Development and Management of the Project:

Bronco Recreation Complex Management (BRCM)

Team Members

	Student 1	Student 2	Student 3	Student 4	Student 5
Name	Mehaa Bhatta	Mitchell Humphries	Nilay Nagar	Veerbasant Reddy	Antoine Si

Table of Contents

Table of Figures

Table of Tables

Abstract

The Bronco Recreation Complex Management (BRCM) does not have an automated system to handle customers, visits, and activities. The purpose of this project was to create such a system for the BRCM to remedy the situation.

This new system will allow for users to perform many actions. These actions include: creating an account, scheduling a visit, adding an activity, editing an activating, generating revenue reports and more. The system will have a primary desktop application which will house data in a SQL database. Additionally some features are to be alive through a web component. There should be no redundancies in the system and should share resources when possible.

The whole project was implemented using Java. For the website component, we used Tomcat, JSP, and Java Servlets in order to present an interactive experience for the user. For the desktop application, JavaFX was used to create an efficient kiosk GUI for customers. Lastly, for the central database, we used PostgreSQL and Hibernate to maintain all information pertinent to the system. All these technologies were then implemented within a Spring Boot dat

Overall, we found that the technologies chosen were inconvenient for the development team and that issues resulted from a lack of familiarity with the specified technologies. Despite these challenges, we successfully implemented a system that effectively and efficiently handles customers, visits, and activities.

Project Plan

Introduction

System-As-Is:

CPP wants to convert its current recreation complex system into an automated Bronco Recreation Complex Management (BRCM) system to ensure effective control of the services provided while reducing operational costs. Currently, all customers, visits, and recreation activities(bodybuilding, swimming, dance, martial arts, etc.) are maintained by hand. For each visit, a manual receipt is provided to the customer including header information such as the date, time, and customer name as well as the recreation activities, quantities, and corresponding individual and total prices. A copy of those receipts is made so that users can later feed this data into spreadsheets. Then, service transaction information is manually retrieved into financial reports for business analysis. Some general complaints about the current system include:

- Slow customer and recreation activity search processes while visits are being created.
- Redundant information of professors who are also students.
- Lack of reports that consolidate visits per customer and/or period.
- Inaccuracy of the prices of activities being charged at the counter.
- Inaccuracy of the information included in the spreadsheets from the receipts.
- Unavailability of an online visit application for activity reservation.

System-To-Be:

The automated BRCM system should address the problems mentioned before through a software-based solution. In particular, a desktop application with a graphical user interface should be developed to provide services such as: customer registration (students, professors), recreation activity registration, visit management, and intelligent revenue report. The system should (requirements):

- Eliminate any redundancies in the customer registration process
- Allow a different discount scheme for students and professors (% of the visit)
- Allow historical price information of recreation activities
- Print (screen) receipts when visits are completed
- Provide a report (screen) with consolidate revenue information by customer and period

In addition, the system should also include an online (Web) component for visit registration–to be used for activity reservation. This component should be fully integrated with the offline component. There should be a status indicating where the visit was generated with the following options: “counter”, “online-pending”, “online-complete”. This status should change from “online-pending” to “online-complete” when the customer completes his/her visit. Processes that should not be addressed: inventory control or activity availability, login/user privileges.

Process Model:

Scrum development model will be used for the BRCM. As of October 25th, 2022 we are collaborating to do 1-week sprints that started every Monday to allow for an agile/scrum-oriented workflow.

Organization of the project:

Specification of roles:

Table 1: Role Specification

Name	Role	Responsibilities
Mitchell Humphries	Project Manager, Designer	General management of processes, provided guidance and general assistance. Resource management and scheduling. Design ER and UML diagrams. Use case specification.
Nilay Nagar	Programer, Designer	Coding and debugging. Created and maintained backend services including database and business layer connection. Design logical models.
Antonie Si	Programer	Coding and debugging. Handled front end services for the desktop application and connection to the business layer.
Veerbasant B Reddy	Analyst	Created use case diagram, and use case specification. Designed paper based prototype model.
Mehaa Bhatta	Analyst, Tester	Prepared a wide range of test cases including: unit test, integration tests, and system tests to ensure functional and nonfunctional requirements were met. Performed tests on the system.

In addition to assigned roles team members assisted one another whenever possible.

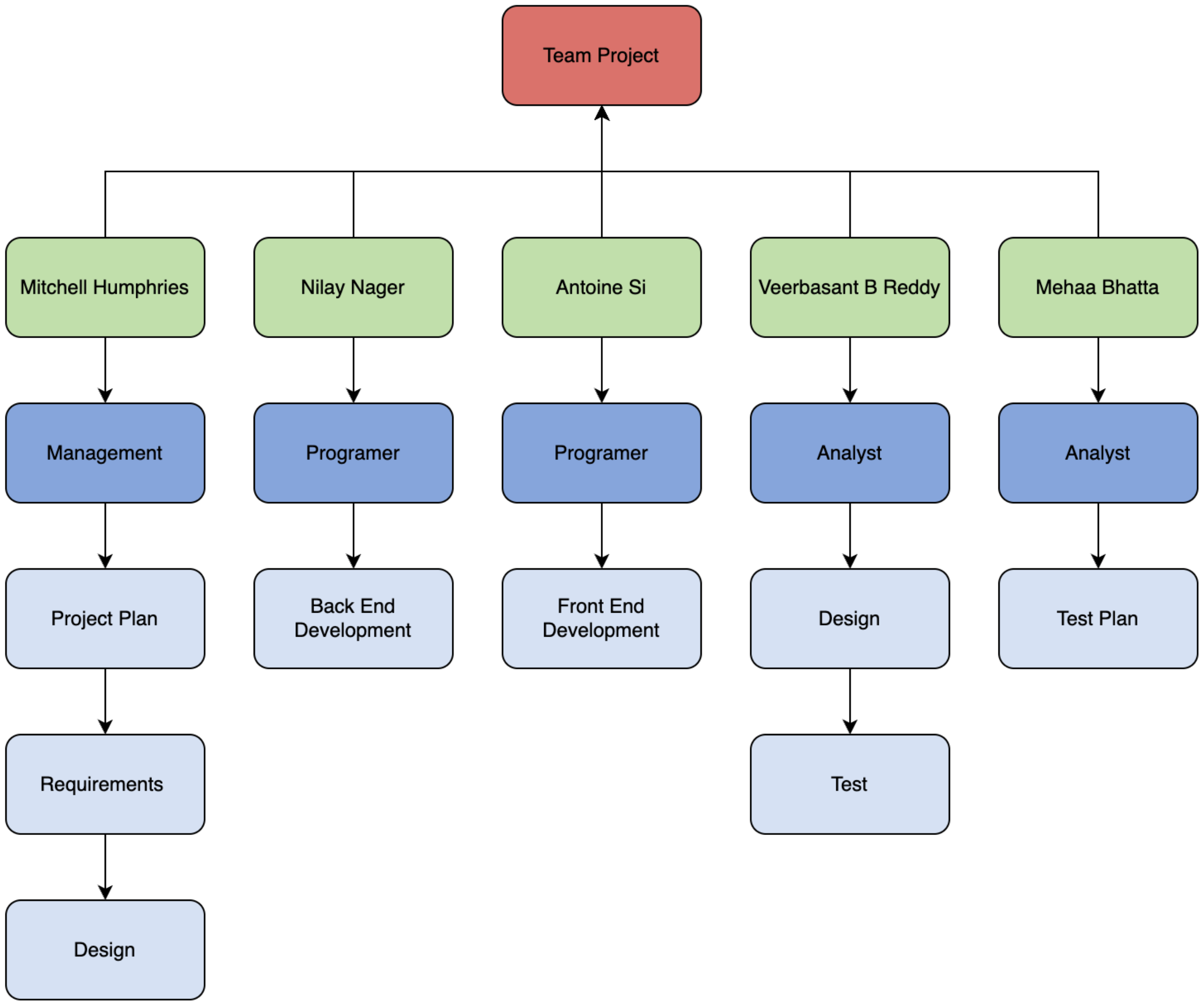
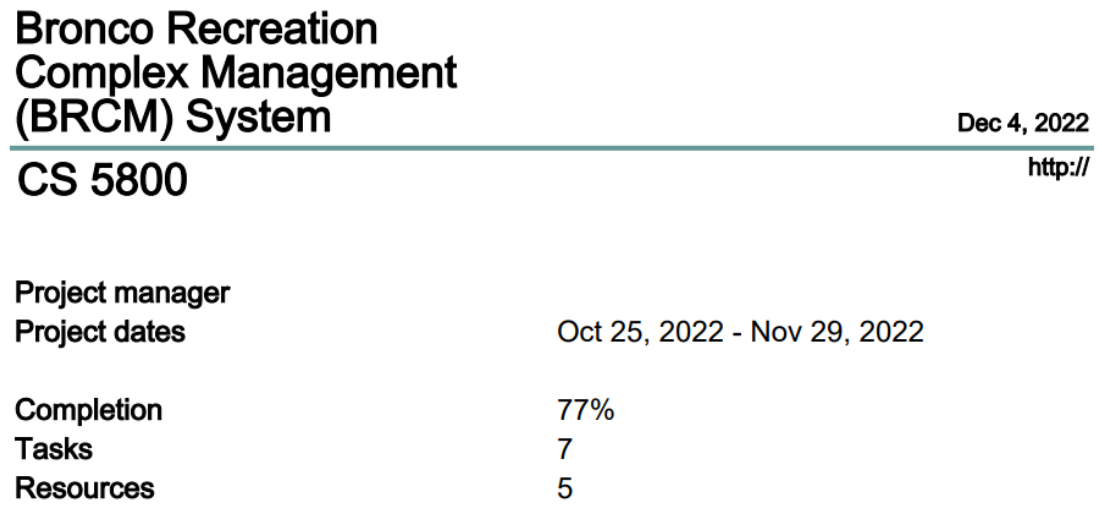


Figure 1: Work Packages

Methods and Techniques

For this project the team will design and implement a new BRCM. The team plans to create a variety of models such as: Entity relationship, UML, Logical, and more. For the implementation the team will use a variety of technologies. Some key technologies are: javaFX for the GUI, Spring Boot for the application layer, Hibernate and PostgreSQL for database storage and management.

