**COMSATS University Islamabad,   
Abbottabad Campus**

**SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)**

**for**

**Treasure Hunt Adventure**  
Version 1.0

***By***

**RAJA FATASH ABBASI CIIT/SP22-BSE-127/ATD**

**HUZAIFA TANVEER CIIT/SP22-BSE-140/ATD**

***Supervisor*Mr. Ahsan Khan**

***Bachelor of Science in Computer Science (2022-2026)***

**Table Of Content**

1 [**Introduction** 5](#_Toc201615559)

**2** [**Design Methodology – Object-Oriented Programming (OOP)** 5](#_Toc201615570)

**3** [**Software Process Model – Iterative Incremental Model** 6](#_Toc201615573)

**4** [**System overview** 6](#_Toc201615576)

4.1[Architectural design 7](#_Toc201615581)

4.2[Process flow/Representation 9](#_Toc201615582)

**5** [**Design models [along with descriptions]** 11](#_Toc201615587)

**6** [**Data design** 18](#_Toc201615588)

[6.1 Data Organization and Entities 18](#_Toc201615590)

[6.2 Data Storage Technologies 18](#_Toc201615598)

[6.3 Data Processing and Flow 18](#_Toc201615599)

**7** [**Algorithm & Implementation** 20](#_Toc201615616)

**8** [**Human interface design** 23](#_Toc201615617)

8.1 [User Interaction Overview 23](#_Toc201615619)

8.2 [AR Gameplay Interface 23](#_Toc201615627)

8.3 [Feedback and Notifications 23](#_Toc201615633)

**9** [**Screen images** 24](#_Toc201615641)

[9.1 Screen objects and actions 24](#_Toc201615642)

**10** [**Appendix I** 24](#_Toc201615643)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

**Supervised by**

**<Supervisor’s Name>**

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Introduction**

*Treasure Hunt Adventures* is an interactive mobile game that blends Augmented Reality (AR) with real-world exploration, designed to offer players a gamified treasure-hunting experience using their smartphones. The game leverages Unity with ARFoundation to place virtual treasures in physical environments, requiring players to move, scan, and interact with AR objects. As they explore different locations, players solve engaging puzzles, collect digital rewards, and compete on a real-time leaderboard.

The project scope completed so far includes the design and partial development of several key modules. These modules include:

* **User Interface Module**: Handles sign-in, registration, onboarding tutorial, and settings.
* **AR Engine Module**: Utilizes ARCore/ARKit and GPS to detect flat surfaces and anchor virtual content.
* **Puzzle & Clue System**: Generates and validates puzzles such as riddles, symbol-matching, and memory challenges.
* **Treasure Collection Module**: Allows users to interact with AR-placed objects to collect virtual treasures.
* **Leaderboard System**: Maintains live rankings based on player performance, synced through Firebase.
* **Rewards & Achievement System**: Tracks progress and awards digital badges, coins, and bonuses to encourage continued play.

The system also integrates with Firebase for user authentication, real-time database storage, and cloud sync. These foundational components lay the groundwork for a robust, engaging, and scalable AR-based mobile game experience.

**Design methodology and software process model**

**Design Methodology – Object-Oriented Programming (OOP)**

The Treasure Hunt Adventures system is being developed using the Object-OrientedProgramming (OOP) methodology. OOP is most appropriate for this project due to its modular structure and the need for clear abstraction between game components such as the AR Engine, Puzzle System, Treasure Collection, and Leaderboard Module. OOP allows us to define classes and objects that represent real-world entities like players, treasures, and puzzles, supporting better code reuse, scalability, and maintainability.

Each module in the game—such as the User Interface, AR functionality, and Firebase integration—is encapsulated within distinct classes with defined responsibilities and interactions. This design ensures that updates, enhancements, or debugging can be carried out without affecting unrelated parts of the system. Moreover, object-oriented principles like inheritance and polymorphism will help in extending game features like additional puzzle types or reward mechanisms in future iterations.

**Software Process Model – Iterative Incremental Model**

The development of this project follows the Iterative Incremental Process Model. This model is chosen due to the dynamic and exploratory nature of AR-based mobile games, where user feedback, testing, and gradual integration play a crucial role. The project is divided into multiple manageable iterations, with each increment adding functionality—starting with core modules such as user authentication and AR scanning, followed by puzzle integration, reward systems, and leaderboard functionality.

This approach allows the development team to build a functional baseline early in the process, test it thoroughly, and improve upon it in each cycle. It also facilitates continuous user validation and minimizes risk by allowing early detection and resolution of issues. The model aligns well with the mobile game development life cycle, especially when deploying modules like Firebase authentication, AR surface detection, and puzzle logic in stages.

**System overview**

*Treasure Hunt Adventures* is a mobile-based augmented reality (AR) game that transforms real-world exploration into an interactive treasure hunting experience. Designed for Android and iOS platforms using Unity and AR Foundation, the game overlays virtual treasure elements in real-world environments by utilizing GPS, camera input, and AR surface detection technologies. Players navigate their surroundings, solve engaging puzzles, collect treasures, and climb the leaderboard—all while interacting with virtual content seamlessly blended into their physical environment.

The system is built with a modular design, featuring distinct components such as User Authentication, AR Engine, Puzzle & Clue Management, Treasure Collection, Leaderboard Tracking, and Rewards & Achievements. Firebase serves as the backend for authentication, real-time data storage, and leaderboard synchronization. The AR Engine leverages ARCore (for Android) or ARKit (for iOS) to detect flat surfaces, place AR objects, and respond to user gestures.

From a functional perspective, the system guides users through a simple onboarding and login process, then transitions them into AR-based gameplay where they explore nearby areas, detect virtual clues, solve memory and logic-based puzzles, and earn digital rewards. The system ensures real-time syncing of game progress and achievements while providing engaging feedback through the UI and animations.

By merging physical activity with cognitive challenges and immersive technology, *Treasure Hunt Adventures* aims to create a playful, educational, and rewarding experience that promotes both mental and physical engagement. Its scalable design and backend integration make it suitable for solo play, with future potential for multiplayer enhancements and commercial partnerships (e.g., branded treasure hunts).

