Development and Management of the Project:
BCDM System

Team Members

	Student 1	Student 2	Student 3	Student 4	Student 5
Name	Bryan Trinh	Fanghua Gu	Dhruv Patel	Meet Butani	Catharine Parmar

Table of Contents

Table of Figures

Table of Tables

Abstract

Dining Management System targeted for converting the Bronco Centerpointe Dining system into a new system to ensure more effective control of the sales while reducing operational costs. The new Bronco Centerpointe Dining Management system provides better control over transactions like improving the efficiency of customer and dish/beverage search, improving order management, improving accuracy of prices of dishes/beverages, improving accuracy of information included on spreadsheets from receipts, creation of an online sales application, and generating intelligent reports.

The goal of this project is to develop a system for the computerization of the dining system which integrates all of the current Centerpointe subsystems. The common transactions of the Bronco Centerpointe Dining system include the maintenance of mass user registration (student, professors, both), maintenance of mass food registration (dish, beverage), order management, and historical price information of food. In addition, the Dining Management System automatically provides the receipt consisting of date, time, customer name, and corresponding order list, consisting of food, quantity, corresponding individual and total price. Users can perform searches of food using a self-guided tool and intelligent reports will be generated to consolidate revenue information by customer and period. All queries and results will be accessible to customers anywhere, and the response to queries will be fast and efficient.

Keywords: dining management system, revenue reports, self-guided, historical price

Project Plan

1. Introduction

1.1 System-As-Is

The Bronco Centerpointe Dining Management (BCDM) system consists of multiple unconnected restaurant subsystems. There are multiple subsystems which include customer registration, transaction receipt, database of transactions, searching for dish/beverage price, and financial reports. All customers, dishes/beverages, and orders are maintained by hand, manually. Each order has a receipt provided to the customer with header information, such as date, time and customer name, in addition to corresponding individual/total prices of dishes/beverages and the quantities of each dish/beverage. All receipts are copied so that the users can manually input those data into spreadsheets. To obtain the sales transaction information, the users must manually retrieve the data to incorporate into financial reports for business analysis. There are many issues which may happen such as redundancies in the customer registration process, lack of tractability, incomplete or ineffective search for dish/beverage price, and inaccuracy in manually pulling and pushing data (prices and orders).

1.2 System-To-Be

The new Bronco Centerpointe Dining Management (BCDM) system integrates all restaurant subsystems based on a software-based solution and is responsible for all subsystems to provide the same functionality for customer registration, transaction recording into database, searching for dish/beverage price, and generating financial reports. The new restaurant system should provide a graphical user interface for users and use ORM Hibernate databases to maintain customers' information, dish/beverage prices, transaction status, and transaction history. Additionally, the software should provide the following: customer (students/professor) registration, dish/beverage registration, order registration/management, and an intelligent revenue report. In addition, an online component for order registration will be implemented for dish/beverage reservation, which will be fully integrated with the offline component. A self-guided tool should be provided for users to search for dish/beverage and their associated price and an intelligent report should be generated by the managers for consolidating revenue information by customer and period. The new restaurant should reduce unnecessary duplicate customer information (for students and professors), allow a different discount scheme for students and professors, allow historical price information of dishes/beverages, print receipts when orders are completed, and provide a report.

1.3 Project Deliverables

Table 1: Project Deliverables Table

Project Deliverable Table			
Project Name	Bronco Centerpointe Dining Management System	**Project ID**	Version 2.0

Project Manager	Bryan Trinh		**Status**	Complete
Sr No	**Deliverable Name**	**Description**	**Owner**	**Status**
1	Prepare project plan	Design process model, method, and techniques. Provide guidelines, procedures, and standards. Resources distribution and effort schedule. Demonstrate the procedures to be followed	Bryan Trinh	Complete
2	Prepare configuration plan	Prepare software configuration management, identify configuration items. Determine environment and infrastructure, and provide versioning and other documents etc.	Bryan Trinh	Complete
3	Requirements specification	Clear high-level goals, use case diagram, and specifications. Provide functional requirements and non-functional requirements. Traceability Matrix is also included	Bryan Trinh, Fanghua Gu	Complete
4	Design	ER diagram, logical model, UML class diagram, and other diagrams etc.	Bryan Trinh, Fanghua Gu	Complete
5	Prepare testing plan	Prepare unit tests, integration tests, and system, acceptance tests, and provide quality assurance plan	Bryan Trinh	Complete
6	Design prototype	Design paper-based prototype and provide technical description of the system	Bryan Trinh	Complete
7	Discussion	Analysis of the results obtained, and lesson learned	Bryan Trinh	Complete

2. Process Model

2.1 Activities to be Undertaken

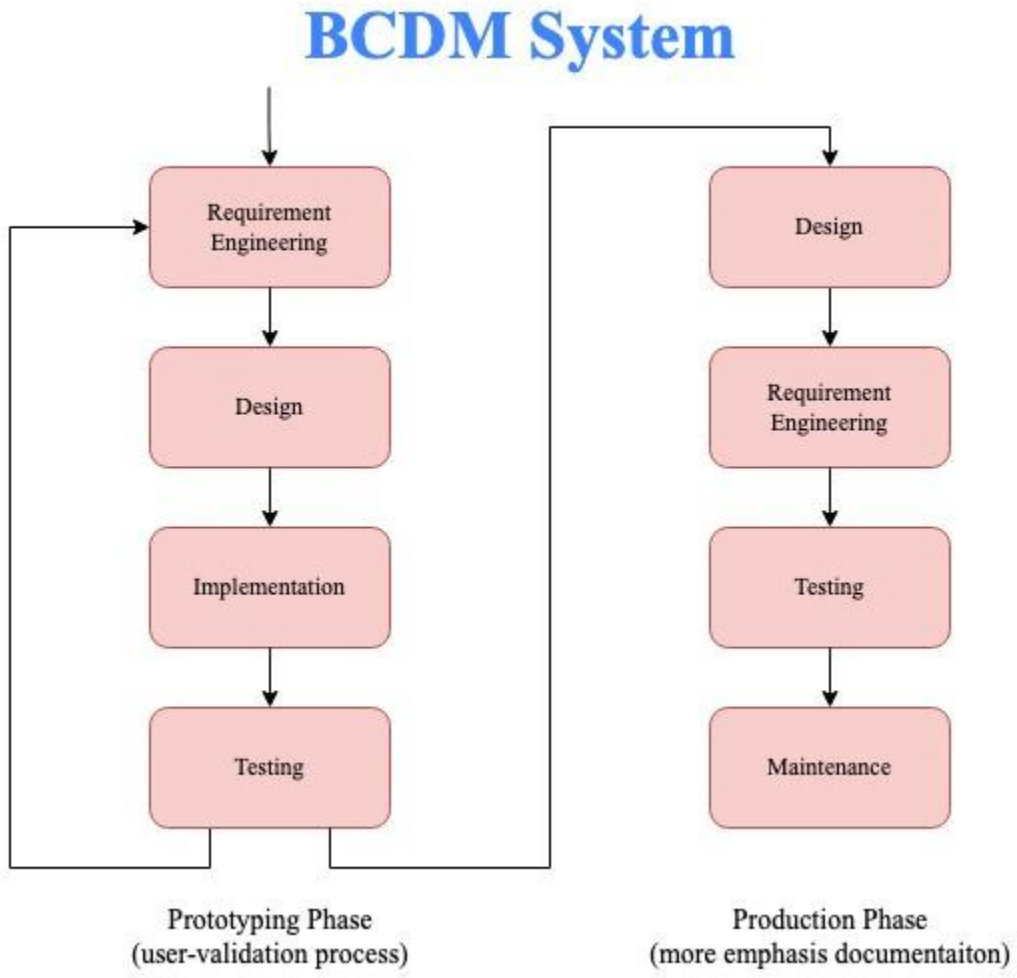

Figure 1. Evolutionary Prototype Model

The BCDM System will use Evolutionary Prototype as we see from Figure 1. Since we have two prototypes, the development of multiple prototypes can iteratively capture the requirements, as the prototypes are intensively tested for each of their respective metrics and requirements before a real production line is set up. We will fix any errors or bugs during the user-validation process and then move to the production phase, which is why the evolutionary prototype model is the perfect model that we implement to create our product.

2.2 Milestones to Measure Progress

Table 2: Milestone to measure progress

Activity	Duration	Constraints	Milestone

Requirements	5		
Design	10	Requirements finished	
Test Plan	5	Design finished	M1
Coding	10	Design Finished	M2
Test	10	Coding finished, Test plan finished	M3

After requirements are finished, we will conduct project design, which is also a constraint of the testing plan, so we set the test plan as milestone M1. Project design is also a constraint of Coding, and we will conduct Coding after Project design finishes. We set the Coding as milestone M2 since we will have a large programmatic percentage of our product in the current prototype to be completed. Coding and test plan are constraints of Test and we set Test as milestone M3. All milestones are to ensure that the project is on a good track and helps to keep it on time and within budget.

3. Organization of Project

3.1 Information, Services, Resources to be Provided

All customer and food resources should be provided by the old Dining system database when the Bronco Centerpointe Dining Management system starts. The customers will input their own credentials to register into the system, using their Bronco ID, and will be able to search and add food items to their orders. Staff can log in with admin credentials to adjust prices and create new food items. Now, all data will be stored on the server, and only current orders will be handled internally by the application, as in user credentials, food items, and prices etc. will be stored on a server. Most dining management services will be inherited from the old system, and also new features will be added such as self-guide tools, intelligent reports, and etc.

3.2 Specification of the Roles

Table 3: Specification of roles

Role	Name	Responsibilities
Project Manager	Bryan Trinh	Design process mode, methods, and techniques, Provide guidelines, procedures, and standards. Resources distributions and effort schedule. Demonstrate the procedures to be followed
Designer	Bryan Trinh, Fanghua Gu	Design ER diagram, logical model, UML class diagram, and other diagrams, etc. Design paper-based prototype and provide technical description of the system
Analyst	Bryan Trinh, Fanghua Gu	Clear high-level goals, use case diagram, and specifications. Provide functional requirements and non-functional requirements. Traceability matrix is also included

Programmer	Bryan Trinh, Fanghua Gu, Dhruv Patel, Meet Butani, Catharine Parmar	Coding and debugging. Designing and testing computer structures. Troubleshooting system errors. Writing computer instructions. Managing database systems
Tester	Bryan Trinh, Dhruv Patel, Meet Butani, Catharine Parmar	Prepare unit tests, integration tests, and systems. Acceptance tests. Reviewing software requirements and preparing test scenarios. Executing tests on software usability. Analyzing test results on database impacts, errors or bugs, and usability

4. Methods and Techniques

Teams will design ER diagram, use case diagram, traceability matrix by using draw.io during the requirement engineering phase, and will also provide UML class diagram, logical diagram, and physical diagram by using draw.io and SQL developer in order to make the project more clear and organized. We will conduct our coding and testing by using Java programming language in Visual Studio, VS Code, and/or Eclipse IDE, and all data or information will be stored in the MySQL database. For UI design, teams will use Java Swing GUI, in addition, automating testing will be provided using Selenium, Cucumber, and JUnit. Since we will use an evolutionary prototype model, we will compose our documentation following Jira software during the production phase

5. Standards, Guidelines, and Procedures

One of the teams constantly reviews requirement specifications, and ensures to meet all the requirements. One of the teams will periodically inspect project progress and project quality before moving to subsequent phases correctly, after that teams will write verification processes. Since the initial level is being currently implemented (CMMI), teams currently implement requirements management, project planning, project monitor and control, and also work on process and product quality assurance, measurement and analysis, and configuration management. After teams finish those processes. the project will move to a repeatable level.

6. Work Packages

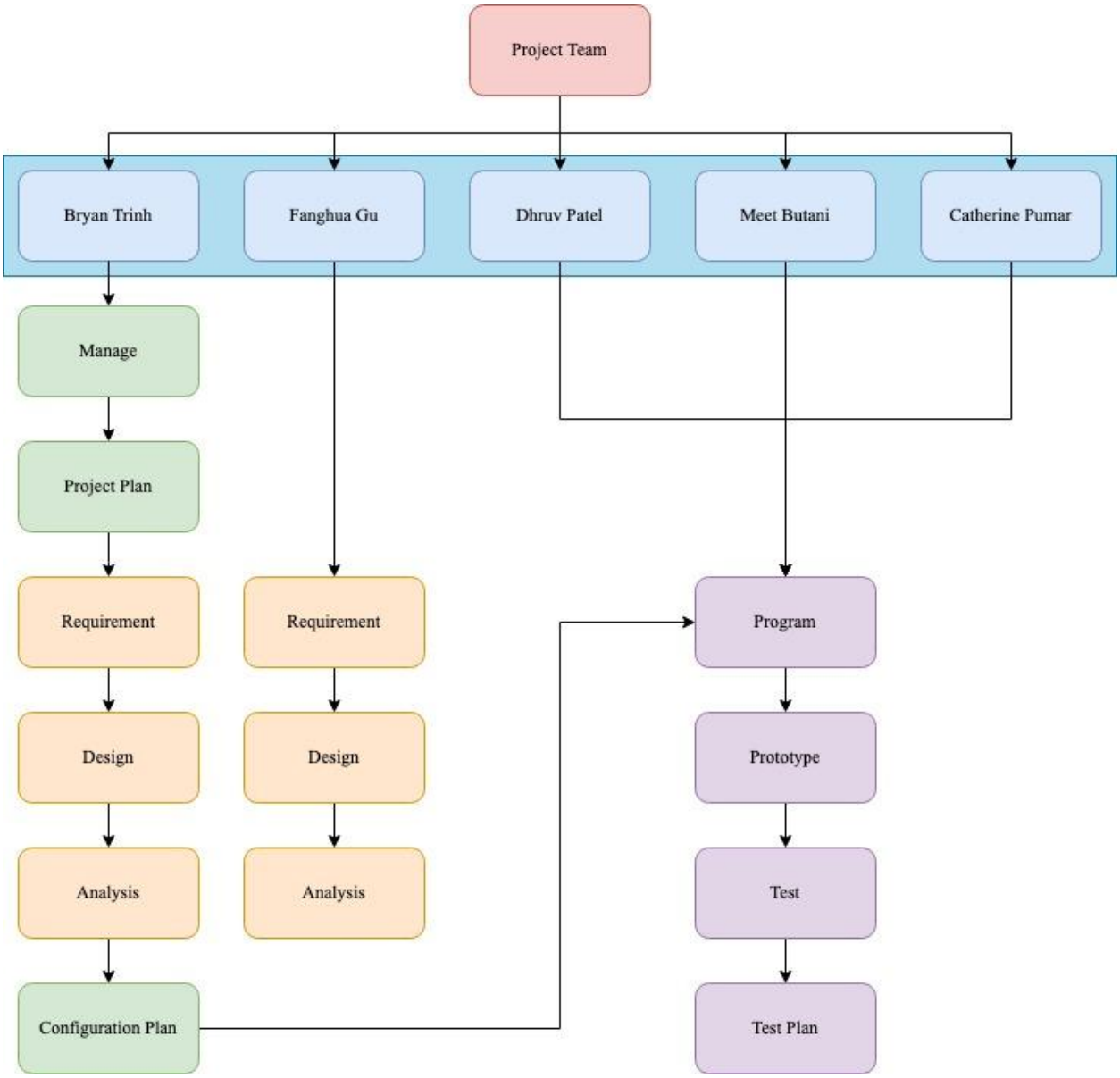

Figure 2. Work packages

7. Resources

7.1 Hardware

The Bronco Centerpointe Dining Management System uses Javascript for frontend, Java for backend, and MySQL for database. Any OS that supports Javascript, Java, and MySQL should be able to support Bronco Centerpointe Dining Management System. For instance: Windows, Linux, and Mac OS. Suggested minimal production environments: Quad Core (64 bit CPU and OS), 4 GB RAM, and 100GB

storage. In addition to the computer and display ,the program needs a normal computer keyboard and mouse for control.

7.2 Personnel

One project manager, Two designers, Five analysts, Five programmers, Five testers
(Multiple roles for certain personnels, with at least 1 personnel in each role)

8. Effort and Schedule

8.1 Effort

External Input

FTR \ DET	<5	5-15	>15
<2	Low	Low	Average
2	Low	Average	High
>2	Average	High	High

External Output / External Inquiry

FTR \ DET	<6	6-19	>19
<2	Low	Low	Average
2-3	Low	Average	High
>3	Average	High	High

Data Functions

RET \ DET	<20	20-50	>50
1	Low	Low	Average
2-5	Low	Average	High
>5	Average	High	High

Type of Functionality	Low	Average	High
Internal Logic Files (ILF)	7	10	15
External Interface Files (EIF)	5	7	10
External Input (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiry (EQ)	3	4	6

Table 0: Complexity and Contribution table for reference

SCREEN_LOGIN

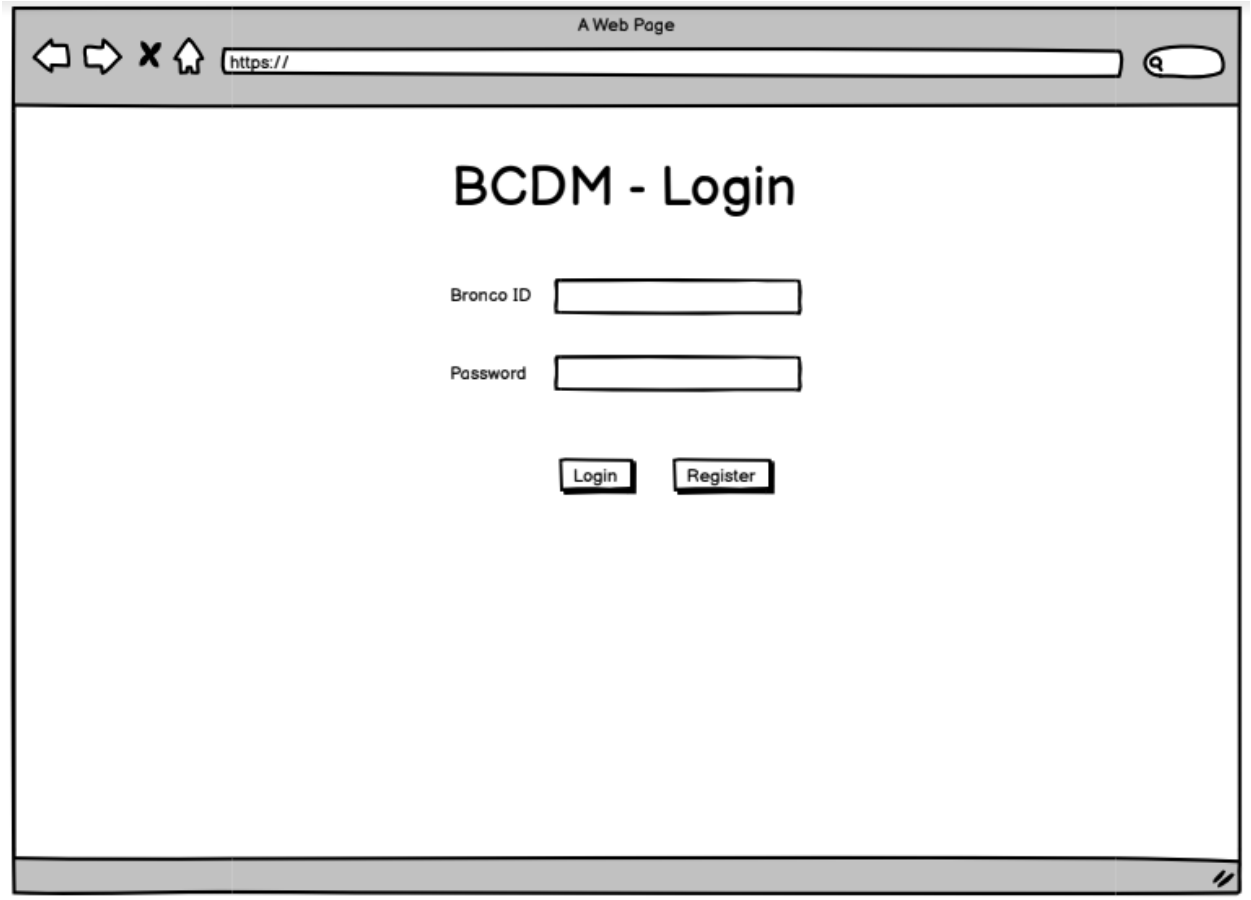

Figure 1: SCREEN_LOGIN with Login, Register

Table 1: SCREEN_LOGIN FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
User	EIF	2	1	LOW	5
Functions					
Buttons					
Login	EI	4	1	LOW	3
				Total	8

