

**CEN 308 SOFTWARE ENGINEERING**

PROJECT DOCUMENTATION

Aces – Clothing Store

Prepared by:

**Anes Mehović**

Proposed to:

**Nermina Durmić, Assist. Prof. Dr.**

**Aldin Kovačević, Teaching Assistant**

Date of submission

24.06.2022

Contents

[1. Introduction 3](#_Toc106988962)

[1.1. About the Project 3](#_Toc106988963)

[1.2. Project Functionalities and Screenshots 3](#_Toc106988964)

[2. Project Structure 8](#_Toc106988965)

[2.1. Technologies 8](#_Toc106988966)

[2.2. Database Entities 9](#_Toc106988967)

[2.3. Architectural Pattern 10](#_Toc106988968)

[2.4. Design Patterns 10](#_Toc106988969)

[3. Conclusion 10](#_Toc106988970)

# 

# 1. Introduction

## 1.1. About the Project

The project that was worked one was an online clothing store that provides multiple arrays of clothing options to its users, rather than providing only one clothing brand or clothing item type. This would help users in a way that it allows the user to minimize the time needed when shopping for new clothes, rather than interacting with multiple clothing sites and not being able to find any sort of clothing brand. This would take away from the daily time that a user might have spent on doing something else. The user can just browse the site logged out, however they can create an account if they do not have one and proceed to purchase an item of their choice. They can edit their profiles also if they find that some information that they have provided is lacking and could use improvement.

## 1.2. Project Functionalities and Screenshots

- **Profile management:**

- Registration

- Login

- Reset Password

- Edit current profile:

- Change current name

- Change current surname

- Change current email

- Change current password

- Change current image

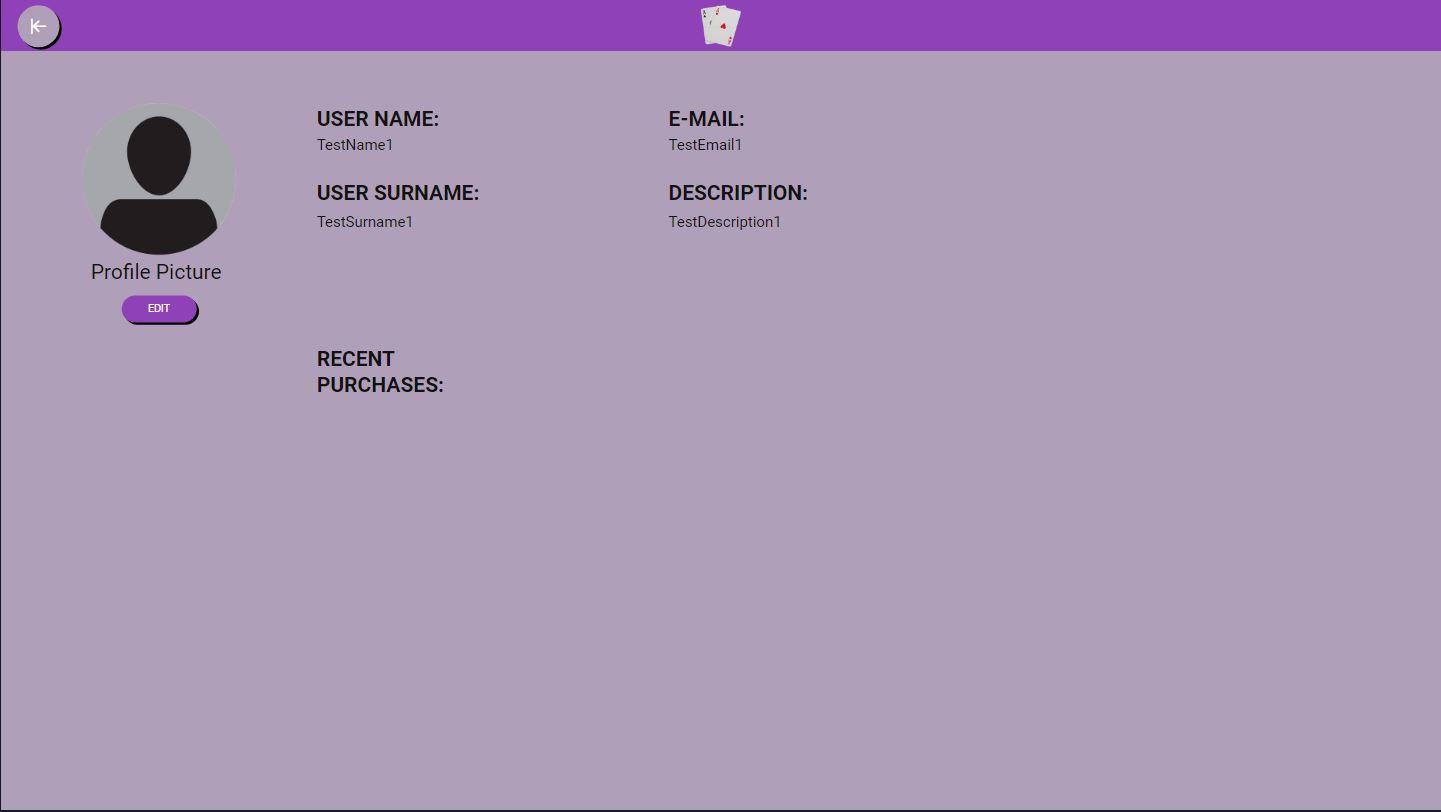
