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

Participatory Design Web Application

Raphael-Demetrios Tsirivakos

Thesis Committee
Professor Michail Zervakis (ECE)
Professor Michail G. Lagoudakis (ECE)
Professor Konstantinos - Alketas Oungrinis (ARCH)

Chania, July 2024

Abstract

Years of research in the field of participatory design highlight the importance of its role in future model cities. Recent studies with a dedicated focus on enhancing the practical application of this design model in urban planning and architecture by leveraging modern methods, turn their interest on 3D web technology. There are valid reasons for this since, utilizing its capabilities can provide a highly accessible environment, suitable to create and share 3D visualizations, enabling ideas exchange with ease. The seamless usability of participatory tools is crucial for users to effortlessly articulate their creative concepts, allowing a more extensive and diverse engagement in the design process. This approach, fosters collaboration beyond the confines of in-person workshops. Web browsers are quite familiar to most people nowadays so, an expanded participation is a natural consequence and constitutes a key point for an effective participatory design. Nevertheless, practical approaches implemented thus far, arguably offer potential for further enhancement in order to facilitate the process of a design model, where different perspectives can come together and co-shape an architectural design. This thesis introduces a web-based information system developed using the Node JS javascript runtime environment, aiming to connect stakeholders of participatory design through a network composed of three main pillars. The first one, is the build of a community, where design proposals can be shared and assessed. The intention is to broaden perspectives by facilitating collaborative design, but also to enhance user's interest in the process. By witnessing how non-experts actively engage, the community becomes a source of inspiration, fostering a dynamic environment for those interested in their projects. The second, is to spur end-users into more active involvement throughout the architectural design process. For this reason, the system provides an 'easy-to-use' 3D editor, where users without being professionals, can unfold their ideas and ensure a higher level of comprehension between them and the designers. The third pillar of the network composition, aspires to provide a more direct and efficient cooperation among members of a team, where they can design together in real-time their projects. This feature leads to a more interactive design process. Consider a team working on a project that has been decomposed in discrete segments; everyone can work on their part of interest, while being able to see the progress of other members live, and provide feedback, since they work in the same digital space simultaneously. The proposed system is designed to enhance participatory design within urban planning and architecture, improving collaboration conditions and providing a space for this design model where it can be expressed.

Contents

List of Figures

Chapter 1

Introduction

1.1 Key Concept

Participatory design, could play a pivotal role in the sustainability of modern societies by embracing diverse perspectives and crafting inclusive solutions within a democratic framework [1]. This approach, employing the appropriate methodologies cultivated over time as its tools, ensures that the developed solutions consider various viewpoints, acknowledging the diverse needs and experiences within communities. By doing so, it fosters a feeling of ownership among stakeholders, ultimately resulting in more effective and sustainable solutions that are embraced by the community and reflecting end-users needs and expectations. [2].

Participatory design model has a versatile nature that emerges collaborative approaches and innovative solutions across a multitude of sectors, amplifying its impact throughout the years:

1. **Urban Planning and Architecture :** Inclusive urban planning involving communities, results in cities that meet resident's needs, creating sustainable, livable spaces, through collaborative design of public areas and infrastructure.

2. **Healthcare and Wellness :** Engaging patients, caregivers and healthcare professionals in participatory design enhances medical facilities, technologies, and services, ensuring patient-centric care and operational efficiency.

3. **Education and Learning Environments :** Empowering students and educators through participatory design, creates engaging and effective learning environments, allowing co-creation of curriculum materials, classrooms and educational tools.

4. **Technology and Product Development :** From software interfaces to physical products, involving end-users in design processes ensures user-friendly, innovative solutions, that align with actual needs and preferences.

5. **Community Development and Social Services :** Participatory design, drives community empowerment by involving residents in the design of public amenities, and social services, creating solutions tailored to local needs.

6. **Environmental Sustainability :** Engaging stakeholders in the design of green spaces, conservation efforts, and renewable energy projects fosters effective and inclusive environmental initiatives.

7. **Inclusive Governance and Policy-making :** Governments use participatory design to involve citizens in policy making, resulting in responsive policies and services that cater to diverse populations.

8. **Transportation and Mobility :** Designing efficient and inclusive transportation systems.

9. **Entertainment and Media :** Creating engaging and user-centric entertainment products.

Participatory design's widespread application across these sectors, underscores its role in fostering collaboration, incorporation and innovation, ultimately leading to outcomes that align with communities and users needs.

This thesis, proposes a modern web-based information system, as a digital gathering place for participatory design processes within urban planning and architecture. 3D web capabilities, can assist us piece together the puzzle and create an engaging environment that accommodates design ideas and encourages involvement from all skill levels.

1.2 Brief Description

The swift evolution of 3D Web technologies, has enabled the development of innovative tools driven by interactive experiences. Participatory design processes, are progressively incorporating 3D web technologies owing to their efficacy in simulating scenarios. This progression offers an opportunity for participatory design, providing a dynamic platform for collaborative creation and exploration. As these technologies advance, their potential to elevate participatory design methodologies becomes increasingly evident, through immersive and interactive environments that enhance collaboration and creativity.

The incorporation of Three.js technology [3], exemplifies the burgeoning capabilities in 3D web technologies. Its seamless integration and robust rendering strength, has elevated the development of interactive experiences across diverse applications. Three.js enables dynamic and immersive environments on the web, revolutionizing how users interact with the browsers. A powerful rendering engine, facilitates the creation and manipulation of detailed 3D models, offering both realism and interactivity. Three.js's adaptability across various platforms and devices, establishes its position as a democratizing force in the 3D web landscape. Its accessibility, ensures that users can seamlessly interact with rich 3D content, regardless of their device's specifications or location, fostering a more inclusive and engaging online environment.

This thesis, presents the implementation of a web application with the use of Node.js in conjunction with Three.js. This selection, stems on a comprehensive evaluation, focused on creating a dynamic and engaging web experience.(Figure 1.1)

Web-based approach eliminates operating system's limitations, making it easier for users to engage. This aligns with our goal of broadening users involvement, a target that Node.js can effectively support. Combining Node.js with Three.js, ensures a responsive program for a large user base, without compromising quality, enabling an engaging web platform that is accessible across operating systems, encouraging broad user participation and interaction.

Leveraging web real-time features to enhance user's experience, the platform enables team collaboration in real-time. This way, the system fosters collaboration beyond the usual standards of design applications, innovating the way real-time communication features can be applied to 3D design processes.

The information system's Full-Stack development, is powered by a **MEAN**-like or **MERN**-like stack, which encompasses **M**ongoDB, **E**xpress.js and **N**ode.js, with EJS (Embedded JavaScript) as a template engine. System's front-end was implemented utilizing vanilla JavScript. This combination of technologies, forms the backbone of the application's implementation, covering both front-end and back-end aspects. MongoDB serves as a database, while Express.js facilitates server-side operations. EJS as a templating engine, simplifies dynamic content generation and server-side rendering as Node.js powers the entire server environment. Collectively, these technologies create a unified environment, enabling us to build a dynamic and efficient information system.

This thesis aims to provide a substantial contribution to the research needed, for an efficient application of participatory design model on urban planning and architecture, that facilitates the engagement through digital platforms and methodologies. The proposed system, aspires to constitute an environment capable of accommodating diverse ideas, unifying different viewpoints of stakeholder groups and distribute them among a wider audience.

Specifically, this practical approach includes:

- **Account Creation**: Stakeholders, are able to sign up and create personalized accounts on the platform, accessing its full suite of features and functionalities.
- **Onboarding Tutorial**: Registering on the platform, users undergo a brief tutorial to familiarize themselves with its features, ensuring that they can effectively utilize its tools from their first interaction.
- **3D Environment Creation**: Users navigate an intuitive interface, which enables the creation of 3D environments and facilitates creative expression in all skill levels.
- **File Import and Export**: The platform supports the import of various 3D file formats, allowing users to enrich their designs. Additionally, projects can be exported for the purpose of archiving or sharing outside the program.

- **Project Management**: Users have access to storage capabilities, ensuring easy access and organization of their design portfolio.
- **Collaborative Workspaces**: Teams can be formed, allowing members to join dedicated rooms for real-time collaborative design sessions.
- **Community Interaction**: The application fosters community engagement by allowing users to interact and comment, providing feedback on shared designs, through a page that serves as an inspiration hub where multiple projects are showcased for discussion and idea exchange.

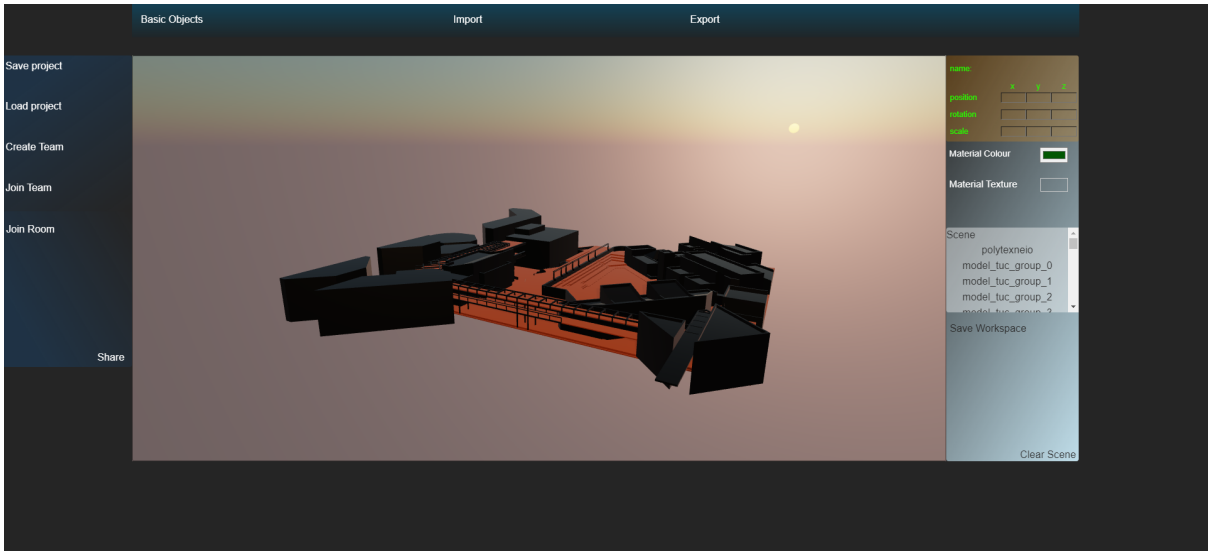

Figure 1.1: Application's 3D Editor interface

1.3 Structure of the Thesis

The structure is divided into chapters where:

Chapter 1 introduces the key concept for the object of this thesis and presents a brief description of the project's purpose.

Chapter 2 analyzes the methodology, guiding the decisions on the information system's structure and presents the network protocols that enabled the desired communication.

Chapter 3 presents the user-centric requirements that the system must fulfill, along with the use case scenarios and the system's manual.

Chapter 4 describes step by step the implementation process for both the Front-end and Back-end of the system.

Chapter 5 explores potential future expansions of the system, while also discussing its deployment process observations and evaluating its strengths and weaknesses.

Chapter 2

Research Overview

2.1 Introduction

This chapter provides the analysis for understanding the methods used to develop and implement the information system described in this thesis.

The implementation of this project requires a specific combination of technologies in order to be able to function the way it was envisioned.

2.2 Development Blueprint

Main technologies and libraries combined to achieve the desired result.

2.2.1 Full-Stack Development

The methodology adopted in this thesis, embodies a deliberate approach aimed at maximizing performance, elevating user experience and ensuring scalability. Each component of our chosen framework, was carefully selected to match the project needs, ensuring its efficiency.

The platform was implemented as a web application (Figure 3.1). This way, any device connected to a browser can access our application without needing a separate app for each operating system, as would be required otherwise to broaden accessibility. It

is cheaper and easier to maintain; if there's an error, fixing it across different platforms would be costly. Instead of fixing each user's issue, upgrades are applied to the server and users receive the updated version the next time they log in. Each operating system uses a different programming language. A web application, simplifies things and has significant cost, time and maintenance benefits.

Figure 2.1: Web Application Data Flow

Subsequently, we examined its development methods. The technology choice was based on the desired characteristics to be fulfilled:

- **High performance**, aiming to satisfy users.
- **Real-time capabilities**, for an efficient collaboration tool.
- **High concurrency**, to serve a large number of users simultaneously.
- **Scalability**, capable of handling increasing loads, users, data, or other resources without significant performance degradation.
- **Extensibility**, allowing easy addition of new features or functionalities.
- **Easy to learn**, enabling easy integration of new team members in its implementation, for a future expansion.

The above criteria led us to Node.js JavaScript runtime environment, as the optimal choice for implementing the information system [4], (Figure 2.2).

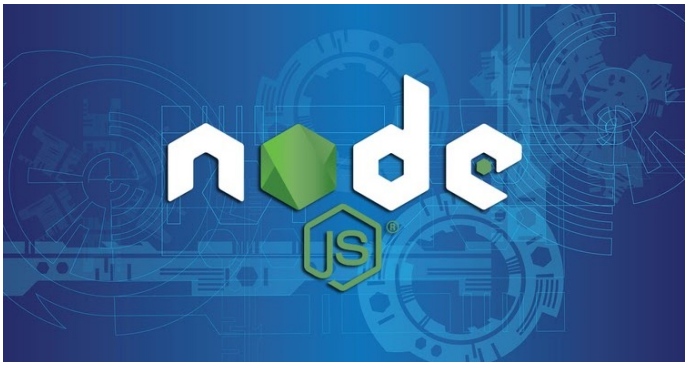

Figure 2.2: Node.js logo

The architecture of Node.js is designed for efficient handling of multiple simultaneous connections and input/output operations (I/O). Unlike traditional server-side technologies like PHP, which typically operate in a synchronous, single-threaded manner, Node.js takes a different approach. It doesn't wait for a task to finish before moving on to the next one. For example, when employing Apache HTTP Server with specific Multi-Processing Modules (MPMs) to reach the same result, PHP may encounter potential bottlenecks and challenges under high loads [5]. Node.js, with its non-blocking, event-driven architecture, efficiently handles multiple tasks concurrently. It employs a callback function system, allowing it to process other tasks without getting stuck or overloading memory by creating separate threads for each request (unlike servers like Tomcat which have limitations, handling around 200 threads). This allows Node.js to avoid the potential performance issues associated with certain server configurations, commonly found in traditional multithreaded setups.

Node.js, operates on a single-threaded model that maximizes the use of this thread. It achieves this, by assigning tasks that would typically block the main thread to background threads, freeing up the main thread to manage other tasks efficiently. This approach, significantly reduces the overall overhead associated with traditional multi-threaded models, enabling Node.js to handle multiple events simultaneously with ease.

This way, the application can host high traffic and numerous users without limitations or performance degradation.(Figure 2.3)

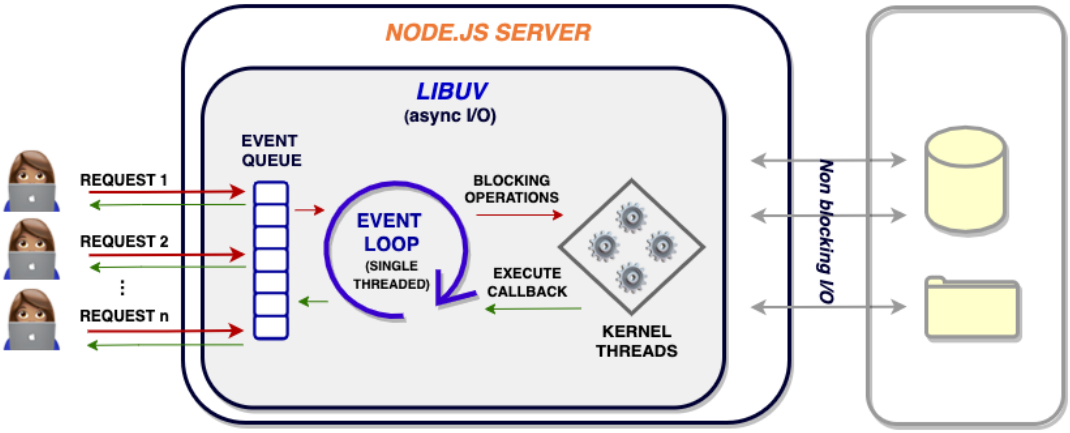

Figure 2.3: Node.js Architecture

Node.js, is suitable for real-time applications due to its low latency capabilities and WebSocket support. Its ecosystem is rich with functional modules and libraries contributed by the community, optimized for various tasks. These functional modules, often follow best performance practices and can be easily integrated into Node.js applications to enhance their capabilities. Provides built-in support for data streaming, allowing the processing of data in small pieces as they are read or written, reducing memory usage and enabling efficient handling of large datasets. Additionally, the use of the same programming language across the entire stack is beneficial in terms of programming time, increase in productivity, ease of learning and extensibility.

Providing users with the ability to store and retrieve their projects, is fundamental for any dynamic and user-centric application. The flexibility of MongoDB in handling various 3D data types aligns seamlessly with Node.jss capabilities, ensuring a smooth data flow between the server and the database. MongoDB's scalability matches the unpredictable data requirements of Three.js applications, offering efficient management of diverse 3D data structures. This combination ensures seamless storage and retrieval of complex 3D data, making MongoDB a robust solution for supporting Three.js projects. Node.js pairs well with MongoDB due to their JavaScript-based foundation; they both utilize JavaScript in their ecosystem. This allows for smoother integration and interaction between the two technologies. Node.js utilizes JavaScript for server-side operations,

while MongoDB stores data in BSON (Binary JSON), a format very similar to JSON (JavaScript Object Notation). Because JSON is native to JavaScript, handling data in Node.js that comes from MongoDB is straightforward and requires minimal conversion or parsing. This compatibility facilitates seamless communication between the database and the server-side environment [6] (Figure 2.4).