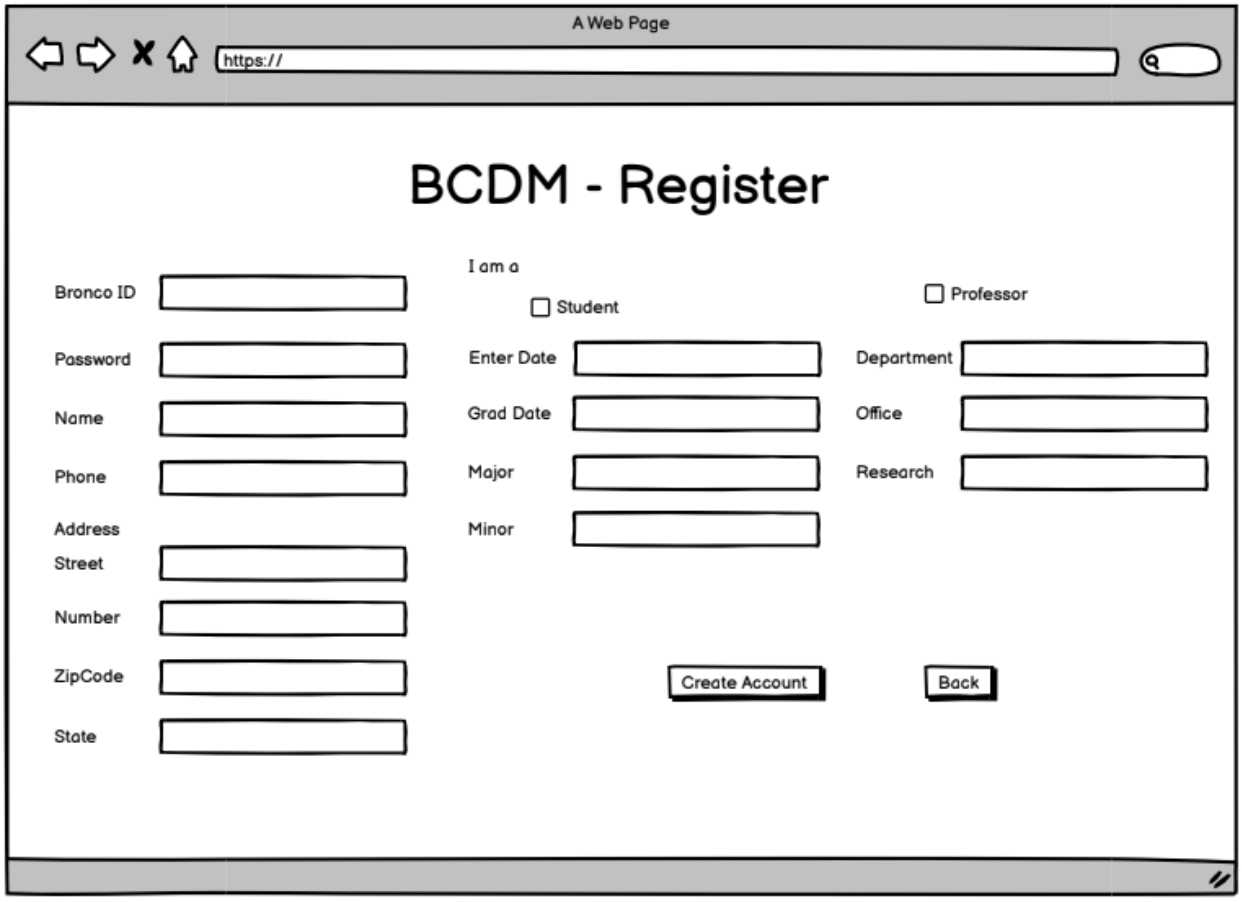

Figure 2: SCREEN_REGISTER with Bronco ID field and password. Note the multi value selection of student and professor

Table 2: SCREEN_REGISTER FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
User (Customer)	EIF	12	1	LOW	5
Address	EIF	4	1	LOW	5
Functions					
Buttons					
Create Account	EI	6	1	LOW	3
				Total	13

SCREEN_ADMIN

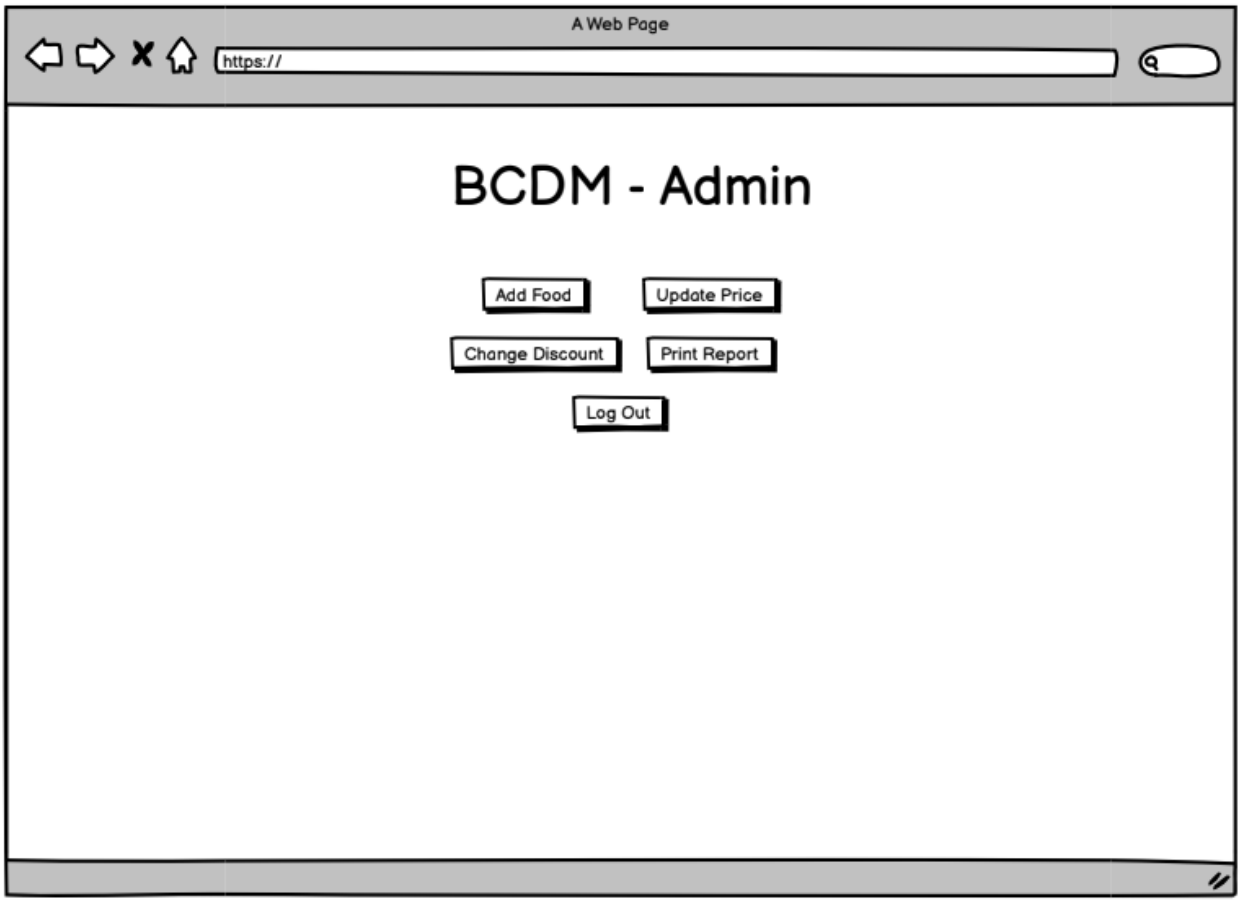

Figure 3: SCREEN_ADMIN with Add Food, Update Price, and Print Report

SCREEN_ADDFOOD

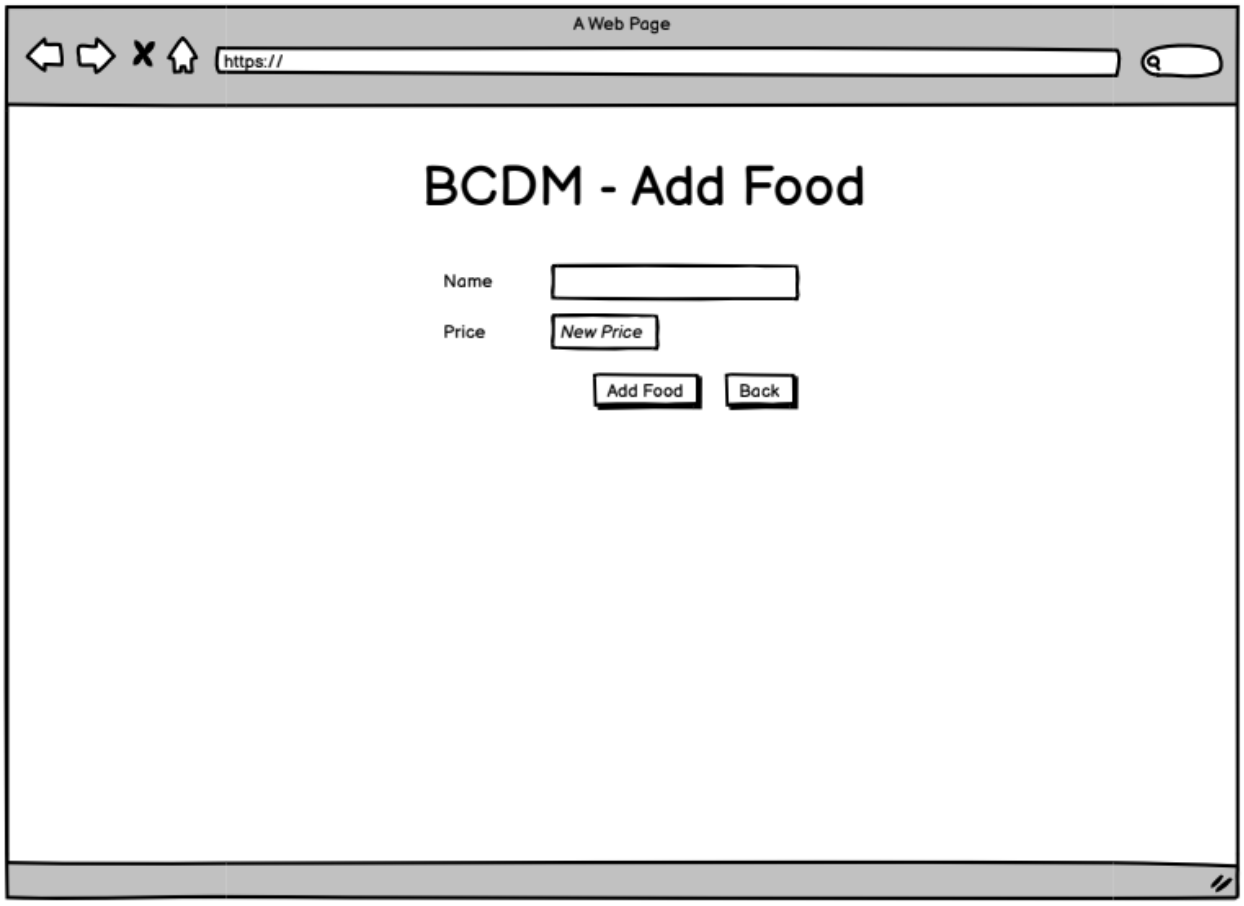

Figure 4: SCREEN_ADDFOOD can add food with name and price fields

Table 3: SCREEN_ADDFOOD FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
Food	EIF	2	1	LOW	5
Functions					
Buttons					
Add Food	EI	4	1	LOW	3
				Total	8

SCREEN_UPDATEPRICE

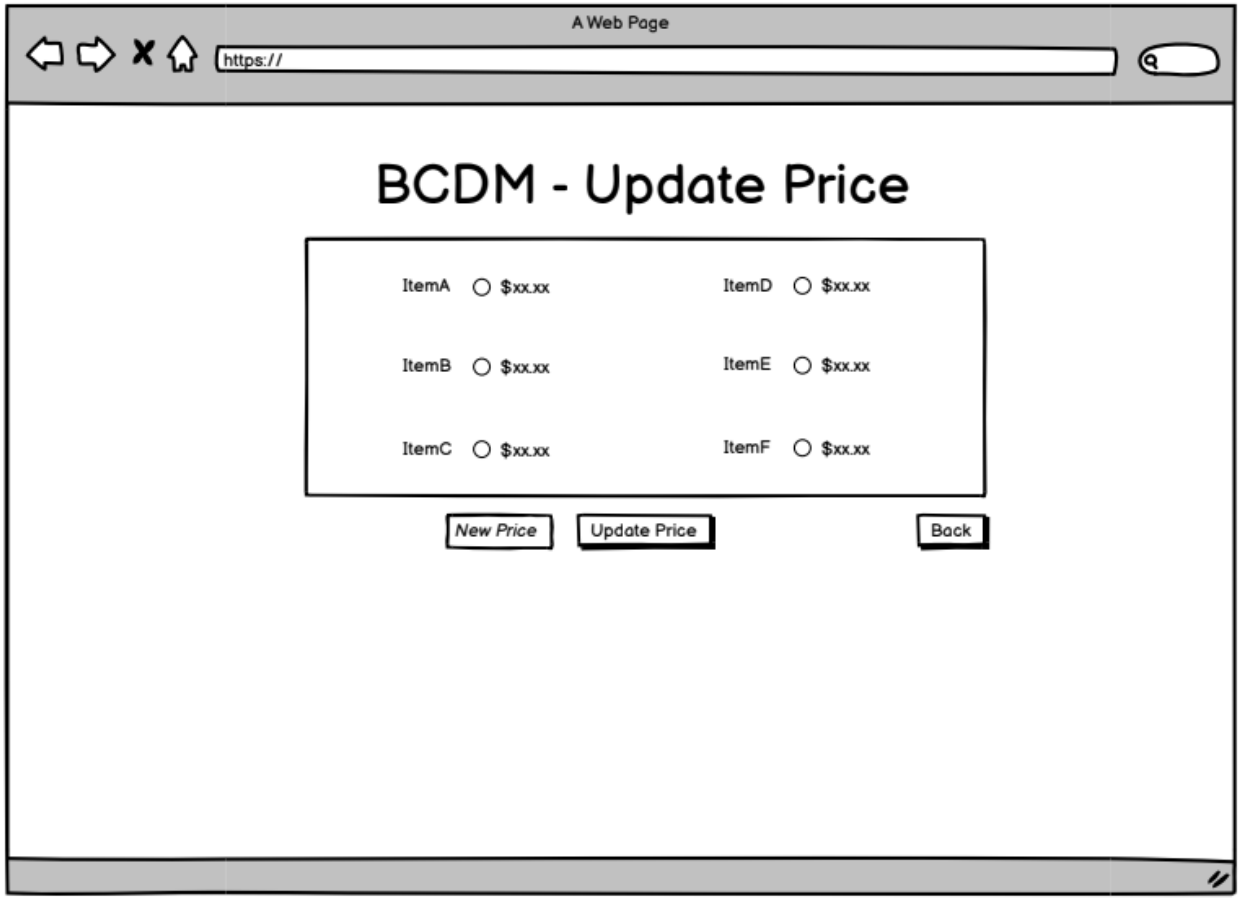

Figure 5: SCREEN_UPDATEPRICE selects an item and has new price field to update a price

Table 4: SCREEN_UPDATEPRICE FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
Food	EIF	2	1	LOW	5
Functions					
Buttons					
Update Price	EI	4	1	LOW	3
				Total	8

SCREEN_CHANGEDISCOUNT

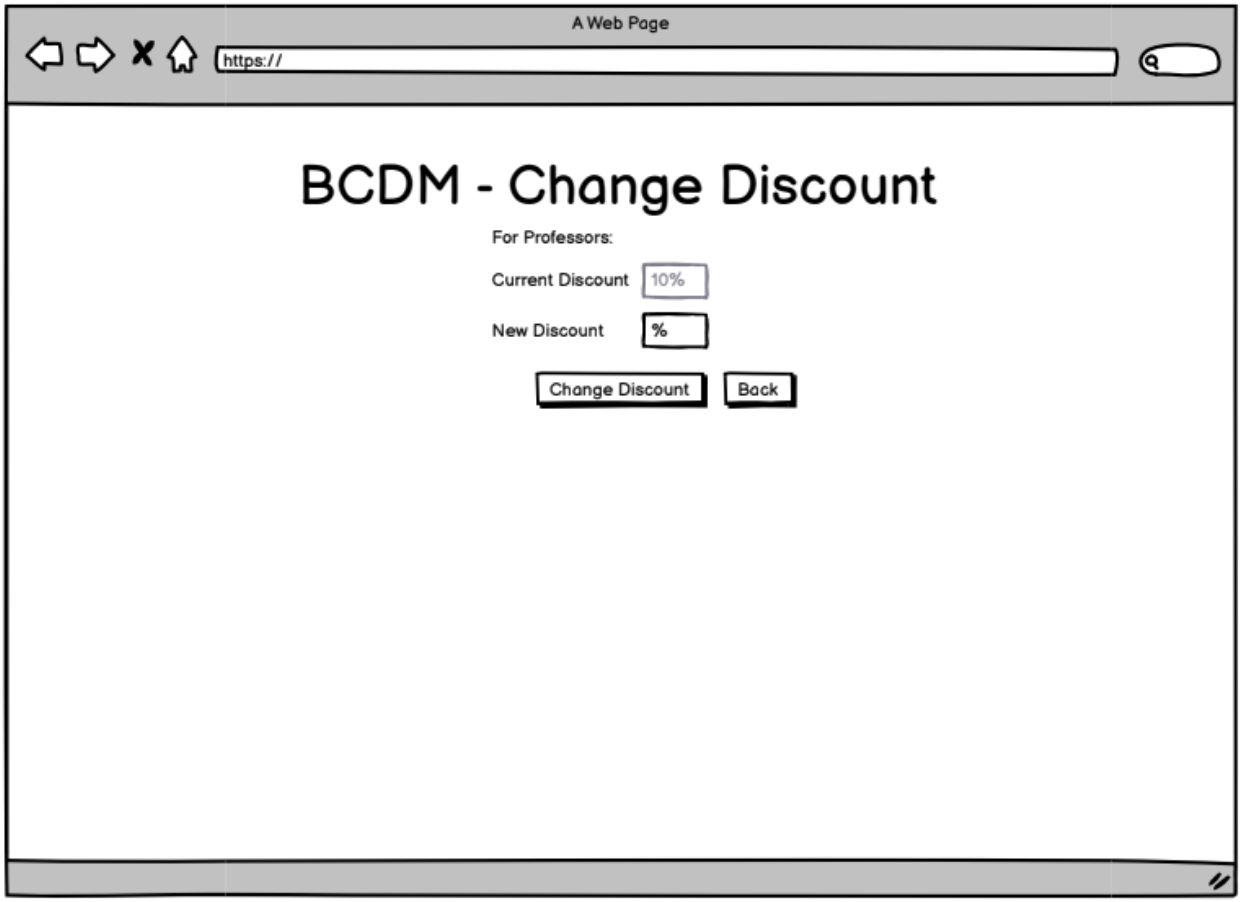

Figure 6: SCREEN_CHANGEDISCOUNT allows admin to change discount scheme

Table 5: SCREEN_CHANGEDISCOUNT FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
Discount	EIF	2	1	LOW	5
Functions					
Buttons					
Change Discount	EI	3	1	LOW	3
				Total	8

SCREEN_PRINTREPORT

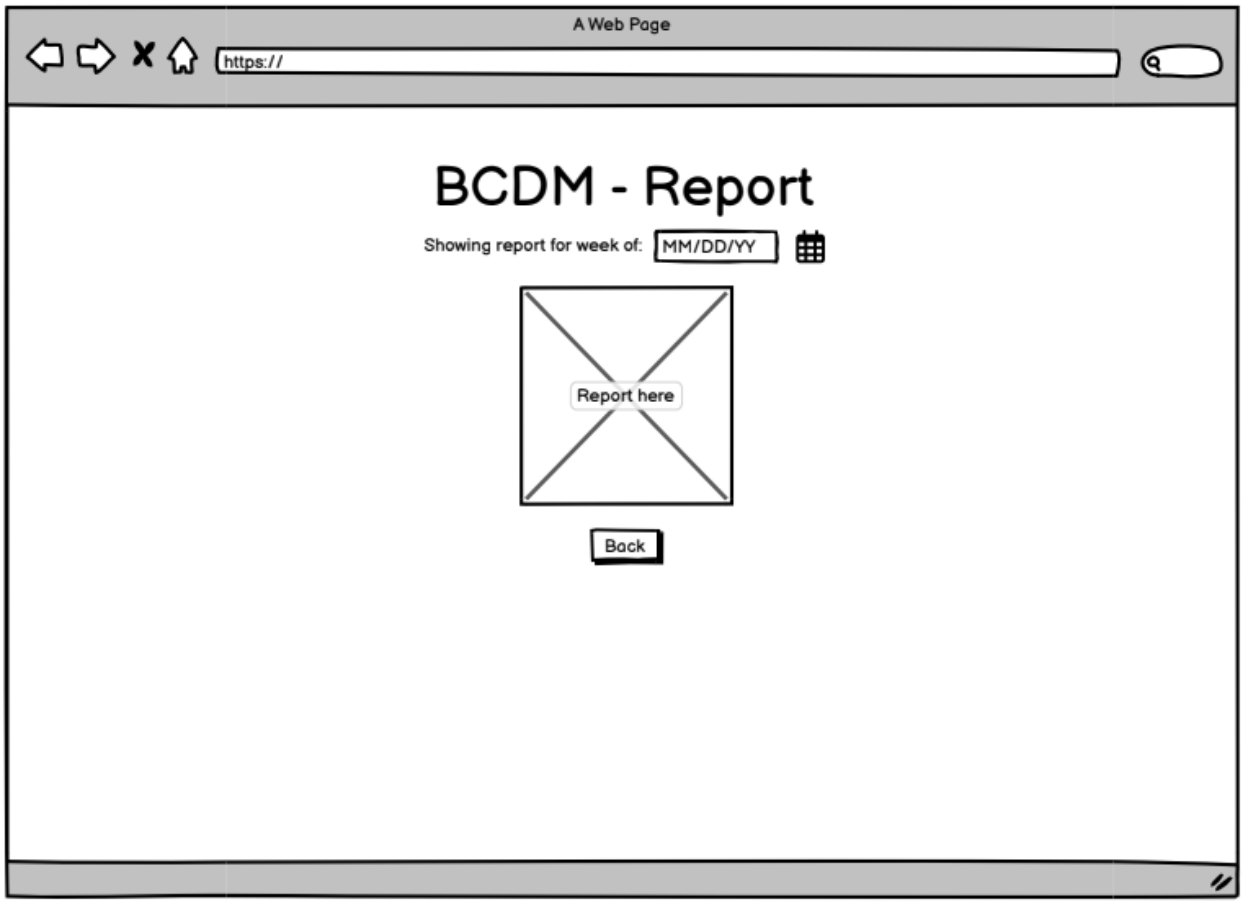

Figure 7: SCREEN_PRINTREPORT displays a report, given a date time frame

Table 6: SCREEN_PRINTREPORT FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Functions					
Buttons					
Change Date	EI	3	1	LOW	3
				Total	3