Effort and Schedule



Bronco Recreation Complex Management (BRCM) System

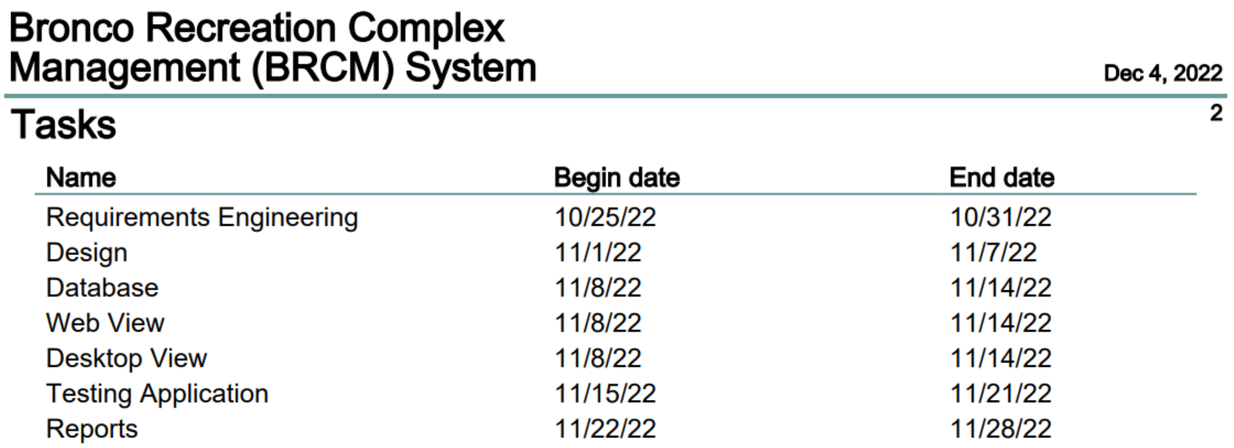
Dec 4, 2022

CS 5800

http://

Project manager	
Project dates	Oct 25, 2022 - Nov 29, 2022
Completion	77%
Tasks	7
Resources	5

Figure 2: Project Information



Bronco Recreation Complex Management (BRCM) System

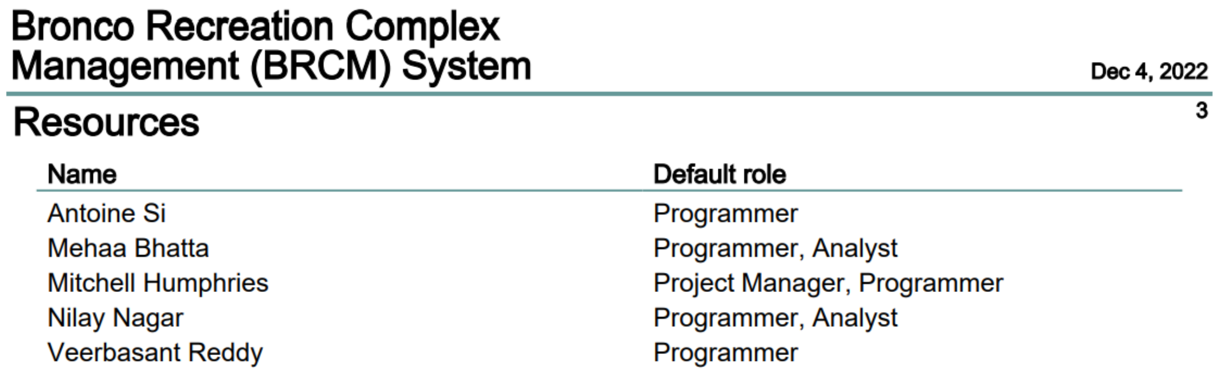
Dec 4, 2022

Tasks

2

Name	Begin date	End date
Requirements Engineering	10/25/22	10/31/22
Design	11/1/22	11/7/22
Database	11/8/22	11/14/22
Web View	11/8/22	11/14/22
Desktop View	11/8/22	11/14/22
Testing Application	11/15/22	11/21/22
Reports	11/22/22	11/28/22

Figure 3: Task Schedule

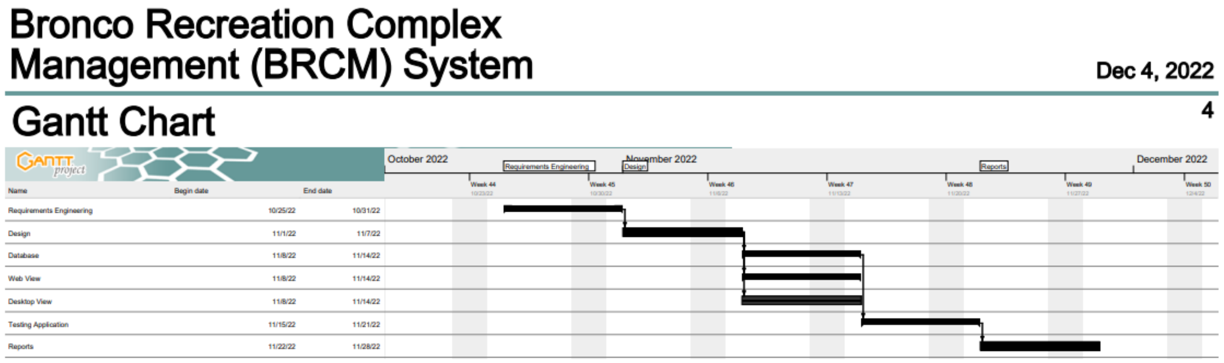


Bronco Recreation Complex Management (BRCM) System — Dec 4, 2022

Resources — 3

Name	Default role
Antoine Si	Programmer
Mehaa Bhatta	Programmer, Analyst
Mitchell Humphries	Project Manager, Programmer
Nilay Nagar	Programmer, Analyst
Veerbasant Reddy	Programmer

Figure 4: Resource Information

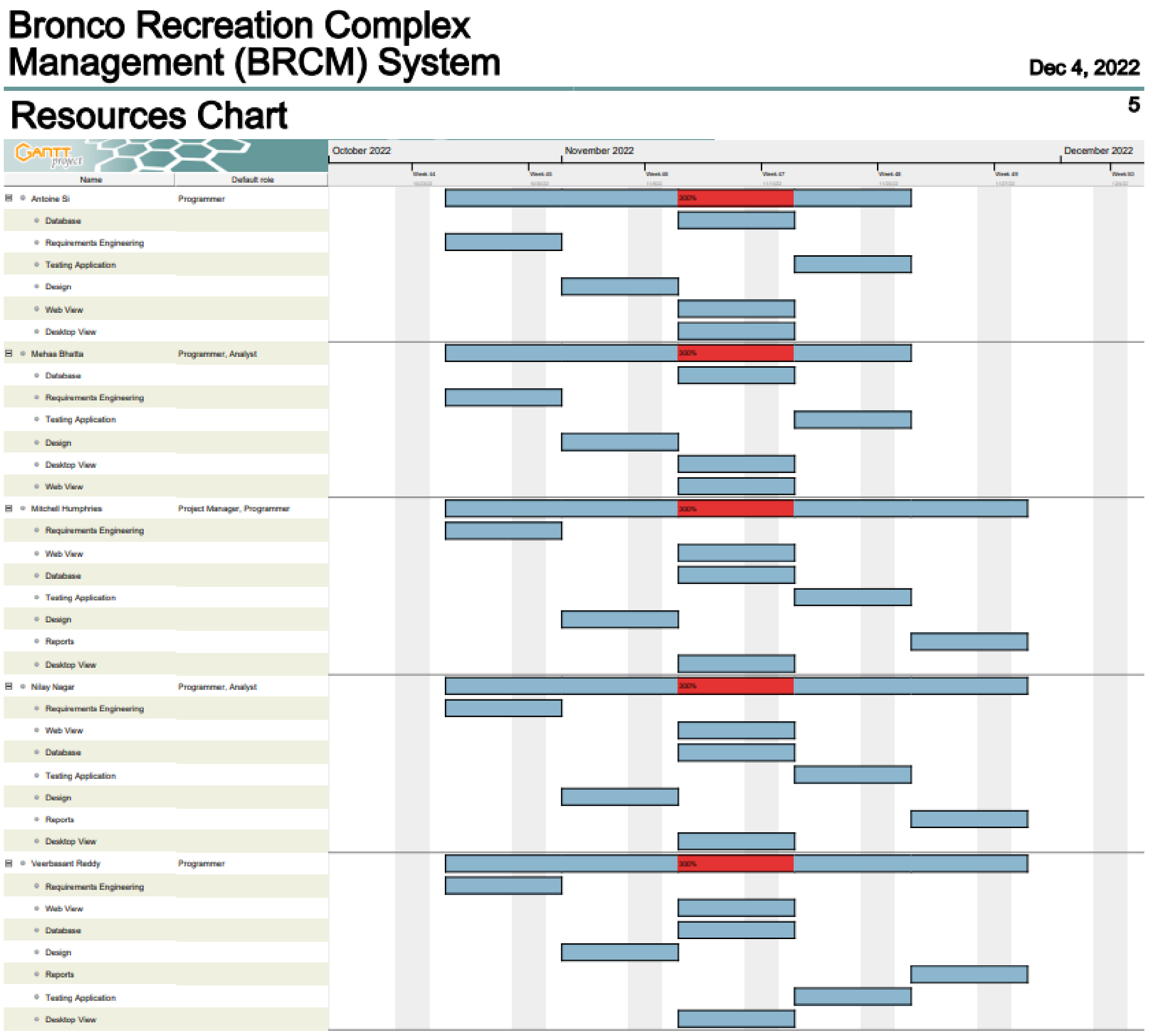


Bronco Recreation Complex Management (BRCM) System — Dec 4, 2022

Gantt Chart — 4

Name	Begin date	End date
Requirements Engineering	10/25/22	10/31/22
Design	11/1/22	11/7/22
Database	11/8/22	11/14/22
Web View	11/8/22	11/14/22
Desktop View	11/8/22	11/14/22
Testing Application	11/15/22	11/21/22
Reports	11/22/22	11/28/22

Figure 5: Gantt Chart


Bronco Recreation Complex Management (BRCM) System

Dec 4, 2022

Resources Chart

5

Name	Default role
Antoine Si	Programmer
Database	
Requirements Engineering	
Testing Application	
Design	
Web View	
Desktop View	
Mehaa Bhatta	Programmer, Analyst
Database	
Requirements Engineering	
Testing Application	
Design	
Desktop View	
Web View	
Mitchell Humphries	Project Manager, Programmer
Requirements Engineering	
Web View	
Database	
Testing Application	
Design	
Reports	
Desktop View	
Nilay Nagar	Programmer, Analyst
Requirements Engineering	
Web View	
Database	
Testing Application	
Design	
Reports	
Desktop View	
Veerbasant Reddy	Programmer
Requirements Engineering	
Web View	
Database	
Design	
Reports	
Testing Application	
Desktop View	

Figure 6: Resource Chart

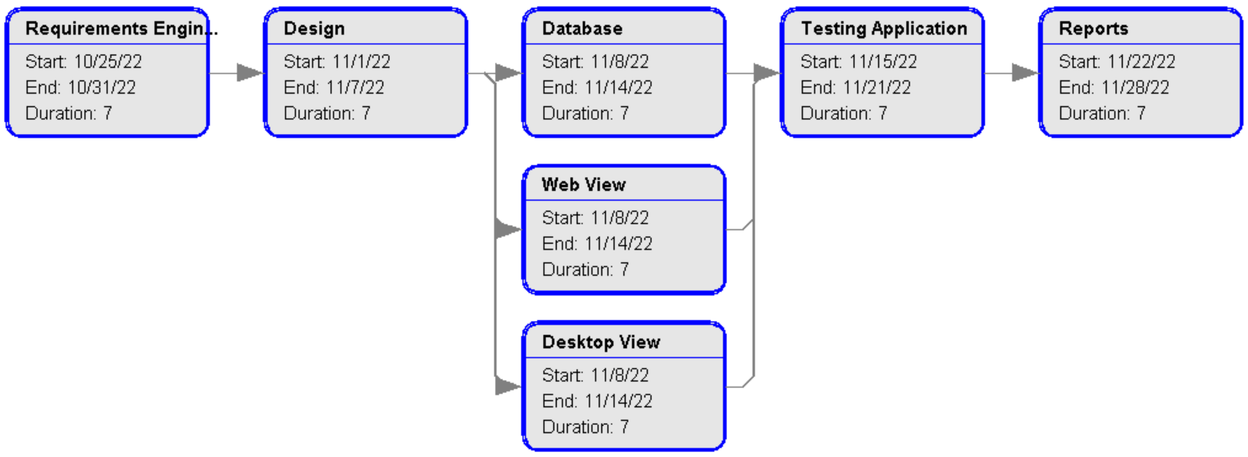


Figure 7: PERT Chart

Delivery

Table 2: Delivery Table

S.No	Deliverable Name	Description	Owner	Status
1	Preparation of Project Plan	Creating a design Model, approach, and methods. Give standards, processes, and recommendations. Distribution of resources and schedule for effort. Explain the steps that must be taken in detail.	Mitchell Humphries	Complete
3	Requirement Specification	High-level objectives, a use case diagram, and specifications that are clear. Both functional and non-functional requirements should be provided. Moreover, a traceability matrix is provided.	Mitchell Humphries Veerbasant Reddy Mehaa Bhatta	Complete
4	Design	Create class diagrams, logical models, ER diagrams, and other types	Veerbasant Reddy, Mitchell Humphries	Complete
5	Prepare Testing Plan	Prepare the system, integration, and unit tests. Acceptance tests and a quality assurance plan are provided.	Mehaa Bhatta Nilay Nagar	Complete
6	Design Prototype	Paper-based prototype design and technical delivery System description	Veerbasant Reddy Nilay Nagar Antonie Si	Complete
7	Discussion	Analysis of the consequences obtained, and training learned	Nilay Nagar, Antoine Si, Mitchell Humphries, Veerbasant Reddy, Mehaa Bhatta	Complete

Github Repository:
https://github.com/Mitch4797/Bronco-Recreation-Complex-Management

Link to DemoUI (missing database connection):
https://drive.google.com/file/d/1ZBdK-jWdmIPXu6cYHDn-889ARh94OHQ-/view?usp=sharing

Requirements Specification

High-Level Goals

- HL_1: Develop a desktop application to manage users visits to the Bronco Recreation Center.
- HL_2: Develop a web page portal which allows users to register for visits to the Bronco Recreation Center.
- HL_3: Develop a backend system which is accessed from both the desktop and web page view which has minimal redundancies.

