Technical University of Crete, Greece
School of Electronic and Computer Engineering

Development of a mobile platform for the self-screening of human pathologies

Ioannis Margaris

Thesis Committee
Professor Costas Balas (ECE)
Associate Professor Antonios Deligiannakis (ECE)
Assistant Professor Vasilis Samoladas (ECE)

Chania, 2016

"If debugging is the process of removing software bugs, then programming must be the process of putting them in." (Edsger Dijkstra)

Abstract

Nowadays mobile applications have become extremely popular since time spent on smart-phones exceeds computer usage. In terms of mobile application popularity, games and social networking are the most popular pastime, rating much higher than news and entertainment.

As consumer's use of mobile devices has increased, so has the variety of applications being published. One of the most upcoming field of mobile applications is the one concerning health, known as mHealth apps. Mobile applications can certainly help many people to take care of their health and wellness, a healthier lifestyle, and also gives access to valuable information whenever they require it.

In this thesis, a mHealth application was developed using Java and Xml, which is a mobile platform for screening human eye pathologies. The application was implemented using the Eclipse IDE (Integrated Development Environment) and the ADT bundle (Android Development Tools). This particular application consists of a set of ophthalmological tests, a way to schedule a test in a given time and date, a SQLite database used for the data storage and some health and nutrition tips.

The results, after the application was tested by a number of users, confirm the validity of the app, since people with eye deficiencies presented low scores on to the respective tests.

Acknowledgements

First of all, I would like to thank my supervisor Professor Costas Balas for his devoted guidance and his enlightening discussions that made this work possible. I also wish to thank Thanasis Papathanasiou for all the support and help he provided me with.

Finally, I wish to express my deep gratitude to my family and friends for their encouragement and support all these years.

Contents

List of Figures

Listings

1 Introduction

In this thesis, a mHealth android application was developed with the main purpose to be used for self screening of eye pathologies. This application allows the users to quickly test and record their progress.

In the next chapter, the theoretical background of the tests implemented in the application is presented, in order to understand how the eye deficiencies affect the human vision.

Afterwards, the step-by-step development of the application shows how to combine a variety of building blocks in order to create a native android app. The tools utilized for the creation of this app and a detailed description of the development of each screen will be illustrated and analyzed in chapter three.

Finally, the results and future work is presented in the last chapter of this thesis.

2 Theoretical Background

Understanding the basic functions of the human eye is the first step towards building something that will help people track or discover some eye pathology. That being said, everything starts when light rays enter the eye through the cornea, the so called clear front "window" of the eye. The cornea's refractive power bends the light rays in such a way that they pass freely through the pupil, the opening in the center of the iris through which light enters the eye, as presented in Figure 2.1. The iris works like a shutter in a camera which means it has the ability to enlarge and shrink, depending on how much light is entering the eye. After passing through the iris, the light rays passes through the eye's natural crystalline lens. This flexible structure works like the lens in a camera, shortening and lengthening its width in order to focus light rays properly. Light rays pass through a dense, transparent, gel-like substance, called the vitreous that fills the globe of the eyeball and helps the eye hold its spherical shape.

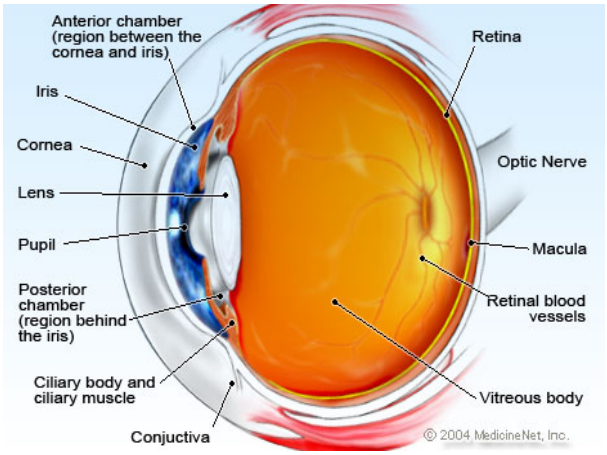


Figure 2.1: The anatomy of the eye by MedicineNet, Inc.

In a normal eye, the light rays come to a sharp focusing point on the retina. The retina functions much like the film in a camera, it is responsible for capturing all of the light rays, processing them into light impulses through millions of tiny nerve endings,

then sending these light impulses through over a million nerve fibers to the optic nerve. Because the keratoconus cornea is irregular and cone shaped, light rays enter the eye at different angles and do not focus on one point the retina, but on many different points causing a blurred, distorted image.

In summary, the cornea is the clear, transparent front covering which admits light and begins the refractive process. It also keeps foreign particles from entering the eye. The pupil is an adjustable opening that controls the intensity of light permitted to strike the lens. The lens focuses light through the vitreous humor, a clear gel-like substance that fills the back of the eye and supports the retina. [1]

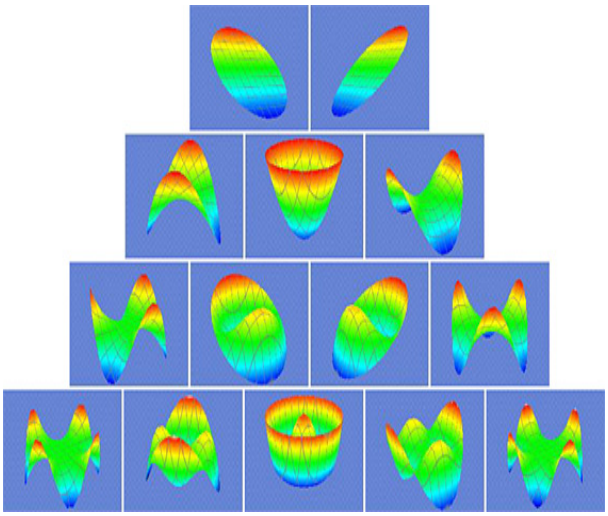


Figure 2.2: Common shapes of aberrations created when a wavefront of light passes through eyes with imperfect vision by Alcon Inc.

The eye's ability to refract or focus light sharply on the retina primarily is based on three eye anatomy features: 1) the overall length of the eye, 2) the curvature of the cornea and 3) the curvature of the lens inside the eye. If the eye is too long, light is focused before it reaches the retina, causing nearsightedness. If the eye is too short, light is not focused by the time it reaches the retina. This causes farsightedness or hyperopia. If the cornea is not perfectly spherical, then the image is refracted or focused irregularly to create a condition called astigmatism. A person can be nearsighted or farsighted with or without astigmatism. If the lens is too steeply curved in relation to the length of the eye and the curvature of the cornea, it causes nearsightedness. If the lens is too flat, the result is farsightedness. More obscure vision errors, known as higher-order aberrations shown in Figure 2.2, are also related to flaws in the way light rays are refracted as they

travel through the eye's optical system. These types of vision errors, which can create problems such as poor contrast sensitivity, are detected through new technology known as wavefront analysis. [10]

Imagine light traveling as a bundle of rays. If you draw lines perpendicular to the tips of a bundle of light rays, you get what is called a wavefront map. In an eye with perfect vision, the wavefront is perfectly flat; but the wavefront of an imperfect eye is irregular. The types of distortions this wavefront acquires as it travels through the eye provide valuable information about vision errors and how to correct them.

Knowing how the eye works helps with the understanding of the errors found in the eye as well. A refractive error, or refraction error, is an error in the focusing of light by the eye and a frequent reason for reduced visual acuity. The number of people globally with refractive errors has been estimated from 800 million to 2.3 billion. An eye that has no refractive error when viewing distant objects is said to have emmetropia or is emmetropic, meaning that the eye is in a state in which it can focus parallel rays of light (light from distant objects) on the retina, without using any accommodation. A distant object in this case is defined as an object 8 meters or further away from the eye. This proves to be an evolutionary advantage by automatically focusing the eye on objects in the distance because it allows an individual to be alert in, say, a prey-predator situation. An eye that has refractive error when viewing distant objects is said to have ametropia or be ametropic. This eye cannot focus parallel rays of light (light from distant objects) on the retina, or needs accommodation to do so. The word "ametropia" can be used interchangeably with "refractive error". Types of ametropia include myopia, hyperopia and astigmatism and are frequently categorized as spherical and cylindrical errors. A spherical error occurs when the focusing power of the eyeball is either too weak or too powerful to focus light on the retina. This can lead to myopia (nearsightedness) or hyperopia (farsightedness). On the other hand, a cylindrical error occurs when the cornea or lens of the eyeball is irregularly shaped, preventing focused light from reaching the retina. Cylindrical errors include astigmatism, in which the cornea or lens is football or donut shaped, and presbyopia, in which the eye's natural lens loses the flexibility required to refocus light.

How refractive errors are treated or managed depends upon the amount and severity of the condition. Those who possess mild amounts of refractive error may elect to leave the condition uncorrected, particularly if the patient is asymptomatic. For those who are symptomatic, glasses, contact lenses, refractive surgery, or a combination of the three are typically used. [10]

2.1 Eye Pathologies

Your eye doctor determines the type and degree of refractive error you have by performing a test called a refraction. This can be be done with a computerized instrument (automated refraction) or with a mechanical instrument called a phoropter that allows your eye doctor to show you one lens at a time (manual refraction). Often, an automated refraction will be performed by a member of the doctor's staff, and then the eye care practitioner will refine and verify the results with a manual refraction.

A refraction may reveal more than one type of refractive error. For example, blurred vision may be due to both nearsighted and astigmatism. An eye doctor will use the results of the refraction to determine a prescription for eyeglasses. A refraction, however, does not provide sufficient information to write a contact lens prescription, which requires a contact lens fitting. Eyeglass lenses and contact lenses are fabricated with precise curves to refract light to the degree necessary to compensate for refractive errors and bring light to a sharp focus on the retina. [10]

2.1.1 Color Vision Deficiency

Color Blindness is an inaccurate term to describe a lack of perceptual sensitivity to certain colors, a more precise term is Color Vision Deficiency (CVD). Color blindness is, however, the most commonly used term though it is misleading if taken literally, because colorblind people can see colors, but cannot make out the difference between some couples of complementary colors.

Color vision deficiency is not related to visual acuity at all and is most commonly due to an inherited condition. Red/Green color vision deficiency is by far the most common form, about 99%, and causes problems in distinguishing reds and greens. Another color vision deficiency, Blue/Yellow, also exists, but is rare and there is no commonly available test for it.

The most commonly used test to detect color vision deficiencies was developed by the Japanese ophthalmologist Shinobu Ishihara (1879-1963). [14]

Ishihara Test

The test is named after Dr. Shinobu Ishihara, who was born in Tokyo in 1879 and attended the Imperial University there on a military scholarship. At the request of the Japanese Army, he undertook postgraduate studies in ophthalmology from 1908, first in Tokyo and later in Germany in 1913-1914. The outbreak of war in Europe forced

Ishihara's return to Japan where he took up a military physician's post in Tokyo, and it is at this time that the Army asked him to develop a color vision test (Figure 2.3) for its conscripts. There were already several such tests in use in Japan, some designed by Japanese ophthalmologists, with another, the 'Stilling' test (after the German ophthalmologist Jakob Stilling, first published in 1878) perhaps the most well known.

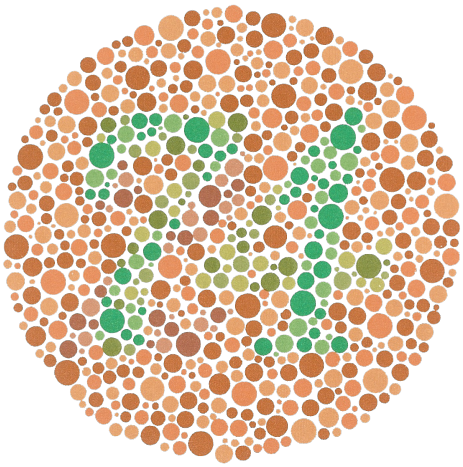


Figure 2.3: An Ishihara plate showing the number 74

The principle on which they were based is called Pseudo-isochromaticism. Simply put, it describes the effect of two (or more) colours appearing equivalent (or 'isochromatic') when in fact they are distinct (thus 'pseudo'). This, of course, applies readily to those with color-deficient vision and can be exploited if colors typically misinterpreted as equivalent are strategically deployed in response or task-based vision tests that expose the error and lead to positive diagnoses. If the tests are well considered, they do not require the subject to actually identify colours by name, as this greatly increases the risk of ambiguous answers.

Like his predecessors, Ishihara also based his test on pseudo-isochromaticism, but with the intention of delivering results that were more easily interpreted and thus more reliable. He began with a circular field in which dots of several sizes and tones were evenly though irregularly distributed. They would ensure that within the field no shape larger than the dots (and composed of them) could be discerned either by outline or by tonal changes, since the configuration of the dots would disintegrate these features, a standard

camouflage technique. Ishihara could now create shapes within the field that could only be identified on the basis of colour (or, more exactly, hue) alone. This would be done in four ways. The first involved a 'transformation'. Two numerals were superimposed in the dot field. Where the numerals overlapped, they were composed of hues that could be distinguished from the background by both normal subjects and by those with color defective vision. Where the numerals did not overlap, one was completed with hues distinct only to color normals, the other with hues distinct only to colour defectives. Of the two numerals, the one identified by the subject indicated the state of their colour vision. In some cases, numerals were strategically paired to maximise their overlap (e.g. 8-3, 6-5, 4-1) and so disguise one within the other more effectively. Here, the shared widths and shapes of the clarendon-like italics proved advantageous, as did their swelling strokes and rounded terminals, which settled seamlessly into the field of dots.

On this basis, a further three tests were devised: a 'vanishing' design where only subjects with normal color vision could distinguish numerals in the dot field; its opposite, a 'hidden' design where numerals could only be discerned by color defectives; and a 'qualitative' design where differing degrees of numeral visibility helped identify mild or severe disorders. And to ensure that those who were unfamiliar with numerals were not excluded, Ishihara created a second series of tests in the same four ways but which only required subjects to trace paths that meandered around the dot field which is displayed in Figure 2.3.

Almost nine decades on from its first edition, the Ishihara test remains widely used, able to quickly screen for color vision defects that other, more exacting tests can then elucidate in detail. It is thus easy to assert its importance to medical practice in general, and specifically to those for whom a diagnosis of color-deficient vision offers an explanation for why their perception of the world is often bewilderingly different from others. But beyond these qualities lies a subtler psychological dimension that is also enlightened. That 'Ishihara' is not a test of mere dysfunction, you either see something or you can't, but instead one of difference: that everyone sees something, it is just not always the same. This is surely a more productive understanding of color-deficient vision, and one that guided Shinobu Ishihara to enduring effect. [11]

Neitz Test

Our global community increasingly relies on color to communicate. Yet 8% to 10% of all males and 0.5% of all females are color-blind. In a classroom of 20 children, it's likely that at least 1 will have a problem discerning color.

Fortunately, there's now an easy, inexpensive way to screen for color vision deficiencies. The Neitz Test of Color Vision is a revolutionary new approach to testing for color blindness. Developed at the Eye Institute of the Medical College of Wisconsin, the Neitz Test is accurate, quick, and inexpensive. It identifies the type and severity of color vision deficiency in just a few minutes. It can be used with people of any age, including very young children. And it can be administered in fluorescent light, daylight, or a combination of the two, making it much more convenient than competing instruments. Because it can be given to large groups at a low cost, the Neitz makes routine screening not only possible, but also easy.

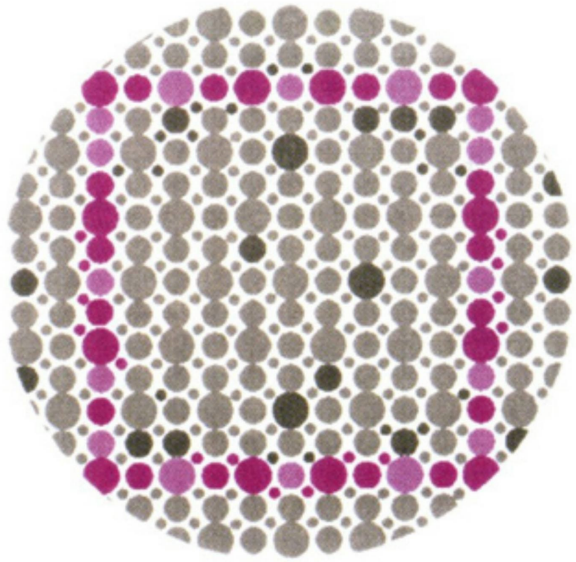


Figure 2.4: A Neitz plate showing a square

Like conventional color vision tests, the Neitz asks individuals to identify colored shapes within grey dotted patterns, as shown in Figure 2.4. Unlike these tests, however, it does not require expensive color plates that eventually fade and need to be replaced. Requiring no special training, the test is quite easy to administer. Elementary school teachers can distribute, administer and collect the test in their classrooms, in less than 5

minutes. This means you can screen an entire student body without disrupting lessons or taking a single child out of class. Reliable, economical and easy to administer on a large scale, the Neitz Test of Color Vision permits routine screening in schools, industry, public safety agencies and the military, thereby reducing the likelihood of accident or educational disadvantage due to poor color vision. [12]

The results of a study from the Illinois College of Optometry show that the Neitz test has high sensitivity and specificity and moderate PPV (Positive Predictive Value) however, the diagnostic aspect of the test fails to agree with the gold standard. In addition, figure ground confusion in the adult subjects may prevent accurate diagnosis. The Ishihara has high sensitivity, specificity, and PPV and is still the test of choice.[7]

Color Sensitivity Test

The Cambridge Colour Test provides a rapid means of screening subjects for colour vision deficiencies; but it also can be used to examine in more detail the changes in colour discrimination that occur as a result of congenital or acquired conditions. It allows the investigator to monitor quantitatively over time the progression or remission of disease. Many drugs affect colour vision and the pharmacologist will find the test well suited to monitoring the short-term or long-term course of such side-effects.

The procedure is easy to use for both the investigator and the subject. It uses the familiar Landolt C stimulus, defined by the two test colors that are to be discriminated, on an achromatic background. The Cambridge Color test uses the proven concept of introducing spatial and luminance noise into the stimulus, which is composed of grouped circles randomly varying in diameter and having no spatial structure. The Landolt C is therefore defined by chromaticity alone, ensuring that the subject's responses are not due to luminance or spatial cues in the stimulus, and thus avoiding the necessity for a preliminary procedure to find isoluminance for the test colours.

The test is conceptually very simple to explain to the subject, who responds to the orientation of the Landolt C, as displayed in Figure 2.5. The chromaticity of the components of the C is varied along the protan, deutan and tritan lines using a standard descending psychophysical staircase procedure. [2]

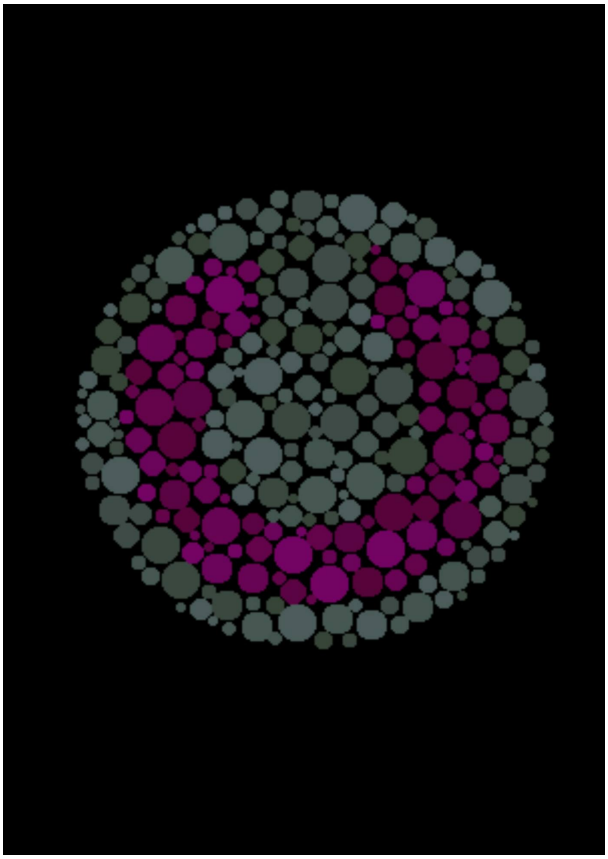


Figure 2.5: Use of the Landolt C stimulus in the Cambrige color sensitivity test

2.1.2 Astigmatism

Astigmatism is a vision condition that causes blurred vision due either to the irregular shape of the cornea, the clear front cover of the eye, or sometimes the curvature of the lens inside the eye. An irregular shaped cornea or lens prevents light from focusing properly on the retina, the light sensitive surface at the back of the eye. As a result, vision becomes blurred at any distance.

The curvature of the cornea and lens causes light entering the eye to be bent in order to focus it precisely on the retina at the back of the eye. In astigmatism, the surface of the cornea or lens has a somewhat different curvature in one direction than another. In the case of the cornea, instead of having a round shape like a basketball, the surface of the cornea is more like a football. As a result, the eye is unable to focus light rays to a single point causing vision to be out of focus at any distance.

