

**TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING**



THESIS

**UTILIZATION OF SOCIAL MEDIA IN TRANSMEDIA STORYTELLING OF
HISTORICAL EVENTS AND USER PERSONALIZATION**

MARKODIMITRAKIS IOANNIS

COMMITTEE:

Professor Antonios Deligiannakis, ECE TUC (Supervisor)

Professor Mania Aikaterini, ECE TUC

Professor Petrakis G.M. Euripides, ECE TUC

CHANIA 2022

ABSTRACT

Η διαμεσική αφήγηση είναι ένας ολοένα και πιο διαδεδομένος όρος ειδικά στην εποχή της ψηφιακής επικοινωνίας που διανύουμε και ορίζεται ως η διαδικασία κατά την οποία μια ιστορία εκτυλίσσεται μέσω διαφορετικών μέσων και πλατφορμών επικοινωνίας. Μεμονωμένα κομμάτια πληροφορίας με ξεχωριστή συνεισφορά το καθένα, ομαδοποιούνται και συνεργάζονται με σκοπό την παροχή μια εξατομικευμένης εμπειρίας στο “ξεδίπλωμα” της ιστορίας από τον χρήστη.

Σκοπός της παρούσας διπλωματικής είναι ο σχεδιασμός και η υλοποίηση ενός πλαισίου τεχνολογιών για την προσωποποιημένη διαμεσική αφήγηση ιστοριών και την αξιοποίηση κοινωνικών δικτύων, το οποίο θα εφαρμοστεί στην περιγραφόμενη διαδικτυακή πλατφόρμα διαμεσικών αφηγήσεων με τίτλο STSP, με στόχο την παρουσίαση ιστορικών γεγονότων στον ελλαδικό χώρο και την προώθηση της τοπικής επιχειρηματικότητας. Αναλυτικότερα η πλατφόρμα επιτρέπει την αναπαράσταση διαφόρων ιστοριών και την συλλογή μετρικών στοιχείων από τα κοινωνικά δίκτυα (Facebook) που αφορούν την αλληλεπίδραση των χρηστών με αυτές. Επιπλέον δίνεται η δυνατότητα στους χρήστες να σχολιάσουν αλλά και να αξιολογήσουν με ποικίλους τρόπους τις ιστορίες και να βρουν στην αρχική τους σελίδα αυτές που παρουσιάζουν ομοιότητα με εκείνες που έχουν αλληλοεπιδράσει ήδη. Οι προτάσεις ιστοριών δημιουργούνται με έναν αλγόριθμο *item-based collaborative filtering*, δηλαδή με βάση την ομοιότητα μεταξύ των βαθμολογημένων και μη, ιστοριών του χρήστη.

Το πλαίσιο εργασίας αποτελείται από διάφορα δομικά στοιχεία με διακριτούς ρόλους που επιτελούν συγκεκριμένες λειτουργίες, με ξεχωριστή συνεισφορά στο τελικό αποτέλεσμα. Η πρότυπη υλοποίηση των στοιχείων αυτών έγινε για την συγκεκριμένη πλατφόρμα αλλά με τις κατάλληλες μετατροπές το περιγραφόμενο πλαίσιο εργασίας μπορεί να αποτελέσει ένα επαναχρησιμοποιούμενο πακέτο λογισμικού που θα αποσκοπεί στην διαχείριση του υλικού μιας πλατφόρμας με στόχο την παραγωγή εξατομικευμένου περιεχομένου αλλά και την οργάνωση και προβολή του υπάρχοντος, με κριτήρια αξιολόγησης που θα αξιοποιούν την πληροφορία που θα έρχεται από το εκάστοτε κοινωνικό δίκτυο και την αλληλεπίδραση των χρηστών.

Table of Contents

1. Introduction	5
1.1 Thesis Contribution	5
1.2 Structure of the Thesis	6
2. Important Software & Terms	8
2.1 Digital Storytelling	8
2.2 Recommendation Systems	9
2.3 STSP Data Model	10
2.4 Social Media Integration	14
2.5 Technological Background	16
3. Requirements Specification and Use Cases	21
3.1 Requirements Analysis	21
3.2 Reference Architecture	24
4. Framework Implementation	26
4.1 Architecture	26
4.2 Implementation Technologies	29
4.2.1 React	30
4.2.2 NextJs	31
4.2.3 Typescript	32
4.2.4 Apollo	32
4.2.5 Hasura	35
4.2.6 PostgreSQL	36
4.2.7 OAuth2	37

4.2.8 Continuous Integration	38
5. Framework Application	40
5.1 Approach.....	40
5.2 GUI.....	40
5.3 Applying Personalization to STSP	45
6. Conclusions – Future Work.....	47
6.1 Conclusions	47
6.2 Future Work	48
7. References.....	49

1. Introduction

The web has evolved significantly in the last 20 years, and nowadays web applications incorporate visual and audio storytelling to captivate users' attention and offer an exclusive and highly individualized experience. This has led to a massive increase in interest in recommendation systems in an attempt to enhance user experience and potentially increase sales. From choosing the best item to purchase and reading the "hottest article" in a newspaper, to watching a similar video on our favorite platform, recommendation systems play a vital role in increasing time spent on a website, and time usually is associated with money in the world of digital advertisement. At this point it is crucial to mention that a large percentage of users do not return to a website or application after their initial visit, thus suggesting that before businesses and organizations have the chance to recommend content to users, their focus should be aimed at ensuring users will keep returning to their platforms to view and interact with more content of interest as well as express their opinions and views about it. For this matter, a storytelling approach is often adopted, an approach that guarantees to etch a brand or service in audiences' minds. Storytelling is one of the most captivating techniques to convey messages and information. The main idea is to create engaging stories that make people vigilant listeners and if utilized properly on social media, turn online audiences into ardent followers.

1.1 Thesis Contribution

The aim of the current thesis is the design and implementation of a framework for personalized transmedia storytelling that will allow online platforms to utilize social media to increase user interaction and drive traffic to their websites, thus creating potential clients for their products/services. The framework consists of several components with clearly defined functionality, and there will be a detailed analysis of how they were implemented. The platform that was utilized for the framework's implementation is STSP, a digital platform for spatiotemporal visualization of historical events that have taken place in Crete and the promotion of services and products of local businesses in the field of tourism. The platform has adopted a storytelling approach to make the content unique and captivating, which will lead to increased user engagement and the addition of user-produced content. The end goal is the creation of personalized services and experiences for the users about their preferences and likes.

As we have already discussed, the main goal of the framework was to increase the transmedia storytelling capabilities of the platform. The first main feature of the framework in this direction is to support users' commenting and interacting with other users, as well as declare their fondness and interests in the form of ratings, likes, and bookmarks regarding the contents of each story. The second main feature was the

establishment of a connection between each story and Facebook posts on the platform's official Facebook page, to initially publish stories on social media and subsequently collect data about user engagement regarding a specific historical event. Thirdly, the framework supports the collection of comments, reactions, and shares about a story that comes from the platform's Facebook page and stores them in the platform's database to model and analyze user interaction coming from social media. After the storage, these can be viewed on the platform's relative page along with the original comments that were made on the platform itself. Evaluation of user interactions coming from both the platform and the Facebook page leads to the fourth and final feature of the framework, which is to support the creation of a smart recommendation system using the collaborative filtering algorithm. The algorithm considers a user's previous interactions with a story and proposes one or more stories that show a high resemblance to those he has already visited and rated.

A very important aspect of the framework is the fact that it is highly flexible, meaning that it can be viewed as a solution for businesses looking to incorporate user interactions coming from all types of social media networks and adjust their content and marketing strategy according to specific metrics of popularity as well as create smart recommendations for each user. Certain tweaks are required for the framework to incorporate certain social media networks, thus allowing for a large degree of customization for the platform owner. The Agile methodology was selected to ensure that frequent inspections would be made, leading to high software quality and organization. The process of constant re-evaluation led to a scalable and well-organized system, where each component of the architecture was meticulously tested before proceeding to the design of the next one.

1.2 Structure of the Thesis

There will be an overview of the content of each chapter of the thesis in this section.

Chapter 1 sets the frame regarding transmedia storytelling in modern web applications, the importance of recommendation systems, and the motivation behind the development of the framework that was designed, as well as a brief analysis of the functionality that it provides.

Chapter 2 explains the necessary terms that are required for a proper understanding of the work that was conducted. There will be an explanation of the term "storytelling" along with the commonly used storytelling tools. The chapter continues with mentions of modern recommendation systems and the data model of the platform on which the framework was implemented. Lastly, the chapter explains the importance of social media in transmedia storytelling and points to some of the most commonly used tools for social media integration.

Chapter 3 presents the requirements analysis that was conducted and displays the reference architecture of the framework that was designed.

Chapter 4 provides a detailed analysis of the architecture diagram of the framework, which contains all the necessary components and subcomponents along with their functionality. Additionally, the chapter provides diagrams of the core technologies that were used for the implementation of the framework and its interconnection with the storytelling platform.

Chapter 5 demonstrates the framework's use in the storytelling platform and clarifies the data model components that were used. It displays the necessary interventions in the graphical user interfaces of the platform and the mechanism behind the production of personalized content.

Chapter 6 summarizes the results that were obtained during the design and implementation process. It refers to general good practices in software development and the importance of social media integration in storytelling platforms. Finally, there are some proposals regarding future work on the framework.

2. Important Software & Terms

This chapter describes in detail all the important terms and tools that were utilized to describe how the framework was developed. All the above are state-of-the-art technologies that are being widely used in the industry and play a vital role in developing modern web applications that are safe, fast, and coherent. The below-mentioned terms include external code libraries, code editors, version control systems, database systems, servers, package managers, storytelling tools and frameworks, and some general elements and principles regarding how modern web development operates, all of which will be analyzed further in the upcoming paragraphs.

2.1 Digital Storytelling

To understand better what transmedia storytelling is, there was a detailed study of the work of **H.Jenkins**. He defines transmedia storytelling as “The process where integral elements of fiction get dispersed systematically across multiple delivery channels to create a unified and coordinated entertainment experience”. This process originated in the entertainment industry and mainly cinemas, but since then marketers have adopted it to promote products and brands in a way that is more appealing to young audiences. The digital era that we are all currently experiencing has paved the way for new forms of storytelling and complex narratives. In a world of digital connectivity, a new cultural context is being created where social media, connectivity, and online-data exchange are of great importance, and this leads businesses to change their marketing strategies appropriately. Overall, digital storytelling leverages an expanded and more participatory sense of audience because it can connect learners in disparate places and situations. Individuals create, post, share, respond to others, critique, and engage in other participatory activities around their digital stories [1].

There are several tools available online for both businesses and individuals so they can create their own unique stories, but the most commonly used ones are the below-mentioned:

- **Microsoft Timeline Storyteller:** It is an open-source expressive visual storytelling environment for presenting timelines in the browser. Its main use is to present different aspects of timeline data using a palette of timeline representations, scales, and layouts, as well as controls for filtering, highlighting, and annotating. The creator of a story must design a series of scenes for each story, where each scene is in a unique state and consists of designs, images, colors, and captions. Timeline Storyteller also allows animated transitions between scenes creating a coherent and desirable user experience [2].
- **Pageflow:** Pageflow is an open-source software and publishing platform for Visual Storytelling, jointly developed with the West German Broadcasting

Corporation “WDR“. Its modular structure allows the continuous addition of new types of pages and features. The latter is the main aim of it, which is to merge various multimedia elements and in conjunction with the full-screen capability, offer a great storytelling experience. The use of HTML5 instead of Flash allows it to be viewed on all devices, making it a great way to avoid using different tools for transmedia storytelling for desktop and mobile [3].

- **StoryMapJS:** A free JavaScript library that helps in telling stories on the web that highlight the locations of a series of events. A different slide can be used for each different location in our story, inside an easily customizable and interactive map that contains all our points of interest in a story. One can add events as part of a larger narrative or even mark the local businesses that can be associated with the location of the story. This is the tool that was selected for the project [4].
- **Europeana:** Digital Storytelling Prototype, which can be seen as a service, that allows users to create and publish collections of content and mix their uploads with content in Europeana and on YouTube. More than 3,000 institutions across Europe have contributed to Europeana, including the Rijksmuseum, the British Library, and the Louvre. Records of over 10 million cultural and scientific artifacts have been brought together on Europeana's platform and are presented in a variety of ways relevant to modern users, such as smartphones or APIs [5].

2.2 Recommendation Systems

In the digital era that we are experiencing, the amount of information that is available is increasing exponentially. This increase in data has led to an increase in interest in filtering these data from the existing storage, leading researchers to propose the concept of recommender systems. Recommendation systems are information filtering systems that deal with the issue of the constant production of information by filtering vital parts of the dynamically generated data according to user interests, preferences, and observed behavior about items. These systems make a well-designed and detailed analysis before they generate a prediction on whether a user would prefer interacting with an item, based on a set of characteristics that depend on the main idea behind the design of the recommendation algorithm.

Recommendation systems are beneficial to both service providers and users. They aid the decision-making process and help in making more accurate assumptions in a business environment. In an online shopping environment, they reduce the transaction costs of finding and selecting an item. In an e-commerce setting, recommender systems increase revenues, since they are effective means of selling more products. All these examples of recommendation systems utilization demonstrate the need for a smart and efficient system that will provide useful and relevant recommendations to the end users.

There are three main approaches as far as designing recommendation algorithms is concerned. These approaches utilize either collaborative filtering, content-based filtering, or hybrid filtering. Collaborative filtering algorithms work by searching a large group of people and finding a smaller set of users with likings similar to a particular user. It looks at the items they like and combines them to create a ranked list of suggestions. On the other hand, content-based recommendation algorithms work by trying to recommend items to users based on their profiles. The user's profile revolves around the user's preferences and likings as well as other characteristics of the user's behavior that the system can recognize. Both of the previously mentioned approaches face the issue of not being able to construct a relationship between the users and the items in the event of insufficient initial data, something which is very evident in the initial period after an application or website launch. In such cases, a possible solution is to use a hybrid recommendation algorithm, which could be implemented in various ways, such as using content and collaborative-based methods to generate predictions separately and then combining the prediction or adding the capabilities of collaborative-based methods to a content-based approach (and vice versa) [6].

2.3 STSP Data Model

This section presents the data model that was designed and implemented for the needs of the platform. Every story that takes place in the context of a story world is governed by a set of rules and constraints and is unique regarding a set of characteristics (era, architecture, tradition, etc.). The main elements of a story are time and place, along with things such as artifacts, language, technology, ideas, and beliefs, all of which can be utilized when describing a story world. In the context of the STSP platform, a detailed analysis of multiple historical sources was conducted to ensure that each historical event would be presented accurately and in an effective way. This process was needed to ensure that all the important parameters of a story would be included during the design and creation of our database. The basic building block of a story is an element. An element takes place in a specific geographic area at a given time. Several interconnected elements constitute a story. Some of them are larger in size and detail and can be viewed as smaller, independent stories with their own media resources.

Apart from the main stories the platform initially contains, there is an infrastructure to support the addition of extra stories that could be individual experiences or narratives of people and could be used by the media to promote products and services. These unique elements are called "user elements" and can be either linked to other user elements or elements from the original real-life stories of the platform, thus forming "chains" of events. In this way, relations have been created between all the basic building blocks of the story platform.

Figure 2.3-1 is a representation of all the entities that exist in the data model and the conceptual relationships between them.

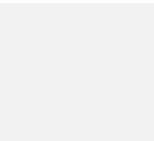
Term	The symbol in the UI	Description	Notes
Super User(Historian)	-	The User creates a story, modifies, formats, and controls the content that is stored in the system.	Not to be confused with the system admin who has all the rights for the technical aspects of the system.
User	-	An authenticated user can add user elements, create event chains and polls, share, comment, and react to a story.	Not to be confused with the guest user, who can only view the content of the platform.
Story Element	-	The building block of information of every Story.	Provided and controlled by the Super User(Historian).
User-contributed Story Element	-	The building block of information of every Story(Provided by Users).	Even though it is provided by the User, it is controlled by the Super User of the Story.
Story		Story Elements and User Contributed Story Elements that are linked.	Provided initially by the Super User(Story Elements) and afterward by the User(User Contributed Story Element).
Chain		Type of Story that links Story Elements and User Contributed Story Elements.	Provided by both the User and the Super User.
Relation		Type of Story that links individual Stories.	Provided by the Super User.

