Alexandria National University

Faculty of Computers and Data Science

Software Engineering Course

Travel Booking System Report

Team Participants

Omar Magdy Abdulla 2205221

Kareem Ahmed Helmy 2205069

Adham Ahmed Reda 2205087

Ismail Ibrahim Aly 2305415

Introduction

Purpose

The purpose of this document is to clarify the specifications for creating a travel booking web application for **Users (customers)** and **Admins**. It will allow users to browse through travel offers, book, and input visa details. Admins can manage offers, bookings, and contact enquiries.

Definitions

- **User:** An individual who searches for travel offers, books, and provides visa information.
- **Admin:** A user with higher privileges, who maintains offers, bookings, and attends to the inquiry of a user.
- **Travel Offer:** An offer for travel locations which users can book.
- **Booking:** A reservation made by a user for a specific travel offer.
- **Visa:** Information or a document needed to travel, which users will provide for their bookings.
- **Contact Message:** A question or a help request sent by users to the system administrators.

Goals and Objectives

- **User Objectives:** Offer a user-friendly interface for users to surf, reserve travel deals, and enter visa information.
- **Admin Goals:** Provide an efficient backend to handle travel offers, bookings, and contact messages.
- **System Objectives:** Make the platform secure, have it perform well, be scalable, and run smoothly on desktop platforms.

Plan

Phase 1: Planning and requirement gathering.

Phase 2: Design phase, which includes database schema, system architecture, and UI.

Phase 3: Development phase with focus on front-end and back-end integration.

Phase 4: Testing phase, which comprises performance, security and usability testing.

Phase 5: Deploy and go-live, and ongoing maintenance and updates.

Requirements

User Requirements

- User Roles: Two types of users—Regular Users (customers) and Admin Users.

Regular User Requirements:

- **Account Management:** Sign-up, login, and account security.
- **Travel Offers:** Browse, search, and view detailed travel offers.
- **Booking:** Make, review, and manage bookings; cancel if within a given time.
- **Visa Submission:** Submit visa details as needed.
- **Contact Support:** Send support questions.
- **Responsive Design:** Use the system on Desktop.

Admin User Requirements:

- **Authentication:** Admin login security.
- **Travel Offer Management:** Add, edit, or delete offers.
- **Booking Management:** Update and view user bookings.
- **Support Management:** Respond and read user inquiries.
- **System Logs:** Display system actions and logs.

System Requirements

Functional Requirements:

- **User Authentication:** Secure login for both users and admins.
- **Booking System:** Allow users to book and manage travel offers.
- **Travel Offer Management:** Admins can create or edit offers.
- **Contact Management:** Admins handle user queries.
- **Database Management:** Structured database for users, bookings, and offers.

Non-Functional Requirements:

- **Performance:** Fast load times and scalability for many users.
- **Security:** Secure passwords, HTTPS, and data protection.
- **Availability:** System must be available 24/7 with backups.
- **Cross-Platform Compatibility:** Accessible on major browsers and devices.

Hardware and Software Requirements:

- **Server:** MySQL.
- **Client:** Modern web browsers and devices.

External Interfaces:

- **Payment Gateway:** Integration for payment processing.
- **OAuth:** Third-party login integration (e.g., Google, GitHub).

Software Process

Type of Software Process

The recommended software process for this Customized Type project is the Agile process. Agile focuses on iterative development, and the team can readily adapt to changes, especially in environments where they change frequently, such as web development. The process involves:

- **Regular Releases:** Releasing useful features in little, incremental increments (sprints), typically every 2–4 Days.
- **Collaboration:** Frequent communication between developers, stakeholders (admins, users), and the project manager to ensure the product is in accordance with the requirements.
- **Flexibility:** Adaptation of project scope and feature changes based on user feedback and market changes.
- **Ongoing Testing:** Ensuring the application is regularly tested for bugs, security vulnerabilities, and usability improvements.

Division of Phases

The software process will be divided into the following phases:

Specification Phase

- Project scope, stakeholders, and user requirements definition.
- User stories creation and high-level feature definition.
- Project timelines and resource allocation.

Design Phase

- System architecture and database design.
- Wireframe and UI/UX prototype designing for the user and admin interfaces.
- Data flow and overall application structure definition.

Implementation Phase

- Front-end and back-end code development, utilizing technologies like PHP, JavaScript, and MySQL.
- Incorporating payment gateways, user authentication, and booking features.
- Developing the admin panel for managing offers and bookings.

Validation (Testing) Phase

- Conducting various types of testing including unit testing, integration testing, security testing, and user acceptance testing (UAT).
- Making sure the system runs efficiently under different conditions and is secure from attacks.
- Debugging and performance tuning.

Evolution Phase

- Regular updates to add features, debug, and optimize performance.
- Handling user feedback and support.

Architectural Design

System Architecture

System Architecture will consist of a three-tier architecture with:

Presentation Layer (Client-Side):

This is the front-end layer of the application where the users will be interacting.

Technologies: HTML, CSS, JavaScript

Responsibilities: Displaying travel offers, booking forms, user login/registration, and server communication through API calls.

Business Logic Layer (Server-Side):

This is the layer in which the main logic and functions of the system will be run.

Technologies: PHP (backend) to process user requests, book processing, travel offers handling, etc.

Responsibilities: Managing business logic such as user authentication, booking processing, managing visa applications, etc. It will also manage database interactions.

Data Layer (Database):

Persist all application data such as users, bookings, travel offers, and contact messages.

Technologies: MySQL for relational database management.

Responsibilities: Storage and retrieval of data, integrity of data, and management of complex inter-relationships among various entities (users, bookings, travel offers).

Application Architecture

Application Architecture shall be applied on top of the Model-View-Controller (MVC) pattern:

Model:

- This segment is the data and business logic of the app.
- **Technologies:** PHP for dealing with data models, and MySQL for firing SQL commands onto the database.
- **Responsibilities:** Talking to the database, data processing, and applying business rules (e.g., booking validation, price calculations).

