UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Travel Tales

-a travel journal application-

Name: Rujac Roxana
Group: 30234

Table of Contents

Project Specification

Travel Tales s is a **web-based travel journal application** that enables users to document their travel experiences interactively. It allows users to **create journals**, **add travel entries**, **upload media**, and **pin locations on a map**. Additionally, the app fosters social interactions by enabling users to **share their travel experiences** and **send virtual postcards**.

Functional Requirements

User Management

- **User Registration & Authentication**

- Sign up, log in, log out
- Password encryption (BCrypt)

- **User Profiles**

- Update profile details (name, email, username)

Journal & Entry Management

- **Create a Journal**

- Add title, cover image, description

-**Add an Entry to a Journal**

- Title & Description
- Location (manual pin or automatic geolocation)
- Attach media (photos/videos)

- **Edit & Delete Entries**

Map Integration

- **Interactive World Map**

- Users can view their travel pins

Social & Community Features

- **Send Virtual Postcards**

Use Case Model

Use Cases Identification

1. Use-Case: User Registration & Authentication

- **Level:** System-level
- **Primary Actor:** New User
- **Main Success Scenario:**
  - User navigates to the **Sign-Up page**.
  - User enters **name, email, password, and username**.
  - System validates the input and checks if the email is already registered.
  - System activates the account, and the user can log in.
  - User enters credentials and logs in successfully.
- **Extensions:**
  - **3a.** If email is already registered, an error message is displayed.