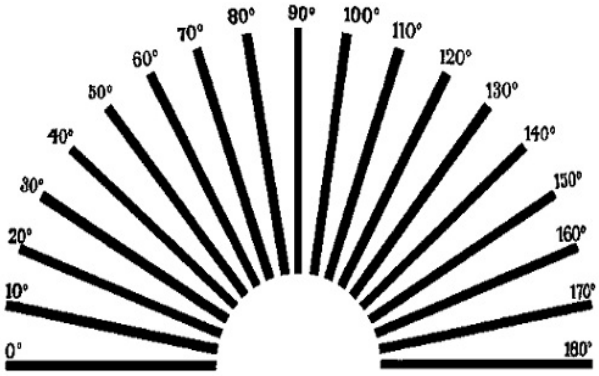


Figure 2.6: Astigmatism angle chart

Astigmatism is a very common vision condition. Most people have some degree of astigmatism. Slight amounts of astigmatism usually do not affect vision or require treatment. However, larger amounts cause distorted or blurred vision, eye discomfort and headaches. Looking at the Astigmatism chart (Figure 2.6) from several distances helps people notice where the distortion begins and ends. It is possible to have astigmatism only at the near, or only at the distance, or at any plane in between. People with natural clear vision see all the lines in the same thickness and exactly the same spacing between the lines.

Astigmatism frequently occurs with other vision conditions like nearsightedness (myopia) and farsightedness (hyperopia). Together these vision conditions are referred to as refractive errors because they affect how the eyes bend or "refract" light. The specific cause of astigmatism is unknown. It can be hereditary and is usually present from birth. It can change as a child grows and may decrease or worsen over time. [5]

2.1.3 Myopia/Hyperopia

Nearsightedness, or myopia, as it is medically termed, is a vision condition in which close objects are seen clearly, but objects farther away appear blurred. Nearsightedness occurs if the eyeball is too long or the cornea, the clear front cover of the eye, has too much curvature. As a result, the light entering the eye isn't focused correctly and distant objects look blurred.

Nearsightedness is a very common vision condition affecting nearly 30% of the population. Some research support the theory that nearsightedness is hereditary. There is also growing evidence that it is influenced by the visual stress of too much close work. Generally, nearsightedness first occurs in school-age children. Because the eye continues

to grow during childhood, it typically progresses until about the age of 20. However, nearsightedness may also develop in adults due to visual stress or health conditions such as diabetes. A common sign of nearsightedness is difficulty with the clarity of distant objects like a movie or TV screen or the chalkboard in school. As in the myopia test presented in Figure 2.7, if someone sees the O blacker or sharper in the red part, they probably have myopia or have a myopic tendency. Conversely, if someones sees the O blacker or sharper in the green part, they are probably hyperope and may have hyperopia. An optometrist can prescribe eyeglasses or contact lenses that correct nearsightedness by bending the visual images that enter the eyes, focusing the images correctly at the back of the eye. Depending on the amount of nearsightedness, you may only need to wear glasses or contact lenses for certain activities, like watching a movie or driving a car. Or, if you are very nearsighted, they may need to be worn all the time.

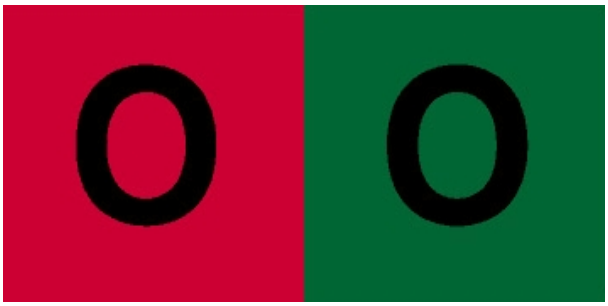


Figure 2.7: Myopia/Hyperopia test

The exact cause of nearsightedness is unknown, but two factors may be primarily responsible for its development: heredity and visual stress. There is significant evidence that many people inherit nearsightedness, or at least the tendency to develop nearsightedness. If one or both parents are nearsighted, there is an increased chance their children will be nearsighted. Even though the tendency to develop nearsightedness may be inherited, its actual development may be affected by how a person uses his or her eyes. Individuals who spend considerable time reading, working at a computer, or doing other intense close visual work may be more likely to develop nearsightedness. Lastly, nearsightedness may also occur due to environmental factors or other health problems. [6]

On the other hand, farsightedness, or hyperopia, as it is medically termed, is a vision condition in which distant objects are usually seen clearly, but close ones do not come into proper focus. Farsightedness occurs if your eyeball is too short or the cornea has too little curvature, so light entering your eye is not focused correctly. Common signs

of farsightedness include difficulty in concentrating and maintaining a clear focus on near objects, eye strain, fatigue and/or headaches after close work, aching or burning eyes, irritability or nervousness after sustained concentration. In mild cases of farsightedness, your eyes may be able to compensate without corrective lenses. In other cases, your optometrist can prescribe eyeglasses or contact lenses to optically correct farsightedness by altering the way the light enters the eyes. [3]

2.1.4 Macular Degeneration

Macular degeneration, also known as age-related macular degeneration (AMD or ARMD), is a medical condition which may result in blurred or no vision in the center of the visual field. This is where the Amsler Grid comes into the picture and can help people detect early on when there is a visual deficiency just by looking at the grid (Figure 2.8). Early on there are often no symptoms. Over time, however, some people experience a gradual worsening of vision that may affect one or both eyes. While it does not result in complete blindness, loss of central vision can make it hard to recognize faces, drive, read, or perform other activities of daily life. Visual hallucinations may also occur but these are not serious and do not represent a mental illness.

Macular degeneration typically occurs in older people. Genetic factors and smoking also play a role. It is due to damage to the macula of the retina. The severity is divided into early, intermediate, and late types. The late type is additionally divided into "dry" and "wet" forms with the dry form making up to 90 percent of cases.

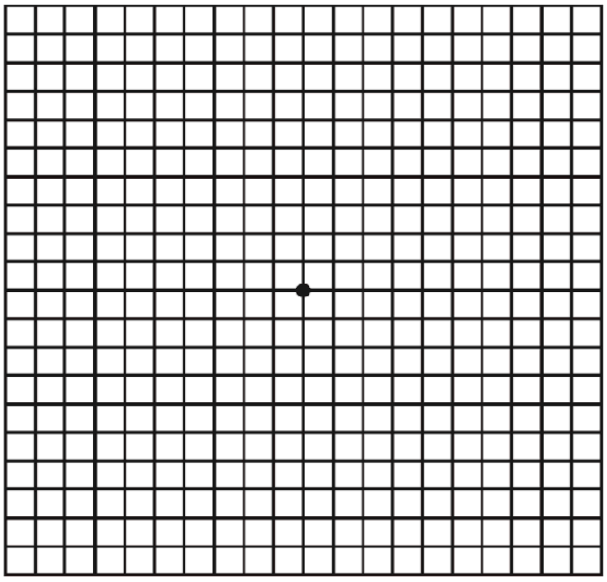


Figure 2.8: The Amsler Grid

Prevention includes not smoking, exercising and eating well. Vitamin supplements do not appear to be useful for prevention. There is no cure or treatment that returns vision already lost. In the wet form, anti-VEGF medication injected into the eye or less commonly laser coagulation or photodynamic therapy may slow worsening. Supplements in those who already have the disease may slow progression.

In 2010 it affected 23.5 million people globally. In 2013 moderate to severe disease affected 13.4 million and it is the fourth most common cause of blindness after cataracts, preterm birth and glaucoma. It most commonly occurs in people over the age of fifty and in the United States is the most common cause of vision loss in this age group. About 0.4% of people 50 to 60 years old have the disease, while it occurs in 0.7% of people 60 to 70, 2.3% of those 70 to 80, and nearly 12% of people over 80 years old. Macular degeneration by itself will not lead to total blindness. For that matter, only a very small number of people with visual impairment are totally blind. In almost all cases, some vision remains, mainly peripheral. Other complicating conditions may possibly lead to such an acute condition (severe stroke or trauma, untreated glaucoma, etc.), but few macular degeneration patients experience total visual loss.

The area of the macula comprises only about 2.1% of the retina, and the remaining 97.9% (the peripheral field) remains unaffected by the disease. Even though the macula provides such a small fraction of the visual field, almost half of the visual cortex is devoted to processing macular information. The loss of central vision profoundly affects visual functioning. It is quite difficult, for example, to read without central vision. Pictures that attempt to depict the central visual loss of macular degeneration with a black spot do not really do justice to the devastating nature of the visual loss. This can be demonstrated by printing letters six inches high on a piece of paper and attempting to identify them while looking straight ahead and holding the paper slightly to the side. Most people find this difficult to do. [15]

2.1.5 Contrast Sensitivity

A contrast sensitivity test measures your ability to distinguish between finer and finer increments of light versus dark (contrast). This differs from common visual acuity testing in a routine eye exam, which measures your ability to recognize smaller and smaller letters on a standard eye chart.

Contrast sensitivity is a very important measure of visual function, especially in situations of low light, fog or glare, when the contrast between objects and their background often is reduced. Driving at night is an example of an activity that requires

good contrast sensitivity for safety. Even if you have 20/20 visual acuity, you can have eye or health conditions that may diminish your contrast sensitivity and make you feel that you are not seeing well. If you have low contrast sensitivity, you may have problems with night driving, including difficulty seeing pedestrians walking alongside poorly lit streets. Or you might notice that your eyes tire more easily while reading or watching television. Poor contrast sensitivity can also increase your risk of a fall if you fail to see that you need to step down from a curb onto similarly colored pavement.

Low contrast sensitivity can be a symptom of certain eye conditions or diseases such as cataracts, glaucoma or diabetic retinopathy. [9]

2.1.6 Visual Acuity

Visual Acuity commonly refers to the clarity of vision. Visual Acuity is dependent on optical and neural factors, i.e. the sharpness of the retinal focus within the eye, the health and functioning of the retina or the sensitivity of the interpretative faculty of the brain.

A common cause of low visual acuity is refractive error (ametropia), or errors in how the light is refracted in the eyeball. Causes of refractive errors include aberrations in the shape of the eyeball, the shape of the cornea, and reduced flexibility of the lens. In the case of pseudomyopia, the aberrations are caused by muscle spasms. Too high or too low refractive error (in relation to the length of the eyeball) is the cause of near-sightedness (myopia) or farsightedness (hyperopia) (normal refractive status is referred to as emmetropia). Other optical causes are astigmatism or more complex corneal irregularities. These anomalies can mostly be corrected by optical means, such as eyeglasses, contact lenses, laser surgery, etc.

Neural factors that limit acuity are located in the retina or the brain (or the pathway leading there). Examples for the first are a detached retina and macular degeneration, to name just two. In some cases, low visual acuity is caused by brain damage, such as from traumatic brain injury or stroke. When optical factors are corrected for, acuity can be considered a measure of neural well-functioning.

Visual acuity is typically measured while fixating, i.e. as a measure of central vision, for the reason that it is highest there. However, acuity in peripheral vision can be of equal (or sometimes higher) importance in everyday life. Acuity declines towards the periphery in an inverse-linear (i.e. hyperbolic) fashion. 20/20 vision is a term used to express normal visual acuity (the clarity or sharpness of vision) measured at a distance of 20 feet. If you have 20/20 vision, you can see clearly at 20 feet what should normally

be seen at that distance. If you have 20/100 vision, it means that you must be as close as 20 feet to see what a person with normal vision can see at 100 feet. [16]

20/20 does not necessarily mean perfect vision. 20/20 vision only indicates the sharpness or clarity of vision at a distance. There are other important vision skills, including peripheral awareness or side vision, eye coordination, depth perception, focusing ability and color vision that contribute to your overall visual ability. Some people can see well at a distance, but are unable to bring nearer objects into focus. This condition can be caused by hyperopia (farsightedness) or presbyopia (loss of focusing ability). Others can see items that are close, but cannot see those far away. This condition may be caused by myopia (nearsightedness).

A comprehensive eye examination by a doctor of optometry can diagnose those causes, if any, that are affecting your ability to see well. In most cases, your optometrist can prescribe glasses, contact lenses or a vision therapy program that will help improve your vision. If the reduced vision is due to an eye disease, the use of ocular medication or other treatment may be used. [4]

2.1.7 Optic Neuritis

Optic neuritis is a demyelinating inflammation of the optic nerve. It is also called papillitis (when the head of the optic nerve is involved) and retrobulbar neuritis (when the posterior of the nerve is involved). It is caused by many different conditions, and it may lead to complete or partial loss of vision.

The most common cause is multiple sclerosis. Major symptoms are sudden loss of vision (partial or complete), sudden blurred or "foggy" vision, and pain on movement of the affected eye. The vision might also be described as "disturbed/blackened" rather than blurry, as when feeling dizzy. Many patients with optic neuritis may lose some of their color vision in the affected eye (especially red), with colors appearing subtly washed out compared to the other eye. [17]

The phenomenon of red desaturation is recognised as an early indicator of neurological field defects. Red desaturation may occur disproportionately to retention of acuity and form perception. It is possible that bitemporal defects from chiasmal compression may be detected by red testing at an earlier stage than using traditional methods. Visual difficulties may be the only symptom of early chiasmal compression, and, if detected early, any visual loss may be reversible. Patients presenting to the optometrist with non-specific visual loss or difficulty should be checked for red field defects.

It has been found that a quarter of patients with a pituitary tumour had visual complaints from two to 10 years before diagnosis [13]

3 | Oculus Application Development

Mary Meeker, an analyst at Kleiner Perkins Caufield Byers, predicted back in 2008 "Mobile to overtake fixed Internet access by 2014". We are now past the mobile tipping point as this report from comScore shows in Figure 3.1. So it is no longer a case of asking whether the mobile market is important, it is now a question of using the statistics to understand how this enormous group uses their mobile devices and what their preferences are, so we can think of, develop and publish applications which will positively influence their lives.

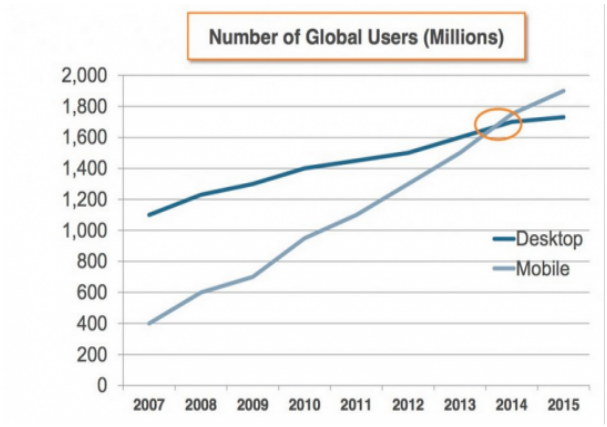


Figure 3.1: Number of global users - Mobile vs Desktop (modified by comScore)

This application which is developed in the contents of this thesis can be categorized in the m-Health applications. m-Health is an abbreviation for mobile health, a term used for the practice of medicine and public health supported by mobile devices. Mobile phones have made a recent and rapid entrance into many parts of the world, with the global mobile phone penetration rate drastically increasing over the last decade due to improvements in telecommunications technology infrastructure which in its own turn leads to reduced costs of mobile handsets. Because they are accessible to users both at

home and on-the-go, mhealth apps are a part of the movement towards mobile health programs in health care.

There are many varieties of m-Health apps available for purchase (or for free) from the app stores. This application's purpose is the self-screening of eye pathologies allowing the users to quickly test and record their progress. It provides them with several tests (Color deficiency test, Astigmatism test, Visual Acuity test, etc.) , an easy way to program the application to run at a desired date and time, some nutrition tips regarding their eye health, and a database which contains their test results in order to keep track of their progress.

3.1 The Tools Utilized

The application was developed using Google's ADT (Android Development Tools) bundle which includes:

- The Eclipse IDE: An integrated development environment used for Android development. Because Eclipse is also written in Java, you can install it on a PC, a Mac, or a Linux computer.
- The Android Developer Tools: a plugin for the Eclipse IDE that extends the capabilities of Eclipse to quickly set up new Android projects, create an application UI (User Interface), add packages based on the Android Framework API (Application Programming Interface), and export signed (or unsigned) .apk files in order to distribute and publish the application
- The latest Android SDK, which was downloaded from the Android SDK manager
- The Android SDK platform tools for debugging and testing purposes
- A system image for the Android Emulator for the testing of the application on different virtual devices

3.2 Oculus's Screens

To understand how this project came together, one must first understand the fundamental components of an application. Each component is a different point through which the system can enter an app. Not all components are actual entry points for the user and some depend on each other, but each one exists as an individual entity and plays a

specific role. Each component is a unique building block that helps the application define its overall behavior. There are four different types of application components: Activities, Services, Content Providers and Broadcast Receivers, from which this project utilizes everything but services.

- Activities: An activity represents a single screen combined with a user interface. For example, to-do list apps might have one activity that shows a list of new tasks, another activity to compose to-do task, and another activity for deleting tasks. Although the activities work together to form a cohesive user experience in the application, each one is independent of the others. As such, a different application can start any one of these activities (if the to-do app allows it). For example, a calendar application can start the activity in the to-do app that composes a new task, in order for the user to save a specific task on a specific date.

- Services: A service is a component that runs in the background to perform long-running operations or to perform a task for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run in order to interact with it

- Content providers: A content provider manages a bundle of the application data. You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your app can access. Through the content provider, other applications (or Activities) can query or even modify the data (if the content provider allows it). For example, any application with the proper permissions can query part of the content provider to read and write information from an SQLite database.

- Broadcast receivers: A broadcast receiver is a component that responds to system-wide broadcast announcements. Many broadcasts originate from the system such as, a broadcast announcing that the screen has turned off, the battery is low, a calendar event came up, or a screen-shot was captured. Apps can also initiate broadcasts for example, to let other apps know that some data has been downloaded to the device and is available for them to use. Although broadcast receivers do not display a user's interface, they may create a status bar notification to alert the user when a broadcast event occurs.

3.2.1 The Splash Screen

Android splash screens are normally used to show the user some kind of progress before the application loads completely. Some applications use splash screens just to show case their app logo (as does this app) for a fixed amount of time, using a timer to dismiss the screen, making the user interface more appealing. Unfortunately, android does not have any inbuilt mechanism to show splash screen so it is necessary to custom build one.

After the creation of the SplashScreen class it is needed to declare a timer and this apps splash screen timer is set to 3000ms (or 3s). Eclipse generates some lines of code on the creation of activities (such as lines 6-8 from the code snippet below which is the on Create method of every activity) so as a convention, throughout this thesis there will be no reference of this type of codes.

The windowManager used in lines 10-12 is configured so this activity can be shown in a full screen mode (hidden status bar). Line 14 or the setContentView() is used to inflate the splash_layout.xml which is the interface the user sees and interact with. Finally, the run() method will be executed in order to start the next activity (which will be the LoginSignUp class) when the timer stops.

An Intent is a messaging object used to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental ways to use them: to start an activity, to start a service or to deliver a broadcast. In this case we see in lines 27-28 the intent is used to start the activity LoginSignUp.

Listing 3.1: The SplashScreen.java

```
1  public class SplashScreen extends Activity {
2
3      // Splash screen timer (3000ms=3s)
4      private static int SPLASH_TIME_OUT = 3000;
5
6      @Override
7      protected void onCreate(Bundle savedInstanceState) {
8          super.onCreate(savedInstanceState);
9
10         requestWindowFeature(Window.FEATURE_NO_TITLE);
11         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
12                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

```
13
14          setContentView(R.layout.splash_layout);
15
16          new Handler().postDelayed(new Runnable() {
17
18              /*
19               * Showing splash screen with a timer. This will be useful when you
20               * want to show case your app logo / company
21               */
22
23              @Override
24              public void run() {
25                  // This method will be executed once the timer is over
26                  // Start your app main activity
27                  Intent gotoLoginSignUp = new Intent(SplashScreen.this,
                        LoginSignUp.class);
28                  startActivity(gotoLoginSignUp);
29
30                  // close this activity
31                  finish();
32              }
33          }, SPLASH_TIME_OUT);
34      }
35
36  }
```

The xml code snippet below shows the structure of the splash screens layout. The most common layouts are the linear layout and the relative layout. The former organizes its children into a single horizontal or vertical row and creates a scroll bar if the length of the window exceeds the length of the screen. The latter enables the developer to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent). Layouts can be built with the use of a variety of views (or widgets), accompanied by their attributes, such as a button, an imageView, a TextView, etc.

With these in mind and a little practice, from now on, it is easy to understand the structures of any xml file found in this thesis.

Listing 3.2: The SplashScreen Layout

```
1   <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:background="@drawable/black_background" >
6
7      <ImageView
8          android:id="@+id/imgLogo"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:layout_centerInParent="true"
12         android:src="@drawable/eye_splash" />
13
14     <TextView
15         android:layout_width="fill_parent"
16         android:layout_height="wrap_content"
17               ...
18         android:textColor="#777777"
19         android:textSize="13sp" />
20
21 </RelativeLayout>
```

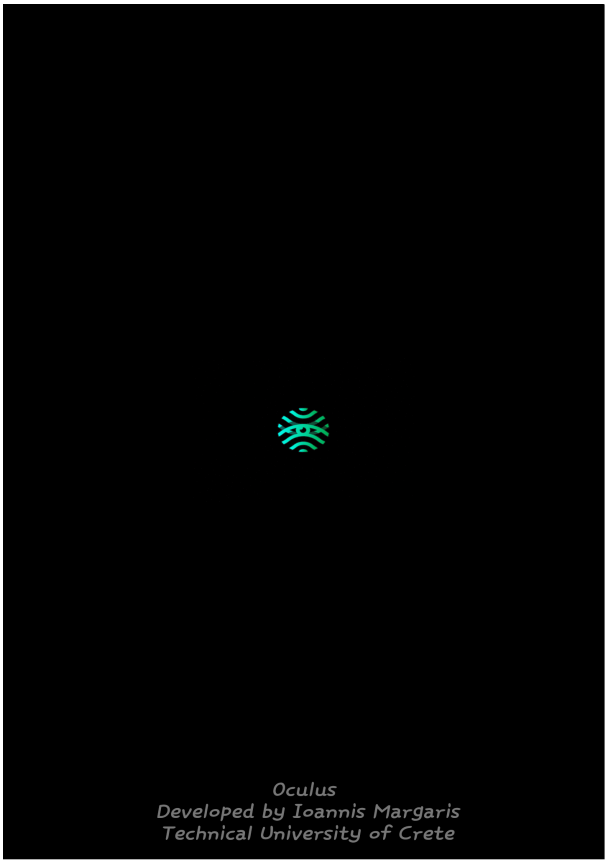


Figure 3.2: The Splash Screen using the xml code from above

3.2.2 The Login System

This app has a login system which consists of the LoginSignUp activity (and its layout) and an SQLite database. The login activity is where the user must input a username, which will then be saved in the database as a unique username.

The login's layout consists of an EditText widget (for inputing text), two Buttons, an ImageView and a TextView (to show text) which can be seen in the xml code below.

Listing 3.3: The LoginSignUp Layout

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:background="@drawable/beach_rocks">
6
7     <View
```

```
8           android:id="@+id/divider"
9              android:layout_width="fill_parent"
10             android:layout_below="@+id/button_login"
11             android:layout_height="2dp"
12             android:layout_marginTop="10dp"
13             android:background="#c0c0c0"/>
14
15      <TextView
16          android:id="@+id/text"
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          android:layout_marginTop="5dp"
20          android:layout_below="@+id/divider"
21          android:text="@string/NewUser"
22          android:textColor="#FFFFFF" />
23
24      <Button
25          android:id="@+id/button_signUp"
26          android:layout_width="wrap_content"
27                 ...
28          android:gravity="center|center_horizontal"
29          android:text="@string/signUp" />
30
31      <ImageView
32          android:id="@+id/imageView1"
33          android:layout_width="wrap_content"
34          android:layout_height="wrap_content"
35          android:layout_alignParentTop="true"
36          android:layout_centerHorizontal="true"
37          android:layout_marginTop="52dp"
38          android:src="@drawable/ic_launcher2" />
39
40      <EditText
41          android:id="@+id/username"
42          android:layout_width="wrap_content"
43                 ...
44          android:textColor="#000000"
45          android:textColorHint="#000000" >
46
```

```
47          <requestFocus />
48      </EditText>
49
50      <Button
51          android:id="@+id/button_login"
52          style="?android:attr/borderlessButtonStyle"
53          android:layout_width="wrap_content"
54                ...
55          android:gravity="center"
56          android:text="@string/login" />
57
58      <TextView
59          android:id="@+id/textView1"
60          android:layout_width="wrap_content"
61          android:layout_height="wrap_content"
62                ...
63          android:textStyle="bold|italic"
64          android:textAppearance="?android:attr/textAppearanceMedium" />
65
66  </RelativeLayout>
```

Implementing the above xml code, Eclipse will produce a layout looking exactly like the login_signup layout used in the LoginSignUp activity (Figure 3.3).