Primary and Secondary Actors

Primary: User (Student, Professor, or StudentProfessor)
Secondary: None

Use Case Diagrams

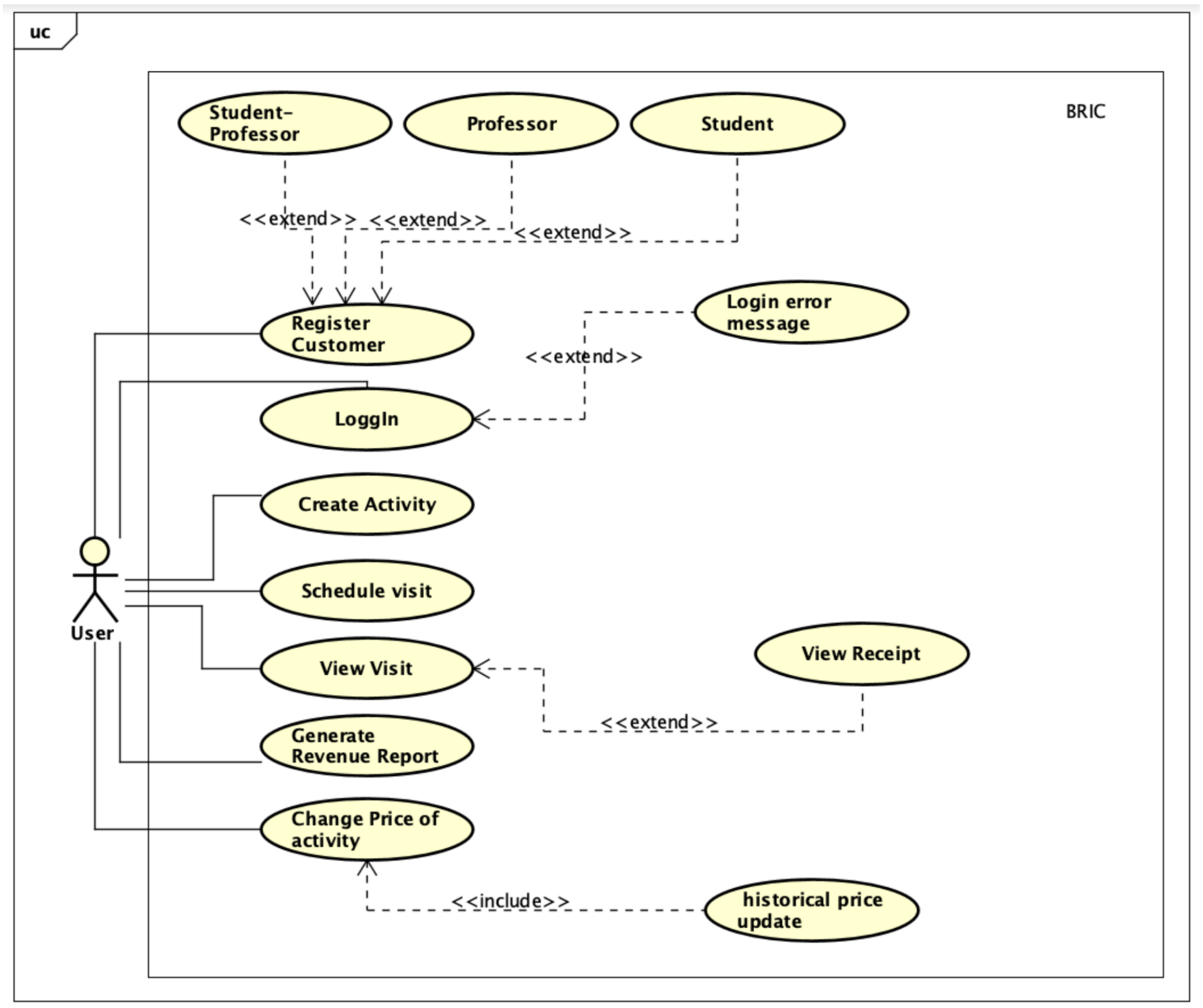


Figure 2: Use Case Diagram

Use Case Specifications

Use cases covered by system:

- Register new customer
- Log-in
- Create new activity
- Schedule visit
- View visit
- Change price
- Generate revenue report

Table 1: Use case: Customer Registration

Use Case: Register New Customer
Primary Actor: User
Secondary Actor [optional]: None
Pre Condition : User is logged in and at home page.
Post Condition: User has an account
Main Success Scenario: 1. User selects create new account. 2. User enters personal information. 3. User selects if they are a student. 4. User provides student information. 5. User clicks register.
Alternative Cours: 3b. User selects they are are a professor. 3c. User selects they are both a professor and student.
Exception Course:

Table 2: Use case: Log-in

Use Case: Log-in
Primary Actor: User
Secondary Actor [optional]: None
Pre Condition : User has an account
Post Condition: User is logged in
Main Success Scenario: 1. User enters username 2. User clicks log in 3. User is presented the home page
Alternative Cours:
Exception Course: 3a. User provides invalid credentials, presented with error message

Table 3: Use case: Create new activity

Use Case: Create new activity
Primary Actor: User
Secondary Actor [optional]: None
Pre Condition : User is logged in and at home page.
Post Condition: A new activity is created.
Main Success Scenario: 1. User selects create new activity. 2. User enters activity information. 3. User clicks confirm activity registration.
Alternative Cours: 2b. User selects home and is returned to the home screen.
Exception Course:

Table 4: Use case: Schedule Visit

Use Case: Schedule Visit
Primary Actor: User
Secondary Actor [optional]: None
Pre Condition : User is logged in and at home page.
Post Condition: User has scheduled a visit.
Main Success Scenario: 1. User selects schedule list. 2. User enters date/time. 3. User selects activities. 4. User confirms visit.
Alternative Cours: 2b. User selects home and is returned to the home screen.
Exception Course:

Table 5: Use case: View visit

Use Case: View Visit
Primary Actor: User
Secondary Actor [optional]: None
Pre Condition : User is logged in and at home page.
Post Condition: User is presented with a history of their visits.
Main Success Scenario: 1. User selects view visits. 2. User is presented with a list of past visits. 3. User selects a single visit 4. User is presented with the receipt from the selected visit.
Alternative Cours: 2b. User selects home and is sent to the home screen.
Exception Course:

Table 6: Use case: Change Activity Price

Use Case: Change Activity Price
Primary Actor: User
Secondary Actor [optional]: None
Pre Condition : User is logged in and at home page.
Post Condition: The price of an activity is changed.
Main Success Scenario: 1. User selects edit activity. 2. User selects an activity 3. User enters new price
Alternative Cours: 2b. User selects home and is sent to the home screen.
Exception Course:

Table 7: Use case: Generate Revenue Report

Use Case: Generate Revenue Report
Primary Actor: User
Secondary Actor [optional]:
Pre Condition : User is logged in and at home screen.
Post Condition: User is presented with a revenue report
Main Success Scenario: 1. User selects Revenue Report. 2. User enters the start and end date of the revenue report. 3. User selects generate reports. 4. User is presented with the revenue report.
Alternative Cours: 2b. User selects home and is sent to the home screen.
Exception Course:

Paper-based Prototype

Screen_0:

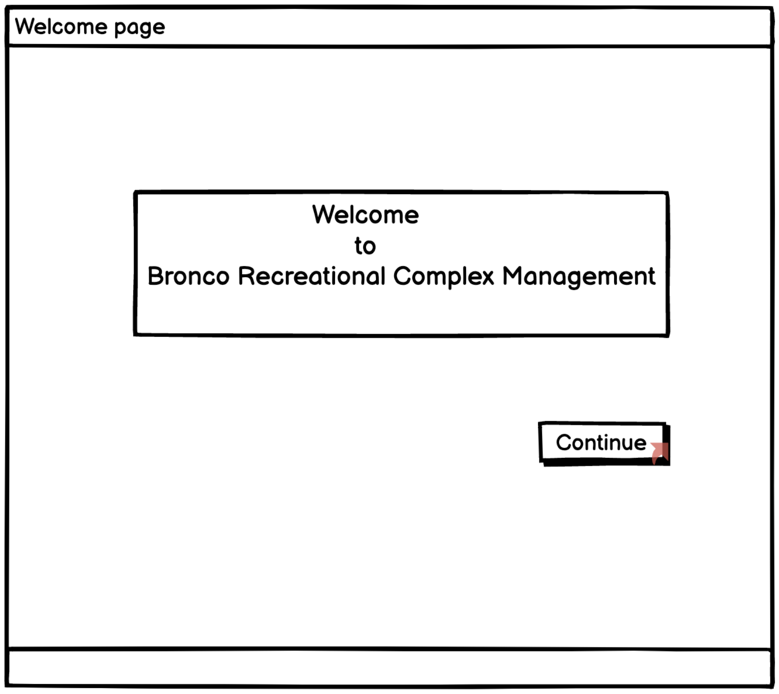


Figure 3: Welcome Screen

Screen_01:

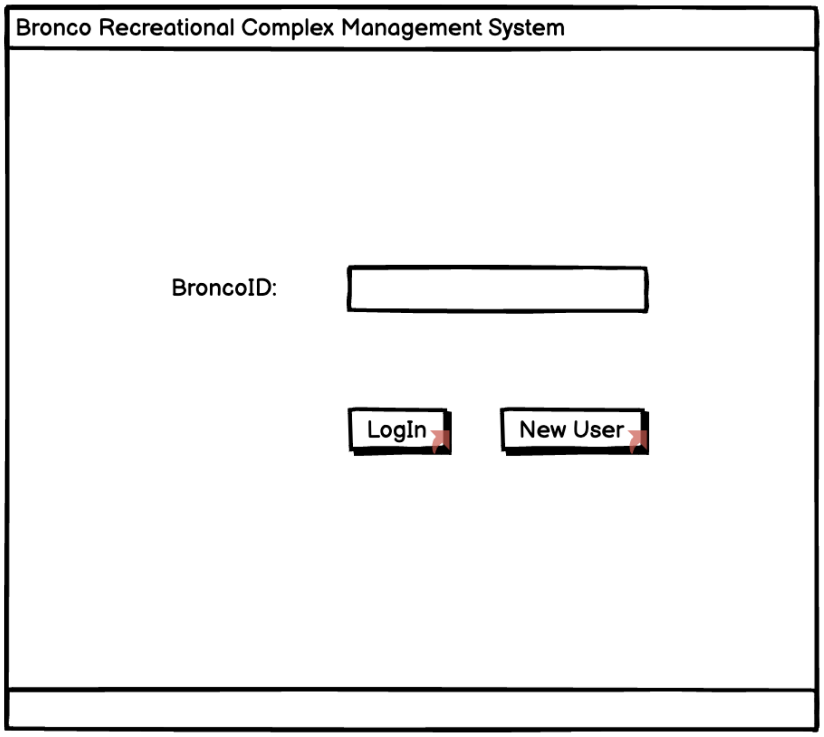


Figure 4: Login Screen

Screen_02:

