Project Wiki: PlatePlan Reservation System

Overview

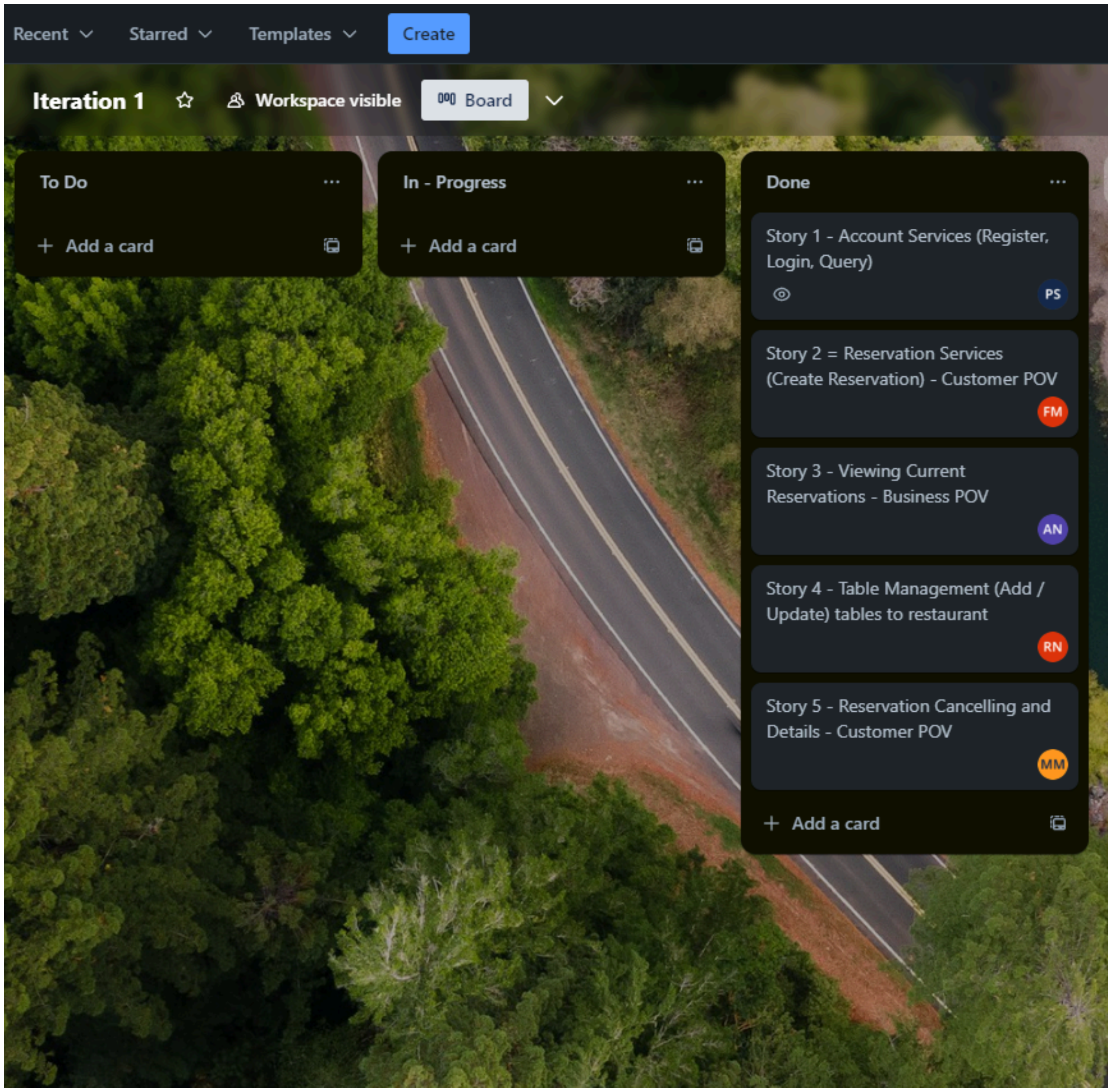
PlatePlan is a comprehensive reservation management system designed to facilitate both customers and businesses in handling table reservations, creating customer and employee accounts, and logging in both. The system comprises a GUI application built using Java, backed by a mock database for testing 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