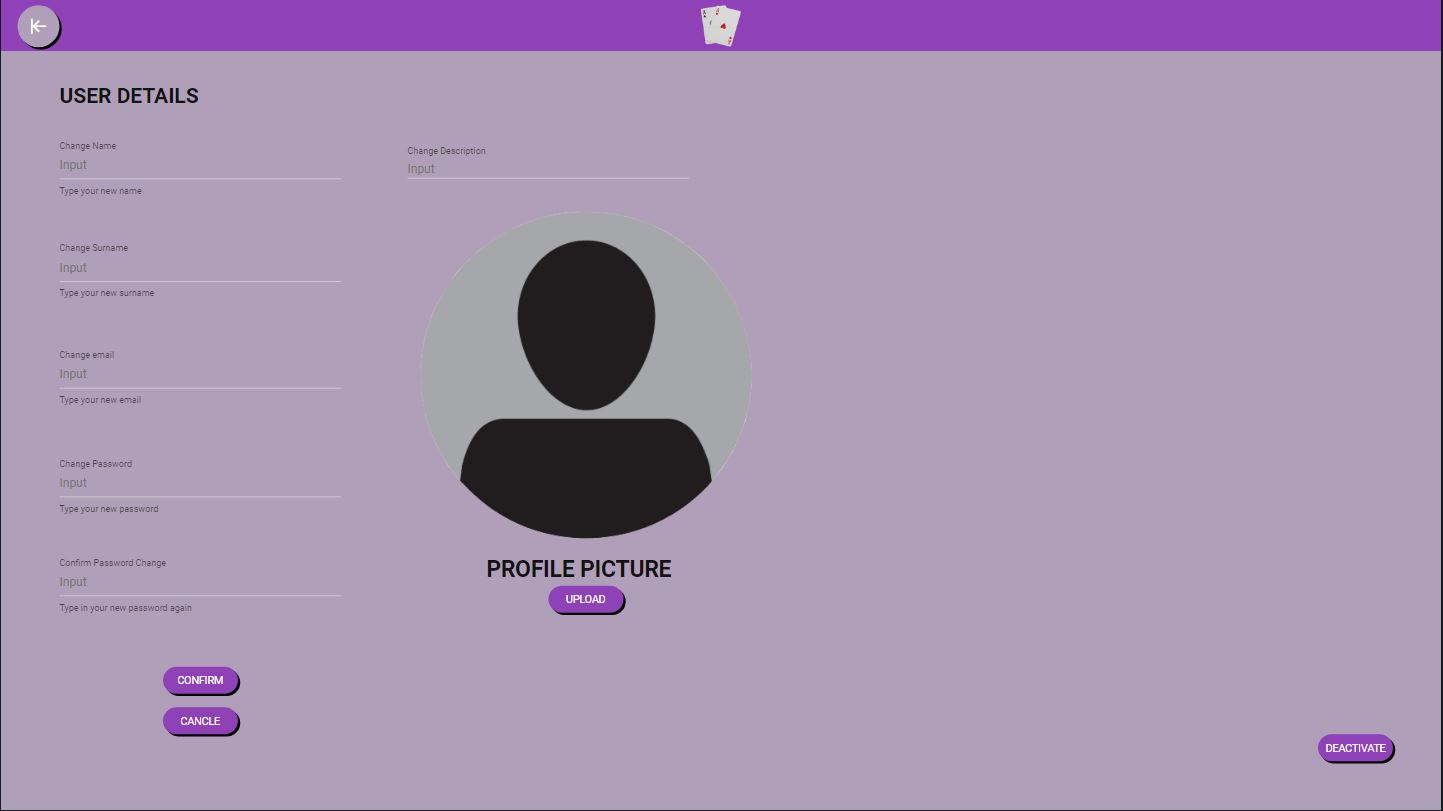
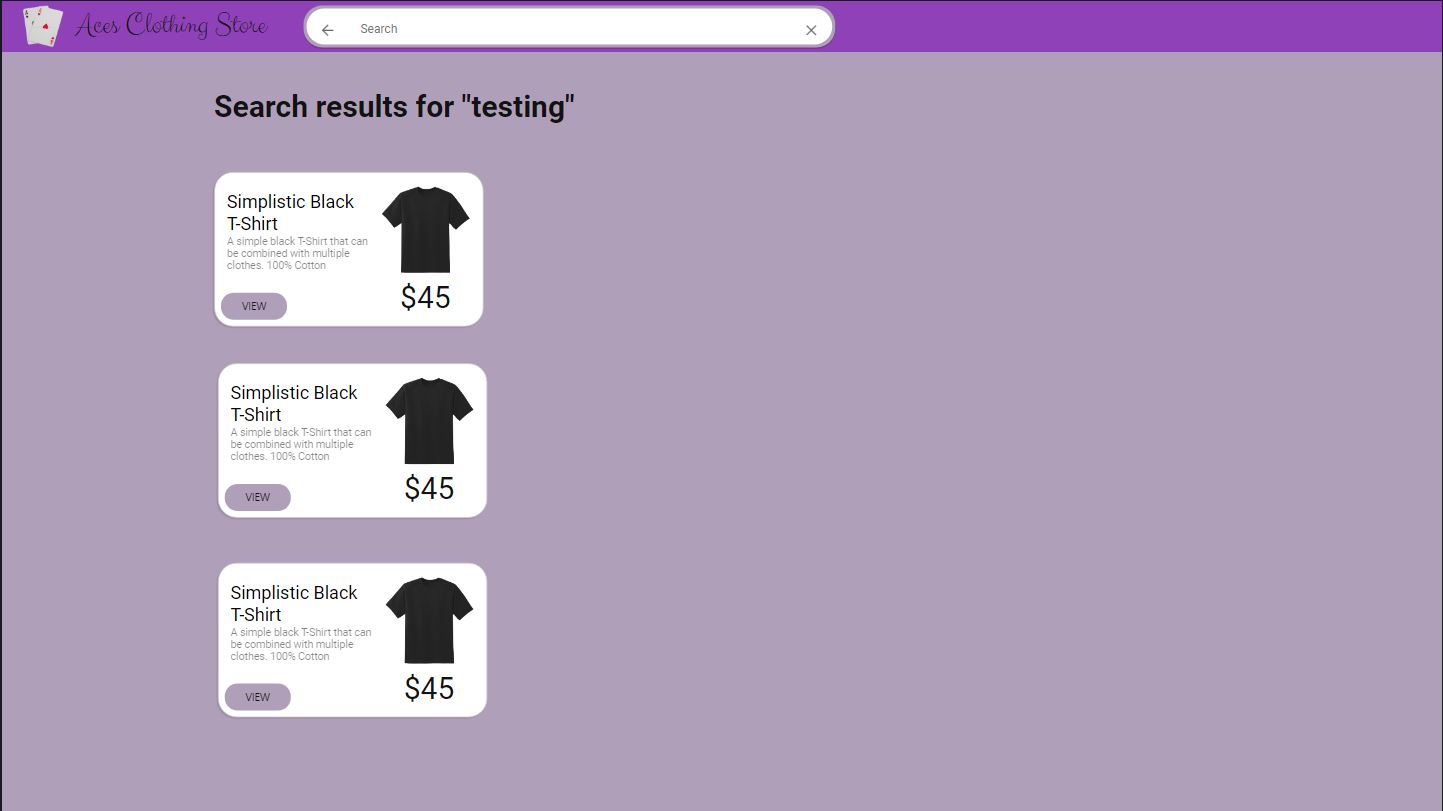
