Tribhuvan University

Institute of Science and Technology

A Project Report on

" TravelApp -Travel Destination Management System "

For

Software Engineering (CSC375)

Submitted to:

Department of Computer Science and Information Technology

Asian College of Higher Studies

Ekantakuna, Lalitpur

In partial fulfillment of the requirements

For the Bachelors of Science in Computer Science and Information Technology

Submitted by:

Dipesh Pun (23081047)

Nishchal Acharya (23081039)

Sandip Kumar Shah (23081029)

Sulav Adhikari (23081003)

Sixth Semester

September 2025

Under the Supervision of:

Mr. Bidur Sapkota

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by **Dipesh Pun, Nishchal Acharya, Sandip Kumar Shah** and **Sulav Adhikari** entitled "**TravelApp - Travel Destination Management System**" in partial fulfilment of the requirements for the Software Engineering subject be processed for evaluation

.....

Mr. Bidur Sapkota

Supervisor

Letter of approval

This is to certify that this project is prepared by **Dipesh Pun, Nishchal Acharya, Sandip Kumar Shah** and **Sulav Adhikari** entitled "**TravelApp -Travel Destination Management System**" in partial fulfilment of the requirements for the degree of **Bachelor in Computer Science and Information Technology** has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

....

Mr. Bidur Sapkota
ACHS

External Examiner

ACKNOWLEDGEMENTS

We are grateful to ACHS College for providing us with the ICT infrastructure and a friendly environment that was essential to the success of our project. We would also like to thank **Mr. Bidur Sapkota, Lecturer**, for providing necessary feedback and guidance in the development of our project *"TravelApp-Travel Destination Management System"* for Software Engineering. He has been a great motivator and helper, encouraging us to keep moving ahead on the project. Last but not the least, our sincere thanks go to everyone who has helped directly and indirectly to complete this project.

With respect,

Dipesh Pun

Nishchal Acharya

Sandip Kumar Shah

Sulav Adhikari

ABSTRACT

TravelApp is a secure backend development project built using Java and Spring Boot, delivering RESTful CRUD APIs for managing personal travel destinations. The system enables user registration and authentication through JWT, ensuring protected access to all endpoints. Data is persistently stored in MySQL, while Swagger UI provides interactive testing and real-time API documentation without requiring a frontend interface.

The development process emphasized version control with Git and GitHub, alongside task tracking via Trello for efficient collaboration. The project showcases modular design, robust security practices, and maintainable code structure, demonstrating a complete lifecycle of secure backend development.

Keywords: *CRUD API, Java, Spring Boot, MySQL, JWT, Swagger, Testing, Git, GitHub, Trello, Secure Backend Development*

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

API	Application Programming Interface
CASE	Computer-Aided Software Engineering
CRUD	Create, Read, Update, Delete
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
JPA	Java Persistence API
JWT	JSON Web Token
REST	Representational State Transfer
UI	User Interface

TABLE OF FIGURE

List of Tables

1. Introduction

1.1. Background

Travel management applications have transformed how individuals plan and organize their trips by offering convenient tools to explore and save destinations. With the growing reliance on web services, there is a clear need for secure, reliable, and scalable backend systems to support such applications. Our TravelApp-Travel Destination Management System addresses this need by providing a simple, secure backend built with RESTful APIs and JWT (JSON Web Token) authentication. [1]

The system allows registered users to create, view, update, and delete their personal travel destinations through basic CRUD operations, ensuring full control over their travel lists. JWT-based authentication protects all API endpoints, preventing unauthorized access and ensuring that only authenticated users can manage their data. By integrating Spring Boot, MySQL, and Swagger UI, the project delivers a clean, functional, and testable backend without a frontend. This focused approach emphasizes core functionality, security, and clarity, making it an ideal foundation for personal travel planning tools that are easy to use, maintain, and extend in the future.

1.2. Objectives

The primary objective of this project is to develop a simple, secure, and functional Travel Destination Management System backend that enables safe travel planning through the following integrated capabilities:

- Full CRUD operations on travel destinations with user-specific access control
- Secure user authentication using JWT with token generation, validation, and expiration
- Persistent storage of user preferences and destination data using MySQL and BCrypt password hashing
- Interactive API documentation and testing via Swagger UI without requiring a frontend

- Modular, maintainable architecture built with Spring Boot, Git, GitHub, and Trello for collaboration and future extensibility

1.3. Scope and Limitations

a. Scope:

- The project focuses on backend development only, handling data processing and server-side logic without a graphical user interface.
- Implements secure CRUD operations and user authentication to ensure data integrity and controlled access.
- Provides API documentation for developers to easily integrate the backend with web or mobile frontends.
- Designed for future scalability, allowing integration with cloud services, analytics, or a complete full-stack system.

b. Limitations:

- The system does not include any user interface or frontend, limiting interaction to backend operations only.
- Advanced search and filter functionalities are not implemented in the current version.
- There is no integration with maps or external third-party APIs at this stage.
- The project is restricted to core backend features, without additional enhancements like analytics or real-time updates.

1.4. Development Methodology

To develop our TravelApp - Travel Destination Management System, we chose to use the Waterfall Methodology because it is a good fit for our project's goals and requirements. The Waterfall Methodology provides for a phased-based plan that can help design a secure backend system in a linear and structured manner through the stages of requirements gathering, design, implementation, testing, deployment, and maintenance. Each phase followed a sequential process and could not begun until the phase was fully complete.

We chose to use the Waterfall model for developing the backend of this travel management system because of its linearity and sequential approach, which align more closely with our requirements that we know will not change and the simplicity of the architecture. The Waterfall model allowed us to move through the distinct phases of requirements gathering, system design, implementation, testing, deployment, and maintenance fully documenting the output of each phase prior before starting the following phase.

This style is fitting for our system because we need the security, reliability of the API, and integrity of the data to be verified before continuing forward. The Waterfall model approach also provided the opportunity to ensure that there was thorough documentation at each phase which maintained clarity on the architecture of the system, design choices and functionalities during its lifecycle.

Waterfall Workflow in the TravelApp Project

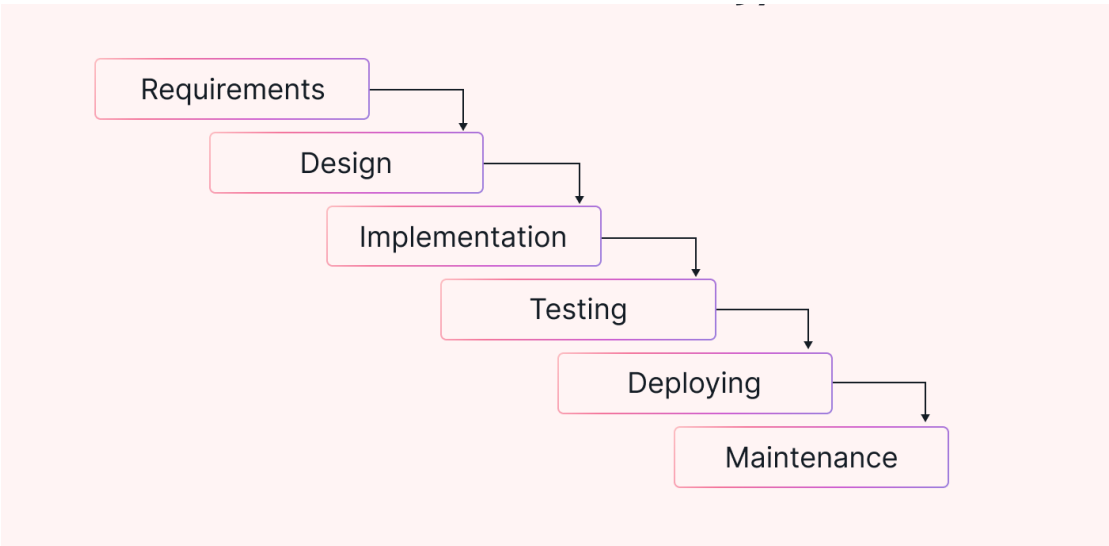

Figure 1 Waterfall Model

Below is a more detailed summary of each section of our TravelApp project using the Waterfall Model:

a. Requirement Analysis

- Functional Requirements: User Registration/Login, JWT Token Generation, Destination CRUD, Mark Destinations as Visited, Search