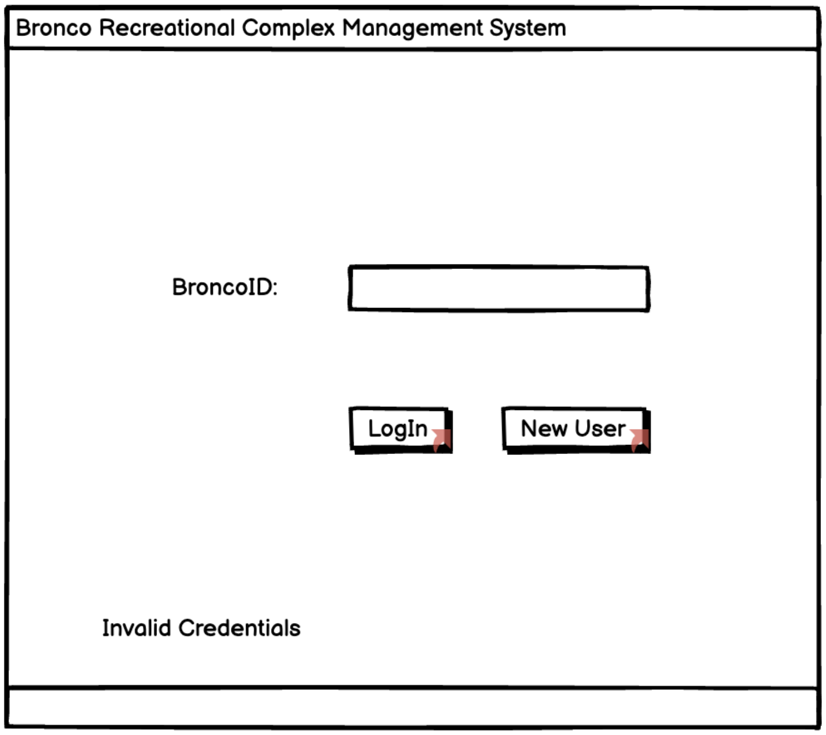


Figure 5: Invalid Login Screen

Screen_03:

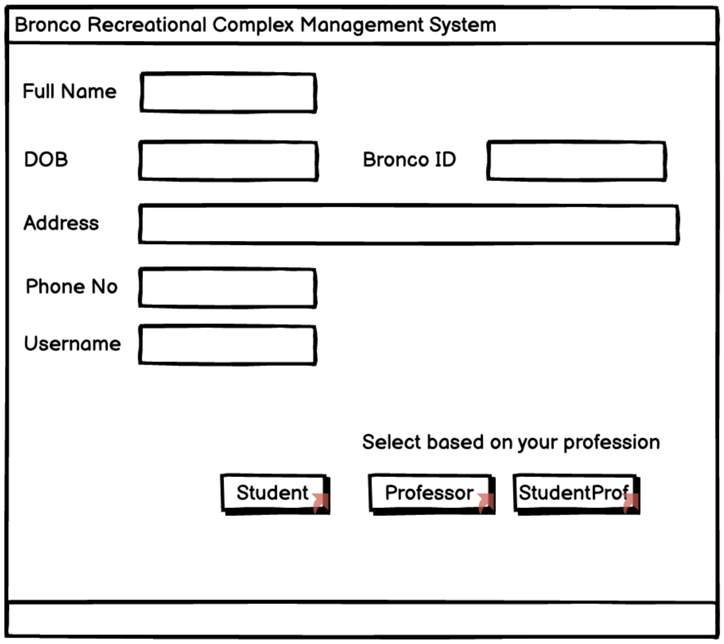

Bronco Recreational Complex Management System
Full Name
DOB
Bronco ID
Address
Phone No
Username
Select based on your profession
Student
Professor
StudentProf

Figure 6: Customer Registration screen

Screen_04:

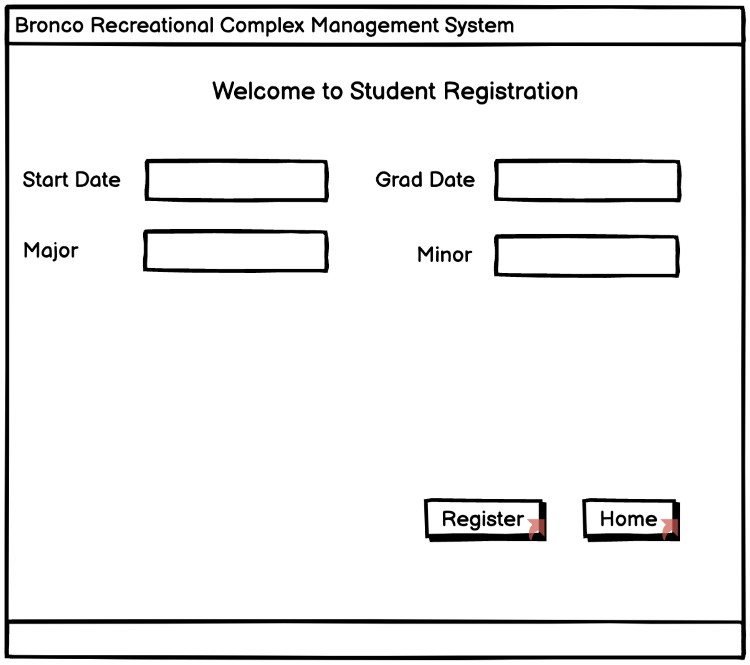

Bronco Recreational Complex Management System
Welcome to Student Registration
Start Date
Grad Date
Major
Minor
Register
Home

Figure 7: Student Registration Screen

Screen_05:

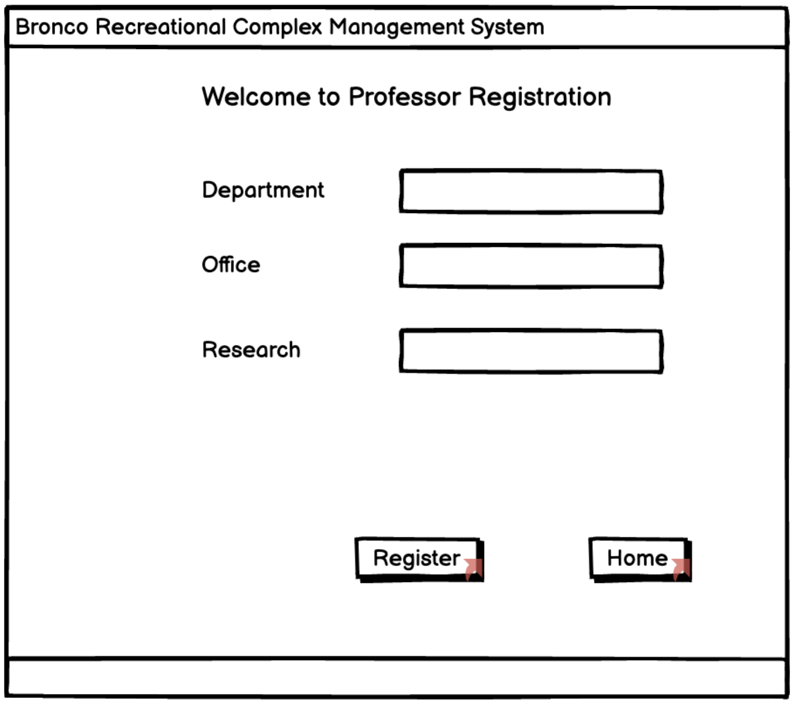

Bronco Recreational Complex Management System
Welcome to Professor Registration
Department
Office
Research
Register
Home

Figure 8: Professor Regitraction Screen

Screen_06:

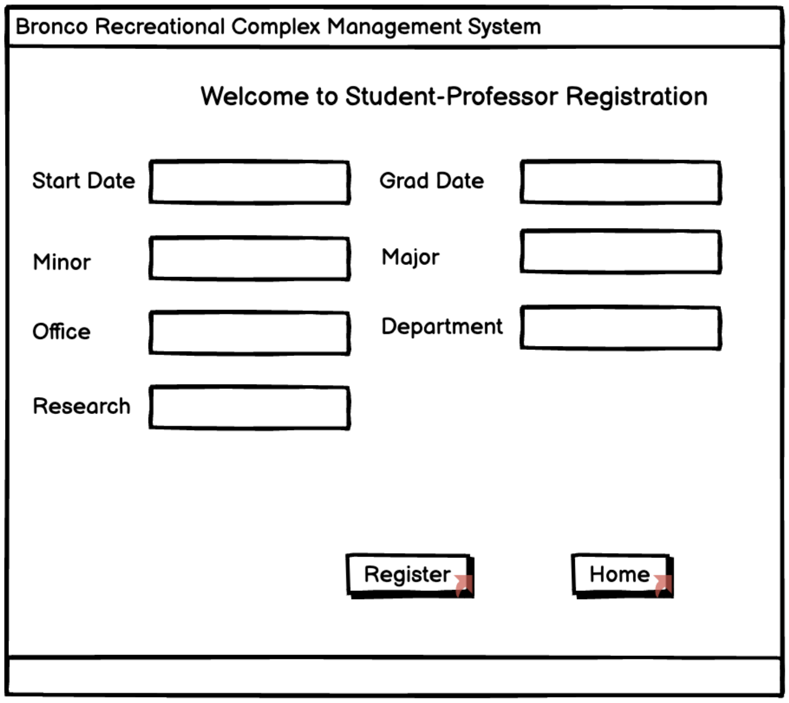

Bronco Recreational Complex Management System
Welcome to Student-Professor Registration
Start Date
Grad Date
Minor
Major
Office
Department
Research
Register
Home

Figure 9: Professor Student Registration

Screen_07:

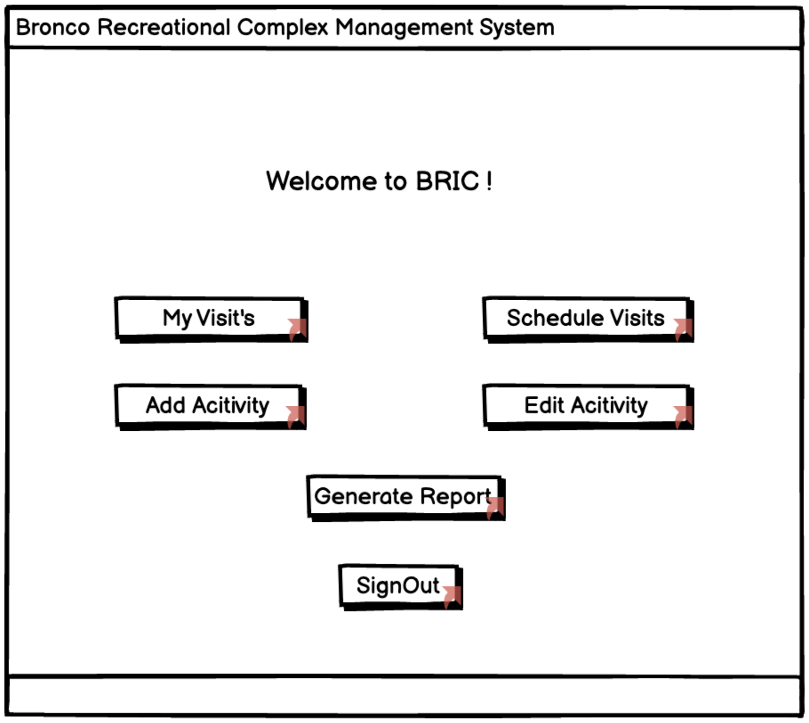
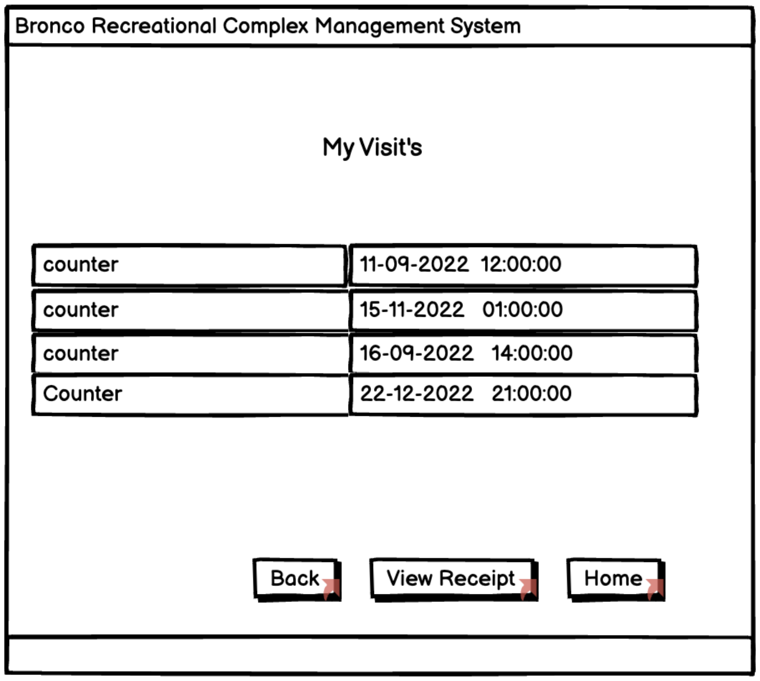


