VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY
THE INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

WEB APPLICATION DEVELOPMENT
IT093IU

FINAL REPORT

Topic: DrawVoice Meeting

By Group – Member List

No.	Full name – Student name	Student ID
1	Huỳnh Trần Khanh	ITCSIU21011
2	Hồ Tiến Đạt	ITCSIU21047
3	Bùi Phương Thanh	ITITIU21311

Instructor: Assoc. Prof. Nguyen Van Sinh
Msc. Nguyen Trung Nghia

December 2024

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

1.1. ABOUT US

KTD Software Entertainment is an emerging startup specializing in innovative software development, creating engaging entertainment software such as games, multimedia apps, and immersive experiences. With a focus on innovation, the company aims to redefine user engagement by blending creativity, technology, and seamless design to meet the evolving needs of the entertainment industry. Our main clients include small and medium enterprises, individual professionals, and organizations seeking reliable software development partnerships.

Company name	KTD Software Entertainment
Team name	We Can Draw
Business	Draw integrate Voice Chat Web application
Customer	Nguyen Trung Nghia Communication Platform
Contact	123 Linh Xuan ward, Thu Duc district, HCMC, Vietnam
Email	ntnghia@hcmiu.edu.vn
Phone number	+84 (076) 350 3524

Table 1.1 General information of the project

1.2. THE PRODUCT'S INFORMATION

Nowadays, Communication platforms have become essential tools for enhancing real-time interactions and collaboration. With the increasing reliance on digital solutions for both personal and professional use, more people are seeking efficient ways to connect and engage online.

Nguyen Trung Nghia Communication platform is a famous and trusted platform based in Ho Chi Minh City, with a longstanding reputation in design and developing communication platforms for teams and enterprises.

Due to the growing demand for simplified communication and collaboration, we have developed the Draw and Voice Chat Web Application to meet this need. Our platform enables users to engage in real-time drawing and voice chat, making interactions faster, more convenient, and more efficient. With this solution, users can easily collaborate and share ideas from the comfort of their homes via a web-based system accessible on computer. Additionally, our application introduces new features such as the ability to draw as well as check the availability of collaborators in real-time.

1.3. WORK BREAKDOWN STRUCTURE

The primary tasks of our project encompass: Login System. Create room chat, Join room chat, Drawboard and Voice chat.

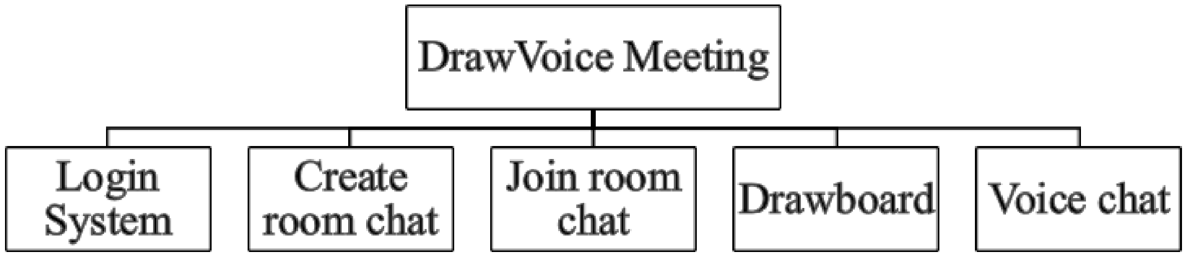


Figure 1.1 Work Breakdown Structure

Each of these tasks is further divided into subtasks, detailing the specific work packages that need to be finished by each team member.

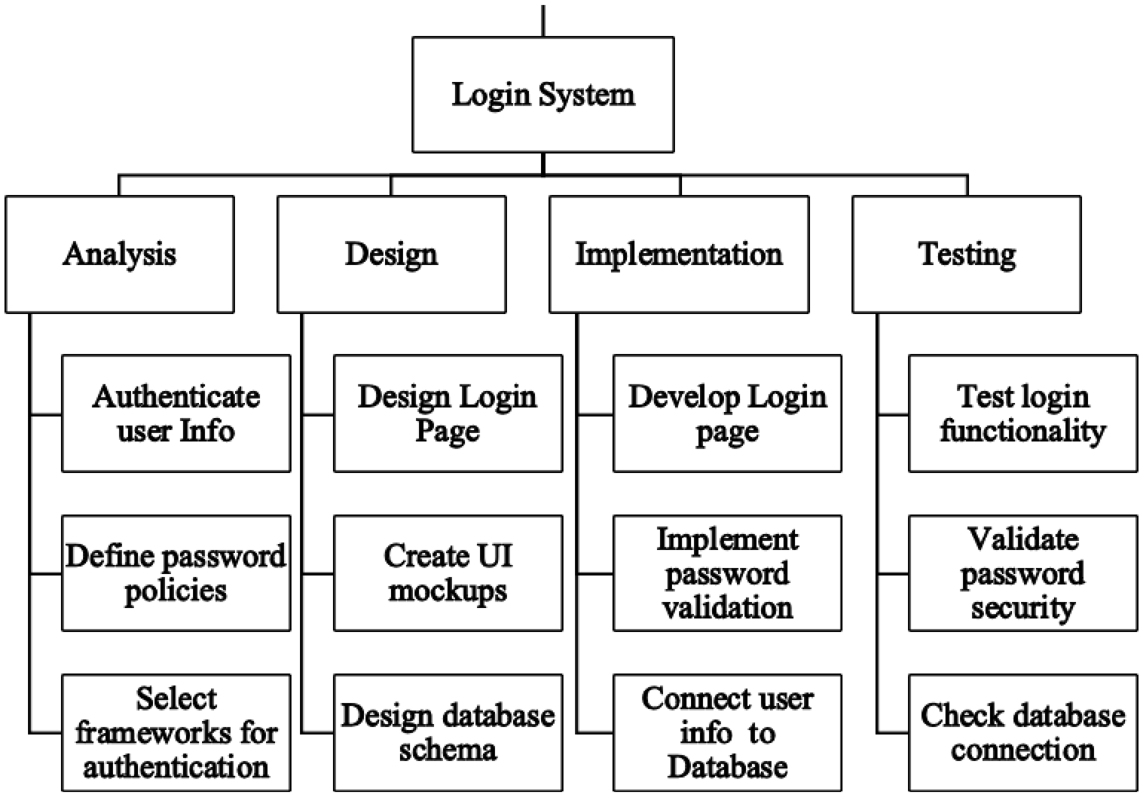


Figure 1.2 Subtasks for Login System

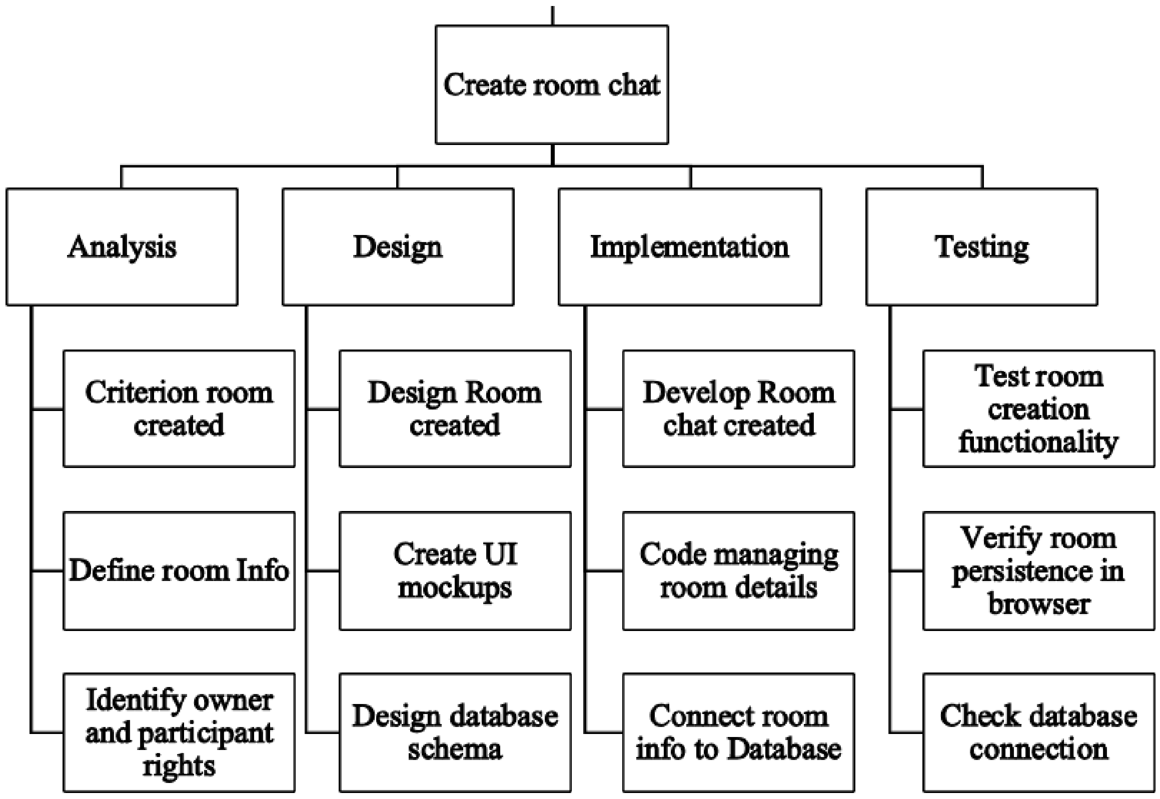

Create room chat
Analysis
Design
Implementation
Testing
Criterion room created
Define room Info
Identify owner and participant rights
Design Room created
Create UI mockups
Design database schema
Develop Room chat created
Code managing room details
Connect room info to Database
Test room creation functionality
Verify room persistence in browser
Check database connection

Figure 1.3 Subtasks for Create room chat

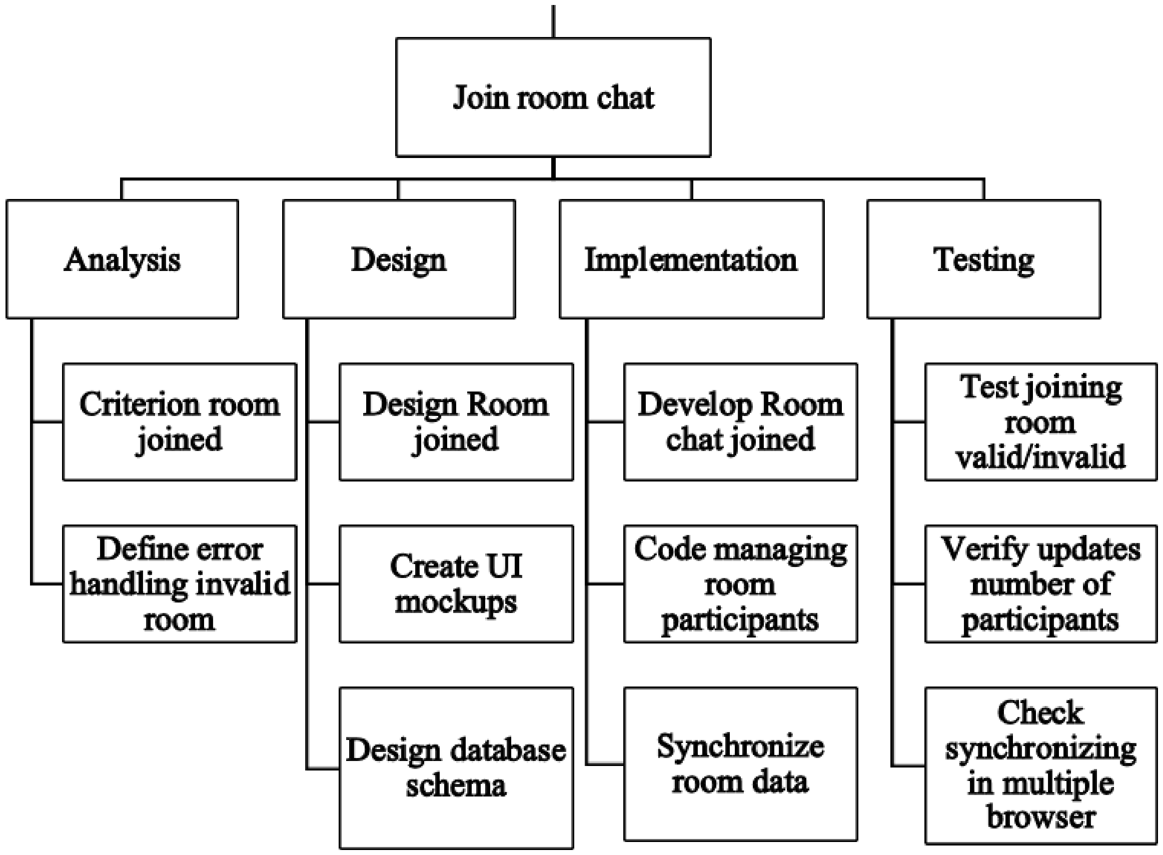

Join room chat
Analysis
Design
Implementation
Testing
Criterion room joined
Define error handling invalid room
Design Room joined
Create UI mockups
Design database schema
Develop Room chat joined
Code managing room participants
Synchronize room data
Test joining room valid/invalid
Verify updates number of participants
Check synchronizing in multiple browser