Name	Student ID
Farah Madkour	219913219
Meem Morshed	219476142
Ricky Nguyen	219461201
Andrew Nong	219661537
Pouya Sameni	216491623

Iteration 1 Stories



Story 1 - Account Services (Register, Login, Query) PS

Story 2 = Reservation Services (Create Reservation) - Customer POV FM

Story 3 - Viewing Current Reservations - Business POV AN

Story 4 - Table Management (Add / Update) tables to restaurant RN

Story 5 - Reservation Cancelling and Details - Customer POV MM

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

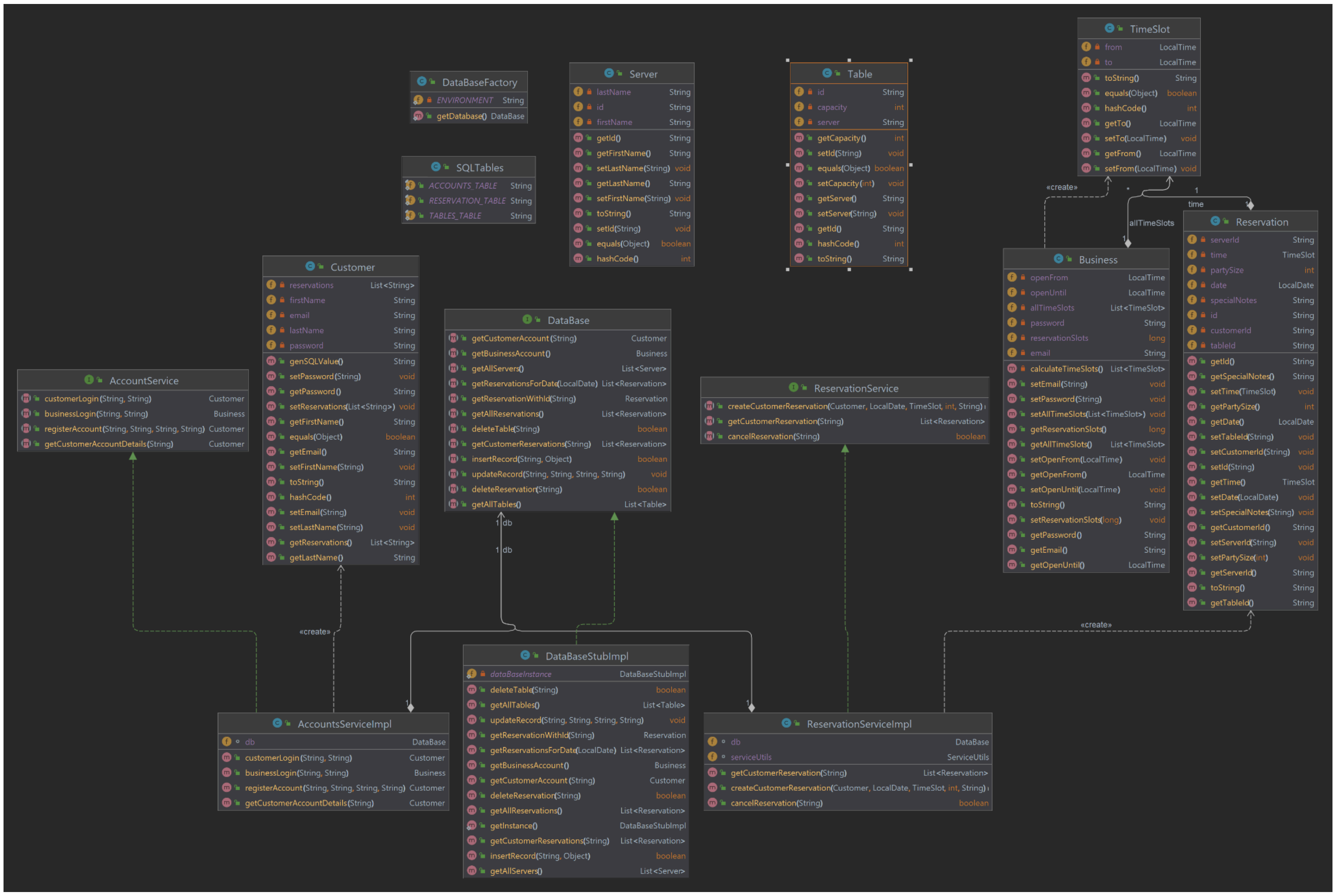
Story 5 revolves around providing customers with the functionality to view their current reservations and the option to cancel any existing reservations. This feature will be accessible directly from the customer menu, enhancing the customer experience by offering flexibility and control over their reservations.