View:

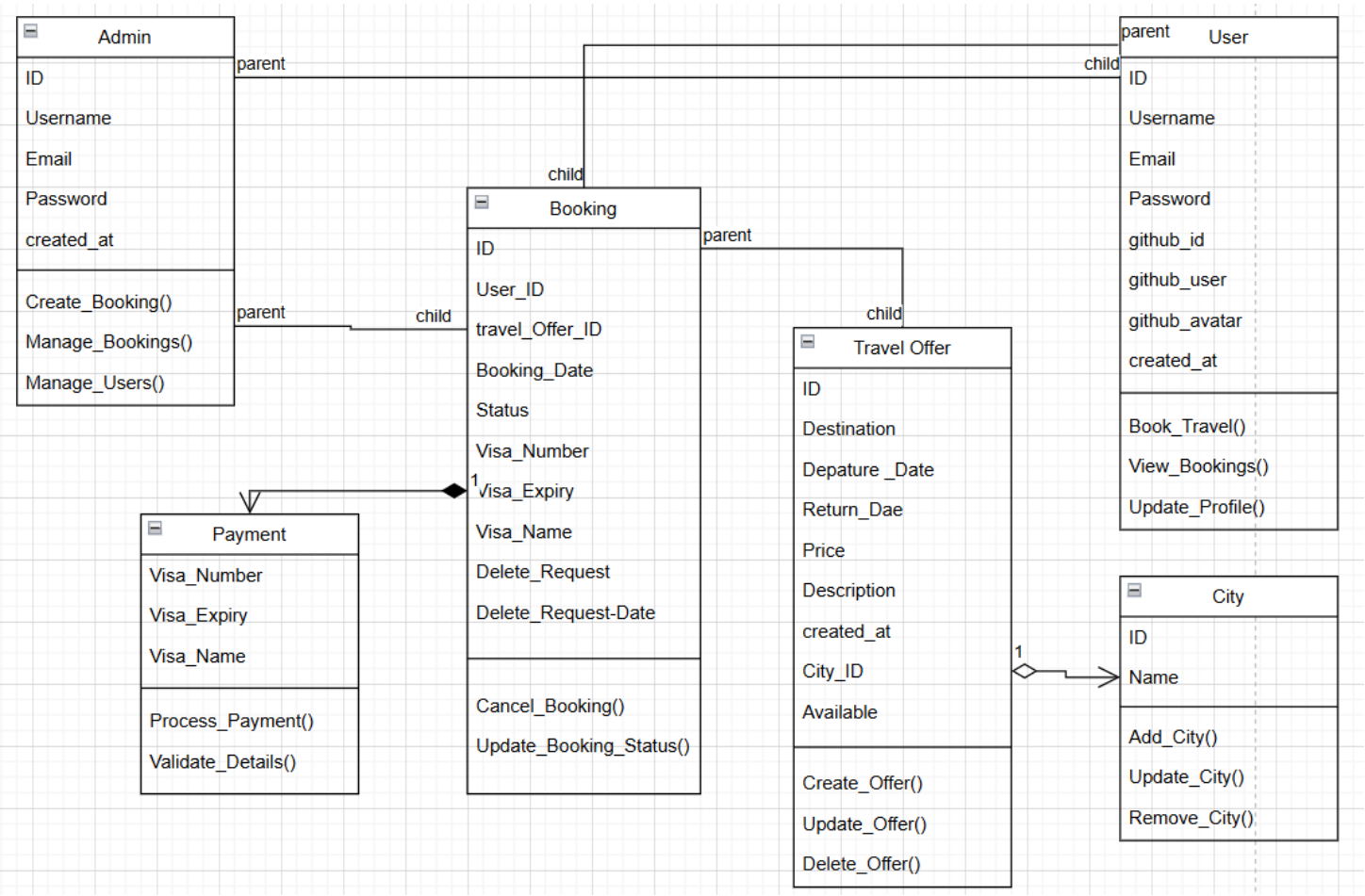
- The View is the user interface (UI), and it's what shows data to the user and accepts input from the user.
- **Technologies:** HTML, CSS, and JavaScript for presenting dynamic and interactive pages.
- **Responsibilities:** Displaying travel offers, booking pages, user consoles, and administrator panels.

Controller:

- The Controller is a middleman between the Model and View. It responds to user requests and updates the View accordingly.
- **Technologies:** PHP for handling user requests and back-end-front-end communications.
- **Responsibilities:** Handling API requests, handling user action (e.g., booking submission), and invoking related business logic from the Model.
- It's a tidy separation of concerns, and thus easy to maintain and scale.

System Modeling

Class Diagram



Admin Class

Attributes: ID, Username, Email, Password, created_at

Functions: Create_Booking(), Manage_Bookings(), Manage_Users()

Relationships: Association with User Table & Booking Table as he Manages them

User Class

Attributes: ID, Username, Email, Password, github_id, github_user, github_avatar, created_at

Functions: Book_Travel(), View_Bookings(), Update_Profile()

Relationships: Association with Booking table as he Books a Travel Offers

Booking Class

Attributes: ID, User_ID (foreign key referencing User), travel_Offer_ID (foreign key referencing Travel Offer), Booking_Date, Status, Visa_Number, Visa_Expiry, Visa_Name, Delete_Request, Delete_Request_Date

Functions: Cancel_Booking(), Update_Booking_Status()

Relationships: Association with Travel Offer Table as it is refer to it but the offer doesn't depend on Bookings

Travel Offer Class

Attributes: ID, Destination, Departure_Date, Return_Date, Price, Description, created_at, City_ID (foreign key referencing City), Available

Functions: Create_Offer(), Update_Offer(), Delete_Offer()

City Class

Attributes: ID, Name

Functions: Add_City(), Update_City(), Remove_City()

Relationships: Aggregation with the Travel Offer Table as a City can exist independently of Travel Offers. Offers belong to a city, but cities are not deleted if offers are removed

Payment Class

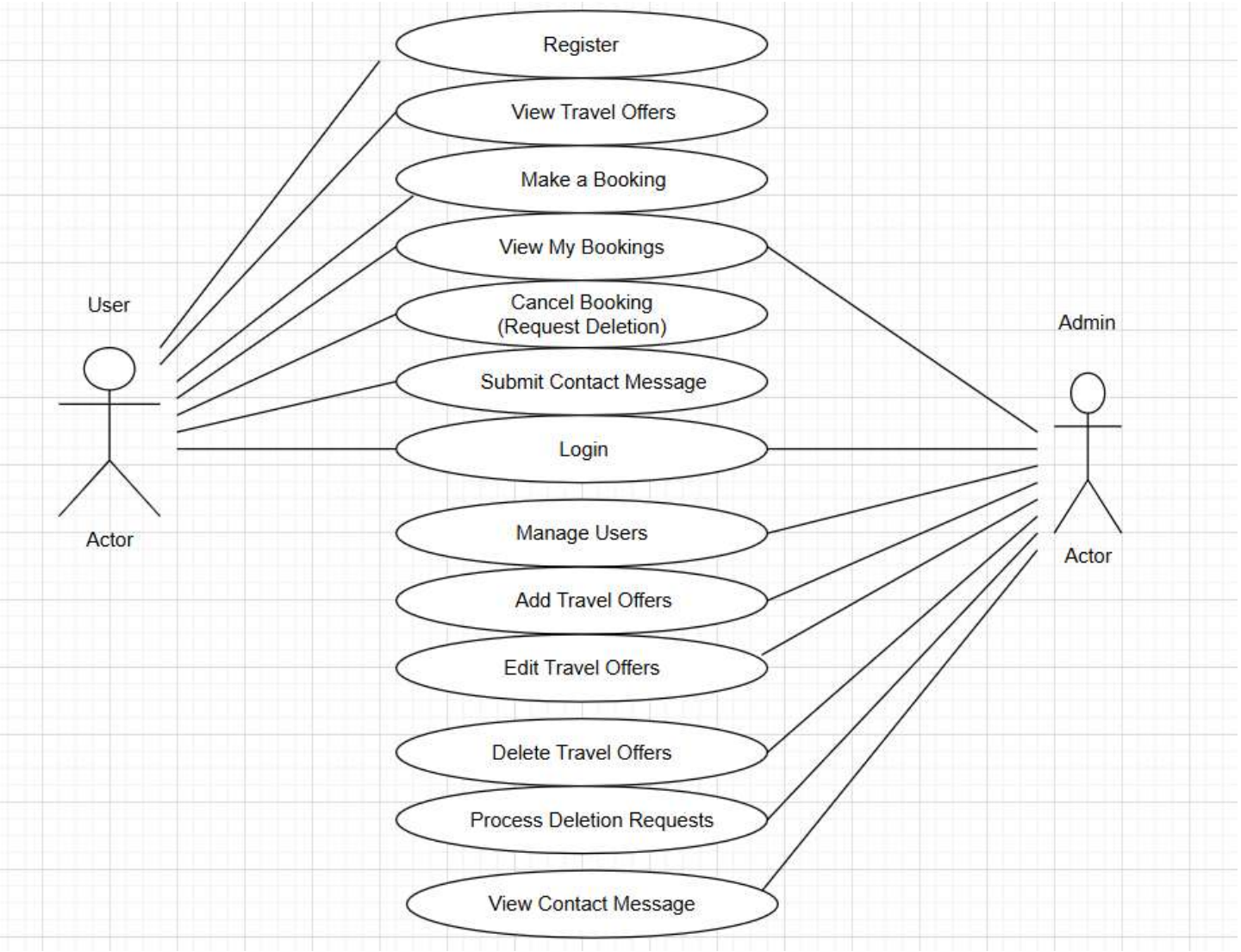
Attributes: Visa_Number, Visa_Expiry, Visa_Name

