TRAVEL GUIDE

Abhijit Mohanta, 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India
amohanta2021@gift.edu.in
Ashish ku Mishra, 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India akmishra2021@gift.edu.in
Prof. Smruti smaraki sarangi, Assistant Professor, Department of CSE, Gandhi Institute for Technology, BPUT, India

Abstract
The Travel Instructor is a web-based travel planning platform designed to provide a seamless and user-friendly experience for exploring, reviewing, and booking travel destinations. Built using Java Servlets, JSP, Hibernate, and a responsive front-end with CSS and JavaScript, the application offers secure user authentication, dynamic destination browsing with search and filter capabilities, a review system for user feedback, and a booking module with real-time price calculation. Key features include session-based authentication, detailed destination pages with highlights and reviews, and a booking system supporting customizable options like guided tours and car rentals. The system leverages a relational database for data persistence and follows the MVC architecture for modularity. While the core functionalities are robust, future enhancements aim to address security concerns (e.g., password hashing), integrate external APIs (e.g., OpenTripMap, Unsplash), and add advanced features like payment gateways and user dashboards. Targeted at travel enthusiasts, Travel Instructor combines functionality and accessibility, laying a strong foundation for a scalable travel planning solution.

1. INTRODUCTION
The Travel Instructor is a sophisticated, user-centric web application designed to revolutionize the travel planning experience by integrating secure authentication, dynamic destination browsing, community-driven reviews, and a streamlined booking system into a single, cohesive platform. Targeting a diverse global audience—travel enthusiasts, casual tourists, budget-conscious travelers, and users with accessibility needs—the application addresses the fragmented nature of travel planning, where users typically navigate multiple platforms for research, feedback, and reservations. Built using a robust technology stack of Java Servlets, JavaServer Pages (JSP), Hibernate ORM, CSS, and JavaScript, the platform adheres to the Model-View-Controller (MVC) architectural pattern, ensuring modularity, scalability, and maintainability. As of May 12, 2025, the prototype represents a fully functional implementation of core functionalities, delivering a responsive, intuitive interface that caters to modern travel demands.

2. LITERATURE REVIEW
Buhalis and Law (2008) trace the evolution of e-tourism, noting that platforms like TripAdvisor and Expedia integrate destination data, reviews, and bookings. However, fragmentation persists, as users must navigate multiple interfaces (Gretzel et al., 2015). Travel Instructor addresses this by consolidating functionalities, reducing user effort.
Travel Instructor aligns with this paradigm by integrating destination browsing (home.jsp, destination.jsp, DestinationDao.java) and review systems (ReviewController.java, ReviewDao.java) into a single platform. Its search and filter functionality (DestinationDao.searchDestinations) mirrors features in TripAdvisor, while the booking system (booking.jsp, BookingController.java) draws inspiration from Expedia's reservation model. However, unlike these platforms, Travel Instructor's prototype focuses on a lightweight, modular architecture using JSP and Hibernate, avoiding the complexity of enterprise-scale systems.

3. SYSTEM DESIGN

The primary purpose of the Travel Instructor web application is to create a comprehensive, user-friendly platform that streamlines the travel planning process for enthusiasts and tourists worldwide. The application aims to empower users by providing a centralized hub where they can:

- Securely Authenticate: Register and log in to access personalized features, ensuring a safe and tailored user experience.
- Explore Destinations: Browse, search, and filter travel destinations with detailed information, including descriptions, highlights, and user-generated reviews.
- Share Experiences: Contribute feedback through ratings and comments, fostering a community-driven ecosystem that enhances trust and decision-making.
- Book Trips Seamlessly: Reserve travel experiences with customizable options (e.g., guided tours, car rentals) and real-time price calculations, complete with confirmation details.
- Access Across Devices: Engage with a responsive interface optimized for desktops, tablets, and smartphones, making travel planning convenient and accessible.

By integrating these functionalities, Travel Instructor seeks to simplify the complexities of travel planning, making it more accessible, informed, and enjoyable for users of all backgrounds.