Planned Due Date: Jan 22nd, 2024
Actual Due Date: Jan 21st, 2024
Number of Days Planned: 4

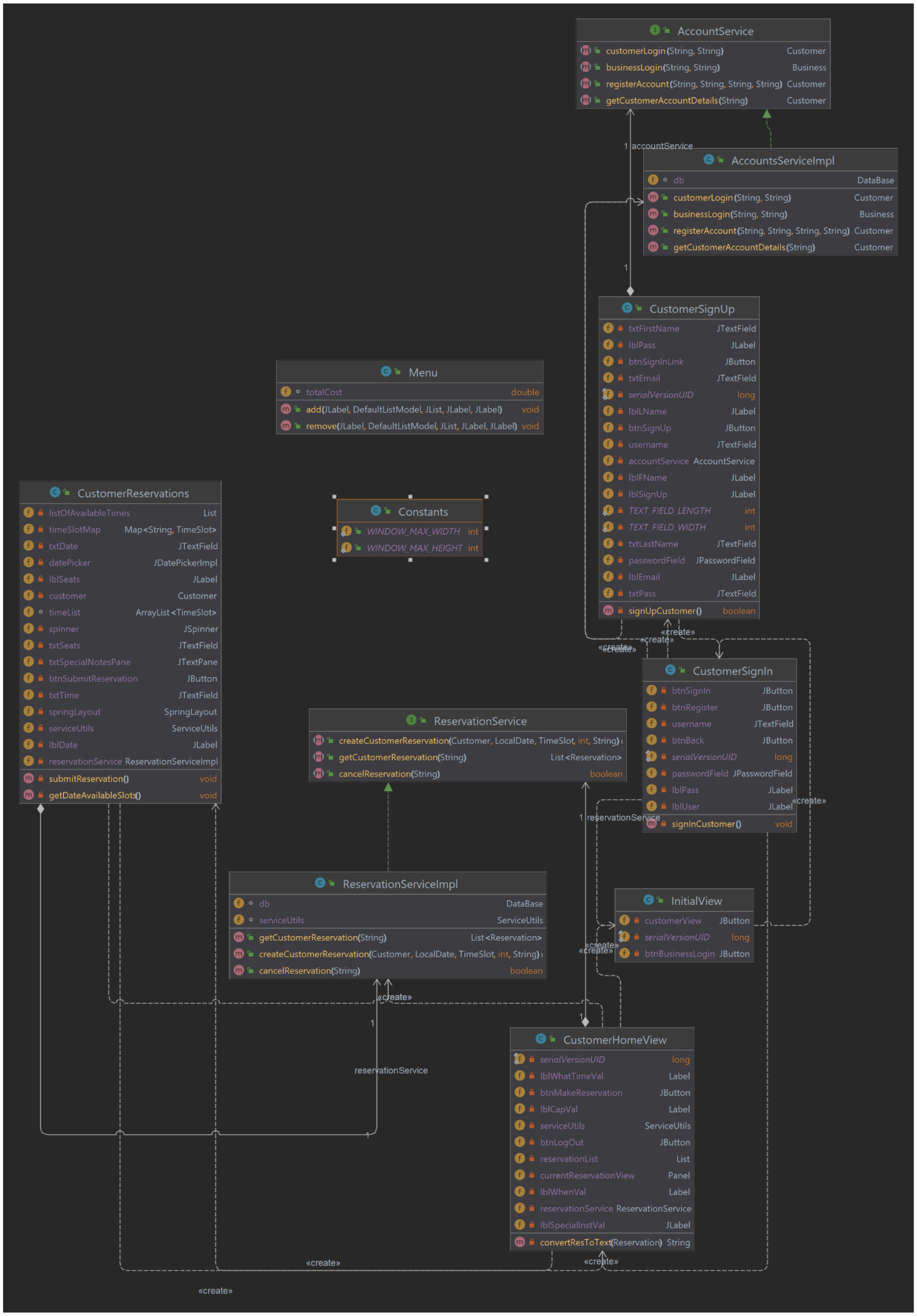
Architecture

