2009

Software Ninjas

[IMPLEMENTATION AND TESTING REPORT]

By Wesley Williams, Trey Bland, Jacob Boniface, and Kyle Snell

TEAM MEMBERS

Wesley Williams
Project Manager
Software Ninjas
waw5709@uncw.edu

Herbert “Trey” Bland
Chief Requirements Engineer
Software Ninjas
hcb8412@uncw.edu

Jacob Boniface
Chief Designer
Software Ninjas
jab5968@uncw.edu

Kyle Snell
Quality Assurance Engineer
Software Ninjas
krs6900@uncw.edu

TABLE OF CONTENTS

INTRODUCTION

PROBLEM DESCRIPTION

Restaurant management can be a daunting task. Management software in restaurants tends to be barely usable with features that seem geared towards tracking money and profits rather than being easy and intuitive to use. Another problem that tends to plague some restaurant management systems is weak or non-existent security. In many systems, some of the needed functionality is only available to the manager, so employees are granted access to the manager-level functions which could potentially spell trouble for the managers. Perhaps it is the designer's lack of domain knowledge, or the lack of user involvement in the design process. Our goal is create restaurant software that is easy to navigate and use and still has the security and features expected from quality restaurant management software.

This report provides a view of how the design will unfold. We begin with an outline of the restaurant management system functions. We then move into the software project plan and the requirements models from the previous report. Next, we cover architecture of the system and sub-systems. Included are class, activity, state, and sequence diagrams that show what our software is supposed to do and when it does it. Then we move into implementation with real screenshots of the running software. Finally we introduce test cases and testing practices.

SCOPE AND OBJECTIVES

It is our main objective is to create software that is usable, intuitive, simple, and functions well consistently. Our chief concern is that our software be user friendly which means making menus effortlessly navigable and grouping UI components in a manner that makes them easy to find. Giving each employee the appropriate level of access to the required components is also imperative for usability and security. Are main focus in this particular software endeavor is create a single working "point of management" system that acts as both a terminal for taking orders and a terminal for generating reports and making changes to employees or items on the menu.

The software is responsible for a host of functions. Our software must be able to add employees, edit their information, and remove employees from the employee database. Menu items must be added, edited, and deleted from the menu item database. Items that can be ordered must be able to be added and removed from an order. All employees must be able to clock in and clock out. Servers must be able to do what all employees do as well as take orders. Managers should be able to do what all employees do and be able to edit item and employee information and generate reports. Reports that should be generated include sales reports showing sales by food category and the total sales from the start of the day. Orders should also be stored in the database to be used to calculate total sales. For a complete look at desired functionality see page six.

SUCCESS CRITERIA

Our software will be successful if we can create system that addresses each stakeholder's concerns. The software must be secure, reliable, and above all usable. It is will also be a great success if we continue to meet the deadlines for deliverables. Ultimately, success will depend on how well our team succeeds during usability testing.

SOFTWARE FUNCTIONS

1. Employee functions
    1.1. clock in
    1.2. clock out
2. Server functions
    2.1. employee functions
    2.2. take orders
        2.2.1. select order from list of existing orders
        2.2.2. add items to existing order
        2.2.3. remove items from existing order
        2.2.4. create new order
        2.2.5. close order
        2.2.6. add tip to order
        2.2.7. add tax to order (automatic)
        2.2.8. total price of order (automatic)
3. Manager functions
    3.1. employee functions
    3.2. server functions
    3.3. manage employees
        3.3.1. add an employee
        3.3.2. remove an employee
        3.3.3. edit an employee
            3.3.3.1. change name
            3.3.3.2. change address
            3.3.3.3. change phone
            3.3.3.4. change wages
            3.3.3.5. edit clock in/out times
    3.4. manage menu items
        3.4.1. add new menu item
        3.4.2. remove menu item
        3.4.3. edit menu item
            3.4.3.1. change item name
            3.4.3.2. change item price
        3.4.4. change tax (all menu items)
    3.5. generate reports
        3.5.1. generate sales report
            3.5.1.1. calculate sales for each category of item (entrée, drink, dessert, and appetizer)
            3.5.1.2. calculate total sales
        3.5.2. generate labor report
            3.5.2.1. calculate labor costs
            3.5.2.2. calculate labor to sales ratio
        3.5.3. generate “clocked in” report

RESOURCE LIST

type	name	description	availability	price
hardware	computers	To write the code	high	$500
	software	Pre-made code to run special hardware components	medium	$25
	Card reader devices	To read debit/credit cards and employee IDs	high	$50
human	programmers	To write the code for the system	medium	$1,000 per week
	manager	To oversee the creation and installation of software	low	$2,500 per week
	Employee ID cards	Hold information about employees	high	$5 per card
software	Window Visio	To write diagrams to plan out work schedules	high	$30
	eclipse	To write java code	high	$0
	C sharp	To write C++ code	high	$0
	Windows XP	For main interface to make the code in	high	$300

Table 1: List of Resources

The hardware resources are the physical components such as the actual point of sale computer and monitor along with any equipment needed for programming and testing. The human resources are the programmers, testers, and technical writers. Software resources are all the required software needed in order to generate code and charts and graphs that illustrate the progress of the work. Many of these resources will be provided free of charge. In an effort to simulate real world cost considerations in software engineering, we have researched several vendors to determine the typical price for the indicated resources. These prices are given in the table above.

RESOURCE ESTIMATION

The resource requirements for this project are minimal. Computers and software development tools are provided for development at the University. The project is designed so that a team a four can accomplish the goals within the two months allotted. Since several of the programmers have had domain experience in restaurant and point of sale operations, there is no need to hire or interrogate a domain expert. To implement the system, only a computer with Windows installed and a magnetic card reader is required. More computers and readers may be required for larger restaurants. The design of the database and user interface will take the most time and will require diligence on the part of the programmers to ensure all work is completed and meets the requirements.

Responsibility Matrix				
Task	Herbert	Kyle	Jacob	Wesley
Database				
Design Tables	X			
Design Queries	X			
Coding for Database	X			
Clock In/Out System		X		
Sales System				
Accept Payment				X
Taking Orders				X
Modify Orders				X
Canceling Orders				X
Reporting System				
Sales Reports			X	
Labor Report			X	
Inventory Report			X	
Configuring Data				
Edit/add Employee		X		
Edit/Add Product		X		
Edit Time Clock Entry		X		

Table 2: Responsibility Matrix

RISK PLAN

Risk	Prob.	Impact	Priority	Actions
Losing a project member	2	9	10	Make sure every team member has updated copies of work done by all team members.
Hardware breaking down	4	2	10	Make sure every team member has updated copies of work done by all team members.
Task takes longer than expected	3	3	2	Trey will be free for helping other members once the initial setup of the database is complete.

Table 3: Risk Assessment

PERFORMANCE ISSUES

There are several performance issues that may arise with this project. There will be many users logged in to the system at any time. If there are too many users on the system there may be both network issues and data integrity issues. We will need to guarantee exclusive access to data to make sure it is not accessed by another part of the software while being changed. We must also be sure to remove any exclusive lock on the stored data when other users require access to that data.

MANAGEMENT AND TECHNICAL CONSTRAINTS

The team is compromised of a group of individuals with diverse programming and technical backgrounds. Most have experience programming in JAVA and .NET, so there are really very little constraints with expertise. Most team members have a good grasp on the problem domain as many of them have worked in the restaurant industry at some point in their lives.

Management is partially done via electronic means such as email and phone calls since many of the members have different and often conflicting schedules. Each member holds the others accountable for the portion of work that was assigned to them. The project manager schedules dates and times for meetings and organizes and assigns tasks to team members. The main challenge will be communication, so the project manager will have to communicate with team members on a near daily basis.

PROJECT MONITORING AND CONTROL MECHANISMS

To monitor status of the project at all times, electronic communication through email is necessary. Team members are to update the other members with their portion of the work so that a consistent picture of the software is available to all members. This ensures that the reports that are delivered depict an accurate snapshot of the project at that time. Furthermore, the project manager schedules one meeting per week for face-to-face communication and to verify the status of each task assigned to a member of the team. This is the time where a team member can offer insight into task assigned to other members.

WORK BREAKDOWN STRUCTURE

1. Project Management
    a. Monitor progress
        i. Schedule meetings
        ii. Peer evaluations
    b. Estimate resources
        i. Determine sources needed
        ii. Check availability
    c. Assess risks
        i. Indentify risks
        ii. Prepare contingencies
    d. Evaluate results
        i. Check to make sure everyone understands the requirements
        ii. Identify weak areas and help strengthen them
2. Requirements
    a. Analyzing Functional Requirements
        i. Taking orders
        ii. Keeping order history
        iii. Tracking employees sales
        iv. Generating sales reports
        v. Tracking employee hours
    b. Analyzing Non-functional requirements
        i. Ease of use
        ii. Security
3. System Design
    a. User Interface Design
        i. Logging on/off
        ii. Clocking in/out
        iii. Ordering food
        iv. Accepting payment
        v. Printing reports
        vi. Configuring menu items
        vii. Editing employee data
        viii. Installing the software
        ix. Removing the software
    b. Database Design
        i. Creating table of employees
        ii. Creating table of orders
        iii. Creating table of items
        iv. Interfacing with UI
4. Testing/Debugging
5. Deployment