Figure 2.3-1 Terminology Table of the STSP Model

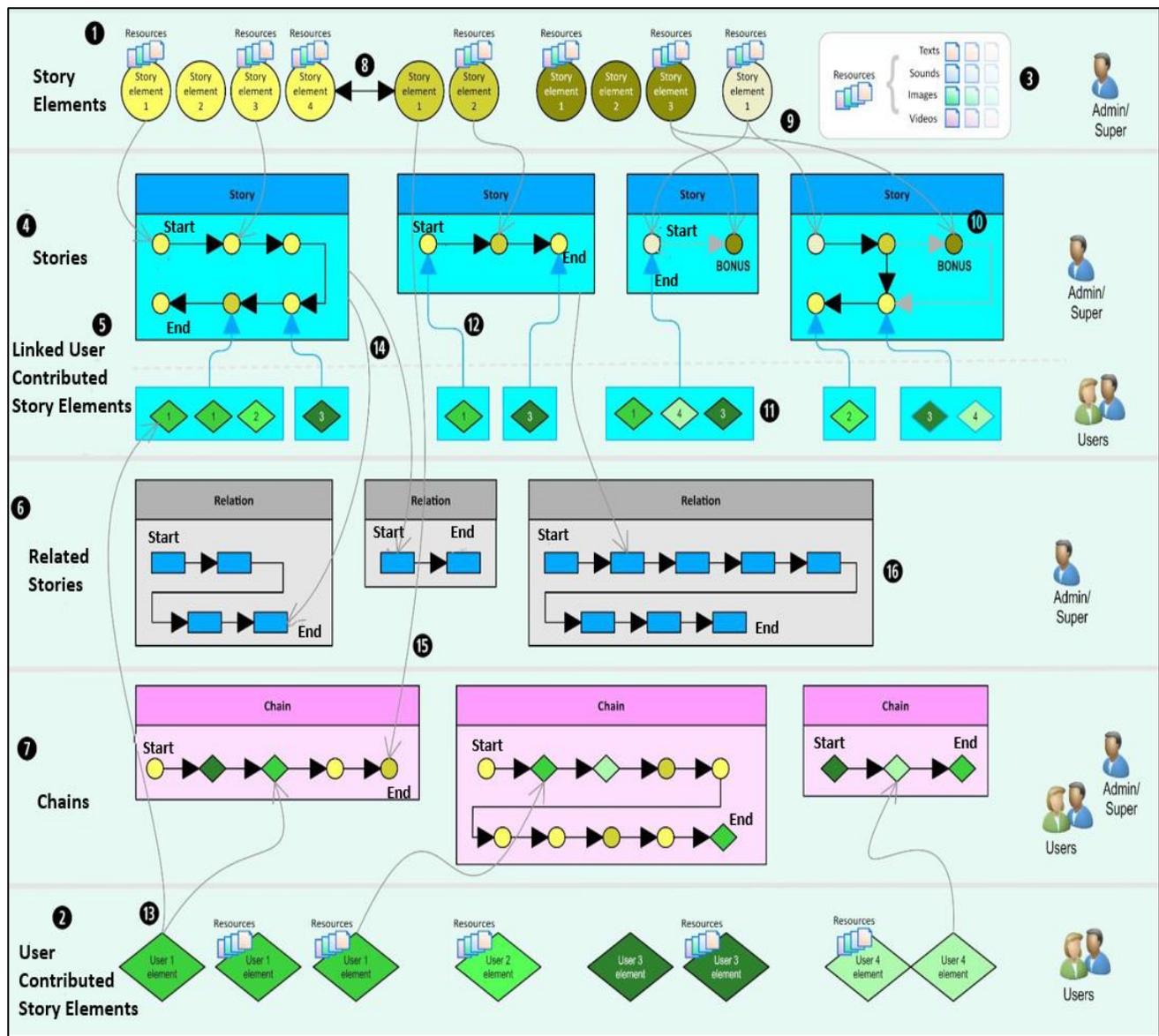


Figure 2.3-2 Basic Building Blocks of the Data Model and their Interconnections

- ① Story Elements are the basic units of information. In each story, there are one or more and they could potentially be linked with media resources. Super Users are responsible for the creation and management of linked relationships.
- ② User's contribution to the platform's stories. Similarly, to the story elements, there can be a great number of them and can be linked with media resources. Super Users manage them, but they are created by authorized platform Users.
- ③ Resources can be video, photographs, sound clips, and texts as well as all the necessary meta-data for their utilization.
- ④ Stories are the higher-level structural units of the platform. They consist of story elements and potentially user-contributed story elements. These elements are placed in chronological order inside a story.

- ⑤ One or more user-contributed elements could be linked to create a story event that is displayed in a story. The platform encourages the user to make this link themselves.
- ⑥ Relation is a type of story that consists of other stories that are linked to each other. These linked relationships are controlled by the superuser(Historian).
- ⑦ Chain is a type of story that consists of story elements and user-contributed story elements that are linked to form a new type of story.
- ⑧ Chain can be used to demonstrate a relation between story elements.
- ⑨ A story element could belong to one or more stories.
- ⑩ Some story elements are not available from the start when a viewer clicks on a story. They can be shown only after a user has completed certain tasks or spent enough time in the platform, and can be characterized as bonus elements, and are part of the general marketing strategy.
- ⑪ User-contributed elements can be willingly linked to a story element inside a story (see 5).
- ⑫ A single user-contributed element can be linked to a story element inside a story (see 5).
- ⑬ User-contributed elements can be displayed either inside a chain or inside a story (Although they cannot be considered official parts of the story, due to them being shown only under the condition that they are linked to a pre-existing story element.)
- ⑭ Several stories can be linked to create a relation, which is effectively a series of stories. One story can appear in different relations.
- ⑮ Story elements can be part of a chain (see 7).
- ⑯ Different relations can pave the path for the creation of new relations, which are up to individual preferences and choices, through their linked stories (see 6).

The last diagram of the subsection is the visual representation of the relations that exist between all major and minor STSP components. It contains details about the spatiotemporal state of a story, the main actors, and their actions as well as parts of the social media aspect of the platform. This is the first edition of the data model and is bound to change in the future.

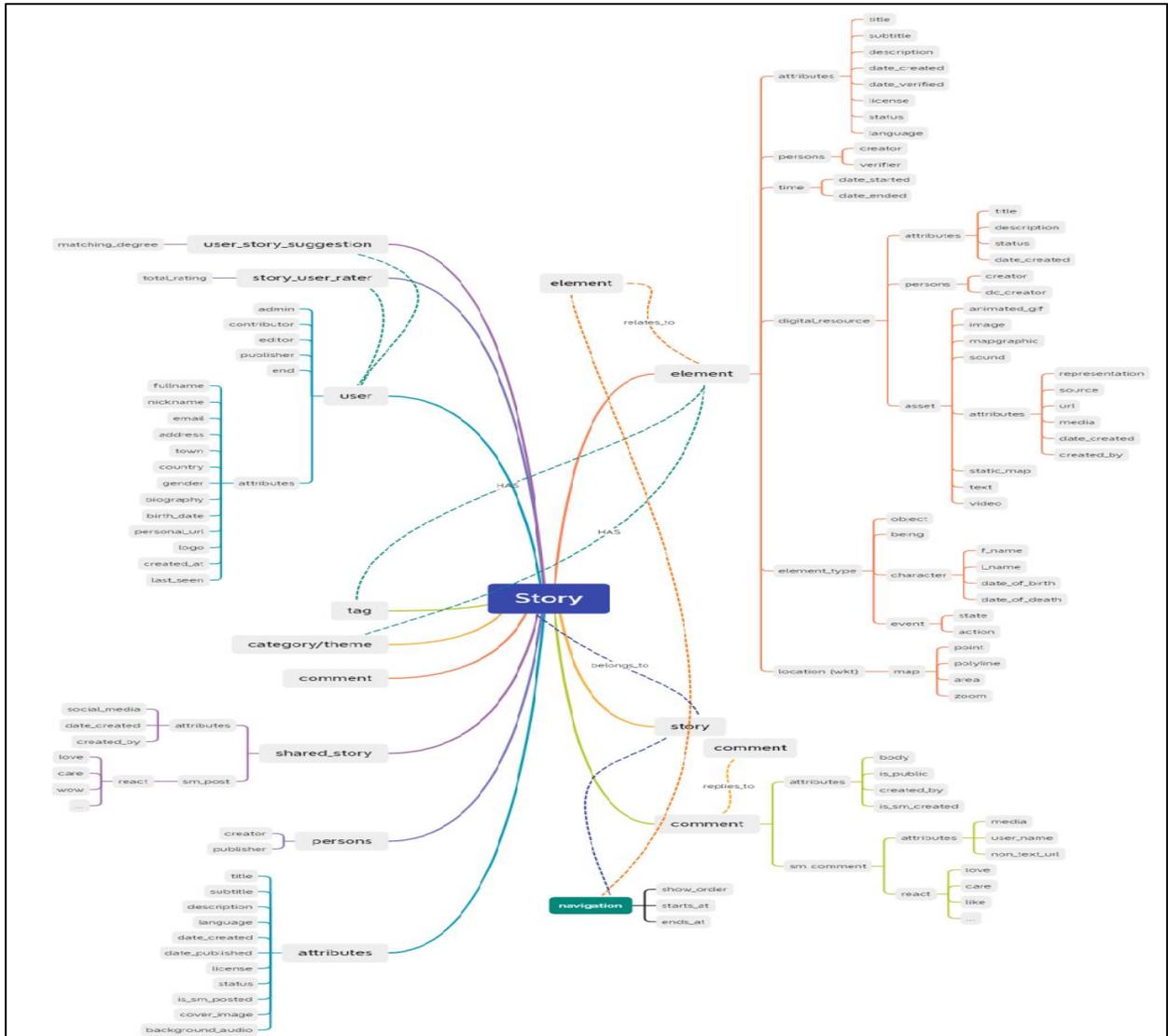


Figure 2.3-3 STSP Basic Entities Model (1st Version)

2.4 Social Media Integration

Social media integration is the act of using social media accounts as an addition to your marketing. It can be viewed as a tool that will promote brand awareness and increase the visibility of marketing campaigns by creating opportunities on various social platforms for customers to link or share information between the website and social media. The two main ways through which integration of social media can be accomplished are:

- Display of information derived from the web platforms to social media to encourage users to visit and interact.
- Allowance of social media accounts to be easily accessible on websites or applications.

It is important to note that the existence of social media-derived elements on the website doesn't directly translate into user engagement, so it is crucial to accomplish this integration without compromising user experience (UX) [7]. There are a few ways to facilitate social media integration, and below there will be a brief analysis of the most used ones :

- **Social Media Sharing Links:** Tools such as Mashable integrate a 'Share button', which is linked to social media, blogs, and product pages, making the sharing of product pages or blog pages from customers to their social channels easy [8]. This is a very common strategy mirrored by many media companies and corporations to aid their website conversions.
- **Social Widgets:** To create a form of two-way communication between the website and the social media platform, brands and organizations utilize widgets, which essentially are an easy way for customers to share information or products that they have found with their network.
- **Social Embeddings:** This is the most common way for businesses to integrate customers' social media into the web platform. Social Logins are a single sign-on for users that utilizes existing information from a social network provider such as Facebook, Twitter, Google, or Pinterest. Social media providers can give additional information to the applications, such as location, interests, birthdays, and more. The utilization of this data is very crucial as it can help in targeting personalized content to the user, which is something that this thesis focuses greatly on.
- **Display Social Media Multimedia Content:** Embedding social media videos, images, and audio is another great way of driving traffic from websites to social media channels. This process is simple and many tools are available to simplify it, especially WordPress plugins such as Smash Balloon Facebook Feed Plugin [9] and RafflePress [10]. Each one of the plugins has different features, but the main idea behind all of them is to showcase custom content fascinatingly and subsequently attract more views, followers, subscribers, etc. The main disadvantages of many of these plugins are that, firstly, they require payment to be efficiently used in a production environment, and secondly, sometimes they significantly slow page speed and are more susceptible to hacks due to infrequent security updates.
- **Social Media-Based Commenting System:** Commenting is a very important part of online engagement. By utilizing commenting systems, businesses increase the involvement of customers with the application's content and encourage the expression of ideas and opinions. Large companies such as Facebook and Google have developed such systems that are easily integrated into existing web platforms. There are also different styles of commenting systems, such as Disqus [11], that provide customized dashboards, analytics, and real-time commenting mechanisms.

By displaying reviews, testimonials, and shoutouts, brands can showcase that they are credible and maintain a following. This motivates new customers to engage with the content and helps in maintaining existing customers that stay associated with the platform. Some of the ways that social proof can be accomplished is by incorporating social media feeds and posts coming from networks such as Twitter and Facebook where users explain their experiences with the product or services.

To sum up, to create a strong digital presence, businesses focus on integrating social media to expand their customer pool. Choosing the right social plugins for a website is crucial to facilitate the increase of user engagement and increase brand recognition. At this point, it should be mentioned that there is not an all-around solution for real-time integration of social media, as the current ones can be viewed more as external components that are utilized. Practically this means, that currently most tools revolve around solely displaying content coming from social media. The framework of discussion in this thesis aims to offer additional capabilities to storytelling platforms and organizations through the storage of content and interactions coming from social media and their utilization to produce user-specific content as well as its association with the native content of the platforms. All the above mentioned will lead to a more systematic integration of social media into the storytelling platforms but more on this matter will be analyzed in the upcoming chapters.

2.5 Technological Background

Since 1989 and the invention of the WWW (World Wide Web), most of the physical services have been transferred to the cloud. Until that period, for a user to perform online activities, they would have to install different desktop applications, but the rapid evolution of the internet changed every aspect of this process. Thanks to WEB 2.0 and WEB 3.0, we can now enjoy dynamic, responsive, and interactive web applications just by having access to the internet and owning a computer. Along with the evolution of the WEB protocols, a major change also took place, related to the web architecture. The traditional, monolithic client-server model shifted to a highly scalable microservices one with an emphasis on the ability to handle large loads of data and user traffic [12]. There are some common building blocks, though, no matter what architectural style one chooses, with which every developer (or web enthusiast) should be familiar before attempting to build web applications.

Server

A computer (or network of computers) that provides a service (or multiple services) over a private network or internet can be referred to as a server. Other devices, known as clients, can connect through different network ports to obtain the provided service. Servers are named based on the type of service they provide, and we may see different servers in a single application, such as Web Servers, Authentication Servers, Database Servers, Application Servers, File Servers, and Mail Servers.

Client

The client can be described as a computer, software, or website that connects with servers to consume their services or resources. The standard example of a client is our web browser when we are visiting a website. The naming convention for clients is similar to the server's, meaning that clients' names are based on the service they consume. The types of clients that we see most often are Web Clients, Database Clients, Email Clients, Online Communication Clients, and File Clients.

There are two main client types: A) **Thick clients** and B) **Thin clients**.

Thick clients don't depend on a server and can be viewed as a standalone app that persists data. Thin clients on the other side, depend entirely on a server similar to SPA(Single Page Applications).