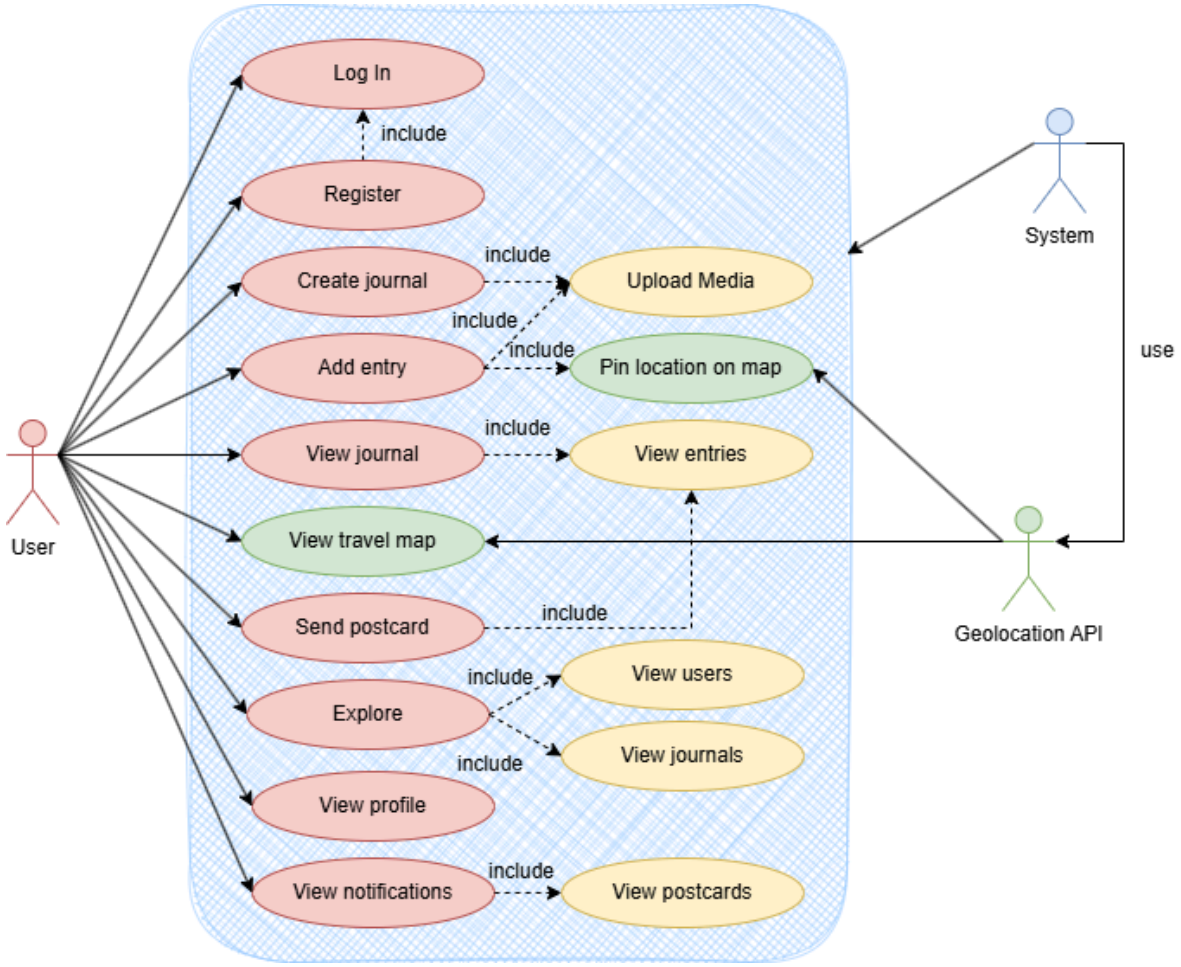
2. Use-Case: Create a Travel Journal

- **Level:** User-level
- **Primary Actor:** Registered User
- **Main Success Scenario:**
  - User logs in and navigates to the "Create Journal" section.
  - User enters **journal title, description, and selects a cover image**.
  - User clicks **"Create"**, and the system saves the journal.
  - The newly created journal appears in the user's dashboard.
- **Extensions:**
  - **2a.** If the title is empty, prompt the user to enter one.
  - **3a.** If an error occurs in saving, display an appropriate message.

3. Use-Case: Add an Entry to a Journal

- **Level:** User-level
- **Primary Actor:** Registered User
- **Main Success Scenario:**
  - User opens an existing journal.
  - User clicks **"Add New Entry"**.
  - User enters:
    - **Title & description**
    - **Location** (auto or manual pin)
    - **Uploads images**
  - User clicks "Save", and the system stores the entry.
  - Entry appears inside the journal with date and media.
- **Extensions:**
  - **3a.** If location is not provided, prompt the user.
  - **3b.** If media upload fails, allow retry or continue without media.

UML Use Case Diagrams



Non-functional Requirements

- **Scalability**

The system should be able to handle an increasing number of users and journal entries without a significant drop in performance. This includes efficiently managing storage for media (photos, videos) and ensuring the backend can scale as needed.