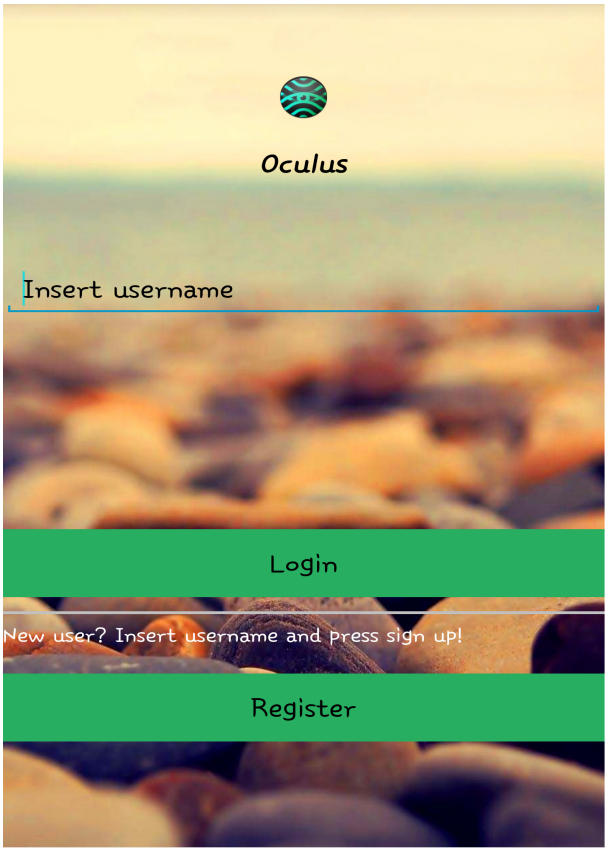


Figure 3.3: The LoginSignUp Activity

The Database

If its the user's first time entering a username, the only constrictions are to not be any other copy of that entry in the database and to not enter an empty username. The database writes its data locally (on the device's memory), so from a security aspect no one can access the applications data due to the lack of root permissions on the android device.

The database supports the creation of multiple users since it assigns unique id's for every user that operates the application on the same android device. Whenever the login activity runs and the user inserts their username, the LoginSignUp activity, shown in the code below, compares it with every entry saved in database's table of users using the Exists() method (line 27 and line 65). If there is no such entry, the user must press the Sign Up button on the login screen, which triggers the InsertUser() method (line

78) and creates a new user. On the contrary, if the Exists() method locates the already created username, the LoginSignUp activity will take the user to the next activity (which will be mentioned later) using an intent (line 38 and line 80).

Listing 3.4: The LoginSignUp.java

```
1
2       setContentView(R.layout.login_signup);
3
4           myDbHelper = new DatabaseAdapter(this);
5
6           imV = (ImageView) findViewById(R.id.imageView1);
7
8           username = (EditText) findViewById(R.id.username);
9           //inputEmail = email.getText().toString();
10          login = (Button) findViewById(R.id.button_login);
11
12          //Create a listener on the login button to
13          login.setOnClickListener(new OnClickListener() {
14           @Override
15           public void onClick(View v) {
16              // TODO Auto-generated method stub
17              //Get the string inserted by the user
18              String user = username.getText().toString();
19              //Check if the user inserted anything
20              if(user.equals("")){
21                 //if input is null show message
22                 Message.message(LoginSignUp.this, "Username is vacant");
23
24              }
25              else{
26                 // Run method Exists to see if username exists in DB
27                 int id = myDbHelper.Exists(user);
28
29                 if(id == 0)
30                 {
31                    // If id=0 user doesn't exist
32                    Message.message(LoginSignUp.this, "Username incorrect or
                          doesnt exist.");
33                 }
```

```
34                else{
35
36                   // If id=1 user exists in DB so let him in the Navigation
                        Drawer class.
37                //Create an Intent
38                   Intent gotoMenu = new Intent(getApplicationContext(),
                        Drawer.class);
39                   gotoMenu.putExtra("Username", user);
40                   startActivity(gotoMenu);
41                   // Call finish() so when the user presses the back button in
                        the Drawer activity
42                   // the login activity is not in the stack, so can't go back
43                   finish();
44                }
45
46             }
47
48          }
49       });
50         signUp = (Button) findViewById(R.id.button_signUp);
51         //create a listener on the sign up button
52         signUp.setOnClickListener(new OnClickListener() {
53          @Override
54          public void onClick(View v) {
55             // TODO Auto-generated method stub
56             //Get the user input
57             String user = username.getText().toString();
58             //check if its null
59             if(user.equals("") ){
60                Message.message(LoginSignUp.this, "Username is vacant");
61
62             }
63             else{
64                // Run method Exists to see if username exists in DB
65                int id = myDbHelper.Exists(user);
66
67                if(id != 0)
68                {
69                   // if (id != 0) then email exists.
```

```
70                  Message.message(LoginSignUp.this, "User already exists.Try
                        another username");
71              }
72              else
73              {
74                  // if id=0 then user doesn't exist. The user must sign up
                        with different email but not
75                  // nessecary to have different password so we call Exists()
                        with query=2
76                  // which will take the 2nd query in Exists(), which checks
                        only the email.
77                  //Call the inserUser method
78                  myDbHelper.insertUser(user);
79                  Message.message(LoginSignUp.this, "Succesfully created
                        user!");
80                  Intent gotoMenu = new Intent(getApplicationContext(),
                        Drawer.class);
81                  gotoMenu.putExtra("Username", user);
82                  startActivity(gotoMenu);
83                  finish();
84
85
86              }
87
88          }
89
90      }
91  });
```

The database, which is listed below, is composed of a table of users , tables for the test scores (with their declared columns) and the string values used for the creation of the tables and the database. Every time the user takes a test, their score (result) will be saved in the corresponding table. When the application was in its first stages there was a decision that had to be made concerning the structure of the database. The first thought was to create a single table comprised of the users and their results, but quickly enough it was decided that the best option was to break down the tables into smaller tables so the database can work faster and use less resources as well. Having a closer look at the structure of the database below makes it easier to understand all of the above, taking

note that the codes are snippets of a larger java file, explaining the use of the three dots in the java code.

Listing 3.5: Part of DatabaseAdapter class where the database is created

```
public static final String DATABASE_NAME = "johnsdatabase";
      //The tables in the Database
      public static final String TABLE_USERS = "users";
      public static final String TABLE_ISHIHARA = "ishihara_table";
      public static final String TABLE_NEITZ = "neitz_table";
      public static final String TABLE_AMSLER = "amsler_table";
      public static final String TABLE_ASTIGMATISM = "astigmatism_table";
                  ...
                  ...
                  ...

      public static final int DATABASE_VERSION = 1;

      //Columns of table users
      public static final String UID = "_id";
      public static final String NAME = "name";
      //public static final String PASSWORD = "Password";

      //Columns of the score tables
      public static final String ISHIHARA_SCORE = "ishihara";
      public static final String ISHIHARA_NAME = "name";

      public static final String NEITZ_SCORE = "neitz";
      public static final String NEITZ_NAME = "name";

      public static final String AMSLER_SCORE = "amsler";
      public static final String AMSLER_NAME = "name";

      public static final String ASTIGMATISM_SCORE = "astigmatism";
      public static final String ASTIGMATISM_NAME = "name";

                  ...
                  ...
                  ...

```

```
36      //Strings used for the table creation
37      private static final String CREATE_TABLE_USERS = "CREATE TABLE
            "+TABLE_USERS+ "("
38            +UID+" INTEGER PRIMARY KEY AUTOINCREMENT, "
39            +NAME+" VARCHAR(255) );";
40
41      private static final String CREATE_TABLE_ISHIHARA = "CREATE TABLE "
            +TABLE_ISHIHARA+ "("
42            +ISHIHARA_NAME+ " VARCHAR(255), "
43            +ISHIHARA_SCORE+ " VARCHAR(255) );";
44
45      private static final String CREATE_TABLE_NEITZ = "CREATE TABLE "
            +TABLE_NEITZ+ "("
46            +NEITZ_NAME+ " VARCHAR(255), "
47            +NEITZ_SCORE+ " VARCHAR(255) );";
48
49      private static final String CREATE_TABLE_AMSLER = "CREATE TABLE "
            +TABLE_AMSLER+ "("
50            +AMSLER_NAME+ " VARCHAR(255), "
51            +AMSLER_SCORE+ " VARCHAR(255) );";
52
53      private static final String CREATE_TABLE_ASTIGMATISM = "CREATE TABLE "
            +TABLE_ASTIGMATISM+ "("
54            +ASTIGMATISM_NAME+ " VARCHAR(255), "
55            +ASTIGMATISM_SCORE+ " VARCHAR(255) );";
56
57                  ...
58                  ...
59                  ...
60
61
62      private Context context;
63
64      public dbHelper(Context context) {
65         super(context, DATABASE_NAME, null, DATABASE_VERSION);
66         this.context = context;
67         //Message.message(context, "Constructor was called");
68      }
69      @Override
```

```
70      public void onCreate(SQLiteDatabase db) {
71         // Create Table VIVZTABLE
72
73         try {
74            db.execSQL(CREATE_TABLE_USERS);
75            db.execSQL(CREATE_TABLE_ISHIHARA);
76            db.execSQL(CREATE_TABLE_NEITZ);
77            db.execSQL(CREATE_TABLE_AMSLER);
78            db.execSQL(CREATE_TABLE_ASTIGMATISM);
79                        ...
80                        ...
81                        ...
82
83            //Message.message(context, "onCreate was called");
84         } catch (SQLException e) {
85
86            Message.message(context, ""+e);
87         }
88
89      }
90
91      @Override
92      public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
93
94         //db.execSQL(CREATE_TABLE);
95
96         try {
97            //Message.message(context, "onUpgrade was called");
98
99
100         } catch (SQLException e) {
101
102            //Message.message(context, ""+e);
103         }
104      }
105
106   }
```

3.2.3 The Navigation Drawer

A navigation drawer is a panel that displays the application's main navigation options on the left edge of the screen. It is hidden most of the time, but is revealed when the user swipes a finger from the left edge of the screen to the right edge or, while the user touches the app icon in the action bar in the top left corner of the screen.

Listing 3.6: The Drawer.java

```
1
2  setContentView(R.layout.drawer_layout);
3
4
5       mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
6       mDrawerList = (ListView) findViewById(R.id.drawerList);
7
8       // Add header to the list
9       View header = getLayoutInflater().inflate(R.layout.header, null);
10      mDrawerList.addHeaderView(header);
11         //Display a manditory message
12
13      Toast.makeText(getApplicationContext(),
14                "Warning!!! Oculus is for screening only.\nIt is not a
                     diagnosis.", Toast.LENGTH_LONG).show();
15
16
17      // Set the adapter for the list view
18      myAdapterDrawer = new MyAdapterDrawer(this);
19      mDrawerList.setAdapter(myAdapterDrawer);
20
21      // Set the lists click listener
22      mDrawerList.setOnItemClickListener(this);
23
24      //Listen to when the Drawer is opened or closed
25      drawerListener = new ActionBarDrawerToggle(this, mDrawerLayout,
26            R.drawable.ic_drawer, R.string.drawer_open,
27            R.string.drawer_closed) {
28
29         mDrawerLayout.setDrawerListener(drawerListener);
30         getActionBar().setHomeButtonEnabled(true);
```

```
31          getActionBar().setDisplayHomeAsUpEnabled(true);
32
33      }
34      //What to do when back button is pressed
35      @Override
36      public void onBackPressed() {
37          new AlertDialog.Builder(this)
38              .setIcon(android.R.drawable.ic_dialog_alert)
39              .setTitle("Closing Activity")
40              .setMessage("Are you sure you want to exit?")
41              .setPositiveButton("Yes", new DialogInterface.OnClickListener()
42          {
43              @Override
44              public void onClick(DialogInterface dialog, int which) {
45                  finish();
46              }
47
48          })
49          .setNegativeButton("No", null)
50          .show();
51      }
52
53
54
55      // Manage drawer list on item clicks
56      @Override
57      public void onItemClick(AdapterView<?> parent, View view, int position,
58              long id) {
59
60         Bundle extras = getIntent().getExtras();
61         String username = extras.getString("Username");
62
63         selectItem(position);
64         switch (position) {
65         case 1:
66            //First Intent takes the user to the Tests
67            Intent showTests = new Intent(Drawer.this,
                  ListActivityWithImages.class);
68            showTests.putExtra("Username", username);
```

```
69              startActivity(showTests);
70          break;
71
72      case 2:
73          //2nd Intent takes the user to schedule a notification
74          Intent scheduleNotification = new Intent(Drawer.this,
                GoToScheduledNotification.class);
75          scheduleNotification.putExtra("Username", username);
76              startActivity(scheduleNotification);
77          break;
78      case 3:
79          //3rd Intent takes the user to past records
80          Intent prevRecords = new Intent(Drawer.this, GoToScores.class);
81          prevRecords.putExtra("Username", username);
82              startActivity(prevRecords);
83          break;
84
85          case 4:
86          //4th Intent shows the nutrition tab
87          Intent showNutrition = new Intent(Drawer.this, Nutrition.class);
88              startActivity(showNutrition);
89          break;
90      case 5:
91          //5th Intent shows about tab
92          Intent showAbout = new Intent(Drawer.this, About.class);
93          showAbout.putExtra("Username", username);
94              startActivity(showAbout);
95          break;
96      }
97  }
98
99  // When an item is pressed, setTitle is called to set the action bar title
100 public void selectItem(int position) {
101     mDrawerList.setItemChecked(position, true);
102
103 }
104
105 // setTitle is implemented here
106 public void setTitle(String title) {
```

```
107         getActionBar().setTitle(title);
108     }
109 
110 }
111 //custom adapter to show the navigation drawer based on a row and then populate
        the list based on that single row
112 class MyAdapterDrawer extends BaseAdapter {
113     //Responsible for fetching my data from string
114     private Context context;
115     String[] loginMenu;
116     int[] images_drawer = {R.drawable.tests,R.drawable.clock_icon,
            R.drawable.prev_records_icon,R.drawable.nutrition, R.drawable.about};
117 
118     public MyAdapterDrawer(Context context){
119         this.context = context;
120         loginMenu = context.getResources().getStringArray(R.array.Login);
121     }
122 
123     @Override
124     public View getView(int position, View convertView, ViewGroup parent) {
125         // TODO Auto-generated method stub
126         View row = null;
127         if (convertView == null) {
128 
129             LayoutInflater inflater = (LayoutInflater)
                    context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
130             row = inflater.inflate(R.layout.single_row_drawer, parent, false);
131         }
132         else
133         {
134             row = convertView;
135         }
136 
137         TextView titleTextView = (TextView) row.findViewById(R.id.textViewDrawer);
138         ImageView titleImageView = (ImageView)
                row.findViewById(R.id.imageViewDrawer);
139 
140         titleTextView.setText(loginMenu[position]);
141         titleImageView.setImageResource(images_drawer[position]);
```

```
142         return row;
143     }
144
145
146
147 }
```

The Drawer.java inflates the drawer_layout.xml and displays a Toast (a temporary notification on the screen) which informs the user that the application is not a diagnosis, it is for screening only (line 12). Lines 35 - 52 are responsible for the back button press, which if pressed will create a dialog interface for the user to choose whether or not to exit the application. The OnItemClick() method, lines 57-98, will be called when the user clicks on an item from the list of items displayed in the navigation drawer, which will then create an intent for each item resulting in the navigation to the next activity. For example, lines 65 - 71 will direct the user to the Tests activity (or ListActivityWithImages.java).

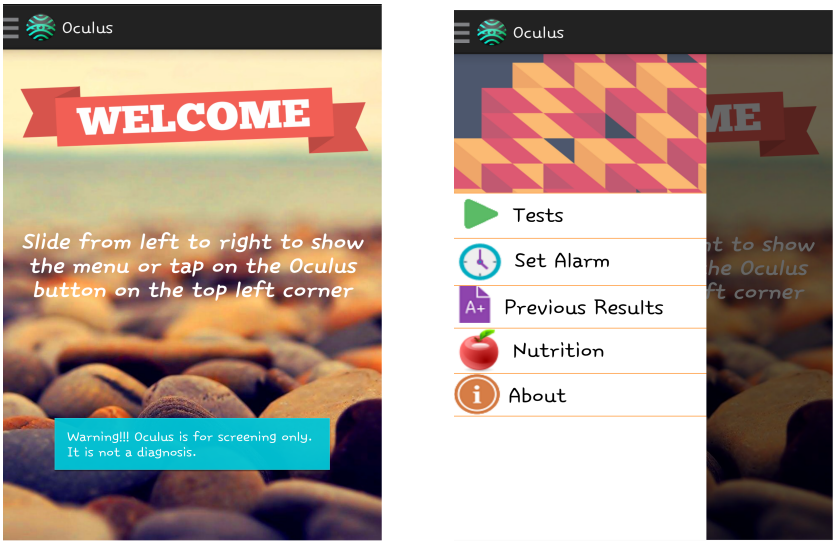
The creation of the drawer's list is done via the MyAdapterDrawer, which basically creates a layout of a row from the list and then recreates the list using that single row layout but with different resources. For example, the Tests item has a green triangle on its left but the Nutrition item has an apple on its left. That happens because the xml code of the single row defines an ImageView on the left of the row. When the adapter creates the list each item on it will have an image on the left side of each row.

Listing 3.7: The drawer layout

```
1
2 <android.support.v4.widget.DrawerLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/drawer_layout"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:background="@drawable/beach_rocks">
8
9     <!-- The main content view -->
10    <FrameLayout
11        android:id="@+id/content_frame"
12        android:layout_width="match_parent"
13        android:layout_height="match_parent"
```

```
14          >
15
16          <ImageView
17              android:id="@+id/welcome_image"
18              android:layout_width="wrap_content"
19              android:layout_height="189dp"
20              android:src="@drawable/welcome3" />
21
22          <TextView
23              android:id="@+id/welcome"
24              android:layout_width="wrap_content"
25              android:layout_height="wrap_content"
26                      ...
27              android:paddingTop="190dp"
28              android:textAppearance="?android:attr/textAppearanceLarge" />
29
30      </FrameLayout>
31
32      <!-- The navigation drawer -->
33
34      <ListView android:id="@+id/drawerList"
35          android:background="#FFFFFF"
36          android:layout_width="240dp"
37          android:layout_height="match_parent"
38          android:layout_gravity="start"
39          android:divider="#ffad60"
40          android:dividerHeight="1dp"
41           >
42       </ListView>
43
44  </android.support.v4.widget.DrawerLayout>
```

Taking all of the above into consideration, implementing the navigation drawer will produce the screens shown below as a result.



(a) Drawer Closed (b) Drawer Opened

Figure 3.4: The navigation drawer (closed and opened)

3.2.4 The Tests

The navigation drawer as mentioned before, is the main activity of this project and provides the user with an easy and simple way to navigate through the application's contents. The drawer list consists of five items: the tests, a notification creator for scheduling a date to run the application, the results of previous tests, a nutrition activity and an about activity.

