SOFTWARE DESIGN DOCUMENT

GROUP B_005

SAFWAAT

OVERVIEW

TABLE OF CONTENT

OVERVIEW

OVERVIEW

OVERVIEW

TABLE OF FIGURES

OVERVIEW

OVERVIEW

1. OVERVIEW

The Software Design Document (SDD) for Safwaat offers crucial definitions to conceptualise and elaborate on the software's design, building upon the summarised requirements and functionalities outlined in the Software Requirements Specifications (SRS) Report. The objective is to provide clear guidance for a design that can be readily implemented by any programmer who reads this document. This report aligns with the IEEE standards (IEEE Std 1016 – 2009) to ensure its formal structure and comprehensiveness.

1.1. SCOPE

The comprehensive Software Design Document (SDD) encompasses various sections, including an overview that defines the project and outlines its key features. It also addresses design constraints, detailing the overall system architecture and data architecture. Furthermore, it provides a concise update on the current progress and outlines the project schedule. To facilitate understanding, the document incorporates UML diagrams that visually illustrate the system's design, subsystems, and modules. These diagrams serve as visual aids to assist programmers in comprehending the information presented in the document accurately and with ease.

1.2. PURPOSE

The Software Design Document (SDD) serves as a comprehensive blueprint that defines and formalises the structure, functionality, and design principles of a software project. Its primary purpose is to provide a clear and detailed description of how the software will be built, outlining the system's architecture, components, interfaces, and their interactions.

This document aims to serve as a guiding reference for developers and programmers, enabling them to understand the software's requirements, design constraints, and technical specifications. By offering a detailed breakdown of the system's design using UML diagrams and explanatory text, the SDD facilitates effective communication among team members, ensuring a consistent and coherent approach to implementation while allowing for easier maintenance and future enhancements of the software system.

1.3. INTENDED AUDIENCE

Safwaat caters to a diverse range of users, first addressing individual learners keen on enhancing their Arabic pronunciation and Tajweed abilities. This user group encompasses students enrolled in Arabic language courses within academic settings, individuals engaged in religious studies, language enthusiasts seeking self-improvement, and self-motivated learners.

OVERVIEW

Secondly, the software extends its applicability to the education sector, specifically targeting educational institutions and language learning centers dedicated to refining Arabic pronunciation skills. Lastly, the project addresses the needs of the religious sector, particularly

Islamic schools seeking resources to aid in teaching Tajweed and Arabic pronunciation to their students. These distinct user categories reflect the versatile application of the software across various domains, emphasising its utility in supporting language learning and religious education. Furthermore, it also permits the users of these categories to indulge themselves in productive learning activities through the collaborative and competitive environment provided by Safwaat..

2. DEFINITIONS

DEFINITIONS TABLE

TERMS	DEFINITIONS
API	Application Programming Interface
CPU	Central Processing Unit
XP	”XP” stands for ”experience.” Refers to the experience level of the user in app
DBMS	Database Management System
State	Refers to the current condition or status of the application.
SDD	Software Design Description
SRS	System Requirements Specification
Unit	Refers to the smallest piece of code that can be logically isolated in a system
Tajweed	Study of the Arabic language's regulatory rules for recitation
League	Indicate the general skill level of players
Module	Refers to reusable components that can contain variables, functions, and other defining statements
Embed	Data transformed into a lower dimensional representation
ER	Entity Relation
UML	Unified Modeling Language

3. CONCEPTUAL MODEL FOR SOFTWARE DESIGN DESCRIPTION

Basic terms, concepts and context of SDD will be given in this part.

3.1. SOFTWARE DESIGN IN CONTEXT

Safwaat is an all-encompassing educational application crafted to empower users by offering robust learning tools, personalized experiences, and effective progress monitoring. Our platform offers a variety of dynamic features, ensuring an engaging and productive learning journey for individuals aspiring to excel in Arabic pronunciation and Tajweed skills.

Central to our platform are key components such as the Friendship Hub, encouraging secure connections among users, a Leaderboard fostering healthy competition and recognition, and Social Media Sharing to extend the reach of achievements. Additionally, users can engage in real-time group discussions and private chat rooms through ChatSpace, personalize their profiles, and display accomplishments using the User Profile feature. Taqraar enables structured discussions and question sessions, while the Consistency Companion aids users in maintaining daily learning routines.

The Learning Unit seamlessly integrates teaching and practice, while the Notifications System ensures users stay informed, respecting their preferences and data privacy. Daily challenges, badges, and rewards in the Quest Arena, along with streamlined organization of learning materials via FlashCraft, are among the features offered. The Level Map facilitates easy navigation through the educational journey. Safwaat caters to users at various proficiency levels, delivering a compelling educational experience that makes learning enjoyable and highly efficient. As a result, users can connect with peers, monitor their progress, and accomplish their Tajweed learning objectives effectively using Safwaat.

The software resources utilized for this project encompass a diverse set of technologies. The frontend development relies on React.js, while the backend is powered by Node.js in conjunction with Express. Additional web technologies such as HTML, CSS, and JavaScript complement the development stack. MongoDB serves as the chosen database solution. Various libraries like Chart.js and react-chartjs-2 cater to analytics, whereas Three.js, react-three-fiber, and react-three-drei support 3D rendering functionalities. Nodemailer and node-cron manage email services and task scheduling, respectively, while socket.io facilitates chatroom functionalities. The project employs mongoDB@6.2 as the MongoDB driver, Redux for app-wide state management, and NPM (Node Package Manager) for dependency management.

3.2. SOFTWARE DESIGN DESCRIPTIONS WITHIN THE LIFE CYCLE

3.2.1. INFLUENCES ON SDD PREPARATION

The primary driver influencing software design within its life cycle is the software requirements specification. The details outlined in the SRS, encompassing aspects such as product perspective, functional and non-functional requirements, as well as interface specifications, along with the expectations and needs of stakeholders, collectively dictate the project's design approach.

3.2.2. INFLUENCES OF SOFTWARE LIFE CYCLE PRODUCTS

Throughout the preparation phase of the SDD or during the project's implementation stage, certain requirements might undergo alterations. Additionally, the SDD plays a role in shaping the test plans and test documentation for the Safwaat.

3.2.3. DESIGN VERFICATION AND DESIGN ROLE IN VALIDATION

Verification and validation procedures will occur subsequent to the creation of test cases. All components of the system will undergo testing based on these cases to ascertain whether the requirements have been met or not.

4. DESIGN DESCRIPTION INFORMATION CONTENT

4.1. INTRODUCTION

Software Design Description of Safwaat identifies how this web application will be designed and implemented. Throughout the document identification, diagrams, user views and user viewpoints are provided.

4.2. SDD IDENTIFICATION

After thorough testing for validation and verification, Safwaat will transition into the real-world environment, offering users an immersive and comprehensive educational experience. Within the application, users will have the capability to engage with a myriad of learning tools and personalized features tailored for mastering Arabic pronunciation and Tajweed skills.

Safwaat's multifaceted platform includes a 'Friendship Hub' fostering connections among users and ensuring secure networking. Moreover, a 'Progress Tracker' will allow users to monitor and showcase their learning milestones via a personalized profile, while features like 'Real-time Discussions' and 'Private Chat Rooms' in the 'ChatSpace' section will facilitate interactive learning experiences.

Additionally, 'Daily Challenges' and 'Quest Arena' rewards will incentivize consistent learning, complemented by an intuitive 'Level Map' for easy navigation through the educational journey. Safwaat aims to cater to learners of all proficiency levels, offering a rich and enjoyable educational expedition where users can connect with peers, monitor their progress, and achieve their Tajweed learning goals seamlessly.

4.3. DESIGN STAKEHOLDERS AND THEIR CONCERNS

Stakeholders for Safwaat encompass users seeking educational content, developers responsible for platform functionality, designers ensuring user-friendly interfaces, educators ensuring educational accuracy, marketers aiming for widespread adoption, quality assurance for seamless user experiences, and management focusing on project success and satisfiability.

In addition to the types of stakeholders, each of them also address their concerns regarding the software product. Users prioritize effective learning tools and data security. Developers aim for a robust, scalable platform. Designers focus on intuitive interfaces and user engagement. Educators ensure accurate educational content. Marketers strategize user engagement, while QA teams ensure seamless experiences. Management oversees project timelines and business alignment.

4.4. DESIGN VIEWS

This project embraces React, harnessing its powerful component-based architecture and functional programming paradigm. The component-based structure simplifies maintenance, allowing for seamless addition, removal, or replacement of components. A notable advantage of this architecture is the ease of global state management, achieved through libraries like Redux, ensuring a centralized and well-maintained global state variable. Furthermore, as the project scales, React's architecture facilitates smooth expansion, providing a modular and organized codebase. Additionally, React's component-based nature simplifies unit testing, making it more straightforward to isolate and test individual components. For instance, when integrating a new activity module, React's design ensures adaptability and easy integration into the existing system, exemplifying its flexibility and scalability.

4.5. DESIGN VIEWPOINTS

The context viewpoint in our Tajweed Application delineates the expected user interactions and stakeholders' roles, establishing a clear system boundary. The user serves as the primary design entity, facilitating information flow between the user and the system, with input-output relations detailed in the design elements. The information view employs UML class diagrams and entity relation diagrams to ensure a persistent representation of data.

Secondly, the composition view outlines team organization, estimated costs, staffing, and scheduling, enhancing project management. Interaction views define service access and service definitions, while the interface viewpoint guides test case development, detailing both external and internal interfaces. State dynamic views utilize diagrams to illustrate state transitions within the application.

4.6. DESIGN ELEMENTS

4.6.1. DESIGN ENTITIES

- User
- FriendRequests
- FriendRequestArchive
- Analytics
- ConsistencyCompanion
- Room
- Messages
- Conversations
- Achievements
- UserAchievements
- Flashcard
- FlashcardSet
- Category
- ExperienceLeaderboard
- ExperienceLeaderboardArchive
- League
- StreakLeaderboard
- LevelMapProgress
- LearningUnit
- TeachingSlides
- 2DModel
- 3DModel
- PracticeSlides
- Topic
- AssessmentMethod
- Questions
- MCQs
- MatchingQuestions
- LabellingQuestions
- DragandDropQuestions
- Challenges
- AchievementsChallenges
- DailyChallenges

4.6.2. DESIGN ATTRIBUTES

Attributes and attributes types of the entities can be seen at entity-relationship diagram in section 5.4. Information Viewpoint.

DEFINITIONS TABLE

TABLE NAME	DESCRIPTION
User	The User entity serves as a central entity which interacts with the entire system. The entity holds information pertaining to the user of the application along with the foreign keys of the entities it interacts with, thereby exhibiting a logical relationship with these entities.
FriendRequests	This entity acquires the details of the friend requests sent by a user to another. It includes the senders id, the receivers id and the status of the request indicating whether the request is accepted or not.
FriendRequestsArchive	The FriendRequestArchive entity serves as a hub for all the friend requests a user sends to other users. It encompasses the requestID attribute of the FriendRequests entity as a foreign key to track the friend requests associated with a particular user.
Analytics	This entity is primarily responsible for recording the user's progress in the application. It interacts with the ConsistencyCompanion entity to retrieve all the analytical information of the user and allow the user to view them in the application as well.
Consistency Companion	The ConsistencyCompanion records the statistical information of the users and tracks their performance. It consists of their earned XP, streak count, lessons learnt, their gems and other essential information, providing a summary of the in achievements acquired in the application.
Room	The Room entity holds the information of the user's joined rooms. It saves the ids of the users present in the group, allowing the users to view or update the status of the Room.
Messages	This entity records all the messages that are exchanged between users in a particular conversation. It includes the conversationId of the Conversation entity as a foreign key to keep track of the messages sent in that conversation, thereby providing data integrity.
Conversations	Conversations refer to the exchange of messages between two users during private chatting. It accomplishes this by recording a list containing the userIds of the two users performing the exchange of messages. This assists in tracking the messages of a conversation as well as retrieving the messages of that conversation.
Achievements	The Achievements entity stores the information regarding the in-app achievements which the user achieves by accomplishing tasks provided by the application.
UserAchievements	UserAchievements holds the personal achievements of a particular user, allowing them to track their own progress and accomplishments.
Flashcard	The Flashcard entity is responsible for storing the data of the flashcards created by the user. It comprises of the term, description, picture and a Boolean to check whether the card is reviewed or not.
FlashcardSet	His entity compiles all the flashcards of a particular category into a single place, allowing ease of access and retrieval of the sets' information when required.
Category	The categories that a user defines for flashcard sets is saved into the Category entity.

ExperienceLeaderboard	This particular entity presents the attributes of a leader board which distributes the registered users according to their earned XP.
ExperienceLeaderboardArchive	This entity holds the archive of a user's performance in a specific leader board.
League	The League entity represents the leagues into which the user can be promoted or demoted. Additionally, it also stores the users which need to be promoted or demoted according to a specific criterion.
StreakLeaderboard	This entity provides the statistical information of the arrangement of all the registered users according to their achieved streak numbers. Hence, it contributes to a healthy competitive environment which promotes user learning paradigms.
LevelMapProgress	The LevelMapProgress entity tracks the progress of a user and the amenities after completion of a particular level.
LearningUnit	A LessonUnit contains informative data of a particular topic which the user wishes to learn. It is comprised of practice slides and teaching slides which further elevates the learning experience of the user.
TeachingSlides	Stores the heading, content, audio, video paths of a particular topic which is being taught in a particular lesson unit.
2DModel	Holds data of 2D model paths used in Teaching slides.
3DModel	Holds the data of 3D model paths used in Teaching slides.
PracticeSlides	Consists of a series of questions and assessment mechanisms to evaluate the learning capability of the user.
Topic	Holds the topic for questions.
AssessmentMethod	Holds the assessment type of questions.
Questions	Holds the topic, assessment type and the question reference.
MCQs	Holds the data of MCQs, question , options and correct option.
MatchingQuestions	Holds the data for Matching Question of Practice Slides.
LabellingQuestions	Holds the data , and diagram link for labelling questions of practice slides
DragandDropQuestions	Holds the data for Drag and Drop Questions of Practice Slides.
AchievementChallenges	This entity describes a particular type of challenge related to the achievements of the user.
DailyChallenges	DailyChallenges provide contextual data for the challenges which a user must accomplish in a single day.
Challenges	This entity constitutes the information regarding a specific challenge which is assigned to a user.

4.6.3. DESIGN RELATIONSHIPS

The relationship of all tables is illustrated in section 5.4. Information Viewpoint

4.6.4. DESIGN CONSTRAINTS

Our Tajweed Application has specific design constraints that influence how it functions and interacts with users. For network constraints, it needs to operate within defined latency limits to ensure real-time responsiveness. A reliable internet connection is crucial for effective feature delivery.

In terms of software, the application's architecture must use cross-platform technologies for consistent execution across different operating systems. Compatibility with popular web browsers like Chrome, Firefox, Safari, and Edge is essential for maintaining uniform features. Performance constraints focus on efficient resource use to prevent excessive CPU or memory usage, ensuring optimal system performance.

The system's design should remain flexible to meet response time targets for user interactions. Additionally, compatibility constraints require maintaining support for older application versions for a smooth transition during updates. These constraints guide the development and optimization of our Tajweed Application.

4.7. DESIGN RATIONALE

We've embraced React's component-based architecture and functional programming paradigm for our Tajweed Application. React's modular structure divides the application into reusable components, each encapsulating its logic, making it easier to manage data control and manipulation. The component-based architecture of React leverages functional programming principles, React allows the creation of pure functions and stateless components, contributing to code simplicity and improved unit testing

We've adopted six key viewpoints for our Tajweed Application, leveraging React's component-based architecture and functional programming paradigm.

- Context Viewpoint:

  Chosen to detail user events in our event-based application, providing a comprehensive understanding of system interactions.

- Logical Viewpoint:

  Selected for its effectiveness in illustrating our system's structure using class diagrams.

- Information Viewpoint:

  Crucial for database management, with a focus on detailed representation through ER Diagrams to enhance system robustness.

- Interface Viewpoint:

  Essential for our web-based application, emphasizing user behaviour. Utilizing user interfaces and component diagrams addresses this aspect effectively.

- Interaction Viewpoint:

  Vital for clear articulation of system reactions to user events like sign-up and sign-in, ensuring transparency for developers and stakeholders.

- State Dynamic Viewpoint:

  Recognizing the significance of state transitions in our web-based application, we selected this viewpoint to navigate the intricacies of states.

4.8. DESIGN LANGUAGES

Unified Modeling Language (UML) is selected as a part of design viewpoint specification.

5. DESIGN VIEWPOINTS

5.1. INTRODUCTION

In this document, six viewpoint are designed for the system as listed below;

- Context Viewpoint
- Logical Viewpoint
- Information Viewpoint
- Interface Viewpoint
- Interaction Viewpoint
- State Dynamic Viewpoint

5.2. CONTEXT VIEWPOINT

There is only one kind of user in our system. The system supplies sixteen services to them, namely Registration, Login, Level Map, Learning Unit, User Profile, Taqraar, Consistency Companion, Notification System, Quest Arena, FlashCraft, SocialMedia Sharing, Leaderboard, FriendshipHub, CHatSpace, 3D Model and Analytics.

A more detailed information about use cases can be found in system requirement specification document.

DESIGN VIEWPOINTS

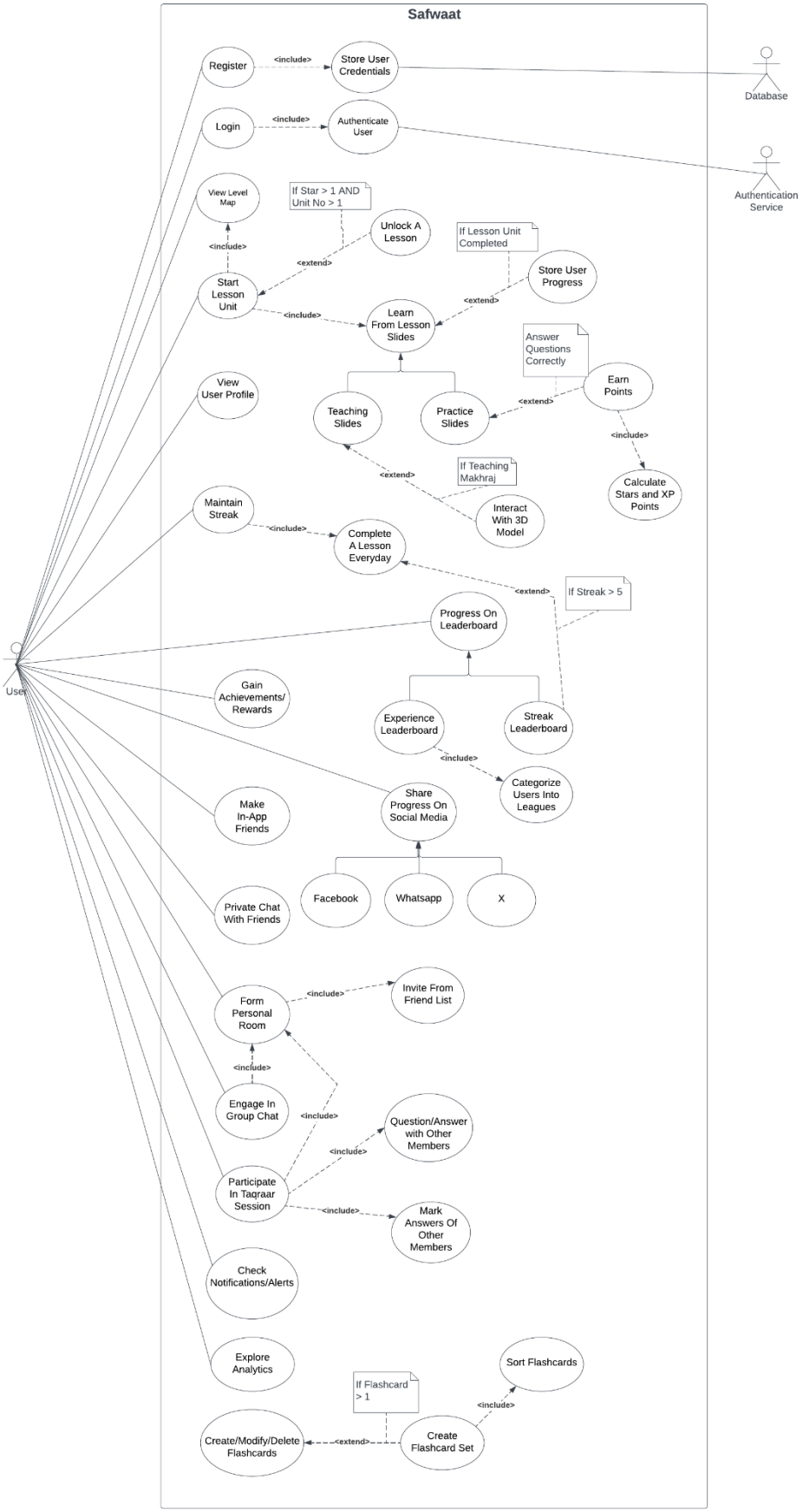


Figure 1. Visualization of Use Case Diagram

5.2.1. Register

The system validates user inputs for correctness and adherence to criteria, displaying error messages for invalid entries and preventing registration until valid input is provided. Upon successful validation, it creates a new user account, ensuring unique usernames to prevent duplication. It provides feedback to the user upon successful registration.

5.2.2. Store User Credentials

The "Store User Credentials in Database" use case encompasses the system's validation of user-provided data like username, password, and email. After confirming the inputs meet the specified criteria, including unique username verification, the system securely stores these credentials in the database. A confirmation message is then displayed to the user, signalling successful registration and safe storage of their information.

5.2.3. Login

The system shall permit the user to enter their username and password to log into their account. It also allows user to login through Google Account.

5.2.4. Authenticate User

The "Authenticate User" use case involves the system's process of verifying a user's identity for access. When a user attempts to log in, they provide their credentials (e.g., username and password). The system then checks these credentials against the stored data in the database, authenticating the user if the input matches the stored information. Upon successful authentication, the system grants access to the user, allowing them to utilize the system's functionalities based on their permissions or role. If authentication fails, the system denies access and prompts the user to retry or seek assistance.

5.2.5. View Level Map

The 'Level Map' use case provides a visual educational journey for users, enabling easy lesson navigation via clickable icons. Users interact efficiently, accessing lesson details and starting tutorials through this map. It enhances learning with intuitive navigation, stores progress data for personalised experiences, simplifies the learning process, motivates users, and supports assessment.

5.2.6. Start Lesson Unit

This use case enables users to commence their educational journey by accessing the lesson content from the Level Map. It acts as an entry point to the structured learning experience, allowing users to navigate and initiate their selected lessons seamlessly.

5.2.7. Unlock A Lesson

Users unlock new lessons within the Start Lesson Unit based on their progression and achievements (Star > 1 and Unit No > 1). This use case ensures a well-paced learning experience, granting access to advanced content as users meet specific proficiency levels.

5.2.8. Learn from Lesson Slides

This use case guides users through the structured educational content of the lessons. It incorporates Teaching Slides, offering concise textual and visual content to convey essential concepts without overwhelming the user. By being a part of the Start Lesson Unit, this use case initiates and supports users' understanding of the lesson material.

5.2.9. Store User Lesson Progress

This use case records and stores user progress within the lesson upon completion of a unit. It ensures that users' learning achievements, points, XP, stars earned, and completion time are securely stored for future reference and tracking.

5.2.10. Teaching Slides

This use case focuses on delivering concise textual and visual content in a structured manner, aimed at helping users grasp essential concepts without feeling overwhelmed. Teaching Slides guide users through fundamental aspects of the lesson content.

5.2.11. Practice Slides

The Practice Slides use case reinforces and applies the knowledge gained from Teaching Slides through diverse assessment methods. Users engage in exercises that require correct responses to advance, with incorrect answers followed by clarifications. These slides foster active learning and skill application.

5.2.12. Earn Points

This use case allows the users to earn points during their practice session and is triggered during practice sessions focusing on teaching Makhraj. It rewards users with points based on their performance. It includes the Calculate Stars and XP Points functionality, motivating users by acknowledging their progress and efforts.

5.2.13. Calculate Stars and XP Points

This use case calculates and attributes stars and XP (Experience Points) to users based on their performance during practice sessions. It quantifies their progress, providing a tangible measure of accomplishment and level progression, thereby motivating users to engage actively in the learning process.

5.2.14. Interact With 3D Mode

This use case allows users to engage interactively with a 3D model embedded within the Teaching Slides. Users interact by providing correct answers to questions, enhancing their understanding and enjoyment of the lesson content.

5.2.15. View User Profile

The User Profile use case allows the user to utilize a wide range of tools aimed at boosting user engagement and customization. It allows users to showcase personal information like name, bio, nationality, and connections. Additionally, it provides options for connecting with friends, monitoring mutual connections, and displaying accomplishments, fostering a vibrant and personalized community experience.

5.2.16. Maintain Streak

This use case is associated with the Consistency Companion feature, focusing on encouraging users to maintain a consistent learning streak. It tracks users' daily completion of lessons, promoting regular engagement within the application to improve Arabic pronunciation and Tajweed skills.

5.2.17. Complete A Lesson Every day

Integrated into the Maintain Streak use case, this functionality prompts users to finish at least one lesson daily. By fostering a routine of daily learning, it reinforces the project's goal of consistent improvement in Arabic pronunciation and Tajweed skills.

5.2.18. Progress on Leaderboard

This use case tracks and displays users' progress within a leaderboard system, showcasing their advancement in learning Arabic pronunciation and Tajweed skills. It aims to motivate users by recognizing their efforts and progress.

5.2.19. Streak Leaderboard

Triggered when users maintain a streak of learning for more than five consecutive days, this use case incentivizes consistent learning habits. It extends from the 'Complete A Lesson Every day' feature and rewards users who sustain a continuous streak of daily lessons.

5.2.20. Experience Leaderboard

Extending the 'Progress on Leaderboard,' this feature emphasizes users' accumulated experience and skill development within the application. It offers a broader view of user proficiency, fostering healthy competition and motivation. Similar to the 'Experience Leaderboard,' this use case focuses specifically on users' streaks of consistent learning. It categorizes users based on their learning streaks, promoting a supportive environment and encouraging regular participation.

5.2.21. Categorize Users Into Leagues

Embedded within the Experience Leaderboard, this use case categorizes users into different leagues based on their proficiency levels and accumulated experience. It fosters a sense of achievement and belonging within specific proficiency groups, enhancing user engagement and motivation in refining Arabic pronunciation and Tajweed skills.

5.2.22. Gain Achievements/Rewards

This use case involves rewarding users with achievements and incentives upon reaching specific milestones or demonstrating exceptional progress in their Arabic pronunciation and Tajweed skills. It fosters a sense of accomplishment and motivation within the application.

5.2.23. Make In-App Friends

Enabling users to connect and interact within the application, this use case facilitates the creation of friendships among users. It aims to build a supportive community, allowing users to share experiences, discuss lessons, and encourage each other in their learning journey.

5.2.24. Private Chat With Friends

Within the application's social framework, this use case provides users with the functionality to engage in private conversations with friends made in-app. It promotes personalized interactions, enabling users to discuss lessons, share insights, and support each other in refining their Arabic pronunciation and Tajweed skills

5.2.25. Share Progress on Social Media

This use case enables users to share their learning progress, achievements, and milestones on various social media platforms. By leveraging platforms like Facebook, WhatsApp, Twitter, and others (represented by use cases 5, 6, and 7), users can showcase their advancements in Arabic pronunciation and Tajweed skills, encouraging others and expanding the application's reach.

5.2.26. Facebook

This particular use case focuses on providing users with the capability to share their learning achievements and progress on the popular social media platform, Facebook. By sharing milestones and accomplishments related to their Arabic pronunciation and Tajweed skills, users can inspire and engage their Facebook network.

5.2.27. WhatsApp

Centered on sharing progress and achievements via WhatsApp, this use case allows users to connect with their contacts directly. It simplifies the sharing process, enabling users to spread awareness and engage their close connections in their learning journey of Arabic pronunciation and Tajweed.

5.2.28. X

Generalized as 'X,' this use case represents the sharing of progress and accomplishments on the microblogging platform 'X'. Users can share snippets of their learning journey, lessons learned, and achievements related to Arabic pronunciation and Tajweed, engaging with a wider audience and fostering discussions on the platform.

5.2.29. Form Personal Room

This use case allows users to create personalized chat rooms within the application's social framework. Users can establish private spaces tailored to specific topics, lessons, or discussions related to Arabic pronunciation and Tajweed. It offers a platform for focused interactions and personalized learning experiences among users.

5.2.30. Invite From Friend List

Enabling users to extend invitations to their friends within the application, this use case facilitates the process of inviting specific contacts to join personalized chat rooms or discussions. It simplifies the interaction process, fostering a sense of community and collaboration among friends interested in refining their Arabic pronunciation and Tajweed skills.

5.2.31. Engage In Group Chat

This use case enables users to participate in group discussions within the application's social environment. Users can interact in group chat rooms created by themselves or their friends, discussing lessons, sharing insights, and supporting each other's learning endeavors regarding Arabic pronunciation and Tajweed. It encourages collaborative learning and fosters a supportive community within the application.

5.2.32. Participate In Taqraar Session

This use case involves users actively engaging in Taqraar sessions within the application. Taqraar sessions include functionalities such as Question/Answer with Other Members and Mark Answers of Other Members. Users participate in discussions, share insights, and engage in collaborative learning to enhance their Arabic pronunciation and Tajweed skills.

5.2.33. Question/Answer with Other Members

Within Taqraar sessions, this use case enables users to ask and respond to questions posed by other participants. It promotes active participation and knowledge sharing, encouraging users to seek clarification and contribute to discussions related to Arabic pronunciation and Tajweed.

5.2.34. Mark Answers of Other Members

This use case allows users to evaluate or mark the responses provided by other participants during Taqraar sessions. It fosters a collaborative learning environment, empowering users to engage in peer-assessment and reinforce their understanding of Arabic pronunciation and Tajweed concepts.

5.2.35. Check Notifications/Alerts

This use case provides users with the functionality to view and manage notifications and alerts within the application. It keeps users informed about new messages, updates, or activities related to their learning progress, ensuring they stay up-to-date with important information and engagements.

5.2.36. Explore Analytics

This use case enables users to access and analyze analytics or insights related to their learning progress, achievements, and engagement within the application. It allows users to track their performance, monitor their advancements in Arabic pronunciation and Tajweed skills, and gain valuable insights to optimize their learning experience.

5.2.37. Create/Modify/Delete Flashcards

This use case empowers users to manage their personalized learning materials by creating, modifying, or deleting flashcards within the application. Users can curate flashcards tailored to Arabic pronunciation and Tajweed rules, allowing for customized study materials to aid in effective learning and skill development.

5.2.38. Create Flashcard Set

When users accumulate more than one flashcard, this use case enables them to organize these flashcards into a set for improved study structure and management. It facilitates the grouping of related flashcards, streamlining the learning process and assisting users in categorizing their study materials effectively.

5.2.39. Sort Flashcards

This use case is embedded within the Create Flashcard Set feature, providing users with the functionality to organize and sort their flashcards systematically. It allows users to arrange flashcards based on specific criteria such as topic, difficulty level, or chronological order, enhancing the efficiency and accessibility of their study materials for Arabic pronunciation and Tajweed skills improvement.

5.3. LOGICAL VIEWPOINT

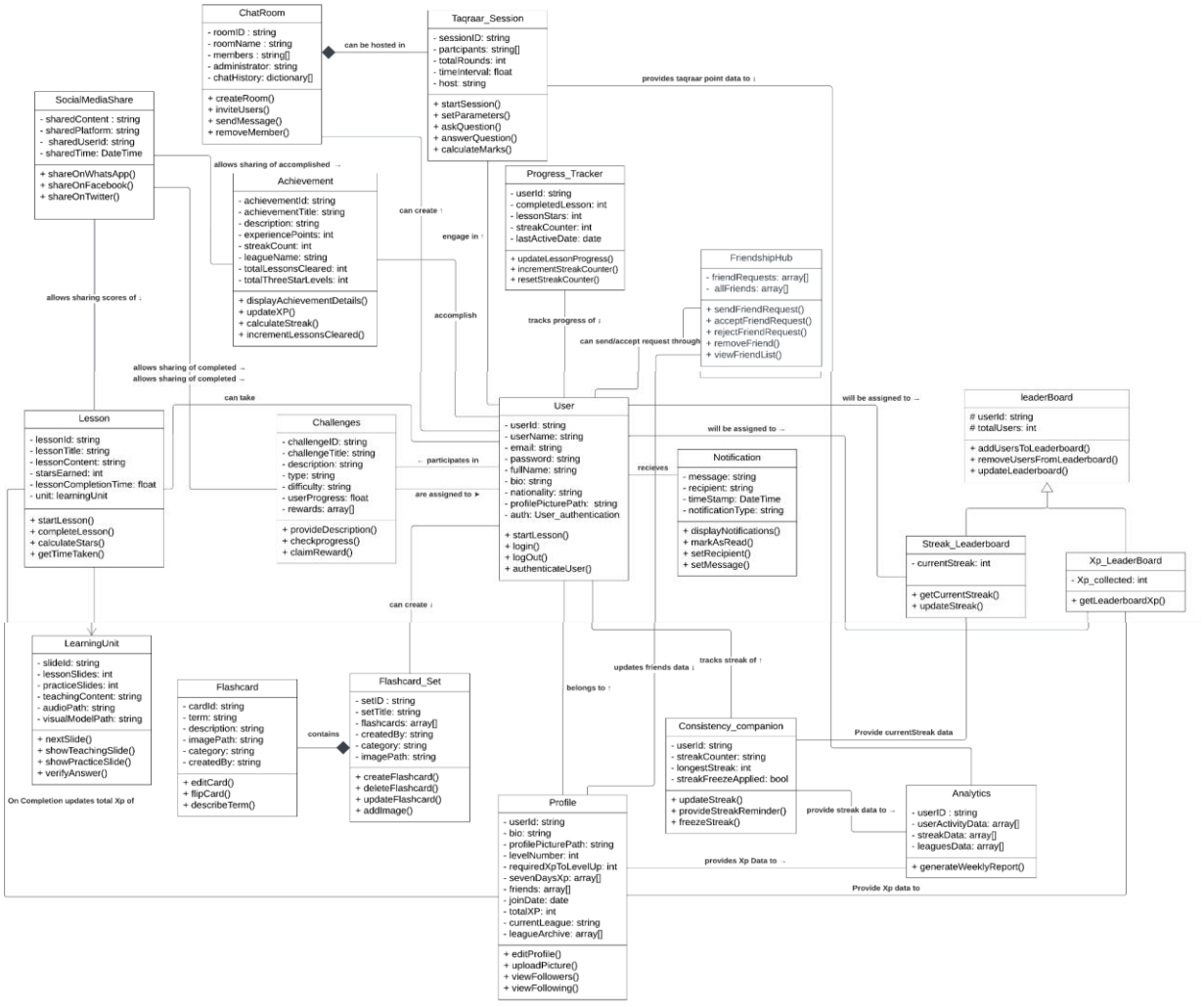


Figure 2. Visualization of Class Diagram

5.3.1. USER CLASS

This class holds information about the user like name, email, password etc. It also has operations which users can perform like starting lesson, logging in and out

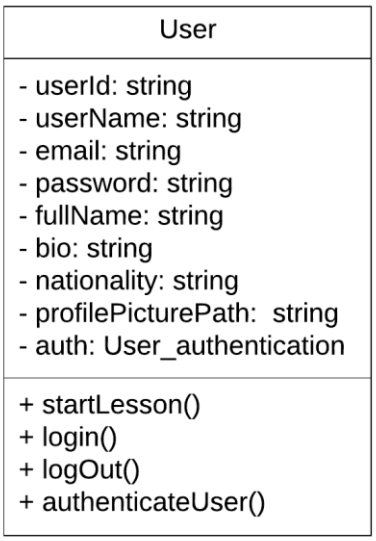


Figure 3. Visualization of User Class Diagram

5.3.1.1. FIELDS

- userId: Unique Integral identifier for the user.
- userName: A string value holding user name.
- email: A string value holding a user's given email address.
- password: A string value holding password entered during registration or login.
- fullName: A string value holding User's full name.
- bio: A long string text holding information about User's biography.
- nationality: A string value holding user's nationality like PK for Pakistan.
- profilePicturePath: A string value holding the file path of the user's profile picture.
- auth: A value holding user’s authentication status and other data like token etc.

5.3.1.2. FUNCTIONS

- startLesson():This function triggers the commencement of a new lesson, preparing the user interface and loading relevant content for an engaging learning experience..
- login(): Facilitating user access, this function verifies login credentials, granting users entry to the Tajweed Application upon successful authentication.
- logout(): Responsible for ending user sessions, this function clears session tokens and updates user status, ensuring a secure logout process.
- authenticateUser(): this function verifies the user's identity through authentication protocols, safeguarding access to sensitive information and features..

5.3.2. CHALLENGES CLASS

This class represents challenges in the system, containing attributes such as challenge ID, type, description, difficulty, user progress, and rewards. It also provides functions for providing challenge descriptions, claiming rewards, and checking progress

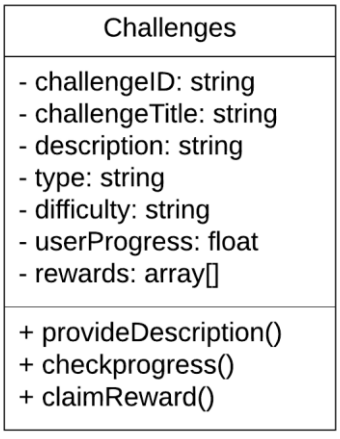


Figure 4. Visualization of Challenges Class Diagram

5.3.2.1 FIELDS

- challengeId: Unique identifier for the challenge.
- type: A string specifying the type of challenge.
- description: A string describing the challenge.
- difficulty: A string indicating the difficulty level of the challenge.
- userProgress: A numerical value representing the user's progress in the challenge.
- rewards: An array containing rewards associated with the challenge.

