CSCE 3513 Software Engineering
Team 02

Restaurant Organization and Floor Layout
(Restaurant Automation Project)

Initial Report
April 4, 2010

Adam Higgins
Tejeshwar Sangameswaran
Andrew Jones
Kellen Zantow

Individual Contributions

Requirements Specification

All members contributed equally

Software Design

Adam Higgins 50%, Tejeshwar Sangameswaran 50%

Coding

All members contributed equally

Debugging

All member contributed equally

Report Preparation

All members contributed equally

Demo and Presentation Preparation

All members contributed equally

Table of Contents

Customer Statement of Requirements

In general, this system is designed to minimize the amount effort required on the part restaurant employees. Instead of interacting with each other in person and through paper, they can simply log in and type a short description on a console and have it electronically entered into the network.

The actors in this system correspond to the different employee types (ie. host, waiter, cook, bus boy, and manager). Each employee has its own privileges in the system, appropriate for his or her tasks. When an employee logs into the system, he or she will see a start up screen with a menu unique to that employee type.
When a waiter logs into a terminal, he can see the layout of the floor and tables. The tables are colored in such a way that they indicate their status (green for available, yellow for occupied, red for dirty). He can then select a table to view its tab and is presented with a hierarchy. He can select a category of dishes and then select a particular food to add to the tab. He can also delete a dish from the tab. After he's done, he can go back to the main menu and choose another option, or he can just log off. This benefits him so that he doesn't have to walk to the back room to give the cooks a piece of paper and walk back to find another table, and changing the order is as easy as a click of a button instead of having to erase or start over.

When a manager logs in, he's greeted with an administrative control panel, wherein he can access, create, or modify employee profiles. He can use it to keep track of any information available, such as inventory info and sales analysis. He can also act as any other employee type in the system, so the manager type is like a subclass of all other types.

The cook screen, which should typically be open at all times, initially has a large gray area. When clicked, a new order attaches itself to the page. Each order has a unique number, with a list of entrees, appetizers, and desserts associated with that order. He can say when the order is complete and it will remove itself from the page and the others will re-organize themselves in sorted order.

One issue to deal with is whether to have the computer terminals stationary or wireless, hand-held devices. If stationary consoles are used exclusively, the waiters would still have to use paper to write things down before they can get to a terminal. On the other hand, hand-held devices would be expensive. Also, fixed devices aren't likely to be lost, unlike wireless devices. For now, assume the system uses

only stationary consoles.

Another problem is, there will most likely be more people on the staff than there are consoles. As a result, a user will have to log in, make the change, and log back out to free up the computer for someone else to use. To minimize the usage time of the consoles, when a user logs in, he's immediately updated on information relevant to his user type. For example, a waiter could log in and be told what orders have become ready to serve since he last logged on. Then he can log off directly after reading that information and go retrieve the food. Some users will remain logged on indefinitely (ie. the host and kitchen staff), since their consoles will be located in areas mostly exclusive to their types, and thus won't have to worry about logging out for another user type.

One of the functions of this system needs to be a floor layout manager. When designing the layout for a specific restaurant, we can arrange the walls and tables so that they match up with the physical layout of the building. The manager will be able to edit the floor layout at any time, as tables are added, removed, or relocated, or even if some construction is done on the walls. There's a default layout of walls and tables that is satisfactory for demonstration purposes.

The system is designed to automatically handle a large range of statistics processing and perform other tasks. It keeps track of employee hours, revenue for a given period, number of each dish sold, etc. and also processes information based on collected data, such as taxes, compensation owed, and so on. Automation is good because your typical human isn't very good at calculations.

In keeping with the requirement of efficiency, the system is designed to function on a minimal amount of clicks and keystrokes.

Glossary of Terms

Busboy

A busboy is a restaurant user who is responsible for cleaning dirty tables.

Cook

A cook is a restaurant user who is responsible for preparing orders.

Host

A host is a restaurant user who is responsible for placing customers at tables.

Manager

A manager is a restaurant user with complete system access. A manager can do all other users tasks, as well as changing master data in the system.

Order

An order is a list of items ordered by customers that is input by a waiter, tells cooks what to prepare, and how much the customer owes.

Payment

A payment is a monetary amount put against the value of an order. A payment may be cash or credit.

Table

A table is a location in a restaurant that is served by a waiter, cleaned by a busboy, and has an order tied to it when the table is occupied.

Waiter

A restaurant user who is responsible for taking customer orders, delivering food, and handling payments.

Functional Requirement Specification

Actors and Goals

Manager

The goals of the manager are to manage table layouts, system users, waiter schedules, and menu items and to analyze data for trends to improve restaurant profits.

Waiter

The goals of the waiter are to take the customers order and enter it into the system, to receive notification of orders being ready and to deliver the orders to the customers, to handle payments, and to mark tables as dirty when customers leave.

Host

The goal of the host is to place customers at available tables when the customers arrive or when a table becomes available.

Busboy

The goal of the busboy is to receive notification of dirty tables, to clean the tables, and the to mark the tables as available.

Cooks

The goal of the cook is to receive the orders from other users, to prepare the food, and to mark the orders as ready.

Use Cases

Casual Descriptions

Deliver Order

The system alerts the waiter when a cook has finished preparing an order. The waiter takes the food to the customer and marks the order as delivered.

Handle Payment

The waiter takes the payment method from the user and inputs the payment information into the system against an order.

Manage Users

A manager can add, change, and delete users in the system.

Manage Menu Items

A manager can add, change, and delete menu items in the system.

Manage Layout

A manager can add, change, and delete different types of tables and walls to the floor layout.

Fully-Dressed Descriptions

Seat Customer

Actor: Host, Manager

Preconditions:
- The user is registered in the system.
- The table to be occupied is open.

Scenario:
1) The user logs into the system.
2) The user navigates to the floor manager.
3) The user selects the table to be occupied and clicks "Set Table Occupied."
4) The user logs out.

Exceptions:
Step 3: If no table is open, the user must wait for one to become available.

Place Order

Actor: Waiter, Manager

Preconditions:
- The user is registered in the system.
- The table for the order is marked as occupied.

Scenario:
1) The user logs into the system.
2) The user navigates to the floor manager.
3) The user selects the table to have an order and clicks "Create Order."

4) The user selects a menu item to add to the order.
5) The user saves the order.
6) The user logs out.

Exceptions:
Step 3: If no tables are occupied, the user cannot create an order.

Extension:
4a. Multiple menu items can be added to an order.

Prepare Food

Actor: Cook, Manager

Preconditions:
- The user is registered in the system.
- The order to be prepared is in the system.

Scenario:
1) The user logs into the system.
2) The user navigates to the cook screen.
3) The user marks the prepared order as "Completed."
4) The user logs out of the system.

Exceptions:
Step 3: If there are no active orders in the system, the user cannot select an order as "Completed."

Extensions:
3a. The user may mark multiple orders as completed at one time.