The application has several tests for the user to choose from. When the user clicks on the Tests item on the drawer list, it navigates to an activity with a list containing all the tests as shown below.

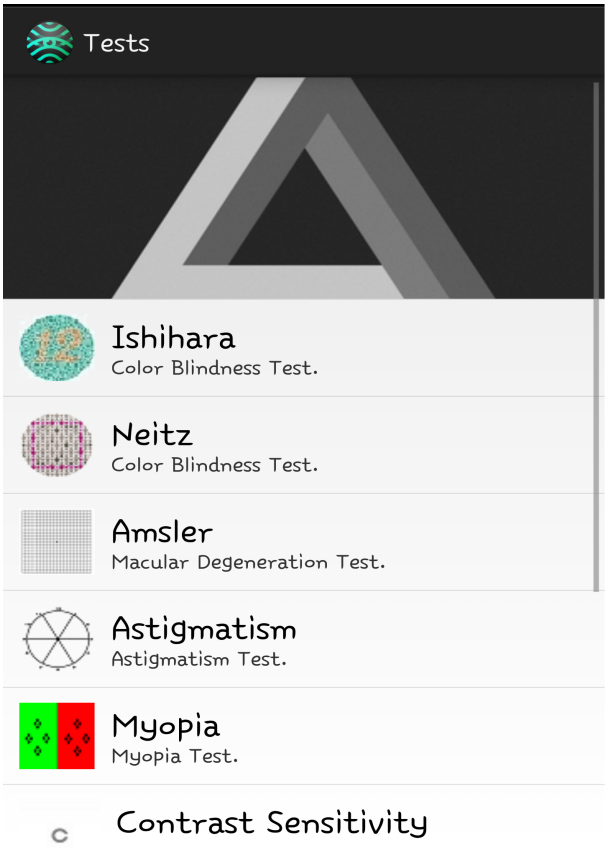


Figure 3.5: The list of tests (or ListActivityWithImages.class)

The layout of ListActivityWithImages.java is comprised from a listView and is populated via the MyAdapter (an array adapter used to populate a listView) which is similar to the adapter used in the drawer.java. The java file implements an onItemClick method of the code below which waits for the user to click on a test. The method then creates an intent that takes the user into the selected test's instructions.

Listing 3.8: The ListActivityWithImages.java

```
1 public class ListActivityWithImages extends Activity {
2   ListView list;
3   String [] Titles;
4   String [] Descriptions;
5   // The images used in the ListView of Tests.
6   int[] images={R.drawable.ishihara_list_icon, R.drawable.neitz_list_icon,
        R.drawable.amsler_list_icon, R.drawable.ast_list_icon,
        R.drawable.myopia_list_icon,R.drawable.contrast_list_icon,R.drawable.acuity_list_icon,R
7
```

```
8   @Override
9    protected void onCreate(Bundle savedInstanceState) {
10      super.onCreate(savedInstanceState);
11      setContentView(R.layout.activity_list);
12      //Set the title of the activity
13      setTitle("Tests");
14
15      Resources res = getResources();
16      Titles =res.getStringArray(R.array.Tests);
17      Descriptions =res.getStringArray(R.array.Descriptions);
18
19      list=(ListView)findViewById(R.id.listView1);
20
21     // Add header to the list
22     View header = getLayoutInflater().inflate(R.layout.header_listview, null);
23     list.addHeaderView(header);
24       //Declare the adapter
25       MyAdapter adapter = new MyAdapter(this, Titles, images, Descriptions);
26       list.setAdapter(adapter);
27
28       //Set the listener on the list
29       list.setOnItemClickListener( new OnItemClickListener(){
30        @Override
31        public void onItemClick(AdapterView<?> parent, View view, int
              position, long id) {
32          // TODO Auto-generated method stub
33          //Get the extra info passed on
34          Bundle extras = getIntent().getExtras();
35          String username = extras.getString("Username");
36          //The switch - case takes the user to the instructions of the test
                selected passing on some extra info
37          switch(position){
38          case 1:
39             Intent ishiharaIntent = new Intent(view.getContext(),
                   Instructions.class);
40             ishiharaIntent.putExtra("Username", username);
41             ishiharaIntent.putExtra("Test", "ishihara");
42             startActivity(ishiharaIntent);
43             finish();
```

```
44              break;
45           case 2:
46              Intent neitzIntent = new Intent(view.getContext(),
                    Instructions.class);
47              neitzIntent.putExtra("Username", username);
48              neitzIntent.putExtra("Test", "neitz");
49              startActivity(neitzIntent);
50              finish();
51              break;
52           case 3:
53              Intent amslerIntent = new Intent(view.getContext(),
                    Instructions.class);
54              amslerIntent.putExtra("Username", username);
55              amslerIntent.putExtra("Test", "amsler");
56              startActivity(amslerIntent);
57              finish();
58              break;
59
60                 ...
61                 ...
62                 ...
63           case 9:
64              Intent RedDesaturationIntent = new Intent(view.getContext(),
                    Instructions.class);
65              RedDesaturationIntent.putExtra("Username", username);
66              RedDesaturationIntent.putExtra("Test", "redDesaturation");
67              startActivity(RedDesaturationIntent);
68              finish();
69              break;
70
71           }
72        }
73       });
74
75     }
76   @Override
77   public boolean onOptionsItemSelected(MenuItem item) {
78       switch (item.getItemId()) {
79       // Respond to the action bar's Up/Home button
```

```
80          case android.R.id.home:
81              NavUtils.navigateUpFromSameTask(this);
82              return true;
83          }
84          return super.onOptionsItemSelected(item);
85      }
86  }
87  //An array adapter as used before
88  class MyAdapter extends ArrayAdapter<String>
89  {
90      Context context;
91      int[] images;
92      String[] titleArray;
93      String[] descriptionArray;
94      MyAdapter(Context c, String[] Titles, int imgs[], String[] desc)
95      {
96          super(c, R.layout.single_row, R.id.textView1, Titles);
97          this.context=c;
98          this.images=imgs;
99          this.titleArray=Titles;
100         this.descriptionArray=desc;
101     }
102     @Override
103     public View getView(int position, View convertView, ViewGroup parent){
104         LayoutInflater
                inflater=(LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
105         View row=inflater.inflate(R.layout.single_row, parent, false);
106
107         ImageView myImage=(ImageView)row.findViewById(R.id.imageView1);
108         TextView myTitle=(TextView) row.findViewById(R.id.textView1);
109         TextView myDescription=(TextView) row.findViewById(R.id.textView2);
110
111         myImage.setImageResource(images[position]);
112         myTitle.setText(titleArray[position]);
113         myDescription.setText(descriptionArray[position]);
114         return row;
115     }
116 }
```

The Ishihara Test

Once the user clicks on the Ishihara test in the ListActivityWithImages activity, the application runs the Instructions.java (one class for all the instructions) which displays the instructions of the selected test. The code below establishes which test has been selected so it can display the appropriate instructions. As mentioned before, the activities pass on some extra information. For example, if the user selected to take the ishihara test, there will be a string containing the name of the test the user selected. This is very useful to identify which instructions will be displayed, after the selection of a test.

Listing 3.9: The Instructions.java

```
1  if (test.equals("ishihara")) {
2
3          //Set the title of the test
4            instr0.setText("Ishihara") ;
5          instr1.setText(R.string.ishiharaInstruction1) ;
6            instr2 .setText(R.string.ishiharaInstruction2) ;
7            instr3 .setText(R.string.ishiharaInstruction3) ;
8
9
10         skip.setOnClickListener(new View.OnClickListener() {
11
12         @Override
13         public void onClick(View v) {
14
15
16
17           Intent ishiharaIntent = new Intent(Instructions.this,
                 IshiharaTest.class);
18           ishiharaIntent.putExtra("Username", username);
19         startActivity(ishiharaIntent);
20         finish();
21         }
22       });
23
24
25        }
26        else if (test.equals("neitz")){
27
```

```
28          instr0.setText("Neitz") ;
29          instr1.setText(R.string.neitzInstruction1) ;
30            instr2 .setText(R.string.neitzInstruction2) ;
31            instr3 .setText(R.string.neitzInstruction3) ;
32
33
34          skip.setOnClickListener(new View.OnClickListener() {
35
36          @Override
37          public void onClick(View v) {
38
39          Bundle extras = getIntent().getExtras();
40          final String username = extras.getString("Username");
41
42           Intent neitzIntent = new Intent(Instructions.this, NeitzTest.class);
43           neitzIntent.putExtra("Username", username);
44          startActivity(neitzIntent);
45          finish();
46          }
47        });
48
49         }
50         else if (test.equals("amsler")){
51
52          instr0.setText("Amsler") ;
53          instr1.setText(R.string.amslerInstruction1) ;
54            instr2 .setText(R.string.amslerInstruction2) ;
55            instr3 .setText(R.string.amslerInstruction3) ;
56            instr4 .setText(R.string.amslerInstruction4) ;
57
58          skip.setOnClickListener(new View.OnClickListener() {
59
60          @Override
61          public void onClick(View v) {
62
63           Intent amslerIntent = new Intent(Instructions.this, Amsler.class);
64           amslerIntent.putExtra("Username", username);
65          startActivity(amslerIntent);
66          finish();
```

```
67          }
68        });
69         }
70                          ...
71                          ...
72                          ...
73
74        else if (test.equals("redDesaturation")){
75
76          instr0.setText("Red Desaturation") ;
77          instr1.setText(R.string.redDesaturationInstruction1) ;
78            instr2 .setText(R.string.redDesaturationInstruction2) ;
79            instr3 .setText(R.string.redDesaturationInstruction3) ;
80
81
82          skip.setOnClickListener(new View.OnClickListener() {
83
84          @Override
85          public void onClick(View v) {
86
87           Intent redDesaturationIntent = new Intent(Instructions.this,
                 RedDesaturation.class);
88           redDesaturationIntent.putExtra("Username", username);
89          startActivity(redDesaturationIntent);
90          finish();
91          }
92        });
93         }
94
95        }
```

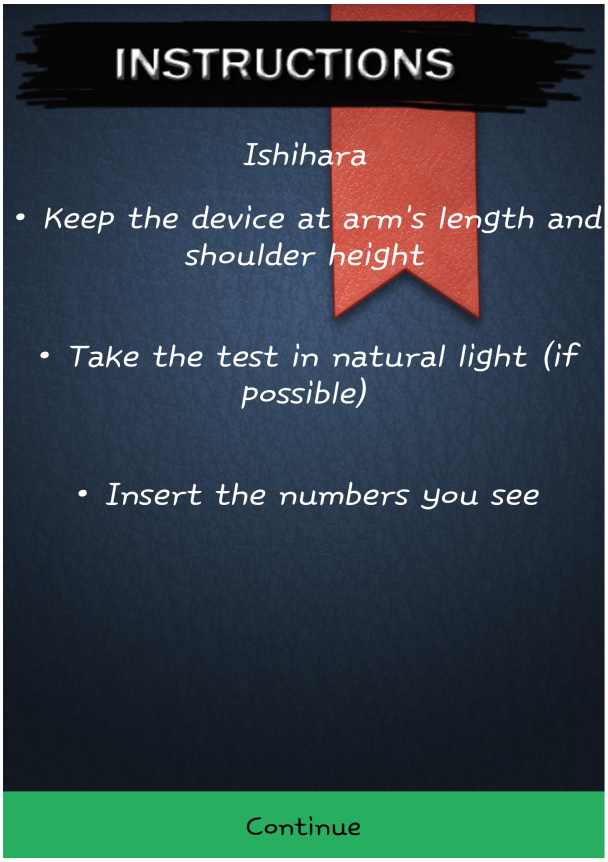


Figure 3.6: The instructions activity

Every test has its own set of instructions. Ishihara's instructions are:

1. Keep the device at arm's length and shoulder height
2. Take the test in natural light
3. Insert the number you see

With these instructions in mind the user can easily take the Ishihara test and see the result/score right afterwards.

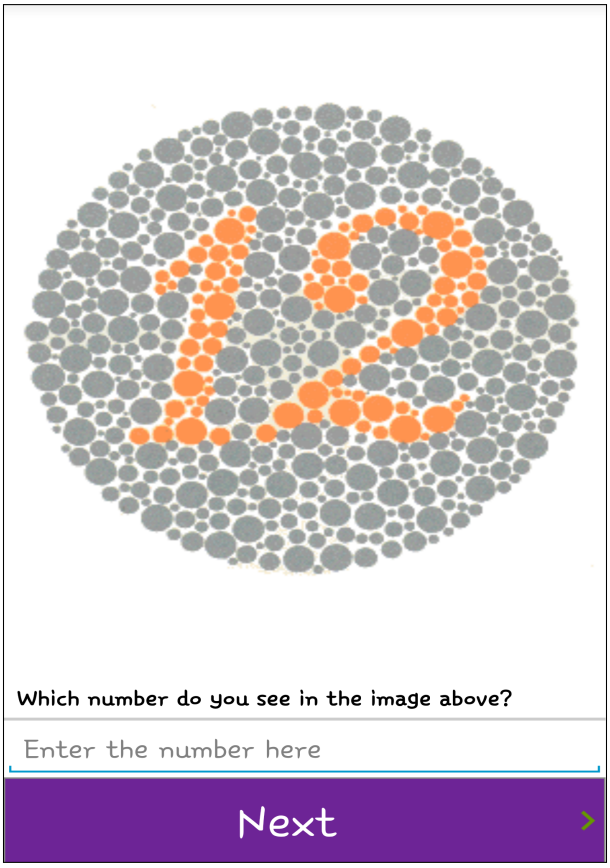


Figure 3.7: The Ishihara Test

The layout of the Ishihara test is described by the following xml code and is nothing more than an ImageView, a TextView, an EditText and a button.

Listing 3.10: The Ishihara layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/display"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"    />
```

```
13
14
15      <TextView
16          style="?android:listSeparatorTextViewStyle"
17          android:layout_width="match_parent"
18          android:layout_height="wrap_content"
19          android:textColor="#000000"
20          android:textAllCaps="false"
21          android:text="@string/question" />
22
23      <EditText
24          android:id="@+id/input"
25          android:layout_width="match_parent"
26          android:layout_height="wrap_content"
27          android:ems="10"
28          android:hint="@string/ishihara_input_hint"
29          android:inputType="phone" />
30
31
32      <Button
33          android:id="@+id/next_button"
34          android:layout_width="match_parent"
35          android:layout_height="60dp"
36                ...
37          android:textSize="30sp" />
38
39  </LinearLayout>
```

The source code of the Ishihara test is quite simple and can be found in the listing below. In the beginning of the test's implementation the best way to keep track of the correct answers was decided to be a non-random order of the images displayed. Since the images are not produced randomly, it was efficient to keep a list with the correct answers and then keep another list for the user's answers. Comparing these lists results in the amount of correct answers the user inserted and this results to their score (percentage).

Listing 3.11: The IshiharaTest.java

```
1  public class IshiharaTest extends Activity {
2
```

```
3     //Declaring variables
4     ImageView display ;
5     EditText input;
6     Button mNextButton;
7     int mCurrentIndex = 0;
8     //Declaring the correct answers
9     int realAnswers[] = {12,15,16,26,29,3,42,45,5,6,7,73,74,8};
10    List<Integer> realAnswersList = new ArrayList<Integer>();
11    String usrAnswers[];
12    List<Integer> usrAnswersList = new ArrayList<Integer>();
13
14    String scoreIshihara;
15    int correctAnswers;
16    int number;
17    int counter=0;
18
19     @Override
20     protected void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22
23         //make activity full screen
24         requestWindowFeature(Window.FEATURE_NO_TITLE);
25         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
26                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
27         //Populate the layout
28         setContentView(R.layout.ishihara_layout);
29
30         display = (ImageView)findViewById(R.id.display);
31         updateImage();
32
33         //Read what the user inserted
34       input = (EditText)findViewById(R.id.input);
35         //input.setInputType(InputType.TYPE_CLASS_NUMBER);
36
37       //Populate the list with the correct answers
38         for (int index = 0; index < realAnswers.length; index++)
39         {
40             realAnswersList.add(realAnswers[index]);
41         }
```

```
42          //Set the listener on the Next button
43          mNextButton = (Button)findViewById(R.id.next_button);
44          mNextButton.setOnClickListener(new View.OnClickListener() {
45              //@Override
46              public void onClick(View v) {
47
48                  //Check if EditText is empty
49                  String ishiharaString = input.getText().toString();
50                  if(ishiharaString.equals(""))
51                  {
52                     Message.message(getApplicationContext(), "Please insert a
                           number first");
53                  }
54                  else
55                  {
56                     number = Integer.parseInt(input.getText().toString());
57                     usrAnswersList.add(number);
58
59                     mCurrentIndex++;
60                      updateImage();
61                      //After every next clear the editText string
62                      input.getText().clear();
63                      //when the index reaches a limit call ScoreIshihara
64                      if (mCurrentIndex == 14 ){
65
66                        scoreIshihara();
67
68                      }
69                  }
70
71              }
72          });
73
74      }
75
76      private void scoreIshihara() {
77
78        //Get the number of correct answers
79        for(int i=0;i<realAnswers.length;i++){
```

```
80          if(usrAnswersList.get(i).equals(realAnswers[i]) ){
81             correctAnswers++;
82          }
83       }
84       //If the user answers everything correct
85       if (realAnswersList.equals(usrAnswersList))
86       {
87
88          Bundle extras = getIntent().getExtras();
89          String username = extras.getString("Username");
90          //Create an Intent to navigate to IshiharaScore.java
91          Intent scoreIshihara = new Intent(getApplicationContext(),
                IshiharaScore.class);
92          scoreIshihara.putExtra("PassFail","Passed");
93          scoreIshihara.putExtra("ColumnToInsert","ISHIHARA");
94          scoreIshihara.putExtra("Username",username);
95          scoreIshihara.putExtra("CorrectAnswers", correctAnswers);
96          startActivity(scoreIshihara);
97          finish();
98
99       }
100      else
101      {
102
103         //if the user has not answered correctly
104         Bundle extras = getIntent().getExtras();
105         String username = extras.getString("Username");
106         //Navigate to Ishihara Score but with another set of info to be passed
                on
107         Intent scoreIshihara = new Intent(getApplicationContext(),
                IshiharaScore.class);
108         scoreIshihara.putExtra("PassFail","Failed");
109         scoreIshihara.putExtra("ColumnToInsert","ISHIHARA");
110         scoreIshihara.putExtra("Username",username);
111         scoreIshihara.putExtra("CorrectAnswers", correctAnswers);
112         startActivity(scoreIshihara);
113         finish();
114      }
115   }
```

```
116     // change the image displayed
117     private void updateImage() {
118         switch (mCurrentIndex){
119         case 0:
120          display.setImageResource(R.drawable.is_12);
121          break;
122         case 1:
123          display.setImageResource(R.drawable.is_15);
124          break;
125         case 2:
126          display.setImageResource(R.drawable.is_16);
127          break;
128         case 3:
129          display.setImageResource(R.drawable.is_26);
130          break;
131                     ...
132                     ...
133                     ...
134         case 13:
135          display.setImageResource(R.drawable.is_8);
136          break;
137         }
138     }
```

The Neitz Test

The Neitz test is very similar to the Ishihara test, both in theory and implementation. It's instructions are:

1. Keep the device at arm's length and shoulder height
2. Take the test in natural light
3. Select the shape you see

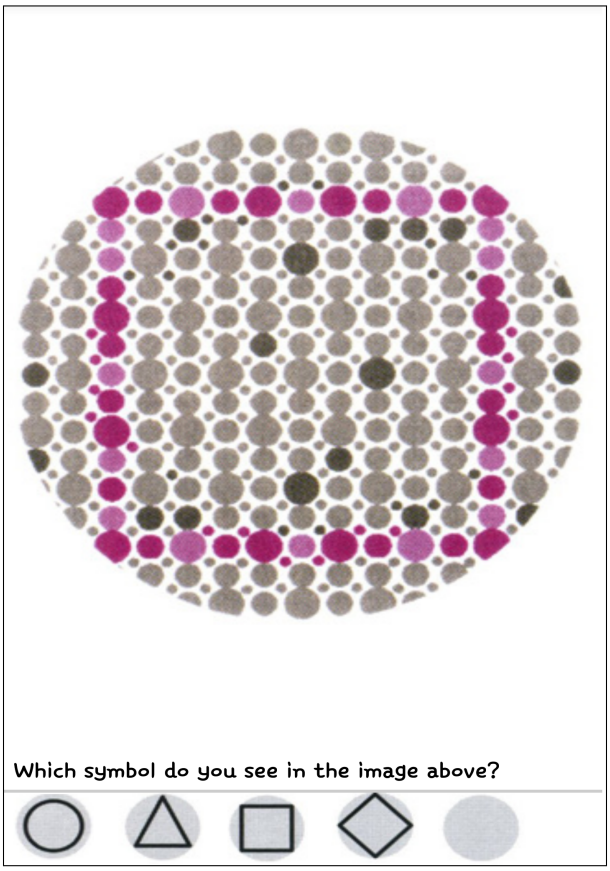


Figure 3.8: The Neitz Test

The layout of the NeitzTest.java is composed of an ImageView, a TextView and a nested linear layout with four Buttons.

Listing 3.12: The Neitz test layout

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/display_neitz"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"   />

    <TextView
        style="?android:listSeparatorTextViewStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textAllCaps="false"
        android:text="@string/question_neitz" />

      <LinearLayout
          android:layout_width="wrap_content"
         android:layout_height="wrap_content"
            >

       <ImageButton
           android:id="@+id/imageButton"
           android:layout_width="0dp"
           android:layout_height="wrap_content"
           android:layout_weight="0.2"
           style="?android:attr/borderlessButtonStyle"
           android:src="@drawable/n_b_1" />

       <ImageButton
           android:id="@+id/imageButton2"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_weight="0.2"
           style="?android:attr/borderlessButtonStyle"
           android:src="@drawable/n_b_2" />
```

```
42        <ImageButton
43            android:id="@+id/imageButton3"
44            android:layout_width="wrap_content"
45            android:layout_height="wrap_content"
46            android:layout_weight="0.2"
47            style="?android:attr/borderlessButtonStyle"
48            android:src="@drawable/n_b_3" />
49        <ImageButton
50            android:id="@+id/imageButton4"
51            android:layout_width="wrap_content"
52            android:layout_height="wrap_content"
53            android:layout_weight="0.2"
54            style="?android:attr/borderlessButtonStyle"
55            android:src="@drawable/n_b_4" />
56        <ImageButton
57            android:id="@+id/imageButton5"
58            android:layout_width="wrap_content"
59            android:layout_height="wrap_content"
60            android:layout_weight="0.2"
61            style="?android:attr/borderlessButtonStyle"
62            android:src="@drawable/n_b_5" />
63
64      </LinearLayout>
65
66  </LinearLayout>
```

The NeitzTest.java follows the same methodology as the Ishihara test. It uses two lists to keep track of the default answers and the user's answers and then comparing them to produce the result.

