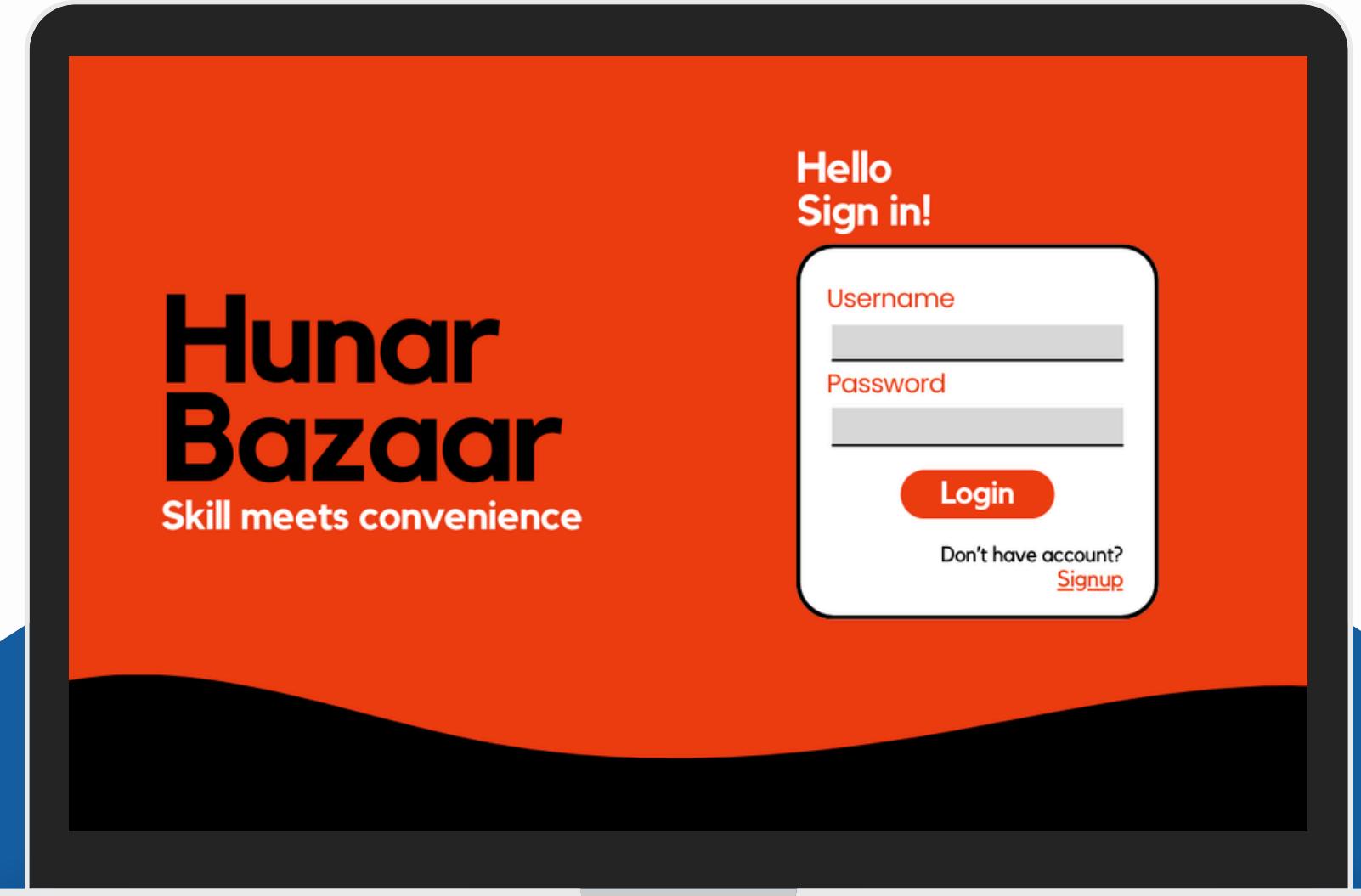




Hunar Bazaar

By: Ahmed, Usman & Irtaza



PURPOSE

The purpose of HunarBazaar is to create a seamless platform for connecting local technicians—such as plumbers, mechanics, and carpenters—with customers across Pakistan. By providing real-time, location-based solutions, the platform addresses the challenges of finding reliable service providers, especially during emergencies. The initial version of the product focuses on enhancing access to verified professionals in major cities like Karachi, Lahore, and Islamabad, ensuring convenience for users and growth opportunities for technicians.

JavaFX
UI LAYER

UI LAYER

The image displays the User Interface (UI) layer of the Hunar Bazaar platform. It features a prominent orange header with the brand name "Hunar Bazaar" and the tagline "Skill meets convenience". A central navigation bar includes links for "Orders", "Chats", and "Become a Seller". Below the header is a grid of service categories, each represented by a white rectangular box containing a skill name. The categories are arranged in four rows and three columns. The skills listed are: Appliance Technician, Blacksmith, Carpenter; Chef/Cook, Electrician, Glass Worker; Lock Smith, Mechanic, Mason; Painter, Plumber, Pest Controller; Plasterer, Sewage Cleaner, Technician; Tiler, Welder, Furniture Repairer.

Hunar
Bazaar

Skill meets convenience

Hello
Sign in!

Username

Password

Login

Don't have an account?