Clean Table

Actor: Busboy, Manager

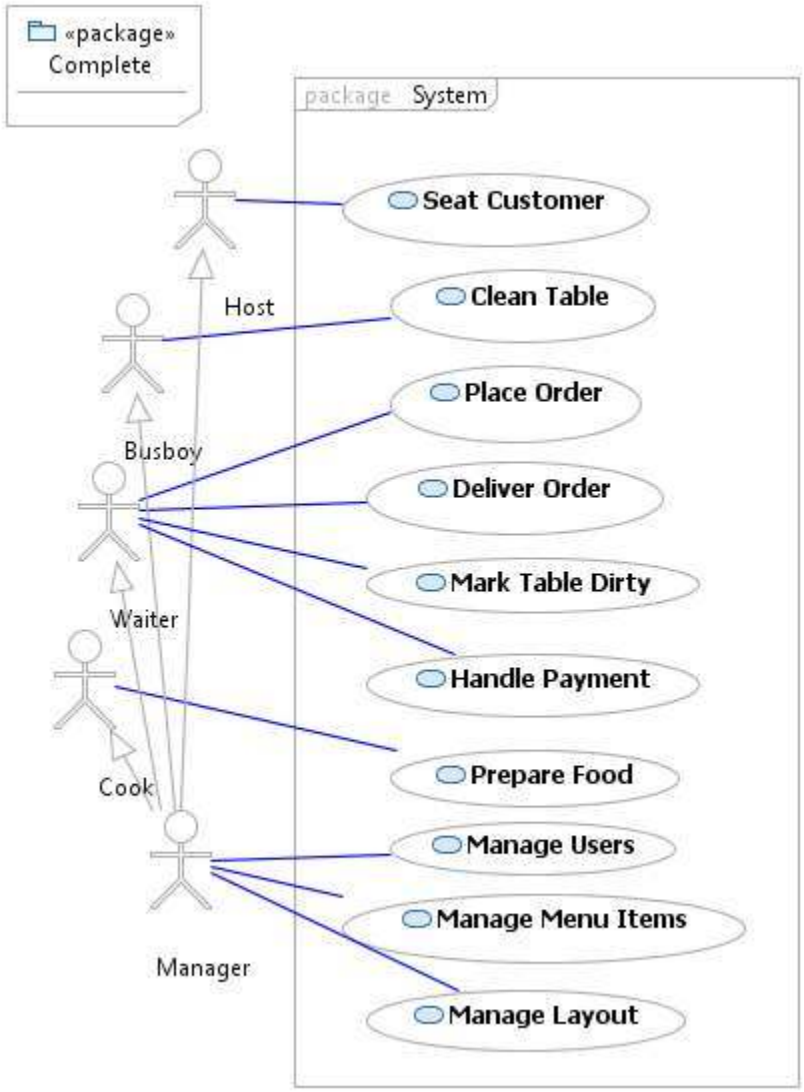
Preconditions:
- The user is registered in the system.
- The table to be cleaned is marked as dirty.

Scenario:
1) The user logs into the system.
2) The user navigates to the floor manager.
3) The user selects the table and clicks "Set Table Open."
4) The user logs out of the system.

Exceptions:

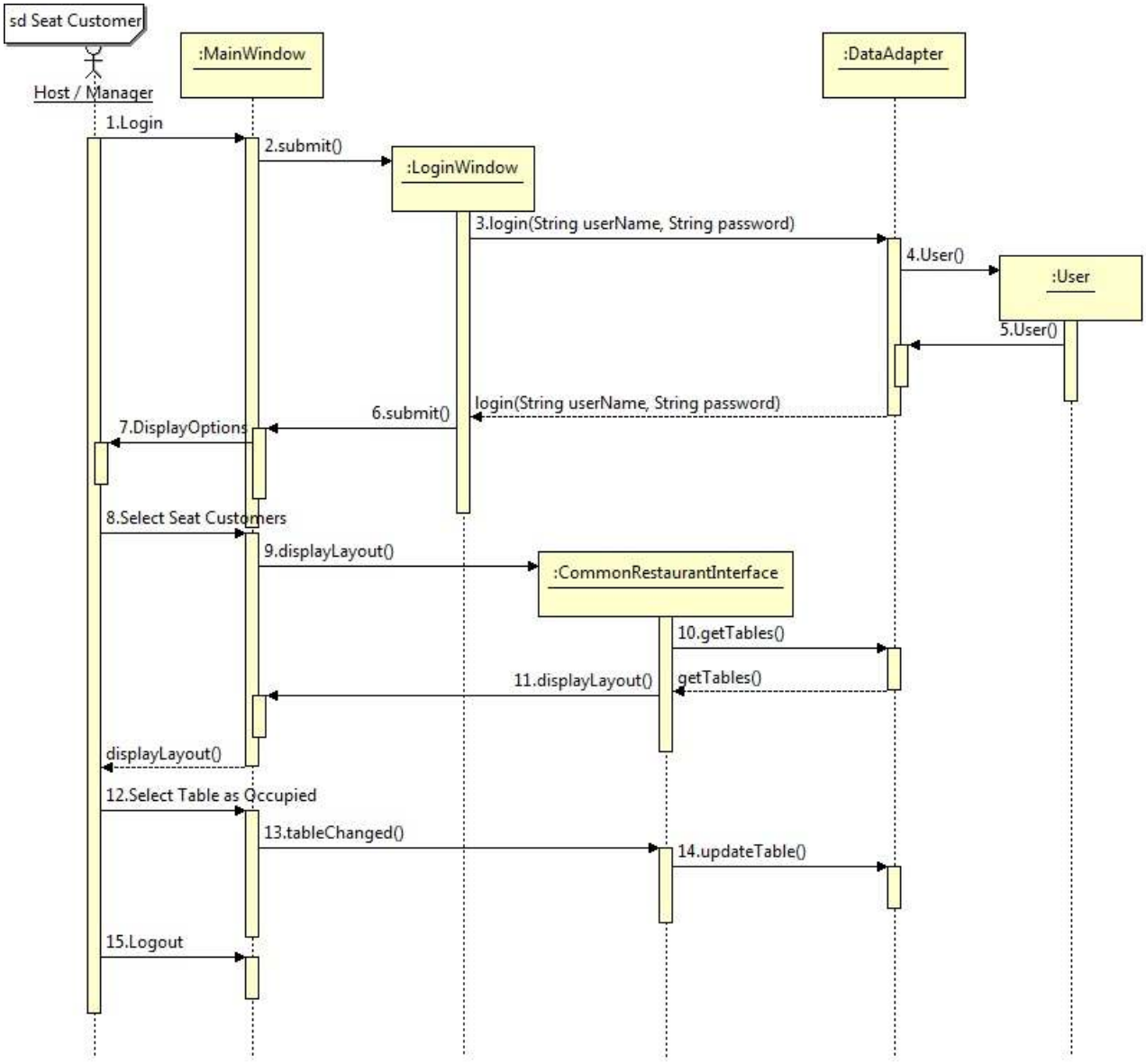
Step 3: If no tables are dirty, the user cannot mark any table as open.