**Architectural design**

**Onion Architecture:**

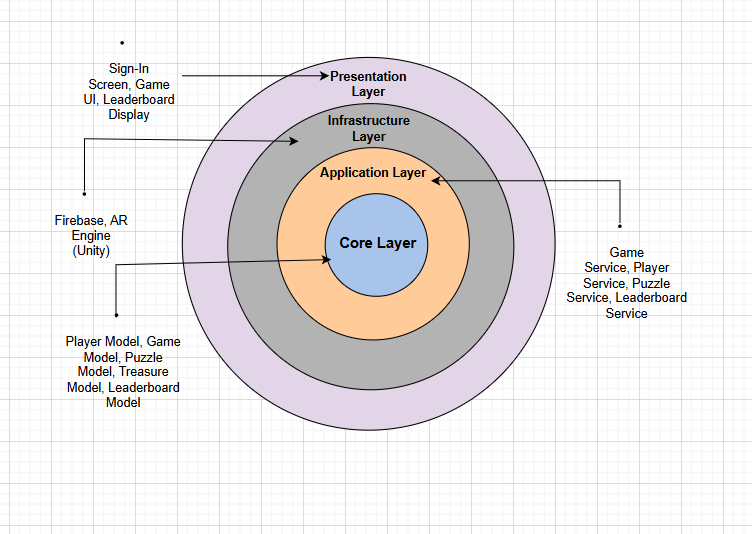


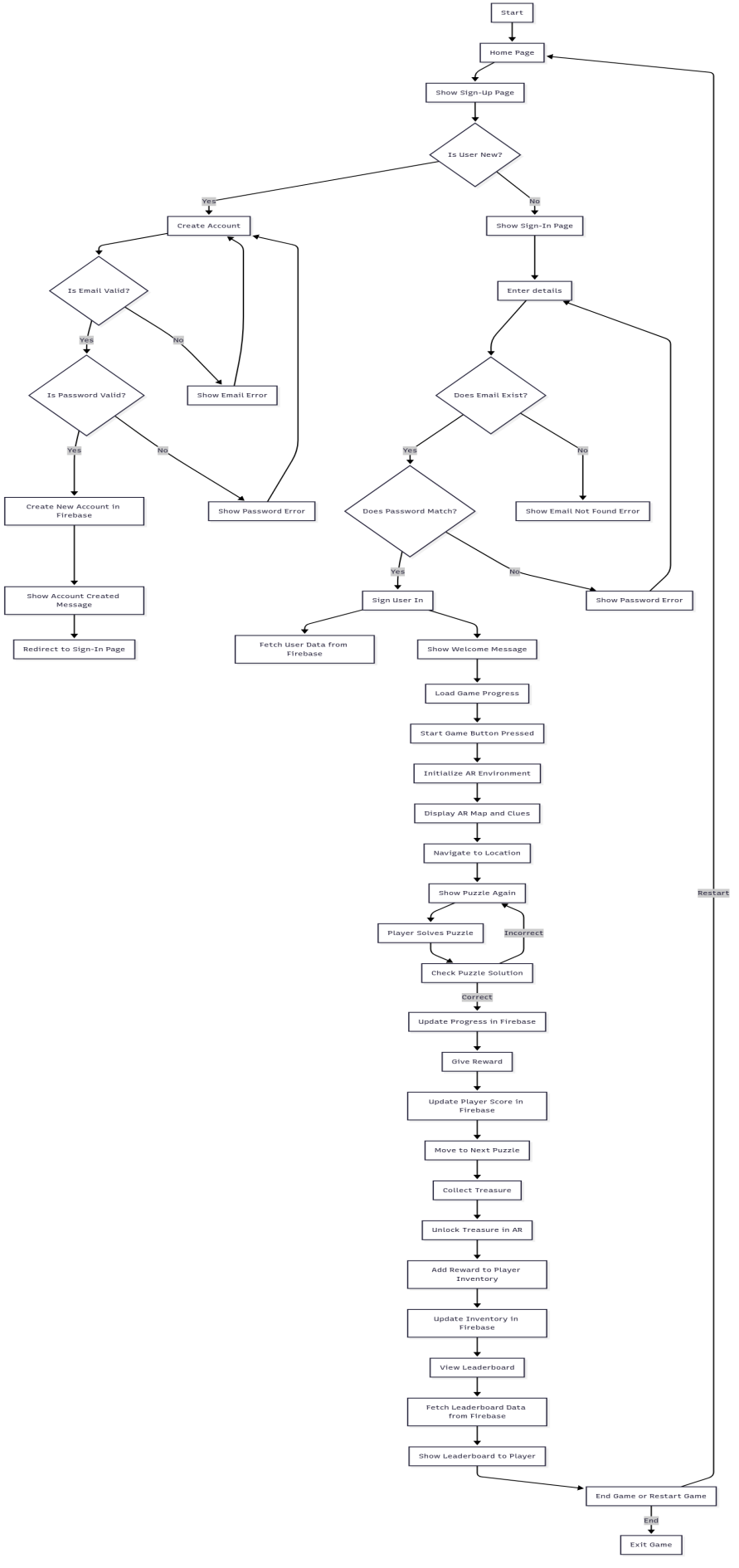
Fig 1.1 Onion Architecture for Treasure Hunt Adventure

**Process flow/Representation**

The activity diagram of *Treasure Hunt Adventure* outlines the complete gameplay flow and system interaction from the player's perspective. The process begins when the player launches the game on their Android device, initiating the AR environment and backend services. The user is then presented with a login or registration interface. If the user is new, they must register by creating an account; otherwise, existing users can log in. Post-authentication, the user has the option to update their profile, such as adding a username or avatar.

Once authenticated, the player starts a new game session. The AR Manager component activates and detects real-world planes using AR Foundation technology. Once a surface is found, virtual treasure clues are spawned within the environment. The user then transitions into the game session phase where they are required to solve a puzzle tied to each treasure. Successful puzzle completion allows the player to collect the treasure.

After collecting the treasure, the system verifies the collection and rewards the player with points or virtual items. The user's score is updated accordingly, and if certain criteria are met, achievements are unlocked. The game session can be ended by the player at any time, after which the user is allowed to view the leaderboard and compare scores with other players globally. Finally, the player can log out of the application, completing the gameplay cycle. This diagram encapsulates authentication, AR interaction, reward progression, and leaderboard integration in a streamlined, user-centric flow.