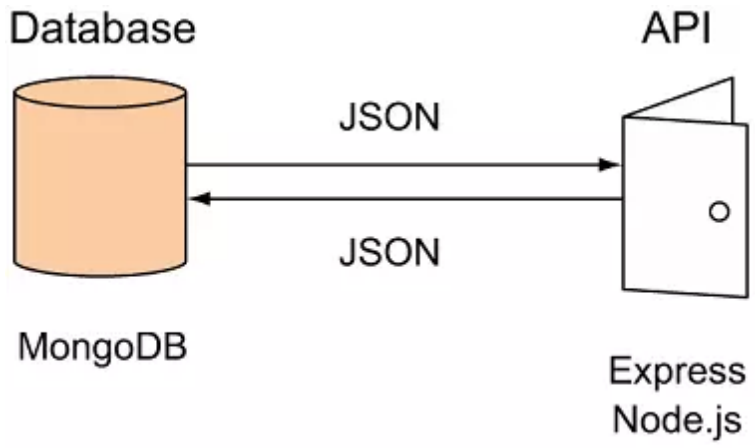

Figure 2.4: MongoDB - Web server communication

It's important to note a key limitation of MongoDB. This database imposes a maximum document size of 16MB per save, which can pose challenges for applications dealing with large files, like ours. However, this limitation can be effectively addressed by utilizing GridFS storage. GridFS is a file storage feature within MongoDB, designed to handle large files by breaking them into smaller chunks, thus overcoming the document size limitation. GridFS operates through two primary collections within MongoDB:

- **fs.files :** This collection stores metadata about the files including filenames and related information such as size, referred to as metadata. In metadata, we are able to include additional data, optimizing our platform's file management and retrieval processes.

- **fs.chunks :** Here lies the actual file data, divided into smaller chunks for efficient storage and retrieval. Each chunk, is indexed and linked to its corresponding metadata in the fs.files collection.

GridFS ensures efficient storage and retrieval of large data files within the database [7] (Figure 2.5).

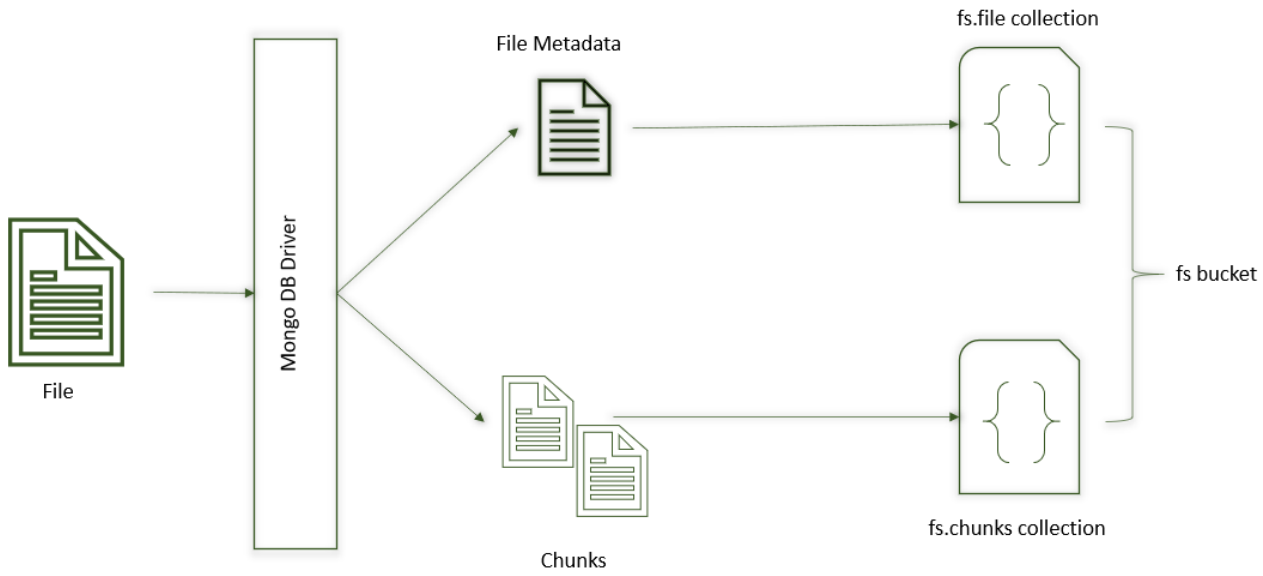

Figure 2.5: GridFS

Express.js, was chosen for shaping the back-end infrastructure due to its simplicity and power. This framework provides a minimalistic yet robust structure that outperforms alternatives enabling flexible development. Its intuitive routing and seamless integration capabilities establishing it as the prime choice for scalable applications. Backed by a strong community and a track record of reliability, Express.js ensures exceptional performance and adaptability which both are crucial elements for modern web development [8]. (Figure 2.6).

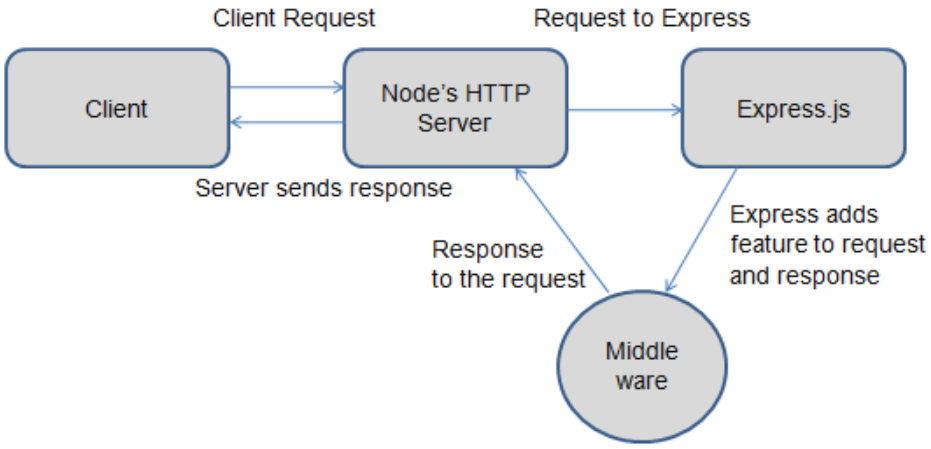

Figure 2.6: Data flow through Express.js

EJS (Embedded JavaScript) serves as a templating engine, enabling dynamic content generation by embedding JavaScript code into HTML templates for server-side rendering. This approach facilitates faster initial page loads and efficient data retrieval from the server, enhancing user experience without relying heavily on client-side processing. EJS can also be used for client-side dynamic content generation. Our system's front-end is implemented using vanilla JavaScript instead of larger frameworks like React or Angular, focusing on simplicity. This approach keeps the code clear, free from the complexities, constraints and performance overhead associated with external frameworks, giving us direct control over web page manipulation. The integration with Express.js is seamless, enabling smooth data flow. By using fewer external framework, we reduce potential conflicts and simplify the development process. However, this choice comes with disadvantages. It requires more code compared to using external frameworks, making it time-consuming and as the codebase expands, maintenance becomes increasingly challenging. (Figure 2.7).

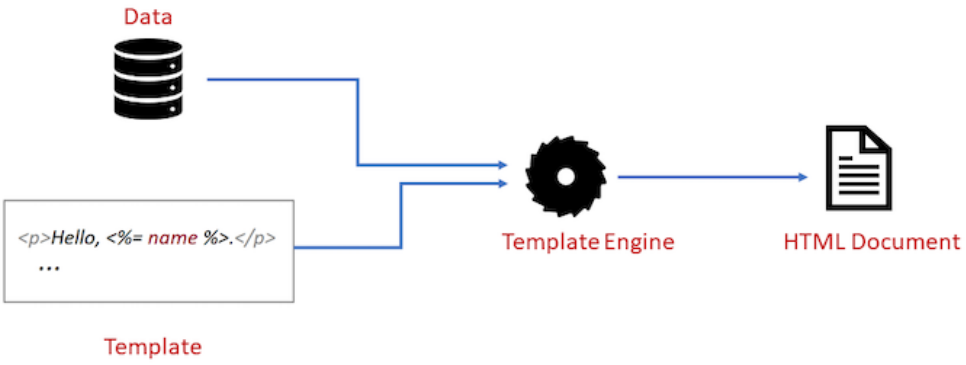

Figure 2.7: EJS workflow

In order to establish a user-friendly environment, user interface design leverages the power of JavaScript and CSS to develop an engaging environment, accessible to all skill levels. Through intuitive design principles guides users seamlessly, offering clear navigation and straightforward layouts. JavaScript enables simplified interactions, allowing users without technical expertise to engage effortlessly with dynamic content and interactive elements. CSS ensures visual consistency, providing aesthetically pleasing and easy to comprehend designs.

These technologies form our tech stack (Figure 2.8) that is structured around the MVC model (Figure 2.9). Model, manages data and business logic through interactions with the MongoDB database. The view, powered by EJS, handles presentation and user interface rendering. The Controller, implemented with Express.js, manages user input, processes it and orchestrates updates to the Model and View.

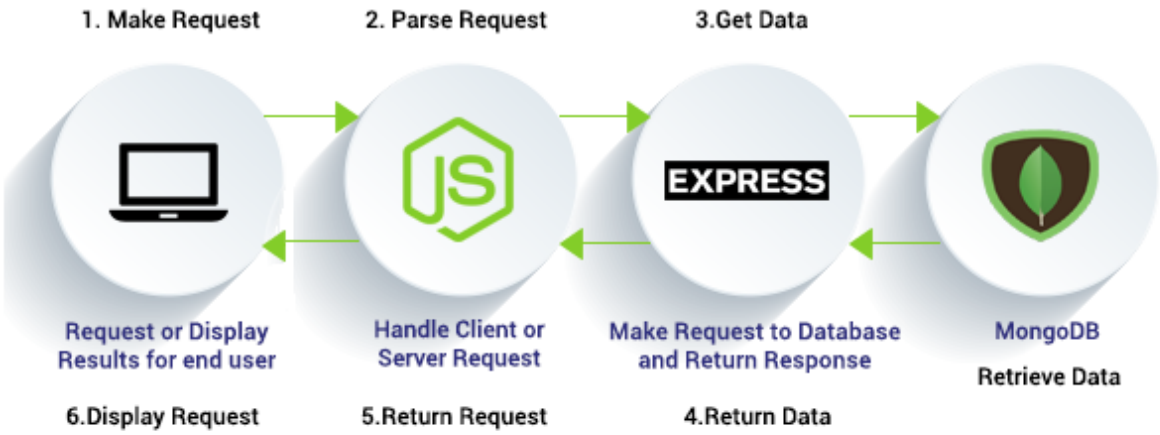

Figure 2.8: Tech Stack

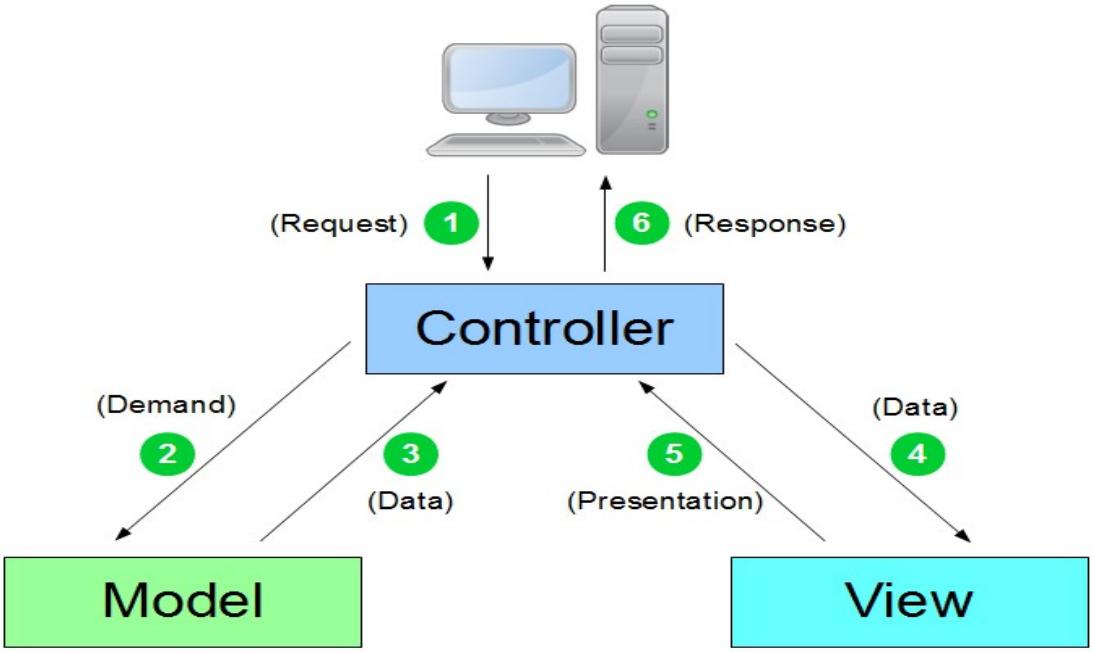

Figure 2.9: MVC model

2.2.2 3D Environment Integration with Three.js library

Three.js [9] [10] (Figure 2.10) is a powerful JavaScript library and application programming interface (API), that simplifies the creation of captivating 3D graphics and interactive experiences for the web. Born out of the necessity to bring three-dimensional visualizations to browsers without the complexities of low-level WebGL programming, Three.js establishes its position in the world of web-based 3D development.

Key Features:

- **Abstraction of WebGL Complexity:** Three.js abstracts the complexities of WebGL, providing developers with a high-level interface to create 3D scenes without delving into low-level graphics programming.

- **Cross-Browser Compatibility:** Ensuring compatibility across various browsers, Three.js shields developers from the nuances of different WebGL implementations. This enables the creation of consistent and immersive 3D experiences on a wide range of devices and platforms.

- **Rich Set of Primitives and Materials:** Three.js offers a diverse set of primitive geometries (e.g., cubes, spheres, planes) and materials (e.g., Lambert, Phong, and Standard materials), that empower developers to craft visually compelling scenes. These elements can be easily manipulated, combined and customized to suit the desired outcomes.

- **Interactivity and Animation:** Leveraging Three.js, developers can imbue their 3D scenes with interactivity and dynamic animations. Whether responding to user input or orchestrating complex motion sequences, Three.js provides a robust framework for creating engaging and responsive content.

- **Extensibility and Community Support:** Three.js boasts a vibrant community and a wealth of extensions, allowing developers to enhance their projects with additional features and functionalities. The library is open-source, encouraging collaboration and the continuous evolution of its capabilities.

Figure 2.10: Three.js

2.2.3 Activating Real-Time Communication with Socket.IO

Leveraging the Socket.IO library [11] [12] (Figure 2.11), enables real-time communication. Initially, between user requests and server responses, ensuring live responsiveness within the application. Moreover, among users collaborating within teams, allowing them to design collectively in a shared digital environment.

Figure 2.11: Socket.IO logo

Socket.IO, is a JavaScript library designed to establish and maintain persistent con-

nections between clients and servers, enabling bidirectional data exchange (Figure 2.12). Introducing rooms and namespaces, enhances communication organization and efficiency, ensuring responsive and engaging user experiences.(Figure 2.13).

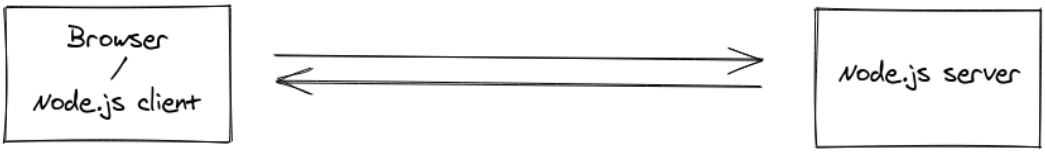

Figure 2.12: bidirectional communication

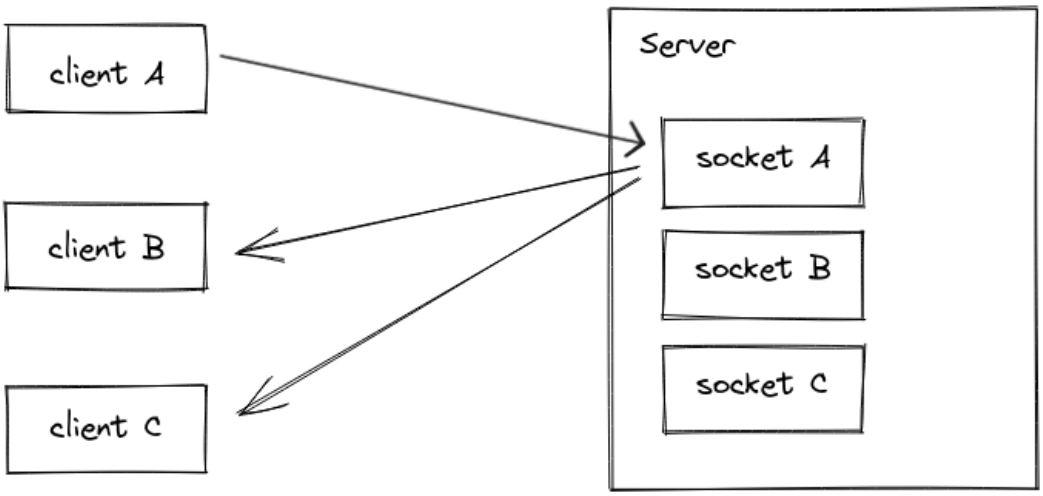

Figure 2.13: Broadcasting events from a client

2.3 Network Protocols

2.3.1 TCP/IP Protocol Stack

The Network Protocol Stack presented below (Figure 2.14), is a layered framework that enables end-to-end communication over a network. This stack, ensures the reliable delivery of data along with a secure and encrypted connection between a client and our server. [13]

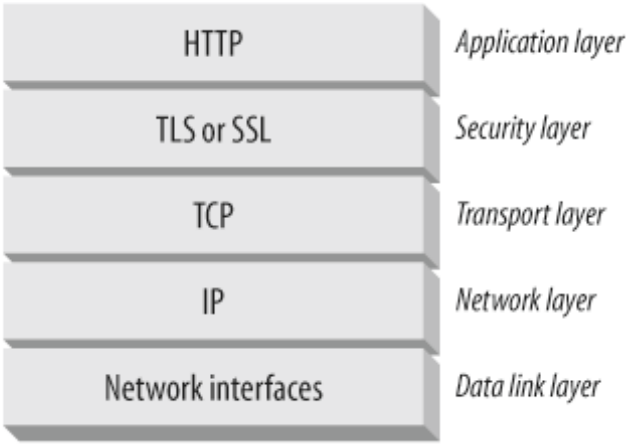

Figure 2.14: Network Protocol Stack

It includes HTTP for application layer communication, TLS for securing data, TCP for reliable transport and IP for network routing. Network interfaces are hardware components or built-in functionalities in devices, that enable them to connect to a network, either wired (Ethernet) or wireless (Wi-Fi).