Hunar Bazaar

Orders Chats Become a Seller

Appliance Technician

Blacksmith

Carpenter

Chef/Cook

Electrician

Glass Worker

Lock Smith

Mechanic

Mason

Painter

Plumber

Pest Controller

Plasterer

Sewage Cleaner

Technician

Tiler

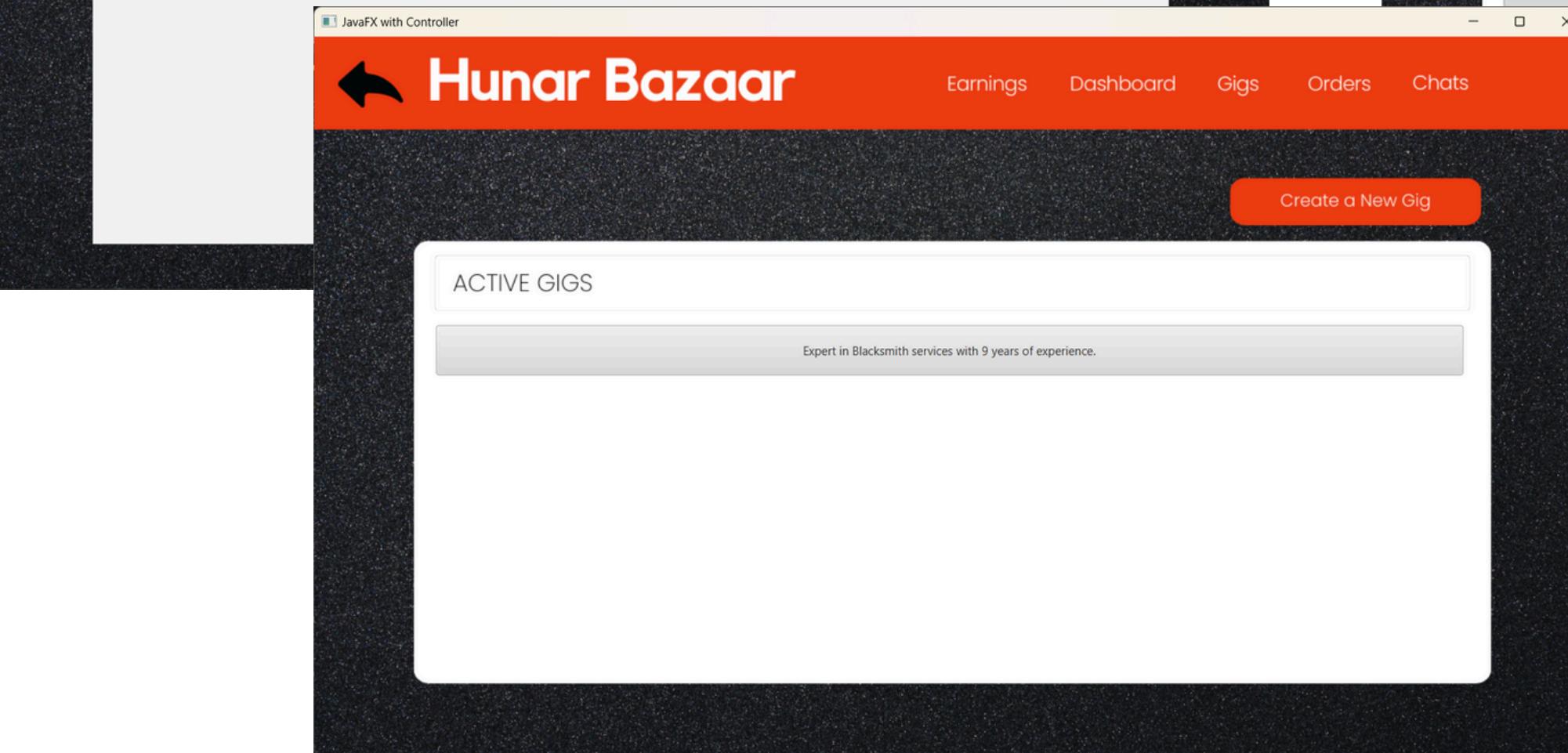
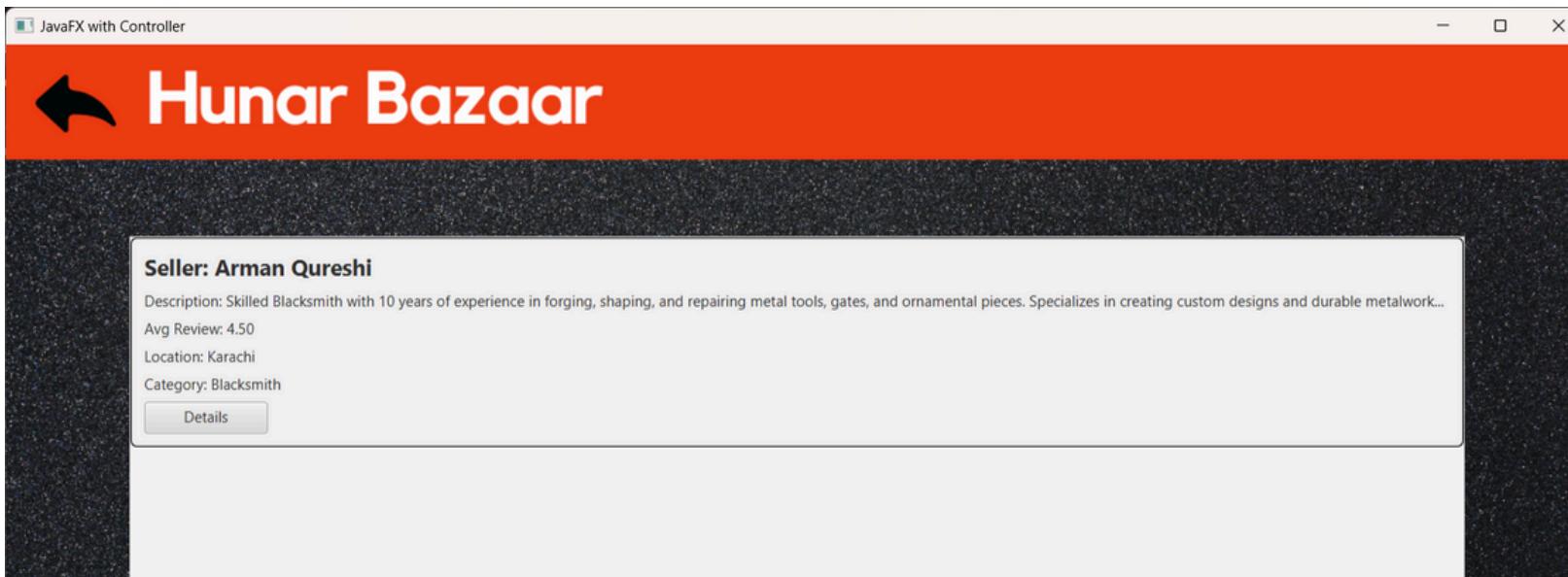
Welder

