CS 492 - Senior Design Project II

Bilkent University

Spring 2026

Detailed Design Report

Group T2525

Ege Ertem 22202433

Can Kütükoğlu 22202619

Sami Bora Akoğuz 22202184

Mehmet Rodi Aydoğdu 22201856

Cem Tarkan Tekcan 22201590

1. Introduction..... 3
1.1. Purpose of the System..... 3
1.2. Design Goals..... 3
1.3. Definitions, Acronyms, and Abbreviations..... 4
1.4. Overview..... 5
2. Current Software Architecture..... 6
2.1. Existing Solutions and Competitors..... 6
2.2. Architectural Limitations of the Current Paradigm..... 7
3. Proposed Software Architecture..... 7
3.1. Overview..... 7
3.2. Subsystem Decomposition..... 8
3.3. Hardware/Software Mapping..... 9
3.4. Persistent Data Management..... 9
3.5. Access Control and Security..... 10
3.6. Diagrams..... 11
4. Subsystem Services..... 14
4.1. User Management & Authentication Service..... 14
4.2. Tour Execution & State Management Service..... 17
4.3. AI Story Creation & Parsing Service..... 17
4.4. Social, Monetization, and Moderation Service..... 18
5. Test Cases..... 19
5.1. Functional Test Cases..... 19
5.2. Non-functional Test Cases..... 35
6. Consideration of Various Factors in Engineering Design..... 40
6.1. Engineering Design Factors..... 40
6.2. Constraints..... 43
6.3. Standards..... 44
7. Teamwork Details..... 44
7.1. Contributing and functioning effectively on the team..... 45
7.2. Helping creating a collaborative and inclusive environment..... 46
7.3. Taking lead role and sharing leadership on the team..... 47
8. Glossary..... 48
9. References..... 50

1. Introduction

1.1. Purpose of the System

The purpose of the Odyssey system is to transform urban exploration by combining the physical world with digital creativity through a mobile app. Traditional tourism and navigation apps usually rely on static content, manual route planning, or passive media like guidebooks and audio tours. Odyssey seeks to address this limitation by gamifying tourism and turning city guides into interactive experiences.

The core differentiator of the system is the integration of Large Language Models (LLMs) to generate dynamic, context-aware narratives and location-based puzzles in real-time. By validating a user's physical presence at designated Points of Interest (POIs) via GPS, Odyssey creates a unique hybrid of digital geocaching and guided tours. The system is designed to serve a diverse user base, ranging from casual tourists and "Normal Users" engaging in free trials to "Premium Users" and "Content Creators" who build and monetize custom routes.

1.2. Design Goals

Usability: The interface is designed for learnability and efficiency. First-time users must be able to navigate and start a tour within 60 seconds of logging in. For Content Creators, the system is designed to allow the creation of a 5-location "Story Mode" tour in under 15 minutes. The design also enforces strict input validation to ensure error prevention during tour creation.

Performance: The system architecture is optimized for low latency and smooth mobile experiences. The mobile client is designed to achieve a cold-start in under 3 seconds on a mid-range device with a 4.5G connection. Backend APIs are designed to process requests in 200ms or less. Furthermore, the Augmented Reality (AR) features are designed to render at a minimum of 24 frames per second to avoid jitter.

Reliability: The core backend services (authentication, tour browsing, payments) are designed to support a 99% uptime. To account for mobile connectivity issues, the design includes an "Offline Mode" that caches the current tour step and tracks location locally if the internet connection is lost. All user-facing write operations are designed to be strictly transactional to ensure data integrity.

Scalability: The persistent data management system is designed to handle immense future growth, specifically architected to support 100,000+ user-generated tours and 1,000,000+ reviews while maintaining query execution times under 1 second.

Modularity and Maintainability: The design strictly decouples the React Native frontend from the Django backend, allowing parallel development.

Security and Compliance: The system design enforces JWT-based authentication, secure social logins via OAuth 2.0, and data transmission encrypted via HTTPS (TLS 1.3). It is structurally designed to comply with GDPR and KVKK by minimizing stored location data and ensuring explicit consent.

1.3. Definitions, Acronyms, and Abbreviations

- **API (Application Programming Interface):** A set of protocols allowing communication between the mobile client, the backend, and third-party services.

- **AR (Augmented Reality):** Technology superimposing computer-generated images onto the real world, used for Odyssey's puzzle mechanics.
- **Cold-Start:** The time required for the application to load from a completely closed state to interactive.
- **GDPR / KVKK:** Legal frameworks governing the collection and processing of personal data (European Union and Turkey, respectively).
- **LLM (Large Language Model):** Deep learning algorithms used to dynamically generate "Story Mode" narratives and puzzles based on system prompts.
- **POI (Point of Interest):** A specific map location (landmark, museum) utilized as a puzzle or story waypoint.
- **React Native:** A cross-platform UI framework used to build the mobile client for both iOS and Android from a single codebase.
- **Django:** A high-level, open-source Python web framework that promotes rapid development and clean, pragmatic design.
- **Token (AI):** The basic unit of text processed by the LLM, representing a variable operational cost.
- **Token (Game Currency):** Virtual in-app currency used to unlock tours or purchase hints.

1.4. Overview

The remainder of this document outlines the detailed system design for Odyssey. **Section 2** analyzes the current software architecture and competitor limitations. **Section 3** details the proposed software architecture, encompassing subsystem decomposition, hardware/software mapping, persistent data management, and access control. **Section 4** describes the specific subsystem services and their interactions. **Section 5** provides

comprehensive integration test cases for validating system functionality. **Section 6** discusses the consideration of various engineering factors (health, safety, economics, etc.) on the design. Finally, **Section 7** outlines the teamwork and leadership dynamics that facilitated the project's execution.

2. Current Software Architecture

2.1. Existing Solutions and Competitors

The current market for interactive tourism is populated by applications that attempt to blend sightseeing with gamification. Platforms like CityFans, Explorial, and Questo allow users to solve puzzles and unlock stories while touring landmarks, while Geocaching has demonstrated a massive demand for location-based puzzle adventures [1,2,3]. Aside from these digital solutions, many users still rely on traditional, non-interactive methods such as physical guidebooks, static maps, or audio tours to navigate new cities.

However, these applications only support manually generated content or static solutions. Applications like Questo depend only on a community of content creators to generate and decide on new routes, which limits scalability and consistency [2]. Moreover, traditional guides also stick with static content which lacks dynamic interaction or personalization. Briefly, no widely renown solution exists to automatically generate dynamic and fresh content without static human intervention, which is the exact need that Odyssey is aimed to fulfill.

2.2. Architectural Limitations of the Current Paradigm

The primary architectural flaw in the current software landscape is the content creation bottleneck. Because existing interactive tour applications rely on static human intervention to create and update tours, they face severe limitations in both scalability and consistency. If a user visits a less popular city, or if community creators become inactive, the platform cannot provide fresh experiences.

In summary, there is currently no widely renowned architectural solution in the market designed to automatically generate dynamic, personalized, and fresh tourism content without static human intervention. This architectural gap – the lack of an automated, real-time content generation engine – is the exact limitation that Odyssey's system design addresses.

3. Proposed Software Architecture

3.1. Overview

Odyssey is a cross-platform mobile application designed to gamify tourism by merging physical exploration with digital storytelling. The system transforms standard city tours into interactive experiences using three primary modes: Puzzle Tours, which challenge users with trivia, gyroscope mechanics, and Augmented Reality (AR); Story Tours, which provide relaxed, narrative-driven routes generated by AI or the community; and Hybrid Tours, which combine both storytelling and interactive challenges.

The system architecture is divided into a mobile client layer and a backend layer. The client Layer is a mobile application that manages local storage, GPS tracking to facilitate

real-time user interaction and other required sensors/interfaces we will require throughout the project. Backend layer is used for hosting essential services including user authentication, social services, and the tour manager. Among the modules of the system, AI Story Creation Module holds great importance as it utilizes a structured previously created prompt to communicate with Large Language Models (LLMs). This allows the system to generate dynamic narratives and puzzles for users dynamically.

There are 4 different roles for managing access in the system: normal users (free trial access), premium users (subscription-based access to exclusive features), content creators, and admins. Odyssey aims to produce a safe, community-driven platform that changes the paradigm of travelling through its innovations, by this system design..

