

Project Wiki: PlatePlan Reservation System

Overview	2
Contents	2
Team Members	2
Iteration 1 Stories	3
Story 1 - Pouya Sameni	3
Story 2 - Farah Madkour	4
Story 3 - Andrew Nong	4
Story 4 - Ricky Nguyen	4
Story 5 - Meem Morshed	5
Architecture	6
Service Architecture	6
Customer Views to Services	7
Business Views to Service	8
Technical Guide	9
Packages Overview	9
Core Components	10
Testing	11
Logic Tested	11
JUnit Coverage Chart	11

Overview

PlatePlan is a comprehensive reservation management system designed to facilitate both customers and businesses in handling table reservations, creating customer and employee accounts, and logging in both. The system comprises a GUI application built using Java, backed by a mock database for testing purposes. The objective of this application is to present an easy and efficient method in which customers and businesses can connect, creating a swift dining experience. This document outlines the project's architecture, including its packages, classes, and functionalities.

Contents

Team Members

Packages Overview

Core Components

GUI Panels

Database Implementation

Testing

Technical Guide

Future Iteration

Team Members

Name | Student ID

Farah Madkour | 219913219

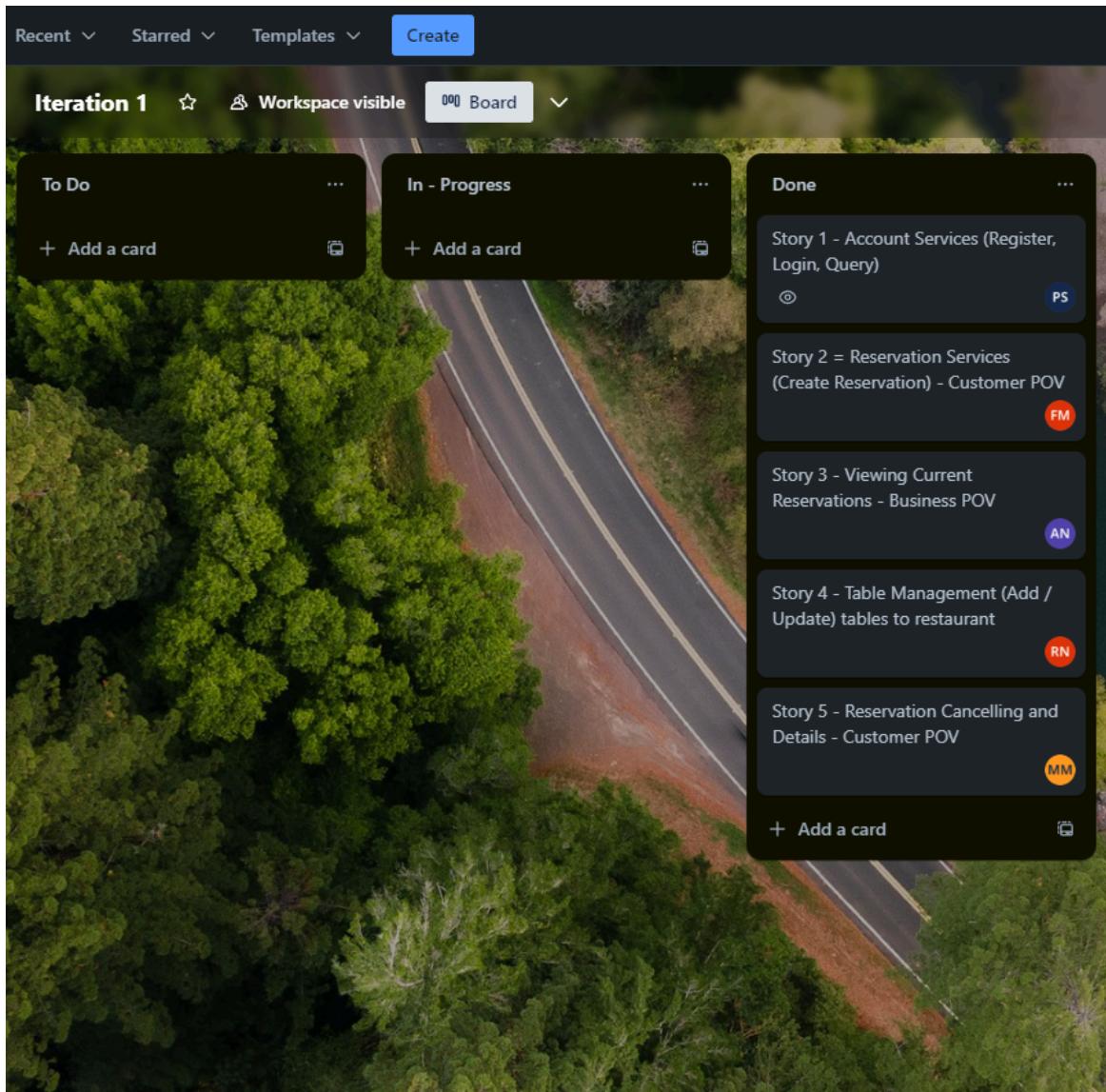
Meem Morshed | 219476142

Ricky Nguyen | 219461201

Andrew Nong | 219661537

Pouya Sameni | 216491623

Iteration 1 Stories



Story 1 - Pouya Sameni

Story one encompasses the foundational implementation of account services, enabling both customers and businesses to log in, and customers to register new accounts. This includes the development of service implementations, front-end components, and database integrations.

Planned Due Date: Jan 22nd, 2024

Actual Due Date: Jan 21st, 2024

Number of Days Planned: 4

Story 2 - Farah Madkour

At its core, this is a reservation system, and story two focuses on enabling customers to make reservations from their perspective. This involves the front-end functionality that allows our customers to create reservations, along with the corresponding reservation service components.

Planned Due Date: Jan 30th, 2024

Actual Due Date: Jan 28th, 2024

Number of Days Planned: 8

Story 3 - Andrew Nong

Story 3 is designed to allow businesses to access and review current reservations. This functionality will provide visibility into various reservation details, such as the identity of the individual who made the reservation, the number of guests included, the specific tables assigned, and the designated server. Moreover, it will also enable businesses to view any special instructions or notes associated with the reservation.

Planned Due Date: Feb 3rd, 2024

Actual Due Date: Feb 4th, 2024

Number of Days Planned: 5

Story 4 - Ricky Nguyen

Story 4 introduces the capability for businesses to effectively manage their table layout. This includes the ability to add or remove tables, assign servers to specific tables, and set a particular capacity for each table to accommodate the appropriate number of customers efficiently.

Planned Due Date: Feb 8th, 2024

Actual Due Date: Feb 9th, 2024

Number of Days Planned: 4

Note: Remainder of the days were spent on additional unit tests, modification in the UI for a better experience. Edge testing and further product enhancement. Additionally time was spent on creating the WIKI, Aggregating Log files and additional documentation

Story 5 - Meem Morshed

Story 5 revolves around providing customers with the functionality to view their current reservations and the option to cancel any existing reservations. This feature will be accessible directly from the customer menu, enhancing the customer experience by offering flexibility and control over their reservations.