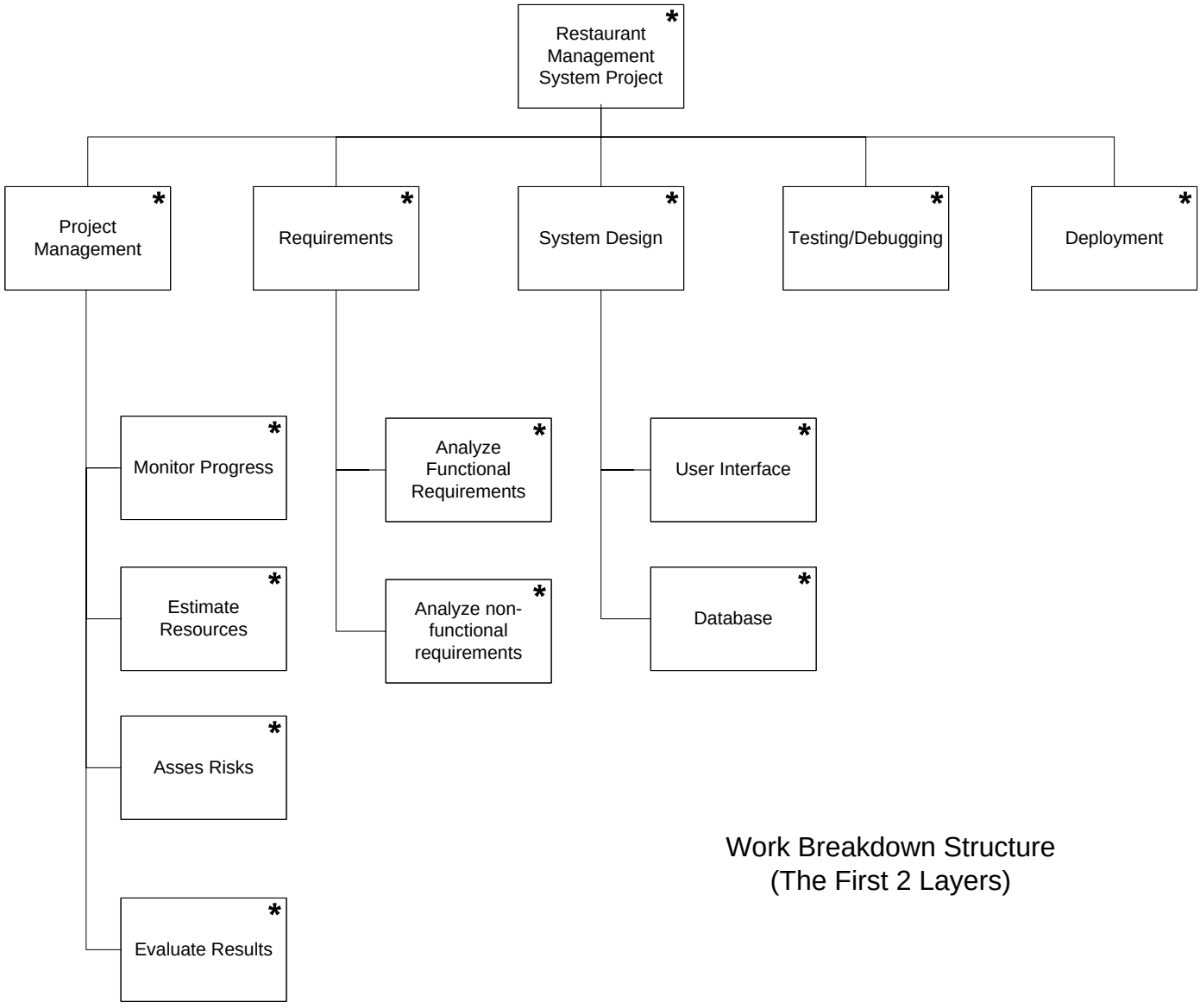
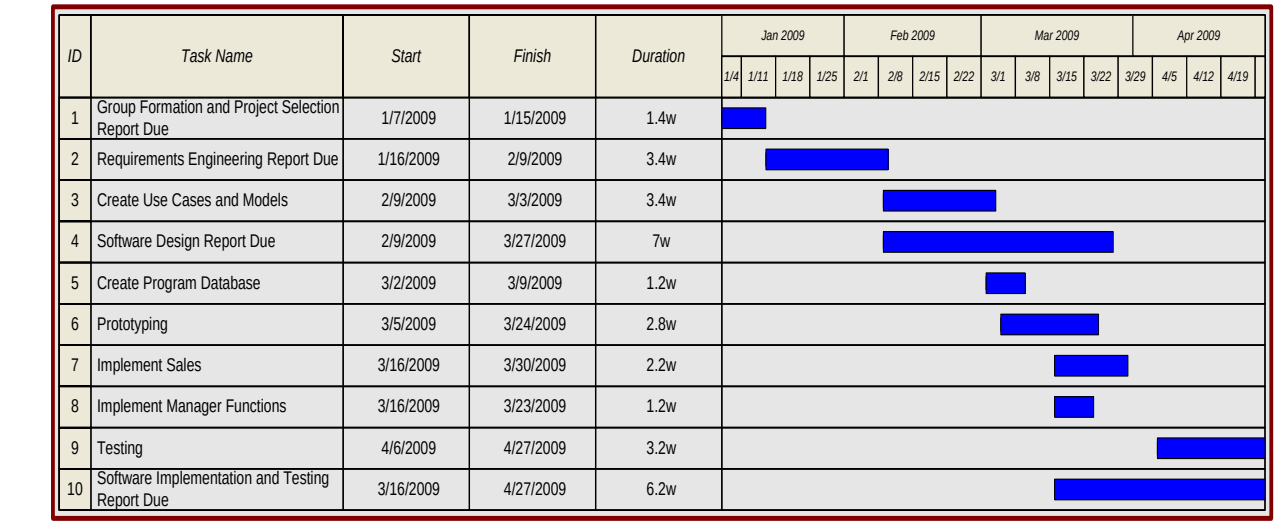


Figure 1: Work Breakdown Chart

GANTT CHART

Table 4: Gantt chart

ID	Task Name	Start	Finish	Duration
1	Group Formation and Project Selection Report Due	1/7/2009	1/15/2009	1.4w
2	Requirements Engineering Report Due	1/16/2009	2/9/2009	3.4w
3	Create Use Cases and Models	2/9/2009	3/3/2009	3.4w
4	Software Design Report Due	2/9/2009	3/27/2009	7w
5	Create Program Database	3/2/2009	3/9/2009	1.2w
6	Prototyping	3/5/2009	3/24/2009	2.8w
7	Implement Sales	3/16/2009	3/30/2009	2.2w
8	Implement Manager Functions	3/16/2009	3/23/2009	1.2w
9	Testing	4/6/2009	4/27/2009	3.2w
10	Software Implementation and Testing Report Due	3/16/2009	4/27/2009	6.2w