3.2. Subsystem Decomposition

React Native frontend and the backend are separated with clear boundaries to provide modularity and maintainability.. The system is separated into the following core subsystems:

- **Client Layer (Mobile App):** React Native and TypeScript is used in the process of development. This subsystem provides the user interface across iOS and Android devices. Moreover, it handles local storage, GPS tracking for real-time interaction, and interfaces with device sensors for gameplay mechanics.
- **Backend Layer (Server):** Django web framework is used in the process of development, this subsystem contains the core API. Essential services are hosted, including user authentication, social services (profiles, friends), and the tour manager. Moreover, Docker containerizes the backend for a streamlined deployment process.

- **AI Story Creation Module:** The communication between the backend layer and the LLM is handled by this subsystem. It is used for parsing the user's city and theme selections, constructing a prompt, validating the returned JSON string, and persisting the generated tour data into the database dynamically.

3.3. Hardware/Software Mapping

Software capabilities are mapped to the specific hardware constraints of modern mobile devices by the physical architecture of the system.

- **Location Services:** Mobile device's built-in GPS sensor is used for getting real-time location data. Although the standard 5-10 meter accuracy limitation of mobile GPS, the system maps this data against third-party API points of interest, in order to provide smooth gameplay.
- **Augmented Reality (AR):** The AR puzzle provides a map directly to the device's camera and AR frameworks (ARCore for Android 7.0+ and ARKit for iOS 11+). Moreover, the software is created in a way to degrade to a 2D interface gracefully if the hardware does not have this capability.
- **Gyroscope Mechanics:** Device orientation based puzzles rely on the physical gyroscope sensor. The software performs a hardware capability check during startup.
- **Offline Fallback Mapping:** The software enters an "offline mode," and caches the current step in local storage and relies solely on offline GPS pings to allow the user to complete their current objective, if the device loses active internet connection.

3.4. Persistent Data Management

Data management is handled by PostgreSQL, which acts as the primary relational database.

- **Scalability Design:** The database schema is designed to handle 100,000+ user-generated tours and 1,000,000+ reviews. Furthermore, it is designed to ensure that complex relational queries execute in under 1 second.
- **Data Integrity:** All user-facing write operations are strictly transactional.
- **Data Privacy:** To comply with KVKK and GDPR, the persistent storage of location history is minimized, encrypted, and anonymized or deleted after a set retention period.

3.5. Access Control and Security

Access control is provided through a role system and modern security protocols.

- **Role-Based Access:** The system enforces four distinct roles: **Normal Users** (base access), **Premium Users** (paid exclusive features), **Content Creators** (tour publishing privileges), and **Super Admins** (system moderation).
- **Authentication Flow:** User authentication provides secure credentials and OAuth 2.0 for social logins in order to decrease the direct handling of sensitive passwords. Session management is handled via JWT access and refresh tokens, stored securely on the device with secure storage framework.
- **Network Security:** All API communication between the React Native client and the Django backend is encrypted via HTTPS.
- **Anti-Cheat Measures:** To protect the token economy and game integrity, calculations regarding XP gain, Token distribution, and Tour Completion are strictly executed on the backend server.