Listing 3.13: The NeitzTest.java

```
1  public class NeitzTest extends Activity {
2
3     ImageView display ;
4     ImageButton imgButton,imgButton2,imgButton3,imgButton4,imgButton5;
5     //Button mNextButton;
6     int mCurrentIndex = 0;
7     //store the correct answers
8     String realAnswers[] =
```

```
        {"square","circle","circle","square","triangle","triangle","circle","square","triangle"
    List<String> realAnswersList = new ArrayList<String>();
    String usrAnswers[];
    List<String> usrAnswersList = new ArrayList<String>();

    int correctAnswers;

     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         //full screen
         requestWindowFeature(Window.FEATURE_NO_TITLE);
         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                              WindowManager.LayoutParams.FLAG_FULLSCREEN);

         setContentView(R.layout.neitz_layout);

         //Populate the list with the correct answers
         for (int index = 0; index < realAnswers.length; index++)
         {
             realAnswersList.add(realAnswers[index]);
         }

         display = (ImageView)findViewById(R.id.display_neitz);
         updateImage();

         // This imageButton is for the Circle
         imgButton = (ImageButton)findViewById(R.id.imageButton);
         imgButton.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View v) {
                 mCurrentIndex++;
                 updateImage();
                 //add to the users answers list
                 usrAnswersList.add("circle");

                 if (mCurrentIndex == 9 ){
                   scoreNeitz();
                 }
```

```
47              }
48          });
49
50          // This imageButton is for the triangle
51          imgButton2 = (ImageButton)findViewById(R.id.imageButton2);
52          imgButton2.setOnClickListener(new View.OnClickListener() {
53              @Override
54              public void onClick(View v) {
55                 mCurrentIndex++;
56                  updateImage();
57
58                  usrAnswersList.add("triangle");
59
60                  if (mCurrentIndex == 9 ){
61                    scoreNeitz();
62                  }
63              }
64          });
65
66          // This imageButton is for the square
67          imgButton3 = (ImageButton)findViewById(R.id.imageButton3);
68          imgButton3.setOnClickListener(new View.OnClickListener() {
69              @Override
70              public void onClick(View v) {
71                 mCurrentIndex++;
72                  updateImage();
73
74                  usrAnswersList.add("square");
75
76                  if (mCurrentIndex == 9 ){
77                    scoreNeitz();
78                  }
79              }
80          });
81
82          // This imageButton is for the diamond
83          imgButton4 = (ImageButton)findViewById(R.id.imageButton4);
84          imgButton4.setOnClickListener(new View.OnClickListener() {
85              @Override
```

```
86              public void onClick(View v) {
87                 mCurrentIndex++;
88                  updateImage();
89
90                  usrAnswersList.add("diamond");
91
92                  if (mCurrentIndex == 9 ){
93                    scoreNeitz();
94                  }
95              }
96          });
97
98          // This imageButton is for nothing
99          imgButton5 = (ImageButton)findViewById(R.id.imageButton5);
100         imgButton5.setOnClickListener(new View.OnClickListener() {
101             @Override
102             public void onClick(View v) {
103                mCurrentIndex++;
104                 updateImage();
105
106                 usrAnswersList.add("nothing");
107
108                 if (mCurrentIndex == 9 ){
109                   scoreNeitz();
110                 }
111             }
112         });
113
114
115     }
116
117     private void updateImage() {
118         switch (mCurrentIndex){
119         case 0:
120          display.setImageResource(R.drawable.n_1);
121          break;
122         case 1:
123          display.setImageResource(R.drawable.n_2);
124          break;
```

```
125         case 2:
126          display.setImageResource(R.drawable.n_3);
127          break;
128                     ...
129                     ...
130                     ...
131
132         case 8:
133          display.setImageResource(R.drawable.n_9);
134          break;
135
136         }
137     }
138
139     private void scoreNeitz(){
140
141       //Get the number of correct answers
142       for(int i=0;i<realAnswers.length;i++){
143          if(usrAnswersList.get(i).equals(realAnswers[i]) ){
144             correctAnswers++;
145          }
146       }
147
148       if (realAnswersList.equals(usrAnswersList))
149       {
150          //correctAnswer++;
151          //scoreIshihara = "Passed";
152          //Toast.makeText(IshiharaTest.this, "You " + scoreAmsler + " the
                Test", Toast.LENGTH_SHORT).show();
153
154          Bundle extras = getIntent().getExtras();
155          String username = extras.getString("Username");
156
157          Intent scoreNeitz = new Intent(getApplicationContext(),
                NeitzScore.class);
158          scoreNeitz.putExtra("PassFail","Passed");
159          scoreNeitz.putExtra("ColumnToInsert","NEITZ");
160          scoreNeitz.putExtra("Username",username);
161          scoreNeitz.putExtra("CorrectAnswers", correctAnswers);
```

```
162             startActivity(scoreNeitz);
163             finish();
164
165         }
166         else
167         {
168             Bundle extras = getIntent().getExtras();
169             String username = extras.getString("Username");
170
171             Intent scoreNeitz = new Intent(getApplicationContext(),
                    NeitzScore.class);
172             scoreNeitz.putExtra("PassFail","Failed");
173             scoreNeitz.putExtra("ColumnToInsert","NEITZ");
174             scoreNeitz.putExtra("Username",username);
175             scoreNeitz.putExtra("CorrectAnswers", correctAnswers);
176             startActivity(scoreNeitz);
177             finish();
178         }
179
180
181     }
182 }
```

The Amsler Grid

The Amsler grid tests for AMD (age related macular degeneration). The instruction set for the proper use of the grid is:

1. Put on your reading glasses (if you wear glasses)
2. Keep the device in arm's length and shoulder height
3. Cover one eye
4. With the other eye, focus on the black dot in the center of the grid and try to establish if there is any blurriness or distortion of the grid

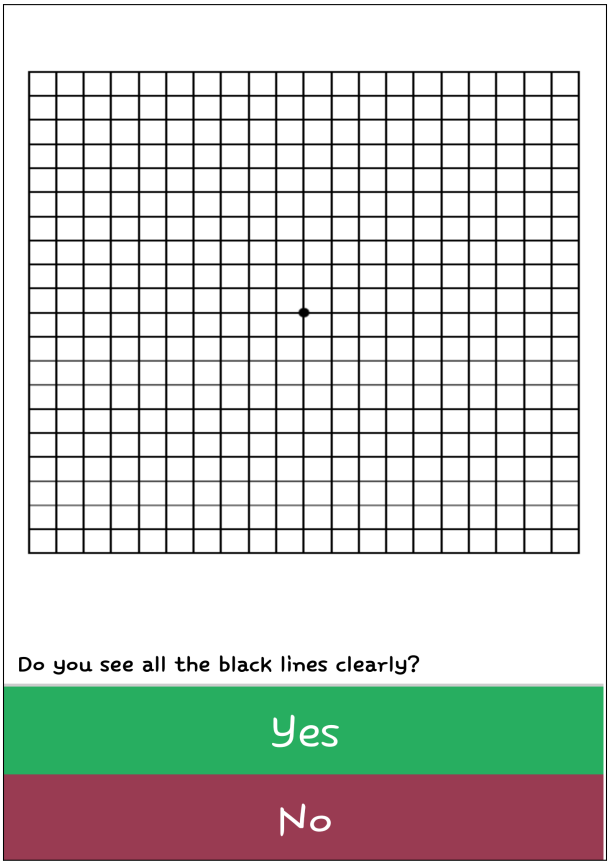


Figure 3.9: The Amsler Grid

The xml layout of the Amsler grid activity is shown below and is composed of an ImageView of the grid, a TextView and a nested linear layout of two buttons.

Listing 3.14: The Amsler grid layout

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:background="#ffffff"
6       >
7
8      <ImageView
9          android:id="@+id/display_amsler"
10         android:layout_width="fill_parent"
11         android:layout_height="wrap_content"
12         android:layout_above="@+id/textv"
```

```
13          android:layout_alignParentTop="true" />
14
15      <LinearLayout
16          android:id="@+id/linearLayout1"
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          android:layout_alignParentBottom="true"
20          android:layout_alignParentLeft="true"
21          android:layout_alignParentRight="true"
22          android:orientation="vertical" >
23
24          <Button
25              android:id="@+id/yes_button"
26              style="?android:attr/borderlessButtonStyle"
27              android:layout_width="match_parent"
28                    ...
29              android:textSize="30sp" />
30
31          <Button
32              android:id="@+id/no_button"
33              style="?android:attr/borderlessButtonStyle"
34              android:layout_width="match_parent"
35                      ...
36              android:textSize="30sp" />
37      </LinearLayout>
38
39      <TextView
40          android:id="@+id/textv"
41          style="?android:listSeparatorTextViewStyle"
42          android:layout_width="match_parent"
43          android:layout_height="wrap_content"
44          android:layout_above="@+id/linearLayout1"
45                ...
46          android:textColor="#000000" />
47
48  </RelativeLayout>
```

The Amsler.java, as shown below, implements two buttons with an onClickListener on each button. When the user clicks on either one it creates an intent that navigates

the user to their test result.

Listing 3.15: The Amsler.java

```
public class Amsler extends Activity {
   ImageView display ;
   Button mYesButton;
   Button mNoButton;
   String scoreAmsler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //full screen mode
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                             WindowManager.LayoutParams.FLAG_FULLSCREEN);
        //inflate the layout
        setContentView(R.layout.amsler_layout);

        display = (ImageView)findViewById(R.id.display_amsler);
        display.setImageResource(R.drawable.amsler);

        //listener on the Yes button
        mYesButton = (Button)findViewById(R.id.yes_button);
        mYesButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

               Bundle extras = getIntent().getExtras();
            String username = extras.getString("Username");
               //Creation of an intent and extra info
               Intent scoreAmsler = new Intent(getApplicationContext(),
                   AmslerScore.class);
               scoreAmsler.putExtra("PassFail","Passed");
               scoreAmsler.putExtra("ColumnToInsert","AMSLER");
               scoreAmsler.putExtra("Username",username);
               startActivity(scoreAmsler);
               finish();
            }
```

```
36          });
37        //listener on the No button
38          mNoButton = (Button)findViewById(R.id.no_button);
39          mNoButton.setOnClickListener(new View.OnClickListener() {
40              @Override
41              public void onClick(View v) {
42
43                 Bundle extras = getIntent().getExtras();
44              String username = extras.getString("Username");
45                 //Creation of an intent and extra info
46                 Intent scoreAmsler = new Intent(getApplicationContext(),
                       AmslerScore.class);
47                 scoreAmsler.putExtra("PassFail","Failed");
48                 scoreAmsler.putExtra("ColumnToInsert","AMSLER");
49                 scoreAmsler.putExtra("Username",username);
50                 startActivity(scoreAmsler);
51                 finish();
52              }
53          });
54
55      }
56  }
```

The Astigmatism Test

The instructions of the Astigmatism test are:

1. Put on your glasses (if you wear glasses)
2. Cover your eye
3. Hold the device in arm's length and shoulder height
4. Look at the center white circle and become aware of the lines radiating out from the center and note if lines appear blacker, grayer, less distinct or wavy.
5. Repeat for the other eye

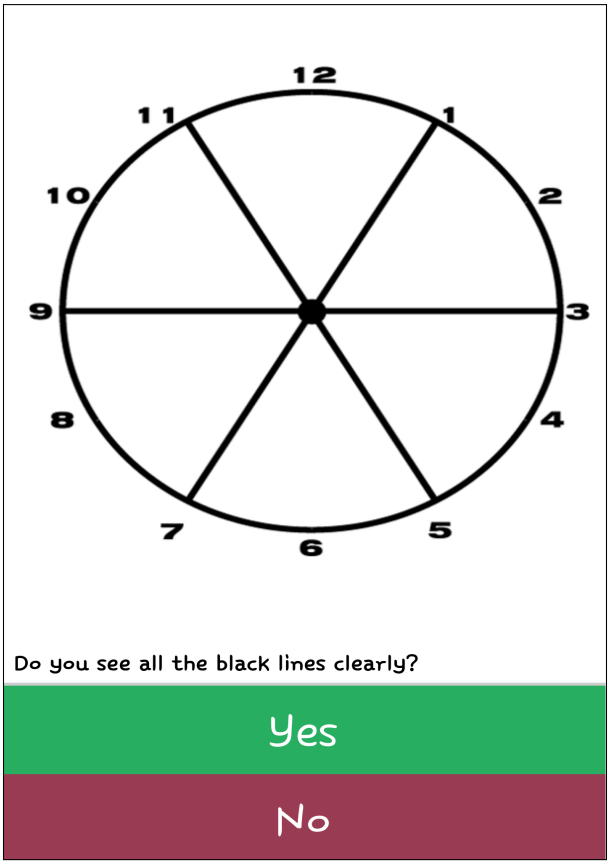


Figure 3.10: The Astigmatism Test

The layout of the Astigmatism test is similar to the Amsler grid test. It consists of an ImageView, a TextView and a linear layout (vertical orientation) with two buttons.

Listing 3.16: The Astigmatism Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/display_astigmatism"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"    />
```

```
13
14      <TextView
15          style="?android:listSeparatorTextViewStyle"
16          android:layout_width="match_parent"
17          android:layout_height="wrap_content"
18          android:text="@string/question_astigmatism"
19          android:textAllCaps="false"
20          android:textColor="#000000" />
21
22        <LinearLayout
23            android:layout_width="match_parent"
24           android:layout_height="wrap_content"
25           android:orientation="vertical" >
26
27           <Button
28              android:id="@+id/yes_button"
29              android:text="@string/yes"
30                       ...
31              android:background="@color/login_green"
32  />
33
34           <Button
35              android:id="@+id/no_button"
36              android:text="@string/no"
37                       ...
38              android:background="#993b52"
39              />
40
41        </LinearLayout>
42
43  </LinearLayout>
```

The Astigmatism.java populates the layout and after the user clicks on the buttons (yes or no), the result is shown.

Listing 3.17: The Astigmatism.java

```
1  public class Astigmatism extends Activity {
2
3     ImageView display ;
```

```
4     EditText input;
5     Button mYesButton;
6     Button mNoButton;
7     int mCurrentIndex = 0;
8     int yesCounter=0;
9     int noCounter=0;
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         //full screen mode
15         requestWindowFeature(Window.FEATURE_NO_TITLE);
16         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
17                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
18         //inflate the layout
19         setContentView(R.layout.astigmatism_layout);
20
21         display = (ImageView)findViewById(R.id.display_astigmatism);
22         updateImage();
23
24         EditText input = new EditText(this);
25         input.setInputType(InputType.TYPE_CLASS_NUMBER);
26         //add listener on the Yes button
27         mYesButton = (Button)findViewById(R.id.yes_button);
28         mYesButton.setOnClickListener(new View.OnClickListener() {
29             @Override
30             public void onClick(View v) {
31                yesCounter++;
32                mCurrentIndex++;
33                 //mCurrentIndex = (mCurrentIndex + 1) % 5;
34
35                 updateImage();
36                 showResult();
37             }
38         });
39         //add listener on the No button
40         mNoButton = (Button)findViewById(R.id.no_button);
41         mNoButton.setOnClickListener(new View.OnClickListener() {
42             @Override
```

```
43            public void onClick(View v) {
44               noCounter++;
45               mCurrentIndex++;
46                //mCurrentIndex = (mCurrentIndex + 1) % 5;
47
48                updateImage();
49                showResult();
50            }
51
52        });
53
54     }
55
56    //method that changes the image displayed
57    private void updateImage() {
58         switch (mCurrentIndex){
59         case 0:
60          display.setImageResource(R.drawable.ast1);
61          break;
62                     ...
63                     ...
64                     ...
65
66         case 4:
67          display.setImageResource(R.drawable.ast5);
68          break;
69         }
70
71     }
72    //method for showing the result
73    private void showResult() {
74       //if all correct
75       if(yesCounter == 5 && noCounter == 0){
76          Bundle extras = getIntent().getExtras();
77          String username = extras.getString("Username");
78          //creation of intent
79          Intent scoreAstigmatism = new Intent(getApplicationContext(),
                AstigmatismScore.class);
80          scoreAstigmatism.putExtra("PassFail","Passed");
```

```
81          scoreAstigmatism.putExtra("ColumnToInsert","ASTIGMATISM");
82          scoreAstigmatism.putExtra("Username",username);
83          scoreAstigmatism.putExtra("CorrectAnswers", yesCounter);
84          startActivity(scoreAstigmatism);
85          finish();
86         }
87       //if not all correct
88       if( (noCounter+yesCounter) == 5 && noCounter != 0 ){
89          Bundle extras = getIntent().getExtras();
90          String username = extras.getString("Username");
91          //creation of intent
92             Intent scoreAstigmatism = new Intent(getApplicationContext(),
                   AstigmatismScore.class);
93             scoreAstigmatism.putExtra("PassFail","Failed");
94          scoreAstigmatism.putExtra("ColumnToInsert","ASTIGMATISM");
95          scoreAstigmatism.putExtra("Username",username);
96          scoreAstigmatism.putExtra("CorrectAnswers", yesCounter);
97             startActivity(scoreAstigmatism);
98             finish();
99          }
100
101
102    }
103
104 }
```

The Myopia/Hyperopia Test

Instructions of the Myopia/Hyperopia test:

1. Hold the device in arm's length and shoulder height
2. Take the test in natural light
3. Cover one eye
4. Focus on the images shown and try to identify in which colored background the black letters are bolder/sharper
5. Repeat for the other eye

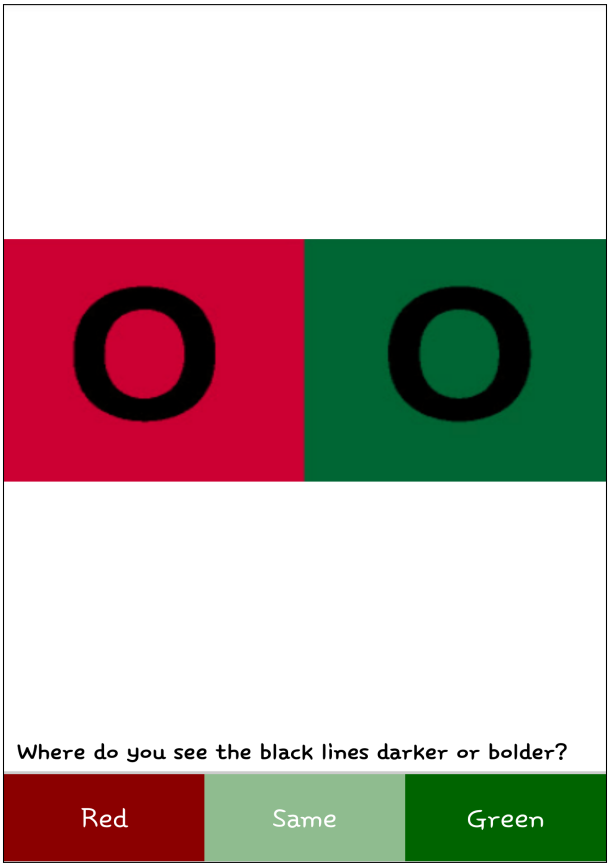


Figure 3.11: The Myopia/Hyperopia Test

The layout of this screen is described by an ImageView, a TextView and a linear layout with two buttons.