SCREEN_SHOP

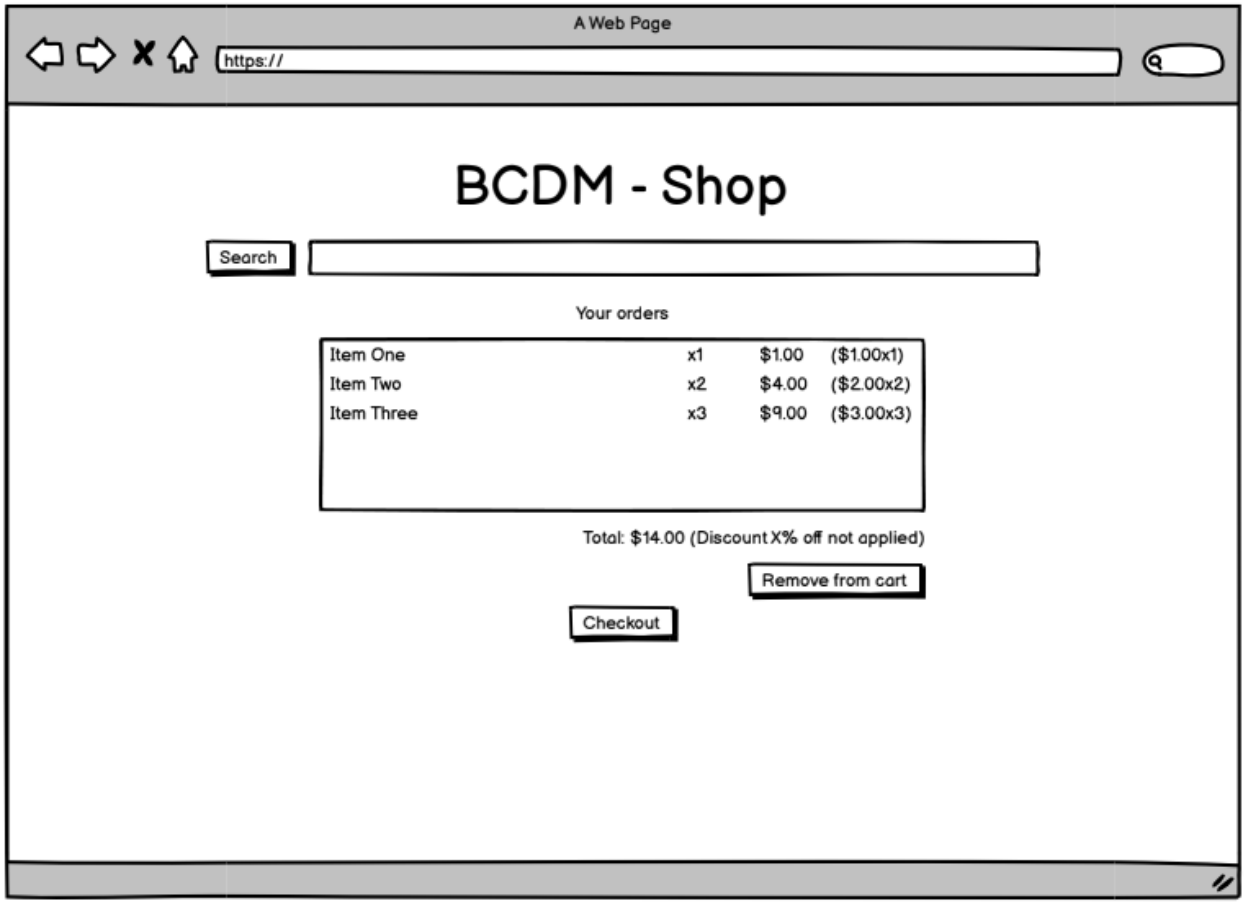

Figure 8: SCREEN_SHOP is the landing screen whenever a Customer logs in successfully

Table 7: SCREEN_SHOP FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
Order	ILF	2	2	LOW	7
Functions					
Buttons					
Search	EQ	2	2	LOW	3
Checkout (order)	EI	3	2	LOW	3
				Total	13

SCREEN_SEARCH

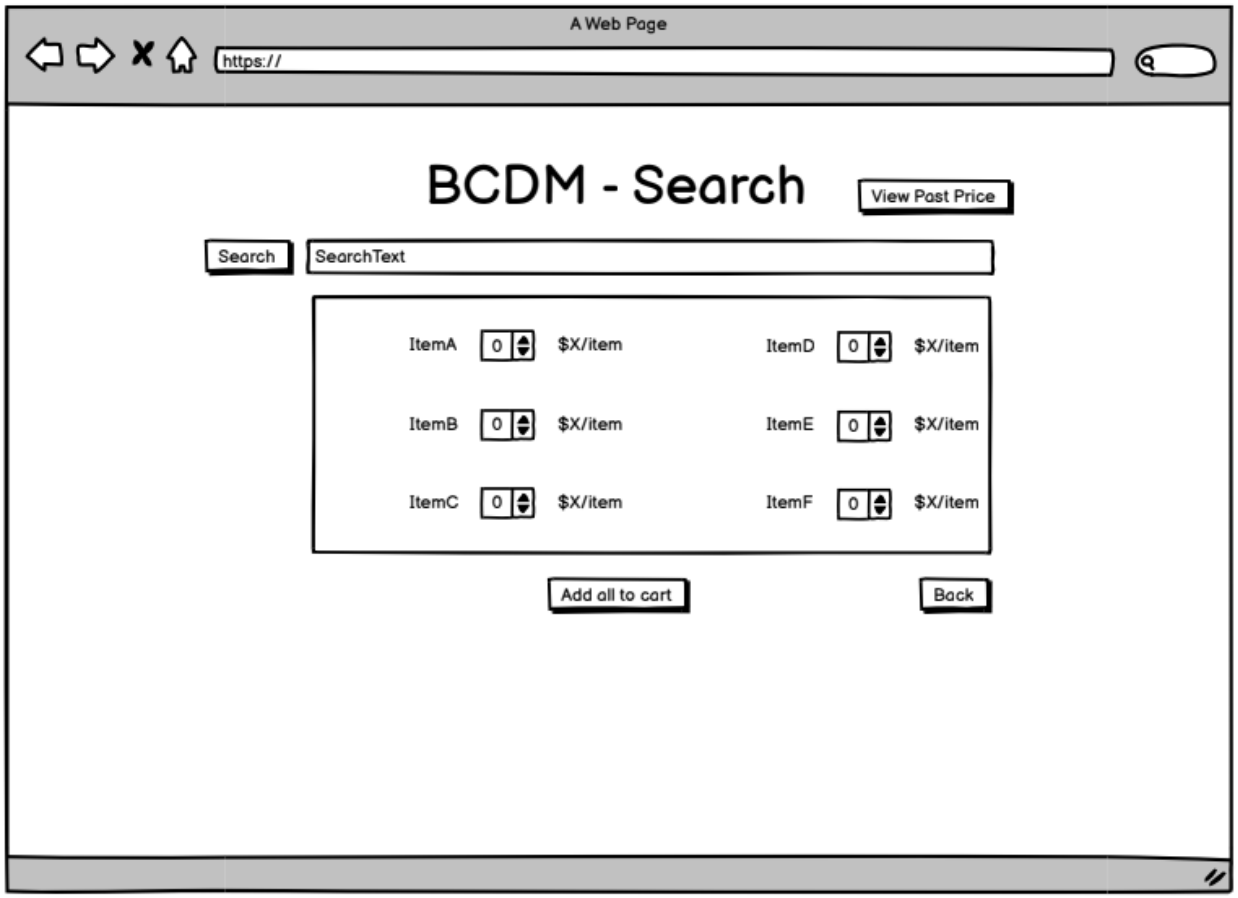

Figure 9: SCREEN_SEARCH employs a self-guided search tool to look up items with the given searchtext

Table 8: SCREEN_SEARCH FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
Food	EIF	2	1	LOW	5
Functions					
Buttons					
Search	EQ	2	2	LOW	3
View Past Price	EI	2	1	LOW	3
				Total	11

SCREEN_PASTPRICE

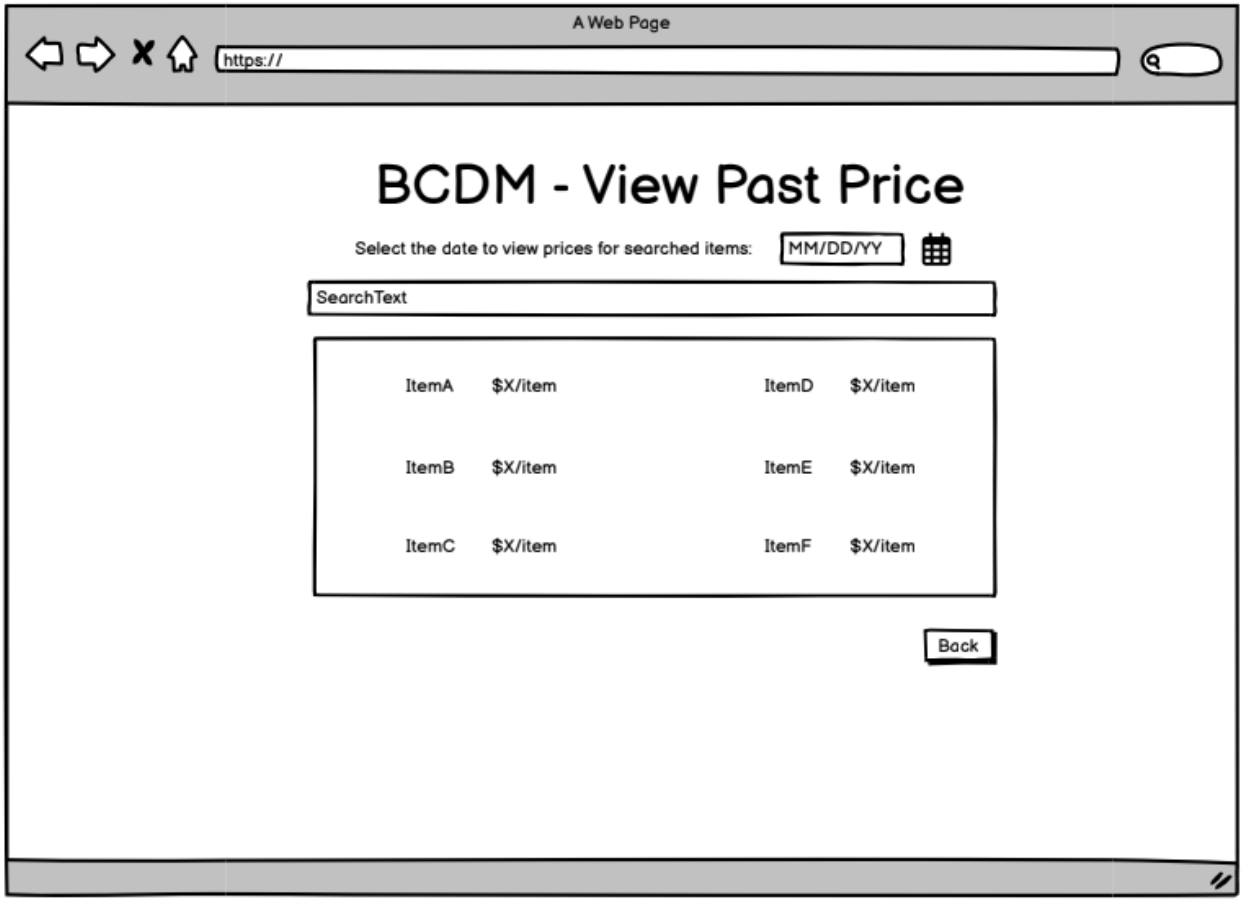

Figure 10: SCREEN_PASTPRICE contains the past price of the searched item, displayed by time

Table 9: SCREEN_PASTPRICE FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
Food	EIF	2	1	LOW	5
Historical Price	EIF	1	1	LOW	5
Functions					
Buttons					
Change Date	EI	3	1	LOW	3
				Total	13

SCREEN_RECEIPT

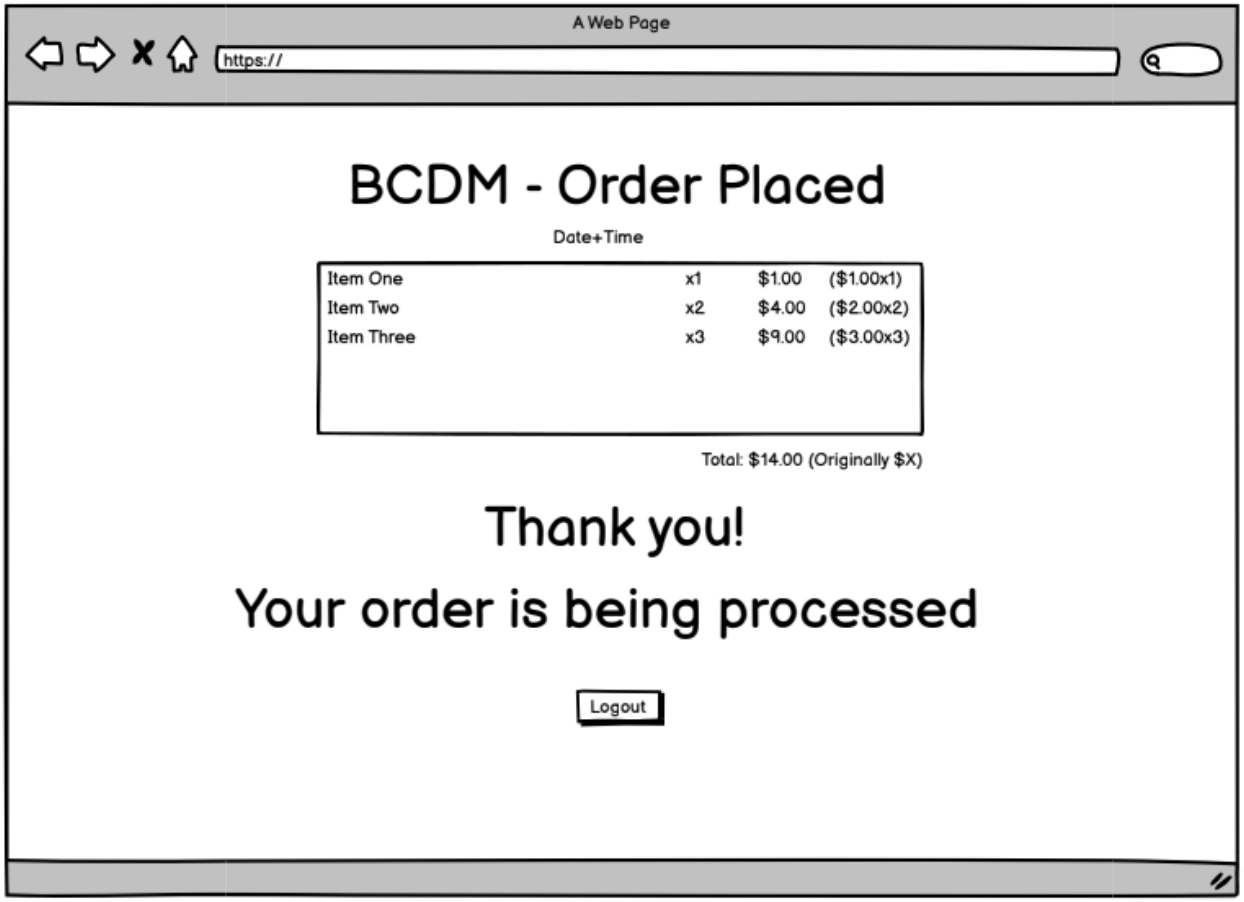

Figure 11: SCREEN_RECEIPT displays the receipt of the order after an order has been placed

Table 10: SCREEN_RECEIPT FP

Elementary Process	Type	DET	RET/FTR	Complex	FP
Files					
Order	ILF	5	2	LOW	7
				Total	7

Total: 8 + 13 + 8 + 8 + 8 + 3 + 13 + 11 + 13 + 7 = 92 FP

8.2 Schedule

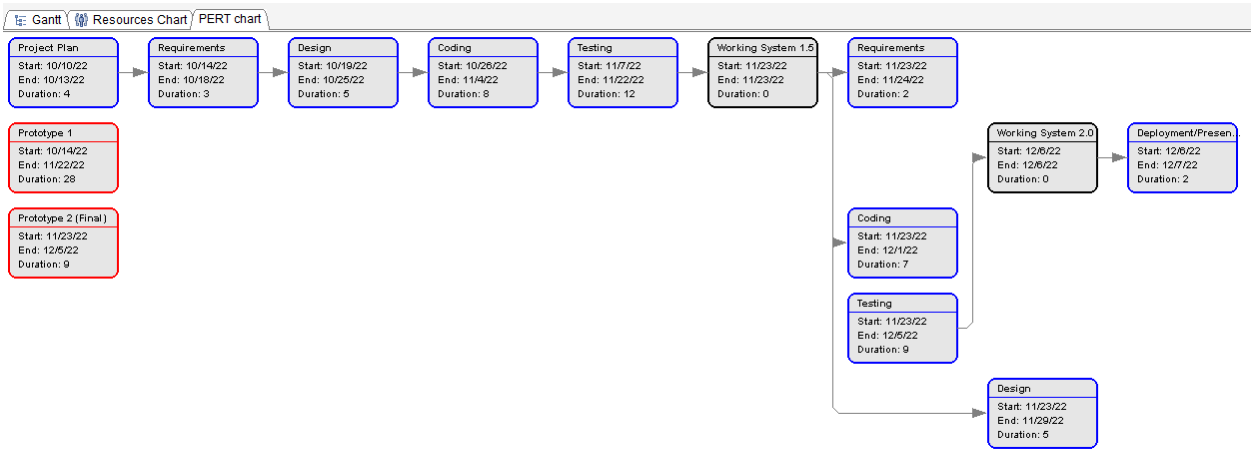

Figure 12: PERT Chart

Bronco Centerpointe Dining Mangement System

Dec 6, 2022

Project manager	Bryan
Project dates	Oct 10, 2022 - Dec 8, 2022
Completion	100%
Tasks	14
Resources	5

Figure 13: Project Information

Bronco Centerpointe Dining Mangement System

Dec 6, 2022

2

Tasks

Name	Begin date	End date
Project Plan	10/10/22	10/13/22
Prototype 1	10/14/22	11/22/22
Requirements	10/14/22	10/18/22
Design	10/19/22	10/25/22
Coding	10/26/22	11/4/22
Testing	11/7/22	11/22/22
Working System 1.5	11/23/22	11/23/22
Prototype 2 (Final)	11/23/22	12/5/22
Requirements	11/23/22	11/24/22
Design	11/23/22	11/29/22
Coding	11/23/22	12/1/22
Testing	11/23/22	12/5/22
Working System 2.0	12/6/22	12/6/22
Deployment/Presentation	12/6/22	12/7/22

Figure 14: Tasks Schedule

Bronco Centerpointe Dining Mangement System

Dec 8, 2022

3

Resources

Name	Default role	Additional Roles
Bryan	project manager	Designer, Analyst, Programmer, Tester
Fiona	Designer	Analyst
Dhruv	Programmer	Tester
Meet	Programmer	Tester
Catharine	Programmer	Tester

Figure 15: Resources Information

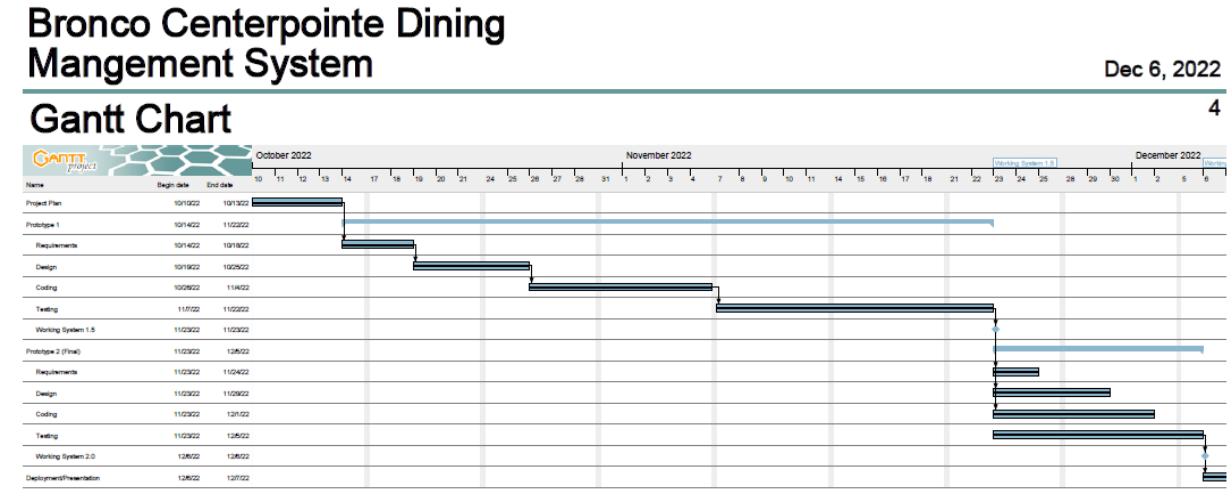

Figure 16: Gantt Chart

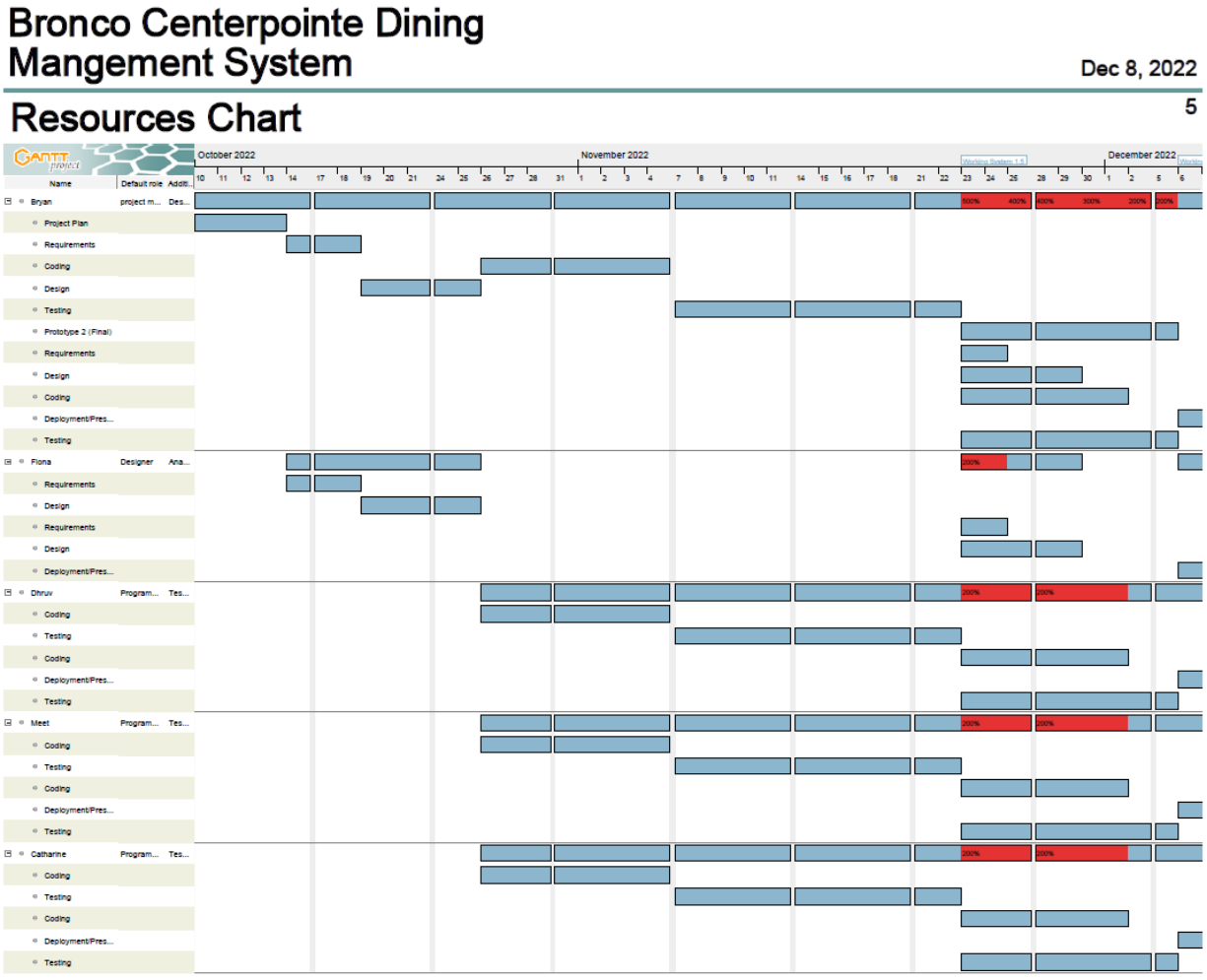

Figure 17: Resources Chart

9. Delivery

Procedures can be limited to customer (student and/or professor) or staff (those with admin credentials)

9.1 Login Procedure

- In order to use the BCDM system, the user must be a student and/or professor of Cal Poly Pomona and have a valid Bronco ID.
- Alternatively, users can log in with admin credentials.