3.6. Diagrams

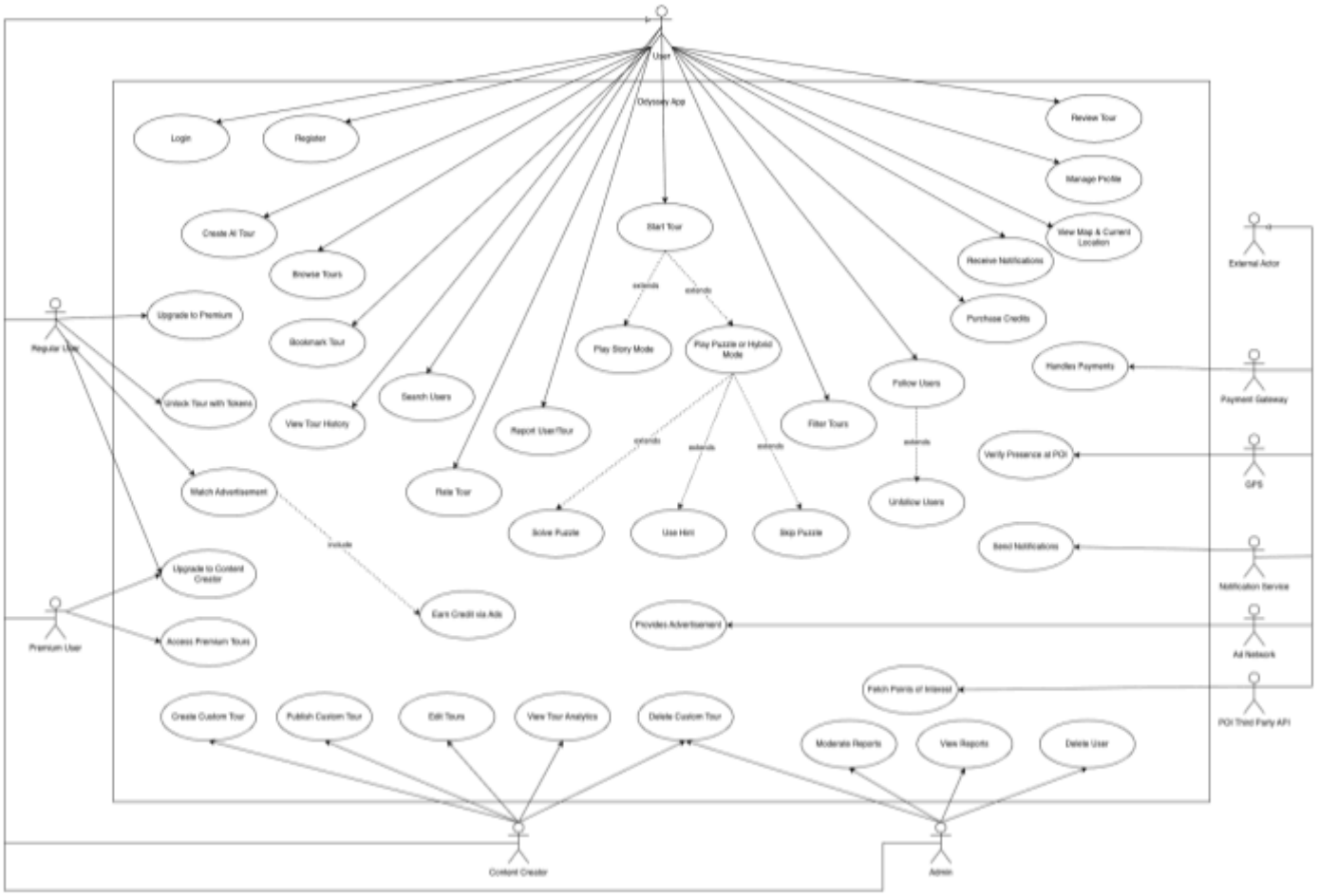

Fig 1. Use Case Diagram

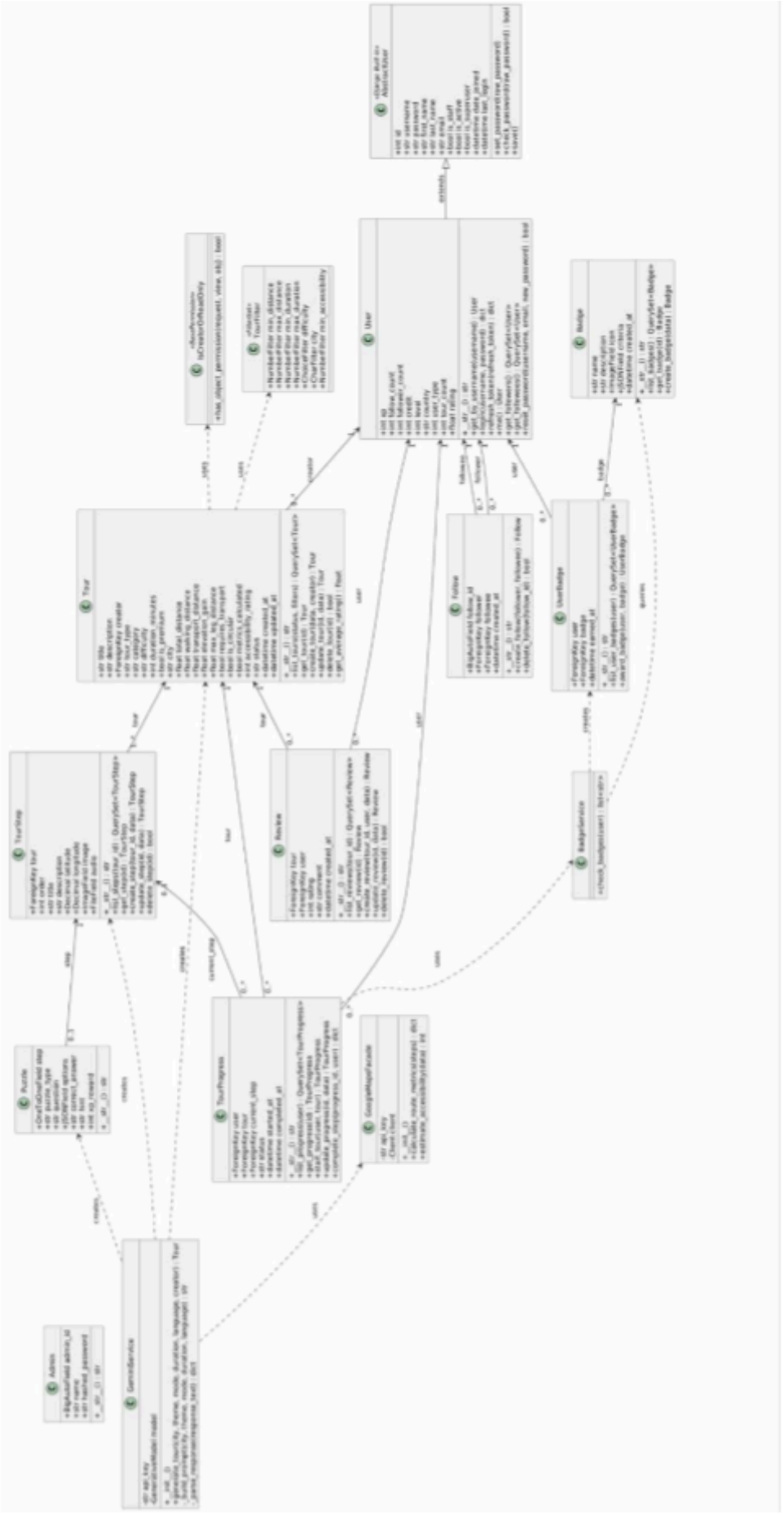

Fig. 2 Class diagram of the django backend

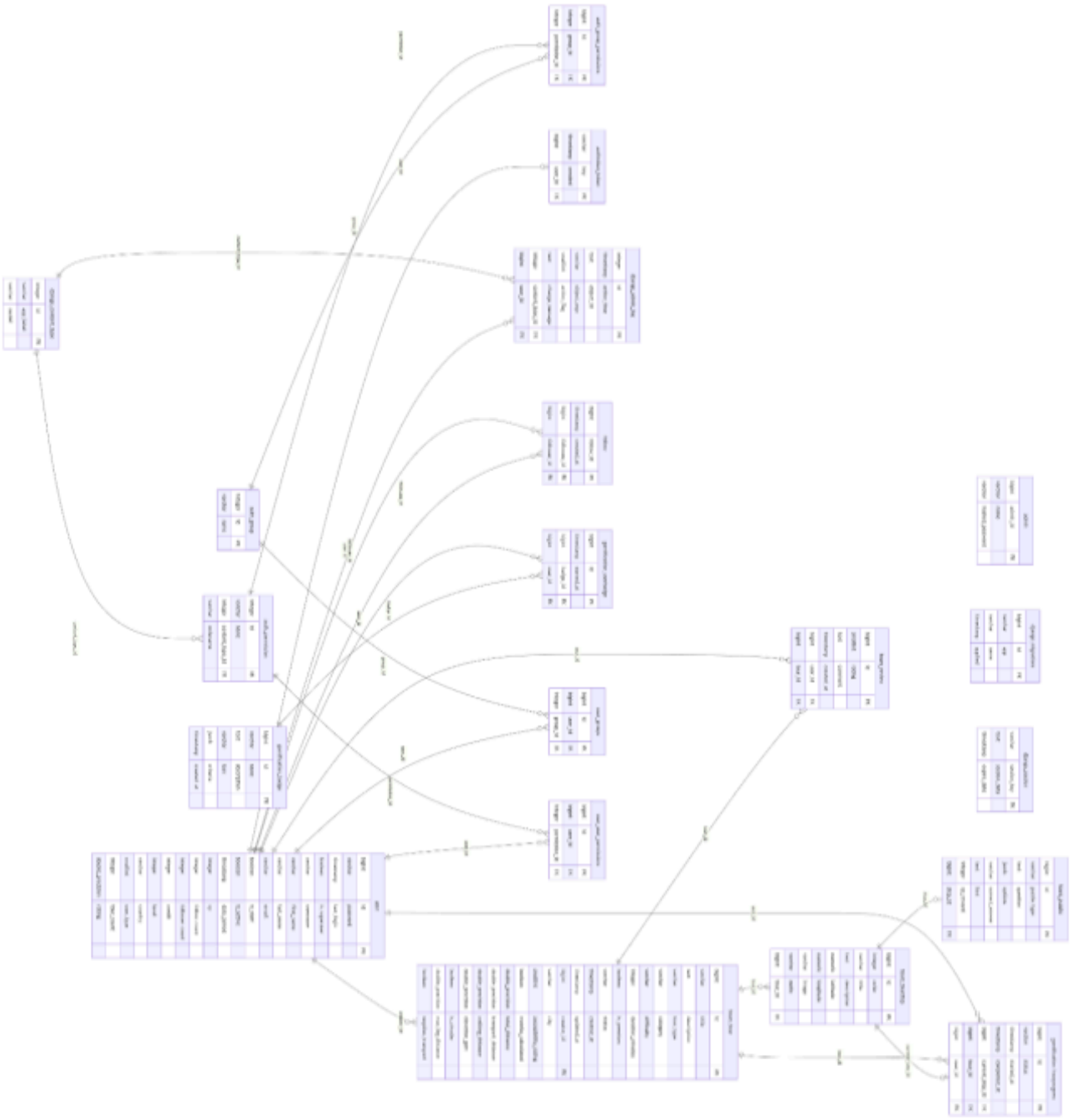

Fig. 3 E/R Diagram

Fig. 4 State diagram of tour and tour progress

4. Subsystem Services

4.1. User Management & Authentication Service

This service provides secure access to the application, managing user states and token distribution. The authentication flow process across several phases is like the following:

- **Input & Validation (Phase 1):** The React Native login.tsx UI takes user credentials and performs initial client-side validation to be sure required fields are filled.
- **Request Preparation & Transmission (Phases 2 & 3):** The frontend API Client configures an HTTP request with a standard 30-second timeout and calls the login endpoint.

- **Server Authentication (Phases 4 & 5):** The Django backend takes a `POST` request at the `/api/users/login/` endpoint. It validates the credentials against the PostgreSQL database. When it receives a successful response (HTTP 200 OK), the server creates JSON Web Tokens (JWT) for both access and refresh. An HTTP 400 Bad Request is returned if the credentials fail.
- **Secure Token Storage (Phase 7):** The client application takes the tokens and securely persists them on the device by Expo's `SecureStore` (`setItem('accessToken', access)`).