**Fig 1.2** Activity Diagram for Treasure Hunt Adventure

**Design models**

**The applicable models may include:**

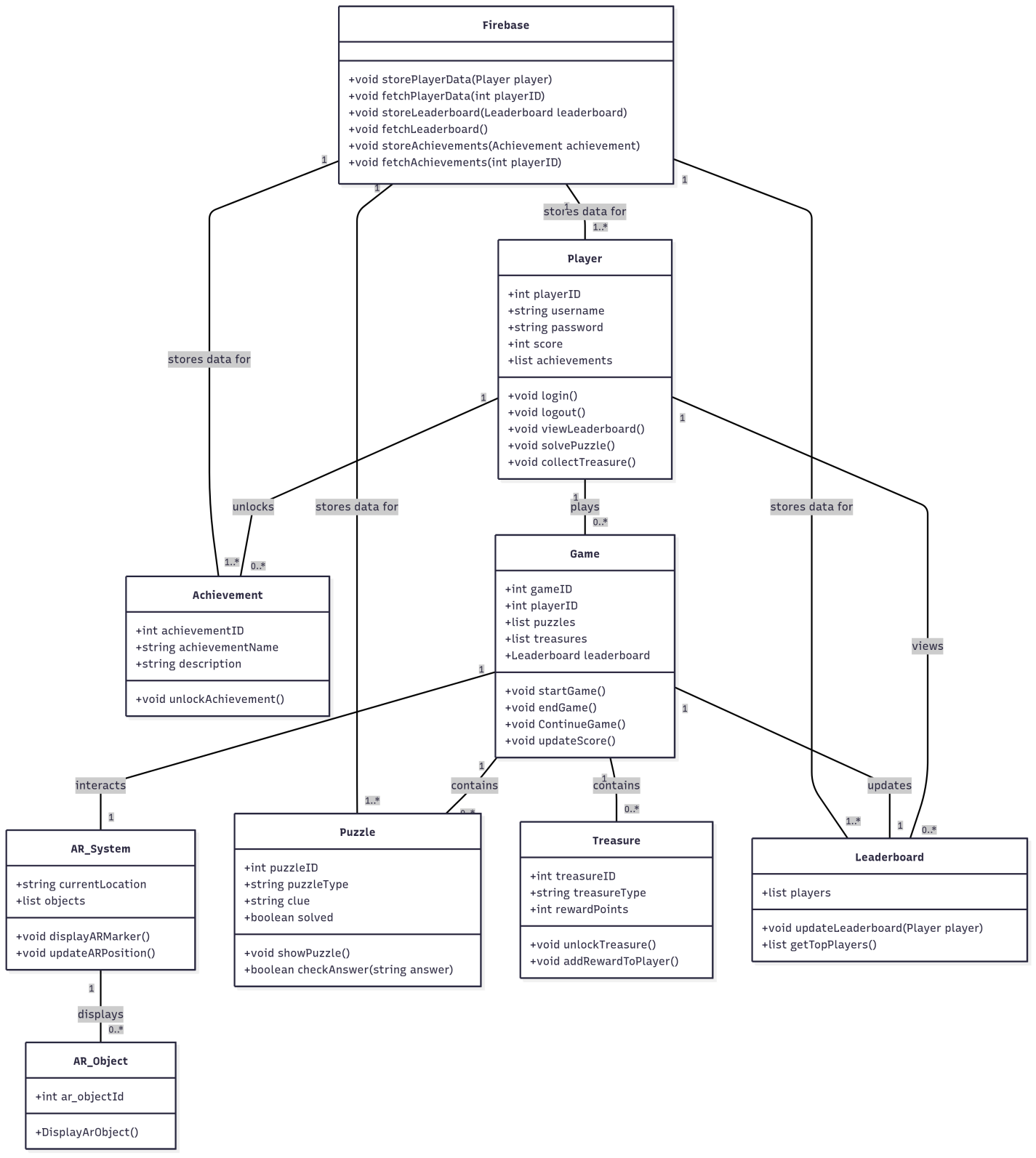
* **Class Diagram**

This class diagram represents the object-oriented structure of the *Treasure Hunt Adventure* mobile AR game. The central entity is the Player class, which holds user credentials, score, and achievements. Each player can initiate multiple GameSession instances, which manage the gameplay timeline, score, and status.

Within a game session, the system aggregates multiple Puzzle and Treasure objects. Puzzles include challenge-specific data like type, question, and difficulty, while treasures are defined by GPS location, collection status, and reward points. The ARManager handles real-world plane detection and virtual object spawning for AR interaction.

Players receive Reward instances based on performance and can unlock Achievements during gameplay. Both are linked back to the player with one-to-many associations. The AuthManager manages sign-up, sign-in, and OTP verification, while the FirebaseManager handles data synchronization, loading, and saving through Firebase.

This architecture ensures modularity, data persistence, and a robust connection between real-world interaction and in-game logic.



**System Sequence Diagram**

The system sequence diagram illustrates the complete user interaction flow within the Treasure Hunt Adventure game. It begins with the registration and login process, where the player submits their credentials through the AuthManager, which communicates with FirebaseManager to save and load user data. Upon successful authentication, the user initiates a game session via GameSession, which logs the start time and session details.