Use Case Diagram

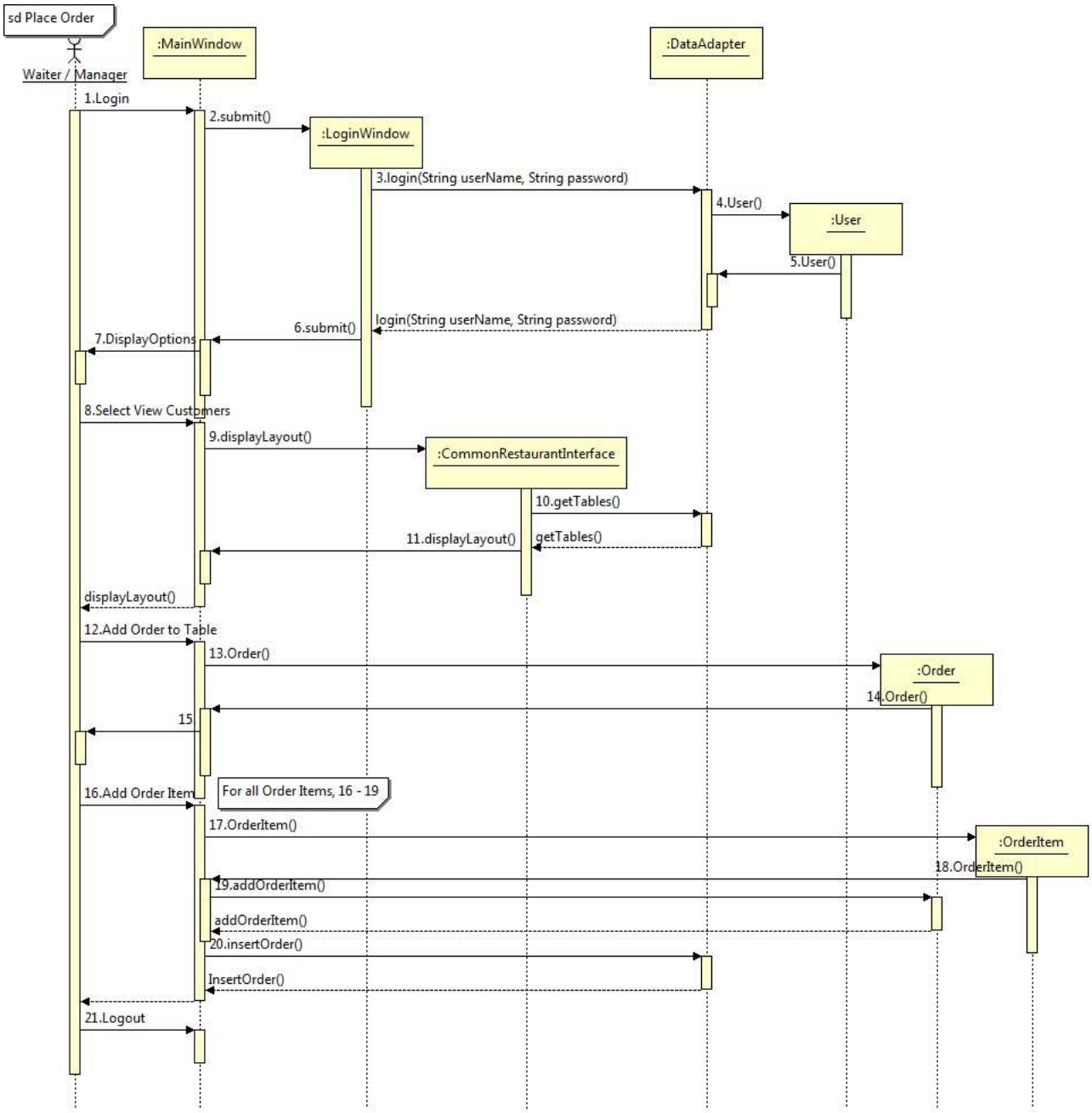


System Sequence Diagram

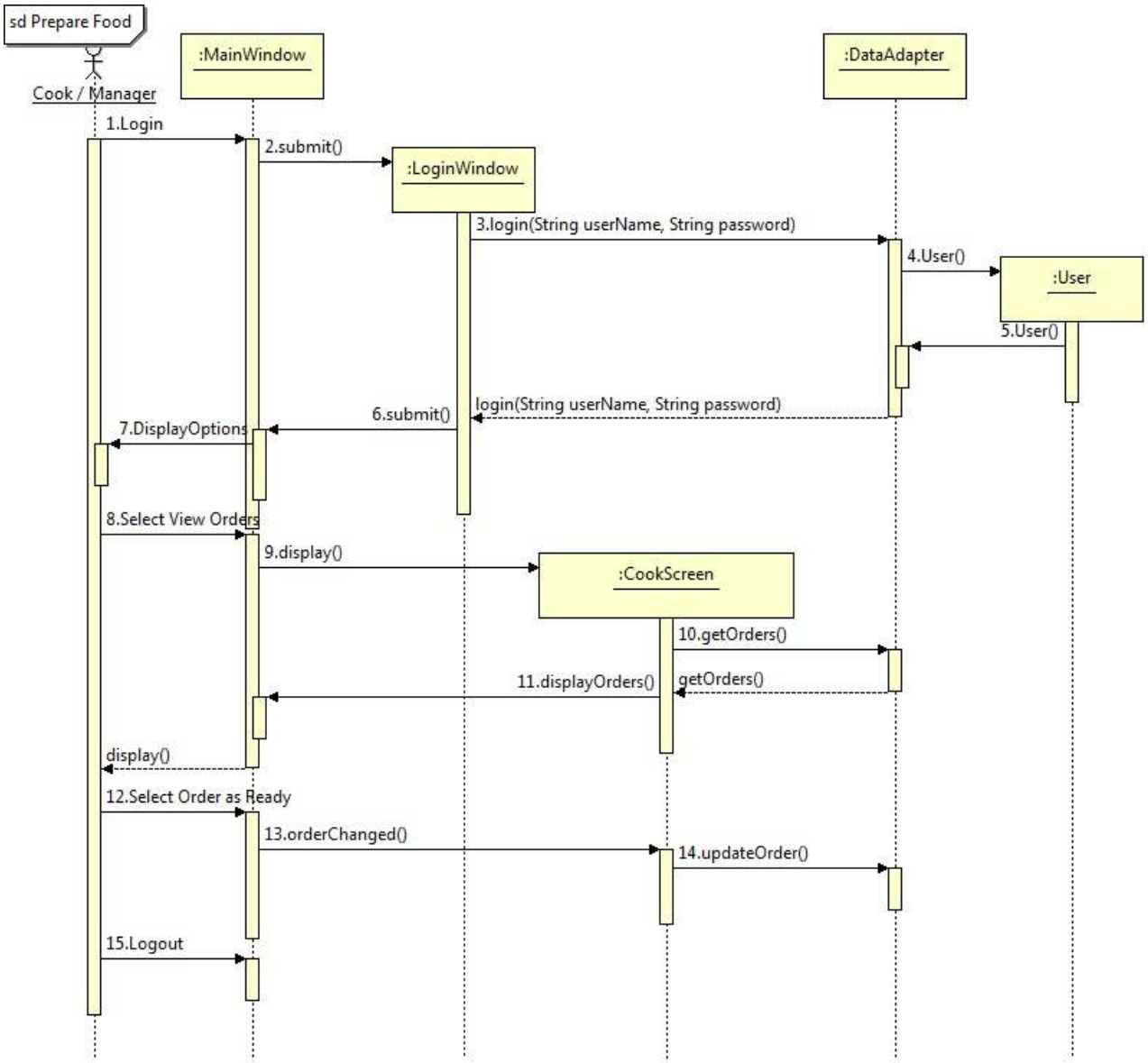
Seat Customer



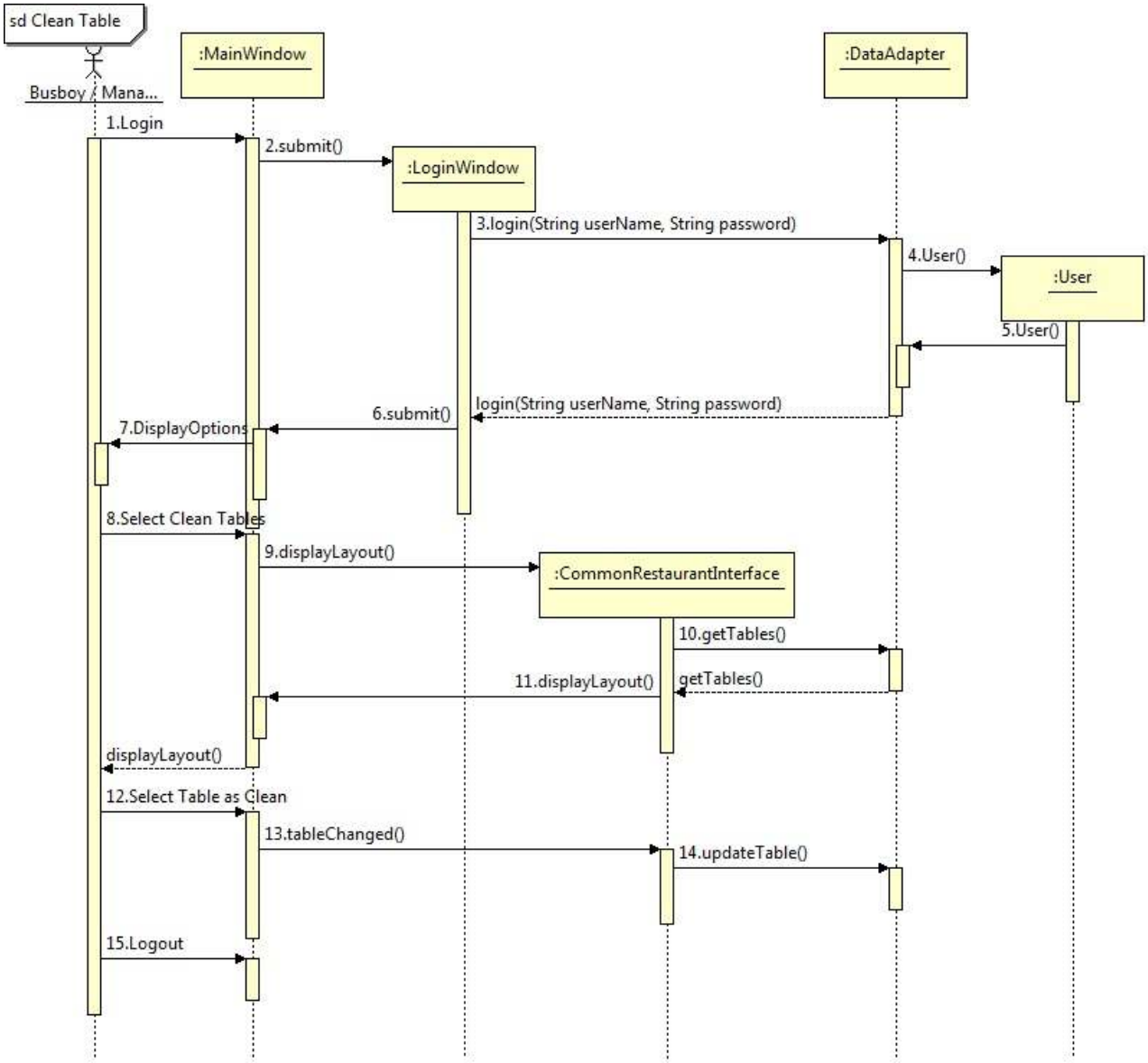
Place Order


sd Place Order
Waiter / Manager
:MainWindow
:DataAdapter
1.Login
2.submit()
:LoginWindow
3.login(String userName, String password)
4.User()
:User
5.User()
6.submit()
login(String userName, String password)
7.DisplayOptions
8.Select View Customers
9.displayLayout()
:CommonRestaurantInterface
10.getTables()
11.displayLayout()
getTables()
displayLayout()
12.Add Order to Table
13.Order()
:Order
14.Order()
15
16.Add Order Item
For all Order Items, 16 - 19
17.OrderItem()
:OrderItem
18.OrderItem()
19.addOrderItem()
addOrderItem()
20.insertOrder()
InsertOrder()
21.Logout