Furniture Repairer

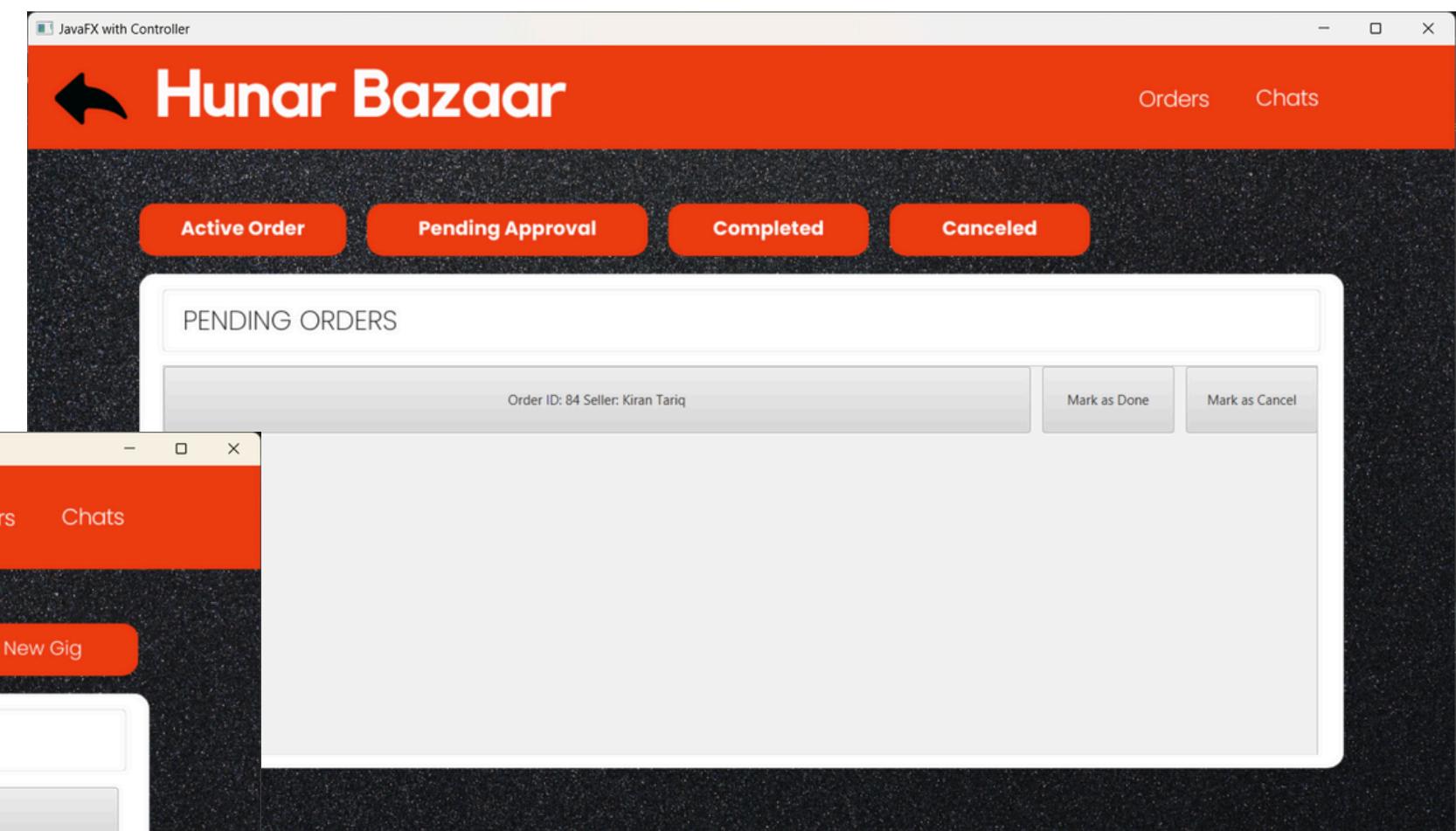
LOGIN

CATEGORIES

GIGS

