# NWEN304 final group document

This e- commerce platform created by team of third year students for NWEN304 as part of group project:

- Elaf Almusaid
- Hayden Pomare
- Kitt McEvoy

## 1.0 Introduction

This document will draw upon and evaluate the overview of the e- commerce platform that allow user to browse shopping items such as clothing, pants, shoes and accessories. The application requirements were to design a secure, private, RESTful, has navigation functionality and scalable application. The design of e-commerce platform was formed to meet the application requirement by using various ranges of resources and programming languages techniques. These include, third party libraries such as mongoose, express, ejs, passport, cookie-parser, chai, mocha and aws-sdk. User can sign in through their email, username and password, the system can also suggest new secure password and agree to the terms & condition to the e-commerce platform. We also have login/logout system were user can login through their Google API. Furthermore, the application development process was divided into stages. The first stage of the application was the security which is one of the major requirements of the application. Decision on security of the application was users needed to be authenticated before using the application and this was achieved by two options, local signup or using media website such as Google to login. Additionally, one of the application's main functions is that user need to login and stay login to be able to browse items and shop.

## 2.0 How to use the platform
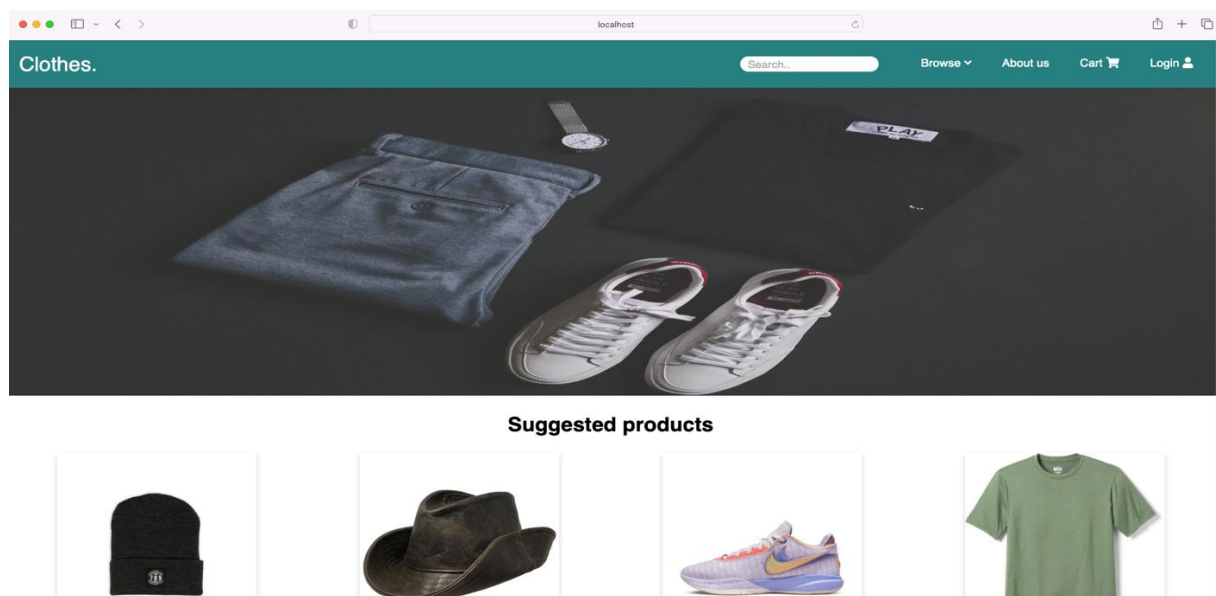
when you first load up the page, you will be led to our main page there are number of ranges things you can do. At the top there are navigation bar with dropdown Browse, About us information, Cart and login. Where the Cart only available once user login and have more accessible features like Orders. If you scroll down you view random items with their title, price and description. With some details in how to contact us which is either by contacting our phone number or visit our social platform.
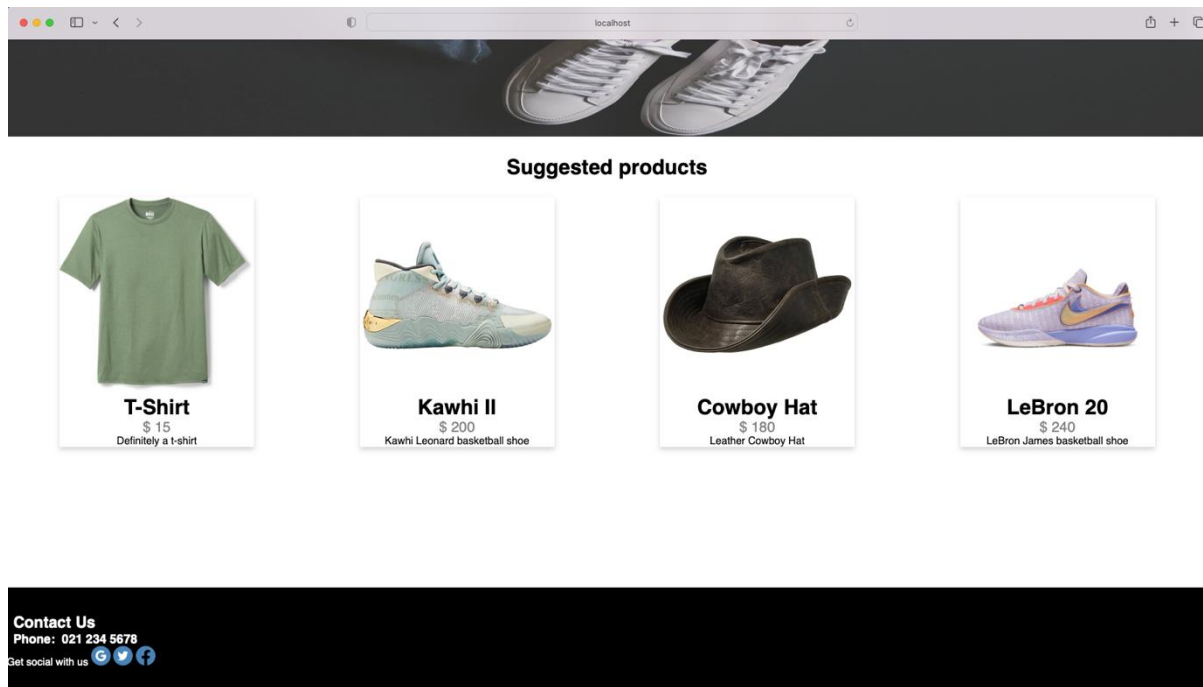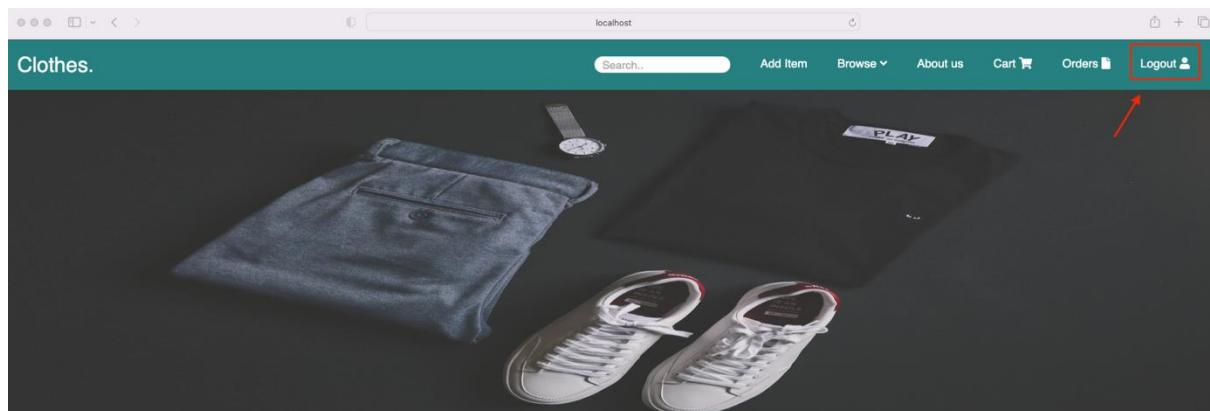


