# *Hotel Reservation System*
# Final Report

Name: Sahib Deep Singh
Team: Tech Travel Titans
Student number: 219170646

## Table of Contents

## Introduction

The "Hotel Reservation System" project, led by the Tech Travel Titans team, represents a comprehensive endeavor to revolutionize the hotel booking landscape. With a core emphasis on enhancing user experience and operational efficiency, our project endeavors to craft an advanced computerized system that optimizes every facet of hotel booking operations. Through meticulous analysis, innovative design, and strategic subsystem decomposition, we aim to create a solution that not only automates processes but also improves accessibility through intuitive online platforms.

Drawing from the rich data outlined in our requirements analysis and system design documents, our project seeks to address the diverse needs of guests, hotel staff, and online travel agencies. By integrating seamlessly with external systems and adhering to stringent security measures, we aspire to deliver a service that not only meets but exceeds industry standards. As a key member of the Tech Travel Titans team, my contributions have been instrumental in shaping the vision and execution of this ambitious project, positioning us for success in the dynamic realm of hotel reservation systems.

## Project Summary

**Overview:**

The hotel reservation system project by Team Tech Travel Titans aims to modernize the hotel reservation industry through automation, increase accessibility through the internet, integration with external systems to provide efficient service. Key features include automated hotel management systems, online booking platform, real-time data management, seamless integration with external platforms.

**Key Features:**

- **Online Booking Portals:** Providing users with convenient online platforms to search for hotels, book rooms, and manage reservations.
- **Personalized User Experience:** Offering tailored recommendations and features based on user preferences and booking history.
- **Real-Time Data Access for Staff:** Improving operational efficiency by granting staff immediate access to reservation information and guest preferences.
- **Automation of Reservation Management:** Streamlining processes such as booking, modification, and cancellation of reservations.
- **System Integration:** Ensuring seamless integration with external platforms such as online travel agencies and payment gateways.

## Summary of contributions

**My Role:**

- **Workload Management**: Managing 25% of the project workload in a group of 4 teammates (25% each), I demonstrated dedication and capability in contributing to major project components.
- **Architectural Planning:** I led my team and played a key role in the high-level architectural planning and subsystem decomposition of the project and implemented designs for understandability and to meet Artifact standards.
- **Core Functionality Development:** I significantly contributed to developing the core functionalities of the system, including analysis where I took care of functional requirements, scenarios and Activity Diagram and design aspects where I majorly looked after Software Architecture and System Decomposition.

**Team Dynamics:**

- Collaborative efforts were crucial in overcoming challenges posed by limited resources and given the time constraints.
- The project's progress was influenced by the team's small size compared to the size of BSA teams in real world, with each member, including myself, taking on substantial responsibilities.

## Lessons Learned from the Project

**Things That Went Well:**

1. **Innovative Design and Planning:** The team's ability to plan and design a complex system showcased strong analytical and design skills.
2. **Effective Collaboration:** Efficient collaboration and workload distribution among team members were key strengths, leading to successful project execution.
3. **Automation Implementation:** Successfully automating reservation management processes significantly improved operational efficiency and user experience.

**Challenges and Difficulties:**

While our project didn't encounter any significant challenges, we did face certain constraints that affected our workflow and project outcomes.

1. **Team Dynamics:** While choosing the right team members can increase productivity and productivity, learning to work effectively with teams, including managing difficult team members, is also a valuable lesson. It requires effective communication, conflict resolution skills and consensus to overcome differences and maintain cohesion among the internal team.

2. **Time Management:** Balancing a large workload was challenging and took up a lot of time for key members. Allocating sufficient time to each task and effective time management among team members was essential to meeting project deadlines and maintaining productivity.