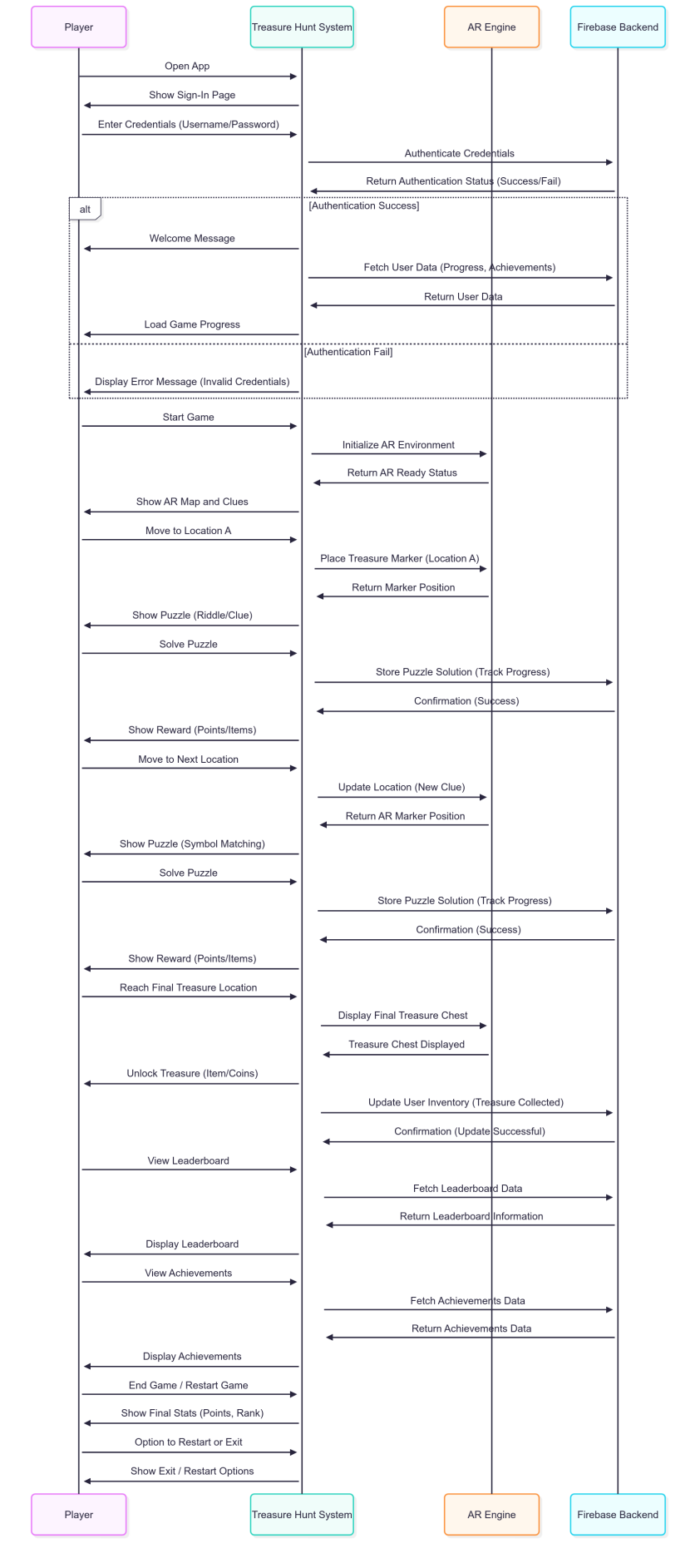


Fig 1.4 System Sequence Diagram for Treasure Hunt Adventure

**Sequence Diagram**

The following diagrams shows the process sequence of our system.