Service Architecture

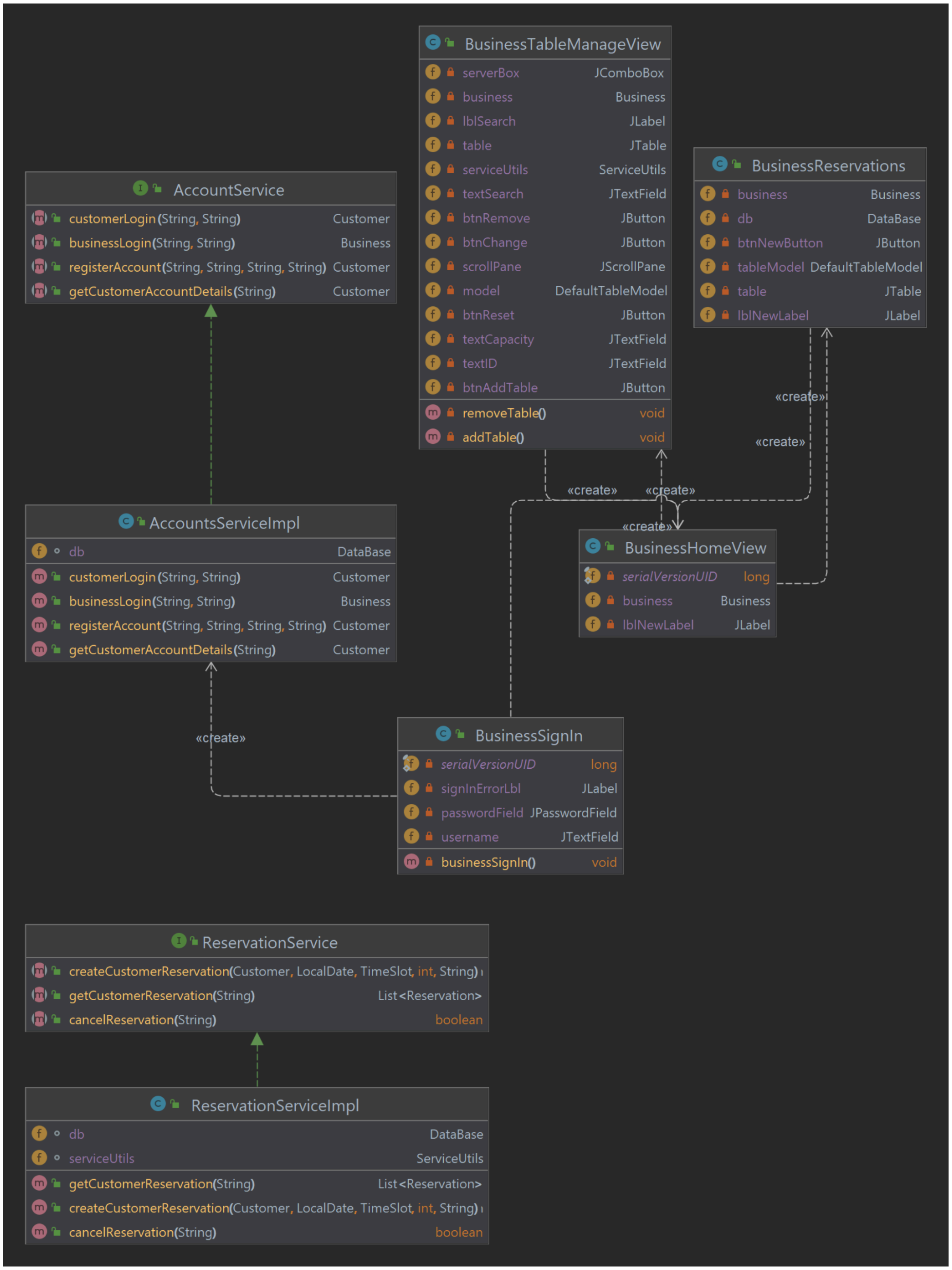
The below diagram shows the relationship between the different services, database and DTOs