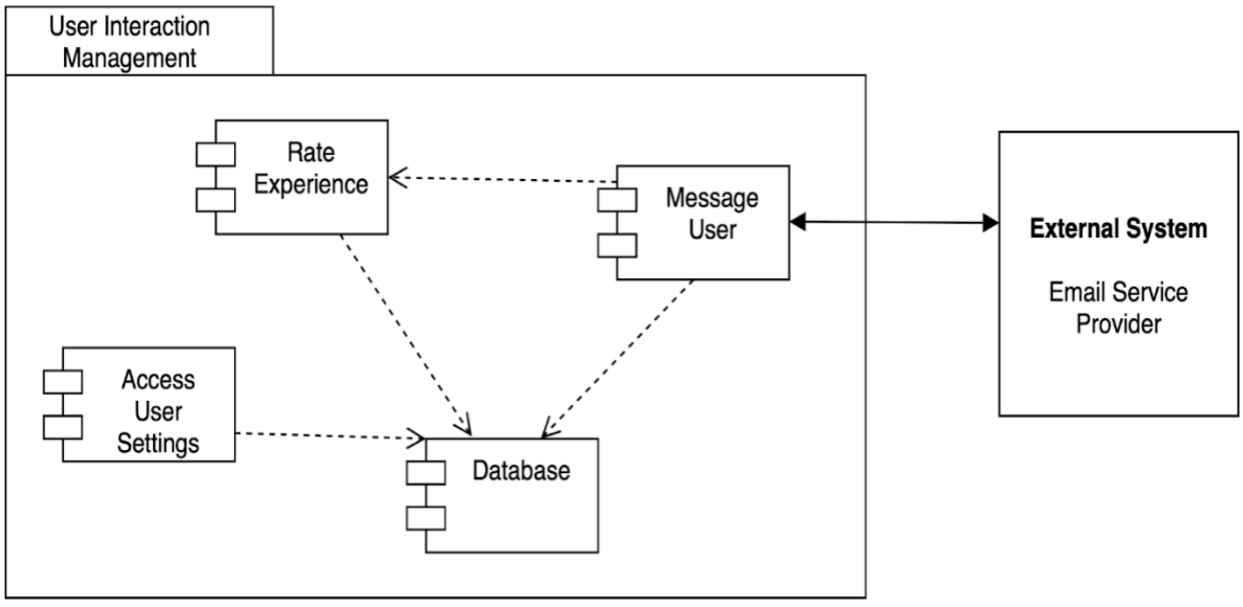
**Improvements in Analysis and Design:**

1. **Iterative Design Process:** Adopting an iterative approach to design could allow for regular feedback and adjustments, ensuring better alignment with user expectations.
2. **Comprehensive Requirement Analysis:** A more thorough initial analysis could lead to a more user-centric design and better alignment with project goals.
3. **Resource Allocation and Planning:** Better resource planning and task allocation could mitigate workload management challenges and detail coverage.

Through this role, the Tech Travel Titans team and I gained valuable insights into teamwork, project management, and program design and development. These lessons will undoubtedly be useful in future efforts, allowing my projects to be more efficient and effective.

## Object Design

**User Interaction Management Subsystem:**



**Overview:**

The User Interaction Management subsystem oversees user interactions and settings within the system, comprising MessageUserManagement, AccessUserSettingsManagement, and RateExperienceManagement. MessageUserManagement facilitates messaging, AccessUserSettingsManagement enables users to customize preferences, and RateExperienceManagement handles feedback and ratings. All components interface with the central database to store user data and interactions

- ***External System:*** *Email Service Provider*
  ***Integration Purpose:*** *Ensure reliable and timely delivery of system-generated emails, such as confirmation emails and notification.*

**Key Components:**

1. **MessageUserManagement:**
- Manages user messaging, including composition, delivery, and storage.
- Enables users to send, receive, and view messages within the system.

2. **AccessUserSettingsManagement:**
- Manages user preferences, including notifications, language, and profile information.
- Facilitates CRUD operations for user settings customization.

3. **RateExperienceManagement:**
- Manages user feedback and ratings for system experiences and hotel bookings.
- Enables users to submit feedback, ratings, and reviews.
- Analyzes feedback to enhance system performance and user satisfaction.

**Class Diagram:**

- **User Class:** Attributes include userID, userName, password, emailAddress, role, phone. Operations include createAccount(), authenticateUser(), logIn(), logOut().

- **Message Class:** Attributes include messageId, senderId, recipientId, content, timeStamp, isRead. Operations include messageUser(), markAsRead().

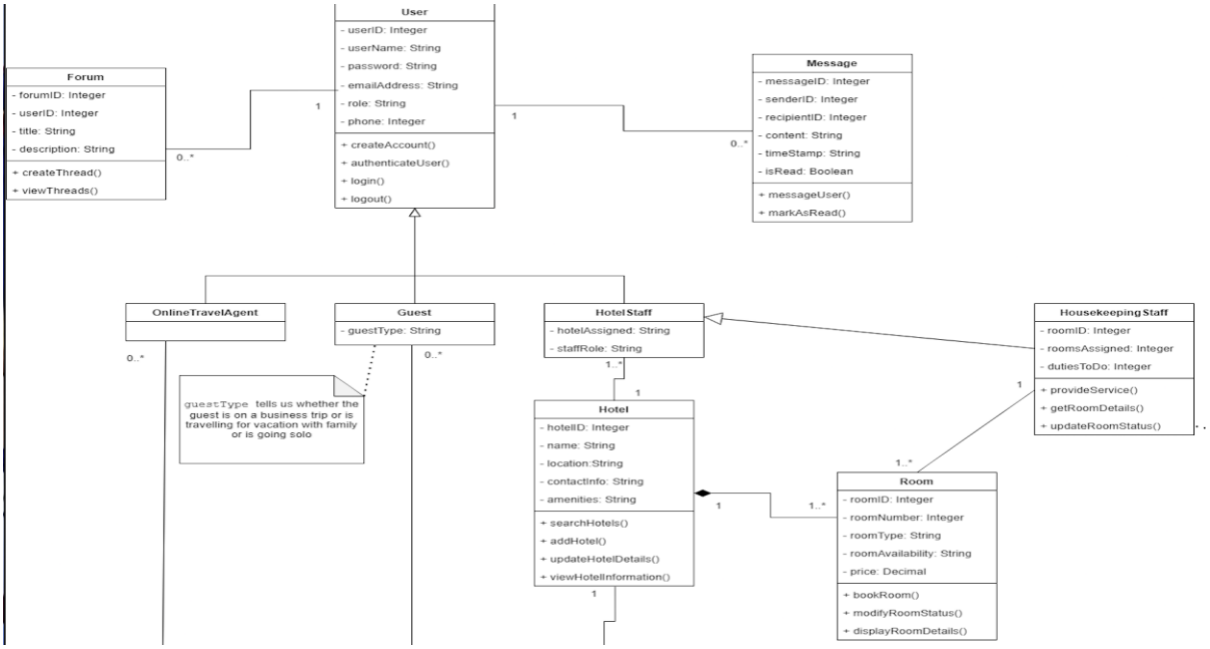
- **Guest Class:** Subclass of User. Attributes include additional guest-specific data. Operations inherit from User class.