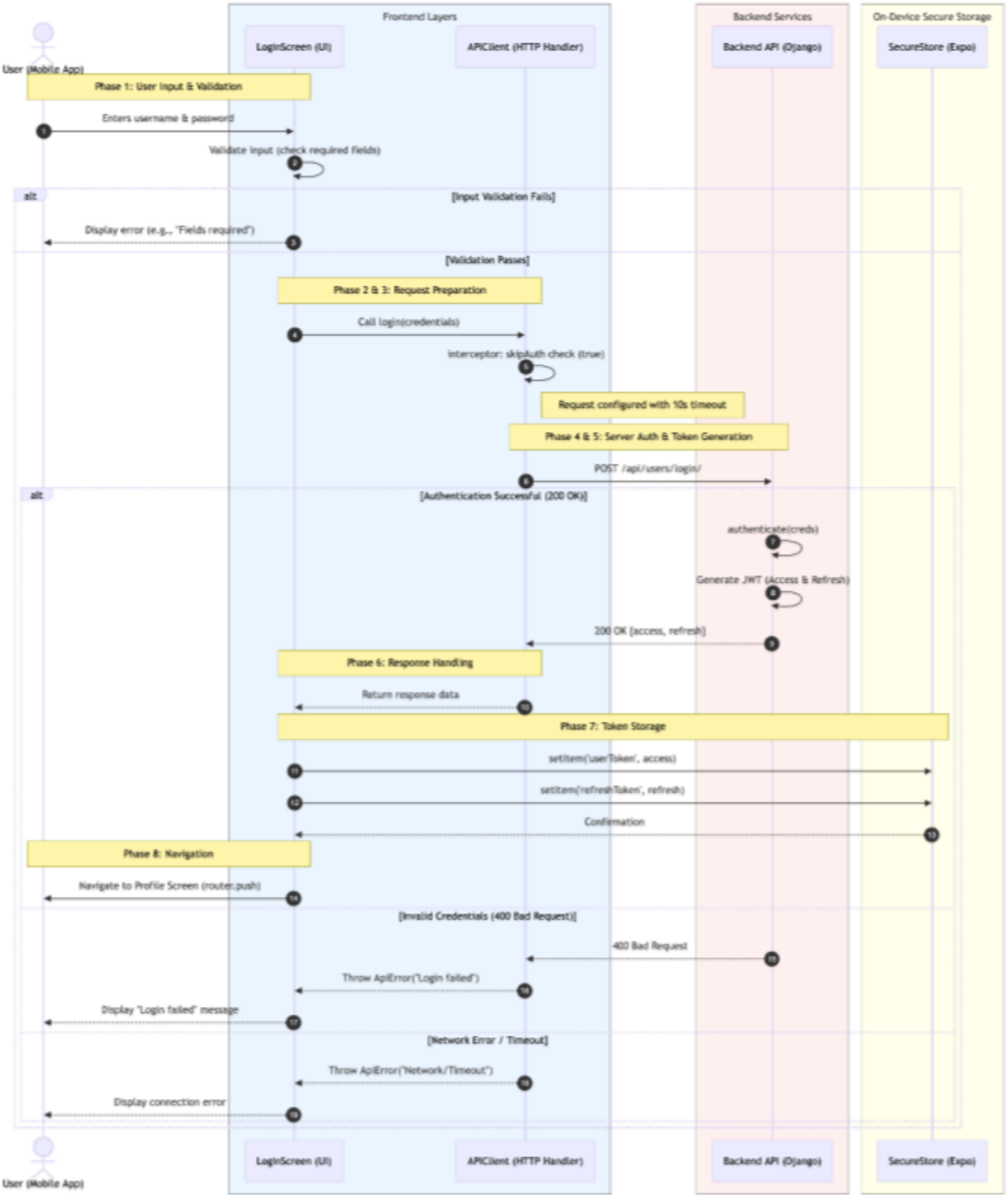

Fig. 1 Sequence diagram of user login process

4.2. Tour Execution & State Management Service

This core gameplay service manages a user's progress through a generated tour. It acts as a state machine that enforces physical verification before unlocking content:

- **Initialization:** When a user starts a tour, the system creates a `tour_progress` record in the database, setting the status to "IN_PROGRESS" and queuing the first location step.
- **Step Verification & Progression:** As the user physically navigates, the mobile client sends GPS data with `/api/tour-progress/{id}/complete-step/`. The backend verifies the physical presence at the current POI. Upon successful verification and puzzle completion, the service fetches the next step, and awards XP to the user.
- **Completion and Badge Evaluation:** When the user completes a step and that step is the final step, the `tour_progress.status` is updated to "COMPLETED". The service then evaluates the user's profile against defined badge criteria and creates a `badge_user` record if a new milestone is reached.
- **Abortion:** If a user manually quits or fails to finish, the state transitions to an "Aborted tour" with an "INCOMPLETE" status.

4.3. AI Story Creation & Parsing Service

- **Prompt Construction:** The backend service receives the user's selected city, theme, and category. It injects these variables into a strictly formatted, pre-created prompt.
- **LLM Communication & Validation:** The service sends this prompt to the external Large Language Model API. The prompt explicitly requires the LLM to return valid JSON containing real-world locations and narrative text.

- **Data Persistence:** Upon receiving the response, the backend parses the JSON string. It validates the output formats and immediately persists the valid POIs, story segments, and puzzle parameters into the PostgreSQL database , making the tour instantly available to the mobile client.

4.4. Social, Monetization, and Moderation Service

- **Profile & Social Graph:** Manages user search, following/follower relationships, and fetching dynamic profile statistics (e.g., total completed tours, badges earned).
- **Token Economy API:** Handles the addition of in-game tokens using real-world currency (via Apple StoreKit / Google Play Billing) or watching ads, and the deduction of tokens when users unlock Premium Tours or purchase puzzle hints.
- **Content Moderation Queue:** Processes user reports regarding unsafe or misleading tour steps. The service logs the report type, timestamp, and reporter ID, flagging the specific tour for admin review and potentially hiding it from search results if a threshold is exceeded.

5. Test Cases

5.1. Functional Test Cases

Test ID	TC-01	Category	Functional	Severity	Critical
Objective	This test case is to verify that a user can create a new account through the mobile application.				
Steps	1. Open the Odyssey app and press the "Register" button in the profile tab. 2. Enter a unique string in the "Username" field. 3. Enter a valid email address in the "E-Mail" field. 4. Enter a password in the "Password" field. 5. Press the “Submit” button to submit and create the account.				
Expected	The system creates the user, returns a success state, and automatically directs the user to the Login screen.				
Date-Result					

Test ID	TC-02	Category	Functional	Severity	Major
Objective	This test case is to verify that the system does not accept attempts for registration with an existing e-mail or username.				
Steps	1. Open the app and press "Register". 2. Enter a "Username" and "Mail" that are already registered in the database. 3. Enter a password and submit.				
Expected	The app does not fail or crash. The UI displays an error message indicating that the username or email is already taken.				
Date-Result					

Test ID	TC-03	Category	Functional	Severity	Critical
Objective	This test case is to verify that a created user can access their account with their own valid credentials.				
Steps	1. Open the app and press "Login". 2. Enter a valid username registered in the Database. 3. Enter the correct corresponding password. 4. Press the "Login" button.				
Expected	The system authenticates the user, generates the necessary session tokens, and navigates the user to their main Profile tab.				
Date-Result					

Test ID	TC-04	Category	Functional	Severity	Major
Objective	This test case is to verify that the “Forgot Password” flow triggers a recovery modal correctly and creates and sends a backend e-mail request.				
Steps	1. On the Login screen, press the "Forgot Password" button. 2. Verify that the password recovery modal appears. 3. Enter a valid registered "Mail" and "Username" into the corresponding modal fields. 4. Press the "Send Recovery Email" button.				
Expected	The modal closes and shows a success message even if the email does not exist in the database. The backend correctly creates and sends a password reset email to the provided address if that e-mail exists in the Database.				
Date-Result					

Test ID	TC-05	Category	Functional	Severity	Critical
Objective	This test case is to verify that the Profile page fetches and shows the user's details and data metrics accurately.				
Steps	1. Login to the application. 2. Navigate to the Profile page. 3. Observe and check the displayed metrics.				
Expected	The page correctly shows the user's selected avatar, username, Badge count, Tour count, XP, Following count, and Followers count				
Date-Result					

Test ID	TC-06	Category	Usability	Severity	Minor
Objective	This test case is to verify that the static metrics in the Profile page do not have any touch features				
Steps	1. Navigate to the Profile page. 2. Tap on the "XP" display area. 3. Tap on the "Tour Count" display area.				
Expected	The UI remains static. No modals appear, and no navigation occurs, confirming these elements are strictly read-only displays.				
Date-Result					

Test ID	TC-07	Category	Functional	Severity	Major
Objective	This test case is to verify that tapping the touchable Badge metric navigates the user to the dedicated Badges view.				
Steps	1. Navigate to the Profile page. 2. Tap on the "Badge count" UI.				
Expected	The application successfully navigates to a detailed list view displaying the user's earned badges				
Date-Result					

