CS5200 PROJECT FINAL REPORT

README

Project Setup and Execution Instructions

Prerequisites:

- **Operating System:** Windows, macOS
- **Python Version:** 3.6 or higher
- **MySQL Server Version:** 8.0 or higher
- **MySQL Workbench**

Required Libraries and Software:

1. **Python 3.x:** Download and install Python from the official website:

   https://www.python.org/downloads/

2. **MySQL Server and Workbench:**

   i. Download and install MySQL Server from the official website: https://dev.mysql.com/downloads/mysql/

   ii. Download MySQL Workbench from the official website: https://dev.mysql.com/downloads/workbench/ for GUI-based database management.

3. **Python Libraries:**

   - Install the required Python libraries using pip:

     - pip install pymysql

     - pip install tabulate

Installation Directories:

- **Python Script:** Place the Python script appGUI.py in a directory of your choice.

- **SQL Scripts:** Place the SQL scripts flight_tracker_dump.sql in the same directory as the Python script for ease of access.

Setup Instructions:

1. **Set Up the Database:**

   a. Open a terminal or command prompt.

   b. Navigate to the directory containing SQL scripts.

   c. Connect to your MySQL server

   d. Once connected, execute the SQL scripts

2. **Configure Database Connection in Python Script:**

   a. Open appGUI.py in a text editor.

   b. Ensure that the connection parameters match your MySQL server settings.

3. **Run the Python Application:**

   a. In the terminal or command prompt, navigate to the directory containing appGUI.py.

   b. In line 11 and line 12, replace default username and password to your localhost's username and password

   c. Run the python script.

4. **Login Credentials:**

   a. **Administrator:**
      i. Username: admin1 and Password: 123456
      ii. OR Username: admin2 and Password: 01234567

   b. **Passengers and Viewers:** No login is required; you can proceed directly to the respective menus.

Note: Ensure that your MySQL server is running before executing the Python script.

Technical Specifications

Project Overview:

The Flight Tracker Database Project is a terminal-based application that allows administrators, passengers, and viewers to interact with flight schedule data. The primary goal is to manage and track airport flight schedules efficiently.

Software and Technologies Used:

- **Programming Language:** Python 3.x
- **Database Management System:** MySQL 8.0
- **Development Tools:**
    - **MySQL Workbench:** For database design and management
    - **Python Libraries:**
        - pymysql: For connecting Python to the MySQL database
        - tabulate: For displaying data in a tabular format in the terminal
- **Operating System Compatibility:** Cross-platform (Windows, macOS, Linux)