- Non-Functional Requirements: Secure Authentication, Responsive APIs, Maintainable Code Base, Scalability

b. System Design

- High-Level Design explained the client-server model as the API is the interface, and the backend consists of secured data processing.

c. Implementation

- Authentication (register/login with JWT)
- Destination management (CRUD)
- Swagger UI to document, display, and test APIs

d. Testing

- JUnit tests for controller and service logic
- Validation of user login, token generation, CRUD operations, and unauthorized access rejection
- System testing via Swagger UI with valid/invalid JWT tokens

e. Deployment

The backend was deployed on a local server with a MySQL database.

f. Maintenance

Once our system has been launched, it is designed to accept minor upgrades, fix bugs, and for future improvements, without data or functionality disruption.

1.5. Report Organization

This document outlines the construction of a back-end travel destination management system using Spring Boot. This system employs a security method through JWT and uses REST APIs for the personalized management of destination properties within a travel agency context. The report is organized to demonstrate the system design and the implementation process. The overall report includes a total of 6 chapters.

Chapter 1: Introduction provides a short overview of the project, problem statement, objectives, scope, limitations.

Chapter 2: Background Study and Literature Review discuss background/overview, related research, and current systems.

Chapter 3: System Analysis describes the functional/non-functional requirements, high-level feasibility study (technical, operational, economic, schedule).

Chapter 4: System Design shows system design from the conception of the system, including use case diagram, flowchart, and algorithm details.

Chapter 5: Implementation and Testing outlines tools and used to implement the system, testing, and results analysis.

Chapter 6: Work Done and Remaining Work outlines features that were or were not implemented together with the prioritization of remaining features and work upcoming on the project.

2. Background Study and Literature Review

2.1. Background Study

Travel planning has evolved from physical maps and handwritten notes to digital tools that help users organize their dream destinations. While many apps focus on searching or booking, few offer a simple, secure space to personally save, track, and manage travel wishlists. Our TravelApp-Travel Destination Management System fills this need with a clean, backend-only solution using RESTful APIs and JWT authentication to ensure secure access. [3]

The system enables users to perform basic but essential operations adding a destination with notes, viewing their list, updating details, or removing places all tied to their authenticated account. Built with Spring Boot and MySQL, and documented via Swagger UI, it provides a lightweight, reliable, and testable backend without a frontend. This focused design prioritizes security, clarity, and extensibility, making it an ideal foundation for personal travel tools that can later connect to mobile apps, websites, or travel blogs. [4]

2.2. Literature Review

Travel planning apps have transformed user behavior, but many suffer from weak security or poor data isolation. Studies reveal that inadequate authentication and session management expose personal travel preferences, while inconsistent CRUD operations lead to data loss or confusion. [5] Our TravelApp counters these issues with JWT-based stateless authentication and user-specific data control, ensuring only the owner can access or modify their destination list. This secure, minimal approach stands in contrast to feature-heavy platforms that sacrifice backend clarity for complexity. [6]

Modern backend development favors Spring Boot for rapid, scalable API creation with built-in security and database support. MySQL offers structured, reliable storage for user and destination data, while JWT enables efficient, session-free authentication reducing server load and enhancing scalability. [7] Swagger UI is widely adopted for generating interactive API documentation, allowing developers to explore endpoints and test requests in real time. [8] Collaborative tools like Git, GitHub, and Trello streamline version control and task management, ensuring code quality and team coordination throughout development. [9]

Following established software engineering principles, the project uses REST for predictable, resource-based APIs and the Waterfall methodology for structured, phase-by-phase progress from requirements to testing and documentation. JUnit validates core functions like login, token verification, and CRUD operations. [10] By integrating these proven technologies and practices, TravelApp delivers a secure, simple, and well-documented backend proving that a focused system with strong authentication, clean data handling, and clear APIs can reliably support personal travel management and serve as a solid base for future full-stack applications. [11]

3. System Analysis

3.1. System Analysis

Performing a system analysis for TravelApp involves examining aspects to ensure components meet user needs and goals.

3.1.1. Requirement Analysis

i. Functional Requirement

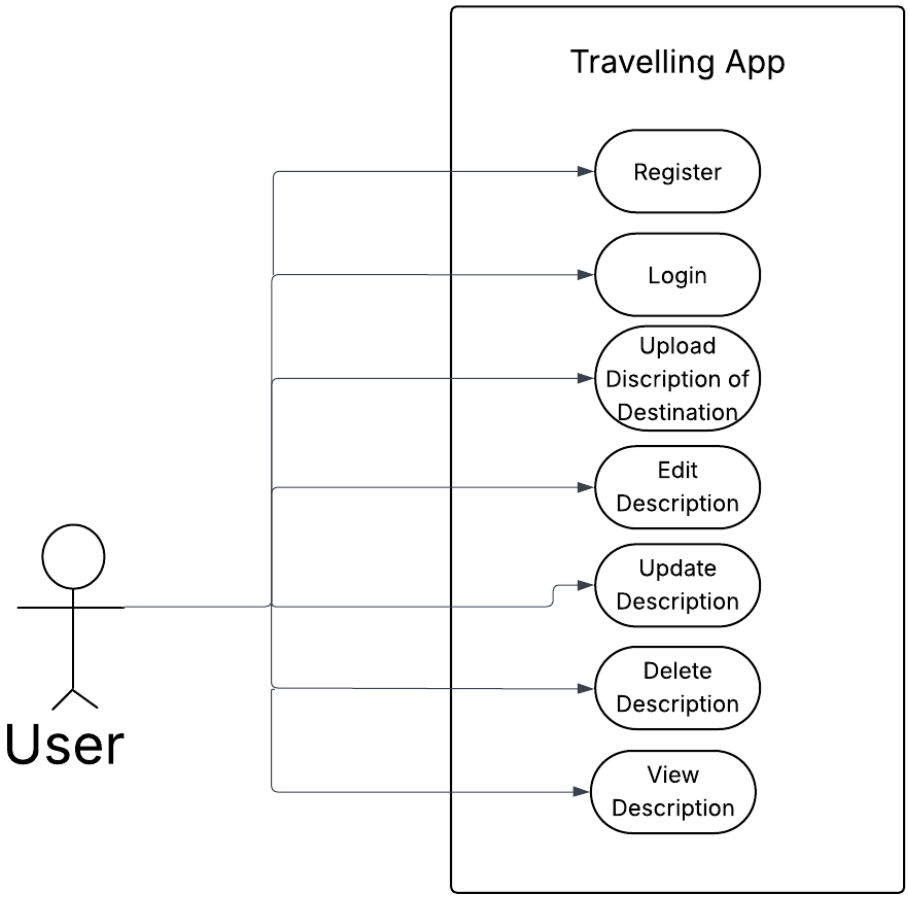

Figure 2 Use Case Diagram of TravelApp

The use Case Diagram shows how the User interacts with the TravelApp system. The User registers with a username, email, and hashed password, then logs in to receive a JWT. This token is required for all secure requests, ensuring only the authenticated User can access their data.