We are then taken to sign up / login page and ask to either sign in locally with email address, username and password. Or through social media platform like Google. If we entered the correct information here we successfully will be taken back to the main page but we will now be logged in.
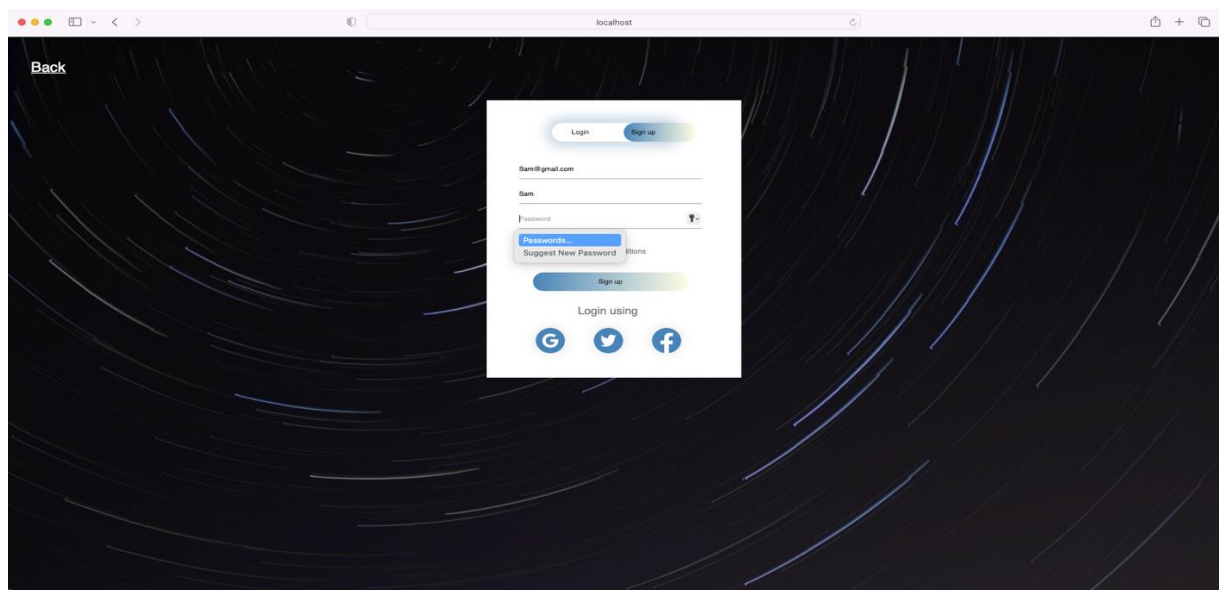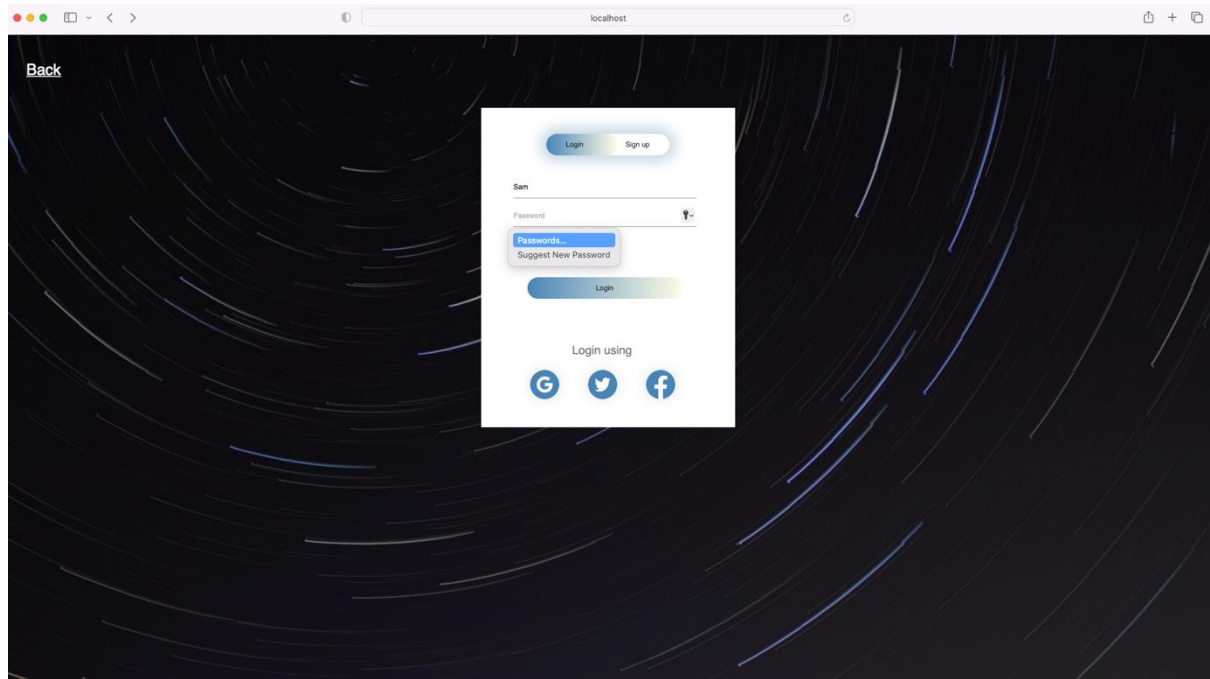
Once we login, will have a user id in the mongoDB and can login next time easily and be able to browse and shop. We can now see from the navigation bar that we have been logged in, and can add items to the cart and see our orders.

