Department of Electrical and Computer Engineering
Principles of Software Design
ENSF 614 – Fall 2025
Term Project

M. Moussavi, PhD, P.Eng

Total Mark: 100

Introduction:

In this assignment you are supposed to practice a complete process of design and development of a software project, using a systematic design methodology.

The last part of the course is focused on system level design and architecture and will be achieved in an "**Active Learning"** pedagogic approach. It means while theoretical concepts will be discussed in lectures, the corresponding practical techniques to build some design components will be developed in the classroom, or during the lab periods. This method shows you the initial directions and helps you to have a better vision on how to continue the details later.

When you are working as a member of group you should assume a full responsibility and your commitments must be achieved at your best capacity. All group members should be available during the lectures and labs, to participate in class/group discussions, and to achieve their own portion of work.

Flight Reservation Application

Project Overview: The purpose of this project is to **analyze, design, and develop** a stand-alone **Flight Reservation System** using **object-oriented principles** and **domain-driven design** concepts. The primary focus will be on **Domain Analysis and Design**, including class modeling, associations, use-case definition, and applying suitable **design patterns**. The project will simulate the operations of a flight booking system for a **single-user GUI** (not a web or client-server system).

The system should allow a user—acting as a **customer**, **flight agent**, or **system administrator**—to:

- View available flights and flight details.
- Make, cancel, or modify a reservation.
- View booking history or flight schedule.
- Manage flight information (add, update, delete flights).
- Maintain customer information and payment record.

Users Roles:

Role	Description	Key Capabilities
Customer	End-user booking flights	Search flights, make/cancel bookings, view reservations, etc.
Flight Agent	Employee managing bookings	Manage customer data, modify reservations, view schedules, etc.
System Administrator	Administer system data	Add/remove flights, manage routes and aircraft, update schedules, etc.

Some of the Functional Requirements:

- **Flight Search & View**
    - Search by origin, destination, date, and airline.
    - Display available seats, flight time, and price.
- **Booking Management**
    - Make new bookings.
    - Cancel or modify existing reservations.
    - Generate booking confirmation.
- **Customer Management**
    - Add new customer profiles.
    - View or edit customer details.
- **Flight Management (Admin only)**
    - Add new flights or update existing flight details.
    - Remove outdated flights.
- **Payment Simulation**
    - Record payment information (simulated, not real processing).
- **Database Connectivity**
    - Use a small embedded/local database (MySQL for SQLite) for storing flight and reservation data.
- **Monthly Promotion News**
    - Customers receiving monthly promotion news (first day of each month)
- **Other, if needed**.

Some of the Non-Functional Requirements:

- **Usability:** Simple GUI for interaction (Java GUI).
- **Portability:** Runs as a stand-alone application on any desktop system.
- **Maintainability:** Use clear modular design with classes and layered architecture.
- **Scalability:** Design allows future extension (e.g., online booking, client server, etc).
- **Other, as needed.**

System Architecture:

A **3-layer design** is recommended:
1. **Presentation Layer:** GUI interface (view + input handling).
2. **Business Logic Layer:** Core classes representing domain entities and operations.
3. **Data Layer:** Handles persistence (database/file access).

Assessment Criteria:
At the end you project will be evaluated based on its functionalities and:
1. How well separation of concepts and different components of software architecture (entities, controllers, views, etc.) are considered and designed.
2. How will major elements of quality-design, such as: modularity, cohesion, and coupling are applied
3. How well the non-functional requirements such as user-friendliness, reusability, maintainability, reliability, etc. are addressed in your application.

4. How your system uses separation of concepts at different system level: presentation layer (view/boundary classes), domain layer (domain/entity classes), which are glued together with control classes.
5. How clear and accurate are you UML diagram, and how well they represent your final product.
6. How the project's requirements are presented in your use-case diagram, and how your use case diagram is traceable in your class diagram.
7. How well the use-cases in your use case diagram are presented in a use-case scenario, or a sequence diagram.

Expected Deliverables

Design Phase (50 marks)

In this phase you should submit a Design Document that includes a clear description of system's requirements, and design artefacts as follows:

- **Introduction to the system under the study**.
  In this part you should have following paragraphs:
  - Brief description of the system
  - Followed by describing some of the major and relatively more complex functionalities such as process of browsing/selecting a flight, process of booking a flight-ticket, and process making payment. In this part you can include four activity diagrams as follows:
    - An activity diagram that show the process login
    - An activity diagram that shows the process browsing/selecting a flight
    - An activity diagram that shows the process booking a flight
    - An activity diagram that only shows the process making payment
- **A systems use-case diagram.**
- **A detailed "Scenario" for each use case**:
  In this part of your report, you should write a scenario (storyboarding) for each use case in your use-case diagram. **All candidate objects and candidate operations should be underlined** (use single-underline for objects and double-underline for operations).
- **Four interaction diagrams:** for four major and important use cases in this system that appear as a control-class in your domain class diagram, you need to draw a sequence diagram.
- **Four state transition diagrams**: One that shows the state of the entire system, one for the process of making payment, one a reservation object, one for a flight object.
- **A class diagram:** that shows classes, relationships among them, attributes, behaviours, cardinalities, etc.
  Note:
  - You class diagram should be traceable in your **use-case** scenarios. Mark will be deducted for classes that are not traceable in the use-case diagram.
  - In your class diagram you don't need to show Java library classes such as Exceptions, Buttons, String, ArrayList, etc.
  - Make sure clearly identify the stereotypes of your classes (such as **`<<entity>>`, `<<control>>`, `<<boundary>>`**)
  - You don't need to show constructors/destructor, getters/setters.
  - **Note:** at the final stage you need to make the architecture of the system more: reusable, scalable, maintainable, reliable, and using necessary concepts such as modular design,

inheritance, realization, aggregation, composition, polymorphism, and appropriate design patterns as needed.

- **A Package Diagram,** that shows how the three layers of the system interact with each other. Make sure to show classes in each package and identify those that are public.

Implementation Phase (50 marks)

In this phase you will implement your proposed design in **Java GUI (using Swing)**, and **MySQL.**

What to submit:

A **zip** file that contains:

1. A **jar** file that contains all `.class` files.
2. A zip file that contains all `.java` files.

Due Date for both parts (Design and implementation): Sunday Nov 30th, before 12 PM. You need to demonstrate your working application, based on a schedule that will be posted on the D2L. During the demonstration, all team members MUST be present to assess their part of work. If a member is absent the member's implementation mark will be considered zero.