Web browsers talk to web servers over TCP connections. A TCP connection is a reliable, bidirectional communication channel established between two devices over a network.

2.3.1.1 Hypertext Transfer Protocol (HTTP)

HTTP, a fundamental web communication protocol within the TCP/IP suite, orchestrates the exchange of information between browsers and servers. It primarily handles resource retrieval, like HTML pages and multimedia content. In the context of security, HTTP can be augmented with TLS resulting in HTTPS (Hypertext Transfer Protocol Secure).

2.3.1.2 Transport Layer Security (TLS)

TLS is a cryptographic protocol designed to secure communication over a computer network. It provides privacy and data integrity between two communicating applications,

preventing eavesdropping, tampering or message forgery. TLS is commonly used to secure web connections, ensuring that data transmitted between a user's web browser and a server remains encrypted and protected from unauthorized access. It is the successor to the earlier SSL (Secure Sockets Layer) protocol and is widely employed for securing sensitive information, such as login credentials and financial transactions on the Internet.

2.3.1.3 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) is one of the protocols in the TCP/IP protocol suite that is used to ensure the reliable delivery of data over a network.

TCP manages data by breaking it into segments, encapsulates them into IP packets and reassembles at the recipient. Providing flow and congestion control, TCP regulates the rate of data transmission ensuring that it aligns with the capacity of the recipient's network and prevents overload. Designed for reliability even on unreliable networks, TCP detects and retransmits lost or corrupted packets ensuring accurate data delivery.

2.3.1.4 Internet Protocol (IP)

The Internet Protocol (IP) is one of the protocols in the TCP/IP protocol suite and it is used for addressing and routing data packets across networks, ensuring the accurate and reliable delivery of data packets between devices on a network.

IP, is essential for guiding data packets across the Internet. It assigns unique addresses (IP addresses) to devices, allowing routers to direct packets from a client to the correct destination. IP determines the most efficient path for packets, ensuring their accurate transmission. The IP packet header, contains the information of the source and destination addresses, aiding routers in informed forwarding decisions. In some cases, data packets are fragmented by IP for efficient transfer across networks with varying Maximum Transmission Unit (MTU) sizes (largest size of a packet).

2.3.1.5 Communication process :

Entering the URL https://www.example.com in the browser, we have :

- **The Scheme**, https:// is the scheme. HTTPS stands for Hypertext Transfer Protocol Secure. This scheme tells the browser to make a connection to the server using Transport Layer Security, or TLS.
- **The Domain Name**, example.com is the domain name of the site, serving as the memorable address that points to a specific server's IP.

1. **DNS Resolution:** browser initiates a Domain Name System (DNS) resolution to translate the human-readable domain (e.g., www.example.com) into an IP address.
2. **TCP Handshake:** once the IP address is obtained, browser initiates a secure connection using the Transmission Control Protocol (TCP). This involves a three-way handshake where browser sends a TCP SYN (synchronize) packet to the server, the server responds with a SYN-ACK (synchronize-acknowledge) packet and browser acknowledges with an ACK (acknowledge) packet. This process, establishes a reliable TCP connection between client's browser and the server.
3. **TLS Handshake within TCP Connection:** with the TCP connection established, browser initiates a secure connection using the Transport Layer Security (TLS) protocol. The TLS handshake occurs within the existing TCP connection, involving an exchange of messages where the browser and the server agree on encryption methods and exchange cryptographic keys to establish a secure communication channel.
4. **HTTP Request within TLS Tunnel:** after the secure TLS tunnel is established within the TCP connection, browser sends an HTTP request to the server. This request specifies the resource (webpage, image, etc.) requested and includes information like the method (GET, POST, etc.) and headers.
5. **Server Processing:** Server receives the HTTP request, processes it and retrieves the requested resource.
6. **HTTP Response within TLS Tunnel:** the server sends back an HTTP response to browser. This response includes an HTTP status code indicating the success or failure of the request, along with the requested resource and additional metadata. The response is sent within the secure TLS tunnel.

7. **Rendering:** browser receives the response, decrypts it if necessary (within the TLS tunnel) and renders the content. This involves parsing HTML, interpreting CSS for styling, executing JavaScript for interactivity and loading additional resources such as images or stylesheets.

8. **TCP Connection Closure:** once the HTTP communication is complete, the TCP connection is closed. The closure can be initiated by either the client or the server, involving a TCP FIN (finish) packet exchange.

9. **TLS Connection Closure:** simultaneously with the TCP connection closure, the TLS-encrypted connection ensuring the security of the data exchange closes.

IP Packets: Throughout this process, the data is transmitted in the form of IP packets. These packets encapsulate the information and ensure its proper routing between browsers and servers over the internet. Each packet includes source and destination IP addresses, enabling the correct delivery of data across the network.

2.3.2 Real-time Communication Protocols

Socket.IO Protocol [14]

WebSocket Protocol Engine.IO Protocol

Engine.IO, is built on top of the WebSocket protocol which operates over TCP connections and serves as a fundamental communication protocol, offering a full-duplex communication channel over a single, long-lived connection. WebSocket, an application layer protocol in the network protocol stack, facilitates real-time bidirectional communication between applications. It provides a persistent, low-latency connection that is well suited for scenarios requiring instantaneous updates.

Engine.IO leverages WebSocket as its primary transport mechanism, providing an efficient and responsive communication channel. However, it goes beyond WebSocket by incorporating additional features, particularly in situations where WebSocket connections

may face challenges. Engine.IO integrates HTTP long polling as a fallback mechanism. This fallback allows communication to adapt to varying network conditions and navigate through potential obstacles, making it robust in different environments.

Building on this foundation, the Socket.IO protocol takes real-time communication a step further. It introduces additional features over the communication channel provided by the Engine.IO protocol. These features include built-in support for reconnection, room-based communication and an event-driven programming model that simplifies handling real-time events in applications.

This stack of protocols, from WebSocket to Engine.IO and Socket.IO, is particularly beneficial in scenarios where low-latency communication is crucial. Examples include real-time chat applications, online gaming platforms, collaborative editing tools and other applications that demand instant updates and seamless communication between clients and servers. The combined capabilities of these protocols make them adaptable and well suited for a wide range of real-time web development use cases.

Communication process and protocols combination:

WebSocket:

1. **Handshake:** The client sends a WebSocket handshake request to the server and upon successful handshake, a persistent full-duplex communication channel is established.

2. **Bi-Directional Communication:** WebSocket enables bidirectional communication, allowing both client and server to send messages.

   WebSocket operates over the Transmission Control Protocol (TCP), ensuring a reliable and ordered data delivery.

Engine.IO:

1. **Built on WebSocket:** Engine.IO primarily uses WebSocket as its transport mechanism for real-time communication.

2. **Fallback Mechanism:** In scenarios where WebSocket connections face challenges, Engine.IO incorporates HTTP long polling as a fallback mechanism, to maintain communication. HTTP long polling also utilizes TCP connections.

Socket.IO:

1. **Extends Engine.IO:** Builds on the foundations of Engine.IO, providing additional features and flexibility.

2. **Event-Driven Model:** Socket.IO uses an event-driven programming model where clients and servers can emit and listen for custom events.

3. **Automatic Reconnection:** Includes built-in support for automatic reconnection, enhancing the reliability of real-time connections.

4. **Additional Features:** Introduces features like room-based communication and a convenient API to simplify real-time application development. The use of TCP contributes to the robustness of these additional features.

2.4 Background

2.4.1 Evolution of Web-based 3D Visualisations

The evolution of web technologies from the era of browser plugins to the contemporary landscape of 3D web capabilities has transformed model visualization. In the early stages, web developers heavily relied on proprietary plugins, such as Adobe Flash Player, to introduce 3D graphics into web browsers [15]. While these plugins offered a gateway to richer content, they were accompanied by security concerns, standardization issues, dependencies on third-party software and notable latency drawbacks. The incorporation of plugins often resulted in slower loading times and increased latency, hindering the seamless delivery of 3D content to users. This latency issue, not only affected the overall

user experience but also posed challenges for developers striving to create responsive and interactive applications.

As a pivotal shift away from plugins, WebGL emerged as an open standard, revolutionizing 3D web graphics by providing a native, standardized API accessible directly within modern browsers [16]. This laid the foundation for the development of immersive and interactive 3D experiences on the web.

To further streamline the 3D model visualization process, making it more accessible, libraries like Three.js were introduced. These libraries abstracted the complexities of working directly with WebGL, empowering developers to create sophisticated 3D content without delving into the complexities of low-level APIs. This marked a significant turning point, enhancing user engagement and collaboration by enabling seamless model viewing experiences on web.

Eliminating the reliance on plugins, improved accessibility, simplified development practices and significantly enhanced performance. The shift towards plug-in-free technologies aligns seamlessly with the overarching trend of crafting lightweight, efficient and user-friendly web applications. This evolution signifies a substantial leap forward in the realm of 3D model visualization, where the convergence of standardized native APIs and higher-level frameworks has ushered in a new era of innovation and user-centric design. The improved performance coupled with streamlined development practices, has resulted in a more responsive and engaging experience for users interacting with 3D models on the web.

2.4.2 3D JavaScript Frameworks

There are several frameworks developed throughout the years to streamline the process of 3D web-based visualisations, including Three.js, Babylon.js and X3D/X3DOM among the most popular choices. The absence of a standardized 3D web visualization framework introduces a challenge for developers to choose 3D web technology. This choice is often

based on specific project requirements, developer preferences and the desired features for implementing 3D graphics.

Comparing Three.js and Babylon.js, both have their strengths. Three.js demonstrates superior performance with lower memory usage. However, developers express a preference for Babylon.js citing quicker feedback during development. While Three.js may excel in runtime performance, Babylon.js offers advantages in terms of development efficiency and responsiveness throughout the coding and debugging phases as indicated in [17]. Checking Github though, Three.js has around 97,000 stars whereas Babylon.js has approximately 21,000. So, Three.js seems to have a larger and more actively engaged developer community compared to Babylon.js. GitHub stars objectively can be an indicator of a project's popularity and the level of interest it has garnered among developers.

X3D/X3DOM and Three.js, show a clear difference in popularity between the academic community on the one hand and the wider development community on the other. X3D/X3DOM is featured in dozens of academic articles, whereas Three.JS is barely mentioned in any. However, the Github stars show that Three.JS is clearly by far more popular among developers than X3DOM as mentioned in [18]. Worth noting, Three.js provides an efficient approach to 3D graphics without the necessity to represent each 3D element as a DOM element in contrast to X3D/X3DOM. Three.js, utilizes WebGL to directly render 3D scenes onto a canvas, resulting in enhanced performance and capabilities for 3D applications. On the other hand, X3D/X3DOM represents 3D scenes as XML documents integrated within the DOM. While this allows for a structured description of scenes, it may introduce some overhead, particularly when managing complex and dynamic 3D graphics. Unlike Three.js, which interfaces directly with WebGL for optimized rendering, X3D/X3DOM's integration into the DOM may incur additional computational costs, potentially affecting the performance of intricate 3D applications.

2.5 Related Work

Numerous research studies highlight the imperative of harnessing the potential of 3D web technologies to encourage stakeholder involvement in urban planning and architecture. Although there are quite studies proposing different systems for the implementation of 3D editors, there are relatively few studies that incorporate real-time collaboration. In [19], a P2P network implemented in order to enable co-design through direct communication between peers without relying on a central server, reducing the strain on server and minimizing costs. This approach results in lower latency compared to client-server architectures, enhancing the real-time nature of collaborative interactions. However, web-based P2P communication comes with limitations such as the 15MB data transfer constraint, which need to be considered when evaluating its suitability for applications related to real-time co-design, as the file sizes in many scenarios frequently surpass the 15MB threshold. A solution could be a chunking mechanism to break down the data into smaller segments that fit within the 15MB limit and then sending these segments sequentially. Related work, regarding collaborative 3D editing tools, was also made in [20]. The researchers, proposed a system designed to enable real-time editing and sharing of 3D objects among multiple users. XMPP, or Extensible Messaging and Presence Protocol, serves as the backbone for user and group management, as well as real-time communication. It allows users to log in, authenticate and join collaborative editing sessions. Through XMPP messaging, users can exchange information about their editing activities, ensuring synchronization of their editing screens in real-time. C3ware's whiteboard service, complements XMPP by providing a collaborative workspace where users can share and edit 3D objects. This service, handles editing operations and synchronization tasks on shared objects stored in cloud storage. When users perform editing actions, such as creating, modifying or deleting objects, C3ware ensures that these changes are propagated to all users in the collaborative session, maintaining consistency across their editing screens. Together, XMPP and C3ware create a collaborative environment where users can work together on 3D models in real-time. XMPP facilitates communication and coordination among users, while C3ware manages the shared workspace and ensures that edits are synchronized across all participants. XMPP and C3ware, may not be optimized for handling large files or multimedia content like images in real-time. While they can support text-based messaging and collaboration on shared objects, handling

large files or multimedia content efficiently in real-time requires specialized infrastructure and protocols. This architecture involves at least three servers. Employing separate servers for messaging (XMPP), collaboration (C3ware), hosting and potentially additional resources for certain types of real-time communication, provides advanced features for collaborative editing but introduces overhead and complexity. Coordinating communication between multiple servers may lead to management challenges, latency issues and increased resource consumption.

Chapter 3

User-Centric Requirements Analysis

3.1 Introduction

In the previous chapter we delved into determining a suitable tech stack for a project of this nature, exploring the characteristics that our system must satisfy to achieve our goals. This chapter examines the essential requirements for optimizing participatory design in urban planning and architecture by leveraging 3D web technologies, focusing on the capabilities users need to effectively utilize the system. To achieve this objective, it is necessary to identify and address the specific requirements governing its features.

A significant challenge in participatory design for urban planning and architecture, lies in bridging the gap between professionals and the general public. 3D web technologies can potentially address this challenge by providing the appropriate tools that enable users to explore proposed designs and offer feedback in a user-friendly manner. A simple interface in the 3D editor would encourage users to engage more actively and create their own designs.

Incorporating participatory design principles into urban planning and architecture leveraging 3D web technologies, also requires addressing technical considerations and system design challenges as mentioned in the previous chapter.

Collaboration lies at the heart of participatory design, given the diverse range of stakeholders involved each offering their individual insights and perspectives. Traditional methods of collaboration within the framework of participatory design, often face challenges such as communication barriers and limited stakeholder engagement. By harnessing 3D web technologies, the proposed system aspires to provide a digital platform

where stakeholders can actively participate in the planning process, visualize proposed designs and contribute ideas.

In the following sections of this chapter we will thoroughly analyze the specific requirements that the proposed system must fulfill and delve into the details of its features. By examining these aspects we aim to provide a comprehensive understanding of how the proposed system addresses the challenges and how the integration of 3D web technologies can pave the way for a more efficient application of the participatory design model in urban planning and architecture processes, leading to more sustainable and inclusive urban development.

3.2 Requirements

The aim of the suggested information system is to enhance participatory design processes by leveraging 3D web technologies. To ensure its effectiveness, it is crucial to fulfill several requirements. These requirements encompass system's tech stack, along with aspects such as data handling or user interface and experience. Adhering to these criteria is essential for implementing a dependable, efficient and user-centric system that can enhance participatory design processes. This section outlines the requirements that the proposed system must satisfy to meet the needs and expectations of its users.

1. **User Interface and Experience**: The system should offer its users an intuitive interface for managing various aspects of the application, including logging in, signing up, tutorial guidance, the 3D editor and the distribution of 3D designs.

   The application must offer users a range of capabilities within the 3D editor feature, facilitating the design process by equipping them with the necessary tools to achieve their desired outcomes. However, to ensure accessibility for non-professionals, the editor's interface should adopt a minimal, user-friendly design. This approach avoids overwhelming or confusing users with an abundance of tools and specialized features known only to experts.