Figure 1.4 Subtasks for Join room chat

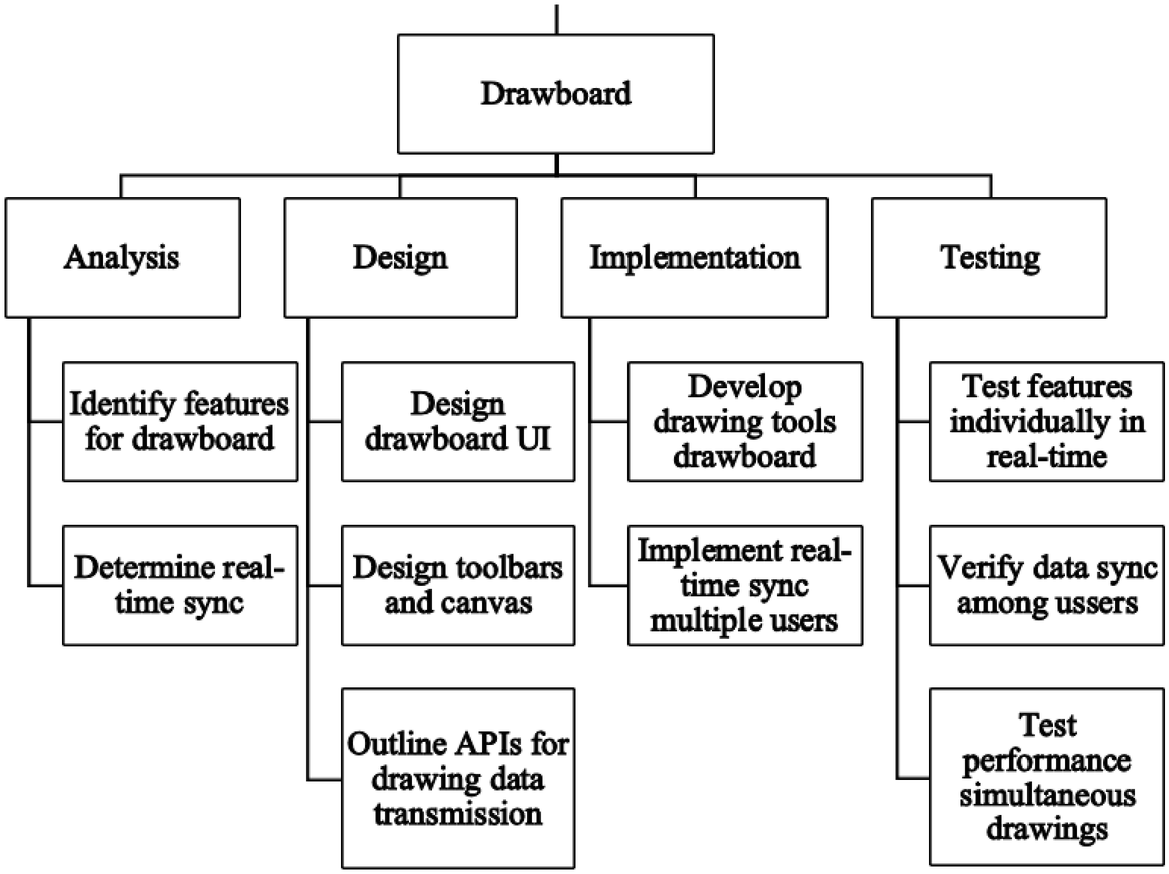

Drawboard
Analysis
Design
Implementation
Testing
Identify features for drawboard
Determine real-time sync
Design drawboard UI
Design toolbars and canvas
Outline APIs for drawing data transmission
Develop drawing tools drawboard
Implement real-time sync multiple users
Test features individually in real-time
Verify data sync among ussers
Test performance simultaneous drawings

Figure 1.5 Subtasks for Drawboard

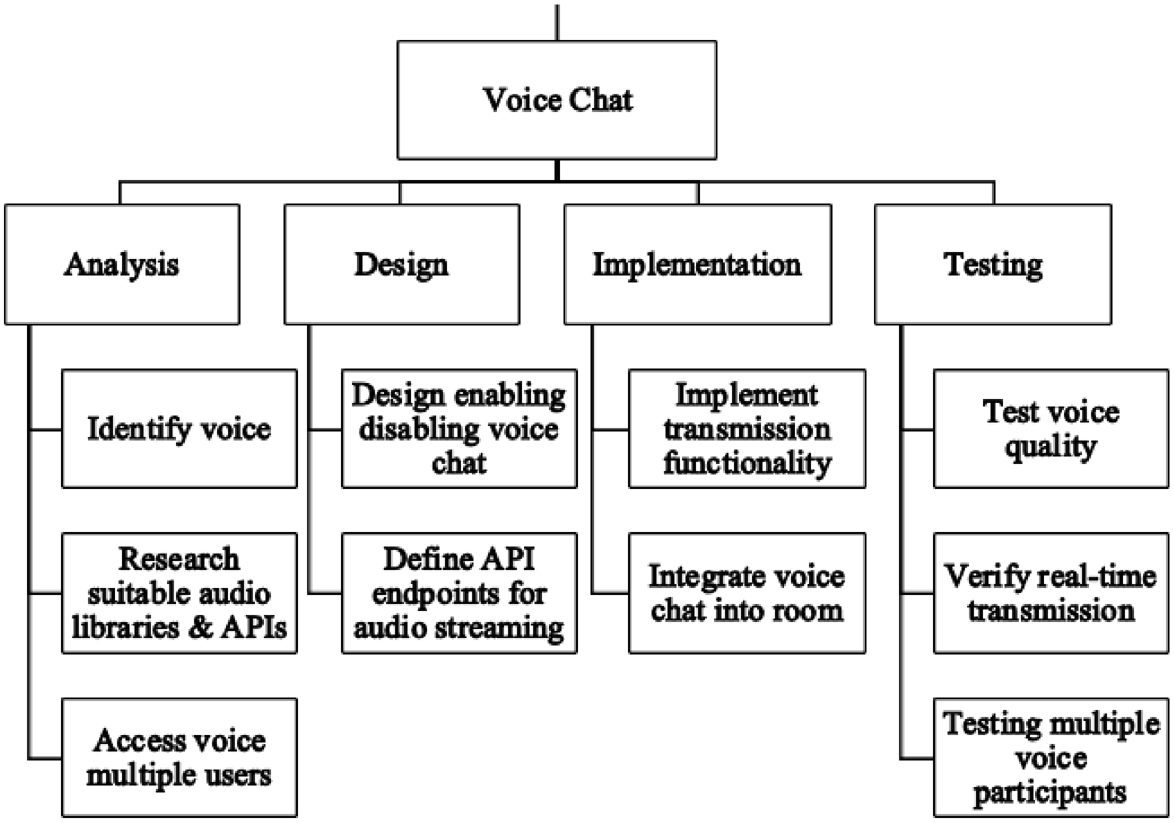

Voice Chat
Analysis
Design
Implementation
Testing
Identify voice
Research suitable audio libraries & APIs
Access voice multiple users
Design enabling disabling voice chat
Define API endpoints for audio streaming
Implement transmission functionality
Integrate voice chat into room
Test voice quality
Verify real-time transmission
Testing multiple voice participants

Figure 1.6 Subtasks for Voice Chat

Following the Software Development Life Cycle, the process includes: Requirement, Analysis, Design, Implementation, Testing, Deployment and Maintenance. However, our project focuses on 4 phases: Analysis, Design, Implementation and Testing in our theory and laboratory Web Application Development class.

1.4. DEVELOPMENT PROCESS

Our team has decided to adopt the Agile process for managing the software development life cycle for our project rather than other processes.

Scrum framework is the most suitable choice of Agile process. A large project is divided into small increments in a sequence of sprints, each which go through several process steps, such as analysis, design, coding and testing: adjustments based on user feedback.

After each iteration, our product is enhanced with improved functionalities and more friendly interface, ensuring that users can interact with the system effectively to meet their needs

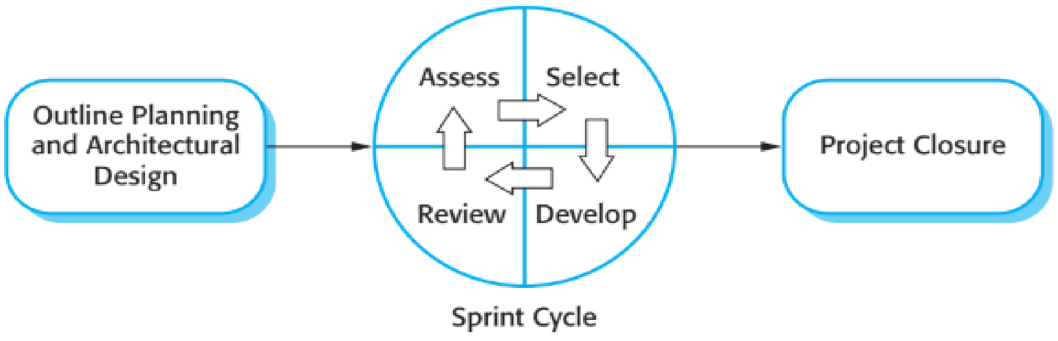


Figure 1.7 The Scrum framework

1.5. DEVELOPMENT ENVIRONMENT

1.5.1. Programming Languages:

- **HTML:** to create the structure of web pages.
- **CSS:** to style and layout web pages.
- **JavaScript:** to add interactivity and dynamic functionality to web pages.
- **Java Server Page:** to build dynamic, server-side web pages.
- **Servlet:** to handle server-side logic in Java-based web apps.
- **MySQL:** to manage relational databases.

1.5.2. Frameworks and Libraries:

- **SQLite:** to manage embedded relational databases
- **WebSocket:** to enable real-time, full-duplex communication between the client and server
- **Jitdi:** to integrate video conferencing and communication tools.
- **Node.js:** as a runtime environment to build server-side apps and manage asynchronous events.

1.5.3. Software Used:

- **VSCode:** to develop front-end and back-end code.
- **Netbeans:** to build Java apps, including JSP, Servlet and connect databases
- **Figma:** to design interfaces and create prototypes collaboratively.
- **Github:** to store and manage code repositories, collaborate with others, and track version control.

1.5.4. Architecture: MVC Model

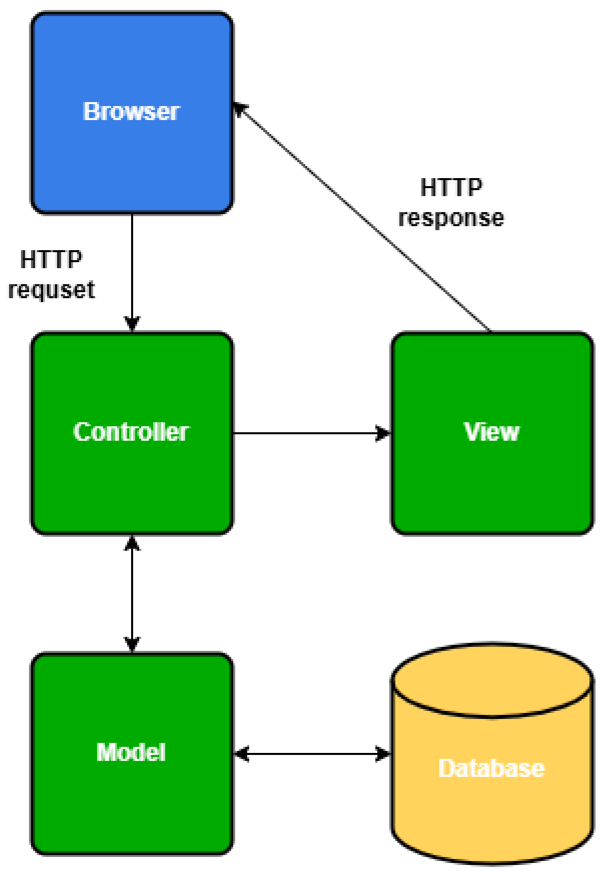


Figure 1.8 Model - View - Controller Model

- **Model** (database layer): handles the application's business logic and is responsible for retrieving the data data from the database, consisting of business objects.
- **View** (UI layer): handles the display of the data and gets needed data attributes from the model, consists of HTML pages and JSPs.
- **Controller** (Business Logic): handles interactions with the user, consisting of serJSPs.