DataBaseFactory
ENVIRONMENT String
getDatabase() DataBase
SQLTables
ACCOUNTS_TABLE String
RESERVATION_TABLE String
TABLES_TABLE String
Server
lastName String
id String
firstName String
getId() String
getFirstName() String
setLastName(String) void
getLastName() String
setFirstName(String) void
toString() String
setId(String) void
equals(Object) boolean
hashCode() int
Table
id String
capacity int
server String
getCapacity() int
setId(String) void
equals(Object) boolean
setCapacity(int) void
getServer() String
setServer(String) void
getId() String
hashCode() int
toString() String
TimeSlot
from LocalTime
to LocalTime
toString() String
equals(Object) boolean
hashCode() int
getTo() LocalTime
setTo(LocalTime) void
getFrom() LocalTime
setFrom(LocalTime) void
«create»
1
time
allTimeSlots
Business
openFrom LocalTime
openUntil LocalTime
allTimeSlots List<TimeSlot>
password String
reservationSlots long
email String
calculateTimeSlots() List<TimeSlot>
setEmail(String) void
setPassword(String) void
setAllTimeSlots(List<TimeSlot>) void
getReservationSlots() long
getAllTimeSlots() List<TimeSlot>
setOpenFrom(LocalTime) void
getOpenFrom() LocalTime
setOpenUntil(LocalTime) void
toString() String
setReservationSlots(long) void
getPassword() String
getEmail() String
getOpenUntil() LocalTime
Reservation
serverId String
time TimeSlot
partySize int
date LocalDate
specialNotes String
id String
customerId String
tableId String
getId() String
getSpecialNotes() String
setTime(TimeSlot) void
getPartySize() int
getDate() LocalDate
setTableId(String) void
setCustomerId(String) void
setId(String) void
getTime() TimeSlot
setDate(LocalDate) void
setSpecialNotes(String) void
getCustomerId() String
setServerId(String) void
setPartySize(int) void
getServerId() String
toString() String
getTableId() String
Customer
reservations List<String>
firstName String
email String
lastName String
password String
genSQLValue() String
setPassword(String) void
getPassword() String
setReservations(List<String>) void
getFirstName() String
equals(Object) boolean
getEmail() String
setFirstName(String) void
toString() String
hashCode() int
setEmail(String) void
setLastName(String) void
getReservations() List<String>
getLastName() String
AccountService
customerLogin(String, String) Customer
businessLogin(String, String) Business
registerAccount(String, String, String, String) Customer
getCustomerAccountDetails(String) Customer
DataBase
getCustomerAccount(String) Customer
getBusinessAccount() Business
getAllServers() List<Server>
getReservationsForDate(LocalDate) List<Reservation>
getReservationWithId(String) Reservation
getAllReservations() List<Reservation>
deleteTable(String) boolean
getCustomerReservations(String) List<Reservation>
insertRecord(String, Object) boolean
updateRecord(String, String, String, String) void
deleteReservation(String) boolean
getAllTables() List<Table>
ReservationService
createCustomerReservation(Customer, LocalDate, TimeSlot, int, String)
getCustomerReservation(String) List<Reservation>
cancelReservation(String) boolean
1 db
1 db
«create»
«create»
DataBaseStubImpl
dataBaseInstance DataBaseStubImpl
deleteTable(String) boolean
getAllTables() List<Table>
updateRecord(String, String, String, String) void
getReservationWithId(String) Reservation
getReservationsForDate(LocalDate) List<Reservation>
getBusinessAccount() Business
getCustomerAccount(String) Customer
deleteReservation(String) boolean
getAllReservations() List<Reservation>
getInstance() DataBaseStubImpl
getCustomerReservations(String) List<Reservation>
insertRecord(String, Object) boolean
getAllServers() List<Server>
AccountsServiceImpl
db DataBase
customerLogin(String, String) Customer
businessLogin(String, String) Business
registerAccount(String, String, String, String) Customer
getCustomerAccountDetails(String) Customer
ReservationServiceImpl
db DataBase
serviceUtils ServiceUtils
getCustomerReservation(String) List<Reservation>
createCustomerReservation(Customer, LocalDate, TimeSlot, int, String)
cancelReservation(String) boolean