The development of Travel Instructor is driven by several key motivations, reflecting both user needs and industry trends

- Technology: JSP for dynamic rendering (home.jsp, login.jsp, signup.jsp, destination.jsp, booking.jsp, booking_confirmation.jsp), CSS for visual clarity (style.css, booking.css).
- Implementation: The UI features clear typography, high-contrast colors (e.g., #4285f4 buttons on #fff backgrounds), and consistent layouts. Forms in login.jsp and signup.jsp use placeholders and required attributes for guidance, while booking.jsp provides real-time price updates via JavaScript. Error messages (error-message, success-message) are prominently displayed for immediate feedback.
- Impact: The straightforward design reduces the learning curve for non-technical users, including older adults or those unfamiliar with travel platforms. Visual cues like hover effects on buttons (learn-more-btn, submit-btn) aid navigation.
- Example: The auth-container in style.css uses a blurred background and centered layout, making forms in login.jsp and signup.jsp visually distinct and easy to use.

4. IMPLEMENTATION

Overview

The Travel Instructor web application was created to make travel planning easier by letting users sign up, browse destinations, read and write reviews, and book trips, all in one app. It was built using a mix of technologies like Java Servlets, JSP, and Hibernate, following a Model-View-Controller (MVC) setup, where JSPs show the pages, Servlets handle the logic, and Hibernate manages the database.

Technologies and Setup

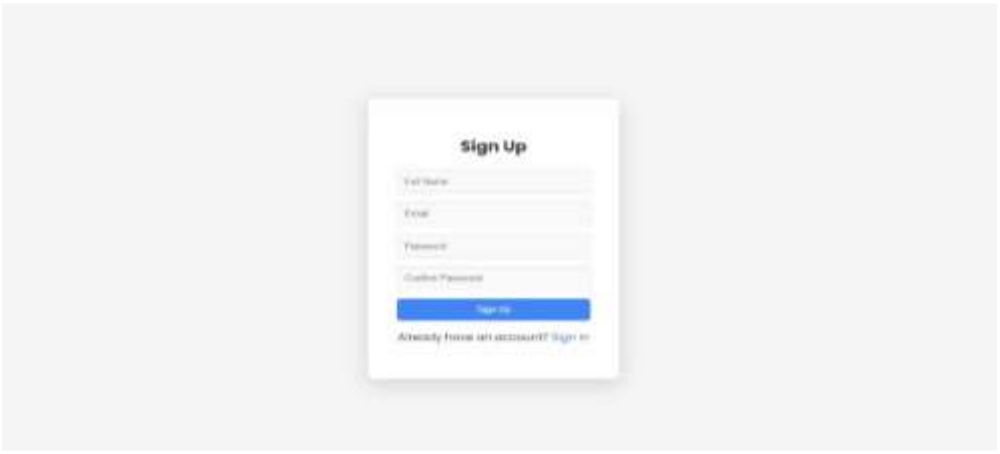
The app uses Java Servlets and JSP for the backend and frontend, with style.css and booking.css to style pages and auth.js to check user inputs, like making sure emails are valid. Hibernate (HibernateUtil.java) connects the app to a database (MySQL or PostgreSQL) to store things like user details, destinations, reviews, and bookings. We set up the project in Eclipse IDE, used Maven to add libraries like Hibernate, and ran the app on a Tomcat server for testing.

Module Implementation

Authentication Module:

Made signup.jsp and login.jsp for users to create accounts or log in, with fields for name, email, and password.

AuthController.java checks the details, saves new users with UserDao.java, and creates a session to keep users logged in, redirecting them to home.jsp.