Test ID	TC-08	Category	Functional	Severity	Major
Objective	This test case is to verify that pressing the Followers/Following metrics navigates the user to the social list format view successfully.				
Steps	1. Navigate to the Profile page. 2. Press the "Followers" count. 3. Navigate back, then press the "Following" count.				
Expected	The application navigates the user to the dedicated views listing their respective followers and followed users successfully.				
Date-Result					

Test ID	TC-09	Category	Functional	Severity	Major
Objective	This test case is to verify that the “Add Friend” modal functionality and the user search query.				
Steps	1. Navigate to the Profile tab. 2. Press the "Add Friend" button. 3. Confirm whether the modal pops up. 4. Enter a known existing username into the search field and find the username on the list.				
Expected	The modal appears without errors. The search query successfully contacts the backend and returns the matching user profile in the modal list.				
Date-Result					

Test ID	TC-10	Category	Functional	Severity	Major
Objective	This test case is to verify that a Normal User can see their interacted tours historically on their profile.				
Steps	1. Login with a Normal User role that has previously completed and archived tours. 2. Navigate to the Profile tab. 3. Scroll to the Tours section.				
Expected	The UI correctly displays lists or tabs for "Completed" and "Archived" tours. "Created" tours are strictly not visible for this role.				
Date-Result					

Test ID	TC-11	Category	Security / Functional	Severity	Major
Objective	This test case is to verify that the role-based access control of the UI ensures “Content Creators” see their authored tours.				
Steps	1. Login with an account with the "Content Creator" role. 2. Navigate to the Profile tab. 3. Scroll through the Tours section.				
Expected	In addition to "Completed" and "Archived" tours, the UI successfully exposes the "Created" tours section, displaying the routes authored by this specific user.				
Date-Result					

Test ID	TC-12	Category	Functional	Severity	Major
Objective	This test case is to verify that the users can browse successfully and search for specific tours within the application.				
Steps	1. Navigate to the Tours tab in the app. 2. Look at the list of available tours. 3. Press the search bar button and enter the name of an existing tour. 4. Press on the tour card from the search results to see its details				
Expected	The search accurately filters the list of tours. Pressing a tour opens the detailed view, showing its description, difficulty, and duration successfully and accurately				
Date-Result					

Test ID	TC-13	Category	Functional	Severity	Critical
Objective	This test case is to verify that starting a tour successfully transitions the user to the active map and gameplay state.				
Steps	1. Navigate to a tour's detail page. 2. Press the "Start Tour" button. 3. Grant location permissions if prompted.				
Expected	The application transitions to the active navigation screen. The map initializes, centers on the user's current location, and displays the routing/POIs for the first step of the tour				
Date-Result					

Test ID	TC-14	Category	Security / Functional	Severity	Major
Objective	This test case is to verify that Normal Users are restricted to AI Tour Creation only.				
Steps	1. Log in with a Normal User account. 2. Tap the + (Create Tour) button in the navigation menu. 3. Observe the creation options presented.				
Expected	The UI displays the "AI-Powered Creation" option as active and accessible, while the "Personal Creation" option is distinctly greyed out and disabled				
Date-Result					

Test ID	TC-15	Category	Security / Functional	Severity	Major
Objective	This test case is to verify that the Content Creators have full access to both creation modes.				
Steps	1. Log in as a Content Creator. 2. Tap the + (Create Tour) button. 3. Observe the creation options.				
Expected	Both the "AI-Powered Creation" and "Personal Creation" buttons are fully active and pressable.				
Date-Result					

Test ID	TC-16	Category	Functional	Severity	Major
Objective	This test case is to verify that the AI Tour form accepts and constraints the user parameters correctly.				
Steps	1. Select "AI-Powered Creation". 2. Enter a "City" and "Tour Theme". 3. Ensure the default mode is set to "Hybrid". 4. Adjust the "Estimated Duration" using the + and - buttons (default 60 mins). 5. Select a language preference (e.g., en, es, tr, fr, de, it). 6. Enter optional additional details and submit.				
Expected	The duration increments/decrements strictly in 15-minute chunks. The form submits successfully and starts the AI generation process.				
Date-Result					

Test ID	TC-17	Category	Functional	Severity	Major
Objective	This test case is to verify that the initial metadata configuration is handled for the manual Personal Tour creation.				
Steps	1. Select "Personal Creation". 2. Enter a Title and Description. 3. Select a "Difficulty" level (e.g., Easy, Medium, Hard). 4. Select one or more "Categories" tags. 5. Press Continue.				
Expected	The application saves the metadata and transitions the user to the map-based POI selection screen.				
Date-Result					

Test ID	TC-18	Category	Functional	Severity	Critical
Objective	This test case is to verify that a Content Creator can successfully drop POI pins on the map for their tour route.				
Steps	1. On the map selection screen, search for a specific location. 2. Press the map to drop a POI pin. 3. Repeat to add at least 3 separate POIs. 4. Proceed to the POI editing phase.				
Expected	The map accurately registers the coordinates for each dropped pin, adding them to the tour's step sequence				
Date-Result					

Test ID	TC-19	Category	Functional	Severity	Critical
Objective	This test case checks the configuration of a standard narrative POI.				
Steps	1. Select a dropped POI to edit its information. 2. Define it as a "Story" location. 3. Leave the "Title" and "Story" fields empty and attempt to save. 4. Fill in the mandatory "Title" and "Story" text. 5. Optionally attach a cover image and address, then save.				
Expected	Step 3 throws a UI validation error. Step 5 successfully saves the POI data				
Date-Result					

Test ID	TC-20	Category	Functional	Severity	Major
Objective	This test case is to verify the configuration of a multiple-choice Trivia puzzle at a POI.				
Steps	1. Select a POI and set it as a "Puzzle" location. 2. Choose the "Trivia" puzzle type. 3. Enter the question text and define the Multiple Choice (MCQ) options. 4. Mark one option as the correct answer and save.				
Expected	The POI correctly saves the trivia parameters and maps the correct answer to the backend validation logic.				
Date-Result					

Test ID	TC-21	Category	Functional	Severity	Major
Objective	This test case is to verify the configuration fields for hardware-based puzzles.				
Steps	1. Select a POI and choose the "AR" puzzle type. Verify that the UI prompts for mesh/3D object parameters. 2. Select another POI and choose the "Gyroscope" puzzle type. Verify that the UI prompts for target angle/orientation values. 3. Fill in the required specifications and save.				
Expected	The UI dynamically changes the input fields based on the selected puzzle type, successfully storing the mesh or angle requirements.				
Date-Result					

Test ID	TC-22	Category	Usability / Functional	Severity	Critical
Objective	This test case is to verify that the users cannot proceed to the final review screen with incomplete POIs.				
Steps	1. During Personal Tour creation, leave one POI completely blank (missing mandatory title/story). 2. Attempt to press the "Next" button.				
Expected	The transition is blocked. The application provides a clear error message indicating which POI is missing mandatory information.				
Date-Result					

Test ID	TC-23	Category	Functional	Severity	Critical
Objective	This test case is to verify that the final submission of a user-created tour to the public database.				
Steps	1. Ensure all POIs have their mandatory fields filled. 2. Proceed to the "Review Your Tour" screen. 3. Verify the preview correctly displays the title, difficulty, and total step count. 4. Tap "Submit Tour" or "Publish".				
Expected	The system validates the final payload, saves the tour to the database, marks it as Public, and redirects the user to the newly created tour's detail page.				
Date-Result					

Test ID	TC-24	Category	Functional	Severity	Major
Objective	This test case is to verify that a user can spend tokens to reveal a puzzle hint during an active tour.				
Steps	1. Start a puzzle-based tour and reach an active puzzle POI. 2. Check the current token balance in the UI. 3. Tap the "Use Hint" button. 4. Confirm the token deduction prompt.				
Expected	The app deducts the correct token amount from the user's wallet and displays a textual/visual hint for the current puzzle on the screen.				
Date-Result					