Customer Views to Services


AccountService
customerLogin(String, String) Customer
businessLogin(String, String) Business
registerAccount(String, String, String, String) Customer
getCustomerAccountDetails(String) Customer
1 accountService
AccountsServiceImpl
db DataBase
customerLogin(String, String) Customer
businessLogin(String, String) Business
registerAccount(String, String, String, String) Customer
getCustomerAccountDetails(String) Customer
1
CustomerSignUp
txtFirstName JTextField
lblPass JLabel
btnSignInLink JButton
txtEmail JTextField
serialVersionUID long
lblLName JLabel
btnSignUp JButton
username JTextField
accountService AccountService
lblFName JLabel
lblSignUp JLabel
TEXT_FIELD_LENGTH int
TEXT_FIELD_WIDTH int
txtLastName JTextField
passwordField JPasswordField
lblEmail JLabel
txtPass JTextField
signUpCustomer() boolean
Menu
totalCost double
add(JLabel, DefaultListModel, JList, JLabel, JLabel) void
remove(JLabel, DefaultListModel, JList, JLabel, JLabel) void
Constants
WINDOW_MAX_WIDTH int
WINDOW_MAX_HEIGHT int
CustomerReservations
listOfAvailableTimes List
timeSlotMap Map<String, TimeSlot>
txtDate JTextField
datePicker JDatePickerImpl
lblSeats JLabel
customer Customer
timeList ArrayList<TimeSlot>
spinner JSpinner
txtSeats JTextField
txtSpecialNotesPane JTextPane
btnSubmitReservation JButton
txtTime JTextField
springLayout SpringLayout
serviceUtils ServiceUtils
lblDate JLabel
reservationService ReservationServiceImpl
submitReservation() void
getDateAvailableSlots() void
«create»
«create»
«create»
CustomerSignIn
btnSignIn JButton
btnRegister JButton
username JTextField
btnBack JButton
serialVersionUID long
passwordField JPasswordField
lblPass JLabel
lblUser JLabel
signInCustomer() void
«create»
ReservationService
createCustomerReservation(Customer, LocalDate, TimeSlot, int, String)
getCustomerReservation(String) List<Reservation>
cancelReservation(String) boolean
1 reservationService
ReservationServiceImpl
db DataBase
serviceUtils ServiceUtils
getCustomerReservation(String) List<Reservation>
createCustomerReservation(Customer, LocalDate, TimeSlot, int, String)
cancelReservation(String) boolean
InitialView
customerView JButton
serialVersionUID long
btnBusinessLogin JButton
«create»
«create»
«create»
1
1
reservationService
CustomerHomeView
serialVersionUID long
lblWhatTimeVal Label
btnMakeReservation JButton
lblCapVal Label
serviceUtils ServiceUtils
btnLogOut JButton
reservationList List
currentReservationView Panel
lblWhenVal Label
reservationService ReservationService
lblSpecialInstVal JLabel
convertResToText(Reservation) String
«create»
«create»
«create»

Business Views to Service


BusinessTableManageView
serverBox JComboBox
business Business
lblSearch JLabel
table JTable
serviceUtils ServiceUtils
textSearch JTextField
btnRemove JButton
btnChange JButton
scrollPane JScrollPane
model DefaultTableModel
btnReset JButton
textCapacity JTextField
textID JTextField
btnAddTable JButton
removeTable() void
addTable() void
BusinessReservations
business Business
db DataBase
btnNewButton JButton
tableModel DefaultTableModel
table JTable
lblNewLabel JLabel
AccountService
customerLogin(String, String) Customer
businessLogin(String, String) Business
registerAccount(String, String, String, String) Customer
getCustomerAccountDetails(String) Customer
«create»
«create»
«create»
«create»
«create»
AccountsServiceImpl
db DataBase
customerLogin(String, String) Customer
businessLogin(String, String) Business
registerAccount(String, String, String, String) Customer
getCustomerAccountDetails(String) Customer
BusinessHomeView
serialVersionUID long
business Business
lblNewLabel JLabel
«create»
BusinessSignIn
serialVersionUID long
signInErrorLbl JLabel
passwordField JPasswordField
username JTextField
businessSignIn() void
ReservationService
createCustomerReservation(Customer, LocalDate, TimeSlot, int, String)
getCustomerReservation(String) List<Reservation>
cancelReservation(String) boolean
ReservationServiceImpl
db DataBase
serviceUtils ServiceUtils
getCustomerReservation(String) List<Reservation>
createCustomerReservation(Customer, LocalDate, TimeSlot, int, String)
cancelReservation(String) boolean

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.