**User Sign-In Sequence Diagram**

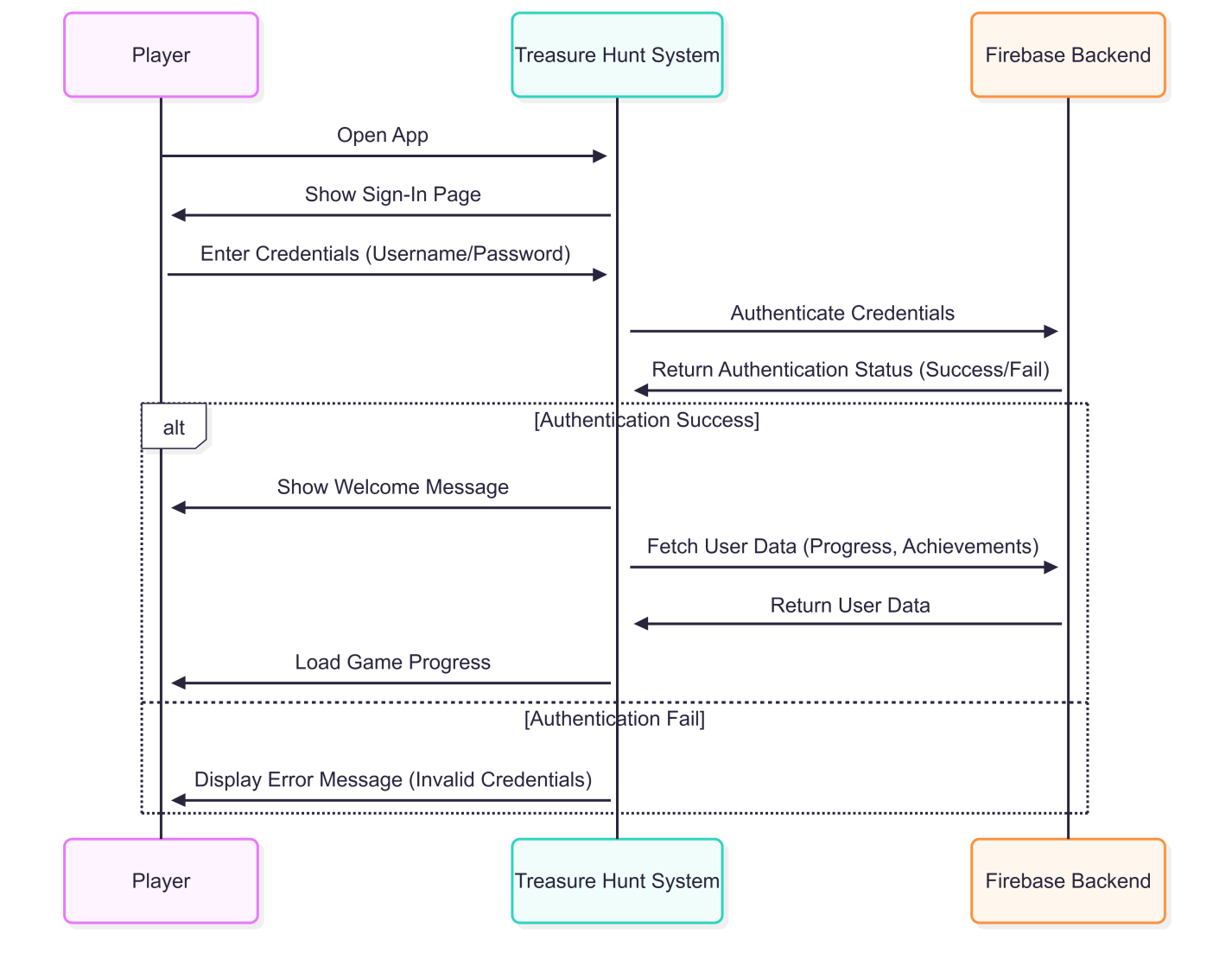


Fig 1.5 system sequence diagram for User sign-in

**AR Navigation (Treasure Hunt) Sequence Diagram**

Fig. 1.6 shows the process sequence for Login Account

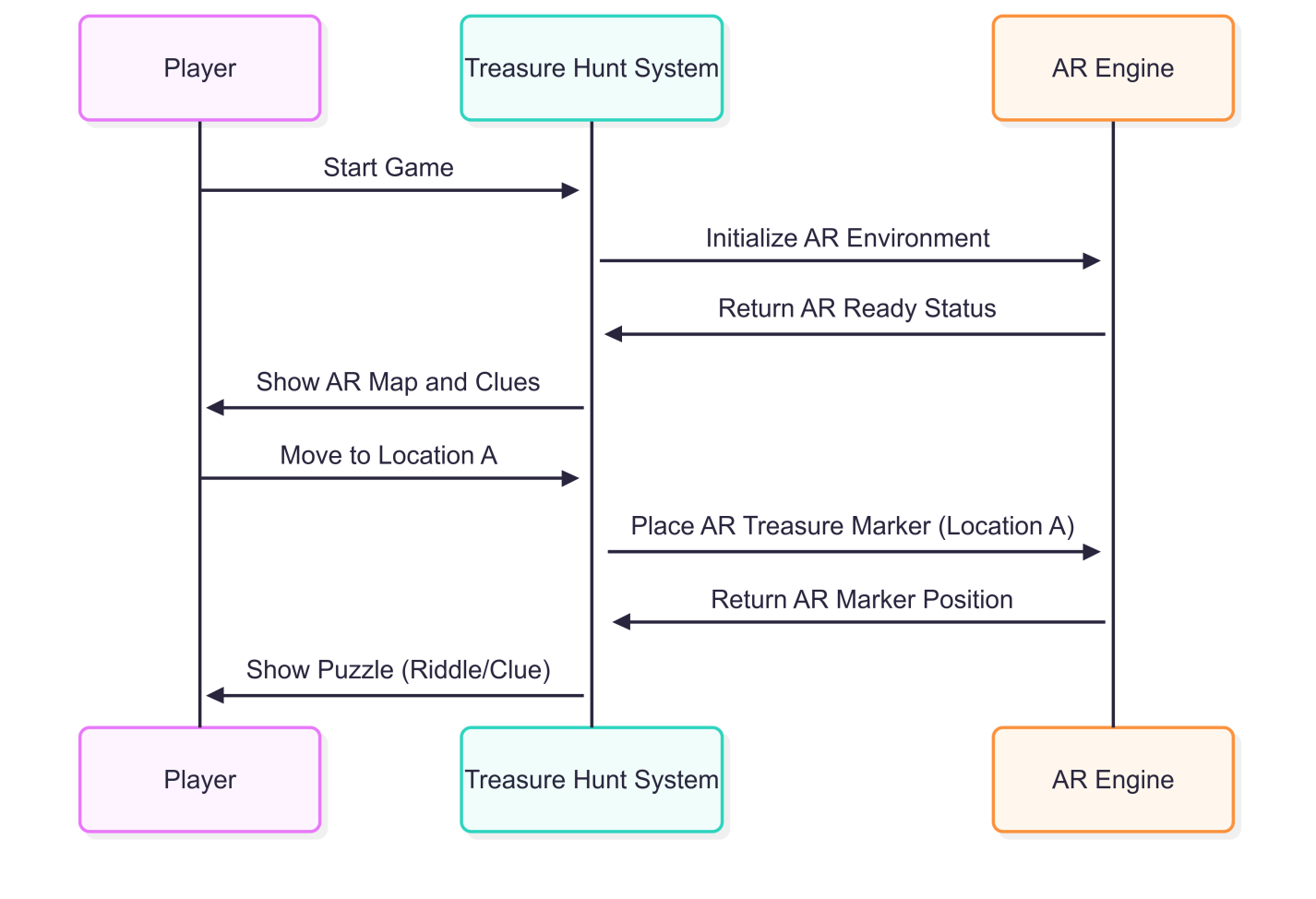


Fig 1.6 system sequence diagram for AR navigation

**Puzzle Solving Sequence Diagram**

Fig. 1.7 shows the process sequence for Reset Password

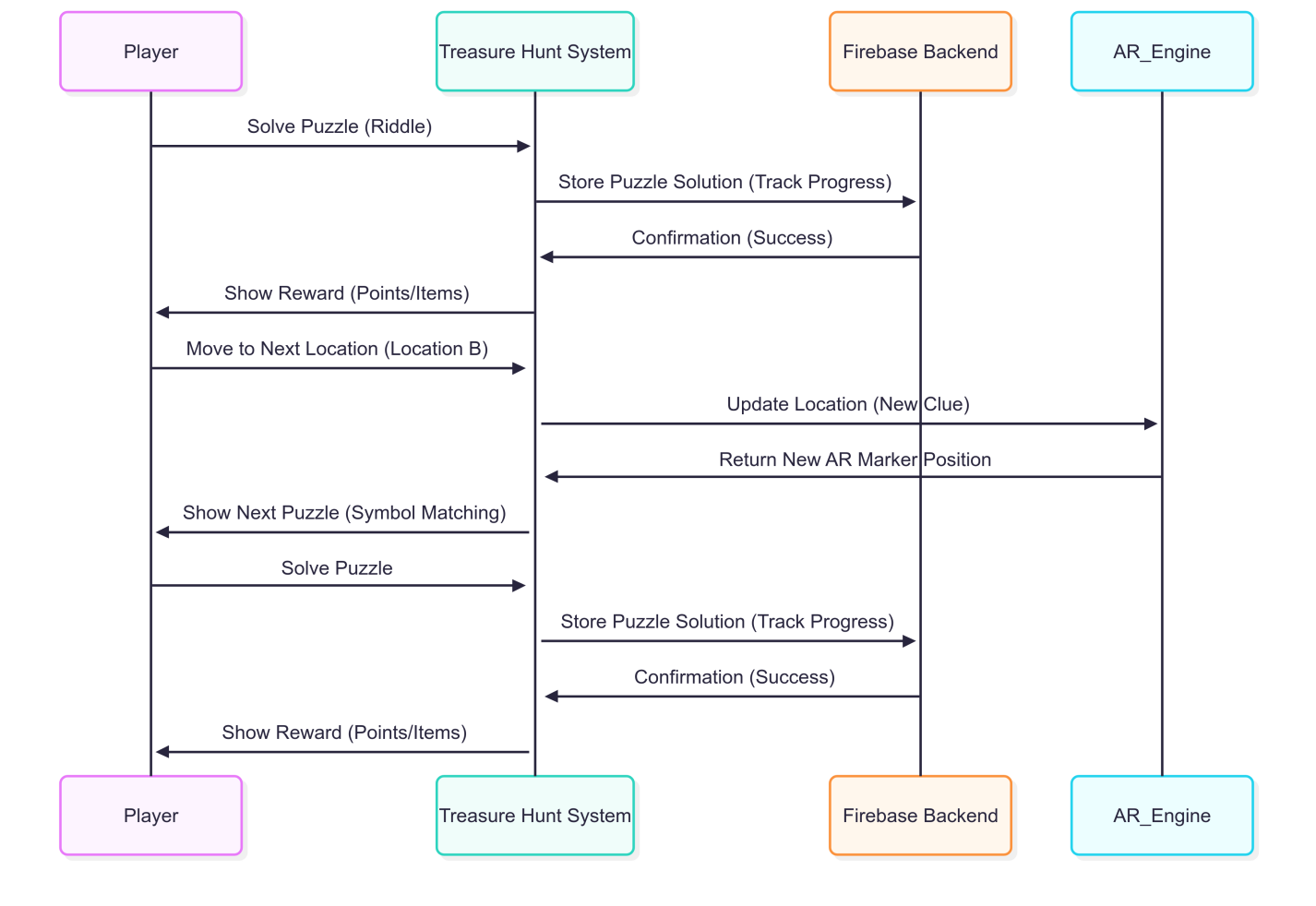


Fig 1.7 system sequence diagram for puzzle solving

**Treasure Collection Sequence Diagram**

Fig. 1.8 shows the process sequence for Update Profile

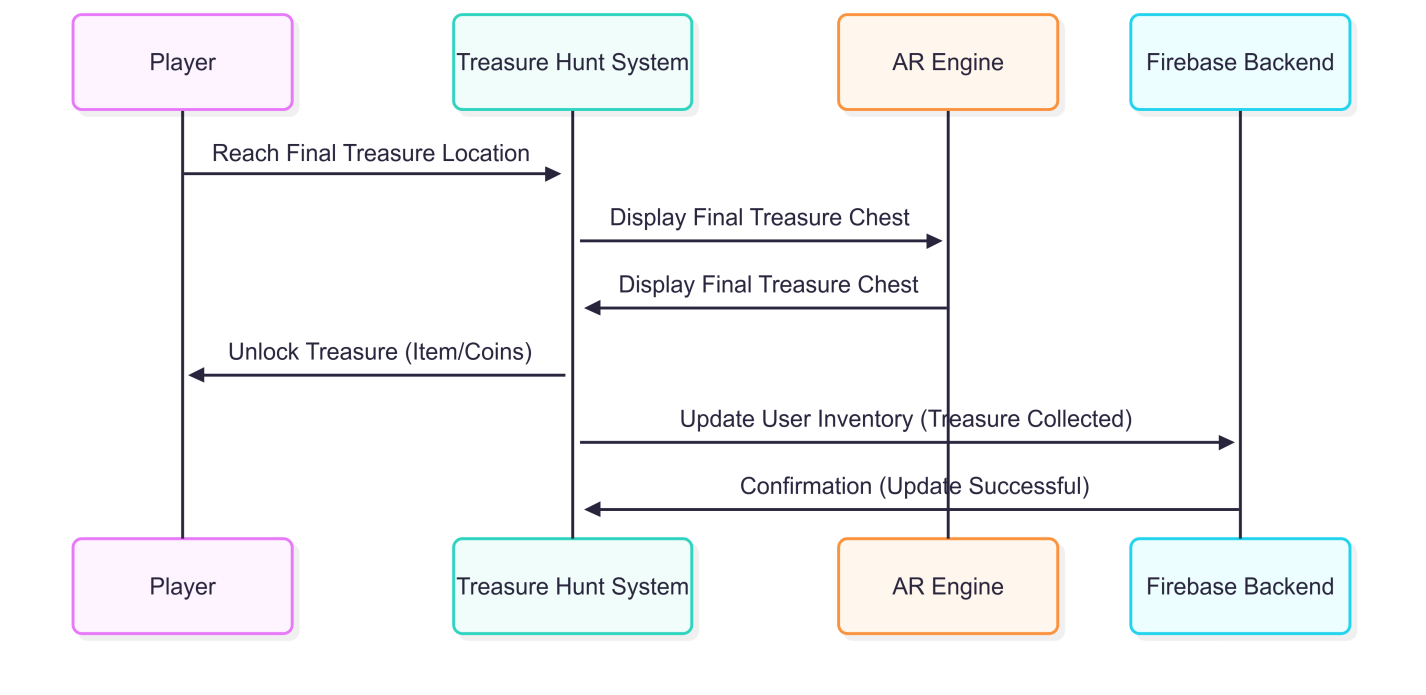


Fig 1.8 system sequence diagram for treausre collection

**Leaderboard and Reward System Sequence Diagram**

Fig. 1.9 shows the process sequence for Start New Game

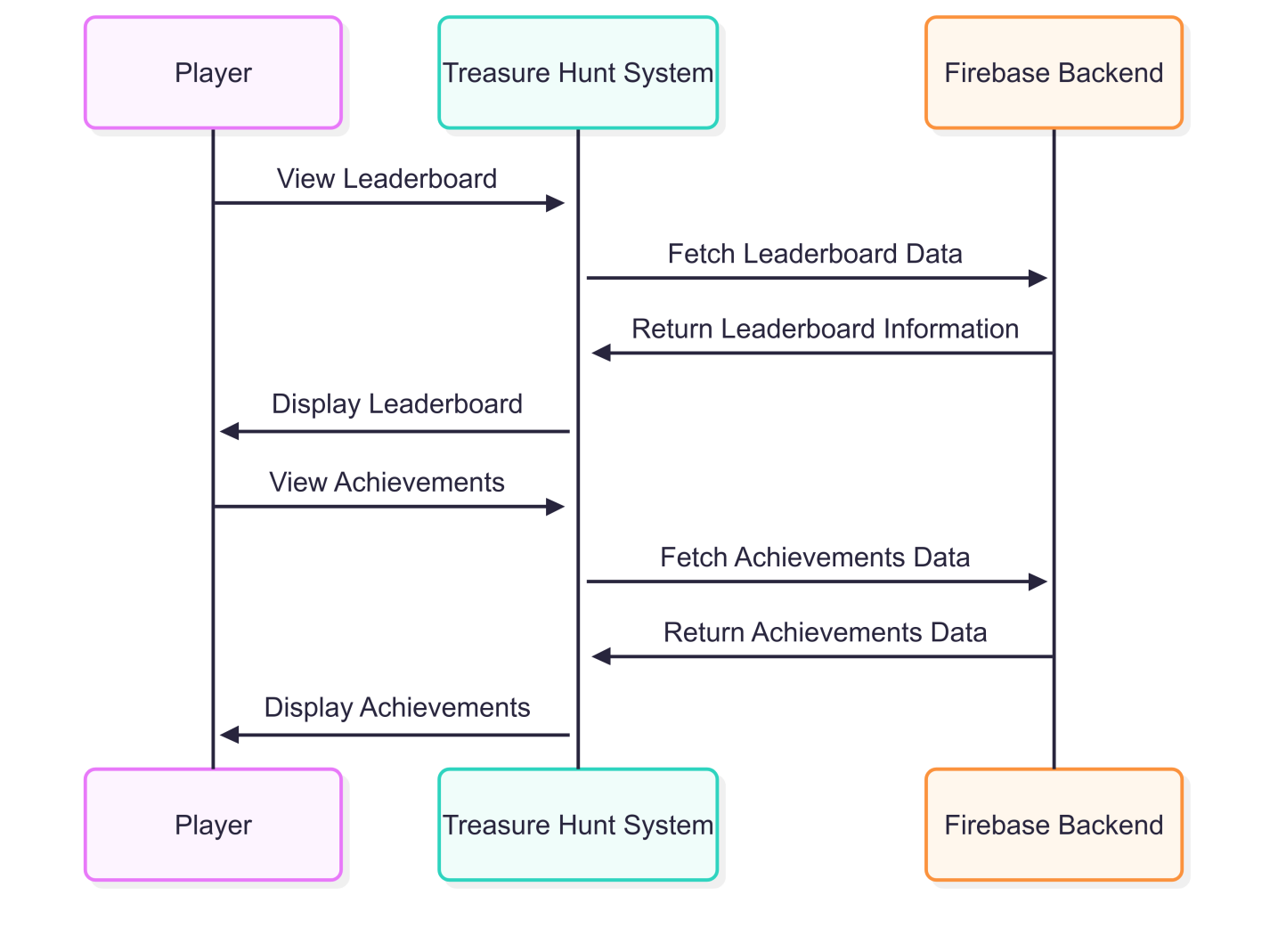


Fig 1.9 system sequence diagram for leaderboard and reward

**JSON Tree**

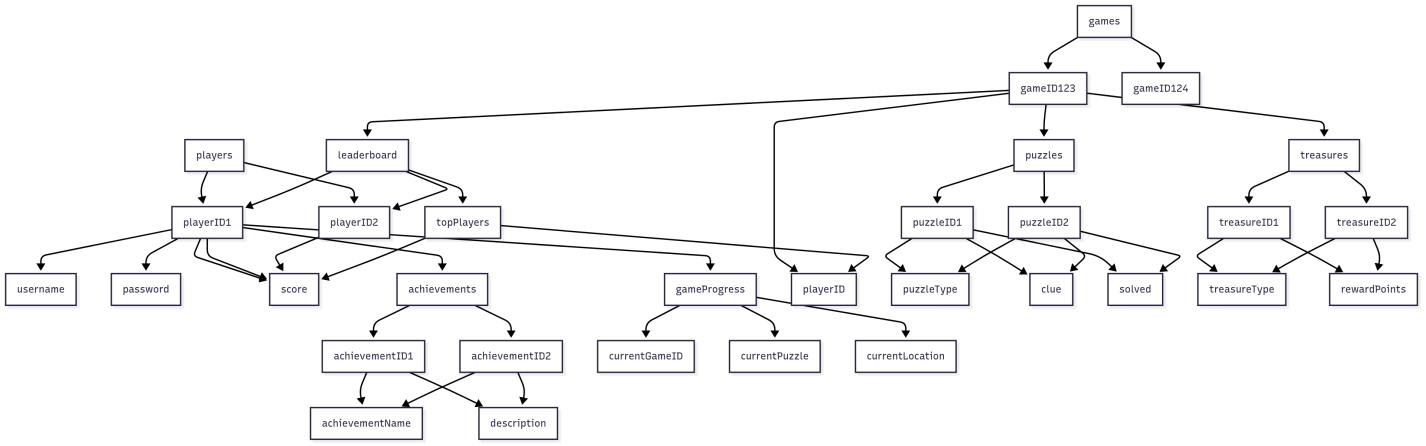
****

Fig 1.10 system sequence diagram for JSON tree

**Data design**

The data design of *Treasure Hunt Adventures* focuses on organizing and managing the game’s dynamic information such as user data, AR elements, puzzles, rewards, and gameplay progress. The system uses **Firebase** as the primary backend, utilizing its Authentication, FirestoreDatabase services to handle real-time data processing and synchronization across devices.

**4.1 Data Organization and Entities**

The system transforms the information domain (players, treasures, puzzles, etc.) into well-structured data entities, each with defined attributes and relationships:

* **Player Entity**: Contains attributes like PlayerID, Username, Email, Score, Avatar, AchievementsUnlocked, and GameProgress.
* **Treasure Entity**: Includes TreasureID, Location (GPS coordinates), Type, IsCollected, and AssociatedReward.
* **Puzzle Entity**: Stores PuzzleID, PuzzleType, Difficulty, ClueText, Solution, and Status.
* **Leaderboard Entry**: Includes PlayerID, Rank, Score, Timestamp.
* **Achievement Entity**: Contains AchievementID, Title, Description, IsUnlocked, and UnlockDate.