Functions: Process_Payment(), Validate_Details()

Relationships: Payment details are tightly bound to a booking and do not exist independently. If a booking is deleted, the payment should be too

---

Use Case Diagram

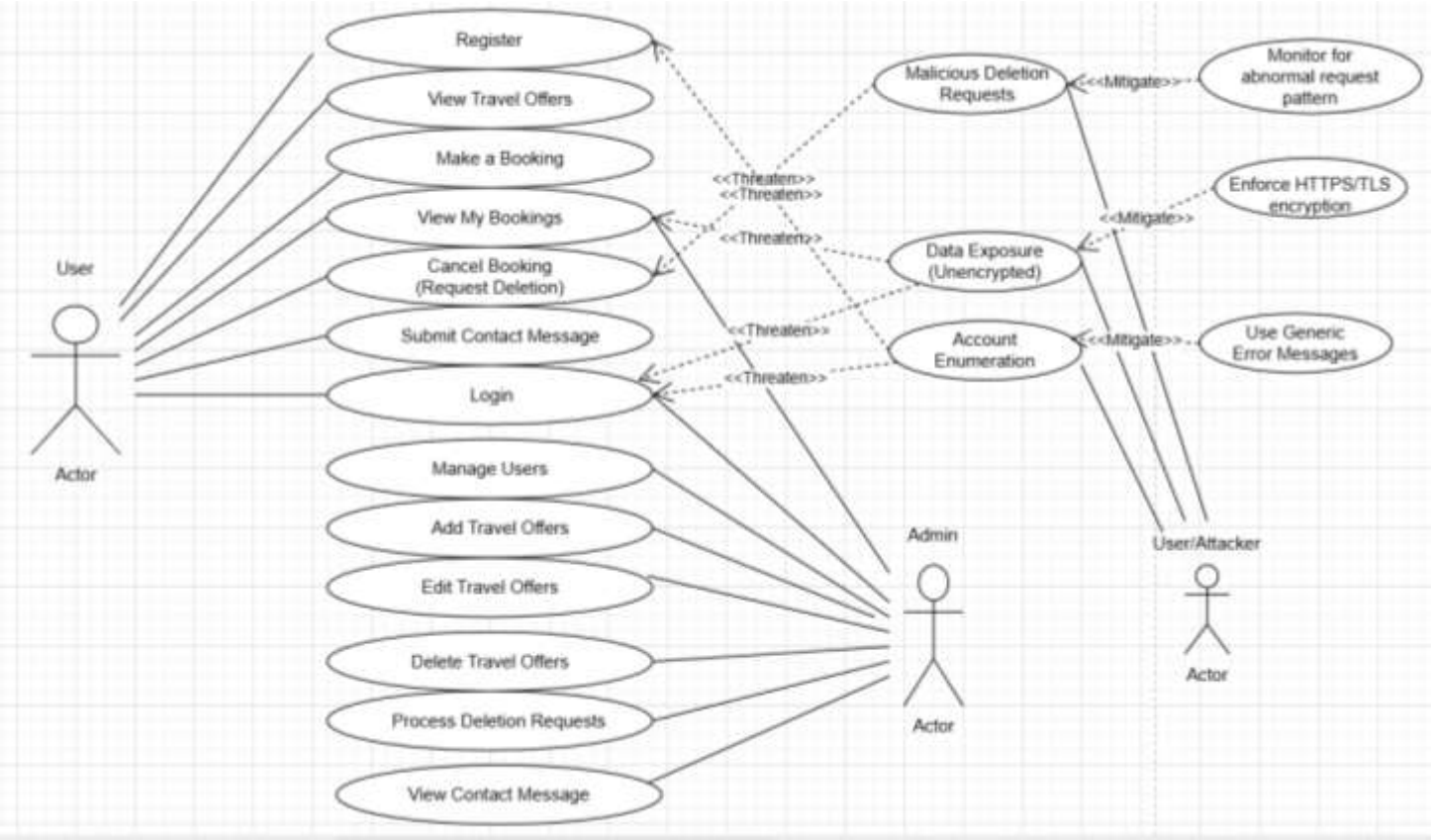


This Use Case Diagram illustrates the functional requirements of a travel booking system by outlining the interactions between two primary actors: User and Admin.