Once logged in, the User can manage destinations via CRUD operations create, read, update, and delete personal entries. The "Mark as Visited" action extends Update, toggling a flag. All operations are User-isolated using the JWT, with include and extend relationships keeping the model clear and focused.

ii. Non-Functional requirement

a. Performance

APIs will be written with a target response time of two seconds or less, under normal load to ensure quality and smooth and efficient user interactions with the application. This will be achieved using optimized queries and lightweight Spring Boot services that can sustain quick response times.

b. Scalability

The system supports horizontal scalability, allowing additional servers or cloud instances to handle higher traffic. This ensures consistent performance even as users and data increase.

c. Maintainability

The system supports horizontal scalability using additional servers or subscription-based cloud computing to handle larger traffic load. Thus, the actions can still sustain good performance and smooth interactions as the number of users and data grows.

d. Security:

Security is ensured through JWT-based authentication and encrypted password storage. Additional measures like input validation and API protection safeguard against unauthorized access.

3.1.2. Feasibility Analysis

Feasibility Analysis looks at whether a project is feasible in technical, economic, legal, operational, and schedule terms.

a. Technical Feasibility

The project is technically feasible due to the use of Spring Boot, MySQL, and JWT authentication, all of which are stable, thoroughly documented, and supported by large communities of developers.

b. Operational Feasibility

The project is operationally feasible in that the project follows a defined development process, but testing is done under a controlled and fixed environment to ensure reliability and functionality prior to launching.

c. Schedule Feasibility

The project is schedule feasible in that works were broken down into milestones to be tracked using a Gantt chart. The chart monitored overall progress and ensured the appointed works were completed on time.

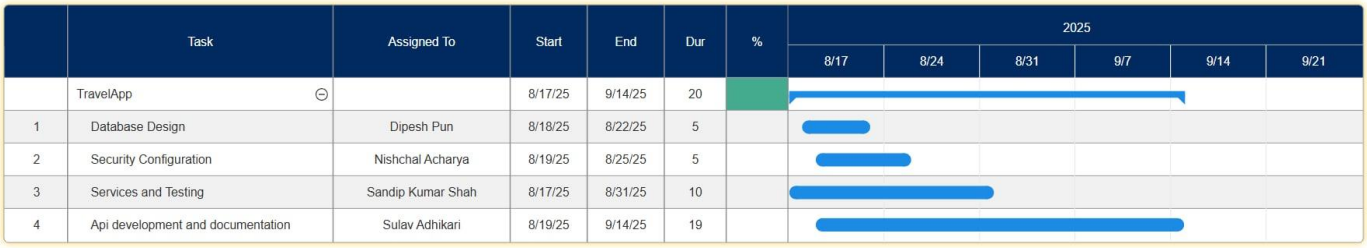

Figure 3 Gantt Chart of TravelApp Project Schedule

The Gantt chart illustrates the project timeline for the TravelApp development, outlining the key tasks, responsible team members, and their respective durations. The overall project runs from August 17, 2025, to September 14, 2025, spanning a total of 20 days. The chart visually presents each task's start and end dates, showing overlapping and sequential activities. The main project goal, "TravelApp," acts as the parent task, with four sub-tasks distributed among different team members: database design, security configuration, services and testing, and API development and documentation.

In detail, Dipesh Pun is responsible for Database Design (August 18–22, 2025), lasting 5 days. Nishchal Acharya handles Security Configuration (August 19–25, 2025), also for 5 days. Sandip Kumar Shah is assigned Services and Testing (August 17–31, 2025), taking 10 days, while Sulav Adhikari manages API development and documentation (August 19–September 14, 2025), lasting 19 days. The blue progress bars in the Gantt chart indicate the timeline overlap and task dependencies, reflecting an organized workflow where multiple processes occur concurrently to ensure efficient project completion.

3.1.3. Analysis

a. Data modeling using ER Diagram

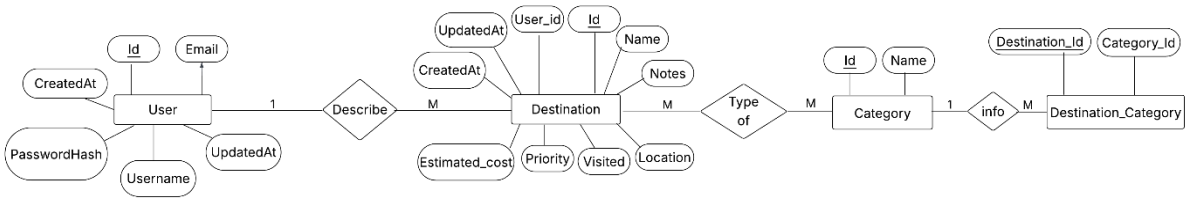

Figure 4 ER Diagram of TravelApp

The ER Diagram shows the TravelApp data structure with three core entities: User, Destination, and Category. User stores Id, Username, PasswordHash, Email, and timestamps, linking to multiple Destinations via User_Id in a one-to-many relationship. Each Destination includes Name, Notes, Type, Location, Priority, Visited, Estimated_Cost, and timestamps, and connects to Category through the Destination_Category junction table to support many-to-many tagging. Category holds Id and Name for grouping destinations. This clean, normalized design ensures secure user isolation, flexible categorization, and efficient querying in MySQL.

b. Context Diagram

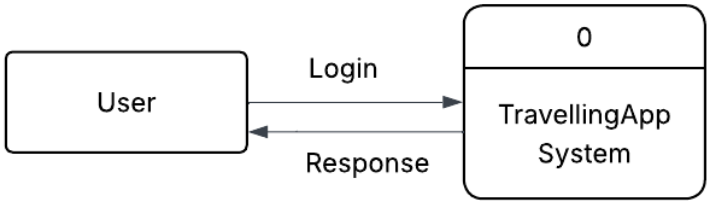

Figure 5 Context Diagram of TravelApp

The Context Diagram shows the TravelApp backend as a single system. It connects only to the User through a two-way link. The user sends requests to register, log in,

and manage destinations, while the system returns tokens, lists, and updates. This simple design keeps the backend secure, focused, and easy to extend.