2. REQUIREMENT ANALYSIS AND DESIGN

2.1. REQUIREMENT ANALYSIS

2.1.1. USE CASE DIAGRAM

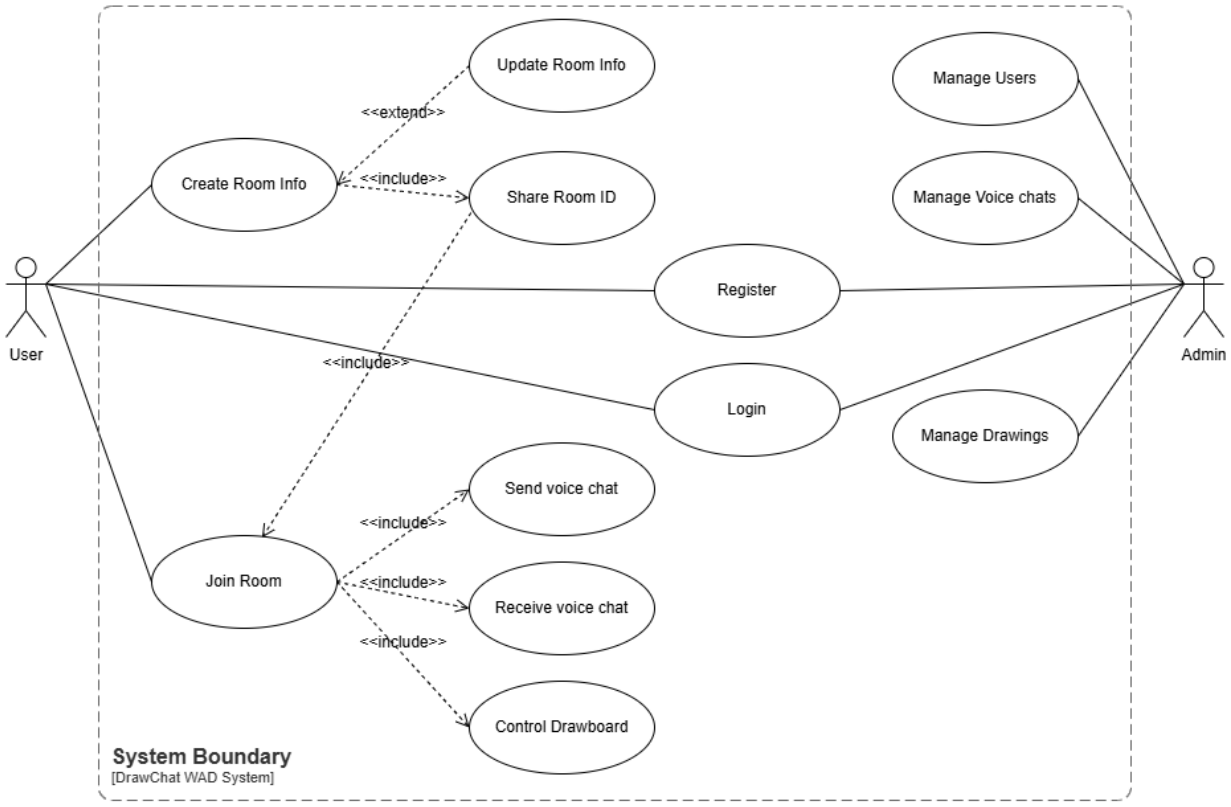


Figure 2.1 Use Case Diagram

2.1.2. USE CASE DESCRIPTION

01. Use Case 1: Register Account

Name: Register Account

Identifier: UC1

Inputs:
1. Account Information (Username, Password, Email)

Outputs:
1. Confirmation message [If successful]
2. Error message. [If fail]

Main Flow:

Actor User	System
1. Access the Register account page.	
2. Enter account information	2. Validate the information
3. Submit	3. Save the information and confirm [if successful]

Table 2.1 UC1 Main Flow

Preconditions:
1. The user is not logged in (for account creation).
2. The user is logged in (for account update).

Postconditions:
1. The user's account is created or updated in the system.

02. Use Case 2: Login

Name: Login

Identifier: UC2

Inputs:

1. Username
2. Password

Outputs:

1. The home page with the user's authorization [If successful]
2. The login page. [If fail]

Main Flow:

Actor User	System
1. Open the login page	1. Display the order status page
2. Enter username and password	
3. Submit	3.1 Check the user's info 3.2 If success, return the home page 3.3 Else return the login page

Table 2.2 UC2 Main Flow

Preconditions:

1. The user has a registered account.

Postconditions:

1. The user is logged into the system.

03. Use Case 3: Create Room Info

Name: Create Room Info

Identifier: UC3

Inputs:
1. Room Information (name, id, description)

Outputs:
1. New room creation confirmation [If successful]

Main Flow:

Actor User	System
1. Select the "Create Meeting" option.	
2. Enter the room details.	
3. Submit	3.1 Creates the room and returns confirmation

Table 2.3 UC3 Main Flow

Preconditions:
1. The user is logged in.

Postconditions:
A new room is created and available for users to join.

04. Use Case 4: Update Room Info

Name: Update Room Info

Identifier: UC4

Inputs:

1. Update Room Info (Name, ID, description)

Outputs:

1. Room info updated. [If successful]
2. Error message. [If fail]

Main Flow:

Actor User	System
1. If user is owner Room ID, the user can update room's info exists in the system	1. Display Page Info room.
2. Submit	2. Confirm updated

Table 2.4 UC4 Main Flow

Preconditions:

1. The user login into the system.
2. The user has the necessary permissions to modify room info.
3. The room need to update which exists in the system

Postconditions:

1. The room info is updated successfully.

05. Use Case 5: Share Room ID

Name: Share room ID

Identifier: UC5

Inputs:

1. Room ID or link

Outputs:

1. Share Room ID or URL.

Main Flow:

Actor User	System
1. Access the room ID or URL from the room's create page.	1. Display Room ID
2. Share the room ID or link via social media, email, or other communication tools.	

Table 2.5 UC5 Main Flow

Preconditions:

1. User has joined a room or has access to the room ID

Postconditions:

1. The room ID is successfully shared, and others can join the room using the shared details.

06. Use Case 6: Join Room

Name: Join Room

Identifier: UC6

Inputs:

1. Room ID or link.

Outputs:

1. Access to the room and all its features (voice chat, drawing)

Main Flow:

Actor User	System
1. Enters the room ID	1. Verify the room's availability
2. If valid, the user can join the room.	

Table 2.6 UC6 Main Flow

Preconditions:

1. User is logged in and has received an invitation or owner to join the room.

Postconditions:

1. User successfully joins the room and can participate in activities.

07. Use Case 7: Send Voice Chat

Name: Logout

Identifier: UC7

Inputs:

1. Audio input is allowed.

Outputs:

1. Audio sent to room participants

Main Flow:

Actor User	System
1. Press the "micro" button or activate the voice chat feature.	1. Transmit the voice message to all participants in the room in real time.

Table 2.7 UC7 Main Flow

Preconditions:

1. User is logged in and has joined an active voice chat room.
2. The user's microphone is functioning and properly configured.

Postconditions:

1. All participants can hear the voice chat.

08. Use Case 8: Receive Voice Chat

Name: Receive Voice Chat

Identifier: UC8

Inputs:

1. Audio input from other participants

Outputs:

1. Audio playback of the voice message

Main Flow:

Actor User	System
1. Adjust volume or mute/unmute their audio if necessary	1. Receive audio message in real time

Table 2.8 UC8 Main Flow

Preconditions:

1. User is logged in and has joined an active voice chat room.
2. The user has audio output capabilities (speakers or headphones).

Postconditions:

1. The user listens as voice messages are sent by other participants.

09. Use Case 9: Control Drawboard

Name: Control Drawboard

Identifier: UC9

Inputs:

1. Drawing tools, interaction with the board

Outputs:

1. Real-time changes to the drawing on the board

Main Flow:

Actor User	System
1. Select a tool (pen, eraser, color, shapes, move, text)	
2. Interact with the board, making annotations or drawings.	2. Update the drawboard in real time for all participants to see

Table 2.9 UC9 Main Flow

Preconditions:

1. User is logged in and has joined the drawboard session.

Postconditions:

1. The drawboard is updated with the user's drawing/annotations.

10. Use Case 10: Manage Users

Name: Manage Users

Identifier: UC10

Inputs:

1. User information (username, email, password)

Outputs:

1. Confirmation message. [If successful]
2. Error message. [If fail]

Main Flow:

Actor Admin	System
1. Select "Manage Users" from the admin dashboard.	
2. View a list of existing users.	
3. Admin can: Add/ Edit/ Delete users	
4. Confirm changes.	4. Update the user database.

Table 2.10 UC10 Main Flow

Preconditions:

1. Admin is logged in and has appropriate permissions to manage users.

Postconditions:

1. Users are added, edited, or removed from the system.
2. The user database reflects the updated information.

11. Use Case 11: Manage Voice chat

Name: Manage Voice chat

Identifier: UC11

Inputs:

1. Voice chat settings

Outputs:

1. Confirmation message. [If successful]
2. Error message. [If fail]

Main Flow:

Actor Admin	System
1. Enter the "Manage Voice Chat" section of the admin interface.	
2. Select an active voice chat session.	2. Display a success message or error if the action cannot be performed.

Table 2.11 UC11 Main Flow

Preconditions:

1. Admin is logged in and has the necessary permissions to manage voice chat.
2. A room with active voice chat exists.

Postconditions:

1. Voice chat settings are applied to the active session.

12. Use Case 12: Manage Drawings

Name: Manage Drawings

Identifier: UC12

Inputs:

1. The drawing is successfully saved by user.

Outputs:

1. Confirmation message. [If successful]
2. Error message. [If fail]

Main Flow:

Actor Admin	System
1. View or manage the saved drawing from the admin dashboard.	1. Confirm the drawing is successfully saved.

Table 2.12 UC12 Main Flow

Preconditions:

1. Admin is logged in to the system.
2. Admin has access to the drawboard.
3. A drawing session is active, and the content to be saved is ready.

Postconditions:

1. The drawing is accessible for future use, modification, or sharing.

2.1.3. FUNCTIONAL REQUIREMENTS

Req.ID	Requirement Name	Detailed Description	Type
001	Register an account	If the user doesn't have an account, then he will be asked to register. The user must provide a unique username, email and password.	Functional requirement
002	Log In	The user log in using their registered credentials (username and password)	Functional Requirement
003	Search Meeting Rooms	Allow users to search for meeting rooms based on keyword Meeting ID	Functional Requirement
004	Create Meeting Room	Allow users to create a new meeting room to communicate with other users with info, including meeting ID, name and description of meeting.	Functional Requirement
005	Join Meeting Room	The Owner (Host) of the meeting room can directly join the meeting room. While participants enter a true meeting ID, they just join.	Functional Requirement
006	Voice Chat	An online chat on the site where users could communicate with other users through voice.	Functional Requirement
007	Draw on Whiteboard	Allow users in a meeting room to draw collaboratively on a shared canvas in real time, using tools like pen, shapes, colors, and eraser.	Functional Requirement
008	Users Account Management	Admin can add, edit, or delete user accounts. User data is updated in real-time in the database.	Functional Requirement
009	User Profile Management	Enable users to update their profile information, manage saved username, email, and change their password.	Functional Requirement
010	Save Drawings	The system provides an option to save the dashboard as an image or file, either locally or in the system database.	Functional Requirement
011	Activity Logs	Admins can access logs showing user activity, including voice chat, drawings created and logins.	Functional Requirement

Table 2.13 Functional Requirements

2.1.4. NON-FUNCTIONAL REQUIREMENTS

Req.ID	Requirement Name	Detailed Description	Type
012	Performance	Ensure fast loading times and responsive user interface to enhance user experience.	Non-functional Requirement
013	Security Requirement	All communications between the system's data server and clients must be encrypted	Non-functional requirement
014		Failure to log in the website (five times or more) should lead to a ban from accessing the website within 24 hours.	Non-functional requirement
015	Scalability	Design the website to handle a large number of users and increase data load, with the ability to scale resources as needed.	Non-functional requirement
016	Accessibility	Ensure that the website is accessible to users with disabilities, following WCAG guidelines.	Non-functional requirement
017	Compatibility	Support multiple devices and browsers to reach a wide range of users.	Non-functional requirement
018	Reliability	Minimize downtime and errors to ensure the website is available and functional at all times.	Non-functional requirement
019	Usability	Design the website with a user-friendly interface and intuitive navigation to make it easy for users to find and use features.	Non-functional requirement
020	Compliance	Ensure compliance with relevant regulations and standards for e-commerce websites, such as GDPR and PCI DSS.	Non-functional requirement
021	Performance Monitoring	Implement tools to draw and voice chat website performance and identify areas for optimization.	Non-functional requirement
022	Backup and Recovery	Regularly backup website data and implement a plan for disaster recovery to minimize data loss in case of emergencies.	Non-functional requirement

023	Localization	Support multiple languages and currencies to cater to a global audience.	Non-functional requirement
024	Feature Integration	Allow users to share drawings and dashboard on shared platforms and integrate social login options for easier registration and login.	Non-functional requirement