- **OnlineTravelAgent Class:** Subclass of User. Attributes include additional online travel agent-specific data. Operations inherit from User class.

- **HotelStaff Class:** Subclass of User. Attributes include additional hotel staff-specific data. Operations inherit from User class.

- **Forum Class:** Attributes include forumId, userId, title, description. Operations include createThread(), viewThread().

**Part of Class Diagram covering some of the classes for User Interactions:**




**UserAccountTransactionManagement Package:**

- This package centrally manages user account management tasks in the system, ensuring the accuracy and integrity of user account data during various operations including creation, modification, and execution.
- The UserAccountTransactionManagement subsystem manages tasks related to user account management and ensures secure and accurate processing. It takes care of tasks such as account creation, information modification and user authentication during login.

**Design Patterns**

- **Singleton Pattern:** Used in the Database Connection class to ensure that a single instance manages all database connections, promoting resource efficiency and data consistency. This model is particularly useful for eliminating the unnecessary overhead associated with creating multiple database connections, thereby improving system performance and scalability.

- **Factory Method Pattern:** Used to instantiate user subclasses (Guest, OnlineTravelAgent, HotelStaff) based on user roles in authentication, contain object creation logic and encourage changes to add new user types. By abstracting in isolation from the model system, the factory path model makes it easy to add new user types without changing the existing authentication logic, thus following the open/closed principle of a regarding software design.

- **Strategy Pattern:** Used for access rights management to dynamically adjust various control options based on application usage, encouraging easy modification and maintenance. This model enables the system to handle changing access control requirements without changing the core logic, enabling flexibility to enhance evolving security requirements, and reducing duplicate rules in multiple applications.

- **Observer Pattern:** Used in MessageUserManagement to notify users when new messages are received, ensuring effective communication and user engagement. By separating the message delivery system from the message creation process, the observer pattern facilitates simple user interfaces in response to message events, enhancing system response and user satisfaction.

## Packages

1) **UserInteractionManagement Package:**

- Overview: This package manages user interaction and configuration of the system. It includes MessageUserManagement, AccessUserSettingsManagement, and RateExperienceManagement components.
- Dependencies: This package may depend on the CentralDatabasePackage for accessing and storing user-related data in the central database.
- Expected Usage: Developers working on user interaction features of the system will mainly interact with this package. It provides functionality for managing user messages, accessing user settings, and user profiles and usage rates.

2) **UserAccountTransactionManagement Package**:

- Overview: This package focuses on handling tasks related to user account management, including account creation, modification, and verification processes.
- Dependencies: It must rely on CentralDatabasePackage to access user account data in the central database and UserInteractionManagementPackage to manage user interaction and configuration.
- Expected Usage: Developers working on user account management functionalities, such as user registration, login, and profile management, will use this package extensively.

3) **CentralDatabasePackage**:

- Overview: This package contains functionality for accessing and managing data stored in a central database. It includes features for database connections, data retrieval, storage, and transformation.
- Dependencies: his package should not have any dependencies on other packages in the system, because it acts as a recovery component for central data storage
- Expected Usage: Developers who manage data persistence and recovery performance across different subsystems will interact with this package. It provides important functionality to ensure data integrity and consistency throughout the system.