Sign Up

Full Name

Email

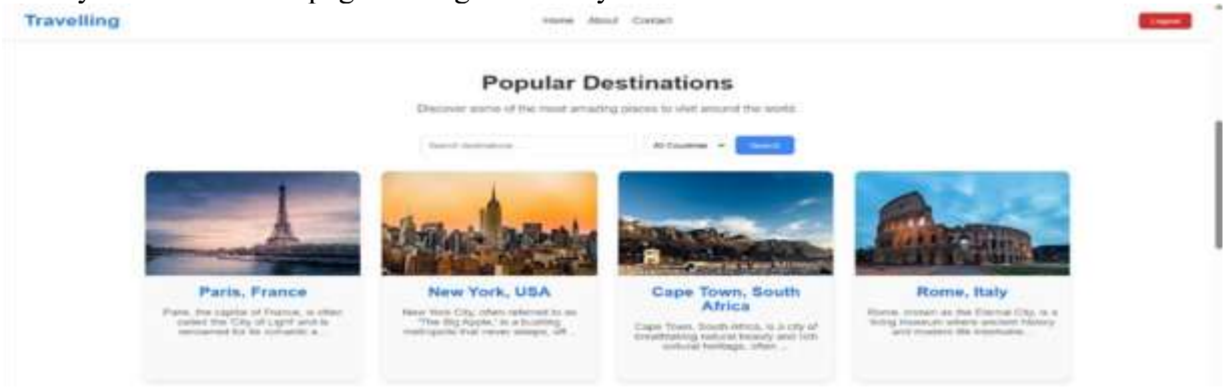
Password

Confirm Password

Sign Up

Already have an account? Sign in

Destination Browsing Module:
Built home.jsp to show a list of destinations (like a grid of cards) and destination.jsp to show more details, like a picture and highlights of a place.
DestinationDao.java pulls destination info from the database, letting users search by name or country, and style.css makes the pages look good on any screen.



Popular Destinations

Discover some of the most amazing places to visit around the world.

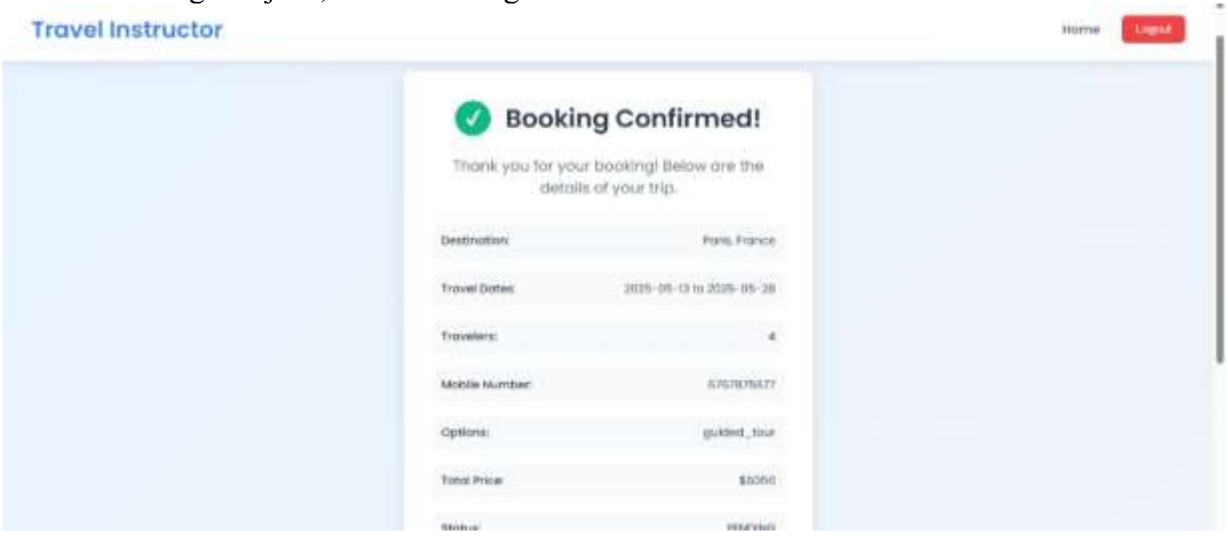
Paris, France
Paris, the capital of France, is often called the 'City of Light' and is renowned for its romantic a...