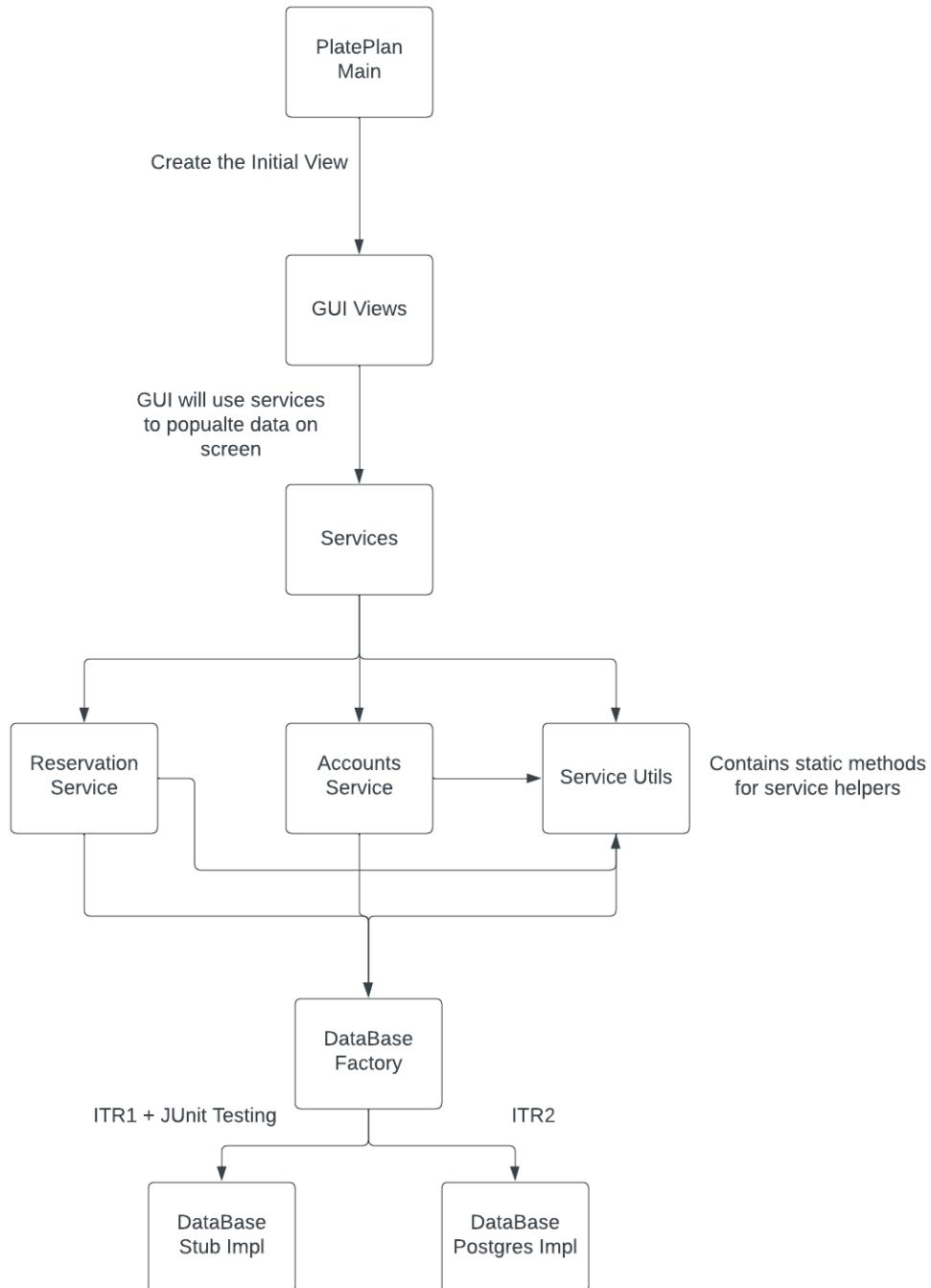
Planned Due Date: Jan 22nd, 2024

Actual Due Date: Jan 21st, 2024

Number of Days Planned: 4

