

**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 (23081003)

Nishchal Acharya (23081003)

Sandip Kumar Shah (230810)

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 External Examiner  
ACHS

# **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**

Travel App is a backend-based travel destination management system that allows users to register, authenticate, and manage their travel destinations. The application implements JWT-based security to protect API endpoints. Swagger UI has been integrated to demonstrate API functionality without a front-end. The project demonstrates REST API development, authentication, and CRUD operations on destinations using Spring Boot, MySQL, and JWT.

# **TABLE OF CONTENTS**

[**SUPERVISOR’S RECOMMENDATION** I](#_Toc208476459)

[**Letter of approval** II](#_Toc208476460)

[**ACKNOWLEDGEMENTS** III](#_Toc208476461)

[**ABSTRACT** IV](#_Toc208476462)

[**TABLE OF CONTENTS** V](#_Toc208476463)

[**LIST OF ABBREVIATIONS** VI](#_Toc208476464)

[**TABLE OF FIGURE** VII](#_Toc208476465)

[**List of Tables** VIII](#_Toc208476466)

# **LIST OF ABBREVIATIONS**

API Application Programming Interface

JWT JSON Web Token

CRUD Create, Read, Update, Delete

UI User Interface

ERD Entity Relationship Diagram

DFD Data Flow Diagram

# **TABLE OF FIGURE**

# **List of Tables**

# **Introduction**

# **Introduction**

With the advent of today's digital era, travel management apps have transformed the way users plan and organize their trips. With the rise of web services, backend systems have become essential for delivering secure and scalable travel management. Our TravelApp - Travel Destination Management System uses RESTful APIs with JWT authentication to provide a secure user-focused experience. [1]

The system analyzes user preferences and destinations, allowing CRUD operations on personalized travel lists. JWT ensures secure access to endpoints, preventing unauthorized actions. By integrating these technologies, the application provides a robust backend that evolves with user interactions, offering a more personalized and secure travel planning experience.

This project focuses on developing a simple and functional backend for travel management, emphasizing secure authentication, destination handling, and API documentation via Swagger UI. By prioritizing these core features, it delivers a backend that is both reliable and extensible. [2]

# **Problem Statement**

The aim is to develop a secure, scalable Travel Destination Management System backend. In the fast-evolving digital world, users require seamless and secure travel planning tools to manage destinations, from discovering new places to marking visited ones, with real-time synchronization across sessions. The goal is to create a robust backend that handles data securely with minimal latency via an intuitive API structure.

A key challenge in travel management ecosystems is the lack of secure, personalized destination recommendations and management. Users often feel overwhelmed by unsecured or generic travel data, leading to poor user experiences. Existing systems may not adequately handle dynamic user preferences or secure data access, resulting in privacy concerns or irrelevant suggestions.

To address these, a hybrid approach using JWT for authentication and RESTful APIs for operations provides a comprehensive solution. JWT ensures token-based security, while APIs focus on user-specific data like destinations' titles, descriptions, and visited status. This backend delivers highly personalized and secure travel management that adapts to individual needs.

# **Aims and Objectives**

The main aim of our project is to develop a Travel Destination Management System backend for secure travel planning, and some of our objectives are:

* To create secure APIs for user authentication and destination management.
* To implement JWT-based security and CRUD operations, providing users with personalized travel tools.

# **Scope and Limitations**

Scope:

* Implement JWT authentication and RESTful APIs for destination CRUD.
* Secure user registration, login, and profile management.
* Allow users to manage destinations, mark as visited, and organize lists.

Limitations:

* New users with limited data may receive basic functionality until interactions build up.
* The system relies on MySQL for data storage, limiting offline access without extensions.
* No frontend integration; testing limited to Swagger UI, restricting cross-platform UI experience.

# **Development Methodology**

For the development of our **TravelApp – Travel Destination Management System**, we have implemented the **Agile Methodology** as it is well-suited to the dynamic and iterative nature of our project. Agile allowed us to build the system incrementally through short development cycles (sprints), ensuring continuous improvement and faster delivery of core features.

This approach emphasized collaboration, adaptability, and iterative progress over rigid sequential phases. Agile enabled us to continuously refine functionalities such as **user authentication, destination CRUD operations, JWT-based security, and API performance** while incorporating feedback at each stage of development.

Unlike the traditional linear Waterfall model, Agile provided flexibility to adjust requirements, improve design choices, and validate implementation continuously. The use of **Spring Boot (Java)** streamlined backend development, while **JUnit testing** ensured that each iteration delivered stable and reliable features.

**Agile Workflow in the TravelApp Project**

Figure 1.1 Agile Development Cycle

Here’s an overview of how each Agile phase was applied in our project:

1. Requirement Gathering & Backlog Creation

* Functional Requirements: User registration/login, JWT token generation, destination CRUD, marking destinations as visited, search functionality.
* Non-Functional Requirements: Secure authentication, responsive APIs, maintainable codebase, and scalability.
* Created a product backlog containing prioritized features to be implemented across multiple sprints.

1. Sprint Planning & Design

* Planned short development cycles (sprints) focusing on high-priority features.
* Designed system architecture including REST endpoints, JWT-based security, and MySQL schema.
* Created wireframes and API contracts for clarity among team members.