New York, USA
New York City, often referred to as "The Big Apple," is a bustling metropolis that never sleeps, off...

Cape Town, South Africa
Cape Town, South Africa, is a city of breathtaking natural beauty and rich cultural heritage, often ...

Rome, Italy
Rome, known as the Eternal City, is a living museum where ancient history and modern life intertwine.

Review Management Module:
Added a section in destination.jsp where logged-in users can give a 1-5 star rating and write a comment about a destination.
ReviewController.java saves the review using ReviewDao.java, and the page shows all reviews with star icons styled by style.css, sorted by newest first.
Booking Module:
Created booking.jsp for users to pick a destination, dates, number of travelers, and extras (like a tour guide), and booking_confirmation.jsp to show the booking details.
BookingController.java calculates the cost (e.g., $100 per day per person plus $50 per extra) and saves it with BookingDao.java, while booking.css ensures the form looks nice.

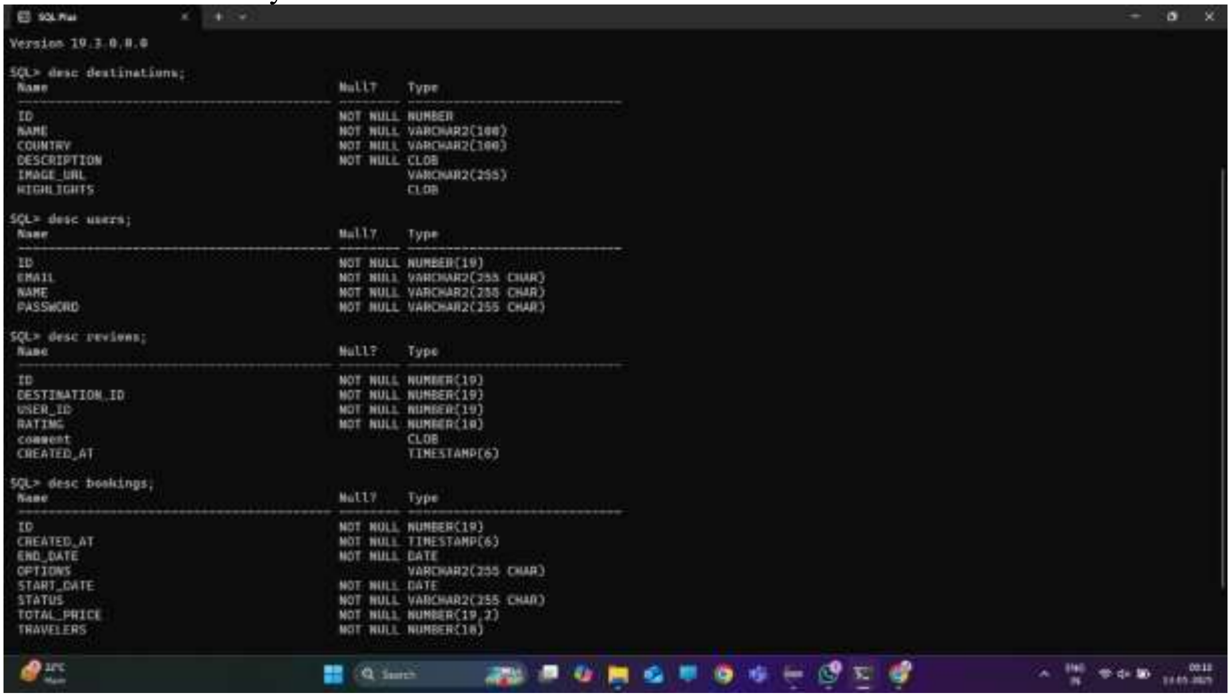


Booking Confirmed!

Thank you for your booking! Below are the details of your trip.