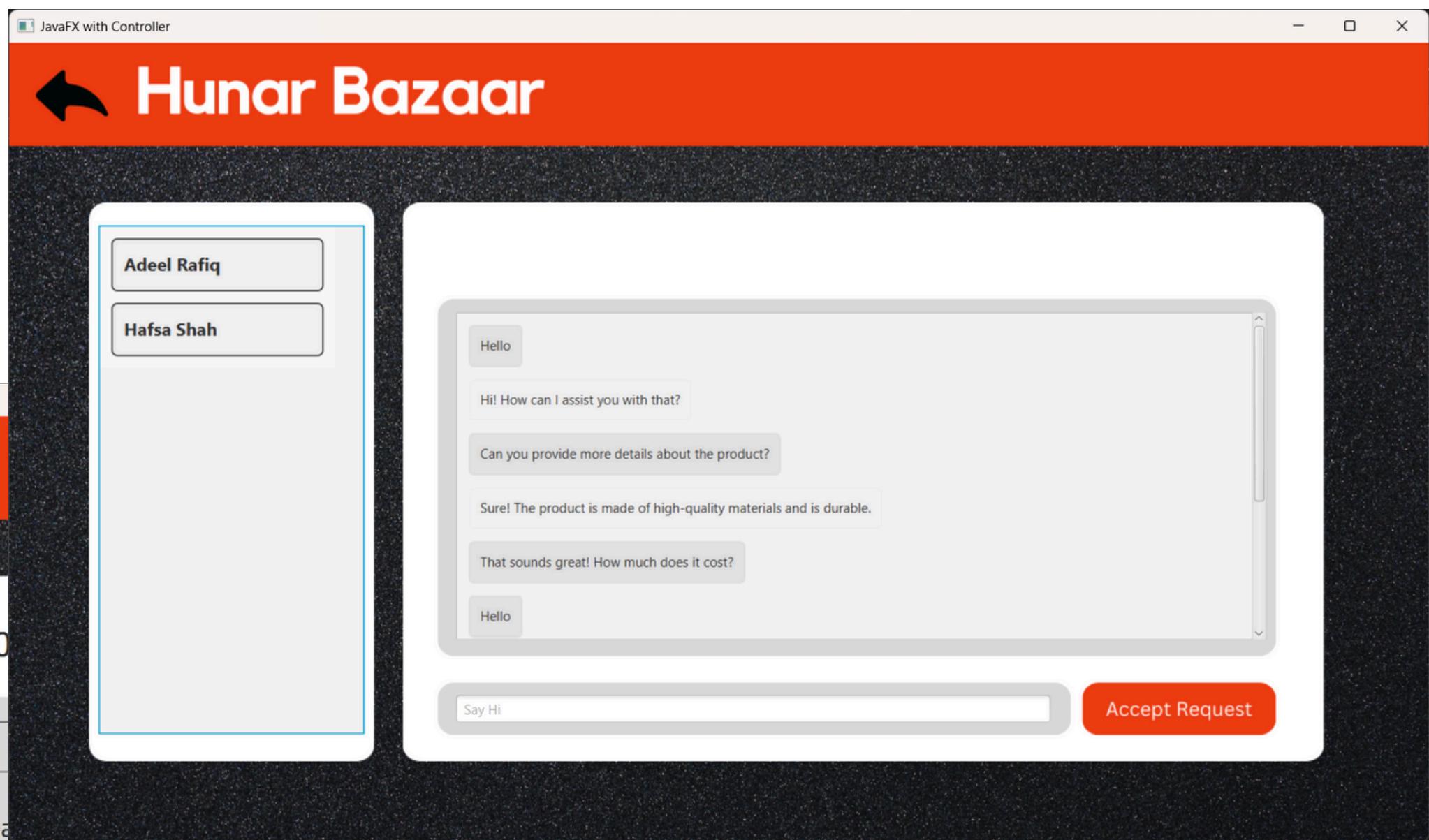
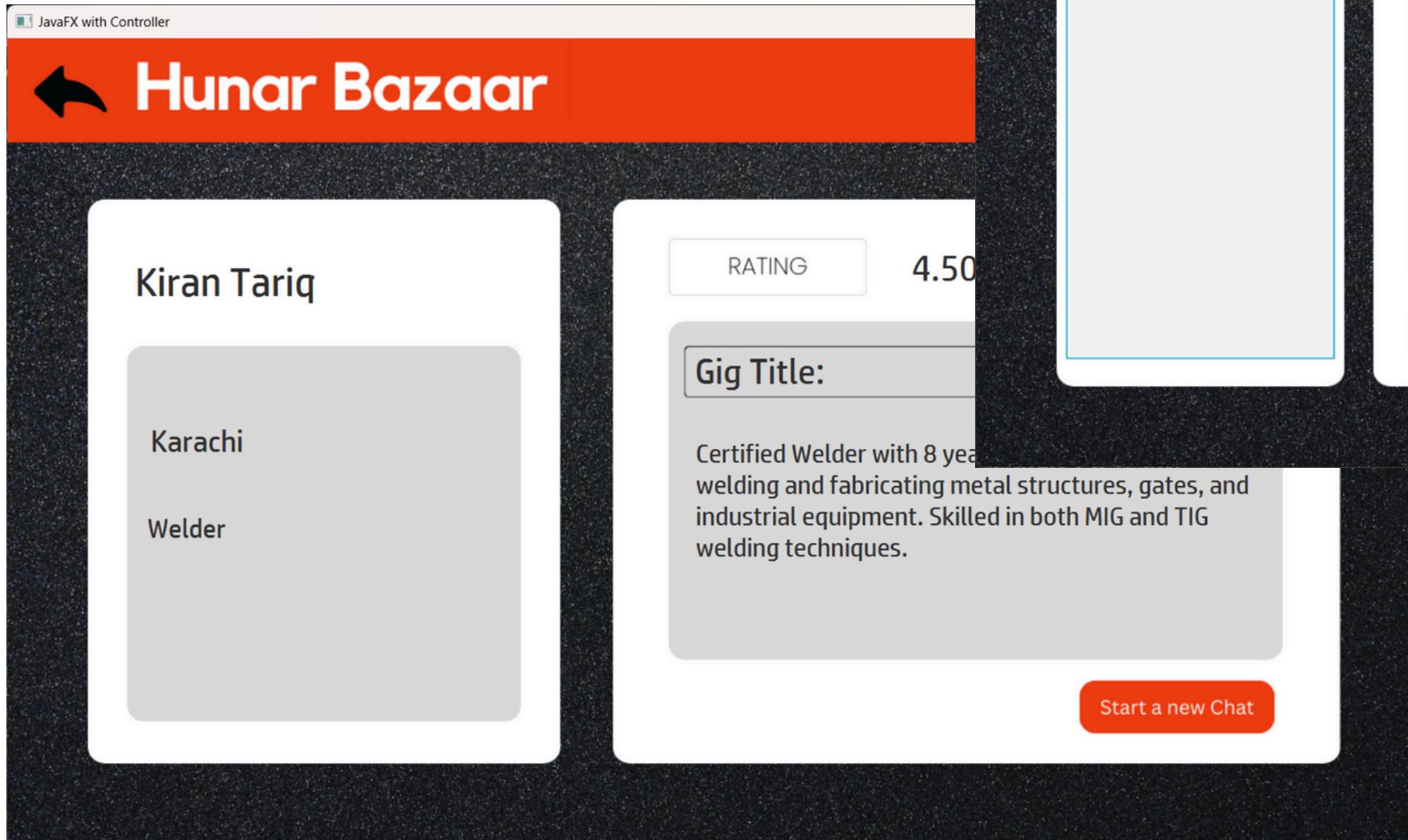