These entities are stored in a **NoSQL structure** within Firebase Firestore, allowing flexible and scalable access patterns suitable for mobile AR applications.

**4.2 Data Storage Technologies**

**4.3 Data Processing and Flow**

During gameplay, data is continuously read and written between the client (Unity AR App) and Firebase. For example:

* When a player collects a treasure, the system updates the Treasure Entity and unlocks a corresponding Achievement, which is then pushed to the Leaderboard.
* Puzzle-solving triggers a validation process, and upon success, the Player Entity is updated with reward information.
* Sync operations ensure all gameplay data is persisted, even when switching devices.

The data design prioritizes real-time responsiveness, data integrity, and minimal latency, all critical for ensuring a seamless AR gaming experience.

|  |  |
| --- | --- |
| Data Storage | Purpose |
| Firebase Firestore | Stores player profiles, puzzle data, treasure status, rewards, and leaderboard records in real time. |
| Firebase Authentication | Manages user accounts, login credentials, and secure access control. |
| Firebase Cloud Storage | Used for storing heavy media files like AR assets, puzzle images, and 3D models. |
| Local Device Storage | Temporarily holds session data and offline progress, which is synced when online. |

**Data dictionary**

This section lists the major objects (entities) used in the system, along with their attributes, data types, and brief descriptions, followed by relevant methods and parameters where applicable. The dictionary is alphabetically ordered for clarity.

**Achievement**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| achievementID | String | Unique identifier for each achievement. |
| title | String | Name of the achievement (e.g., “Puzzle Master”). |
| description | String | Description of how the achievement is earned. |
| isUnlocked | Boolean | Indicates whether the player has unlocked the achievement. |
| unlockDate | Timestamp | The date/time the achievement was earned. |