Test ID	TC-25	Category	Functional	Severity	Major
Objective	This test case is to verify that a user can skip a puzzle step by spending tokens or watching an advertisement.				
Steps	1. Reach an active puzzle POI during a tour. 2. Tap the "Skip Puzzle" option. 3. Select "Pay with Tokens" (ensure sufficient balance). 4. Verify the puzzle is bypassed. 5. Reach the next puzzle and select "Watch Ad to Skip". Watch the ad to completion.				
Expected	In both scenarios, the system registers the step as completed, deducts tokens or finishes the ad callback, and unlocks the navigation to the next POI.				
Date-Result					

Test ID	TC-26	Category	Functional	Severity	Critical
Objective	This test case is to verify that submitting answers to a Trivia puzzle yields the correct progression or error state.				
Steps	1. Reach a Trivia puzzle POI during an active tour. 2. Select an incorrect multiple-choice option and tap "Submit". 3. Observe the UI feedback. 4. Select the correct multiple-choice option and tap "Submit".				
Expected	Step 2 triggers an "Incorrect Answer" UI state and prevents progression. Step 4 triggers a "Correct" success state, awards XP, and unlocks the next tour step.				
Date-Result					

Test ID	TC-27	Category	Functional	Severity	Minor
Objective	This test case is to verify that a user can successfully leave a rating and text review after finishing a tour.				
Steps	1. Complete the final step of an active tour to reach the "Finished tour" summary screen. 2. Tap the star rating UI to select a 4-star rating. 3. Enter a text review in the provided text box. 4. Tap "Submit Review".				
Expected	The system saves the review to the database, updates the tour's average rating globally, and redirects the user back to the tours screen.				
Date-Result					

Test ID	TC-28	Category	Functional	Severity	Minor
Objective	This test case is to verify that the users can save tours to a personal bookmark list.				
Steps	1. Navigate to a tour's detail page from the search/browse tab. 2. Tap the "Bookmark" (save) icon. 3. Navigate to the user's Profile page. 4. Open the "Saved Tours" section.				
Expected	The bookmark icon toggles to a filled state. The selected tour correctly appears in the user's "Saved Tours" list on their profile.				
Date-Result					

Test ID	TC-29	Category	Functional	Severity	Major
Objective	This test case is to verify that submitting a report successfully logs the issue for moderation.				
Steps	1. Open a published tour's detail page. 2. Tap the "Report" button. 3. Select a report type (e.g., "Safety Issue" or "Wrong Location"). 4. Enter an optional text comment and tap "Submit".				
Expected	The app displays a "Report Submitted" confirmation message. The backend moderation queue receives the payload with the tour ID, reporter ID, and timestamp.				
Date-Result					

Test ID	TC-30	Category	Functional	Severity	Minor
Objective	This test case is to verify that unfollowing a user correctly updates the social graph and UI.				
Steps	1. Navigate to the Profile page and tap "Following". 2. Locate a currently followed user in the list and tap "Unfollow". 3. Navigate back to the Profile page and check the "Following" count.				
Expected	The "Following" count immediately decrements by 1. The unfollowed user's activity no longer appears in the follower feed/notifications.				
Date-Result					

Test ID	TC-31	Category	Functional	Severity	Critical
Objective	This test case is to verify that the users can successfully purchase in-game tokens using real-world currency.				
Steps	1. Navigate to the app's wallet or store section. 2. Tap a token bundle (e.g. "100 Tokens for x ₺"). 3. Complete the mock transaction via the native OS prompt (Apple StoreKit test environment).				
Expected	The native payment gateway returns a success callback. The backend verifies the receipt, and the user's token balance increases by the purchased amount.				
Date-Result					

Test ID	TC-32	Category	Functional	Severity	Critical
Objective	This test case is to verify that purchasing a Premium subscription unlocks exclusive features.				
Steps	1. Tap "Upgrade to Premium" in the settings. 2. Complete the mock transaction via the native OS billing prompt. 3. Navigate to a "Premium Only" tour.				
Expected	The user's role in the database updates to "Premium". The previously locked "Premium Only" tour is now fully accessible without prompting for individual token unlocks.				
Date-Result					

Test ID	TC-33	Category	Functional	Severity	Major
Objective	This test case is to verify that a Normal User can spend tokens to permanently unlock a specific paid tour.				
Steps	1. Ensure the user account has a balance of at least 50 tokens. 2. Navigate to a paid tour requiring 50 tokens. 3. Tap "Unlock Tour" and confirm the prompt.				
Expected	The user's wallet balance decrements by 50. The "Start Tour" button becomes active, and the tour remains permanently unlocked for that user.				
Date-Result					

Test ID	TC-34	Category	Functional	Severity	Major
Objective	This test case is to verify that watching a rewarded video ad grants free tokens.				
Steps	1. Navigate to the wallet/store section. 2. Tap "Earn Free Tokens via Ad". 3. Let the test advertisement play completely to the end. 4. Close the ad interface.				
Expected	The third-party ad network sends a success callback to the app, and the backend deposits the designated amount of free tokens into the user's account.				
Date-Result					

Test ID	TC-35	Category	Functional	Severity	Major
Objective	This test case is to verify that a Content Creator can modify an existing published tour.				
Steps	1. Log in as a Content Creator and navigate to "Created Tours" on the Profile page. 2. Select a published tour and tap "Edit". 3. Modify the Title of the first POI and save the changes. 4. View the public detail page of that tour.				
Expected	The system successfully updates the database record, and the changes are immediately reflected on the public-facing tour details.				
Date-Result					

Test ID	TC-36	Category	Functional	Severity	Major
Objective	This test case is to verify that a creator can permanently delete their own tour.				
Steps	1. Log in as a Content Creator and select a created tour. 2. Tap "Delete Tour". 3. Confirm the deletion prompt. 4. Search for the tour by name in the main Browse tab.				
Expected	The tour is soft-deleted from the database, disappears from the creator's profile, and is no longer discoverable in public search results. It can still exist in completed tours.				
Date-Result					

Test ID	TC-37	Category	Functional	Severity	Minor
Objective	This test case is to verify that the creator analytics dashboard loads correctly.				
Steps	1. Log in as a Content Creator. 2. Navigate to the Tour Analytics dashboard. 3. Check the displayed metrics (Total Plays, Average Rating).				
Expected	The dashboard successfully fetches and displays accurate aggregated data for all tours authored by that creator.				
Date-Result					

Test ID	TC-38	Category	Security / Functional	Severity	Major
Objective	This test case is to verify that an Admin can process user reports and hide inappropriate content.				
Steps	1. Log in with a Super Admin account. 2. Navigate to the Admin Moderation Dashboard. 3. Select a tour that has been flagged by users. 4. Tap "Hide Tour" or "Delete Tour".				

Expected	The Admin dashboard correctly lists flagged content. The action successfully changes the tour's visibility status, removing it from Normal User access.
Date-Result	

Test ID	TC-39	Category	Security / Functional	Severity	Critical
Objective	This test case is to verify that an Admin can permanently remove a user account.				
Steps	1. Log in with a Super Admin account. 2. Navigate to the User Management console. 3. Search for a specific test user account and tap "Delete User". 4. Attempt to log in to the app using the deleted user's credentials.				
Expected	The system deletes the user profile and cascades deletions to their associated data. The login attempt in Step 4 fails with a "User not found" error.				
Date-Result					

5.2. Non-functional Test Cases

Test ID	TC-40	Category	Reliability	Severity	Major
Objective	This test case is to verify that losing internet access during an active tour triggers Offline Mode safely.				
Steps	1. Start a "Story Mode" tour while connected to a stable cellular network. 2. Swipe down on the mobile OS and enable "Airplane Mode" to disconnect from the internet. 3. Physically walk to the next designated POI coordinate. 4. Attempt to unlock the current cached step.				
Expected	The app does not crash. It displays a clear "Offline Mode" indicator, successfully verifies the GPS coordinate locally, and allows the user to read the current step's story.				
Date-Result					

Test ID	TC-41	Category	Compliance	Severity	Critical
Objective	This test case is to verify that the app halts core functionality until mandatory legal location-tracking agreements are accepted.				
Steps	1. Install a fresh build of the application and create a new account. 2. Log in and attempt to navigate to the "Map" tab or start a tour. 3. Observe the UI response. 4. Tap "I Agree" on the consent modal.				
Expected	Step 2 triggers a mandatory KVKK/GDPR location tracking consent modal, blocking access to map features. Step 4 permanently dismisses the modal and grants access to the features.				
Date-Result					

