Norse Rune Learning, Puzzle, and Divination App & Rune Quest Web Development Project - Software Architecture & Methodologies

Overview

The Norse Rune App is a web-based application that combines learning, puzzles, and divination tools. It provides users with an engaging way to explore the meanings and history of Norse runes, solve rune-based puzzles, and perform virtual rune readings for personal insights or guidance.

Rune Quest's web development project uses a structured **Model-View-Controller (MVC) architecture** built using **Node.js, Express, React, and MongoDB**. Project management follows the **Waterfall methodology**, while **Kanban** is used for task management via **Trello** and wireframing through **Figma**.

General Features

1. Rune Learning Module

- **Interactive Guide**:
    - Displays each rune from the Elder Futhark alphabet.
    - Includes information about its name, pronunciation, meaning, reversed meaning (if applicable), and historical context.
- **Learning Tools**:
    - Flashcards to practice rune symbols recognition.
    - Quiz and puzzles to test knowledge of runes.

2. Rune Puzzles & Quiz

- **Puzzles**:
    - Solve the rune word to the English translation.
    - Codebreaker style rune puzzle.
    - Solve the meaning for a given rune quiz.

3. Divination

- **Casting Options**:
    - Single Rune Draw: Provides a single random rune fora general insight reading.
    - Three-Rune Spread: Past, Present, and Future.
    - Five-Rune Spread: Adds Advice and Outcome Positions.
- **Interpretation Features**:
    - Contextual relationships between runes in a spread.
    - Personalized readings based on user input themes.
    - Weighted rune selection for relevant themes.

4. User Dashboard

- Track progress in learning and puzzles.
- Save and revisit past divination readings.

5. Visual Enhancements

- Rune graphics include upright and reversed orientations.
- Thematic visuals like Norse-style decorations or animations for rune casting.

Software Architecture

Programming Paradigm

The Rune Quest project follows an **object-oriented programming (OOP) paradigm**, as JavaScript (used in both Node.js and React) supports encapsulation, inheritance, and polymorphism. Additionally, a **data-oriented** approach is considered when structuring MongoDB data models.

Application Architecture

The project uses the **Model-View-Controller (MVC) architecture**:

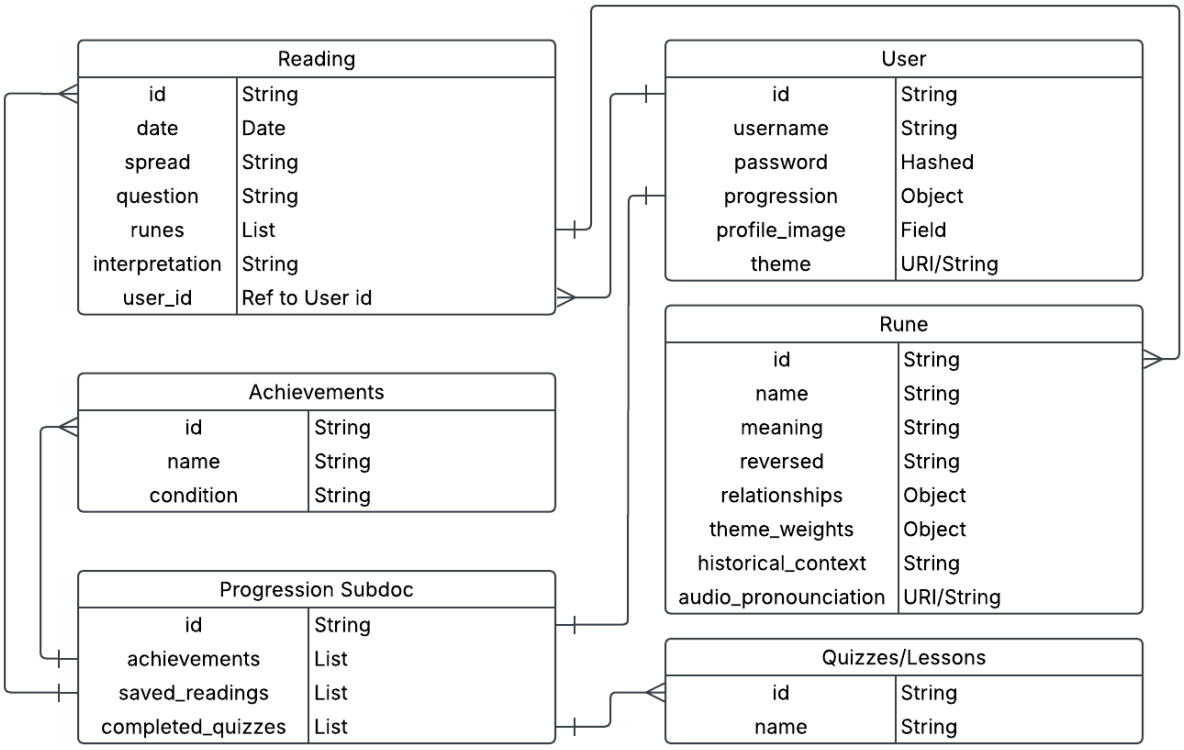
- **Model (M):** Defines data structures and interactions with MongoDB.
- **View (V):** React components render UI elements dynamically as a **single-page application (SPA)**.
- **Controller (C):** Express.js handles logic and routes between the front and back end.