PERT CHART

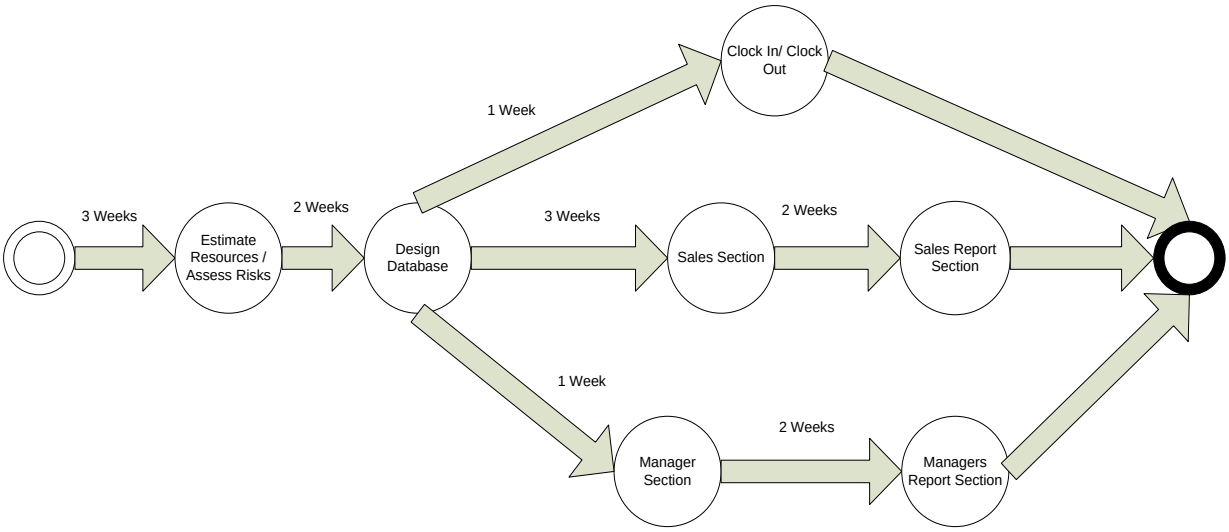


Figure: Pert Chart

ACTIVITY DIAGRAMS

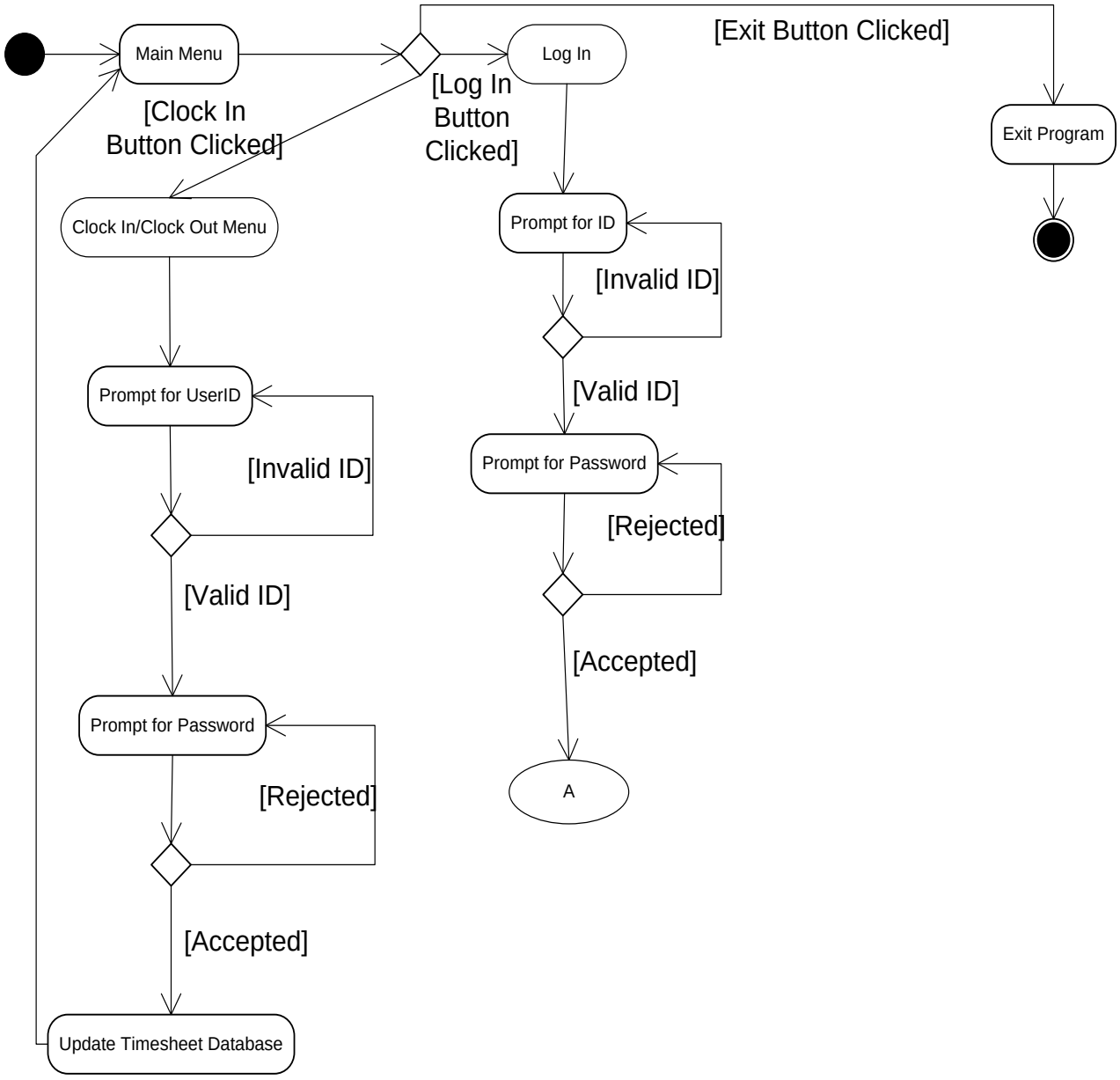


Figure: Activity Diagram 1

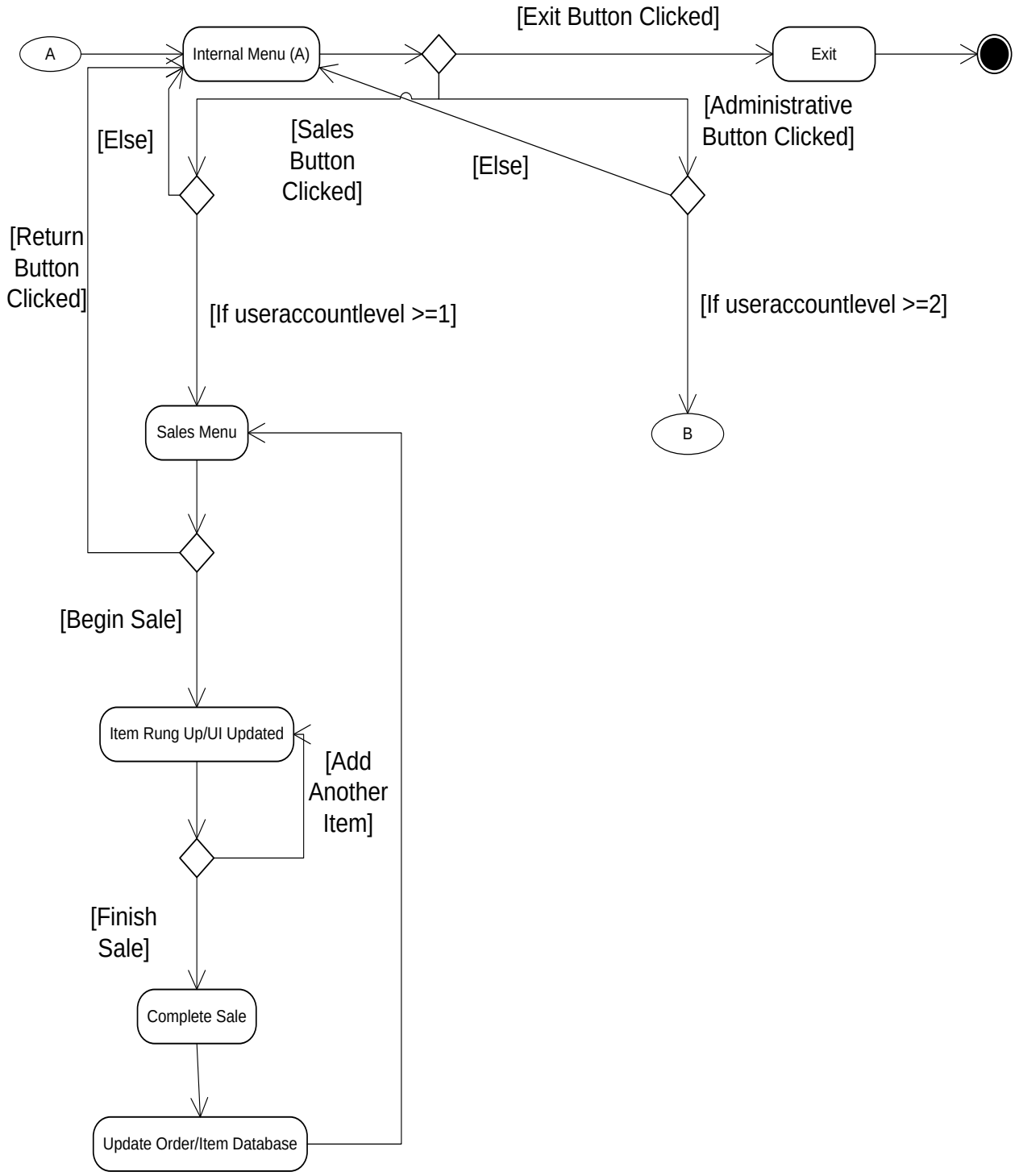


Figure: Activity Diagram 2

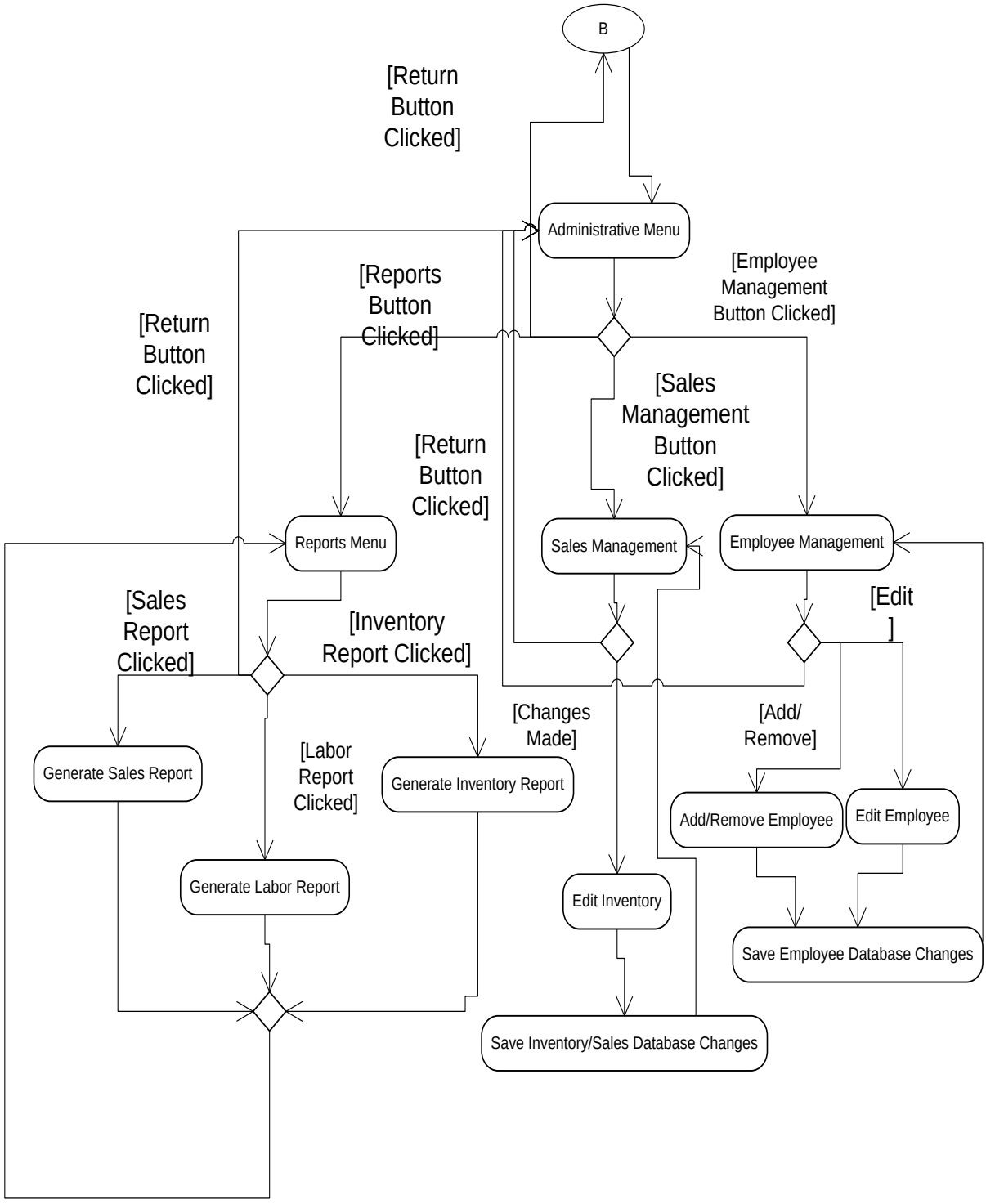


Figure: Activity Diagram 3

USE CASE DIAGRAM

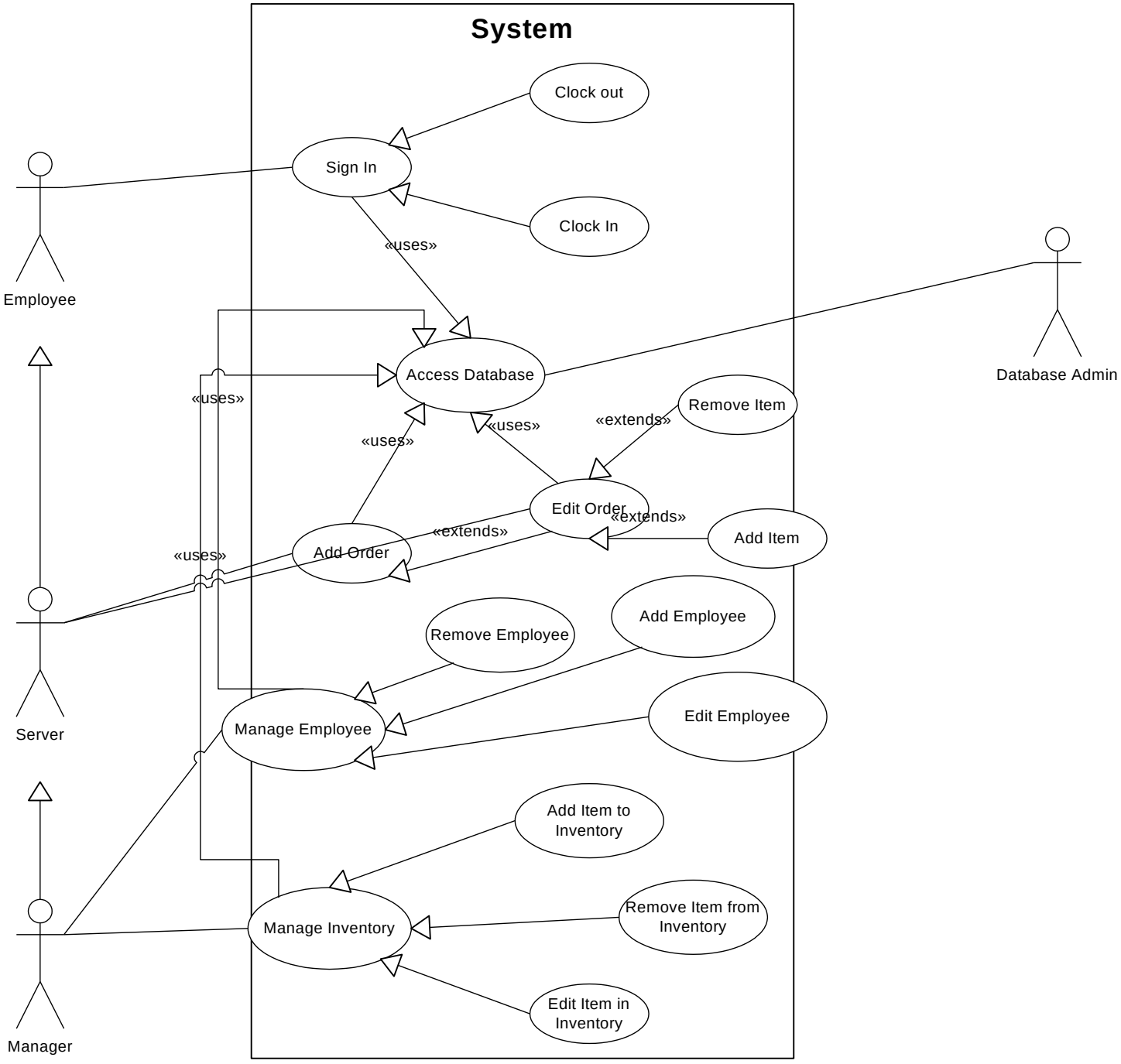


Figure: Use case diagram

DESIGN MODELS

DESIGN CONSTRAINTS

There are several design constraints that must be addressed. The software will be designed to run in a Windows environment. In order for the user interface to be a simple and navigable, we will be limiting the amount of submenus and components on the graphical user interface. One of our goals is also security so we must accommodate user ids and/or passwords. We are limited on time, so we have decided not to include the inventory functions which represent a marginal difference from our plans during the requirements phase.

ARCHITECTURAL DESIGN

The architectural styles that suit this project are the three-tiered style and the shared-repository style. They naturally lend themselves to situations calling for a central or underlying database of information. In this case, the information is orders, employees, and menu items. This information has to be retrieved from the database by the various subsystems, rearranged and formatted for output to the UI where it will be displayed in some fashion.

Below is one of the suggested architectures for this project, the thee-tier design. This particular design has a hierarchal aspect where “ordering system” is an aggregate of the “cashing out” and “taking orders” subsystems. This style is much more organized and is easy to follow.

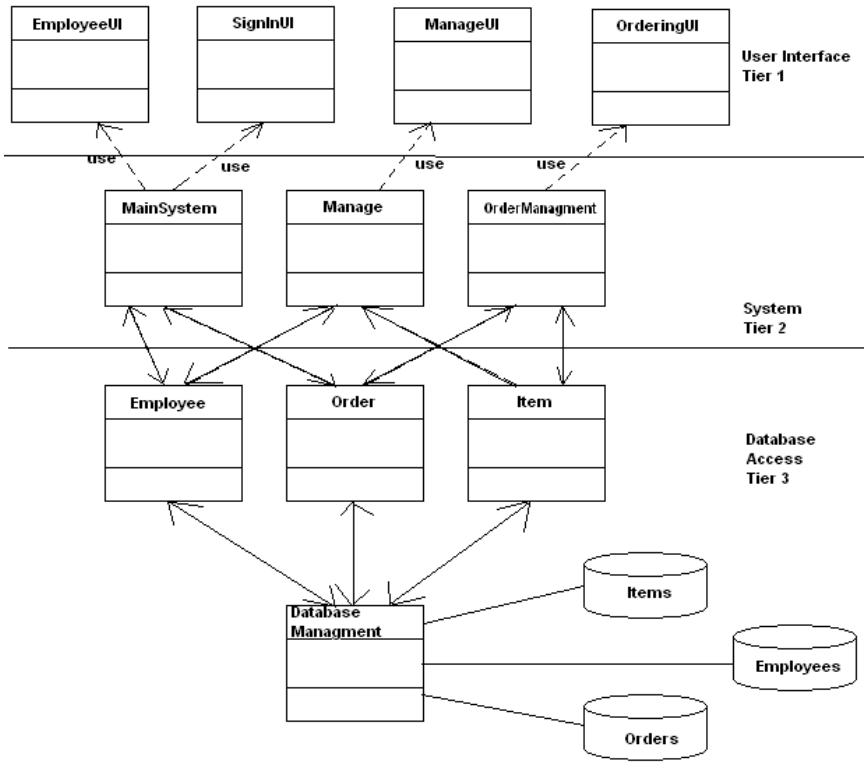


Figure 2: Three Tier Architecture