Architecture

High Level Architecture

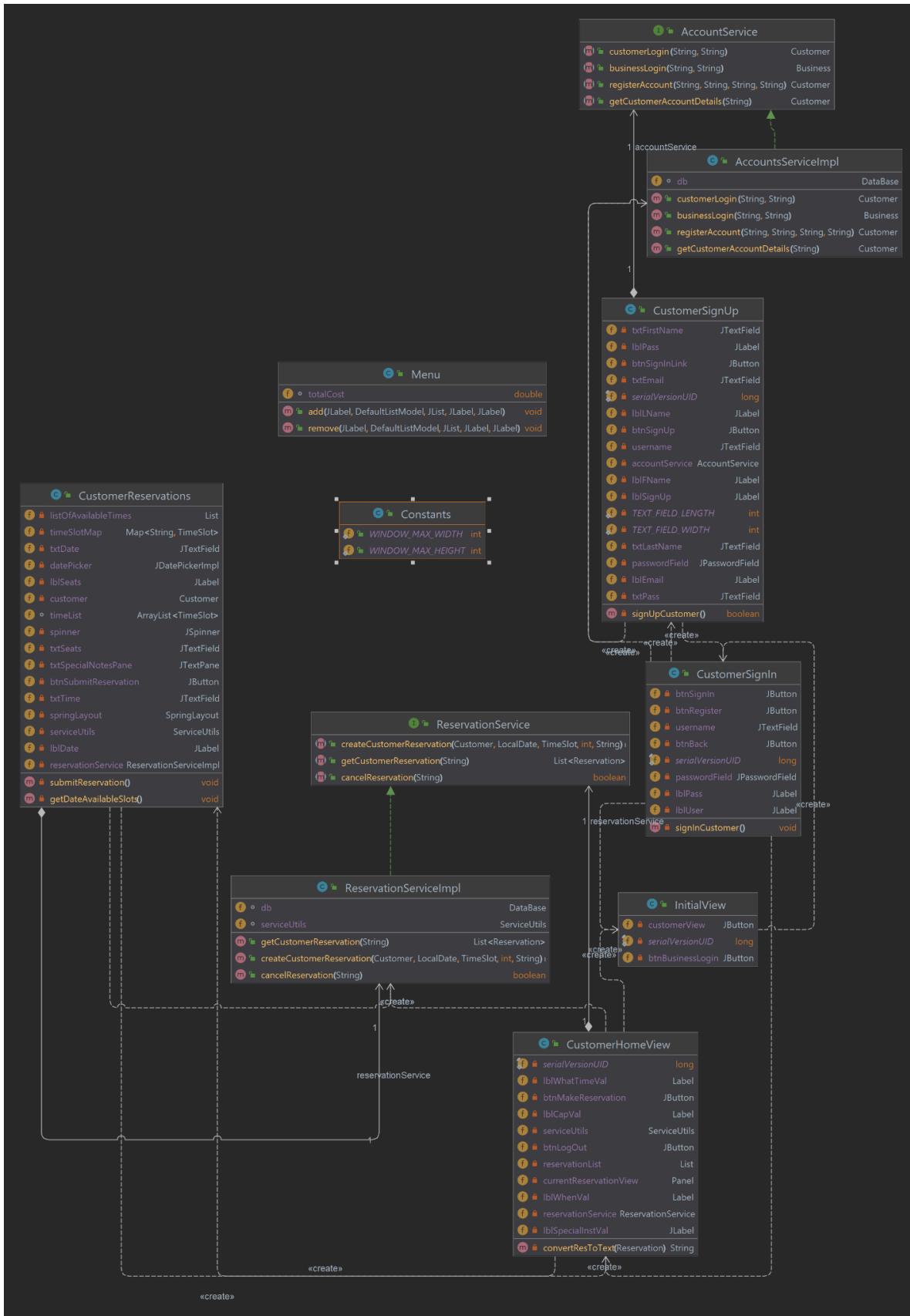


Detailed Service Architecture