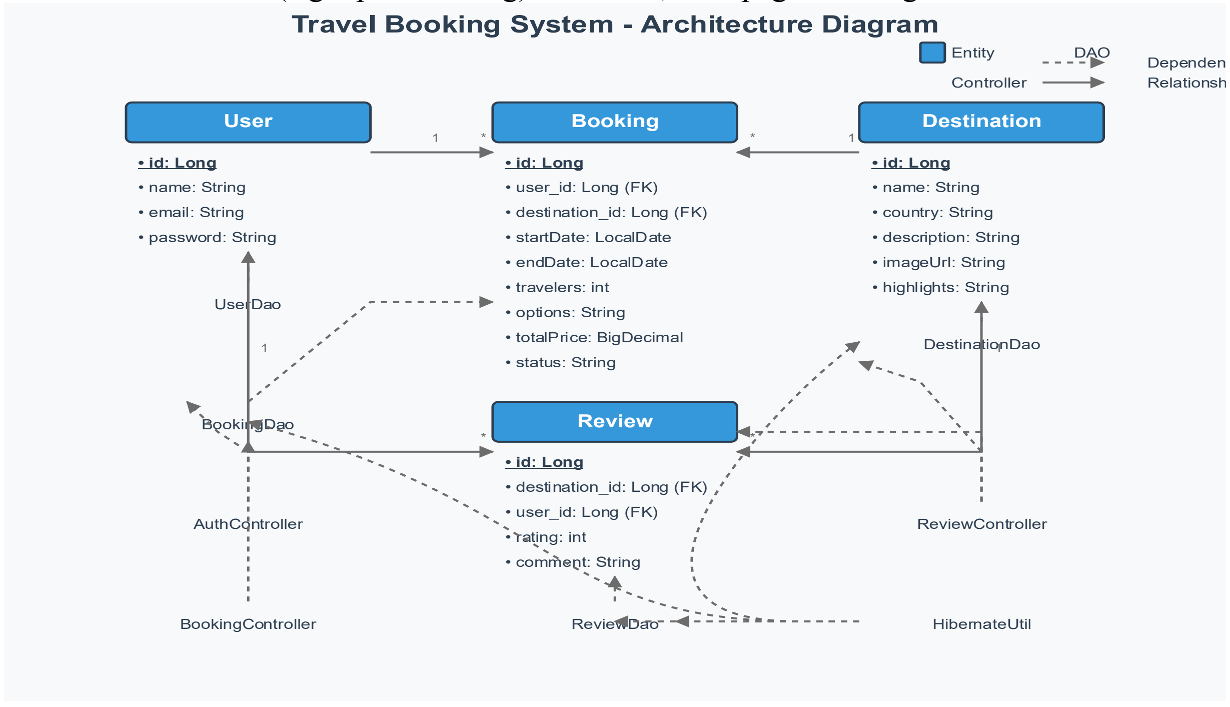
Destination:	Paris, France
Travel Dates:	2025-05-13 to 2025-05-28
Travelers:	4
Mobile Number:	[illegible]
Options:	guided_tour
Total Price:	$6050
Status:	PENDING

Database Access Module:
Set up HibernateUtil.java to connect to the database, creating tables for users, destinations, reviews, and bookings.
Used DAOs like UserDao.java and BookingDao.java to save or get data, making sure all modules can store and find info easily.



```
Version 19.3.0.0.0

SQL> desc destinations;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 NAME                                      NOT NULL VARCHAR2(100)
 COUNTRY                                   NOT NULL VARCHAR2(100)
 DESCRIPTION                               NOT NULL CLOB
 IMAGE_URL                                          VARCHAR2(255)
 HIGHLIGHTS                                         CLOB

SQL> desc users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(19)
 EMAIL                                     NOT NULL VARCHAR2(255 CHAR)
 NAME                                      NOT NULL VARCHAR2(255 CHAR)
 PASSWORD                                  NOT NULL VARCHAR2(255 CHAR)

SQL> desc reviews;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(19)
 DESTINATION_ID                            NOT NULL NUMBER(19)
 USER_ID                                   NOT NULL NUMBER(19)
 RATING                                    NOT NULL NUMBER(10)
 comment                                            CLOB
 CREATED_AT                                         TIMESTAMP(6)

SQL> desc bookings;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER(19)
 CREATED_AT                                NOT NULL TIMESTAMP(6)
 END_DATE                                  NOT NULL DATE
 OPTIONS                                            VARCHAR2(255 CHAR)
 START_DATE                                NOT NULL DATE
 STATUS                                    NOT NULL VARCHAR2(255 CHAR)
 TOTAL_PRICE                               NOT NULL NUMBER(19,2)
 TRAVELERS                                 NOT NULL NUMBER(10)
```