5.3.2.2 FUNCTIONS

- provideDescription(): This function retrieves and returns the detailed description of a specific challenge, providing users with insights into the challenge requirements.
- claimRewards(): This function allows users to claim rewards they have earned by successfully completing the associated challenge
- checkProgress(): This function assesses and retrieves the current progress of the user in the challenge, offering real-time updates on their journey towards completion.

5.3.3. ACHIEVEMENT CLASS

This class represents achievements in the system, containing attributes such as achievement ID, title, description, experience points, streak count, league name, total lessons cleared, and total three-star levels. It also provides functions for displaying achievement details, updating experience points, calculating streaks, and incrementing lessons cleared.

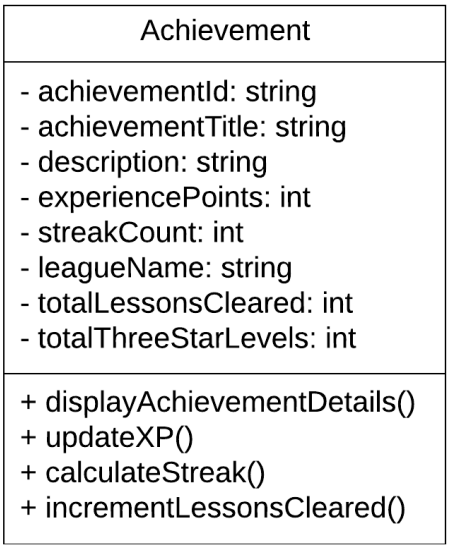


Figure 5. Visualization of Achievement Class Diagram

5.3.3.1 FIELDS

- achievementId: Unique identifier for the achievement.
- achievementTitle: A string representing the title of the achievement.
- description: A string describing the achievement.
- experiencePoints: An integer indicating the experience points associated with the achievement.
- streakCount: An integer representing the streak count for the achievement.
- leagueName: An integer representing the league name associated with the achievement.
- totalLessonCleared: An integer representing the total number of lessons cleared for the achievement.

5.3.3.2 FUNCTIONS

- displayAchievementDetail():This function displays comprehensive information

about a specific achievement, providing users with insights into their accomplishments.

- updateXP(): Responsible for updating the experience points associated with

the achievement, this function ensures accurate tracking of user progress and rewards..

- calculateStreak(): This function calculates and updates the streak count for the

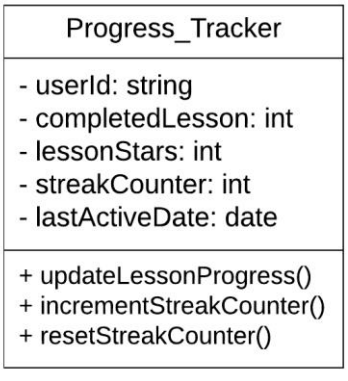
achievement, reflecting the user's consecutive success in meeting certain milestones.

- incrementLessonsCleared(): By incrementing the total lessons cleared count,

this function recognizes and acknowledges the user's continuous learning and progress within the achievement.

5.3.4. PROGRESSTRACKER CLASS

This class represents a progress tracker for users in the system, containing attributes such as user ID, completed lesson, lesson stars, streak counter, and last



active date. It provides functions for updating lesson progress, incrementing the streak counter, and resetting the streak counter.

Figure 6. Visualization of Progress Tracker Class Diagram

5.3.4.1 FIELDS

- userId: Unique identifier for the user associated with the progress tracker.
- completedLesson: A string representing the completed lesson by the user.
- lessonStar: An integer indicating the number of stars earned for completed lessons.
- streakCounter: An integer representing the streak counter for the user.
- lastActiveDate: A date indicating the last active date of the user.

5.3.4.2 FUNCTIONS

- updateLessonProgress(): This function is responsible for updating the user's lesson progress, ensuring accurate tracking of completed lessons and ongoing learning.
- incrementStreakCounter(): By incrementing the streak counter, this function recognizes and rewards users for maintaining a consistent and consecutive learning streak.
- resetStreakCounter(): When invoked, this function resets the streak counter for the user, allowing them to start anew and build a fresh streak in their learning journey.

5.3.5. FRIENDSHIPHUB CLASS

This class represents a hub for managing friendships in the system, containing attributes such as friend requests and a list of all friends. It provides functions for sending friend requests, accepting/rejecting friend requests, removing friends, and viewing the friend list.

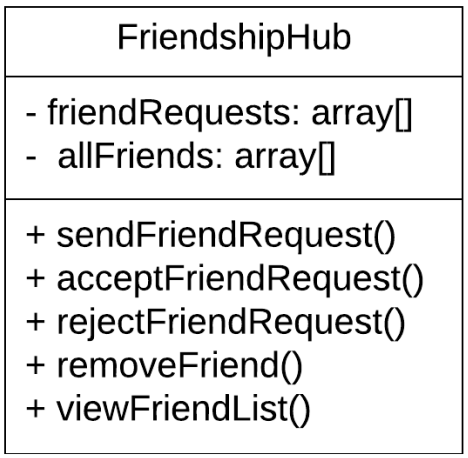


Figure 7. Visualization of FriendHub Class Diagram

5.3.5.1 FIELDS

- friendRequest: An array holding pending friend requests.
- allFriends: An array containing the list of all friends.

5.3.5.2 FUNCTIONS

- sendFriendRequest(): This function triggers the process of sending a friend request to another user, facilitating the establishment of a new connection within the Friendship Hub.
- acceptFriendRequest(): this function acknowledges and accepts a pending friend request, formalizing a friendship and reinforcing social connections within the Friendship Hub..
- rejectFriendRequest(): This function dismisses a pending friend request, signaling a decision not to establish a connection with the user who initiated the request.
- removeFriend(): Activating this function removes a friend from the user's friend list, effectively terminating the established friendship within the Friendship Hub.
- viewFriendList(): By executing this function, users can retrieve and view a comprehensive list of all friends currently linked within the Friendship Hub, enhancing social awareness and connectivity.

5.3.6. NOTIFICATION CLASS

This class represents notifications in the system, including attributes such as the notification message, recipient, timestamp, and notification type. It provides functions for displaying notifications, marking them as read, setting the recipient, and updating the message.

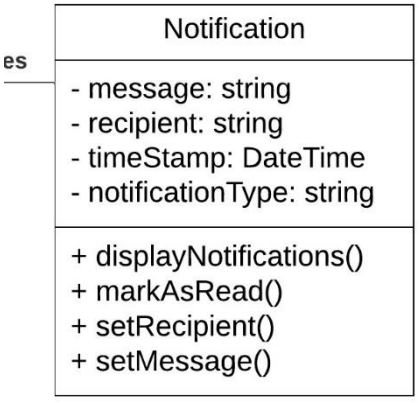


Figure 8. Visualization of Notification Class Diagram

5.3.6.1 FIELDS

- message: A string holding the content of the notification message.
- recipient: A string representing the recipient of the notification.
- timeStamp: A timestamp indicating when the notification was generated.
- notificationType: A string indicating the type or category of the notification.
- receivedMessages : This is the messages that are sent to the user.
- ratingList : This is the list of rating that are done to the user.

5.3.6.2 FUNCTIONS

- displayNotification(): This function triggers the display of a notification message to the user, providing immediate visibility to important updates or information.

- markAsRead(): this function marks the notification as read, signifying that the user has acknowledged and viewed the content of the notification.

- setRecipient(): By executing this function, users can set the recipient of the notification, directing specific messages to intended individuals within the application.

- setMessage(): This function allows users to update or set the content of the notification message, ensuring the flexibility to convey diverse information.

5.3.7. CONSISTENCY COMPANION CLASS

This class represents a companion responsible for tracking and encouraging user consistency, managing streak-related information and providing reminders. It includes attributes such as the user ID, streak counter, longest streak, and whether a streak freeze has been applied. The companion offers functions to update the streak, provide streak reminders, and freeze the streak.

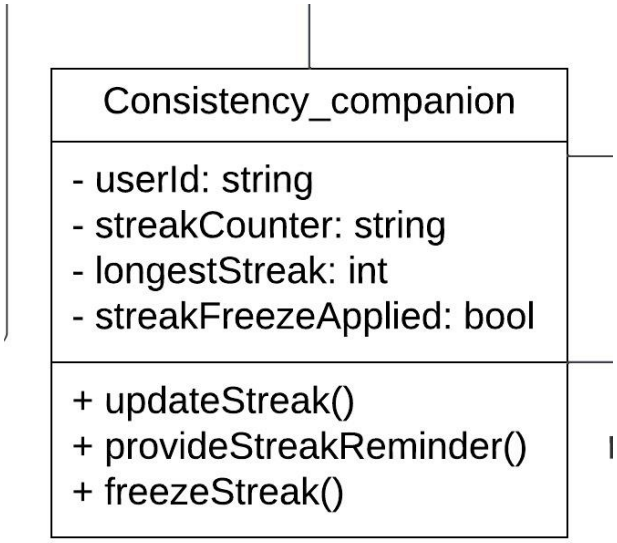


Figure 9. Visualization of Consistency Companion Class Diagram

5.3.7.1 FIELDS

- userId: A unique identifier for the user associated with the companion.
- streakCounter: An integer representing the current streak count.
- longestStreak: An integer indicating the user's longest consecutive streak.
- streakFreezeApplied: A boolean indicating whether a streak freeze has been applied.

5.3.7.2 FUNCTIONS

- updateStreak(): This function dynamically updates the streak counter, reflecting the user's ongoing activity and progress within the application.

- provideStreakReminder(): By executing this function, the application provides timely reminders to users, encouraging them to sustain their streaks

- freezeStreak(): This function allows users to apply a streak freeze, offering a temporary reprieve from maintaining daily activities while still preserving their streak within the system.

5.3.8. ANALYTICS CLASS

This class manages analytical data related to user activity, streaks, and league performance. It includes attributes such as the user ID, user activity data, streak data, and league-specific data. The class provides a function to generate a weekly report based on the collected analytics.

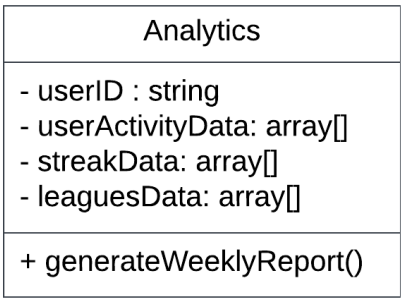


Figure 10. Visualization of Analytics Class Diagram

5.3.8.1 FIELDS

- userID: A unique identifier for the user associated with the analytics.
- userActivityData: An array containing data related to user activities.
- streakData: An array storing information about user streaks.
- leagueData: An array holding data related to user performance in leagues.

5.3.8.2 FUNCTIONS

- GenerateWeeklyReport(): This function compiles and generates a detailed

weekly report, providing users with insights into their activities, streaks, and performance within the application.

5.3.9. PROFILE CLASS

This class represents user profiles, containing attributes that define a user's identity, achievements, and progress. It includes functionalities for editing the profile, uploading a profile picture, and viewing followers and following.

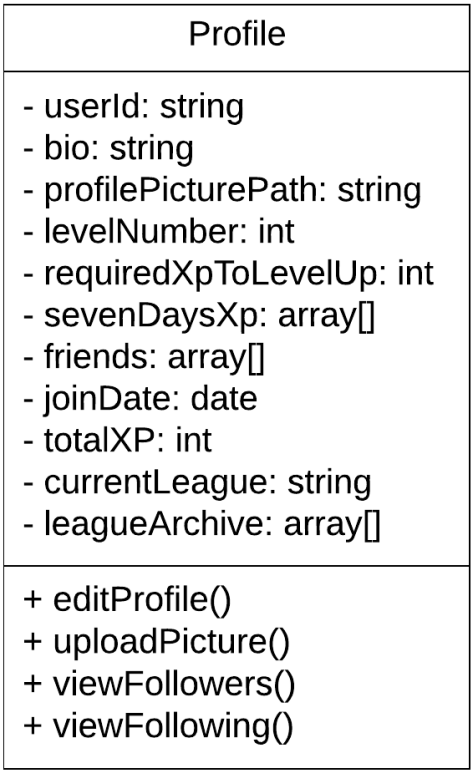


Figure 11. Visualization of Profile Class Diagram

5.3.9.1 FIELDS

- userId: A unique identifier for the user associated with the profile.
- bio: A long string text holding information about the user's biography.
- profilePicturePath: A string value holding the file path of the user's profile picture.
- levelNumber: An integer indicating the user's current level.
- requiredXpToLevelUp: An integer representing the experience points required to level up.
- sevenDaysXp: An array containing XP data for the last seven days.
- friends: An array holding information about the user's friends.
- joinDate: The date when the user joined the application.
- totalXP: An integer indicating the total experience points earned by the user.
- currentLeague: A string representing the user's current league.
- leagueArchive: An array documenting the user's participation in past leagues.

5.3.9.2 FUNCTIONS

- editProfile(): This function facilitates the modification of profile details, ensuring a personalized and up-to-date user experience..
- uploadPicture(): This function enables users to enhance their visual identity by uploading a profile picture, adding a personalized touch to their profile.
- viewFollowers(): this function allows users to gain insights into their follower base, viewing a list of individuals interested in their activities..
- viewFollowing(): this function displays the users whom the profile owner is currently following, fostering a connected community experience.

5.3.10. SOCIAL MEDIA SHARE CLASS

This class represents the sharing of content on various social media platforms. It includes attributes related to the shared content, the platform used for sharing, the user ID of the shared content, and the timestamp of the sharing event. The class provides functionalities for sharing on WhatsApp, Facebook, and Twitter.

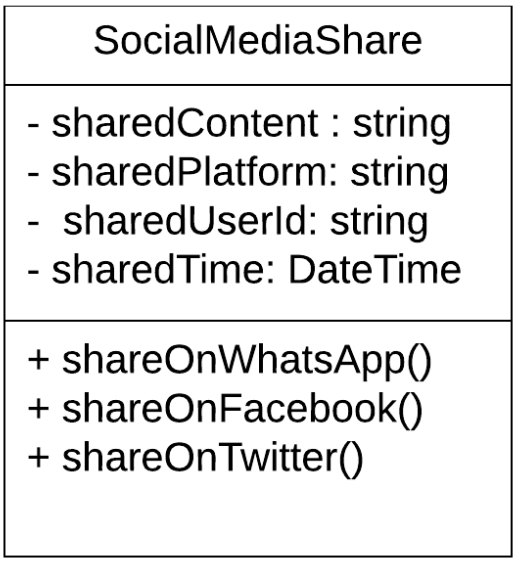


Figure 12. Visualization of Social Media Sharing Class Diagram

5.3.10.1 FIELDS

- sharedContent: A string representing the content shared on social media.
- sharedPlatform: A string indicating the social media platform used for sharing.
- sharedUserId: A unique identifier for the user who shared the content.
- sharedTime: The timestamp indicating when the content was shared.

5.3.10.2 FUNCTIONS

- shareOnWhatsApp(): Initiates the process of sharing content on WhatsApp.
- shareOnFacebook(): Initiates the process of sharing content on Facebook.
- shareOnTwitter(): Initiates the process of sharing content on Twitter.

5.3.11. LESSON CLASS

This class represents a lesson within the learning system. It includes attributes related to the lesson's identifier, title, content, stars earned, completion time, and the learning unit to which it belongs. The class provides functionalities for starting a lesson, completing a lesson, calculating stars earned, and retrieving the time taken to complete the lesson.

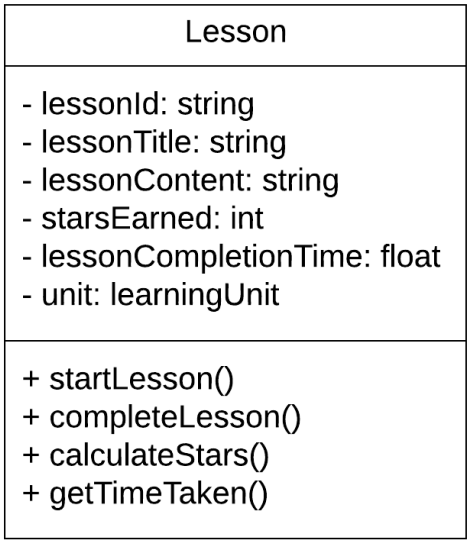


Figure 13. Visualization of Lesson Class Diagram

5.3.11.1 FIELDS

- lessonId: A unique identifier for the lesson.
- lessonTitle: The title of the lesson.
- lessonContent: The content or material covered in the lesson.
- starsEarned: The number of stars earned by the user upon completing the lesson.
- lessonCompletionTime: The time taken by the user to complete the lesson.
- unit: The learning unit to which the lesson belongs.

5.3.11.2 FUNCTIONS

- startLesson(): This function initiates the lesson, marking the beginning of the user's learning process within the application.
- completeLesson(): This function marks the lesson as completed, allowing the system to track user progress and achievements.
- calculateStars(): This function determines the number of stars earned by the user based on their performance during the lesson.
- getTimeTaken(): This function retrieves the time taken by the user to complete the lesson, offering insights into learning efficiency.

5.3.12. LEARNING UNIT CLASS

This class represents a learning unit within the educational system. It includes attributes related to the unit's slide identifier, the number of lesson and practice slides, teaching content, audio path, and the path to a 3D visual model. The class provides functionalities for navigating to the next slide, displaying teaching slides, showing practice slides, and verifying answers.

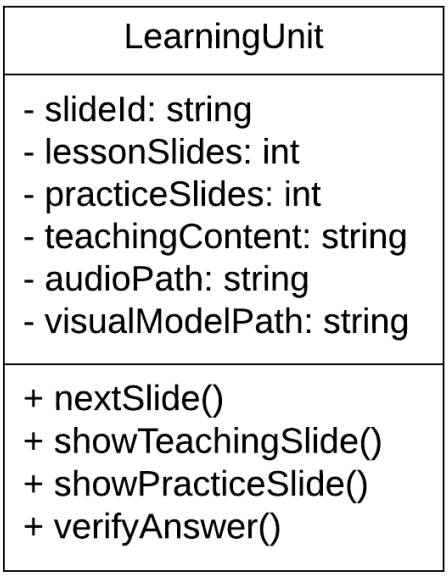


Figure 14. Visualization of Learning Class Diagram

5.3.12.1 FIELDS

- slideId: A unique identifier for the slides within the learning unit.
- lessonSlides: The number of slides containing lesson content.
- practiceSlides: The number of slides designated for practice activities.
- teachingContent: The instructional content provided within the learning unit.
- audioPath: The file path to audio content associated with the learning unit.
- visualModelPath The file path to a 3D visual model for enhanced learning.

5.3.12.2 FUNCTIONS

- nextSlide(): This function advances the learning unit to the next slide, facilitating a sequential flow in the educational content.
- showTeachingSlide(): This function displays the teaching content on the current slide, offering users access to instructional material.
- showPracticeSlide(): This function presents the practice content on the current slide, engaging users in hands-on activities for skill reinforcement.
- verifyAnswer(): This function validates user answers during practice slides, providing feedback on correctness and contributing to the learning assessment

.

5.3.13. TAQRAAR SESSION CLASS

This class represents a Taqraar session within the application, facilitating collaborative learning and discussion. It includes attributes related to the session's identifier, participants, total rounds, time intervals, and the host. The class provides functionalities for starting the session, setting parameters, asking questions, answering questions, and calculating marks.

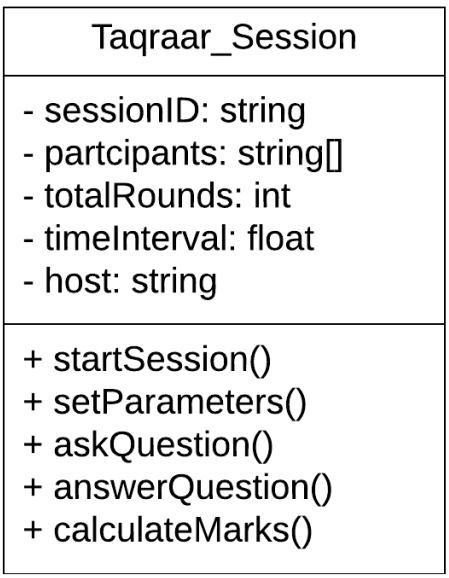


Figure 15. Visualization of Taqraar Session Class Diagram

5.3.13.1 FIELDS

- sessionId: A unique identifier for the Taqraar session.
- participants: An array containing the participants involved in the session.
- totalRounds: The total number of discussion rounds planned for the session.
- timeInterval: The duration of time allocated for each round of discussion.
- host: The user or entity serving as the host for the Taqraar session.

5.3.13.2 FUNCTIONS

- startSession(): This function initiates the Taqraar session, creating a virtual environment for interactive discussions.
- setParameters(): This function allows the host to define session parameters, including round count and time intervals.
- askQuestion(): This function enables the host to pose questions, fostering a structured and purposeful discussion
- answerQuestion(): This function enables users to contribute by providing their answers or insights to the posed questions.
- calculateMarks(): This function computes marks or scores based on participant interactions and responses during the session.

5.3.14. CHAT ROOM CLASS

This class represents a chat room within the application, facilitating communication and interaction among users. It includes attributes related to the room's identifier, name, members, administrator, and chat history. The class provides functionalities for creating a room, inviting users, sending messages, and removing members.

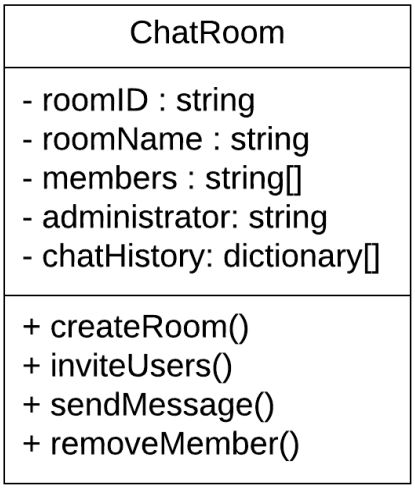


Figure 16. Visualization of Chat Room Class Diagram

5.3.14.1 FIELDS

- roomId: A unique identifier for the chat room.
- roomName: The name or title assigned to the chat room.
- members: An array containing the users who are members of the chat room.
- administrator: The user designated as the administrator or owner of the chat room.
- chatHistory: A data structure storing the history of messages exchanged within the chat room.

5.3.14.2 FUNCTIONS

- createRoom(): This function establishes a new chat room within the application, offering a dedicated space for communication.
- inviteUsers(): This function Invites users to join the chat room, fostering a collaborative and inclusive communication space.
- sendMessage(): This function enables members to send messages within the chat room, facilitating real-time communication.
- removeMember(): This function allows the administrator to manage the chat room by removing specific members.

.

5.3.15. FLASHCARD CLASS

This class represents a flashcard within the application, used for learning and memorization purposes. It includes attributes related to the card's identifier, term, description, associated image, category, and creator. The class provides functionalities for editing the card, flipping the card to reveal additional information, and describing the term.

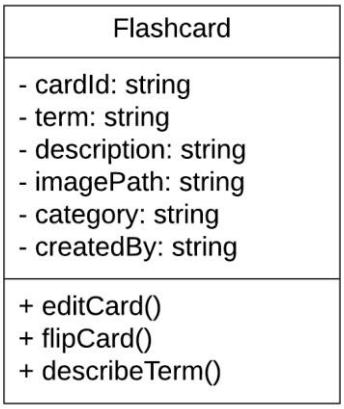


Figure 17. Visualization of FlashCard Class Diagram

5.3.15.1 FIELDS

- cardId: A unique identifier for the flashcard.
- term: The main term or keyword featured on the flashcard.
- description: Additional information or details related to the term on the flashcard.
- imagePath: The file path or URL pointing to an image associated with the flashcard.
- category: The category or subject to which the flashcard belongs.
- createdBy: The user who created or added the flashcard.

5.3.15.2 FUNCTIONS

- editCard(): It enables the user to modify or update the content of the flashcard, ensuring customization according to preferences.
- flipCard(): It simulates the action of flipping the flashcard, providing an engaging and interactive learning experience.
- describeTerm(): This function provides comprehensive details about the term featured on the flashcard, aiding in a deeper understanding of the subject.

5.3.16. FLASHCARD SET CLASS

This class represents a set of flashcards within the application and is composed of individual flashcards. It includes attributes related to the set's identifier, title, associated flashcards, creator, category, and an optional image. The class provides functionalities for creating, deleting, and updating flashcards within the set, as well as adding an image to the set.

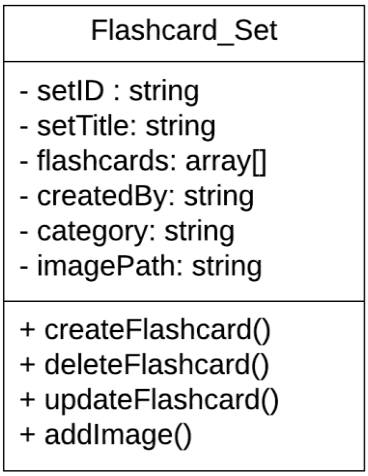


Figure 18. Visualization of FlashCard Set Class Diagram

5.3.16.1 FIELDS

- setId: A unique identifier for the flashcard set.
- setTitle: The title or name of the flashcard set.
- flashcards: An array containing individual flashcards that belong to the set.
- createdBy: The user who created or owns the flashcard set.
- category: The category or subject to which the flashcard set belongs.
- imagePath: The file path or URL pointing to an image associated with the flashcard set.

5.3.16.2 FUNCTIONS

- createFlashcard(): This function allows the user to add a new flashcard to the set.
- deleteFlashcard(): This function facilitates the removal of a flashcard from the set.
- updateFlashcard(): This function provides functionality to modifies the content of an existing flashcard within the set.
- addImage(): This functions allows user to attach an image to the flashcard set.

5.3.17. LEADERBOARD CLASS

This class serves as a superclass for streak and Xp leaderboards within the application. It encompasses attributes and functionalities shared among streak and Xp leaderboards

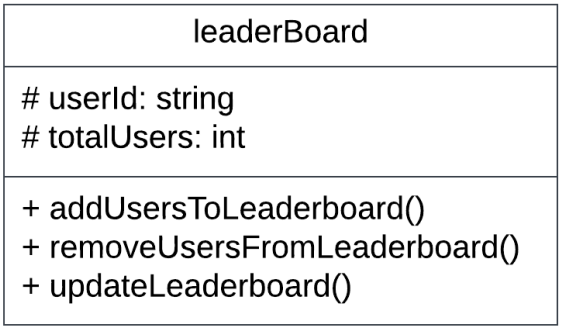


Figure 19. Visualization of Leaderboard Class Diagram

5.3.17.1 FIELDS

- userId: The unique identifier for the user associated with the leaderboard.
- totalUser: The total number of users featured on the leaderboard.

5.3.17.2 FUNCTIONS

- addUsersToLeaderboard(): This function dynamically incorporates users into the leaderboard, ensuring an accurate representation of the active user base.
- removeUsersFromLeaderboard(): It facilitates the removal of users from the leaderboard, maintaining accuracy and relevance.
- updateLeaderboard(): This function conducts real-time updates and recalculations on the leaderboard, ensuring accuracy in user rankings.

5.3.18. STREAK LEADERBOARD CLASS

This class is a subclass of the Leaderboard superclass, specifically tailored for tracking streak-related data on the leaderboard.

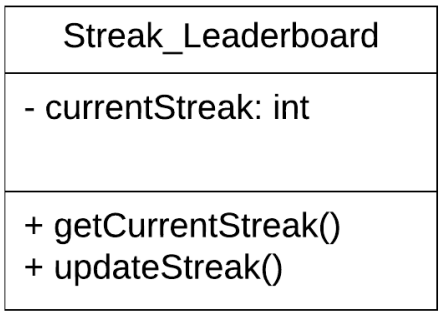


Figure 20. Visualization of Streak Leaderboard Class Diagram

5.3.18.1 FIELDS

- currentStreak**:** The current streak value associated with the user on the streak leaderboard.

5.3.18.2 FUNCTIONS

- getCurrentStreak(): This function enables retrieval of the current streak value for a user on the streak leaderboard.

- updateStreak(): This function dynamically adjusts the streak value based on user activities and interactions.

5.3.19. XP LEADERBOARD CLASS

This class is a subclass of the Leaderboard superclass, focusing on tracking experience points (XP) related data on the leaderboard.

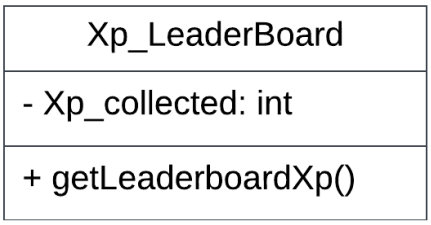


Figure 21. Visualization of XP Leaderboard Class Diagram

5.3.19.1 FIELDS

- xpCollected: The total experience points collected by the user.

5.3.19.2 FUNCTIONS

- getLeaderboardXp(): This function fetches and provides the total experience points for a user on the XP leaderboard contributing to the accurate representation of a user's standing by displaying their accumulated experience points.

5.4. INFORMATION VIEWPOINTS

ER Diagram of the system is illustrated as below and elements and attributes of the diagram is explained in section 4.6.1. Design Entities and 4.6.2. Design Attributes.

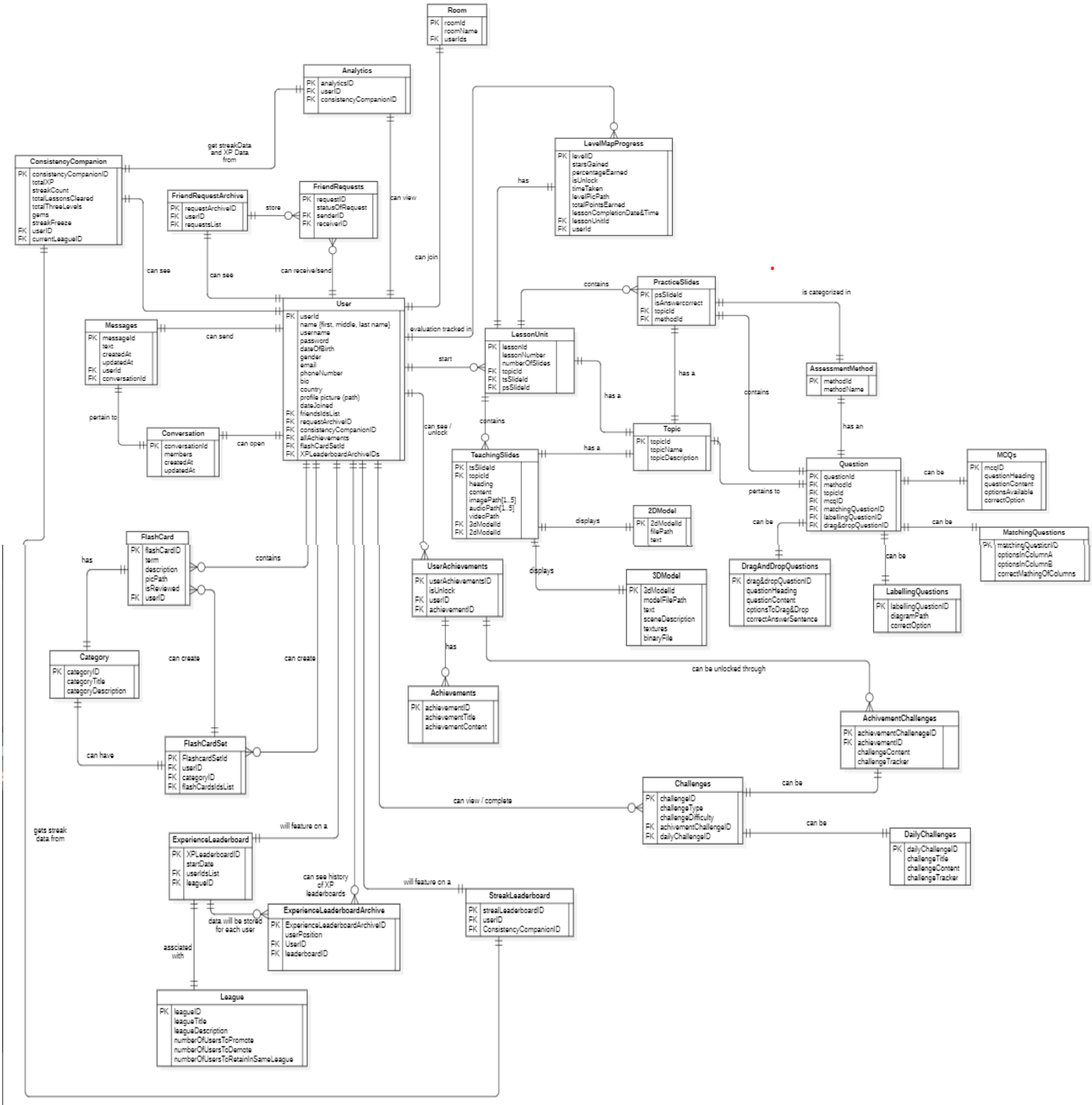


Figure 22. Visualization of ER Diagram

5.5. INTERFACE VIEWPOINTS

5.5.1. SYSTEM INTERFACE

This viewpoint provides details on how to properly utilize the functions that a design object offers. The design contains users, web server, SAFWAAT server, backend server and the Atlas component. The website is connected to user via web browser, and the SAFWAAT application is using APIs from backend server and some third libraries, and the backend server is connected to the Atlas Cloud database. As a result of this connection, the website arises as a MERN stack application, deploying a three-tier architecture.