This structure ensures **the separation of concerns** and **modular code organization**.

Client/Server Architecture

- **Front-End:** React (View layer)
- **Back-End:** Node.js + Express (Controller layer)
- **Database:** MongoDB (Model layer)
- **Communication:** RESTful API endpoints handled via Express
- **Authentication:** JSON Web Tokens (JWT) or session-based authentication

Entity Relationship Diagram:



Data Structures

1. Rune Dataset

Structure for storing rune data:

```
Rune = {
  "name": String, // Rune name (e.g., "Fehu")
  "meaning": String, // Upright meaning
  "reversed": String, // Reversed meaning
  "relationships": Map, // Contextual relationships with other runes
  "theme_weights": Map, // Weights for relevance to specific themes (e.g., love, career)
  "historical_context": String, // Historical background
  "audio_pronunciation": URL, // Link to audio file for pronunciation
}
```

2. User Data

Structure for tracking user progress and preferences:

```
User = {
  "id": String, // Unique identifier for the user
  “username”: String, // Username unique to user
  “password”: Hashed, // User password
  "completed_quizzes": List, // List of completed quiz IDs
  "saved_readings": List, // List of past rune readings
  "preferences": {
    "theme": String, // Default theme for divinations
    "difficulty": String // Preferred puzzle difficulty level
  }
}
```

User Stories

Persona: Adam Adamson
Occupation: none
Description: As a complete beginner with no prior experience with Norse Runes I want a web API that provides basic and easy to understand resources to assist in teaching me the fundamentals in an engaging way.

Need/Justification:

1. **Fundamental Understanding of Runes:**
   As someone with no knowledge of runes, I need to learn the basics, such as what runes are, where they come from, and why they were used. This foundational knowledge will give me context and make my learning journey meaningful and interesting.

2. **Beginner-Friendly Interface:**
   I'm looking for an easy to navigate interface that breaks down complex information into digestible pieces with clear explanations, simple language, and beginner-friendly content that will make the learning process easy to begin.

3. **Clear and Simplified Visuals:**
   I need visual representations of the runes to help me recognize their shapes and better understand their meanings. Runes are very abstract compared to my native language, so visuals help me connect the symbols to their meanings much easier.

Want/Feature/Function:

1. **Interactive Rune Introduction:**
   I'd like the API to offer an introduction to Nordic runes with interactive elements, such as a tutorial or visual guide that explains the basic concept of runes and their origins so as to not overwhelm me with information in the beginning.