c. Process Modelling using DFD level 0

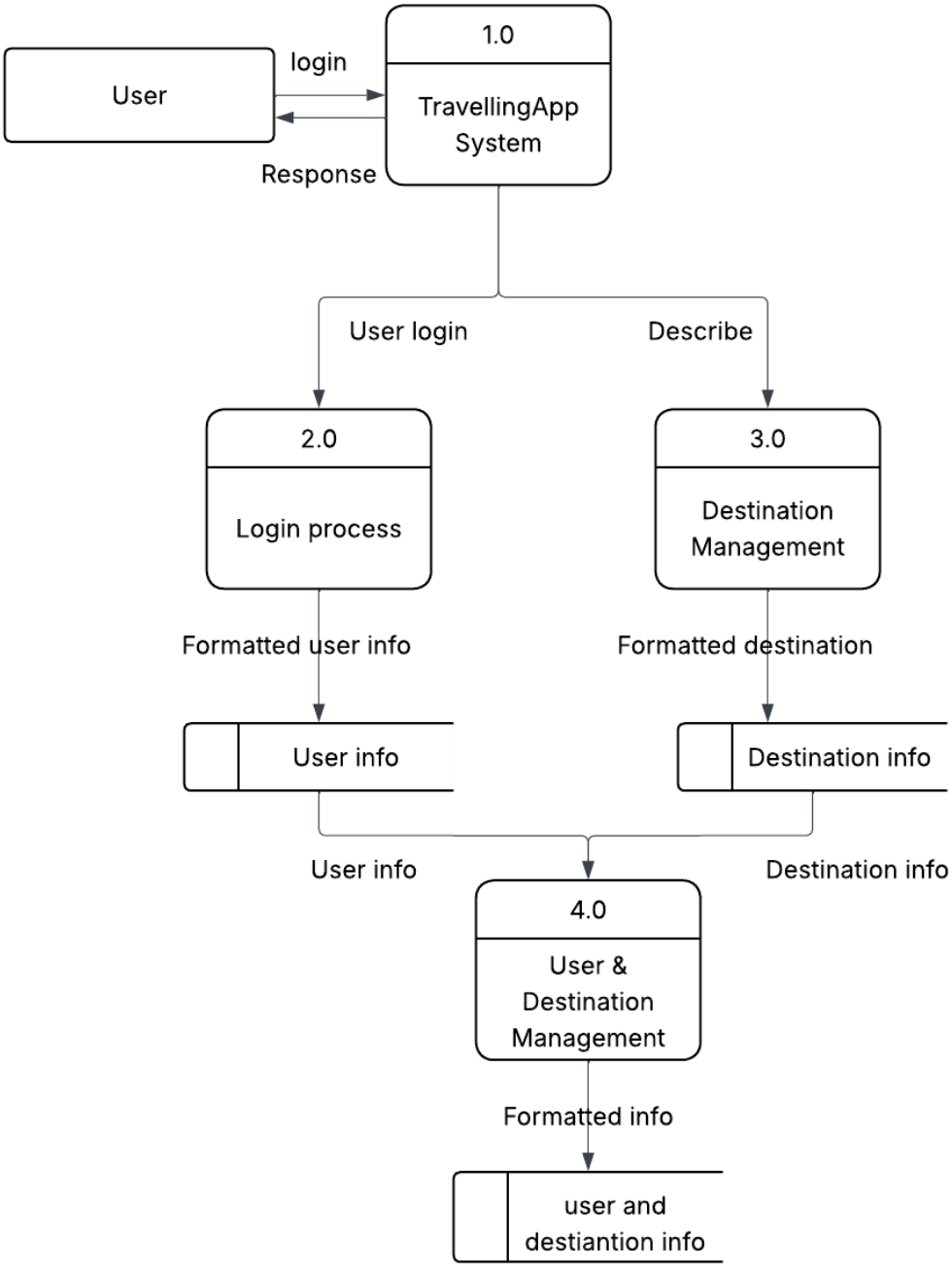

Figure 6 DFD level 0 of TravelApp

The Level 0 Data Flow Diagram (DFD) of the TravellingApp System illustrates the main functional components and how data flows within the system. The central process, TravellingApp System (1.0), interacts with external entities to manage travel-related operations. It is divided into two major subprocesses: Login Process (2.0) and

Destination Management (3.0). The Login Process manages user authentication, collecting and storing User Info, while the Destination Management process focuses on handling Destination Info, such as travel locations, details, and preferences.

Both user and destination data are then processed in User & Destination Management (4.0), which integrates this information to create personalized travel recommendations and maintain synchronized records. The output, labeled as User and Destination Info, represents the system's final organized data that can be used for tailored suggestions or further analysis. This DFD provides a clear, high-level overview of how different components of the TravellingApp interact and how data moves efficiently between them to ensure secure and personalized travel management.

4. Design - Flowchart of the System

4.1. Design - Flowchart of the System

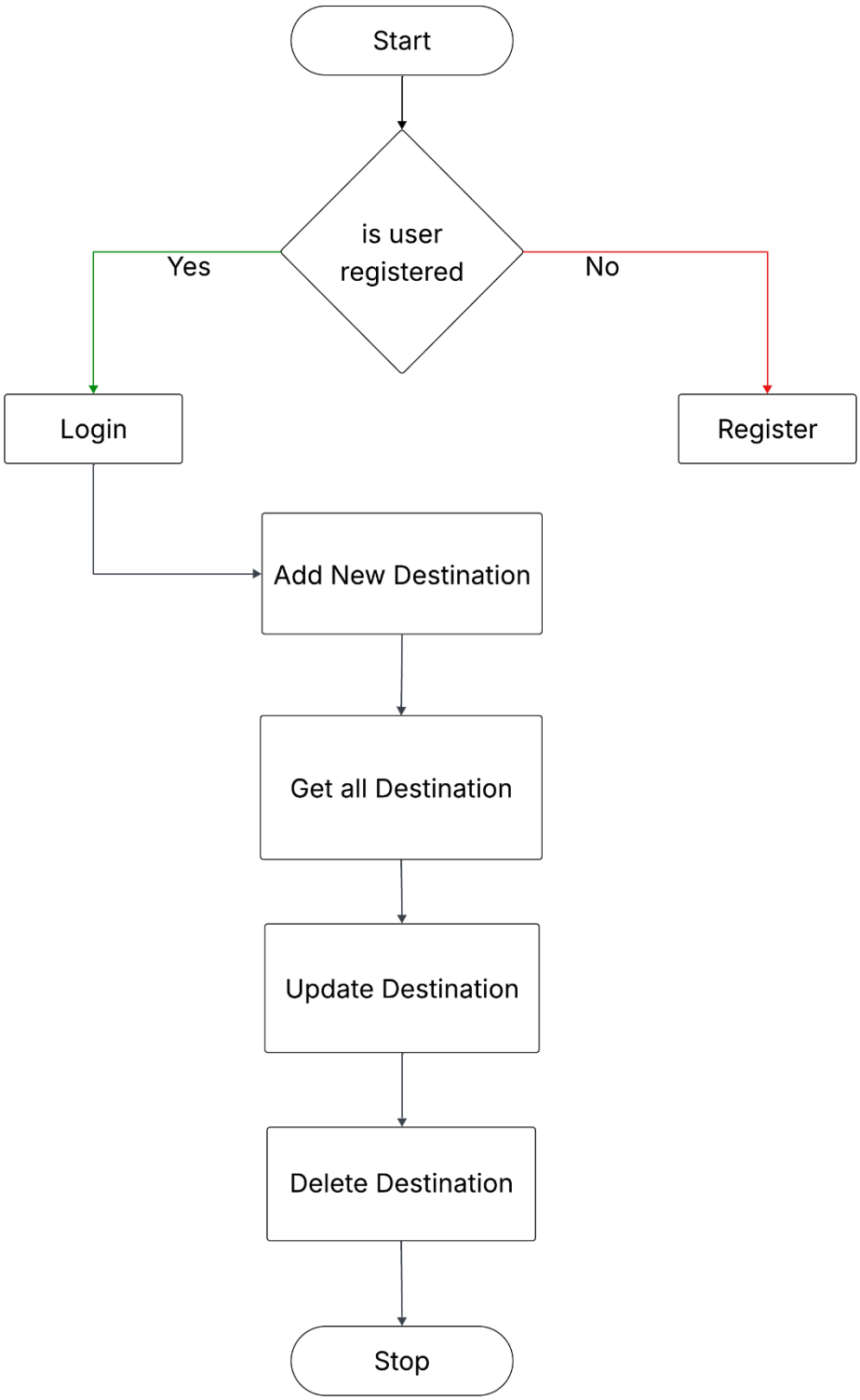

Figure 7 Flowchart of TravelApp

TravelApp process begins with a registration check: "Is user registered?". If No, the user Registers by submitting a username, email, and password — securely hashed with BCrypt and stored in MySQL. If yes, the user Logs in using valid credentials and receives a JWT token for secure access.

Once authenticated, the user proceeds to destination management: Add New Destination to save a place, get all Destination to view their list, Update Destination to edit or mark as visited, and Delete Destination to remove entries. All operations require the JWT for authorization and are restricted to the user's own data. The flow concludes at Stop when the session ends or the user logs out.