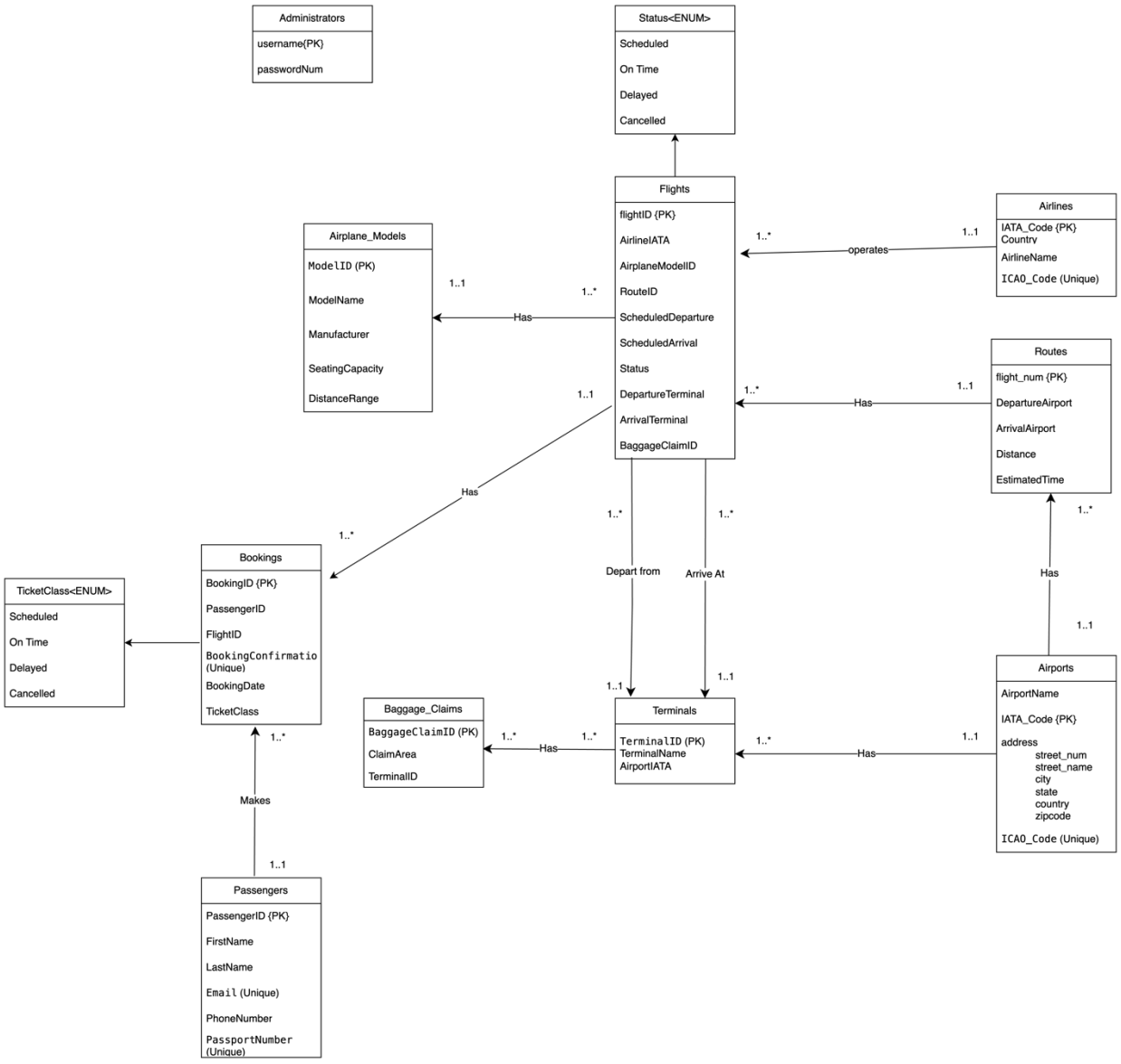
Frameworks and Libraries:

- **pymysql:** A pure-Python MySQL client library.
    - **Download Page:** https://pypi.org/project/PyMySQL/
- **tabulate:** A Python library for creating simple ASCII tables.
    - **Download Page:** https://pypi.org/project/tabulate/

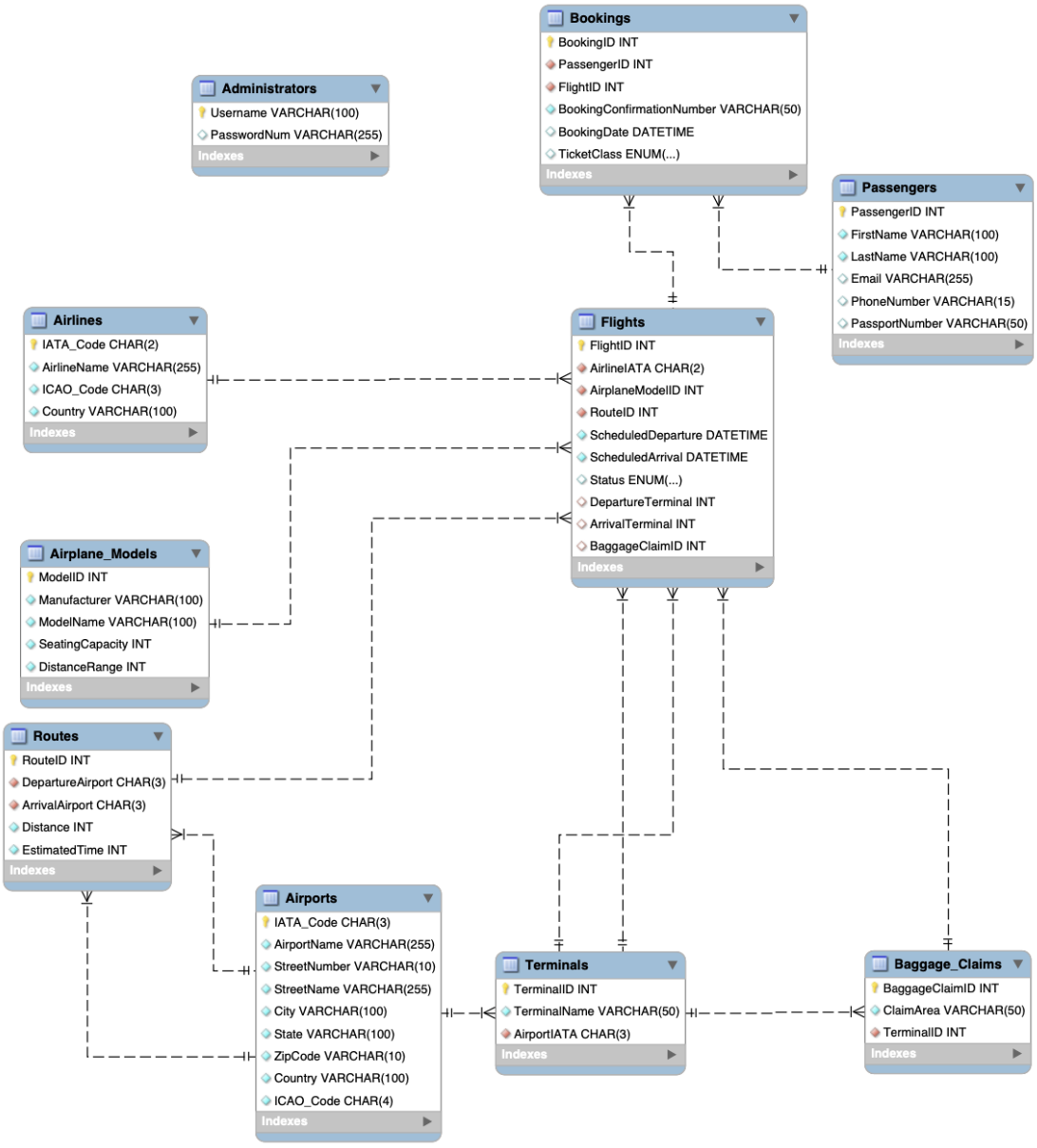
Machine Requirements:

- Any machine capable of running Python 3.x and MySQL Server.
- Sufficient permissions to install Python libraries and MySQL.

Conceptual Design (UML Diagram)



Logical Design (Database Schema)



1. Airports

- **Attributes**:
    - IATA_Code (PK)
    - AirportName
    - StreetNumber

    - StreetName
    - City
    - State
    - ZipCode
    - Country
    - ICAO_Code (Unique)
- **Description**: This table represents airports, including their codes and location details.
- **Constraints**:
    - IATA_Code as the primary key.
    - ICAO_Code and AirportName are unique to avoid duplicate entries.

2. Airlines

- **Attributes**:
    - IATA_Code (PK)
    - AirlineName
    - ICAO_Code (Unique)
    - Country
- **Description**: Represents airlines with their unique codes and operational country.
- **Constraints**:
    - IATA_Code as the primary key.
    - ICAO_Code and AirlineName must be unique.

3. Airplane Models

- **Attributes**:
    - ModelID (PK)
    - Manufacturer
    - ModelName

    - SeatingCapacity
    - DistanceRange
- **Description**: Contains details about airplane models and their specifications.
- **Constraints**:
    - ModelID as the primary key.
    - Ensures unique combinations of Manufacturer and ModelName.

4. Terminals

- **Attributes**:
    - TerminalID (PK)
    - TerminalName
    - AirportIATA (FK → Airports.IATA_Code)
- **Description**: Represents terminals within airports.
- **Constraints**:
    - TerminalID as the primary key.
    - AirportIATA as a foreign key referencing Airports.

5. Baggage Claims

- **Attributes**:
    - BaggageClaimID (PK)
    - ClaimArea
    - TerminalID (FK → Terminals.TerminalID)
- **Description**: Represents baggage claim areas within terminals.
- **Constraints**:
    - BaggageClaimID as the primary key.
    - TerminalID as a foreign key referencing Terminals.

6. Routes

- **Attributes**:

- RouteID (PK)
- DepartureAirport (FK → Airports.IATA_Code)
- ArrivalAirport (FK → Airports.IATA_Code)
- Distance
- EstimatedTime

- **Description**: Defines flight routes between airports.
- **Constraints**:
  - RouteID as the primary key.
  - Unique combination of DepartureAirport and ArrivalAirport.
  - Both DepartureAirport and ArrivalAirport are foreign keys referencing Airports.