9.2 Registration Procedure

This procedure is made to create new customers, however any user can register a new account

- In order to use the BCDM system, the user must be a student and/or professor of Cal Poly Pomona and have a valid Bronco ID.
- User can register their Bronco ID and password to use the BCDM system, and select their role of student, professor, or select both to be a student professor

9.3 Search Procedure

This procedure is limited to only customers.
For customer user:

- Originally, if the user wants to search for a food item, they would look up a menu and/or ask the cashier if the establishment has the particular food item. In our application implementation, if the user wants to look up a food item (dish or beverage), they would provide a search text for the food and the application would populate the search field with food items with their respective names matching the text.

9.4 Cart Procedures

Originally orders are written on an order paper. In our application implementation, adding to an order paper is synonymous with adding to a virtual shopping cart (cart for short).

9.4.1 Add to Cart Procedures

This procedure is limited to only customers.
For customer user:

- Originally, if the user wants to add a food item to their order, they would indicate the item on the menu to the cashier along with a quantity, and the cashier will manually add it to an order paper. In our application implementation, if the user wants to add a food item to their cart for purchase, they must select a quantity to purchase and add it to their order. There are no worries for inventory management.
- In addition, an individual total will be calculated for individual cost (multiplied by quantity) and displayed next to the food in the cart. The total price of all cart items will be calculated dynamically.

9.4.2 Remove from Cart Procedures

This procedure is limited to only customers.
For customer user:

- Originally, if the user wants to remove a food item from their order, they would indicate the item in the order (cart) to the cashier, and the cashier will manually remove the item from the order paper. In our application implementation, if the user wants to remove a food item from their cart for purchase, they must select the item from their cart and select “remove from cart” selection to remove all instances of the food item from their cart.

9.4.3 Checkout Cart Procedures

This procedure is limited to only customers.
For customer user:

- Originally, if the user wants to place their order after adding food item(s), the cashier will sum up the cost of all items and provide the total for payment. In our application implementation, the user will select “check out cart” and the payment will happen automatically, as it is billed to Cal Poly Pomona’s student account. A receipt will additionally be generated.
- To thank professors for their continued support, we have provided a discount of 10% upon checkout towards any customers who are professors.

9.5 Add New Food Item Procedure

This procedure is limited to only staff.
For staff:

- Originally, if the staff wants to add a new food item to the menu, they would need to add a new field to the physical menu, along with its price. In our application implementation, if the staff wants to add a new food item, they would first need to log in using admin credentials, and select the “add food” option, filling out fields for name and price, syncing with our database.

9.6 Update Food Price Procedure

This procedure is limited to only staff.
For staff:

- Originally, if the staff wants to update a food item’s price in the menu, they would need to remove the old price and manually update with the new price on the menu. In our application implementation, if the staff wants to update a food item’s price in the menu, they would first need to log in using admin credentials, and select the “update price” option, select the food item they would like to update, and enter the new price to update, syncing with our database.

9.7 Update Discount Scheme Procedure

This procedure is limited to only staff.
For staff:

- Originally, there was no discount scheme. In our application implementation, if the staff wants to update the default discount scheme for all professors, they would first need to log in using admin

credentials, and select “update discount” option, and enter the new discount percentage, syncing with our database.

9.8 Printing Receipt Procedure

This procedure is limited to only customers.
For customer user:

- Originally, if the cashier wants to print a receipt for the customer, they would need to print 2 copies of the receipt, one for the customer and one for the restaurant to store the receipt in an excel file. In our application implementation, if the user wants to check out their cart, a receipt will be generated, and a copy of the receipt will be synced with our database.

9.9 Display Historical Price Procedure

This procedure is limited to only customers.
For customer user:

- Originally, there was no procedure to display historical prices. In our application implementation, the user can click the “view past price” button next to the searched food items to display a page with past prices with the dates associated.

9.10 Intelligent Report Procedure

This procedure is limited to only staff.
For staff:

- Staff can get an intelligent report of consolidated revenue information by customer and period.

Configuration Plan

1. Introduction

1.1 Conventions

- Initially, each developer will work on “dev” branch
- After group consensus, all work on current branch will be merged over to main branch
- Before merging over to the main branch, “dev” branch must be thoroughly tested for all metrics and requirements during the current prototype development cycle.

1.2 Terms

REQ - Requirements
BLD - Builds
MN - Manuals
DEV - Development
PROD - Production

DEVS - Development Server
BETAS - Beta Server
PRODS - Production Server

1.3 Abbreviations Identify Configuration Manager(s)

BT - Bryan Trinh
FG - Fanghua Gu
DP - Dhruv Patel
MB - Meet Butani
CP - Catherine Pumar

2. Software Configuration Management

List of process activities

- Have a group meeting at least once a week (Monday) to discuss what we are going to do and update on the current process
- Identify high-level goals
- Identify primary and secondary actors
- Identify use case diagrams and specifications
- Identify requirements
- Identify configuration items
- Identify responsibility for each configuration item
- Create repository
- Commit

- Generate versions and baselines

3. Identify Configuration Items

Configuration items that will be produced throughout the project life cycle

- Project plan
- Configuration plan
- Requirements specification
- Design specification
- Paper-based prototype
- Test plan
- Quality plan

4. Identify Responsible for each Configuration Item

Table 4: Configuration Item Responsibility

Configuration Item	Responsible Person
Project Plan	BT
Configuration Plan	BT
Requirement Specification	BT, FG
Design Specification	BT, FG
Paper-based Prototype	BT
Implementation	BT, FG, DP, MB, CP
Test Plan	BT, FG, DP, MB, CP

5. Tools

Version control tools will be used

- GitHub: Control versions of the app
- Discord: Primary communication platform for personnels
- IDE: Visual Studio, VS Code, Eclipse

6. Environment and Infrastructure

- There are going to be 3 different servers: dev, beta, production: dev server is for developing, beta server is for internal testing, production server is for the releasing
- All the manual docs are stored on the dev server

7. Configuration Policy

Nomenclature for configuration items such as <project> - <type-artifact> - <name>, where <project> is the name of the project, <type-artifact> is the configuration item, <name> is the name of the document

7.1 Project Plan

BCDMS-ProjectPlan-ProcessModel
BCDMS-ProjectPlan-Organization
BCDMS-ProjectPlan-MethodsAndTechniques
BCDMS-ProjectPlan-Standard
BCDMS-ProjectPlan-WorkPackages
BCDMS-ProjectPlan-Resources
BCDMS-ProjectPlan-Delivery

7.2 Configuration Plan

BCDMS-ConfigurationPlan-Conventions
BCDMS-ConfigurationPlan-Terms
BCDMS-ConfigurationPlan-SoftwareConfigurationManagement
BCDMS-ConfigurationPlan-ConfigurationItems
BCDMS-ConfigurationPlan-Tool
BCDMS-ConfigurationPlan-EnvironmentAndInfrastructure
BCDMS-ConfigurationPlan-ConfigurationAndChangeControl
BCDMS-ConfigurationPlan-SystemBuilding
BCDMS-ConfigurationPlan-ReleaseManagement
BCDMS-ConfigurationPlan-ContingencyPlan

7.3 Requirements Specification

BCDMS-RequirementsSpecification-High-Level Goals
BCDMS-RequirementsSpecification-PrimaryAndScondaryActors
BCDMS-RequirementsSpecification-UseCaseDiagram
BCDMS-RequirementsSpecification-UseCaseSpecification
BCDMS-RequirementsSpecification-Assumptions
BCDMS-RequirementsSpecification-DomainProperties
BCDMS-RequirementsSpecification-FunctionalRequirements
BCDMS-RequirementsSpecification-NonFunctionalRequirements
BCDMS-RequirementsSpecification-TraceabilityMatrix

7.4 Design Specification

BCDMS-DesignSpecification-ERDiagram
BCDMS-DesignSpecification-LogicalModel
BCDMS-DesignSpecification-SequenceDiagram
BCDMS-DesignSpecification-StateMachineDiagram

BCDMS-DesignSpecification-SystemArchitectureViewAndStylePattern

7.5 Paper-based Prototype

BCDMS-PaperBasedPrototype-TechnicalDescription

7.6 Test Plan

BCDMS-TestPlan-Scenarios
BCDMS-TestPlan-UnitTests
BCDMS-TestPlan-IntegrationTests
BCDMS-TestPlan-SystemAcceptanceTests

8. Document Version

Definition of a version numbering scheme for configuration items and baselines
BCDMS.major.minor[.fix][-configuration]
BCDMS - Bronco Centerpointe Dining Management System
.major - Main Version with 5 or more new features
.minor - Minor Version with less than 5 new features or updating features
[.fix] - bug fixing
[-configuration] - configuration label

9. Evolution of Version Number

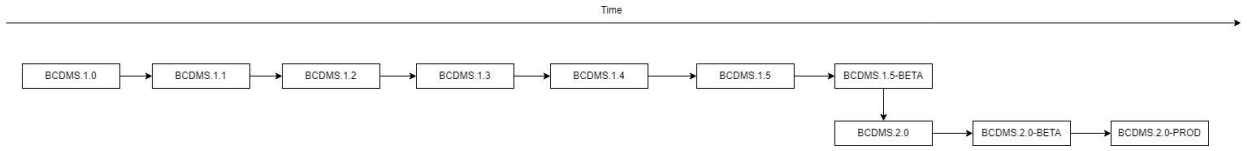

Figure 1: Version Evolution

BCDMS.1.0
    Created sample backend code skeleton
BCDMS.1.1
    Implemented sample login via Bronco ID and default password (Customer/Staff)
    Implemented GUI for Login, and Admin and Customer Shop Portal
BCDMS.1.2
    Implemented Customer registration, and GUI
    Implemented create new food item in database
    Implemented update food item price in database
BCDMS.1.3
    Implemented adding items to cart, checking out, and GUI
BCDMS.1.4
    Implemented receipt, and search feature
BCDMS.1.5

Implemented discount scheme
BCDMS.1.5-BETA
Push version BCDMS.1.5 to beta server and test
BCDMS.2.0
Improved GUI, created desktop application
BCDMS.2.0-BETA
Push version BCDMS.2.0 to beta server and test
BCDMS.2.0-PROD
Push version BCDMS.2.0-BETA to production server to deploy

10. Configuration and Change Control

We are going to use Discord to update the progress of the project, BT is the one who manages it. There are different channels to keep track of work.

In the General text channel, we have simple day to day communication and quick online updates in a Scrum-like methodology. Take note of the pinned message pop-up in the upper right corner in Figure 1. Messages can be pinned on the bulletin board for ease of access, in this case the Github repository. All text channels will have their own bulletin board instance.

In the Meeting text channel, BT would book group study rooms in the library for in person meetings, usually every Monday at 4pm. Meetings hosted at any other time will have their information updated in there as well.

In the Links text channel, there are links of whatever relevant information discussed in meetings will be posted there.

In the Backend text channel, there will be information and communication pertaining to the backend developers. Likewise the Frontend text channel also has the same feature. Any tasks can be pinned to the bulletin board, with the assignee, and once finished, it will be unpinned.

The General voice channel allows members to join a call, with a screen sharing function similar to Zoom, to provide verbal communication, and the team will occasionally have meetings on this voice channel through Discord.

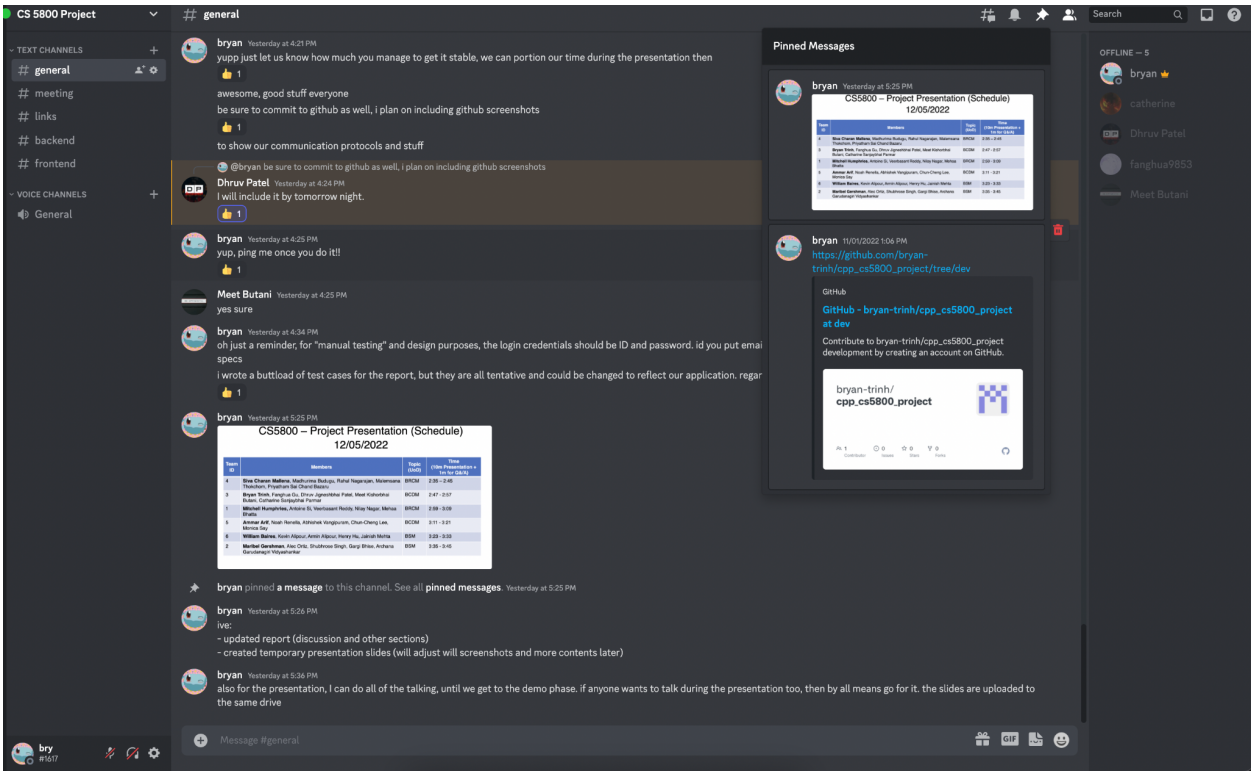

Figure 1: Screenshot of our Discord server

11. System Building

1. Create repository
2. Configure Java development environment
3. Configure MySQL environment and manage data to match with Bronco Centerpointe Dining Management system's database
4. Follow use cases and implement those functionalities
5. Follow test cases to validate and verify requirements
6. Compile the project and then write documentations on how to use it

12. Release Management

Strategy to system delivery, client system version management

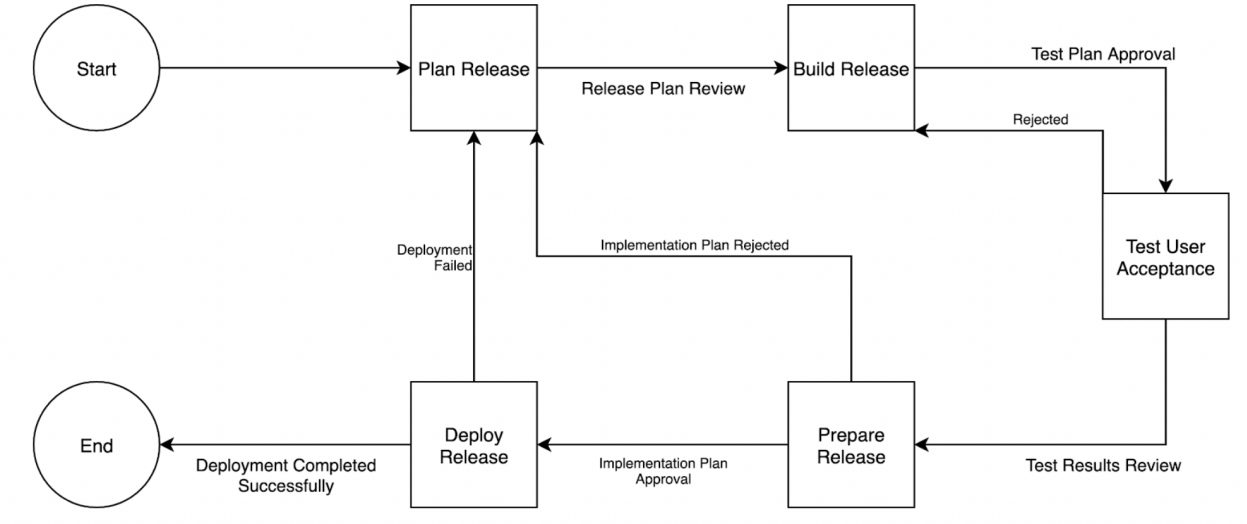

Figure 2: Release Management

13. Contingency Plan

Procedures to be executed in case of data loss or compromise:

- Separate the development database and production database
- Backup the production database every month
- Before updating new version to production server, backup the production database
- Try to make clear and nice description of comments and pull requests, so developers can easily find the cause of data loss or compromise
- Comment all changes and push to own branch whenever a developer makes substantial changes
- Make logs for every database update and save logs separate from database
- When data loss or compromise occurs, developers can try to rollback to the latest working backup and update the backup by logs

Requirement Specification

1. High-Level Goals (Unique Identifiers)

G1: Deliver a sufficient way to use the dining management system to make orders for student, professors, and student professors
G2: Deliver a sufficient way for staff to modify prices, add food items, and adjust discount scheme
G3: Provide efficient methods to manage order activity

2. Primary and Secondary Actors

Primary actors: customer (student, professor, student professor), staff
Secondary actors: dining management system

3. Use Case Diagram(s)

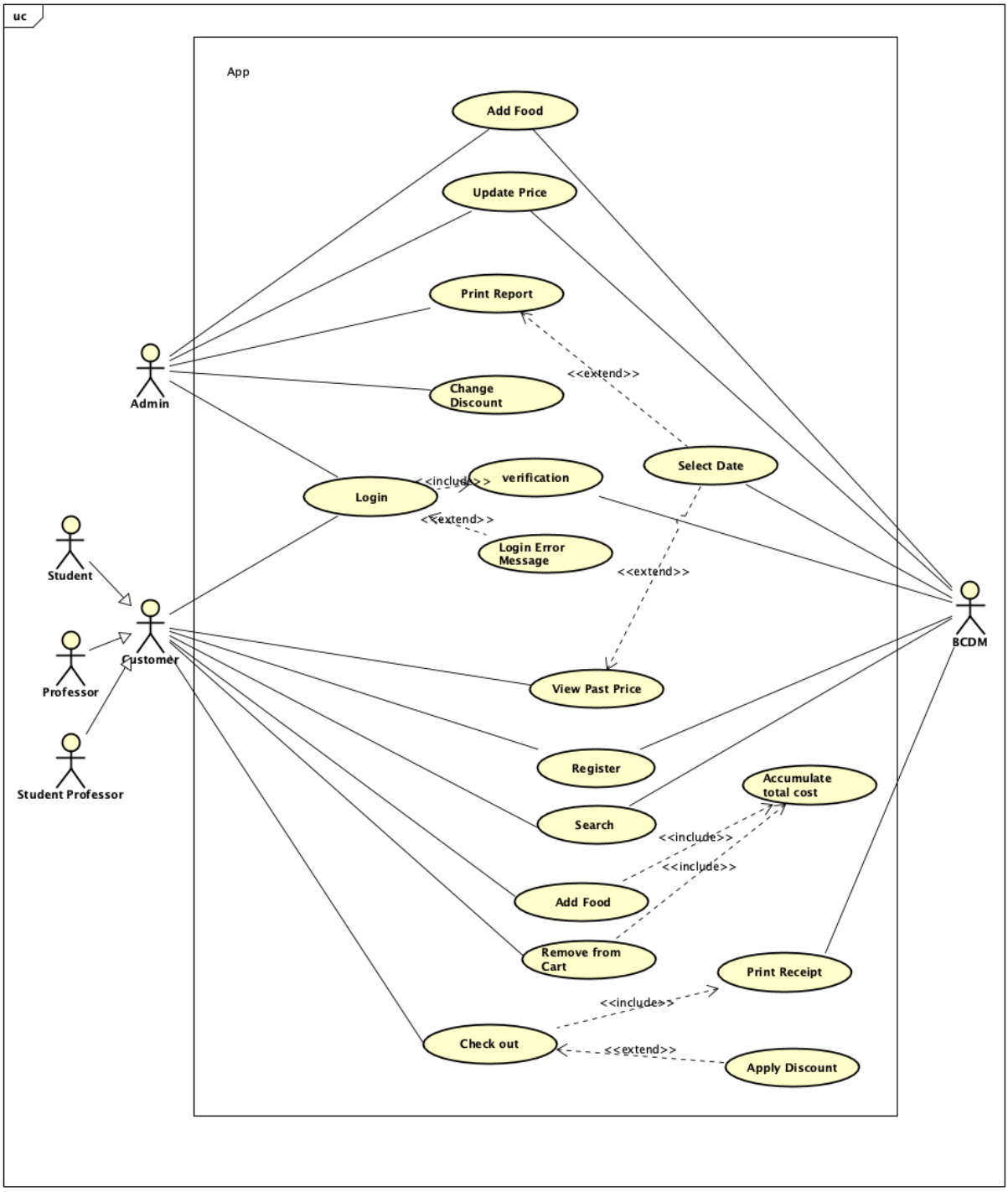

Figure 1: Use case diagram

4. Use Case Specification(s)

BCDM: Bronco Centerpoint Dining Management App
Customer: student, professor, student professor
Staff: admin

Table 1: Login Use Case

Use Case: Login
Primary Actor: customer, staff
Secondary Actor [optional]: BCDM
Precondition: The application is up and running
Postcondition: customer/staff successfully logged in
Main Success Scenario: 1.Customer/Staff accesses the login page 2.Customer/Staff informs his/her credentials 3.Customer/Staff clicks the button “Login” 4.Verify Password 5.An authentication success message is displayed 6.Customer/Staff is granted access to the application
Alternative Course [optional]:
Exception Course [optional]: 5a. Extend UC Display Login Error /An authentication failure message is displayed. Go back to step 2.