4.2. Class Diagram

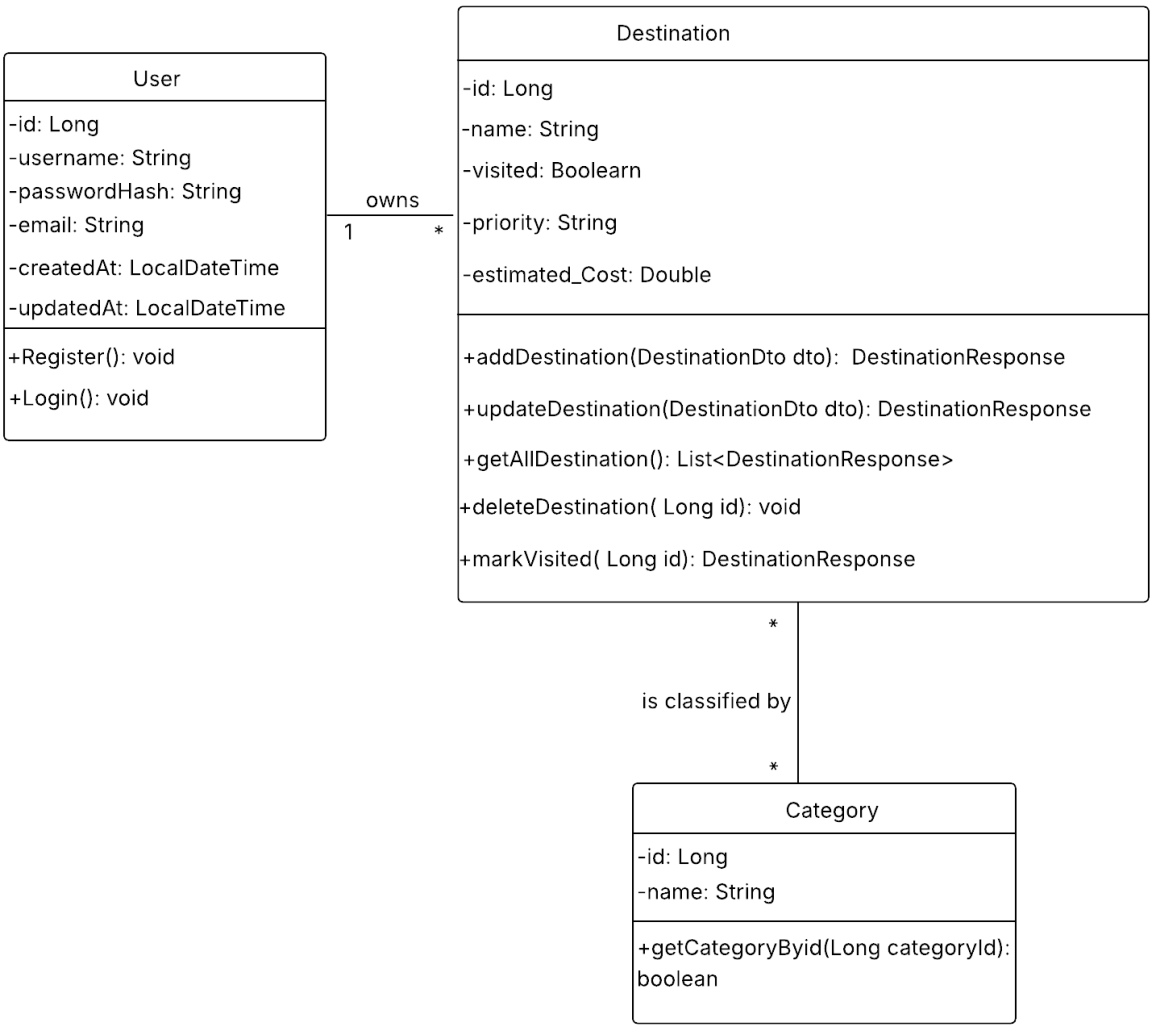

Figure 8 Class Diagram of TravelApp

The Class Diagram presents the core structure of the TravelApp backend, showing three main entities: User, Destination, and Category, along with their attributes, methods, and relationships. The User class manages authentication and profile data with fields like id, username, passwordHash, email, createdAt, and updatedAt. It includes two key methods: Register () to create a new account with secure password hashing, and Login() to authenticate and generate a JWT token for session access.

The Destination class represents a travel place saved by the user, containing id, name, visited (boolean), priority, and estimated_Cost. It supports full CRUD operations via methods such as addDestination(), updateDestination(), getAllDestination(), deleteDestination(), and markVisited(). These ensure users can manage their personal travel list securely. The Category class allows grouping destinations with id and name, and includes getCategoryById() to fetch category details. This modular, object-oriented design — built with Spring Boot and mapped to MySQL — ensures data integrity, user isolation, and clean separation of concerns, forming a solid, extensible foundation for the backend.

5. Implementation and Testing

5.1. Implementation

5.1.1. Tools Used

1. Case Tools
   a. IDE:
      - IntelliJ IDEA

   b. Diagramming Tools:

      - Lucid Chart for diagrams

   c. Version Control:

      - Git for tracking changes

   d. Framework:

      - Spring Boot

2. Programming Languages

      - Java

3. Database Platforms

      - MySQL: For relational data storage.

4. Other Tools:

      - Swagger and Postman

5.2. Testing

5.2.1. Test Cases for Unit Testing

S.N.	Test Name	Expected Outcome	Actual Outcome	Result
1	Login and Validation	JWT token generated	Token received	Pass
2	Destination Testing	Destination saved	Saved as expected	Pass
3	User Service Test	Updated successfully	Updated	Pass