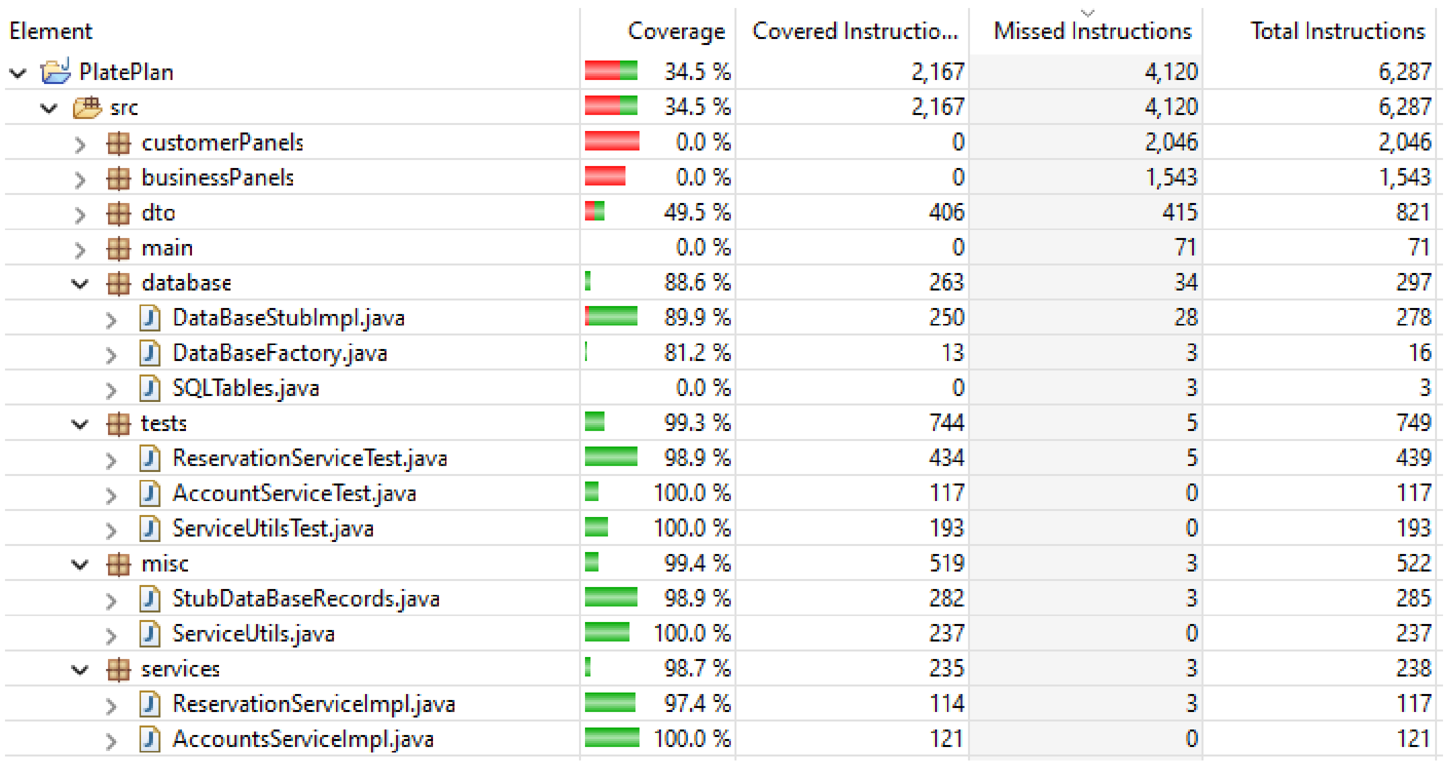
Testing

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

Coverage Chart



Element	Coverage	Covered Instructio...	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

Future Iteration

For the next iteration, new big user stories will be introduced. The application will receive an expansion of functionality in both the business and customer perspective. These new implementations will allow for management of staff work schedules, customers may be able to send feedback, and businesses can review and respond to feedback. The goal of this next iteration is to further the connection between customer and business which enable complete transparency.