The next architecture is the shared repository style where each subprogram has access to the database(s). In this architecture, we lose the hierarchal capacity. This architecture is far simpler than the previous one. In this example, the subsystems communicate with the database like they do in the other, but there is no sense of ordering or hierarchy.

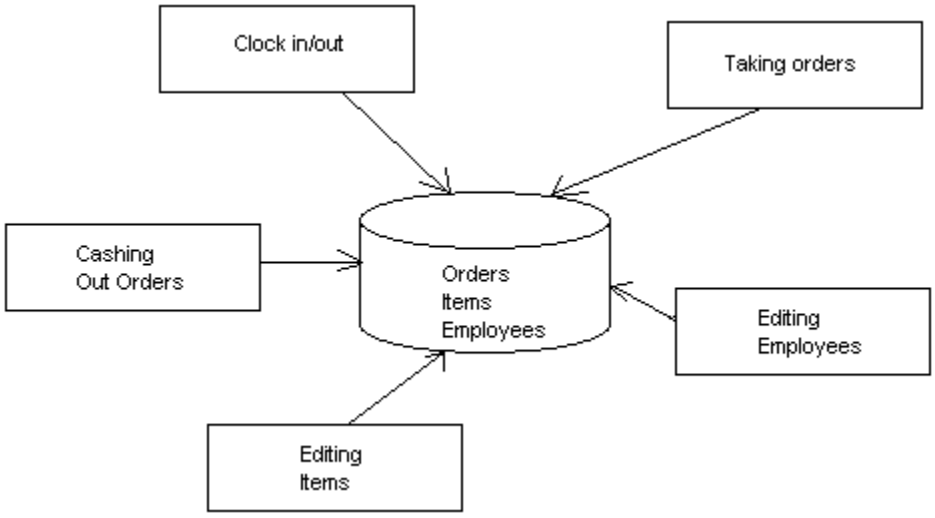


Figure 3: Shared Repository Architecture

SUBSYSTEMS

Each subsystem involves one of the primary functions of restaurant management. The clock in and clock out subsystem is responsible for clocking in and clocking out employees as well as signing in employees to take orders or manage the restaurant. The "taking orders" subsystem is responsible for creating new orders as well as adding and removing menu items from an order. The "cashing out" subsystem will be responsible for accepting payment and closing the order. The "editing employees" subsystem allows a user with manager privileges to add, remove, or edit existing information about an employee. And finally the "editing items" subsystem will govern the adding, removing, and editing of menu items. Class diagrams follow.

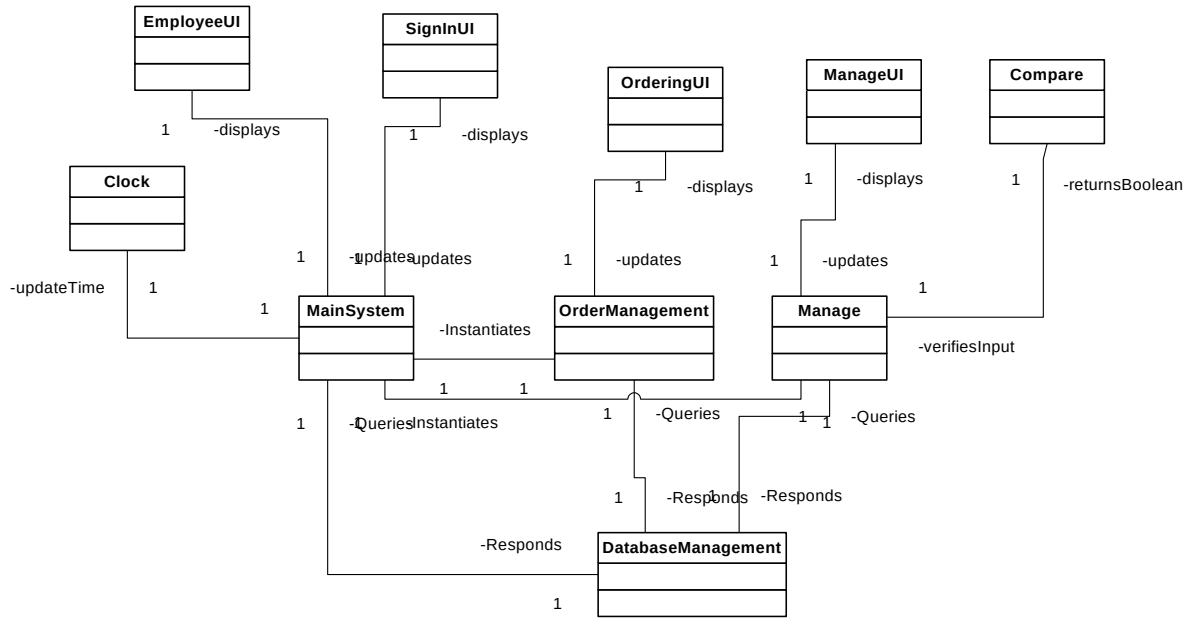


Figure 4: Class Diagram of Subsystems

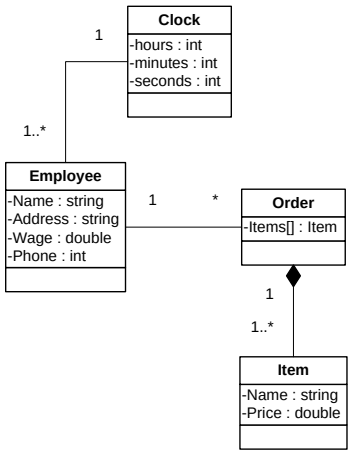


Figure 5: Class Diagram of objects

SEQUENCE DIAGRAMS

The edit Employee and edit Item functions were very simple design. First the user logs in to the management menu then either selects an item or a employee and selects the edit button and proceeds to either the item or employee edit menu. Then the user fills in whatever categories that are going to be changed and clicks OK or clicks cancel to not make any changes at all (note: no change will be made if nothing is filled in). If changes are made then the database for the item or employee is updated. The user will remain in the loop between the menus until the user hits done which logs them out of the system.