The component and deployment diagrams for interfaces can be seen as below;

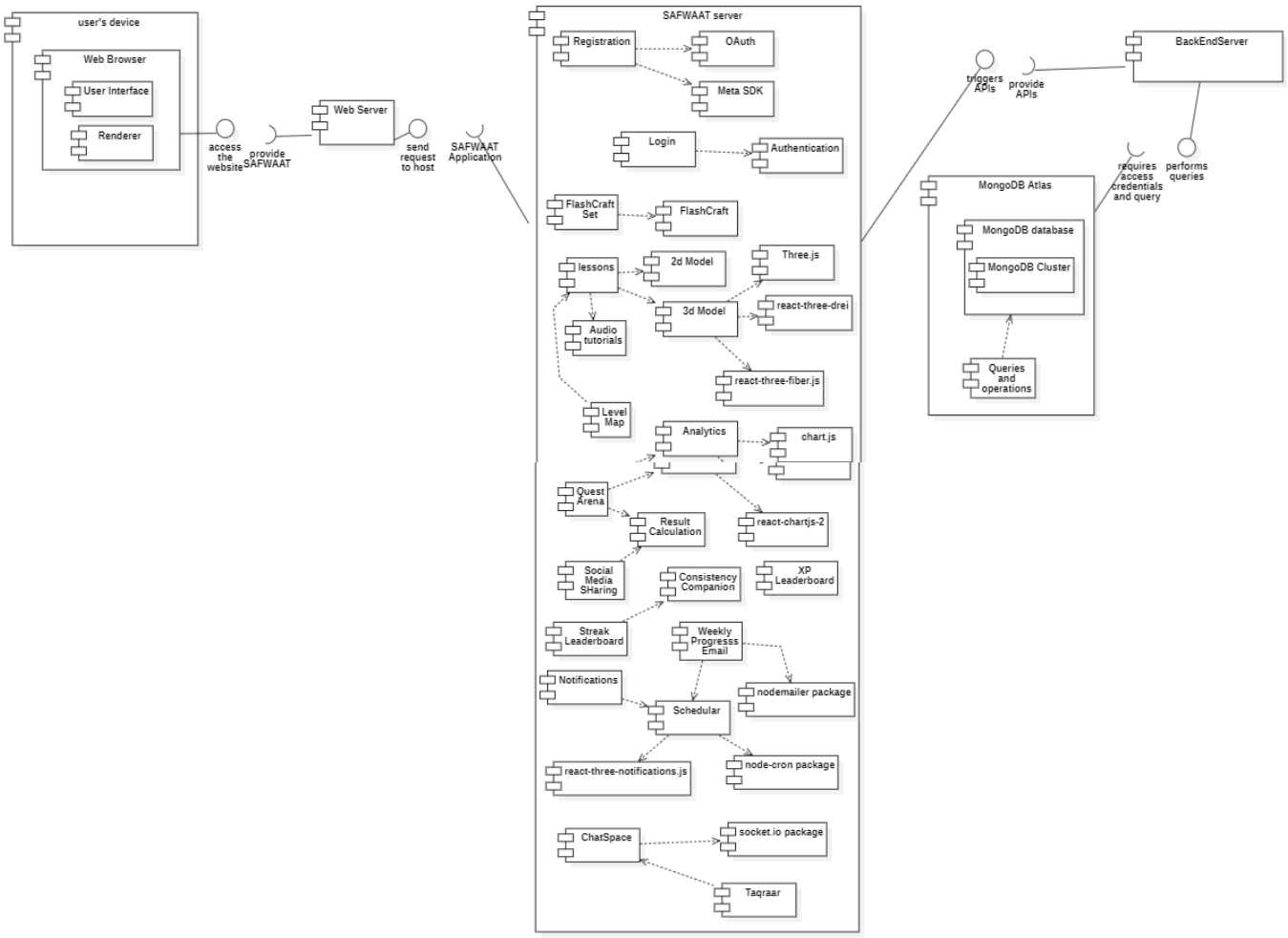


Figure 23. Visualization of component diagram

DESIGN VIEWPOINTS

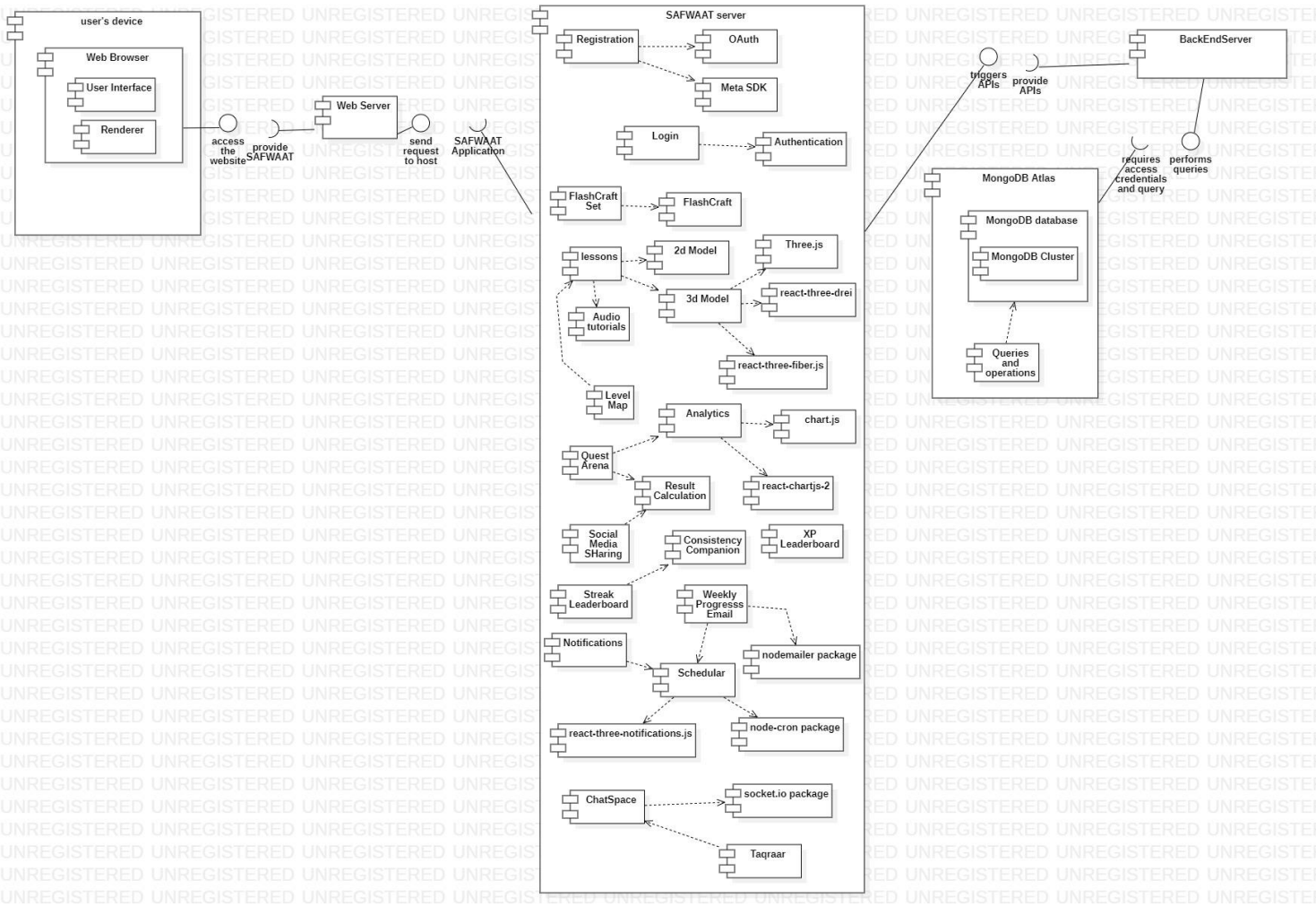


Figure 23.1 Visualization of deployment diagram

5.5.1.1. USER'S DEVICE COMPONENT

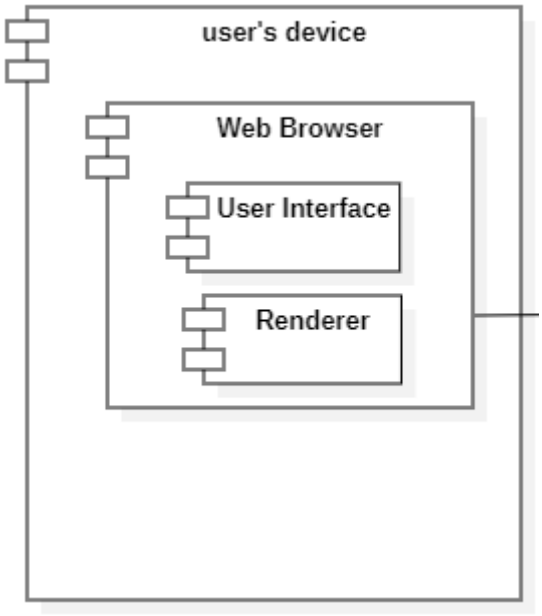


Figure 24. Visualization of user's device component

The user, accessing the web application through a standard web browser, is an external entity represented in the component diagram. Serving as the initiator, the user interacts with the application's frontend components. The communication link between the user and the Web Server Component signifies the exchange of requests (e.g., HTTP requests) and responses.

5.5.1.2. WEB SERVER COMPONENT

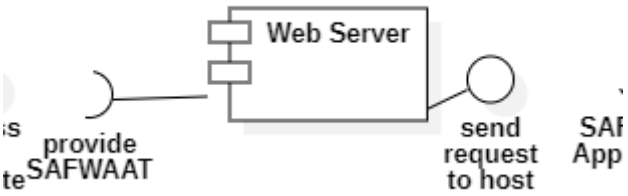
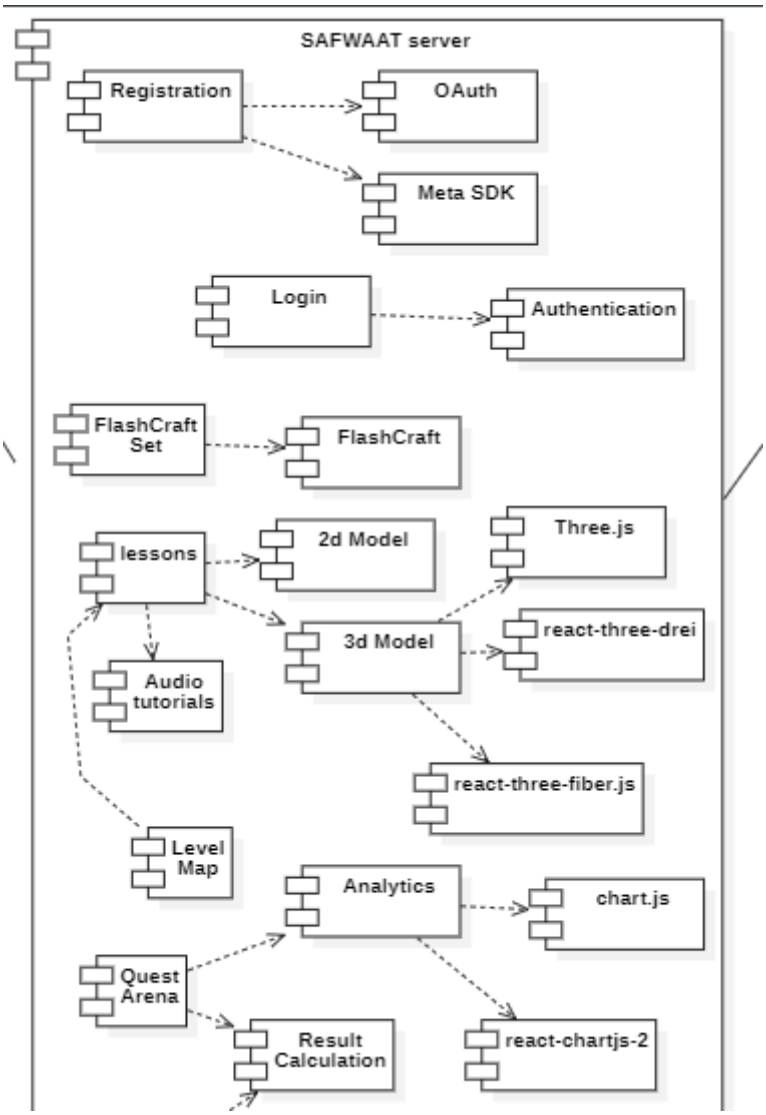


Figure 25. Visualization of Web Server component

It Acts as the entry point for users, facilitating the retrieval and presentation of static content, contributing to the overall user experience.

5.5.1.3. SAFWAAT SERVER COMPONENT



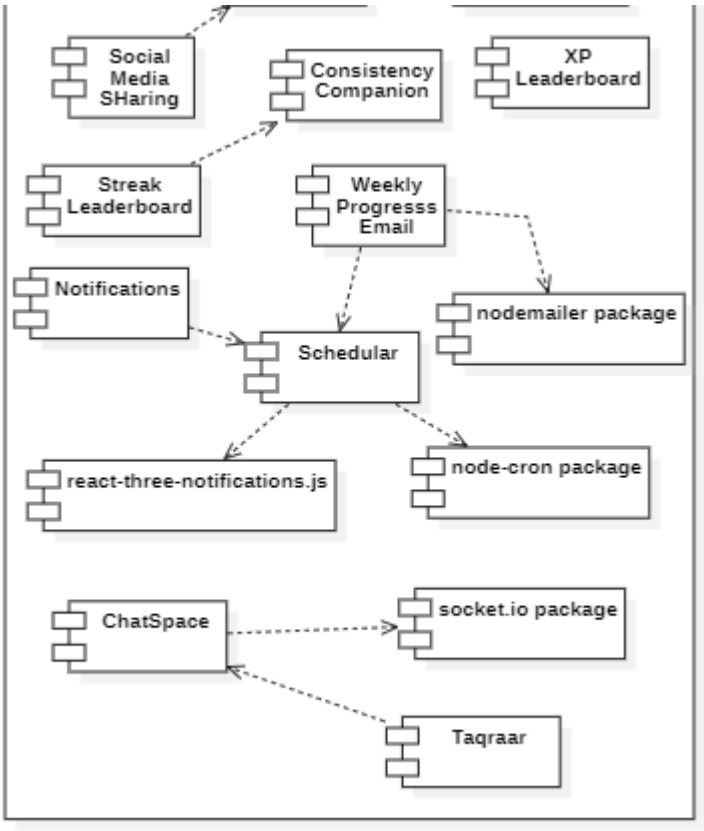


Figure 26. Visualization of SAFWAAT Server component

The "Safwaat Server" component in the diagram serves as the central hub for the SAFWAAT Arabic Learning App. It hosts a variety of features, including user authentication, notification handling, assessment processing, and leaderboard management. This component acts as the bridge to the backend server, facilitating seamless communication between the app and the server to ensure the smooth functioning of all aspects, from user data management to real-time updates and notifications.

5.5.1.4. BACKEND SERVER COMPONENT

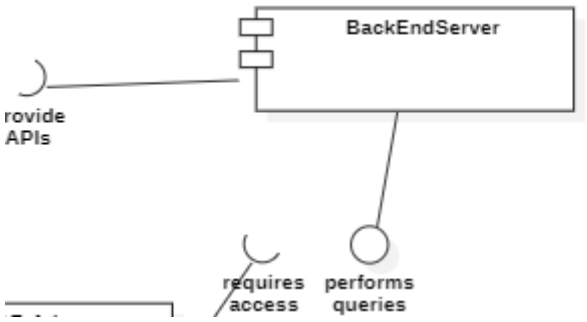


Figure 27. Visualization of Backend Server component

The "Backend Server" component functions as the backbone of the SAFWAAT Arabic Learning App. It supports critical functionalities such as database management, user authentication,

content delivery, and overall system coordination. This component is responsible for handling and processing requests from the Safwaat Server, ensuring secure data storage, and managing the flow of information between the app and the database. It plays a pivotal role in maintaining the integrity and reliability of the application's backend infrastructure.

5.5.1.5. MONGO DB ATLAS COMPONENT

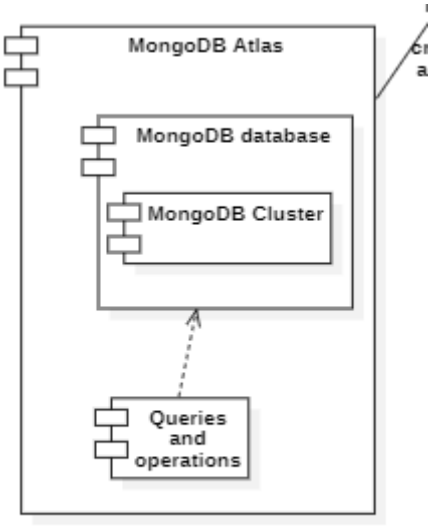


Figure 28. Visualization of MongoDB Atlas component

The "MongoDB Atlas" component serves as the cloud-hosted database solution for the SAFWAAT Arabic Learning App. It is designed to securely store and manage the application's data, offering scalability and reliability. MongoDB Atlas provides a flexible and robust database environment, ensuring seamless access to user information, learning progress, and other essential data points. This component contributes to the overall efficiency and performance of the application by leveraging MongoDB's cloud-based infrastructure for reliable data storage and retrieval.

5.5.2. USER INTERFACE

All of the user interface layouts are in SRS document.

5.5.2.1. COLOR STYLE LIBRARY INTERFACE:

The web must have three types of colors: Primary Color, Secondary Color, and Accent Color, all which all shown.

DESIGN VIEWPOINTS

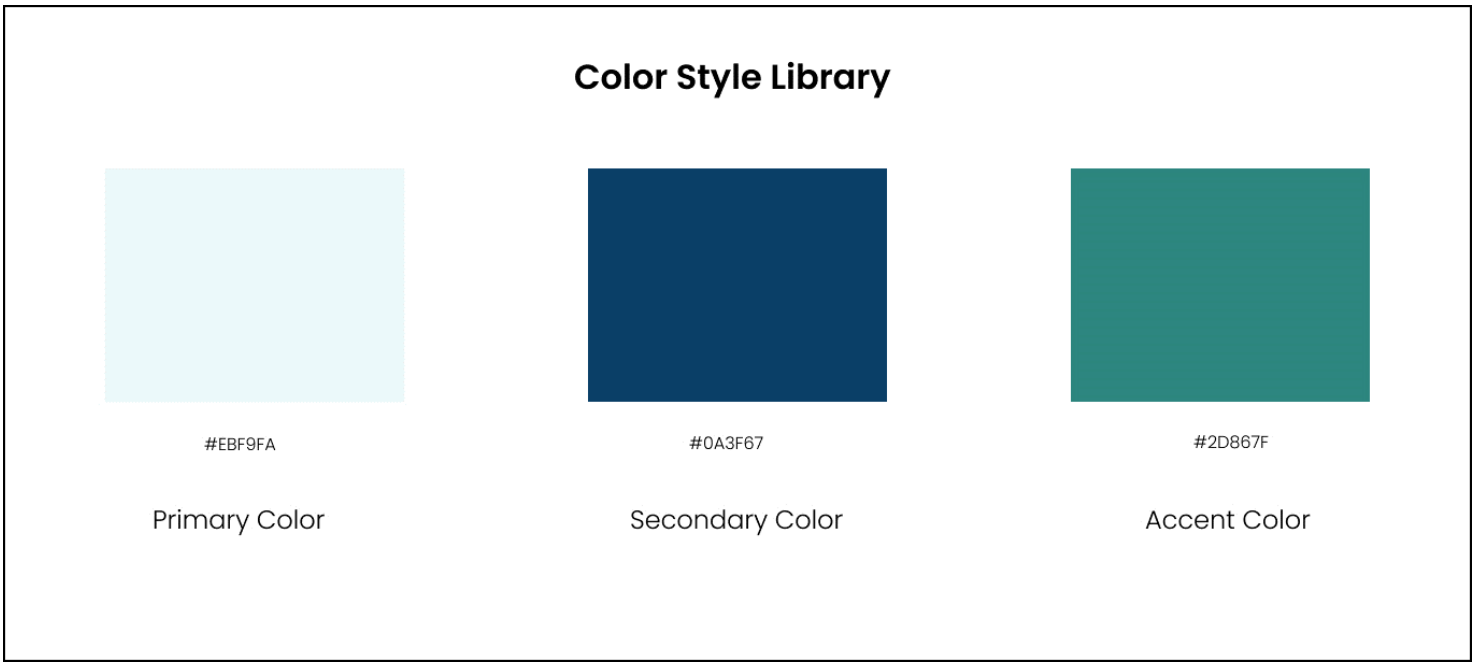


Figure 29. Visualization of Color Library User Interface

5.5.2.2. SIGN UP USER INTERFACE

The Registeration page asks user to fill multiple fields that are; First name, Lastname, Email address, Username, Password, Date of Birth, and the Gender.Additionally, user also has the option to directly sign up via Google Account.

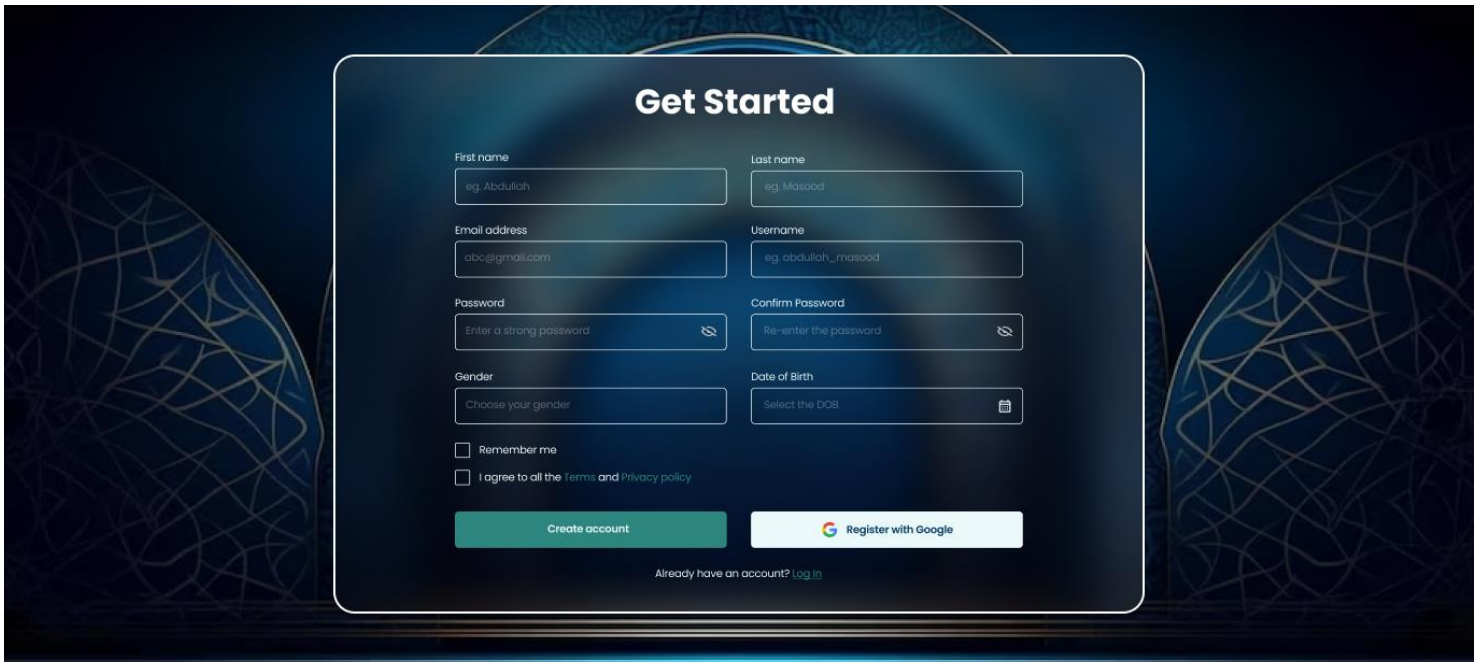


Figure 30. Visualization of Sign Up User Interface

5.5.2.3. CREATE ACCOUNT CLICK STATE

The design shows the state of the page after the user has clicked the 'Create Account' and how the user-filled information will appear.

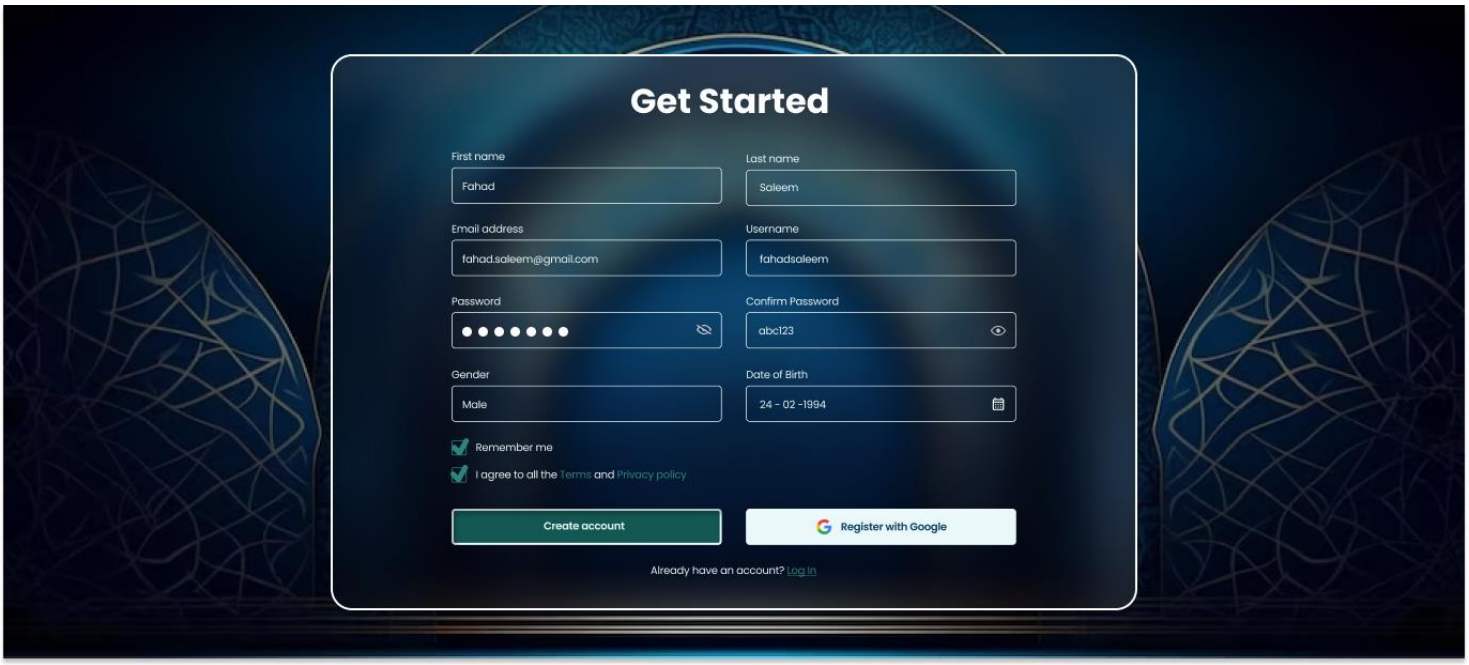


Figure 31. Visualization of create account click state User Interface

5.5.2.4. SIGNUP ERRORS

The UI Design shows how the error will be indicated to the user when user makes error such as: password and confirm password fields are different, or email address does not exist, or account on that email address already exists, or username already exists.

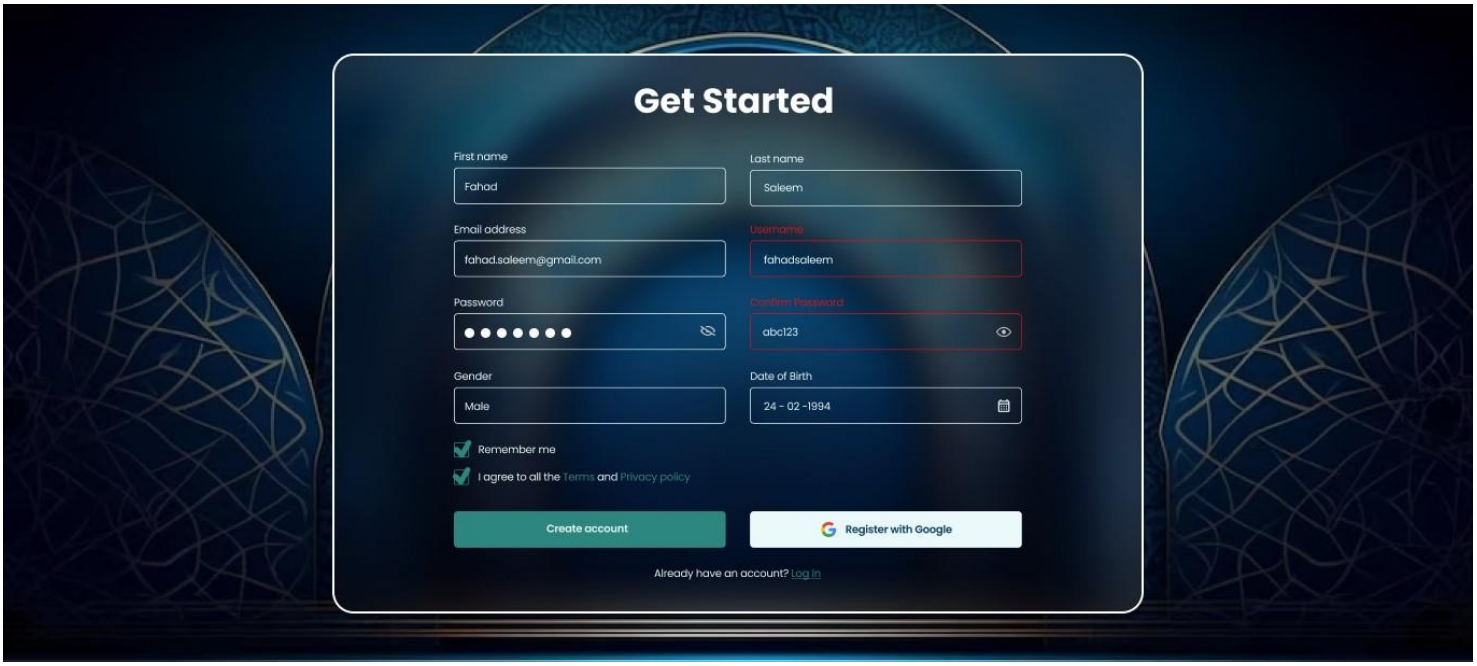


Figure 32. Visualization of Sign Up Errors User Interface

5.5.2.5. LOGIN PAGE

The page asks the user to input the username or the password which will then by verified by the system. User can also click on the eye icon to view the hidden password. Forgot Password option enables the user to rest the password if forgotten. Moreover, user also has the option to Sign in with Google account directly with ease.

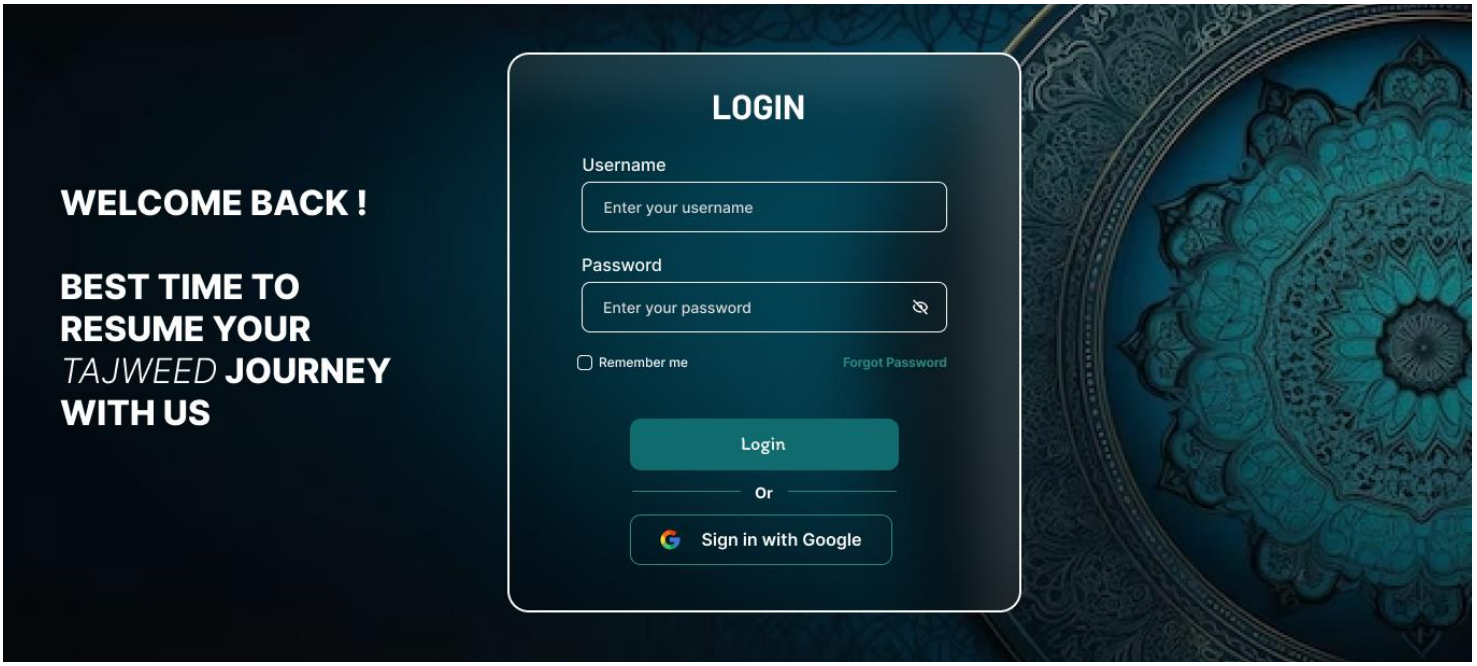


Figure 33. Visualization of Login Page User Interface

5.5.2.6. LOGIN PAGE WITH USER DATA

The design displays how the interface will appear after the user has entered the required credentials.

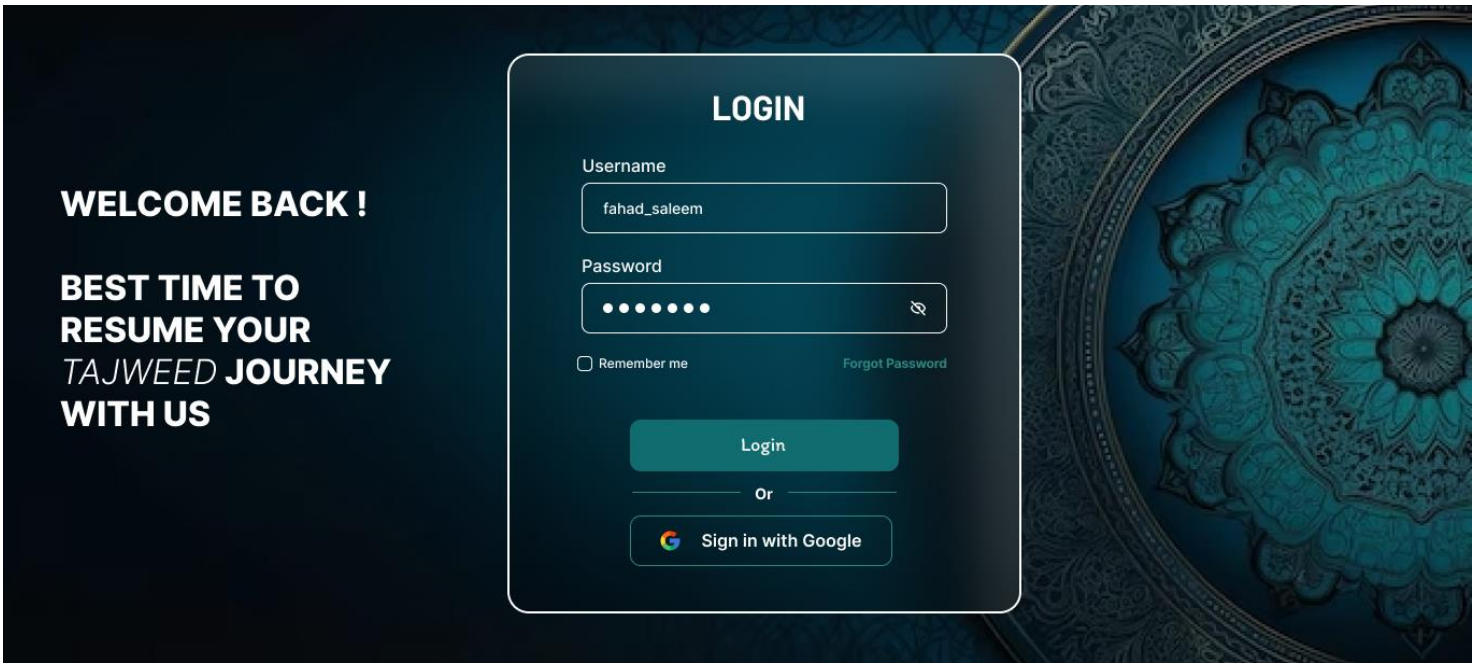


Figure 34. Visualization of Login Page with User Data User Interface

5.5.2.7. LOGIN PAGE WITH ERROR

If user has entered incorrect username or password, it will be highlighted by the red color, and an alert message will be shown.

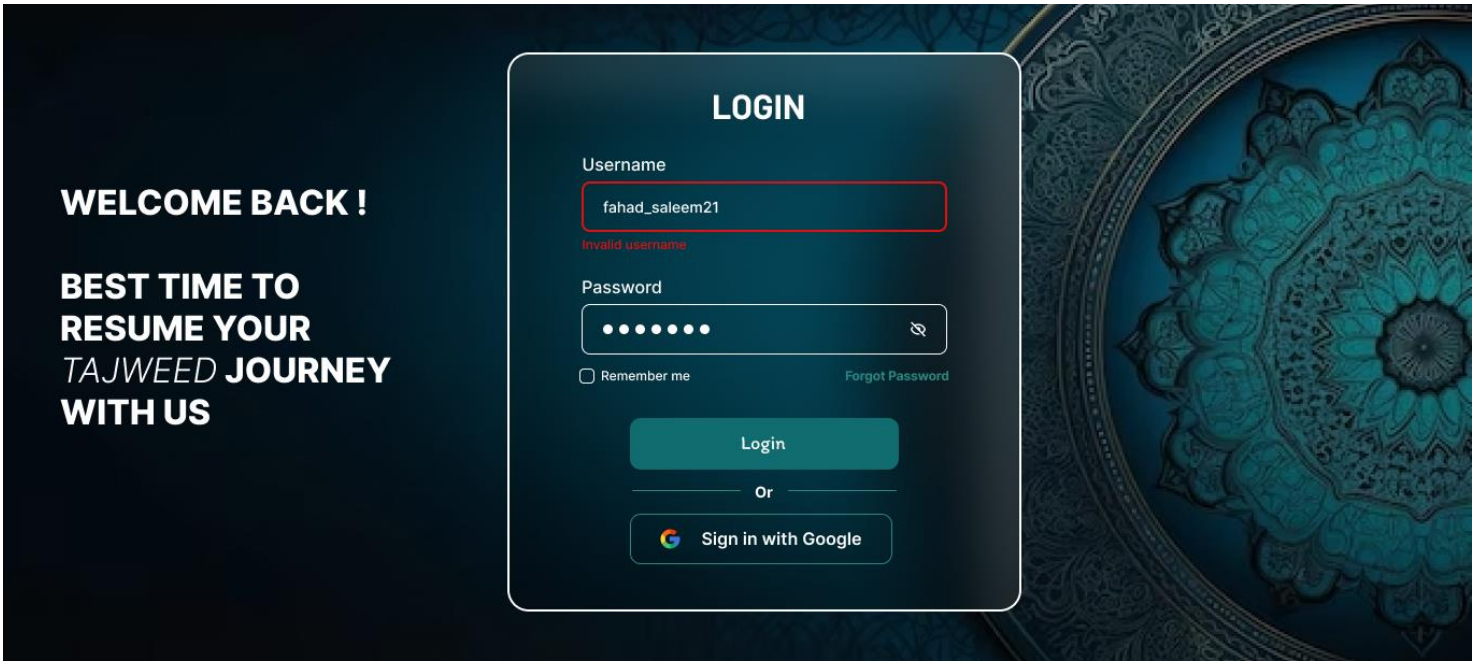


Figure 35. Visualization of Login Page with Error User Interface

5.5.2.8. LEVEL MAP

The design shows how the level map will probably appear in the system. The level map includes scenery and a path which signifies a journey which the user has to cover to reach to their destination of being better at Tajweed. It also shows lesson units, which are clickable, an once the user clicks on one of those, they can start their lesson slides for that unit.

Figure 36.1 Visualization of Level User Interface

5.5.2.9. 3D MODEL

The UI Design demonstrates about one type of 'Teaching Slide' which uses 3D model animation to teach about articulation point. User can click on 'Play' button to start the animation of the model. A dark background is chosen for the slide, for enhanced dark-contrast visibility. The model is interactive, meaning the user can zoom in/zoom out or change the viewing angle of the model anytime with ease.