- **Performance & Responsiveness**

The system should provide fast response times for user actions like uploading journal entries, fetching maps, or sending postcards. The UI should remain smooth and responsive for the best user experience.

- **Security & Privacy**

User data (journal entries, photos, and locations) must be protected through authentication, authorization, and encryption. The system must follow data privacy best practices to keep users' personal travel information secure.

Supplementary Specifications

Design Constraints

- **Backend:** Must use **Java Spring Boot** for developing the backend.
- **Frontend:** Must use **React** for building an interactive and dynamic UI.
- **Database:** Must use **MySQL** as the primary database for structured data storage.
- **Authentication:** Must use **Spring Security** for secure user authentication.

Glossary

1. **MySQL Database =** A relational database management system (RDBMS) that stores structured data in tables with relationships. MySQL uses SQL (Structured Query Language) for queries.

2. **Geolocation API =** A web API that retrieves the user's current location (latitude & longitude) and integrates it with mapping services.

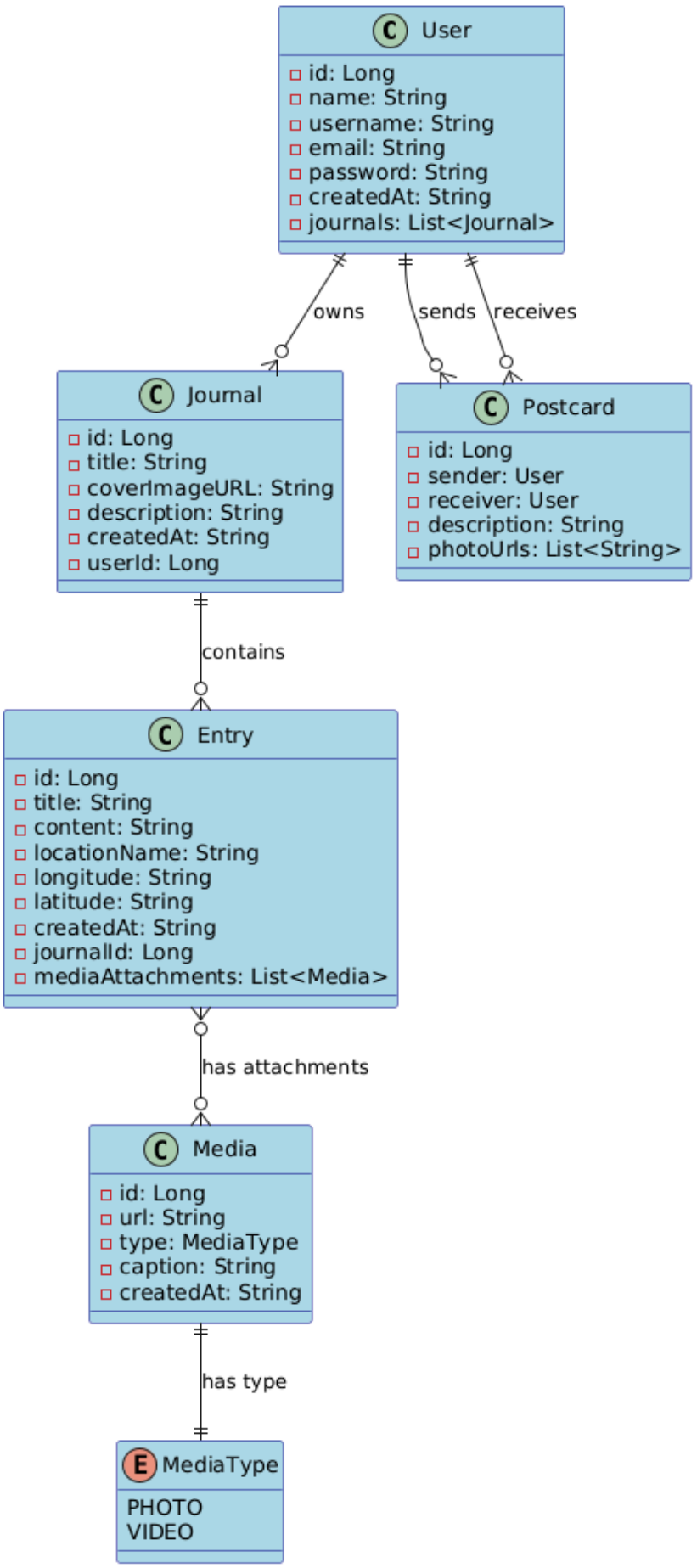
3. **API (Application Programming Interface)** - A set of rules and protocols that allows different software applications to communicate with each other.