Table 1 Test Cases for Unit Testing of TravelApp

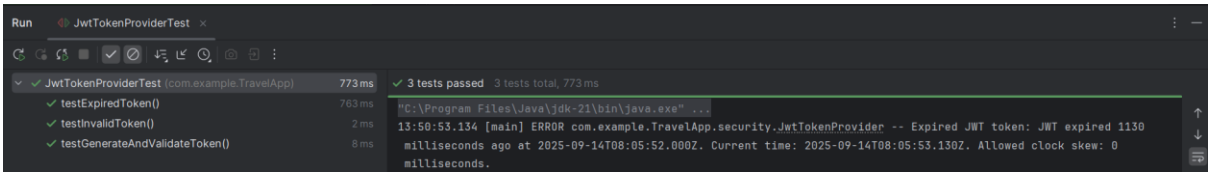

Figure 9 Login and Validation Test

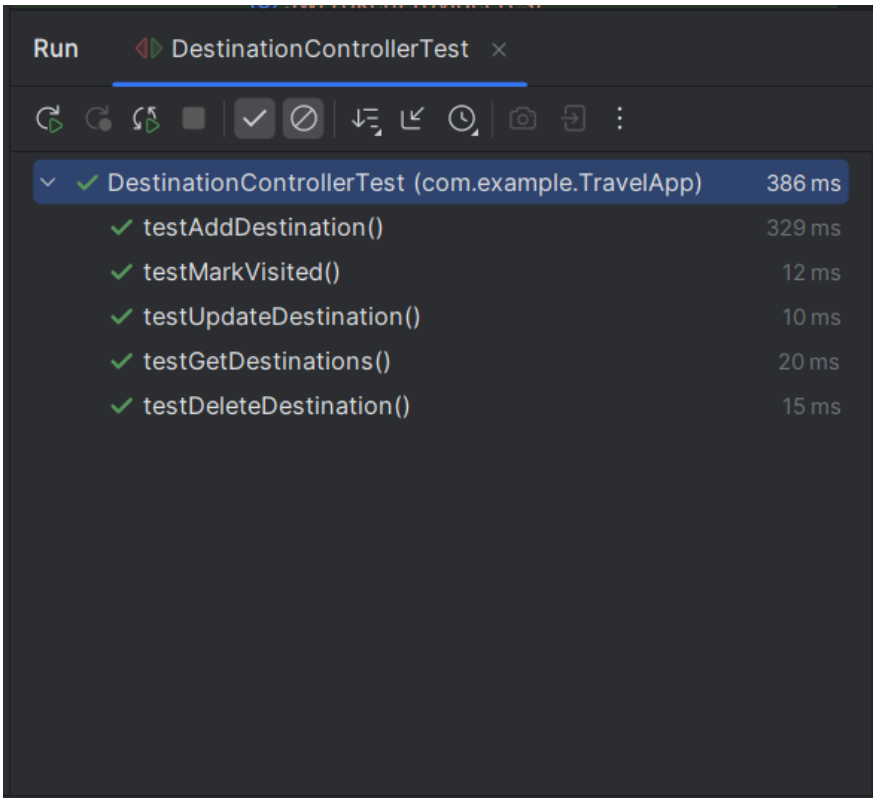

Figure 10 Destination Controller Test of TravelApp

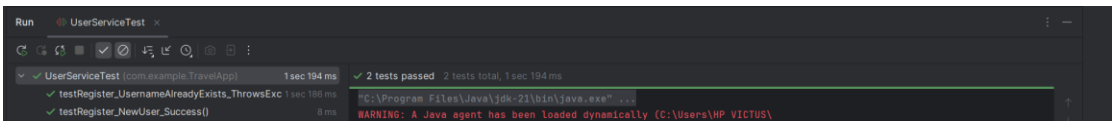

Figure 11 User Service Test of TravelApp

5.3. Result Analysis

The results of the unit tests presented in Table 5.1 indicate that all elements tested (login, registration, destination management, and validation of tokens) functioned correctly, with all tests passing. The APIs were secure, demonstrating the ability to generate and validate JWT tokens, supporting secure authorization and authenticating users. The APIs were performant, handling requests to create, read, and update destinations efficiently. The consistent correlation between expected and actual results indicates reliability and proper functioning. In conclusion, the successful unit tests demonstrate that the system is both secure and efficient in the array of tests conducted.

6. WORK DONE AND WORK REMAINING

6.1. Work Done

- User Registration/Login: Secure with JWT.
- Destination CRUD: Manage lists.
- Mark Visited: Update status.
- Swagger UI: For testing.

6.2. Remaining Work

- Frontend Integration: Add UI.
- Recommendation Engine: Add collaborative filtering.
- Search Feature: For destinations.
- Multi-Device Sync: For better experience.

7. Conclusion

The TravelApp-Travel Destination Management System has successfully developed a secure and optimized backend using Spring Boot, MySQL, and Json web token (JWT) authentication protocol. The project was organized through timeary sequential phases as followed the Waterfall methodology requirements analysis, design, implementation, testing, deployment with all collected artifacts intending to improve transparency, documentation, and reliability. The system allows the user to register and securely log-in, followed by the ability to perform complete CRUD ability on relevant travel destinations all protected endpoints validated with Swagger UI.

This project was successful not only academically, meeting the course requirements for Software Engineering (and a pass), but also creates a solid backend suitable for production. The project included best practices with RESTful API design, security with tokens, and implemented database management, which will lay a good foundation for future work, like a front-end or recommendation functionality.

8. References

[1] Spring Boot Documentation.

https://docs.spring.io/spring-boot/documentation.html

[2] JSON Web Token Introduction. https://jwt.io/introduction

[3] Recommender Systems Applied to the Tourism Industry: A Literature Review. *https://www.researchgate.net/publication/381717210_Recommender_systems_applied_to_the_tourism_industry_a_literature_review*

[4] JWT Security Best Practices. *https://curity.io/resources/learn/jwt-best-practices/*

[5] Building REST API Using Spring Boot: A Comprehensive Guide. *https://medium.com/@pratik.941/building-rest-api-using-spring-boot-a-comprehensive-guide-3e9b6d7a8951*

[6] Everything You Need to Know About the Tripadvisor Popularity Ranking. *https://www.tripadvisor.com/business/insights/resources/tripadvisor-popularity-ranking*

[7] A Collaborative Location Based Travel Recommendation System through Enhanced Recommender System for Smart City Environment.

https://pmc.ncbi.nlm.nih.gov/articles/PMC4812910/

[8] A Study on a JWT-Based User Authentication and API Assessment Scheme Using IMEI in a Smart Home Environment.

https://www.researchgate.net/publication/317865652_A_Study_on_a_JWT-Based_User_Authentication_and_API_Assessment_Scheme_Using_IMEI_in_a_Smart_Home_Environment

[9] Restaurant Recommender System Based on Sentiment Analysis. *https://www.sciencedirect.com/science/article/pii/S2666827021000578*

[10] On the Need for Configurable Travel Recommender Systems.

https://arxiv.org/abs/2407.11575

[11] The Digital Tourist: Examining User Experience in Travel Apps Using Text Mining and Topic Modelling.

https://www.tandfonline.com/doi/full/10.1080/02508281.2025.2545802?src=exp-la

[12] Travel Itinerary Recommendation Using Interaction-Based Recurrent Neural Networks. *https://www.sciencedirect.com/science/article/pii/S0957417424031610*

[13] Satisfaction and User Reviews in Tourism Using Big Data and Text Mining: A Bibliometric Study.

https://www.researchgate.net/publication/387193730_Satisfaction_and_User_Reviews_in_Tourism_Using_Big_Data_and_Text_Mining_A_Bibliometric_Study