2. **Interactive Rune Recognition Quiz:**
   I'm looking for an API that offers an interactive quiz or flashcard game where I can test my knowledge by matching rune symbols to their meanings, providing hands-on practice and helping me to retain the information.

3. **Beginner Rune Practice Games:**
   I'd like the API to include games or quizzes where I can practice identifying runes and their meanings such as, a game where I match the rune with its meaning or pronunciation..

4. **Simple Learning Path with Milestones:**
   I'd like the API to guide me through a controlled learning path, with ongoing milestones such as "First Rune Learned," "First Phrase Written," and "Complete Basic Lesson." to keep me feeling like I'm constantly progressing.

Persona: Donald Donaldson
Occupation: Student/Studying
Description: As a student, I'm someone with only a brief knowledge of runes with a desire to study them further.

Need/Justification:

1. **Structured Learning Path:**
   Within the API, I'm looking for a structured learning path that progressively increases difficulty. I want the ability to edit my lessons to avoid subjects I'm not as interested in.

2. **Visual Resources**:
   I'm looking for an API to provide visual representations of the runes, such as images or diagrams. As a practical learner, a visual approach helps me understand and grasp subjects faster and keeps me more likely to come back and continue taking lessons.

3. **Practical Applications:**
   I need examples of how runes can be used for personal purposes such as, writing words, creating symbols, or crafting messages. A personal understanding of how runes can be applied helps me both practice and use my rune knowledge outside of my lessons.

Want/Feature/Function:

1. **Rune Image Gallery:**
   I'm looking for the API to include a gallery of images or diagrams within its database. This allows me to visualize the ways each rune is shaped

2. **Audio Pronunciation:**
   The web API should provide an audio feature that shows the proper annunciation of each rune within its database

3. **Personalized Rune Creation Tool:**
   I want the API to have a tool that allows me to create my own unique phrases and symbols using runes.

4. **Interactive Rune Flashcards:**
   The API should offer interactive flashcards that display a rune on one side and its meaning, pronunciation, and an example of its use on the other side.

Persona: Robert Roberston
Occupation: Historian/Teacher
Description: I'm searching for a web API that will enable me to verify my knowledge and further my studies and teaching into rune knowledge, usage and understanding.

Need/Justification:

1. **Easy access to in-depth rune information:**
   In the API, I'm looking for easy access to a large and well detailed library of rune information accessible from any device that I'm also able to add to. Access to this detailed library allows me to quickly fact check any information I'm looking for and add additional information the database may be missing.

2. **Advanced learning tools:**
   When using the API, I'm looking to be able to constantly test my current knowledge. Finding advanced quizzes and challenges has been difficult, I want an application that gives me harder and more advanced quizzes and challenges. This will make it easier for me to evaluate where my current strengths and weaknesses are.

3. **Ability to explore regional and lesser-known runes:**
   I'd like an API that has a database of both broad, well known runes and runes from smaller regions. I've found that when I'm searching for and learning about lesser-known runes it's common to find conflicting information across sources, I'm looking to have access to less common runes from particular regions that receive less attention.

Want/Feature/Function:

1. **Comprehensive database of Runic Variants and Dialects:**
   I would like the API to include not just information on speaking and understanding the runic language but their historical use and sources, such as mythical tales, historical examples etc

2. **Comprehensive Rune Database with Search:**
   In the web API I would like the ability to search through a database using different classes, such as name, historical meaning, proper pronunciation.

3. **Advanced Quizzes and Challenges:**
   I'm looking for the API to include challenging learning tasks and quizzes to test even an expert's knowledge while also providing thorough feedback about what areas I did well and what areas need further looking into.

4. **Rune Translation Tool:**
   I'm looking for a tool that allows me to either input, photograph or search the web for Nordic historical text and automatically and swiftly translate it.