Users can register, login, view available travel offers, book and manage bookings, send contact messages, and cancel bookings. Admins have additional rights including login, user management, adding/editing/deleting travel offers, handling deletion requests, and viewing contact messages. The diagram clearly separates user-level activities from admin operations and indicates a well-planned and organized approach of system access and control

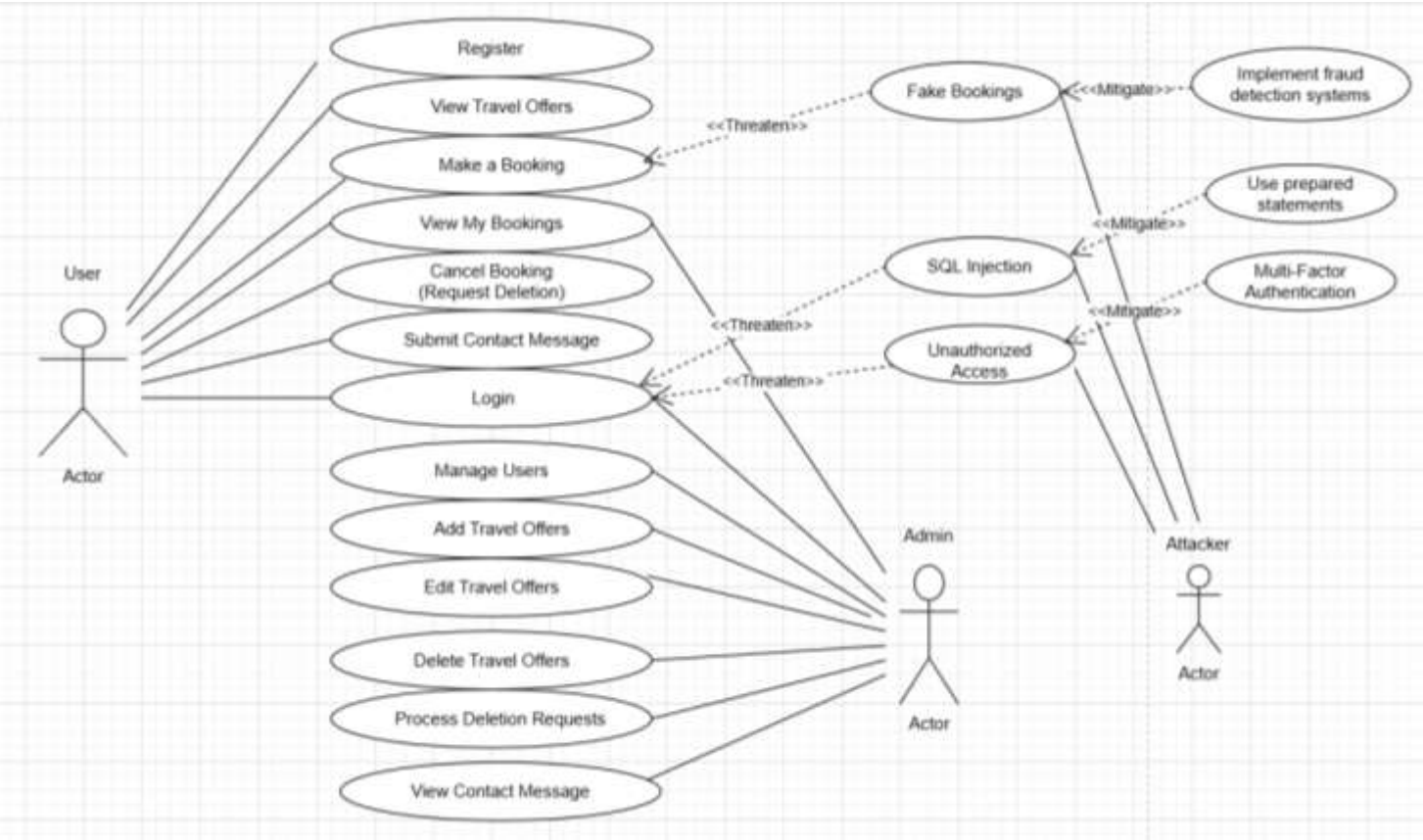
---

Misuse Case Diagram



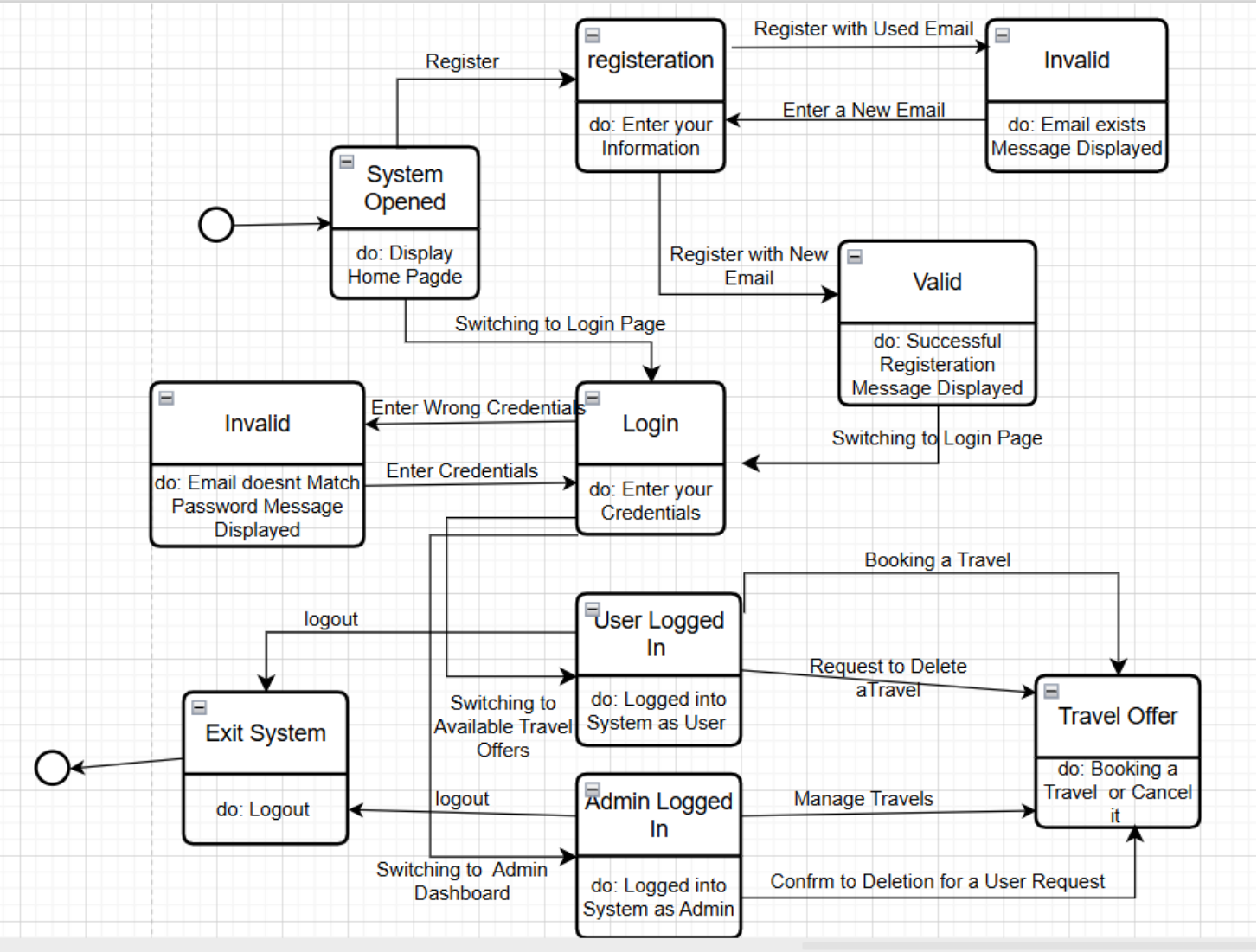
This misuse case diagram shows potential risks of the travel booking system from Malicious or Unintended User Behavior. It shows how genuine system activities like registration, login, and sending contact messages can be utilized by an attacker (or user/attacker) to perform Malicious Delete Requests, Data Exposure through unencrypted pathways, and Account Enumeration. To reduce these threats, the system uses protection measures such as Monitoring for Abnormal Patterns of Requests, Use of HTTPS/TLS Encryption, and Generic Error Message return for maintaining information leaks out. This figure helps in understanding how the system can be hardened against specific misuse situations