2. **Interactive 3D Visualization**: The user should be able to edit and navigate the visualization of the 3D environment, therefore, a user interface window visualizing 3D scenes should be included, complemented by shaders and lights to enhance realism and visual quality of the environment.

3. **Transformation Controls**: The system should enable users access to transformation controls that empower them to manipulate their 3D objects with precision. These controls enable users to perform transformations, including translation, rotation and scaling, with ease. Whether adjusting the position of an object, rotating it to achieve the desired orientation or scaling it to fit within the scene, users can effortlessly fine-tune their designs to meet their desired results.

4. **Basic Objects**: Users have to be provided with a convenient way to enrich their 3D scenes with fundamental shapes. With the push of a button, users can seamlessly include objects such as cubes, spheres, cylinders and tetrahedrons into their creations. These basic shapes serve as building blocks, enabling users to start their design process with ease.

5. **Coordinates Panel**: The system should provide real-time information on the position, rotation and scale of objects in a user's 3D scene, along with the name of the currently managed object. This tool enables precise adjustments, ensuring objects are placed and oriented exactly as desired. Additionally, it aids users in identifying the object they are manipulating, streamlining their workflow and enhancing efficiency within the design process.

6. **Import**: The application should support import capabilities. This feature allows the import of various types of 3D model files and images. Enabling users to integrate external assets into their designs, they can enrich their projects thus facilitating the creation of more complex and compelling designs.

7. **Export**: The export functionality facilitates seamless sharing and utilization of designs across different platforms. By enabling users to export their designs in various formats, empowers them to showcase their work to a wider audience. This capability enhances the visibility of user creations encouraging wider collaboration and feedback, blunting participatory design.

8. **Colorization and Texturing**: The ability to colorize and apply textures to 3D objects, adds realism to designs, enhancing their visual appeal. By offering these tools to users, for color and texture manipulation, the editor enables them to imbue

their creations with personality and character, ultimately elevating the quality and impact of their designs.

9. **Scene Panel**: The addition of a scene panel serves as a guiding tool by displaying a comprehensive overview of the elements within the scene. It prevents confusion and ensures that users have full control over their designs. This panel enables interaction with the objects, facilitating easy identification inside a complex scene. Moreover, this feature paved the path for the creation of a merge button, enabling users to merge multiple 3D models into a single entity with which they can interact. This functionality, retains the flexibility for users to manipulate both the merged 3D model and its individual components. Providing users with an intuitive interface that displays the project's structure in an organized way, further enhances the ease of use of the editor, resulting in a more comprehensive and encouraging experience for users to engage with.

10. **Undo Redo**: This functionality allows users to confidently experiment with design changes, knowing they can easily revert to previous states if needed. This feature provides users with greater flexibility and control over their design process, empowering them to explore creative ideas and iterate on their designs with confidence. Offering seamless undo redo capabilities, the editor enhances the user experience and promotes a more flexible design workflow.

11. **Remove and Clone**: The addition of remove and clone functionality, further enhances user's control over their designs. With the ability to remove unwanted elements or duplicate existing ones, users can refine their designs with precision and efficiency.

12. **Share**: Share functionality enables users to showcase their projects on a dedicated page within the application, where other users can provide feedback. This feature fosters a sense of community where creators can receive valuable input and insights from stakeholders, enhancing the design process. By facilitating communication and interaction among users, share functionality promotes knowledge sharing and collective learning, enhancing the evolution of design processes.

13. **Project Management**: In order to ensure continuity throughout the design process, enabling users to revisit and build upon their work without losing progress, the system must provide the functionality of saving and retrieving projects. This capability allows users to seamlessly manage their projects and maintain organization, facilitating a more efficient and productive design workflow enhancing the overall design process.

14. **Teams**: The information system should enable users to engage in co-design activities virtually, transcending geographical barriers and fostering a culture of teamwork. By providing options to create and join teams, the application supports real-time collaboration, allowing users to work together towards common design goals. This collaborative environment, evolves traditional co-design practices by leveraging web capabilities.

15. **Temporary Storage**: To ensure uninterrupted user experience, the system should provide seamless navigation within the application while safeguarding workspace data, even if unsaved in the database. This mechanism serves as a safeguard against unexpected events, such as system lag or browser crashes, where users risk losing their progress. In such scenarios, binary data is temporarily stored on the server, allowing users to retrieve their work upon refreshing the page or revisiting the 3D editor. Leveraging the local storage capabilities of the browser, the system efficiently handles small-sized data, crucial for reconstructing the project's structure in the scene panel upon data retrieval. Moreover, this mechanism fosters collaboration by enabling team members who join an ongoing design session, to seamlessly catch up with the data from those who initiated the workspace, ensuring continuity and cohesion within collaborative projects.

3.3 3D visualization system

As mentioned in the previous chapter, we are harnessing Three.js for the implementation of the 3D environment. Delving further into this, the diagram below illustrates its functionality.

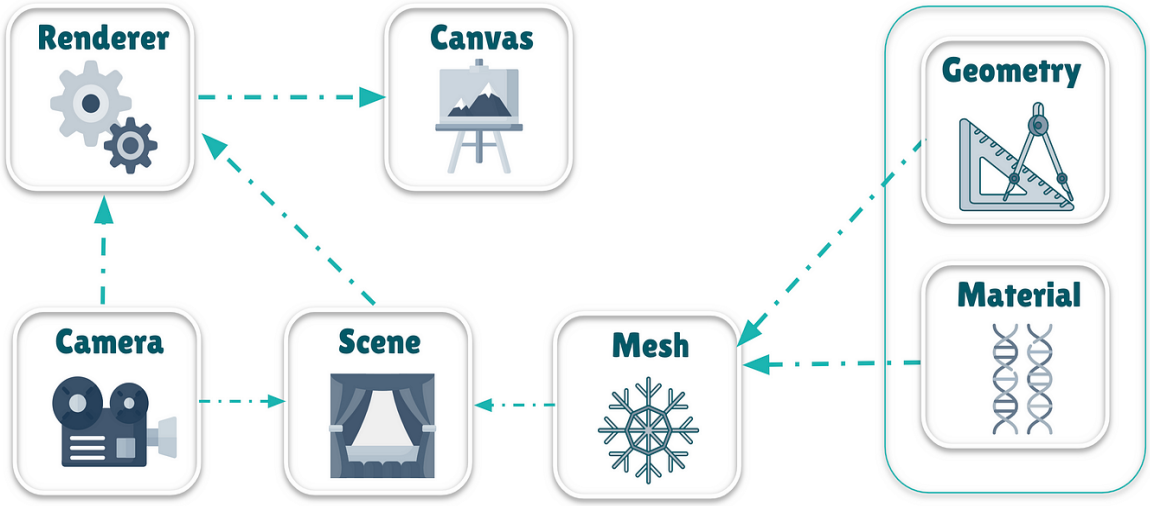

Figure 3.1: Three.js Flow Diagram

The renderer is one of the main elements of a Three.js project and is responsible for rendering 3D graphics on an HTML canvas. Using the render method which takes the camera and the scene as parameters, we render the scene on the canvas. Meshes are the objects of the scene. In order to see them, we need to illuminate and animate them. Each one of them is defined by its geometry which includes attributes for position, normal and UV coordinates. The position attribute describes the 3D coordinates (x, y, z) of each vertex, delineating the object's shape and spatial arrangement through vectors. The normal attribute, composed of three-dimensional vectors, specifies the surface direction at each vertex. These normals, are essential for the computers to understand how light should interact with the surface, creating shading and highlighting effects to produce realistic visuals. UV coordinates, represented by two-dimensional vectors, are encapsulated in the UV attribute. These coordinates, dictate how textures are mapped onto the object's

surface, ensuring precise texture placement and seamless appearance. Collectively these attributes contribute to the geometry of the object, shaping its form, surface properties and texture mapping, thereby enhancing the overall visual quality of rendered scenes. Additionally, meshes consist of materials that define the texture and color of the objects.

Applying this functionality along with the OrbitControls and TransformationControls modules from the Three.js library, we can address the second, third, fourth and fifth requirements as well as the part of the first requirement that refers to the 3D editor. Utilizing GLTFLoader, OBJLoader and MTLoader modules, we address the sixth requirement enabling the import of .gltf .obj .mtl and .glb in our 3D editor. Additionally, the GLTFExporter and STLExporter modules, utilized for the export functionality, satisfying our seventh requirement. Leveraging TextureLoader module, we do address the part of the eighth requirement referred to the texturing of objects. The utilization of undo-manager module, manages the tenth requirement and by using the remove and clone methods, provided by Three.js, we satisfy the eleventh requirement.

3.4 Use Case Scenarios

In order to fulfill all required specifications, the platform incorporates different views where the desired functionalities mentioned above are met. These include a login page, a sign-up page, a tutorial guidance, the 3D editor page which contains the majority of features and a home page for distributing designs. To access the platform, users are prompted to sign up and follow the tutorial guidance which covers the basic features of the 3D editor. Once this initial setup is completed, users are able to log in for subsequent sessions using their credentials. The user interface provides a header at the top of all pages once a user is logged in, allowing them to navigate within the application or log out. This component is commonly referred to as a partial. The index page, is the 3D editor. In the center of this page, there is a rectangular window called canvas, initially integrated with a sky as the starting environment, using the Sky module from the Three.js library, intended to enhance realism. Above this window there is a menu containing the basic objects and import/export functionalities. On the left side of the screen there is a menu that contains the functionalities of saving a project, load a project, create team, join a team, join room and share. On the right side, there is a panel that comprises a coordinates panel, providing real-time information on the position, rotation and scale of

objects as well as the name of the current object that is being managed. Below this there is a colour pallet allowing users to colourize objects and a texture loader enabling their texturing. Subsequently, this panel contains a dynamic scene panel that comprises the scene objects. Below this section, there is a 'save workspace' button and a 'clear scene' button. These are the components which contribute to the implementation of the 3D editor page. If users join a room dedicated to a team, the application navigates them to a page similar to 3D editor's but with slight differences. There, users are provided with the same interface as the 3D editor's page, except the abilities to create or join a team, join a room and share a project. In addition, on the right side of the page there is a real-time update of the online/offline users that constitute the team. Home page comprises 3D designs shared from users by the use of the share button on 3D editor page. There, every shared project is presented within a box containing the designer's username, a description, a 3D scene where the project can be visualized, real-time reaction buttons and a real-time comment area.

3.4.1 Log In View

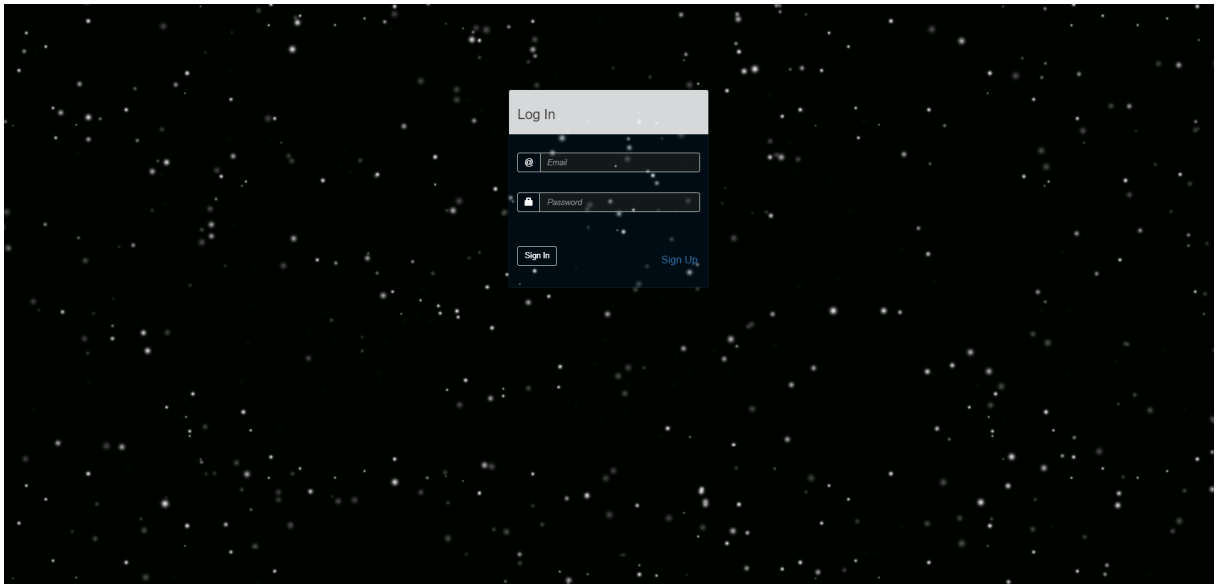

Figure 3.2: Log In View

Interacting with the interface depicted at Figure 3.2, users have the ability to log into the application or navigate to the sign-up view. If the user types a wrong username or password, sign in button redirects the user to log in page. Upon authentication, the user is redirected to 3D editor's page.

3.4.2 Sign up View

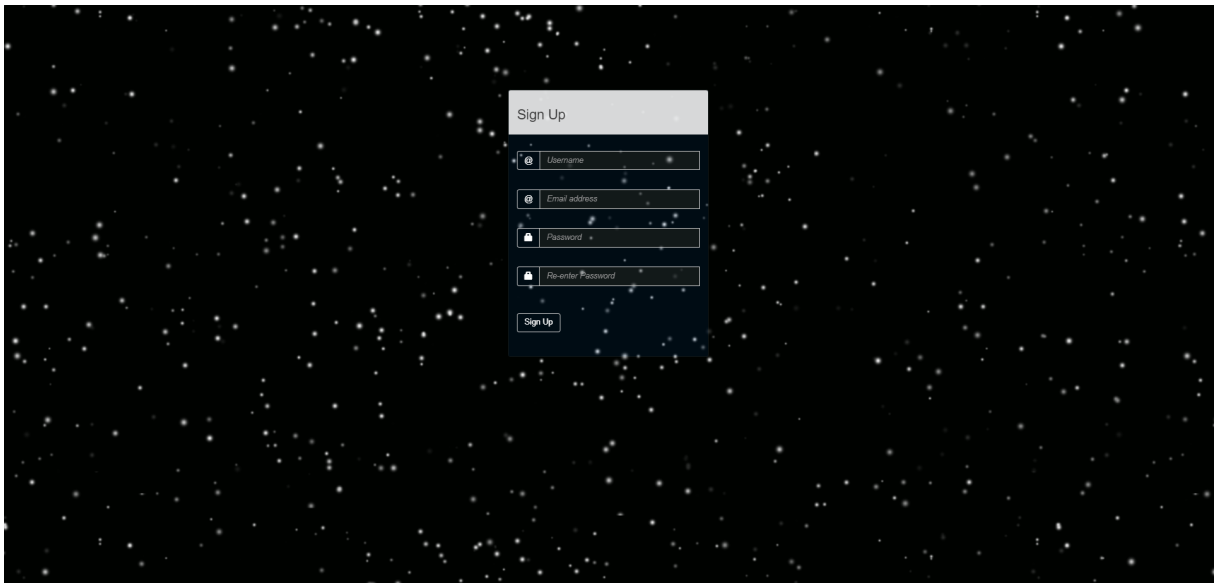

Figure 3.3: Sign up View

The interface presented at Figure 3.3 is responsible for signing up users to the platform. Sockets are connected on input fields, in order to enable a client-server communication and inform users if a username is available for use, if an e-mail has the correct format or it's unavailable and if the password entered by the user matches the one entered in the Re-enter Password input field. If a condition fails to meet the specified criteria, the user is redirected to sign-up page. Otherwise, the user is successful signed-up on platform and is redirected to our welcome page.

3.4.3 Welcome View and Brief Tutorial

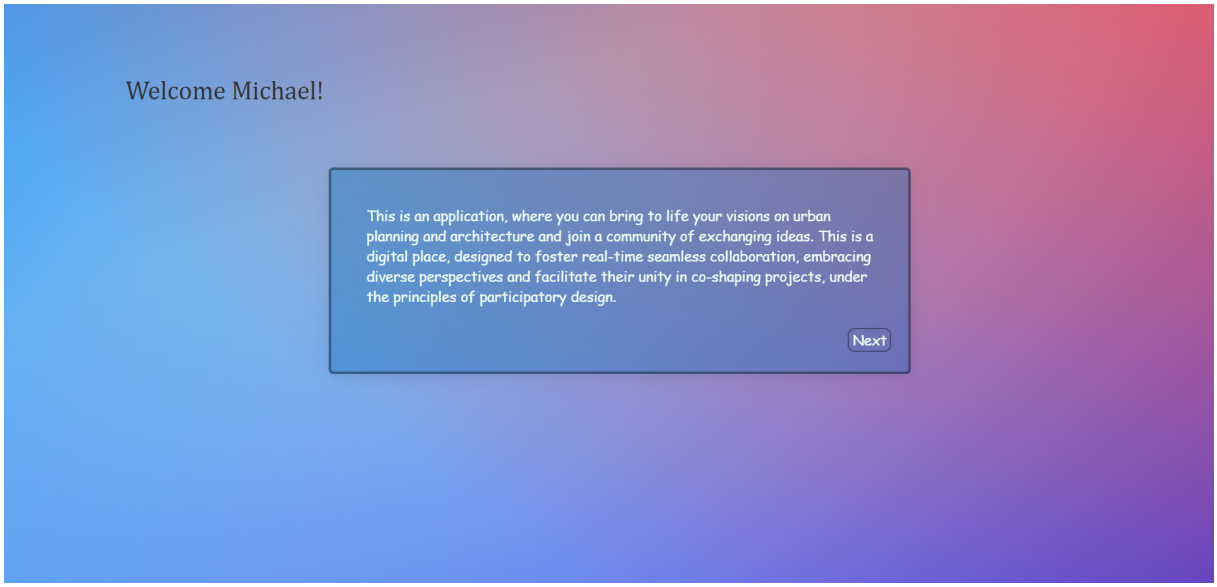

Figure 3.4: Welcome View

The interface shown in Figure 3.4 is our welcome page. This page welcomes the new user and introduces the purpose of the application.