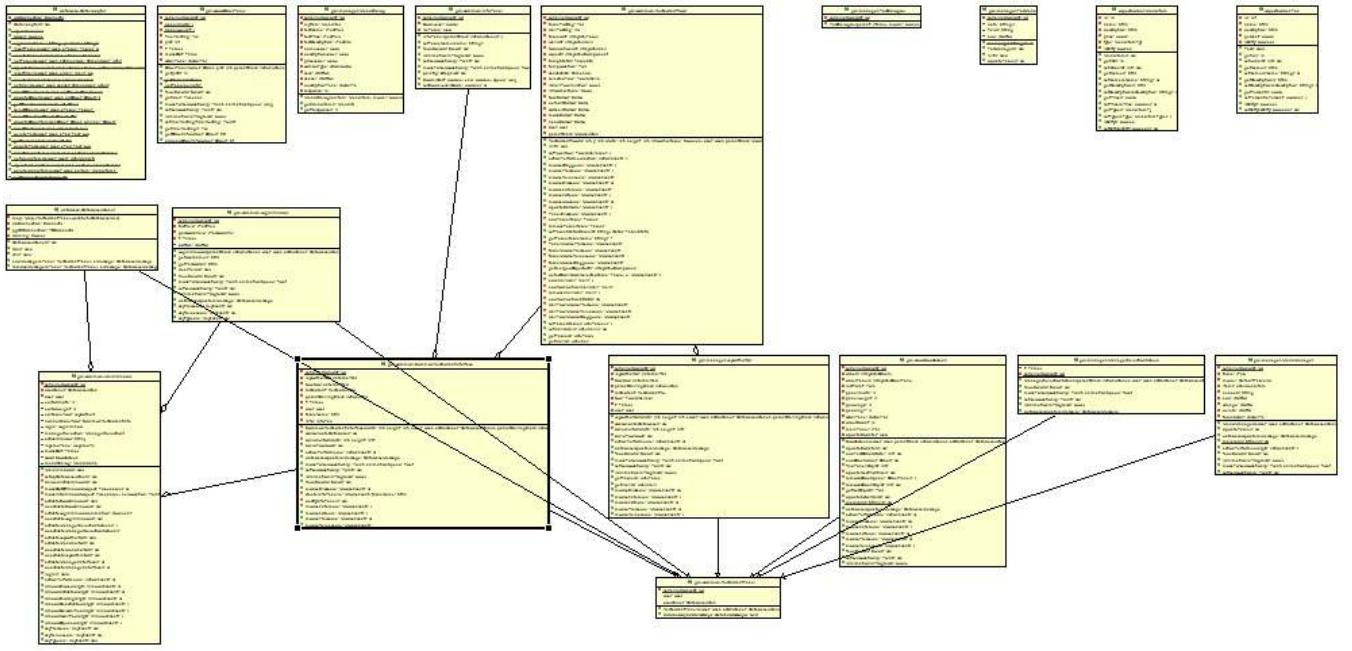
Prepare Food

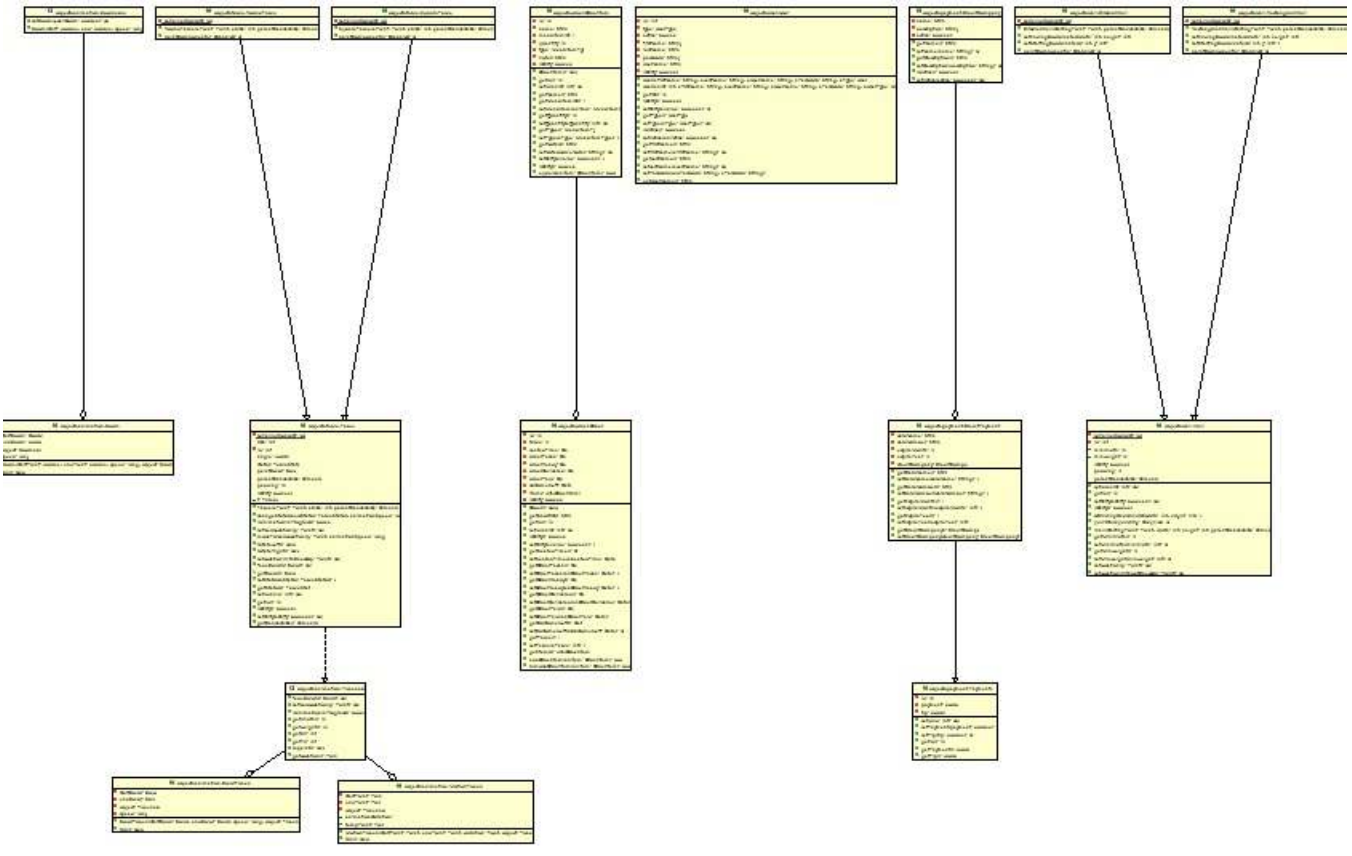


Clean Table


sd Clean Table
Busboy / Mana...
:MainWindow
:LoginWindow
:DataAdapter
:User
1.Login
2.submit()
3.login(String userName, String password)
4.User()
5.User()
login(String userName, String password)
6.submit()
7.DisplayOptions
8.Select Clean Tables
9.displayLayout()
:CommonRestaurantInterface
10.getTables()
getTables()
11.displayLayout()
displayLayout()
12.Select Table as Clean
13.tableChanged()
14.updateTable()
15.Logout

Class Diagram and Interface Specification



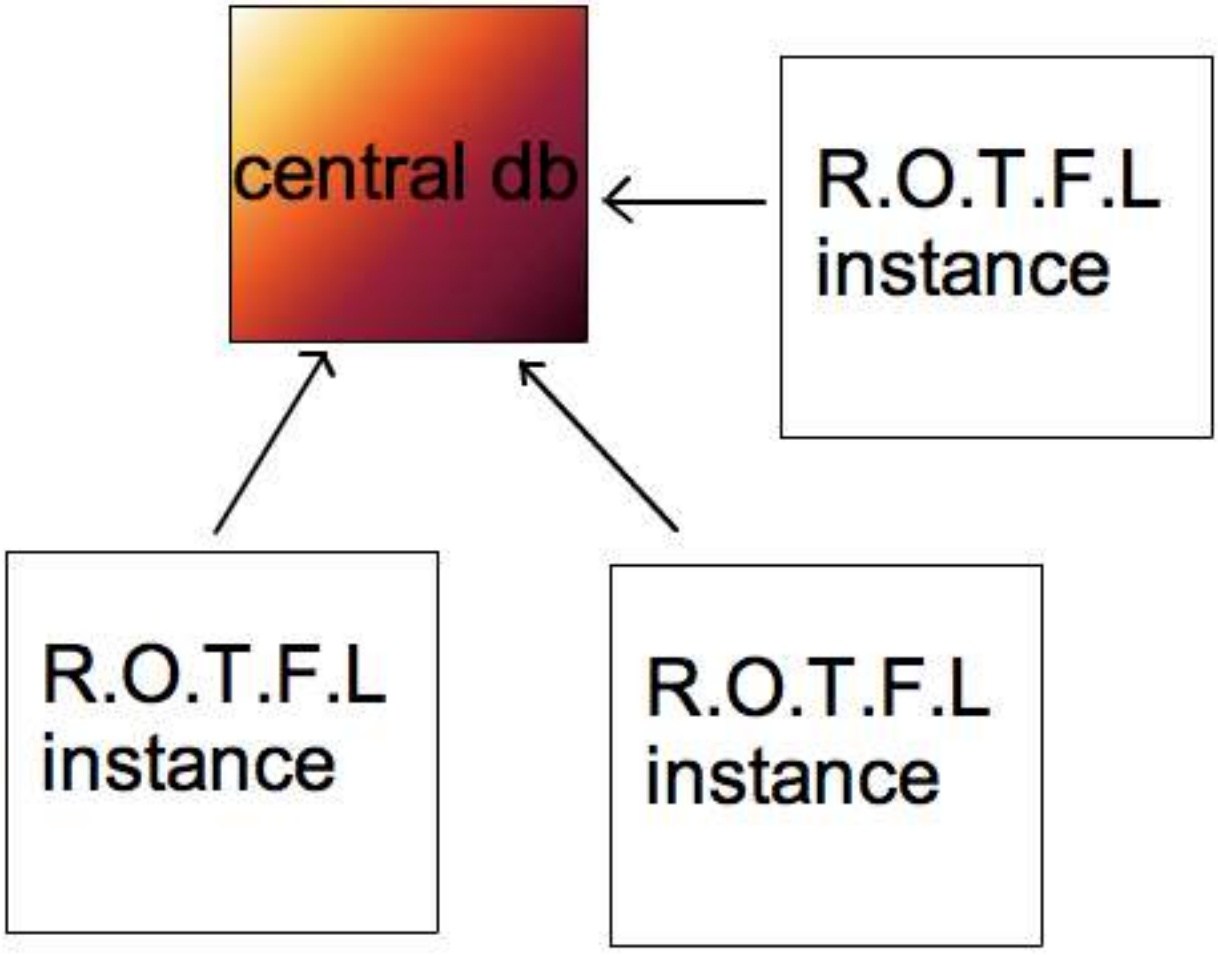


Note: Class diagram is managed with an Eclipse plug-in, AmaterasUML, and can be viewed in high resolution from the repository using this plug-in.

System Architecture and System Design

Architecture Styles

The final completed program executable is completely stand-alone. The same program can be deployed on several different machines. They are all kept in sync with each other through the central database. Each executable can logon as a single username and perform actions which will be cascaded onto all the other nodes.