ORDERS



MANAGE GIG

INFO UI



CHAT UI

DESIGN PATTERNS

Low Coupling

High Cohesion

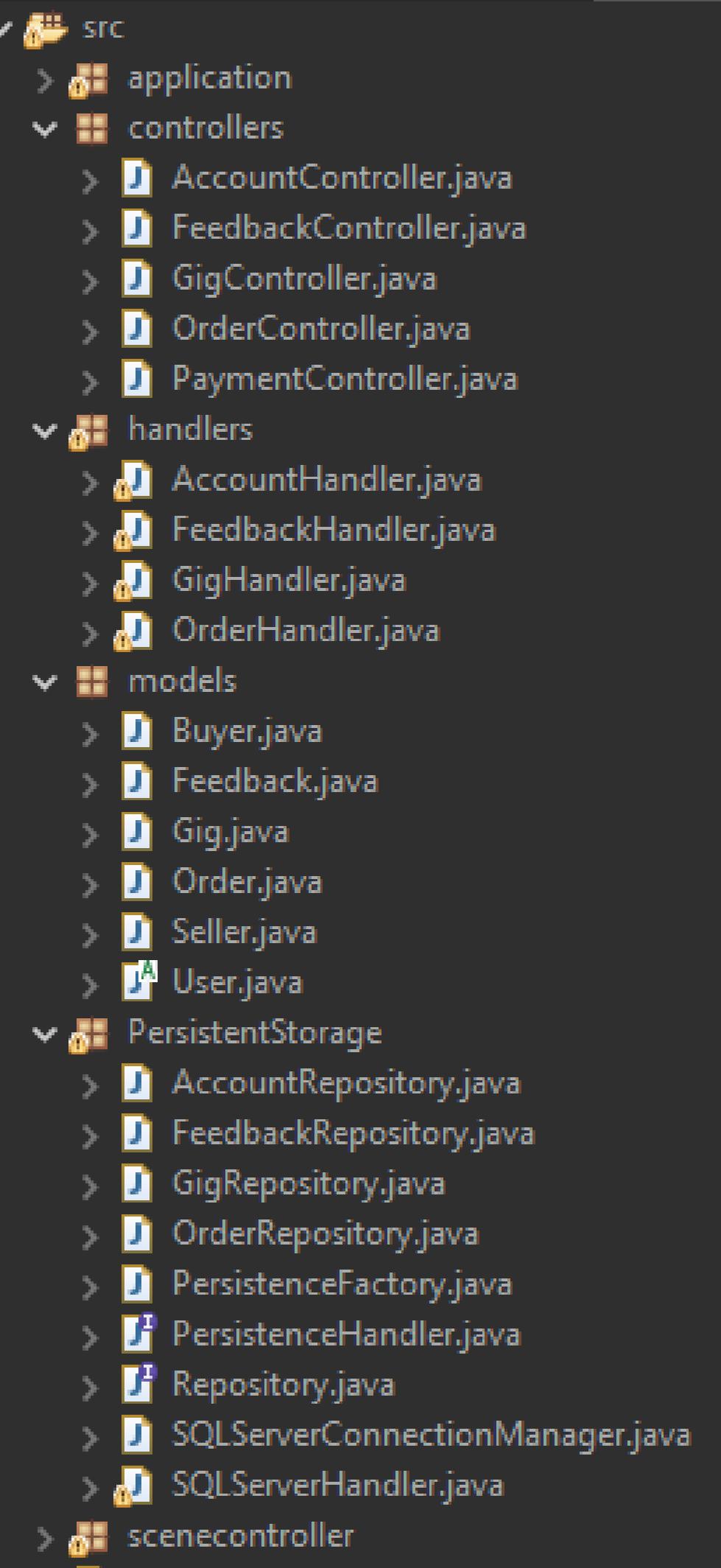
Factory

Singleton

Strategy

...

HIGH COHESION



```
src
  > application
  < controllers
    > AccountController.java
    > FeedbackController.java
    > GigController.java
    > OrderController.java
    > PaymentController.java
  < handlers
    > AccountHandler.java
    > FeedbackHandler.java
    > GigHandler.java
    > OrderHandler.java
  < models
    > Buyer.java
    > Feedback.java
    > Gig.java
    > Order.java
    > Seller.java
    > User.java
  < PersistentStorage
    > AccountRepository.java
    > FeedbackRepository.java
    > GigRepository.java
    > OrderRepository.java
    > PersistenceFactory.java
    > PersistenceHandler.java
    > Repository.java
    > SQLServerConnectionManager.java
    > SQLServerHandler.java
  < scenecontroller
```

```
package PersistentStorage;

+ import java.sql.Connection;[]

public class SQLServerConnectionManager {[]

    // Connection string for SQL Server
    private static final String CONNECTION_STRING =
        "jdbc:sqlserver://IRTAZA-KHAN\\SQLEXPRESS:1433;" +
        "Database=firsttemp;encrypt=false;trustServerCertificate=true;integratedSecurity=true;"

    /**
     * Get a connection to the SQL Server database.
     *
     * @return Connection object
     * @throws SQLException if a database access error occurs
     */
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(CONNECTION_STRING);
    }
}
```

SINGLETON
FACTORY

```
public class PersistenceFactory {  
  
    // Singleton instance  
    private static PersistenceFactory instance = null;  
  
    // Private constructor to prevent direct instantiation  
    private PersistenceFactory() {}  
  
    // Synchronized method to get the Singleton instance  
    public static synchronized PersistenceFactory getInstance() {  
        if (instance == null) {  
            instance = new PersistenceFactory();  
        }  
        return instance;  
    }  
  
    // Factory method to create persistence handlers  
    public PersistenceHandler createPersistenceHandler(String handlerType) {  
        if (handlerType.equalsIgnoreCase("SQLServer")) {  
            return new SQLServerHandler();  
        } //else if (handlerType.equalsIgnoreCase("MySQL")) {  
        //    return new MySQLHandler(); // Implement MySQLHandler similarly  
    //} //else if (handlerType.equalsIgnoreCase("XML")) {  
        //return new XMLPersistence();  
    //}  
    else if (handlerType.equalsIgnoreCase("MongoDB")) {  
        return new MongoDBHandler();  
    }  
    else {  
        throw new PersistenceException("Unsupported Persistence Handler: " + handlerType);  
    }  
}
```

