



**Department of Electrical,
Computer, & Biomedical Engineering**
Faculty of Engineering & Architectural Science

Course Title	Object Oriented Eng Analysis and Design
Course Number	COE528
Semester/Year	W2022

Instructor	Prof. Olivia Das
------------	------------------

Assignment/Lab Number	Final Project
Assignment/Lab Title	Bookstore Application

Submission Date	April 3 rd , 2022
Due Date	April 3 rd , 2022





Student LAST Name	Student FIRST Name	Student Number	Section	Signature
Sandhu	Sukhroop	501038466	1	
Kelsey	Charles	501022515	1	
Debnath	Prayash	501041739	1	
Ly	Alan	500974484	1	

Table of Contents

Introduction	2
Objectives	2
Implementation	2
Use-Case Diagram	2
State Design Pattern	4

Introduction

The report for the bookstore application final project in the COE 528 course is presented below. This project took four weeks to complete and utilized concepts that are listed in the objectives section.

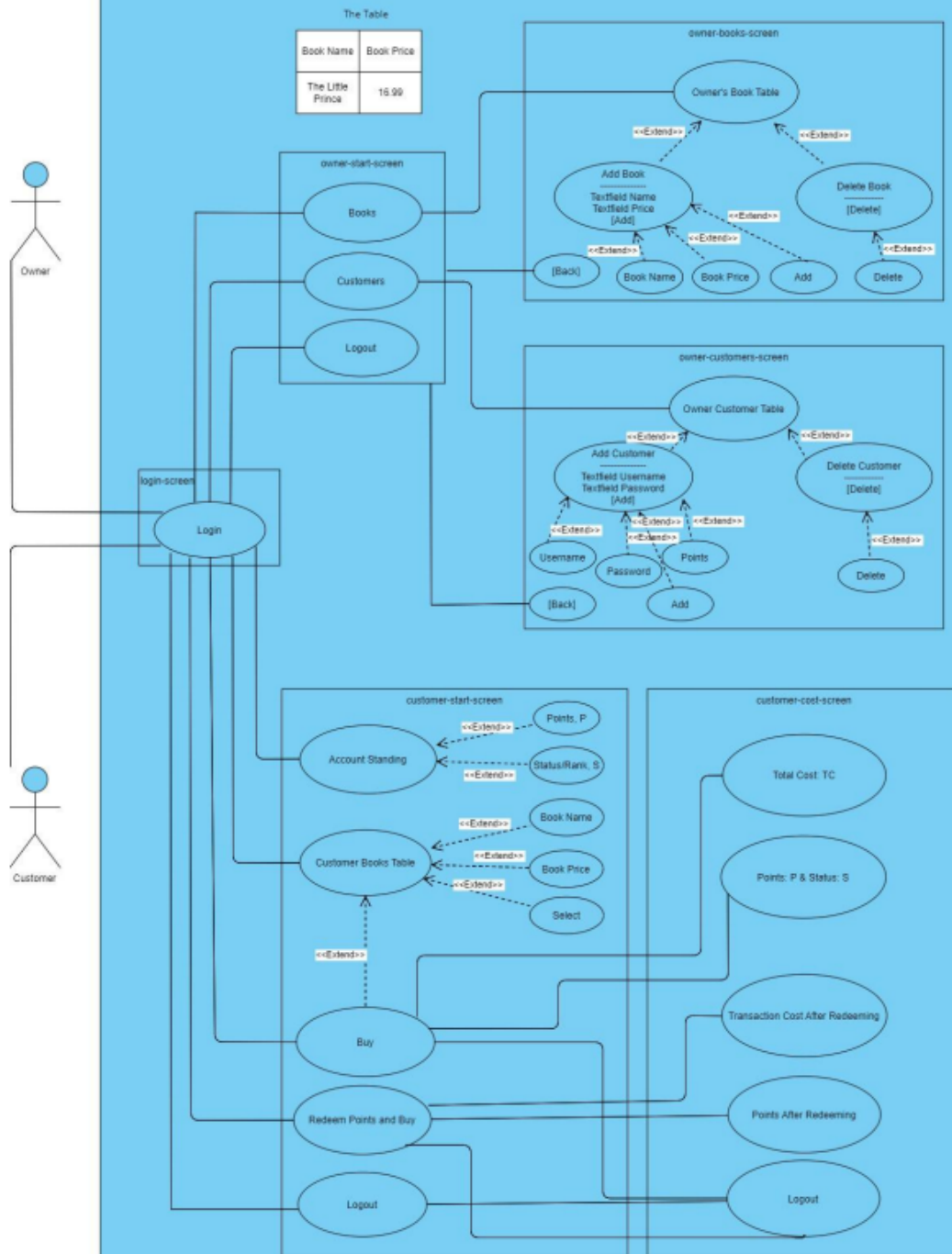
Objectives

The purpose of this project is to create a bookstore application that uses concepts such as use case diagrams, UML diagrams, state design patterns, and the implementation of the code using Java and JavaFX. The application must have two different points of access which are the owner of the store and the customer. Respectively, each point of access will have different options and menus pertaining to its role.

Implementation

Use-Case Diagram

The use-case is used as they help explain how the system as a whole should function without directly implementing it. In addition, the use-case acts as the first step in the brainstorming process to map out and analyze the problem. When presenting the problem, it can be split into multiple different parts that allow the implementation of each individual part to become simpler. In this case, for example, it helps indicate how the owner and customer sides of the application should behave separately from each other. From *Figure 1* below, it can be seen how the relationships are made. The owner and customer are both related to the logic screen but from there, they each have their own screens which are the owner-start-screen and customer-start-screen.