### 3.0 Design Requirements

#### Security

Security is a set of measures taken when designing, coding and deploying the application. We way we implement our code take these measures into account to prevent the system from being exposed to internal and external threats/ attacks such as cross site scripting (XSS) and cross site request forgery (CSRS). To achieve high security when designing the application, we used session and cookies for user authentication and only allow these users to Create, Get and delete items. With the aid of a third-party library both local and google authentication was achieved. For local authentication, "passport-local" and for Google authentication "passport-google-oauth2" library was used. The passwords are encrypted before being send to the database for storage. We also, implemented cookie using third party library called "cookie-parser". Cookie simply is a way to know that user is authenticated we can store the information in the browser on the client side.

#### RESTfull

One of the main specifications of the application was ensuring that it supports RESTful as a distributed system approach. These involves 4 HTTP verbs. Three out of the four verbs were used whilst making the application, these included GET, POST, DELETE. The fundamental areas of POST were implemented when a new user sign up, login, add item, edit item by id, a POST verb was used to send information to the database and return a message (200) indicating that a successful request was stored. Secondly, the verb DELETE was utilized to have the ability to delete item from mongoDB by using the id of item to be removed. Finally, the verb GET was employed when a user successfully logged in, logout, browse, cart, order, API items, API users and error handling pages.

#### MongoDB

Another server side the application hosted is MongoDB, which ensures efficiency. The decision to use MongoDB database as using mongoose drive as mongoose is an object database modelling tool that allows users to skip some lower level operation and based on the fact that we can store items and user's information. More importantly, mongoDB has greater suitability for NodeJS and advance schema-based feature.

Adding item to the database page have different field to fill such as Title, Description of the item, Size, Color, Price, Image URL and uploading image from a file.



After adding the items, we can then see them through API without a GUI
Using: http://localhost:3000/api/items

We then can see the items successfully added to MongoDB both in MongoDB compass Under the items folder or through a MongoDB cluster.
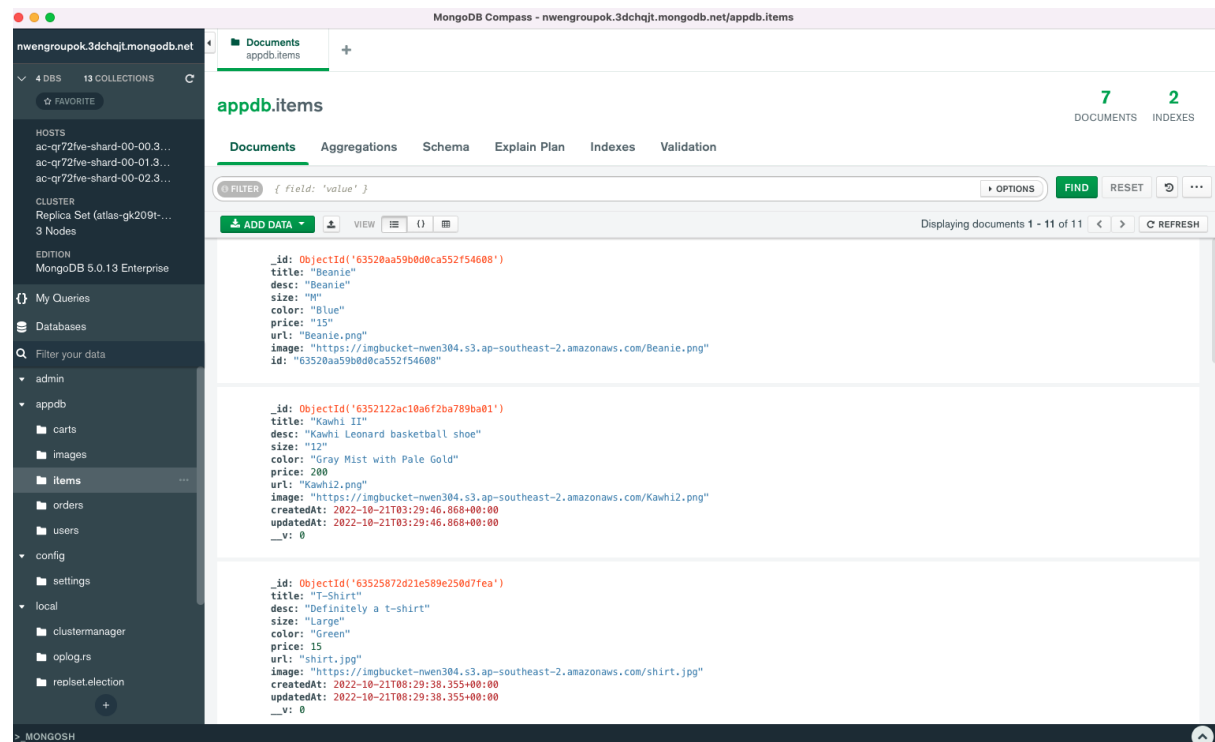