4. **JPA (Java Persistence API)** - A Java specification for managing relational data in applications using object-relational mapping.

5. **Repository** - A design pattern that provides an abstraction layer for data access operations.

6. **REST (Representational State Transfer)** - An architectural style for designing web services using standard HTTP methods.

Domain Model



Architectural Design

Conceptual Architecture

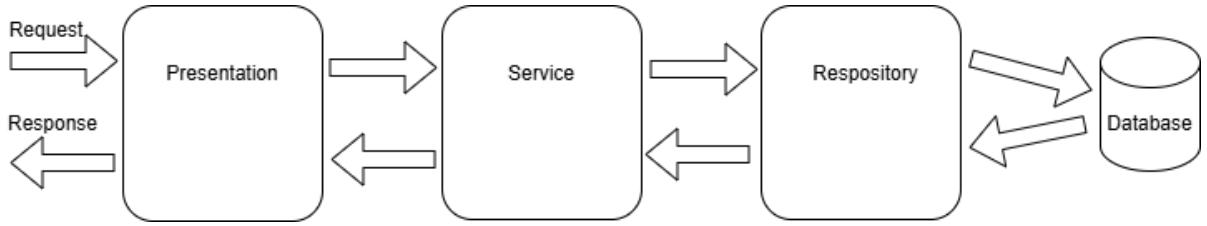
Multi-layer monolithic architecture with 4 layers, encapsulating 2 backend modules and a frontend application. Layers in orders:

-Presentation Layer

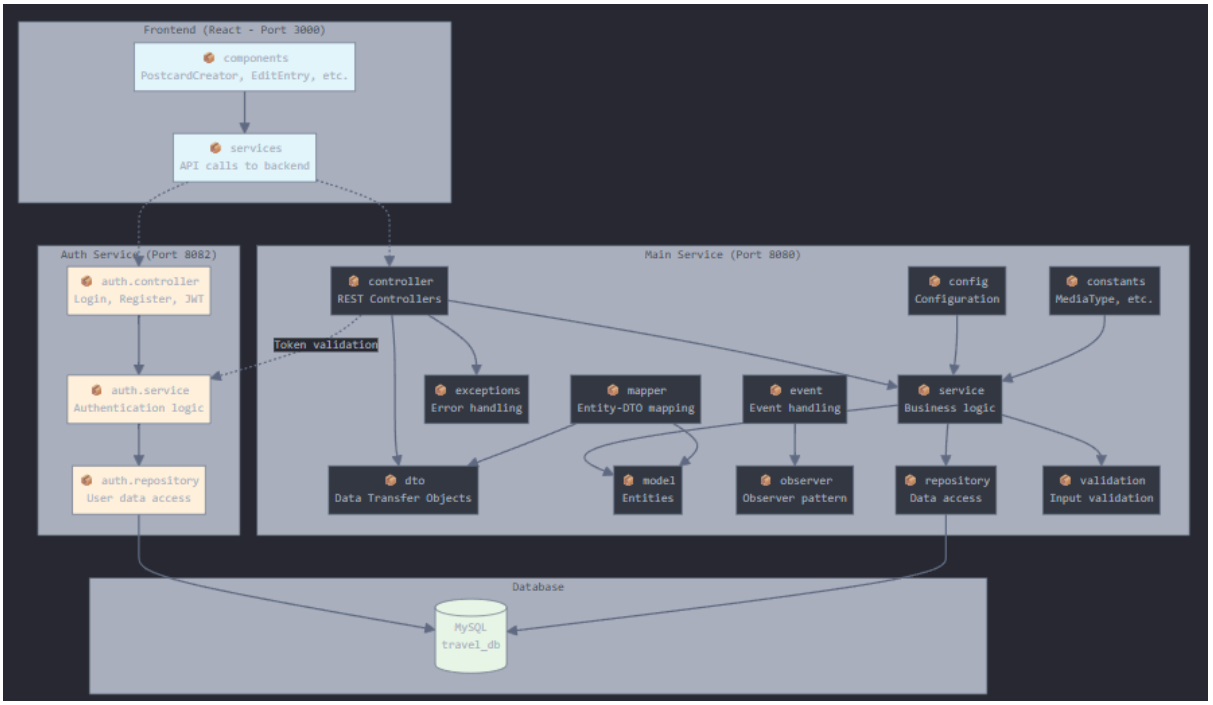
-Service Layer

-Repository Layer

-Database Layer

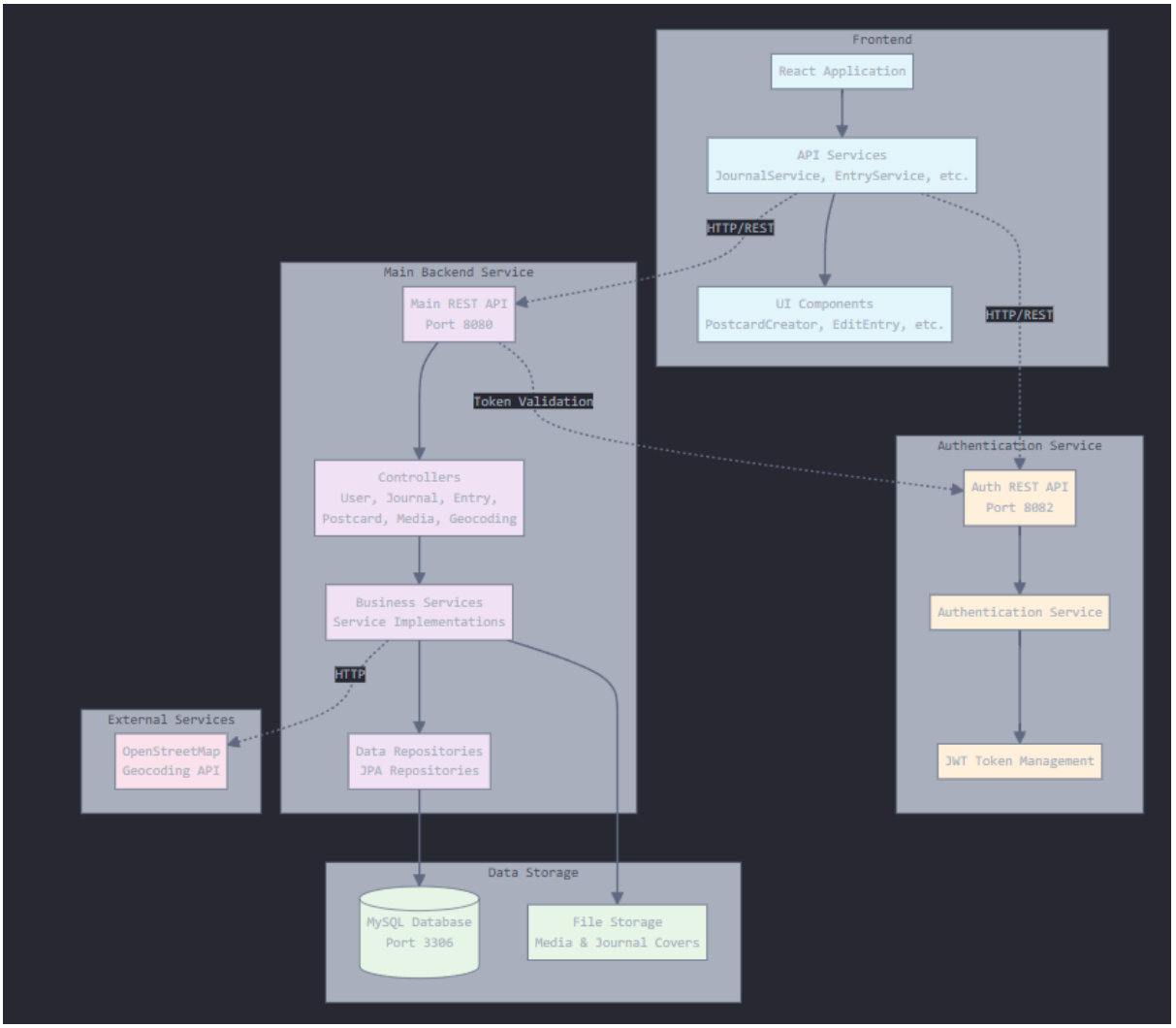