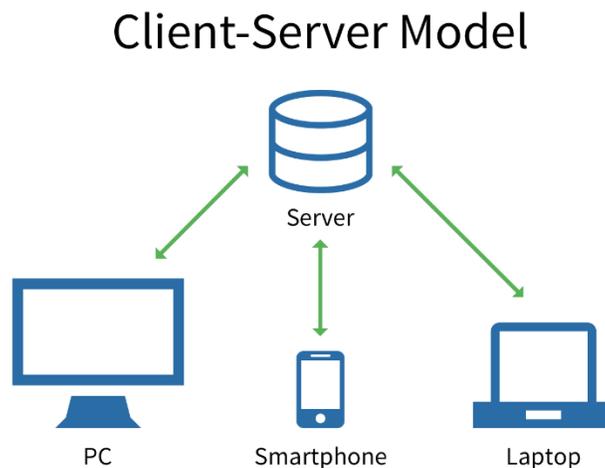


Figure 2.5-1 Client-Server Model in Web Applications

Single Page Applications

A SPA(Single Page Application) is a type of web app that loads only a single web document, and then updates the body content of that single document via JavaScript APIs, without refreshing the whole page leading to performance gains and a more dynamic experience without the need of loading new external files. The goal is to minimize the refresh time and escape from the typical request-response cycle. The main advantage of SPA is speed and as Amazon quotes "1 second of additional delay in page load costs 1% of sales (which, considering Amazon's number of sales, is \$1.6 billion per year.)",so it is easily understandable why SPAs gain more and more attention from developers every single day. [13] It is important to note though, that it is not always better to build an application

as a SPA, because if users have low-power devices, they will experience poor performance and memory leaks may be observed. [14]

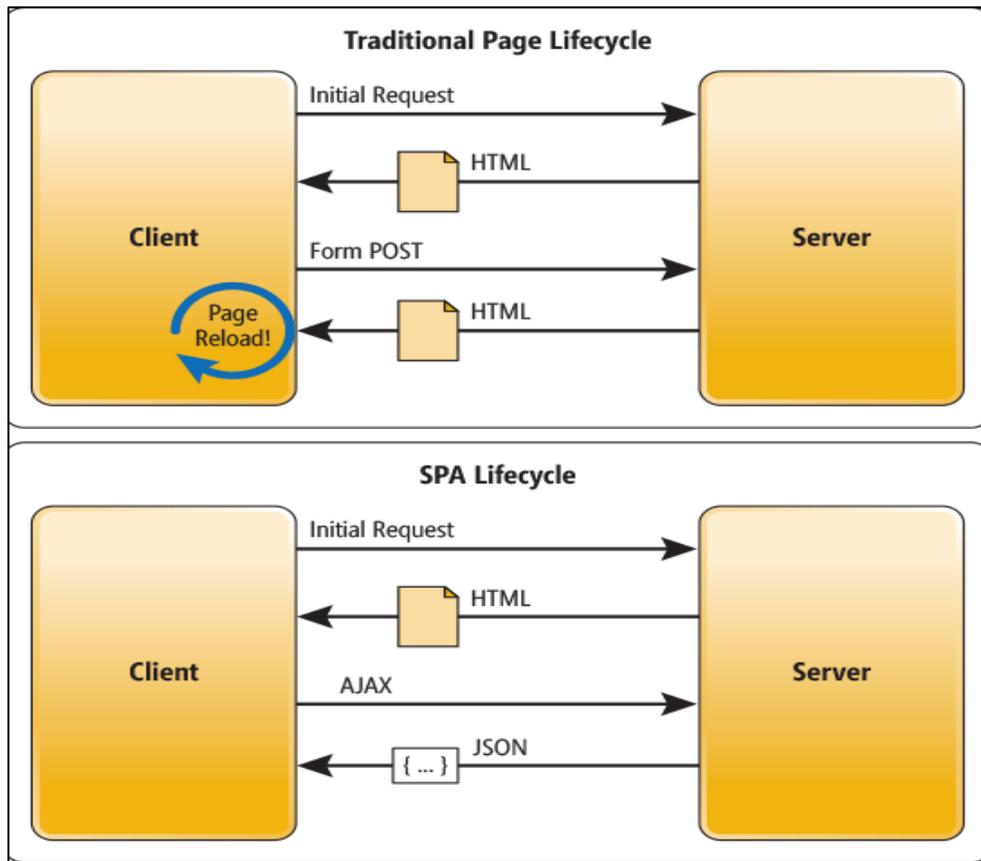


Figure 2.5-2 Traditional Architecture vs SPA

Representational State Transfer (REST)

Representational State Transfer, or simply Rest, is an architectural style to facilitate communication between computer systems on the web. Systems that have adopted this style are called restful and are characterized by how they are stateless and separate the concerns of the client and server. Practically, this means that we can develop and implement the client and the server independently, without information about how the first, or the latter, was developed. The greatest advantage of this method is that potential changes to one do not affect the other, thus achieving client-server separation. The only requirement to maintain this modularity/separation is that each side knows the format of the message to send to the other one. By using REST interfaces, different clients hit the same REST endpoints, perform the same actions, and receive the same responses.

Microservices

The traditional client-server model entails some dangers as a malfunction in one of the parts affects the whole system, and this can lead to a constant need for maintenance work. The microservices architecture model aims to decompose large systems into small services that operate independently and constitute building blocks for larger systems. A microservice refers to a loosely coupled and isolated service that is responsible for a particular process. In real-world web-based software systems, most microservices are RESTful APIs running inside a virtual machine or container. The main benefit of this architectural style is that systems can be easily scaled, and faults can be isolated and fixed much more resiliently.

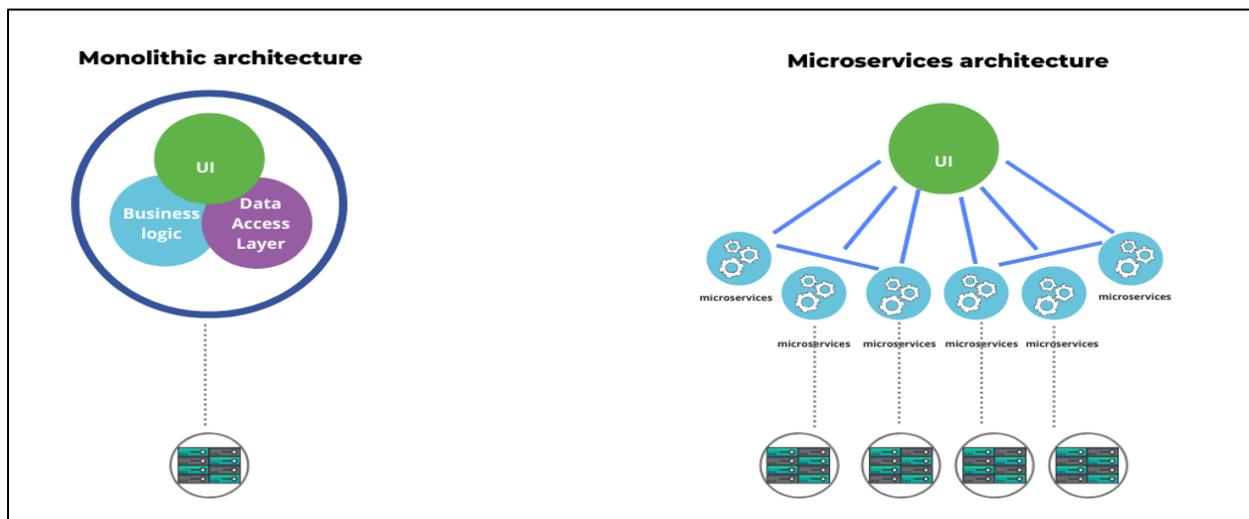


Figure 2.5-3 Monolithic Architecture vs Microservices Architecture

GraphQL

GraphQL is an alternate option to build APIs in REST. It is a query language that is used in many applications, as most of them have data hosted on a remote server in a database. The API only must expose an interface to put-away information that meets application requirements. GraphQL defines specifications on how the client will request all sorts of data from a remote server. The server's response is sent back as a result of the client's specific query. Overall, the main advantages of GraphQL over REST are the fact that firstly, we can obtain multiple resources with a single request, instead of multiple requests to different endpoints, and secondly, we can specify exactly what type of information we want to fetch from a server, thus avoiding over fetching which can lead to performance issues. [15]

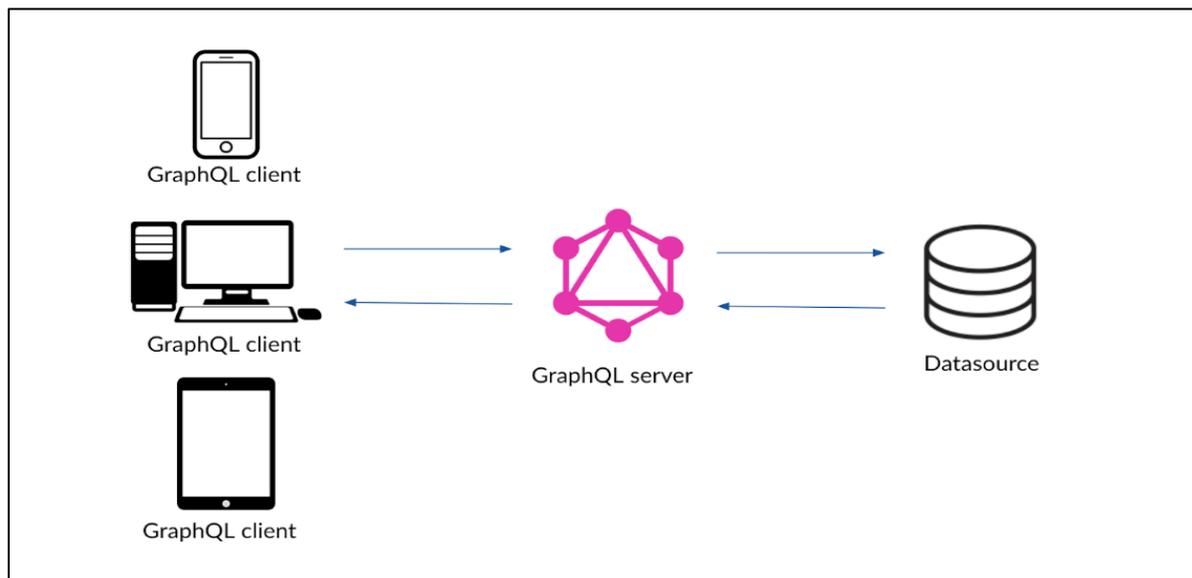


Figure 2.5-4 GraphQL Interface for Data Retrieval from a Backend Source

Web Components

Over the last decade, web components have evolved due to industry change. Some of these are tools, some are ideas, and others are approaches, but all of them are related on the basis that they are being used to manage a balance of complex requirements and performance with safety and simplicity. The need for a faster and richer user experience led to the creation of concepts such as single-page web apps. Lightweight and dynamic applications that could manage large amounts of data exchange and rapid state changes, with the unique characteristic of being able to manage application logic on the server rather than the browser. This changed the old-fashioned view of the distinction between frontend and backend technologies and paved the way for new, innovative frameworks such as Node.js, React.js, Angular.js, etc. [16].

There are three main elements of modern-day web components :

- Custom elements are a set of JavaScript APIs that you can call and define in a customizable way to fulfill the application's needs.
- This acts as a "DOM" that is attached to the page's components. In this way, all unique elements and recourses are isolated, which makes the development process easier and offers better performance for users.
- Html Templates: readily available pages of HTML that can be reused and called when needed.
- By using them, developers aim to handle more complexity and workload while minimizing resources and time spent on development.

Moving forward, there will be an attempt to showcase the analysis that was conducted for the documentation of framework requirements, the corresponding reference architecture that was designed, and the relevant use case diagram.

3. Requirements Specification and Use Cases

This chapter describes all the requirements that were set for the platform that has been developed [17]. Firstly, there will be a brief mention of the requirements that were set and relate to both the framework and platform functionality along with the corresponding use case diagrams. Secondly, having established these requirements, there is going to be a display of the reference architecture of the system and brief analysis of the functionality that each. It is crucial to mention that the development process began, only after the design and cost analysis were completed.

3.1 Requirements Analysis

This section analyzes firstly all the requirements that were set for the framework to support the desired functionality and have certain quality characteristics. Secondly, there were some requirements for the platform to be able to support the interconnections to the framework and facilitate the framework development. For each set of requirements, there will also be a relevant use case diagram.

Framework Requirements

It is important to note, that each of these requirements was set to secure that the framework implementation in the application is leading to the accomplishment of four major goals: Personalized content served to people at the right times (recommendation engine), the interaction of the users with the content (co-creation and co-management of user-contributed elements), collection of social media interactions and publishment of platform stories to social networks.

1. The framework should transfer a story from the web platform to the social media page once it's published.
2. The framework should collect comments, ratings, and reactions from the platform's social media page.
3. The framework should check at regular intervals about new content that is posted on the social media page and subsequently transfer it to the platform.
4. The framework should transfer new comments that are made to a story page to the relevant social media page.
5. The framework should store the content from social media and display it on the relevant story page.
6. The framework should adapt the inbound content from Facebook to the platform's storage model.
7. The framework should adapt the outbound content from the platform to the social media API model.
8. The framework should utilize the stored reactions, ratings, and comments from both social media and the native platform to create the necessary vector matrices for the recommendation algorithm.

- The framework should run the recommendation algorithm at regular intervals to produce new and updated personalized story suggestions.

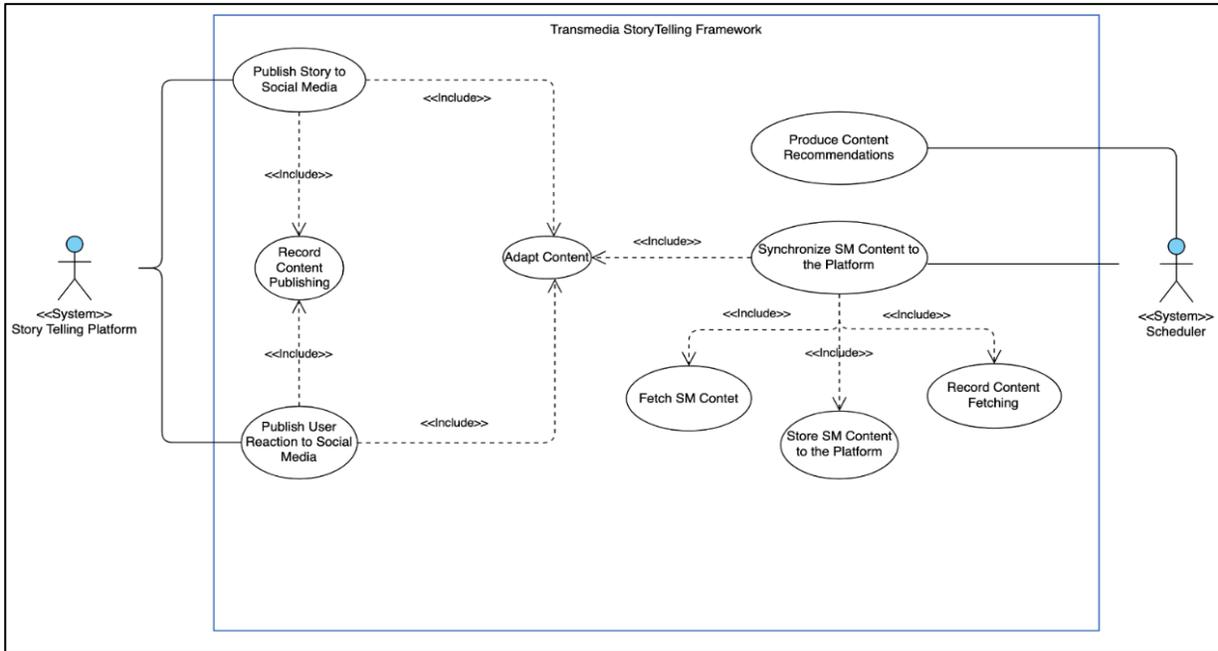


Figure 3.1-1 Framework Requirements Use Case Diagram

Storytelling Platform Requirements

In general, storytelling platforms have the infrastructure to support a specific data model as well as user interaction such as commenting systems or forms of rating. Having established this, emphasis on the requirements analysis will be given to those that mainly facilitate the development of the framework. The first thing that should be ensured and refers to the data model of the platform, which is utilized by the framework, is that changes to the basic structure of stories in the form of additions or extensions should be simple and not cause malfunctions. Extending the list of stories should be done simply. Any type of interconnections that have been made between basic data blocks of the platform such as story elements, user-contributed elements, social media elements, chains, relations, stories, and authorized users should never be affected.

As far as the web platform is concerned, it was deemed necessary that it should fill a series of requirements, a lot of which are crucial for the proper functionality of the framework and its interconnection to the platform, such as :

- The platform should support role-based access control and make the distinction between authorized and non-authorized users.
- The platform should support user registration and authentication.
- The platform should support viewing specific story elements inside a story.

4. The platform should support viewing the tags of a story.
5. The platform should support the rating, commenting, and sharing of a story.
6. The platform should support comment replying.
7. The platform should support user-contributed media elements additions to stories.
8. The platform should support users' addition of a story to their favorite stories.
9. The platform should support users' bookmarking of a story.
10. The platform should support the creation of a profile page for each user.
11. The platform should support the modification of users' personal information on their profile pages.
12. The platform should support users' viewing of the list of their favorite stories.
13. The platform should support users' viewing of the list of their bookmarked stories.
14. The platform should support users' viewing of the stories that they have recently accessed.
15. The platform should support users' viewing of the average rating of each story.
16. The platform should support users' viewing of their chains.
17. The platform should support user's viewing of their story relations

Figure 3.1-2 displays the use case diagram of platform requirements.