APPENDIX

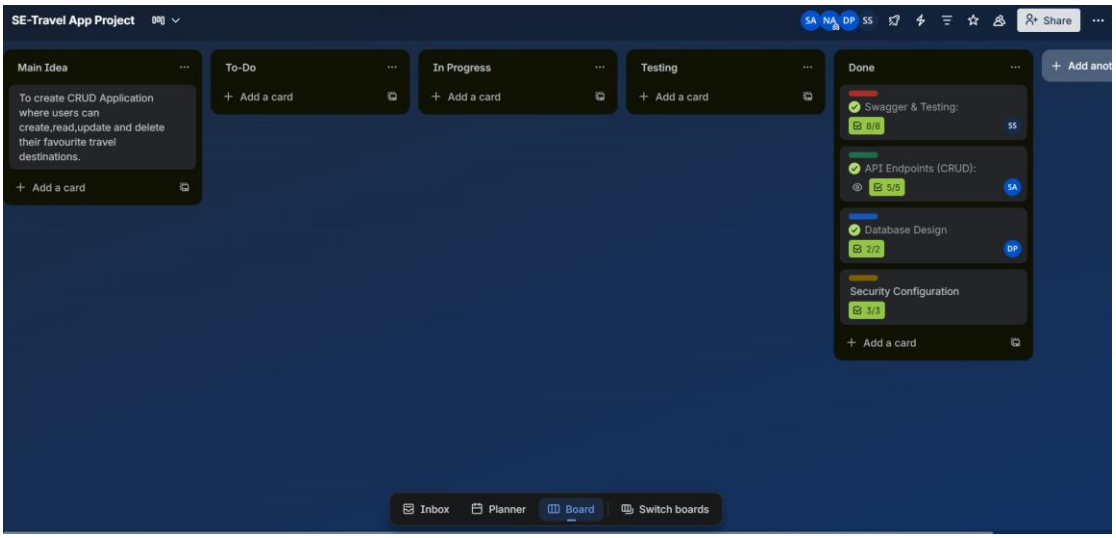

Figure 12 Trello Proof

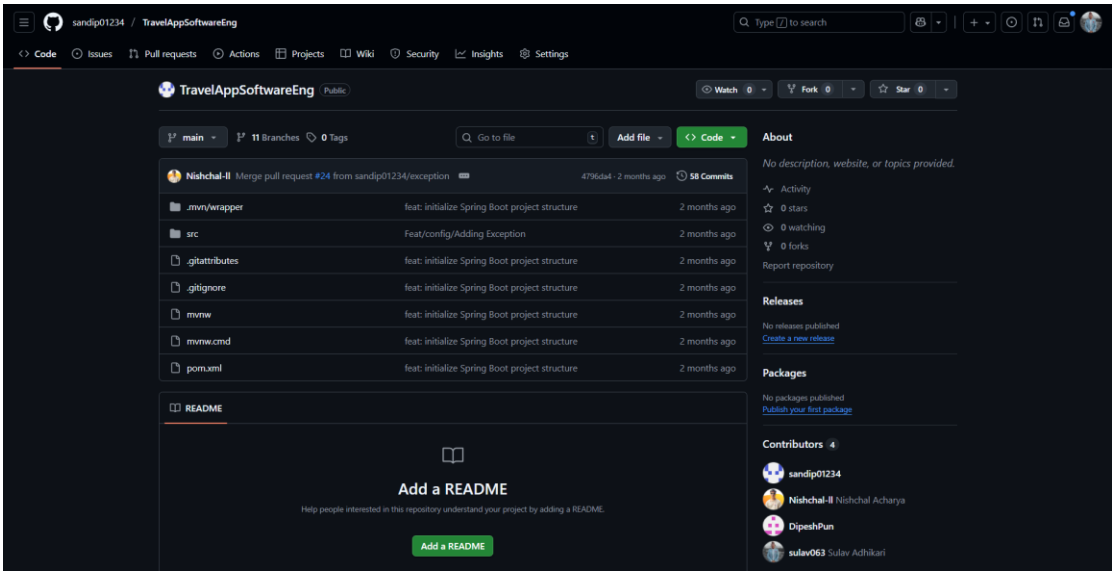

Figure 13 GitHub Proof of TravelApp

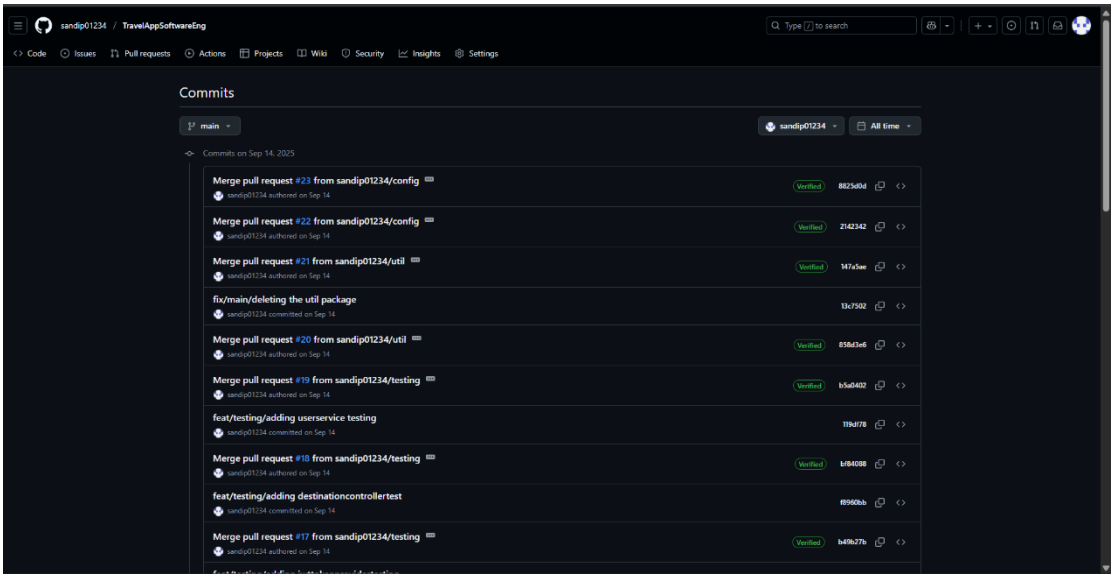

Figure 14 GitHub Commits by Sandip Kumar Shah

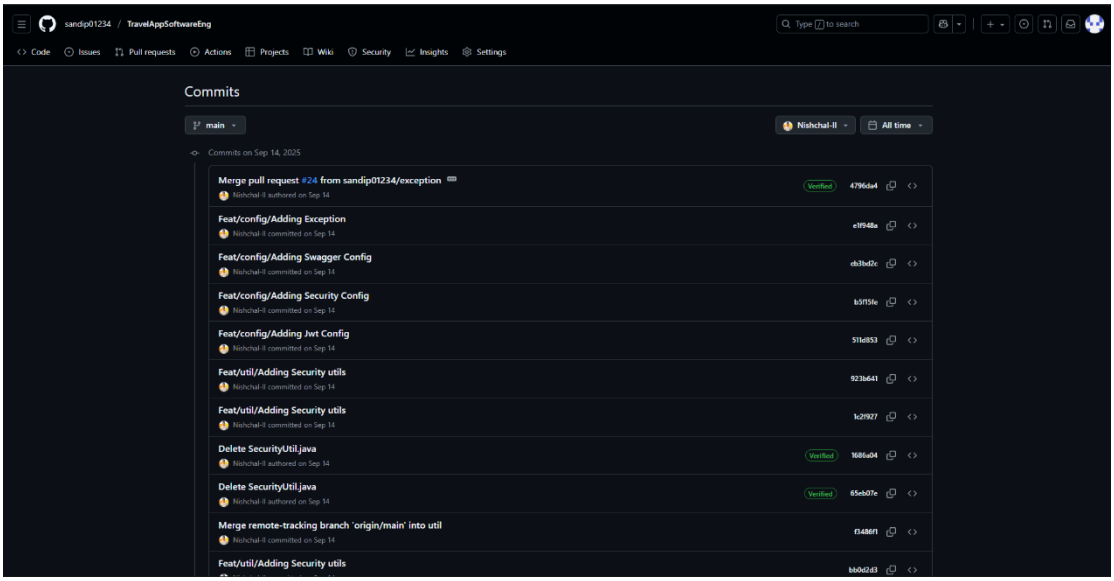

Figure 15 GitHub Commits by Nishchal Acharya