Test ID	TC-42	Category	Security	Severity	Critical
Objective	This test case is to verify that the users cannot bypass premium token deductions via UI manipulation.				
Steps	1. Log in with an account that has exactly 0 game tokens. 2. Navigate to a "Premium Tour" that costs 50 tokens. 3. Attempt to rapidly tap the "Unlock Tour" button while toggling Airplane mode to simulate a race condition.				
Expected	The server-side validation acts as the ultimate authority. The request is denied, the UI displays an "Insufficient Tokens" error, and the premium tour remains locked.				
Date-Result					

Test ID	TC-43	Category	Compatibility	Severity	Major
Objective	This test case is to verify that the app gracefully handles devices lacking specific hardware sensors.				
Steps	1. Install the application on a lower-end test device known to lack a physical gyroscope sensor. 2. Navigate to "Personal Creation" to create a new tour. 3. Attempt to add a "Gyroscope" puzzle to a POI.				
Expected	The application detects the missing hardware upon startup. The UI either greys out or hides the "Gyroscope" puzzle option to prevent the creation of unplayable content.				
Date-Result					

Test ID	TC-44	Category	Performance	Severity	Major
Objective	This test case is to verify that the AR camera view renders smoothly without severe frame drops.				
Steps	1. Start a tour featuring an Augmented Reality puzzle on a mid-range test device. 2. Arrive at the POI and activate the AR camera interface. 3. Pan the camera quickly across the physical environment for 30 seconds.				
Expected	The AR view maintains a visually stable frame rate (minimum 24 frames per second). The application does not freeze, crash, or exhibit nauseating jitter during rapid movement.				
Date-Result					

Test ID	TC-45	Category	Performance	Severity	Major
Objective	This test case is to verify that the active background location tracking operates within acceptable energy limits.				
Steps	1. Charge a test device to exactly 100% battery. 2. Start a tour and leave the app running in the background while physically walking for exactly 1 hour. 3. Check the device's battery percentage after 60 minutes.				
Expected	The application consumes no more than 15% of the total battery capacity during the hour of active GPS background tracking.				
Date-Result					

Test ID	TC-46	Category	Reliability	Severity	Critical
Objective	This test case is to verify that the AI-generated coordinates are cross-referenced with Google Maps to prevent impossible POI placements.				
Steps	1. Request an AI-generated tour through the UI. 2. Once generated, view the tour's route on the map interface. 3. Zoom in to verify the placement of the POI pins.				
Expected	The backend validation successfully intercepts any hallucinated coordinates from the LLM, ensuring all POI pins are placed on traversable, real-world landmasses and not in oceans or inaccessible zones.				
Date-Result					

Test ID	TC-47	Category	Reliability	Severity	Minor
Objective	This test case is to verify that the application successfully receives OS-level push notifications.				
Steps	1. Log into "Account A" on the test device and background the app. 2. On a separate device, log into "Account B" and follow "Account A". 3. Observe the lock screen of the first test device. 4. Tap the incoming notification.				
Expected	An OS-level push notification appears stating "Account B started following you". Tapping the notification opens the app directly to the Follower Feed or Profile page.				
Date-Result					

Test ID	TC-48	Category	Usability	Severity	Minor
Objective	This test case is to verify that changing the language preference translates hardcoded UI elements seamlessly.				
Steps	1. Navigate to the Settings page. 2. Change the language preference from "English" to "Turkish". 3. Navigate back to the Profile and main Navigation menus.				
Expected	All hardcoded navigational labels, button texts, and static metrics (e.g., "Followers", "Settings") update immediately to the selected language without requiring an app restart.				
Date-Result					

Test ID	TC-49	Category	Performance	Severity	Major
Objective	This test case is to verify that the app maintains swift loading times when fetching database lists.				
Steps	1. Connect the device to a standard 4.5G network. 2. Perform a "pull-to-refresh" gesture on the main "Tours" feed, which triggers an API call to the Django backend. 3. Observe the loading indicator.				
Expected	The loading indicator disappears and the UI populates with fresh tour data in less than 1 second, confirming the backend's sub-200ms processing capability.				
Date-Result					

6. Consideration of Various Factors in Engineering Design

6.1. Engineering Design Factors

Public Health (Effect Level: 6/10): Because the application encourages physical movement and walking tours, public health was a significant consideration. The system design incorporates accessibility considerations and walking-duration safety guidance in tours itself. Furthermore, the UI includes general health disclaimers to ensure users are aware of their physical limits during extended "Story Mode" or "Hybrid Mode" tours.

Public Safety (Effect Level: 9/10): Public safety is the most critical non-technical factor in this project, scoring a 9/10. The core mechanic relies on real-time location tracking (GPS) and Augmented Reality (AR), which introduces the risk of "distracted walking" and potential traffic accidents. To mitigate this liability, the design includes a mandatory "physical awareness" warning upon each app launch. Additionally, the game logic is strictly

constrained to prevent the AI from generating clues, waypoints, or AR puzzles in hazardous or restricted areas. The system also limits user interaction if the GPS detects movement speeds exceeding a walking pace (e.g., driving) to prevent accidents.

Security (Effect Level: 9/10): Security architecture is paramount due to the collection of real-time location data and the implementation of an in-game economy. To protect user privacy and comply with GDPR/KVKK , the database schema is designed to store minimal location data purely for navigation verification, which is encrypted and subject to automated retention policies. Authentication relies on OAuth 2.0 and JWT to minimize credential handling. To secure the token economy, the system design centralizes all XP and Token calculations on the backend and includes hardware checks to block execution on rooted/jailbroken devices, preventing GPS spoofing.

Public Welfare (Effect Level: 3/10): To support public welfare, the app is designed with fair access and inclusivity in mind. The system architecture includes an "Offline Mode" to support users who may not have continuous cellular data access or who are in low-connectivity zones. The design also ensures the AR components degrade gracefully to a 2D interface for users with lower-end mobile devices.

Global Factors (Effect Level: 6/10): As a tourism application, Odyssey inherently targets a global audience. The architecture addresses this through inclusive design and localization capabilities. The frontend React Native application isolates UI text from the codebase to facilitate future translation into multiple languages beyond the initial English release.

Cultural Factors (Effect Level: 8/10): When generating content via the AI module, cultural sensitivity is vital. The system utilizes structured pre-determined prompts to ensure the LLM avoids sensitive locations or inappropriate topics. The architecture also aims to maintain respectful content and localized tone when generating historical or cultural narratives.

Social Factors (Effect Level: 7/10): The social dynamics of the platform are managed through community-driven moderation features. The design includes reporting endpoints that allow users to flag inappropriate routes, unsafe instructions, or malicious users. If a tour receives multiple reports, the backend service automatically flags it for admin review and temporarily hides it from search results to protect the community.

Environmental Factors (Effect Level: 4/10): Environmentally, the design attempts to guide users respectfully through urban spaces. On a technical level, to reduce the environmental impact of device energy consumption, the system architecture restricts background location tracking to consume no more than 15% of the device's battery per hour of active use.

Economic Factors (Effect Level: 9/10): Economic constraints heavily influenced the system's architecture. The design strictly monitors and limits the usage of Generative AI tokens to prevent exceeding fixed monthly budgets. Furthermore, the monetization model maps directly to proprietary payment gateways to comply with platform economic policies

Factors	Effect (Out of 10)
Public Health	6
Public Safety	9
Security	9
Public Welfare	3
Global Factors	6
Cultural Factors	8
Social Factors	7
Environmental Factors	4
Economic Factors	9

6.2. Constraints

The system design was shaped by the following specific constraints:

- **Economic Constraints:** The architecture must efficiently manage LLM token usage to stay within budget. Deployment is restricted to platforms offering free student tiers during initial phases.
- **Technical Constraints:** Development is strictly limited to React Native for cross-platform compatibility. The core AI features require an active internet connection. AI generation must return a response within 60 seconds to maintain immersion.
- **Legal and Ethical Constraints:** The database architecture must facilitate the anonymization and deletion of location data to strictly comply with GDPR and

KVKK. The UI must prominently feature disclaimers regarding the factual accuracy of AI-generated "thriller" narratives.