7. Flights

- **Attributes**:
  - FlightID (PK)
  - AirlineIATA (FK → Airlines.IATA_Code)
  - AirplaneModelID (FK → Airplane_Models.ModelID)
  - RouteID (FK → Routes.RouteID)
  - ScheduledDeparture
  - ScheduledArrival
  - Status (ENUM: 'Scheduled', 'On Time', 'Delayed', 'Cancelled')
  - DepartureTerminal (FK → Terminals.TerminalID)
  - ArrivalTerminal (FK → Terminals.TerminalID)
  - BaggageClaimID (FK → Baggage_Claims.BaggageClaimID)
- **Description**: Represents individual flights, including schedule, status, and terminals.
- **Constraints**:

    - FlightID as the primary key.
    - Unique combination of RouteID, ScheduledDeparture, and AirlineIATA.

8. Passengers

- **Attributes**:
    - PassengerID (PK)
    - FirstName
    - LastName
    - Email (Unique)
    - PhoneNumber
    - PassportNumber (Unique)
- **Description**: Contains information about passengers.
- **Constraints**:
    - PassengerID as the primary key.
    - Email and PassportNumber must be unique.

9. Administrators

- **Attributes**:
    - Username (PK)
    - PasswordNum
- **Description**: Stores administrator login credentials.
- **Constraints**:
    - Username as the primary key.

10. Bookings

- **Attributes**:
    - BookingID (PK)
    - PassengerID (FK → Passengers.PassengerID)
    - FlightID (FK → Flights.FlightID)

    - BookingConfirmationNumber (Unique)
    - BookingDate
    - TicketClass (ENUM: 'Economy', 'Business', 'First')
- **Description**: Represents flight bookings made by passengers.
- **Constraints**:
    - BookingID as the primary key.
    - Unique BookingConfirmationNumber.

As we can see from the diagram above. The logical design of the provided database schema consists of ten interrelated entities, each representing a key component of the flight tracking system. The **Airports** table defines each airport with attributes such as IATA_Code (primary key), AirportName, and location details like City, State, and Country. It enforces unique constraints on ICAO_Code and AirportName to prevent duplicates. The **Airlines** table stores airline information, using IATA_Code as the primary key, along with attributes like AirlineName and ICAO_Code, which are also unique. The **Airplane_Models** table represents specifications for airplane models, with attributes such as Manufacturer, ModelName, SeatingCapacity, and DistanceRange, ensuring unique combinations of manufacturer and model names.

The **Terminals** table links terminals to airports via the AirportIATA foreign key and includes TerminalID as the primary key. Associated with terminals is the **Baggage_Claims** table, which represents baggage claim areas, using BaggageClaimID as the primary key and referencing the TerminalID foreign key. The **Routes** table defines flight paths between airports with a RouteID primary key and foreign keys DepartureAirport and ArrivalAirport, which reference the IATA_Code of airports. Each route includes details like Distance and EstimatedTime and enforces a unique combination of departure and arrival airports.

The **Flights** table captures flight schedules and operational details. It includes FlightID as the primary key and foreign keys linking to the relevant airline (AirlineIATA), airplane model (AirplaneModelID), route (RouteID), terminals (DepartureTerminal and ArrivalTerminal), and baggage claims (BaggageClaimID). Attributes like ScheduledDeparture, ScheduledArrival, and Status (with ENUM values such as 'Scheduled' or 'Cancelled') provide flight-specific data, and a unique constraint ensures no duplicate flights for the same route, departure time, and airline. The **Passengers** table tracks traveler details with

attributes like PassengerID (primary key), Email, and PassportNumber, ensuring uniqueness for personal identifiers.

The **Administrators** table manages system users with Username as the primary key, storing login credentials for administrative tasks. The **Bookings** table links passengers to flights, with BookingID as the primary key and foreign keys referencing Passengers and Flights. Additional attributes like BookingConfirmationNumber (unique), BookingDate, and TicketClass (ENUM: 'Economy', 'Business', 'First') provide reservation details. Overall, this logical design ensures normalization up to 3NF by minimizing redundancy and dependency issues, validates relationships with referential integrity constraints, and incorporates ENUM constraints to standardize attribute values. The schema is scalable and prepared for real-world use cases, allowing future enhancements like new terminals, routes, or airlines without disrupting existing functionality.

User Flow of the System

The application features a terminal-based user interface with three main user roles: Administrator, Passenger, and Viewer.

Main Menu

Upon launching the application, users are presented with the following options:

1. **Administrator**
2. **Passenger**
3. **Viewer**
4. **Exit**

Administrator Flow

Authentication:

- The administrator is prompted to enter a username and password.
- Authentication is verified against the Administrators table.