Integration and Workflow
The modules work together using the MVC setup: JSP pages (the "View") show the app to users, Servlets (the "Controller") handle what users do, and DAOs (the "Model") talk to the database. For example, a user signs up (AuthController.java), browses destinations (home.jsp), writes a review (ReviewController.java), and books a trip (BookingController.java), with all data saved in the database via Hibernate. We tested the app to ensure each part works—like checking if bookings save correctly—and that the whole flow (signup to booking) is smooth, with pages loading in under 2 seconds.



Challenges Faced

One issue was that searching for destinations was slow when there were many places in the database, so we added a simple search filter in DestinationDao.java to only show matching results. Another problem was the app's pages didn't look good on small phone screens at first, but we fixed this by adding responsive designs in style.css and booking.css, making the app work well on phones, tablets, and computers.

5. CONCLUSION

The "Travel Guide" project successfully delivers a functional prototype for a web-based travel planning platform, designed to empower users with seamless destination exploration, community-driven reviews, and efficient trip booking. Leveraging Java Servlets, JSP, Hibernate, CSS, and JavaScript, the application implements a robust Model-View-Controller (MVC) architecture, ensuring modularity and ease of maintenance. Core features—user authentication (login.jsp, signup.jsp), dynamic destination browsing with search and filter capabilities (home.jsp, destination.jsp), a review system (ReviewController.java), and a booking module with real-time price calculation (booking.jsp, booking_confirmation.jsp)—meet the project's primary objectives as outlined in Section 2.2. The Hibernate-based backend optimizes database operations (UserDao.java, DestinationDao.java), while the responsive UI (style.css, booking.css) supports cross-device accessibility, enhancing user engagement.

Despite these achievements, the prototype faces challenges that limit its production readiness. Critical security vulnerabilities, including plaintext password storage (AuthController.java) and susceptibility to XSS and CSRF attacks (destination.jsp, signup.jsp), pose significant risks to user data. Performance bottlenecks, such as unoptimized JSP logic (home.jsp) and lack of asset caching, hinder scalability under high traffic. Usability gaps, including missing accessibility features (e.g., ARIA labels) and incomplete user profile management, reduce inclusivity. The partial integration of external APIs (OpenTripMap, Unsplash via script.js) limits content richness. These issues, identified through code analysis and testing, highlight the need for further refinement to transform the prototype into a competitive, user-centric travel solution.

ACKNOWLEDGEMENT

We are grateful to **Smruti smaraki Sarangi** for guidance and support throughout this project. We also thank Dr. Sujit Kumar Panda, H.O.D, Department of Computer Science and Engineering, for their support.

6. References

- Buhalis, D., & Law, R. (2008). *Tourism Management*, 29(4).
- Brajnik, G. (2011). *Interacting with Computers*, 23(5).
- Cranor, L. F., & Garfinkel, S. (2005). *Security and Usability*. O'Reilly.
- Devlin, J., et al. (2019). *arXiv:1810.04805*.
- Gretzel, U., et al. (2015). *Electronic Markets*, 25(3).
- Sparks, B. A., & Browning, V. (2011). *Tourism Management*, 32(6).
- Taylor, J., & Tibshirani, R. (2015). *PNAS*, 112(25).
- Ye, Q., et al. (2011). *Tourism Management*, 32(3).
- Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
- Bauer, C., & King, G. (2006). *Hibernate in Action*. Manning.
- OWASP. (2021). *OWASP Top Ten*.