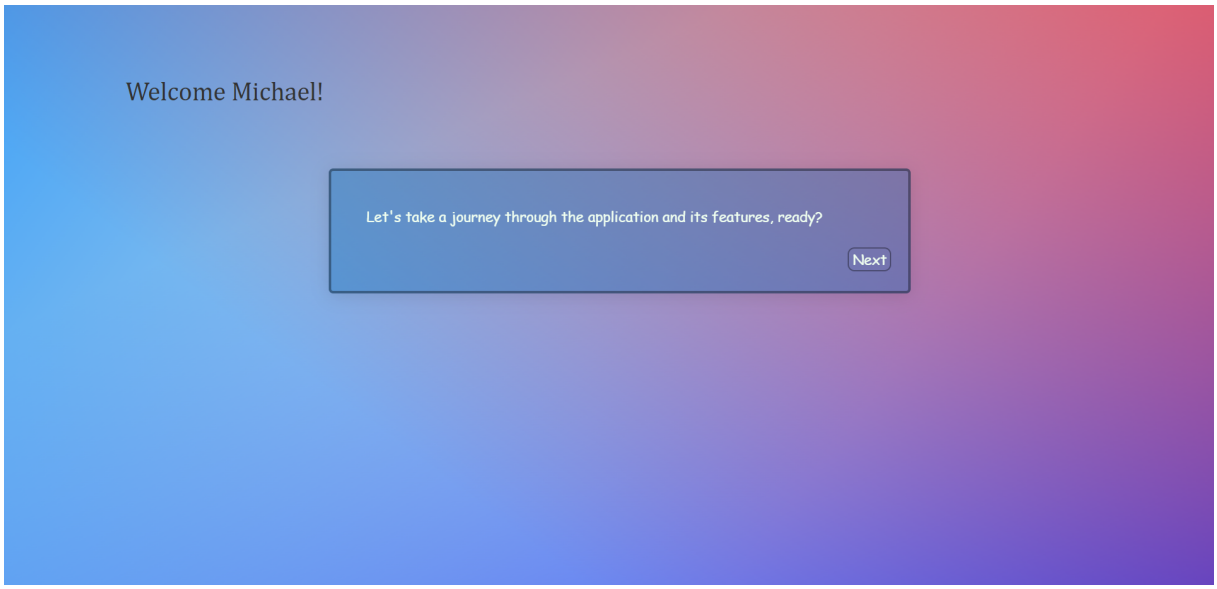

Figure 3.5: Tutorial Start

In Figure 3.5 is presented the next step of the welcome page, where users are prompted to begin the tutorial. By clicking 'Next', users are guided through a brief tutorial that explains the fundamental features of the 3D editor. This tutorial aims to encourage user engagement by equipping them with the necessary knowledge to start their first project from their initial visit to the platform. The steps of this process, include introducing the home page, the three-dimensional space and the keys that enable users to perform actions such as transforming controls, undo, redo, delete objects, duplicate objects, apply colors and add textures to objects. In each step, users are provided with interactive guidance on performing these actions, ensuring they have a hands-on tutorial experience rather than just observing. Users are encouraged to actively engage with the interface, allowing them to gain practical experience in manipulating objects, applying transformations and utilizing various controls effectively. On the Table 3.1, all steps are presented.

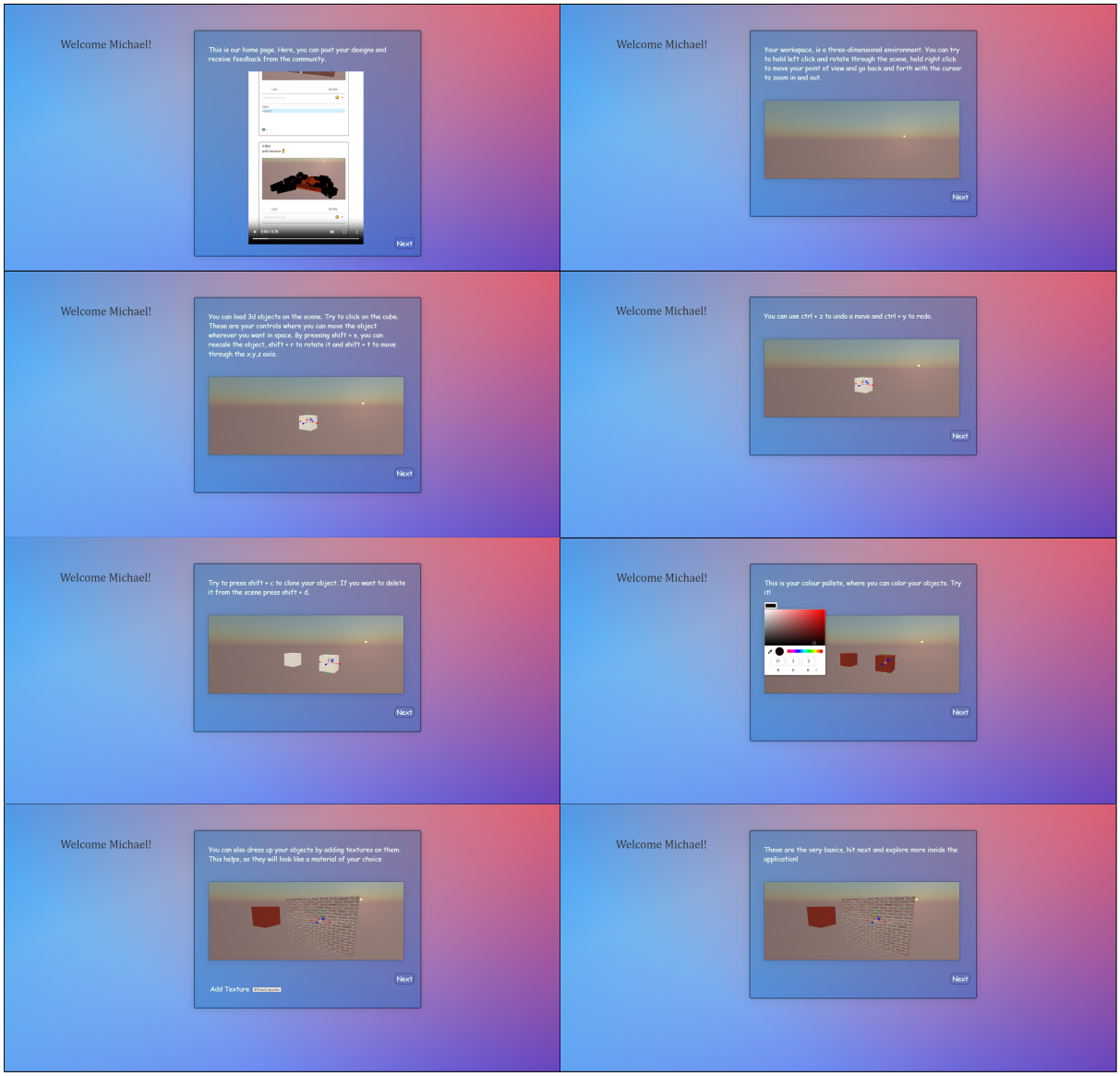

Table 3.1: Tutorial Steps

3.4.4 3D Editor View

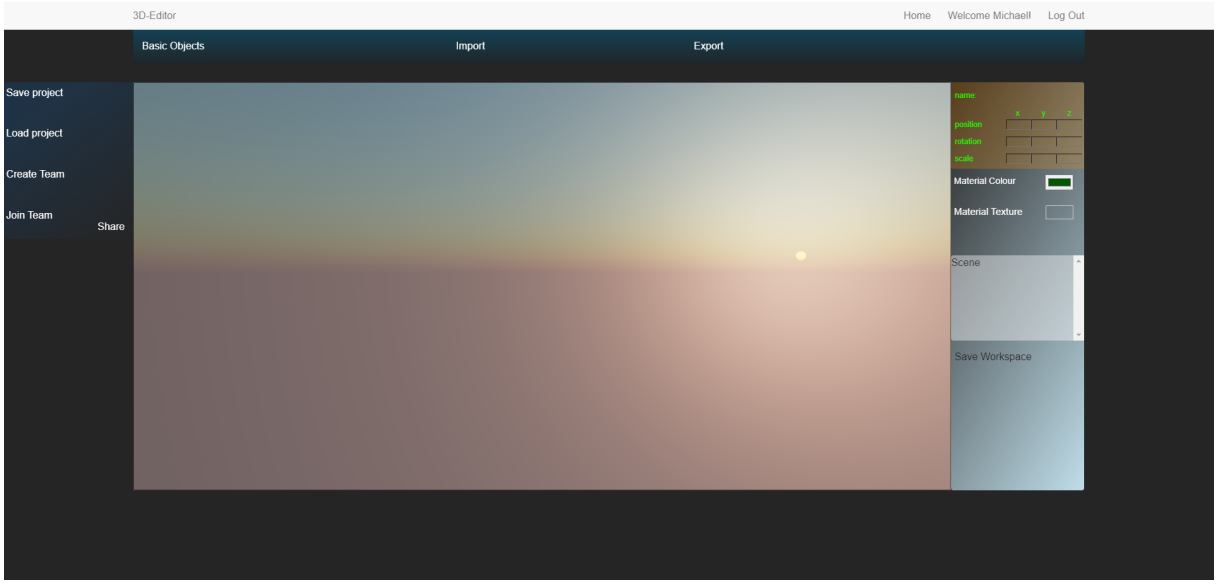

Figure 3.6: 3D Editor View

The interface shown in Figure 3.2, represents the initialized state of the 3D editor. Users can navigate within the application and log out, using the header which includes 3D-Editor, Home and Log Out options. Interacting with the canvas allows users to navigate the 3D environment and manage the objects it contains. The menu located at the top of the interface, consisting 'Basic Objects', 'Import' and 'Export' options, enables users to enrich their scene with basic objects, 3D models and images, as well as export the 3D environment. The menu on the left side, enables saving a project, load an already saved, create a team, join an existing one and share your design on home page. These actions are connected via sockets with our server, allowing for a real-time environment making our system asynchronous. If a user create a team or join an existing one, the menu expands to include an additional option to join a room, associated with the team. The panel displayed at the upper right of the screen, is the coordinates panel that enables real-time information on the position, rotation and scale of objects in a user's 3D scene, along with the name of the currently managed object. Below is the colour palette button for the objects. When pressed, it expands to provide a palette of colours.Then, there is a

texture loader, enabling users to apply textures to objects. If a user uploads a texture, a thumbnail of the texture is presented in the empty box displayed to its right. Following that, there is the Scene panel where the objects contained within the scene are displayed, allowing users to press an object's name to select it. Subsequently, the 'save workspace' button is placed, enabling temporary storage for the user's session. Below this button, the 'clear scene' button is displayed if the scene is not empty, allowing users to empty the scene.

3.4.5 3D Editor Team's Room View

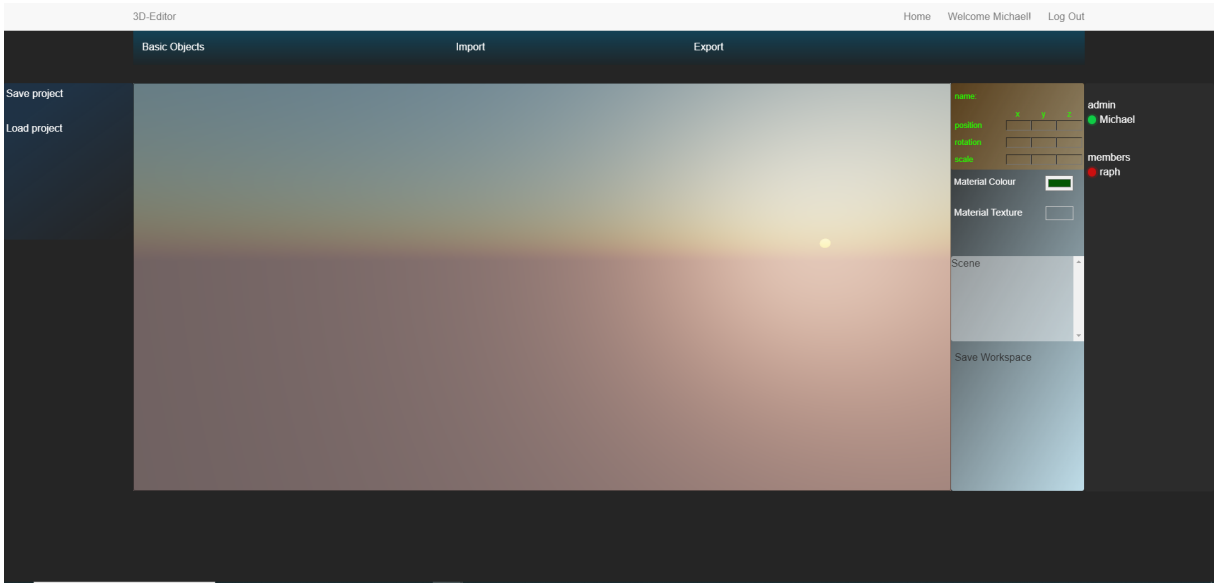

Figure 3.7: 3D Editor Team Room View

The interface depicted in Figure 3.7, presents a room of a team. The overall view is similar to the 3D editor's, except create team, join team, join room and share options on the left side's menu. On the right side of the screen, there is an additional field indicating the online status of team members. On this page, the 'save workspace' button enables the transmission of the room's session data to a member that just logged into the room via socket connection.

3.4.6 Home Page View

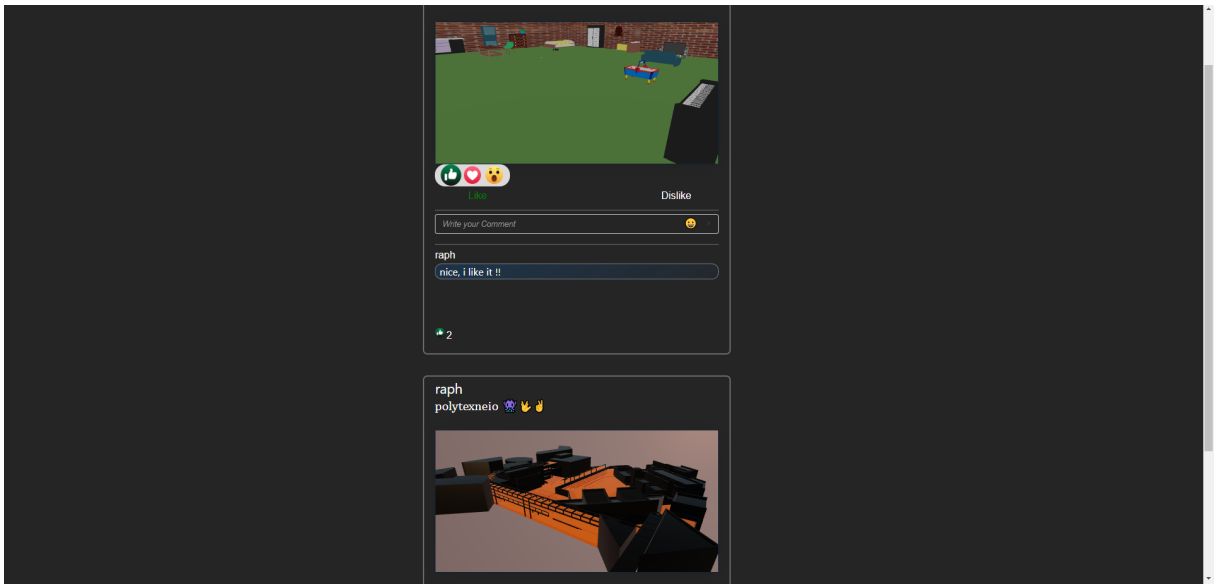

Figure 3.8: Home Page View

Figure 3.8 presents the home page. This page comprises the designs that users shared in a structure of multiple boxes. These boxes containing the designer's name, a relevant description, the shared scene where users have the ability to navigate within, a like and a dislike button, a field enabling users to comment the design and a field where comments and likes are displayed.

3.5 System's Manual

The Systems manual, serves as a comprehensive guide to every component within the application, highlighting their features and functionalities.

3.5.1 Manual of 3D Editor

Outlined below are the components constituting the 3D editor.

3.5.1.1 Upper Screen Menu

Figure 3.9: Upper Screen Menu

Figure 3.9 presents the upper screen menu that contains three buttons, 'Basic Objects', 'Import' and 'Export'.

- **Basic Objects :**

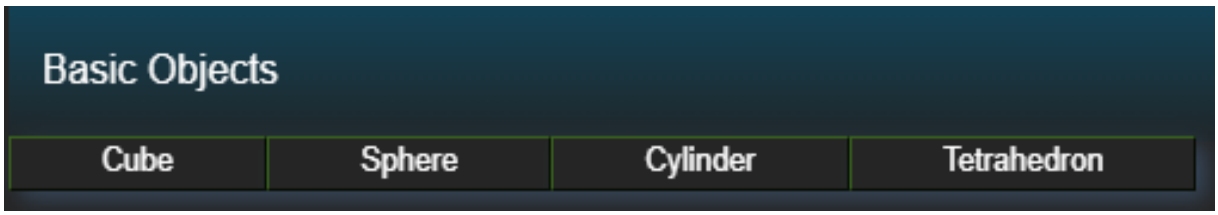

Figure 3.10: Basic Objects

Basic Objects button, when pressed, displays a row of options allowing the user to add fundamental shapes to their 3D scene.