Table 2.14 Non-functional Requirements

2.2. DESIGN

2.2.1. SYSTEM ARCHITECTURE MODEL

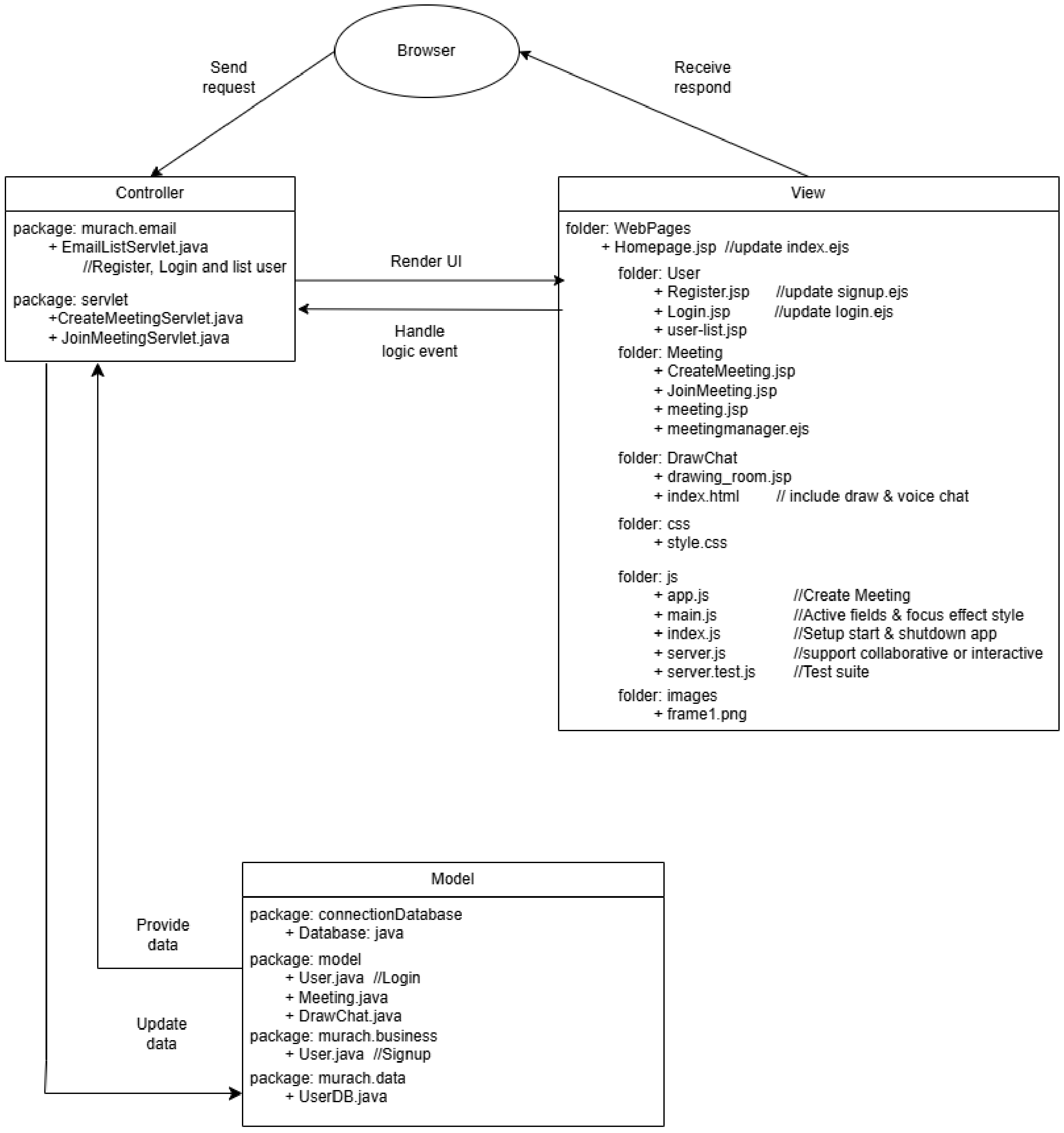


Figure 2.2 MVC Architecture

2.2.2. ENTITY-RELATIONSHIP DIAGRAM (ERD)

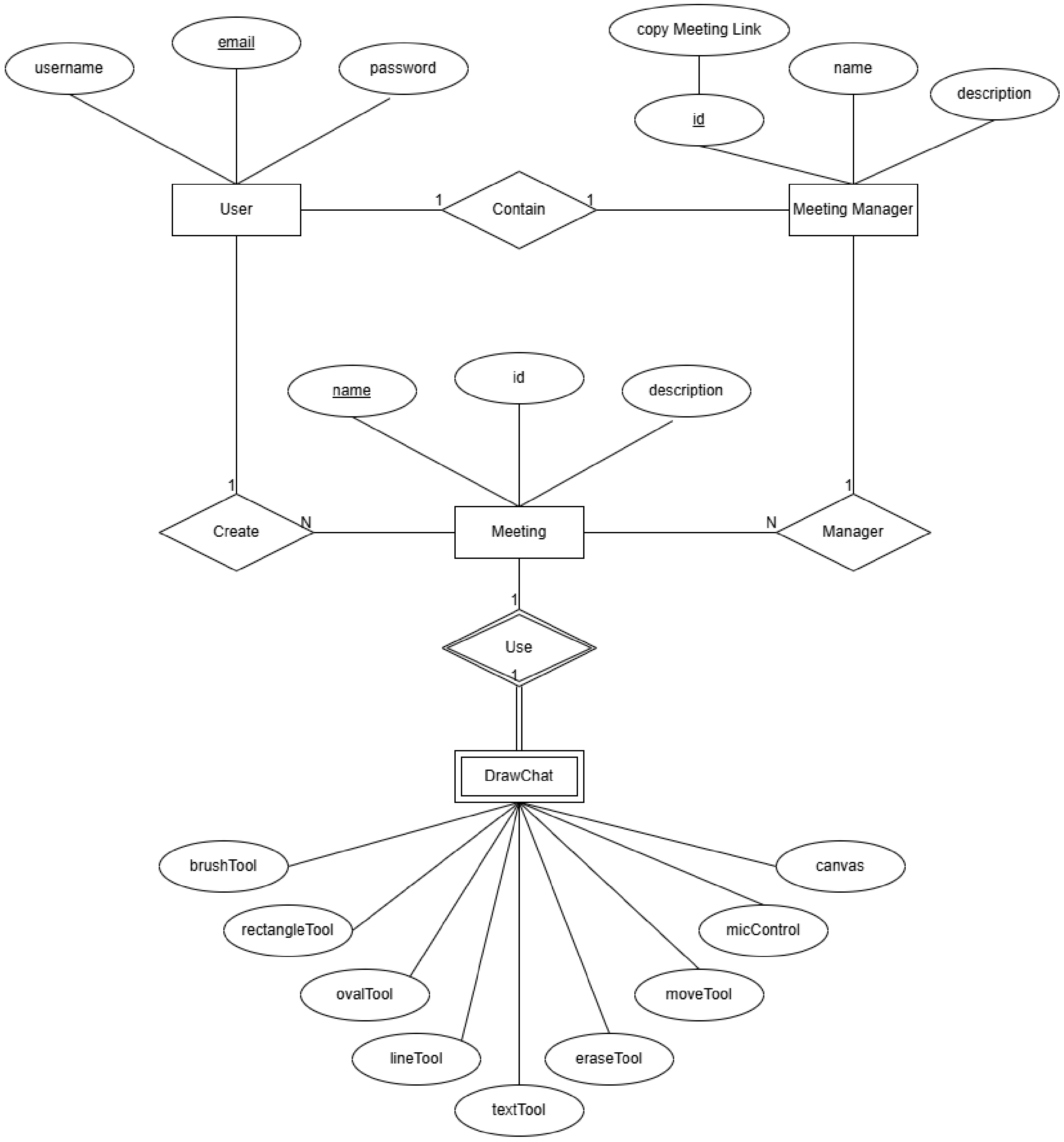


Figure 2.3 ERD Diagram

2.2.3. CLASS DIAGRAM

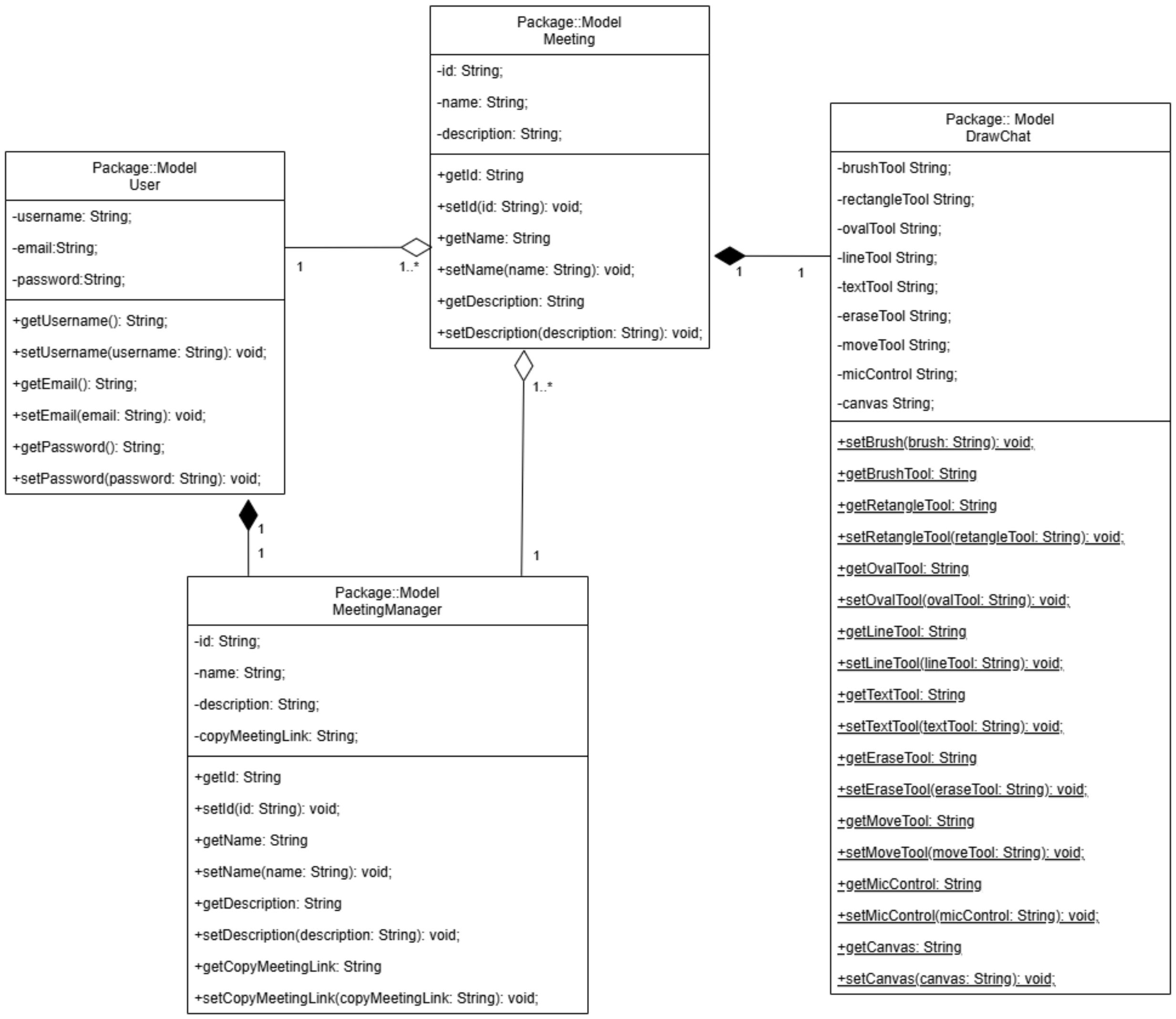


Figure 2.4 Class Diagram

2.2.4. USE CASE DIAGRAM

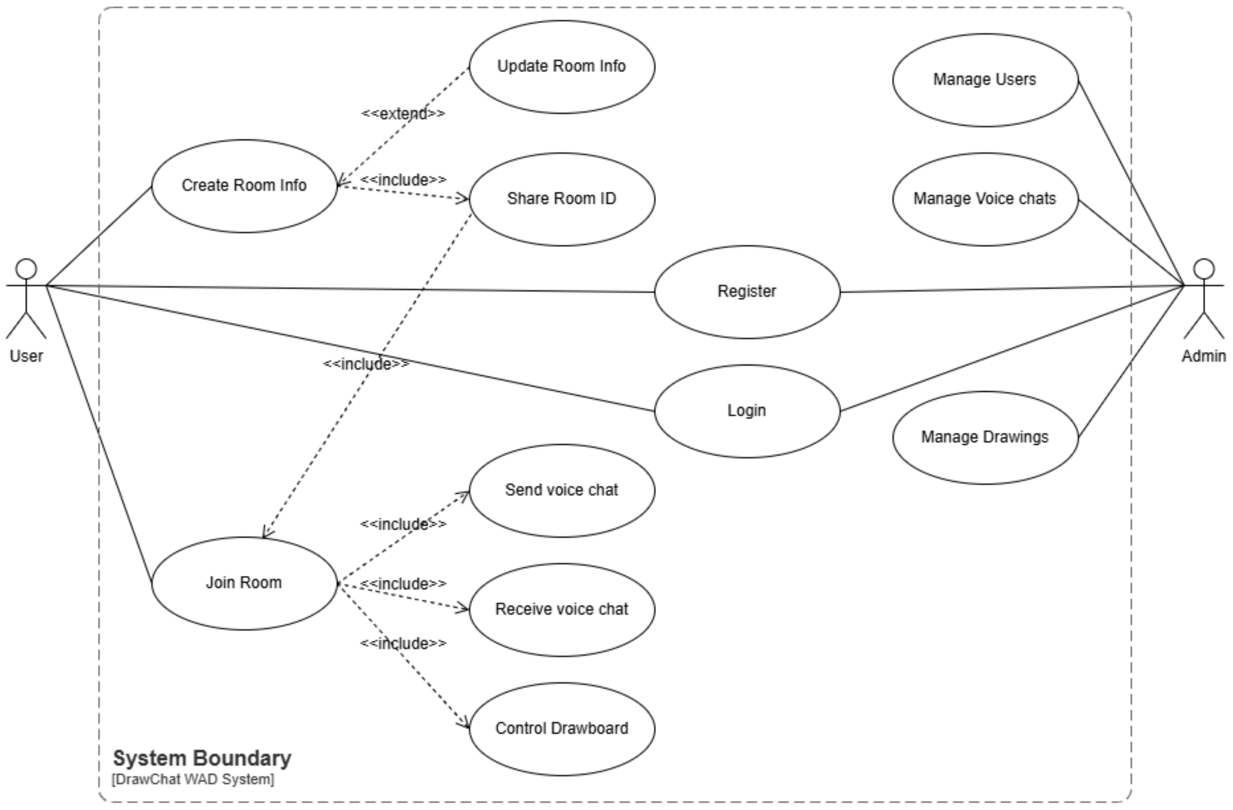


Figure 2.5 Use Case Diagram

2.2.5. SEQUENCE DIAGRAM

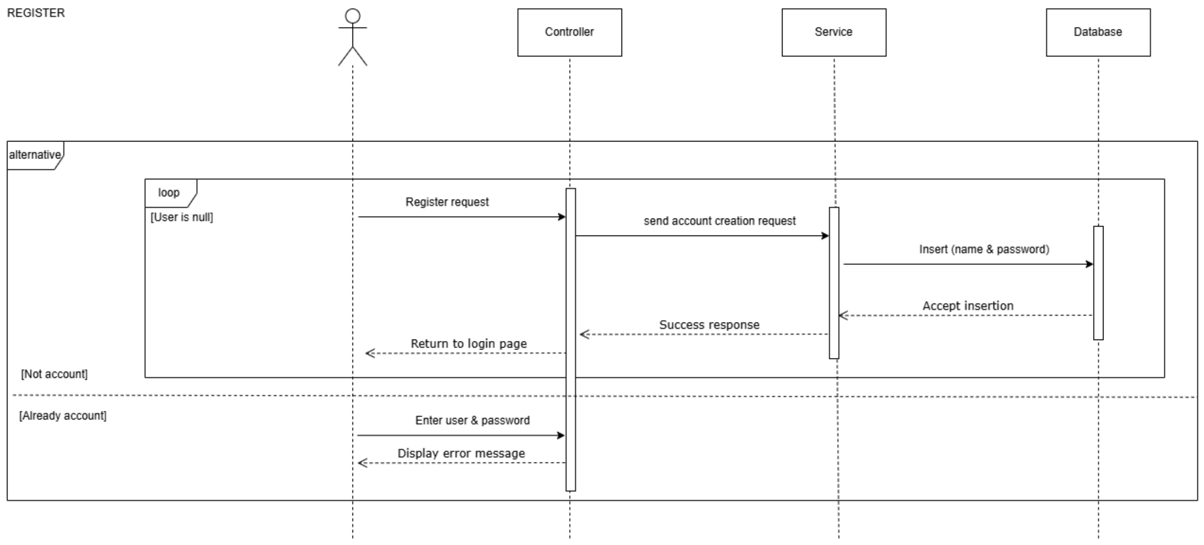


Figure 2.6 Create Account Sequence Diagram

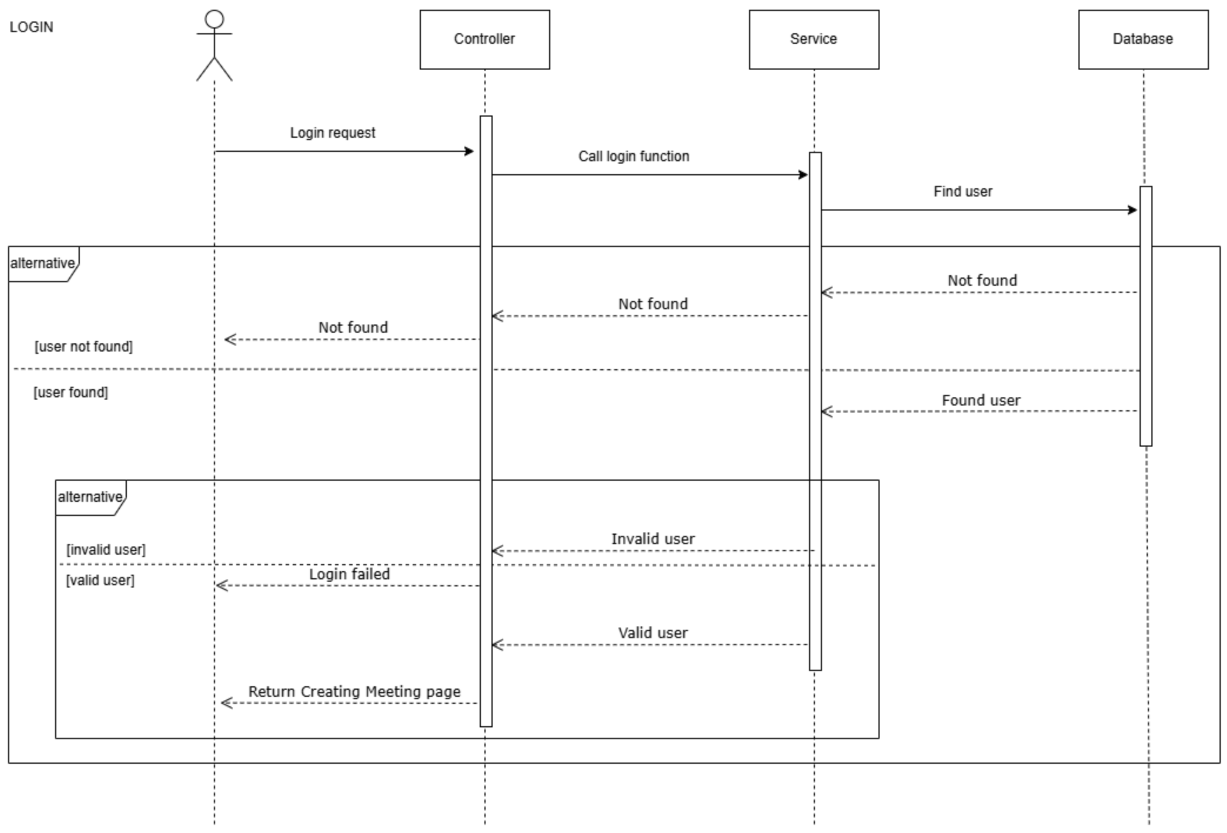


Figure 2.7 Login System Sequence Diagram

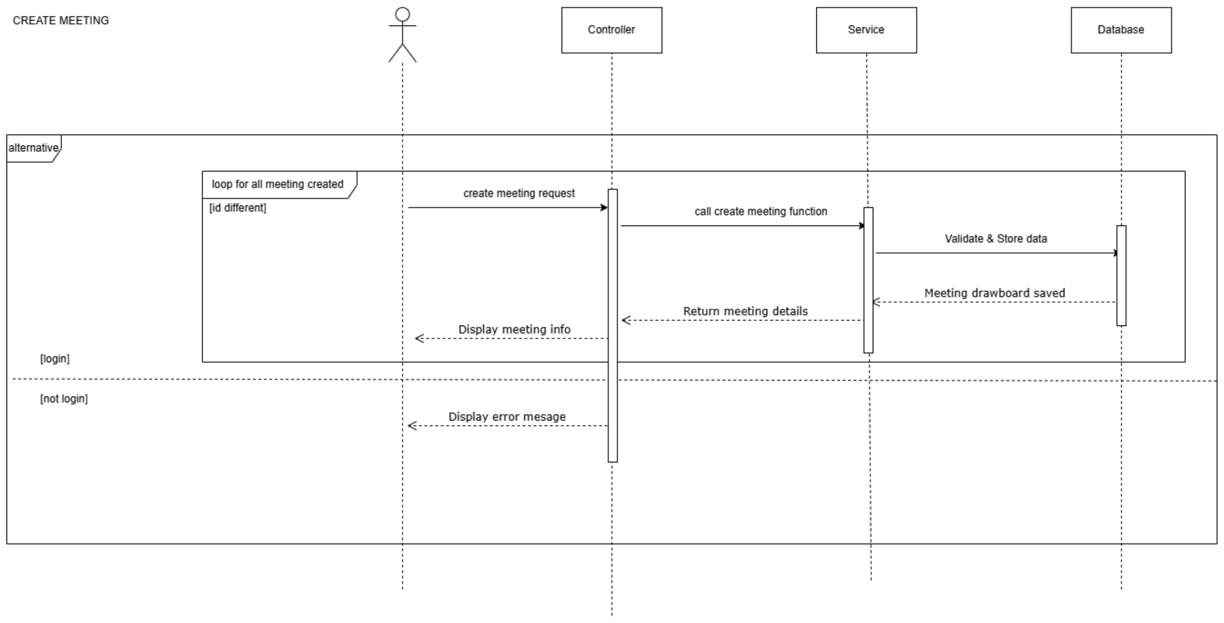


Figure 2.8 Create meeting Sequence Diagram

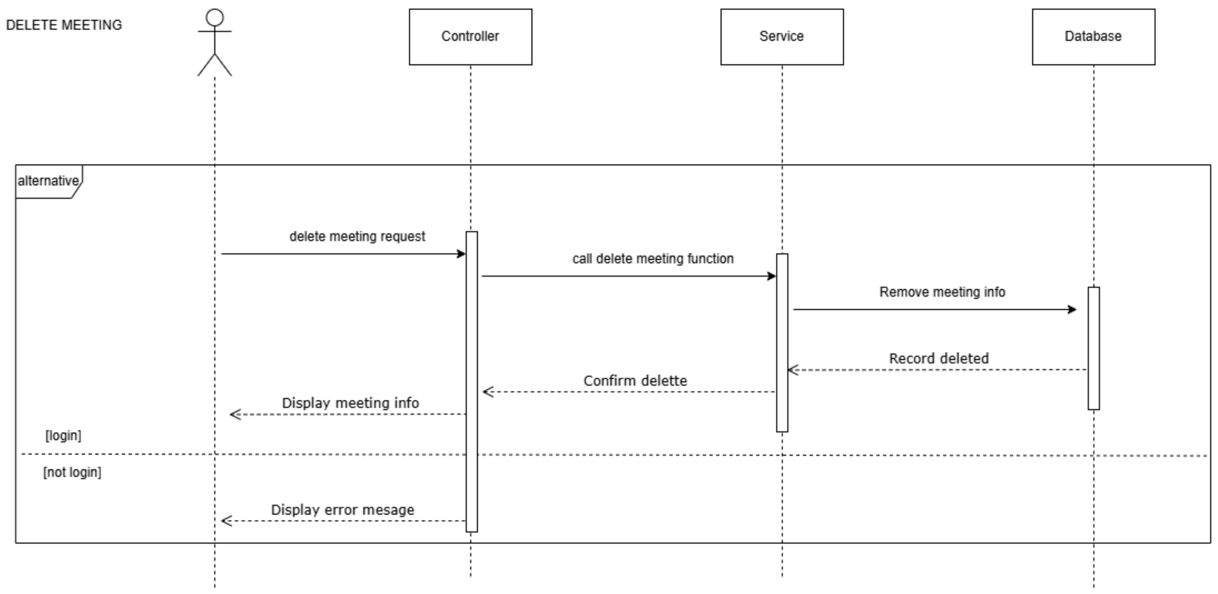


Figure 2.9 Delete meeting Sequence Diagram

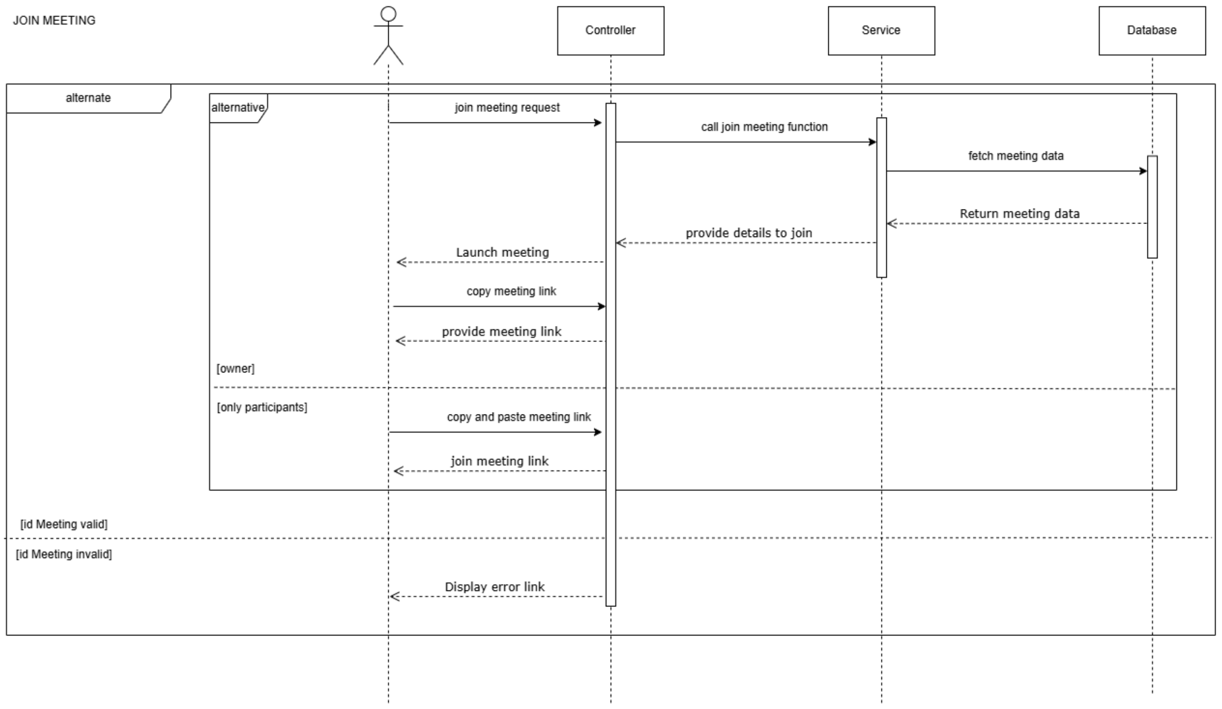