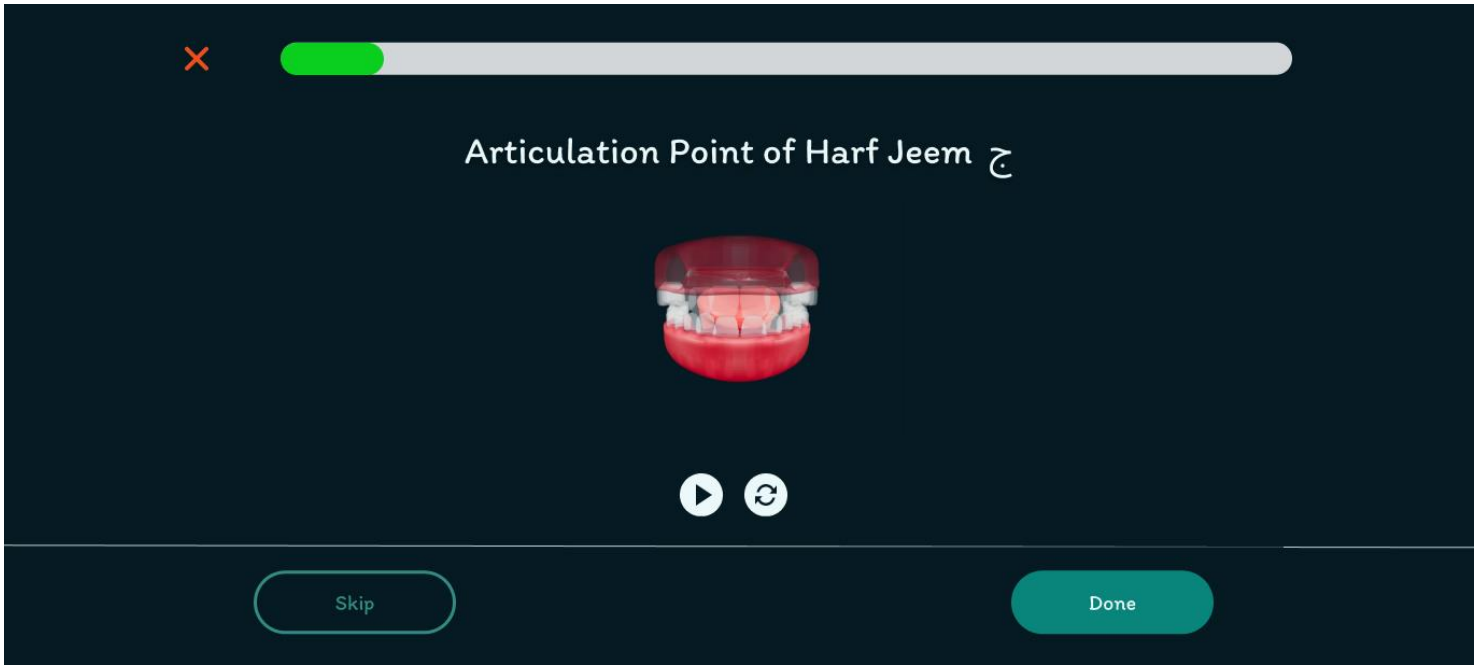


Figure 36.2 Visualization of 3D Model User Interface

5.5.2.10. 3D MODEL PLAYING

It shows the state after the user has played the animation. The model changes accordingly to show precisely where the tongue touches or moves towards for pronouncing a letter.

Figure 37. Visualization of 3D Model Playing User Interface

5.5.2.11. 3D MODEL SIDE

This UI design shows the side of the 3D Model which the user can move to in real-time. Side view would give a clearer perception of the movement of the tongue and the mouth for better understanding and learning. Moreover, the opacity of the teeth of the modal has been reduced immensely otherwise they would create hindrance in viewing of the tongue.

Figure 38. Visualization of 3D Model Side User Interface

5.5.2.12. 3D MODEL UPPER VIEW AND LOOP

This design exhibits that upper palate of the Model is also made transparent to a great extent for enhanced clarity and vision keeping in mind the needs of the users and stakeholders. It also shows the change in 'Loop' button upon being clicked. Loop button will allow user to play the animation continuously.

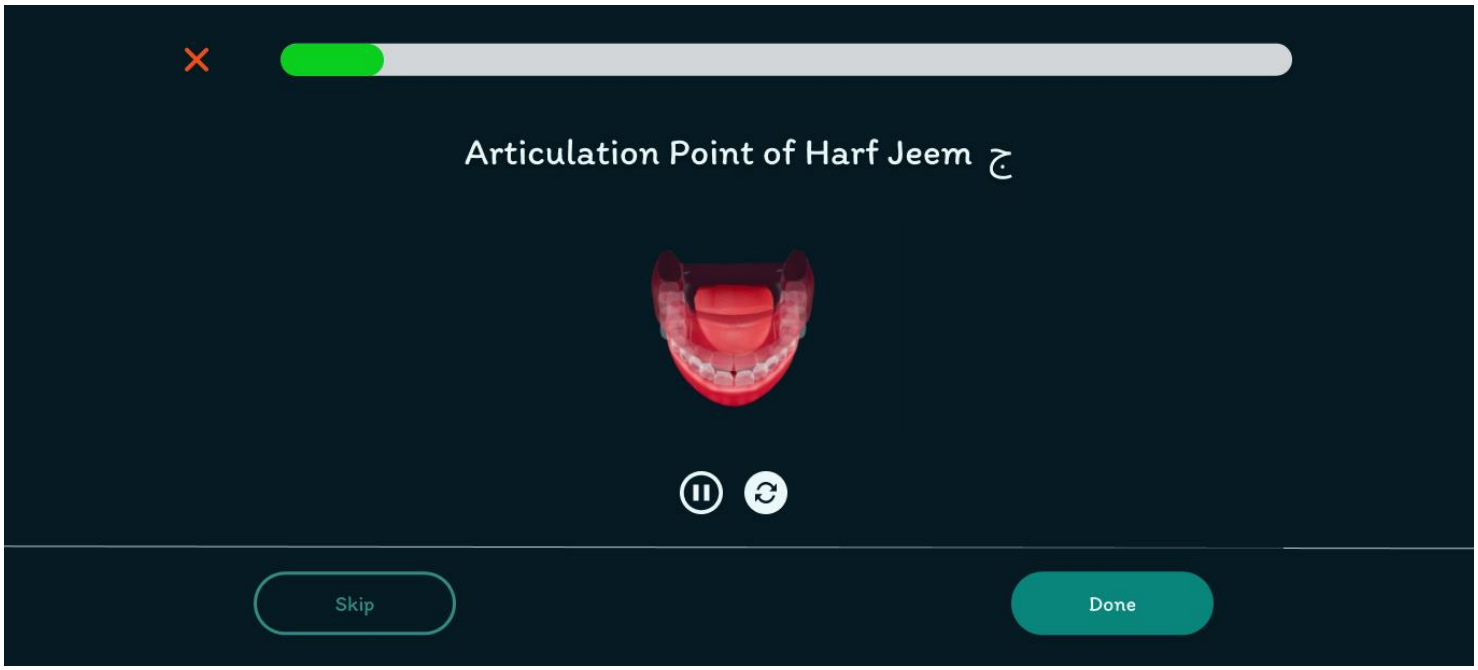


Figure 39. Visualization of 3D Model Upper View and Loop User Interface

5.5.2.13. TEACHING SLIDE

The design illustrates one of the way in which the knowledge is provided to the user to learn, so that subsequent questions in 'Practice' Slides will be based on the information provided to the user.

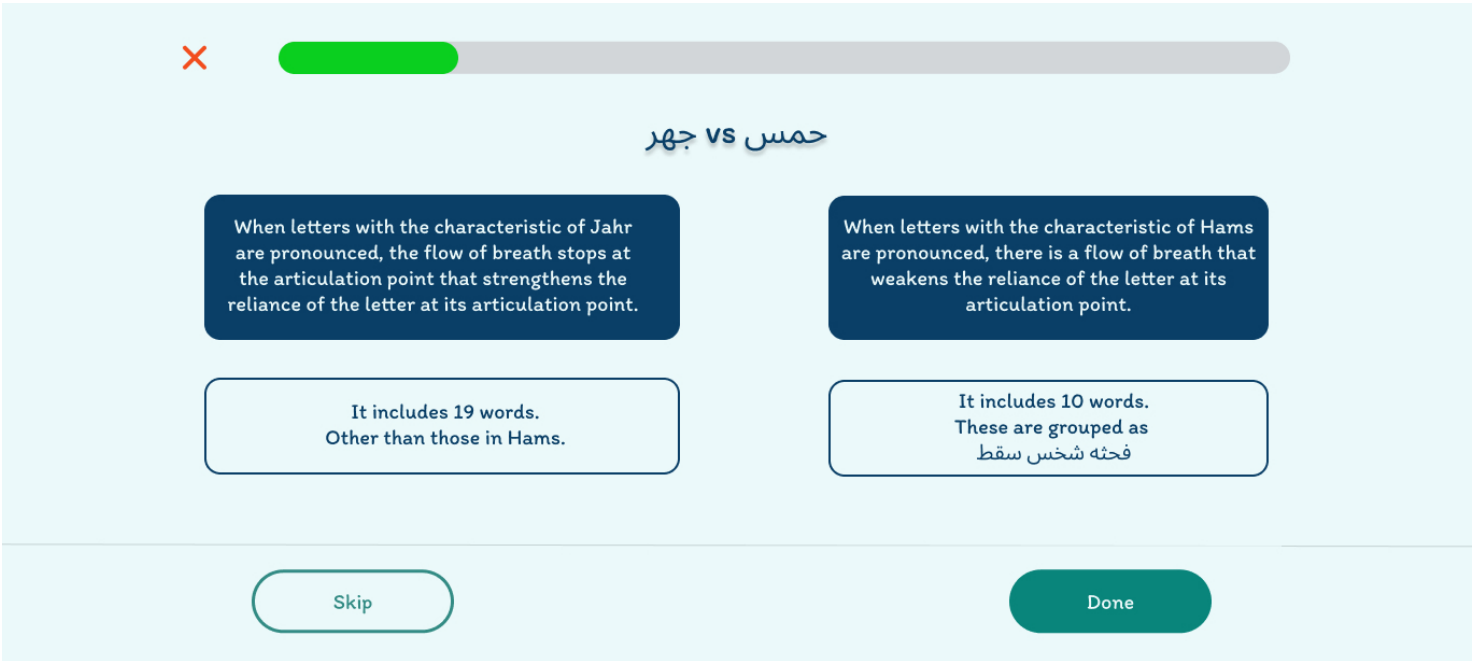


Figure 40. Visualization of teaching Slide User Interface

5.5.2.14. DRAG AND DROP SLIDE

The options will be provided to the user and two boxes can be seen at the bottom. The user can drag each of the option to any of the two boxes wherever they think it fits the best.

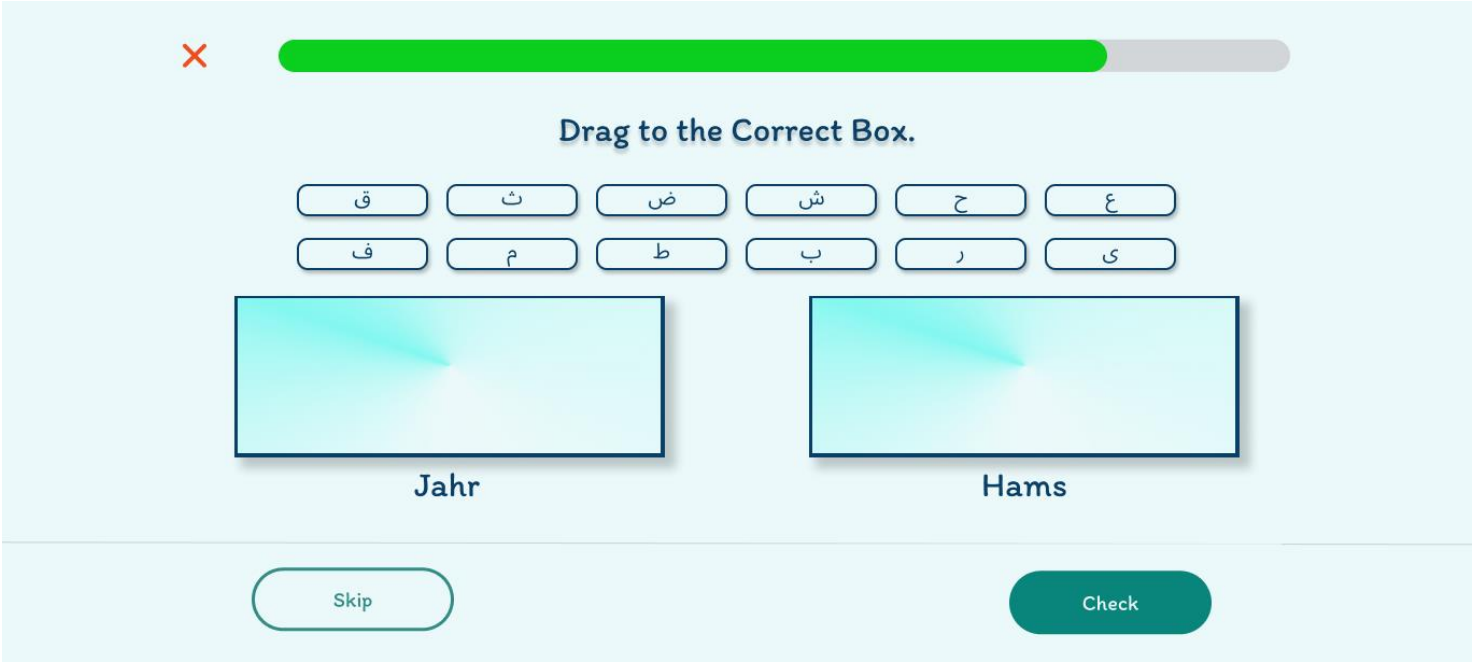


Figure 41. Visualization of Drag and Drop Slide User Interface

5.5.2.15. DRAG AND DROP SLIDE COMPLETED

After the user has dragged all the options, the UI will appear in this manner where each option is placed wherever the user has dragged them. The Check button is now enabled so that can proceed with the testing of the answers given.

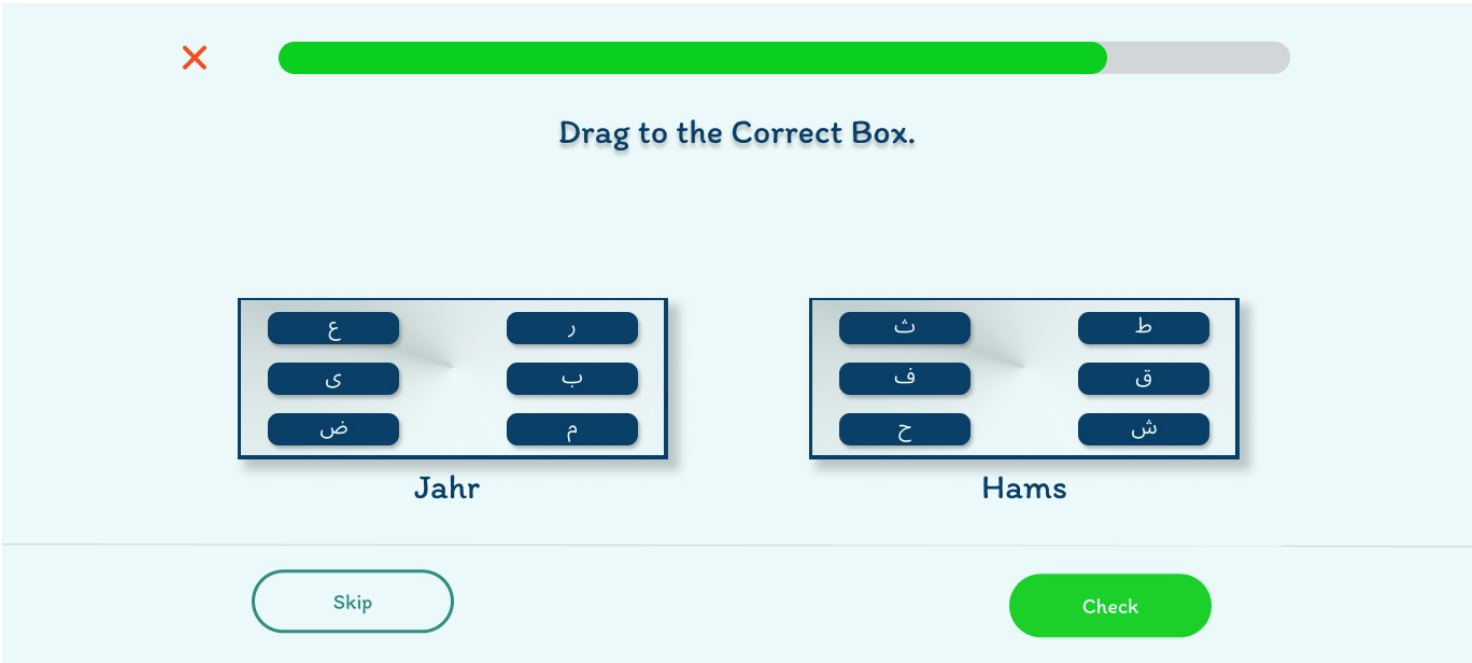


Figure 42. Visualization of Drag and Drop Slide Completed User Interface

5.5.2.16. DRAG AND DROP SLIDE CORRECT

Once the user has requested the answered to be checked, if all are correct, the UI view would appear like this where green highlights all the correct answers and a 'Continue' button to proceed to the next slide.

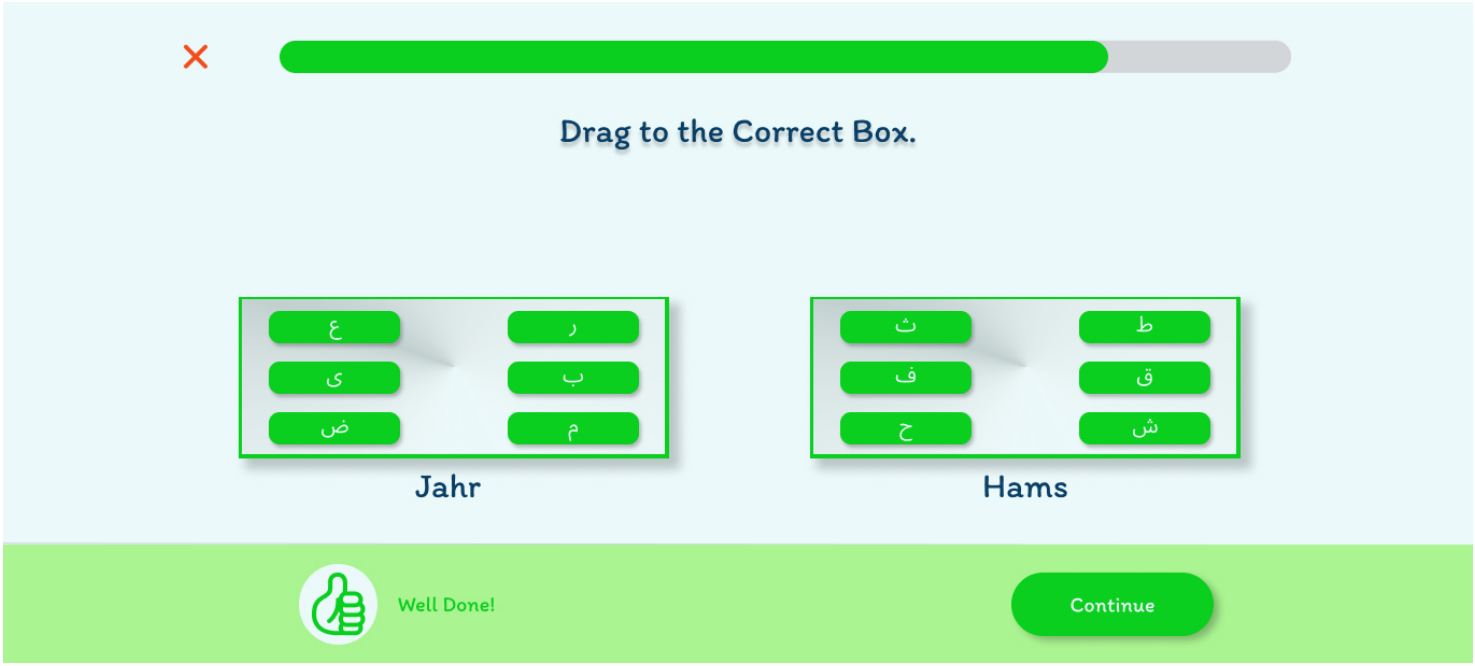


Figure 43. Visualization of Drag and Drop Slide User Correct Interface

5.5.2.17. DRAG AND DROP SLIDE WRONG

Upon wrong answers, the UI will vividly highlight the wrong answers by red so that users can know their errors to improve and perform better next time.

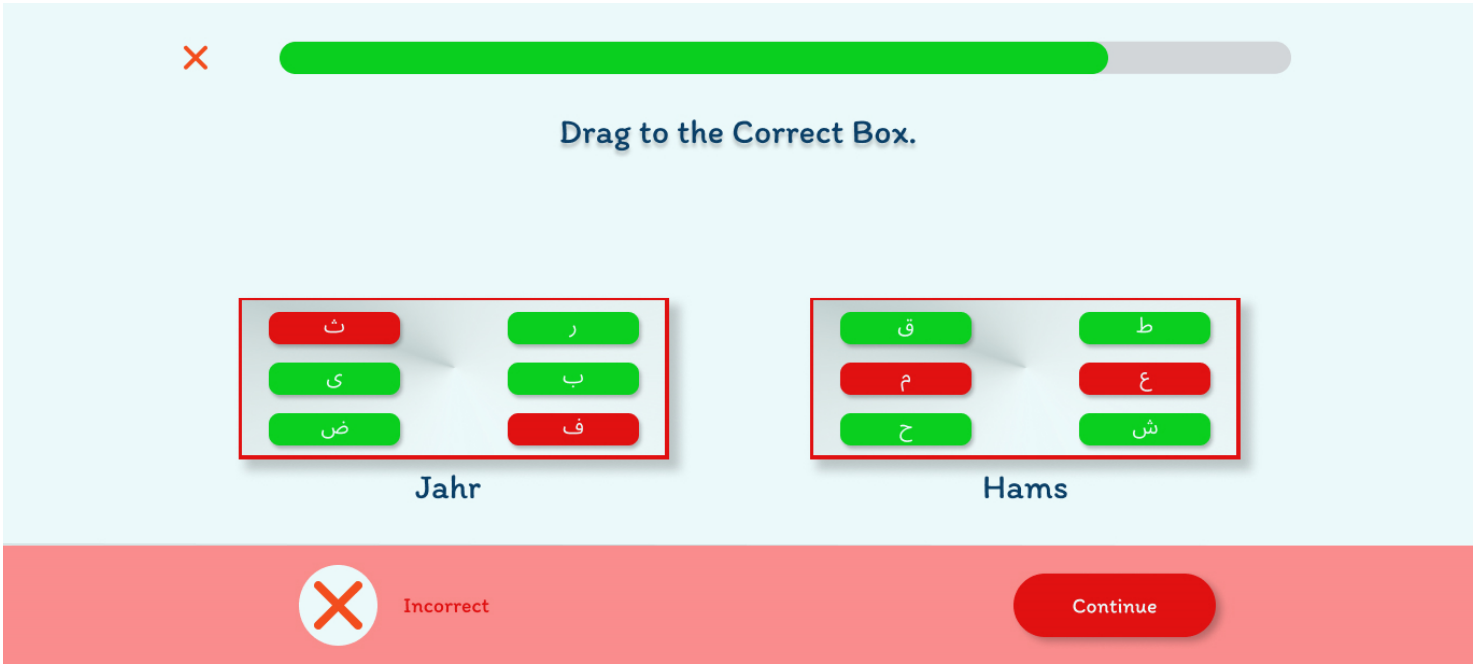


Figure 44. Visualization of Drag and Drop Slide Wrong User Interface

5.5.2.18. PRACTICE SLIDE MCQ

The design shows multiple options for a question one which is correct. The user can click and choose any one of the answers and then click on 'Check' Button to let system check the answer. User can also choose to click on 'Skip' button in which case the slide will be skipped and would be marked wrong.

At the top, progress bar is shown, which gives an idea of how much portion of the lesson unit is covered by the user at that point in time.

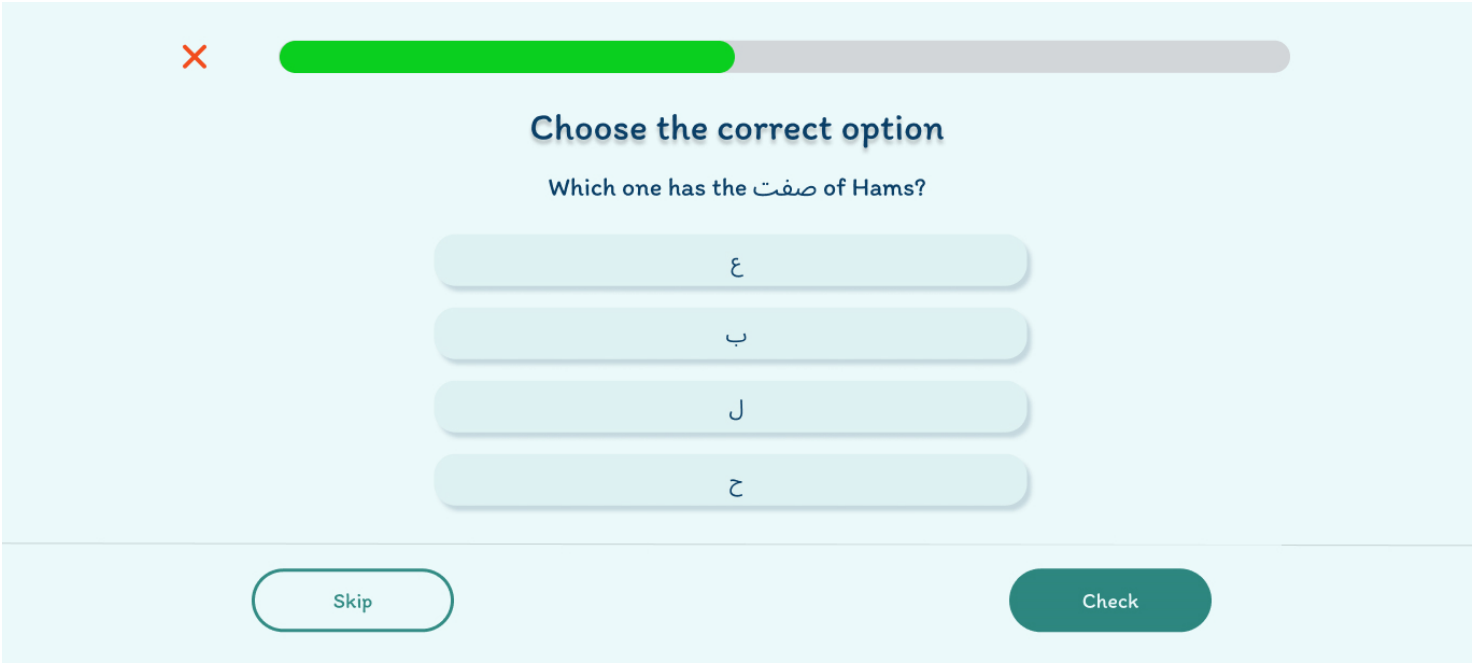


Figure 45. Visualization of Practice Slide MCQ User Interface

5.5.2.19. PRACTICE SLIDE MCQ SELECTED

It shows how the page will appear once the user has selected a particular option. When an option is selected the state of the 'Check' button will be changed from disabled to active state which is highlighed by a change to a brighter color, indicating the user that they can know check their answer.

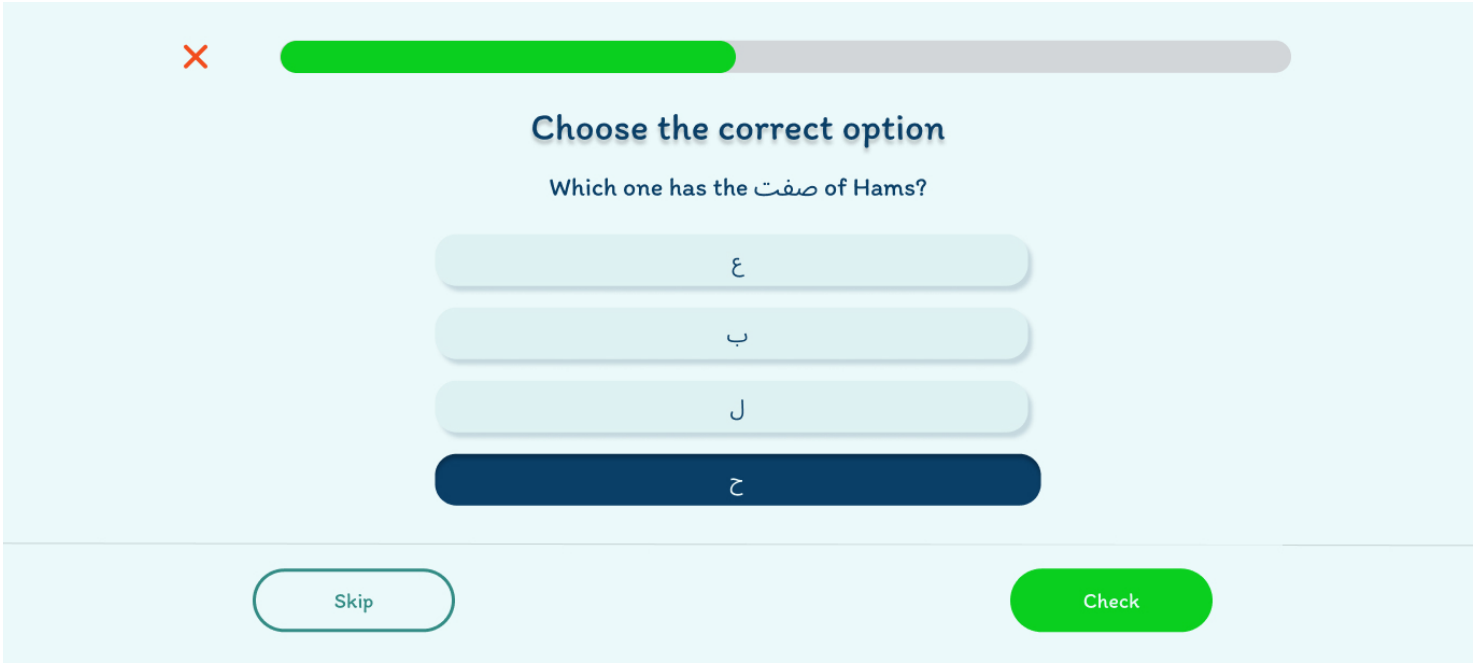


Figure 46. Visualization of Practice Slide MCQ Selected User Interface

5.5.2.20. PRACTICE SLIDE MCQ CORRECT

The design shows the appearance of the webpage after the user has answered the question correctly. The user would be appreciated and would be given an option to continue to the next slide.

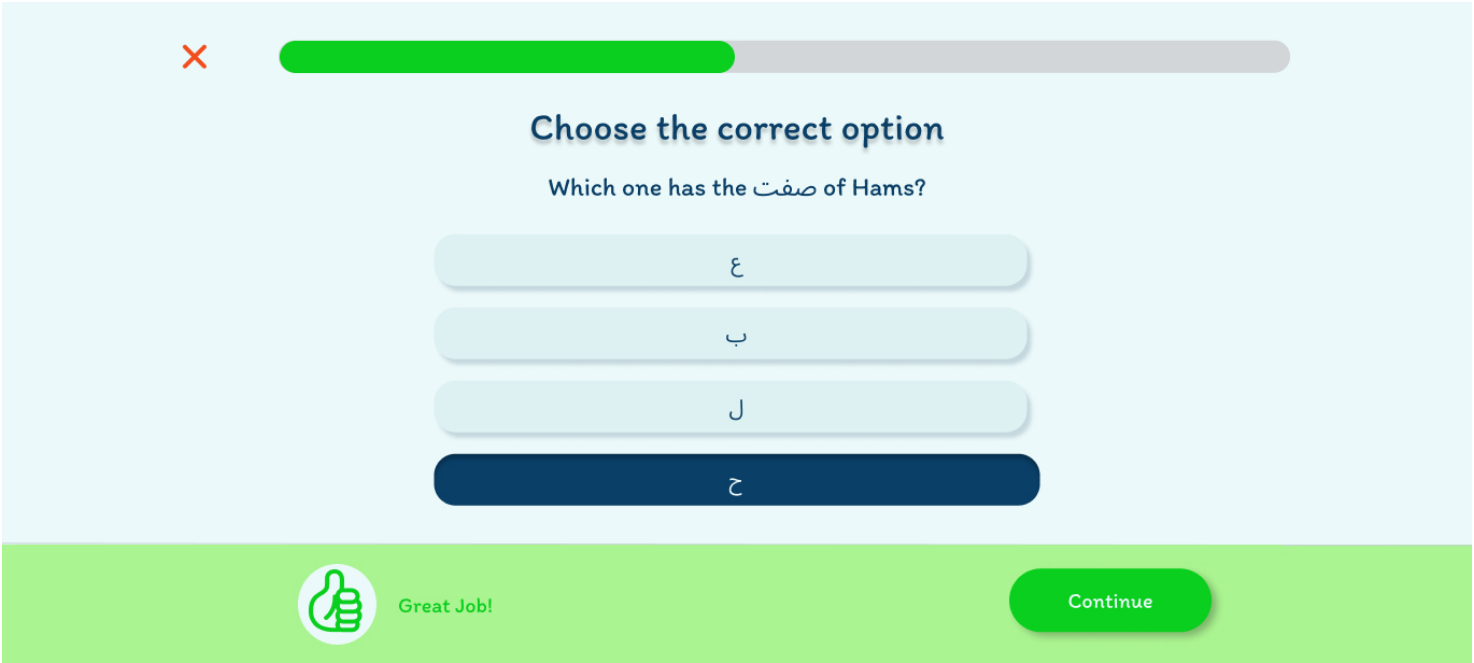


Figure 47. Visualization of Practice Slide MCQ Correct User Interface

5.5.2.21. PRACTICE SLIDE MCQ WRONG

The design shows the appearance of the webpage after the user has answered the question wrong. The user would be shown a correct answer and would be given an option to continue to the next slide.

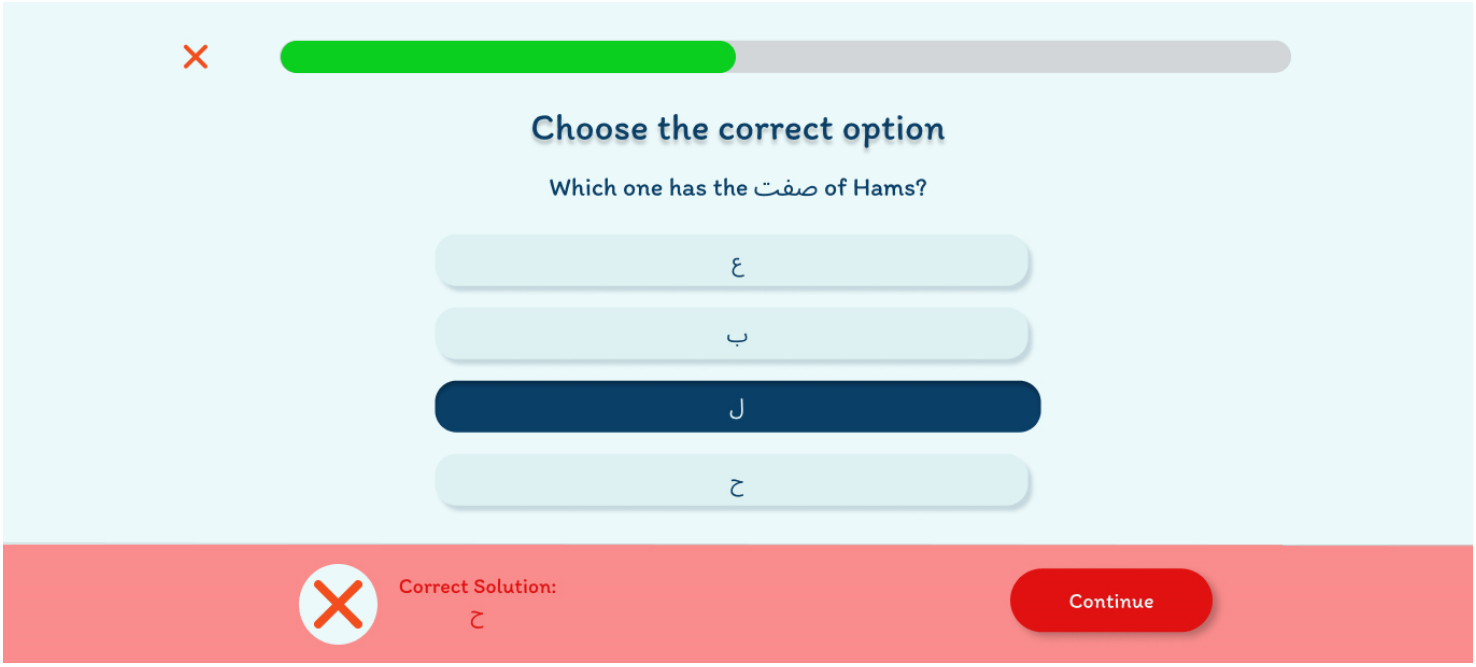


Figure 48. Visualization of Practice Slide MCQ Wrong User Interface

5.5.2.22. MATCHING SLIDE

This Practice slide will provide two columns and the user has to match the options from the left column to any of the options in the right column. The user can first switch between the options in one column but when they choose an option on a column and right afterwards click on an option on the different column, it would be considered as a match to the last selected option of the other column.