Listing 3.18: The Myopia/Hyperopia layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/display_myopia"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"   />
```

```
13
14    <TextView
15        style="?android:listSeparatorTextViewStyle"
16        android:layout_width="match_parent"
17        android:layout_height="wrap_content"
18                  ...
19        android:text="@string/question_myopia" />
20
21       <LinearLayout
22           android:layout_width="wrap_content"
23          android:layout_height="wrap_content"
24
25             >
26
27          <Button
28              android:id="@+id/red_button"
29              style="?android:attr/borderlessButtonStyle"
30              android:layout_width="270dp"
31                      ...
32              android:textColor="#FFFFFF"
33               />
34
35          <Button
36             android:id="@+id/same_button"
37             android:layout_weight="0.3"
38             android:text="@string/same"
39                      ...
40             android:background="@color/DarkSeaGreen" />
41
42          <Button
43             android:id="@+id/green_button"
44             android:layout_weight="0.3"
45             android:text="@string/green"
46                      ...
47             android:background="@color/DarkGreen"
48             />
49
50
51       </LinearLayout>
```

```
52
53 </LinearLayout>
```

The user must distinguish in which background the black lines (the letter 'O') is sharper/bolder (or if it seems the same). Depending on the button pressed, the user will have a different result. The distance vision is deficient if the user sees the letter O sharper or blacker, either in the red or the green color. Generally, if the O is blacker or sharper in the red part, the user probably has myopia or has a myopic tendency. Conversely, if the O is blacker or sharper in the green part, the user is probably hyperope or has a hyperopic tendency.

Listing 3.19: The Myopia.java

```
1  public class Myopia extends Activity {
2
3     ImageView display ;
4     //ImageView display2 ;
5     Button mRedButton;
6     Button mGreenButton;
7     Button mSameButton;
8     int mCurrentIndex = 0;
9     int redCounter, sameCounter, greenCounter =0;
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         //full screen mode
15         requestWindowFeature(Window.FEATURE_NO_TITLE);
16         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
17                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
18
19         setContentView(R.layout.myopia_layout);
20
21         display = (ImageView)findViewById(R.id.display_myopia);
22         updateImage();
23         //add listener to the Red button
24         mRedButton = (Button)findViewById(R.id.red_button);
25         mRedButton.setOnClickListener(new View.OnClickListener() {
26             @Override
```

```
27              public void onClick(View v) {
28                 mCurrentIndex++;
29                 redCounter++;
30                 updateImage();
31                 if(mCurrentIndex == 2){
32                    showResult();
33                  }
34
35              }
36          });
37          //add listener to the Same button
38          mSameButton = (Button)findViewById(R.id.same_button);
39          mSameButton.setOnClickListener(new View.OnClickListener() {
40              @Override
41              public void onClick(View v) {
42                 mCurrentIndex++;
43                 sameCounter++;
44                 updateImage();
45                 if(mCurrentIndex == 2){
46                    showResult();
47                  }
48
49              }
50          });
51          //add listener to the Green button
52          mGreenButton = (Button)findViewById(R.id.green_button);
53          mGreenButton.setOnClickListener(new View.OnClickListener() {
54              @Override
55              public void onClick(View v) {
56                 mCurrentIndex++;
57                 greenCounter++;
58                 updateImage();
59                 if(mCurrentIndex == 2){
60                    showResult();
61                  }
62
63              }
64          });
65
```

```
66      }
67      private void updateImage() {
68          switch (mCurrentIndex){
69          case 0:
70           display.setImageResource(R.drawable.myopia3);
71           break;
72          case 1:
73           display.setImageResource(R.drawable.myopia2);
74           break;
75
76          }
77
78      }
79      //method to show the result
80      private void showResult() {
81
82        // When Red Button pressed at least once print message for myopia
83        if(redCounter == 2 || redCounter == 1){
84           Bundle extras = getIntent().getExtras();
85           String username = extras.getString("Username");
86           //creation of an intent
87           Intent scoreMyopia = new Intent(getApplicationContext(),
                 MyopiaScore.class);
88           scoreMyopia.putExtra("PassFail","Failed");
89           scoreMyopia.putExtra("ColumnToInsert","MYOPIA");
90           scoreMyopia.putExtra("Username",username);
91           scoreMyopia.putExtra("ButtonPressed","red");
92           startActivity(scoreMyopia);
93           finish();
94        }
95        // When pressed "Same" twice you are ok
96        if(sameCounter == 2){
97           Bundle extras = getIntent().getExtras();
98           String username = extras.getString("Username");
99           //creation of an intent
100          Intent scoreMyopia = new Intent(getApplicationContext(),
                 MyopiaScore.class);
101          scoreMyopia.putExtra("PassFail","Passed");
102          scoreMyopia.putExtra("ColumnToInsert","MYOPIA");
```

```
103             scoreMyopia.putExtra("Username",username);
104             scoreMyopia.putExtra("ButtonPressed","same");
105             startActivity(scoreMyopia);
106             finish();
107         }
108         //When Green button pressed at least once print message for hyperopia
109         if(greenCounter == 2 || greenCounter == 1){
110             Bundle extras = getIntent().getExtras();
111             String usersEmail = extras.getString("UsersEmail");
112             //creation of an intent
113             Intent scoreMyopia = new Intent(getApplicationContext(),
                    MyopiaScore.class);
114             scoreMyopia.putExtra("PassFail","Failed");
115             scoreMyopia.putExtra("ColumnToInsert","MYOPIA");
116             scoreMyopia.putExtra("UsersEmail",usersEmail);
117             scoreMyopia.putExtra("ButtonPressed","green");
118             startActivity(scoreMyopia);
119             finish();
120         }
121     }
122
123
124 }
```

The Contrast Sensitivity Test

The instructions for taking this test are:

1. Keep the device in arm's length and shoulder's height
2. Take the test in natural light
3. Select the correct direction of the letter 'C'

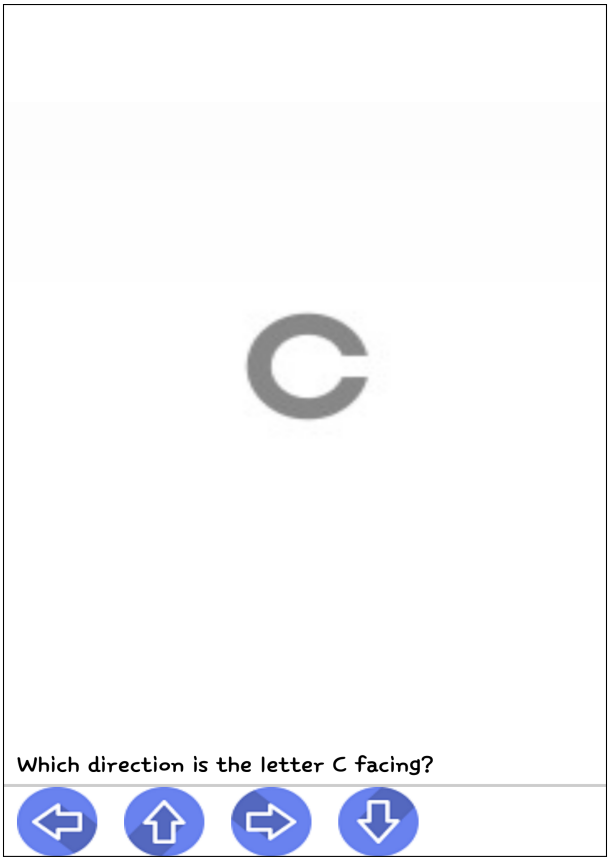


Figure 3.12: The Contrast Sensitivity Test

The layout of the Contrast Sensitivity test, as shown below, is structured using an ImageView for the descending in contrast images, a TextView and a linear layout with four button used for the direction of the letter 'C'.

Listing 3.20: The Contrast Sensitivity layout

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/display_contrast"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"   />

    <TextView
        style="?android:listSeparatorTextViewStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textAllCaps="false"
        android:text="@string/question_contrast" />

      <LinearLayout
          android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:weightSum="1"      >

       <ImageButton
           android:id="@+id/imageButton"
           android:layout_width="0dp"
           android:layout_height="wrap_content"
           android:layout_weight="0.25"
           style="?android:attr/borderlessButtonStyle"
           android:src="@drawable/arrow_left" />

               ...
               ...

       <ImageButton
           android:id="@+id/imageButton4"
           android:layout_width="0dp"
           android:layout_height="wrap_content"
```

```
42            android:layout_weight="0.25"
43            style="?android:attr/borderlessButtonStyle"
44            android:src="@drawable/arrow_down" />
45
46        </LinearLayout>
47
48    </LinearLayout>
```

As for the java code implemented, it builds two lists which hold the default answers and the user's answers. In the end it compares the two lists, produces the proper result/score and directs the user to the result/score activity.

Listing 3.21: The ContrastSensitivity.java

```
1  public class ContrastSensitivity extends Activity {
2
3     ImageView display ;
4     ImageButton imgButton,imgButton2,imgButton3,imgButton4;
5     int mCurrentIndex = 0;
6     //Store the correct answers
7     String realAnswers[] = {"right","down","left","up","left","up",
8     "right","down","down","left","up","right",
9     "left","up","right","down"};
10    List<String> realAnswersList = new ArrayList<String>();
11    String usrAnswers[];
12    List<String> usrAnswersList = new ArrayList<String>();
13
14    int correctAnswers;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         //full screen mode
20         requestWindowFeature(Window.FEATURE_NO_TITLE);
21         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
22                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
23         //inflate the layout
24         setContentView(R.layout.contrast_layout);
25
26         //Populate the list with the correct answers
```

```
27          for (int index = 0; index < realAnswers.length; index++)
28          {
29              realAnswersList.add(realAnswers[index]);
30          }
31
32          display = (ImageView)findViewById(R.id.display_contrast);
33          updateImage();
34          //add listener on the button
35          // This imageButton is for the Left
36          imgButton = (ImageButton)findViewById(R.id.imageButton);
37          imgButton.setOnClickListener(new View.OnClickListener() {
38              @Override
39              public void onClick(View v) {
40                  mCurrentIndex++;
41                  updateImage();
42
43                  usrAnswersList.add("left");
44
45                  if (mCurrentIndex == 16 ){
46                    contrastScore();
47                  }
48              }
49          });
50          //add listener on the button
51          // This imageButton is for the UP
52          imgButton2 = (ImageButton)findViewById(R.id.imageButton2);
53          imgButton2.setOnClickListener(new View.OnClickListener() {
54              @Override
55              public void onClick(View v) {
56                 mCurrentIndex++;
57                  updateImage();
58
59                  usrAnswersList.add("up");
60
61                  if (mCurrentIndex == 16 ){
62                    contrastScore();
63                  }
64              }
65          });
```

```
66          //add listener on the button
67          // This imageButton is for the Right
68          imgButton3 = (ImageButton)findViewById(R.id.imageButton3);
69          imgButton3.setOnClickListener(new View.OnClickListener() {
70              @Override
71              public void onClick(View v) {
72                 mCurrentIndex++;
73                  updateImage();
74
75                  usrAnswersList.add("right");
76
77                  if (mCurrentIndex == 16 ){
78                    contrastScore();
79                  }
80              }
81          });
82          //add listener on the button
83          // This imageButton is for the DOWN
84          imgButton4 = (ImageButton)findViewById(R.id.imageButton4);
85          imgButton4.setOnClickListener(new View.OnClickListener() {
86              @Override
87              public void onClick(View v) {
88                 mCurrentIndex++;
89                  updateImage();
90
91                  usrAnswersList.add("down");
92
93                  if (mCurrentIndex == 16 ){
94                    contrastScore();
95                  }
96              }
97          });
98
99
100     }
101   //method to change the image displayed
102     private void updateImage() {
103         switch (mCurrentIndex){
104         case 0:
```

```
105          display.setImageResource(R.drawable.contrast1);
106          break;
107         case 1:
108          display.setImageResource(R.drawable.contrast2);
109          break;
110                      ...
111                      ...
112                      ...
113
114         case 15:
115          display.setImageResource(R.drawable.contrast16);
116          break;
117         }
118     }
119     //Method to show results
120     private void contrastScore(){
121
122       //Get the number of correct answers
123       for(int i=0;i<realAnswers.length;i++){
124          if(usrAnswersList.get(i).equals(realAnswers[i]) ){
125             correctAnswers++;
126          }
127       }
128       //if all answers are correct
129       if (realAnswersList.equals(usrAnswersList))
130       {
131          //correctAnswer++;
132          //scoreIshihara = "Passed";
133          //Toast.makeText(IshiharaTest.this, "You " + scoreAmsler + " the
                Test", Toast.LENGTH_SHORT).show();
134          Log.i("TAG", "mpike");
135          Bundle extras = getIntent().getExtras();
136          String username = extras.getString("Username");
137          //Create intent
138          Intent scoreContrast = new Intent(getApplicationContext(),
                ContrastSensitivityScore.class);
139          scoreContrast.putExtra("PassFail","Passed");
140          scoreContrast.putExtra("ColumnToInsert","CONTRAST_SENSITIVITY");
141          scoreContrast.putExtra("Username",username);
```

```
142         scoreContrast.putExtra("CorrectAnswers", correctAnswers);
143         startActivity(scoreContrast);
144         finish();
145 
146       }
147       else
148       {
149         Bundle extras = getIntent().getExtras();
150         String username = extras.getString("Username");
151         //create intent
152         Intent scoreContrast = new Intent(getApplicationContext(),
              ContrastSensitivityScore.class);
153         scoreContrast.putExtra("PassFail","Failed");
154         scoreContrast.putExtra("ColumnToInsert","CONTRAST_SENSITIVITY");
155         scoreContrast.putExtra("Username",username);
156         scoreContrast.putExtra("CorrectAnswers", correctAnswers);
157         startActivity(scoreContrast);
158         finish();
159       }
160 
161 
162     }
163 }
```

The Color Sensitivity Test

The instructions for taking this test are:

1. Keep the device in arm's length and shoulder's height
2. Take the test in natural light
3. Select the correct direction of the letter 'C'

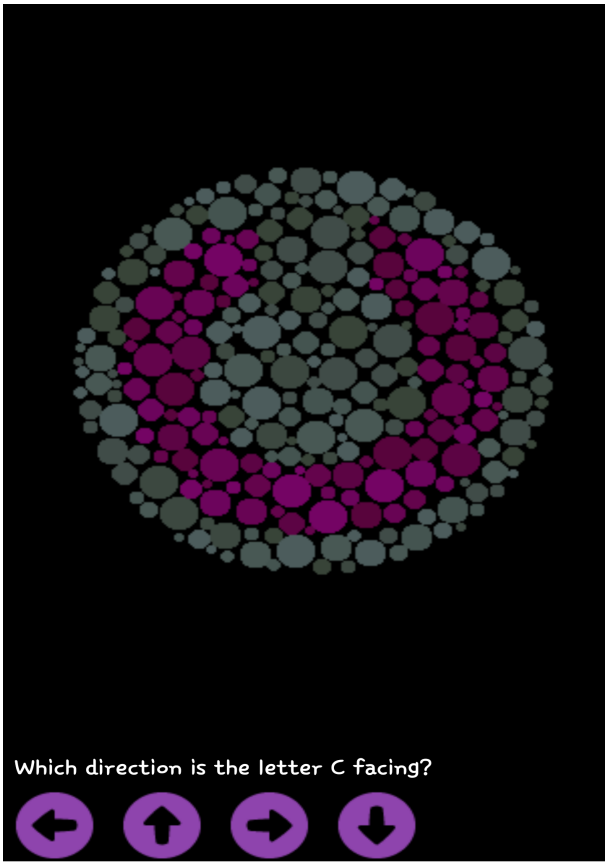


Figure 3.13: The Color Sensitivity Test

The layout of the Color Sensitivity test, as shown below, is structured using an ImageView for the descending in color images, a TextView and a linear layout with four buttons used for the direction of the letter 'C'.

Listing 3.22: The Color Sensitivity layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/display_color"
        android:layout_width="match_parent"
        android:layout_height="0dp"
```

```
12          android:layout_weight="1"    />
13
14
15      <TextView
16          style="?android:listSeparatorTextViewStyle"
17          android:layout_width="match_parent"
18          android:layout_height="wrap_content"
19          android:textColor="#FFFFFF"
20          android:textAllCaps="false"
21          android:text="@string/question_contrast" />
22
23        <LinearLayout
24            android:layout_width="wrap_content"
25           android:layout_height="wrap_content"
26           android:weightSum="1"       >
27
28         <ImageButton
29             android:id="@+id/imageButton"
30             android:layout_width="0dp"
31             android:layout_height="wrap_content"
32             android:layout_weight="0.25"
33             style="?android:attr/borderlessButtonStyle"
34             android:src="@drawable/purple_arrow_left" />
35
36                 ...
37                 ...
38
39         <ImageButton
40             android:id="@+id/imageButton4"
41             android:layout_width="0dp"
42             android:layout_height="wrap_content"
43             android:layout_weight="0.25"
44             style="?android:attr/borderlessButtonStyle"
45             android:src="@drawable/purple_arrow_down" />
46
47        </LinearLayout>
48
49  </LinearLayout>
```

As for the java code implemented, it builds two lists which hold the default answers and the user's answers. In the end it compares the two lists, produces the proper result/score and takes the user to the result/score activity.

Listing 3.23: The ColorSensitivity.java

```
1  public class ColorSensitivity extends Activity {
2
3     ImageView display ;
4     ImageButton imgButton,imgButton2,imgButton3,imgButton4;
5     int mCurrentIndex = 0;
6     //Store the correct answers
7     String realAnswers[] =
          {"up","right","left","right","left","up","left","right","up","right","left","down"};
8     List<String> realAnswersList = new ArrayList<String>();
9     String usrAnswers[];
10    List<String> usrAnswersList = new ArrayList<String>();
11
12    int correctAnswers;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         //full screen mode
18         requestWindowFeature(Window.FEATURE_NO_TITLE);
19         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
20                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
21         //inflate the layout
22         setContentView(R.layout.color_sensitivity);
23
24         //Populate the list with the correct answers
25         for (int index = 0; index < realAnswers.length; index++)
26         {
27             realAnswersList.add(realAnswers[index]);
28         }
29
30         display = (ImageView)findViewById(R.id.display_color);
31         updateImage();
32         //add listener on the button
33         // This imageButton is for the Left
```

```
34          imgButton = (ImageButton)findViewById(R.id.imageButton);
35          imgButton.setOnClickListener(new View.OnClickListener() {
36              @Override
37              public void onClick(View v) {
38                  mCurrentIndex++;
39                  updateImage();
40
41                  usrAnswersList.add("left");
42
43                  if (mCurrentIndex == 12 ){
44                    colorScore();
45                  }
46              }
47          });
48          //add listener on the button
49          // This imageButton is for the UP
50          imgButton2 = (ImageButton)findViewById(R.id.imageButton2);
51          imgButton2.setOnClickListener(new View.OnClickListener() {
52              @Override
53              public void onClick(View v) {
54                 mCurrentIndex++;
55                  updateImage();
56
57                  usrAnswersList.add("up");
58
59                  if (mCurrentIndex == 12 ){
60                    colorScore();
61                  }
62              }
63          });
64          //add listener on the button
65          // This imageButton is for the Right
66          imgButton3 = (ImageButton)findViewById(R.id.imageButton3);
67          imgButton3.setOnClickListener(new View.OnClickListener() {
68              @Override
69              public void onClick(View v) {
70                 mCurrentIndex++;
71                  updateImage();
72
```

```
73                usrAnswersList.add("right");
74
75                if (mCurrentIndex == 12 ){
76                  colorScore();
77                }
78            }
79        });
80        //add listener on the button
81        // This imageButton is for the DOWN
82        imgButton4 = (ImageButton)findViewById(R.id.imageButton4);
83        imgButton4.setOnClickListener(new View.OnClickListener() {
84            @Override
85            public void onClick(View v) {
86               mCurrentIndex++;
87                updateImage();
88
89                usrAnswersList.add("down");
90
91                if (mCurrentIndex == 12 ){
92                  colorScore();
93                }
94            }
95        });
96
97    }
98    private void updateImage() {
99        switch (mCurrentIndex){
100       case 0:
101        display.setImageResource(R.drawable.color1);
102        break;
103       case 1:
104        display.setImageResource(R.drawable.color2);
105        break;
106                ...
107                ...
108                ...
109
110       case 11:
111        display.setImageResource(R.drawable.color12);
```

```
112         break;
113
114        }
115     }
116
117     private void colorScore(){
118
119       //Get the number of correct answers
120       for(int i=0;i<realAnswers.length;i++){
121          if(usrAnswersList.get(i).equals(realAnswers[i]) ){
122             correctAnswers++;
123          }
124       }
125       //If all answers are correct
126       if (realAnswersList.equals(usrAnswersList))
127       {
128          //correctAnswer++;
129          //scoreIshihara = "Passed";
130          //Toast.makeText(IshiharaTest.this, "You " + scoreAmsler + " the
               Test", Toast.LENGTH_SHORT).show();
131
132          Bundle extras = getIntent().getExtras();
133          String username = extras.getString("Username");
134          //create an intent
135          Intent color = new Intent(getApplicationContext(),
               ColorSensitivityScore.class);
136          color.putExtra("PassFail","Passed");
137          color.putExtra("ColumnToInsert","COLOR_SENSITIVITY");
138          color.putExtra("Username",username);
139          color.putExtra("CorrectAnswers", correctAnswers);
140          startActivity(color);
141          finish();
142
143       }
144       else
145       {
146          Bundle extras = getIntent().getExtras();
147          String username = extras.getString("Username");
148          create an intent
```