Persistent Data Storage

Restaurant Organization and Layout Manager uses Postgresql, a relational database to persist data. The database persists users,

tables, orders, order items, menu items, payments, adjustment masters, adjustments, taxes, scheduling, and credit companies, as well as the following relationships: orders to tables, order items to orders, menu items to order items, payments to orders, adjustments to adjustment masters, adjustments to orders, taxes to menu items, and credit companies to credit payments. Java JDBC is the connection protocol of choice because of the ease of use. It requires only a database management system installed on the server instead of an additional server program**.**

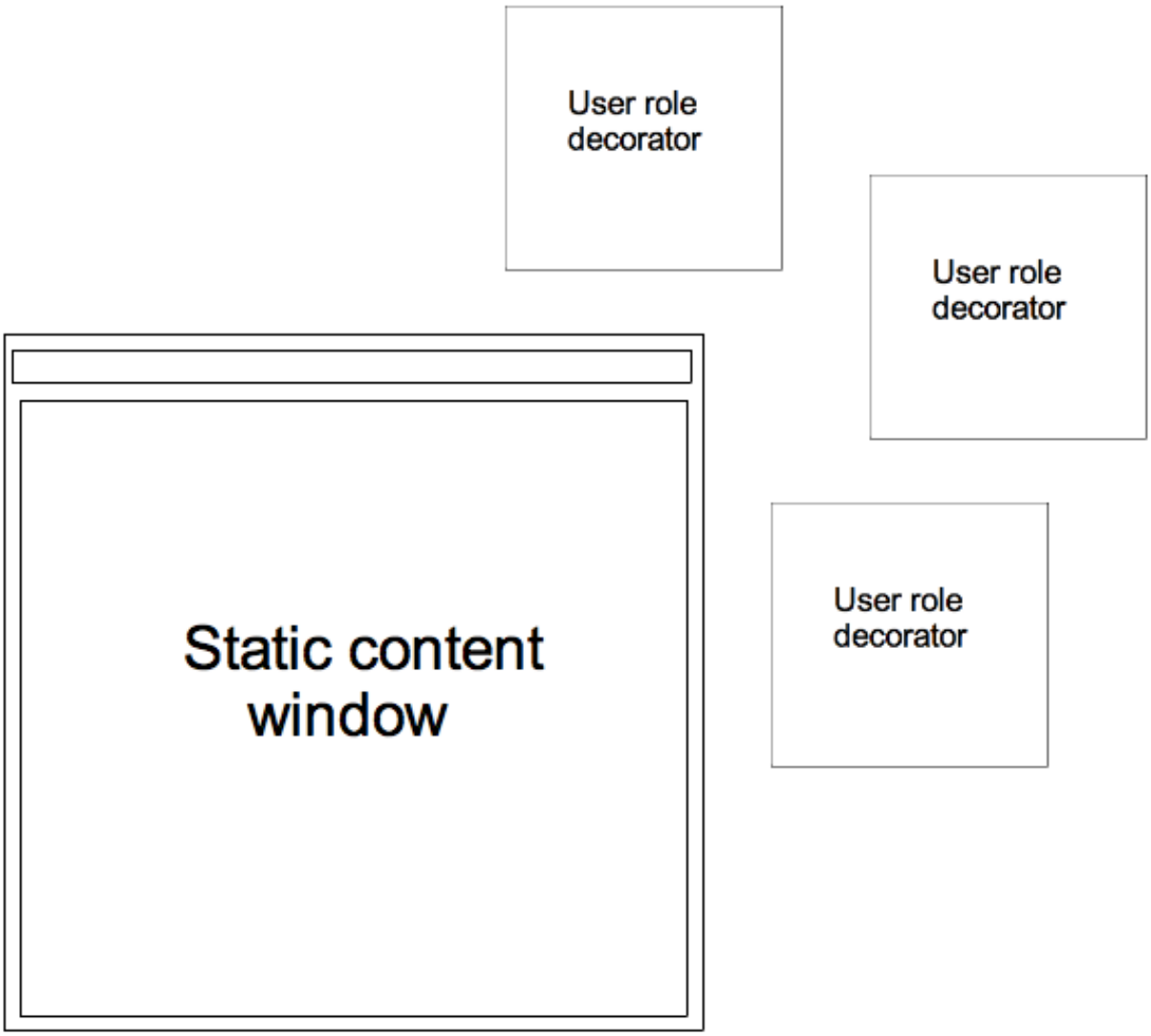
Algorithms and Data Structures

No complex algorithms or data structures were used in the making of this program.

User Interface Design and Implementation

Preliminary Design

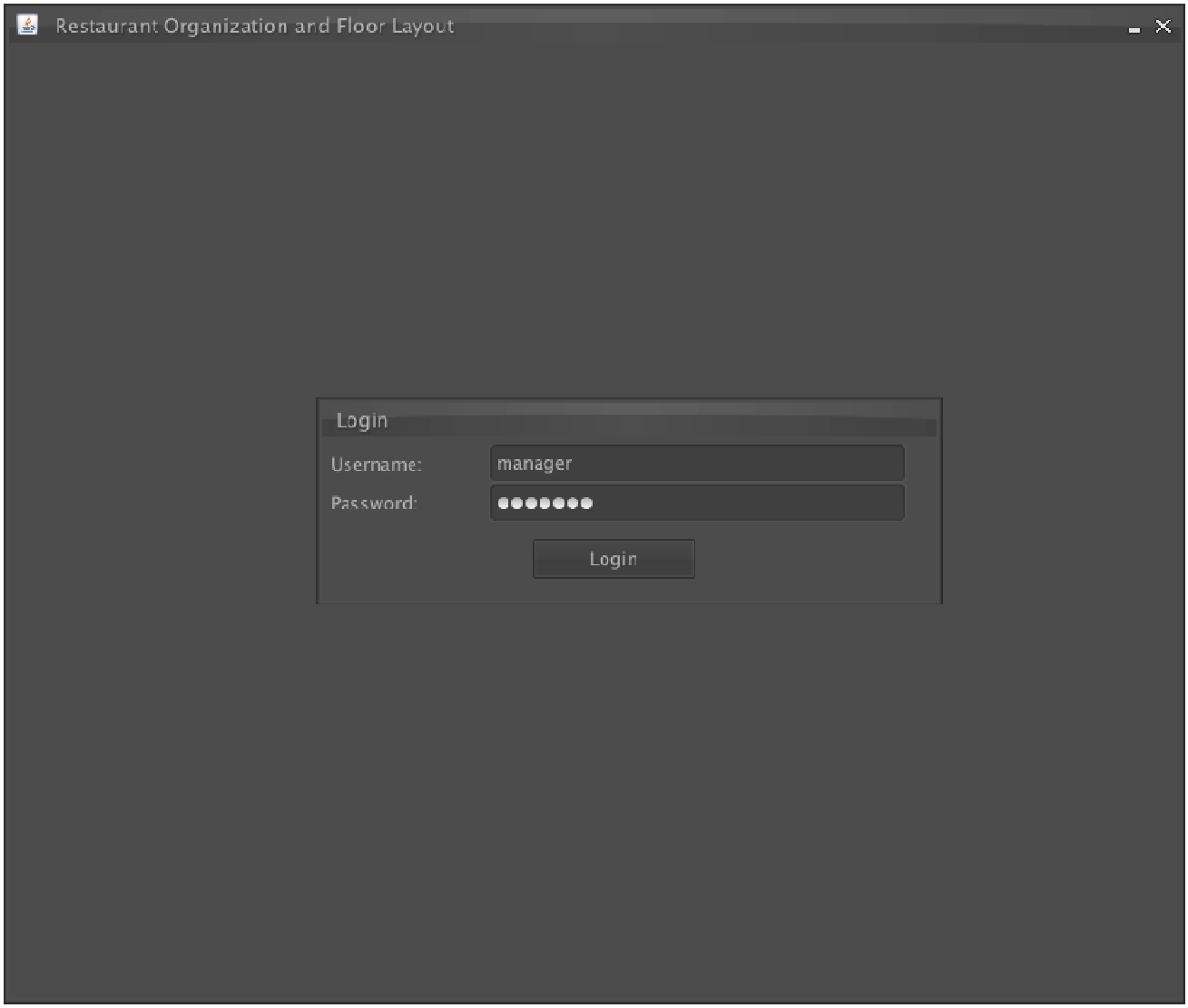
This project has several user roles all of whom have different functions. So, to accomodate this, sperate roles will have to have completely customized views. The most effecient way to implement this would be to have a static panel and a decorater to be passed into it with dynamic content. This content would reflect the roles of the user.