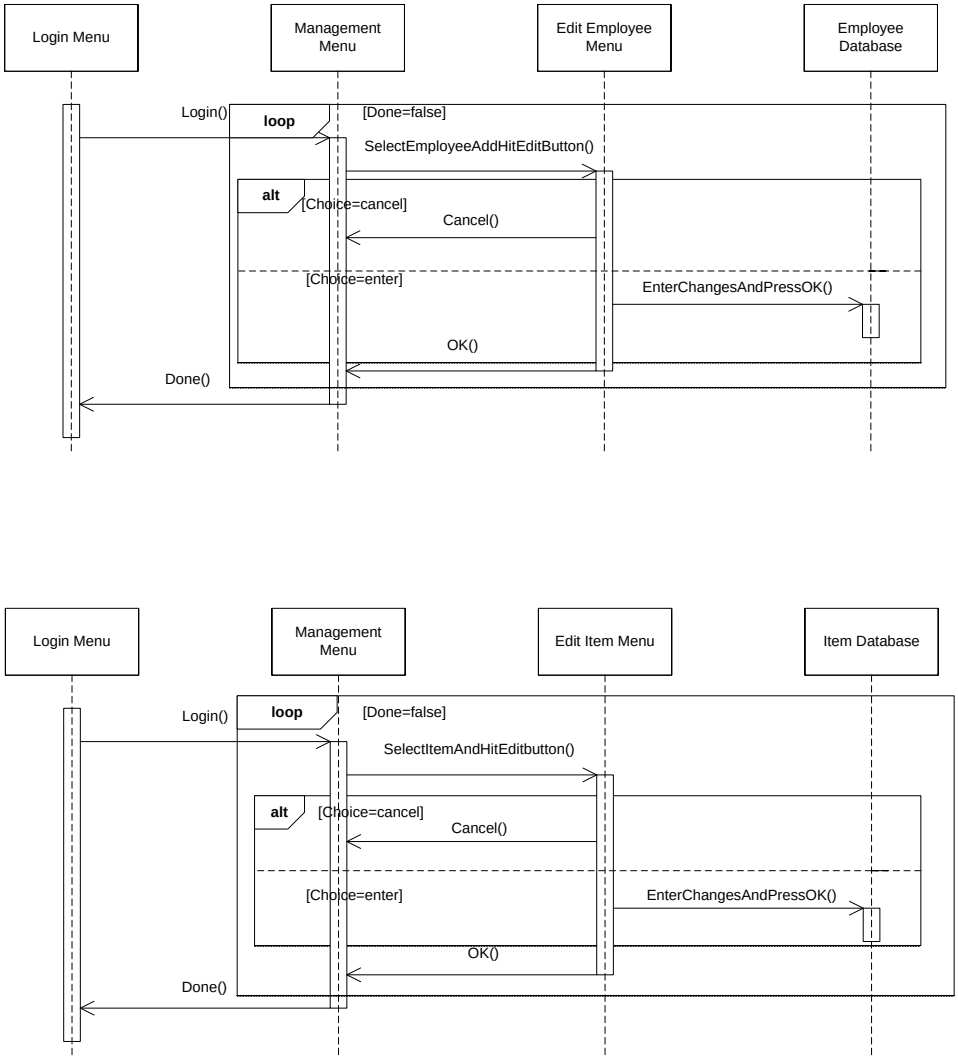


Figure 6: Edit employee (top) and edit item (bottom) sequence diagrams

To add and remove employees, the user logs in to the management menu, and in the case of removing an item or employee, simply selecting an employee or item from the list on the screen and click the delete button underneath the list. To add employees or item the user clicks add under the appropriate list, then the user is prompted to enter data for the employee or item being added. The user keeps in this loop making these four decisions until they select done which exits the loop and logs the user out.

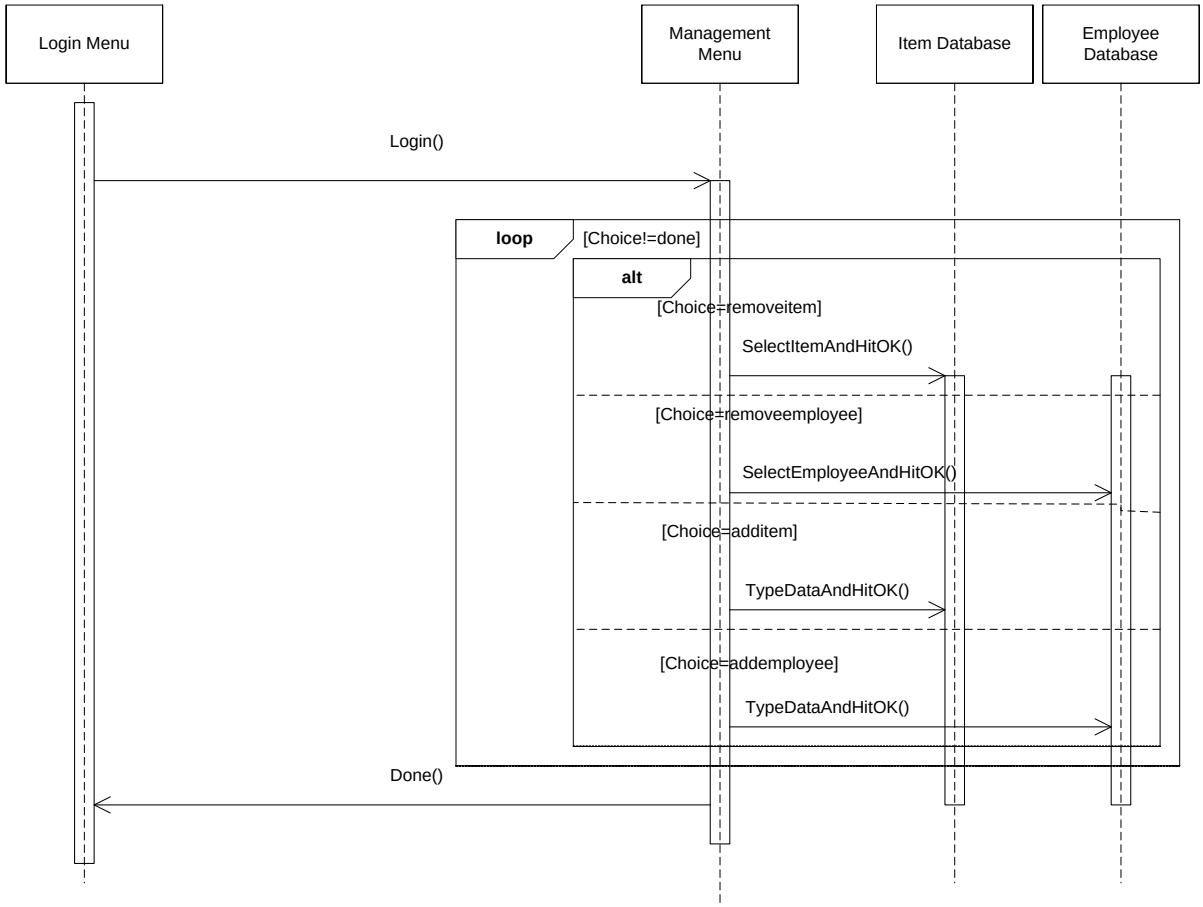


Figure 7: Add/Remove Sequence Diagram

To take orders the user first logs into the Ordering System menu there once there the user selects from the lists available what items are being order and clicks the add button under the specific list. If necessary the user can select an item from the order list click the remove button below it to remove an item. When the user is finished the user clicks the done button and to exit the loop. The user is logged off and the *item* and *order* databases are updated.

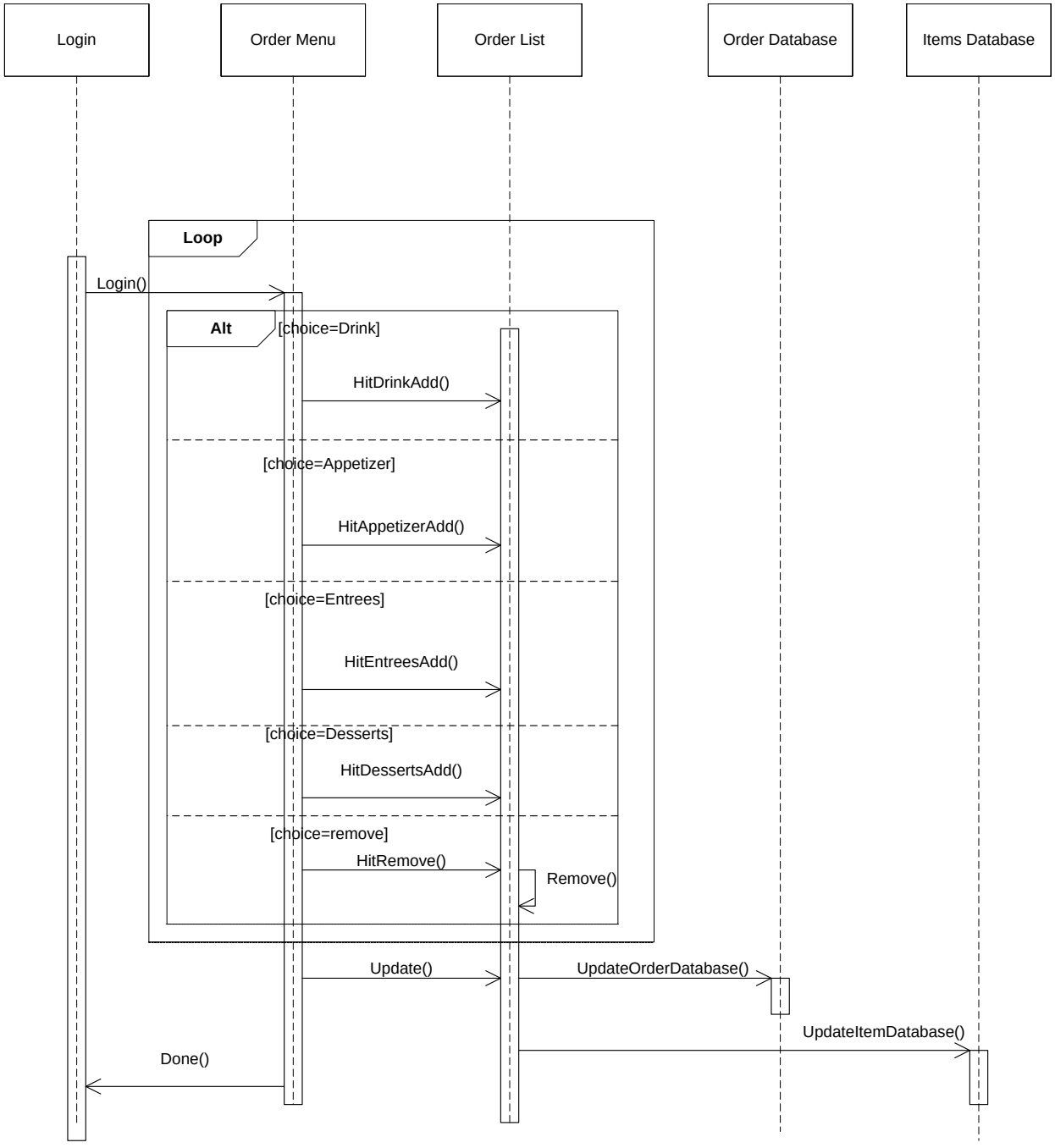


Figure 8: Ordering System

STATE DIAGRAMS

Below are state diagrams for the overall software as well as for the subsystems.

