## Task 1: Requirements and Subsystems

Functional and Non-functional Requirements: Document the specific functional

and non-functional requirements of your system. Identify key requirements and

explain why they are architecturally significant.

Subsystem Overview: List and describe the main subsystems of your project.

Provide a brief description of each subsystem’s role and functionality.

**Functional Requirements:**

|  |  |  |
| --- | --- | --- |
| **INDEX** | **FUNCTION** | **REQUIREMENT** |
| **1.1** | User Authentication | The system shall allow users to **sign up, log in, and log out** securely. |
| **1.2** | Data Storage & Retrieval | The system shall store user data and allow retrieval of past activity. It should also allow the system to store and retrieve data of levels and lessons per user. |
| **1.3** | Stock Market Simulation | The system should be able to simulate a level where users can play a level in which we simulate a stock market in a gamified manner. The stocks will be based on historical data and the users input would not have effects on the stock. |
| **1.4** | Virtual Currency | The system shall enforce a virtual currency per user which they earn per lesson and invest in every simulation level. |
| **2.1** | News Feed | Before the start of a level, the system shall display news feed for the particular level. The simulation time and entry amount as virtual money will be displayed. The news is historical in nature. |
| **2.2** | Simulation Time selection | The user shall be allowed to select the amount of time they want the simulation to be played out. |
| **2.3** | Buy-In System | The system shall allow users to use the virtual currency and play the simulation. |
| **3.1** | Stock Market | The simulation shall provide multiple stock options into which the user can invest. |
| **3.2** | Historical Stock Market Data | The simulation shall be based on historical stock market data. It is from a pre-defined data set. |
| **3.3** | Buy and sell stocks | The system shall allow players to buy/sell stocks in the simulation game. |
| **3.4** | |  | | --- | |  |  |  | | --- | | Portfolio Tracking | | The system shall track **user portfolios**, including stock holdings, cash balance, and transaction history. |
| **3.5** | Simulation End | The system shall automatically **end the simulation** after the pre-set simulation time. |
| **3.6** | Pause and Resume Strategy | The user won’t be able to pause a simulation and if the user exits the simulation, the user will lose all the buy in. They cannot play the same level again to gain virtual coins. |
| **4.1** | Lessons | The system shall provide **different lessons** with structured reading material and mini games after that. |
| **4.2** | Lesson Navigation | The system shall allow users to **navigate between lessons** in a structured manner. |
| **4.3** | Quizzes | Each Lesson would have a quiz where the user can test their knowledge and earn virtual currency. |
| **5.1** | Leaderboard | The system shall display the highest scores per level |
| **6.1** | Level Creation | A level designer shall have all the APIs available to create and curate a level. Basic crud operations would be provided per parameter. |
| **6.2** | Level parameter setting | The system shall allow the level designer to set the hyperparameters per level |
| **6.3** | Level unlock strategy | Each level would have a different buy in and also a lesson requirement which you need to fulfill before you can play a level. |
| **7.1** | Competitive levels | You can play a level with your friends using a code. It should be a real time multiplayer simulation. |
| **8.1** | User Profile and Stats | A profile page should show user statistics, progress, past performance, and ranking history |

**Non Functional Requirements:**

|  |  |  |
| --- | --- | --- |
| **INDEX** | **QUALITY ATTRIBUTE** | **DESCRIPTION** |
| **1** | Usability | * The system shall have an intuitive UI/UX for easy navigation across simulation levels, lessons, and stock market interactions. * The system shall provide tooltips, tutorials, and feedback messages for first-time users. * The system shall ensure accessibility compliance (e.g., WCAG standards for color contrast, text size). |
| **2** | Compatibility and  Interoperability | * The system shall be compatible with major browsers (Chrome, Firefox, Edge, Safari). |
| **3** | Availability | * The system shall have an uptime of at least 99% to ensure reliability. * The system shall allow users to access their profiles, progress, and simulation history at any time. |
| **4** | Security | * The system shall use **OAuth 2.0** or **JWT-based authentication** for secure login. * The system shall enforce **role-based access control (RBAC)** for different users (players, level designers, admins). |
| **5** | Performance | * The system shall respond to user actions (buying/selling stocks, submitting quizzes) in **under 500ms**. * The stock market simulation should update **in real time with minimal latency** (< 1 second). * The game shall handle at least **5 concurrent multiplayer users** without performance degradation. |
| **6** | Maintainability | * The system shall follow **modular architecture** to allow easy updates and bug fixes. * The API shall be **well-documented** for future development and third-party integrations. * The system shall use **version control (Git)** and CI/CD pipelines for smooth deployment. |
| **7** | Scalability | * The system shall support **thousands of concurrent users** without a significant drop in performance. * The stock simulation should scale to accommodate increasing levels and historical data without lag. * The backend shall be **horizontally scalable** to handle traffic spikes. |
| **8** | Data Integrity | * The system shall ensure **data consistency** in all stock transactions and portfolio records. * The database shall implement **ACID properties** to prevent data corruption. * In case of a system crash, users should not lose transaction history or virtual currency balances. |

* Data management - stores and retrieves stock market data
* Levels system - incharge of displaying the data and simulating and allowing level creators to create levels.
* Lesson system - stores all different lessons and content for the lessons and quizzes.
* User system – it manages all the user related activities and profile storing the user progress, portfolio, currency balance and leader board.