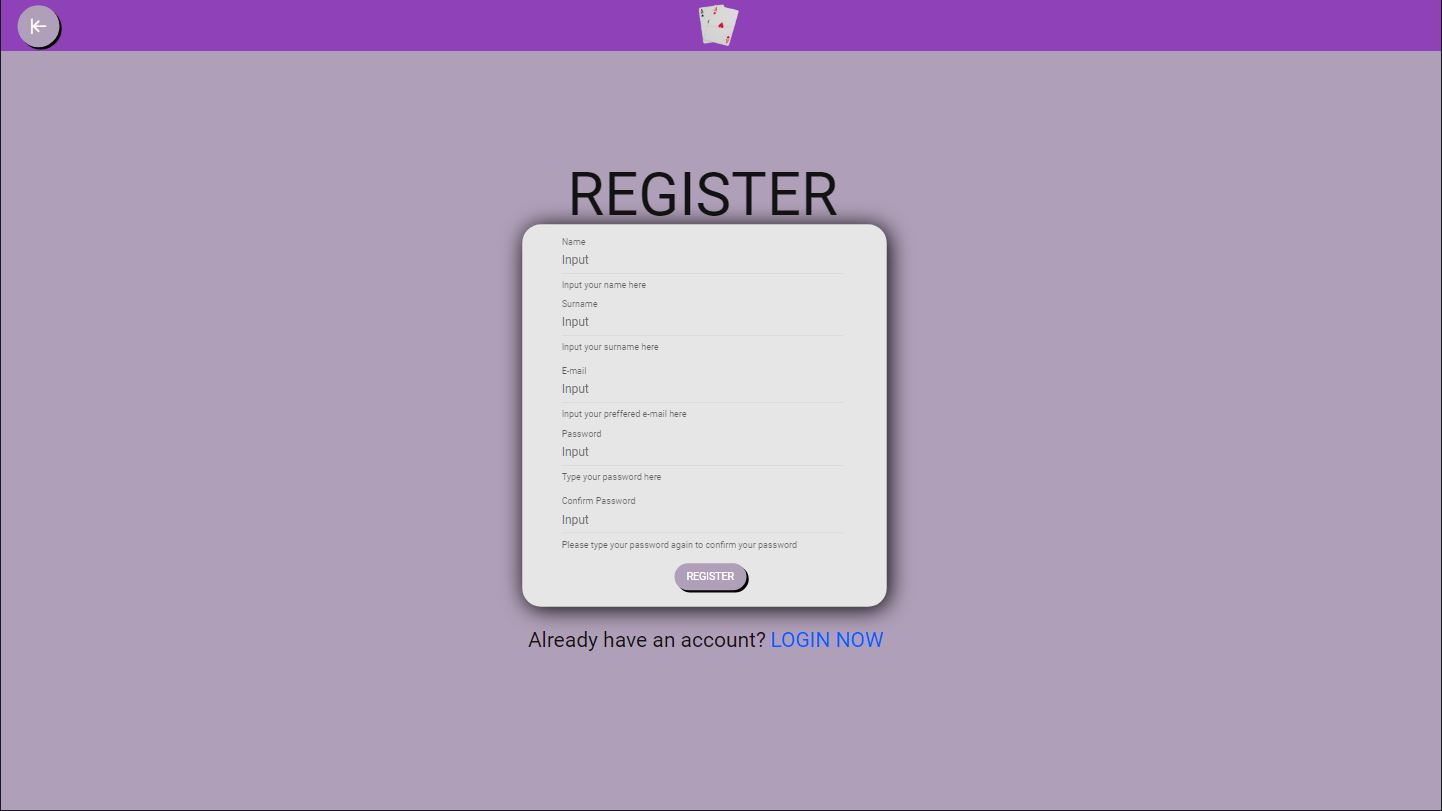
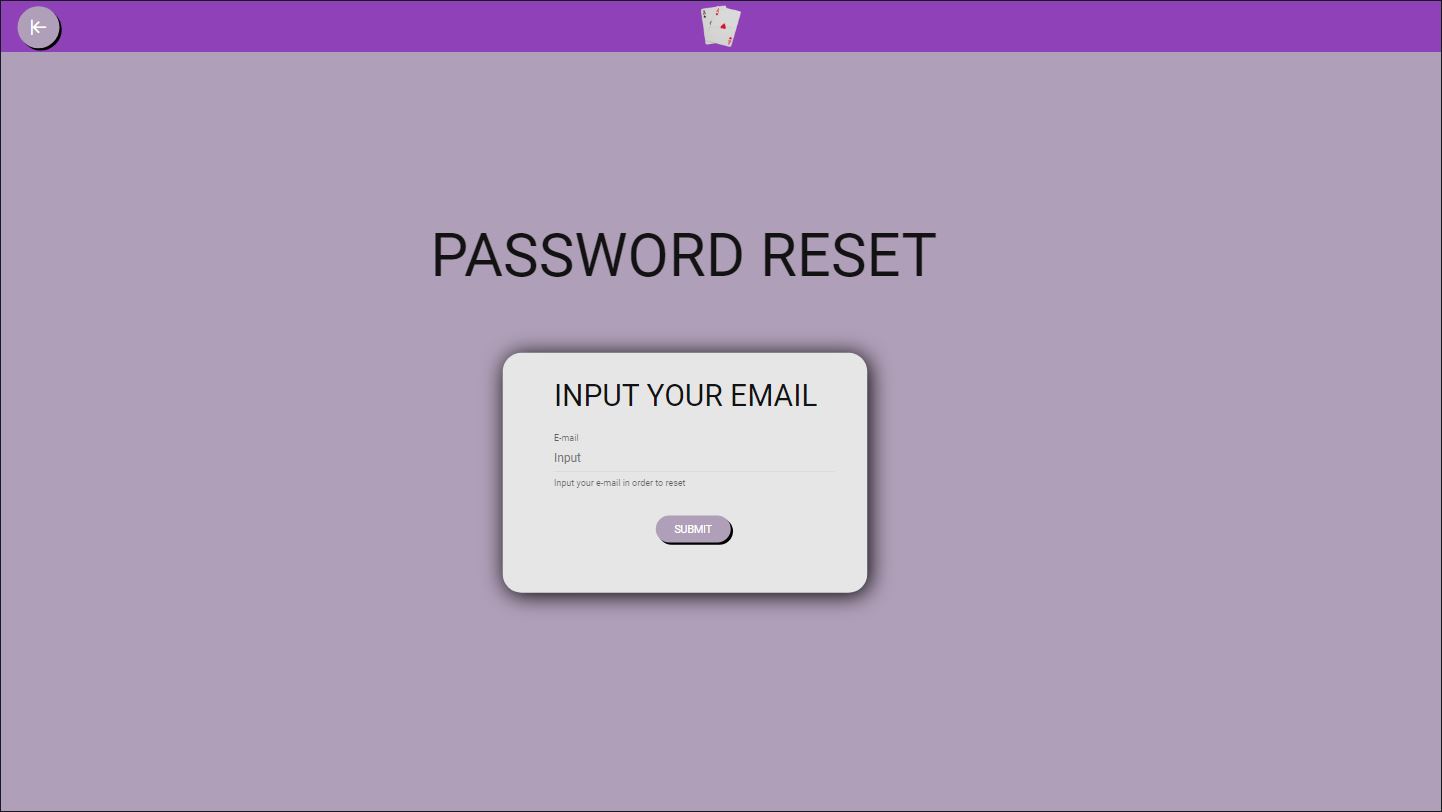