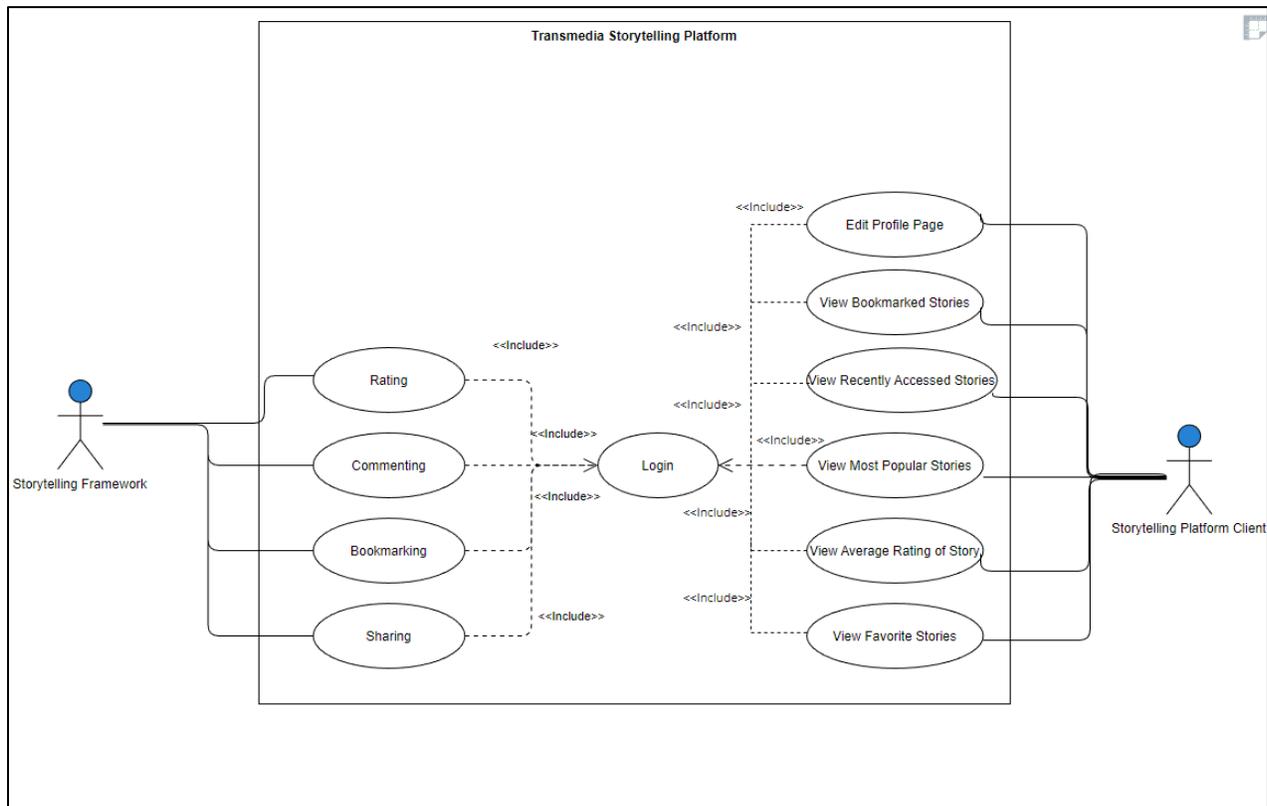


Figure 3.1-2 Platform Requirements Use Case Diagram

3.2 Reference Architecture

In figure 3.2-1 there is a display of the reference architecture of the framework that highlights its features and its interconnections to the web platform, as well as the communication with the social media APIs.

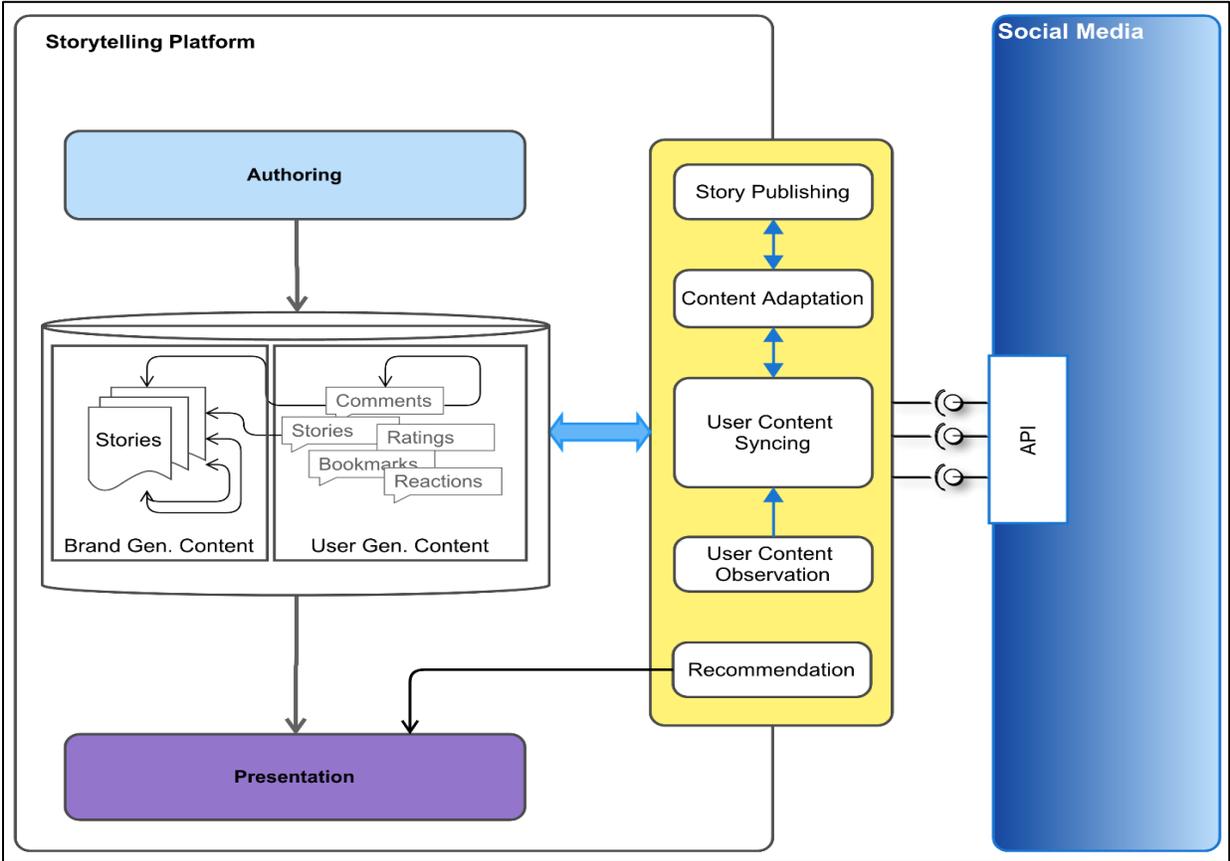


Figure 3.2-1 Reference architecture of the framework-Interconnections between the Framework and STSP Platform

It is important to emphasize that the five components (*Story Publishing, Content Adaptation, User Content Syncing, User Content Observation, and Recommendation Engine*) that are located inside the yellow container are the main components of the framework that carry out the desired functionality, which was a product of the requirements analysis that was done in the previous subchapter. Each of the components fulfills the below-mentioned requirements that were set for the framework. The number next to each subcomponent refers to the corresponding requirement number in the previous subchapter:

- Story Publishing: 1

- Content Adaptation: 6,7
- User Content Syncing: 2,4,5
- User Content Observation: 3
- Recommendation: 8,9

Social Media API could be any major API of a social network, but in our case Graph API of Facebook was utilized. As one can notice, at the storage level there is content that is derived both natively from the platform as well as coming from social media and is managed by the super users of the platform. The number next to each part of the storytelling platform refers to the corresponding platform requirements numbers in the previous subchapter:

- Platform's interconnection to the framework: 1,2,5,6,7,8,9
- Presentation of transmedia content in the platform:
3,4,10,11,12,13,14,15,16,17

The next step in the description of the framework is to go in-depth into its structural units of it and reference the technologies that were used for their implementation.

4. Framework Implementation

This chapter will present the main components of the framework architecture that were touched upon in subchapter 3.2 along with their proposed functionality to obtain a better idea of the thought process behind the framework design. Each major component will be analyzed along with its subcomponents, their role in the framework, and their position in the architecture layer. Additionally, in subsection 4.2, there will be a deeper dive into the technologies that were selected to design these components as well as the thought process behind choosing them.

4.1 Architecture

Figure 4.1-1 displays the two-layered component and subcomponent architecture of the framework. Having recognized the five main components of the framework from chapter 3, their implementation was based on fulfilling the major requirements that were set for it in subchapter 3.2 and related to social media integration, an increase of user interaction, story content publishing, and personalized content suggestions.

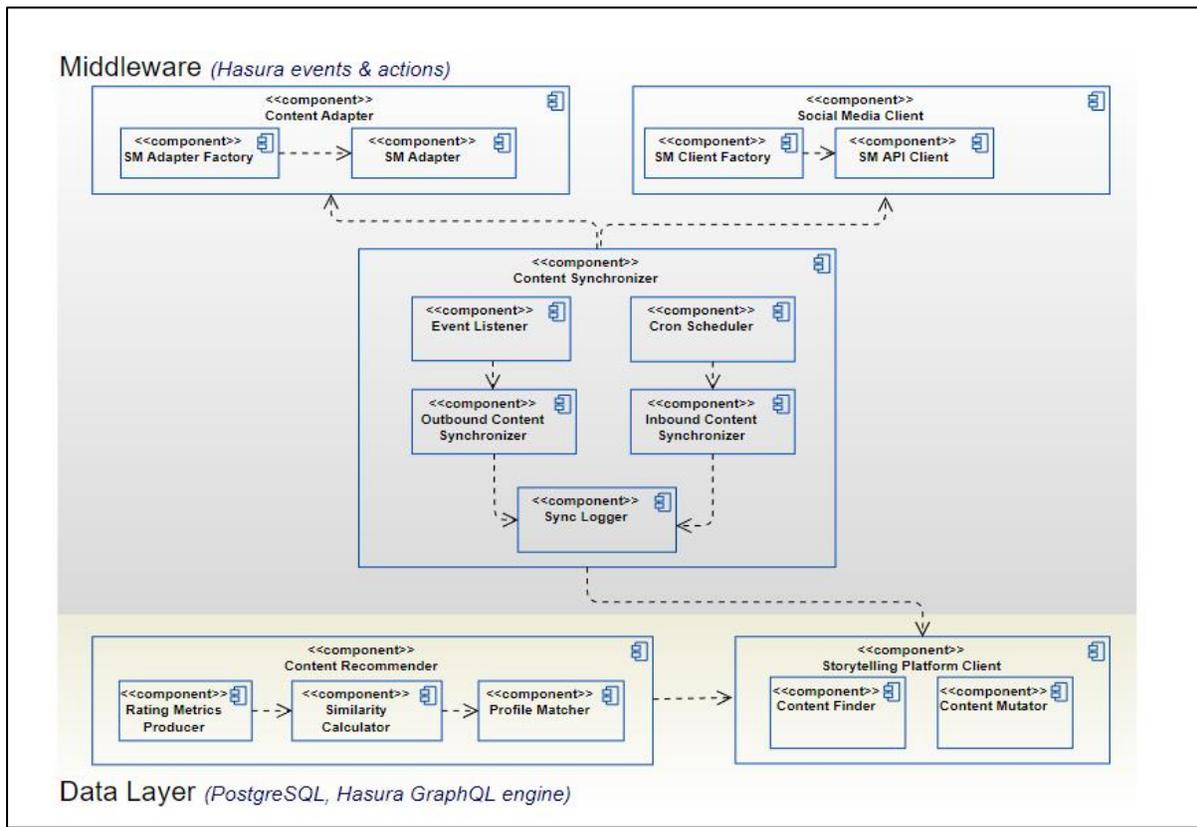


Figure 4.1-1 Framework Component Architecture-Display of Subcomponents and Their Main Functionality

Content Synchronizer

To ensure that the transfer between the native STSP platform and the social media page is updated, it was necessary to create a component that would handle this process. Sync loggers are keeping track of what content has been transferred at any given moment, and event listeners are being utilized to send platform-generated content to social media pages right after its creation. To automate the process of receiving content from the page, a Cron scheduler was used that executed this task at specified intervals.

Social Media Client

This component's purpose is the integration of any API of major social media networks. It handles the type of content that is shown in social media, and the proper metadata, and establishes communication with the API.

Content Adaptor

The major functionality is handling the structure of the content that is exchanged between the storytelling platform and social media. The social media-generated content needs to be adapted to the storage component data model and, correspondingly, the content that is transferred from the platform to the social media page needs to be adapted to the API standards.

Client of a Storytelling Platform

This represents the functionality that is being executed on the native storytelling platform that interconnects to the framework and mainly refers to the content that is stored there and CRUD operations on it.

Content Recommender

This chapter describes the algorithm behind the implementation of the recommendation engine component, which is a crucial part of the proposed framework. In more detail, it is an item-based collaborative filtering recommendation system that utilizes cosine similarity [18].

Cosine similarity is a metric used to measure how similar the two items or documents are irrespective of their size. It measures the cosine of an angle between two vectors projected in multi-dimensional space. This allows us to measure the similarity of documents of any type. It is defined as follows:

$$\text{Similarity}(p, q) = \cos \theta = \frac{p \cdot q}{\|p\| \|q\|} = \frac{\sum_{i=1}^n p_i q_i}{\sqrt{\sum_{i=1}^n p_i^2} \sqrt{\sum_{i=1}^n q_i^2}}$$

Figure 4.1-2 Cosine Similarity Equation for Computing Similarity between Two Vectors

The next step is the most important one, and it is to generate the output interface in terms of prediction. As soon as the set of most similar items based on the similarity measures is isolated, the focus is shifted to examining the target user ratings and using a technique to obtain a prediction. In this implementation of the algorithm, the weighted sum method was utilized, which is essentially a method that computes the prediction of item *i* for a user *u* by computing the sum of the ratings given by the user on the items that are similar to *i*. The total rating of a user for a story is calculated based on his star rating and whether they have liked or bookmarked a story. Each rating is weighted by the corresponding similarity *S_{i,j}* between items *i* and *j*. The final step is to scale the weighted sum by the sum of the similarity terms to make sure the prediction is within the predefined range.

$$P_{u,i} = \frac{\sum_{\text{all similar items, } N} (s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, } N} (|s_{i,N}|)}$$

Figure 4.1-3 Prediction of Rating of an Item for a Specific User

For example, if User 1 has rated two items (Story 1 and Story 3) out of a set of three stories, the algorithm will calculate the potential rating for Story 2. The following figure illustrates the above-mentioned process more clearly.

$$r(U_1, I_2) = \frac{r(U_1, I_1) * s_{I_1 I_2} + r(U_1, I_3) * s_{I_3 I_2}}{s_{I_1 I_2} + s_{I_3 I_2}} = \frac{(2 * 0.9) + (3 * 0.869)}{(0.9 + 0.869)} = 2.49$$

Figure 4.1-4 Rating of Item_2 for User_1

- $r(\mathbf{U}_1, \mathbf{I}_1)$ = Rating of Item_1 by User_1

- $r(\mathbf{U}_1, \mathbf{I}_3)$ = Rating of Item_3 by User_1
- $r(\mathbf{U}_1, \mathbf{I}_2)$ = Proposed Rating of Item_2 for User_1
- $S_{I_1 I_2}$ = Similarity between Item_1 and Item_2
- $S_{I_3 I_2}$ = Similarity between Item_3 and Item_2

The same process continues for all possible ratings of an item that can be calculated for a user. The final step is to compute the item with the highest rating, which will be the outcome of the algorithm.

4.2 Implementation Technologies

In this subchapter, there will be a quick presentation of each of the core implementation technologies that were utilized in the framework and the mechanisms that were developed for the framework's interconnection with the web platform. Figures 4.2-1 and 4.2-2 display these technologies and their use in the architecture. The same core technologies were utilized to implement the framework functionality.