Abuse Case Diagram



This Abuse case diagram represents potential security weaknesses in the travel booking system. It demonstrates how a regular user and admin activities can be exploited by an attacker through Fake bookings, SQL injection, and Unauthorized Access. In order to offset these attacks, the system utilizes Fraud detection, Prepared statements, and Multi-Factor Authentication as the mitigation strategies. The diagram simplifies Abuse scenarios together with the respective security mechanisms for protecting the system.

State Machine Diagram



State Machine Diagram Model The Behaviour of the System in Response to External & Internal Events. It Show system States as Nodes, and Events as Arcs Between These Nodes,

When an Event occurs , The System Moves From One State to Another State.

The System starts by showing the home page from which users can register or login.

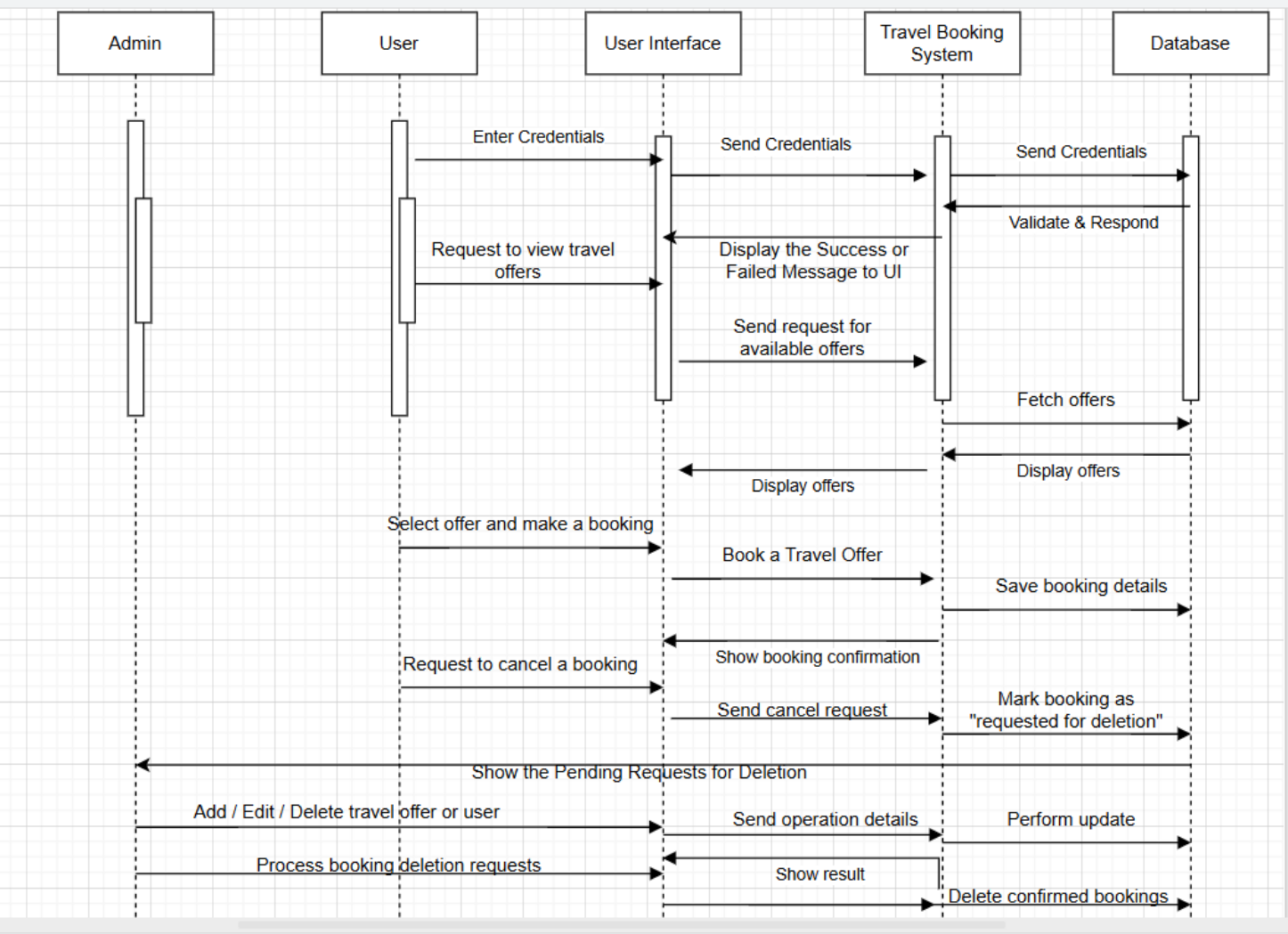
After Registration, the user logs in; otherwise, an error message is shown.

Upon log in, users or admins are directed to their respective dashboards.

Users can look at, book, or cancel travel offers, but admins manage travel details.