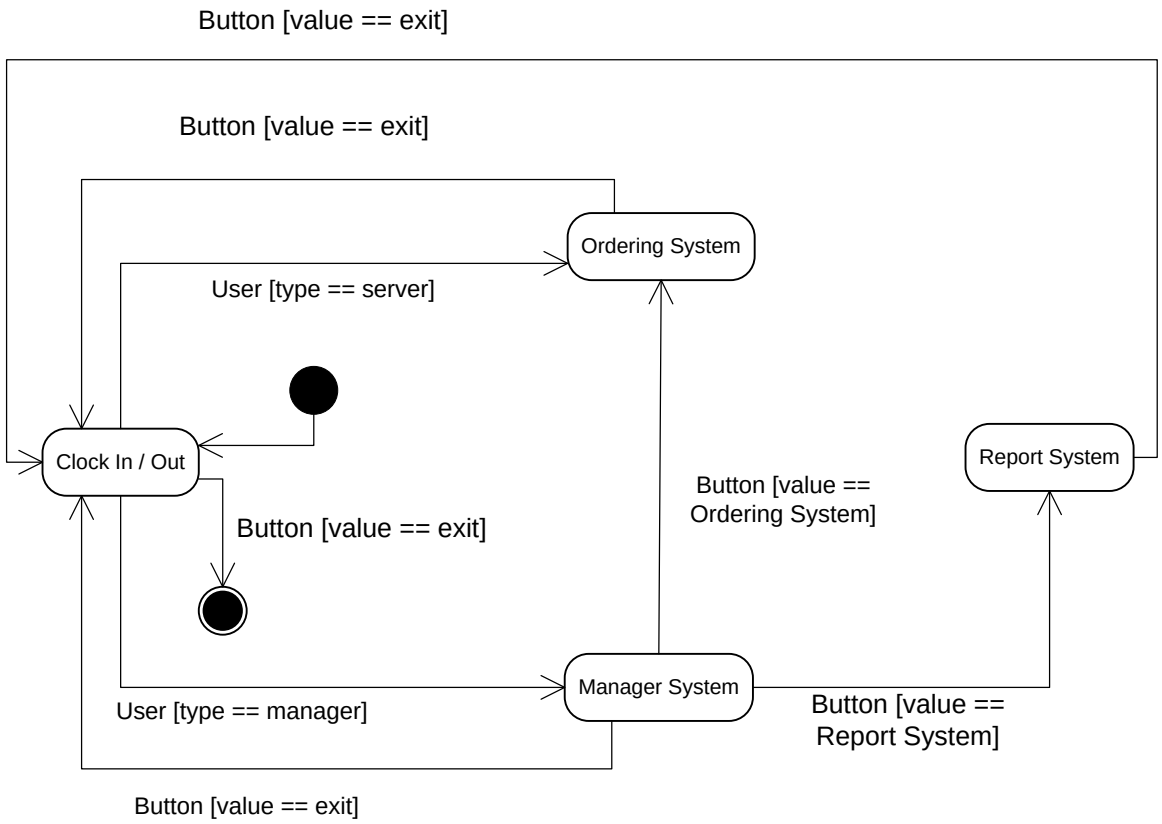


Figure 9: Main State Diagram

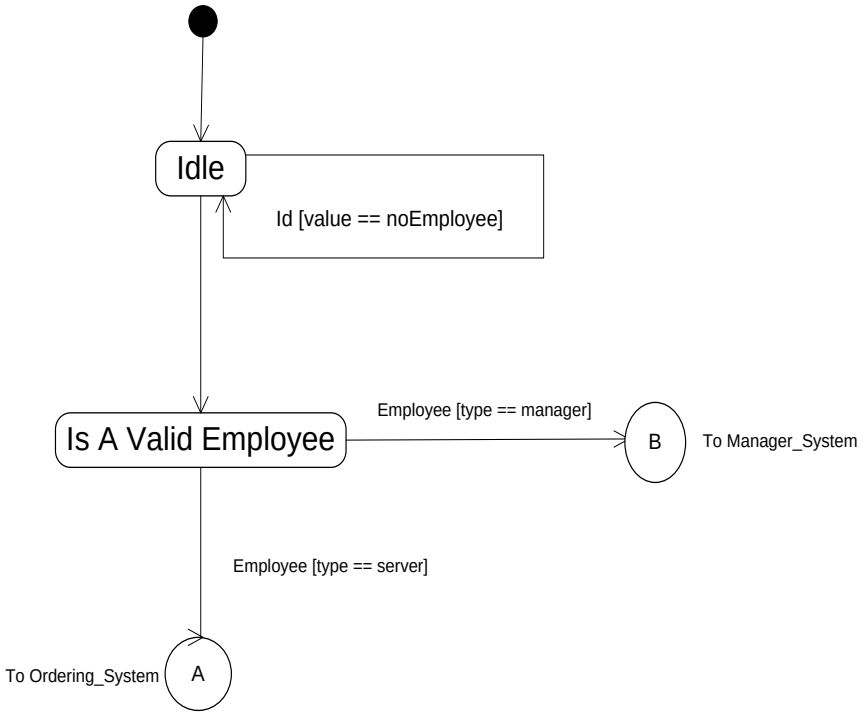


Figure 10: Login System

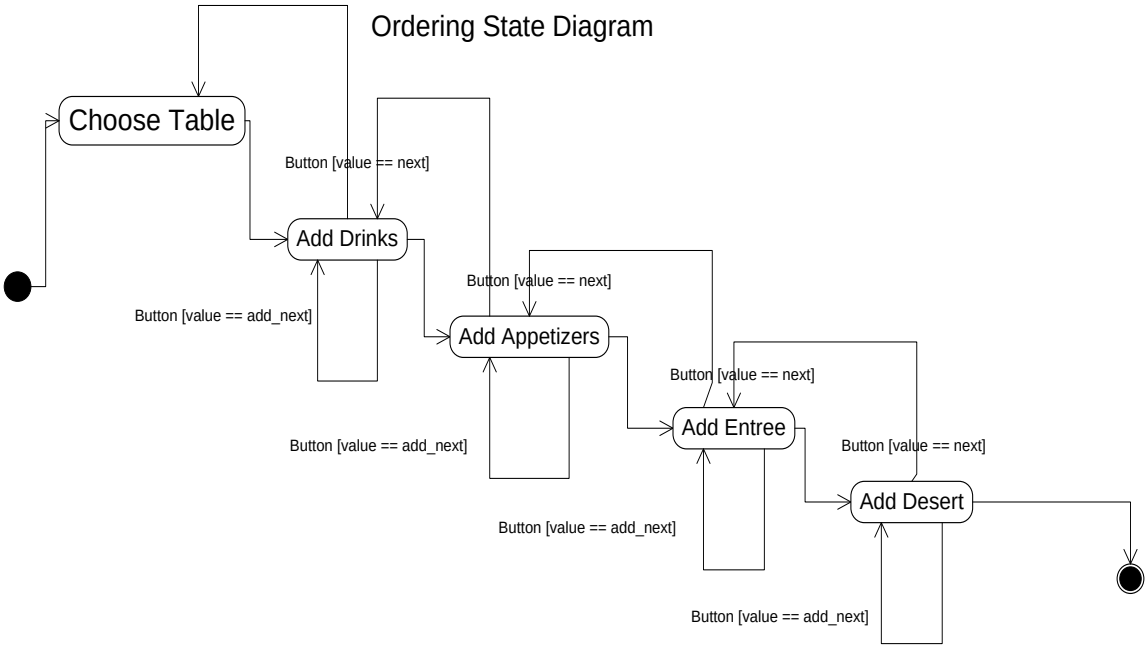


Figure 11: Ordering System state diagram

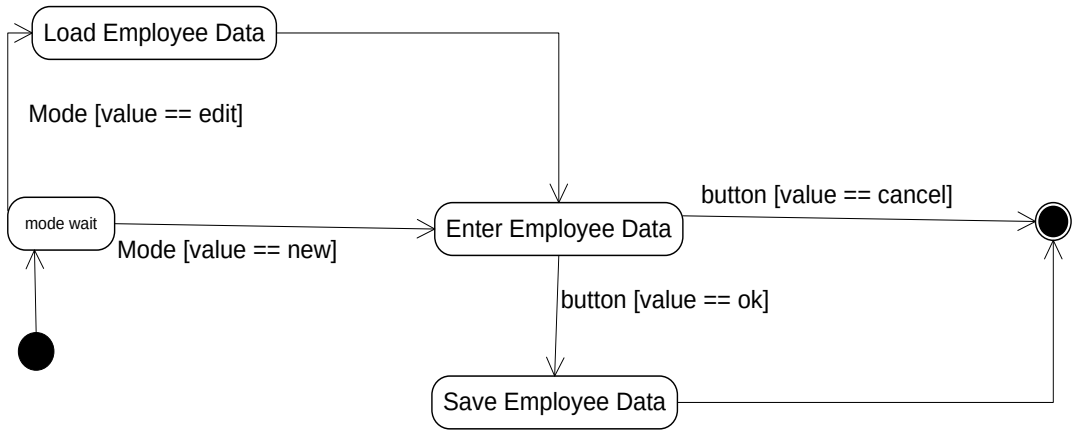


Figure 12: Add/Edit Employee State Diagram

IMPLEMENTATION

The implementation phase of the project has by far been the most challenging of the phases of this project. Many hopes and dreams were shattered. Because of the significant time constraint, many functions that initially seemed trivial to implement became very time consuming to implement. Much functionality has been left out either due to time or to manpower as many project members had other projects during the same period of time. Below is a list of functions that we intended to implement. The highlighted ones are the ones that were actually implemented as of the time of this writing.