1. Incremental Development (Spring Boot Implementation)

* Used Spring Boot for rapid development and REST API creation.
* Each sprint delivered incremental features such as authentication, destination management, and API documentation with Swagger.
* Git was used for version control, ensuring collaborative and iterative development.

1. Continuous Testing (JUnit & Integration Testing)

* Applied JUnit for unit testing service classes and controller logic.
* Verified user login, JWT token validation, CRUD operations, and exception handling.
* Conducted integration testing to ensure secure data transactions and API reliability.

1. Review & Retrospective

* At the end of each sprint, reviewed implemented features with the team.
* Collected feedback for improvements and adjusted the backlog for the next sprint.

1. Deployment

* Successfully deployed the backend with stable API performance.
* Ensured compatibility with frontend clients and mobile integration.

1. Maintenance & Iteration

* Post-deployment, handled bug fixes, performance improvements, and added new features.
* Agile allowed us to continuously enhance the system while keeping it stable and secure.

# **Report Organization**

This report presents the development of a backend travel destination management system using Spring Boot. The system utilizes JWT for security and REST APIs for personalized destination management. The report is organized to show the design and implementation. There is 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 covers the background, related research, and existing systems.

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

Chapter 4: System Design provides system design, including use case diagram, flowchart, and algorithm details.

Chapter 5: Implementation and Testing describe tools, testing, and result analysis.

Chapter 6: Work Done and Remaining Work details implemented features and pending ones.

# **Background Study and Literature Review**

# **Background Study**

Recommender Systems (RSs) in travel domains filter information to suggest destinations based on user preferences. [3]

The Travel Destination Management System uses user history for personalized suggestions, while authentication ensures secure access. JWT provides token-based security, enhancing user trust. [4]

Secure APIs give the system a structured way to manage data without exposing sensitive information. This relies on user interactions for personalization. [5]

Analysis of platforms like TripAdvisor shows key features: authentication, destination management, and integrations. Design emphasizes secure, user-friendly APIs. [6]

# **Literature Review**

Under the study of various researchers, Recommender systems in travel use content-based and collaborative methods. Hybrid approaches improve accuracy by combining user data and security. [7]

Studies on JWT reveal its role in secure API authentication, reducing session overhead while handling token validation effectively. [8]

Papers on RESTful APIs with Spring Boot highlight sentiment analysis and natural language for better recommendations, improving accuracy by 80%. [9]

Improved algorithms integrate clustering for personalized recommendations, outperforming basic methods. [10]

Interactive web players for travel apps provide superior experiences but require engagement. [11]

Platforms like TripAdvisor offer rankings and user interactions for seamless functionality. [12]

Studies on user satisfaction use text mining to identify preferences not captured in surveys. [13]

# **System Analysis**

# **System Analysis**

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

# **Requirement Analysis**

1. Functional Requirement

Figure 3.1 Use Case Diagram of the System

The use case diagram shows interactions between User and System. Users register, login, manage destinations, mark visited. The System handles JWT authentication and CRUD.

1. Non-Functional requirement
2. Performance

The APIs are designed to respond within two seconds under normal load, ensuring smooth and efficient user interactions. Optimized queries and lightweight Spring Boot services help maintain fast response times.

1. 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.

1. Maintainability

With modular coding practices and layered architecture, the system allows easy updates without disrupting existing data. JUnit tests and proper documentation further enhance maintainability.

1. Security:

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

# **Feasibility Analysis**

Feasibility Analysis assesses practicality in technical, economic, legal, operational, schedule terms.

1. Technical Feasibility

The project is technically feasible since it uses Spring Boot, MySQL, and JWT, all of which are reliable, well-documented, and supported by large developer communities.

1. Operational Feasibility

The system is operationally feasible as it follows a structured development process with testing in controlled environments, ensuring reliability before deployment.

1. Schedule Feasibility

The project is schedule feasible as tasks were **broken into milestones and tracked using a Gantt chart,** ensuring timely progress and completion.

Figure 3.2 Gantt chart

# **Analysis**

1. Figure 3.3 Data modeling using ER Diagram

ER diagram shows User and Destination entities with one-to-many relationship. User has id, username, password, email. Destination has id, user\_id, title, description, visited.

1. Context Diagram

Figure 3.4 Context Diagram

Overview of user interactions with secure APIs for destination management.

1. Process Modelling using DFD level 1

Figure 3.5 DFD level 1

Processes: Register/Login (1.0), Manage Destinations (2.0), Mark Visited (3.0), Secure Access (4.0). Databases: User Info, Destination Data.

# **Design - Flowchart of the System**

# **Design - Flowchart of the System**

Figure 4.1 Flowchart of the System

The process starts with registration/login, generates JWT, allows CRUD on destinations, validates tokens.

# **Algorithm Details**

Travel systems personalize experiences using secure algorithms. JWT algorithm generates tokens based on user credentials.

We implemented JWT for authentication. Below is a user-destination matrix (rows: users, columns: destinations, 1: visited):

Figure 4.2 User-Destination Interaction Matrix

# **Implementation and Testing**