```
149         Intent color = new Intent(getApplicationContext(),
                ColorSensitivityScore.class);
150         color.putExtra("PassFail","Failed");
151         color.putExtra("ColumnToInsert","COLOR_SENSITIVITY");
152         color.putExtra("Username",username);
153         color.putExtra("CorrectAnswers", correctAnswers);
154         startActivity(color);
155         finish();
156       }
157     }
158 }
```

The Visual Acuity Test

The instructions for taking this test are:

1. Keep the device in arm's length and shoulder's height
2. Take the test in natural light
3. Select the correct direction of the letter 'E'

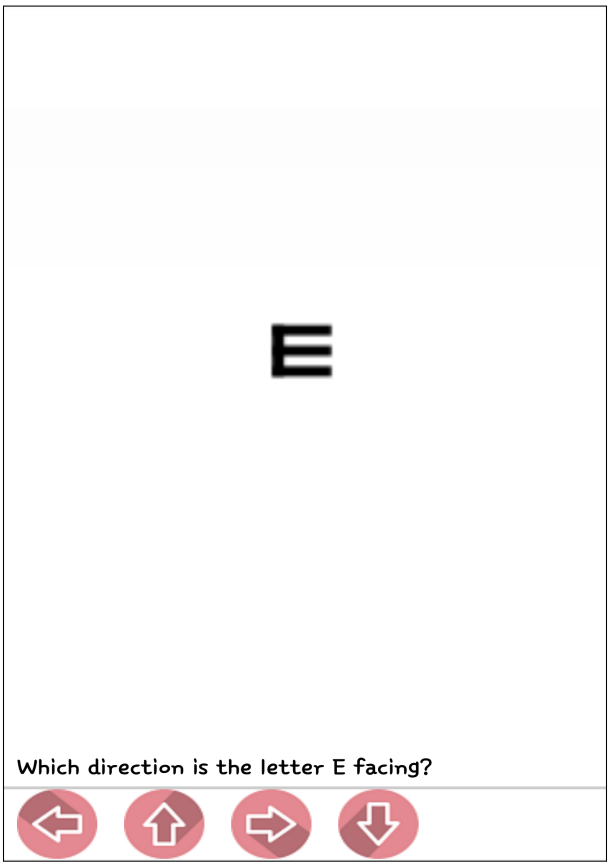


Figure 3.14: The Visual Acuity Test

The layout of the Visual Acuity test, as shown below, is structured using an ImageView for the descending in size images, a TextView and a linear layout with four buttons used for the direction of the letter 'E'.

Listing 3.24: The Visual Acuity Test layout

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/display_visualAcuity"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"   />


    <TextView
        style="?android:listSeparatorTextViewStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textAllCaps="false"
        android:text="@string/question_visualAcuity" />

      <LinearLayout
          android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:weightSum="1"      >

       <ImageButton
           android:id="@+id/imageButton"
           android:layout_width="0dp"
           android:layout_height="wrap_content"
           android:layout_weight="0.25"
           style="?android:attr/borderlessButtonStyle"
           android:src="@drawable/pink_arrow_left" />

                 ...
                 ...
       <ImageButton
           android:id="@+id/imageButton4"
           android:layout_width="0dp"
           android:layout_height="wrap_content"
```

```
            android:layout_weight="0.25"
            style="?android:attr/borderlessButtonStyle"
            android:src="@drawable/pink_arrow_down" />

    </LinearLayout>

</LinearLayout>
```

As far as the implementation goes, it builds two lists which hold the default answers and the user's answers. In the end it compares the two lists, produces the proper result/score and takes the user to the result/score activity.

```
public class VisualAcuity extends Activity {

   ImageView display ;
   ImageButton imgButton,imgButton2,imgButton3,imgButton4;
   int mCurrentIndex = 0;
   //store the correct answers
   String realAnswers[] = {"right","down","left","up","down",
   "left","up","right","down","left","up",
   "right","down","left","up","right","left","up",
   "right","down","right","down","left","up"};
   List<String> realAnswersList = new ArrayList<String>();
   String usrAnswers[];
   List<String> usrAnswersList = new ArrayList<String>();

   int correctAnswers;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //full screen mode
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                             WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.visual_acuity_layout);

        //Populate the list with the correct answers
```

```
28          for (int index = 0; index < realAnswers.length; index++)
29          {
30              realAnswersList.add(realAnswers[index]);
31          }
32
33          display = (ImageView)findViewById(R.id.display_visualAcuity);
34          updateImage();
35          //add listener on the button
36          // This imageButton is for the Left
37          imgButton = (ImageButton)findViewById(R.id.imageButton);
38          imgButton.setOnClickListener(new View.OnClickListener() {
39              @Override
40              public void onClick(View v) {
41                  mCurrentIndex++;
42                  updateImage();
43
44                  usrAnswersList.add("left");
45
46                  if (mCurrentIndex == 24 ){
47                    visualAcuityScore();
48                  }
49              }
50          });
51          //add listener on the button
52          // This imageButton is for the UP
53          imgButton2 = (ImageButton)findViewById(R.id.imageButton2);
54          imgButton2.setOnClickListener(new View.OnClickListener() {
55              @Override
56              public void onClick(View v) {
57                 mCurrentIndex++;
58                  updateImage();
59
60                  usrAnswersList.add("up");
61
62                  if (mCurrentIndex == 24 ){
63                    visualAcuityScore();
64                  }
65              }
66          });
```

```
67          //add listener on the button
68          // This imageButton is for the Right
69          imgButton3 = (ImageButton)findViewById(R.id.imageButton3);
70          imgButton3.setOnClickListener(new View.OnClickListener() {
71              @Override
72              public void onClick(View v) {
73                 mCurrentIndex++;
74                  updateImage();
75
76                  usrAnswersList.add("right");
77
78                  if (mCurrentIndex == 24 ){
79                    visualAcuityScore();
80                  }
81              }
82          });
83          //add listener on the button
84          // This imageButton is for the DOWN
85          imgButton4 = (ImageButton)findViewById(R.id.imageButton4);
86          imgButton4.setOnClickListener(new View.OnClickListener() {
87              @Override
88              public void onClick(View v) {
89                 mCurrentIndex++;
90                  updateImage();
91
92                  usrAnswersList.add("down");
93
94                  if (mCurrentIndex == 24 ){
95                    visualAcuityScore();
96                  }
97              }
98          });
99
100
101     }
102
103     private void updateImage() {
104         switch (mCurrentIndex){
105         case 0:
```

```
106          display.setImageResource(R.drawable.vision1);
107          break;
108         case 1:
109          display.setImageResource(R.drawable.vision2);
110          break;
111                      ...
112                      ...
113                      ...
114
115         case 23:
116          display.setImageResource(R.drawable.vision24);
117          break;
118
119         }
120     }
121     //method to show results
122     private void visualAcuityScore(){
123
124       //Get the number of correct answers
125       for(int i=0;i<realAnswers.length;i++){
126          if(usrAnswersList.get(i).equals(realAnswers[i]) ){
127             correctAnswers++;
128          }
129       }
130       //if all answers are correct
131       if (realAnswersList.equals(usrAnswersList))
132       {
133
134          Bundle extras = getIntent().getExtras();
135          String username = extras.getString("Username");
136          //create an intent
137          Intent scoreAmsler = new Intent(getApplicationContext(),
               VisualAcuityScore.class);
138          scoreAmsler.putExtra("PassFail","Passed");
139          scoreAmsler.putExtra("ColumnToInsert","VISUAL_ACUITY");
140          scoreAmsler.putExtra("Username",username);
141          scoreAmsler.putExtra("CorrectAnswers", correctAnswers);
142          startActivity(scoreAmsler);
143          finish();
```

```
144
145         }
146         else
147         {
148             Bundle extras = getIntent().getExtras();
149             String username = extras.getString("Username");
150             //create an intent
151             Intent scoreAmsler = new Intent(getApplicationContext(),
                    VisualAcuityScore.class);
152             scoreAmsler.putExtra("PassFail","Failed");
153             scoreAmsler.putExtra("ColumnToInsert","VISUAL_ACUITY");
154             scoreAmsler.putExtra("Username",username);
155             scoreAmsler.putExtra("CorrectAnswers", correctAnswers);
156             startActivity(scoreAmsler);
157             finish();
158         }
159     }
160 }
```

The Red Desaturation Test

This test can indicate if someone suffers from ON (Optic Neuritis). As explained in the second chapter, one of optic neuritis's symptoms is the desaturation of the color red. Knowing this, if someone sees a red image slightly washed out (more orange than red) then there is a possibility that this person suffers from ON.

The tests instructions are simple:

1. Hold the device in arm's length and shoulder's height
2. Take the test in natural light
3. Focus on the first and second image and try to identify whether the given red areas are rich or desaturated and washed out.

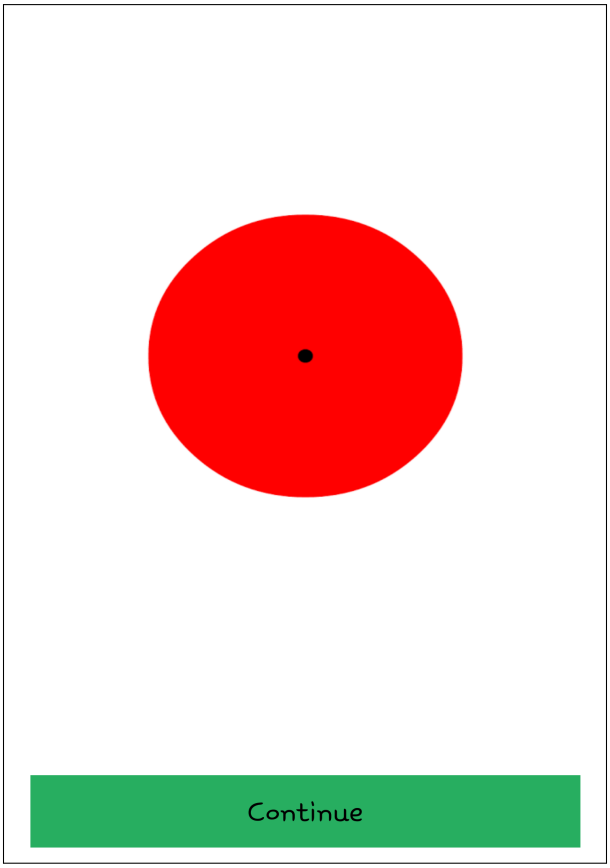


Figure 3.15: The Red Desaturation Test

The layout of this test has just two widgets, an ImageView and a button as shown in the xml code below.

Listing 3.25: The Red Desaturation layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="#FFFFFF"
    tools:context="com.example.thesis.RedDesaturation" >

    <Button
```

```
13          android:id="@+id/button1"
14          android:layout_width="fill_parent"
15          android:layout_height="wrap_content"
16                ...
17          android:background="@color/login_green"
18          android:text="@string/Continue" />
19
20      <ImageView
21          android:id="@+id/imageView1"
22          android:layout_width="wrap_content"
23          android:layout_height="wrap_content"
24          android:layout_above="@+id/button1"
25          android:layout_centerHorizontal="true"
26          android:layout_marginBottom="38dp"
27
28           />
29
30  </RelativeLayout>
```

The java code for this test inflates the layout and shows the ImageViews to the user. It then asks the user if the images where the same rich red color or not. Depending on the answer it directs the user to the result/score layout.

Listing 3.26: The RedDesaturation.java

```
1  public class RedDesaturation extends Activity {
2
3     ImageView img;
4     Button next;
5     int mCurrentIndex = 0;
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9        super.onCreate(savedInstanceState);
10
11       // Make activity full screen
12         requestWindowFeature(Window.FEATURE_NO_TITLE);
13         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
14                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
15
```

```
16        setContentView(R.layout.red_desaturation_layout);
17
18        img = (ImageView)findViewById(R.id.imageView1);
19        updateImage();
20        //add listener on the Next button
21        next = (Button) findViewById(R.id.button1);
22        next.setOnClickListener(new View.OnClickListener() {
23
24           @Override
25           public void onClick(View v) {
26              mCurrentIndex++;
27              updateImage();
28              if(mCurrentIndex == 2){
29                    showResult();
30                  }
31
32
33           }
34        });
35
36
37     }
38     //method to change the image
39     private void updateImage() {
40          switch (mCurrentIndex){
41          case 0:
42           img.setImageResource(R.drawable.red_desaturation1);
43           break;
44          case 1:
45           img.setImageResource(R.drawable.red_desaturation2);
46           break;
47
48          }
49
50      }
51      //Method to show results
52  private void showResult() {
53
54        Bundle extras = getIntent().getExtras();
```

```
55        //"PassFail" contains PASS or FAIL
56
57        final String username = extras.getString("Username");
58
59        Intent gotoRedDesaturation2 = new Intent(getApplicationContext(),
              RedDesaturation2.class);
60        gotoRedDesaturation2.putExtra("PassFail","Passed");
61        //gotoRedDesaturation2.putExtra("ColumnToInsert","AMSLER");
62        gotoRedDesaturation2.putExtra("Username",username);
63        startActivity(gotoRedDesaturation2);
64        finish();
65
66    }
67 }
```

3.2.5 Scheduled Notification

This activity allows the user to schedule a notification and is based on the AlarmManager class. Alarms provide a way to perform time-based operations outside the lifetime of the application. For example, an alarm can be used to initiate long-running operation, such as starting a service once a day to remind a user to take some medicine. Alarms have these characteristics:

- They fire Intents at set times and/or intervals.
- They are used in conjunction with broadcast receivers to start services and perform other operations.
- They operate outside of the application, so they can be used to trigger events or actions even when the app is not running, and even if the device itself is asleep.
- They help minimize the application's resource requirements. For example, schedule operations without relying on timers or continuously running background services.[8]

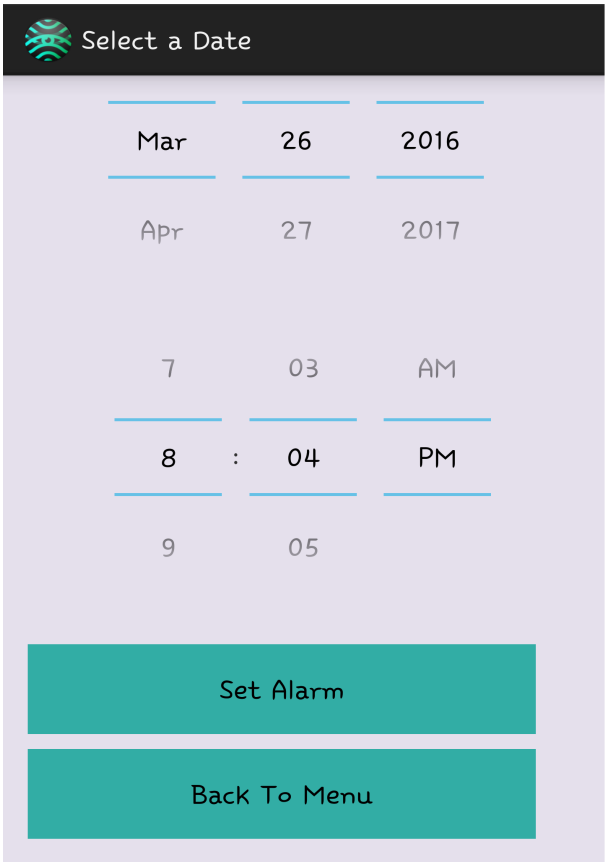


Figure 3.16: Scheduling a Notification

The layout of the screen contains five view widgets. A DatePicker for the user to pick a future date to run Oculus, a TimePicker for setting a specific time, a TextView to show the confirmation of the alarm and two buttons.

Listing 3.27: Schedule Notification Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="#e5e0ec"
    android:orientation="vertical"
    tools:context=".MainActivity" >
```

```
12
13      <ScrollView
14          android:layout_width="match_parent"
15          android:layout_height="match_parent">
16          <LinearLayout
17              android:layout_width="match_parent"
18              android:layout_height="wrap_content"
19              android:orientation="vertical">
20
21              <DatePicker
22                  android:id="@+id/pickerdate"
23                  android:calendarViewShown="false"
24                  android:layout_width="wrap_content"
25                  android:layout_height="wrap_content"
26                  android:descendantFocusability="blocksDescendants"
27                  android:layout_gravity="center" />
28
29              <TimePicker
30                  android:id="@+id/pickertime"
31                  android:layout_width="wrap_content"
32                  android:layout_height="wrap_content"
33                  android:descendantFocusability="blocksDescendants"
34                  android:layout_gravity="center"/>
35              <TextView
36                  android:id="@+id/info"
37                  android:layout_width="match_parent"
38                  android:layout_height="wrap_content"/>
39
40              <Button
41                  android:id="@+id/setalarm"
42                  android:layout_width="303dp"
43                          ...
44                  android:background="@drawable/buttonshape_gotonotif"
45                  android:text="@string/setAlarm" />
46              <Button
47                  android:id="@+id/goback"
48                  android:layout_width="303dp"
49                  android:layout_height="wrap_content"
50                          ...
```

```
51                android:background="@drawable/buttonshape_gotonotif"
52                android:text="@string/backToMenu" />
53
54        </LinearLayout>
55    </ScrollView>
56
57 </LinearLayout>
```

The code implemented to inflate the layout is shown below. It populates the widgets, reads the input from the date and time pickers, creates the alarm using this info and finally creates a pending intent that runs in the background until the selected time and date.

Listing 3.28: The ScheduledNotification.java

```
1  public class ScheduledNotification extends Activity{
2
3     DatePicker pickerDate;
4     TimePicker pickerTime;
5     Button buttonSetAlarm,goBack;
6     TextView info;
7
8     final static int RQS_1 = 1;
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12       super.onCreate(savedInstanceState);
13       setContentView(R.layout.scheduled_notification);
14       //set activity title
15       setTitle("Select a Date");
16
17       info = (TextView)findViewById(R.id.info);
18       pickerDate = (DatePicker)findViewById(R.id.pickerdate);
19       pickerTime = (TimePicker)findViewById(R.id.pickertime);
20       //get calendar instance
21       Calendar now = Calendar.getInstance();
22
23       pickerDate.init(
24             now.get(Calendar.YEAR),
```

```
25            now.get(Calendar.MONTH),
26            now.get(Calendar.DAY_OF_MONTH),
27            null);
28
29        pickerTime.setCurrentHour(now.get(Calendar.HOUR_OF_DAY));
30        pickerTime.setCurrentMinute(now.get(Calendar.MINUTE));
31        //add listener on the GoBack button
32        goBack = (Button)findViewById(R.id.goback);
33        goBack.setOnClickListener(new OnClickListener() {
34
35           @Override
36           public void onClick(View v) {
37              Bundle extras = getIntent().getExtras();
38              String username = extras.getString("Username");
39              //create an intent to take the user back
40              Intent goToDrawer = new Intent(getBaseContext(),Drawer.class);
41              goToDrawer.putExtra("Username", username);
42              startActivity(goToDrawer);
43
44           }
45        });
46        //add listener on the SetAlarm button
47        buttonSetAlarm = (Button)findViewById(R.id.setalarm);
48        buttonSetAlarm.setOnClickListener(new OnClickListener(){
49
50           @Override
51           public void onClick(View arg0) {
52              Calendar current = Calendar.getInstance();
53
54              Calendar cal = Calendar.getInstance();
55              cal.set(pickerDate.getYear(),
56                    pickerDate.getMonth(),
57                    pickerDate.getDayOfMonth(),
58                    pickerTime.getCurrentHour(),
59                    pickerTime.getCurrentMinute(),
60                    00);
61              //Check if the date selected has already passed
62              if(cal.compareTo(current) <= 0){
63                 //The set Date/Time already passed
```

```
64                 Toast.makeText(getApplicationContext(),
65                     "Invalid Date/Time",
66                     Toast.LENGTH_LONG).show();
67             }else{
68             //else call the setAlarm method
69                setAlarm(cal);
70             }
71
72         }});
73     }
74     //method to set the alarm to given date and time
75     private void setAlarm(Calendar targetCal){
76
77        info.setText("\n\n***\n"
78              + "Alarm is set @ " + targetCal.getTime() + "\n"
79              + "***\n");
80        //create a pending intent
81        Intent intent = new Intent(getBaseContext(), AlarmReceiver.class);
82        PendingIntent pendingIntent =
              PendingIntent.getBroadcast(getBaseContext(), RQS_1, intent, 0);
83        AlarmManager alarmManager =
              (AlarmManager)getSystemService(Context.ALARM_SERVICE);
84        alarmManager.set(AlarmManager.RTC_WAKEUP, targetCal.getTimeInMillis(),
              pendingIntent);
85     }
86
87 }
```

3.2.6 Previous Test Results

The activity of the previous test results provides the users with a progress of their results. It allows them to monitor their conditions or make them aware of conditions that have not been noticed in the past. After taking a test, the results are recorded in the database and with the help of the getAllScores() method in the DatabaseAdapter.java the user can choose to see the results of any test taken previously. As we can see the java code of the getAllScores method identifies which test to retrieve the results from, creates a sqlite query, retrieves the results and finally inserts them into an array so they can be

displayed.

Listing 3.29: The getAllScores method

```
1
2  public void getAllScores(String username, String test) {
3
4       if(test.equals("ishihara")){
5          //create a sqlite query
6          String selectQuery = "SELECT * FROM " + dbHelper.TABLE_ISHIHARA + "
               WHERE " +dbHelper.NAME+ " = ?";
7          String[] whereArgs = {username};
8          //open DB and get the data
9            SQLiteDatabase db = helper.getWritableDatabase();
10           Cursor cursor = db.rawQuery(selectQuery, whereArgs);
11           //while getting data add them to a string
12           while (cursor.moveToNext() ){
13
14              // Select from the db
15                 String ishiharaScore = cursor.getString(1);
16                 String showIshiharaScore = "User: "+username+"\n"+ "Score: "
                       +ishiharaScore+ "%";
17                //add that string to an array
                     ScoreGridView.ArrayofScores.add(showIshiharaScore);
18
19               } ;
20
21      }
22      else if(test.equals("neitz")){
23
24         String selectQuery = "SELECT * FROM " + dbHelper.TABLE_NEITZ + " WHERE
               " +dbHelper.NAME+ " = ?";
25         String[] whereArgs = {username};
26
27           SQLiteDatabase db = helper.getWritableDatabase();
28           Cursor cursor = db.rawQuery(selectQuery, whereArgs);
29
30           while (cursor.moveToNext() ){
31
32              // Select from the db
```

```
33                String neitzScore = cursor.getString(1);
34                String showNeitzScore = "User: "+username+"\n"+ "Score: " +
                      neitzScore+ "%";
35                ScoreGridView.ArrayofScores.add(showNeitzScore);
36
37              } ;
38
39
40      }
41
42                  ...
43                  ...
44                  ...
45
46
47      else if(test.equals("red_desaturation")){
48
49         String selectQuery = "SELECT * FROM " +
               dbHelper.TABLE_RED_DESATURATION + " WHERE " +dbHelper.NAME+ " = ?";
50         String[] whereArgs = {username};
51
52           SQLiteDatabase db = helper.getWritableDatabase();
53           Cursor cursor = db.rawQuery(selectQuery, whereArgs);
54
55           while (cursor.moveToNext() ){
56
57              // Select from the db
58                String redDeScore = cursor.getString(1);
59                String showRedDeScore = "User: "+username+"\n"+ "Result: " +
                      redDeScore;
60                ScoreGridView.ArrayofScores.add(showRedDeScore);
61
62              } ;
63      }
64
65   }
```

Viewing these results requires two activities, one for choosing the test of which the user requests the results and secondly, an activity to display them in a list.