**Figure 1. Use-Case Diagram**

State Design Pattern

The purpose of the State Design Pattern is to help allow objects to alter their behaviour based on the state it is currently in or when their internal state changes. This change is something that must happen at the run-time, and it is the object that will appear to change its class. The state object makes the change in states, however, the current class is unchanged. Also, a state or class cannot be changed by itself. When looking at our application, there are two states for the user, with one being the manager/owner and the other being the customer. Each state is implemented into its own class using the state pattern. The User class behaves like the client whereas the GUI behaves like the context. This means that there would be two main states that each have numerous sub-states. The Owner state would be one of the initial states that contain 3 sub-states which are Books, Customers, and Logout. Accessing the Books state would present the user with a list of all available books and the ability to add or remove books. The Customer's state would present the user with a list of all existing customers and the ability to add new customers or remove existing ones. The Logout state would return the user to the login screen. The Customer state also has 3 sub-states which are Buy, Redeem points and Buy, and Logout. Similar to the Owner sub-states, the Logout state returns the user to the login screen. The Buy state presents the user with the total cost, current status, and how many points the user has. The last sub-state is the Redeem points and Buy state which will prompt a screen similar to the Buy State, but this time, including a discount. As shown in *Figure 2*, the use of the state design pattern helps to show that the manager and customer have different functionalities.

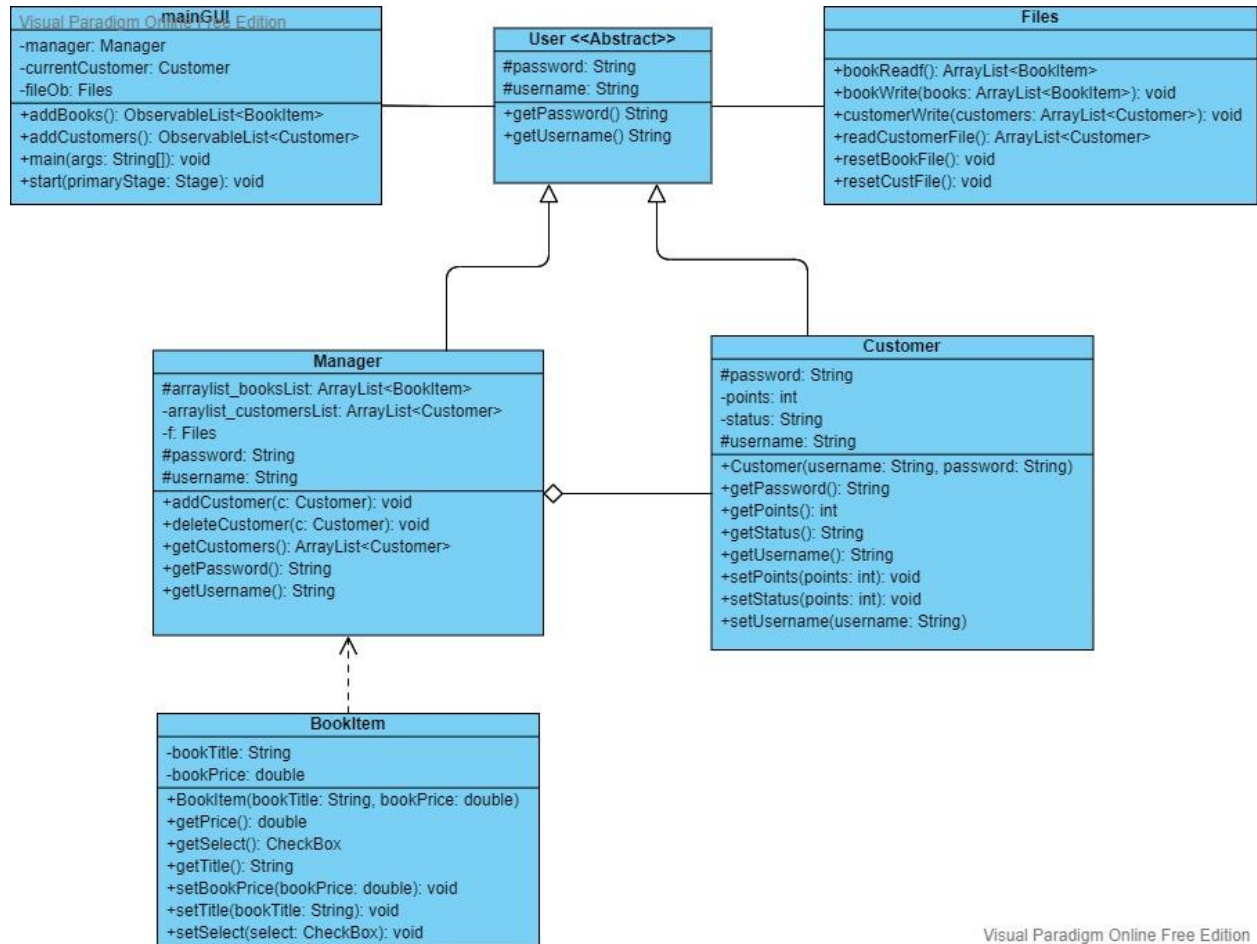


Figure 2. State Design Pattern