Figure 2.10 Join meeting Sequence Diagram

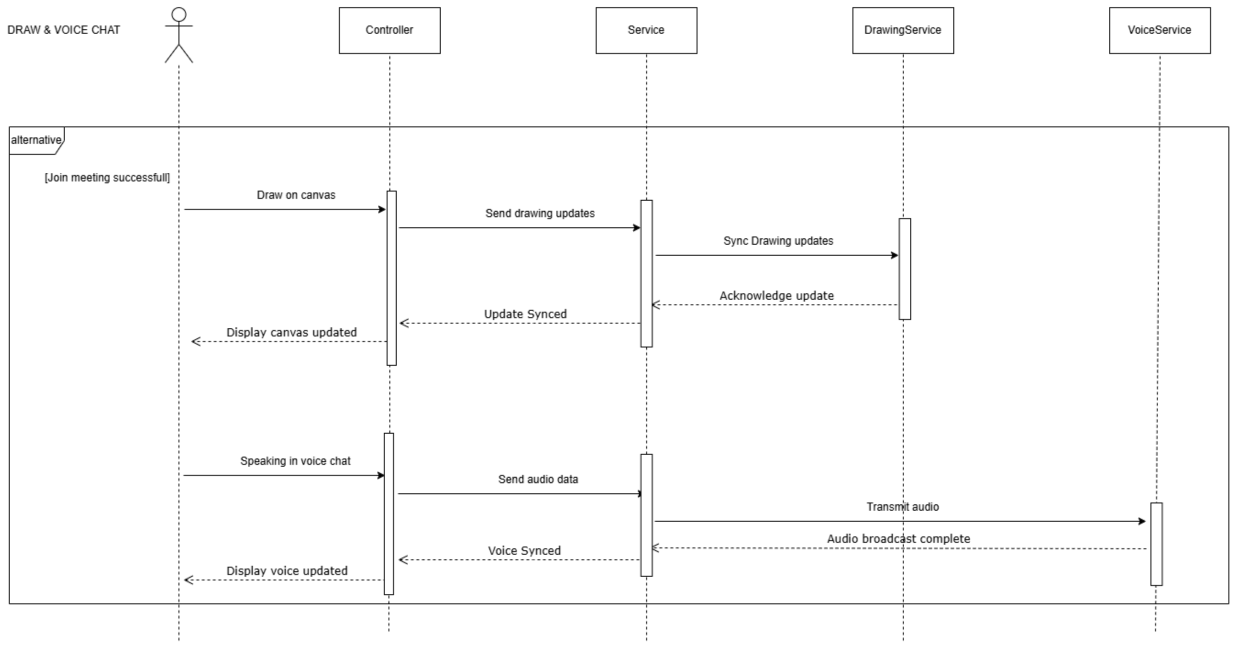


Figure 2.11 Draw & Voice chat features Sequence Diagram

3. IMPLEMENTATION

3.1. Implementation of Pre-Meeting Landing Page

3.1.1. User Authentication

The user authentication system was designed with a strong emphasis on security, particularly in the handling of passwords. The login and signup process leverages a "server-relief" protocol, where the client-side JavaScript uses the SubtleCrypto API to hash the password with PBKDF2. This process is personalized using a string derived from the username, ensuring that each user's password hashing is unique. Specifically, the personalization string is constructed as "MEETING MEETING 747589285774729292[" + username + "]" to enhance security. The hashed password is then sent to the server, where it undergoes a lightweight SHA-256 