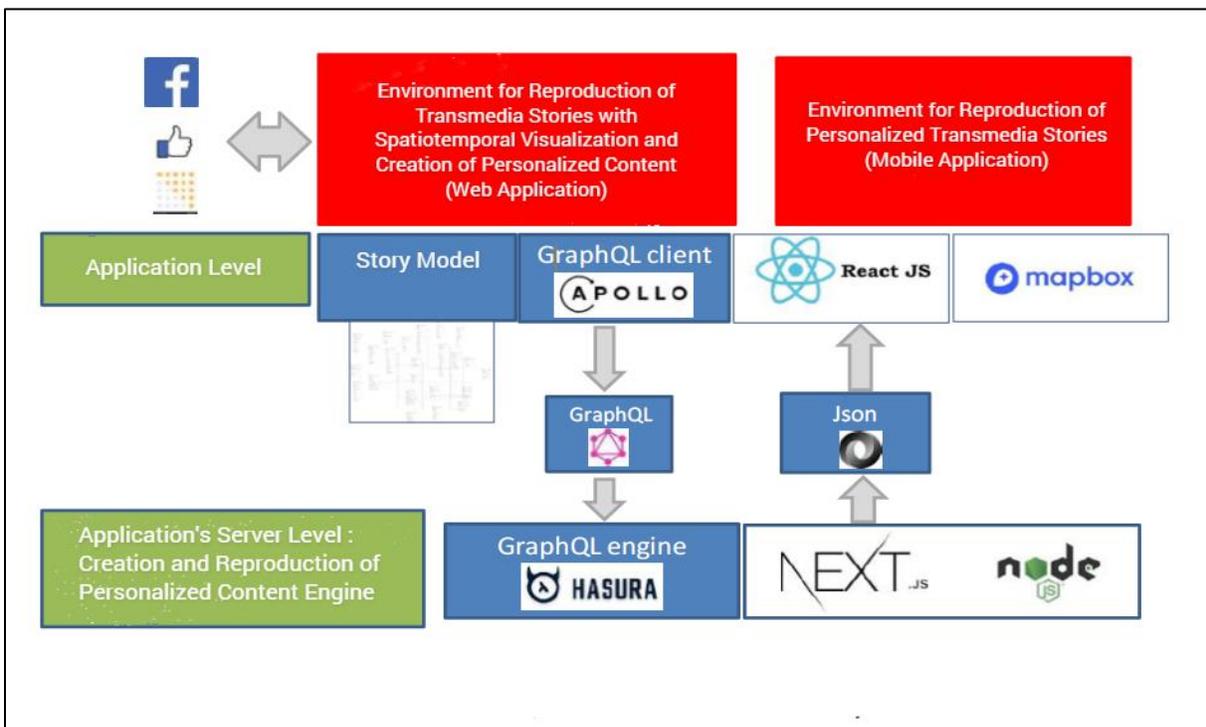


Figure 4.2-1 Three-Tier Web Architecture for the Creation and Enrichment of Multimedia Stories

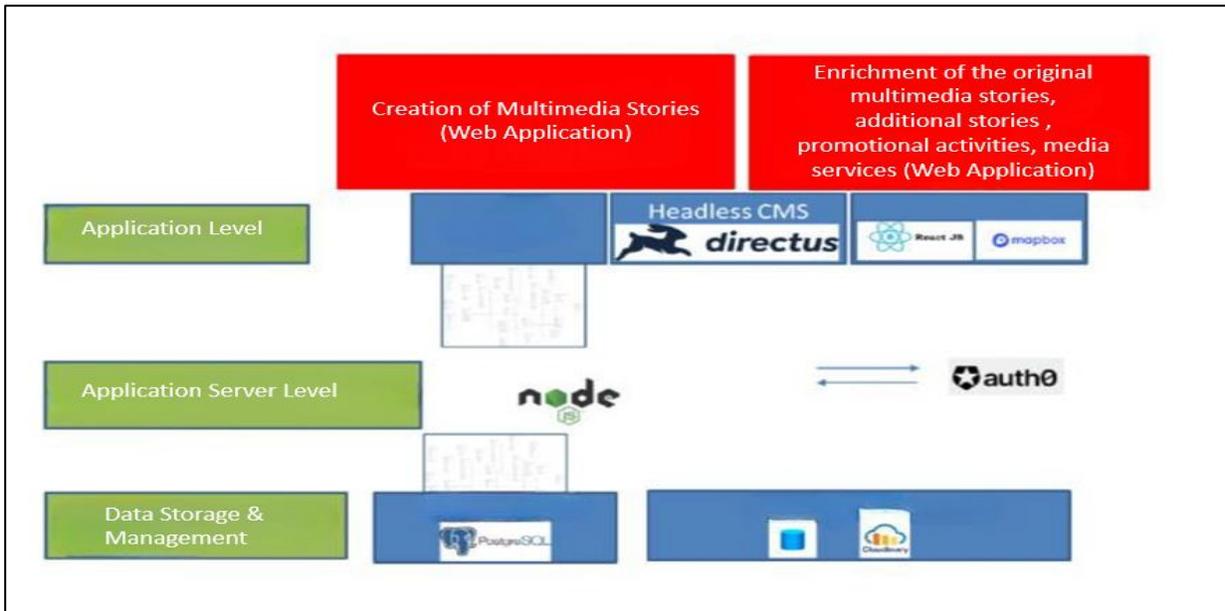


Figure 4.2-2 Three-Tier Web Architecture for Creation and Reproduction of Personalized Content

4.2.1 React

React is a JavaScript framework for building user interfaces simply and efficiently. Launched in 2013 by the Facebook developers' team, it quickly rose in popularity and today it is used in most large companies (Apple, PayPal, Netflix, etc.). By using it, we build encapsulated, autonomous components that manage their state and can be used as building blocks for complex UIs. What makes React one of the most used frontend libraries is that it's very fast, something that is achieved by the unique ability to update and change only the right components whenever a change is detected in the state of the page data.

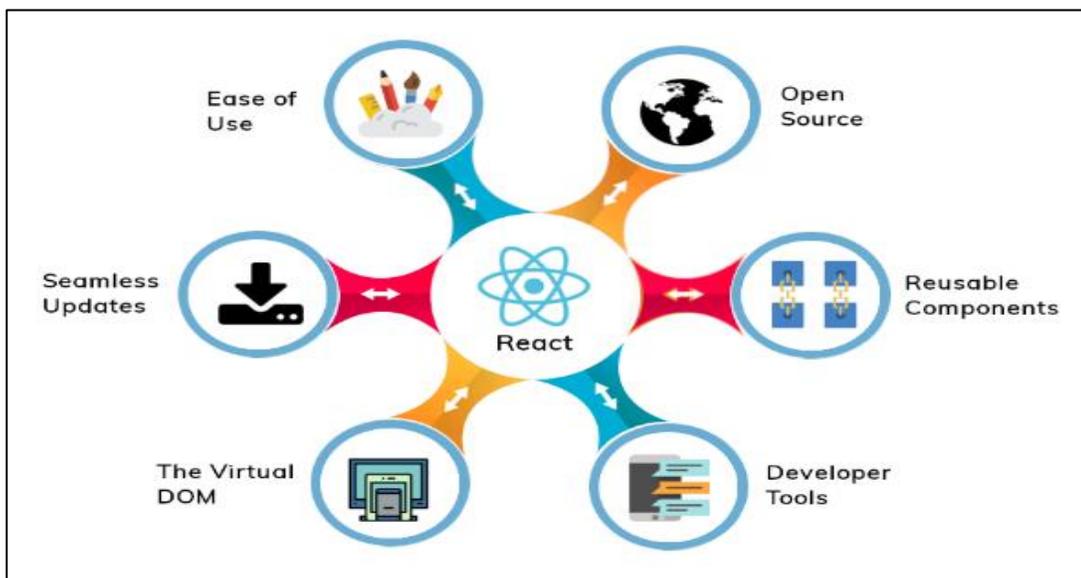


Figure 4.2-3 React.js Main Features

Reusability

Developers can create components that will be used in multiple applications, thus effectively reducing time spent in development and debugging mode and ensuring flawless performance. The blend of HTML and JavaScript is also a defining factor in making it simpler, as the library contains various functions that transform HTML components into the required functions.

Virtual Dom

React has the unique ability to tackle a common search engine failure to read JavaScript-heavy apps. As a solution, it can run on the server, rendering, and returning the virtual DOM to the browser as a regular webpage [20].

One-way Data Binding

React is using a unidirectional flow of data, something that forces developers to use the callback feature to edit components and prevents them from editing them directly. The JS component that ensures that controlling of data flow is being accomplished from a single point is called Flux. By utilizing it, developers achieve more flexible and effective apps that can be controlled a lot easier.

Helpful Developer Toolset

Facebook has included several developer tools for React and both Chrome and Firefox in the React.js framework, all of which aid developers immensely in observing component hierarchies, checking present state, discovering parent and child components, etc. The list of tools is constantly updated not only by the Facebook team but also by the community, so one must be constantly updated about new additions in the React toolset that could greatly reduce debugging time and ease the development process.

4.2.2 NextJs

In an attempt to improve the platform's performance and incorporate server-side rendering, NextJs was chosen. NextJs is a wrapper for the React framework, effectively expanding development capabilities with the most useful ones being the ability to render pages on the server and an improvement in the site's ranking in Google search engines. Pre-rendering is an incredibly useful feature as the HTML is generated at build time, so it can be reused on each request. More specifically, pages are cached by the CDN with no extra configuration to boost performance [21]. NextJs supports a unique file-based system based on the concept of pages, along with the ability to do client-side transitions between them, similar to how a SPA works. Interpolation can also be used for path creation when we want to achieve dynamic route segments. The latter feature proved to be very crucial in our application, as the need to have dynamic URLs in a multitude of API calls was evident. To use code over specific configurations, NextJS uses middlewares. The native Web API

Objects are extended to give more control over how one manipulates and configures a response based on incoming requests. The most common types of modifications that are dependent on these incoming requests are rewriting, redirecting, adding headers, or even streaming HTML.

4.2.3 Typescript

A very common bug that is seen in large JavaScript applications is that often a variable can be reassigned or coerced into a value of a different type with no issues or warning, thus leading to bugs that are often hard to locate. Typescript was created in 2012 to allow for optional static type checking, which would be particularly useful when developing large-scale applications. TypeScript will check types at compile time and throw an error if the variable is ever given a value of a different type. However, the error does not prevent the code from executing. It can be viewed as spellcheck that will inform the developer if something is problematic but won't alter how the code executes. By using Typescript, it was ensured that runtime errors and debugging time would be minimized.

4.2.4 Apollo

Apollo is an implementation of GraphQL that can transfer data between the server and the UI of an application [22]. Apollo's platform is made up of a combination of components that fall into three main categories:

- **Open-Source Components:** Apollo contains both server and client components. Server components define the schema that will be used and a group of resolvers, each with a specific role in schema implementation. Commercial plugins can be connected to any requests sent to the server, making it rather extensible. The client component is set up, so it can manage the application's data. It is very commonly used with JavaScript frontend frameworks such as React, Angular, Vue, etc.
- **Cloud Services:** The Apollo platform consists of a service that registers GraphQL schemas. It can register every client of the schema and the known operations that can be performed on it. Additionally, a pipeline exists along with a storage layer that takes in information about the GraphQL operations processed by the Apollo server.
- **Platform-Gateway:** Apollo's platform gateway is essentially a configuration of the server and the plugins. Smaller "schemas" refer to each other in a larger "master schema". When answering queries, the gateway builds a query plan, fetches the data from the GraphQL server, and combines everything into a single result.

Figure 4.2-4 displays these components in a clearer manner.

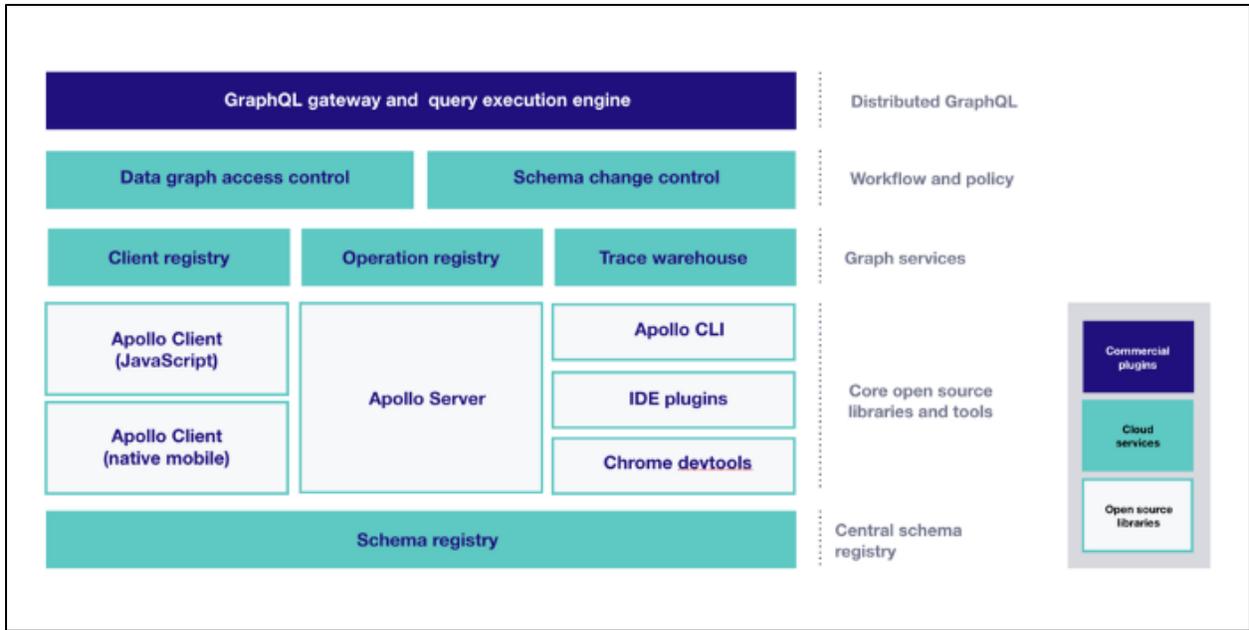


Figure 4.2-4 Apollo Major Components

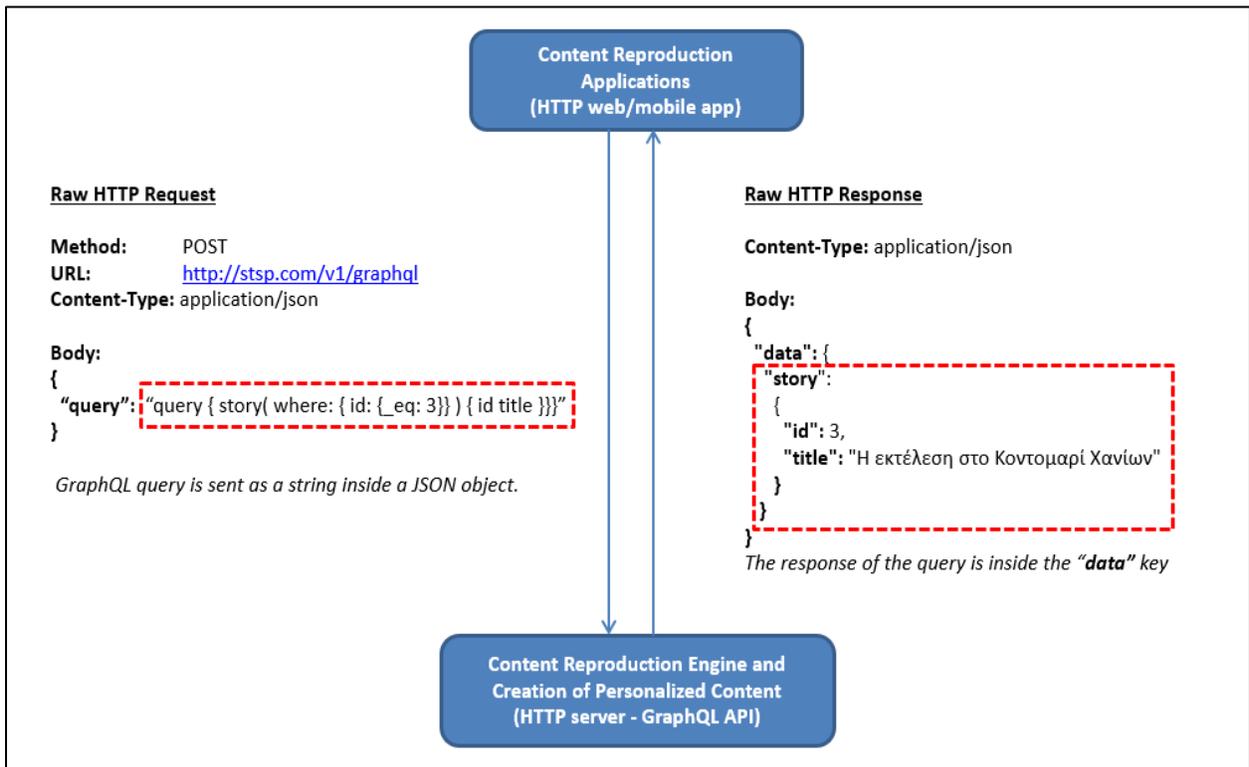


Figure 4.2-5 Apollo GraphQL Usage in the Architecture – Data Fetching Process

As figure 4.2-5 demonstrates, by sending a GraphQL request in the proper JSON form through the GraphQL API, the server can manage it and subsequently request the data from the database server. Finally, the requested data is sent back in a JSON object. At this point, the need for management of this data arises. The tool that is utilized to resolve that need is Apollo Client, a comprehensive state management library for JavaScript that enables the management of both local and remote data with GraphQL. Apollo's unique ability to cache queries and store query results saves computational resources and automatically updates UI components upon query result change or arrival. These characteristics of Apollo dramatically improve performance and make it a definite choice for dynamic content-serving platforms, which require the fetching of page data for server-side rendered pages.

As has already been mentioned, a major goal of the platform was to incorporate user interaction within each unique story. To manage and store the data that is created by the users, additional rendering is being done, but this time it's on the client side by utilizing Apollo Client. In this way, server-side rendering is combined with client-side rendering to make sure that each page that is pre-rendered on the server is updated in real-time (GraphQL subscription) after each user's contribution to a story. This is a very crucial mechanism of the architecture as it guarantees that each page of the platform always has updated data. By having updated data in each user interface, we can support commenting, rating, reacting, bookmarking, and accurate story recommendations to authenticated users of the platform.

Figure 4.2-6 is a brief depiction of the above-mentioned mechanism, which, as we have already discussed, ensures that pages are being pre-rendered at the server and, additionally, are being enriched by user-contributed elements in real time.