STRATEGY

```
package strategy;

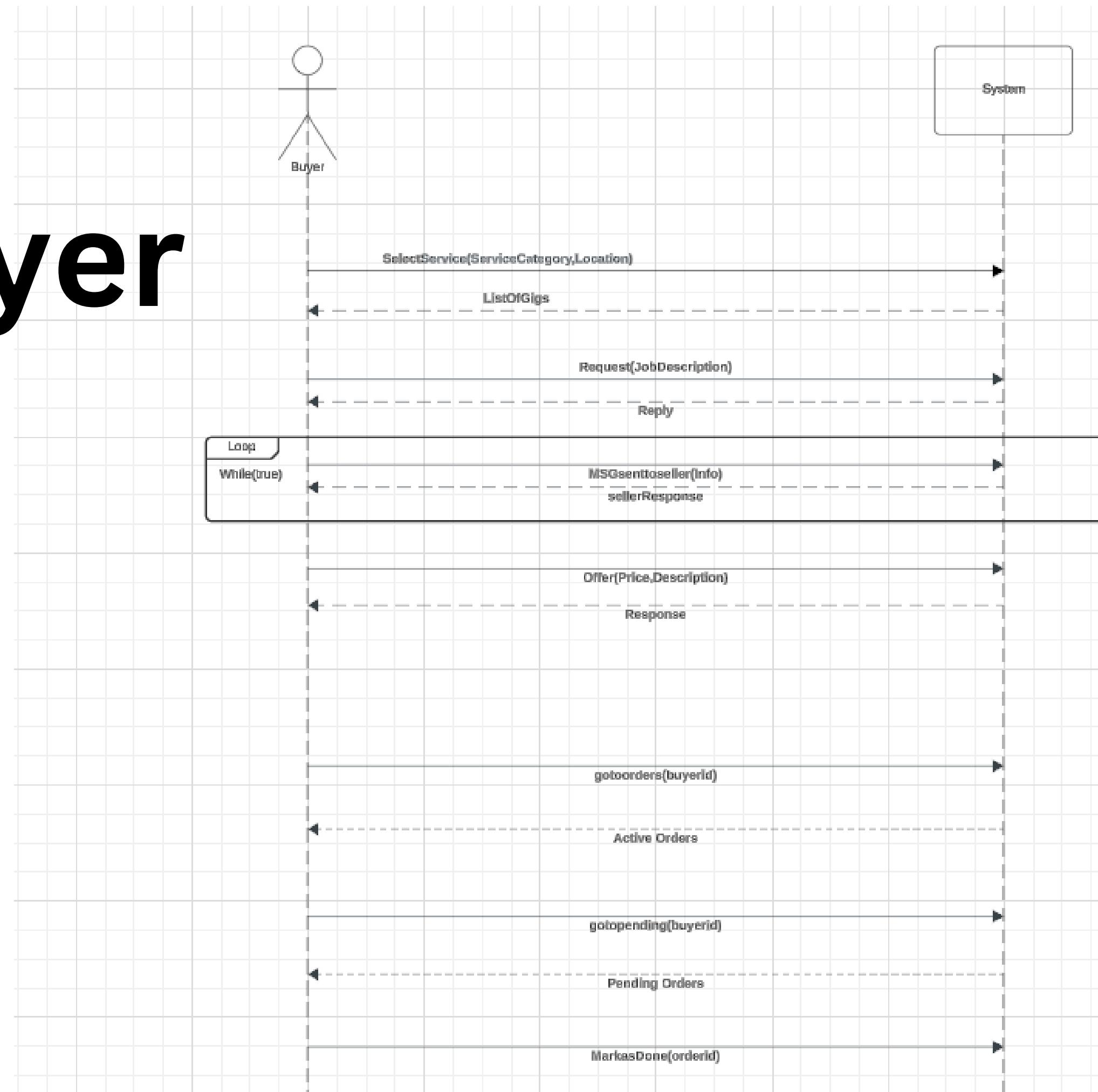
public class homepagecontext {
    private homepagestrategy homePageStrategy;

    public void setHomePageStrategy(homepagestrategy homePageStrategy) {
        this.homePageStrategy = homePageStrategy;
    }

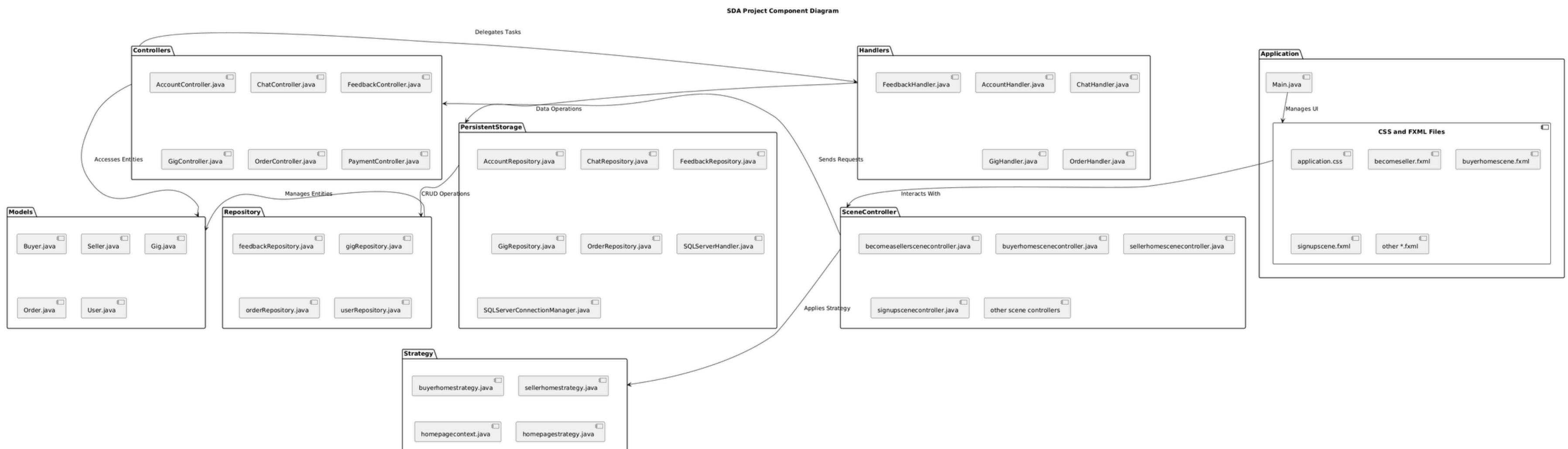
    public void displayHomePage() {
        if (homePageStrategy != null) {
            homePageStrategy.displayHomePage();
        } else {
            System.out.println("No strategy set!");
        }
    }
}
```