We can also read the users id without a GUI, by using: http://localhost:3000/api/users

A logged in user can successfully visit a Cart page where they can see the item card with some details such as price and a button to pressed to check out
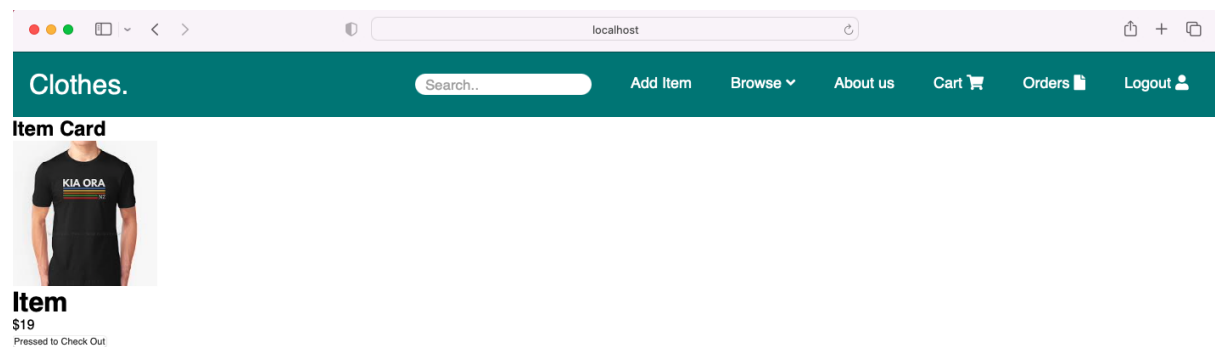


A logged in user can successfully visit an Orders page where they can see the payments summary details display and a button to track their order.
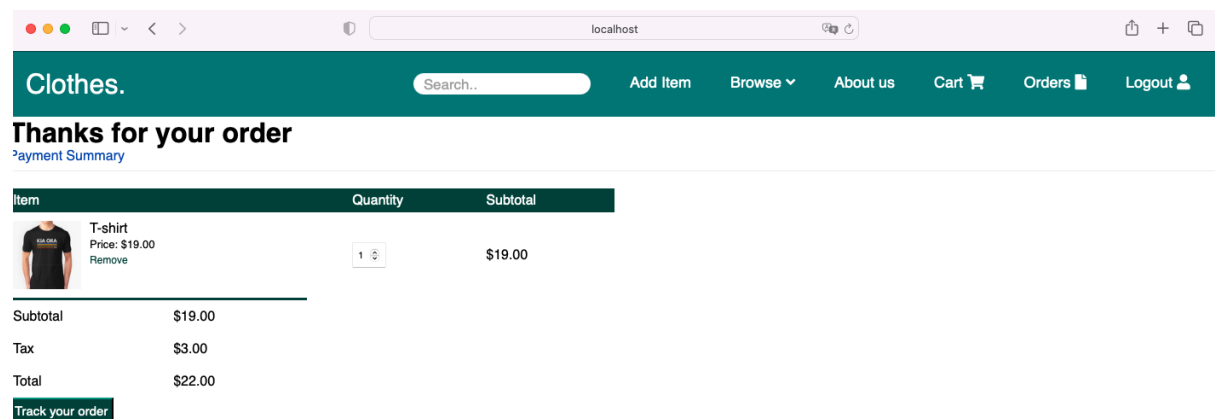
**4.0 Error handling**