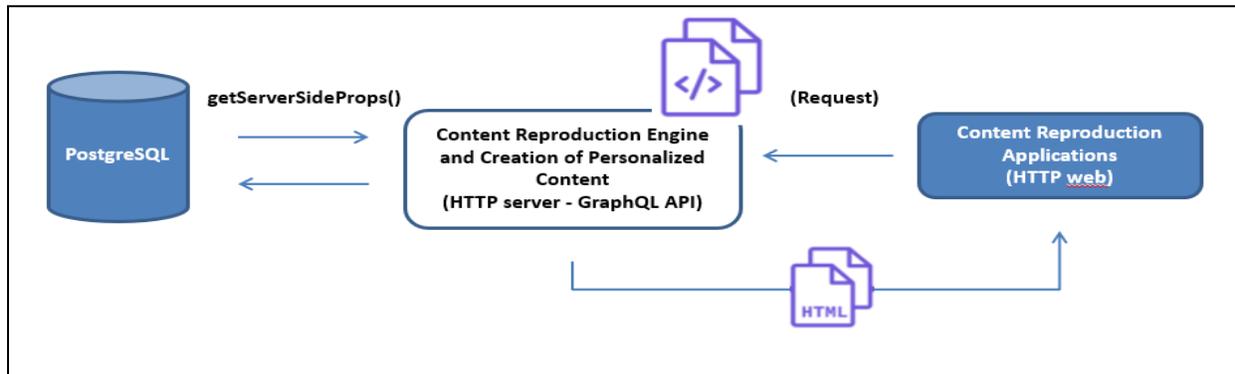


Figure 4.2-6 Usage of Apollo Client in managing Server-Side Rendered Page Data in the Web Application

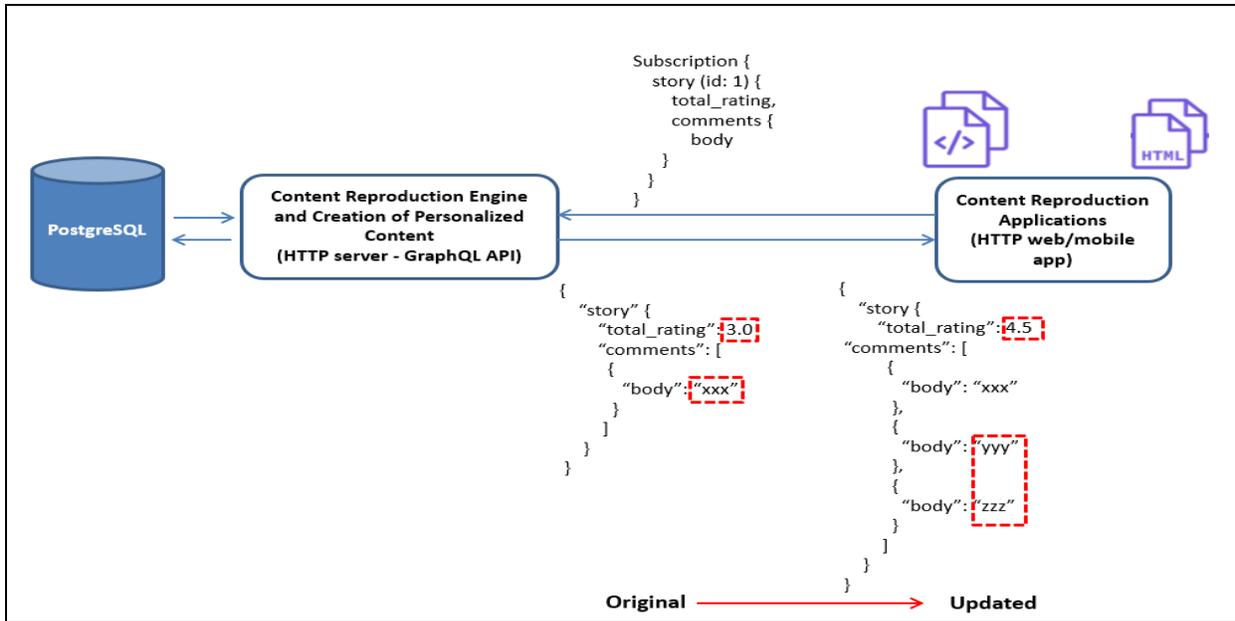


Figure 4.2-7 Apollo Client's Usage for Managing Client-Side Data- Example of GraphQL Subscription for Fetching Data in Real Time

4.2.5 Hasura

Hasura is a real-time GraphQL API engine that makes data from databases instantly accessible over a real-time GraphQL API, offering developers a fast and reliable way to build APIs and ship apps. For Hasura to work, it must be connected to whatever databases, Rest servers, GraphQL servers, or third-party APIs we want to have a unified, instantaneous, real-time GraphQL API. In our platform, it related to the Postgres database tables, all of which (along with their relationships) were subsequently reflected directly in the structure [23].

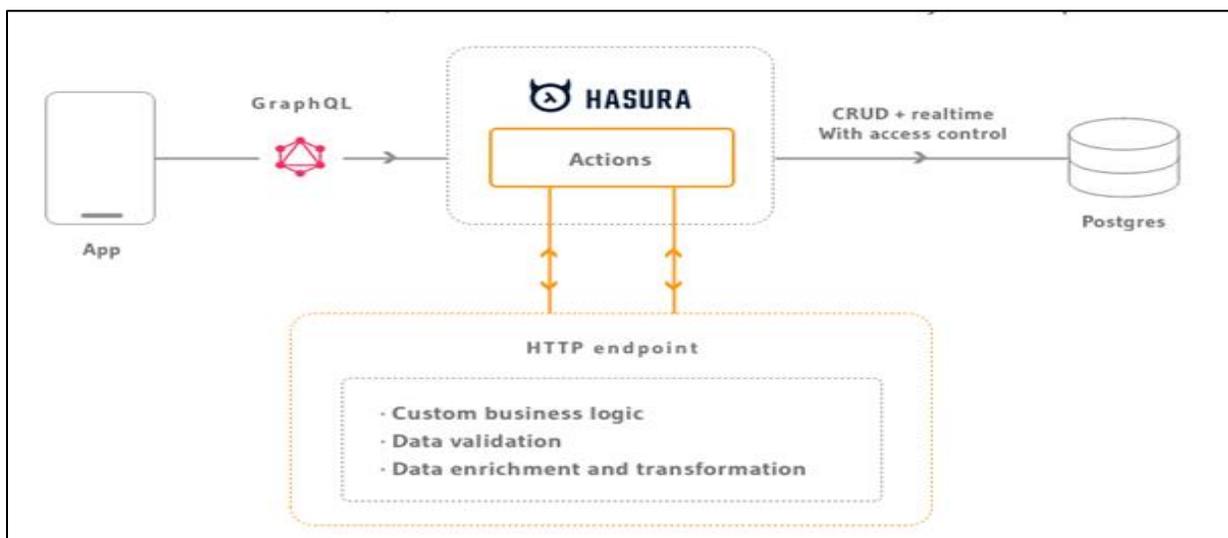


Figure 4.2-8 Hasura Actions Model Flow Diagram

In our application, Hasura was used to create a unified data access layer. By connecting Hasura to the database, we achieved real-time access to all our data sources without having to write any backend code. Actions are a way to extend the Hasura schema with custom business logic using custom queries and mutations. Actions can be added to Hasura to handle various use cases such as data validation, data enrichment from external sources, and any other complex business logic. There are two main types of actions used in our application :

- **Query Action:** After querying some data from an external API or doing some validations or transformations before sending back the data, a Query Action can be used.
- **Mutation Action:** To perform data validations or do some custom logic before manipulating the database, a Mutation Action can be used.

Along with Hasura Actions, it was deemed necessary to also utilize Hasura Event Triggers. Event triggers reliably capture events on specified tables of our database and invoke webhooks to carry out our custom logic. After a mutation operation, triggers can call a webhook asynchronously [24].

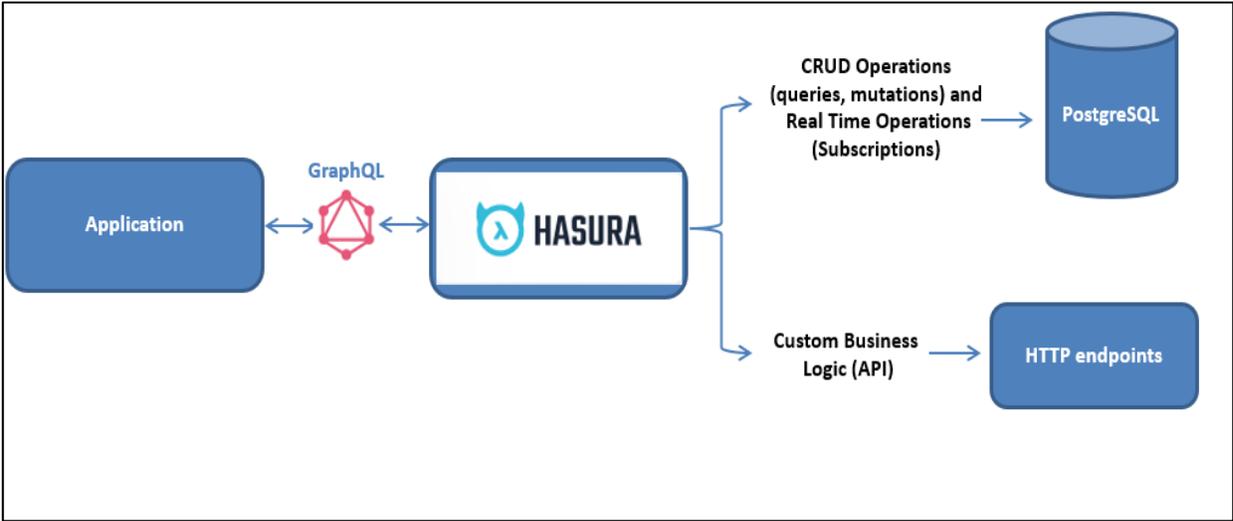


Figure 4.2-9 Integrating Hasura in an Application Environment to both add Custom Business Logic and Manage Database

4.2.6 PostgreSQL

PostgreSQL is an open-source object-relational database system that came out in 1986 and was created as a part of a semester project for the University of California at Berkeley. It extends the SQL language and offers features that safely store and scale complex data workloads, while simultaneously ensuring that the database remains secure with tools such as TDE and data masking. Being

an open-source project, PostgreSQL allows companies to use, modify, and implement it as per business needs, thus ensuring a large degree of freedom during the development process. The constant contribution of the community has been driving innovation for the past 25 years to such a degree that PostgreSQL currently supports a multitude of extensions not only for SQL data models but also for NoSQL.

The choice to use PostgreSQL in our project was made as it was the best-fitting option for our type of data. The joint clause was a major benefit as it enabled us to retrieve all the necessary data from the chosen tables using SQL mechanisms. If we chose a NoSQL database with JSON-style data, such as MongoDB, we would add unnecessary complexity to our requests [25].

4.2.7 OAuth2

The OAuth2 authorization framework is a protocol that allows a user to grant a third-party website or application access to the user's protected resources without necessarily revealing their credentials or even their identity. OAuth introduces an authorization layer and separates the role of the client from that of the resource owner. In OAuth, the client requests access to resources controlled by the resource owner and hosted by the resource server and is issued a different set of credentials than those of the resource owner. Instead of using the resource owner's credentials to access protected resources, the client obtains an access token—a string denoting a specific scope, lifetime, and other access attributes. Access tokens are issued to third-party clients by an authorization server with the approval of the resource owner. Then the client uses the access token to access the protected resources hosted by the resource server [26]. The previously mentioned process is viewed in figure 4.2-10.

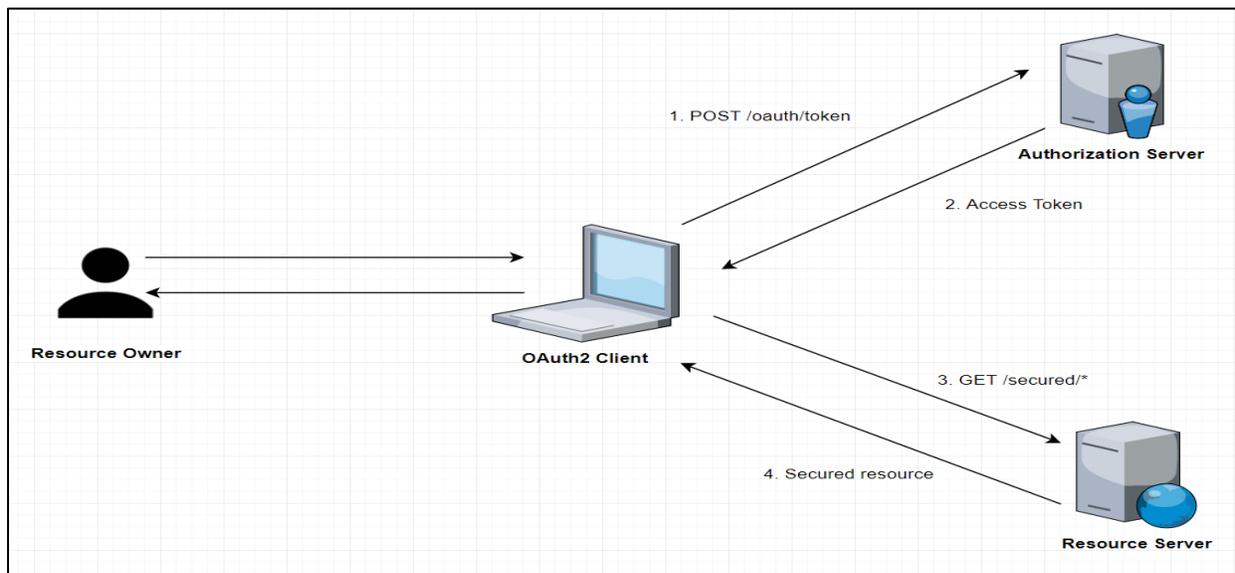


Figure 4.2-10 OAuth2 Implementation Diagram

In our application, an OAuth2 protocol was selected with JWT as bearer tokens as it allows us to encode all relevant parts of an access token into the access token itself instead of having to store them in a database. JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way of securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed [27].

Our GraphQL engine is configured to use JWT authorization mode to authorize all incoming requests to the Hasura GraphQL engine server. The idea is that the authorization server will return JWT tokens, which are decoded and verified by the GraphQL engine, to authorize and get metadata about the request. The JWT is decoded, the signature is verified, and then it is asserted that the requested role of the user (if specified in the request) is in the list of allowed roles. If the desired role is not specified in the request, then the default role (public) is applied. If the authorization passes, then all the *x-Hasura-** values in the claim are used for the permission system. Figure 4.2-11 illustrates this mechanism in a flow diagram which showcases its use in the architecture.

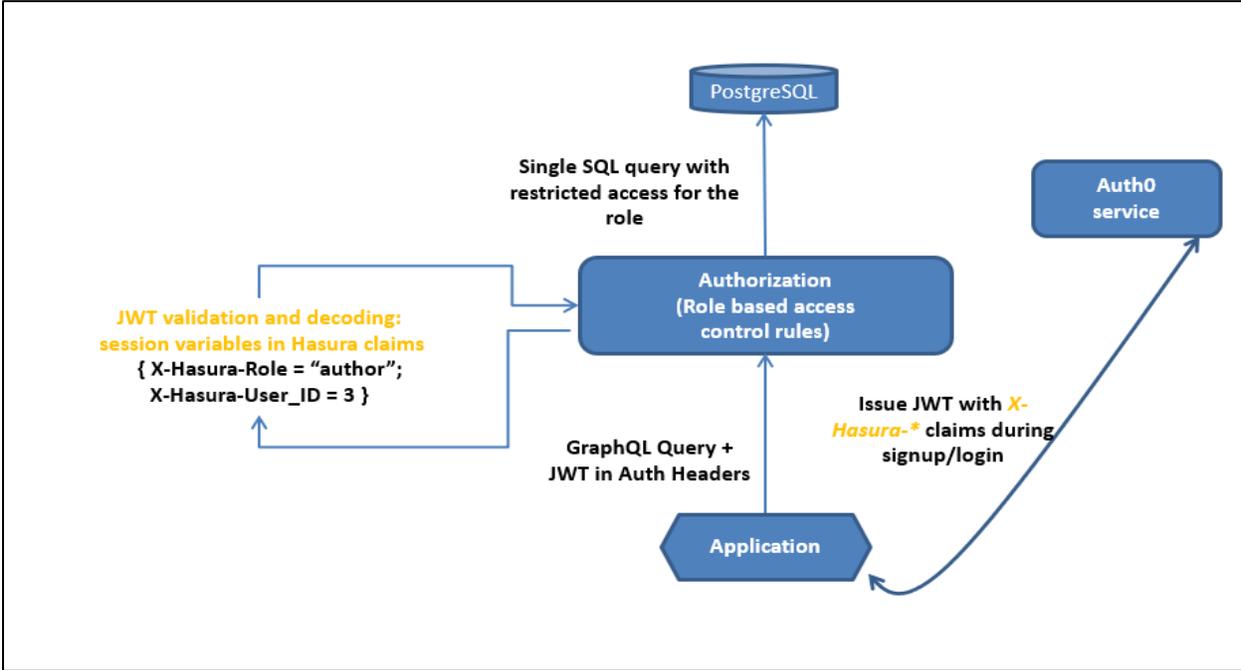


Figure 4.2-11 JWT-Based Authentication and Authorization System

4.2.8 Continuous Integration

Continuous integration (CI) is the practice of automating the integration of code changes from multiple developers into a single software project. It allows developers to frequently merge code changes into a central repository, where builds and tests are then run. Automated tools are used to assert the new code's correctness before integration.