DIAGRAMS

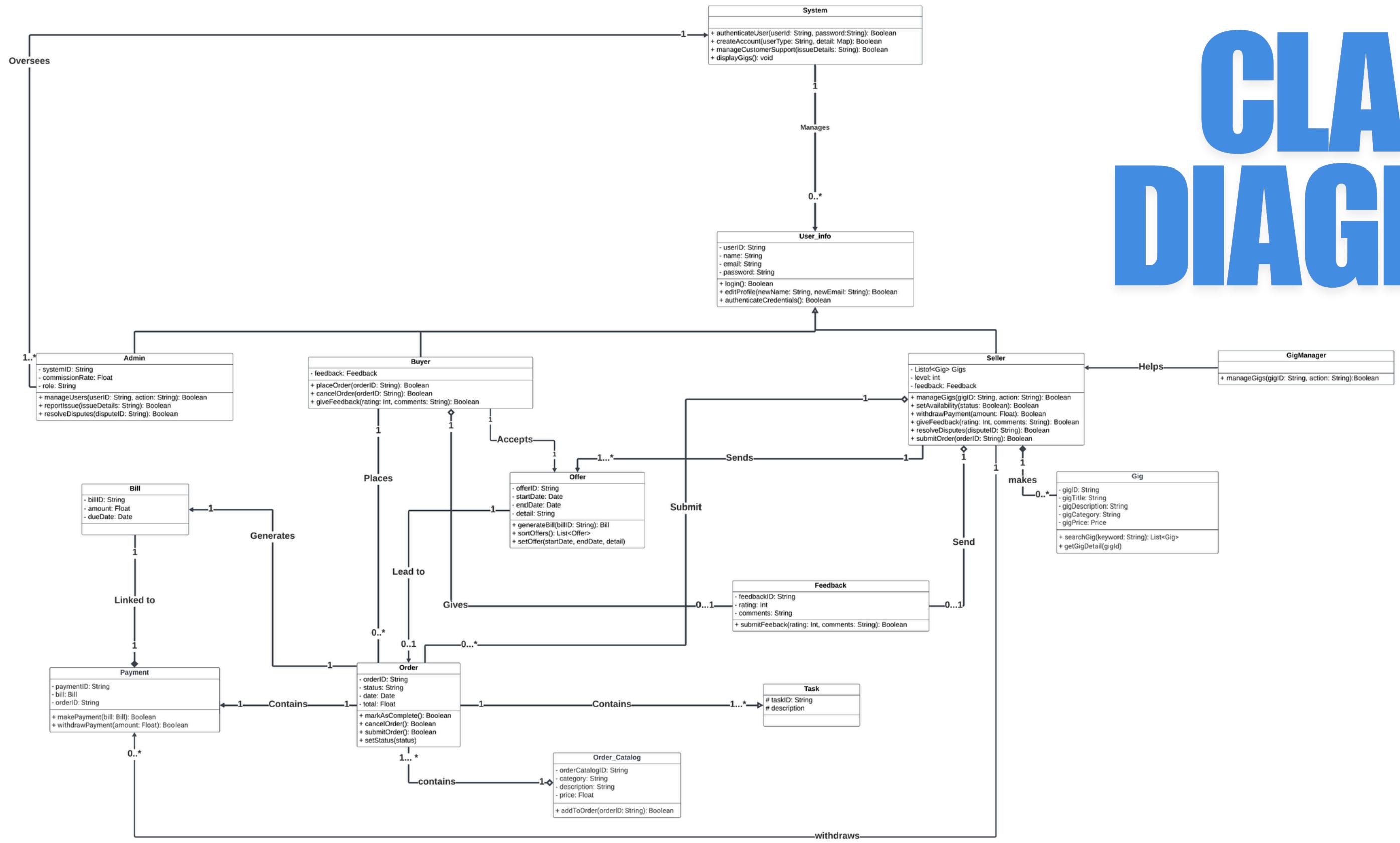
SSD for Buyer



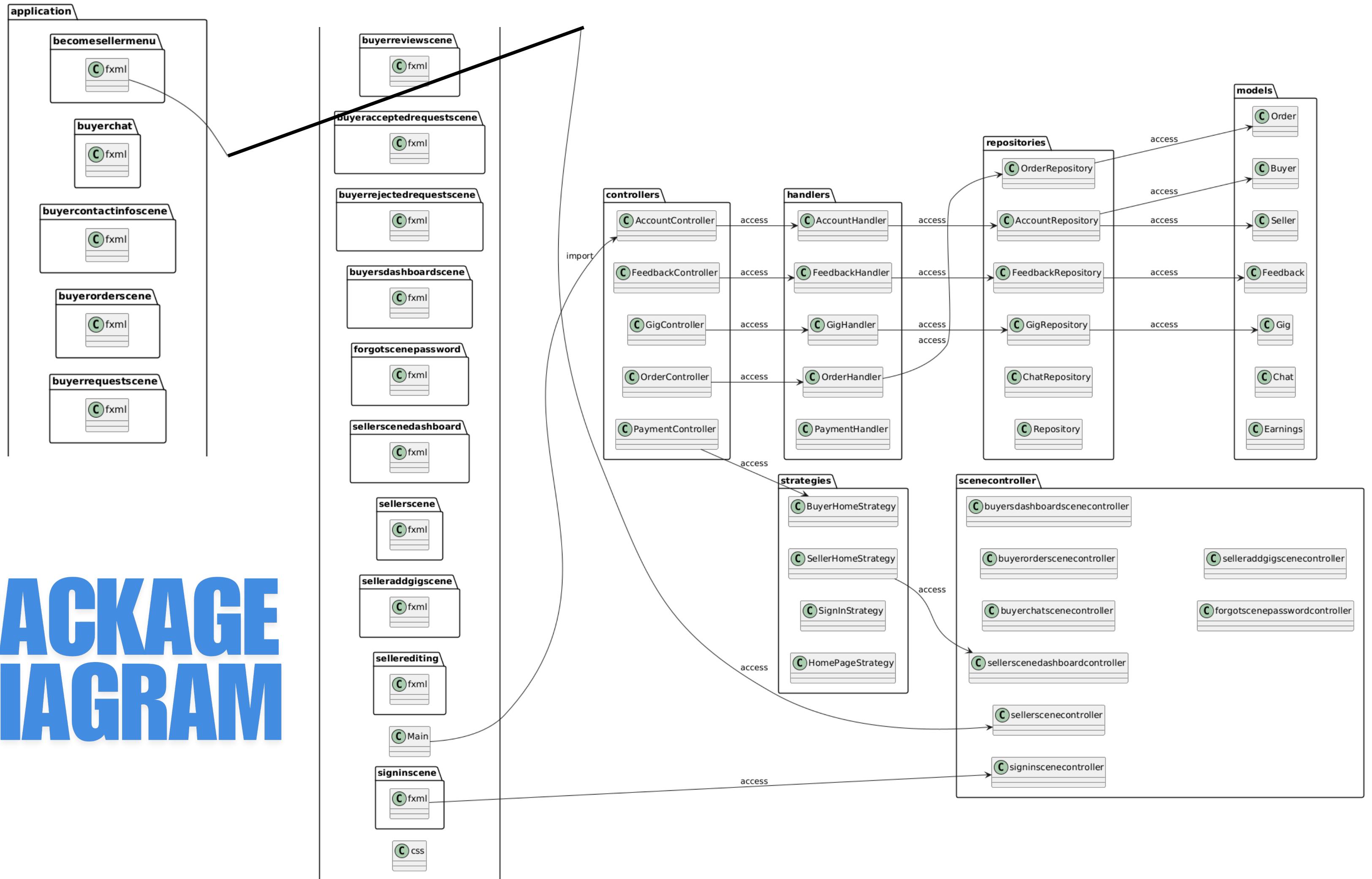