hashing before being stored in the database. This approach minimizes the server's computational load while maintaining a high level of security.

The login process mirrors this hashing mechanism, ensuring that the stored password hash can be securely verified without exposing the original password. Additionally, secure session handling was implemented using cookies, ensuring that user sessions are managed in a way that prevents unauthorized access.

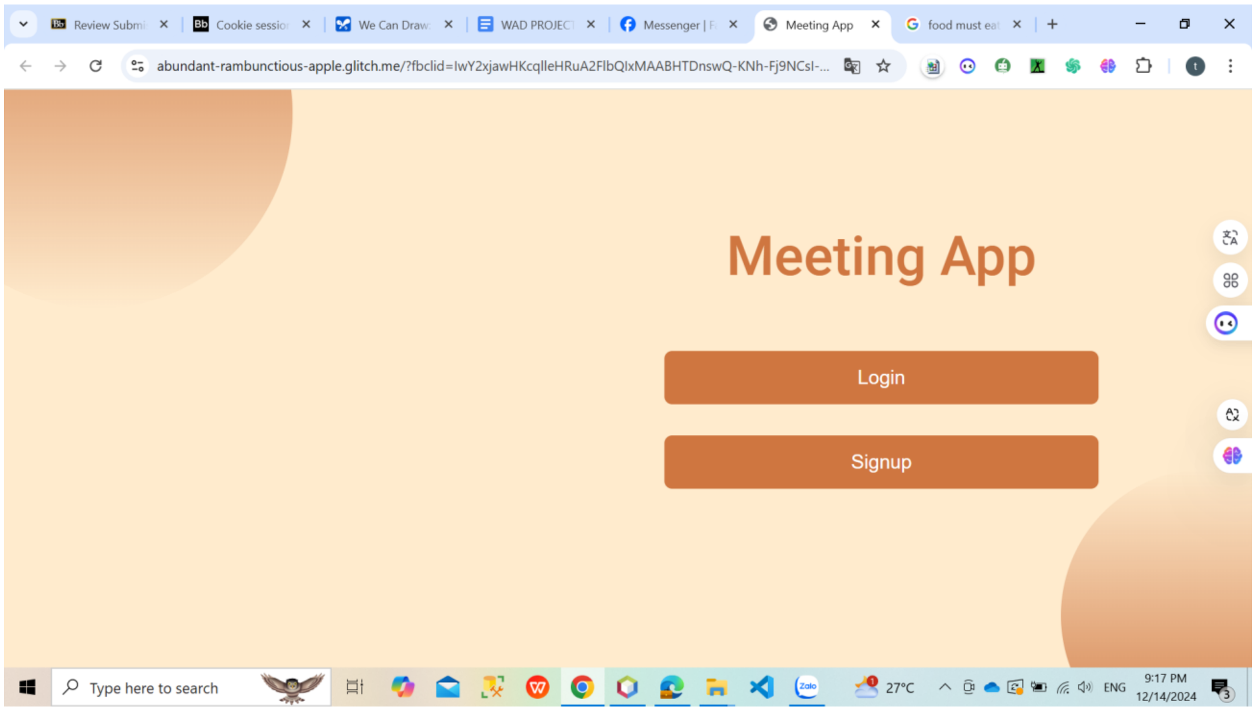


Figure 3.1 Home page interface

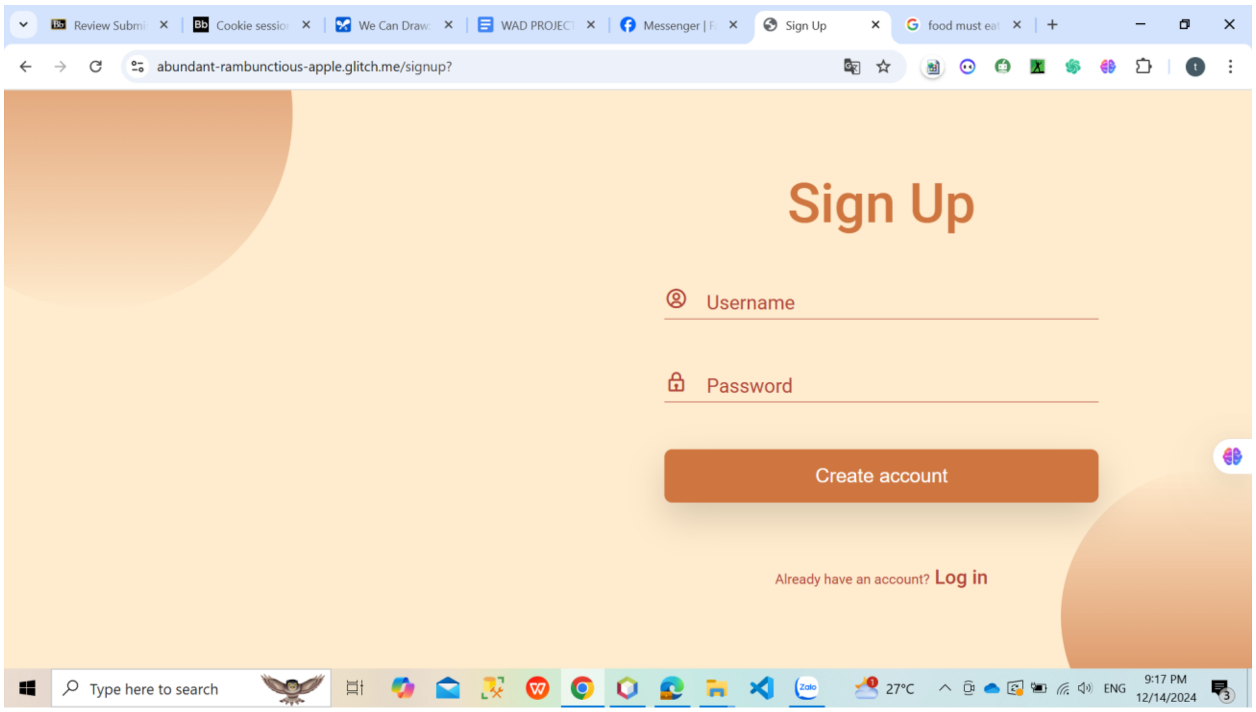

abundant-rambunctious-apple.glitch.me/signup?
Sign Up
Username
Password
Create account
Already have an account? Log in

Figure 3.2 Register page interface

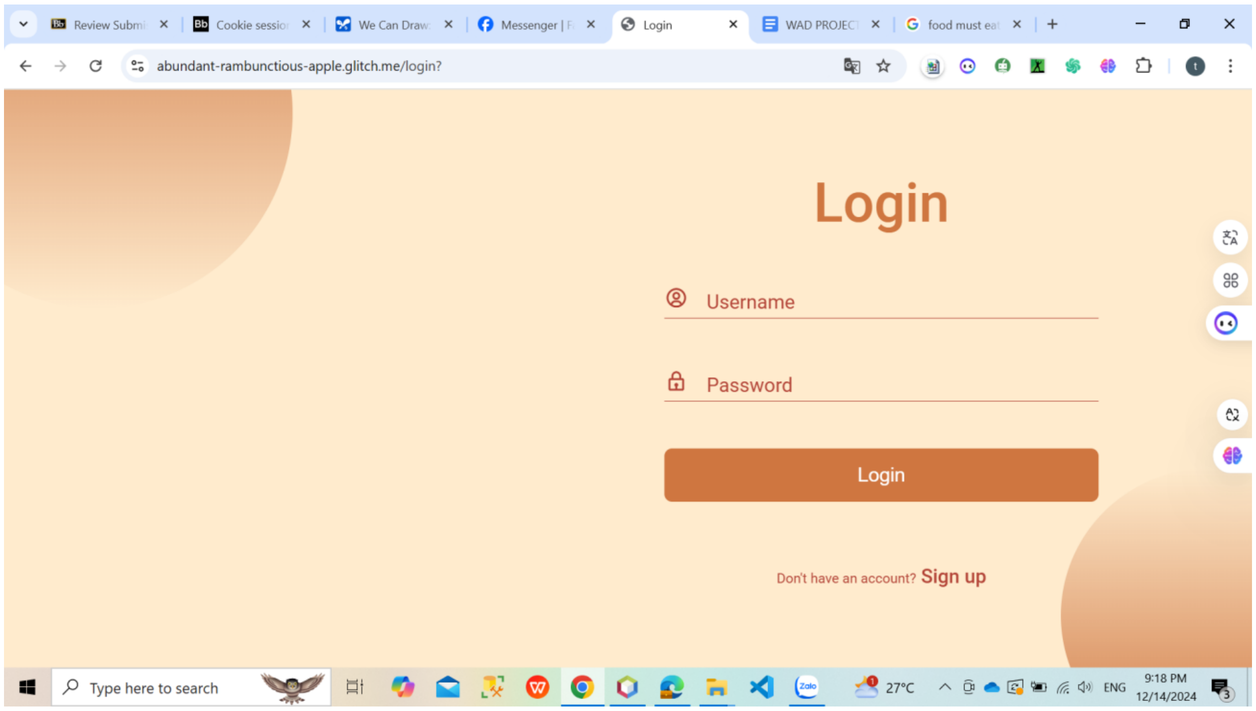

abundant-rambunctious-apple.glitch.me/login?
Login
Username
Password
Login
Don't have an account? Sign up