Ethical Web Development

As part of our efforts to adhere to ethical web development practices over the course of this project, the development team will endeavour to do the following:

- Rune Quest will be designed to run on all major modern browsers, and the design of the website will be tested thoroughly to ensure that it looks and operates well on a range of devices, primarily including smart phones, tablets and desktops.
- Any browser that supports the latest versions of React and ExpressJS will be compatible. All user interface components will be made easily accessible for any user, and any forms will be clearly defined and inclusive.
- Rune Quest will not require any personal information from a user and as such will not cause any concerns for a user's security or privacy.
- User data such as progress and settings preferences will be easy to export by the user for their own interest.
- Any sensitive data, which will only include a user's password will be encrypted and be unavailable for direct reading from any other user.
- All links will be clearly defined and human readable.
- All code will be optimised for performance and follow DRY principles. Code comments will be used frequently to help any developer understand the purpose of code and files. Open source will be used primarily.
- Code will be tested, and any errors will be handled appropriately.
- GitHub will be used for source control and all commits will have appropriate comments attached.
- HTTPS will be used once we can be confident in doing so.

API Endpoints

1. Rune Data

- **GET /API/runes**: Returns a list of all runes with essential data.
- **GET /API/runes/:id**: Returns detailed information for a specific rune.

2. User Data

- **GET /API/user/:id**: Returns user progress and preferences.
- **POST /API/user/:** Creates new user upon signup, using default preferences/
- **PATCH /API/user/:id/preferences**: Updates user preferences.

3. Divination

- **POST /API/divination**: Accepts user input (spread type, question, theme) and returns a rune reading.

4. Puzzles

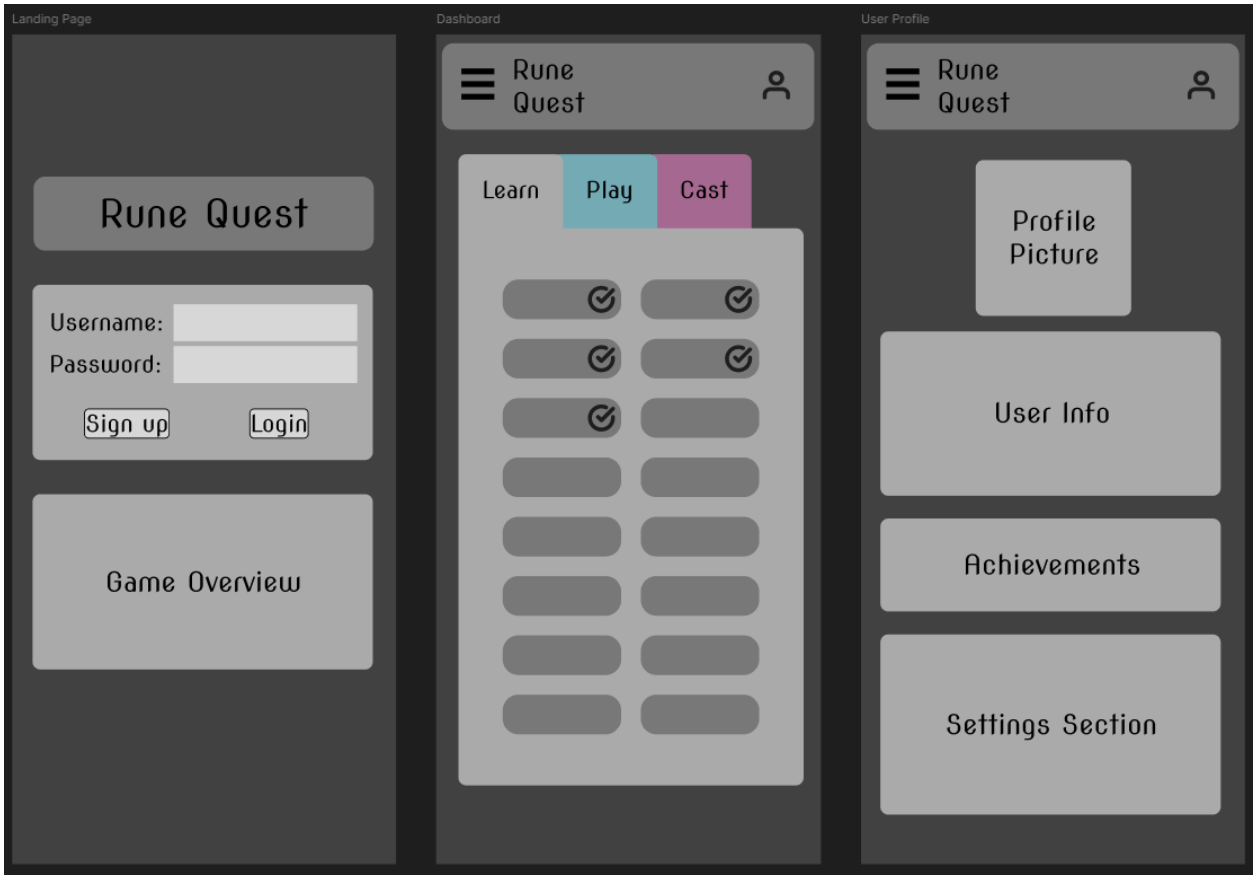
- **GET /API/puzzles**: Returns available puzzles with their status (locked/unlocked).
- **PATCH /API/puzzles/:id/complete**: Marks a puzzle as completed.