Table 2: Register Use Case

Use Case: Register
Primary Actor: customer
Secondary Actor [optional]:BCDM
Precondition: The application is up and running
Postcondition: customer successfully registered
Main Success Scenario: 1. Customer clicks the register button on the login page 2. Customer input his/her information to register 3. The registration page switches to the login page
Alternative Course [optional]:

Exception Course [optional]:

Table 3: Add Food Use Case

Use Case:Add Food
Primary Actor: staff
Secondary Actor [optional]: BCDM
Precondition: User is authenticated
Postcondition:Food is added successfully
Main Success Scenario: 1.staff clicks on the add food option in the application menu 2.staff inputs the name of the food and price on the text field 3.staff clicks the add food button.
Alternative Course [optional]:
Exception Course [optional]:

Table 4: Update Price Use Case

Use Case:Update price
Primary Actor: staff
Secondary Actor [optional]: BCDM
Precondition: User is authenticated
Postcondition: Price is updated successfully
Main Success Scenario: 1.staff clicks on the update price option in the application menu 2.staff chooses the desired item and input the new price on the textfield. 3.staff clicks the update price button.
Alternative Course [optional]:
Exception Course [optional]:

Table 5: Print Report Use Case

Use Case: Print report
Primary Actor: staff
Secondary Actor [optional]: BCDM

Precondition: User is authenticated
Postcondition:Report is print successfully
Main Success Scenario: 1.staff clicks on the print report option in the application menu. 2.staff selects the desired date. 3.The corresponding report is printed on the middle of the screen.
Alternative Course [optional]:
Exception Course [optional]: 3a. Go back to step 2 to select a different date for the report.

Table 6: Search Use Case

Use Case: Search
Primary Actor: customer
Secondary Actor [optional]: BCDM
Precondition: User is authenticated
Postcondition:Searched items are displayed on the screen
Main Success Scenario: 1.Customer inputs the name of the desired item on the textfield. 2.Customer clicks on the search button on the shop page 3.The searched items are displayed on the screen
Alternative Course [optional]:
Exception Course [optional]:

Table 7: Remove from Cart Use Case

Use Case: Remove from cart
Primary Actor: customer
Secondary Actor [optional]: BCDM
Precondition: User is authenticated
Postcondition: Food is removed from the cart successfully
Main Success Scenario: 1.Customer chooses the unwanted item 2. Customer clicks the remove from cart button on the shop page

Alternative Course [optional]:
Exception Course [optional]:

Table 8: Checkout Use Case

Use Case: Checkout
Primary Actor: customer
Secondary Actor [optional]: BCDM
Precondition: User is authenticated
Postcondition: Checkout is completed
Main Success Scenario: 1.Customer clicks the checkout button on the shop page 2.The shop page switches to receipt page
Alternative Course [optional]:
Exception Course [optional]: 1a: Customer can input the code in add discount textfield

Table 9: Add to Cart Use Case

Use Case: Add to cart
Primary Actor: customer
Secondary Actor [optional]:
Precondition: User is authenticated
Postcondition: Food is added to the cart successfully
Main Success Scenario: 1.Customer inputs the name of the desired item on the textfield. 2.Customer clicks on the search button on the shop page 3.The searched items are displayed on the screen 4.Customer selects the quality of each item. 5.Customer clicks add all to cart button. 6.The search page switches back to the shop page 7.The total accumulated prcied is displayed
Alternative Course [optional]:
Exception Course [optional]:

Table 10: View Past Price Use Case

Use Case: View Past Price
Primary Actor: customer
Secondary Actor [optional]: BCDM
Precondition: User is authenticated
Postcondition: The past price is displayed
Main Success Scenario: 1.Customer inputs the name of the desired item on the textfield. 2.Customer clicks on the search button on the shop page 3.The searched items are displayed on the screen 4.Customer clicks the view past price on the upper right conner 5.The shop page switches to past price page 6.The past price page contains the past price of the searched item, displayed by time.
Alternative Course [optional]:
Exception Course [optional]:

5. Assumptions

5.1 Credentials

- User should be a member of Cal Poly Pomona, with some Bronco ID
- User should be one of the following: student, professor, student professor, or staff (those with admin credentials)

5.2 Security

- No user can place an order without logging in with some Bronco ID
- Only staff (admin) can print and view reports
- Only customers can view past prices
    - Nonexisting past price will be labeled as N/A

6. Domain Properties

- A food item can be ordered in unlimited quantity by anyone
- After a customer places an order online, the order will be marked as online-complete the next time they log into the desktop application
- An order can not be placed without a customer
- An admin can only add new food items, update food prices, and print reports
- A customer can only place orders and view past prices

7. Functional Requirements

REQ_01- Customer Registration

Rationale: a customer is defined by a BroncoID, and has the attribute First Name Last Name (both combined to Name), DOB, Phone, Address(Street, Number, Zipcode, City, State)

REQ_02- Order Registration

Rationale: an order is defined by an ID, and has the attribute Date, Time, Customer, multiple ProductList(Food, Quantity), derived attribute Total Price.

REQ_03- Food registration

Rationale: a food is defined by an ID, and has the attribute Name.

REQ_04-Historical Price

Rationale: a historical price is defined by Date and Price.

REQ_05- Relationship between Customer and Order

Rationale: each order must belong to a single customer, but a customer can have multiple orders. This relationship has an attribute Price Discount to inform the customer the price after applying discount.

REQ_06- Relationship between Food and Order

Rationale: an order can contain multiple food and food can be selected in one or more orders.

REQ_07- Relationship between Food and Historical Price

Rationale: a food can have multiple prices but a historical price must belong to one or more food.

REQ_06- Relationship between Historical Price and Order

Rationale: a historical price can refer to multiple orders, but an order must refer to one or more historical prices.

8. Non-functional Requirements

8.1 Product Requirements

NREQ_PR_1 - Usability Requirement
Rationale:

- The system shall allow the users to access the system from the Bronco Centerpointe Dining Management System User interface. The system uses a Swing GUI client as an interface. Since all users are familiar with the usage of UI, no more training is required
- A user should be a member of Cal Poly Pomona and has an active Bronco ID which can be used to register a customer account
- The system is user friendly and online help makes using the system easy
- The system shall generate food item results which match the search terms when users conduct the self-guided search
- The system shall generate an intelligent report when staff (admin) send a request to the system
- The users can add any amount of food items to their order (which could be duplicate). All food items added in the order must have a corresponding quantity
- The system shall provide a receipt to the user upon checking out their order
- The users can view past prices for each searched item, for a selected date

NREQ_PR_2 - Availability Requirement
Rationale:

- The Bronco Centerpointe Dining Management System is available for any registered user and admin, and is used 24 hours for 365 days a year. The system should be operation 24 hours a day for 7 days a week
- The information is refreshed at regular intervals, pending updates
- The format for food item queries shall be accessible to any customer, and those queries shall take less than 3 seconds

NREQ_PR_3 - Reliability Requirement
Rationale:

- The Bronco Centerpointe Dining Management System should accurately provide real time food item search. The system has to be 100% reliable and shall provide 100% access reliability
- Mean time to repair - even if the BCDM system fails ,the system will be recovered back up in 3 hours or less
- Restarting the BCDM system is acceptable when a failure event occurs
- Failure rate less than or equal to 3%
- The system will automatically mark online-pending orders to online-complete upon the respective customer logging into the desktop application

NREQ_PR_4 - Performance Requirement
Rationale:

- The system shall respond to the user in less than 5 seconds from the time all kinds of requests such as food item search, customer registration, intelligent report generation, and etc are submitted. The responses to view results shall take no more than 6 seconds to appear on the UI page. The search result shall be complete and effective, pending user's spelling
- The system shall support 10,000 concurrent users and should be able to handle a large amount of data. Thus it shall accommodate a high number of food items and users without fault
- The system shall handle expected and unexpected errors in ways that prevent loss of information and downtime period

- The system has strong traceability to customer's orders, especially when ordering online

NREQ_PR_5 - Supportability (including maintainability and portability)
Rationale:

- Any changes (new user registration, database changes) must be verified at least once per day by admin
- In the future, the system shall be ported in 64-bit Linux OS and Android OS, pending longevity of project at the next annual shareholder's meeting
- The system shall be maintained by staff and shall be updated at least once every 3 months

8.2 Organizational Requirements

NREQ_OR_1 - Implementation Requirement
Rationale:

- Deploying and commissioning the system shall take 3 months or less, and the implementation involves BCDM system installation, maintenance, staff and user training, easy to use search and tracking policies, strong user management, intuitive navigation, and extensive administrative permission for staff accounts

NREQ_OR_2 - Software Constraints
Rationale:

- The quality of the database is maintained in such a way that it can be very user friendly to all the users of the database, including staff. The BCDM database shall be MySQL
- To develop the web server of the BCDM system, we will use Java and Java Swing GUI to design the UI pages. Other related software associated with the BCDM system shall be written in Java, to comply with BCDM's policy

NREQ_OR_3 - Hardware Constraints
Rationale:

- the system requires a database in order to store persistent data. The database should have backup capabilities

8.3 External Requirements

NREQ_ER_1 - Security Requirements
Rationale:

- The databases may crash at any time due to virus or OS failures. Therefore the database shall be backed up by staff and it shall be recovered within thirty minutes when the failure happened.
- There are different categories of users, namely staff, customers (student, professors, student professors). Depending on the category of user, the access rights are different and the user will be able to access category-exclusive privileges. All staff (admin) will be able to modify the food item price and add new food items, in addition to viewing generated reports. All customers only have the right to retrieve the information about the database
- All system data must be backed up every 36 hours and the backup copies stored in a secure location which is not in the same building as the system

NREQ_ER_2 - Legal Requirement
Rationale:

- User information shall be protected. The developmental process to be used must be explicitly defined and must be conformant with ISO 9000 standards

9. Traceability Matrix

Table 11: Traceability Matrix

Requirement	Design	Implementation	Test Case
REQ_1	Staff	SCREEN_LOGIN SCREEN_ADMIN	TS_01
REQ_2	Customer	SCREEN_REGISTER SCREEN_LOGIN	TS_01 TS_02 TS_03 UT_01 UT_02 UT_03 UT_04 UT_05 UT_06 IT_01 IT_02 IT_03 IT_04 IT_05 ST_01 ST_02 ST_03
REQ_3	Search	SCREEN_SHOP SCREEN_SEARCH	TS_02 IT_04 ST_03 UT_03 UT_04
REQ_4	Order	SCREEN_SEARCH SCREEN_RECEIPT	
REQ_5	Checkout	SCREEN_SHOP SCREEN_RECEIPT	TS_03 IT_05 UT_05
REQ_6	Historical Price	SCREEN_PASTPRICE SCREEN_UPDATEPRICE	

Design

1. Entity Relationship Diagram (conceptual model)

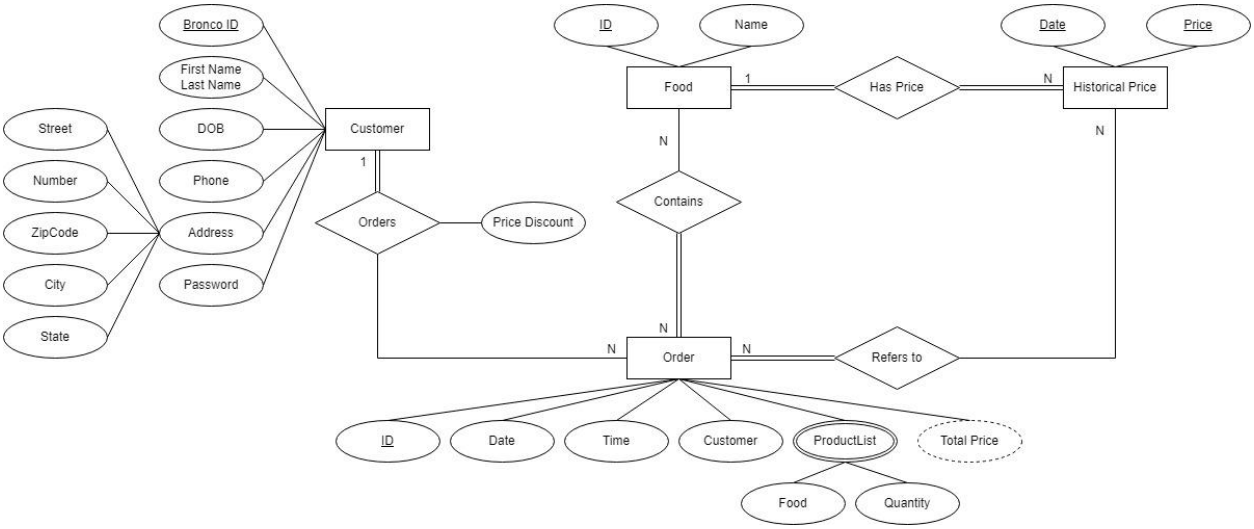

Figure 1: ER Diagram (conceptual model)

2. Logical Model

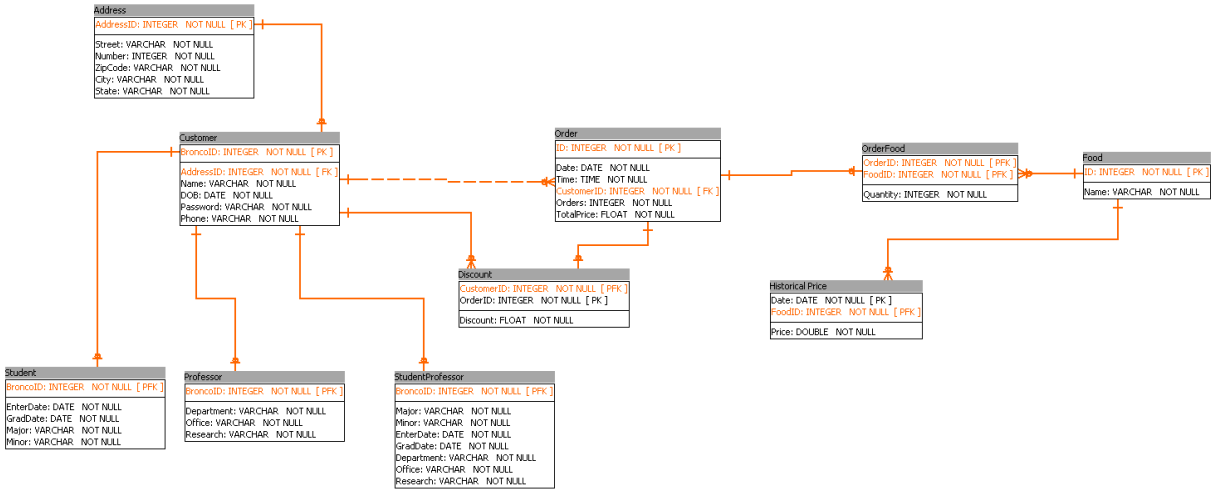

Figure 2: Logical Model

3. Class Diagram

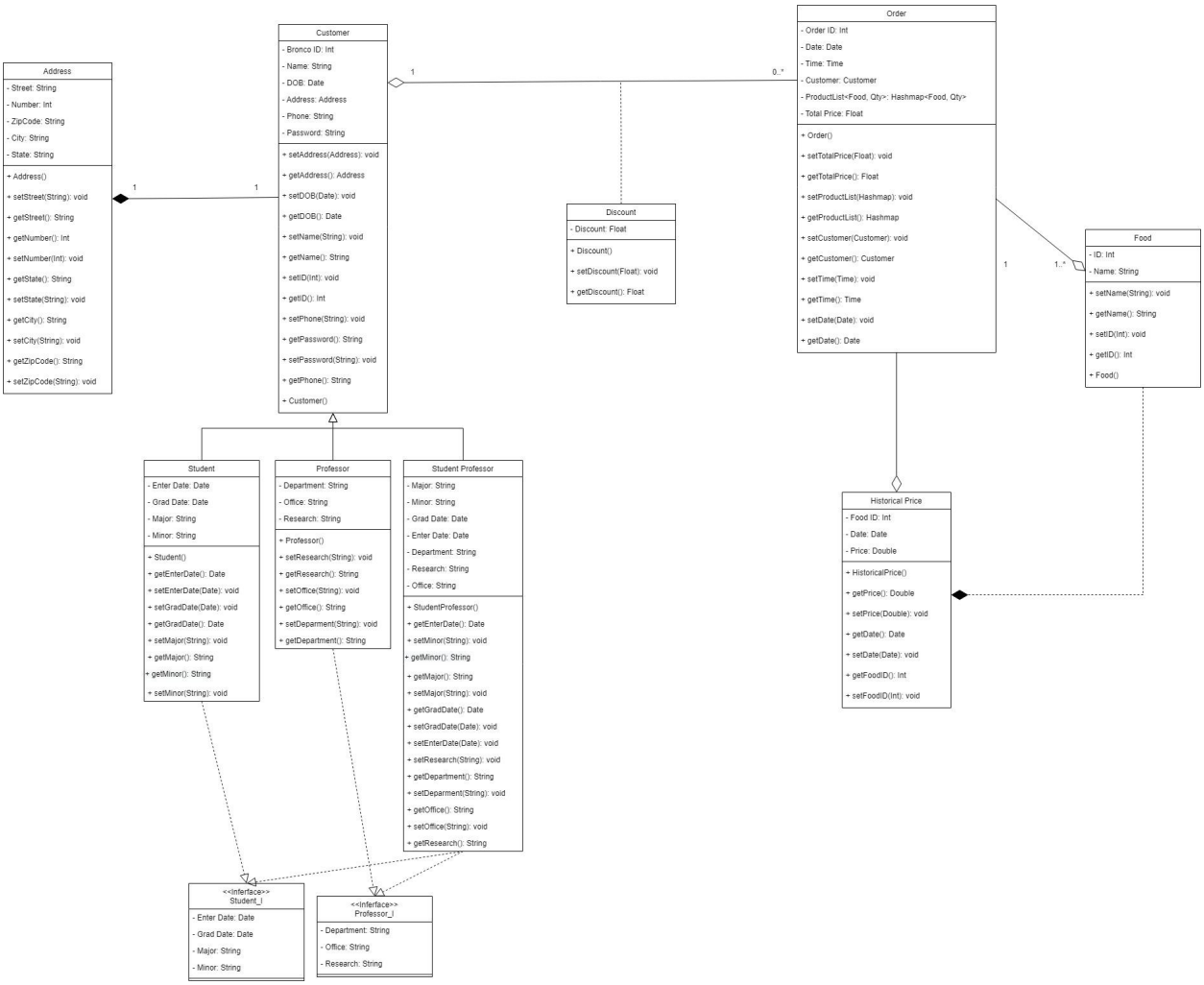

Figure 3: UML Class Diagram

4. Sequence Diagram

4.1 Checkout Cart

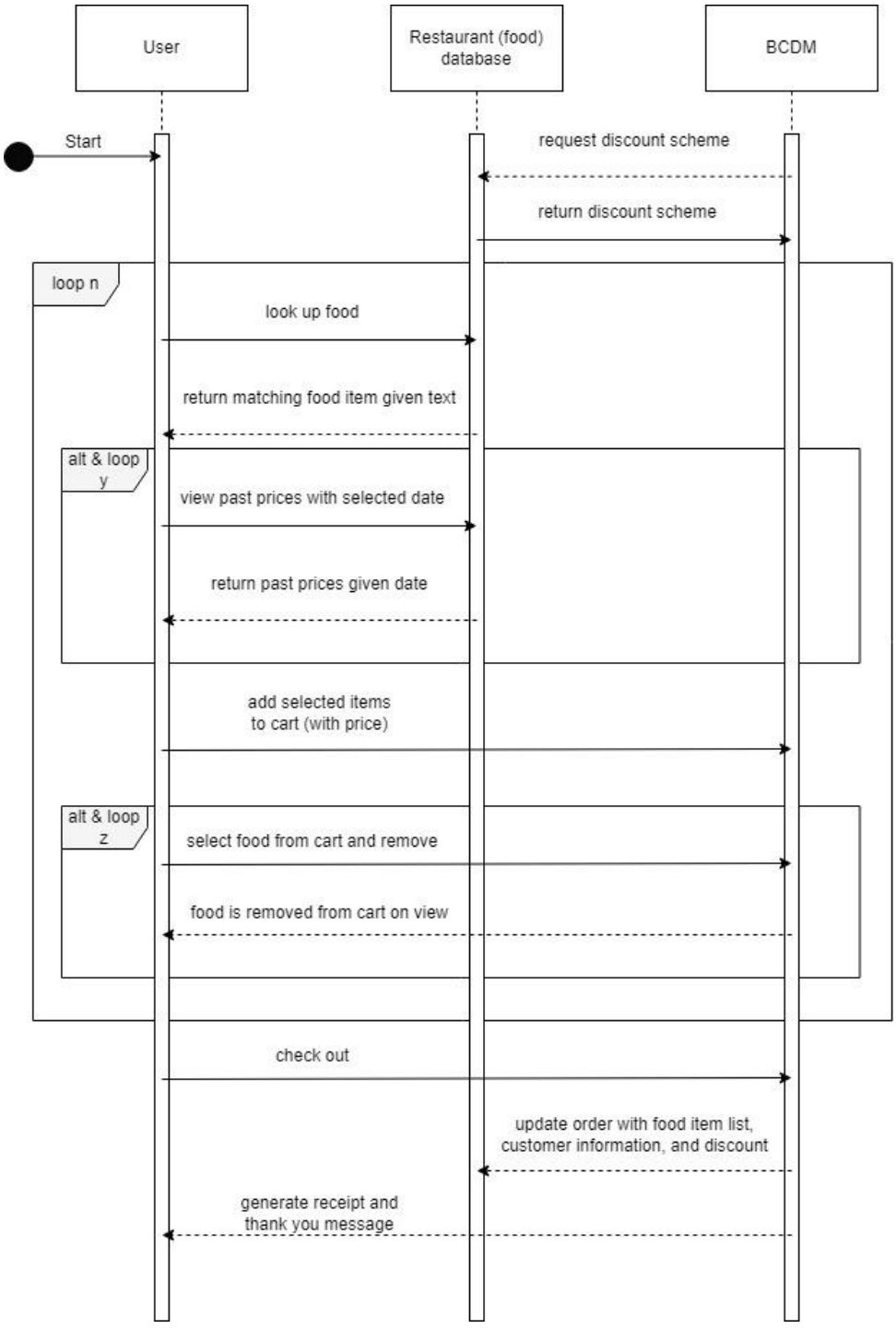

Figure 4: Sequence Diagram (Checkout cart)