Package Design

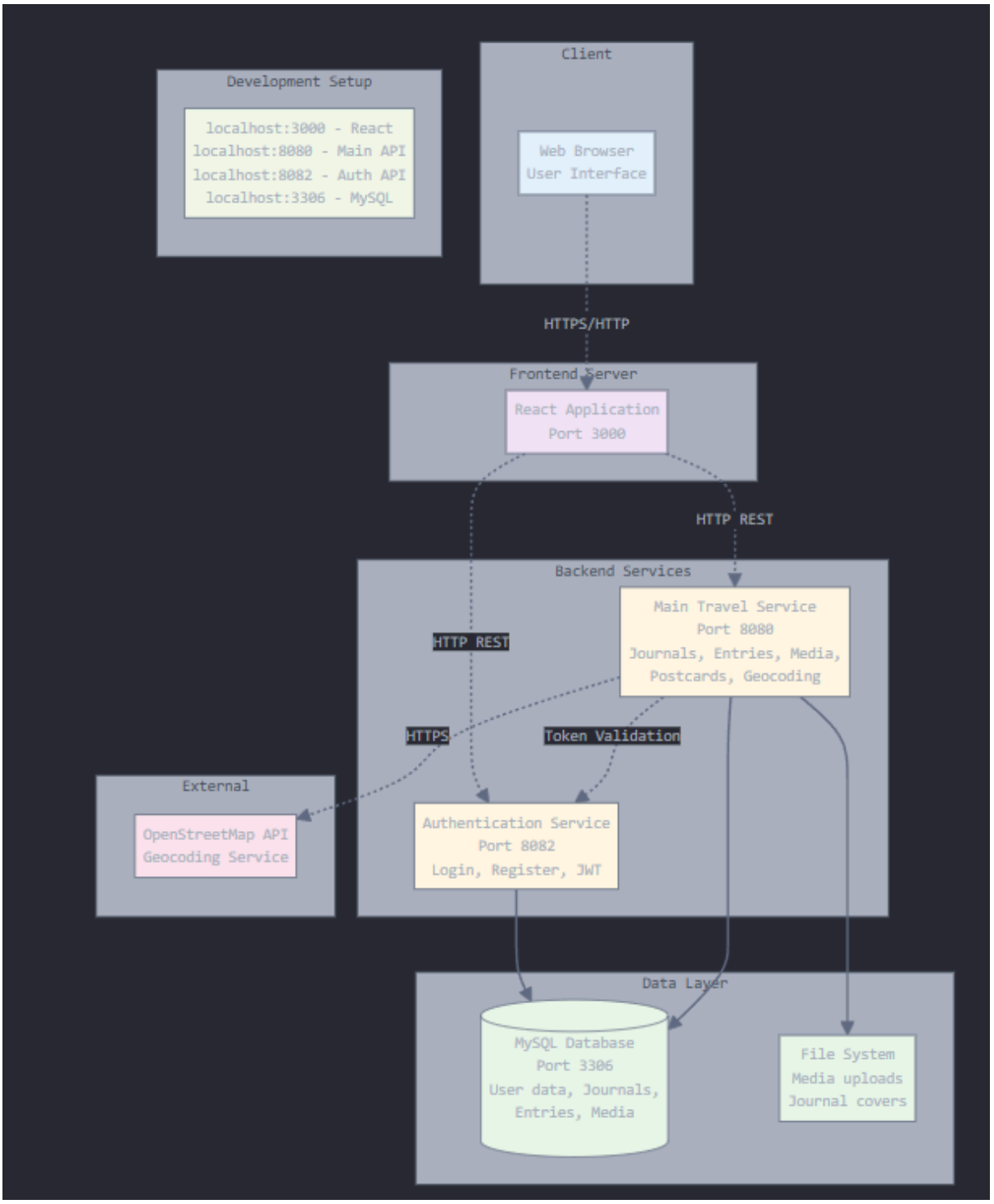


Component and Deployment Diagram

Component diagram:

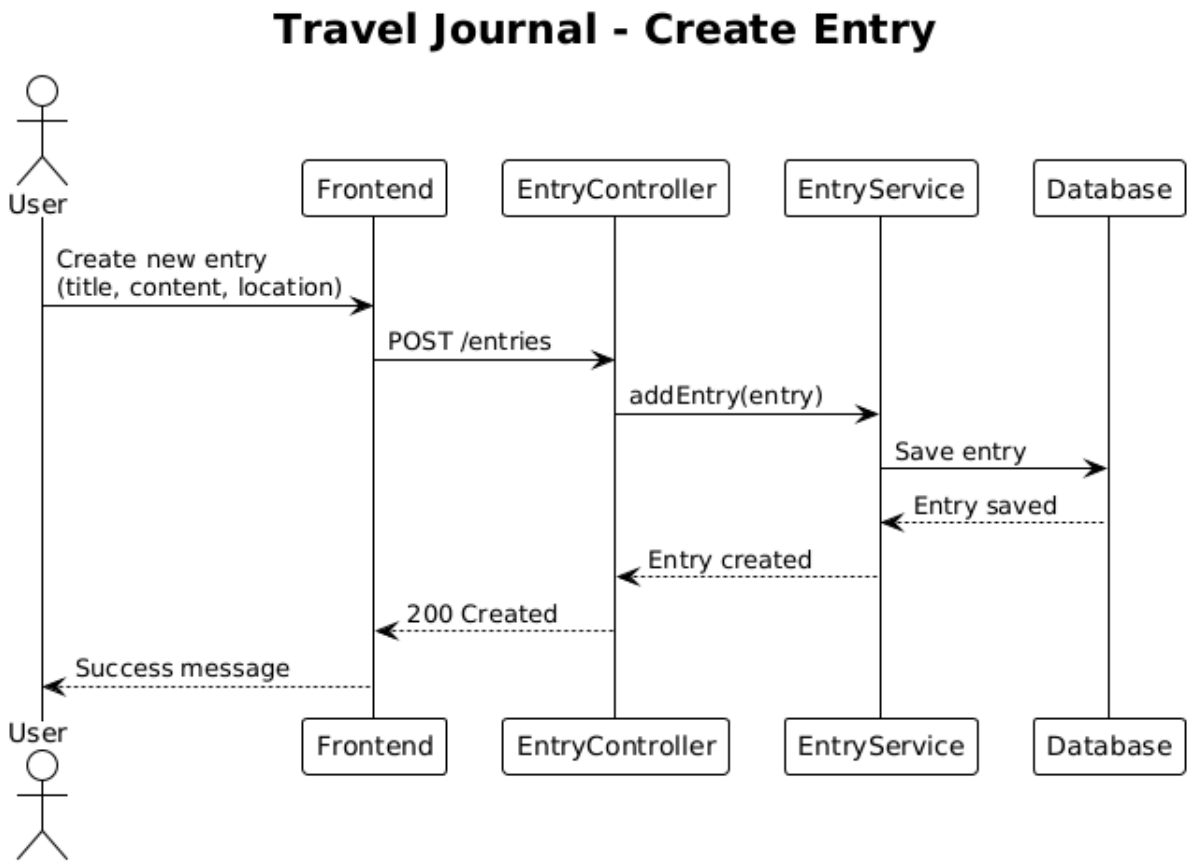


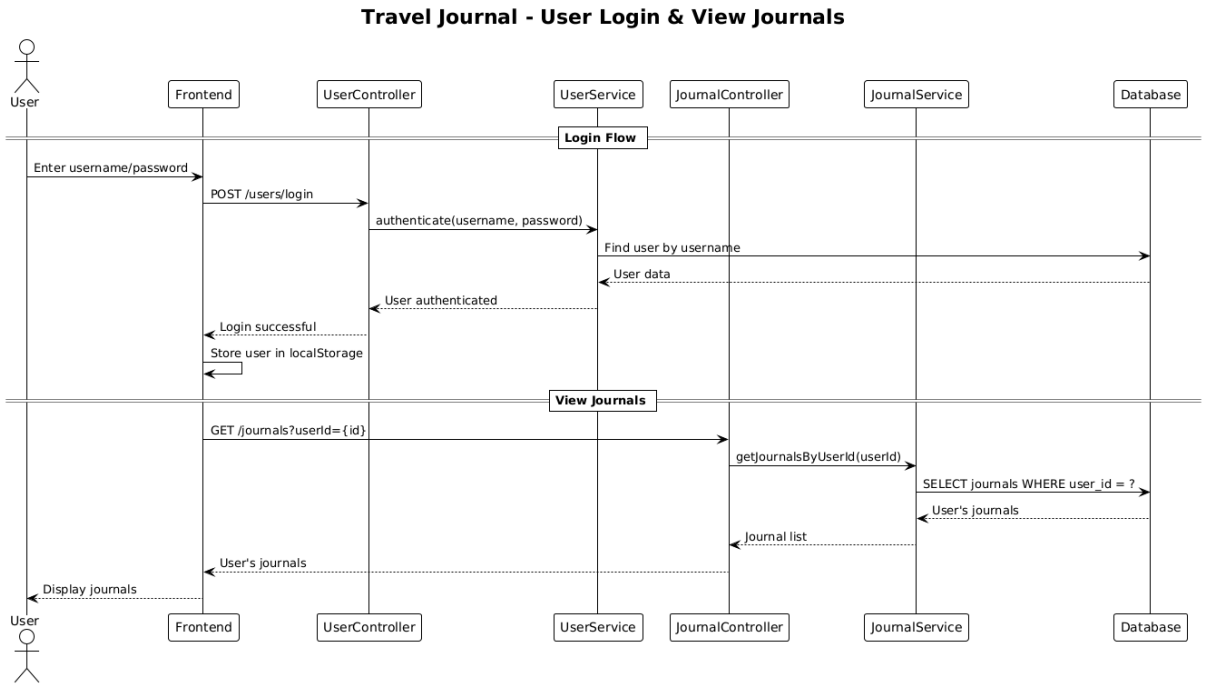
Deployment diagram:



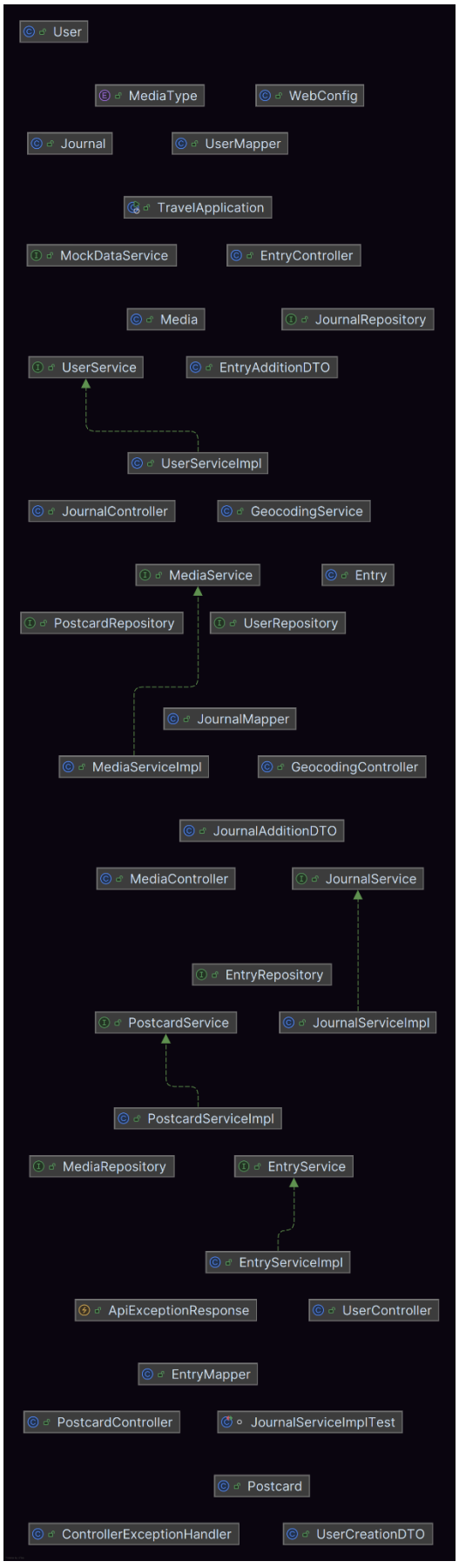
Design Model

Dynamic Behavior

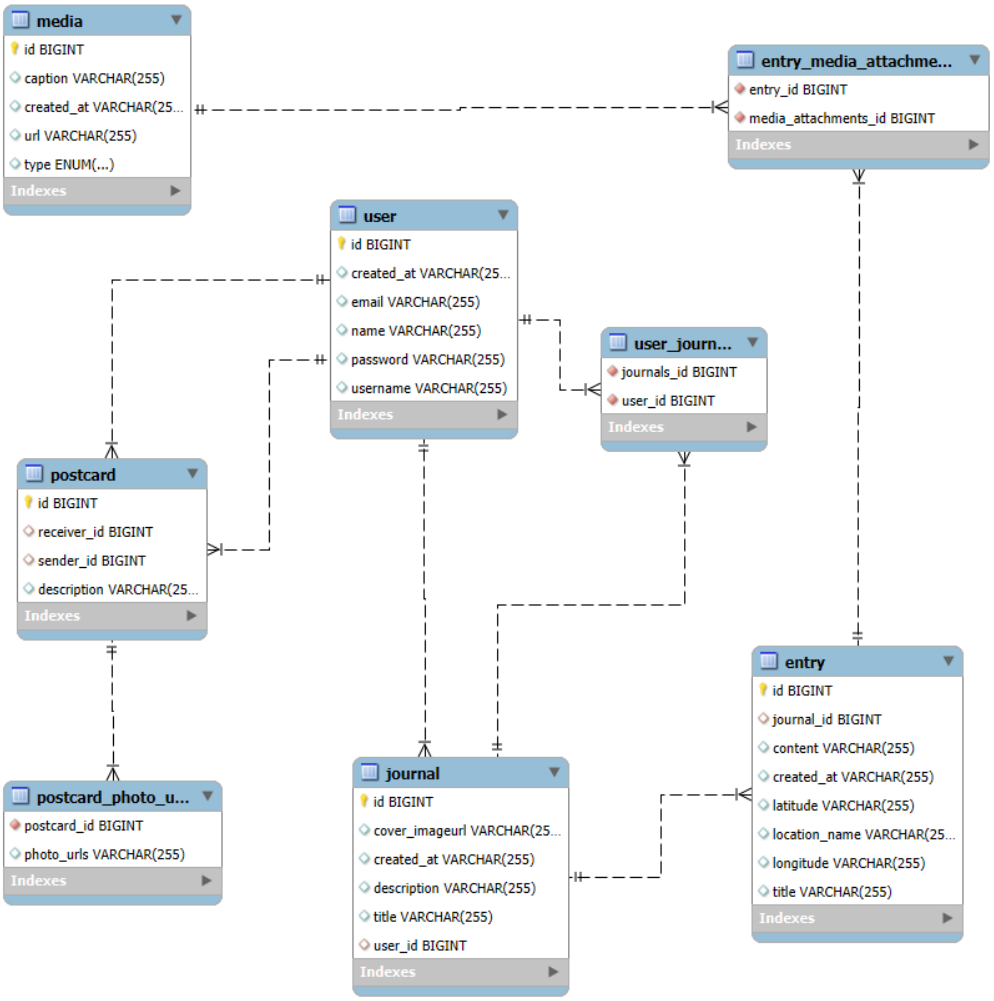




Class Diagram



Data Model



System Testing

The travel journal application underwent comprehensive testing using both automated and manual methodologies to ensure robust functionality and user experience.

Unit Testing

Business logic testing was implemented at the service layer using JUnit 5 and Mockito framework, with database access properly mocked. Unit tests were developed for the following core components:

- **UserService**: User creation, authentication, profile updates, and retrieval operations
- **JournalService**: Journal creation, modification, deletion, and listing functionality

- **EntryService**: Entry management including creation, updates, location geocoding, and media attachments
- **MediaService**: File upload handling, media type determination, and storage management

The primary objective was to verify correct method behavior within services, proper exception handling, and accurate repository method invocations.

Manual Testing (Functional Testing)

Beyond automated testing, the application was thoroughly validated through manual testing in the user interface (UI), simulating real-world usage scenarios. The following functionalities were tested:

- User registration and authentication workflows
- Journal creation and management operations
- Entry creation with location data and media uploads
- Navigation between journals and entries
- Image upload and display functionality
- Location geocoding and mapping features

Integration Testing

Complete end-to-end workflows were verified to ensure proper inter-module communication, including user authentication, journal management, entry creation with media attachments, and location services integration.

Future Improvements

Performance Optimizations:
-File Storage: Migrate to cloud storage (AWS S3, Google Cloud Storage) with CDN integration for faster media delivery

User Experience Improvements:
-Advanced Search: Implement full-text search with filters by date, location, tags, and media type
-Social Features: Add ability to share journals publicly, follow other users, and create collaborative journals

Technical Enhancements:
-Microservices Architecture: Break down monolithic structure into microservices for better scalability
-Real-time Features: Implement WebSocket support for real-time updates and notifications

Feature Expansions:
-Advanced Maps Integration: Interactive maps with route planning, visited locations tracking, and points of interest
-Export Functionality: Allow users to export journals as PDF books or create physical photo albums

Conclusion

The travel journal application successfully delivers a complete digital solution for documenting travel experiences. Built with Spring Boot and React, it enables users to create journals, write entries with photos and location data, and organize their memories effectively.

All core functionalities were implemented including user authentication, journal management, entry creation with media uploads, and location services. The clean architecture ensures maintainability while comprehensive testing validates system reliability.

Bibliography

1. https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/
2. https://www.geeksforgeeks.org/mvc-design-pattern/
3. https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/
4. https://www.geeksforgeeks.org/introduction-to-spring-security-and-its-features/
5. https://spring.io/guides/gs/accessing-data-jpa