Figure 10: Home Screen

Screen_08:



counter	11-09-2022 12:00:00
counter	15-11-2022 01:00:00
counter	16-09-2022 14:00:00
Counter	22-12-2022 21:00:00

Figure 11: View Visits Screen

Screen_09:

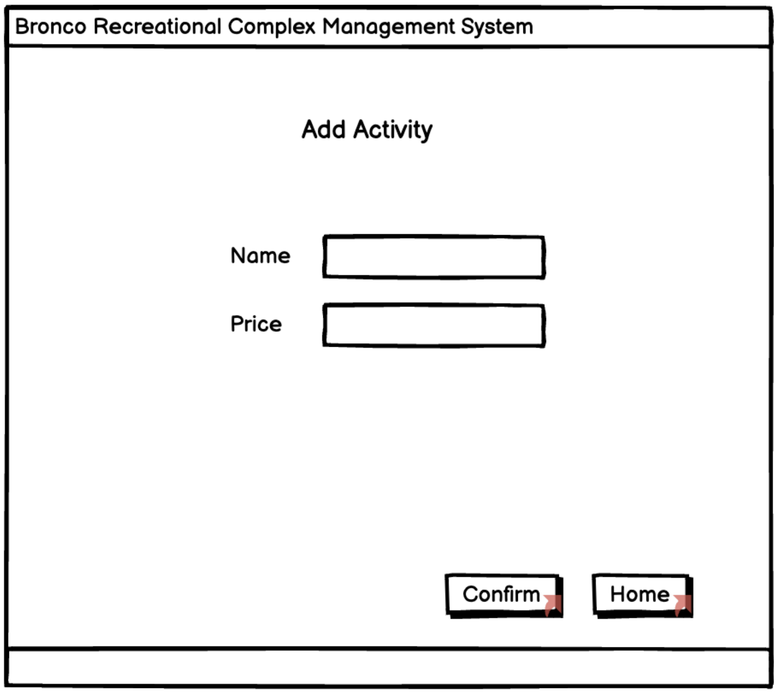


Figure 12: Add Activity Screen

Screen_10:

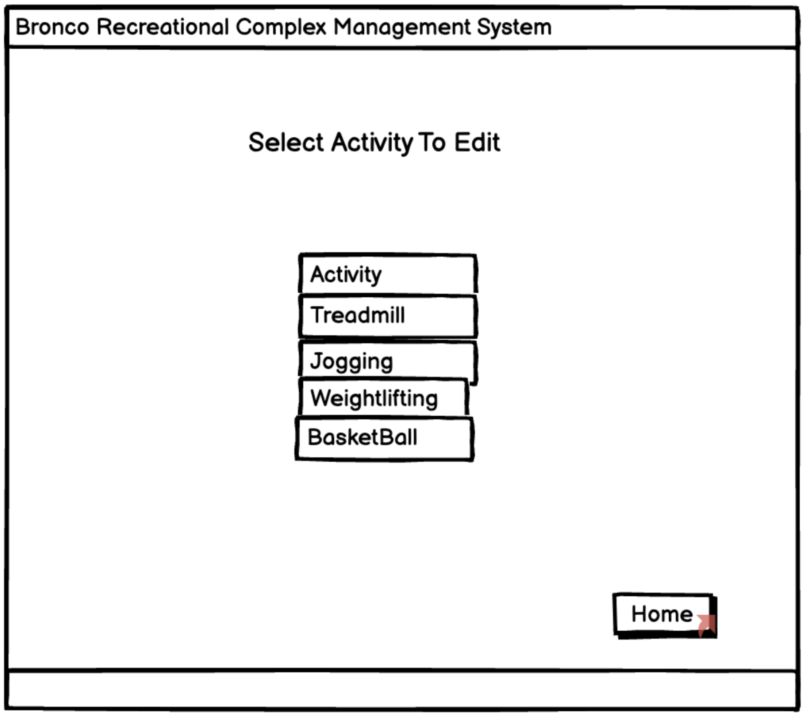
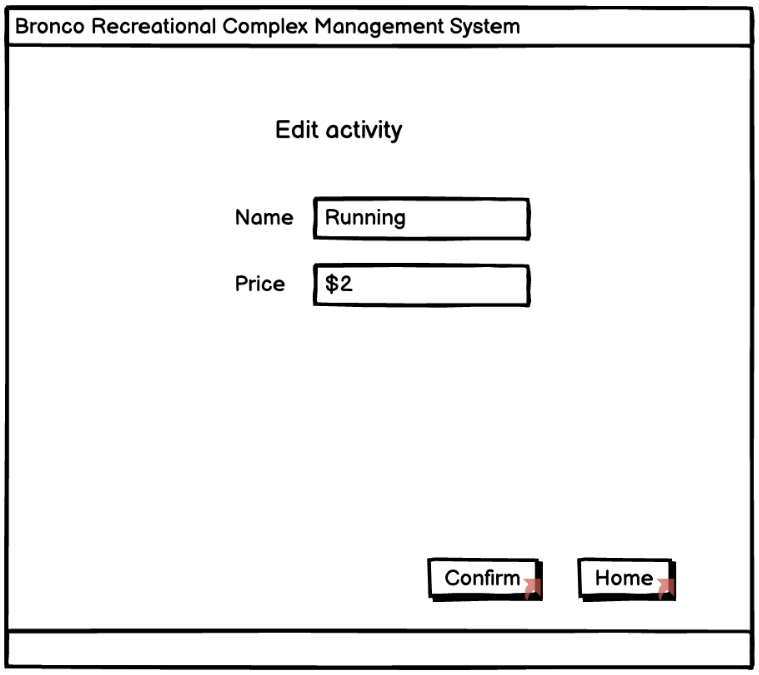


Figure 13: Select Activity to Edit Screen

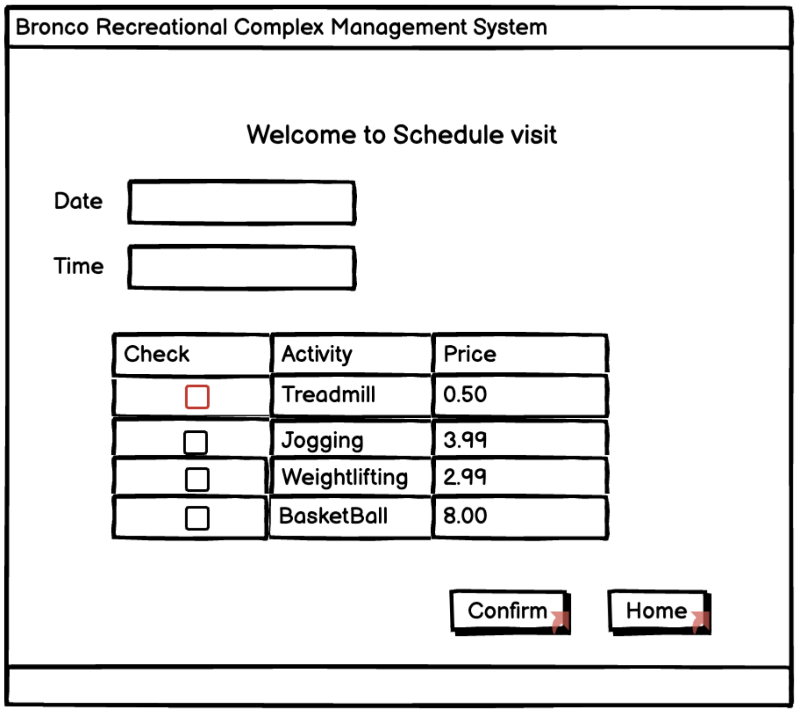
Screen_11:



Name	Running
Price	$2

Figure 14: Edit Activity Screen

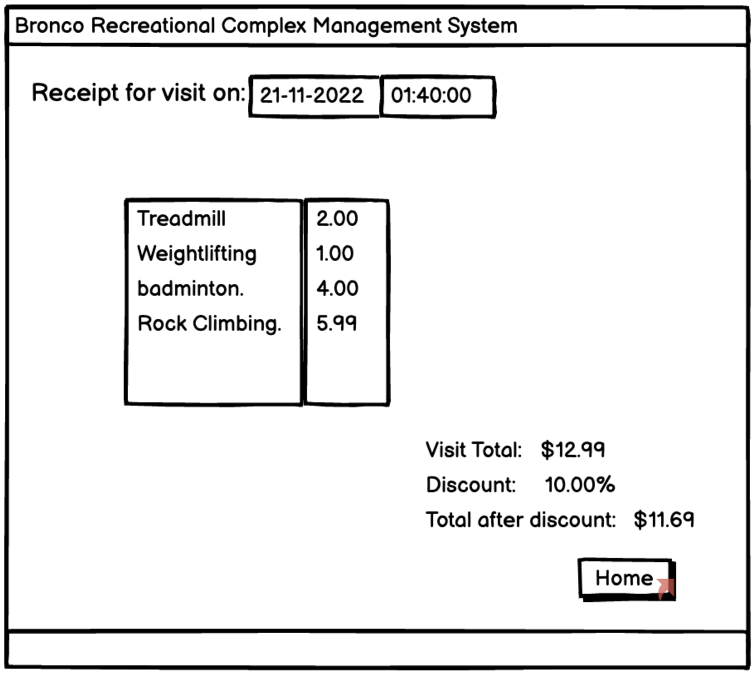
Screen_12:


Bronco Recreational Complex Management System
Welcome to Schedule visit
Date
Time
Confirm
Home

Check	Activity	Price
☐	Treadmill	0.50
☐	Jogging	3.99
☐	Weightlifting	2.99
☐	BasketBall	8.00

Figure 15: Schedule Visit Screen

Screen_13:


Bronco Recreational Complex Management System

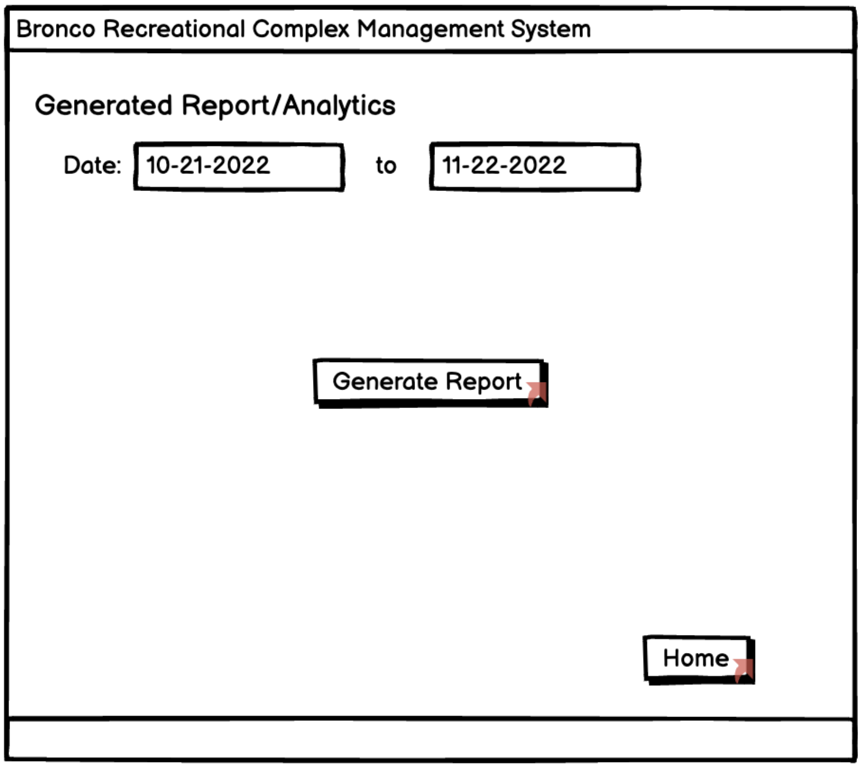
Receipt for visit on: 21-11-2022 01:40:00

Treadmill	2.00
Weightlifting	1.00
badminton.	4.00
Rock Climbing.	5.99

Visit Total: $12.99
Discount: 10.00%
Total after discount: $11.69

Figure 16: View Receipt

Screen_14:

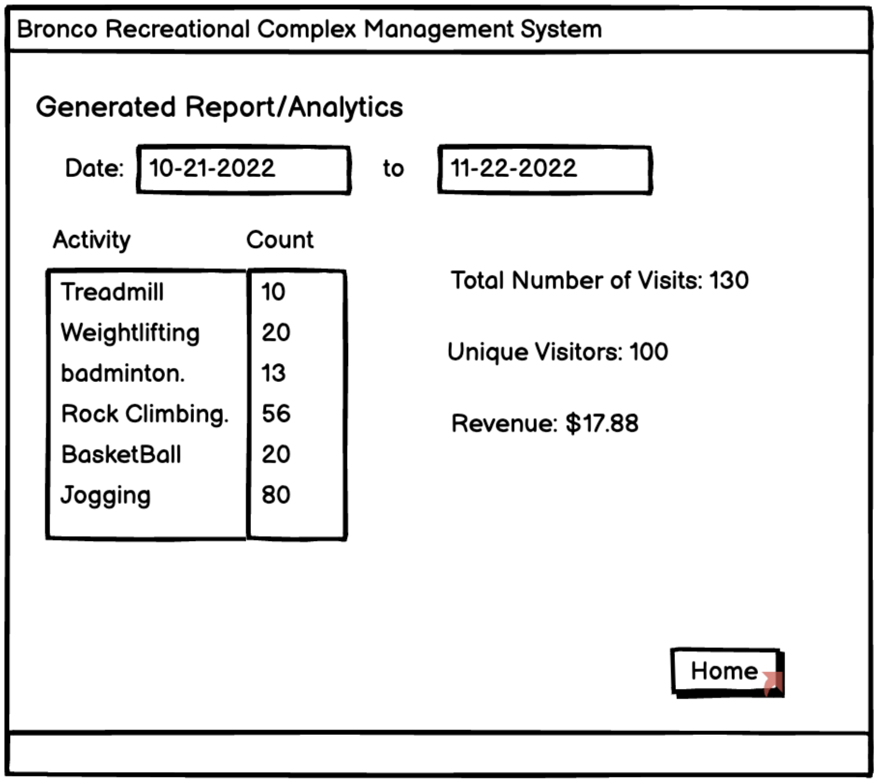

Bronco Recreational Complex Management System

Generated Report/Analytics

Date: 10-21-2022 to 11-22-2022

Figure 17: Generate Report Screen

Screen_15:



Generated Report/Analytics

Date: 10-21-2022 to 11-22-2022

Activity	Count
Treadmill	10
Weightlifting	20
badminton.	13
Rock Climbing.	56
BasketBall	20
Jogging	80

Total Number of Visits: 130

Unique Visitors: 100

Revenue: $17.88

Figure 18: Report Screen

Assumptions

- AS_01: All activities are available to all users at all times. Any amount of users can be assigned to an activity at once.
- AS_02: Payment is handled by a separate system.
- AS_03: All visitors will be students and or professors who have a valid broncoID number.
- AS_04: Security for the system will be handled by a third party.

Domain Properties

- A customer can only do an activity once per visit.
- The price of any activity can only be changed at the start of the day. Additionally an activity has the same price all day.

Functional Requirements

Customer

REQ_01 - Customer Registration

Rationel: A CUSTOMER is identified by their broncoID and have attributes:Name, Address, DOB, and phone number. Additionally a customer has a type: Student, Profesor, StudentProfessor

REQ_02 - Student

Rationel: A STUDENT has attributes: Enter Date, Grad Date, Major, Minor, Discount

REQ_03 - Professor

Rationel: A STUDENT has attributes: Department, Office, Research, Discount

REQ_04 - StudentProfessor

Rationel: A STUDENTPROFESSOR has attributes: Enter Date, Grad Date, Major, Minor, Department, Office, Research, Discount

Visit

REQ_05 - Visit Registration

Rationel: A VISIT is identified by a vistID and has attributes: Cost, Time, and Date.

REQ_06 - Customer Visit Relationship

Rationel: A CUSTOMER may have 0 or more VISIT. A VISIT can only belong to one customer.

Activity

REQ_07 - Activity Registration

Rationel: An ACTIVITY is identified by an activityID. An ACTIVITY has attributes: name and price.

REQ_08 - Activity Visit Relationship

Rationel: Each VISIT can have multiple ACTIVITY. An ACTIVITY can belong to 0 or more VISITS.

REQ_09 - Historical_Price

Rationel: A HISTORICAL_PRICE is identified by a historicalPriceID. A HISTORICAL PRICE has attributes: Date and Price.

REQ_10 - Historical Price Activity Relationship

An ACTIVITY can have 0 to many HISTORICAL_PRICES. A HISTORICAL PRICE can only belong to a single activity.

Non-Functional Requirements

NON_REQ_01 - Usability

Rationel: The system must be stable so it can fully replace the system as is.

NON_REQ_02 - Availability

Rationel: Customers should be able to use desktop and web applications to access the BRCM system.

NON_REQ_03 - Performance

Rationel: System should issue notifications in the event of an error or an exception.

NON_REQ_04 - Usability

Rationel: Customers should be able to readily understand UIs thanks to their user-friendly design.

NON_REQ_05 - Scalability

Rationel: The system should continue to work properly as the number of customers and activities registered changes.

NON_REQ_06 - Performance

Rationel: The system should be usable round-the-clock.

Developer Activities

- Multiple users cannot schedule for the same activity at the same time.
- An activity's price cannot be changed more than once per day.

Technologies

TRQ_01 - Java

Rationel: As requested by the customer the system will be implemented using java.

TRQ_02 - PostgreSQL

Rationel: PostgreSQL will allow for the creation of a fully featured persistent database.

TRQ_02 - Hibernate

Rationel: Hibernate is an ORM which allows for the easy storage of entire classes as a single row in a SQL database.

TRQ_03 - Thymeleaf templates

Rationel: Thymeleaf templates will handle the web view of the application.

TRQ_04 - JavaFX

Rationel: This GUI will allow for the creation of a desktop application, thus satisfying HL_01.

TRQ_05 - Spring Data JPA

Rationel: Provides annotations/methods to avoid boilerplate db code.

TRQ_06 - Spring Boot

Rationel: Dependency Injection and provides an easy implementation.

Traceability Matrix

Table 8: Traceability Matrix

Requirement	Design	Implementation	Test Case
REQ_01	User	Screen 3	TS_01, UT_01,UT_02,TS_02, UT_03, IT_01, IT_02, IT_03, IT_04, IT_05
REQ_05	Vist	Screen 12	TS_01,TS_02, UT_04, UT_05
REQ_07	Activity	Screen 12	TS_01,TS_02,UT_04, UT_05, UT_06, UT_07,
REQ_09	Historical Price	Screen 11	TS_01, TS_02
REQ_10	Report	Screen 15	TS_01, TS_03, UT_08