**LeaderboardEntry**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| playerID | String | Unique reference to the Player. |
| rank | Integer | The position of the player in the leaderboard. |
| score | Integer | Total score of the player. |
| timestamp | Timestamp | Time when the score was recorded. |

**Player**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| playerID | String | Unique identifier for the player. |
| username | String | Player's display name. |
| email | String | Registered email address. |
| avatar | String (URL) | Link to the profile picture/avatar. |
| score | Integer | Cumulative score earned. |
| achievementsUnlocked | List | List of unlocked achievements. |
| gameProgress | JSON/Object | Snapshot of last known progress. |

**Puzzle**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| puzzleID | String | Unique identifier for each puzzle. |
| puzzleType | String | Type of puzzle (e.g., riddle, memory). |
| difficulty | String | Difficulty level (easy, medium, hard). |
| clueText | String | The hint or description for the puzzle. |
| solution | String | Correct answer to the puzzle. |
| status | String | Solved / Unsolved status. |

**Treasure**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| treasureID | String | Unique ID for the treasure. |
| location | GeoPoint (lat, long) | GPS coordinates of the treasure. |
| type | String | Type of treasure (chest, scroll, gem). |
| isCollected | Boolean | Whether it has been collected by the player. |
| associatedReward | String | Reward/achievement unlocked upon collection. |

|  |  |  |
| --- | --- | --- |
| **Method** | **Parameters** | **Description** |
| spawn() | location: GeoPoint | Places treasure in AR world. |
| scollect() | – | Marks the treasure as collected and triggers reward. |

**Algorithm & Implementation**

This section outlines the core functionality of the system’s major object-oriented components using Procedural Description Language (PDL) or pseudocode. It reflects the internal behaviour of objects like Player, Puzzle, Treasure, and others, based on their responsibilities.