- **Import :**

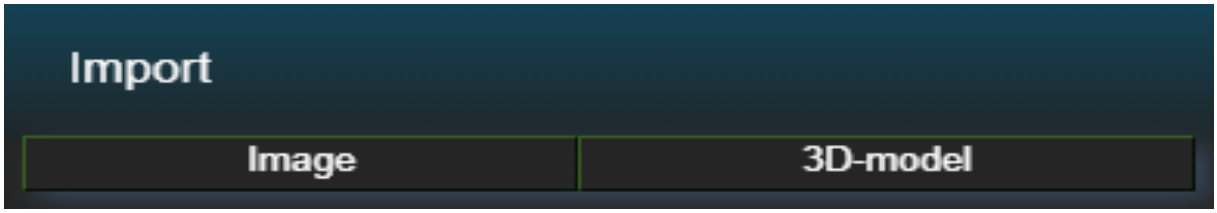

Figure 3.11: Import

Import button, pops up a row of two options enabling users to import images in JPG and PNG formats, as well as 3D models in GLTF, OBJ along with MTL and GLB file formats.

- **Export :**

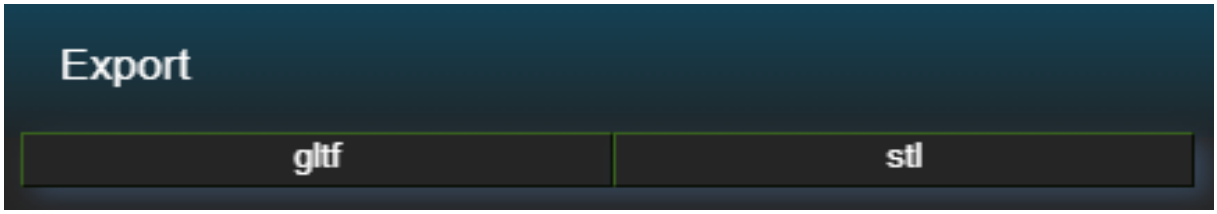

Figure 3.12: Export

If a user press export button, GLTF and STL options are displayed, allowing users to export their design either in GLTF format or in STL format.

3.5.1.2 Left Screen Menu

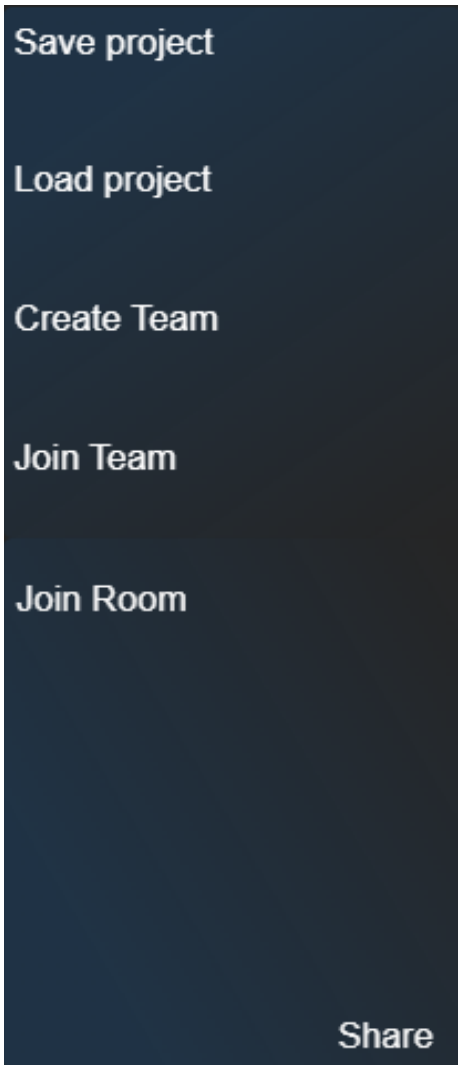

Figure 3.13: Left Screen Menu

The interface depicted in Figure 3.13, presents the menu at the left side of the screen.

- **Save Project :**

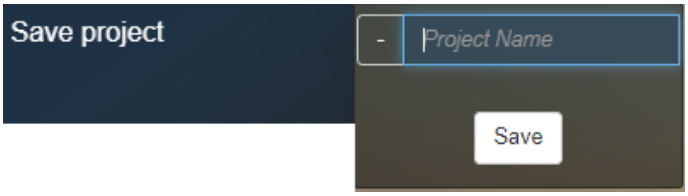

Figure 3.14: Save Project

Save Project button, pops up a window that enables users name their projects and save them to the database.

- **Load Project :**

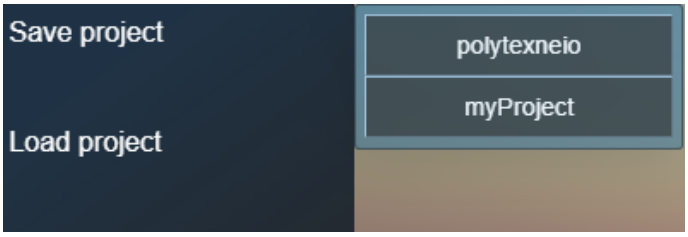

Figure 3.15: Load Project

When the Load Project button is pressed, a list of the user's saved projects are displayed as buttons. Upon selecting a specific project, it is loaded into the 3D scene.

- **Create Team :**

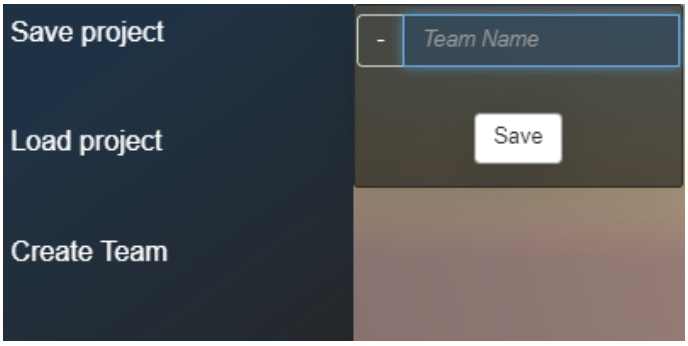

Figure 3.16: Create Team

Create team button, pops up a window similar to the save project's button where the user is enabled to create a team.

- **Join Team :**

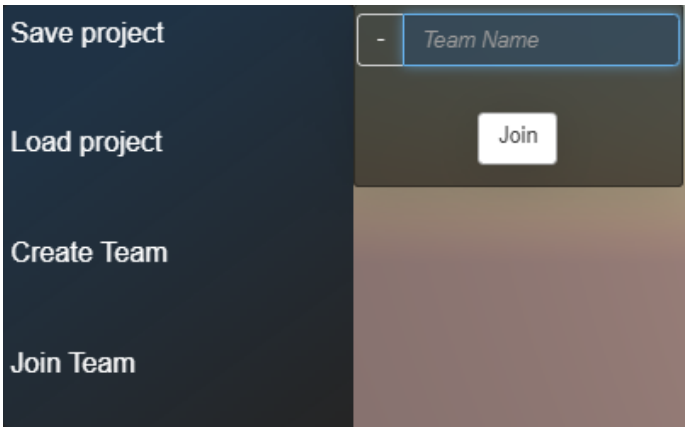

Figure 3.17: Join Team

If a user press join team button, a window that enables user to join an existing team is displayed.

- **Join Room :**

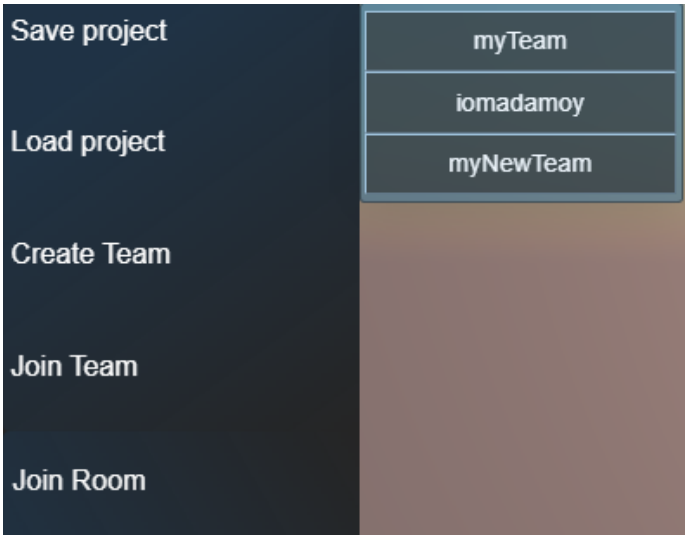

Figure 3.18: Join Room

When Join room button is pressed, a list of the teams that the user is registered is displayed. Upon selecting a specific team, user is redirected to the team's dedicated room.

- **Share :**

Figure 3.19: Share

Share button, displays a window that contains a field where users are enabled to write a description relevant to their design, a picture of the 3D environment that they are about to share and a share button to complete the share process by publishing their design to the home page.

3.5.1.3 Right Screen Panel

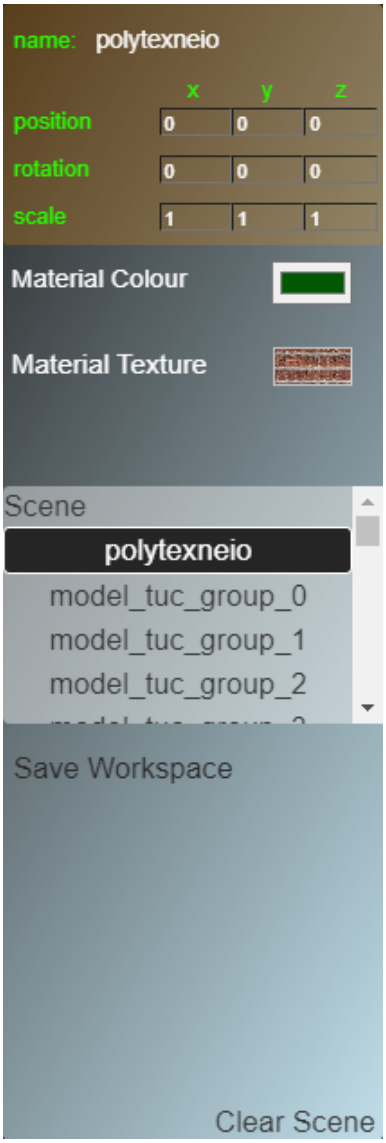

Figure 3.20: Right Screen Panel

The interface depicted in Figure 3.20, presents the panel at the right side of the screen.

- **Coordinates Panel :**

Figure 3.21: Coordinates Panel

This panel provides real-time information on the position, rotation and scale of an object in a user's 3D scene, along with its name. Users can modify these values directly from the panel. Any changes made are instantly reflected in the scene.

- **Material Colour :**

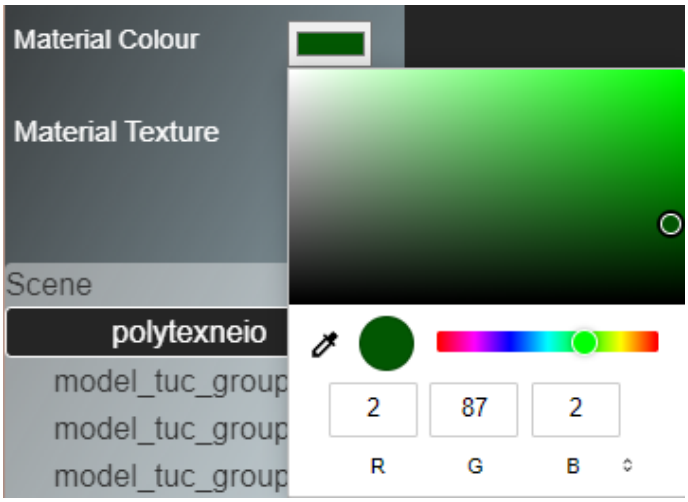

Figure 3.22: Material Colour

Material Colour button, pops up a colour palette that enables them to colourize their objects.

- **Material Texture :**

Figure 3.23: Material Texture

Pressing Material Texture button, users can load a texture and apply it to their objects. The box presented to the right of the button depicts the loaded texture, allowing users to see the texture they are applying.

- **Scene Panel :**

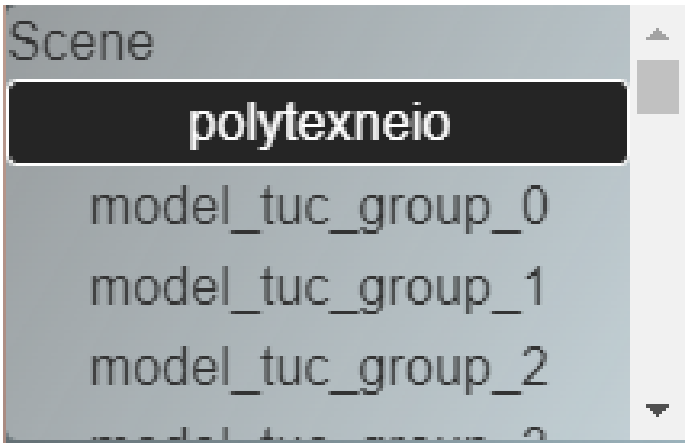

Figure 3.24: Scene Panel

The Scene Panel displays an overview of the elements within the scene. If a loaded file consists of multiple objects, the filename is presented on the scene panel too, serving as a merge button. This feature enables users to manage both the entire

object and its individual components. When a user presses the merge button, it turns black to indicate that the object is in merge state. Then if users want to manage its components, they can click on one of them in the Scene Panel and split the object, returning merge button to its initial color. This functionality was implemented utilizing the BufferGeometryUtils module provided by Three.js library, that enables the merge of multiple geometries into one.

- **Save Workspace :**

Figure 3.25: Save Workspace

The 'Save Workspace' button serves as a temporary storage solution for a user's session. In Chapter 3.2 we mentioned the significance of this feature for our system. Delving further into its implementation, our web server utilizes an object capable of carrying the binary data saved by users. With each save action, the previous data is overwritten. Each user's data is stored within this object under a room key, associated with their unique URL which is assigned upon authentication. This unique URL serves as an identifier for the specific room the user interacts with ensuring that saved data is correctly associated with the corresponding room. This feature was implemented in response to the limited space of 5MB in the browser's local storage. However, to address the need for rebuilding our scene panel upon data retrieval, we utilized the browser's local storage for saving small-sized data necessary for this process. If the user logs out their saved data is erased from our server.

- **Clear Scene :**

Figure 3.26: Clear Scene

'Clear Scene' button enables users to clear the 3D environment, providing a convenient alternative to individually erasing each object, facilitating the start of new projects.

3.5.2 Manual of 3D Editor On Teams Mode

Given the similarity of the interface to the standard 3D editor, we will focus on the additional features specific to Teams Mode.

3.5.2.1 Online/Offline Panel indicator

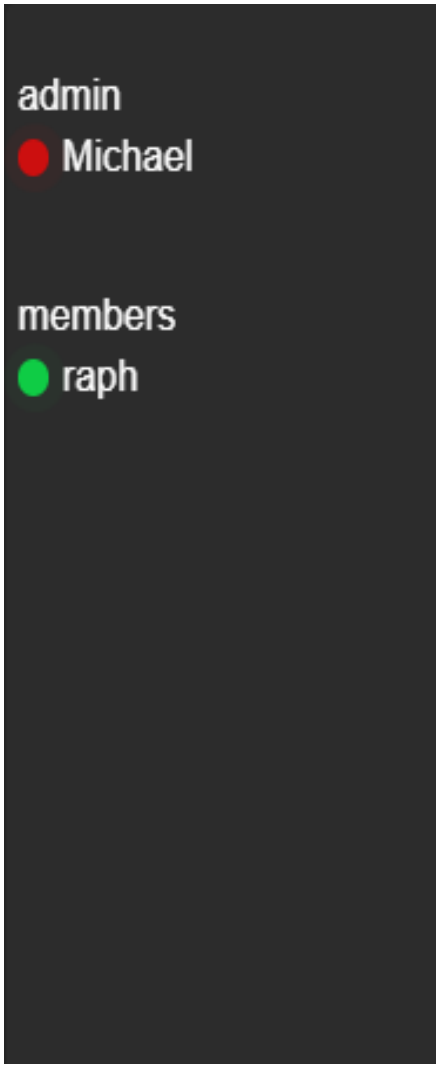

Figure 3.27: Online/Offline Panel indicator

The interface depicted in figure 3.27 presents the Online/Offline Panel indicator. This panel shows real-time updates on whether collaborators are online and actively editing the project or offline and unavailable for collaboration.