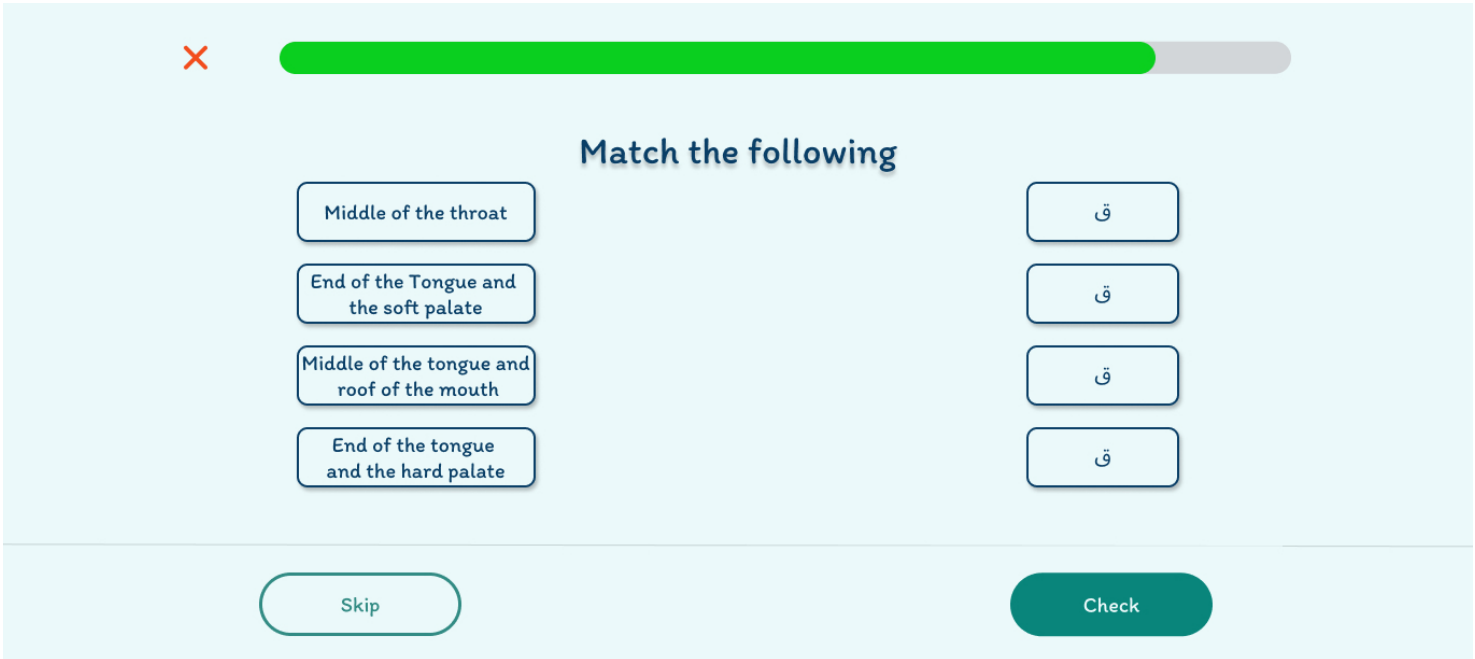


Figure 49. Visualization of Matching Slide User Interface

5.5.2.23. MATCHING SLIDE SELECTED

The selected options will be highlighted in this manner as shown in the design and right after that the system will check the match made.

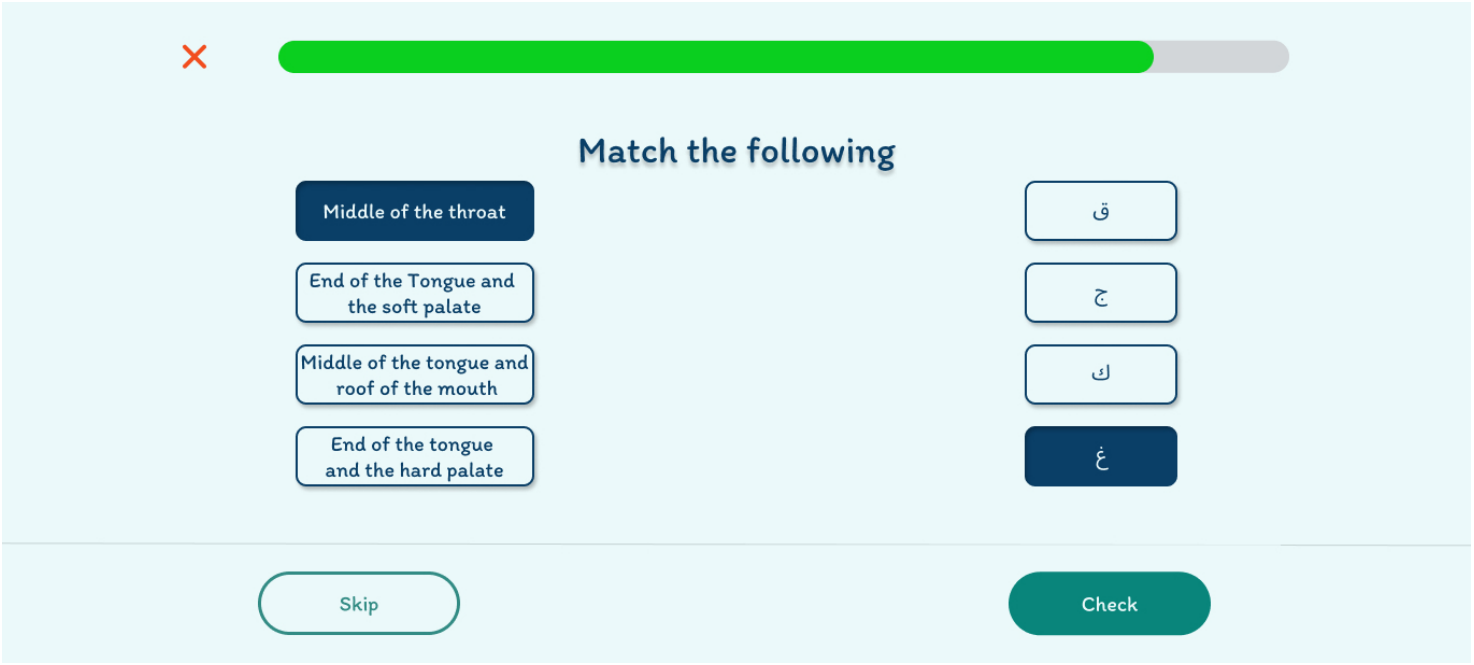


Figure 50. Visualization of Matching Slide Selected User Interface

5.5.2.24. MATCHING SLIDE DISABLED

The previously made matches, whether they were correct or wrong, would be disabled so that the user can not make any further changes to those.

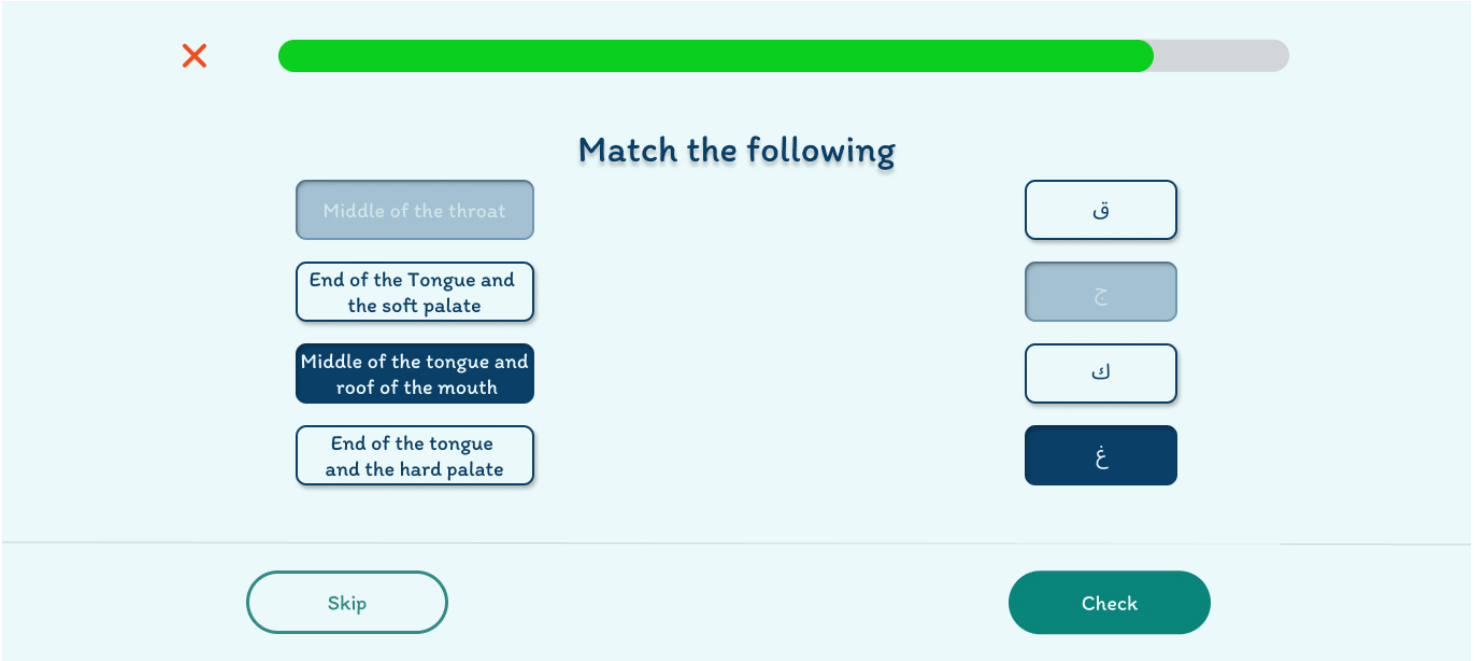


Figure 51. Visualization of Matching Slide Disabled User Interface

5.5.2.25. MATCHING SLIDE CHECK

Once the user has given the request to check the answers, the system will highlight the correct and wrong matches accordingly. Red color signifies incorrect matches whereas green color signifies correct matches.

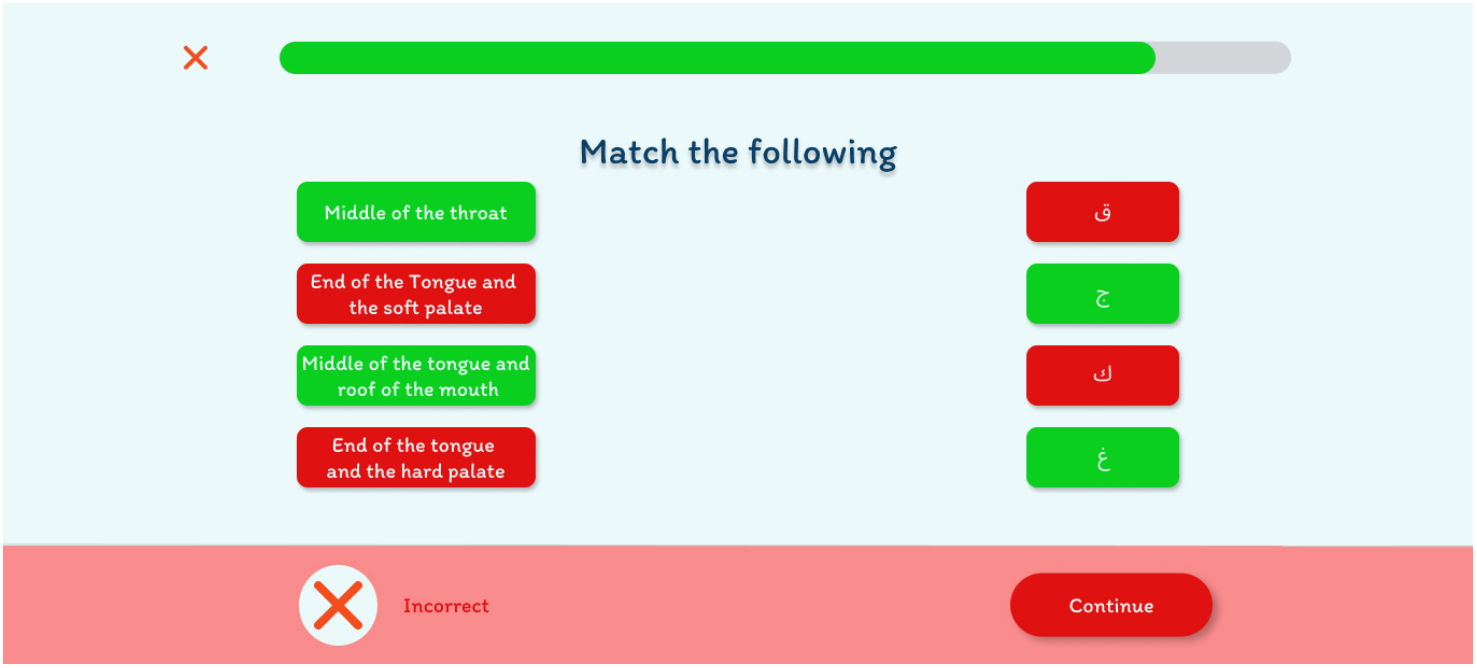


Figure 52. Visualization of Matching Slide Check User Interface

5.5.2.26. AUDIO SLIDE

The slide allow users to play the audio of the question and answer accordingly. When the user clicks on any option, audio of that option will also play which is highlighted by the change of stroke color on the interface.

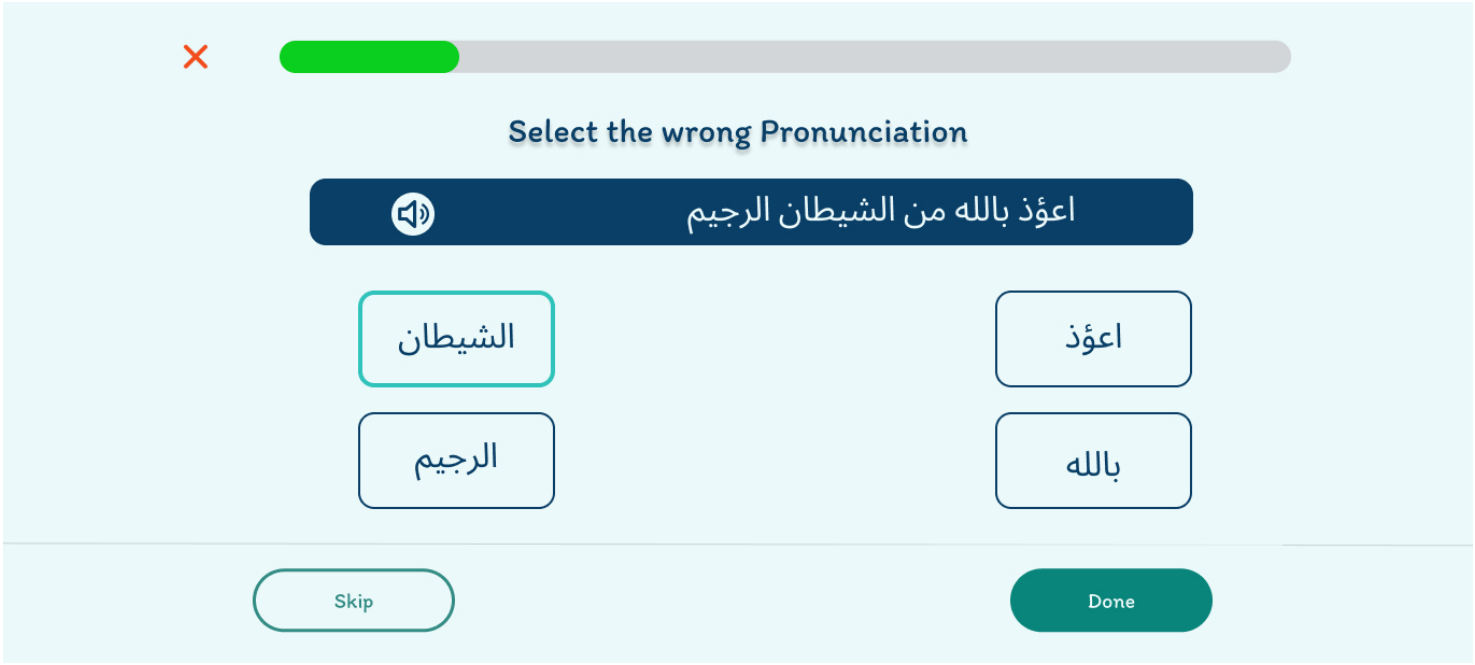


Figure 53. Visualization of Audio Slide User Interface

5.5.2.27. MOTIVATION SLIDE

This slide will appear to motivate the users to keep learning more and to keep them engaged with the learning journey.

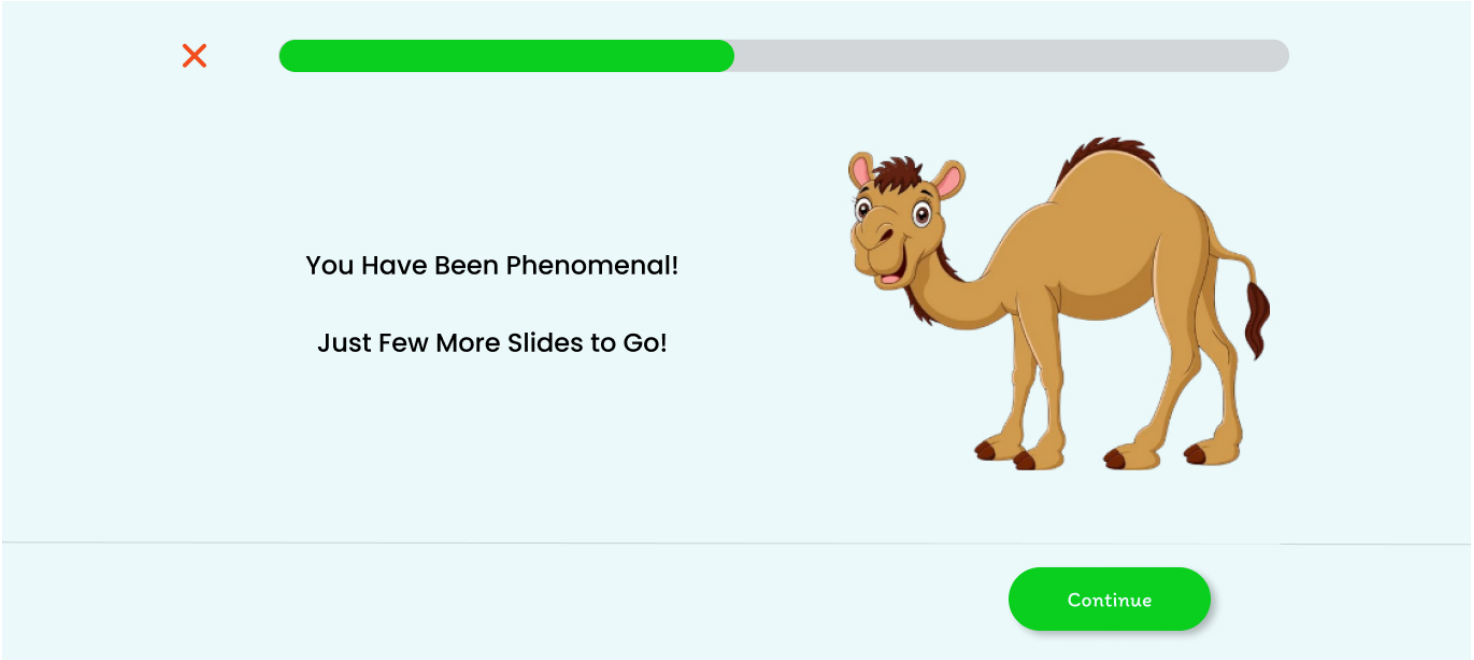


Figure 54. Visualization of Motivation Slide User Interface

5.5.2.28. CANCEL

The red button on the top left corner allows the user to exit the lesson unit. When it is clicked, an alert appears telling the user that the progress will be lost and to reaffirm the decision.

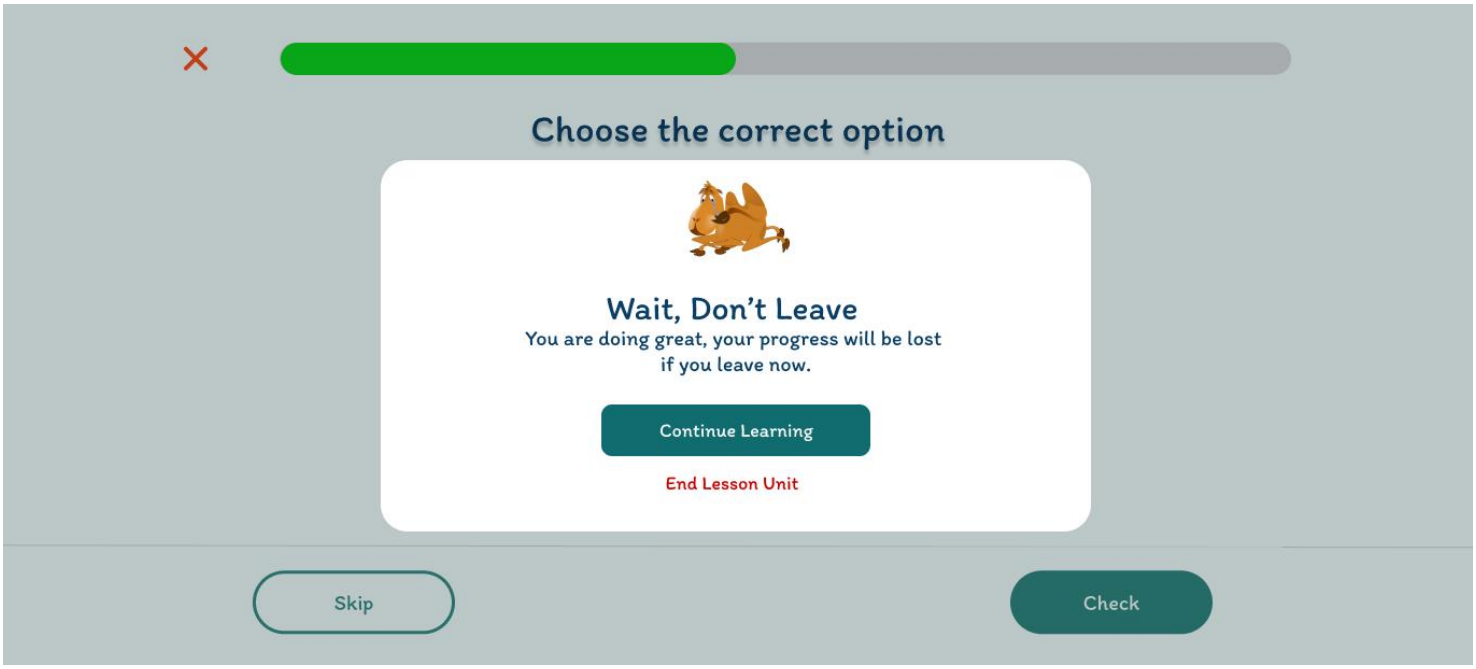


Figure 55. Visualization of Cancel User Interface

5.5.2.29. CHATSPACE INITIAL STATE

The design demonstrates the user interface of ChatSpace feature of the system. When the user open the ChatSpace, all the previous chats will appear along with the last messages of those chats and the profile picture of those users. There are two options Group chat and private chat while the design here shows the UI of the private chat. It also shows relatively blank space on the right, waiting for the user to select a chat, to show its whole chat.

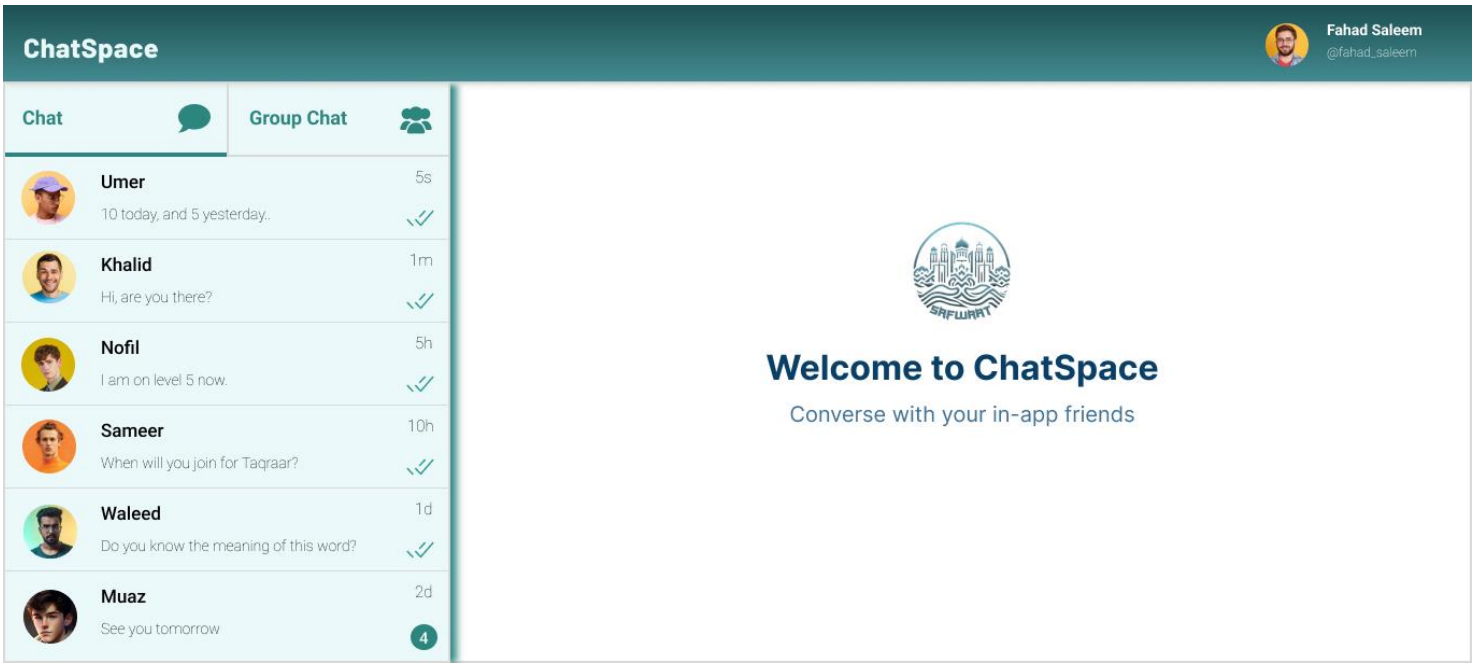


Figure 56. Visualization of ChatSpace Initial State User Interface

5.5.2.30. CHATSPACE CHATS

It shows the whole chat with the particular user and when those messages were received or when the messages were sent to the that user. It also shows their online status while a place is provided for the user to type in new messages and send to the their in-app friend.

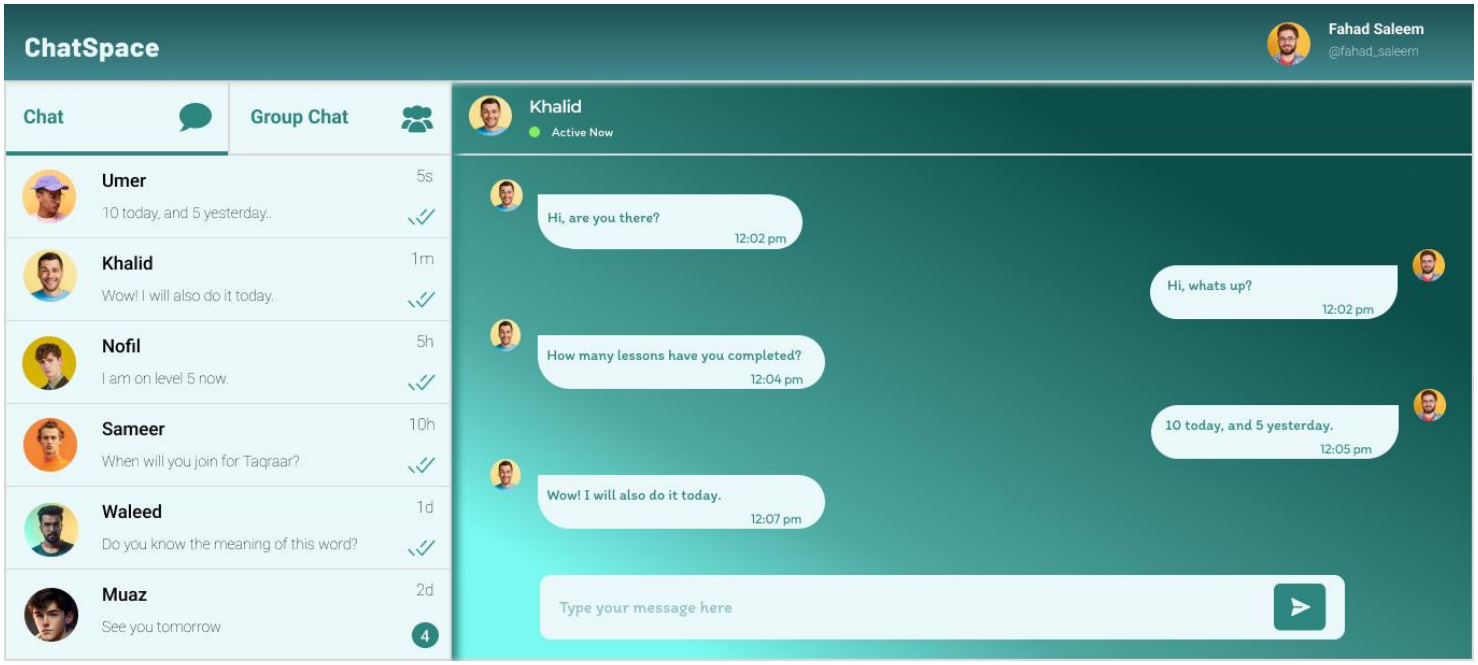


Figure 57. Visualization of ChatSpace Chats User Interface

5.5.2.31. FLASHCARD SET CREATION

The design shows the blanks that the user has to fill such as title, category, and optional description to create a flashcard set. The user has to fill the information to be displayed on the flashcard: the term which will be displayed on the front side of the flashcard and the Definition which will be displayed on the back side of the flashcard.

The UI also provides option to add more flashcards and an option to create the flashcard set when the user has entered enough flashcards to be added in the set.

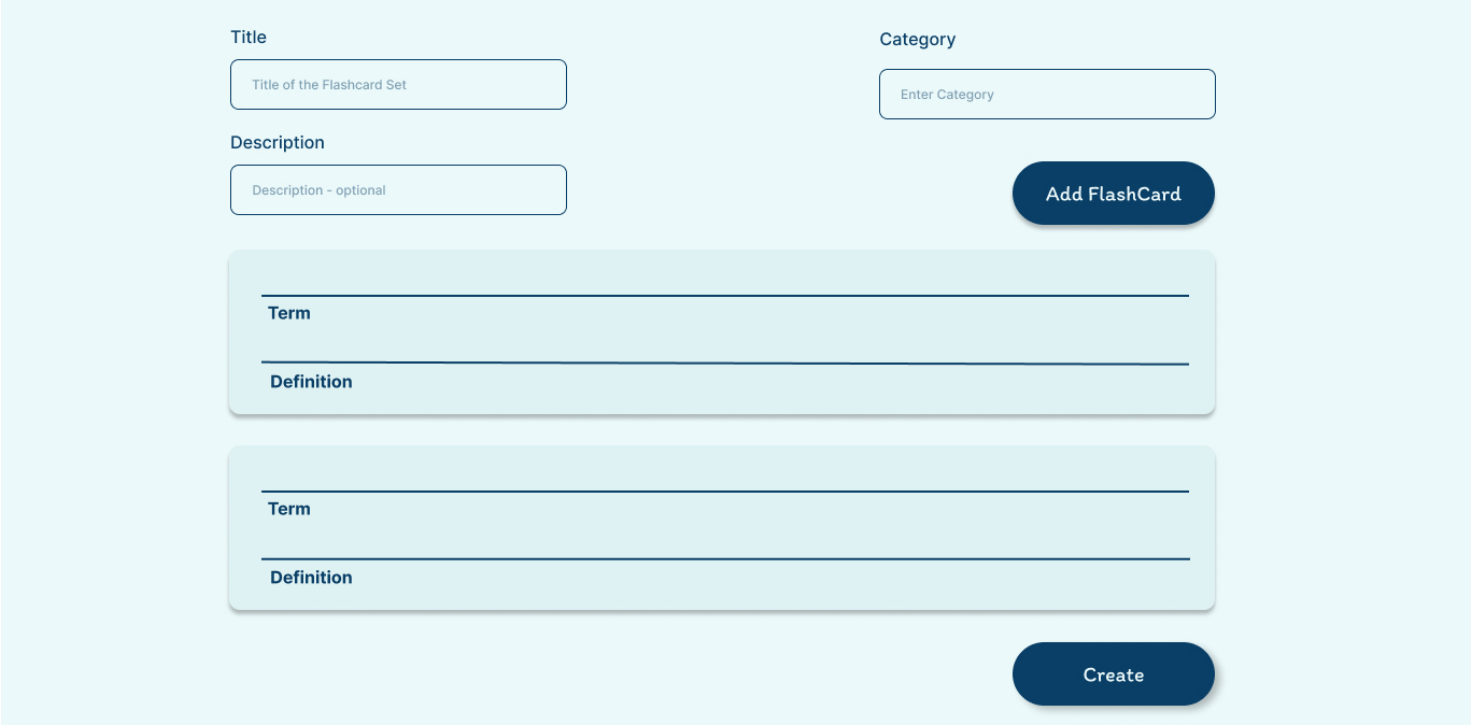


Figure 58. Visualization of FlashCard Set Creation User Interface

5.5.2.32. FLASHCARD SET COMPLETION

The UI appearance when the user has filled in all the necessary data is shown and how the option will change its appearance on-click.

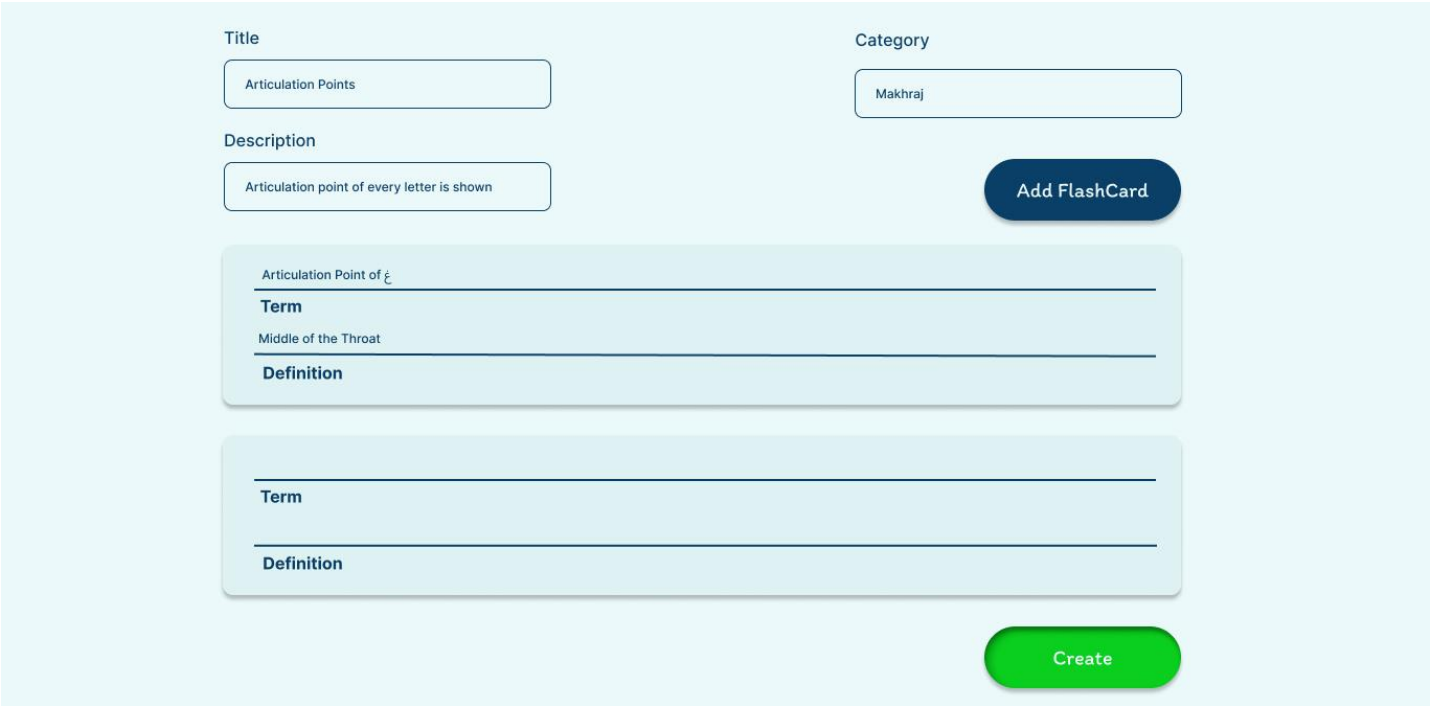


Figure 59. Visualization of FlashCard Set Completion User Interface

5.5.2.33. FLASHCARD SET ERROR

It shows the error that will be generated when the user tries to create a flashcard set with less than 2 cards. Clicking on the backdrop will take the user to the same interface as before.