This user design allows, minimal code rewrite with maximum flexibility. Each individual decorator can be designed and tested independently.

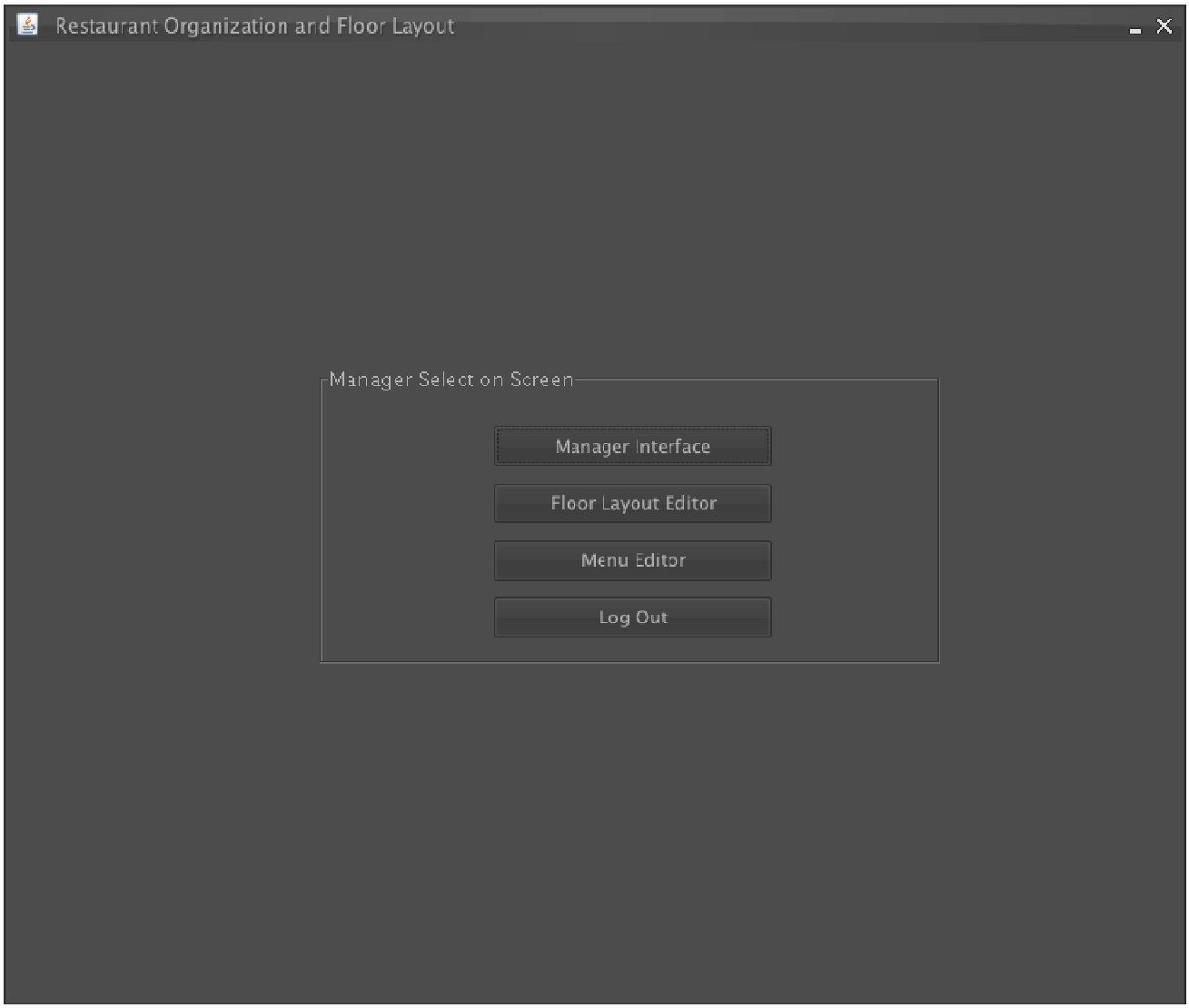
Each of these modules inherit from a common abstract class. They follow a strict set of rules so that they can be easily plugged into a parent system.

An example implementation of this system would look like this



As you can see, the login window is an internal frame inside the main frame. That internal frame can be switched anytime to another window. For example, after logging in as manager, the user will now

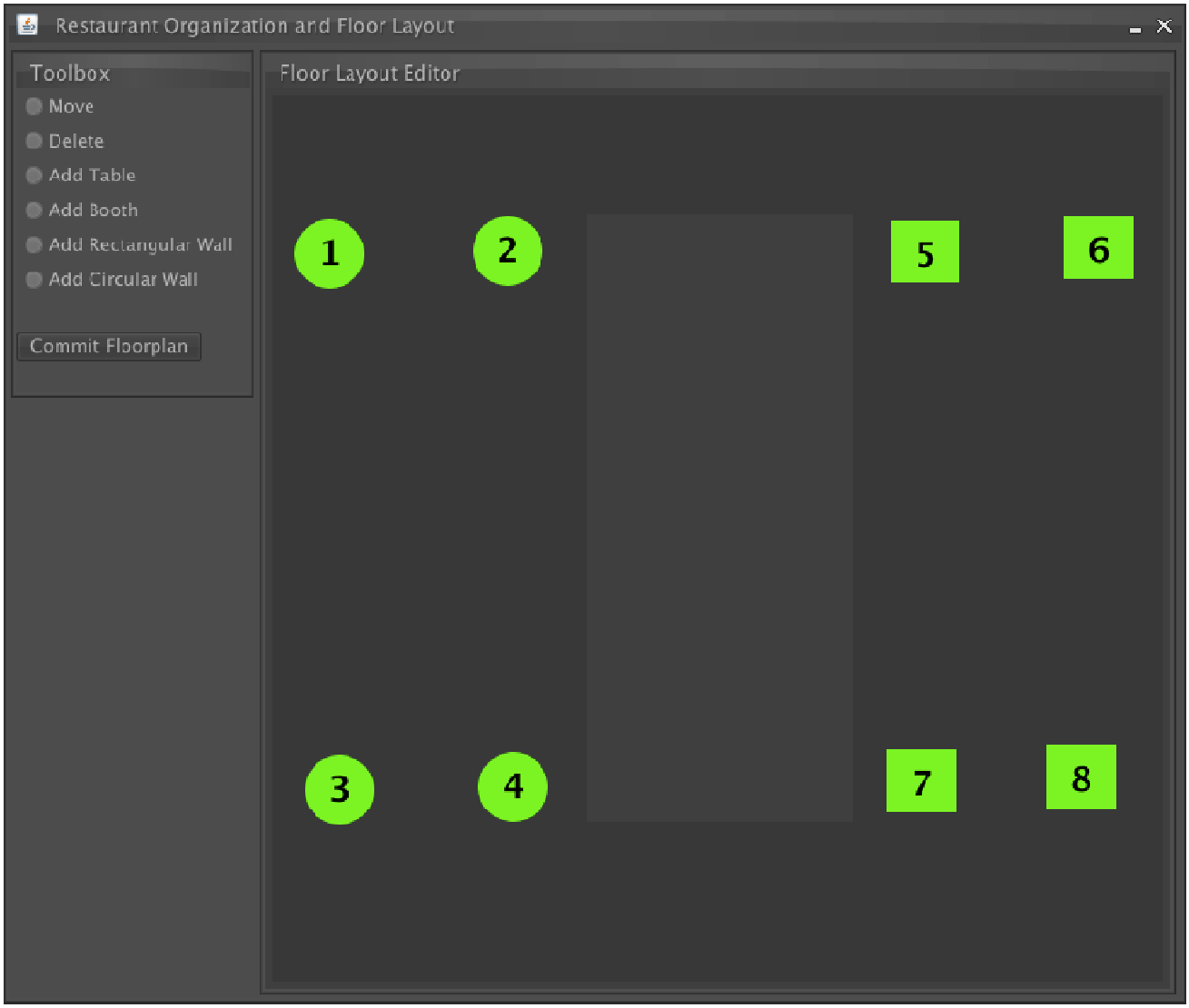
see this screen:



Each internal frame is aware of the role of the user and has direct connection the the database. Every change in the database is automatically cascaded down to all clients. This way, all the clients are always up-to-date.