3.5.2.2 Save Workspace Usage On Teams

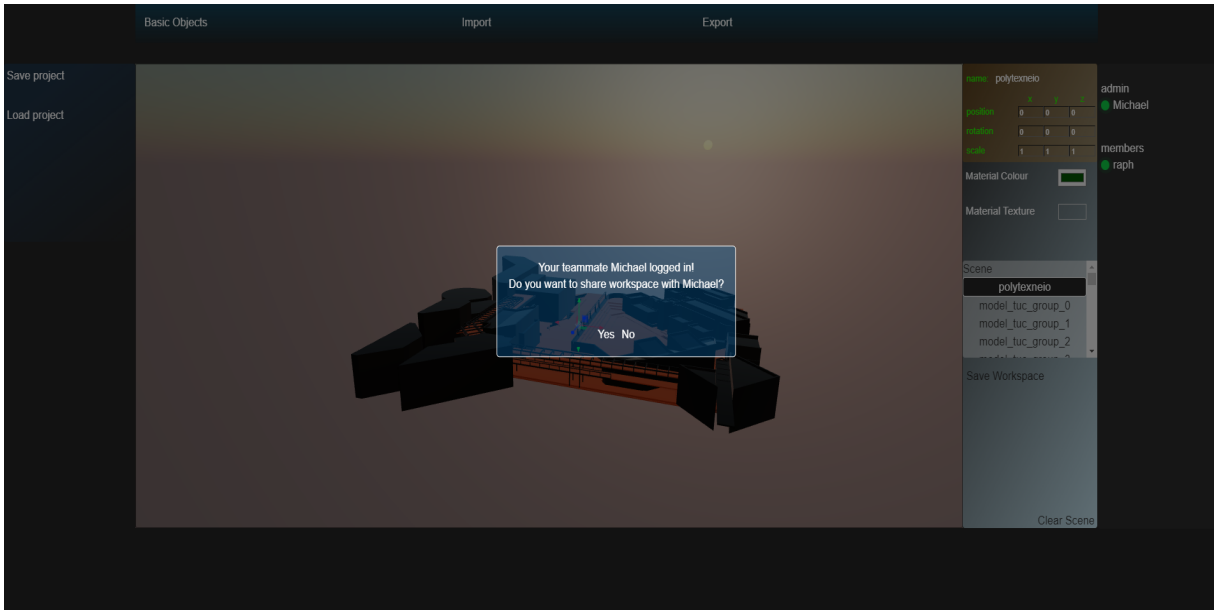

Figure 3.28: Save Workspace Usage On Teams

If a user comes online while another team member is actively editing, we prompt the online members to share their workspace with the newly online user. By pressing yes, server utilizes the Save Workspace button and transmits the data to the other user. This approach ensures that all team members are up-to-date with the design process and can seamlessly contribute to the project.

3.5.3 Manual of Home Page

Below, we will examine the components constituting the home page.

3.5.3.1 Post Box

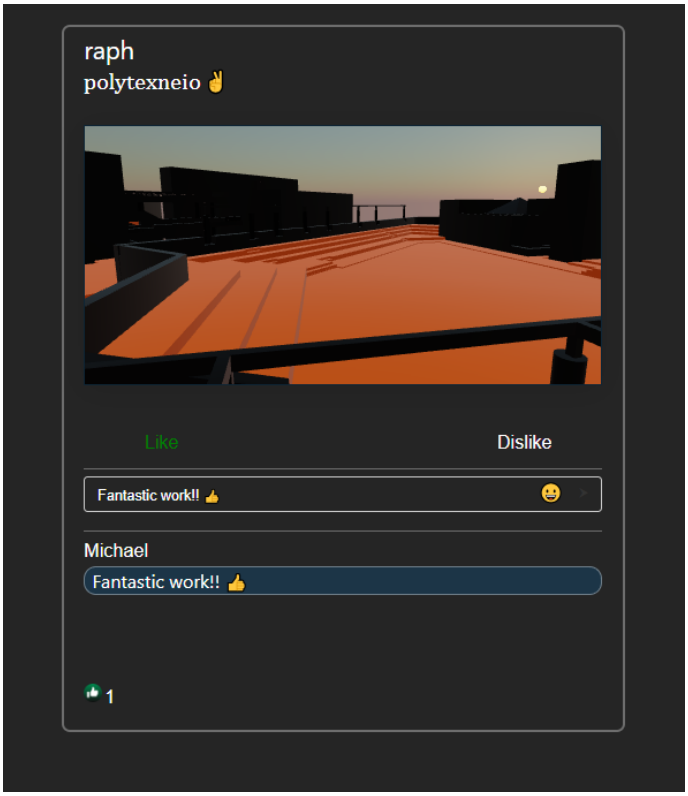

Figure 3.29: Post Box

Figure 3.29 presents a post on home page. This post is a rectangular window that contains the designer's name, a description relevant to its design, the shared project in a small canvas enabling users to navigate within, a like and a dislike button, a field enabling users to comment and use emojis and a field where comments and likes are displayed. These actions are updated in real-time for all connected users. If a user likes a post, like area goes green as shown in the figure. We have enabled users to like a post once with the intention of providing genuine feedback on the community's reactions. If they like the post again, the like toggles between green and white and the number of likes displayed below, increases or decreases accordingly.

Chapter 4

Implementation

4.1 Introduction

The implementation of the information system detailed in this thesis, was orchestrated utilizing the Node.js JavaScript runtime environment. This choice was primarily driven by its capability to manage numerous simultaneous connections and input/output operations with efficiency. Additionally, Node.js was selected for its inherent scalability and real-time capabilities, which are key components for our system ensuring the required level of responsiveness and performance.

Express.js was utilized to set up the initial configuration of our web server and create our RESTful API, that is responsible for routing requests and managing data exchange between the client and server components of our application. Leveraging Express.js we were able to create endpoints and middleware to efficiently handle various functionalities such as authentication, validation and error handling, facilitating seamless communication and interaction between the client-side and server-side components of our system.

MongoDB serves as our database due to its flexible document-based model, which aligns perfect with the nature of the data that our system needs to handle. GridFS, a tool provided by MongoDB, was utilized to surpass the limitation of 16 MB per save, an initial constraint of MongoDB, ensuring efficient storage and retrieval of large files within our database.

To achieve 3D visualization, Three.js was integrated into our system, enabling navigation

and editing of 3D space. Utilizing its rich set of modules, we were able to implement various functionalities that enrich user's 3D editing and visual experience within the application. To enable this capability we leveraged the EJS view engine, empowering us to generate dynamic HTML pages. This allowed for the integration of Three.js components, while also enabled dynamic retrieval and visualization of back-end data to users. Opting for vanilla JavaScript instead of integrating additional frameworks aided us in maintaining a lightweight and efficient front-end architecture, minimizing dependencies, constraints and potential overhead ultimately contributing to a smoother user experience.

Socket.IO enabled real-time communication between a client and our web server. This way our application becomes more responsive and interactive, allowing users to receive instant updates and notifications. Furthermore, it empowered us to provide users with the capability to create teams for collaborative work within dedicated rooms. Leveraging sockets, the system efficiently supports real-time data sharing among team members, fostering collaboration.

4.2 Back-End

4.2.1 Node JS Web Server

The implementation started by setting up a Node.js web server. Initially, Node.js was installed on our computer. Visual Studio Code provided the development environment for writing and managing the code, while Node.js was responsible for executing JavaScript and running the web server.

After installing Node.js, the Node Package Manager (npm) becomes available. Node Package Manager is a powerful tool that comes bundled with Node.js and serves as a package manager for JavaScript. It allows us to easily install, manage and update dependencies and packages required for our application. When we install a package using npm, it automatically resolves and installs any dependencies required by that package. It also generates a package-lock.json file to lock down the version of each installed package, ensuring consistent behavior across different environments. Node Package Manager utilizes a package.json file to store metadata about the project and its dependencies. This file

contains information such as the project name, version, description, entry point, dependencies and scripts. It serves as a central configuration file for npm-managed projects. Nodemon package, automated the process of running our server by continuously monitoring our files for changes and restarting the server automatically, making the development workflow smoother and more productive. Without it, if we made changes to our code, we would need to manually stop the server and then restart it to see the changes take effect. This manual process can be time-consuming and inefficient, especially during development when frequent code changes are made.

With the server environment prepared, the foundation was laid for integrating Express.js, MongoDB, EJS views and our static files, along with numerous modules from npm to activate crucial functionalities for our system.

4.2.2 Server Components - Integration Steps

1. The integration process started with the import of Express.js framework and the utilization of its fundamental methods (use,post,get,put,delete,set,listen etc.). These methods enabled us to mount and activate the functionality of imported modules within our application like body-parser, that provides our web server with the ability to parse incoming HTTP request bodies and form data, make use of express.static to serve static files to clients, as well as many others, that we will further examine below. We were empowered to establish application-level configurations, such as specifying the location of our views or setting our view engine, initiate our server connection and define our route handlers to create our RESTful API that serves clients by handling incoming requests, performing CRUD operations and responding with appropriate data or messages.

2. After setting up Express.js we continued with the establishment of the MongoDB connection using Mongoose, a popular Object Data Modeling (ODM) library for MongoDB and Node.js. Mongoose simplifies the interaction with MongoDB by providing methods and a structured schema-based solution. This enabled us to define data models using schemas according to our application's specific requirements and by using Mongoose methods, we could manage them. Schemas, serve as

blueprints for organizing and representing data within MongoDB collections, allowing for precise control over the data structure. To handle large-sized files efficiently, we integrated GridFS, a MongoDB tool designed to serve this purpose. GridFS, allows us to manage and serve large-sized files. When transmitting a large-sized file, it's essential to consider the limitations and constraints of the HTTP protocol. HTTP requests, have restrictions on the size of the data being sent in the request. Multipart form data, is a specific format designed to overcome these limitations by allowing a file to be divided into multiple parts, each with its own headers and content. This format is well-suited for transmitting large-sized files because it enables the file to be split into smaller chunks, making it easier to handle and transmit over the network. By using multipart form data, large-sized files can be transmitted more efficiently, as the file data splits into smaller parts, reducing the risk of exceeding size limits imposed by the HTTP protocol or intermediary systems (such as proxies or firewalls). For handling multipart form data file uploads, we utilized Multer, a middleware designed specifically for this purpose in Express.js applications. Multer, seamlessly integrates with GridFS, enabling efficient processing and storage of large-sized files uploaded by users.

3. Sessions were employed to manage user interactions within our web application, leveraging express-session middleware. Express-session, facilitates the handling of user sessions by storing session data on the server-side and associating each session, with a unique identifier that allows server to identify and associate incoming requests with the corresponding session data, enabling interaction with the user across requests. Sessions are further utilized to create and manage unique URLs for users, addressing the issue of unauthorized access to specific parts of our application. Unique URLs, serving as our access control mechanisms, while also helping us in some processes to identify and manage individual sessions serving as an alternative to session IDs. This approach facilitates personalized content delivery and more importantly, acts as a mechanism for restricting access to unauthorized users. Users are required to access the application through their unique URL, that is obtained after authentication, ensuring that only authenticated users have access to their sessions and associated data, enhancing the security of our web application.

Authentication was implemented using passport module, a popular authentication middleware for Node.js provided by npm. This package enables the implementation of various authentication strategies. The configuration involves defining strategies, serialization and deserialization functions. The LocalStrategy module was utilized for authenticating users based on their username and password stored in the database. After a login request, LocalStrategy module receives the username and password submitted by the user, checks the database's collection where users are stored and validates the data provided. Serialization is the process of converting a user object into an ID which is stored in the user's session data. In our case, after successful login, we serialize a user object into its MongoDB ID. This ID field, serves as a unique identifier for each user record in the database. By using this ID as the serialized representation of the user, the session data can efficiently reference and retrieve the user's information when needed. When we need to retrieve the user's information during the session, we can use this ID to efficiently fetch the corresponding user record from the database utilizing the deserialization method. This method reverses the above process, converting the user ID back into a user object. It is called, whenever a request is made to the server, allowing passport module to retrieve the user's information from the database, based on the ID stored in the session data during serialization.

4. In this step we'll list both the imported modules mentioned previously and those not yet referenced, describing the functionalities they enable for our web server.

   - **body-parser middleware:** body-parser, enabled our web server to parse incoming HTTP request bodies and form data, which is essential for handling data submitted through forms or API requests.
   - **path module:** path API integration allows us to join directory paths, making them accessible and visible to our server. This functionality, is essential for accurately specifying the location of files and directories that need to be served or accessed by the server.

- **mongoose module:** mongoose, is an ODM (Object Data Modeling) library for MongoDB. It provides a straightforward, schema-based solution for defining data models, enabling us to structure our application data and includes methods for interacting with our MongoDB database.
- **fs module:** fs API, also known as the file system module, provided us with the ability to interact with the file system on the server. It's useful for tasks like reading/writing files, working with directories and managing file-related operations on the server-side. In our case we utilized it to read the key and certificate files necessary for configuring HTTPS. Our application, primarily deals with files on the client-side. Therefore, we utilize the FileReader API for reading files. This built-in in browsers API provides client-side functionality for reading files asynchronously in JavaScript. It enabled us to handle file inputs from the user interface and process the contents of those files directly within the browser, without the need for server interaction.
- **bcryptjs module:** bcryptjs was utilized for hashing passwords, based on the bcrypt algorithm, before storing them into our database and compare hashed passwords during the authentication process.
- **Socket.IO module:** Utilizing the Socket.IO module on our server, enabled real-time bidirectional communication between our server and connected clients.

4.3 Front-End

4.3.1 EJS (Embeeded JavaScript)

After setting up our web server environment, we integrated the EJS template engine into our development stack. This decision, empowered us to create dynamic HTML pages according to user-specific data, bridging our back-end and front-end development. In simple terms, what we achieved is that when a user requests web page, we perform server-side processing to generate the HTML resulting to faster page loads in contrast with plain HTML and client-side processing to generate the page and retrieve user's data from our database. Additionally, we are able to retrieve data from our database rendering them along with the page on the client-side. Server-side rendering (SSR), empowered us

to efficiently generate dynamic HTML pages that incorporate server-side data, enhancing the user experience by minimizing processing overhead on the client-side. Additionally, EJS supports the use of partials, which are reusable components that can be included in multiple pages of a website. This promotes code reusability and simplifies maintenance by avoiding duplication of code. In contrast, plain HTML lacks such built-in mechanisms for code organization and reusability, often resulting in increased complexity and inefficiency as a project grow in size.

4.3.2 Three.JS - Significant Modules

In the preceding chapters we delved into the features, functionalities and workflow of Three.js, exploring its capabilities. Below, we quote the most significant modules that were utilized after importing the Three.js library, providing essential features and functionality to our project.

- **WebGLRenderer:** The core renderer for Three.js, responsible for rendering 3D scenes using WebGL, ensuring high-performance rendering across platforms and devices.
- **GLTFLoader:** Enables loading models of the GLTF format, facilitating interoperability and efficient rendering of 3D models.
- **OBJLoader:** Supports loading models of the OBJ format, a widely used format for exchanging 3D assets.
- **MTLLoader:** Enables the loading of materials and textures defined in the MTL format, ensuring accurate rendering of complex materials.
- **TextureLoader:** Provides functionality for loading and handling textures, adding realism and detail to 3D scenes.
- **DRACOLoader:** Integrates Draco compression for efficient loading and rendering of 3D models, reducing file sizes and enhancing performance.
- **GLTFExporter:** Allows for the exporting of 3D models in the GLTF format, ensuring compatibility with various platforms and applications.

- **STLExporter:** Facilitates the export of 3D models in the STL format, commonly used in 3D printing and computer-aided design (CAD) applications.
- **BufferGeometryUtils:** Used specifically for merging geometries in our project. It provides methods like mergeBufferGeometries(), which allows us to merge multiple buffer geometries into a single geometry.
- **TransformControls:** Provides interactive manipulation controls for transforming objects in 3D space.
- **OrbitControls:** Enables intuitive camera control for navigating and exploring 3D scenes.
- **SpotLight:** Represents a light source that emits light in a specific direction from a single point, typically used to create focused or spotlight-like effects in a scene.
- **HemisphereLight:** Provides a light source that simulates ambient light coming from all directions, typically used to create more natural lighting environments in 3D scenes.

4.3.3 General Client-Side Significant Modules and Interfaces

- **UndoManager:** The UndoManager module provides functionality for implementing undo and redo functionality in web applications. It allows users to revert changes made to data or content. By managing a history of actions, the UndoManager enables users to navigate through their changes and revert to previous states.
- **FileReader:** FileReader interface is a part of the File API in JavaScript provided by web browsers, allowing web applications to asynchronously read the contents of files stored on the user's computer. It is commonly used for tasks such as uploading files, processing file contents, and displaying file data within web applications. The FileReader interface, provides methods for reading various types of files, including text files, binary files and images, making it a versatile tool for handling file input and output in client-side JavaScript applications.

- **Sky.js:** Sky.js is a JavaScript module that provides functionality for creating dynamic and interactive sky and atmosphere effects in 3D web applications. It offers features such as realistic sky rendering, dynamic lighting effects, cloud simulation and more, enhancing the visual realism of 3D scenes.

- **Blob:** Blob interface represents raw binary data, typically used for handling file data or other binary data in JavaScript. It provides methods for creating and manipulating blobs, as well as properties for accessing information about the blob's size, type and other metadata. When working with files in web applications, the FileReader API is used to read the contents of files (if they are large-sized, otherwise we can use data URIs) selected by the user. However, FileReader requires a Blob object as input in order to read the file's contents. This means that when a user selects a file using a file input element, we need to create a Blob object representing the selected file which can then be passed to FileReader for reading. Using the URL.createObjectURL() method, we generate a unique URL that can be used to reference the binary data stored in the Blob object. This is a convenient way to represent these binary data within the application and display media, download files, as well as to optimize the performance since we don't need to encode binary data into the HTML of our web page. Instead, we can reference to the Blob data using this URL, which can improve performance and reduce memory usage, especially for large binary data. Additionally, the alternatives such as data URIs can represent limited amount of data. Using Blob URLs we exceed this limitation. However, while Blob URLs do not have inherent limitations on the size of data they can represent, it's important to note that Blob data is still subject to browser and system limitations. Extremely large Blob objects, may still encounter performance issues or memory constraints, depending on the browser and system resources available.

