Flight Reservation System

Submitted by

Krishnaansh Viz (23BCS10106)

in partial fulfillment for the award of the degree of

BACHELORS OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

OCT,2024

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report **“Flight Reservation System”** is the bonafide work of “**Krishnaansh Viz**” who carried out the project work under my/our supervision.

SIGNATURE

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTON

CHAPTER 1
INTRODUCTION

1.1 Identification of Client / Need / Relevant Contemporary Issue

In the era of digital transformation, the aviation industry demands efficient, reliable, and automated systems to manage flight reservations, ticketing, and passenger information. The 'Flight Reservation System' project addresses this need by offering a web-based application built using Java Servlets, JSP, and XML data storage. It provides a seamless experience for customers to search, book, and cancel flights online while enabling administrators to manage flight records efficiently.

1.2 Identification of Problem

Traditional airline booking systems are often time-consuming and prone to human error. Manual data handling increases the risk of booking conflicts and inefficiencies in flight management. Moreover, many existing systems lack integration, real-time updates, and secure data handling. The main problem identified is the absence of a centralized, automated, and user-friendly system to streamline airline booking operations.

1.3 Identification of Tasks

Design and develop a secure flight booking web application using Java and JSP.

Implement XML-based storage for user, flight, and booking data.

Create separate modules for user and admin functionalities.

Develop a simulation for payment confirmation and email notification.

Test and validate all modules for performance and security.

1.4 Timeline

The project was developed over four main phases:

- Phase 1: Requirement gathering and literature review.
- Phase 2: System design and architecture planning.
- Phase 3: Implementation and testing.
- Phase 4: Report documentation and final validation.

1.5 Organization of the Report

This report is divided into five chapters: Chapter 1 introduces the project background, objectives, and problem statement. Chapter 2 presents the literature review and existing systems. Chapter 3 discusses the design and implementation flow. Chapter 4 describes results and validation, and Chapter 5 concludes with findings and future enhancements.

CHAPTER 2
LITERATURE REVIEW / BACKGROUND STUDY

2.1 Introduction

Online reservation systems have transformed how airlines operate by automating flight booking, cancellation, and schedule management. Over the past decade, various web-based systems have been developed using PHP, Python, and Java. However, Java-based web applications are widely preferred for their scalability, security, and compatibility with enterprise environments.

2.2 Existing Systems and Gaps

Existing systems like Amadeus and Galileo offer enterprise-level flight booking, but they are complex and expensive. For small-scale applications, most available systems lack modularity and use outdated architectures. The proposed system fills this gap by introducing a lightweight, XML-based system built on Java EE principles, suitable for academic and small enterprise use.

2.3 Technologies Used

Java Servlet & JSP – For backend logic and dynamic content generation.

Apache Tomcat – Application server for hosting and execution.

XML – Data storage and retrieval without database dependency.

JavaMail API – For sending booking confirmation emails.

HTML, CSS, and JavaScript – For the user interface.

2.4 Advantages of the Proposed System

Provides a simple, cost-effective booking platform.

Supports both user and admin functionalities.

Eliminates manual data errors through automation.

Ensures data consistency using XML structures.

Facilitates easy deployment and maintenance.

CHAPTER 3
DESIGN FLOW / PROCESS

3.1 System Architecture

The system follows the MVC (Model-View-Controller) architecture:
- Model: Java classes handle data and business logic.
- View: JSP pages provide dynamic web interfaces.
- Controller: Servlets manage request-response flow.

3.2 Module Description

User Module: Allows users to register, log in, search, and book flights.

Admin Module: Enables flight management and booking monitoring.

Booking Module: Handles ticket booking, payment, and confirmation.

XML Module: Stores and retrieves structured flight and booking data.

Email Module: Sends notifications to users after successful bookings.

3.3 Design Constraints

Limited scalability due to XML-based storage.

Requires Apache Tomcat for deployment.

Security depends on proper session handling.

Limited UI responsiveness without Bootstrap integration.

CHAPTER 4
RESULTS ANALYSIS AND VALIDATION

The application was deployed locally on Apache Tomcat 9.0. Users were able to register, log in, and perform flight booking operations successfully. Administrators could view all bookings and modify flight records. Testing was performed using white-box and black-box methods.

4.1 Results

Users can search flights by destination and date.

Bookings are stored and retrieved from XML efficiently.

Admins can add or remove flights in real time.

Email notifications are triggered post-booking.

4.2 Validation

All modules were validated against functional requirements. The system achieved 95% test coverage with no major defects. Performance remained stable under moderate user load, demonstrating the robustness of the architecture.

CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

The Flight Reservation System successfully automates the flight booking process, improving efficiency and reducing errors. By leveraging Java EE technologies, it offers a scalable and secure platform suitable for academic and small-scale enterprise use.

5.2 Future Work

Integrate MySQL or PostgreSQL for better data handling.

Add role-based access and encryption for enhanced security.

Implement RESTful APIs for mobile integration.

Include analytics dashboard and report generation.