User Effort Estimation

A lot of effort was put into the project to avoid program crashed due to malformated input and redundant data.
The GUI is "fool-proof", We will now demonstrate a sample of how we implemented this through the floor layout editor (pictured below)



The user has the power to add and remove walls and tables. But to make sure that the user does not give in an invalid input, the user's mouse movements are locked within the restaurant floor layout. This prevents them from giving in co-ordinates which are out of bounds.

We have also implemented an advanced heuristics based collision detection system. This system automatically detects collision and prevents users placing tables on top of each other and on walls. This implementation is highly optimized to be spatially aware; which means it detects collisions closer to it first than the ones which are far away.

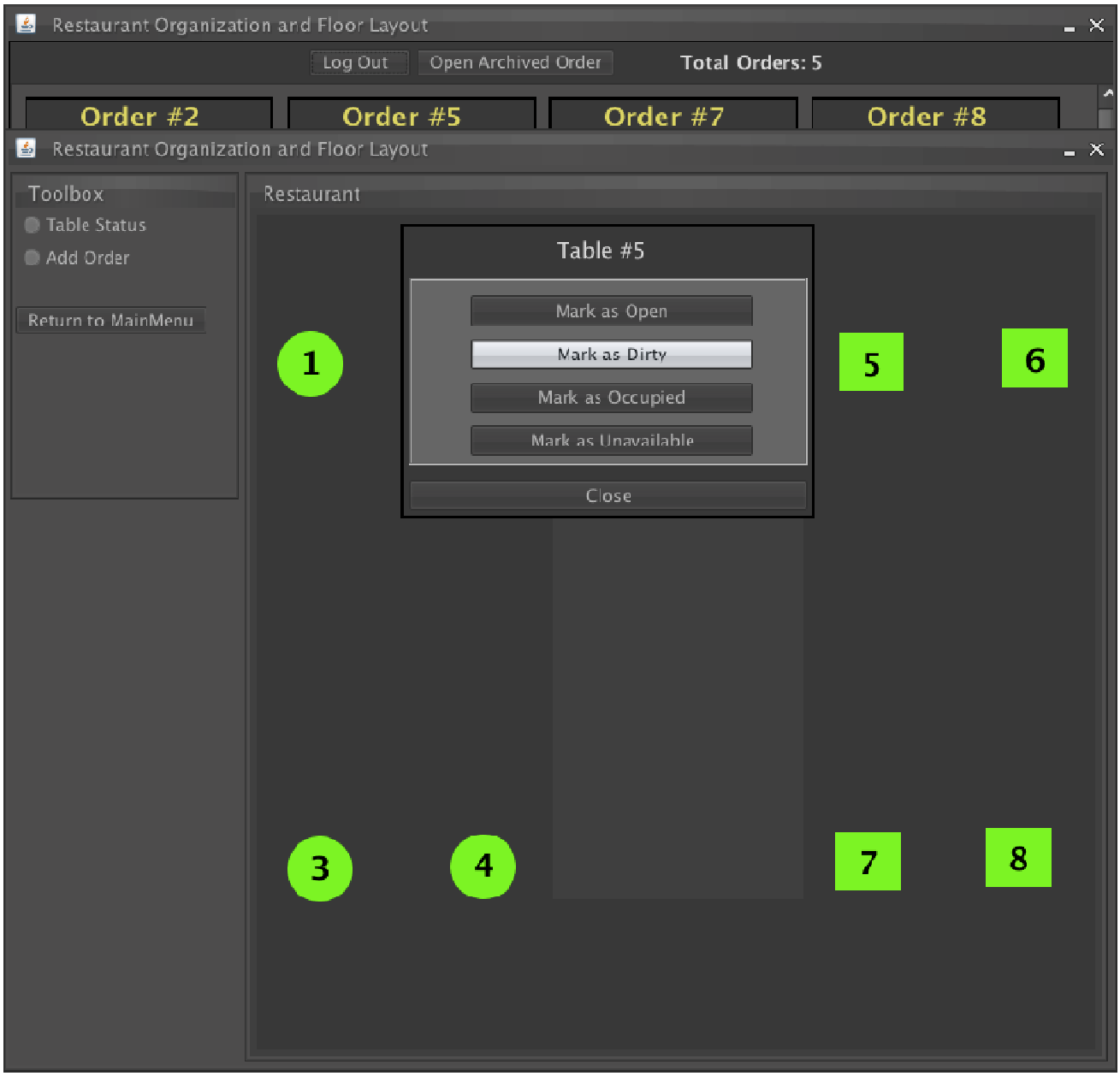
Rich Client Interface

Aestheitics is also of atmost importance in a sofware. We have gone to broad lenghts to build an animation framework just for our project.

Its a timing based engine which can do an assorted types of animations. It currently supports:

1. Motion based animations
2. Color based animation
3. Sized based animations

The framework is setup in such a way that several of these effects can be coupled together to create beautiful interfaces. They give a feeling of the program being more responsive that it actually may be.
A good illustration of where the engine is implemented is in the menu screen; where the person is given a variety of actions that can be performed is presented to the user in a very sheek and elegant way.



Another place where the animation framework plays a vital role is in the cook screen. The orders are autonomous units which are decorated with the animation framework. They are then timed and animated to give a very pleasing effect (picture below).

Progress Report and Plan of Work

Progress Report

Currently Implemented Functionality:

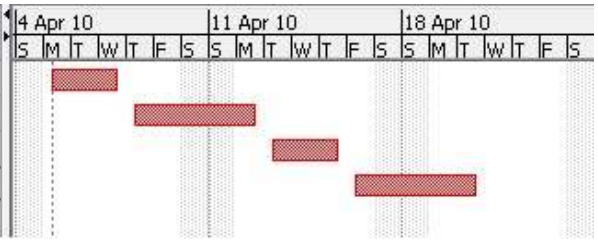
- Manage Table Layouts
- Manage Menu Items
- Seat Customer
- Clean Table
- Mark Table Dirty

Currently In Progress:

- Prepare Food

Plan of Work

	Name	Duration	Start
1	Place Order	3 days	4/5/10 8:00 AM
2	Deliver Order	3 days	4/8/10 8:00 AM
3	Handle Payment	3 days	4/13/10 8:00 AM
4	Manage Users	3 days	4/16/10 8:00 AM



Breakdown of Responsibilities

Adam Higgins

- Database Design
- Data Objects
- Asynchronous Database Listeners
- MenuManager

Tejeshwar Sangameswaran

- Front End design
- Rich Client Interface
- Layout Editors
- Animation Framework

Andrew Jones

- Data Management Screens

Kellen Zantow

- Cook Screen