CS673 Software Engineering
Team 6 : College Street
Software Design Document

Your project Logo here if any

Team Member	Role(s)	Signature	Date
Theerarun Tubnonghee (Steve)	Team Leader/	*Theerarun Tubnonghee*	10/19/2023
Aishwarya Raja	Configuration Mgmt	*Aishwarya Raja*	10/19/2023
Nidhi Desai	Quality Assurance	*Nidhi Desai*	10/19/2023
Subhajit Das (Jeet)	Back-End Lead	*Subhajit Das*	10/19/2023
Vedant Gupta	Design/Product Implementation	*Vedant Gupta*	10/19/2023
Yin Xiancheng(Xanthus)	DevOps (combine of FE/BE, domain, ...)	*Yin Xiancheng*	10/19/2023
Chenyang Lyu (Nick)	Front-End Lead	*Chenyang Lyu*	10/19/2023

Revision history

Version	**Author**	**Date**	**Change**
1.0	**Team 6**	**10/19/2023**	**Writeup**
1.1	**Steve/Jeet**	**11/11/2023**	**SA/SD/BL/DP**

Introduction

Software Architecture

Class Diagram

UI Design (if applicable)

Database Design (if applicable)

Security Design

Business Logic and/or Key Algorithms

Design Patterns

Any Additional Topics you would like to include.

References

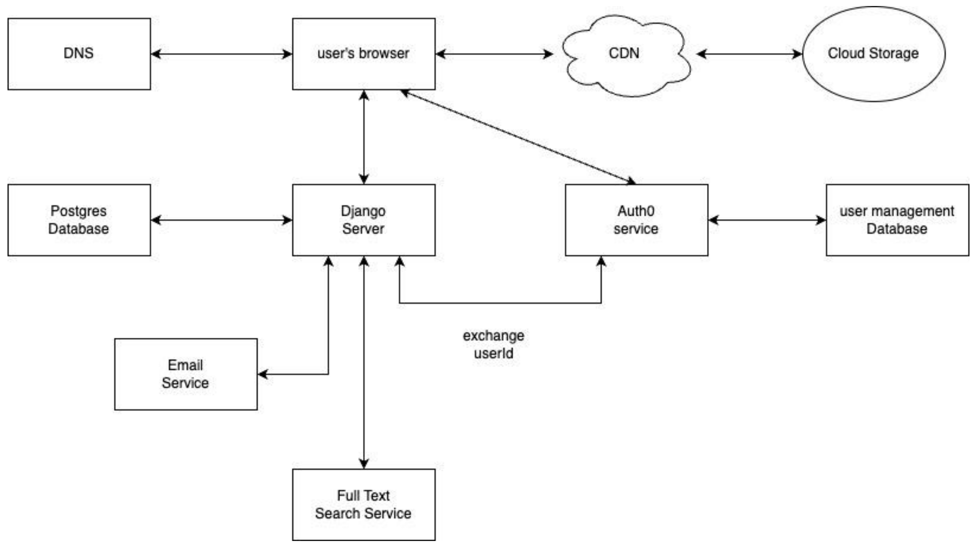
Glossary

• Introduction

This document will be an overview of our project design and implementation for backend and frontend. We will describe the Software Architecture, Class Diagram, UI design, Database design, Security design, Business Logic, and Design pattern that we use for the project.

• Software Architecture

The SA diagram depicts our college-street project infrastructure involving several interconnected services. The user's browser communicates with a Django server. It also connects to an email service, used for sending verification emails as well as forgot password. The architecture includes an Auth0 service, for the implementation of a secure authentication and authorization system which interfaces with a separate user management database, which we later changed it to our in house Django authentication system. A full-text search service is also connected to the Django server, indicating that the application has a feature to search through the entire list of products. The Content Delivery Network (CDN) service will be used for the image processing of the products which will be pushed to our cloud network - We will be using Cloudinary for it. Arrows between components show the direction of data flow and dependencies within the system.



MVC

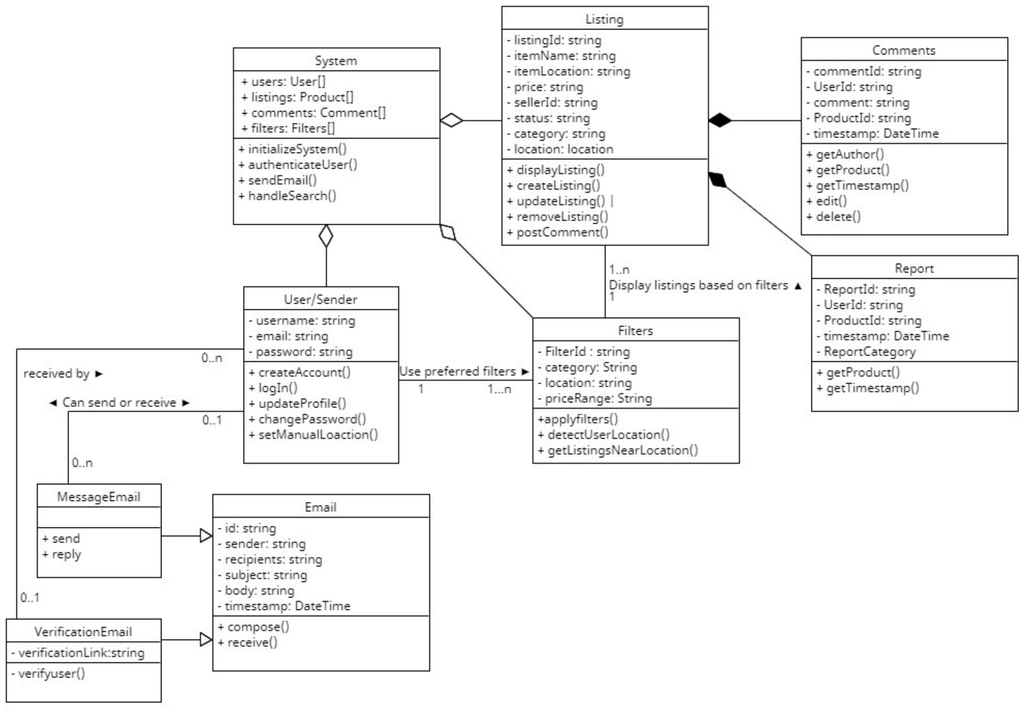
We use MVC (Model-View-Controller) Pattern, which separates the application logic into three interconnected components: the model (data and business logic), the view (user interface), and the controller (input handling and coordination).