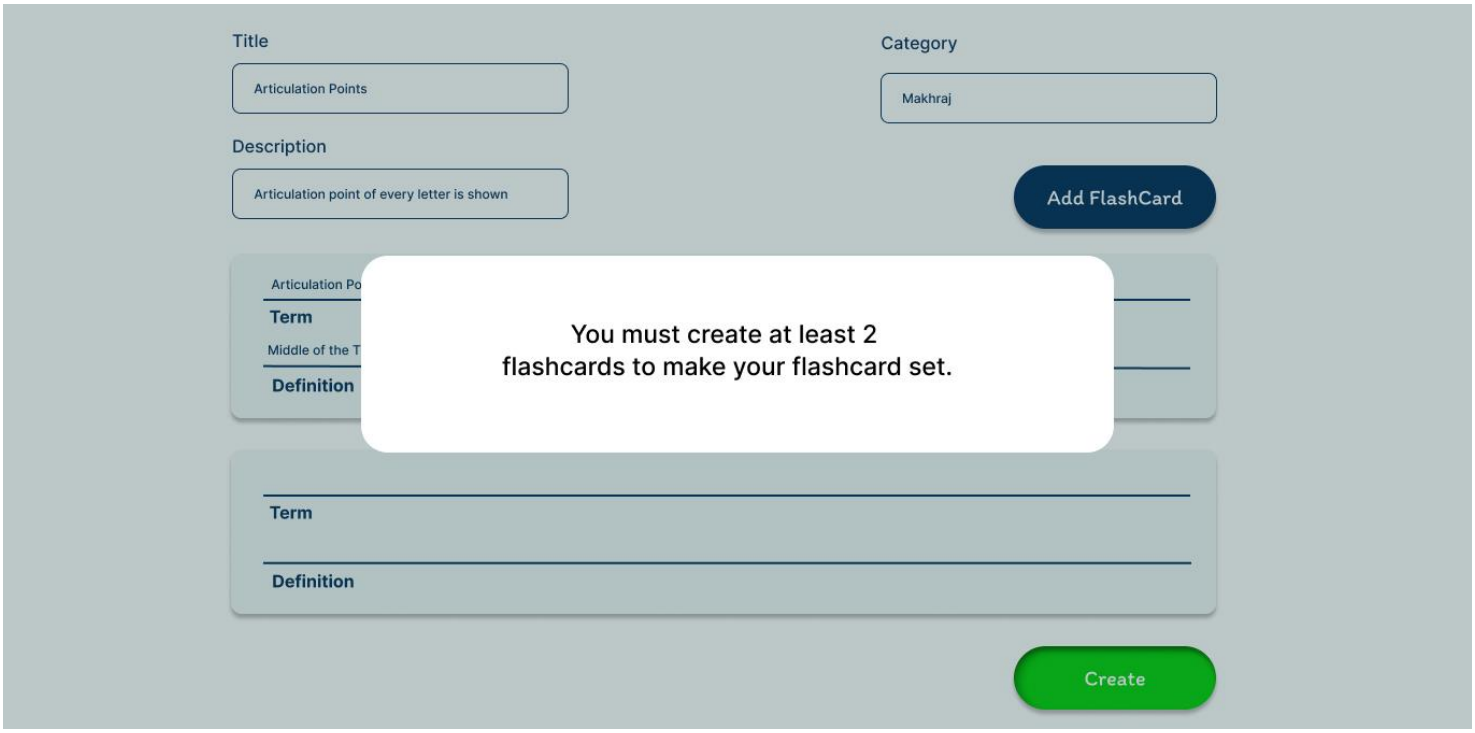


Figure 60. Visualization of FlashCard Set Error User Interface

5.5.2.34. FLASHCARD FRONT

The appearance of the flashcard on the front side and the instructions provided to the user are shown.

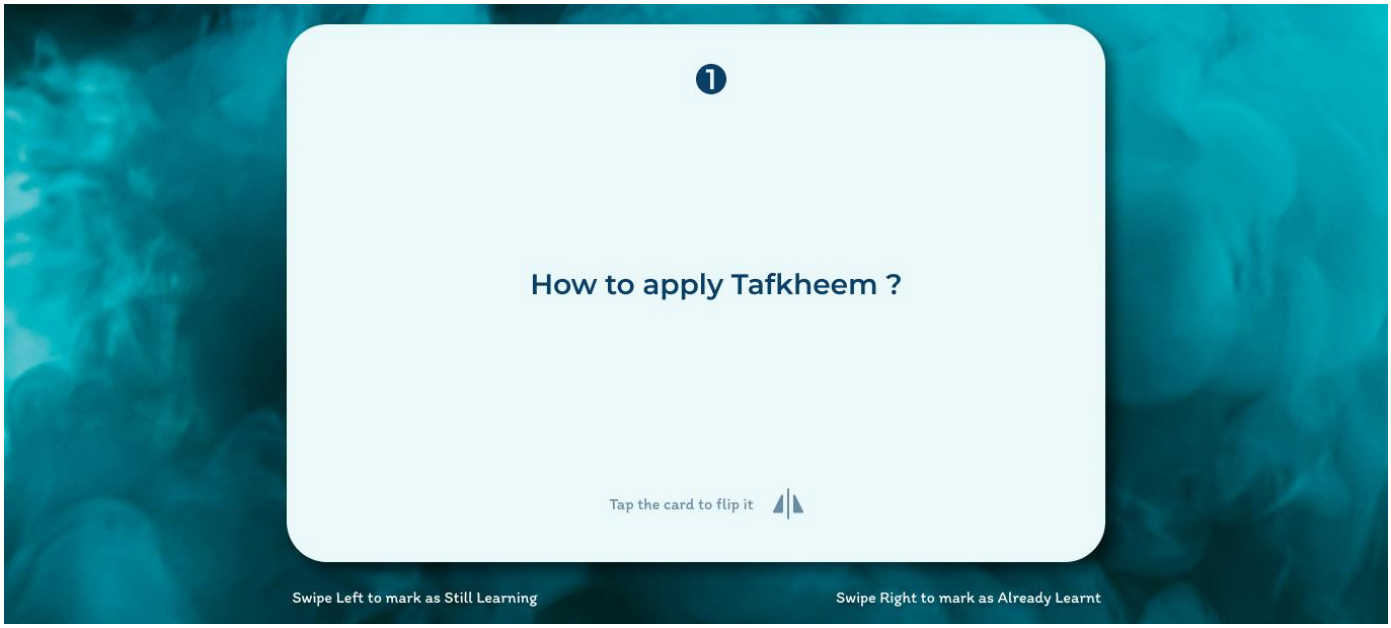


Figure 61. Visualization of FlashCard FrontUser Interface

5.5.2.35. FLASHCARD BACK

The flip side view of the flashcard is shown when the user taps on the card as instructed. If the user taps now, they will again be directed to the frontside seamlessly. The user can swipe the card to the right if he knows the information on the flashcard otherwise swipe it to the left if still learning.

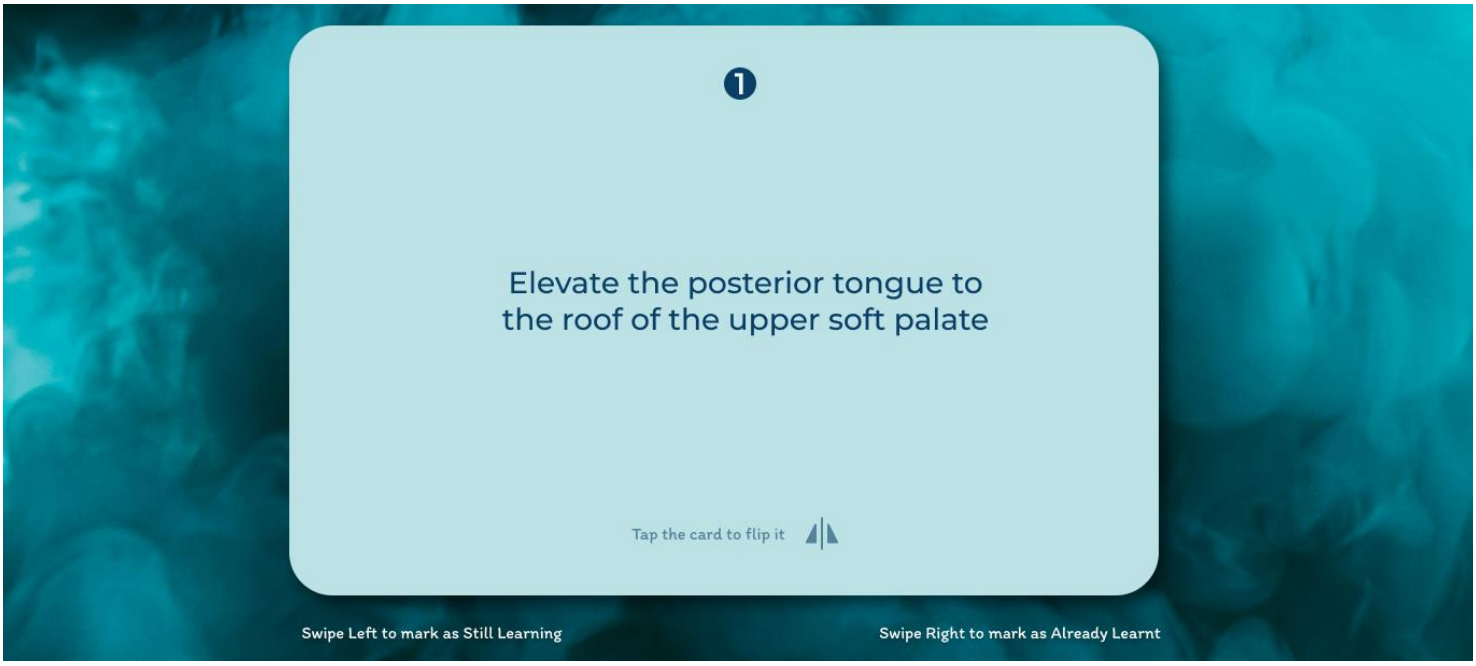


Figure 62. Visualization of FlashCard Back User Interface

5.5.2.36. FLASHCARD FRONT OF IMAGE TYPE

The UI provides an option to the user to add pictures as well. The design shows the flashcard front view of that other type of flashcard.

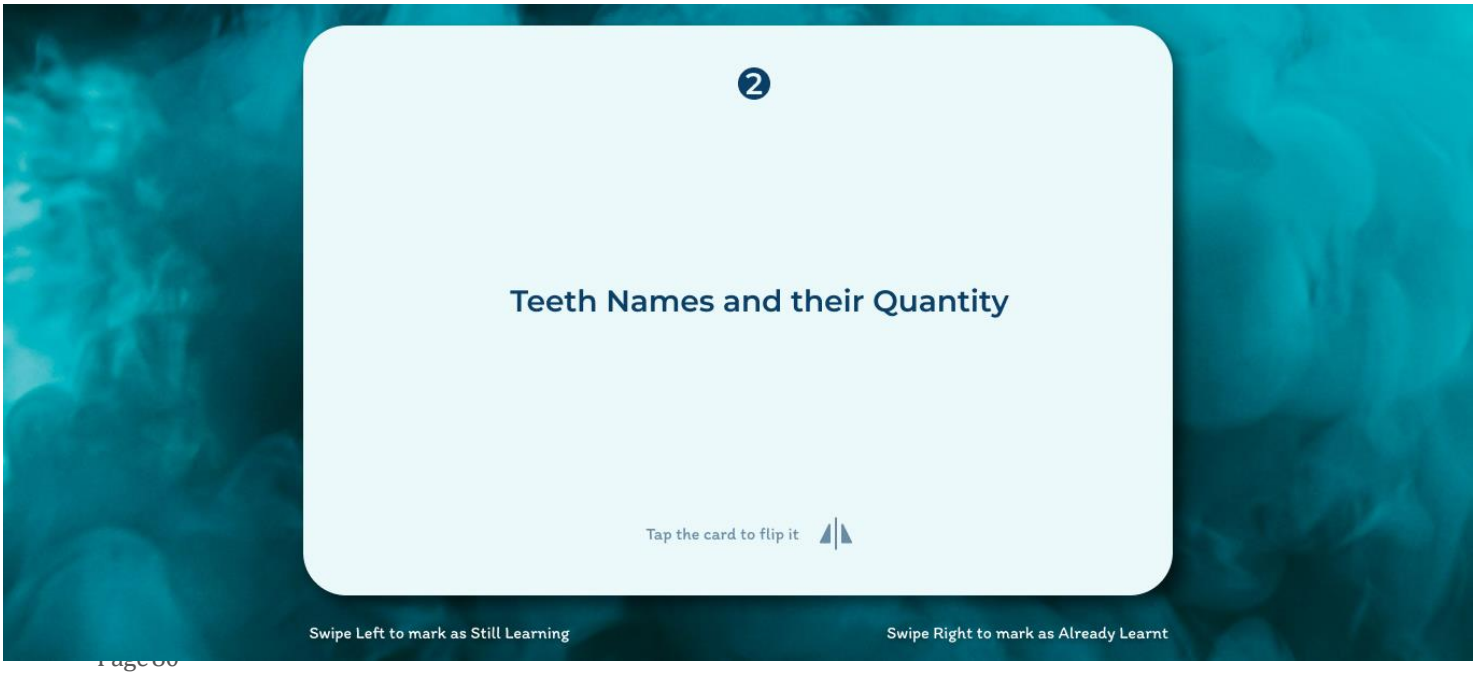


Figure 63. Visualization of FlashCard Front Of Image Type User Interface

5.5.2.37. FLASHCARD BACK OF IMAGE TYPE

The image that the user has added as the description of the term when creating flashcard will shown to the user on the backside of flashcard.

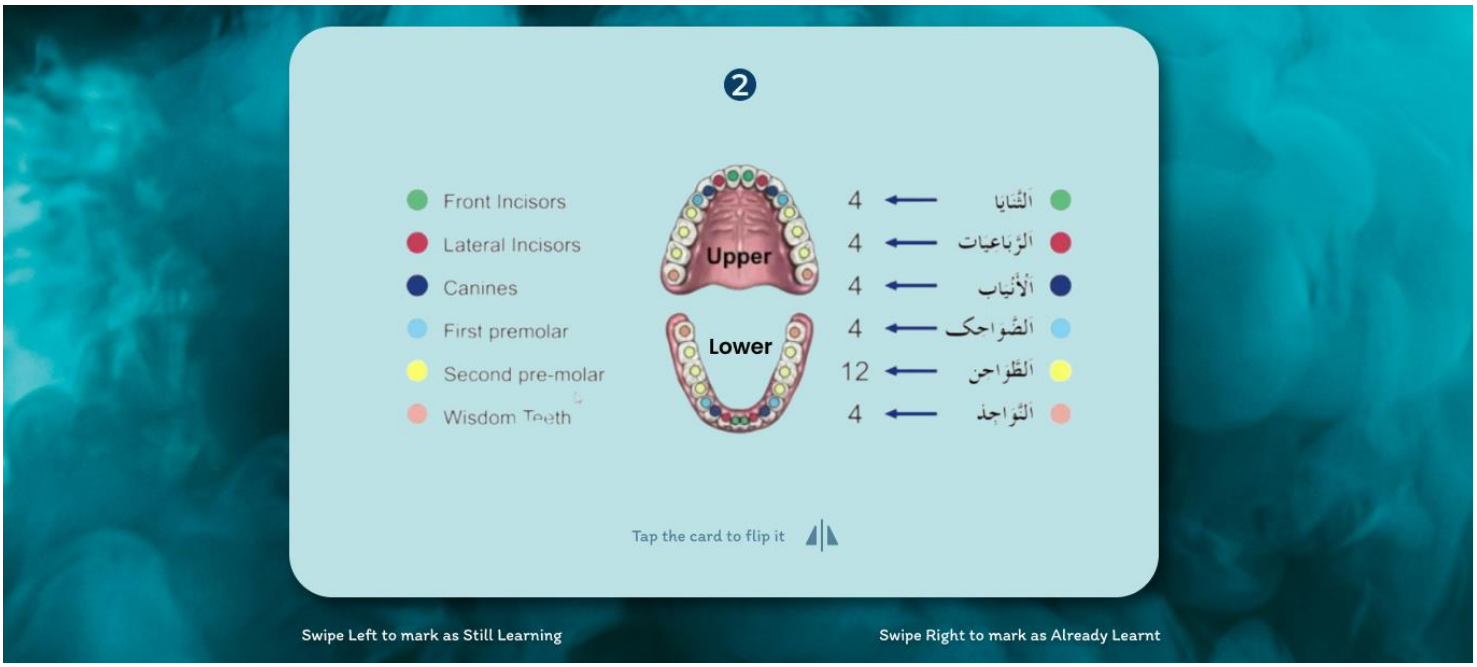


Figure 64. Visualization of FlashCard Back Of Image type User Interface

5.5.2.38. FLASHCARD FINAL VIEW STILL LEARNING

When the user has reviewed all the flashcards, they will be shown their results and the count of how many they know and how many are in 'still learning' phase. An option to 'Keep Reviewing' is provided so that the user can learn or revise the 'Still Learning' cards.

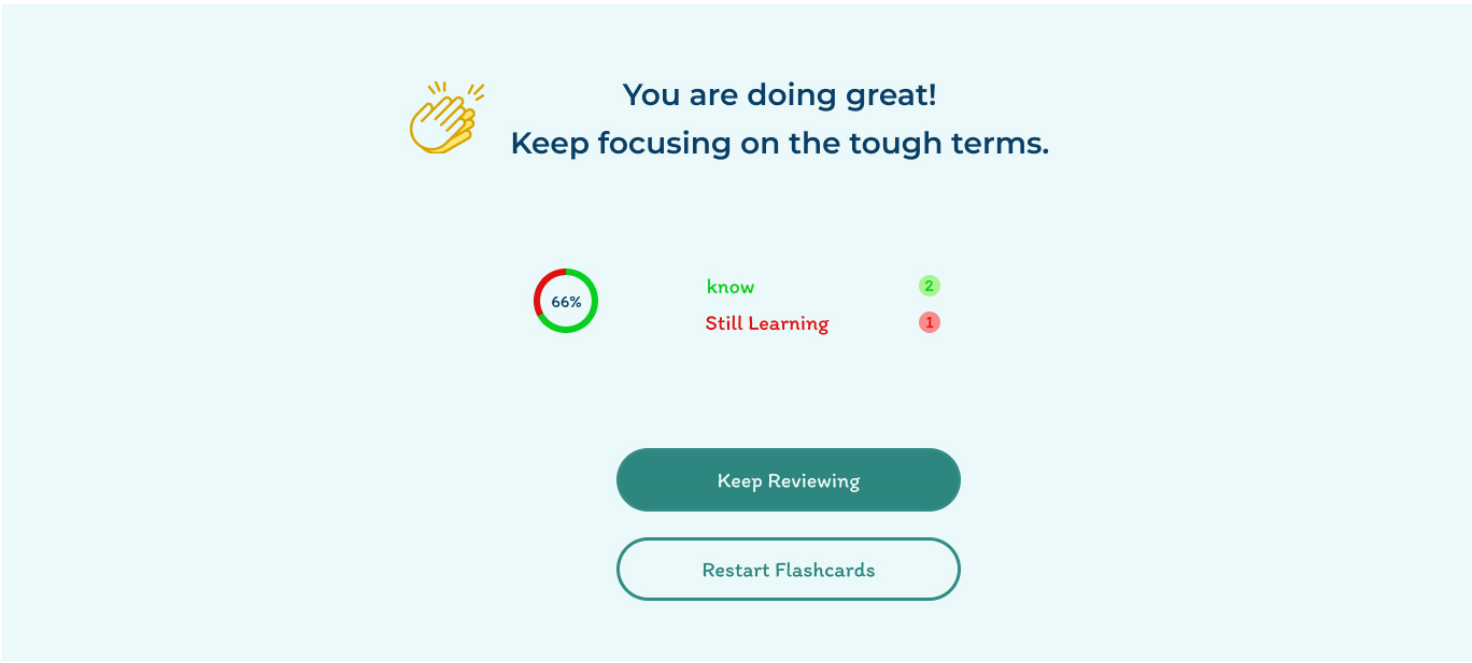


Figure 65. Visualization of FlashCard Final View Still LEarning User Interface

5.5.2.39. FLASHCRAD FINAL VIEW ALL KNOWN

This design illustrates the UI when all the flashcards turned out be already learnt by the user.

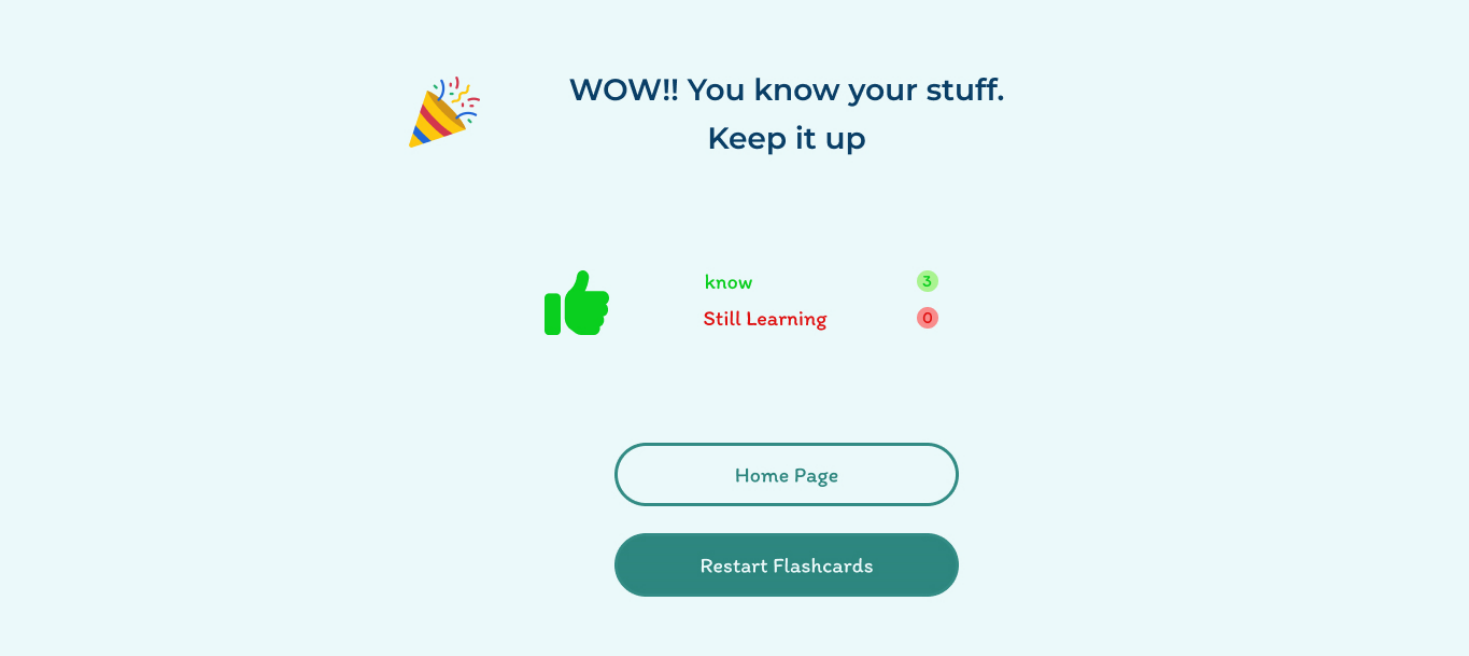


Figure 66. Visualization of FlashCard Final View All Known User Interface

5.5.2.40. STREAK BOARD

The Steak Board displays a calender highlighting the range of streak that the user has along with the streak count. It also allows the user to switch to previous or next month.

Figure 67. Visualization of Streak Board User Interface

5.5.2.41. XP LEADERBOARD BRONZE LEAGUE

This UI illustrates the ranking of user on basis of Xp points in the assigned league which is currently bronze league. It also provides information about how many members are to be promoted to next league.

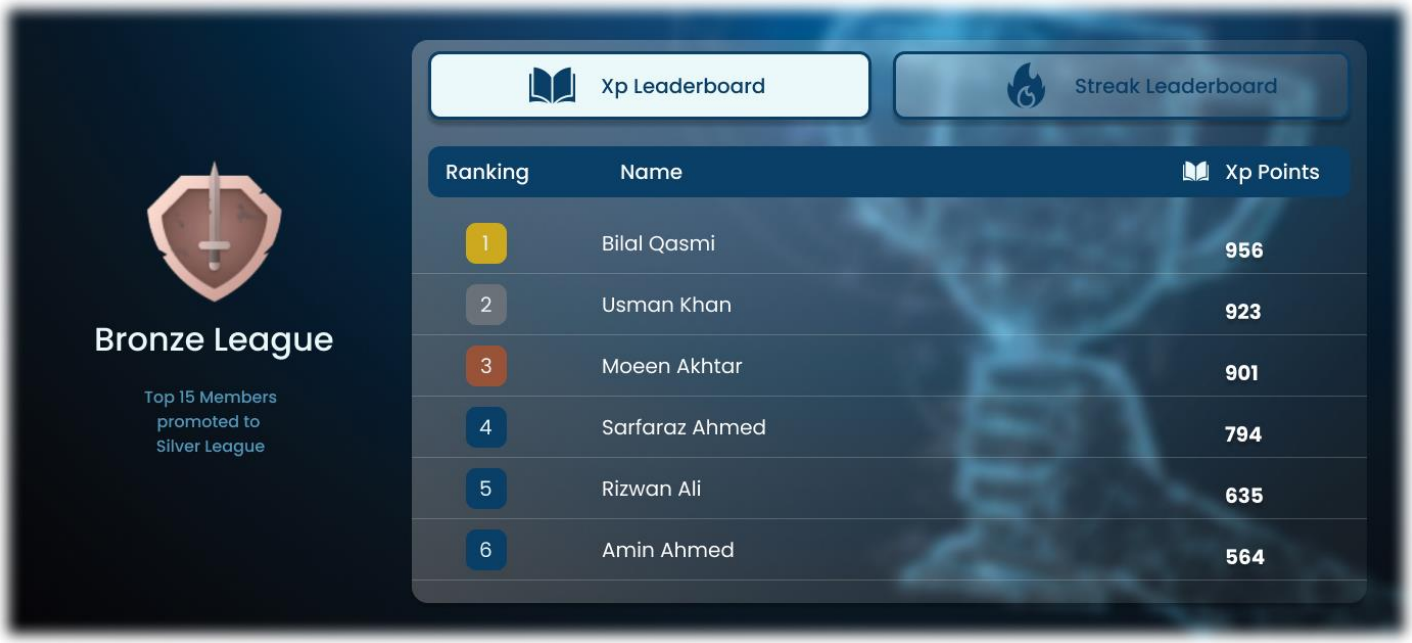


Figure 68. Visualization of Xp Leaderboard Bronze League User Interface

5.5.2.42. XP LEADERBOARD GOLD LEAGUE

This UI is another variant of above UI with different league badge and promotion criteria as well as higher average Xp point.

Figure 69. Visualization of Xp Leaderboard Gold League User Interface

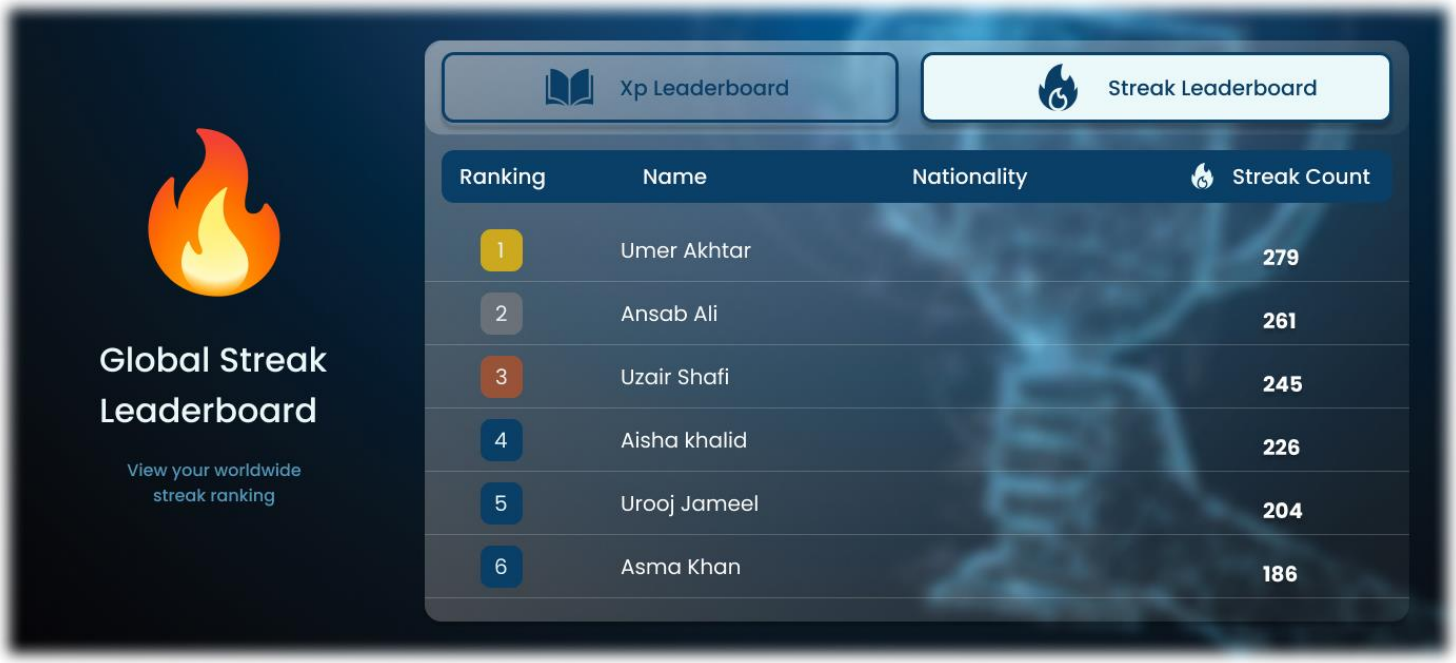
5.5.2.43. XP LEADERBOARD DIAMOND LEAGUE

This UI depicts final league in the system. It doesn't provide promotion criteria as it is final league instead it tells that top three ranks will be rewarded heavily as compared to other competitors.

Figure 70. Visualization of Xp Leaderboard Diamond League User Interface

5.5.2.44. STREAK LEADERBOARD

The Streak leaderboard UI depicts the global ranking of user on basis of streak count. Additionally, since streak leaderboard is global this UI also shows the national flag of the country



the user belongs to.

Figure 71. Visualization of Streak Leaderboard User Interface

5.6. INTERACTION VIEWPOINTS

5.6.1. Login and Registration

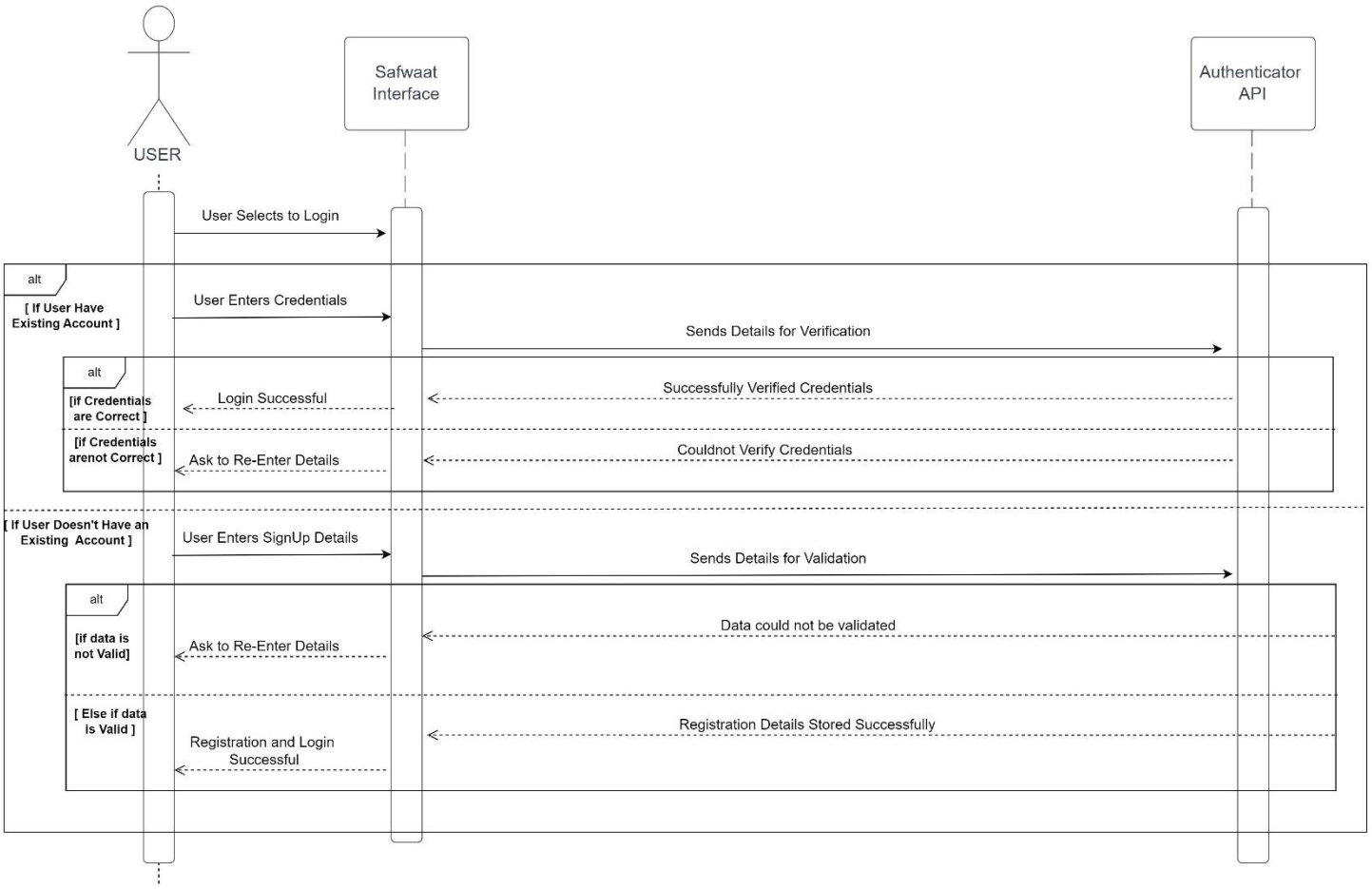


Figure 72. Visualization of Login and Registration Sequence Diagram

In the sequence diagram encompassing the "Registration" and "Login" features, the User interacts with the UI Interface to initiate either the registration or login process.

In the login scenario, the User enters their username and password through the UI Interface, with the system permitting access if the credentials match existing records. Additionally, the system allows login through a Google Account using the Authenticator API. The sequence guarantees a secure and user-friendly experience, with seamless transitions between registration and login functionalities. For registration, the system validates user inputs, ensuring correctness and adherence to specified criteria. In case of invalid input, appropriate error messages are displayed, preventing registration until valid details are provided. Upon successful validation, the system creates a new user account, stores credentials in the database, and ensures the uniqueness of usernames. Feedback is then provided to the User.

5.6.2. TAQRAAR

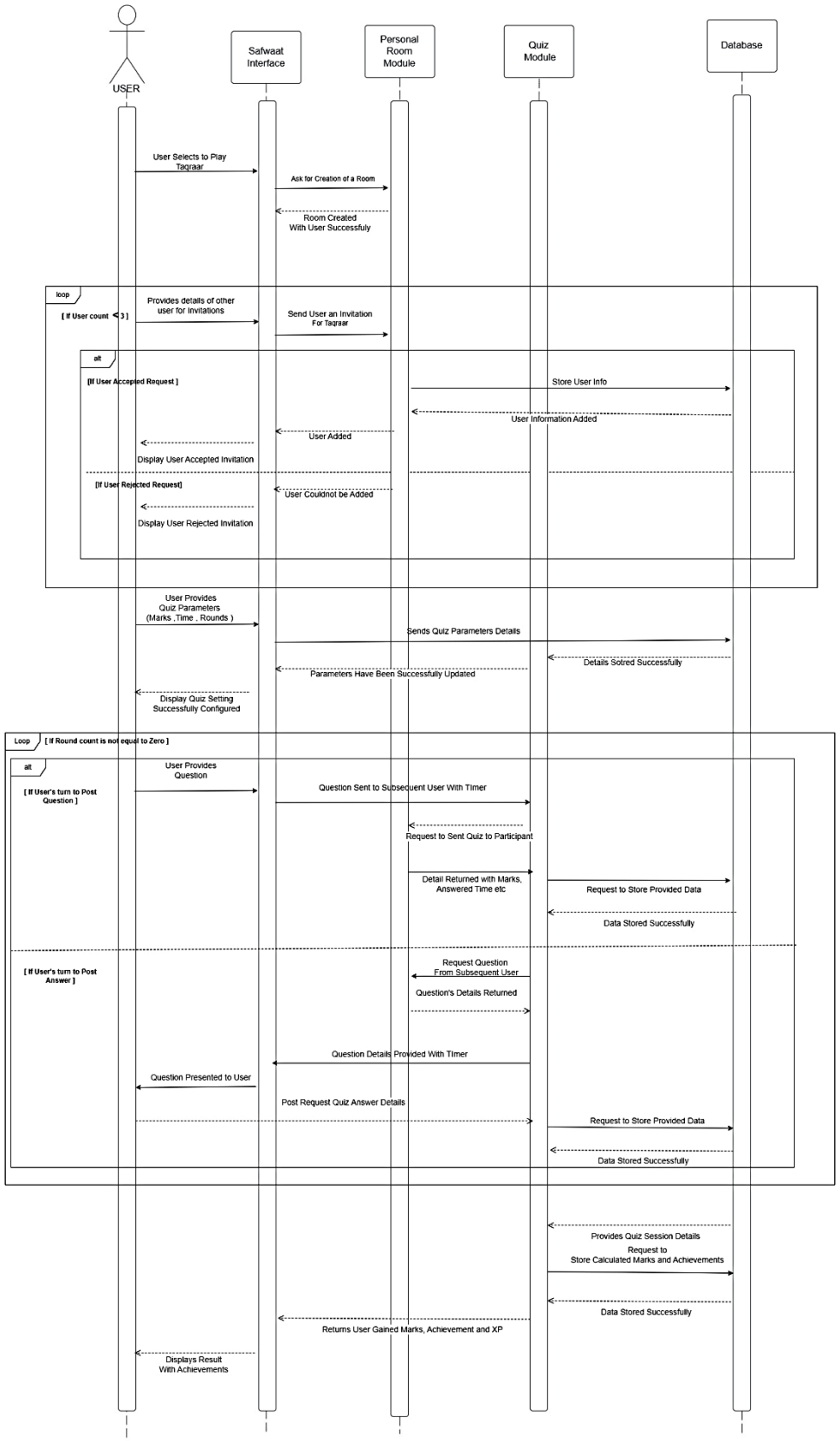


Figure 73. Visualization of Taqraar Sequence Diagram

DESIGN VIEWPOINTS

In Taqraar feature, a User, acting as a host in a Personal Room, initiates a quiz session. The User interacts with the UI Interface to set parameters for the session, including selecting participants from the room, specifying the number of rounds, defining time intervals for answering questions, and assigning uniform mark values. The system ensures a round-robin fashion for participants to post questions and provides a timer for answering. After each round, participants receive feedback on their answers and scores. The system keeps track of the overall progress, calculates total marks, and displays user rankings. The session concludes as per the host's specifications, showcasing earned achievements in the user profile. The Quiz Module becomes usable when the user count reaches three, emphasizing a collaborative and engaging quiz experience within the defined Personal Room.