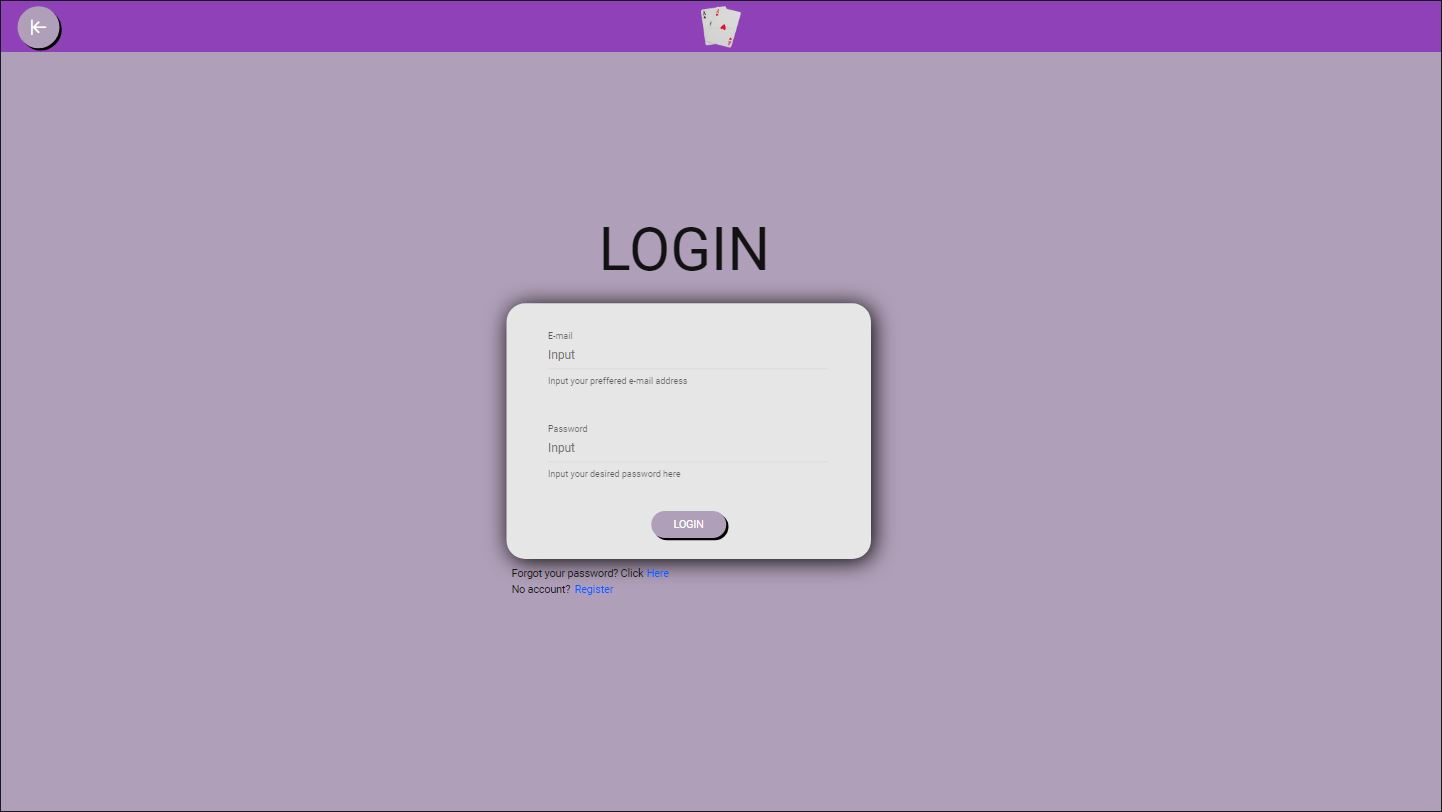
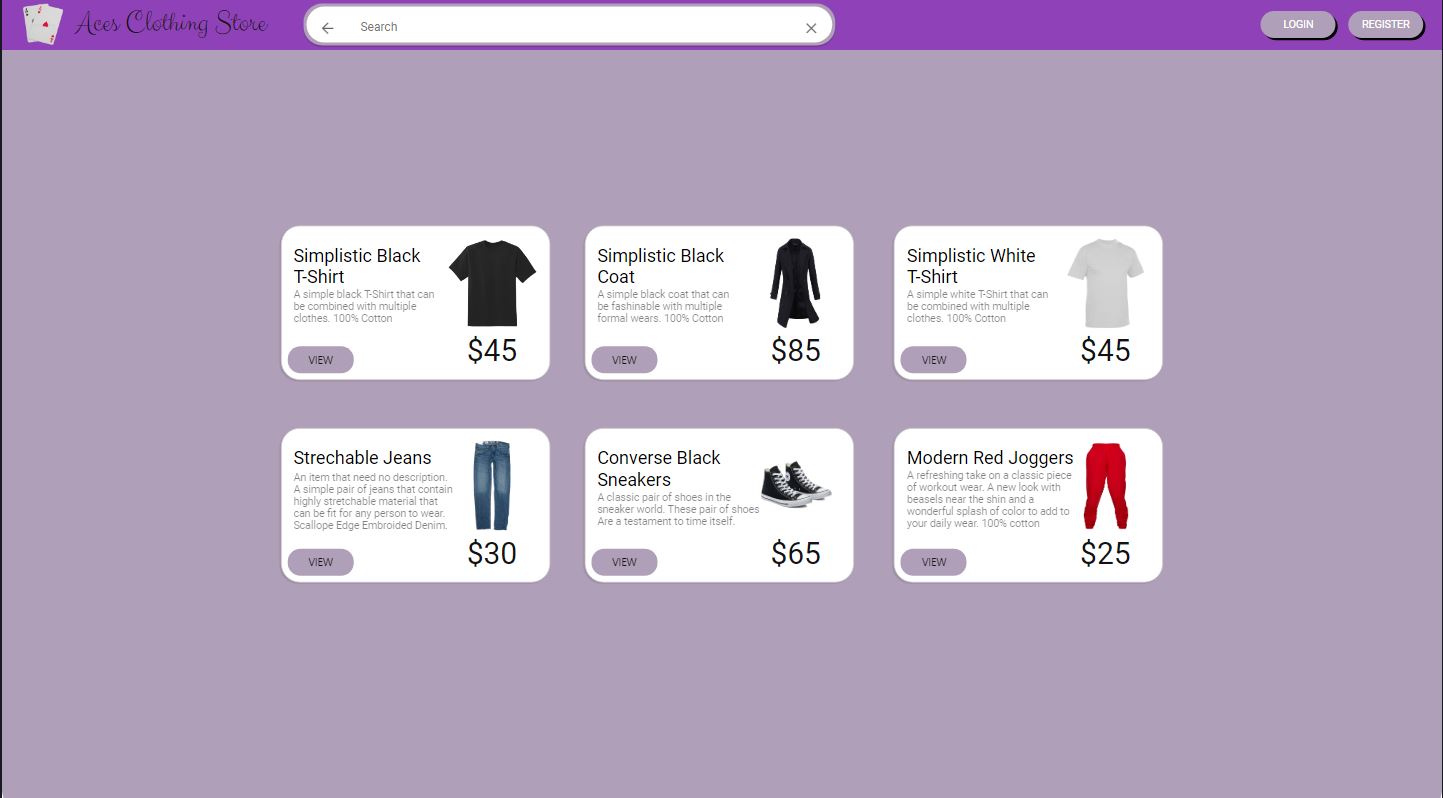