5. State Machine Diagram

5.1 Admin/Staff SMD

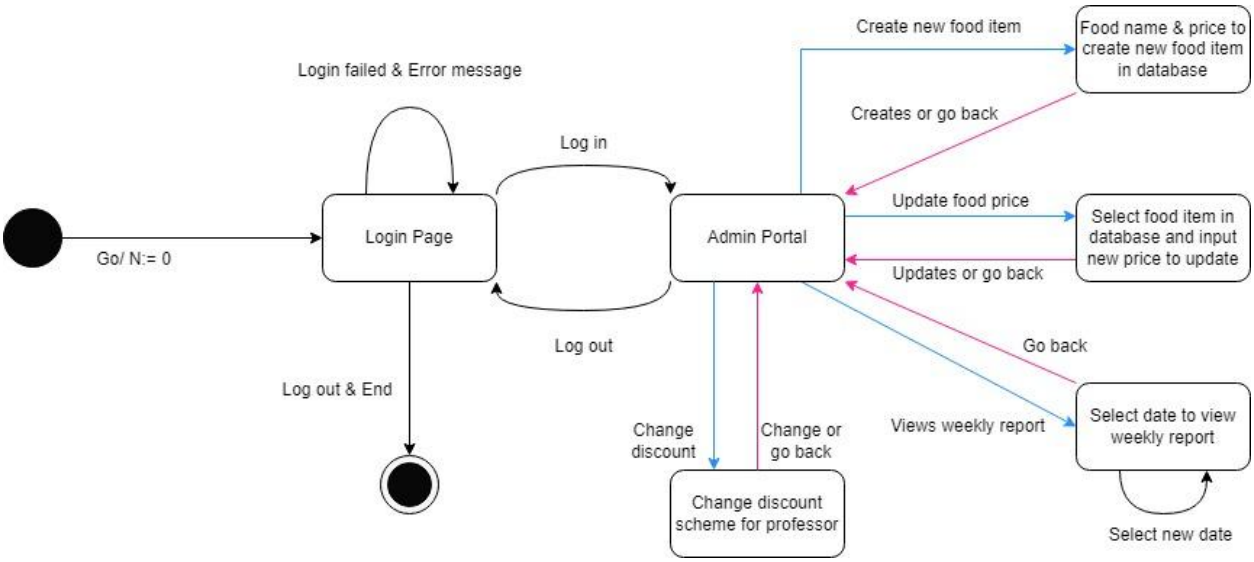

Figure 5: State Machine Diagram (For Staff/Admin)

5.2 Customer SMD

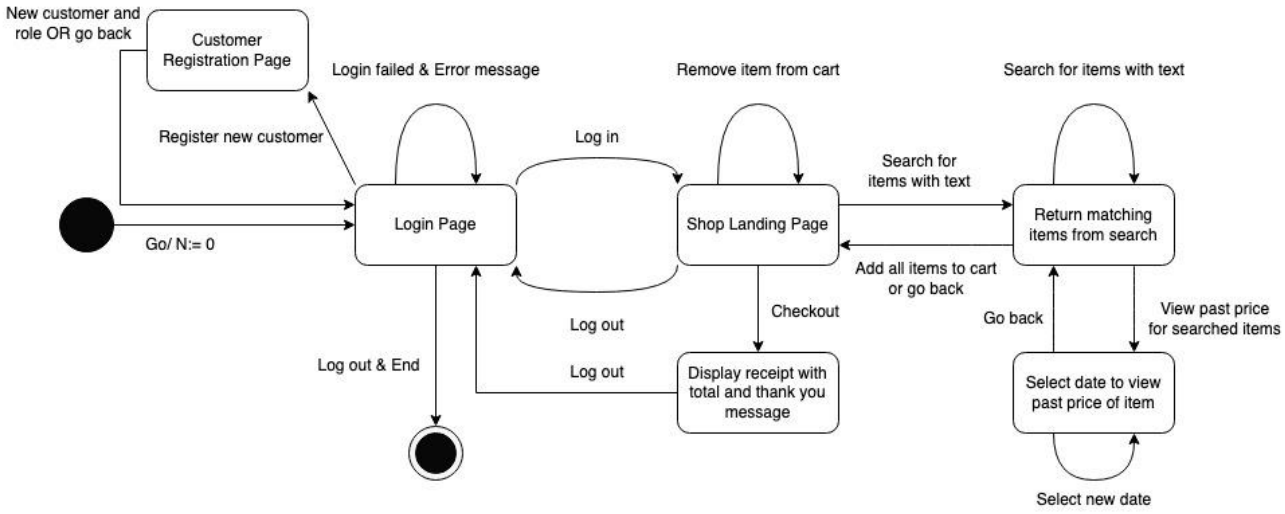

Figure 6: State Machine Diagram (For Customer)

6. System Architecture View and Style/Pattern

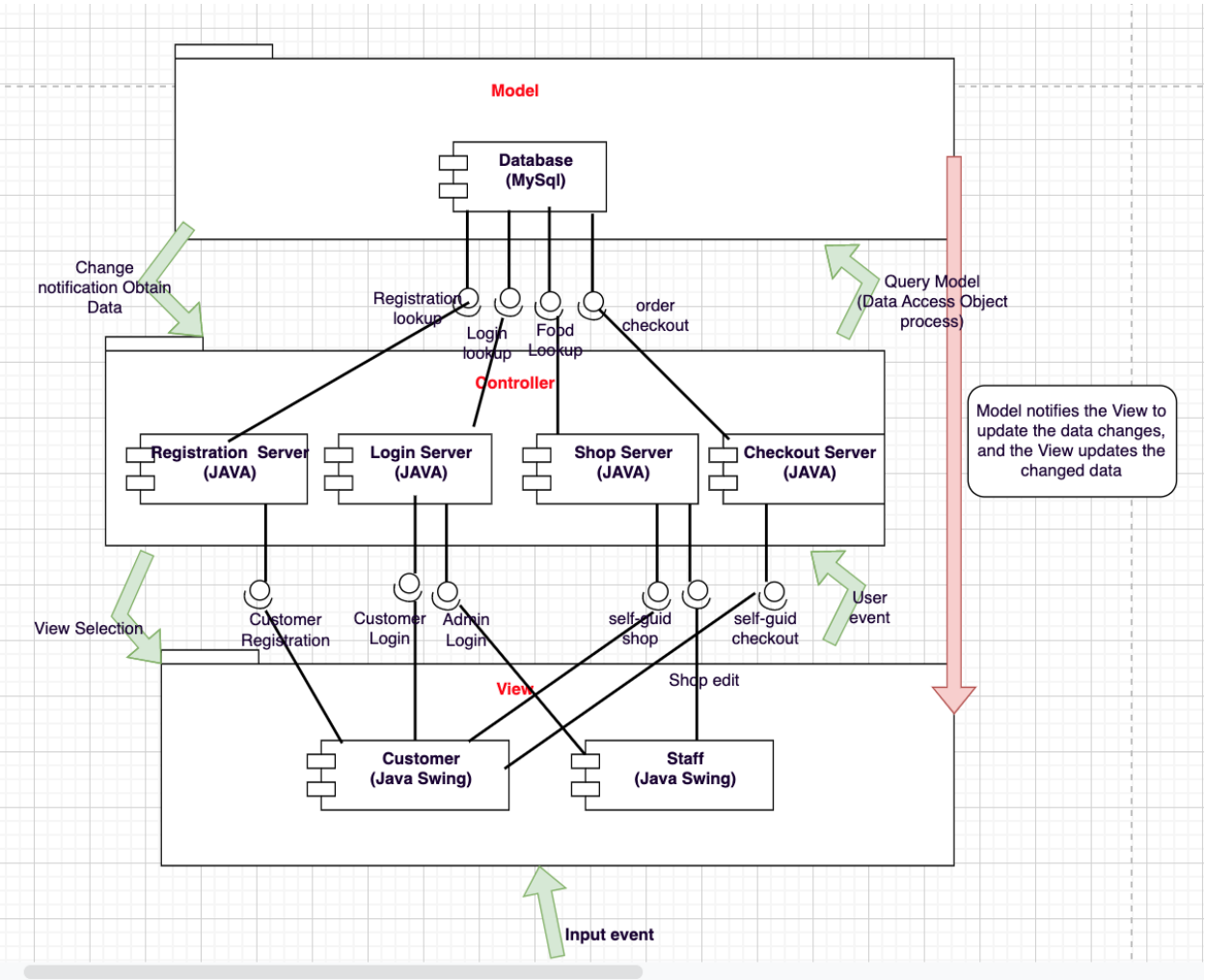

Figure 7: System Architecture Diagram

We will use MVC System Architecture Pattern to design this project, as we know, the Model layer is business data, View layer is the user interface and Controller is used to handle dispatcher events. When the View layer accepts a user interface request(input event) such as self-guided shop, checkout order etc, it will transfer requests to Controller, and Controller will operate Model layer and manipulate the database such as Add, Delete, Update, Search functions. Finally, the Model layer will notify the View layer to update the data changes, and View updates changed data.

Paper-based Model

1. Technical Description of the System

SCREEN_LOGIN

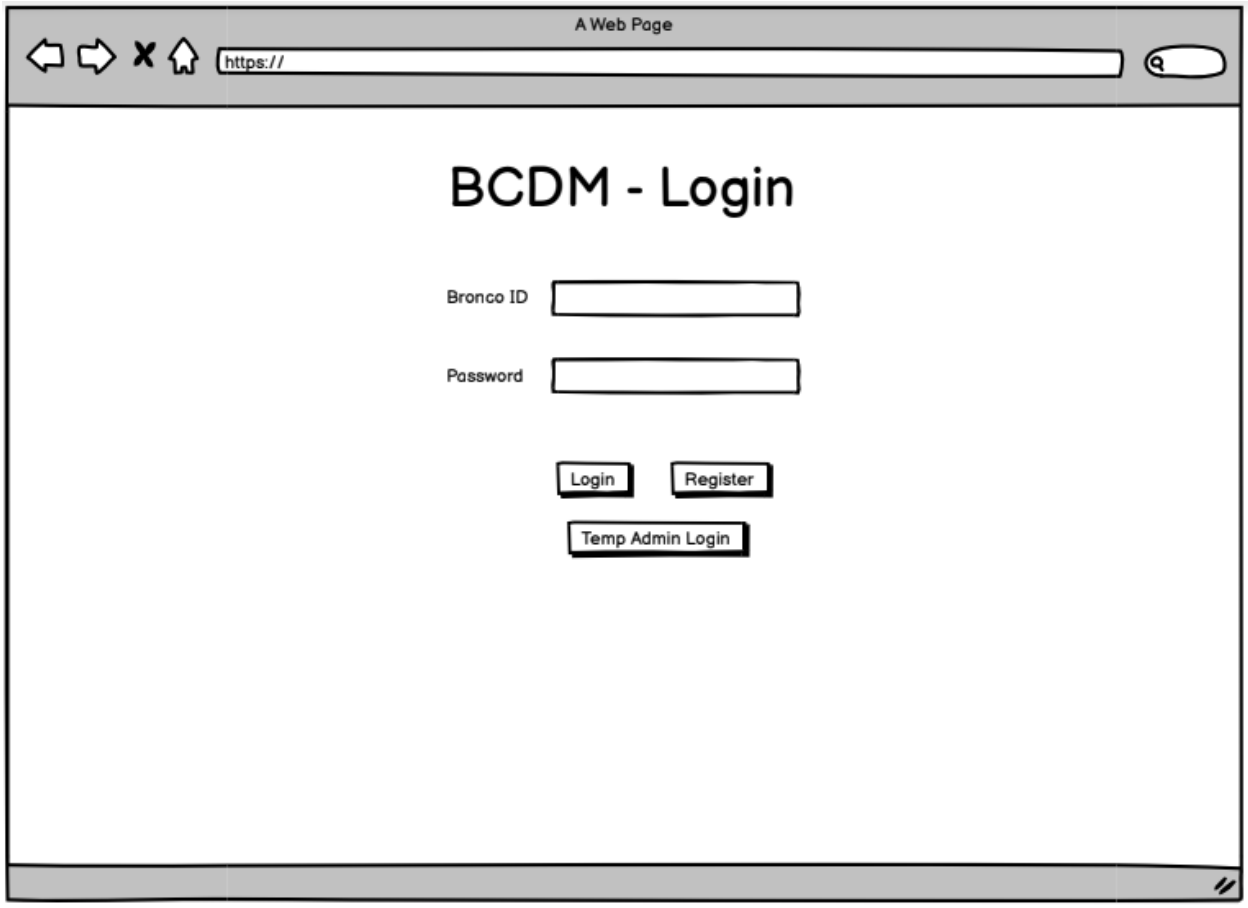

Figure 1: SCREEN_LOGIN with Login, Register, and a Temp Admin Login for prototype demoing purposes

SCREEN_REGISTER

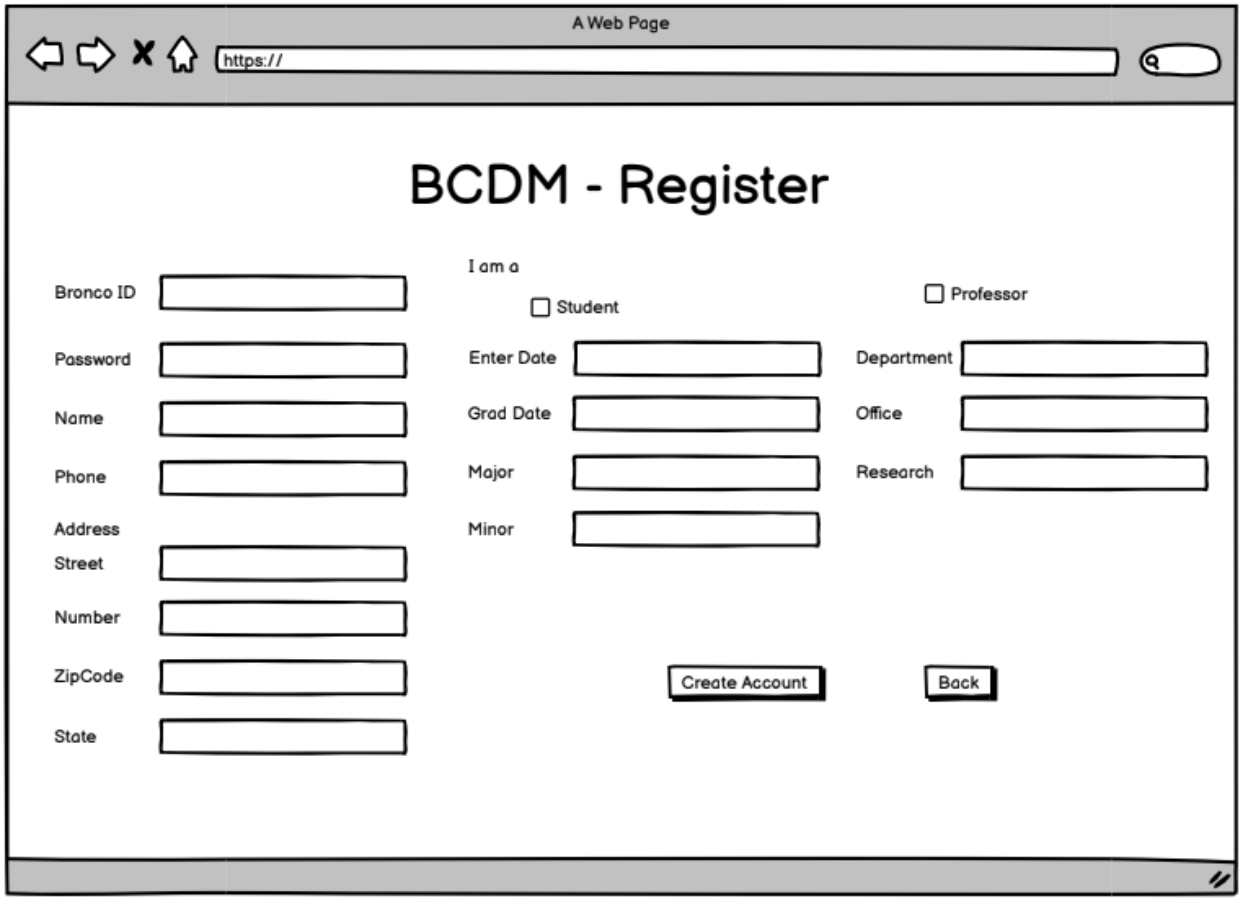

Figure 2: SCREEN_REGISTER with Bronco ID field and password. Note the multi value selection of student and professor