Design

Entity Relationship Diagram

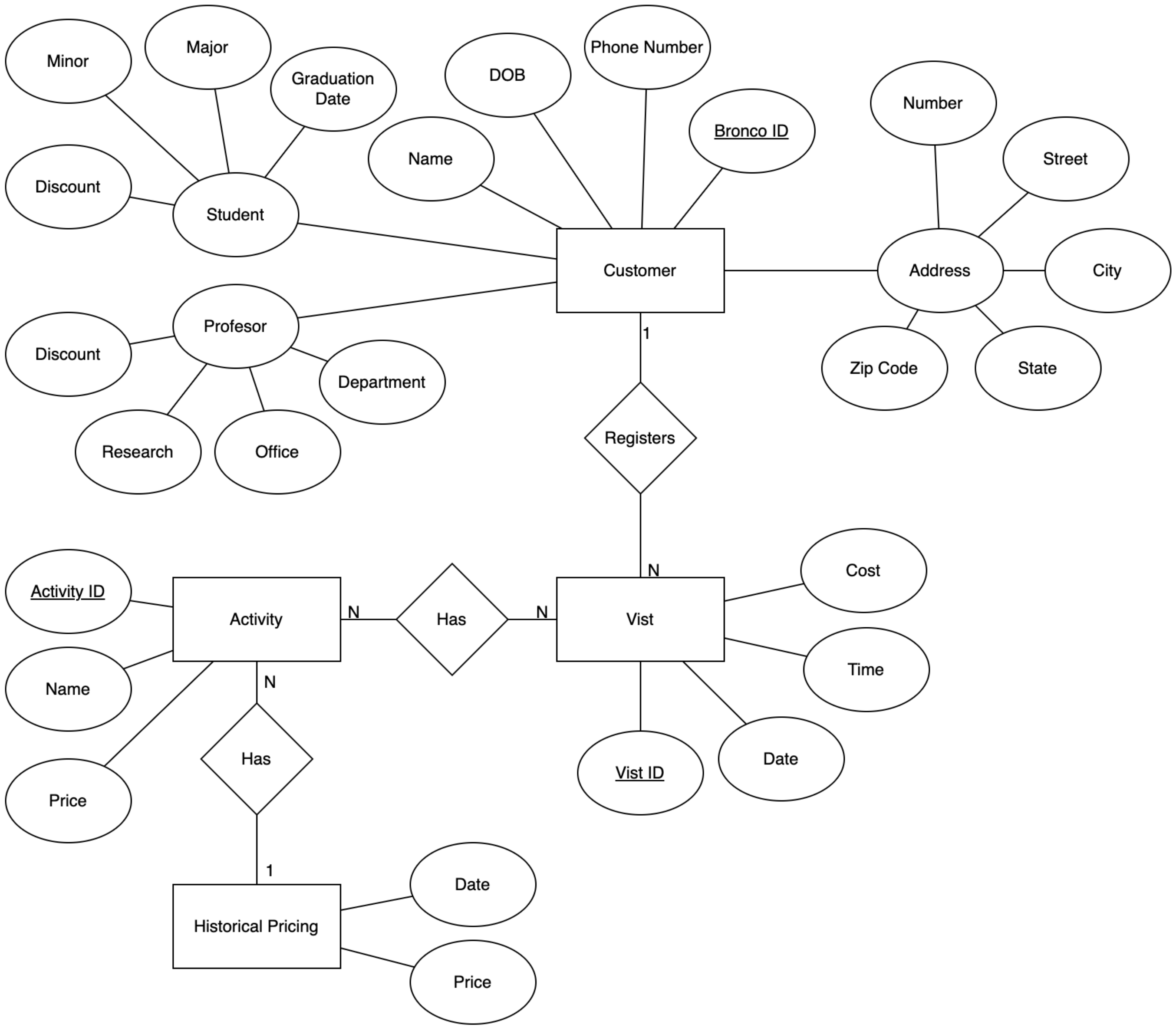


Figure 1: Entity Relationship Diagram

Data Logical Model

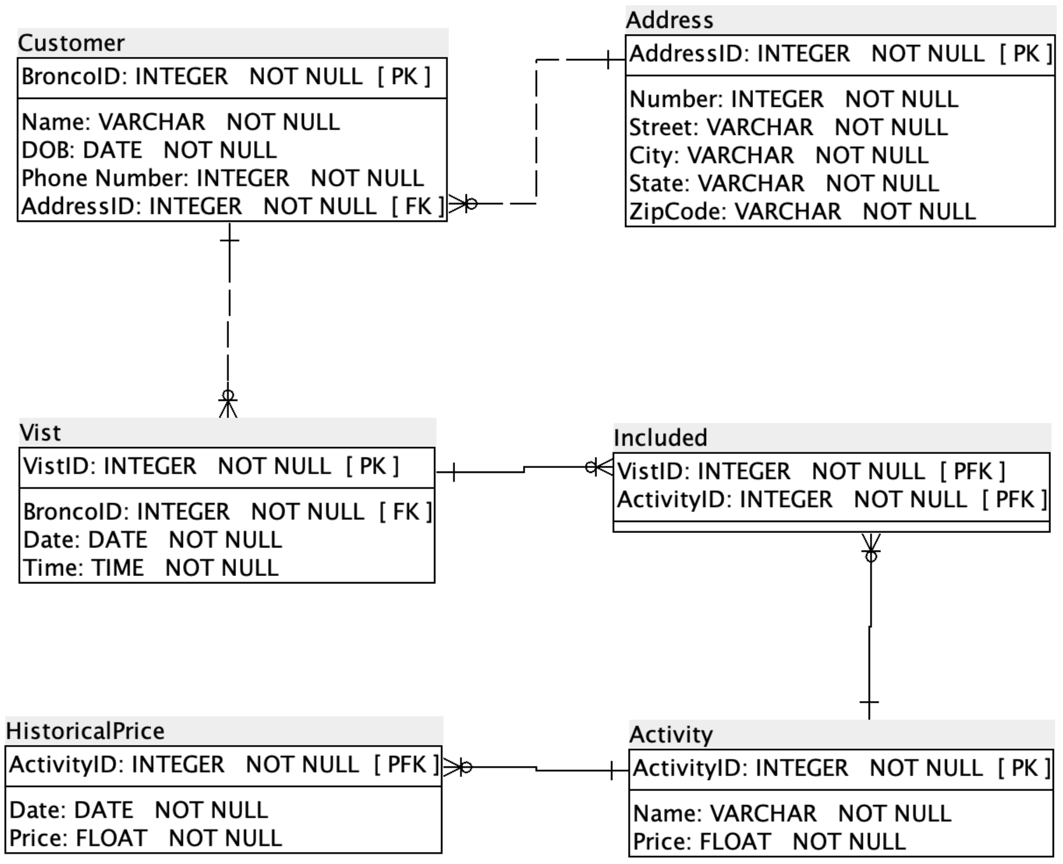

Customer
BroncoID: INTEGER NOT NULL [PK]
Name: VARCHAR NOT NULL
DOB: DATE NOT NULL
Phone Number: INTEGER NOT NULL
AddressID: INTEGER NOT NULL [FK]
Address
AddressID: INTEGER NOT NULL [PK]
Number: INTEGER NOT NULL
Street: VARCHAR NOT NULL
City: VARCHAR NOT NULL
State: VARCHAR NOT NULL
ZipCode: VARCHAR NOT NULL
Vist
VistID: INTEGER NOT NULL [PK]
BroncoID: INTEGER NOT NULL [FK]
Date: DATE NOT NULL
Time: TIME NOT NULL
Included
VistID: INTEGER NOT NULL [PFK]
ActivityID: INTEGER NOT NULL [PFK]
HistoricalPrice
ActivityID: INTEGER NOT NULL [PFK]
Date: DATE NOT NULL
Price: FLOAT NOT NULL
Activity
ActivityID: INTEGER NOT NULL [PK]
Name: VARCHAR NOT NULL
Price: FLOAT NOT NULL

Figure 2: Logical Model

System Architecture View and Style/Pattern

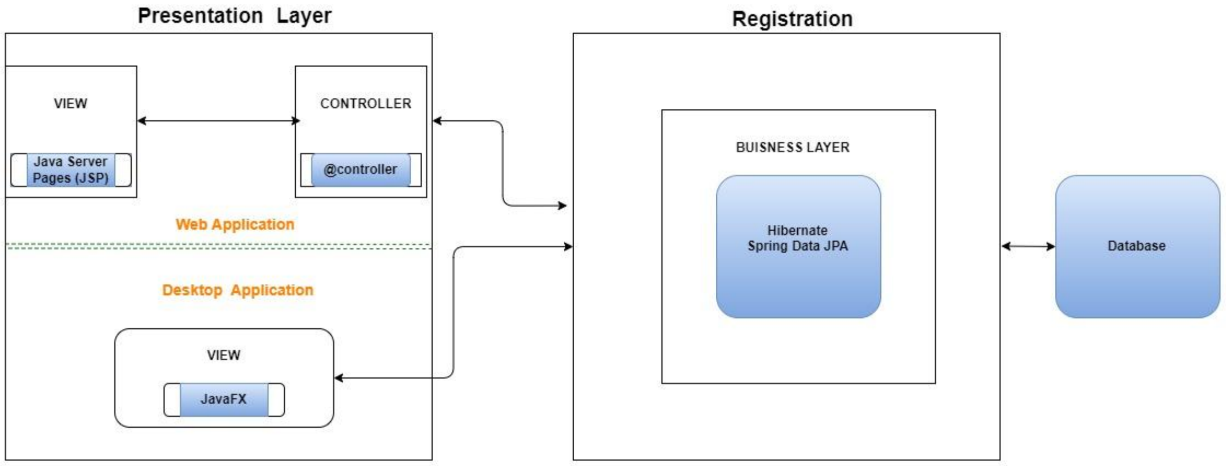


Figure 3: High Level Overview

Class Diagram

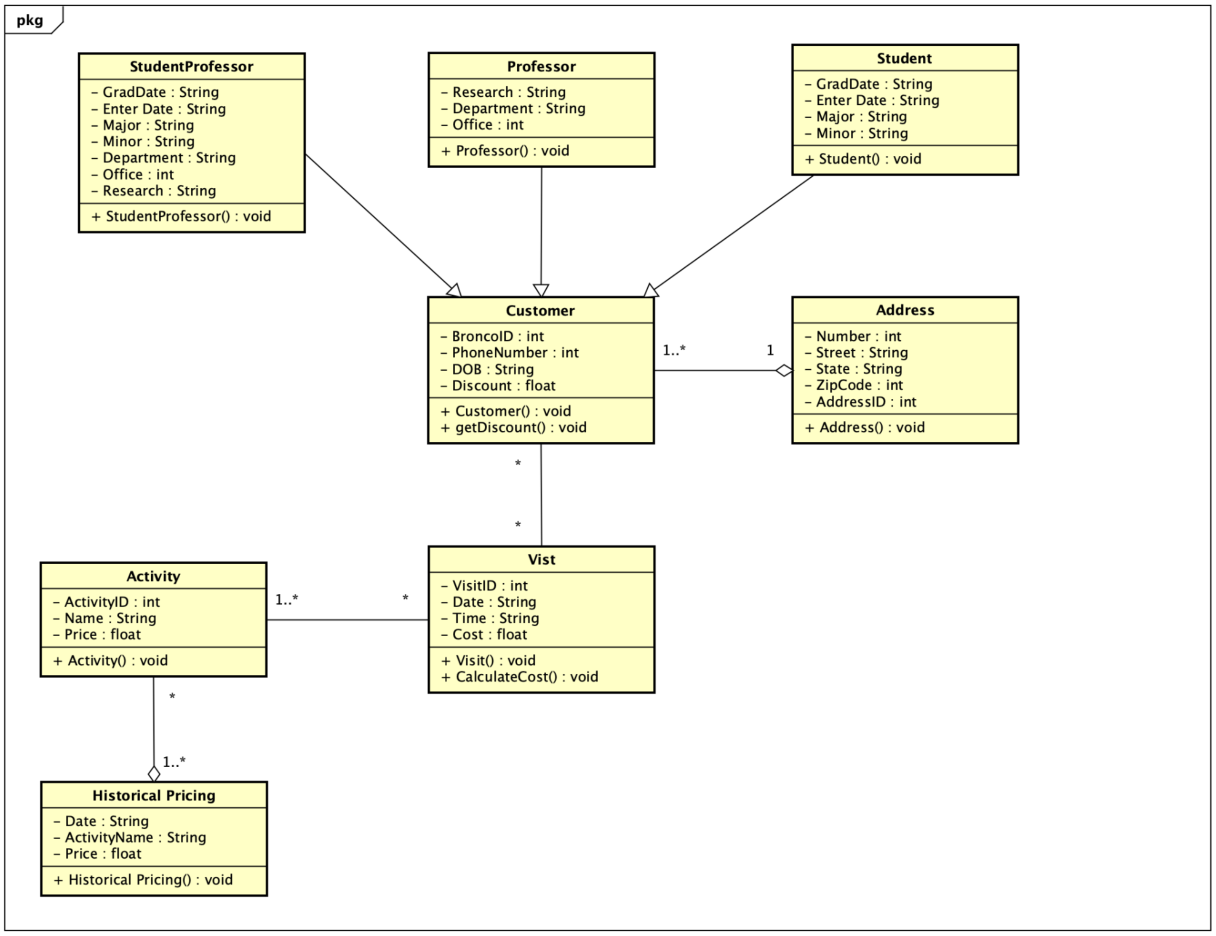


Figure 4: UML Diagram

Testing Plan

Test Scenarios:

Table 1: Test Scenario

Test Scenario		
ScenarioID	Title	Description
TS_01	Log In	Users are able to create an account from the desktop application. With their username they are then able to login to both the webview and desktop
TS_02	Schedule Visit	Users are able to schedule a visit from the desktop application after logging in. During scheduling they will select which activities
TS_03	Generate Receipt	Users are able to print a receipt from the desktop application after logging in. The user is able to view a list of past visits and may select any visit to generate the receipt.
TS_04	Generate Report	Users are able to generate an earnings report for all visits between two given days.

Unit Tests

Table 2: UT_01

Test Scenario ID	TS_01
Test Scenario	Log In
Test Case ID	UT_01
Test Case	Sign up for a new account
Pre - Conditions	Be at the login screen
Instructions	1. User clicks new-user. 2. User enters all fields. 3. User selects which customer type they are. 4. User enter the remaining required information.