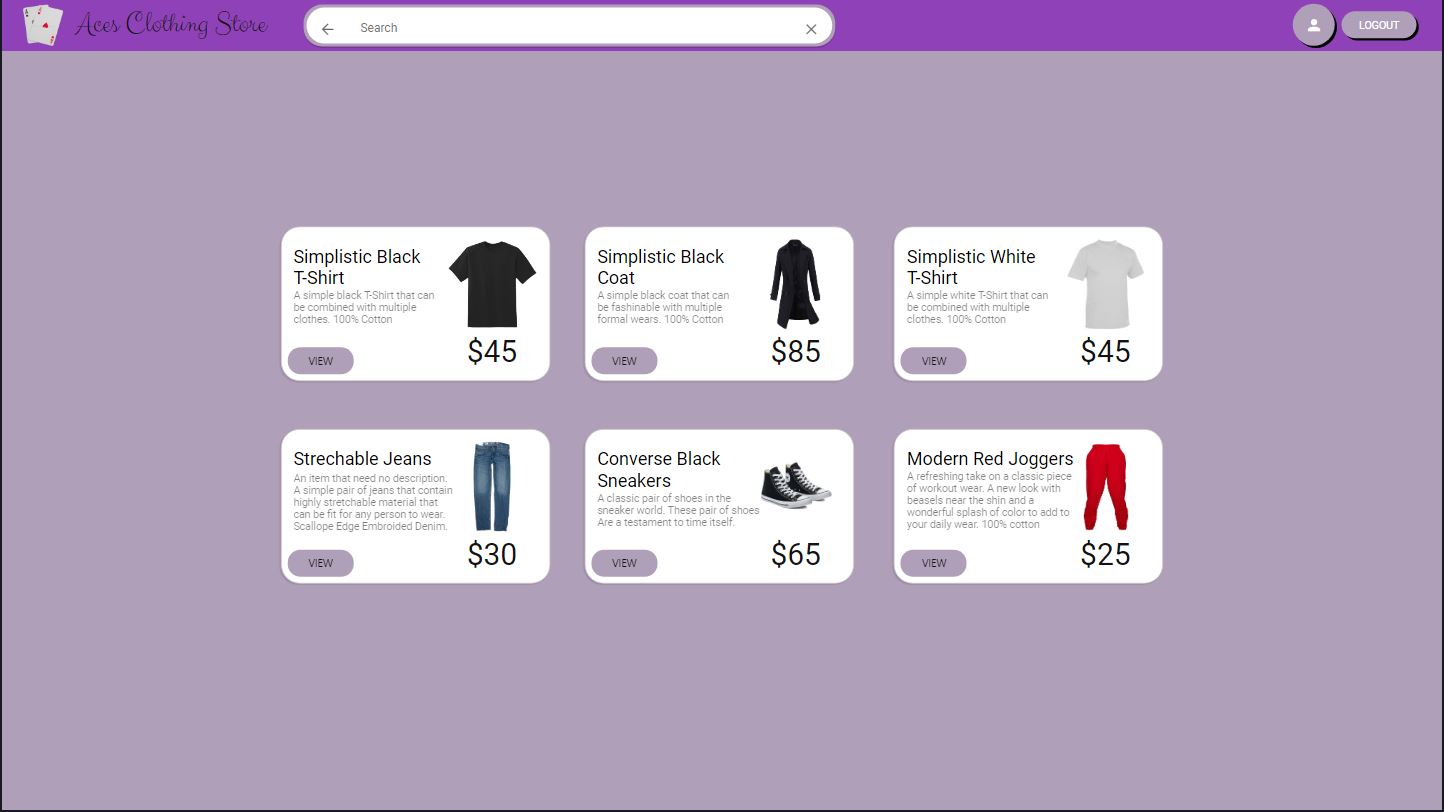
- Deactivate profile entirely