Wireframes

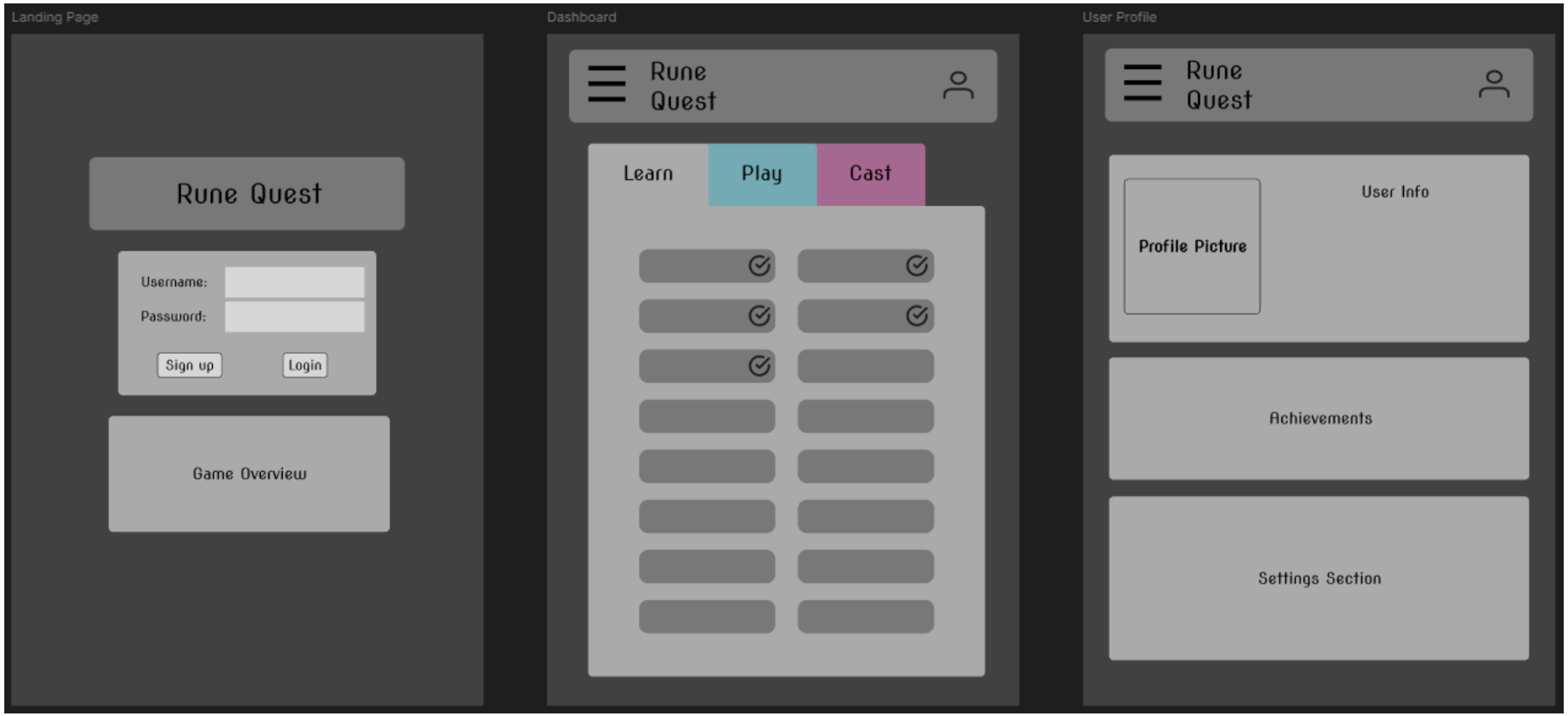
Wireframes were created using **Figma**. Key wireframes include:

- **Landing Page:** Displays the game overview and signup options.
- **Dashboard:** Displays main modules such as learning, puzzles, and rune casting.
- **Profile Page:** Shows user stats, lists completed and available quests.

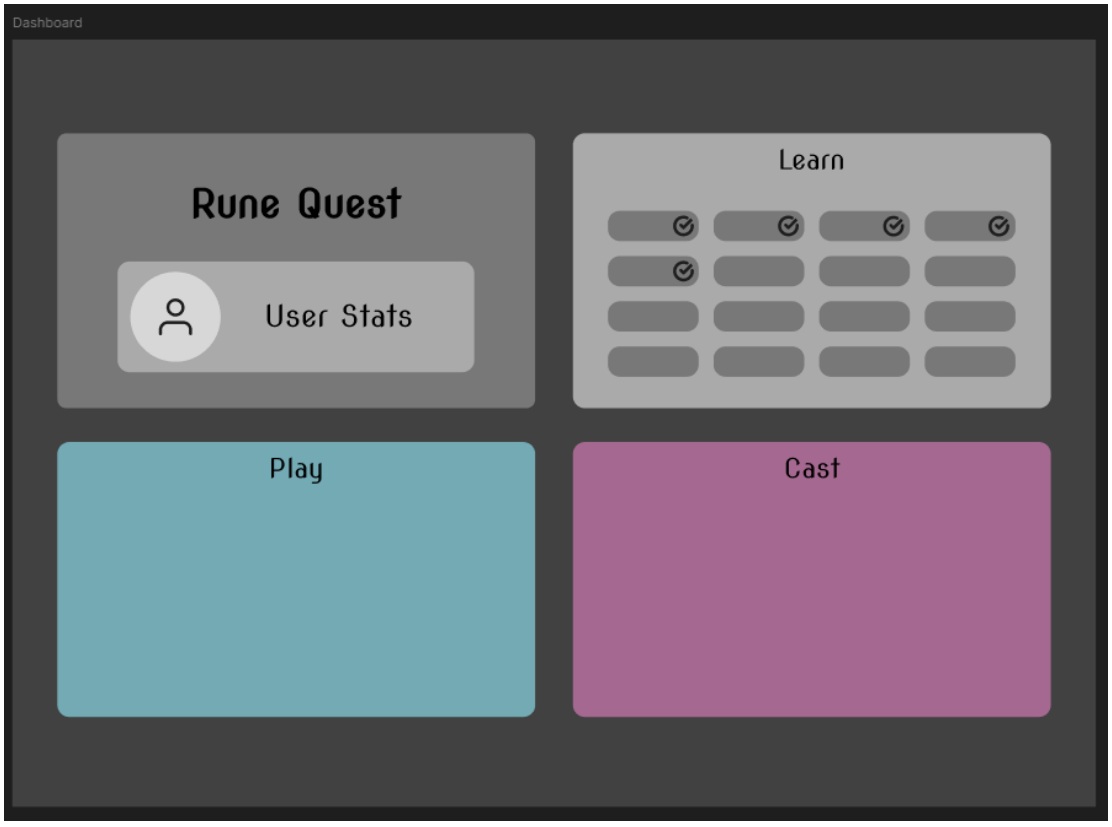
Mobile Wireframes (395 x 852):