SCREEN_ADMIN

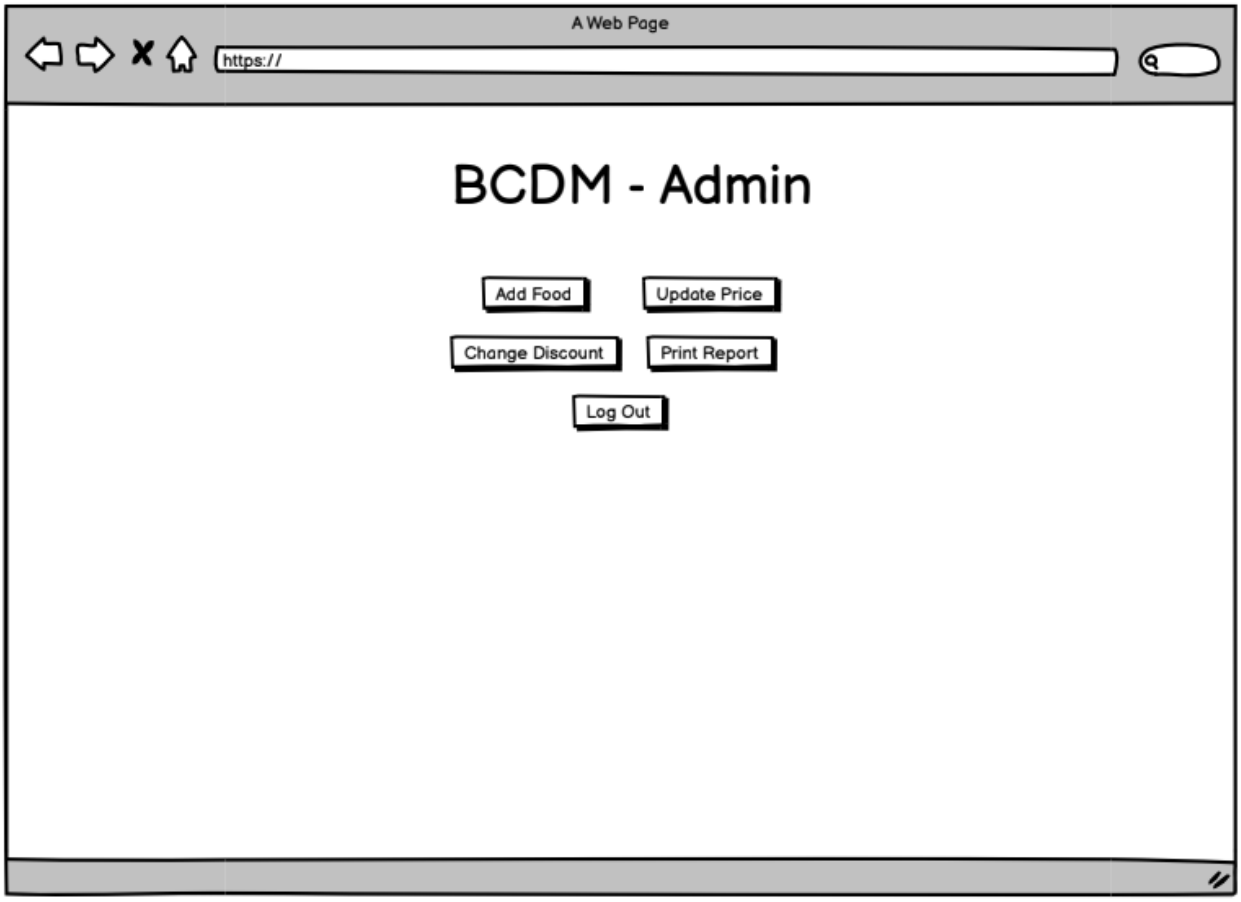

Figure 3: SCREEN_ADMIN with Add Food, Update Price, and Print Report

SCREEN_ADDFOOD

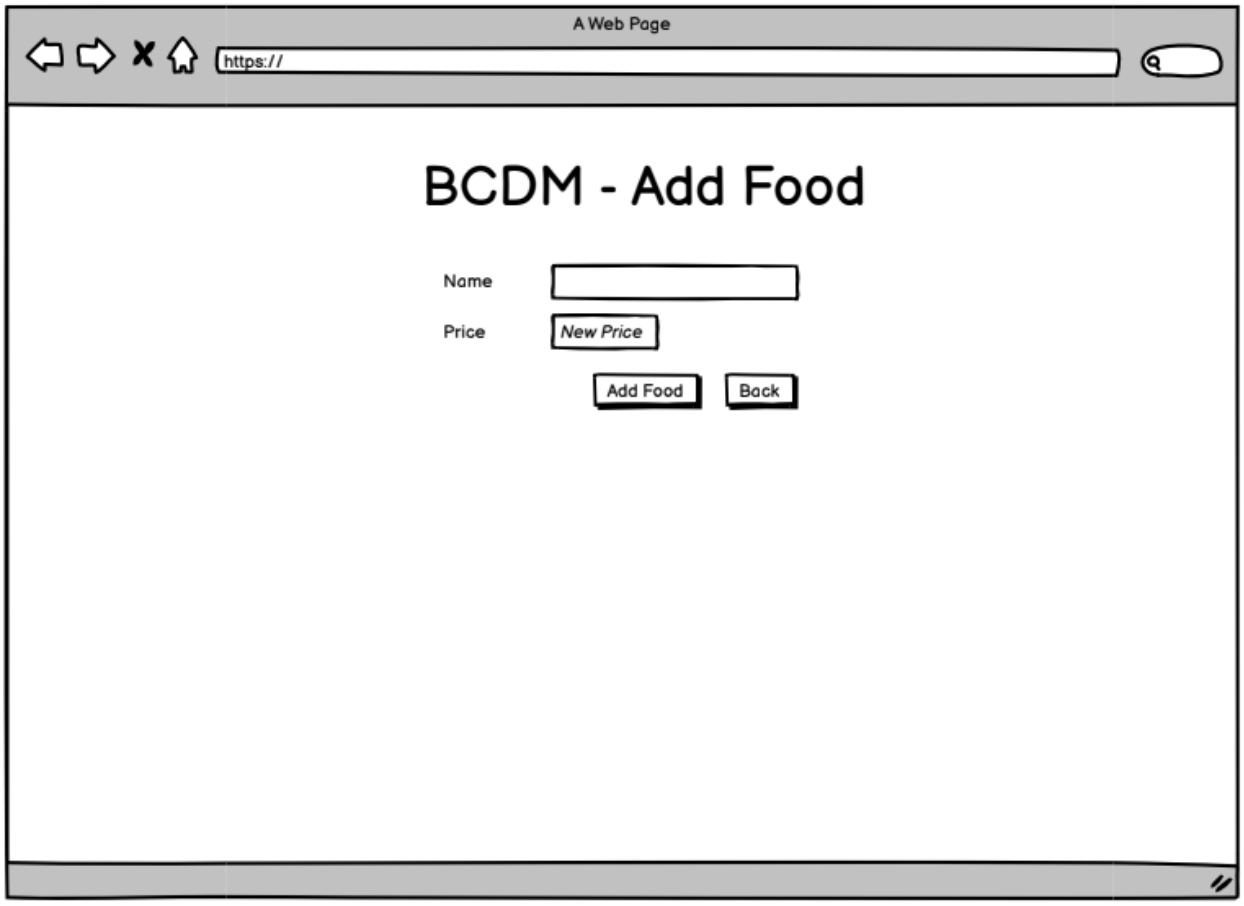

Figure 4: SCREEN_ADDFOOD can add food with name and price fields

SCREEN_UPDATEPRICE

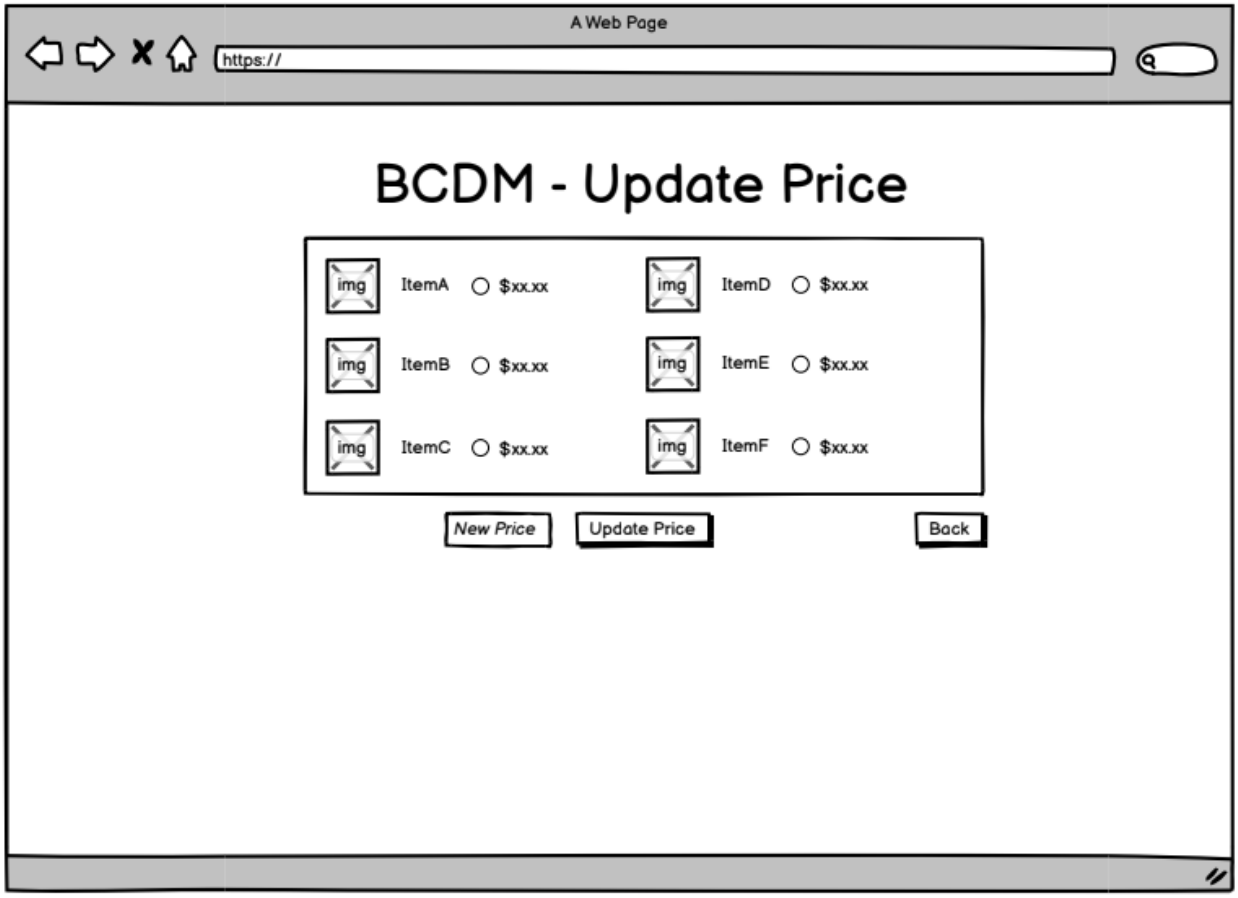

Figure 5: SCREEN_UPDATEPRICE selects an item and has new price field to update a price

SCREEN_CHANGEDISCOUNT

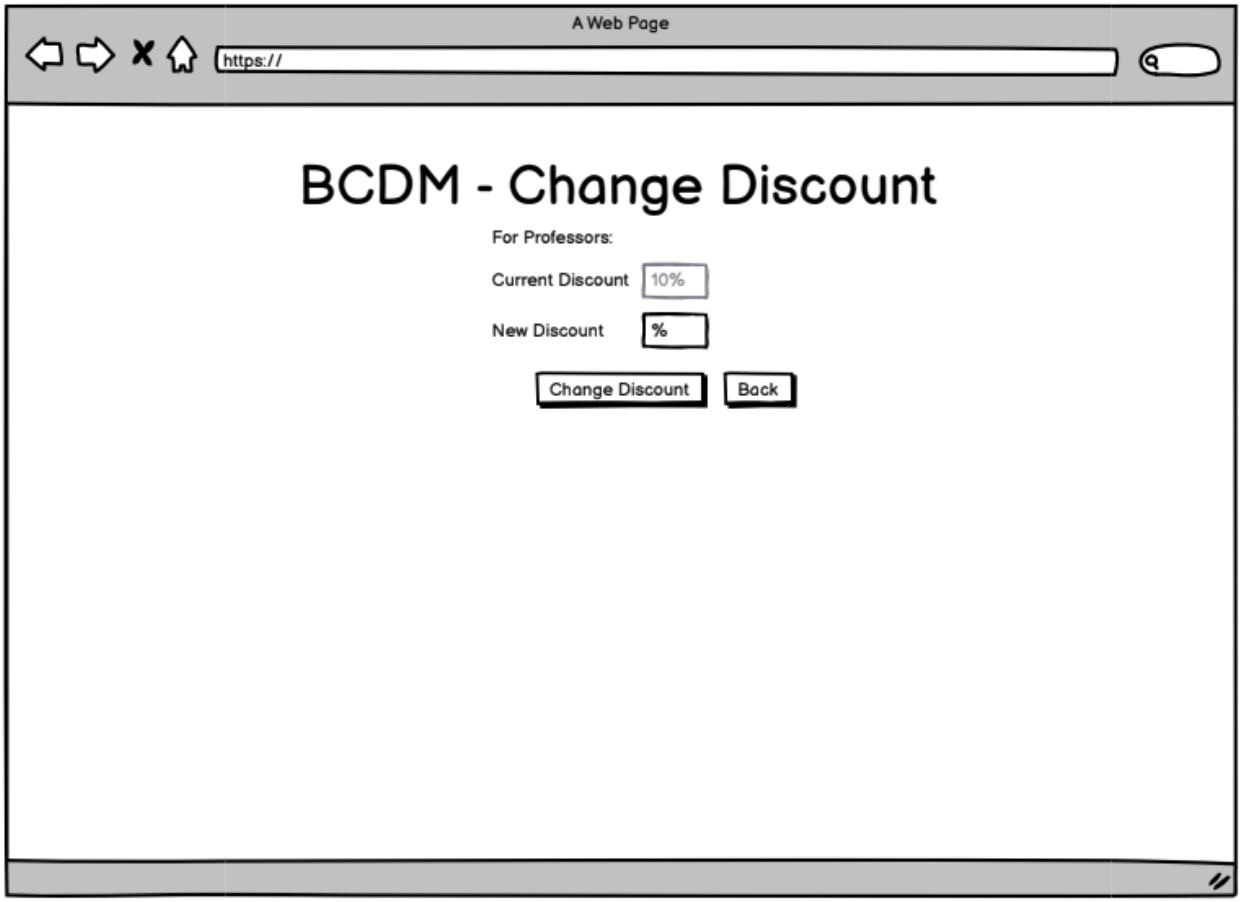

Figure 6: SCREEN_CHANGEDISCOUNT allows admin to change discount scheme

SCREEN_PRINTREPORT

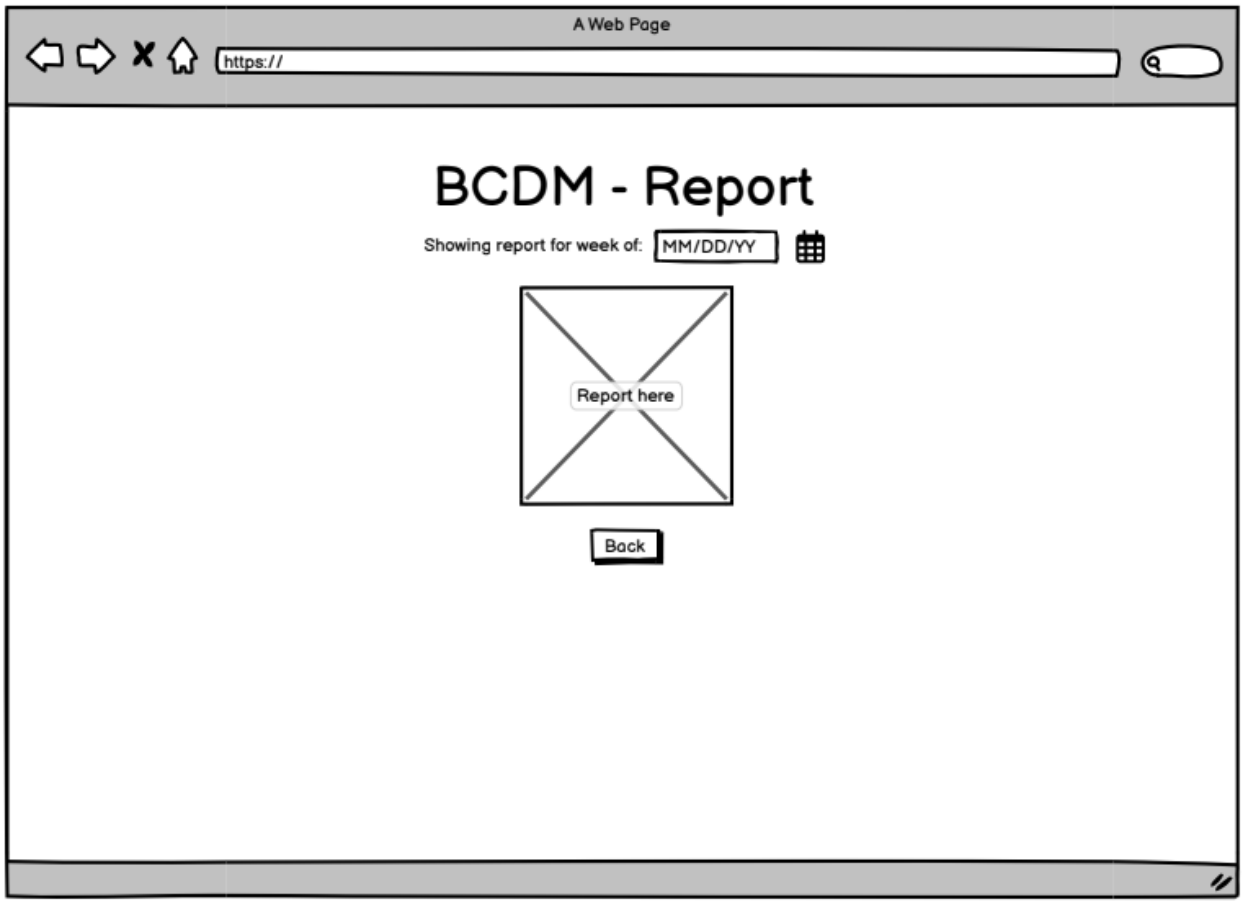

Figure 7: SCREEN_PRINTREPORT displays a report, given a date time frame

SCREEN_SHOP

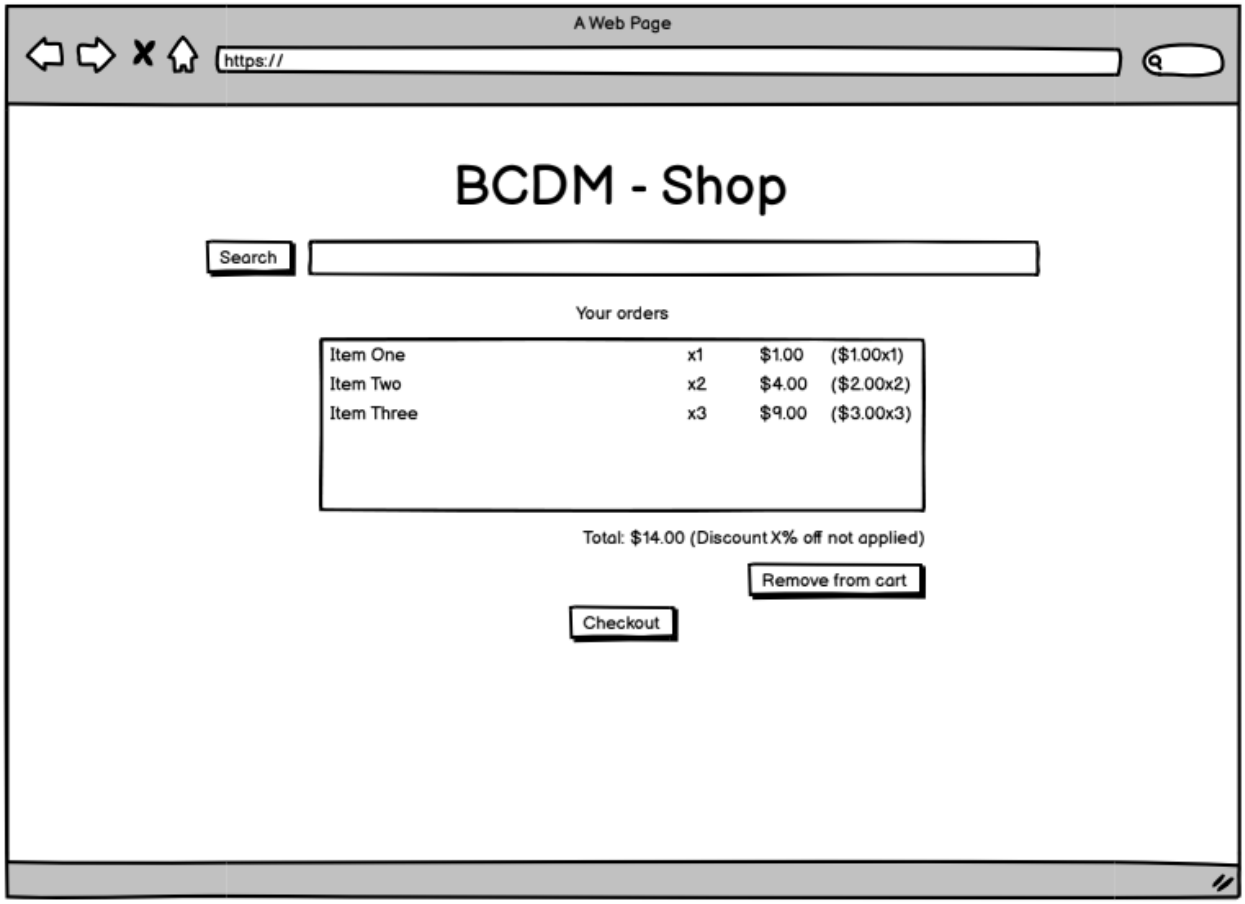

Figure 8: SCREEN_SHOP is the landing screen whenever a Customer logs in successfully

SCREEN_SEARCH

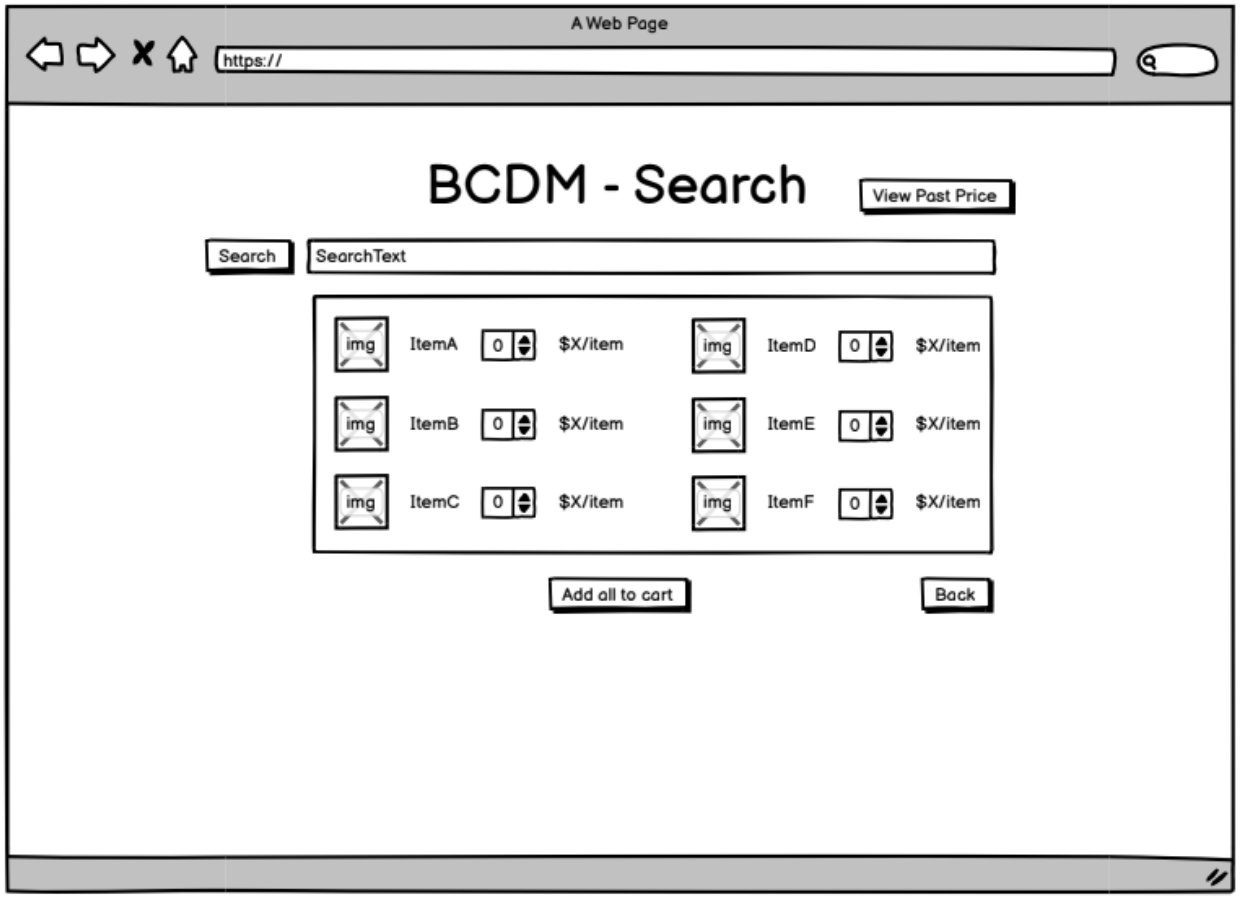

Figure 9: SCREEN_SEARCH employs a self-guided search tool to look up items with the given searchtext