- **Browsing:**

- Search for products

- Open products in detail

- Sort products



# 2. Project Structure

## 2.1. Technologies

Technologies used to build the project are as follows:

- **Backend:**

- Php

- JavaScript

- Java -> used with selenium for unit testing

- **Frontend:**

- JavaScript -> in form of React and React Native coding

- HTML

- CSS

- **Database:**

- SQL

- **IDEs:**

- Eclipse IDE 2022‑06

- Visual Studio Code

- Visual Studio 2019

- MySQL Ultimate

Coding standards for this project are the Coding standard for JavaScript in Visual Studio and React/React Native in Visual Studio (JavaScript coding style for frontend developers)

## 2.2. Database Entities

- **Product:**

- id

- name

- description

- category\_id

- inventory\_id -> used in identifying the amount available of the current item

- price

- **Product Category:**

- id

- name

- description

- **Product Inventory:**

- id

- quantity

- **Order:**

- id

- order\_id

- product\_id

- user\_id

- quantity

- total

- **User:**

- id

- username

- password

- name

- surname

- email

- picture

## 2.3. Architectural Pattern

The architectural pattern for this project is the pattern called Client to Server. The reason that this architectural pattern was chosen is because the pattern is for one most commonly used in types of e-commerce sites and two is because it’s easier to manage the database information by having the server provide the client with the needed information. The reason that patterns such as microservices were not chosen is because there is not a lot of additional functionality on an e-commerce site. What this means is that since the only option for the user is to purchase items, that means that the user only needs one service done, which is why a singular connection from a client to the server is often the fastest way for the client to complete that task, rather than having multiple connections to different unnecessary services.

## 2.4. Design Patterns

For this project one design pattern was used. That design pattern is Abstract Factory. The reason that this design pattern was used is because it fits with the store concept. The abstract factory allows to add new items to the store without creating an entire code structure to add it. This way you have a factory that creates a factory for that type of item.

Location of pattern: *se-project-store/backend/src/addingItems.php*

# 3. Conclusion

Honestly I think that I was not successful in fully implementing my idea for the application. The application that I envisioned was perfect in every way. What is presented in the current file is merely a shadow of what is supposed to be a ready to sell application. There are multiple parts that are quite obviously missing from the main application, however I think still that this project gave me an opportunity to explore and find new methods of implementing different coding standards into a real world based scenario.

As mentioned in the beginning, the current state of the app is not satisfactory. Sure the app frontend is clear written and it can be launched on its own, however the backend lacks connection to the frontend, which diminished what the app is supposed to present. However, now I have a base template for the store design that I can implement a real working store onto in the future.

The hardest part to figure out is the connection to the inventory of the application, i.e. how to add inventory and reduce inventory once a purchase is complete or once a restock happens.

The part that I cannot currently handle is the connection with the backend. I still am fairly new to the usage of React JS and React Native services, so I am still trying to figure out the best approach for connection the database with the backend and then the backend with the frontend.

//TODO: Connect backend

//TODO: Freshen up the look of the frontend

//TODO: Add credit card payment instead of on delivery payment

//TODO: Connect database inventory management to backend factory