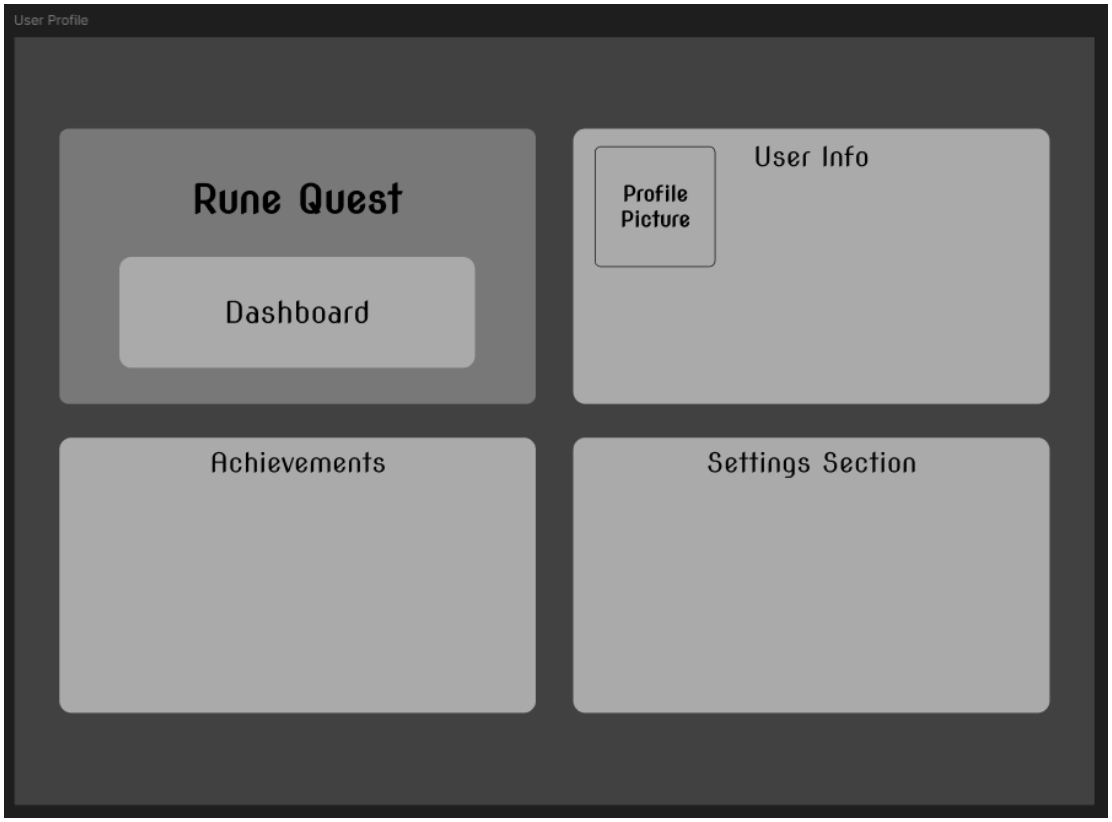


Tablet Wireframes (834 x 1194):



Desktop Wireframes (1440 x 1024):




User Profile
Rune Quest
Dashboard
Profile Picture
User Info
Achievements
Settings Section