FUNCTIONS IMPLEMENTED (HIGHLIGTED)

1. Employee functions
   - 1.1. clock in
   - 1.2. clock out
   - 1.3. sign in
   - 1.4. sign out
2. Server functions
   - 2.1. employee functions
   - 2.2. take orders
     - 2.2.1. select order from list of existing orders
     - 2.2.2. add items to existing order
     - 2.2.3. remove items from existing order
     - 2.2.4. create new order
     - 2.2.5. close order
     - 2.2.6. add tip to order
     - 2.2.7. add tax to order (automatic)
     - 2.2.8. total price of order (automatic)

3. Manager functions
    3.1. employee functions
    3.2. server functions
    3.3. manage employees
        3.3.1. add an employee
        3.3.2. remove an employee
        3.3.3. edit an employee
            3.3.3.1. change name
            3.3.3.2. change id
            3.3.3.3. change wages
            3.3.3.4. edit clock in/out times
    3.4. manage menu items
        3.4.1. add new menu item
        3.4.2. remove menu item
        3.4.3. edit menu item
            3.4.3.1. change item name
            3.4.3.2. change item price
        3.4.4. change tax (all menu items)
    3.5. generate reports
        3.5.1. generate sales report
            3.5.1.1. calculate sales for each category of item (entrée, drink, dessert, and appetizer)
            3.5.1.2. calculate total sales
        3.5.2. generate labor report
            3.5.2.1. calculate labor costs
            3.5.2.2. calculate labor to sales ratio
        3.5.3. generate “clocked in” report

Despite the many functions that were not implemented, much of the framework for the unimplemented functions exists and can easily be added to in the future. The graphical user interface was by far the most time consuming task of the entire project. Perhaps using a builder of some sort, would have proven more efficient.

USER INTERFACE DESIGN

Because of the nature of our project, an intuitive graphical user interface is required. Initially designed by Jacob Boniface, the user interface design below is the JAVA Swing equivalent of the earlier design. There are a few alterations that had to be made. The buttons on the far right side on each screen have been removed and put onto a single menu accessible after login. On the first screen visible to the user we have removed all buttons except the sign in button. The functionality that was originally on this particular screen has been moved to subsequent menus and screens. In addition, we have added the clock functionality to every screen in the program so that no matter which screen an employee is viewing, he or she will be able to keep track of time.

Figure 13: First user screen

The next screen is where the employee can choose which tasks he or she wants to do. Options that appear grayed out are not available for the user that has logged in. Managers have all options, servers have all options except the management button, and finally normal employee are only granted access to system to log in and log out. All orders that are currently "owned" by this particular employee are listed in the list above the "add" and "edit" buttons. Below are three screenshots that show this.

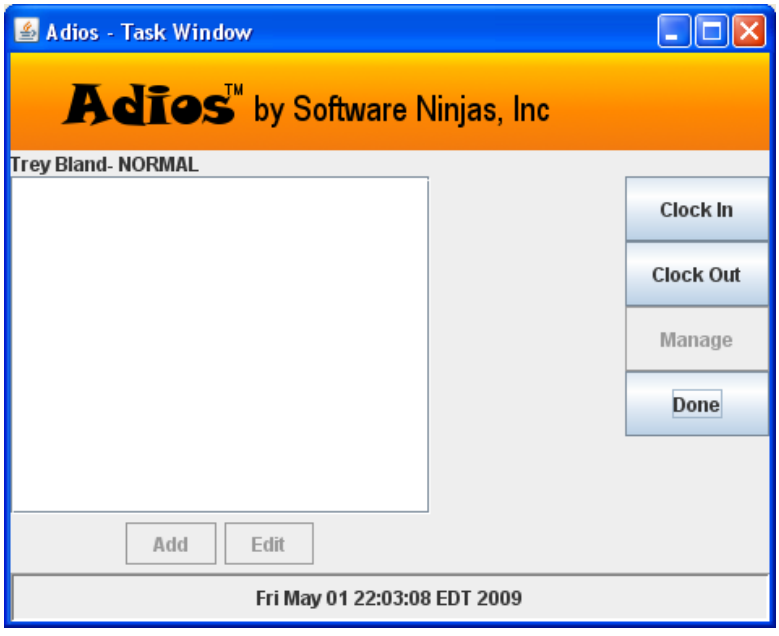


Figure 14: Employee Task Menu

Figure 15: Server Task Menu

Figure 16: Manager Task Menu

Management menu functions include the ability to add and remove employees as well as items. Although our group did not implement report generation, report generation would be completed from the management menu. Below are screenshots of the management menu and the various submenus for editing employees and items.