Both admins and users can log out to exit the system

Sequence Diagram



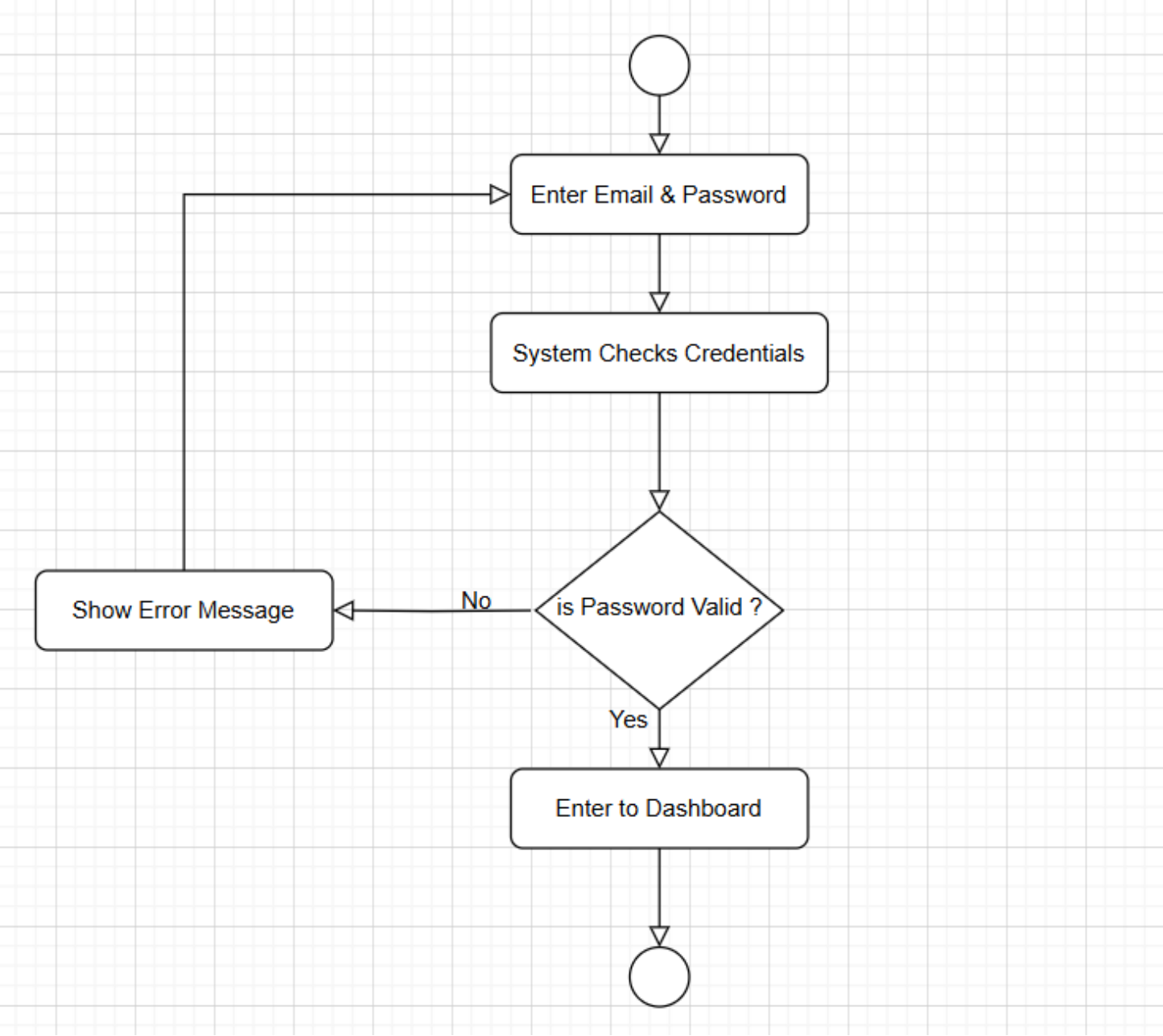
The Sequence Diagram illustrates the interaction between the User, Admin, User Interface, Travel Booking System, and Database components.

It starts with user authentication, in which the system verifies credentials and returns success or failure.

After logging in successfully, users are able to view travel offers, choose an offer, and make a booking request.

The system processes the cancellation of reservations by flagging them for deletion and alerting the Admin to take appropriate action. Administrative tasks involve adding, editing, and deleting travel offers or users, and managing and verifying booking cancellations.

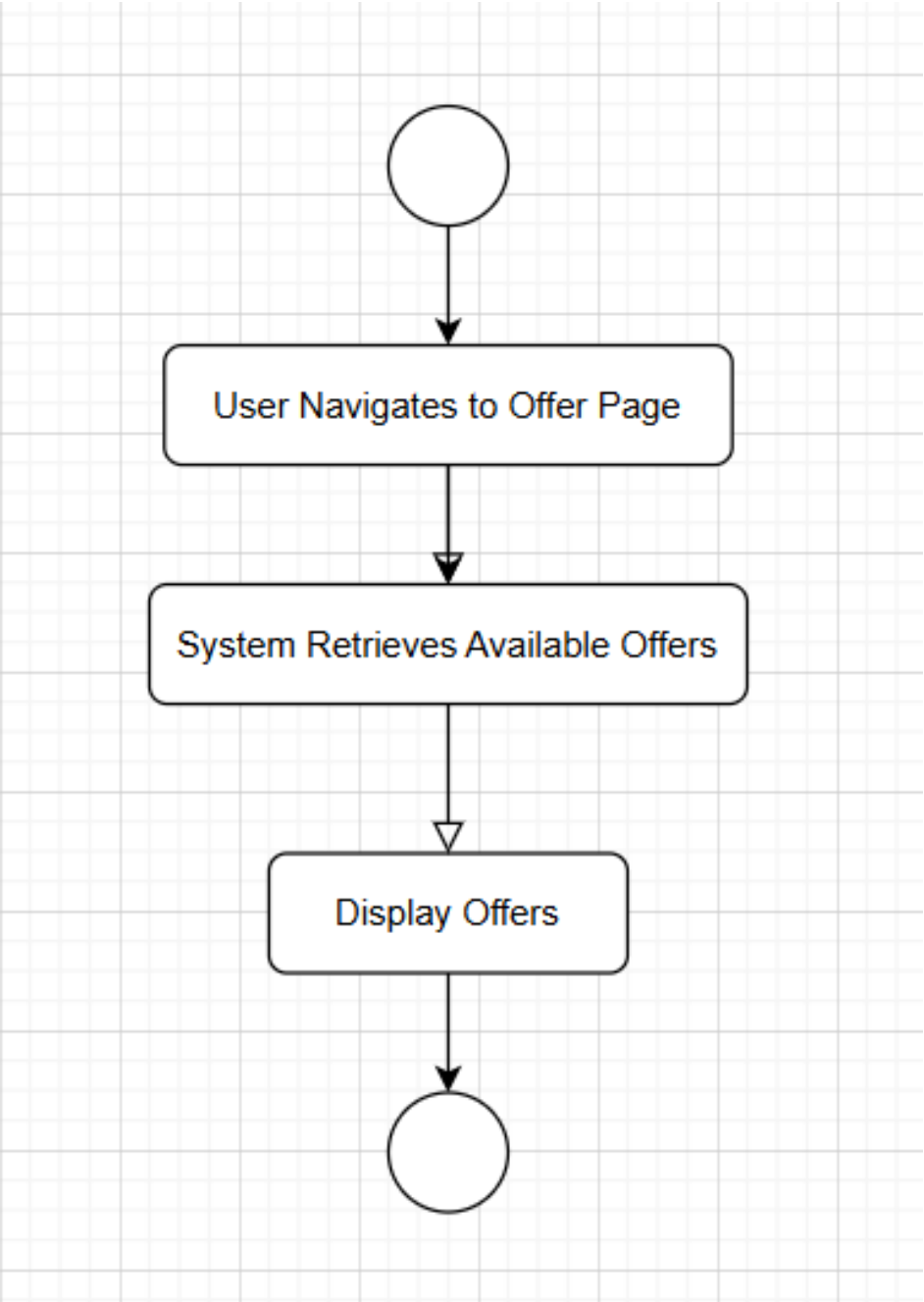
Activity Diagram: User Registration / Login



The user enters their email and password to log in. The system validates credentials and either grants access or shows an error message based on correctness