Figure 3.3 Login page interface

3.1.2. Meeting Management

Once authenticated, users gain access to a personalized meeting list. This list allows users to create, edit, and delete meetings, each of which is assigned a unique identifier. The meeting ID is generated using a cryptographically secure random string, ensuring that each meeting link is both unique and tamper-proof. The base URL for meetings is https://glory-buttercup-wolverine.glitch.me/, and the unique meeting ID is appended to this URL using a # sign, resulting in a format like https://glory-buttercup-wolverine.glitch.me/#hfiwbfiwnfiwnfiejehgjwi. Users can create meetings by providing a name and description. Upon submission, the server generates the meeting ID and stores the details in the database. The meeting list is dynamically rendered using EJS templates, ensuring a responsive and user-friendly interface. Each meeting entry includes options to edit or delete the meeting, as well as buttons to copy the meeting link or join the meeting directly. The "Copy Meeting Link" feature uses the Clipboard API to allow users to easily share meeting URLs.

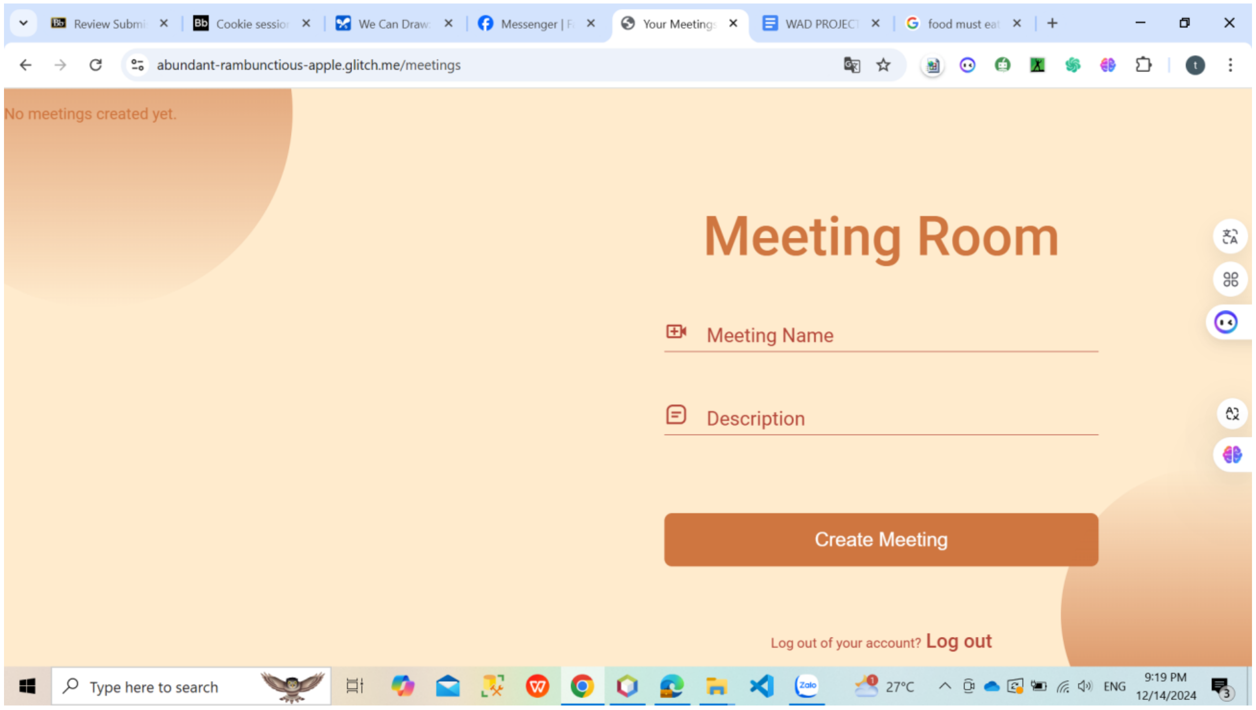
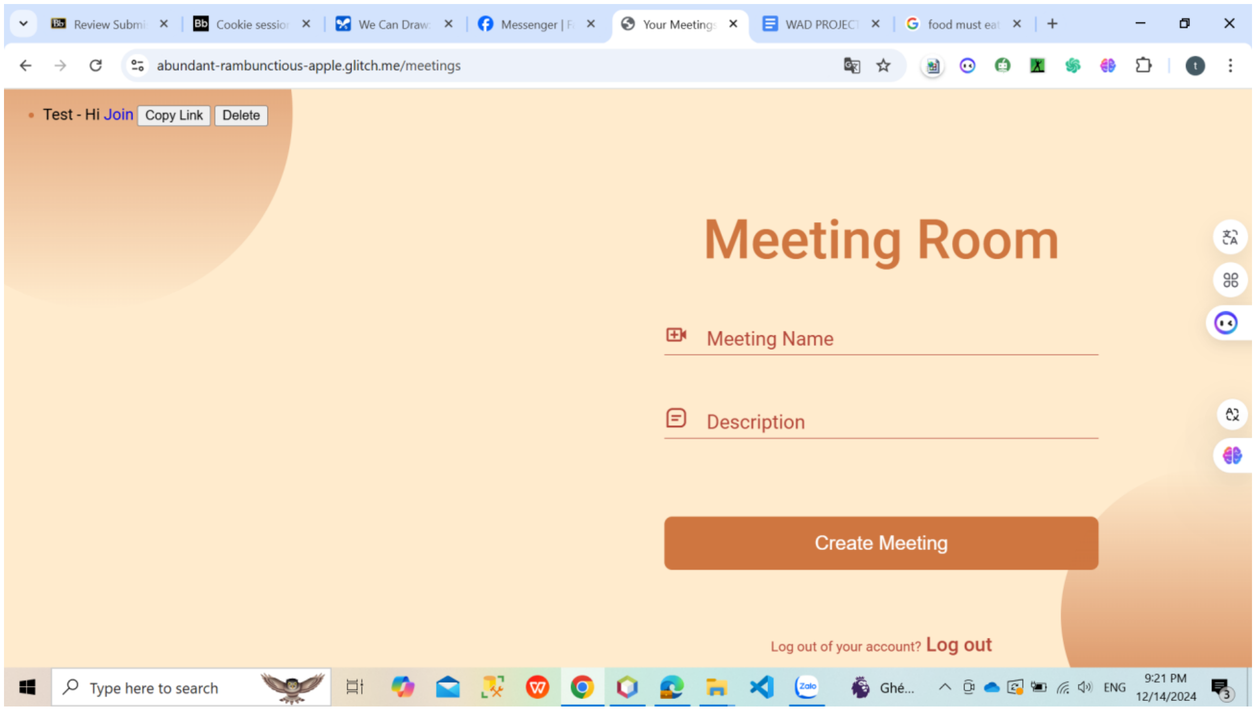
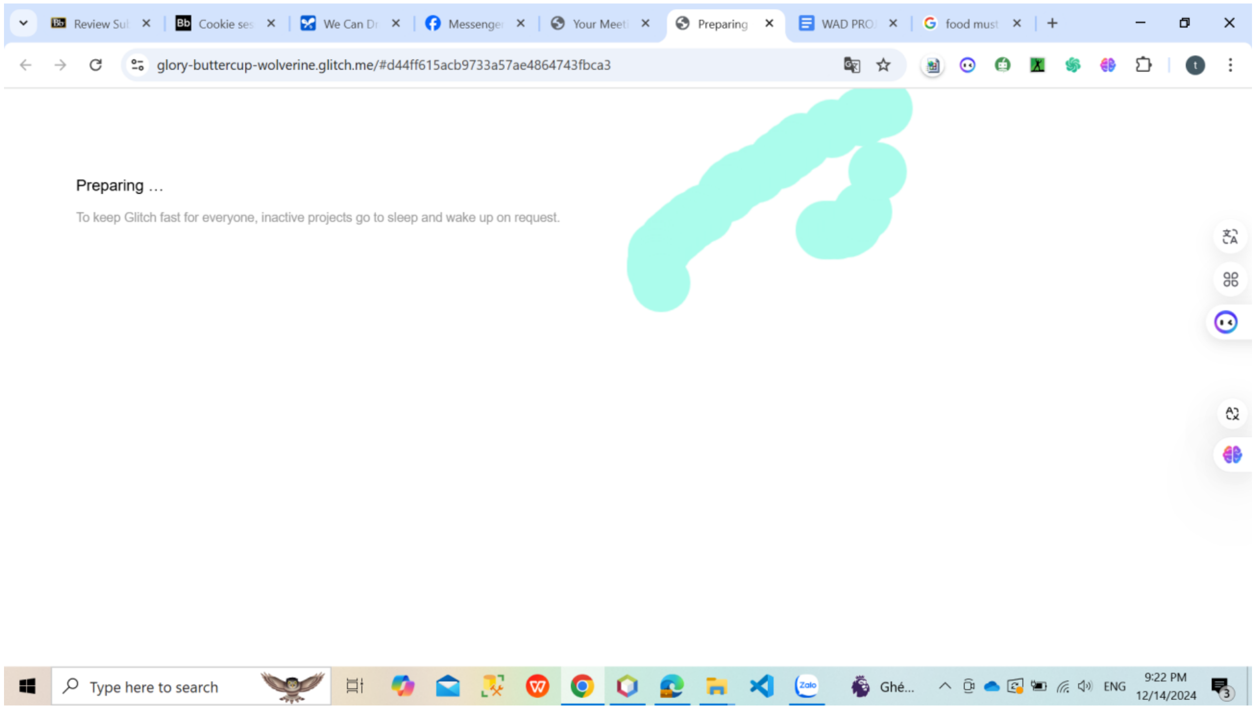

abundant-rambunctious-apple.glitch.me/meetings
No meetings created yet.
Meeting Room
Meeting Name
Description
Create Meeting
Log out of your account? Log out

Figure 3.4 Meeting Room interface



abundant-rambunctious-apple.glitch.me/meetings
Test - Hi Join Copy Link Delete
Meeting Room
Meeting Name
Description
Create Meeting
Log out of your account? Log out

Figure 3.5 Meeting manager in Meeting Room interface



glory-buttercup-wolverine.glitch.me/#d44ff615acb9733a57ae4864743fbca3

Preparing ...

To keep Glitch fast for everyone, inactive projects go to sleep and wake up on request.

Figure 3.6 Accept link project interface

3.1.3. Security and Session Management

Security was a top priority throughout the implementation. In addition to the server-relief password hashing, the system employs secure session management using cookies. The cookies are configured with the HttpOnly and Secure flags to prevent client-side access and ensure that they are only transmitted over HTTPS. Additionally, the SameSite attribute is set to Strict to mitigate cross-site request forgery (CSRF) attacks.
The session data is stored on the server, with the session ID being the only information transmitted via cookies. This ensures that sensitive user information is never exposed to the client.