Login/ signup

We have a number of different error states that can handle when it comes to both login and signup. First it checks for a valid email address. If a user tries to enter a name in state of email address a message tells the user to enter an email address.
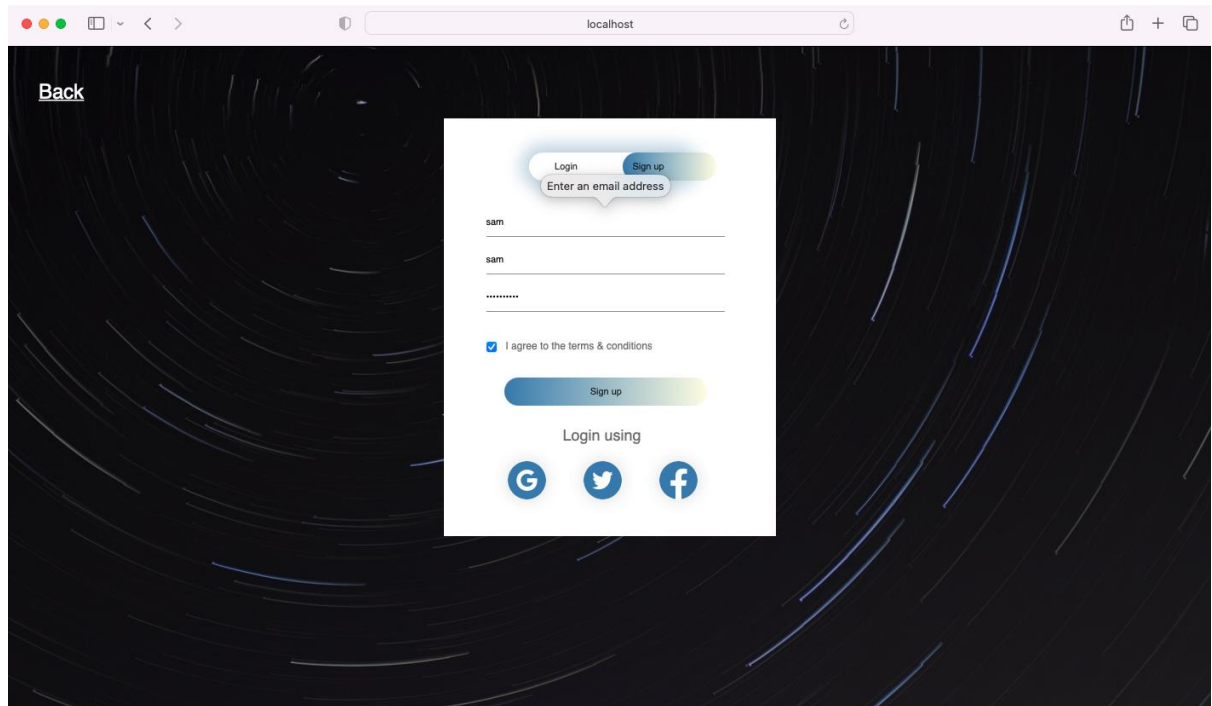