---

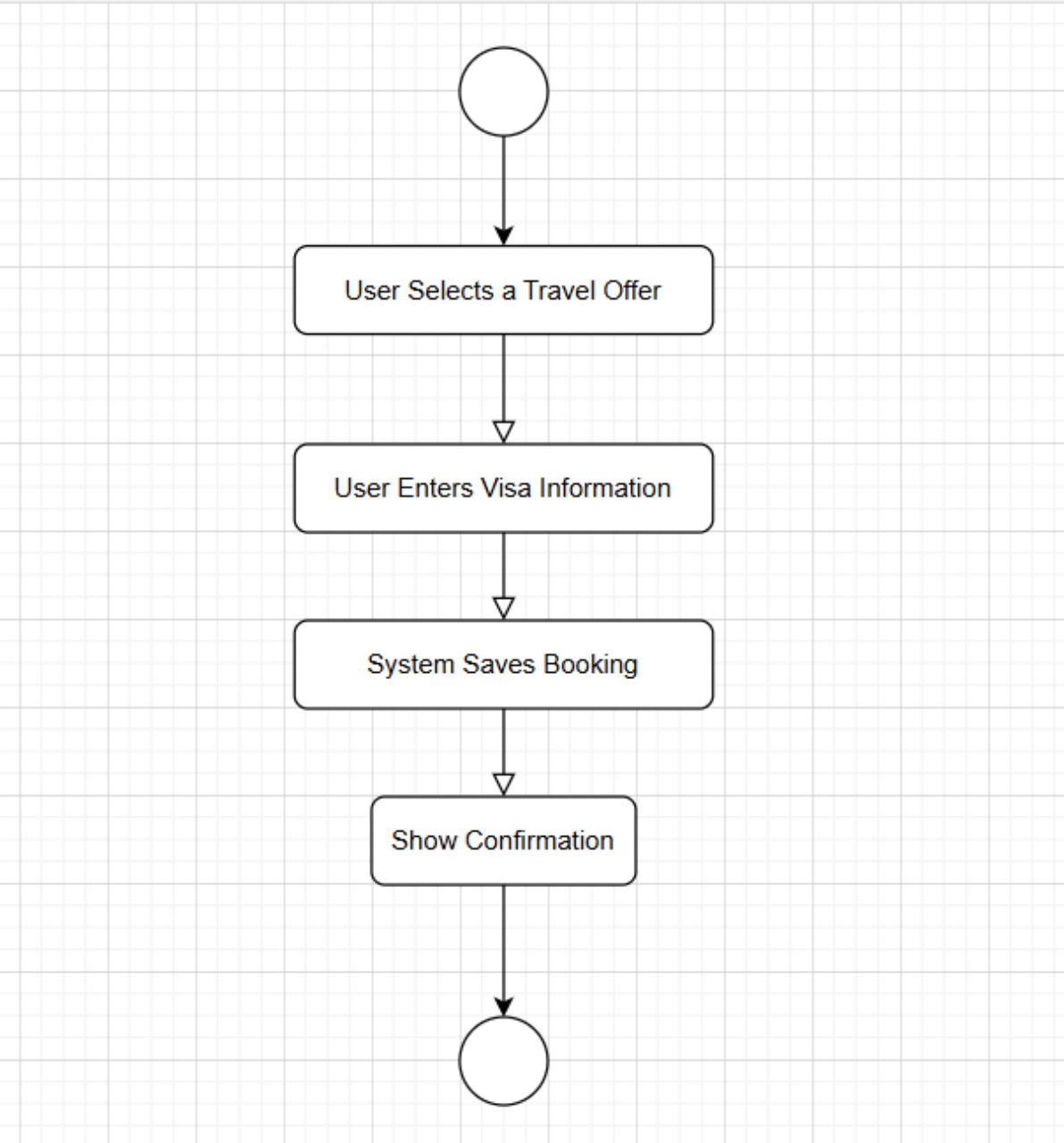
Activity Diagram: 🌍 View Travel Offers



The user navigates to the travel offers section. The system fetches and displays all available offers to the user in a readable format