5.6.3. Level Map, Learning Unit, 3D Model and Social Media Sharing

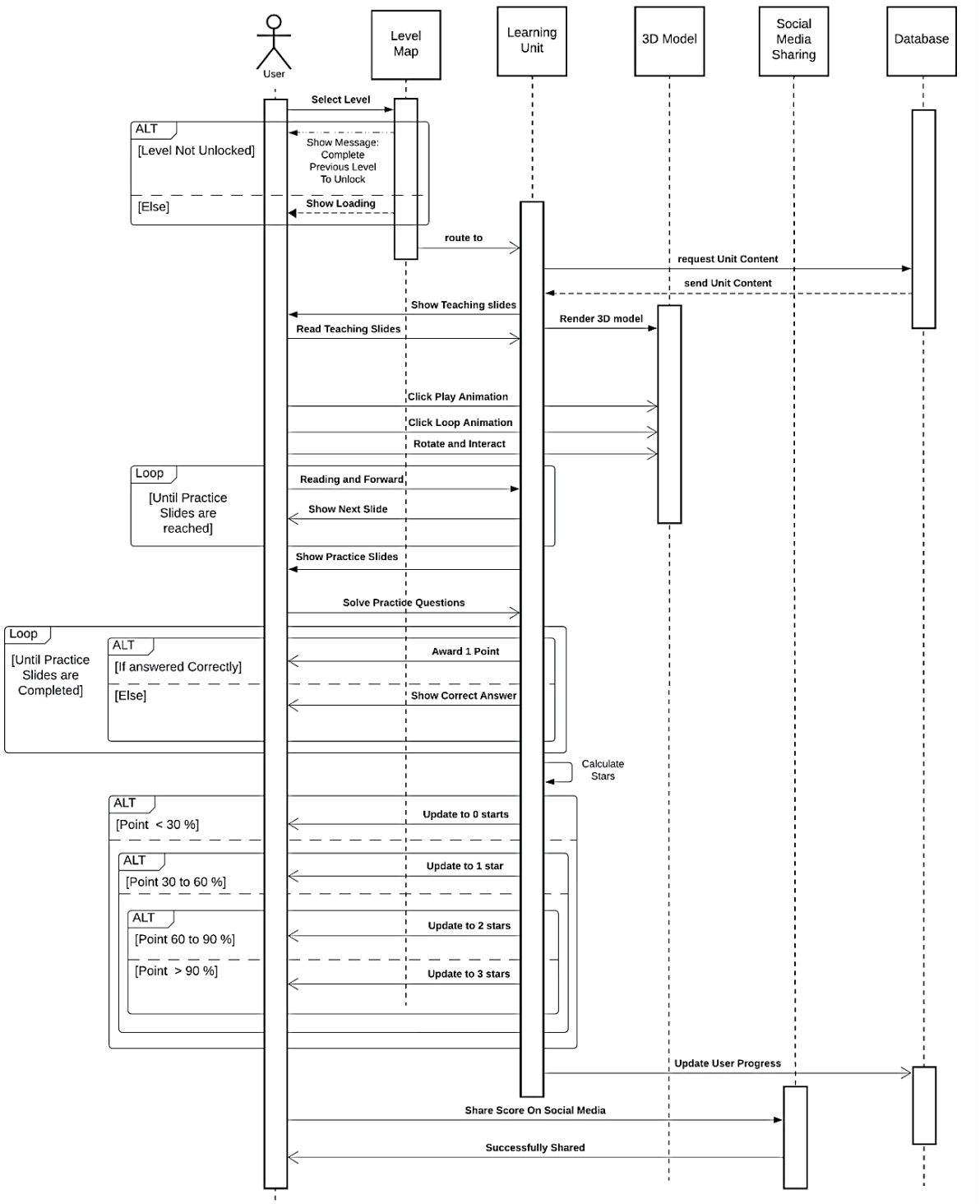
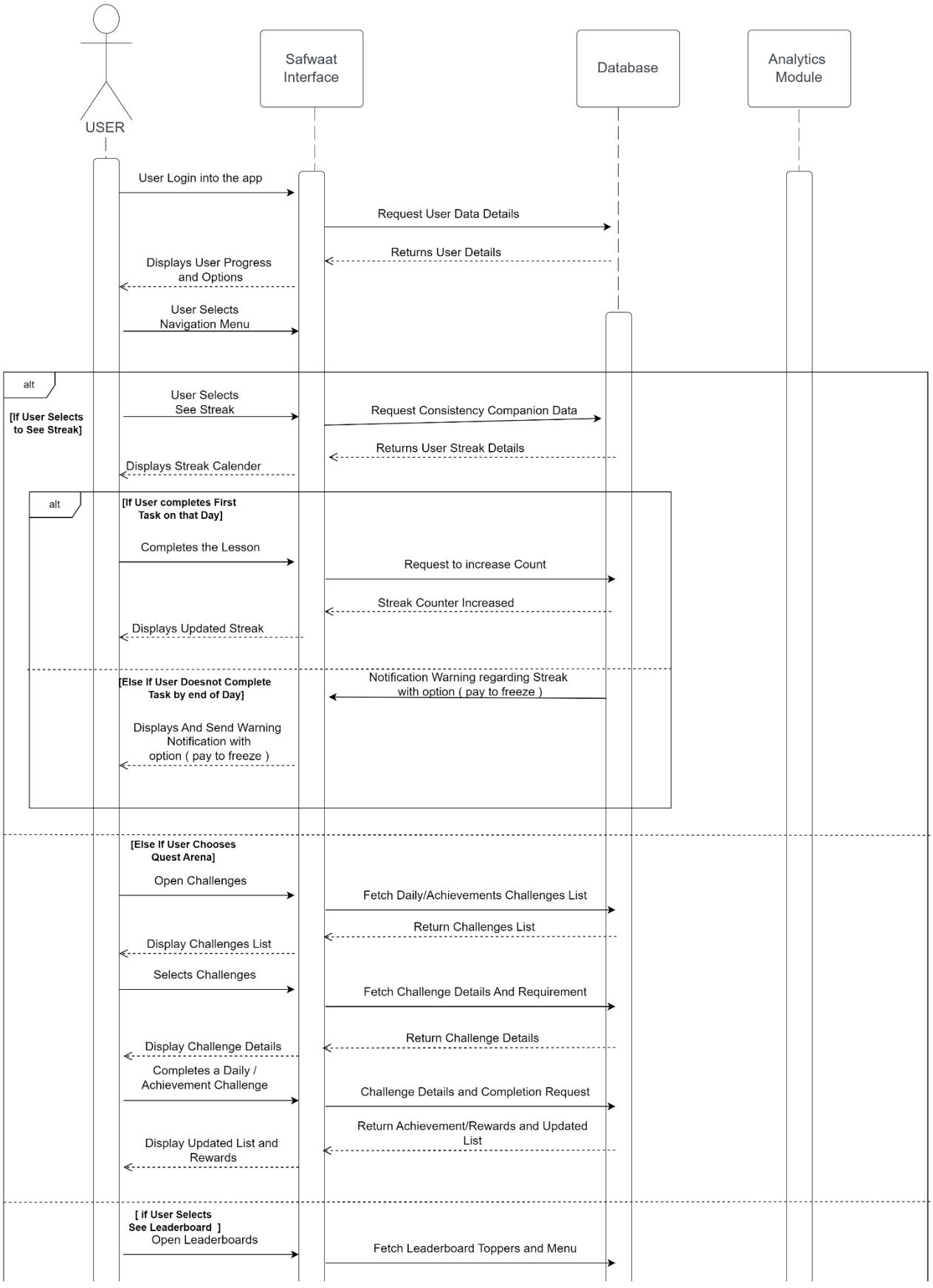


Figure 74. Visualization of Level Map, Learning Unit, 3D Model, Social Media Sharing Sequence Diagram

DESIGN VIEWPOINTS

The system guides users through a sequence of learning steps, from selecting a level and accessing content to practising with interactive slides and animations. Users can earn points and stars, track their progress, and even share their achievements on social media, making the app an engaging and effective way to learn about Tajweed.

5.6.4. Consistency Companion, Leaderboard, Quest Arena and Analytics



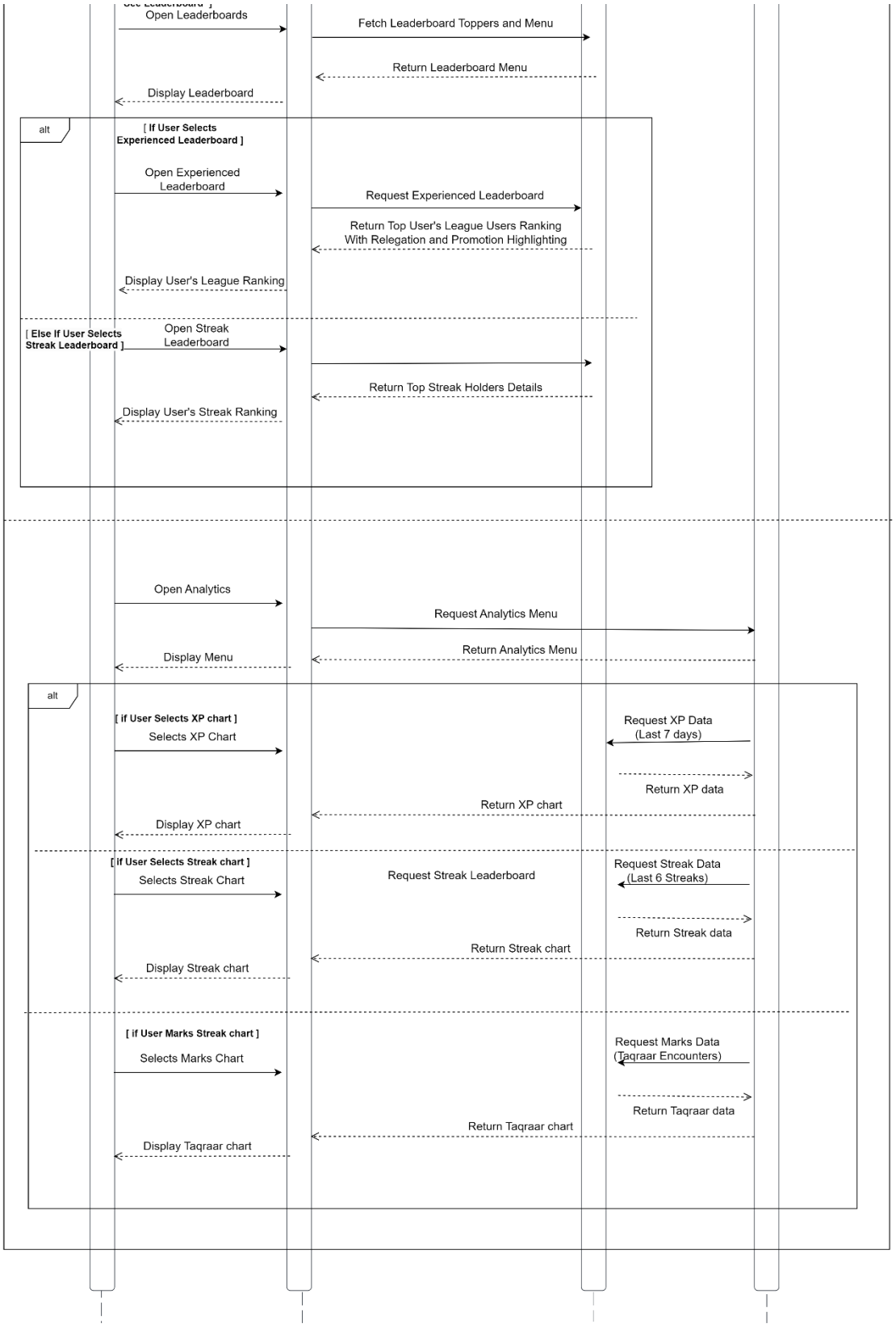
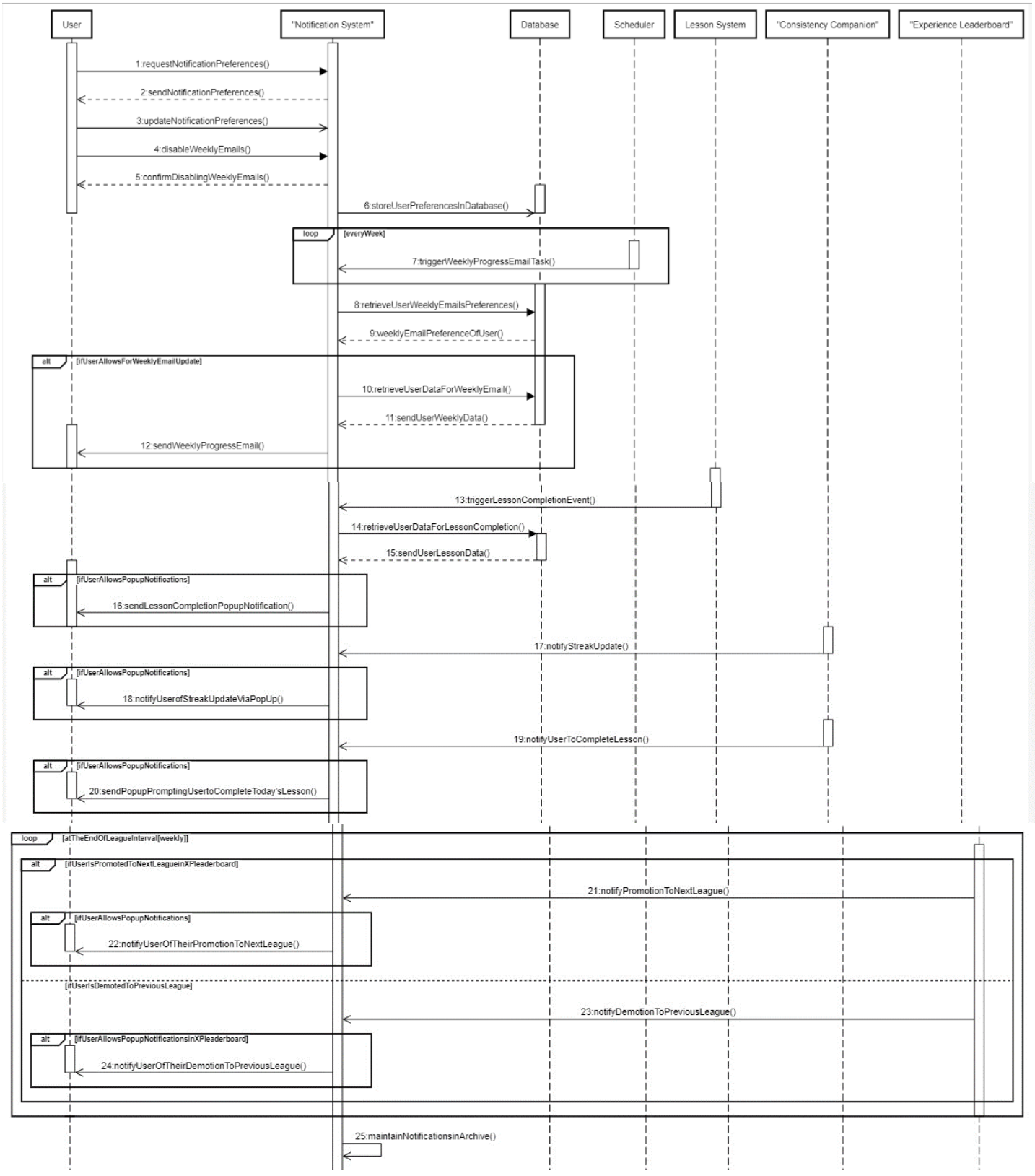


Figure 75. Visualization of Consistency Companion, Leaderboard, Quest Arena and Analytics Sequence Diagram

DESIGN VIEWPOINTS

In the integrated sequence diagram featuring the "Consistency Companion," "Quest Arena," "Leaderboard," and "Analytics," the User navigates through the website, interacting with the UI Interface to access various features seamlessly connected by the Website Navigation System. The sequence begins with the Consistency Companion, where the User engages in daily learning activities, and the system updates streaks and progress in the Database. Subsequently, the User navigates to the Quest Arena, participating in challenges and achievements, with the system dynamically assigning and updating challenges based on the User's experience level. Progress and rewards are tracked in the Database. The User then explores the Leaderboard, viewing rankings and promotions, and the system updates this information in real-time. Finally, the User accesses Analytics, generating insights through charts and graphs, with data retrieved from the Database. Throughout this integrated sequence, the Website Navigation System ensures a cohesive user experience, allowing seamless transitions between features while maintaining accurate and up-to-date information in the Database.

5.6.5. NOTIFICATION



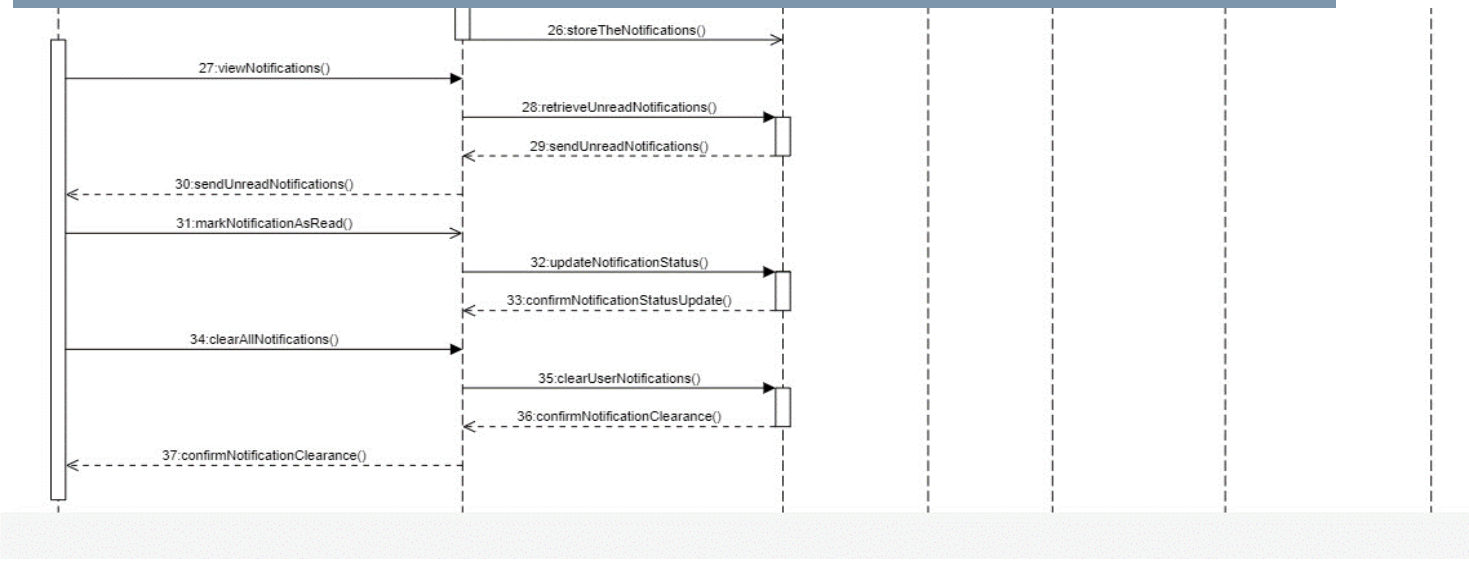


Figure 76. Visualization of Notifications Sequence Diagram

This sequence diagram outlines the flow of interactions within a Notification System, illustrating the communication between the user, the Notification System, associated subsystems, and the underlying database. The user initiates actions such as requesting and updating notification preferences, disabling weekly emails, and managing notifications. The Notification System handles these requests, storing and confirming preferences in the database. Weekly tasks, lesson completion events, and interactions with companion systems like the Consistency Companion and Experience Leaderboard are depicted, with optional popup notifications based on user preferences. The system maintains a notification archive, stores notifications in the database, and supports user actions such as viewing, marking as read, and clearing notifications. The sequence also includes confirmations for database updates and clearance. Overall, this sequence diagram provides a comprehensive visualization of the Notification System's functionality, encompassing user interactions, system triggers, and database interactions.

5.6.6. FLASHCRAFT

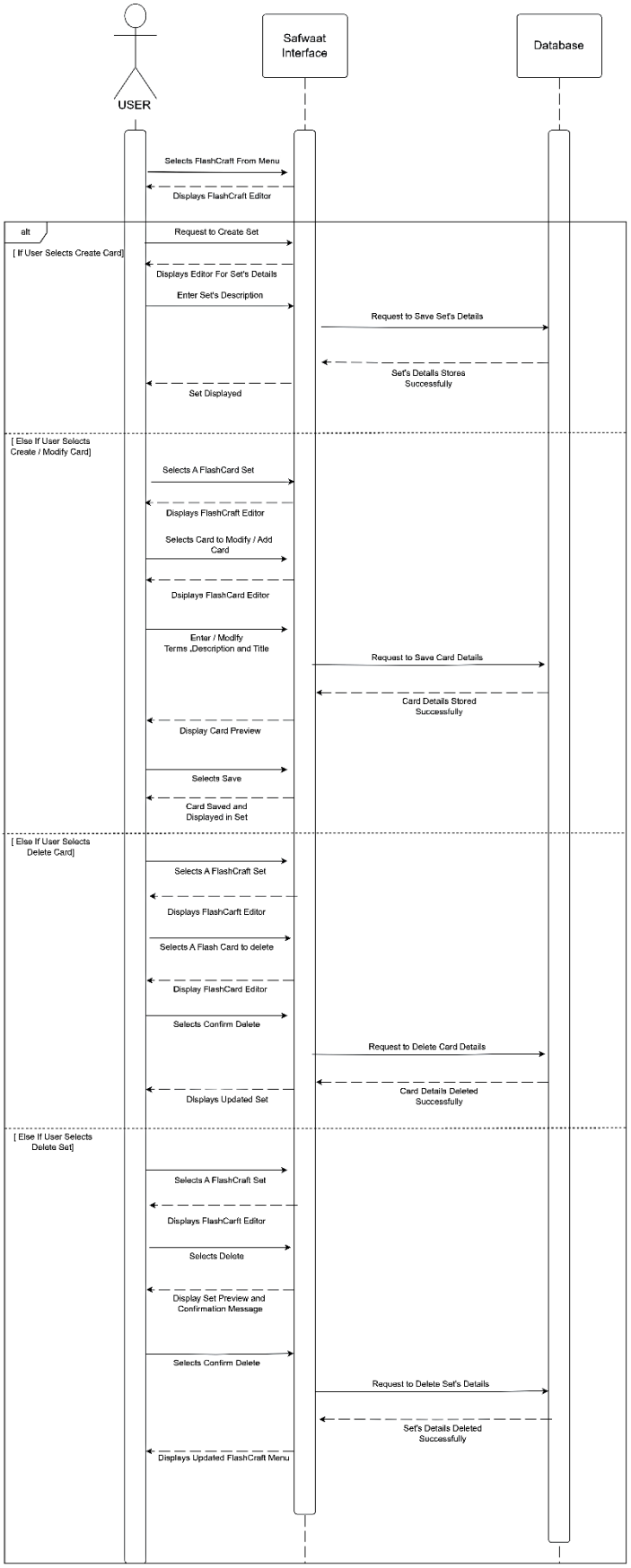


Figure 77. Visualization of Flashcraft Sequence Diagram

DESIGN VIEWPOINTS

In the sequence diagram for the "FlashCraft" feature, the process begins with the User interacting with the UI Interface to initiate the creation of flashcards. The system responds by allowing the user to specify flashcard details, including terms, descriptions, and set information. The UI Interface communicates with the Database to store this flashcard data. Users can perform actions such as modifying existing flashcards, removing specific ones, or deleting entire sets, and the system ensures data integrity with confirmation prompts. The feature also covers categorization, image insertion, sorting, sharing, and gamification elements, each represented in the sequence by interactions between the User, UI Interface, and Database. The Database plays a crucial role in storing and retrieving flashcard-related information throughout these interactions, maintaining a cohesive and organized flashcard management system.

5.6.7. USER PROFILE, FRIENDSHIP HUB, CHAT SEQUENCE DIAGRAM

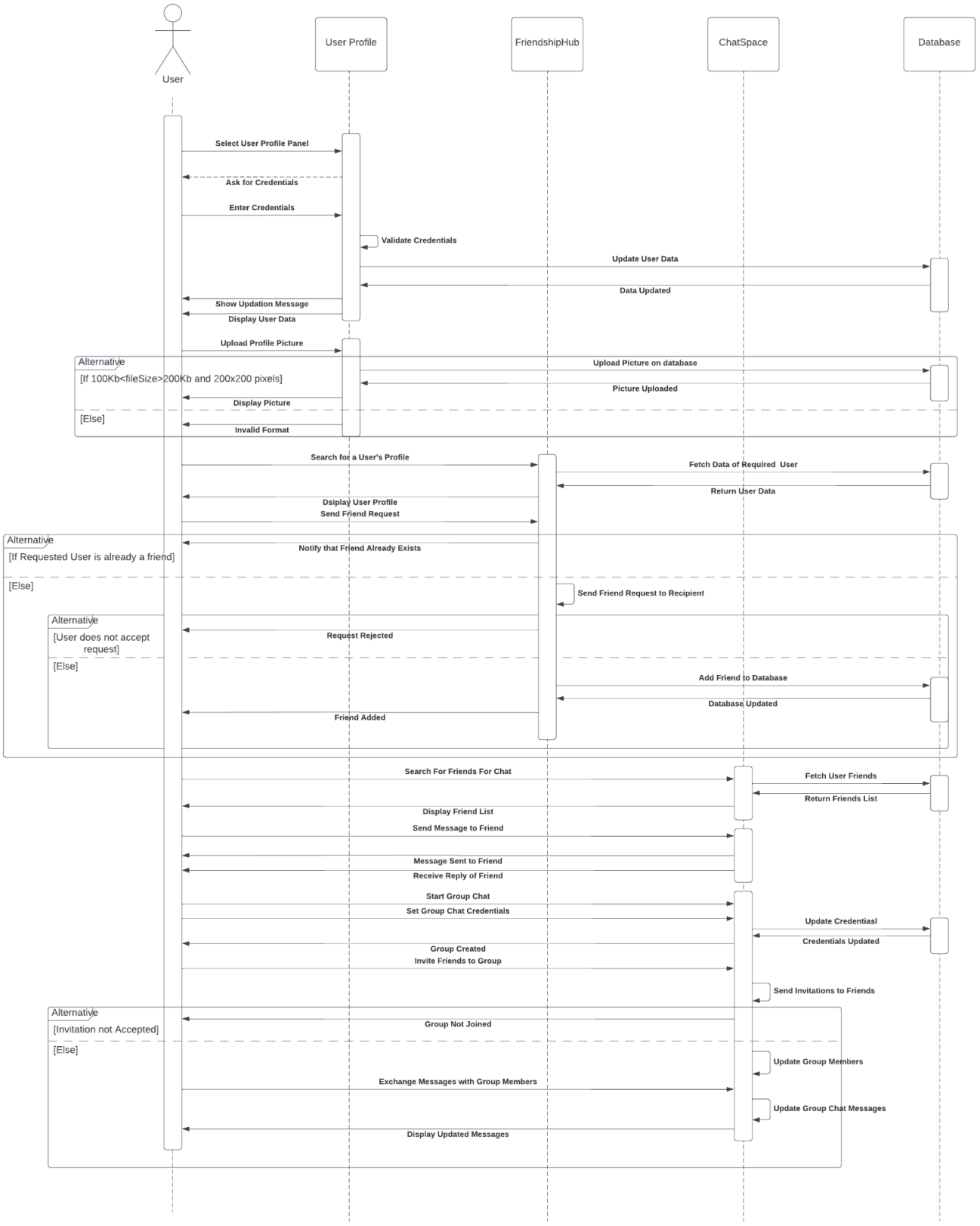


Figure 78. Visualization of User Profile, Friendship Hub, Chat sequence diagrams

The sequence diagram demonstrates how the user can browse through the functionalities of the system and create a collaborative environment for boosted learning. For this, the user completes his personal profile to achieve a more distinguished position among other users. Furthermore, the sequence diagram elaborates the necessary actions required for sending and receiving friend requests to enhance the social network of the user. This broadens the user's learning paradigm, thereby making them more prominent on leaderboards and scoreboards. Additionally, the diagram also depicts the sequence of actions for communicating with their friends by utilizing the ChatSpace functionality.

5.7. STATE DYNAMIC VIEWPOINTS

5.7.1. LOGIN AND REGISTRATION

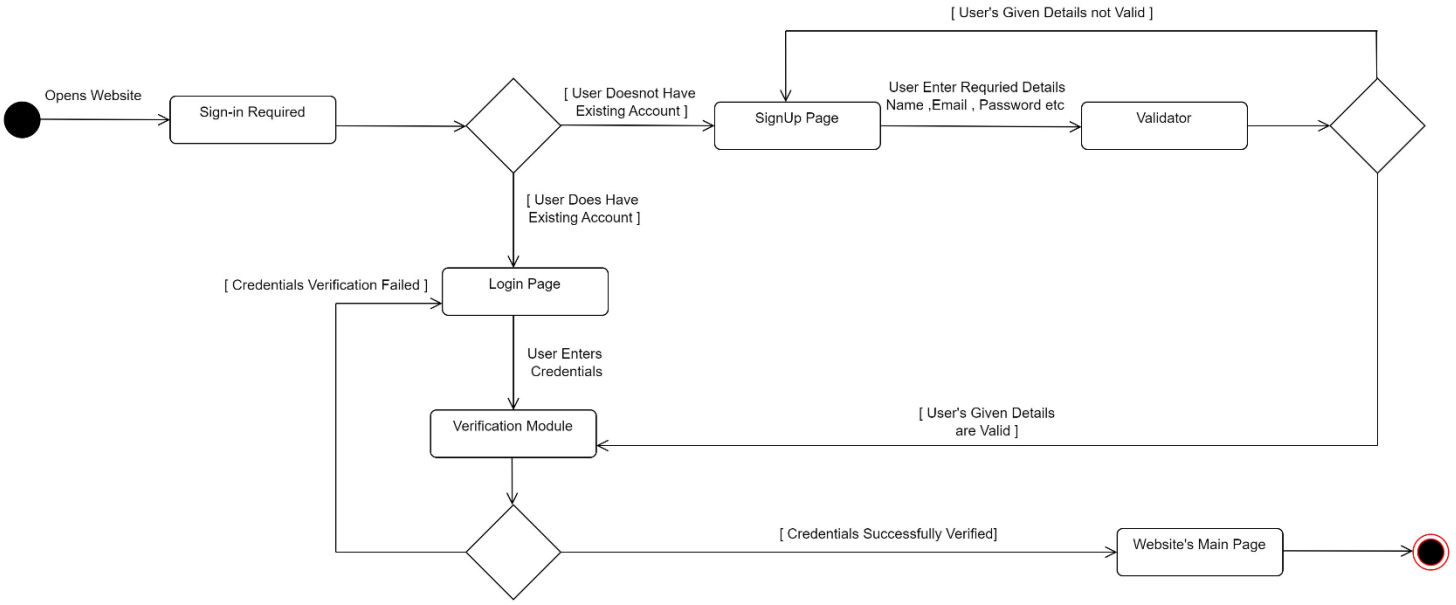


Figure 79. Visualization of Login and Registration State Diagram

- **Sign-in Required**: The state indicating that signing in is required for accessing certain website.
- **Website Main Page:** The initial state where users can navigate to different sections of the website's main content.
- **Sign-up Page:** The state where users enter their details to register for a new account.
- **Validator (for Registration Data):** The state representing the validation process for user input during registration.
- **Verification Module (for Login):** The state representing the verification process for user login credentials.
- **Login Page:** The state where users enter their credentials to log into their account.

5.7.2. TAQRAAR

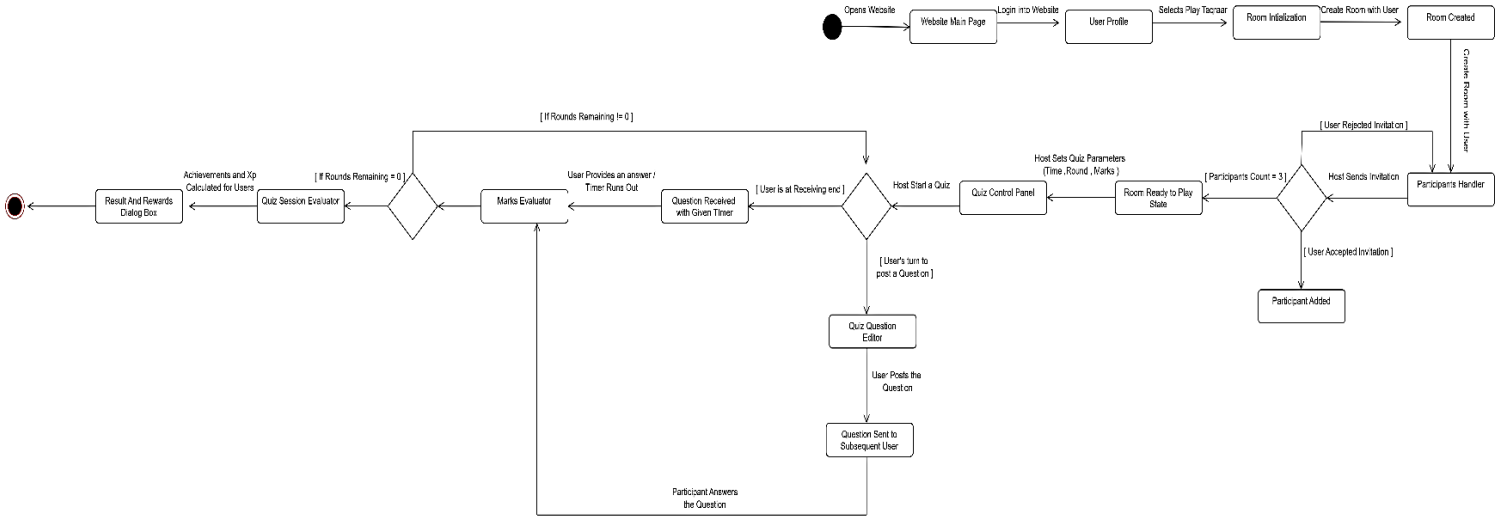


Figure 80. Visualization of Taqraar State Diagram

- **Website Main Page:** The initial state where users can navigate to different sections of the website, including Taqraar.
- **User Profile Page:** Represents the state where users can view their achievements and total marks.
- **Room Initialization:** State where a host initiates the process of creating a quiz session.
- **Room Created/Established:** Represents the state when the host successfully creates a quiz room.
- **Participants Handler:** Manages the selection of participants by the host for inclusion in the quiz session.
- **Room Ready to Play:** Indicates the readiness of the quiz room to start the session.
- **Quiz Control Panel:** Represents the state where the host sets parameters for the quiz session.
- **Quiz Question Editor:** State where participants take turns posting questions for other members.
- **Question Sent to Subsequent User**: Indicates that a question has been sent to the next participant in a round-robin fashion.
- **Question Received Within Timer**: Represents the state when a participant receives a question and provides an answer within the specified time.
- **Marks Evaluator**: Evaluates and assigns marks by participants for the answers provided.
- **Quiz Session Evaluator:** Keeps track of the progress of each round and the overall session.
- **Result and Rewards Dialog Box:** Displays the final results, earned achievements, and rewards for participants.