SCREEN_PASTPRICE

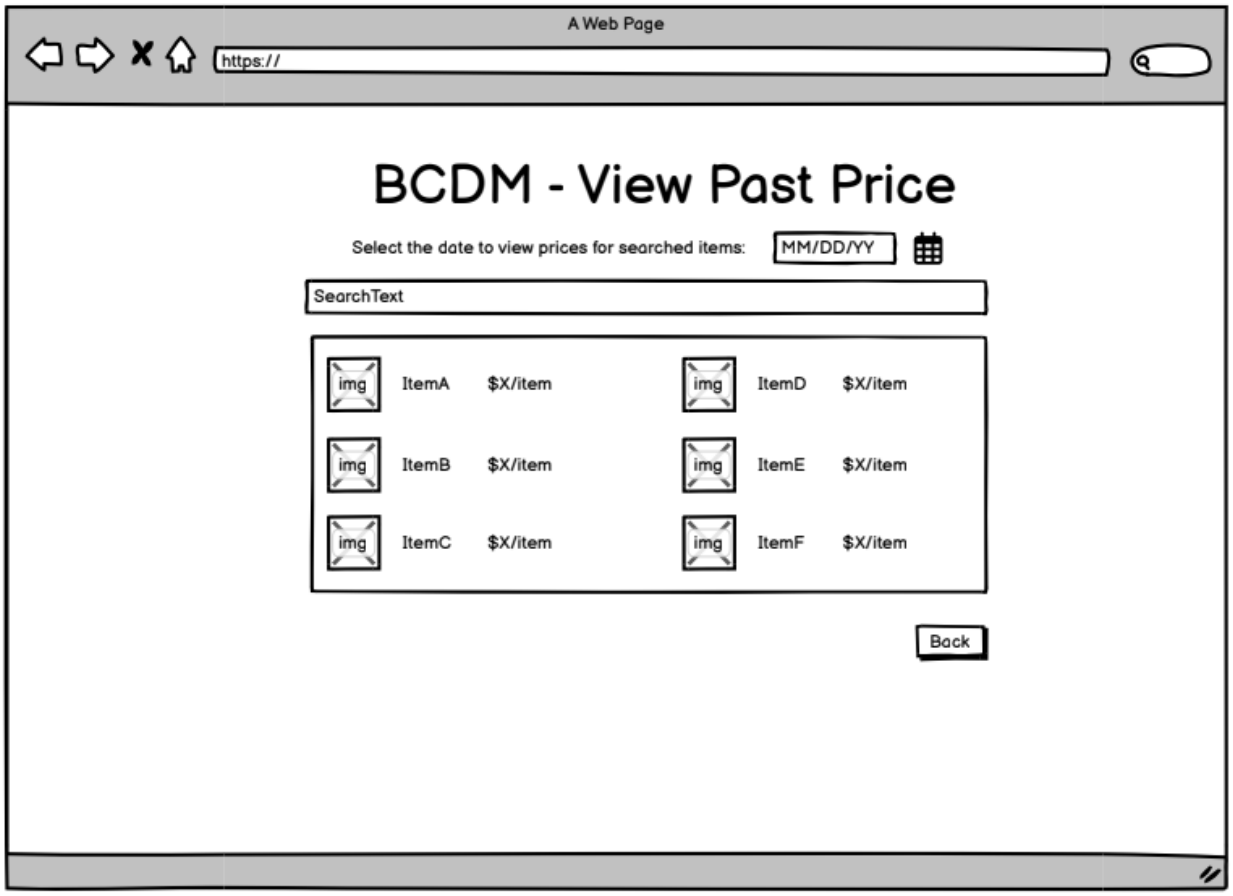

Figure 10: SCREEN_PASTPRICE contains the past price of the searched item, displayed by time

SCREEN_RECEIPT

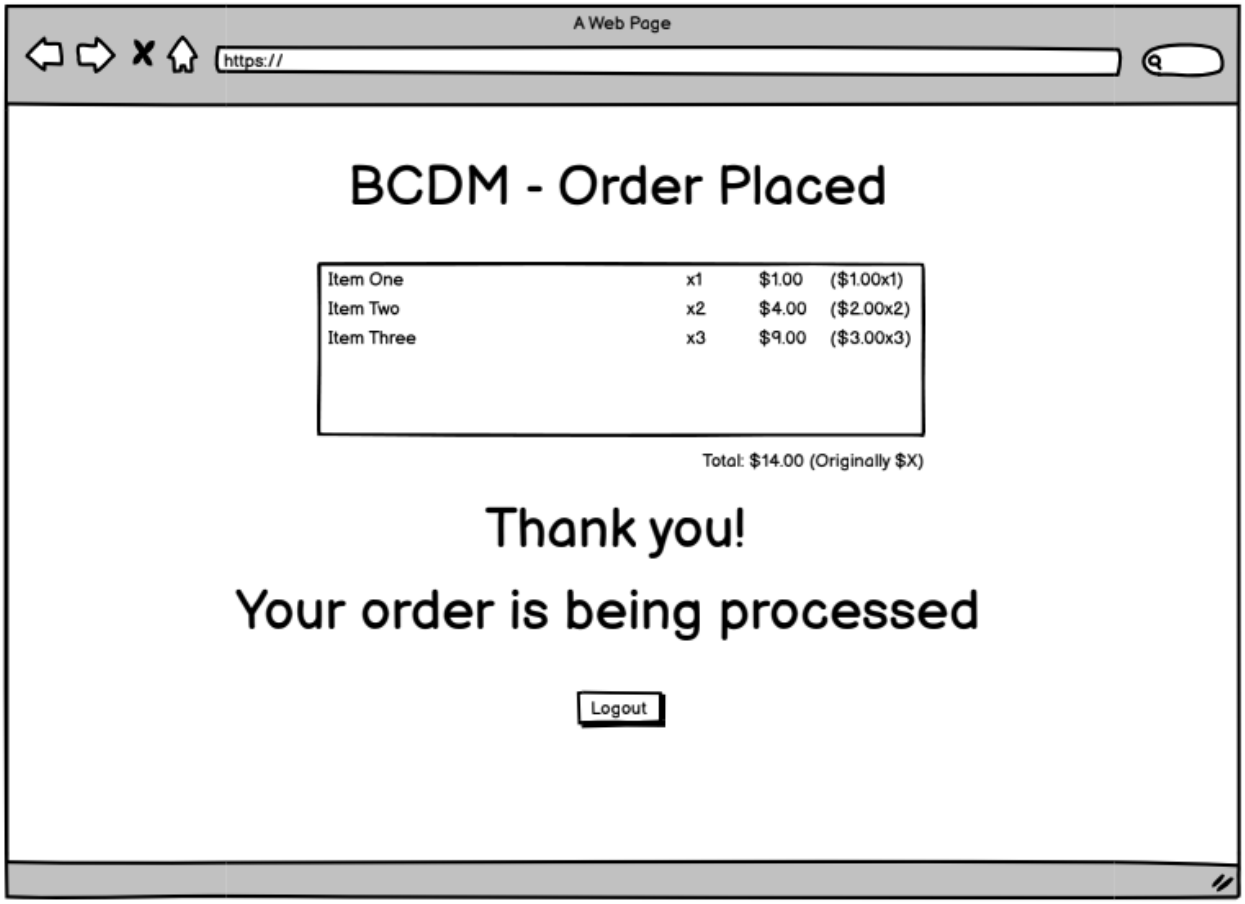

Figure 11: SCREEN_RECEIPT displays the receipt of the order after an order has been placed

Implementation

1. Github Repository Link

This project will use the following Github link for the repository: https://github.com/bryan-trinh/cpp_cs5800_project

We also included the following:

- Data Physical Model: PostgreSQL
- Graphical User Interface: Java Swing
- Source Code: Best used with Eclipse IDE
- Persistence: Hibernate
- Web Development: JSP and Java Servlets

2. Demo Video Link

Here is the link to our video demo:
https://drive.google.com/file/d/1sIQAn01mav1Q2eEEthfG3t6QpITBQLI5/view?usp=share_link

Testing Plan

1. Test Scenarios

Table 1: Test Scenarios

Test Scenarios		
Scenario ID	Title	Description
TS_01	Log in	Users are able to log into the system by providing a registered Bronco ID and password pair, then the application redirects to the next screen based on the user's credential and permissions. If the user has placed an online order with status online-pending, then the order is marked as online-complete.
TS_02	Search	Customers are able to search for food items by name on the self-guided search
TS_03	Checkout	Customers are able to place their order and receive their receipt for their total, with a discount applied if applicable.

2. Unit Tests

Table 2.1: UT_01

Test Scenario ID	TS_01
Test Scenario	Log in
Test Case ID	UT_01
Test Case	When the log in error message is displayed, clicking the "OK" button will close the error message
Preconditions	1. The log in error message is displayed
Instructions (Steps)	1. Press the "OK" button
Expected Output	1. Error message disappears
Type of Implementation	Manual
Phase	Implementation
Dependencies	IT_02
Requirement Tested	REQ_01

Table 2.2: UT_02

Test Scenario ID	TS_02
Test Scenario	Log in
Test Case ID	UT_02
Test Case	When inputting the correct user credentials, the login() function will return True
Preconditions	1. Making the login() function take ID and password as arguments and returns an array [id, password] 2. The “Log in” screen is loaded 3. Valid ID password pair is [“2”, “123”] and is in the database
Instructions (Steps)	1. Input “2” into the ID field 2. Input “123” into the password field 3. Press “Login”
Expected Output	1. Login() function will return True
Type of Implementation	Automated (JUnit)
Phase	Implementation
Dependencies	IT_01, IT_02
Requirement Tested	REQ_01

Table 2.3: UT_03

Test Scenario ID	TS_02
Test Scenario	Search
Test Case ID	UT 03
Test Case	When the search error message is displayed, clicking the “OK” button will close the error message
Preconditions	1. The search error message is displayed
Instructions (Steps)	1. Press the “OK” button
Expected Output	1. Error message disappears
Type of Implementation	Automated (JUnit)
Phase	Implementation
Dependencies	IT_04

Requirement Tested	REQ_03

Table 2.4: UT_04

Test Scenario ID	TS_02
Test Scenario	Search
Test Case ID	UT_04
Test Case	When searching for a food item, the search function will query and obtain a response using whatever text is inside the “search text” field
Preconditions	1. User logs in as customer successfully and Shop screen is loaded
Instructions (Steps)	1. Send the parameter “coke” in the search() function 2. Search() function queries up a SELECT statement from SQL database
Expected Output	1. Search() function returns something (could be nothing)
Type of Implementation	Automated (JUnit)
Phase	Implementation
Dependencies	ST_03
Requirement Tested	REQ_03

Table 2.5: UT_05

Test Scenario ID	TS_03
Test Scenario	Checkout
Test Case ID	UT_05
Test Case	When the checkout error message is displayed, clicking the “OK” button will close the error message
Preconditions	1. The checkout error message is displayed
Instructions (Steps)	1. Press the “OK” button
Expected Output	1. Error message disappears
Type of Implementation	Automated (JUnit)
Phase	Implementation
Dependencies	IT_05

Requirement Tested	REQ_02

Table 2.6: UT_06

Test Scenario ID	TS 01
Test Scenario	Log In
Test Case ID	UT_06
Test Case	When inputting a non-registered user into log in, the login() function will return False
Preconditions	1. Making the login() function take ID and password as arguments and returns an array [id, password] 2. The “Log in” screen is loaded 3. Valid ID password pair is [“2”, “123”] and is in the database
Instructions (Steps)	1. Input “1” into the ID field 2. Input “123” into the password field 3. Press the “Log In” button
Expected Output	1. Login() function will return False
Type of Implementation	Automated (JUnit)
Phase	Implementation
Dependencies	IT_01, IT_02
Requirement Tested	REQ_01

3. Integration Tests

Table 3.1: IT_01

Test Scenario ID	TS_01
Test Scenario	Log in
Test Case ID	IT_01
Test Case	The login() function is triggered by the “Log in” button
Preconditions	4. Making the login() function take ID and password as arguments and returns an array [id, password] 5. The “Log in” screen is loaded 6. Valid ID password pair is [“2”, “123”]
Instructions (Steps)	1. Enter a valid ID “2” on the Bronco ID field 2. Enter a valid password “123” on the password field

	3. Click the “Log in” button
Expected Output	The login() function is fired and returns an array as result which result[0] is equal to the value in the id and result[1] is equal to the value in the password field.
Type of Implementation	Manual
Phase	Testing and Integration
Dependencies	UT_01, UT_02, UT_05
Requirement Tested	REQ_01

Table 3.2: IT_02

Test Scenario ID	TS_01
Test Scenario	Log in
Test Case ID	IT_02
Test Case	Login error message displays
Preconditions	1. Mock the login() api responds an error 2. The “Log in” screen loaded up. 3. The ID “1” is not in the database
Instructions (Steps)	1. Enter an ID “1” on the Bronco ID field 2. Enter a password on the password field 3. Click the “login” button
Expected Output	1. The application received the error response. 2. An error message is displayed with an “OK” button.
Type of Implementation	Manual
Phase	Testing and Integration
Dependencies	UT_01, UT_02, UT_05
Requirement Tested	REQ_01

Table 3.3: IT_03

Test Scenario ID	TS_01
Test Scenario	Log in
Test Case ID	IT_03
Test Case	Notification message to pick up online-pending order displays

Preconditions	1. Online orders have been placed by the web application 2. The user logs in as a Customer successfully 3. Mock the login() api respond to online-pending order
Instructions (Steps)	1. The application receives a notification response 2. A notification response is displayed with an “OK” button
Expected Output	3. The application received the error response. 4. An error message is displayed with an “OK” button.
Type of Implementation	Manual
Phase	Testing and Integration
Dependencies	
Requirement Tested	

Table 3.4: IT_04

Test Scenario ID	TS_02
Test Scenario	Search
Test Case ID	IT_04
Test Case	Search error message displays
Preconditions	1. Mock the search() api responds an error 2. The “Shop” or “Search” screen loaded up. 3. The Search text field is empty
Instructions (Steps)	1. Press the “search” button
Expected Output	1. The application received the error response. 2. An error message is displayed with an “OK” button.
Type of Implementation	Manual
Phase	Testing and Integration
Dependencies	UT_03
Requirement Tested	REQ_03

Table 3.5: IT_05

Test Scenario ID	TS_03
Test Scenario	Checkout
Test Case ID	IT_05

Test Case	Checkout error message displays
Preconditions	1. Mock the checkout api responds an error 2. The “Shop” screen loaded up 3. Order list is empty
Instructions (Steps)	2. Press the “search” button
Expected Output	3. The application received the error response. 4. An error message is displayed with an “OK” button.
Type of Implementation	Manual
Phase	Testing and Integration
Dependencies	UT_05
Requirement Tested	REQ_02

4. System/Acceptance Tests

Table 4.1: ST_01

Test Scenario ID	TS 01
Test Scenario	Login
Test Case ID	ST_01
Test Case	Successfully login by Customer
Preconditions	1. The “log in” screen is loaded up 2. Credential {id=”1”, password=”test”} is a valid customer account
Instructions (Steps)	1. Enter an ID “1” in the Bronco ID field 2. Enter a correct password “test” in the password field 3. Click the “log in” button
Expected Output	1. The application redirects to the self-guided search screen on the Shop screen 2. The shop screen loads up
Type of Implementation	Manual
Phase	After Integration Test
Dependencies	IT_01
Requirement Tested	REQ_01

Table 4.2: ST_02

Test Scenario ID	TS_01
Test Scenario	Login
Test Case ID	ST_02
Test Case	Successfully login by Staff (admin)
Preconditions	1. The “log in” screen is loaded up 2. Credentials {id=”-1”, password=”123”} is valid admin account
Instructions (Steps)	1. Enter an Admin ID “-1” in the Bronco ID field 2. Enter a correct password “123” in the password field 3. Click the “log in” button
Expected Output	1. The application redirects Admin screen 2. The Admin screen loads up
Type of Implementation	Manual
Phase	After Integration Test
Dependencies	IT_01
Requirement Tested	REQ_01

Table 4.3: ST_03

Test Scenario ID	TS_02
Test Scenario	Search
Test Case ID	ST_03
Test Case	General Search
Preconditions	1. The user is a Customer and is logged in 2. Either the Shop screen or Search screen is loaded up
Instructions (Steps)	1. Enter “coke” on the search bar 2. Click the “search” button before the search bar
Expected Output	1. The application redirects to the screen which displays the list of found foods 2. All food items on the screen include keywords “coke”
Type of Implementation	Manual
Phase	After Integration Test
Dependencies	IT_05

Requirement Tested	REQ_01

Discussion

The Bronco Centerpointe Dining Management System (BCDM) project helps us review and understand the knowledge covered in the class such as software life cycle, configuration management, team organization, software testing, and design patterns. We understand how to apply those principles of software engineering to the design, development, maintenance, testing, and evaluation of computer software when we work on this project and fully integrate software engineering principles into the real situation. The BCDM System project consists of 6 parts: Project plan, Configuration Plan, Requirements Specification, Design, Paper-based Prototype, and Testing Plan, and some of them require us to overcome certain difficulties to accomplish. Here are some challenges encountered during the project, and the steps to overcome and resolve them.

In the Project Plan, the difficulty is how to determine our process model. As we know, the Evolution Prototype is used to obtain the requirements of some aspect of the system. The development of one or more prototypes may help to better understand the requirements of this project and then we can get some feedback from users when prototypes are tested intensively so that we can modify and perfect our requirements and coding and then move to the production phase. That's the reason why we select Evolution Prototype as our process model. What's more, we also need to search what are the perfect hardware requirements to support the library system. We knew that building constructing a project plan early on would provide ample time for everyone to work on their portions: BT building different classes and a sample backend, BT and FG working on the project report, DP working on understanding Hibernate and work on a sample connection between code and database, MB and CP working on the GUI. After our initial prototype stage, DP, MB, and CP worked vigorously to achieve a working application and it was a huge learning curve that the programmers have overcome since everyone on the team needed a Java refresher. The seamless transition from Hibernate to the linking of front-end back-end of DP allowed for MB and CP to integrate their polished GUI easily. The team has managed to incorporate the deliverables into a robust application, which is easy to use for both Customers and Staff.

In the Identify configuration items, we thought we needed to select some tools which can be used for the project at the beginning, finally, we figured out that configuration items are those documents we generated to control the project. Discord was used as the medium to communicate from member to member, and post announcements on a virtual bulletin board for all members to take notice. Communication was the biggest deterring factor of this project as tools were selected and agreed upon by the entire group. Initially communication was not optimal, however as the weeks went on, communication greatly improved and the messages in the server allowed for ease of inquiries and troubleshooting. However, this was one of the tricky parts we have encountered.

ER-Diagram is a necessary and important part of our project and many other diagrams and Assumptions and functional requirements should be based on it. We were working on ER-Diagram so hard together for 4 hours and we used Google Docs and Discord to get everybody on the same page and to design it. Some teams live in different cities and have different availability, especially during holiday season, but we always are able to have the time to set up weekly meetings together after class on Monday no matter how difficult it is. Although we could not get all of our deliverables, we obtained a working prototype that addresses most of the deliveries. We have accumulated lots of valuable experience in the BCDM project, which will provide references for our future work.

Reference

1. Minh, N. H. (2020, March 11). Java Servlet and JSP Hello World Tutorial with Eclipse, Maven and Apache Tomcat. Java Servlet and JSP hello world tutorial with Eclipse, Maven and Apache Tomcat. Retrieved November 28, 2022, from https://www.codejava.net/coding/java-servlet-and-jsp-hello-world-tutorial-with-eclipse-maven-and-apache-tomcat
2. Kumar, M. (2014, July 14). Web Application With Hibernate, JSP and Servlet using Eclipse | Java Web Tutor. Javawebtutor.com. Retrieved November 28, 2022, from https://www.javawebtutor.com/articles/hibernate/hibernate_web_example.php
3. Hibernate - Web Application - GeeksforGeeks. (2022, February 19). GeeksforGeeks. Retrieved November 28, 2022, from https://www.geeksforgeeks.org/hibernate-web-application/
4. JGraph Ltd(2005). Draw.io Diagram (V 7.1.9)[Diagram Modeling software] Published in 2005. Retrieved from http://app.diagrams.net/
5. 2. Kenji Hiranabe(1999). Astah Professional JUDE(V 1.4)[UML Modeling software] Published in 1999. Retrieved from http://astah.net/
6. Best of BI (2009) SQL Power Architect (V 1.0.7)[Data modeling Software] Published in 2009. Retrieved from http://www.bestofbi.com/
7. Produle System(2008). MockFlow(Basic)[UI mockup Software] Published in 2008. Retrieved from https://mockflow.com/#login
8. Textbook: Vliet H. V., “Software Engineering: Principles and Practice”, 3rd Edition, Wiley.