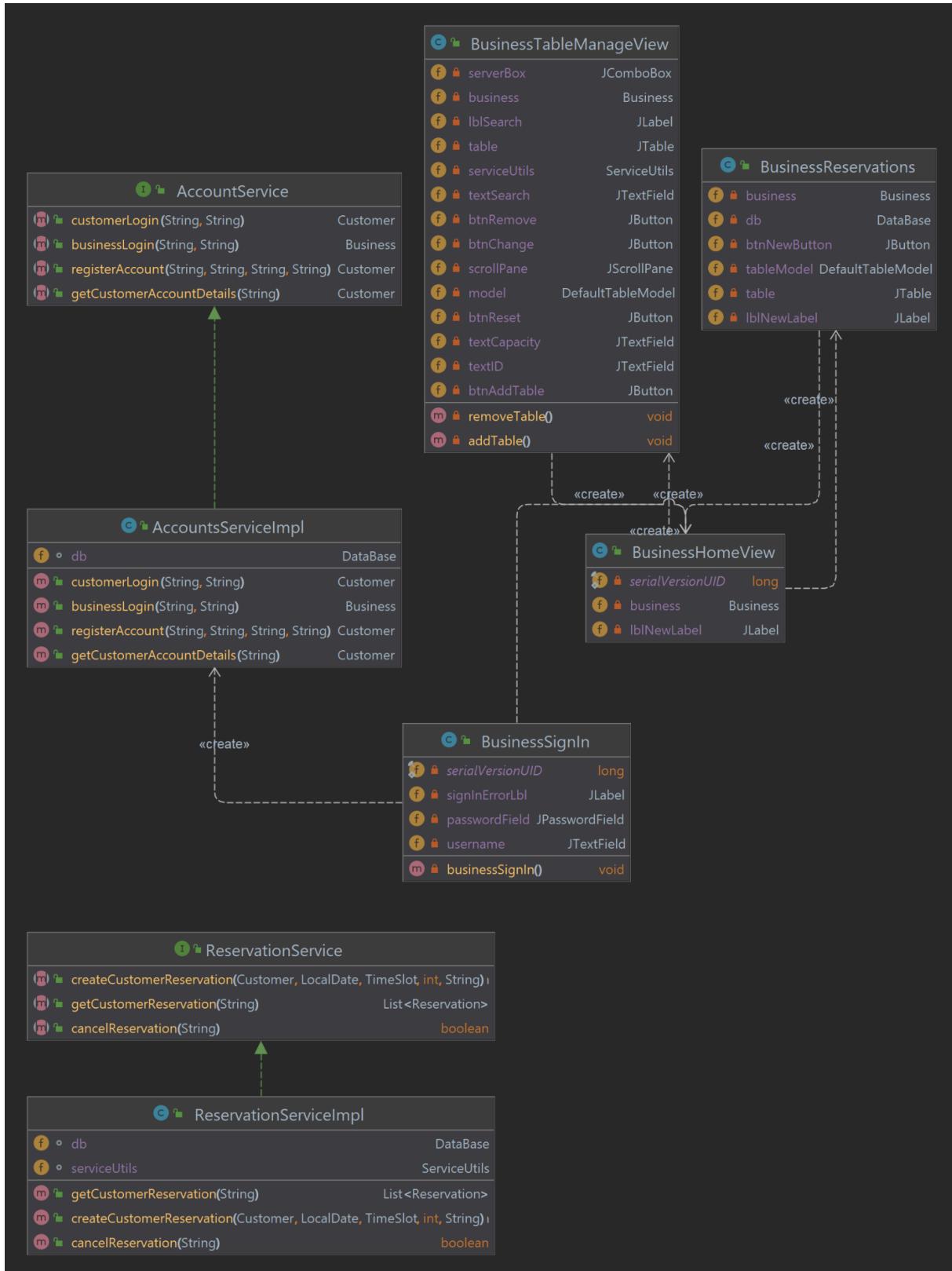
The below diagram shows the relationship between the different services, database and DTOs



