UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Travel Journal

Name: Rujac Roxana
Group: 30234

Table of Contents

Deliverable 1

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**, **send virtual postcards**, and **collect travel badges**.

Functional Requirements

User Management

- **User Registration & Authentication**

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

- **User Profiles**

- Update profile details (name, bio, profile picture)
- View other users' profiles

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**

- **Time Capsule Mode**

- Lock an entry for a set period

Map Integration

- **Interactive World Map**

- Users can view their travel pins
- Click on a pin to open a travel entry

- **Location Search**

- Search for a place & auto-fetch coordinates

Social & Community Features

- **Send Virtual Postcards**

- Customizable digital postcards

- **Travel Challenges & Badges**

- Earn badges for visiting locations

Use Case Model 1

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.

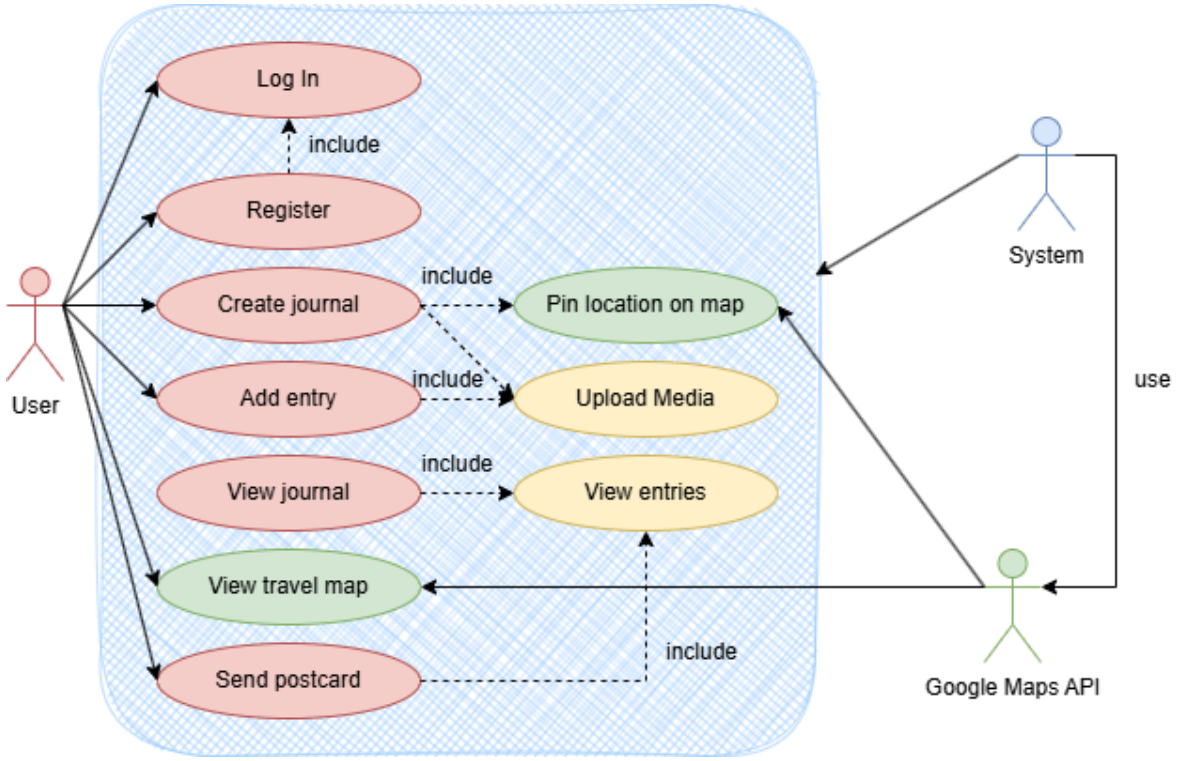
3. Use-Case: Add an Entry to a Journal

- **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/videos**
    - 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.

4. Use-Case: View Travel Map & Entries

- **Level:** System-level
- **Primary Actor:** Registered User
- **Main Success Scenario:**
    - User navigates to the **World Map**.
    - System loads user's pinned locations.
    - User clicks on a **pin** to view related journal entries.
    - System displays the **entry details** (title, photos, description).
- **Extensions:**
    - **2a.** If the map fails to load, show an error message.
    - **3a.** If no entries exist for a location, display a message.

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 RESTful API.
- **Frontend:** Must use **Thymeleaf** for building an interactive and dynamic UI.
- **Database:** Must use **MySQL** as the primary database for structured data storage.
- **Authentication:** Must use **JWT (JSON Web Token)** for secure user authentication.

Glossary

1. **Model-View-Controller (MVC)** = A software design pattern that separates an application into three interconnected components:
- Model: Represents the data and business logic.

- View: Handles the user interface (UI) and presentation.

- Controller: Manages user input and updates the model and view accordingly.

2. **RESTful API** = A Representational State Transfer (REST) API follows a set of constraints to allow web services to communicate over HTTP.

3. **JSON Web Token (JWT)** = A compact, secure way to transmit information between parties as a JSON object. Used for user authentication and authorization in web applications.

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

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

Deliverable 2

Domain Model

[Define the domain model and create the conceptual class diagrams]

Architectural Design

Conceptual Architecture

[Define the system's conceptual architecture; use an architectural style and pattern - highlight its use and motivate your choice.]

Package Design

[Create a package diagram]

Component and Deployment Diagram

[Create the component and deployment diagrams.]

Deliverable 3

Design Model

Dynamic Behavior

[Create the interaction diagrams (2 sequence) for 2 relevant scenarios]

Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

Data Model

[Create the data model for the system.]

System Testing

[Describe the testing methides and some test cases.]

Future Improvements

[Present some features that apply to the application scope.]

Conclusion

Bibliography