A source code version control system is the crux of the CI process. The version control system is also supplemented with other checks like automated code quality tests, syntax style review tools, and more. CI is intended to be used in combination with automated unit tests written through the practices of test-driven development. This is done by running and passing all unit tests on the developer's local computer before committing to the main project. This helps avoid one developer's work-in-progress breaking another developer's copy. It was a major tool in the development of the framework as it allowed for tracking of code changes and reverting to a previous version of the code when it was deemed necessary.

So far, even though the analysis of the framework architecture and its implementation technologies has been completed, there has not been a lot of discussion on how the framework was applied to the platform except for some sporadic references in the initial chapters. Chapter 5 will dive deeper into this matter and complete the description of the interconnections between the framework and the platform.

5. Framework Application

The chapter describes the necessary elements that were utilized from the STSP platform to make the implementation of the framework feasible. There will be an analysis of what was created from a social media standpoint, what parts of the data model were utilized, the interventions in the GUI and lastly how item-based collaborative filtering was applied in the platform.

5.1 Approach

For the platform to be able to integrate the framework and successfully utilize its features, there were a series of tasks that had to be fulfilled. Firstly, in the context of communicating with social media and more specifically with Facebook, a dedicated Facebook page was created that would serve as a way of displaying the platform's content and an environment that would encourage user interaction. Every story that was published on the platform was subsequently transferred to the official Facebook page in a dedicated post along with its dedicated story elements such as images, text, audio, etc.

Comments and ratings coming from the Facebook page are being utilized along with ratings, bookmarks, and reactions that are made in the native STSP platform as input in the recommendation engine. Using both types of reactions and ratings, the recommendation engine produces a story suggestion which is displayed on the home page of STSP along with other similar stories. Additionally, elements that come from social media, such as pictures and videos, are stored in the STSP storage component and could become potential story elements in the form of user-contributed elements.

For real-time comment fetching, the subscription mechanism of the STSP web client that was mentioned in the previous chapter is being utilized to show real-time comments that are being made on the STSP platform as well as replies to them, but in chronological order.

5.2 GUI

This chapter will present some of the basic user interfaces of the platform and the elements of them that were utilized in the development of the framework. The main design pattern that was followed was the Atomic Design pattern [28]. The goal was to make digital interaction as simple, fluid, intuitive, and efficient as possible. By utilizing Jacob's Nielsen 10 usability heuristics, it was ensured that the whole user interface design was held up to a high business standard [29]. It should be noted that slight changes may have occurred since the writing of this chapter, as the development process of the platform is ongoing.

Figures 5.2-1 and 5.2-2 present the home page, in which the users can perform a series of tasks that are interconnected to the framework's features such as :

- Share a story on their personal Facebook page
- Explore their recommended story for the day
- Check their bookmarked stories
- Check their liked stories
- Check the stories that are related to their preferences

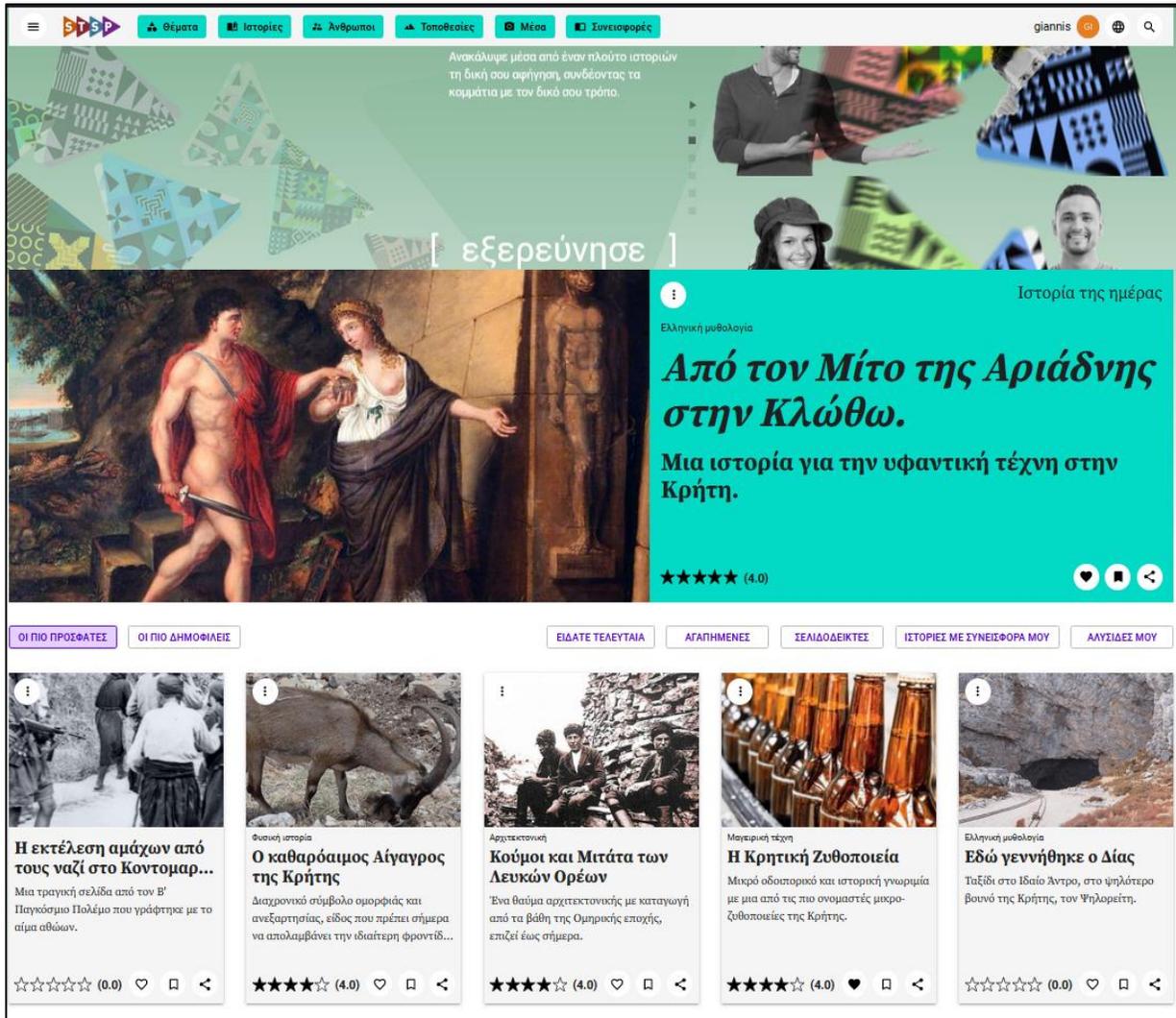


Figure 5.2-1 STSP Home Page-User Recommended Story

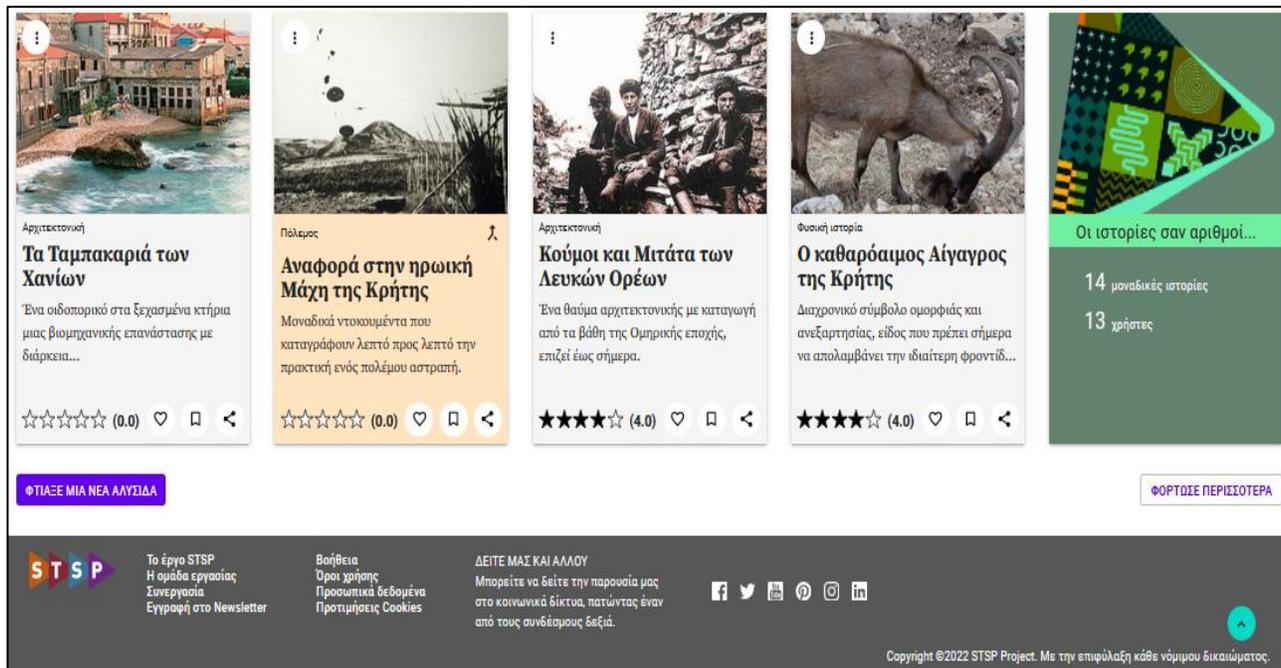


Figure 5.2-2 Home Page of STSP- Liked, Recently Viewed and Bookmarked Stories, Recommended Story, Popular Stories, Average Rating of Stories

Figures 5.2-3 and 5.2-4 present the details of the story of the platform. They contain important information related to each historical event that is part of a story, user-contributed elements, multimedia elements, related stories, and an interactive comment section. The subscription mechanism of the STSP client is utilized here to ensure new comments are being shown in real-time. Overall users can :