Customer Views to Services



Business Views to Service



Technical Guide

To run the PlatePlan application:

1. Clone the repository.
 2. Open the project in a Java IDE (e.g., IntelliJ IDEA, Eclipse)- we built this project using eclipse.
 3. Run the main method in the PlatePlanMain class.
 4. From here you will be presented with the home screen where you can choose to log in as a customer or business owner.
 5. For demonstration purposes, use the email: max@email.com and the password: password, to login the customer side. For addition accounts please look into the file labeled **StubDataBaseRecords.java**
 6. To login to the business side use the **username**: alfredo and **password**: password
- For testing, execute the test cases in the tests package.

NOTE: You may also run the executable .jar file that is precompiled

Packages Overview

The project is structured into several packages, each with a distinct role:

Package Name	Purpose
businessPanels	Includes the JPanels that are used for the different views that relate to business features.
customerPanels	Includes the JPanels that are used for the different views that relate to customer features.
database	Includes the database factory, database interface and the stub database for iteration 1. The database factory will return the correct implementation of the database methods
dto	DTO contains all the data transfer objects that are used between methods and use to pass data around such as customer, reservations etc.
main	Includes the main method of our project
misc	Includes java files that are not categorized. This includes the stub database values, and service utils which is a series of methods used by the application to help other existing services
service_interfaces	Includes the interfaces of the different services for the project such as AccountService, ReservationService
services	Includes the implementation of the services from service interfaces.
tests	Includes the JUnit tests for our project

Core Components

Service Interfaces

ReservationService: Manages reservation-related operations.

AccountService: Handles account registration and authentication.

Utility Classes

ServiceUtils: Provides utility methods supporting various operations such as manipulating data transfer objects (Servers and Tables).

StubDataBaseRecords: In-memory mock database for testing.

DTOs

Customer, Business, Server, Table, Reservation, TimeSlot: Represent different entities like customers, reservations, etc. Constructors, Getters, and Setters.

GUI Panels

Customer Panels:

CustomerReservations: Interface for customers to make reservations.

CustomerSignIn: Panel for customer login.

CustomerSignUp: Registration interface for new customers.

CustomerHomeView: Home screen for logged-in customers.

InitialView: Home screen for all users, customers and business.

Business Panels:

BusinessHomeView: Main dashboard for business users.

BusinessReservations: Panel to view and manage reservations.

BusinessSignIn: Login interface for business accounts.

BusinessTableManageView: Interface for managing restaurant tables.

Database Implementation

DataBase: Interface defining database operations.

DataBaseStubImpl: Mock implementation for testing.

Testing

Unit tests are provided under the tests package to validate the functionality of both service layers and database operations.

Logic Tested

Java File
AccountServiceImpl.java
ReservationServiceImpl.java
DataBaseStubImpl.java
ServiceUtils.java

JUnit Coverage Chart

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
PlatePlan	34.5 %	2,167	4,120	6,287
src	34.5 %	2,167	4,120	6,287
> customerPanels	0.0 %	0	2,046	2,046
> businessPanels	0.0 %	0	1,543	1,543
> dto	49.5 %	406	415	821
> main	0.0 %	0	71	71
> database	88.6 %	263	34	297
> DataBaseStubImpl.java	89.9 %	250	28	278
> DataBaseFactory.java	81.2 %	13	3	16
> SQLTables.java	0.0 %	0	3	3
> tests	99.3 %	744	5	749
> ReservationServiceTest.java	98.9 %	434	5	439
> AccountServiceTest.java	100.0 %	117	0	117
> ServiceUtilsTest.java	100.0 %	193	0	193
> misc	99.4 %	519	3	522
> StubDataBaseRecords.java	98.9 %	282	3	285
> ServiceUtils.java	100.0 %	237	0	237
> services	98.7 %	235	3	238
> ReservationServiceImpl.java	97.4 %	114	3	117
> AccountsServiceImpl.java	100.0 %	121	0	121