5.7.3. CONSISTENCY COMPANION, LEADERBOARD AND ANALYTICS

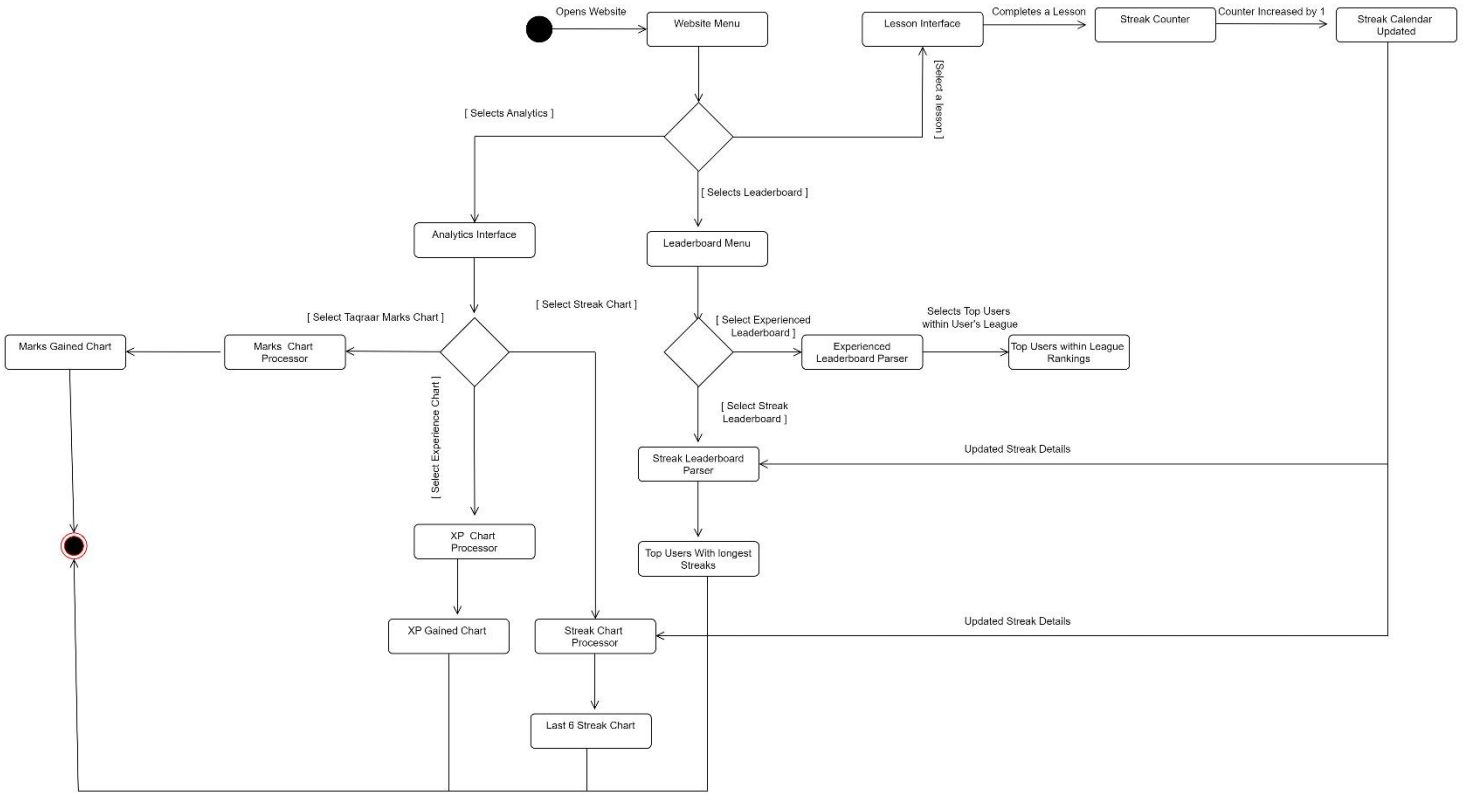


Figure 81. Visualization of Consistency Companion, Leaderboard and Analytics State Diagram

- **Website Menu:** The initial state where users can navigate to different sections of the website, including the Leaderboard.
- **Lesson Interface:** The state representing the lesson interface where users can engage in educational activities.
- **Streak Counter:** The state representing the streak counter, tracking the user's consecutive days of engagement.
- **Streak Counter Updated:** The state representing the updated streak counter after a user completes a lesson.
- **Leaderboard Menu:** The state where users can access different types of leaderboards, including Experience Leaderboard and Streak Leaderboard.
- **Analytics Interface:** The state representing the analytics interface where users can view different statistical charts.
- **Experience Leaderboard Panel:** The state representing the panel displaying the Experience Leaderboard for different leagues.
- **Streak Leaderboard Panel:** The state representing the panel displaying the Streak Leaderboard with users' current streaks.
- **Top Users Within User's League Ranking:** The state showing the top users within the same league as the user in the Experience Leaderboard.
- **Top Users with Longest Streak Ranking:** The state showing the top users with the longest streak in the Streak Leaderboard.

- **State Chart Processor:** The state representing the processing of data for generating the state chart on the website.

- **Last 6 Streak Chart:** The state displaying the chart illustrating the user's streak for the last 6 days.
- **XP Gained Chart:** The state displaying the chart illustrating XP gained over the last 7 days.
- **Marks Chart Processor:** The state representing the processing of data for generating the marks chart on the website.
- **Marks Gained Chart:** The state displaying the chart illustrating marks gained over a specific period.

5.7.4. QUEST ARENA

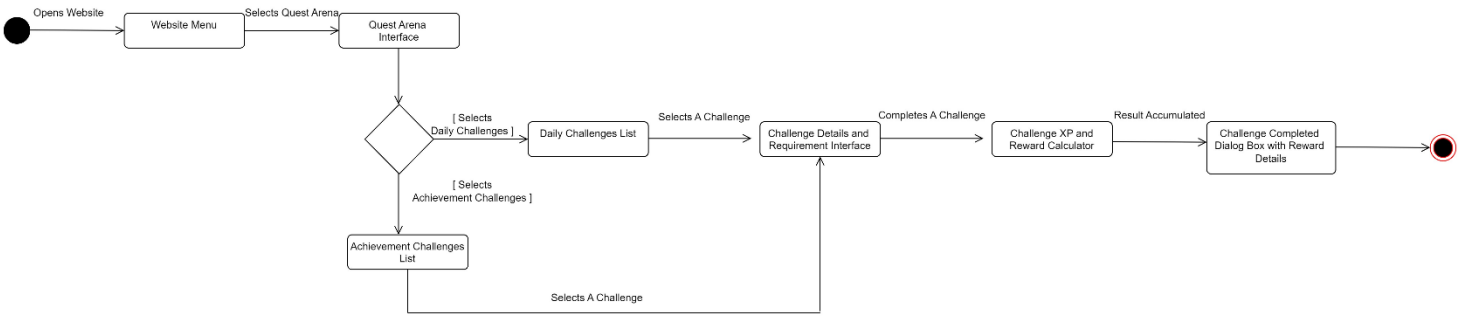


Figure 82. Visualization of Quest Arena State Diagram

- **Website Menu:** The initial state where users can navigate to different sections of the website, including the Quest Arena.
- **Quest Arena Interface:** The main interface where users can view available challenges and quests, categorized into Daily Challenges and Achievements.
- **Daily Challenges List:** The state where users can view the list of available daily challenges.
- **Achievement List Challenges:** The state where users can view the list of available achievement challenges.
- **Challenges Details and Requirements Interface:** The state where users can view the details and requirements

  Of a specific challenge.
- **Challenge XP and Reward Calculator:** The state where the system calculates the XP and rewards for completing

  a challenge.
- **Challenge Completed Dialog Box with Reward Details:** The state where users are notified of completing a challenge along with reward details.

5.7.5. NOTIFICATION SYSTEM

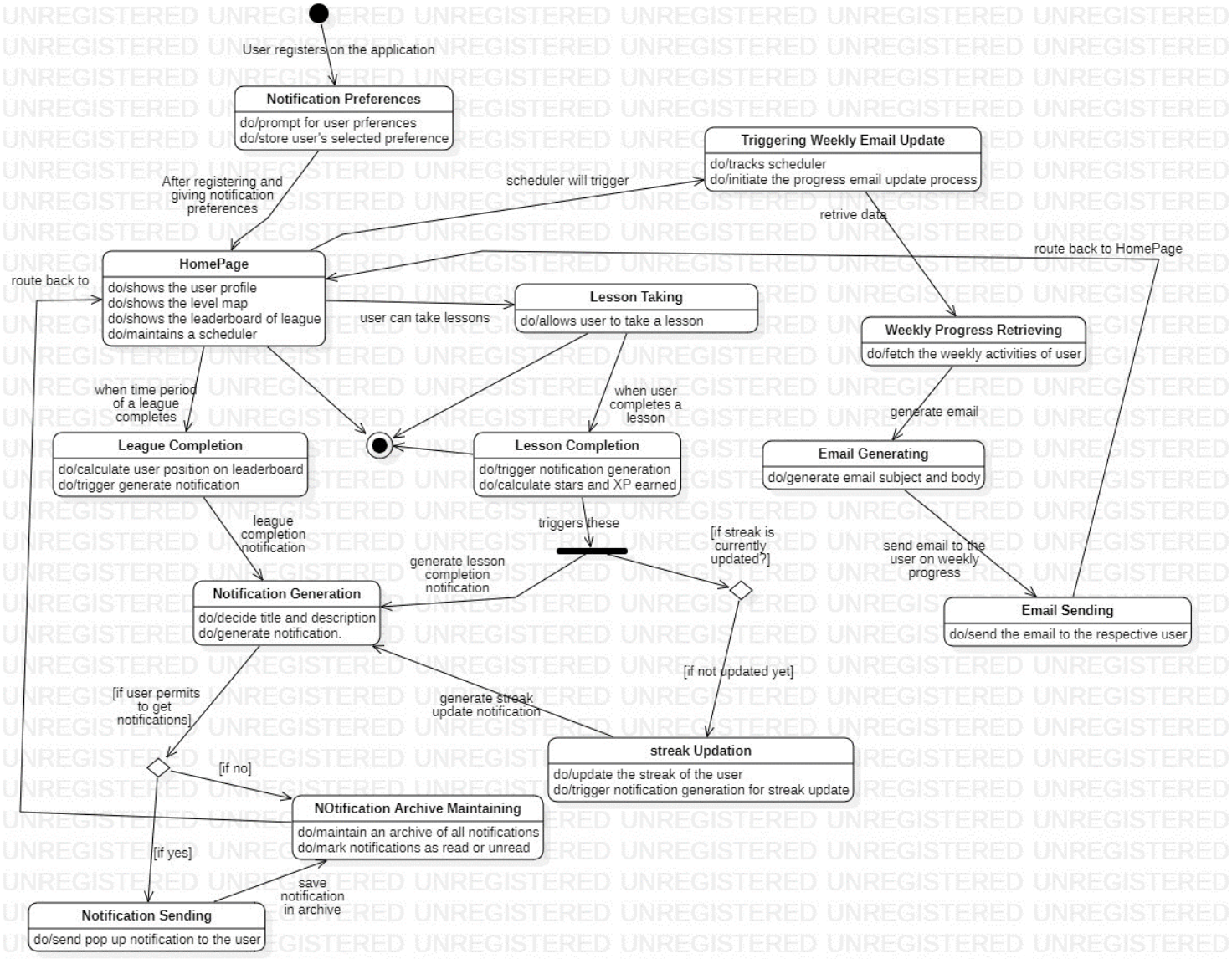


Figure 83. Visualization of Notification System State Diagram

- **Notification Preferences:** During the registration process, "Notification Preferences" is a state where users control their notification settings, choosing to receive or disable various notifications like weekly emails, lesson pop-ups, and streak updates. It reflects user customization for a personalized learning experience.

- **HomePage:** In the "Homepage" state following registration, users are greeted with a tailored learning experience. It serves as the central hub, offering quick access to lessons, challenges, and personalized content. The state symbolizes the starting point for users to engage with the Arabic learning app's diverse features.

- **Triggering Weekly Email Update:** "Triggering Weekly Email Update" initiates the process of sending personalized weekly progress emails to users who opt for this notification. It ensures timely and informative updates, fostering user engagement and motivation within the Arabic learning app.

DESIGN VIEWPOINTS

- **Weekly Progress Retrieving:** "Retrieving Weekly Progress" involves fetching user-specific data from the database to compile personalized weekly progress reports. This state ensures accurate and up-to-date information for crafting meaningful updates during the weekly progress email generation in the Arabic learning app.
- **Email Generating:** The "Email Generating" state crafts personalized email content by generating both the subject and body, utilizing the data retrieved in the previous state, this step ensures that the weekly progress email is rich in relevant information, enhancing user engagement in the Arabic learning app.
- **Email Sending:** In the "Email Sending" state, the system initiates the process of delivering the generated weekly progress email to the user.
- **Lesson Taking:** In the "Lesson Taking" state from the homepage, users are actively engaged in the learning process as they have the opportunity to take a lesson. This state facilitates a dynamic and interactive learning experience, allowing users to enhance their Arabic language skills through various activities and exercises offered within the lesson.
- **Lesson Completion:** In the "Lesson Completion" state, the system performs essential activities, including triggering the generation of notifications to update users on their progress. Additionally, it calculates the stars and XP earned by the user during the completed lesson.
- **League Completions:** In the "League Completion" state from the homepage, the system calculates the user's position on the leaderboard, reflecting their performance in the league. Additionally, it triggers the generation of notifications to inform the user about their league status and any associated rewards or achievements.
- **Streak Updation:** In the "Streak Update" state, the system checks and updates the user's streak if it has not been updated for the current day. This ensures accurate tracking of the user's learning consistency and encourages regular engagement with the application.
- **Notification Generating:** In the "Notification Generating" state, which can be triggered by both the "League Completion" and "Lesson Completion" states, the system determines the title, body, and theme of the notification. This state customizes the content based on the specific event, ensuring that users receive relevant and engaging notifications about their achievements or progress.
- **Notification Archive Maintaining:** In the "Maintaining Notification Archive" state, the system organizes and stores notifications in the archive for future reference. This ensures that users can review their past notifications, providing a comprehensive history of their achievements, progress, and important updates within the application.
- **Notification Sending:** In the "Notification Sending" state, the system sends notifications to users who have allowed the app to show notifications.

5.7.6. USER PROFILE, FRIENDSHIP HUB, CHAT

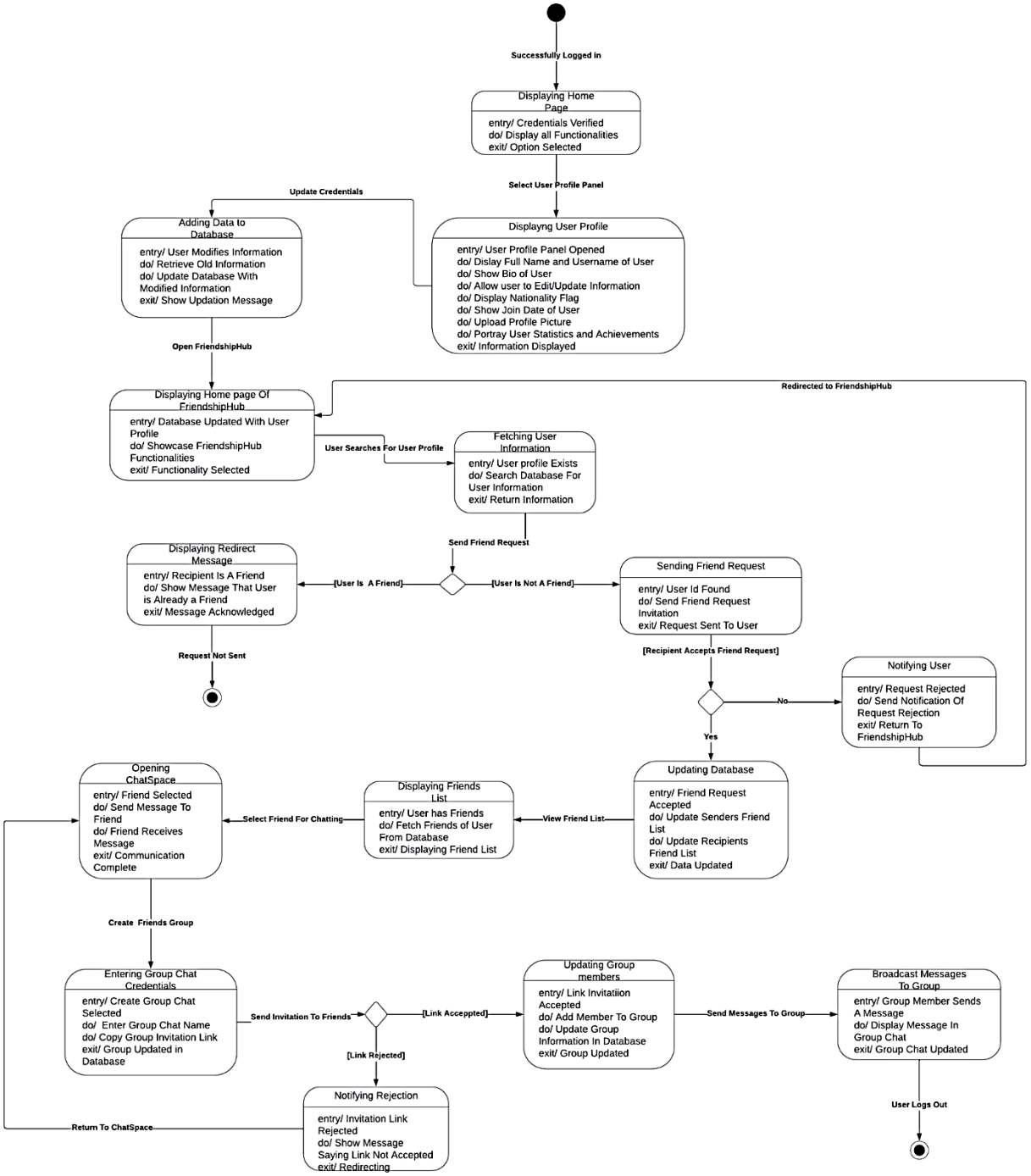


Figure 84. Visualization of User Profile, Friendship Hub, Chat State Diagram

- **Displaying Home Page:** Upon successful user login, the system verifies the credentials in the "credentials verified" entry activity. following this, the state initiates the "display all functionalities" activity, presenting the home page with all available functionalities relevant to the logged-in user. once the user interacts with an option or functionality, the state progresses to the "option selected" exit activity, allowing the user to navigate

further within the system.

- **Displaying User Profile:** When a user opts to view the "User Profile Panel," triggering the state transition, the system enters the "User Profile Panel Opened" entry activity. This state involves multiple activities aimed at presenting a comprehensive user profile. Activities include displaying the user's full name and username, showcasing the user's bio, providing options to edit or update information, showing the user's nationality flag, indicating the account's join date, enabling profile picture uploads, and presenting user statistics and achievements. Once all relevant user information is displayed or interactable, the state progresses to the "Information Displayed" exit activity.
- **Adding Data To Database:** On initiation of the "Update Credentials" action triggering the state transition, the system enters the "User Modifies Information" entry activity. Within this state, the system proceeds with activities necessary for the modification process. These activities involve retrieving the old information from the database using the "Retrieve Old Information" action and subsequently updating the database with the modified user data. Once the database is successfully updated, the system progresses to the "Show Updation Message" exit activity, notifying the user about the successful completion of the data update process.
- **Displaying Home Page Of Friendshiphub:** The FriendshipHub home page state engages the user in activities aimed at showcasing the functionalities available within FriendshipHub. These activities involve presenting various features and options provided by FriendshipHub to the user. Once the user interacts with a specific functionality or option, the state progresses to the "Functionality Selected" exit activity, allowing the user to navigate or interact further within FriendshipHub.
- **Fetching User Information:** Triggered by the user's action to "Search For User Profile," this state begins with the confirmation in the "User Profile Exists" entry activity. Within this state, the system executes activities focused on retrieving user information from the database. These activities involve querying the database to gather the requested user information. Once the necessary information is obtained, the state concludes with the "Return Information" exit activity, providing the retrieved user data.
- **Displaying Redirect Message:** This state verifies the condition in the "recipient is a friend" during the entry activity. While in this state, the system proceeds with a singular activity, displaying a message that acknowledges the existing friendship status, informing the user that they are already connected as friends. Upon the user's acknowledgment of this message, the state concludes its purpose, exiting through the "Message Acknowledged" exit activity.
- **Sending Friend Request:** This state is activated when the system confirms that the user is not currently connected as a friend, as per the prevailing condition. Upon confirming the existence of the user ID, the system proceeds with the necessary actions. It initiates the process of sending a friend request invitation to the identified user. Once the friend request is generated and sent, the state concludes, indicating successful completion through the action of the request being sent to the user.
- **Notifying User:** The Notifying state serves as a key figure in providing feedback to the user, and is prompted by the action of the recipient rejecting the friend request. Within this state, the system proceeds with the specific action of sending a notification to inform the user about the rejection of their friend request. After successfully sending the rejection notification, the state concludes its purpose, allowing the user to return to the FriendshipHub home page.
- **Updating Database:** This state is triggered by the action of the recipient accepting the friend request, validating this action in the initial phase. Inside this state, the system executes necessary tasks to ensure the

database reflects the updated friendship status. It involves updating the sender's friend list and concurrently updating the recipient's friend list to incorporate the new connection. Upon successfully completing these updates, the state concludes its operations, confirming that the data has been effectively updated.

- **Displaying Friends List:** Activated upon the user's request to "View Friend List," this state confirms the existence of friends associated with the user. Once verified, the system proceeds to retrieve the user's friends' information from the database as the primary action within this state. Subsequently, the system presents the fetched friend list to the user, providing a comprehensive display of their connections within the platform.
- **Opening Chatspace:** The state is triggered when the user chooses a friend to initiate a chat, confirming the selection of the friend in the initial phase. Within this state, the system performs essential actions by sending a message to the chosen friend and ensuring that the friend receives the message. Upon successful completion of these actions, signifying the communication's initiation, the state concludes, marking the communication between the user and the selected friend as complete.
- **Entering Group Chat Credentials:** Upon selection of the "Create Group Chat" option, the system progresses to this state, confirming the selection for initiating a group chat. Here, the system conducts crucial activities, involving the input of a group chat name and the generation of an invitation link for the group. Upon completing these tasks, signifying the group's setup, the state concludes by updating the group details in the database.
- **Notifying Rejection:** The Rejection Notice state proceeds with a singular activity, displaying a message indicating that the link has not been accepted. Following this notification, the state concludes its purpose by redirecting the user to the chat space for further interaction.
- **Updating Group Members:** Upon acceptance of the invitation link, this state proceeds with crucial activities, adding the new member to the group and subsequently updating the group information in the database. Following the successful completion of these actions, signifying the addition of the new member, the state concludes with the group information being updated accordingly.
- **Broadcast Messages To Group:** The broadcasting state executes a critical activity by displaying the sent message in the group chat. Once the message is successfully displayed in the group conversation, the state concludes its function with the group chat being updated accordingly.

5.7.7. LEVELMAP, LEARNING UNIT, 3D MODEL, SOCIAL MEDIA SHARING

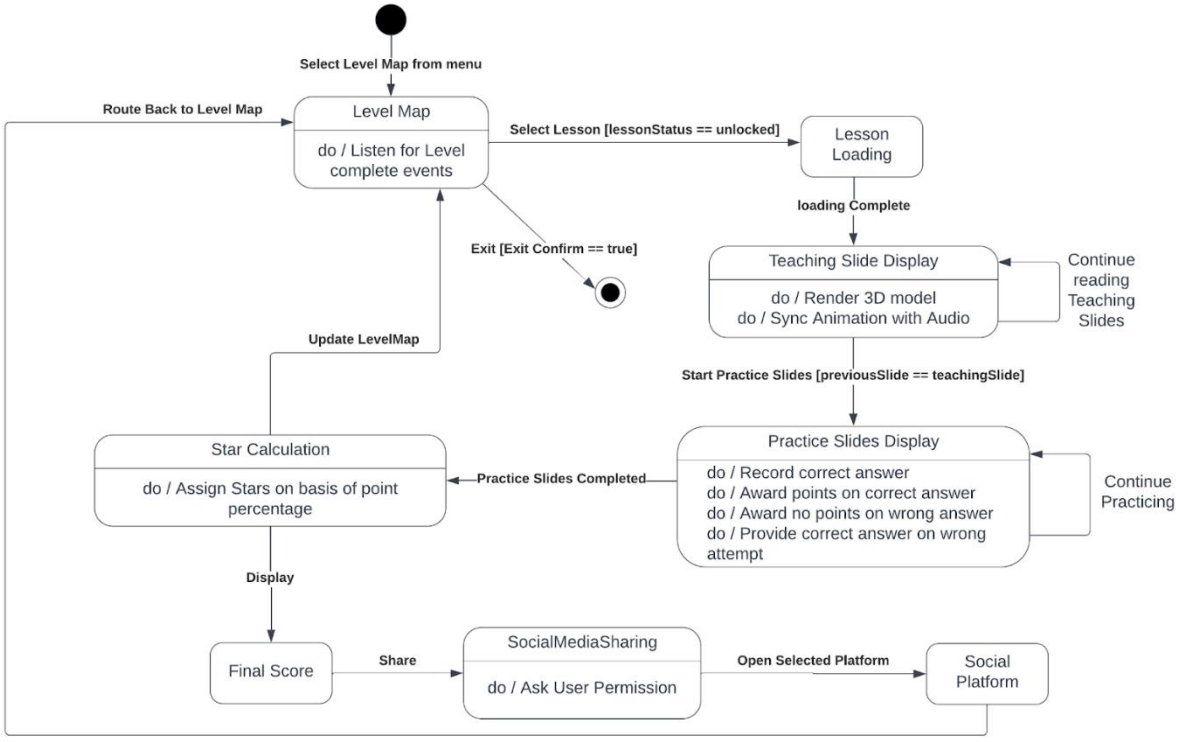


Figure 85. Visualization of LevelMap, Learning Unit, 3D Model, Social Media Sharing State Diagram

- **Level Map:** The Level Map acts as a central hub for accessing learning content. It displays all available levels, their completion status (locked or unlocked), and provides a visual overview of the learning journey. Users can browse through the map, select a level to begin their learning, and easily return to the map for further exploration.
- **Lesson Loading:** The Lesson Loading state bridges the gap between selecting a lesson and engaging with its content. It displays a loading indicator to keep the user informed while the application retrieves necessary resources, such as text, images, audio, and video. This state ensures a smooth transition and prevents user frustration by providing feedback on the loading process.
- **Teaching Slide Display:** The Teaching Slide Display state transforms complex concepts into visually engaging learning experiences. Users navigate through a series of slides, each presenting key information about Tajweed rules and pronunciation. Audio narration and 3D model visualization enhance the comprehension of the material.
- **Practice Slide Display:** Following the acquisition of theoretical knowledge, the Practice Slide Display state tests the user's understanding through interactive questions. These questions are designed to apply the learned concepts to various scenarios, solidifying knowledge and identifying areas for improvement. Immediate feedback is provided for each answer attempt, helping users track their progress and learn from their mistakes.
- **Star Calculation:** The Star Calculation state serves as a motivational tool by evaluating the user's performance on practice slides. Based on their accuracy and completion time, users are awarded stars, providing a tangible representation of their achievements. This gamified approach encourages continued learning and fosters a sense of accomplishment.

- **Final Score:** Upon completing a level, the Final Score state provides a comprehensive overview of the user's performance. It displays their overall score. This serves as both a feedback mechanism and a source of motivation, allowing users to track their progress and identify areas for improvement as they move forward.
- **Social Media Sharing:** The Social Media Sharing state allows users to share their achievements with their social circles. This celebrates their learning progress, fosters a sense of community, and potentially encourages others to explore the Tajweed Learning application. Users can customize their message and choose the platform they want to share on, maximizing the reach and impact of their achievements.
- **Social Platform:** The Social Platform state ensures a seamless transition between the application and the chosen social media platform. This promotes the application and encourages others to join the learning community.

5.7.8. FLASHCRAFT

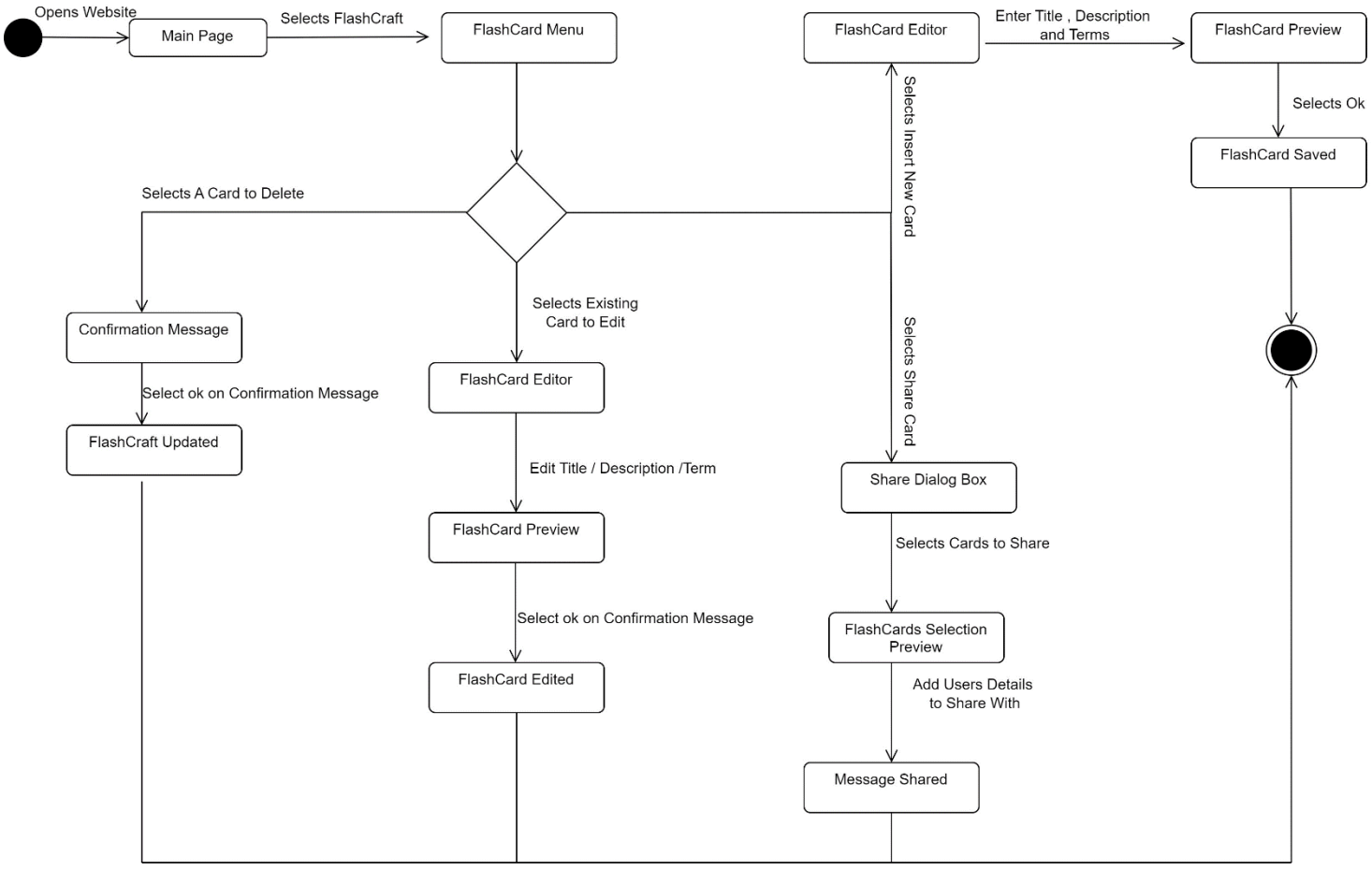


Figure 86. Visualization of FlashCraft State Diagram

- **Main Page:** The initial state where users can access different features of the FlashCraft.
- **Flashcard Menu:** The state where users can view, create, and manage flashcards and flashcard sets.
- **Flashcard Editor:** The state where users can create, modify, or delete flashcards within a set.
- **Confirmation Message:** The state where a confirmation prompt is displayed before certain actions, such as deleting a flashcard set.

DESIGN VIEWPOINTS

- **Flashcard Set:** Represents a set of flashcards created by the user.
- **Flashcard:** Represents an individual flashcard within a flashcard set.
- **Confirmation Message:** Represents the confirmation prompt shown before deleting a flashcard set, ensuring user confirmation.
- **Share Dialog Box:** Represents the dialog box where the user can share a flashcard set, promoting collaborative learning.
- **Flashcard Selection Preview:** Represents the state where the user can preview or study flashcards in a set.
- **Message Shared:** Represents the state when the user successfully shares a flashcard set.
- **Flashcard Updated:** Represents new list after modification in Flashcard Content.

6. PLANNING OF THE PROJECT

The Planning of the project is listed below:

PLANNING TABLE

DATE	TASK
12.12.2023	Login , User Profile Set Up and Registration Modules should be completed.
13.12.2023	Construction of Taqraar Module would be initialized .
20.12.2023	Lesson Unit & Level Map Feature with consistency companion should be completed .
21.12.2023	Notification System Module should be completed.
27.12.2023	.Quest Arena , Leaderboard and Analytics should be completd.
29.12.2023	FlashCraft Module should be completed and Integrated.
1.12.2023	Friendship Hub Module and 3d Models should be completed and Integrated.
2.12.2024	Taqraar Module should be completed and integrated .
3.12.2024	System Modules Integration and Backend Functionality Check.
8.01.2024	System Testing should be done.
10.01.2024	System Release.

PLANNING OF THE PROJECT

6.1 TRELLO

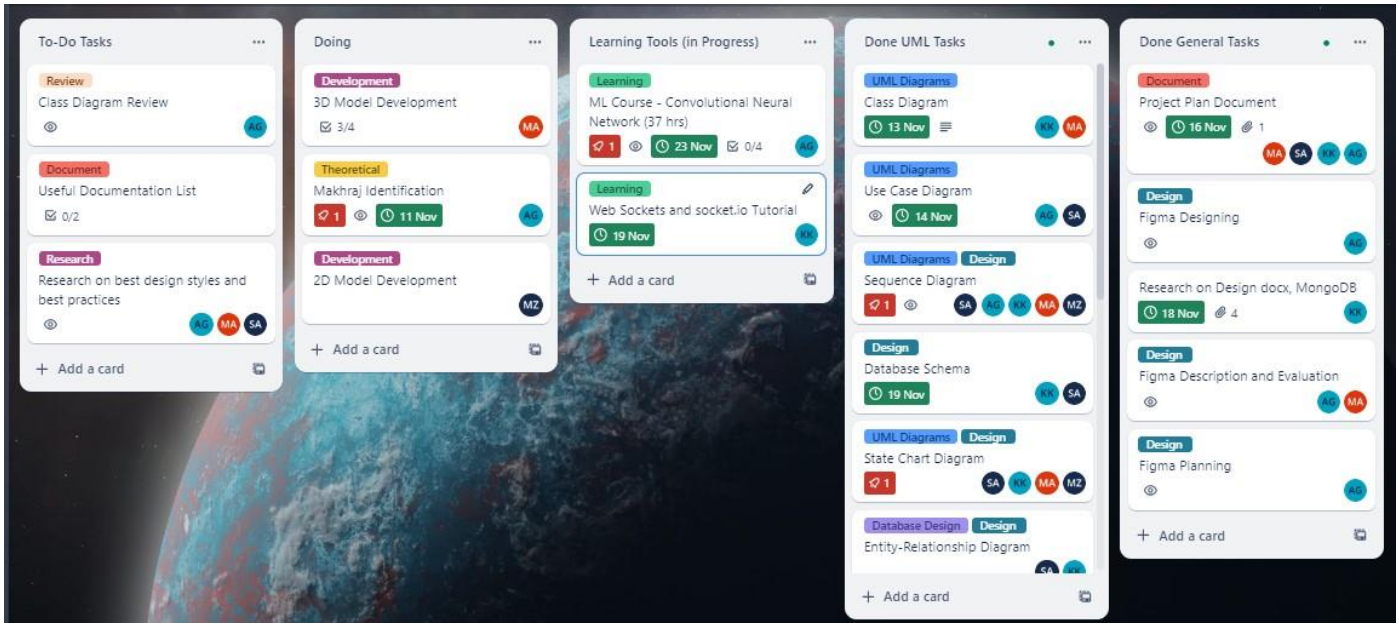


Figure 87. Trello Board -1

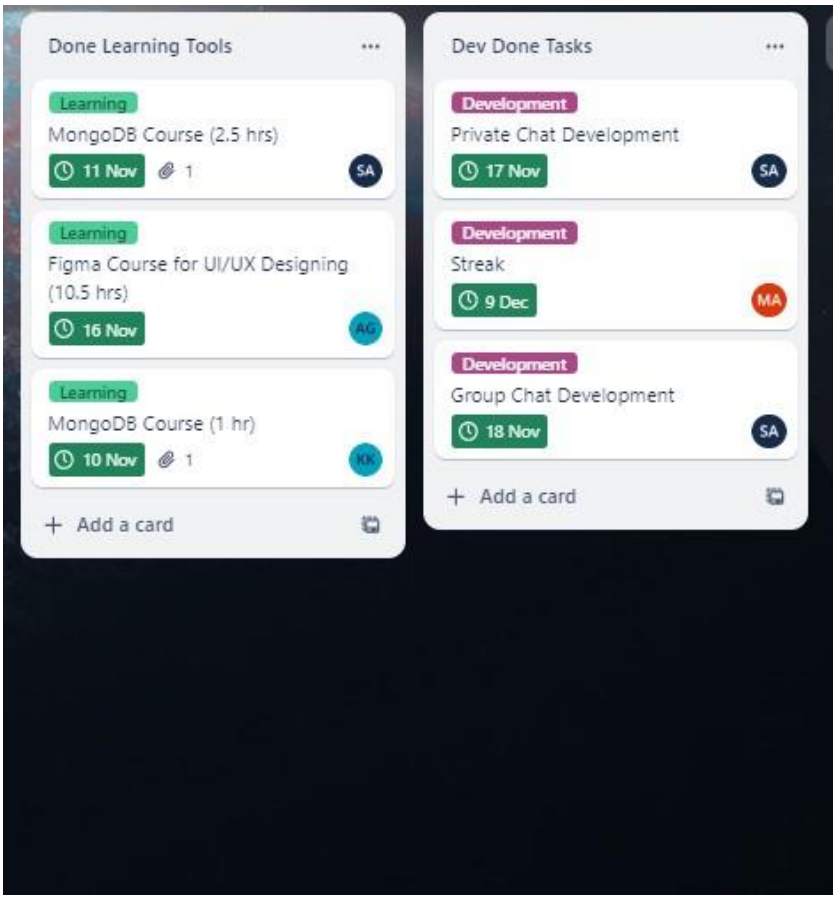


Figure 88. Trello Board -2