# Airline Ticket Reservation System — README

This document explains how to set up, run, and grade the Part 3 web-based Airline Ticket Reservation System. It also lists the files and what each file does, and maps features to the project requirements.

## 1) Quick Start

### Go to the project root (the folder containing app.py)

```
cd "/CS-3083 Database/project/part3"
```

### Create (or refresh) a Python virtual environment

```
python3 -m venv .venv
```

### Activate it

macOS/Linux:

```
source .venv/bin/activate
```

Windows PowerShell:

```
.\.venv\Scripts\Activate.ps1
```

### Optional but recommended

```
python -m pip install -U pip
```

### Install dependencies

If requirements.txt exists:

```
python -m pip install -r requirements.txt
```

Otherwise:

```
python -m pip install Flask PyMySQL python-doten
```

Run the app

```
python app.py
```

then open http://127.0.0.1:5000 in browser

If port 5000 is busy, stop the other process or set PORT=5050 in .env and make app.py read it.

## 2) Database Setup

Create a MySQL database (Airline Ticket Reservation System).

In a MySQL client / phpMyAdmin, run:

- sql/create_tables.sql
- sql/insert.sql

### Environment variables (.env)

Create a file named .env in the project root and set:

```
FLASK_DEBUG=1
MYSQL_HOST=localhost
MYSQL_PORT=8889
MYSQL_USER=root
MYSQL_PASSWORD=root
MYSQL_DB=Airline Ticket Reservation System
SECRET_KEY=dev
```

## 3) File Index (what's in each file)

```
app.py
  Flask app & routes:
  - Auth: login/logout (customer & staff), session handling
  - Customer: home, search, purchase (name/number validation), my flights,
reviews
  - Staff: view flights with filters, create flight, change status,
           add airplane, view ratings & comments, ticket-sales reports
  - DB: PyMySQL connection via env vars, prepared statements everywhere

templates/
  layout.html               Base template (nav + flash messages)
  home.html                 Public home (search link, login, register)
```

```
  login.html                Login form (customer & staff)
  register_customer.html    Customer registration
  register_staff.html       Staff registration

  customer_home.html        Customer dashboard
  customer_search.html      Search UI + buy form
  customer_myflights.html   Purchased flights
  customer_reviews.html     Ratings / comments UI

  staff_home.html           Staff "View Flights" (Default: next 30 days) +
filters
  staff_create_flight.html  Create Flight form (+ shows next 30 days list)
  staff_change_status.html  Update flight status
  staff_add_airplane.html   Add airplane (ownership check)
  staff_ratings.html        Avg rating + comments per flight
  staff_reports.html        Ticket sales (range / last month / last year)

static/
  styles.css                Optional styles

sql/
  create_tables.sql         Schema (Part 2 + constraints)
  insert.sql                Sample data (optional)

.env                        Environment config
requirements.txt            Python dependencies (if included)
README_file.md              This README
USE_CASES.md                All use cases description
Summary.md                  Team contribution
```

## 4) Running Details

### Activate env & run

```
cd /path/to/part3
python3 -m venv .venv
source .venv/bin/activate           # Windows:
.\.venv\Scripts\Activate.ps1
python app.py
```

### Demo accounts

- Customer: hw3345@nyu.edu (use the seeded password)
- Staff: JetBlue staff account (username(email)/password as seeded)

### Purchase constraints implemented

- Can only buy future flights and not CANCELLED flights.
- Card name must match the logged-in user's account name.

- Card number must be digits only.
- Defensive checks for missing fields and unknown flights.

### Staff filters / defaults

- View Flights default shows next 30 days for the staff's airline.
- Filters include period (current/future/past), date range, from/to IATA, cities.
- Create Flight page also lists next 30 days below the form.

## 5) Troubleshooting

- Cannot connect to DB: verify .env values and ensure MySQL is running.
- Port in use: change PORT or kill the other process.
- Packages missing: python -m pip install Flask PyMySQL python-dotenv.
- Unicode/locale issues on Windows: run from PowerShell and make sure the console uses UTF-8.