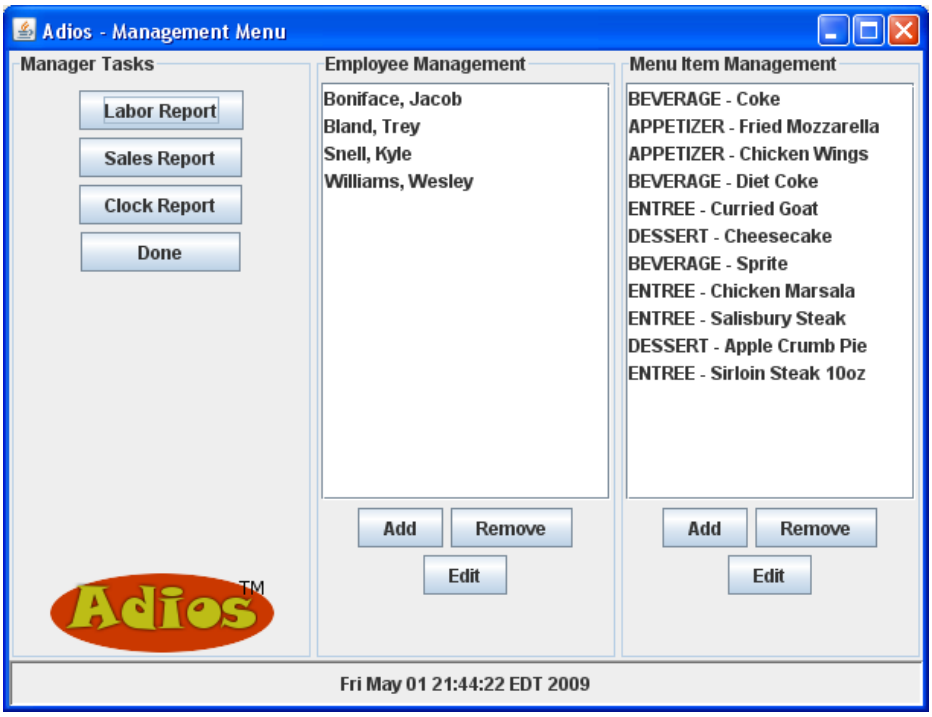


Figure 17: Ordering menu

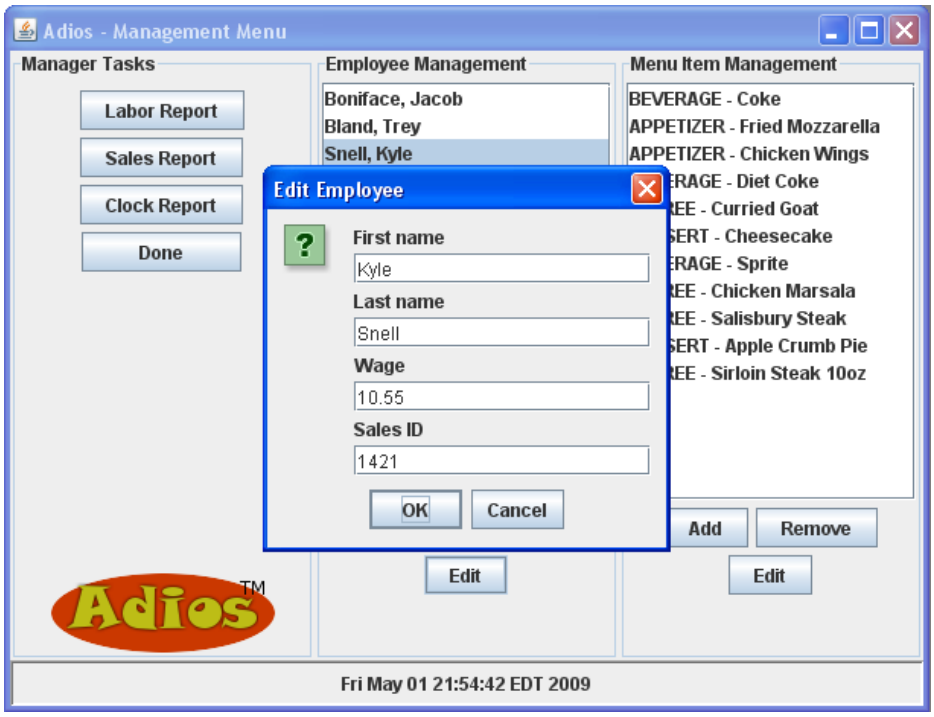


Figure 18: Edit Employee

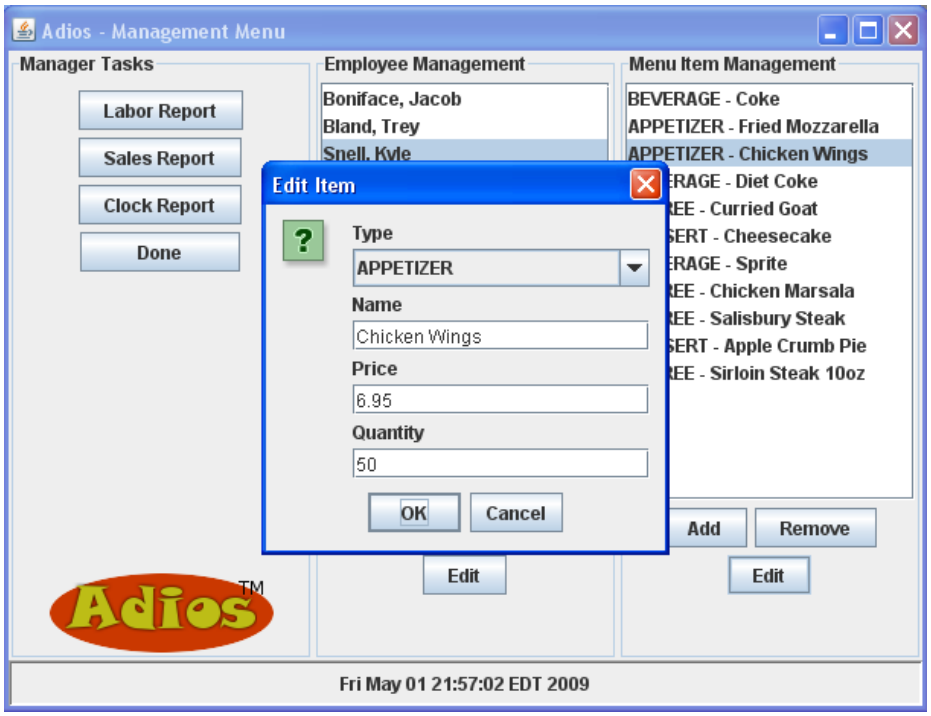


Figure 19: Edit Item Menu

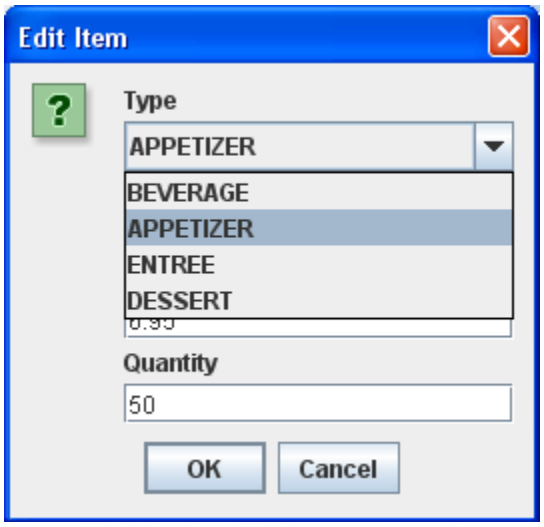


Figure 20: Item Type Selection

The ordering screen allows a server or manager to create an order by selecting from four lists that represent the four types of food: beverage, appetizer, entrée, or dessert. When the item is selected, pressing the add button below the list adds the item to a fifth list which is the actual customer order. When the screen is exited, the information is stored in the database. All the screens are periodically updated with the current contents of the database. The figure below is the ordering screen.

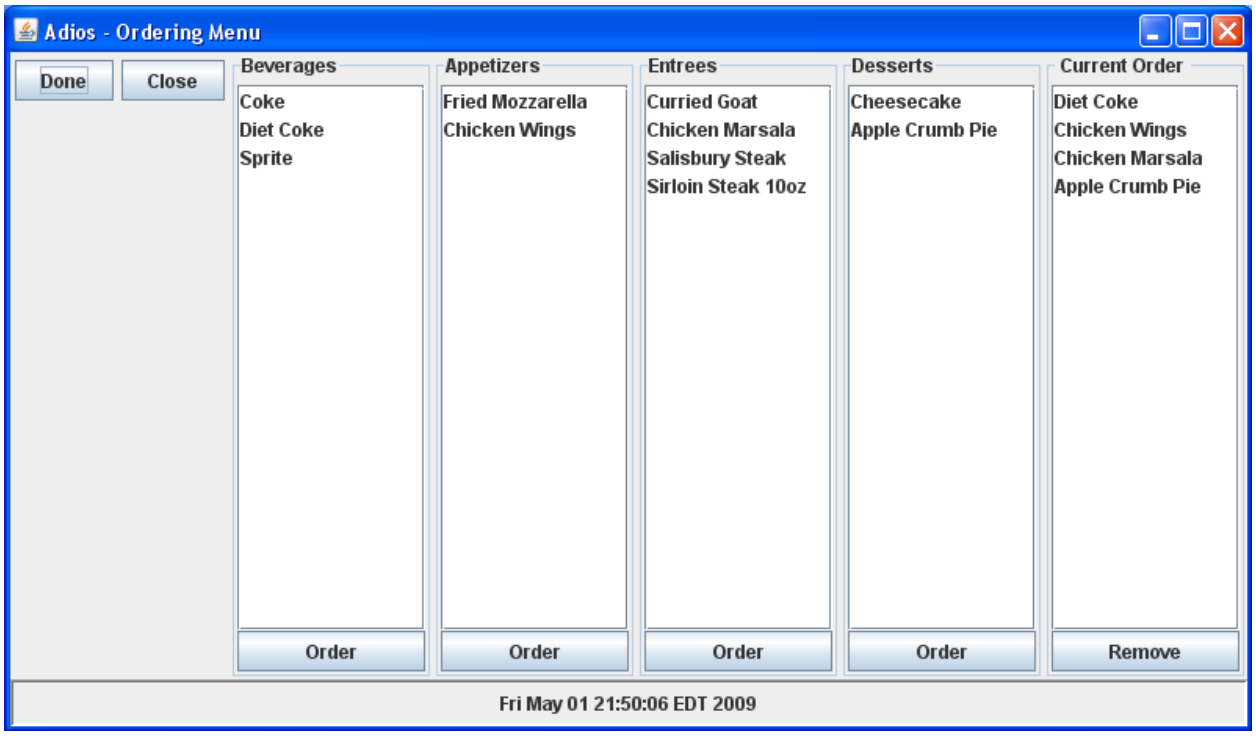


Figure 21: Ordering Screen

TEST PLAN

The test plan for the Adios system can be summed up into our three major goals: reliability, security, and usability. Reliability is the first of the goals. The software must be able to do the requested function and perform in a predictable way. The second goal is security. The program must provide the correct functionality to the employee authorized to execute those functions. Lastly, the third and most important goal is usability. We want to make sure our product does more the simply work; we want it to be usable.

Reliability and security testing can be accommodated by merely constructing a few test cases and comparing what the result should be versus what the result is. Usability testing, however, is completely different. Usability testing would require some domain experts to use the software and perhaps even deploy the software in a restaurant environment. Due to the time constraints of the project we were unable to perform any formal usability testing.

Test cases were created to test adding, deleting, and editing both items and employees. Specifically these test cases make certain that employee and items are stored and retrieved from the database correctly. Test cases were also generated to perform boundary testing on how many entries could be successfully added or updated. In addition, test cases where created to verify the function of the compare class, which is used to validate input. The table of test cases listed on the following page shows what kinds of tests were performed on the “add new employee” function, the intended results, and the actual results.

Test Case #	Description	Intended result	Actual result	Completed by
1	Add new employee (0, Devon, Simmonds, 12.35, 3245)	Employee(1, Devon, Simmonds, 12.35, 3245) MANAGER	Employee(1, Devon, Simmonds, 12.35, 3245) MANAGER	Wesley Williams
2	Add new employee (0, John, Smith, 21.55, 1245)	Employee(2,John, Smith, 21.55, 1245) SERVER	Employee(2,John, Smith, 21.55, 1245) SERVER	Wesley Williams
3	Add new employee (0, Dan, Rather, 14.35, 378)	Employee(3, Dan, Rather, 14.35, 378) NORMAL	Employee(3, Dan, Rather, 14.35, 378) NORMAL	Wesley Williams
4	Add new employee(0, 45, 54, 23.12, 234)	Error	Employee(0, 45, 54, 23.12, 234) NORMAL	Wesley Williams
5	Add new employee(0, Hugh, Laurie, -13.45, 546)	Error	Error	Wesley Williams
6	Add new employee(0, William, Shatner, 27.68, -1532)	Error	Error	Wesley Williams
7	Add new employee(0, Hernan, Cortez, 5.68, 10)	Error	Employee(0, Hernan, Cortez, 5.68, 10)	Wesley Williams

As you can see, two of the test cases above did not pass. There are some bugs with the compare class that need to be worked out. Namely, the compare class should be able to refuse any sales ID (the last argument in the method signature) that is less than three digits and it should prevent a user from entering numbers as the only characters in names. Further testing is required to verify complete operation. These bugs and bugs found in other tests would need to be fixed and retested before release.

IMPLEMENTATION AND TESTING PHASE RESPONSIBILITES

Most of the coding was done in CIS 2004. This is a computer lab located in the Computer Information Systems building at the University of North Carolina at Wilmington. The lab was the chosen place for meetings because along with computers installed with the software we required, the room afforded us the opportunity to use the white board for drawing diagrams and clarifying implementation details. Implementation and testing started in early April and continued until April 26th. Below is a table of responsibilities. It includes what was done as well as who completed it.

Task	Herbert	Kyle	Jacob	Wesley
UI				
Design	X	X	X	X
Coding				X
Database				
Table Design	X	X	X	X
Query Design	X			

Coding	X			
Program				
Sales System			X	X
Management System				X
Logon System				X
Input Validation		X	X	
Testing				
Testing Approach	X	X	X	X
Test Cases	X			X
Presentation				
PowerPoint			X	
Report	X	X	X	X

Table 5: Implementation responsibility matrix

IMPLEMENTATION & TESTING - MAJOR PROBLEMS

By far the biggest challenge encountered was time constraints. Implementation takes an extraordinary amount of time and a large amount of coordination. Scheduling project meetings around every group member's schedule has been nearly impossible. Many of the group members were unable to devote the amount of focus that the implementation stage required. Both the former and the latter problem may be more of an issue in the academic environment where priorities of the different group members are skewed in a variety of directions. Another issue that cropped up was knowledge of the JAVA programming language. At least two of the four group members were unfamiliar with JAVA's Swing API, which is JAVA's primary user interface package. Again, this may not be as much of an issue in software engineering outside the academic arena.

One of tools we found very useful, in situations where member responsibilities need to be hashed out, is the responsibility matrix. It has really been the only tool that has allowed us to continue making progress. Everyone is assigned a task, and everyone is held accountable for the completion of their assigned task. It also allows us to track tasks that need to be done. The responsibility matrix has proven to be an invaluable tool in the software engineering process

TOOLS

Tool		Description
Computer		MS Windows XP, 2 GB DDR2 RAM
Eclipse		JAVA integrated development environment
GIMP		Graphics design program
MS Word		Word processor

Table 6: Tools Required in Implementation

DATA DICTIONARY

Clock in (out)

To keep track of hours worked, an employee must clock in with an ID.

Item

Item is anything that is sold though the POS. Item attributes include name, id number, price and quantity. Any item will also have a location that it prints when it is ordered and a category that it belongs to on the menu. This will also include any ingredients that other products are made.

Item type

An item type is an enumerated variable that holds string constants representing the different kinds of items that can be ordered. For instance, a soda would be listed under the BEVERAGE item type, and mozzarella would be listed under the APPETIZER item type.

Employee

Employee is anyone that has access to the system. Employee attributes include id number, name, address, phone number, current pay rate, password and level of authorization (for system security purpose only)

Employee type

An employee type is an enumerated variable that holds string constants representing the different levels of employment within the restaurant. There are three possible strings, MANAGER, SERVER, and NORMAL.

Orders

An order is a collection of items. Order attributes include an id number, an employee that ordered the item and the id of the item ordered. An order would be similar to a receipt grouping in a retail point-of-sale system.

Sign in (out)

To receive the functionality granted to servers and managers, an authorized employee must sign in to the system so that all transactions that occur are presented as being from said employee