A Second check to see if a password matches the request format with minimum 8 characters, at least 1 letter, 1 number and 1 special character.
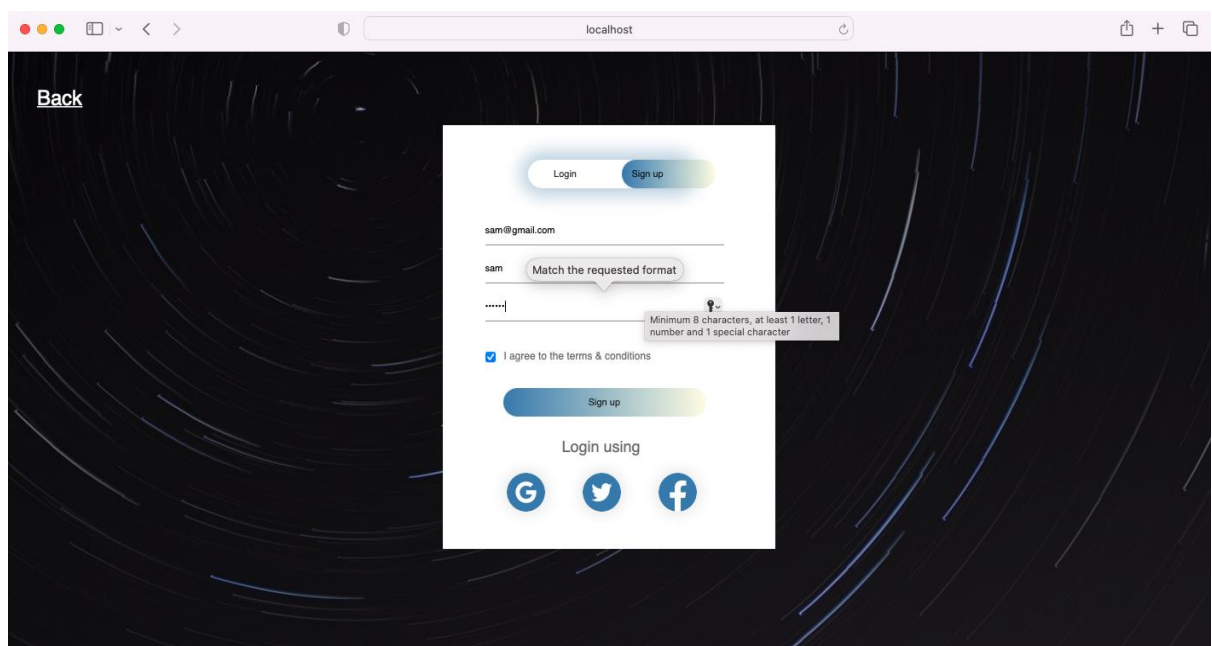
Third check, if a user login with wrong information of username or password it gets directed to a 401 page with a login failed message.
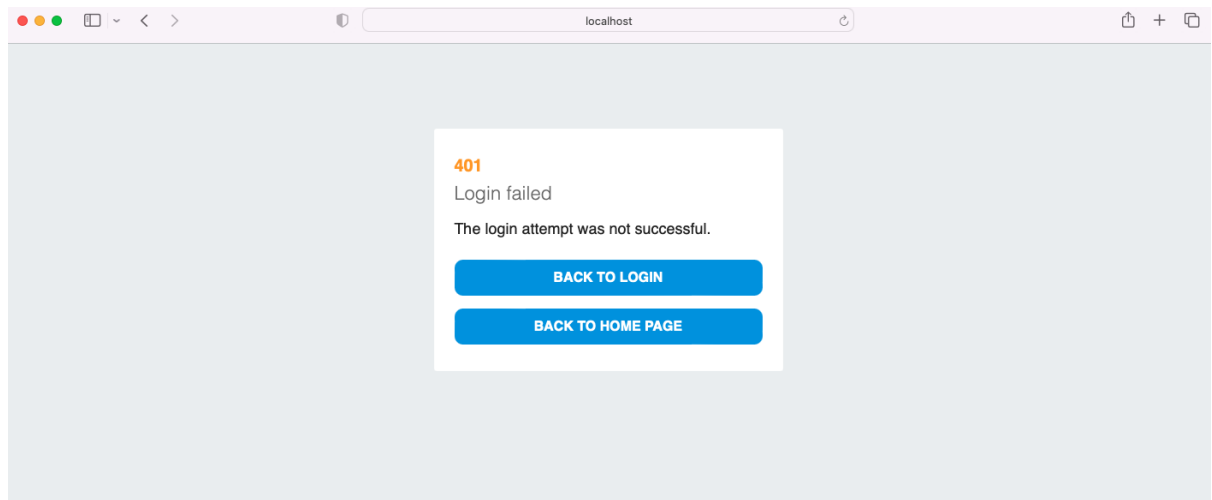


Fourth check, if a user tries to access a page that is not there with wrong API address it gets directed to a 404 with Page Not Found message.