**Class: Player**

**Function:** login(email, password)

IF Firebase.Authenticate(email, password) == SUCCESS THEN

Load player profile from Firestore

Set session status to ACTIVE

ELSE

Display error message to user

END IF

**Function: updateProfile(username, avatar)**

pseudocode

CopyEdit

Update local user data with new username and avatar

Push updated data to Firebase Firestore

Notify user of success or failure

**Function: logout()**

pseudocode

CopyEdit

Clear local session data

Sign out from Firebase Auth

Redirect user to Login screen

**Class: Puzzle**

**Function: checkAnswer(input)**

pseudocode

CopyEdit

IF input == solution THEN

status = "Solved"

RETURN true

ELSE

RETURN false

END IF

**Function: generatePuzzle()**

pseudocode

CopyEdit

SELECT random puzzle from Firebase WHERE difficulty == current level

SET currentPuzzle = selected puzzle

DISPLAY clueText to user

**Class: Treasure**

**Function: spawn(location)**

pseudocode

CopyEdit

IF ARSurfaceFound(location) THEN

Instantiate virtual treasure model at detected plane

Store treasure metadata locally

ELSE

Prompt player to move or scan again

END IF

**Function: collect()**

pseudocode

CopyEdit

IF Player is within collection radius THEN

Mark treasure as collected

Trigger reward system

Sync treasure status with Firebase

ELSE

Show message: “Get closer to collect treasure”

END IF

**Class: Leaderboard**

**Function: updateScore(playerID, score)**

pseudocode

CopyEdit

FETCH current score from Firestore

ADD new score to existing score

UPDATE Firestore with new total

REFRESH leaderboard UI

**Function: getTopPlayers()**

pseudocode

CopyEdit

QUERY Firestore for top 10 players ORDER BY score DESC

RETURN list of usernames and scores

DISPLAY in leaderboard screen

**Class: Achievement**

**Function: unlock(achievementID)**

pseudocode

CopyEdit

IF achievementID NOT IN player.achievementsUnlocked THEN

ADD achievementID to unlocked list

SHOW notification to player

SAVE progress to Firebase

END IF

**Human interface design**

The *Treasure Hunt Adventures* system is designed with a user-centric interface that provides an intuitive and seamless experience, particularly for mobile users engaged in real-world exploration. The application interface follows mobile design best practices, with a clear layout, consistent navigation, minimal taps to access core features, and engaging feedback through visual cues, animations, and sound.

**User Interaction Overview**

Upon launching the application, the user is greeted with a clean and interactive login or sign-up screen powered by Firebase Authentication. After successful authentication, the user is navigated to the **Main Menu**, where they can access:

* **Start AR Hunt** – Initiates the augmented reality gameplay session.
* **Achievements** – View earned badges and progress stats.
* **Inventory** – Check collected treasures or puzzle items.
* **Leaderboard** – View top players and their ranks.
* **Settings** – Manage sound, difficulty, profile info, and logout.

Each button in the UI is clearly labeled with icons and text, making navigation simple even for first-time users.

**AR Gameplay Interface**

When the AR session starts, the system activates the device’s camera and overlays virtual content on the real environment using ARCore or ARKit. Players will see:

* **On-screen indicators** for detected planes.
* **Interactive treasure objects** placed on flat surfaces.
* A **puzzle popup** when reaching a treasure spot, allowing players to solve a challenge.

Touch-based interactions such as tap, drag, or double-tap are used to collect items or interact with the environment. Players receive immediate feedback through visual animations, sound effects, and real-time score updates.

**Feedback and Notifications**

The system provides constant feedback through:

* **Popup messages** for success/failure in puzzles.
* **Progress bars** showing game advancement.
* **Toasts and alerts** for actions like treasure collection or AR errors.
* **Achievement unlocks** displayed with badge animations.
* **Firebase sync confirmations** ensuring data is safely stored.

Additionally, session data such as score, collected treasures, and completed puzzles are visually reflected in the player’s profile and leaderboard in real-time.

**Screen images**

Display screenshots showing the interface from the user’s perspective. These can be hand-drawn, or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

**8.2 Screen objects and actions**

A discussion of screen objects and actions associated with those objects

A screen shot of a video game

AI-generated content may be incorrect.

A screen shot of a video game

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer screen

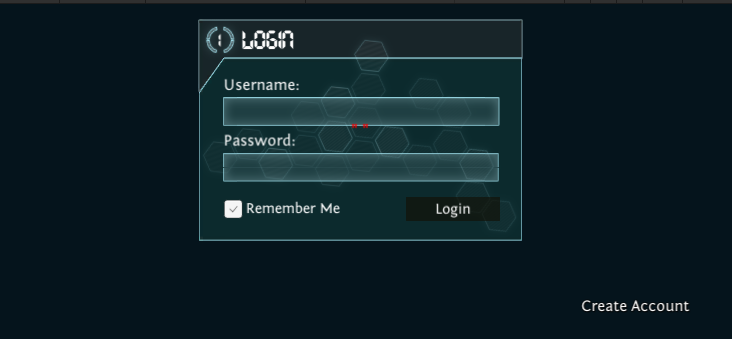
AI-generated content may be incorrect.

A screen shot of a message

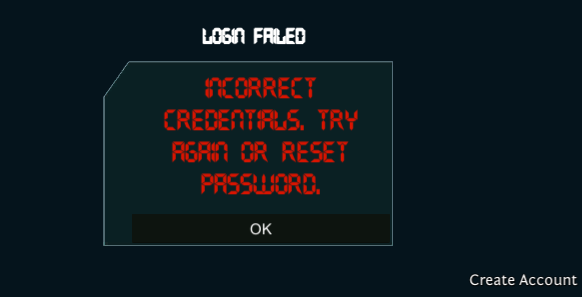
AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a video game

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Appendix I**

* How to design using UML (OOP): For guidance please follow the instructions mentioned in the link: http://agilemodeling.com/artifacts/
* How and when to design ER diagrams: For guidance please follow the instructions mentioned in the link:

<http://people.inf.elte.hu/nikovits/DB2/Ullman_The_Complete_Book.pdf>

* Data flow diagrams: For guidance please follow the instructions mentioned in the link and book:
  + http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm
  + Software Engineering –A Practitioner’s approach by Roger Pressman
* Architecture diagram: For guidance please follow the instructions mentioned in the link and book:
  + Ian Sommerville – Software Engineering 9th Edition– Chapter 6