**File Organization:**

- **UserInteractionManagement Package:**
  MessageUserManagement.java (Observer pattern)
  AccessUserSettingsManagement.java
  RateExperienceManagement.java

- **UserAccountTransactionManagement Package:**
  UserAccountTransactionManagement.java (Strategy pattern for Access Rights Management)

- **CentralDatabase Package:**
  DatabaseConnection.java (Singleton pattern)
  DataRetrieval.java
  DataStorage.java

This file structure ensures that each package has its relevant resources and encourages modularity and ease of maintenance. Developers can easily identify and implement specific functionality in the system based on package structure. In addition, the CentralDatabasePackage is central to data management throughout the system.

## Class interfaces

- **Class Diagram structure in detail including parameters as needed for Contracts understanding:**

  **[User]**
  - userID: String
  - userName: String
  - password: String
  - emailAddress: String
  - role: String
  - phone: String

---
+ createAccount(): void
+ authenticateUser(username: String, password: String): boolean
+ logIn(username: String, password: String): boolean
+ logOut(): void

**[Message]**
- messageId: String
- senderId: String
- recipientId: String
- content: String
- timeStamp: Date
- isRead: boolean
---
+ messageUser(senderId: String, recipientId: String, content: String): void
+ markAsRead(messageId: String): void

**[Forum]**
- forumId: String
- userId: String
- title: String
- description: String
---
+ createThread(userId: String, title: String, description: String): void
+ viewThread(threadId: String): void

**[Guest]**
---
<<subclass of User>>
+ additionalGuestData: String

**[OnlineTravelAgent]**
---
<<subclass of User>>
+ additionalAgentData: String

**[HotelStaff]**
---
<<subclass of User>>
+ additionalStaffData: String

- **Contracts:**

```
context User::createAccount():
   pre:
      self.role = 'Guest' or self.role = 'OnlineTravelAgent' or self.role = 'HotelStaff'
      and self.userName <> null and self.password <> null
   post:
      self.userName <> null and self.password <> null
      and self.userName = userName and self.password = password
      and self.role = 'Guest' or self.role = 'OnlineTravelAgent' or self.role = 'HotelStaff'
```

```
context User::authenticateUser(username: String, password: String):
   pre:
      username <> null and password <> null
   post:
      result = true or result = false
```

```
context User::logIn(username: String, password: String):
      pre:
         username <> null and password <> null
   post:
        result = true or result = false
```

```
context Message::messageUser(senderId: String, recipientId: String, content: String):
   pre:
      senderId <> null and recipientId <> null and content <> null
   post:
      self.senderId = senderId and self.recipientId = recipientId and self.content = content
```

```
context Message::markAsRead(messageId: String):
      pre:
         messageId <> null
      post:
          self.isRead = true
```

```
context Forum::createThread(userId: String, title: String, description: String):
   pre:
      userId <> null and title <> null and description <> null
   post:
      self.userId = userId and self.title = title and self.description = description
```

```
context Forum::viewThread(threadId: String):
   pre:
      threadId <> null
   post:
      // postcondition
```

**Justification for Contracts:**

1. **User Class:**

- Invariant: The user role must be ‘Guest’, ‘OnlineTravelAgent’, or ‘HotelStaff’. Additionally, the username and password must not be null.
- Preconditions for methods: Usernames and passwords cannot be null for account creation, authentication, and login.
- Postconditions: After creating an account, both username and password should be set and not null. Additionally, the role should match one of the valid roles.

2. **Message Class:**

- Invariant: None.
- Preconditions for methods: Sender ID, recipient ID, and message content cannot be null for sending a message. Message ID cannot be null for marking as read.
- Postconditions: When sending a message, the sender ID, recipient ID, and content are formatted accordingly. When marking a message as read, the isRead attribute is set to true.

3. **Forum Class:**

- Invariant: None.
- Preconditions for methods: User ID, title, and description cannot be null for creating a thread.
- Postconditions: When creating a thread, the user ID, title, and description are formatted accordingly. Viewing a thread does not have specific postconditions mentioned as it may depend on the specific requirements of the system.

## Additional Comments to the Course

**Strengths of the Course:**

1. **Application Oriented Approach:** This course effectively connects theoretical principles with practical implementation, exemplified through projects like the Hotel Reservation System. This method facilitates a clear understanding of how theories are applied in real-world contexts.
2. **Emphasis on Project-Based Learning:** Incorporating a group project like the Hotel Reservation System provided hands-on experience, fostering teamwork and problem-solving abilities among groupmates.

**Areas for Improvement:**

1. **Increased Exposure to Industry Practices:** Integrating more insights from industry case studies or real-world examples of system design challenges and solutions would provide students with a deeper understanding of current industry practices and trends. This exposure could include examining successful implementations of large-scale systems or exploring how emerging technologies influence system design decisions.

**Overall Feedback:**

The course is highly structured and informative, providing students with valuable knowledge and skills in systems analysis and design. It strikes a balance between theoretical understanding and practical application, and thoroughly prepares students for real-world situations in system design and development. The inclusion of a team project is particularly beneficial, making collaboration and ideas more effective.