COMPONENT DIAGRAM



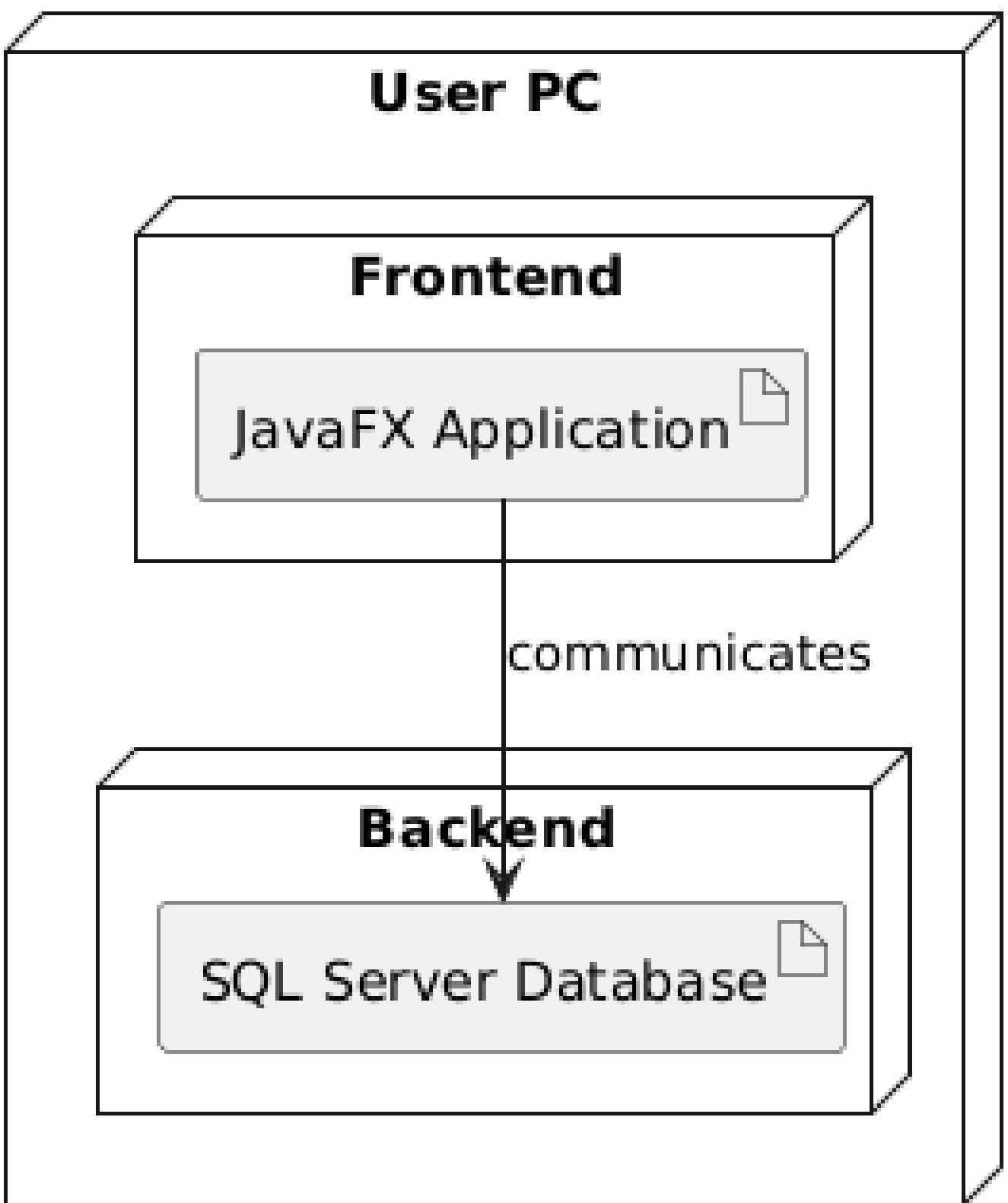
CLASS DIAGRAM



PACKAGE DIAGRAM



DEPLOYMENT DIAGRAM



THANK YOU