- Comment
- Like
- Rate
- Bookmark
- Share
- Contribute to the story

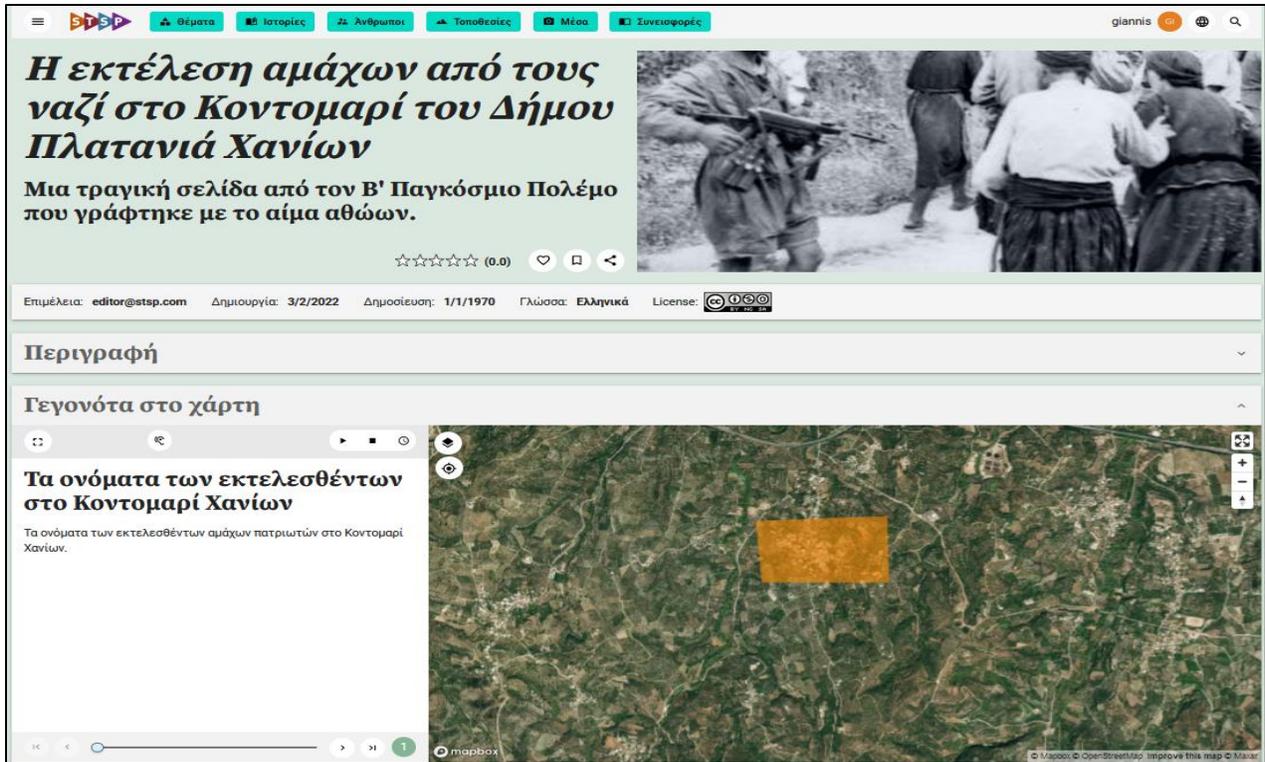


Figure 5.2-3 Rating, Share, Like, Bookmark of a Story

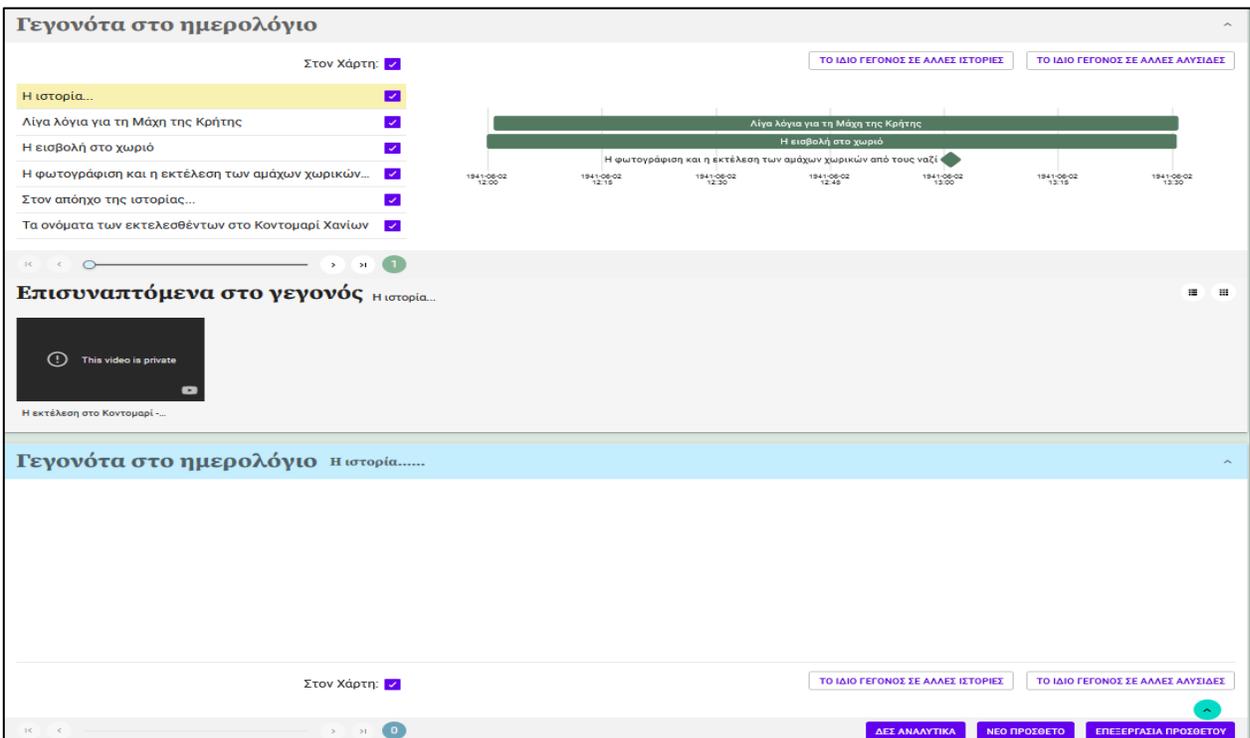


Figure 5.2-4 Exploring of Story Elements -User Contributed Elements

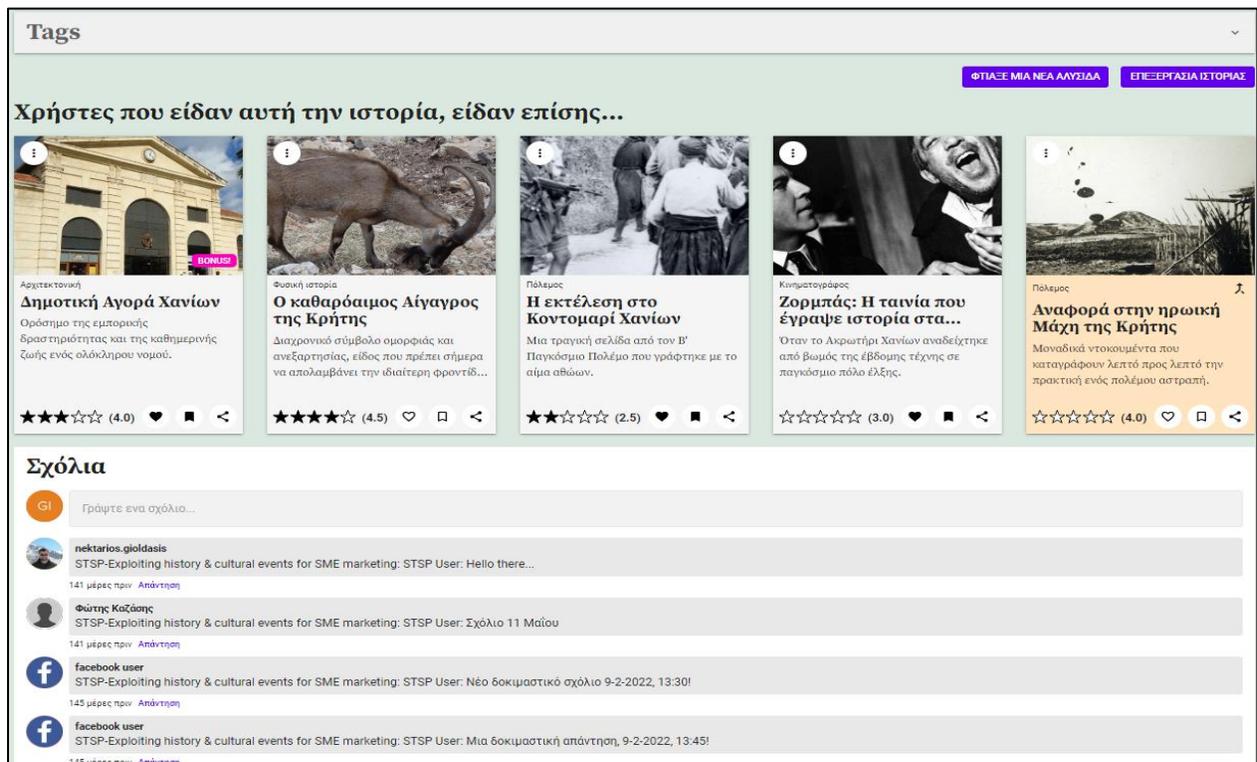


Figure 5.2-5 Story Comment Section-Display of Facebook Comments

Figure 5.2-6 presents the profile page where users can edit personal information. Additionally, they can see stories that they have interacted with in the past, such as favorite stories, recently viewed stories or stories that have been rated. They can change any type of action they have performed in a story(bookmark, like, rating, etc.), something that will influence the recommendation engine when the process for a new suggestion of a story arrives.

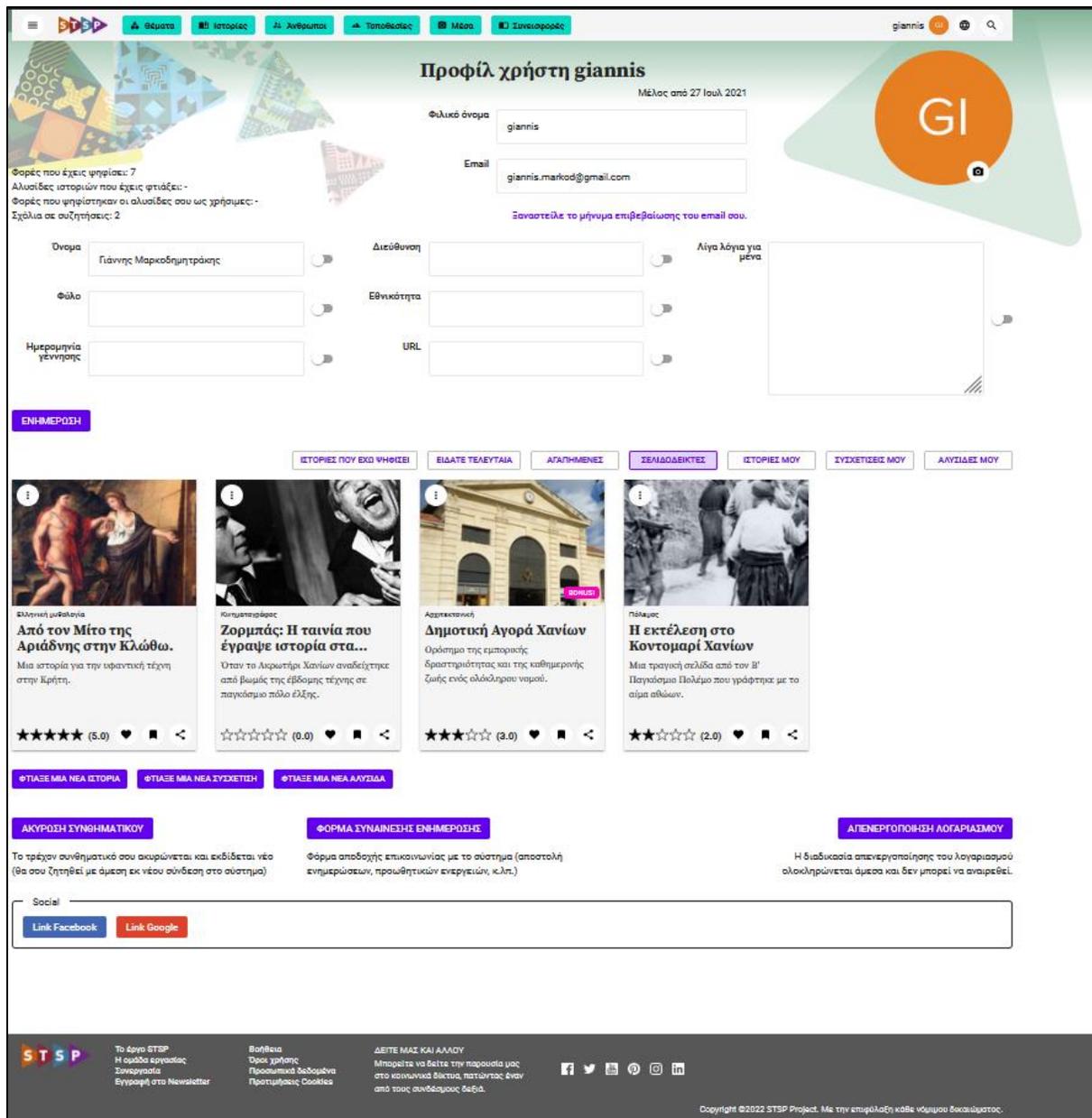


Figure 5.2-6 User Profile Page- Recently Viewed, Bookmarked, Liked, Commented Stories of the User

5.3 Applying Personalization to STSP

It has already been discussed that the recommendation engine of the platform works by implementing an item-based collaborative filtering algorithm. In this subchapter, there will be a more detailed analysis of the process and a reference to the STSP components that are being utilized. As it was mentioned previously, both STSP comments from the native platform and comments from social media are being stored in the database in specific tables. Apart from them, the framework also

ensures that ratings, bookmarks, and reactions coming from both sources are stored. All this data is being stored in the 5 major tables which are named: *sm_comment*, *sm_comment_react*, *sm_post*, *sm_post_react*, *story_user_rater*. Immediately after the storage of a new type of rating or comment, a series of Hasura triggers are activated, which are responsible for producing a total rating, which refers to the opinion of a user about a story. The total rating final number is calculated by considering a series of factors such as whether the user has bookmarked/rated/liked/commented on a story as well as the average rating of a story, and it is subsequently stored in a new table, named *story_user_rater*, which will be used to compute the required similarities.

Moving forward, after the similarities have been calculated, they are stored in the *story_story_similarity* table of the database, with the next step being their utilization for computing the potential rating for each unexplored story for the user. The highest-rated story becomes the recommendation for the user that is displayed on the home page of the STSP platform and is stored in the table named *user_story_suggestion*.

The final step is to check the user's rated stories at regular intervals using Cron Triggers to ensure that the recommended story that he views on the main page hasn't been rated because, in the event of that, there is a need for the story to be removed from the *user_story_suggestion* matrix and the recommendation algorithm to run again and suggest a new story [30].

6. Conclusions – Future Work

In this section, there is going to be a brief overview of the conclusions of this thesis and some suggestions for future modifications that will improve the system's functionality.

6.1 Conclusions

The main goal of this thesis was to design a framework for transmedia storytelling platforms that would aid the utilization of social media. The uniqueness lies in the fact that to create and manage the components that would execute the desired functionality, there was a necessity for learning several new technologies and frameworks, as well as testing libraries and getting familiar with continuous integration. In particular, the following conclusions have been drawn from this process are:

- Storytelling is a very crucial aspect of modern web applications that strive to integrate user engagement. Having a well-designed and executed storytelling strategy will help captivate the user and will influence him to interact with the platform and potentially buy products and services. More and more storytelling tools are being created every day, meaning that it is very important to pick the right one to create memorable emotions for the user.
- Being part of a professional project in development requires methodical work and time management. The standards for the code are different and more emphasis is given on whether something works efficiently rather than just being functional.
- Building a secure application should be the number one goal for every modern web developer. Optimal web application security starts in the design phase and continues well after the web application release, therefore, it is very crucial to take an agile approach. Role management is vital as the system should only permit the right people to access the right resources at the right time and for the right reasons.
- The microservices architecture fits very well into the agile development model. Each developer can own and focus on one service, thus working more effectively and efficiently as the potential conflicts with other developers are eliminated. Since microservices are loosely coupled, developers can add new features and upgrades without having to upgrade the entire system. In addition, the microservice architecture can allow for various modifications to be implemented. Developers can add new frameworks, data sources, and lists without having to undergo a system-wide reconfiguration.
- Social media integration is crucial for platforms that wish to expand their audience. In our case, the Graph API (Facebook API) was the way to get data into and out of the Facebook platform. Overall, the implementation of

Graph's API was hard as it is a large and complex API with documentation and examples written in PHP and CURL. Additionally, the Facebook approval process for scope permissions is the most rigorous of any network. One needs to write up testing steps, submit a video, and detail the reasoning for needing the permissions. Rejection is almost guaranteed, even for the slightest missing step in the video process. Also, every new permission requires starting the process over again. Having all these in mind, the application currently operates with test users and hasn't been approved as the development process is continuing, thus meaning that were we to ask for Facebook's approval to go live, the permissions would be revoked after each new feature that was added.

6.2 Future Work

Below are two major aspects of the system that could be improved in the future to ensure a better recommendation system and the integration of additional social media APIs to gather user attention from multiple platforms.

- Since at the moment it is difficult to assess the platform's traffic rate, it is crucial to note that the proposed item-based collaborative filtering algorithm for the system may not work as intended in the beginning, mainly because of the initially low number of users. This is known as the "new community problem," and it refers to the startup of the system when virtually no information the recommender can rely upon is present [31]. In our case, in the beginning, only a few interactions are available, and even though the collaborative algorithm will produce some recommendations, the quality of those recommendations will be poor. To resolve this, the system could potentially force the user to provide some personal data for their preferences in the sense that the user must dedicate an amount of effort to using the system in its "dumb" state—contributing to the construction of their user profile—before the system can start providing any intelligent recommendations. This technique is called preference elicitation, and it can be implemented either explicitly (by querying the user) or implicitly (by observing the user's behavior).
- The database model is currently able to support the integration of data coming mainly from Facebook. It would be beneficial to support data exchange between other social media networks as it would attract a large number of users that could potentially interact with the platform and participate in the upcoming advertising campaigns.

A detailed study of each API of these networks as well as changes in the model of the database is required to incorporate these different APIs. Brand promotion should include a social media API strategy to reap maximum benefits from the social media influence on consumers.

7. References

- [1] K. T. A. a. P. H. CHUA, "Digital Storytelling as an Interactive Digital Media Context," 2010. [Online]. Available: <https://idnarrative.pressbooks.com/chapter/digital-storytelling-as-an-interactive-digital-media-context/>.
- [2] Microsoft, "Microsoft Timeline Storyteller," Microsoft, 2017. [Online]. Available: <https://timelinestoryteller.com/>.
- [3] Pageflow., Codevice Solutions, 2022. [Online]. Available: <https://www.pageflow.io/en/>.
- [4] K. lab, "StoryMapJS," 2022. [Online]. Available: <http://orangeline.knightlab.com/templates/pages/storymap.html>.
- [5] Europeana, Europeana Foundation, 2017. [Online]. Available: <https://pro.europeana.eu/data/digital-storytelling-prototype>. [Accessed 2022].
- [6] Y. F. O. F.O. Isinkaye, "Recommendation systems: Principles, methods, and evaluation," Egyptian Informatics Journal, 2015..
- [7] G. Belani, "6 Examples of Effective Social Media Integration," Quintly, 2021. [Online]. Available: <https://www.quintly.com/blog/social-media-integration>.
- [8] Mashable, 2022. [Online]. Available: <https://mashable.com/>.
- [9] M. inc., "Smash Ballon," 2022. [Online]. Available: <https://smashballoon.com/?smashid=1635>.
- [10] Rafflepress, 2022. [Online]. Available: <https://rafflepress.com/about/>.
- [11] Disqus, "Disqus," [Online]. Available: <https://disqus.com/>. [Accessed 2022].
- [12] S. Suranga, "BetterProgramming," July 2020. [Online]. Available: <https://betterprogramming.pub/10-must-know-concepts-of-modern-web-architecture-9ecbefef8bc>.
- [13] K. Eaton, Benton Institute, March 2012. [Online]. Available: <https://www.benton.org/headlines/how-one-second-could-cost-amazon-16-billion-sales>.
- [14] HUSPI, "HUSPI BLOG," 2019. [Online]. Available: <https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications/>.
- [15] A. Sikandaar, "Mobile Live," 2020. [Online]. Available: <https://www.mobilelive.ca/blog/graphql-vs-rest-what-you-didnt-know/>.
- [16] M. Developers, "MDN Web Docs," 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Web_Components.

- [17] M. Dabbagh, "Functional and non-functional requirements prioritization: An empirical evaluation of IPA, AHP-based, and HAM-based approaches," 2015.
- [18] J. P. M. Jiawei Han, "ScienceDirect," 2021. [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/cosine-similarity>.
- [19] G. K. J. K. a. J. R. Badrul Sarwar, "Item-Based Collaborative Filtering Recommendation," University of Minnesota, 2001.
- [20] S. Stefanov, "React: Up & Running: Building Web Applications," 2019.
- [21] Vercel, "NextJs," 2020. [Online]. Available: <https://nextjs.org/>.
- [22] R. S, "Bits and Pieces," February 2019. [Online]. Available: <https://blog.bitsrc.io/should-i-use-apollo-for-graphql-936129de72fe>.
- [23] S. Kurzynowski, "The Software House," July 2020. [Online]. Available: <https://tsh.io/blog/hasura-instant-realtime-graphql-apis/>.
- [24] H. Inc., "https://hasura.io," June 2021. [Online]. Available: <https://hasura.io/learn/graphql/hasura/custom-business-logic/>.
- [25] M. Borozenets, "PostgreSQL for the next project," FULCRUM, [Online]. Available: <https://fulcrum.rocks/blog/why-use-postgresql-database/>. [Accessed 2021].
- [26] T. McKinnon, "auth0," auth0, 2022. [Online]. Available: <https://auth0.com/docs/authenticate/protocols/oauth>.
- [27] A. D. Team, "JWT.IO," 2022. [Online]. Available: <https://jwt.io/introduction>.
- [28] B. Frost, "Atomic Design Methodology," 2016. [Online]. Available: <https://atomicdesign.bradfrost.com/chapter-2/>.
- [29] J. Nielsen, "10 Usability Heuristics for User Interface Design," April 1994. [Online]. Available: <https://www.nngroup.com/articles/ten-usability-heuristics/>.
- [30] H. Inc., "Hasura.io," 2021. [Online]. Available: <https://hasura.io/docs/latest/scheduled-triggers/create-cron-trigger/>.
- [31] G. K. R. Al Mamunur Rashid, "Learning preferences of new users in recommender systems: an information theoretic approach," December 2008.