	5. User submits
Expected Outputs	User now has an account created and is sent back to the login screen.
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 3: UT_02

Test Scenario ID	TS_01
Test Scenario	Log In
Test Case ID	UT_02
Test Case	Successful Login
Pre - Conditions	1. Be at the login screen. 2. User has an account.
Instructions	1. User enters username. 2. User selects login.
Expected Outputs	User logs in and is sent to the home page.
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 4: UT_03

Test Scenario ID	TS_01
Test Scenario	Log In
Test Case ID	UT_03
Test Case	Invalid Credential Login

Pre - Conditions	Be at the login screen.
Instructions	1. Enter invalid username. 2. Select Login.
Expected Outputs	User is told they enter invalid credentials.
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 5: UT_04

Test Scenario ID	TS_02
Test Scenario	Schedule Visit
Test Case ID	UT_04
Test Case	User schedules a visit
Pre - Conditions	User is registered and logged in
Instructions	1. User is at Schedule visit screen 2. User chooses Date and Time to schedule an activity. 3. User checks the Activity(s) to be reserved during his schedule. 4. User presses Confirm. 5. Receipt page is shown with details of activities chosen and total amount 6. User presses Home page 7. User is sent to Home page
Expected Outputs	Visit is scheduled
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_05, REQ_06, REQ_07, REQ_08, REQ_09

Table 6: UT_05

Test Scenario ID	TS_02
Test Scenario	Schedule Visit
Test Case ID	UT_05
Test Case	User views activities but does not schedule one
Pre - Conditions	User is registered and logged in
Instructions	1. User is at Schedule visit screen 2. User chooses Date and Time to schedule an activity. 3. User views available Activity(s) during that date and time. 4. User presses Home 5. User is sent to Home page
Expected Outputs	User views activities
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_05, REQ_08

Table 7: UT_06

Test Scenario ID	TS_03
Test Scenario	Generate Receipt
Test Case ID	UT_06
Test Case	View Visits
Pre - Conditions	1. User is logged in at Home screen. 2. User has made at least one visit.
Instructions	User selects view visits.
Expected Outputs	User is shown a list of past visits.
Type of Implementation	MANUAL

Phase	Implementation
Dependencies	
Requirement Tested	REQ_07, REQ_9, REQ_10

Table 8: UT_07

Test Scenario ID	TS_03
Test Scenario	Generate Receipt
Test Case ID	UT_07
Test Case	Generate a receipt for a given visit.
Pre - Conditions	1. User is logged in. 2. User is at the Home page. 3. User have made visits in the past.
Instructions	1. User selects view visits. 2. User chooses a visit from the list. 3. User selects a visit. 4. User selects generate receipt.
Expected Outputs	User is shown a receipt for the selected visit.
Type of Implementation	MANUEL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_07, REQ_9, REQ_10

Table 9: UT_08

Test Scenario ID	TS_04
Test Scenario	Generate Report
Test Case ID	UT_08
Test Case	Generate a report
Pre - Conditions	1. User is logged in. 2. User is at the home screen.

Instructions	User clicks generate Report.
Expected Outputs	
Type of Implementation	MANUEL
Phase	Implementation
Dependencies	
Requirement Tested	

Integration Tests

Table 10: IT_01

Test Scenario ID	TS_01
Test Scenario	Log In
Test Case ID	IT_01
Test Case	Creating a professor account
Pre - Conditions	1. User is at Home page
Instructions	1. User clicks on the New User button 2. User is sent to the New User window 3. User fills in the fields. 4. User clicks on select based on profession: Professor 5. User is sent to the Professor Register window 6. User fills out appropriate information for Professors 7. User clicks Register 8. User is redirected to the Home page. 9. User clicks Login.
Expected Outputs	1. User is successfully registered and logged in as Professor
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 11: IT_02

Test Scenario ID	TS_01
Test Scenario	Log In
Test Case ID	IT_02
Test Case	Creating a student account
Pre - Conditions	1. User is at Home page
Instructions	1. User clicks on the New User button 2. User is sent to the New User window 3. User fills in the fields. 4. User clicks on select based on profession: Student 5. User is sent to the Student Register window 6. User fills out appropriate information for Student 7. User clicks Register 8. User is redirected to the Home page. 9. User clicks Login.
Expected Outputs	1. User is successfully registered and logged in as Student
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 12: IT_03

Test Scenario ID	TS_01
Test Scenario	Log In
Test Case ID	IT_03
Test Case	Creating a student and professor account

Pre - Conditions	1. User is at Home page
Instructions	1. User clicks on the New User button 2. User is sent to the New User window 3. User fills in the fields. 4. User clicks on select based on profession: StudentProf 5. User is sent to the Student-Professor Register window 6. User fills out appropriate information for Student-Professor 7. User clicks Register 8. User is redirected to the Home page. 9. User clicks Login.
Expected Outputs	1. User is successfully registered and logged in as Student-Professor
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 13: IT_04

Test Scenario ID	TS_01
Test Scenario	Log In
Test Case ID	IT_04
Test Case	Creating a student and professor account successfully
Pre - Conditions	User is at Home page
Instructions	1. User clicks on the New User button 2. User is sent to the New User window 3. User fills in the fields. 4. User clicks on select based on profession: StudentProf 5. User is sent to the Student-Professor Register window 6. User fills out appropriate information for Student-Professor 7. User clicks Register 8. User is redirected to the Home page. 9. User clicks Login.

Expected Outputs	User is successfully registered and logged in as Student-Professor
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 14: IT_05

Test Scenario ID	TS_02
Test Scenario	Log In
Test Case ID	IT_05
Test Case	Creating a student and professor account unsuccessfully
Pre - Conditions	User is at Home page
Instructions	1. User clicks on the New User button 2. User is sent to the New User window 3. User fills in the fields. 4. User clicks on select based on profession: StudentProf 5. User is sent to the Student-Professor Register window 6. User fills out wrong information for Student-Professor 7. User clicks Register 8. User is redirected to the Home page. 9. User clicks Login.
Expected Outputs	User is unsuccessfully registered and logged in as Student-Professor. Error message is generated
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

System Tests

Table 15: ST_01

Test Scenario ID	TS_02
Test Scenario	Schedule Visit
Test Case ID	ST_01
Test Case	Schedule visit and choose activity
Pre - Conditions	User is logged in
Instructions	1. The user clicks on “Schedule Visit” 2. User enters the date for the visit 3. User enters the time of the visit 4. User verifies that the name displayed is correct 5. User checks the activities that they want to participate in from the list of activities 6. User clicks “Submit” 7. User is sent back to the “Home Screen”
Expected Outputs	User should be sent to the account home screen
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 16 :ST_02

Test Scenario ID	TS_02
Test Scenario	Generate Receipt
Test Case ID	ST_02
Test Case	User views their visits and receipts
Pre - Conditions	The user has scheduled a visit and is on the home screen

Instructions	1. The user clicks on My Visits 2. The user clicks on one of the dates for a visit 3. User clicks View Receipt
Expected Outputs	1. A receipt for the visit should be displayed
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Table 17: ST_03

Test Scenario ID	TS_02
Test Scenario	Generate Receipt
Test Case ID	ST_03
Test Case	User successfully creates an online reservation
Pre - Conditions	User is logged in the system
Instructions	1. User clicks Create reservation 2. User chooses a date and time for the reservation 3. User clicks on the activity to participate in 4. User clicks confirm
Expected Outputs	Reservation created successfully
Type of Implementation	MANUAL
Phase	Implementation
Dependencies	
Requirement Tested	REQ_01, REQ_02, REQ_03, REQ_04

Discussion

The Bronco Recreation Complex Management (BRCM) project has helped us apply the knowledge we gained in CS5800, Advanced Software Engineering. As a group we were able to use our knowledge of software life cycle, team organization, software testing, and software system design. While our group was unsuccessful in developing a fully functional application the knowledge gained and lessons learned were immense.

In the project plan we were able to successfully divide up into roles, choose a development strategy, and create a plan. Our group chose to go with a scrum development strategy. This was chosen so we could all play to our strengths through the entirety of the project. We held scrum meetings approximately twice a week. These meetings were very helpful as it allowed us to move resources to various aspects of the project as needed. Requirements specification was handled as a team. By analyzing the provided UoD we were able to develop a clear list of high level goal, function and non functional requirements, and plan a high level overview of how the completed system would work with the selected technologies. After developing the requirement specifications the team moved on to the design stage. In the design stage the entity relationship diagram, UML diagram, and logical view models were all planned. This acted as a reinforcement for all the object-oriented programming that has been discussed in class. After the design stage we were able to begin implementation.

During the implementation phase is where our group encountered issues. Rather than using a traditional Java Servlet approach, it attempted to implement the system as a spring boot application with a JSP webview, and JavaFX desktop. Hibernate and postgreSQL would handle data access and management. JavaFX and Hibernate/postgreSQL were selected to be used as it was what was discussed in class. Spring Boot was also used because of the recommendation of a team member, with all others agreeing after discussing. However this ultimately led to critical compatibility issues. We were unable to integrate both JavaFX and Spring Boot. Due to the late discovery of this issue we were unable to sufficiently create our application. Given the opportunity to before this project again a more traditional set of technologies would have been used. Wrapping up the Project, we recovered by splitting our project into 2 views. Here, we worked on developing the web differently from the desktop version. For this, the web version used the FXML as a controller and used Hibernate to connect to the database. For the desktop version, JavaFX would request and receive the instruction to the controller.
Next, for the web view, we linked the database with the interface which implements the CRUDRepository<entity_class, id>interface. Then, the controller which is connected with the browser also communicates with the service interface which is also connected to the repository which now communicates with the CRUDRepo. Through this process , we recovered and created two separate but well working systems. Using components that included a navigable JavaFX GUI, a thymeleaf webview to see customer information, and more. However many of these choices

were to allow for a simple demonstration and are not ready for delivery. Despite the failure to complete the implementation, we do believe our plan would have worked had we chosen a different set of technologies.Given the opportunity to do this project again a more traditional set of technologies would have been used. Additionally as a future plan, we will also try to come up with a solution to the same problem we faced.

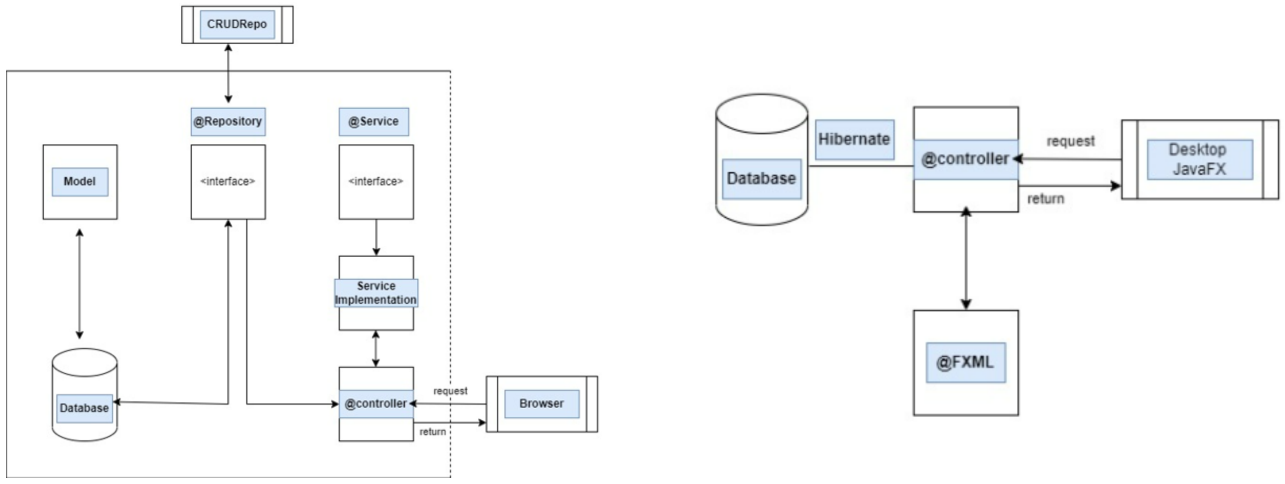


Figure 1: Updated View for recovery