---

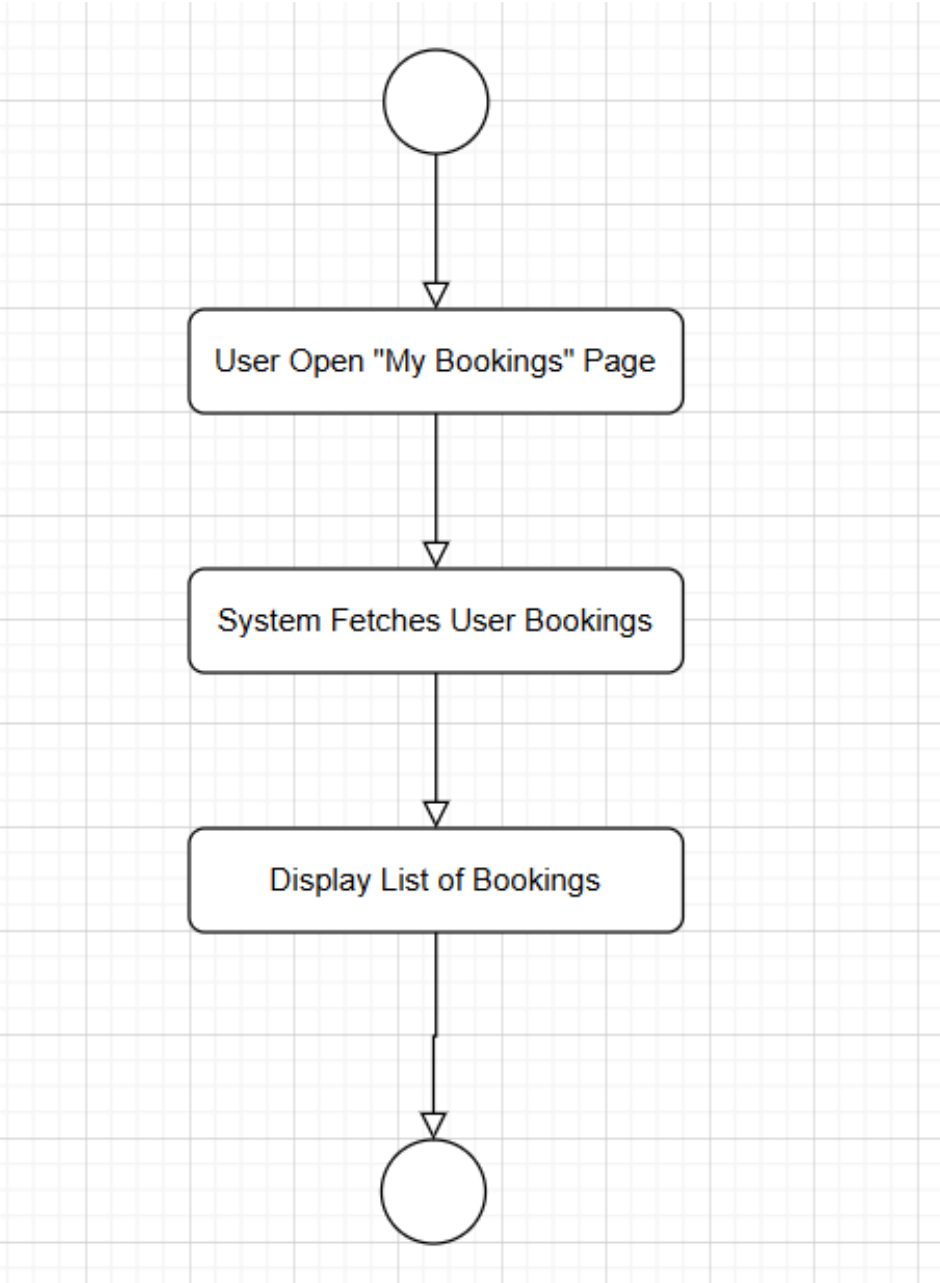
Activity Diagram: 🧮 Make a Booking



After selecting a travel offer, the user provides visa information. The system saves the booking and confirms it back to the user

---

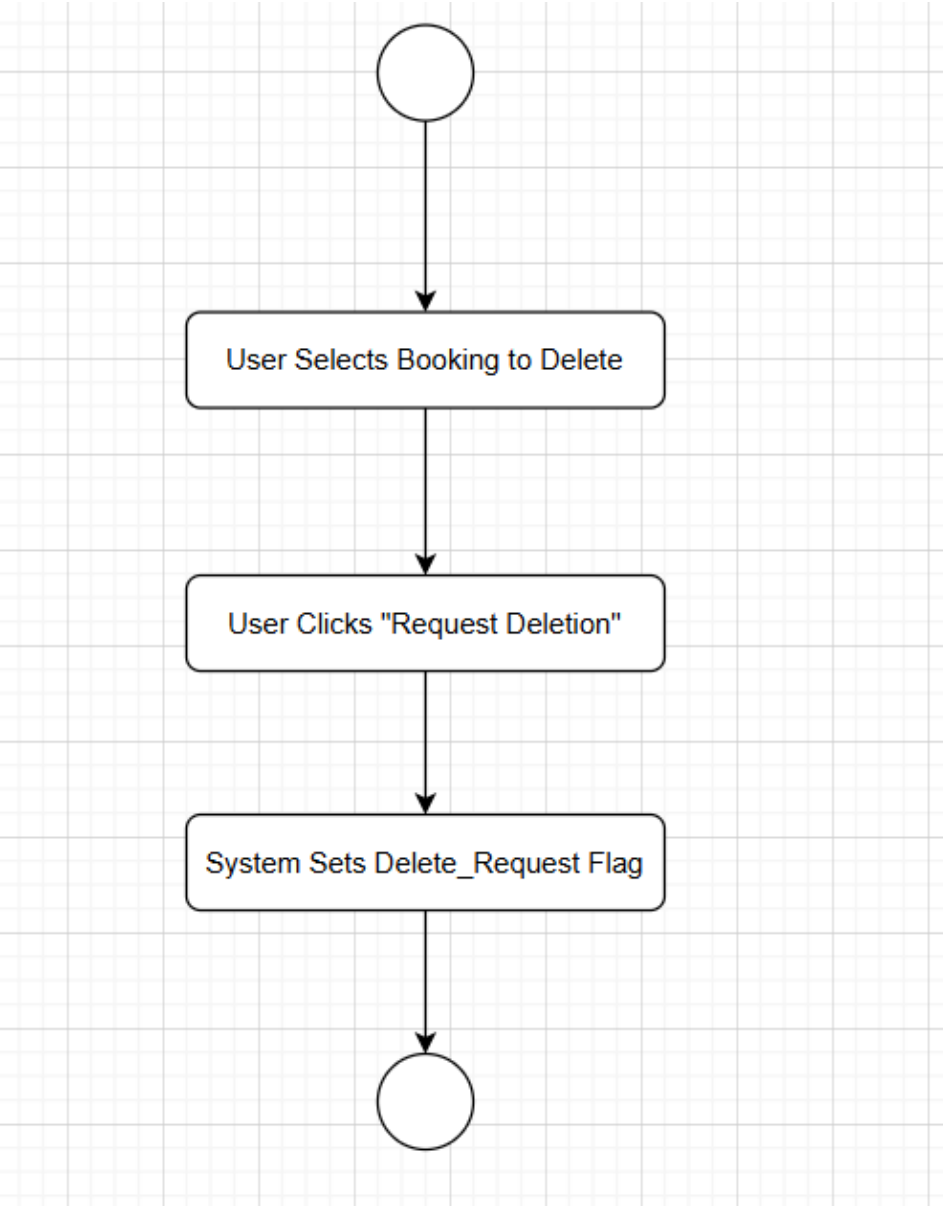
Activity Diagram: 🕮 View My Bookings



Users open their personal booking page. The system retrieves and displays all previous and current bookings made by the user

---

Activity Diagram: ❌ Request Booking Deletion



The user selects a specific booking and requests its deletion. The system marks the booking with a deletion request flag for admin action

---

Design & Implementation

Design Description

- The travel booking web app has two user roles: admins and users. Admins manage travel offers and bookings, while users can browse offers and make bookings.
- The system follows a three-tier architecture with a front-end (HTML, CSS, JavaScript), a back-end (PHP), and a database (MySQL).
- The database includes tables for users, admins, bookings, cities, travel offers, and contact messages, with foreign key relationships to ensure data consistency.

Implementation (Development Environment & Coding)

- Visual Studio is used for development, with PHP for back-end logic and MySQL for managing data. The front-end uses HTML, CSS, and JavaScript for an interactive user experience.
- The system follows the MVC pattern to separate the front-end, business logic, and database, ensuring maintainability and scalability. Features include user authentication, booking management, and dynamic content updates.

Testing

Development Testing

- **Unit Testing: Individual components, such as user authentication, booking creation, and admin management functionalities, were tested to ensure correct behavior. Each module was tested independently to verify that it meets functional requirements.**
- **Integration Testing: After unit tests, the interactions between the front-end and back-end, such as form submissions and data retrieval (offers, users)**
- **Usability Testing: The user interface was tested for ease of use and accessibility. This included testing forms, navigation, and responsiveness across different devices and screen sizes.**
- **Security Testing: The application underwent tests for vulnerabilities such as SQL injection, XSS attacks, and proper data validation (e.g., passwords) to ensure data integrity and security.**

Release Testing

- **System Testing: A comprehensive test of the entire system was performed, including testing the database, user role management (admin vs. user), and end-to-end booking process. This ensures that the system works as expected in a real-world environment.**
- **Acceptance Testing: Before release, the system was tested by potential users to confirm that it meets the original requirements and expectations. The user feedback helped in identifying any final issues or improvements before deployment.**