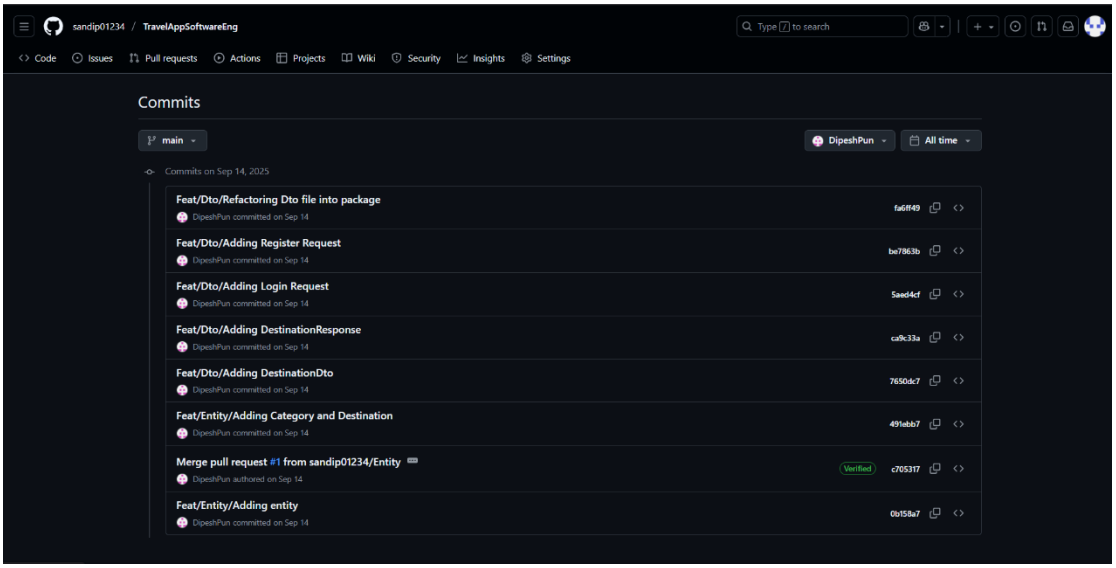

Figure 16 GitHub Commits by Dipesh Pun

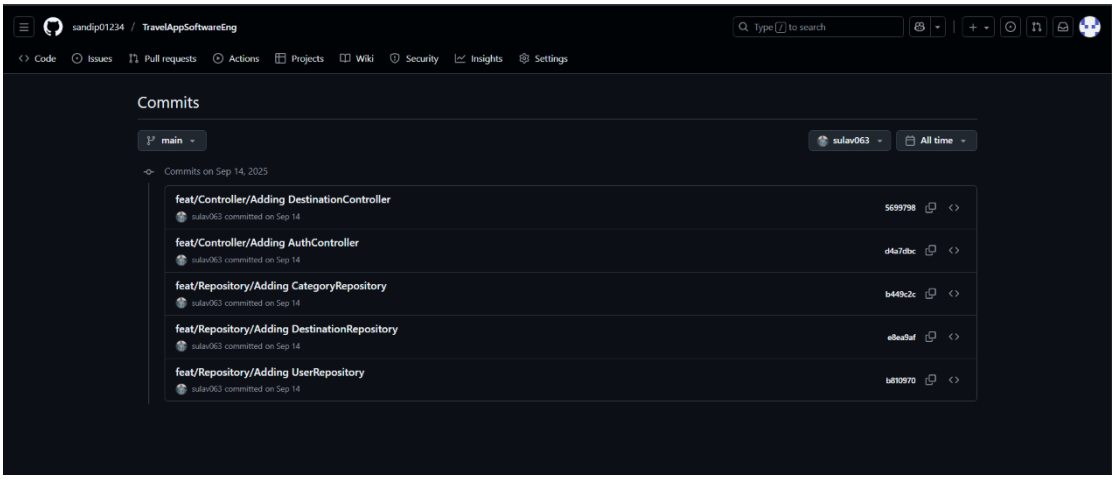

Figure 17 GitHub Commits by Sulav Adhikari

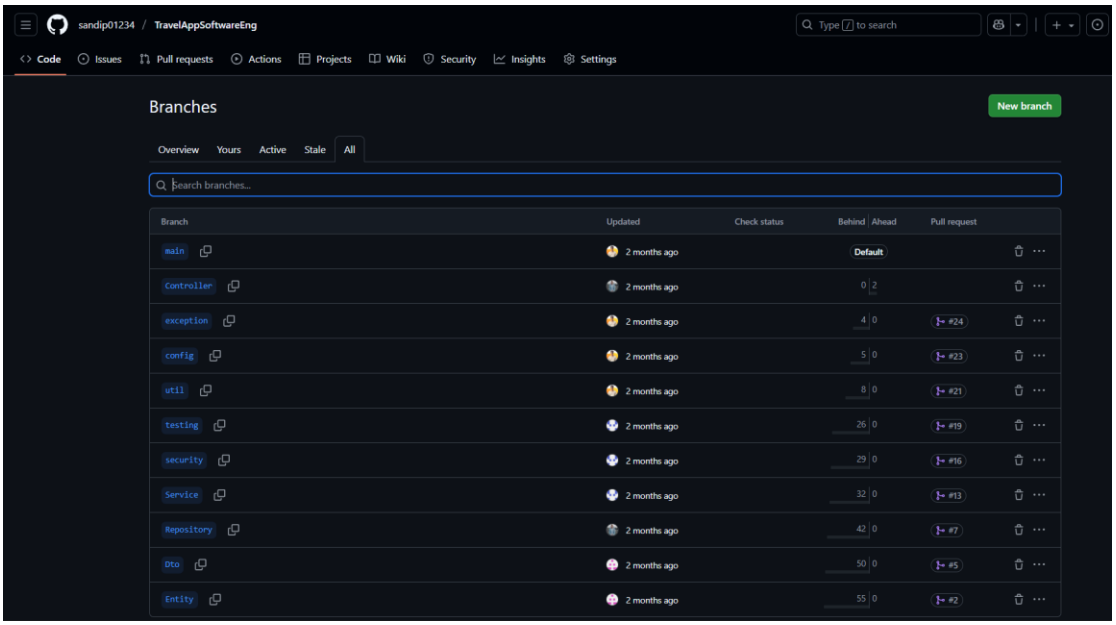

Figure 18 GitHub Branches

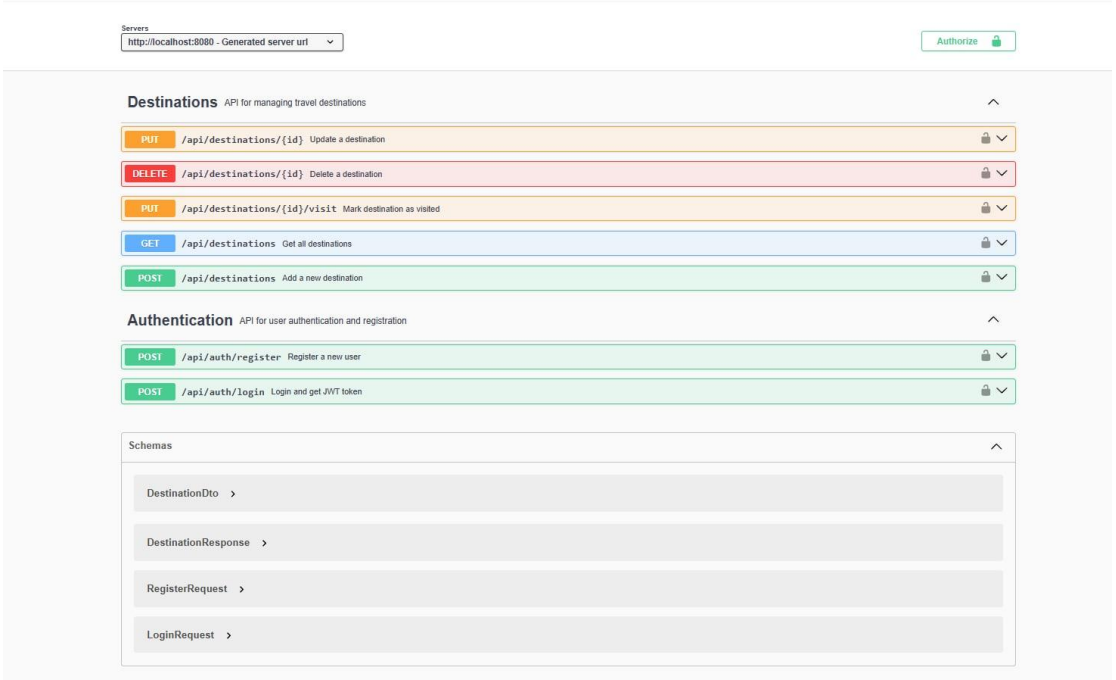

Figure 19 Swagger UI TravelApp