- **LocalStorage:** The LocalStorage interface, provides a simple key-value storage mechanism that allows web applications to store data persistently on the client-side. It is typically used for storing small amounts of data such as user preferences, session data or application settings. LocalStorage provides methods for reading, writing and deleting data stored in the local storage area of the web browser.

Chapter 5

Conclusion

5.1 Introduction

This thesis discusses the development of an information system, harnessing web technologies and innovative approaches to enhance the efficiency and streamline participatory design processes in urban planning and architecture. Throughout the document, the rationale behind the choice of the components that constitute the system's structure has been analyzed, along with its features and the steps toward its implementation. Nevertheless, significant work and research remain necessary for the potential real-world application of a similar system. This system would leverage all cutting-edge technologies to provide an optimal environment that engages stakeholders and inspires broader participation. This concluding chapter encompasses the deployment of the platform, initially as an indicator of the application's robustness and reliability, as well as for experimental purposes under real-world conditions. This section also includes an analysis of the system's advantages and disadvantages, along with discussions on potential avenues for future research and development within the realm of participatory design in urban planning and architecture. By identifying the future exploration areas, this thesis endeavors to inspire continued research and development within the field, ultimately contributing to the enhancement of this architectural design model.

5.2 From Local Testing to Live PaaS Deployment

In order to evaluate the functionality and efficiency of our platform under real-world conditions, we implemented the essential modifications to deploy our Software-as-a-Service (SaaS) to a Platform-as-a-Service (PaaS) provider called Render.

PaaS is a cloud computing service model that provides a platform allowing users to develop, run and manage applications without the complexity of building and maintaining the underlying infrastructure. With PaaS, developers can focus on coding and deploying applications, while the PaaS provider handles tasks such as infrastructure provisioning, scaling and maintenance. This model offers greater agility, scalability and cost-effectiveness compared to on-premises infrastructure, enabling accelerated application development and deployment processes.

Deploying an application from local-host to a PaaS, introduces challenges for development teams due to potential differences in code behavior and performance compared to the local development environment. These differences can arise from variations in environment configuration, dependencies and libraries and the underlying infrastructure.

The deployment proceeded smoothly, as we only had to update some library versions and adjust the URL paths to align with our domain name. Then our SaaS was successfully deployed and became accessible, indicating a successful transition from development to deployment. It's essential to note that we opted for the free version which imposes restrictions on available resources, so extensive testing under high traffic conditions was practically unfeasible. As a result, our observations were derived from a scenario involving a limited number of users. We were allocated 0.1 of the PaaS's CPU power and 512MB of RAM. When a user sends a request to the server, the server typically retrieves the necessary data from storage, such as a database or file system and loads it into RAM for processing. The server's RAM plays a crucial role in facilitating data processing by providing fast access and serving as temporary storage for data and program instructions during execution. Having sufficient RAM ensures that the server can handle multiple concurrent requests efficiently without running out of memory, leading to better performance and responsiveness of the web application. The CPU accesses and executes the

instructions stored in RAM, to perform various tasks such as processing requests, executing server-side logic and generating responses to be sent back to users. Both RAM and CPU, work together to ensure a smooth and efficient operation of applications and overall system performance. Insufficient RAM or CPU resources, can lead to performance bottlenecks, slowdowns or application crashes. Therefore, having an optimal balance of both components is crucial for maintaining an application's reliability and performance. Furthermore, it's important to acknowledge that a user's device significantly impacts the performance of web applications too, as it handles client-side operations such as rendering web pages, executing scripts and handling user interactions. Just like the server's RAM and CPU, the resources available on the user's device, including RAM, CPU and GPU, play a pivotal role in determining the application's responsiveness and overall user experience. Insufficient resources on the client-side, can lead to slow page load times (SSR can mitigate this issue, as we discussed in the previous chapters), unresponsive interfaces and degraded performance.

While testing our application in teams mode, we noticed a worth mentioning delay in data transmission that was not evident when running the application on localhost. We observed that, the larger the volume of data to be transmitted within the socket connections, the longer the delay until the transmission completes. Exceeding an upper threshold of 20MB per file import, the socket connections were closing and reopening instantly resulting to communication breakdown. Both issues could be influenced by the limitations of the network bandwidth provided by the PaaS. If the network bandwidth is limited, it could result in slower data transmission rates, especially when dealing with large volumes of data. Moreover, the PaaS provider may impose restrictions on the total amount of data that can be transmitted within a certain time frame to ensure fair usage and prevent network congestion. We encountered the same behaviour in our temporary storage functionality, which was also implemented with socket connections to transmit and temporary save user's data on our server.

Limited CPU power and RAM could indirectly contribute to these issues. For example, if the server's CPU is under-powered it may struggle to efficiently process

incoming and outgoing data, leading to bottlenecks in data transmission. Similarly, insufficient RAM could result in slower processing speeds or the inability to handle large volumes of data effectively, which could impact data transmission performance. However, the direct impact of CPU and RAM limitations on these issues may not be as significant as factors like network bandwidth or platform restrictions.

We also observed that when importing and transmitting files below 20MB in our Three.js scene, when their cumulative size exceeded 45MB, at times 96MB, 64MB or even 32MB we encountered the same behaviour described above. This variety in the accepted cumulative size without the connection being closed, is most possibly due to the change of traffic on the PaaS. During the transmission process, the data being sent and received temporarily occupies system memory. However, once the transmission is complete the memory used to store the transmitted data is released, freeing up resources for other tasks. So, after the transmission ends, the transmitted data no longer burdens the RAM although it's worth noting that the process of managing socket connections and transmitting data may still consume system resources, including CPU cycles and network bandwidth, especially if the socket connection remains open for further communication like ours. Another reason for this, is that the PaaS may have limitations on the total amount of data that can be transmitted over a single connection within a specific time frame as we previous mentioned. Even though each individual file is within the acceptable size range, the cumulative data transferred may exceed these limits after multiple transmissions, leading to the closure of the connection.

Beyond these differences between the two environments (local-host and PaaS), the rest of our application functionalities worked as expected. These observations, led us to the conclusion that if we deploy our application and are allocated with more resources, our SaaS will work sufficiently.

5.3 Platform's advantages and disadvantages

5.3.1 Advantages

- **System scope:**

1. Our platform was implemented as a web application to broaden accessibility, a crucial factor for an efficient participatory design and has significant cost, time and maintenance benefits.
2. Node.js provided us with a powerful environment that enables the implementation of a high-performance, scalable and real-time application.
3. EJS enabled server-side rendering, resulting to faster page loads. By rendering HTML on the server and sending it to the clients, EJS minimized the time users spent waiting for content to appear, ultimately improving their overall experience.

Through the integration of these technologies, our development process successfully met our system's requirements.

- **User scope:**

1. The platform offers a comprehensive set of essential features, similar to those found in professional 3D space editors. However, it stands out by prioritizing user-friendliness, ensuring that non-professionals can easily navigate and utilize its functionalities. By omitting complex features that could potentially confuse users, the platform focuses on enhancing engagement and empowering individuals of all skill levels to create and edit 3D content effortlessly. These features, include 3D model and image import, basic object insertion, color and texture management, clone and delete functionality, undo/redo capability, object and scene information display, as well as storage capabilities.
2. Facilitates collaborative design by enabling team creation, further enhancing its utility. This feature sets it apart from other 3D editors, marking a significant step forward in real-time collaboration during a design process among users.
3. By providing a page where designers can share their design proposals and every user can seamlessly navigate through them and provide valuable feedback, the platform enhances the participatory design model. This fosters a sense of community among stakeholders, where different perspectives unite and contribute toward the end result. Such collaborative interaction not only enriches

the design process but also promotes inclusivity and encourages collective creativity, ensuring that all voices are heard and valued. This, results in more diverse and innovative outcomes, reflecting the true essence of collaborative design.

By combining the features mentioned above with user-friendly design and innovative collaborative tools, the platform fosters a sense of community and empowers individuals and teams alike, to create and edit 3D content effortlessly regardless of their expertise.

5.3.2 Disadvantages

- **System scope:**

  1. Node.js's utilization comes with its trade-offs like every technology. It's not the best choice for server-side CPU-intensive tasks such as cryptographic operations, image or video processing, data compression or decompression, complex mathematical calculations, data analysis, rendering 3D graphics or machine learning due to its single-threaded nature. In such cases, other platforms like Python or Java may offer better performance. Another significant disadvantage of Node.js, is its memory consumption. Node.js applications can consume more memory compared to traditional server-side technologies like PHP. This increased memory usage, is attributed to the overhead of managing asynchronous operations and event loop concurrency. Therefore, it's essential to allocate adequate resources to ensure that the application operates smoothly without encountering memory-related issues, leading to increased infrastructure costs.

  Since there is no need to perform CPU-intensive operations on our server and our system's requirements align perfectly with Node.js's key features, the increase in memory consumption is a manageable trade-off. By allocating additional resources to our system, we could ensure that it functions as intended, allowing us to effectively leverage Node.js's strengths.

- **User scope:**

1. Not all user's personalized data is being server-side rendered through EJS. This is because utilizing client-side APIs for data retrieval is also an essential knowledge and therefore, they are also employed, allowing us to further develop our expertise and practice in their utilization sacrificing some of the data retrieval speed.
2. While Node.js and Three.js are both cross-browser compatible technologies, there are slight differences on the platform's styles among them since our attention was not really focused in the styling to ensure consistency across all browser platforms. Therefore, some adjustments in our CSS instructions would be needed to accommodate any variations in rendering among different browsers.

5.4 Future Work

The information system proposed in this thesis, is an approach to the implementation of an environment, capable of fostering the participatory design on urban planning and architecture through its features. However, additional capabilities are necessary for the system to provide its users with all the tools that could facilitate and inspire design processes, whether by unlocking Three.js's full potential or creating useful interfaces. Then, it is quite interesting to also examine technologies that could be incorporated into our system, allowing us to envision new paths for the application's usage.

Below, we propose some addons that are essential for the system's optimal utilization, as well as others that we find interesting for enhancing the system from the user's perspective.

- **Assets library :**

  An assets library within the 3D editor would serve as a resource repository, offering a diverse array of pre-made 3D models, textures, materials, animations and lighting setups. Enabling users to skip time-consuming or complex tasks

such as creating models from scratch, encourages creative exploration and facilitates the engagement for all skill levels. An essential resource, as it also enhances the efficiency elevating the overall quality of projects.

– **Geographic Information Systems (GIS) :**

Combining GIS spatial analysis with the 3D visualization capabilities of Three.js, could offer a versatile platform for various applications including urban planning and architecture. By integrating GIS into a Three.js software, urban planners and architects gain the ability to visualize, examine and optimize design decisions in real-world conditions. This visualization capability, facilitates decision-making, design's optimization and collaboration among stakeholders. In [21], Three.js was evaluated for its efficiency and is considered a valid solution. While Three.js itself does not provide direct support for parsing GeoJSON or KML files, there are libraries to parse and convert them into Three.js geometries for visualization. Three.js lacks of built-in streaming capabilities, however, researches were able to develop their own streaming algorithm and memory management rather easily leveraging WebGL API. When visualizing large geographic areas with detailed terrain or imagery, such as satellite imagery, the volume of data can be large. Without built-in or custom streaming capabilities, Three.js may struggle to load and render these data efficiently. In [22] Three.js was combined with the Cesium.js library to leverage the specific features of both libraries. Cesium aims directly at geospatial applications and compared to other frameworks it can stream terrain data and imagery. It is also the only Open-Source framework with direct support for geospatial coordinates. The current release 1.7 includes basic support for KML, the main file format of Google Earth. Thus, users easily integrate data created or exported from Google Earth into their Cesium-based applications. Three.js offers extensive capabilities for custom 3D graphics, animations and interactions. Combining the two libraries, enables the creation of hybrid visualizations that incorporate both geospatial and custom 3D elements. Overal, GIS can enhance the capabilities of urban planning and architecture applications, enabling stakeholders to create more sustainable and livable cities.

– **Chat and Video call capabilities :**

Communication tools, such as chat and video calls can facilitate collaboration and decision-making processes in urban planning and architecture like they did in [23]. These tools enable stakeholders to engage in real-time discussions, share ideas and provide feedback without switching between different software applications. This feature, streamlines decision-making processes and facilitates knowledge exchange resulting to accelerated project timelines.

– **AR and VR capabilities :**

AR and VR technologies have revolutionized urban planning and architecture in recent years by offering immersive experiences and advanced visualization capabilities. Their effectiveness in facilitating urban planning, is fairly a topic of discussion in several studies. Researchers, explore the impact of these technologies on design processes, highlighting their several benefits. Related work, can be found in [24], discussing about VR effectiveness, and in [25], where AR technology was utilized to visualize 3D layout plans of urban development projects. Integrating AR and VR modules into web applications using Node.js and Three.js, has become increasingly accessible. npm (Node Package Manager) provides a repository of AR and VR libraries and modules that can be easily installed and incorporated into web projects.

– **Game Engine Operational State :**

Although this extension is outside the scope of urban planning, its effectiveness in a participatory design model could be experimented with and assessed. The idea of a software that enables users to form teams and collaborate on game development seems interesting and offers innovative potential. Three.js is suitable for designing games, particularly on the web, therefore, we could leverage its capability by enabling users to engage in collaborative game development, paving the way for a new era of creativity and exploration in game development, under participatory design's principles.

References

[1] Sanoff, H.: Participatory design. DEPARCH Journal of Design Planning and Aesthetics Research **1**(2) (2022) 12 1

[2] Kumar, A., Lodha, D., Mahalingam, A., Prasad, V., Sahasranaman, A.: Using design thinking to enhance urban re-development: a case study from india. Engineering Project Organization Journal **6**(2-4) (2016) 155–165 1

[3] Danchilla, B.: in beginning webgl for html5. Springer (2012) 173203 3

[4] Tilkov, S., Vinoski, S.: Node.js: Using javascript to build high-performance network programs. IEEE Internet Computing **14**(6) (Nov.-Dec. 2010) 80–83 8

[5] Chaniotis, I., Kyriakou, K., Tselikas, N.: Is node.js a viable option for building modern web applications? a performance evaluation study. Computing **97** (2015) 10231044 9

[6] Bangare, S., Gupta, S., Dalal, M., Inamdar, A.: Using node. js to build high speed and scalable backend database server. Int. J. Res. Advent Technol (2016) 6164 11

[7] Plugge, E., Membrey, P., Hawkins, T.: Gridfs. in the definitive guide to mongodb: The nosql database for cloud and desktop computing. Apress: New York, NY, USA (2010) 8395 12

[8] Mardan, A.: Using express.js to create node.js web apps. Apress, Berkeley, CA (2018) 5187 12

[9] Three.js: site. https://threejs.org/ 16

[10] Dirksen, J.: Three.js essentials. Packt Publishing: Birmingham, UK (2014) 16

[11] Socket.IO: site. https://socket.io/ 17

[12] Mardan, A.: Socket.io and express.js. Apress, Berkeley, CA **In: Pro Express.js** (2014) 17

[13] Fall, K., Stevens, W.: Tcp/ip illustrated. Addison-Wesley **volume 1: The Protocols** (2011) 18

[14] Socket.IOProtocol: site. https://socket.io/docs/v4/socket-io-protocol/ 22

[15] Curtis, H.: Flash web design: the art of motion graphics. New Riders Publishing (2000) 24

[16] Matsuda, K., Lea, R.: Webgl programming guide: Interactive 3d graphics programming with webgl. Addison-Wesley (2013) 25

[17] Johansson, J., Folino, E., Bahrehmand, A., Agenjo, J., Blat, J.: Performance and ease of use in 3d on the web: Comparing babylon.js with three.js. (2021) 26

[18] Evans, A., Romeo, M., Bahrehmand, A., Agenjo, J., Blat, J.: 3d graphics on the web: A survey. Comput. Graph **41** (2014) 4361 26

[19] Desprat, C., Jessel, J.P., Luga, H.: 3devent: a framework using eventsourcing approach for 3d web-based collaborative design in p2p. Proceedings of the 21st International Conference on Web3D Technology **10** (2016) 7376 27

[20] Yeoun-Ui, H., Jae-Hwan, J., Myung-Joon, L.: 3devent: a framework using eventsourcing approach for 3d web-based collaborative design in p2p. Proceedings of the 21st International Conference on Web3D Technology (2015) 189–198 27

[21] Kramer, M., Gutbell, R.: A case study on 3d geospatial applications in the web using state-of-the-art webgl frameworks. Proceedings of the 20th International Conference on 3D Web Technology, Ser. Web3D 15, Association for Computing Machinery, New York, NY, USA (2015) 189–197 76

[22] Mete, M.O., Guler, D., Yomralioglu, T.: Development of 3d web gis application with open source library. Selcuk University Journal of Engineering, Science, and Technology 6 (Special) (2018) 818824 76

[23] Nyamsuren, P., Lee, S.H., Hwang, H.T., Kim, T.J.: A web-based collaborative framework for facilitating decision making on a 3d design developing process. Journal Computational Design Engineering **2** (2015) 148156 77

[24] Van Leeuwen, J.P., Hermans, K., Jylha, A., Quanjer, A.J., Nijman, H.: Effectiveness of virtual reality in participatory urban planning. ACM International Conference Proceeding Series (2018) 128136 77

[25] Rohil, M.K., Ashok, Y.: Visualization of urban development 3d layout plans with augmented reality. Results in Engineering **14** (2022) 110 77