Administrator Menu Options:

1. **Add an Administrator:**

    a. Input new admin username and password.
    b. Calls the AddAdministrator stored procedure.

2. **Add a Flight:**

    a. Inputs required flight details such as airline, airplane model, route, scheduled times, status, terminals, and baggage claim.
    b. Calls the AddFlight stored procedure.

3. **Update Flight Status:**

    a. Inputs flight ID and new status.
    b. Calls the UpdateFlightStatus stored procedure.

4. **Back to Main Menu:**

    a. Returns to the main menu.

Passenger Flow

Options:

1. **Enter First Name and Last Name:**

    a. Inputs first and last name.
    b. Calls the GetPassengerBookings stored procedure to retrieve booking confirmation numbers.
    c. For each booking, calls GetFlightInfoByConfirmation to display flight details.

2. **Enter Booking Confirmation Number:**

    a. Inputs booking confirmation number.
    b. Calls the GetFlightInfoByConfirmation stored procedure to display flight details.

3. **Back to Main Menu:**

    a. Returns to the main menu.

Viewer Flow

Process:

1. **Select Airport:**

    a. Displays a list of available airport IATA codes by calling GetAirportIATACodes.
    b. User inputs the desired airport code.

2. **Select Date:**

    a. Displays available flight dates by calling GetFlightDates.
    b. User inputs the desired date.

3. **Choose to View Departing or Arriving Flights:**

    a. Options:

i. **View Departing Flights:** Calls GetDepartingFlights with the selected date and airport.

ii. **View Arriving Flights:** Calls GetArrivingFlights with the selected date and airport.

iii. **Back to Main Menu:** Returns to the main menu.

4. Display Flights:

a. Flights are displayed in a tabulated format using the tabulate library.

Lessons Learned

Technical Expertise Gained

- **Database Design:**
    - Learned to design a normalized relational database schema for complex data relationships.
    - Gained experience in defining primary keys, foreign keys, and constraints.
- **SQL Programming:**
    - Developed skills in writing stored procedures and functions in MySQL.
    - Handled error signaling and transaction management.
- **Python and MySQL Integration:**
    - Connected Python applications to a MySQL database using pymysql.
    - Executed stored procedures from Python and processed results.
- **User Interface Development:**
    - Created a terminal-based user interface.
    - Utilized the tabulate library for better data presentation.

Insights

- **Data Domain Insights:**
    - Gained an understanding of the aviation industry's data requirements.
    - Recognized the importance of accurate scheduling and status information.
- **Time Management:**
    - Realized the significance of iterative development and incorporating feedback promptly.
    - Managed time effectively to accommodate design changes.

Alternative Design Approaches

- **Initial vs. Final Design:**
    - Initially included a Route entity with a FlightNumber as an attribute, which was revised.
    - Moved scheduling and status attributes from Flights to a new Journeys entity in the initial redesign but later consolidated them into Flights based on feedback.
- **Elimination of Weak Entities:**
    - Removed the Route weak entity and integrated origin and destination airports directly into the Routes table.
- **Handling User Types:**
    - Simplified user roles by having separate tables for Administrators and Passengers instead of a general Users table with subtypes.

Non-Working Code

- **Stored Procedure Errors:**
    - Encountered issues with stored procedures not handling exceptions properly.
    - Resolved by adding appropriate error signaling and checking for existing records.
- **Data Integrity Constraints:**
    - Faced challenges with foreign key constraints during data insertion.
    - Corrected by ensuring that referenced records exist before insertion.

Future Work

Planned Uses of the Database

- **Flight Booking System:**
    - Extend functionality to allow passengers to book flights directly through the application.
- **Real-Time Updates:**
    - Integrate real-time flight tracking APIs to update flight statuses automatically.

Potential Areas for Added Functionality

- **User Registration and Authentication:**
    - Implement a full-fledged user authentication system for passengers and administrators.
- **Graphical User Interface (GUI):**
    - Develop a web-based or desktop GUI for enhanced user experience.
- **Data Analytics:**
    - Incorporate analytics features for administrators to monitor operational metrics and performance.
- **Notifications System:**
    - Enable email or SMS notifications for passengers regarding flight status changes.

Justification for No Future Uses

- **Limited Scope:**
    - The current scope focuses on flight tracking and schedule management. Expanding beyond this may dilute the project's primary objectives.