- **Implementation Constraints:** The scope is limited to the development capacity of a 5-person engineering team using Git/GitHub for version control. Token context windows restrict the length of AI-generated stories.
- **Hardware/Sensor Constraints:** AR features are constrained to devices supporting ARCore/ARKit, requiring graceful UI degradation for unsupported hardware. Gyroscope puzzles require physical hardware checks.

6.3. Standards

Documentation & Modeling: IEEE 830-1998 standardizes the requirement structures, while UML 2.5.1 dictates the structural and dynamic system diagrams.

Programming Style: The Django backend strictly adheres to the PEP 8 style guide for Python. The React Native frontend utilizes TypeScript Strict Mode to prevent runtime errors.

Data & Communication: All client-server transmission is secured using HTTPS . Data payloads between the frontend and LLM are strictly formatted as JSON.

Regulatory: The system design follows GDPR and KVKK regulations regarding explicit consent for location tracking. Secure user access is standardized using OAuth 2.0 protocols.

7. Teamwork Details

The development of the Odyssey application was undertaken by a five-person engineering team: Ege Ertem, Can Kütükoğlu, Sami Bora Akoğuz, Mehmet Rodi Aydoğdu, and Cem Tarkan Tekcan. To ensure efficient collaboration and successful project delivery, the team adopted an Agile-inspired methodology, dividing the workforce into specialized sub-teams while maintaining synchronized integration efforts.

7.1. Contributing and functioning effectively on the team

To parallelize the implementation phase and maximize efficiency, the team was split into two primary clusters based on domain expertise, with specific workloads distributed during active sprints. During a recent sprint, individual contributions were clearly defined to tackle new functionalities, bugs, and missing features:

- Can Kütükoğlu took responsibility for implementing profile pictures, specifically integrating the Dicebear library to allow users to generate custom avatars from queries.
- Mehmet Rodi Aydoğdu functioned as the core developer for media and external API accuracy, taking on the integration of cloud storage for tour pictures and redesigning the Gemini AI output to cross-check randomly generated coordinates with Google Maps.
- Ege Ertem focused on the application's global accessibility by handling the translation of hardcoded elements to support multiple app languages.
- Bora Akoğuz contributed to user experience by developing the Settings page, allowing users to modify personal settings such as email, password, avatar, and privacy configurations.

- Cem Tarkan Tekcan functioned as the API integration supervisor and debugger, taking responsibility for normalizing inconsistent endpoint naming conventions and fixing critical Swagger documentation errors. He also assisted with the Settings implementation.

7.2. Helping creating a collaborative and inclusive environment

Can Kütükoğlu contributed to the collaborative environment by regularly sharing implementation updates with the team. He supported inclusiveness by explaining the decision making process clearly. In this way, technical and non-technical requirements and concerns could be discussed together as a team. He also participated in the team alignment management by analyzing the team's workload and deciding roles with teammates and being open to feedback.

Mehmet Rodi Aydoğdu supported inclusiveness by keeping the frontend team informed on the changes to the backend. He also wrote clear explanations and todo's on his pull requests so that the 2 mandatory reviewers knew what was done and what needed to be added. He also actively communicated with other team members while taking critical decisions.

Ege Ertem helped the team by creating a safe work environment. This made sure that everyone was working as one and all together without having any conflicts. He dedicated his place to this project, where many contributions were made. He planned the face-to-face meetings and gathered everyone at a specific location. He made sure everyone bonded better in the team, which in return improved our communication and teamwork capabilities.

Sami Bora Akoğuz contributed by setting up the Github repository and assigning automatic jobs and repository rules that ensured that each pull request is viewed by at least 2 people and that passes the tests. This ensured that at least 1 person from each team was involved in a change in the codebase. He also took an active role during meetings and bonding activities.

Cem Tarkan Tekcan ensured that everyone knew what each other was doing during the project. He facilitated clear communication between teams and wrote clear documentation on each task that needed to be done. He also scanned the requirements of the deliverables beforehand and ensured everyone knew what was needed.

7.3. Taking lead role and sharing leadership on the team

Ege Ertem took the lead role during WP 1 (Project Initiation & Problem Definition), steering the team through problem finding, brainstorming, and the critical supervisor meetings.

Can Kütükoğlu took the lead role during WP 2 (Analysis, Design & Requirements), managing the market literature review, defining the system models, and overseeing the UI/UX mockups.

Sami Bora Akoğuz took the lead role during WP 3 (Initial Implementation MVP), orchestrating the coding environment setup, Docker containerization, and the delivery of the initial backend and frontend MVP.

Cem Tarkan Tekcan took a distinct leadership role in managing project deliverables and documentation requirements. For instance, he took the initiative to review the CS 492

Detailed Design Report guidelines independently, summarizing critical requirements (such as the need for 50+ test cases) and strategizing how the team could repurpose existing backend test scripts to fulfill these goals efficiently

Mehmet Rodi Aydogdu took the lead during the LLM integration phase where he handled the automated AI Tour creation and its integration. He ensured clear communication between the frontend and backend teams.

8. Glossary

- **API (Application Programming Interface):** A set of protocols and tools that allows different software applications to communicate with each other. In this project, it refers to backend services and third-party services (like maps).
- **AR (Augmented Reality):** A technology that superimposes a computer-generated image on a user's view of the real world, utilized in the project for puzzle mechanics.
- **Blue-Green Deployment:** A technique that reduces downtime and risk by running two identical production environments (Blue and Green). Only one serves live traffic while the other is updated.
- **Cold-Start:** The time it takes for the application to load from a completely closed state to a fully interactive state.
- **Cross-Platform:** Software developed to work on multiple mobile operating systems (iOS and Android) using a single code base (in this case, React Native).
- **Datadog / Sentry:** Monitoring and observability platforms used to track application errors, performance metrics, and logs.
- **GDPR (General Data Protection Regulation):** A legal framework that sets guidelines for the collection and processing of personal information from individuals who live in the European Union.
- **Generative AI:** A type of artificial intelligence technology capable of generating text, images, or other media in response to prompts. Used here to create "Story Mode" narratives.

- **Geocaching:** An outdoor recreational activity, in which participants use a Global Positioning System (GPS) receiver or mobile device and other navigational techniques to hide and seek containers.
- **GPS (Global Positioning System):** A satellite-based navigation system used to determine the ground position of an object.
- **Hotfix:** A small piece of code developed to correct a specific bug in a software product that is already live in production, usually requiring immediate attention.
- **JSON (JavaScript Object Notation):** A lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate. Used here for structuring AI puzzle outputs.
- **KVKK (Kişisel Verilerin Korunması Kanunu):** The Turkish Law on the Protection of Personal Data, roughly the Turkish equivalent of GDPR.
- **LLM (Large Language Model):** A deep learning algorithm that can recognize, summarize, translate, predict, and generate text and other content based on knowledge gained from massive datasets.
- **Localization:** The process of adapting the application interface and content to a specific locale or market (e.g., translating text from English to other languages).
- **Mocked Data:** Artificial data used during testing to simulate real-world conditions (e.g., Mocked GPS data allows testing location features without walking around physically).
- **Model Context Protocol (MCP):** An open standard that provides a universal interface for AI models to securely discover and access external data, tools, and resources.
- **OpenAPI (Swagger):** A specification for building, documenting, and consuming RESTful web services.
- **POI (Point of Interest):** A specific point location that someone may find useful or interesting, such as a landmark, museum, or puzzle location.
- **React Native:** An open-source UI software framework created by Meta Platforms, used to develop applications for Android and iOS by enabling developers to use React along with native platform capabilities.

- **Rolling Deployment:** A deployment strategy where the new version of an application replaces the old version across the server fleet one by one or in batches, ensuring zero downtime.
- **Token (AI):** The basic unit of text (part of a word) that an LLM processes. Usage costs are calculated based on the number of tokens processed.
- **Token (Game Currency):** A virtual currency within the application used to unlock tours or buy hints, distinct from AI tokens.
- **XP (Experience Points):** A unit of measurement used in the game to quantify a user's progression and level advancement.

9. References

[1] *Cityfans*. CityFans [Online]. Available: https://cityfans.com/en. [Accessed: 16-Nov-2025]

[2] *Questo*. Questo, 12-Apr-2025. [Online]. Available: https://questoapp.com/. [Accessed: 16-Nov-2025]

[3] "FAQ", *Explorial.* Explorial. [Online]. Available: https://explorial.com/ [Accessed: 16-Nov-2025]