3.2. Implementation of Meeting Page

At its core, the application is designed to facilitate two primary components: a collaborative drawing board and real-time audio communication. The drawing board serves as a shared canvas where multiple users can draw, erase, and add text simultaneously, while the audio communication feature, integrated with Jitsi Meet, ensures that users can discuss their ideas in real time. Together, these components create an environment where visual and verbal collaboration are tightly intertwined.
The collaborative drawing board is a central feature of the application, offering a versatile set of tools that cater to various creative needs. Users can select from tools such as a brush for freehand drawing, a rectangle tool for creating geometric shapes, an oval tool for drawing circles or ellipses, a line tool for straight edges, and a text tool for adding annotations. Additionally, users can erase parts of their drawings or move the canvas to explore different areas. The board also includes a color picker, allowing users to choose their preferred drawing color, and a real-time display of the current mouse or touch coordinates, which aids in precise drawing.
One of the standout features of the drawing board is its ability to synchronize changes in real time. When one user makes a change, such as drawing a line or adding text, that change is instantly reflected on the canvases of all other participants. This real-time collaboration is made possible through WebSocket communication, ensuring that all users stay in sync without delay. Furthermore, the application ensures that the state of the drawing board is preserved by storing drawing events in a database. This means that when users rejoin a session, they can pick up where they left off, with the canvas restored to its previous state.
Another noteworthy aspect of the drawing board is its preview mode. When using tools like rectangles, ovals, lines, or the eraser, the application displays a preview of the shape or action before it is finalized. This allows users to visualize their changes and make adjustments as needed, enhancing the overall user experience.
The drawing board is built using HTML5 canvases, which provide a flexible and performant platform for rendering graphics. The application uses two canvases: a main canvas that stores the final drawing and a preview canvas that displays temporary previews for shape tools. This dual-canvas approach ensures that users can see immediate feedback while drawing without affecting the final output.
WebSocket communication plays a crucial role in the application's architecture. Each drawing event, such as a brush stroke or a text annotation, is encoded as a JSON object and sent to the server. The server then broadcasts the event to all other participants in the same session, ensuring

that everyone's canvas is updated simultaneously. Drawing events are also stored in a database, allowing the application to persist the canvas state even after users leave the session. When a user rejoins, the application retrieves the stored events from the database and replays them to restore the canvas to its previous state.

The application also includes a resizing feature that ensures the canvas adapts to the user's viewport, providing a consistent experience across devices. Whether accessed on a desktop, tablet, or smartphone, the canvas remains responsive and easy to use.

The audio communication feature, integrated with Jitsi Meet, complements the drawing board by enabling users to discuss their ideas while collaborating visually. Users can join a meeting room and control their microphones using buttons in the toolbar. The ability to enable or mute the microphone ensures that users can manage their audio input as needed, enhancing the overall collaboration experience.

To enhance privacy and security, the application generates meeting room names using a SHA-256 hash of a unique identifier. This ensures that each room is distinct and secure, preventing unauthorized access. Additionally, user display names are generated using a hash of random data, providing anonymity while maintaining a unique identifier for each participant.

The audio communication feature is implemented using the Jitsi Meet External API, which provides a robust and scalable solution for real-time audio communication. The API is configured to start with audio and video muted, and the pre-join page is disabled, ensuring a smooth user experience. Users can toggle their microphones on or off using buttons in the toolbar, and the state of the microphone is synchronized across all participants.

The application also uses a separate WebSocket connection for audio data, enabling real-time audio streaming between participants. This ensures that audio communication remains low-latency and responsive, even as users interact with the drawing board.

The server-side architecture of the application is designed to support real-time communication and persistent storage. The WebSocket server organizes clients into groups based on a unique identifier, ensuring that drawing events and audio data are synchronized within the same session. When a client sends a drawing event, the server stores it in the database and broadcasts it to all other clients in the same group. This ensures that all participants see the same canvas state in real time.

The application also provides an endpoint to retrieve all drawing events for a specific session. This endpoint returns the events as a binary stream, which is used to initialize the canvas when a user joins the session. By retrieving and replaying these events, the application ensures that the canvas state is preserved and consistent across all users.

To maintain the connection, the server periodically sends ping messages to clients, ensuring that the connection remains active.

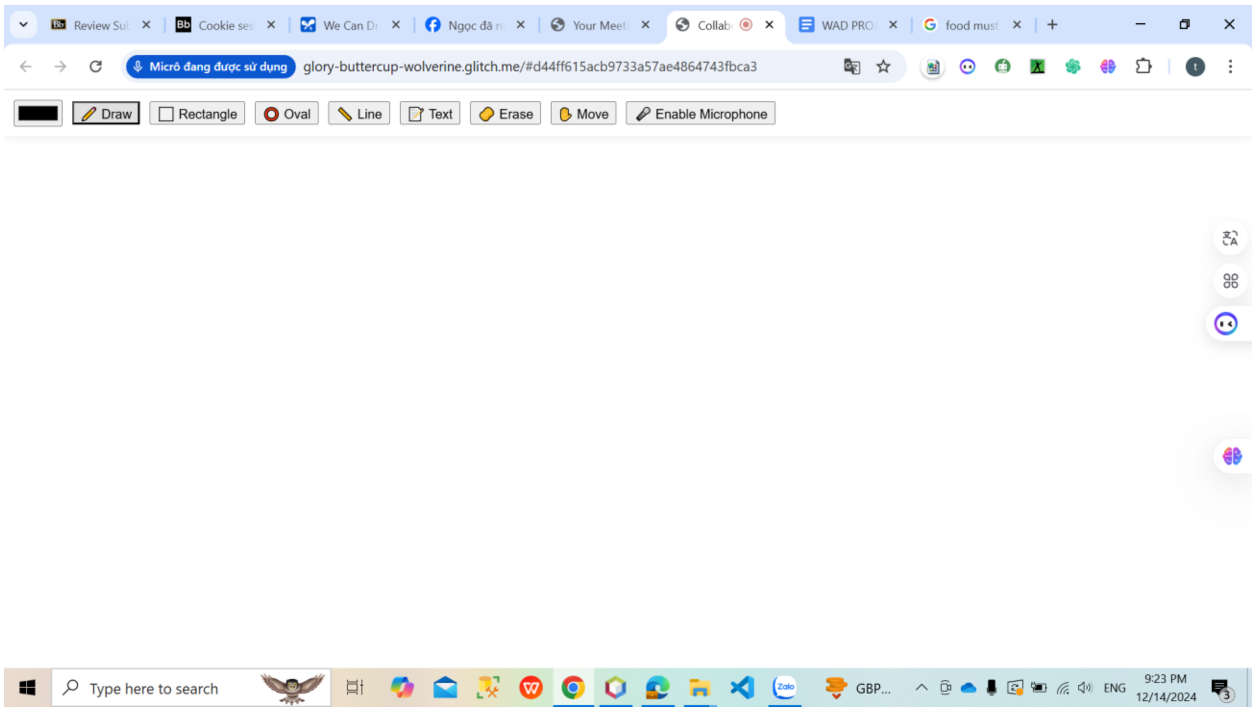


Figure 3.7 Meeting interface

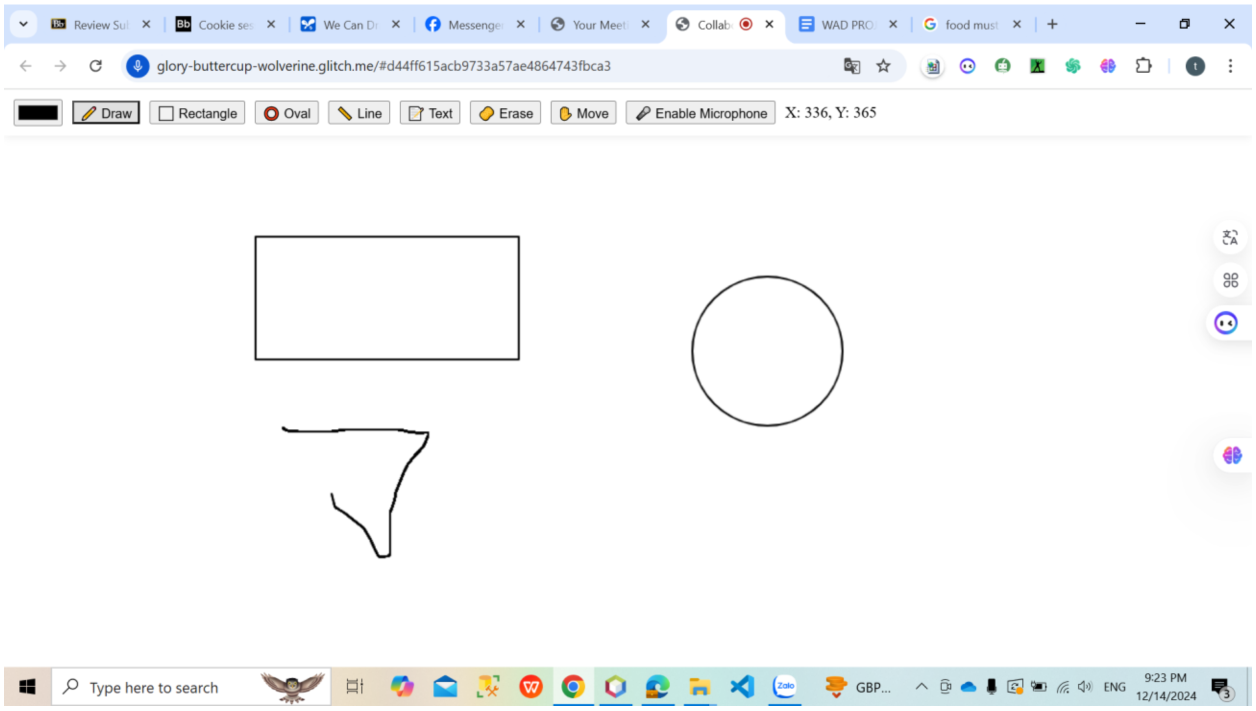

glory-buttercup-wolverine.glitch.me/#d44ff615acb9733a57ae4864743fbca3
Draw
Rectangle
Oval
Line
Text
Erase
Move
Enable Microphone
X: 336, Y: 365

Figure 3.8 Draw and Voice chat features interface

4. DISCUSSION AND CONCLUSION

4.1. Discussion

Initial Challenges	- Communication breakdowns led to uneven task distribution. - Difficulties in applying knowledge from the Web Application Development course. - Technical challenges in connecting Node.js with Servlet.
Team Resilience	- Fostered open dialogue and improved communication. - Strengthened team dynamics for more effective collaboration. - Implemented better workflows and time management strategies.
Technical Success	- Overcame technical hurdles, achieving seamless integration of functionalities. - Successfully connected Node.js with Servlet.
Future Potential	- Completion of the project showcases perseverance and problem-solving skills. - Promises exciting opportunities for future development. - Potential to become a valuable tool for artists in a shared online creative space.

Table 4.1 Problem discussed

4.2. Conclusion

This project, "We Can Draw," envisioned as a real-time collaborative drawing platform similar to a visual Zoom, provided a rich learning experience interwoven with significant challenges. Early in the project lifecycle, we encountered obstacles stemming from communication breakdowns, which led to uneven task distribution and amplified the inherent difficulties of applying newly acquired knowledge from our concurrent Web Application Development course. The technical intricacies of connecting Node.js with Servlet presented a particularly demanding hurdle. Furthermore, the compressed project timeline contributed to significant time management struggles, often resulting in a sense of being overwhelmed.

Despite these initial setbacks, our team demonstrated resilience and adaptability. Through open dialogue and a renewed commitment to clear communication, we fostered a stronger team dynamic, ultimately leading to more effective collaboration. We implemented improved workflows and time management strategies, enabling us to address the uneven workload distribution. Crucially, we successfully overcame the technical challenges, achieving seamless integration of all desired functionalities, including the previously troublesome connection between Node.js and Servlet. The successful completion of "We Can Draw" stands as a testament to our perseverance and problem-solving abilities. This robust foundation promises exciting avenues for future development, including expanded features, enhanced user experience, and further exploration of real-time collaborative technologies. We are confident that "We Can Draw" has the potential to become a valuable tool for artists seeking a shared online creative space.

5. REFERENCES

- **SCRUM**
  - Scrum.org (no date) *Scrum Guide*. Available at: https://www.scrum.org/ (Accessed: 15 December 2024).
  - Sommerville, I. (2011) *Software Engineering* (9th ed.). Pearson.
- **Agile**
  - AgileMethodology.org (no date) *Agile Methodology*. Available at: http://agilemethodology.org/ (Accessed: 15 December 2024).
- **MVC Model**
  - Karunakaran, P. (2020) *Introducing Play Framework: Java Web Application Development*.
  - (2017) *Java EE Web Application Primer: Building Bullhorn - A Messaging App with JSP, Servlets, JavaScript, Bootstrap, and Oracle*.
- **Web-page Template**
  - Figma (no date) *Web-page template*. Available at: https://www.figma.com/ (Accessed: 15 December 2024).
  - YouTube (2024) *Web Design Tutorial*. Available at: https://youtu.be/b7gc_4TrXkg?si=DIWdnyVcSMl5fTGu (Accessed: 15 December 2024).
- **Database Management System (DBMS)**:
  - Oracle (no date) *Oracle Database*. Available at: http://www.oracle.com/ (Accessed: 15 December 2024).
  - SQLite (no date) *SQLite Database*. Available at: https://www.sqlite.org/ (Accessed: 15 December 2024).
- **HTML, CSS, JavaScript Techniques**
  - W3Schools (no date) *HTML, CSS, JavaScript tutorials*. Available at: https://www.w3schools.com (Accessed: 15 December 2024).
- **Text Data and Image Data**
  - Boxicons (no date) *Boxicons: Icons for your project*. Available at: https://boxicons.com/?query= (Accessed: 15 December 2024).
    Google Fonts (no date) *Google Fonts*. Available at: https://fonts.google.com/ (Accessed: 15 December 2024).
- **Node.js**
  - Node.js (no date) *Node.js JavaScript runtime*. Available at: https://nodejs.org/en (Accessed: 15 December 2024).