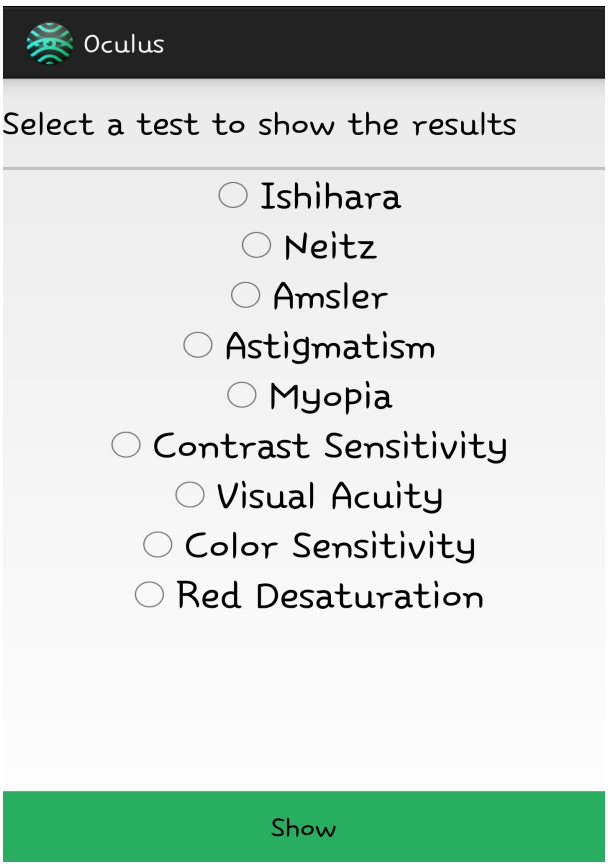


Figure 3.17: Choosing a test to show its previous results

The layout of this activity, as shown below is implemented by using a TextView, a RadioGroup including nine radioButtons, each representing a test and a button which takes the user to the results of the selected test.

Listing 3.30: The layout for choosing a test to show results

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
```

```
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11              ...
12        android:textAppearance="?android:attr/textAppearanceLarge" />
13
14     <View
15        android:id="@+id/score_divider"
16        android:layout_width="fill_parent"
17        android:layout_height="2dp"
18        android:layout_marginTop="15dp"
19        android:layout_below="@+id/textView1"
20        android:background="#c0c0c0" />
21
22    <Button
23        android:id="@+id/button1"
24
25        style="?android:attr/borderlessButtonStyle"
26        android:layout_width="match_parent"
27        android:layout_height="wrap_content"
28                  ...
29        android:text="@string/ShowMe" />
30
31    <ScrollView
32        android:layout_width="fill_parent"
33        android:layout_height="wrap_content"
34        android:layout_above="@+id/button1"
35        android:layout_alignParentLeft="true"
36        android:layout_below="@+id/score_divider" >
37
38        <RadioGroup
39            android:id="@+id/radioGroup"
40            android:layout_width="match_parent"
41            android:layout_height="144dp" >
42
43            <RadioButton
44                android:id="@+id/radioButton1"
45                android:layout_gravity="center_horizontal"
46                android:checked="false"
47                android:text="@string/Ishihara"
```

```
48              android:textSize="25sp" />
49
50          <RadioButton
51              android:id="@+id/radioButton2"
52              android:layout_gravity="center_horizontal"
53              android:checked="false"
54              android:text="@string/Neitz"
55              android:textSize="25sp" />
56
57                    ...
58                    ...
59                    ...
60
61
62          <RadioButton
63              android:id="@+id/radioButton9"
64              android:layout_gravity="center_horizontal"
65              android:checked="false"
66              android:text="@string/RedDesaturation"
67              android:textSize="25sp" />
68      </RadioGroup>
69  </ScrollView>
70
71 </RelativeLayout>
```

The implementation of the activity can be found below. The code takes the input (selected RadioButton) and creates an intent that carries information to the next activity which lets the application identify which test was selected so it can call the getAllScores() method described before and show the results.

Listing 3.31: The GoToScores.java

```
1  public class GoToScores extends Activity {
2     //declare the radiobuttons
3     RadioButton rdbtn1,rdbtn2,rdbtn3,rdbtn4,rdbtn5,rdbtn6,rdbtn7,rdbtn8,rdbtn9;
4
5     @Override
6     public void onCreate(Bundle savedInstanceState) {
7        super.onCreate(savedInstanceState);
```

```
8       setContentView(R.layout.go_to_scores);
9
10      RadioGroup radioGroup = (RadioGroup) findViewById(R.id.radioGroup);
11
12      rdbtn1 = (RadioButton)findViewById(R.id.radioButton1);
13      rdbtn2 = (RadioButton)findViewById(R.id.radioButton2);
14      rdbtn3 = (RadioButton)findViewById(R.id.radioButton3);
15      rdbtn4 = (RadioButton)findViewById(R.id.radioButton4);
16      rdbtn5 = (RadioButton)findViewById(R.id.radioButton5);
17      rdbtn6 = (RadioButton)findViewById(R.id.radioButton6);
18      rdbtn7 = (RadioButton)findViewById(R.id.radioButton7);
19      rdbtn8 = (RadioButton)findViewById(R.id.radioButton8);
20      rdbtn9 = (RadioButton)findViewById(R.id.radioButton9);
21      //add listener on the button
22      Button btnShowScores = (Button)findViewById(R.id.button1);
23      btnShowScores.setOnClickListener(new View.OnClickListener() {
24
25
26         @Override
27         public void onClick(View v) {
28            Bundle extras = getIntent().getExtras();
29              String username = extras.getString("Username");
30
31              //check which radiobutton was selected
32            if(rdbtn1.isChecked()) {
33               Intent ishiharaScores = new Intent(GoToScores.this,
                     ScoreGridView.class);
34               ishiharaScores.putExtra("Username", username);
35               ishiharaScores.putExtra("Test", "ishihara");
36               startActivity(ishiharaScores);
37               finish();
38            }
39            else if(rdbtn2.isChecked()){
40               Intent neitzScores = new Intent(GoToScores.this,
                     ScoreGridView.class);
41               neitzScores.putExtra("Username", username);
42               neitzScores.putExtra("Test", "neitz");
43               startActivity(neitzScores);
44               finish();
```

```
45                          ...
46                          ...
47                          ...
48
49              }else if(rdbtn9.isChecked()){
50                 Intent redDesScores = new Intent(GoToScores.this,
                       ScoreGridView.class);
51                 redDesScores.putExtra("Username", username);
52                 redDesScores.putExtra("Test", "red_desaturation");
53                 startActivity(redDesScores);
54                 finish();
55              }
56           }
57
58        });
59
60     }
```

This takes the user into the next activity which is the display of the results.

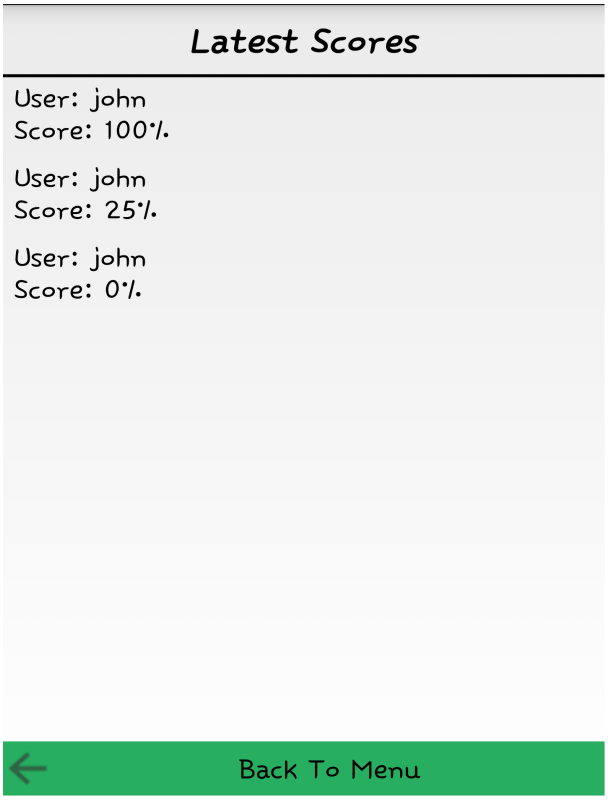


Figure 3.18: The display of previous results

The layout of the results activity is comprised of a TextView, a GridView and a Button. When the user selects a test to show the results, the getAllScores() method reads the database and displays the results that were retrieved in a GridView (in this case a GridView with one column).

Listing 3.32: The score GridView layout

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:gravity="center|center_vertical"
6     android:orientation="vertical" >
7
8     <View
```

```
9         android:id="@+id/divider"
10        android:layout_below="@+id/textView1"
11        android:layout_width="match_parent"
12        android:layout_height="2dp"
13        android:background="#000000" />
14
15    <GridView
16        android:id="@+id/gridView1"
17        android:layout_width="match_parent"
18        android:layout_height="253dp"
19              ...
20        android:numColumns="1"
21        android:verticalSpacing="5dp" >
22
23    </GridView>
24
25    <TextView
26        android:id="@+id/textView1"
27        android:layout_width="match_parent"
28        android:layout_height="46dp"
29              ...
30        android:textAppearance="?android:attr/textAppearanceLarge"
31        android:textStyle="bold|italic" />
32
33
34    <Button
35        android:id="@+id/button1"
36        android:layout_width="match_parent"
37        android:layout_height="36dp"
38                ...
39        android:gravity="center|center_horizontal"
40        android:text="@string/backToMenu" />
41
42 </RelativeLayout>
```

The ScoreGridView.java is responsible for calling the getAllScores method and adapting the data from the database to the GridView using an ArrayAdapter.

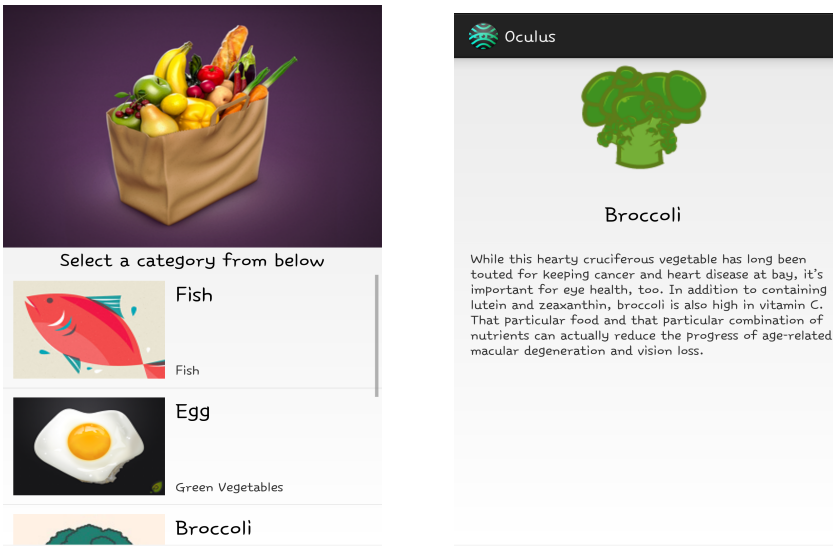
Listing 3.33: The ScoreGridView.java

```
1
2  public class ScoreGridView extends Activity {
3
4     private GridView gridView;
5
6      public static ArrayList<String> ArrayofScores = new ArrayList<String>();
7      Button btn,export;
8      DatabaseAdapter myDbHelper;
9
10     /** Called when the activity is first created. */
11     @Override
12     public void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.score_gridview);
15
16         setTitle("Previous Results");
17         //instance of our DB
18         myDbHelper = new DatabaseAdapter(this);
19
20         Bundle extras = getIntent().getExtras();
21
22         String username = extras.getString("Username");
23         String test = extras.getString("Test");
24
25
26
27         // Reading all contacts
28         //List<Contact> contacts = db.getAllContacts();
29
30         // Clear the list so we have an empty one every time
31         // this activity is created
32         ArrayofScores.clear();
33
34         // Get the scores from the DB
35         myDbHelper.getAllScores(username,test);
36
37         gridView = (GridView) findViewById(R.id.gridView1);
```

```
38
39          ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
40                  android.R.layout.simple_list_item_1, ArrayofScores);
41        //set the adapter
42          gridView.setAdapter(adapter);
43          //add listener on the button
44          btn = (Button)findViewById(R.id.button1);
45          btn.setOnClickListener(new View.OnClickListener() {
46
47           @Override
48           public void onClick(View v) {
49              //create an intent
50              Intent backtomenu = new Intent(getApplicationContext(),
                    Drawer.class);
51                 Bundle extras = getIntent().getExtras();
52                 String username = extras.getString("Username");
53                 backtomenu.putExtra("Username", username);
54                 startActivity(backtomenu);
55                 finish();
56           }
57        });
58      }
59
60
61
62  }
```

3.2.7 The Nutrition Guide

The application has its own nutritional section providing the users with some helpful information regarding their health and more specifically their eye health.



(a) List of Nutrition Tips (b) A selected Nutrition Tip

Figure 3.19: The list of nutrition tips (a) and a selected one (b)

The (a) figure above has a layout that can be described by the xml code below which is constructed by using an ImageView, a TextView and a ListView. In addition, the (b) figure consists of an ImageView and a textView.

Listing 3.34: The Nutrition Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:id="@+id/image_nutr"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:adjustViewBounds="true"
        android:src="@drawable/all_fruits" />
```

```
13
14      <TextView
15          android:id="@+id/textView1"
16          android:layout_width="match_parent"
17          android:layout_height="wrap_content"
18                    ...
19          android:textSize="20sp" />
20
21      <ListView
22          android:id="@+id/listNutrition"
23          android:layout_width="match_parent"
24          android:layout_height="wrap_content"
25          android:layout_alignParentBottom="true"
26          android:layout_alignParentLeft="true"
27          android:layout_below="@+id/textView1" >
28
29      </ListView>
30
31  </RelativeLayout>
```

The nutrition activity (Nutrition.java) implements a ListView using an ArrayAdapter which takes a row of a simple list and then adapts it to every item on the ListView changing the images and information of every item. Every item of the list, when clicked on, opens a different activity that displays the nutritional information of the selected item as shown below.

Listing 3.35: The Nutrition.java

```
1  public class Nutrition extends Activity {
2
3     ListView list;
4     ImageView image_nutr ;
5     ImageButton mCarrot;
6     ImageButton mOrange;
7     String [] TitlesNutr;
8     String [] DescriptionsNutr;
9     int[] imagesNutr={R.drawable.fish, R.drawable.egg2, R.drawable.broccoli,
          R.drawable.carrot, R.drawable.orange};
10
```

```
11      @Override
12      protected void onCreate(Bundle savedInstanceState) {
13          super.onCreate(savedInstanceState);
14          //full screen mode
15          requestWindowFeature(Window.FEATURE_NO_TITLE);
16          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
17                               WindowManager.LayoutParams.FLAG_FULLSCREEN);
18
19          setContentView(R.layout.nutrition_layout);
20          //set the title of the activity
21          setTitle("Nutrition");
22
23          image_nutr = (ImageView)findViewById(R.id.image_nutr);
24
25          Resources res = getResources();
26          TitlesNutr =res.getStringArray(R.array.Categories);
27          DescriptionsNutr =res.getStringArray(R.array.DescriptionsNutr);
28
29          list=(ListView)findViewById(R.id.listNutrition);
30          MyAdapter adapter = new MyAdapter(this, TitlesNutr, imagesNutr,
                DescriptionsNutr);
31          list.setAdapter(adapter);
32          //add listener on the list
33          list.setOnItemClickListener( new OnItemClickListener(){
34           @Override
35           public void onItemClick(AdapterView<?> parent, View view, int
                 position, long id) {
36
37              switch(position){
38              case 0:
39              //create an intent
40                 Intent fish = new Intent(view.getContext(), Fish.class);
41                 startActivity(fish);
42                 break;
43                       ...
44                       ...
45                       ...
46
47              case 4:
```

```
48            //create an intent
49               Intent orange = new Intent(view.getContext(), Orange.class);
50               startActivity(orange);
51               break;
52
53            }
54         }
55        });
56        // Create an array adapter as used before in the app
57        class MyAdapter extends ArrayAdapter<String>
58        {
59         Context context;
60         int[] images;
61         String[] titleArray;
62         String[] descriptionArray;
63         MyAdapter(Context c, String[] Titles, int imgs[], String[] desc)
64         {
65            super(c, R.layout.single_row_nutrition, R.id.textView1, Titles);
66            this.context=c;
67            this.images=imgs;
68            this.titleArray=Titles;
69            this.descriptionArray=desc;
70         }
71
72         @Override
73         public View getView(int position, View convertView, ViewGroup parent){
74            LayoutInflater inflater=(LayoutInflater)context.getSystemService
75            (Context.LAYOUT_INFLATER_SERVICE);
76            View row=inflater.inflate(R.layout.single_row_nutrition, parent,
                 false);
77
78            ImageView myImage=(ImageView)row.findViewById(R.id.imageView1);
79            TextView myTitle=(TextView) row.findViewById(R.id.textView1);
80            TextView myDescription=(TextView) row.findViewById(R.id.textView2);
81         myImage.setImageResource(images[position]);
82            myTitle.setText(titleArray[position]);
83       myDescription.setText(descriptionArray[position]);
84
85            return row;
```

```
86              }
87            }
88          }
89  }
```

4 | Conclusions and Future Work

Concluding, the functionality of the app was successful, which was verified from a number of users that tested the application. This is justified by the fact that users with specific eye deficiencies failed or presented low scores in the respective tests.

The development of an application is a never ending process. There is always some space for improvement and evolution. In this case, as mentioned before, the application is a native android application. So it is logical to adapt the app in order to be used in other operating systems (OS) such as iOS and windows. A simple way to achieve a cross-platform application is to create a hybrid version that will be developed once, but used in every OS.

The targeted version of the app (found in the manifest.xml) is Android 6.0 but it is known that there will be multiple software updates from now on. Every new software version comes with some new key features that make new apps more appealing to the user and the developers as well. So using the new features to develop a better version of this application will be the key to making people use the app even more.

Finally, a fundamental change in the applications core will be the conversion of the local database into a online one. This will help the users to easily back up their medical progress or even share them with a medical professional for a detailed diagnosis. Having an online database will free the applications resources, the reserved memory will be a cloud storage.

Bibliography

[1] How the human eye works. URL http://www.nkcf.org/how-the-human-eye-works/. 4

[2] The cambridge colour test. URL http://www.crsltd.com/tools-for-vision-science/measuring-visual-functions/cambridge-colour-test/. 10

[3] A. O. Association. Hyperopia, . URL http://www.aoa.org/patients-and-public/eye-and-vision-problems/glossary-of-eye-and-vision-conditions/hyperopia?sso=y. 14

[4] A. O. Association. Visual acuity, . URL http://www.aoa.org/patients-and-public/eye-and-vision-problems/glossary-of-eye-and-vision-conditions/visual-acuity?sso=y. 17

[5] A. O. Association. Astigmatism, . URL http://www.aoa.org/patients-and-public/eye-and-vision-problems/glossary-of-eye-and-vision-conditions/astigmatism?sso=y. 12

[6] A. O. Association. Myopia, . URL http://www.aoa.org/patients-and-public/eye-and-vision-problems/glossary-of-eye-and-vision-conditions/myopia?sso=y. 13

[7] S. B. D. L. J. H. A. Birr. Comparison of the neitz test of color vision to the ishihara color vision tests and the anomaloscopic classification. 2004. 10

[8] Google. Alarms. URL http://developer.android.com/training/scheduling/alarms.html. 103

[9] G. Heiting. Contrast sensitivity testing, . URL http://www.allaboutvision.com/eye-exam/contrast-sensitivity.htm. 16

[10] G. Heiting. Refractive errors and refraction: How the eye sees, . URL http://www.allaboutvision.com/eye-exam/refraction.htm. 5, 6

[11] E. Kindel. Ishihara. URL http://www.eyemagazine.com/feature/article/ishihara. 8

[12] J. Neitz. The neitz test. URL http://www.neitzvision.com/content/neitztest.html. 10

[13] Optician. The red desaturation confrontation disc. URL http://www.opticianonline.net/the-red-desaturation-confrontation-disc/. 18

[14] G. A. Sarcone. Color blindness or color vision deficiency. URL http://www.archimedes-lab.org/colorblindnesstest.html. 6

[15] Wikipedia. Macular degeneration, . URL https://en.wikipedia.org/wiki/Macular/_degeneration. 15

[16] Wikipedia. Visual acuity, . URL https://en.wikipedia.org/wiki/Visual_acuity. 17

[17] Wikipedia. Optic neuritis, . URL https://en.wikipedia.org/wiki/Optic_neuritis. 17