Our backend service will be written in Django and django-rest framework which by default uses MVC design pattern.

Our client side code is written in ReactJS which uses Flux type Design pattern but we are not following any certain design pattern for our clients. We are using ReactJS for our UI/UX design but React internally uses Flux design pattern.

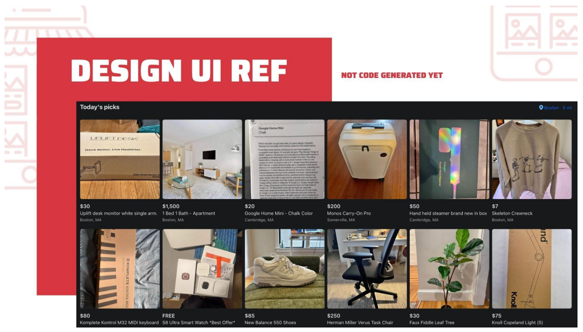
● Class Diagram

Below is the class diagram, v0, for the current version.



- ## UI Design

  The ui design reference is from the airbnb and facebook marketplace. The UI will be clean on white theme with a tint of red for the excitement of the color, as for BU. However, using too much red in such an application could cause a lot of confusion while navigating, we decide to have the red as an accent color and use black and white as a main navigations.



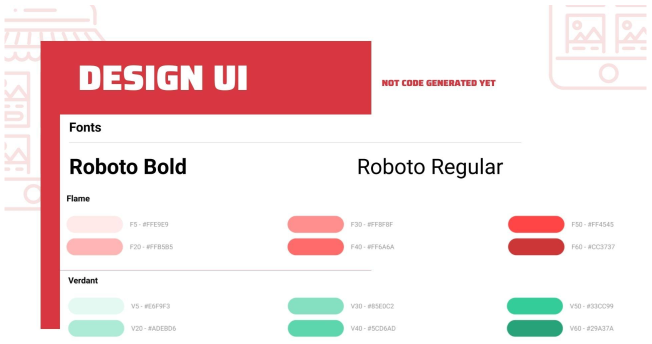
DESIGN UI REF

NOT CODE GENERATED YET

Today's picks

Boston · 5 mi

$30	Uplift desk monitor white single arm.	Boston, MA
$1,500	1 Bed 1 Bath - Apartment	Boston, MA
$20	Google Home Mini - Chalk Color	Cambridge, MA
$200	Monos Carry-On Pro	Somerville, MA
$50	Hand held steamer brand new in box	Cambridge, MA
$7	Skeleton Crewneck	Boston, MA
$80	Komplete Kontrol M32 MIDI keyboard	
FREE	S8 Ultra Smart Watch *Best Offer*	
$85	New Balance 550 Shoes	
$250	Herman Miller Verus Task Chair	
$30	Faux Fiddle Leaf Tree	
$75	Knoll Copeland Light (5)	



DESIGN UI

NOT CODE GENERATED YET

Fonts

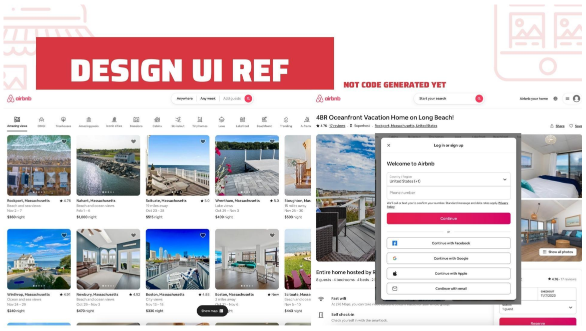
Roboto Bold Roboto Regular

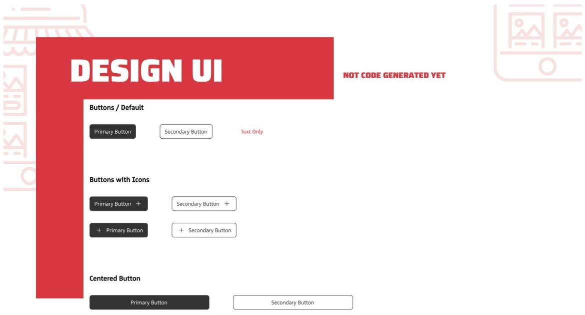
Flame

F5 - #FFE9E9	F30 - #FF8F8F	F50 - #FF4545
F20 - #FFB5B5	F40 - #FF6A6A	F60 - #CC3737

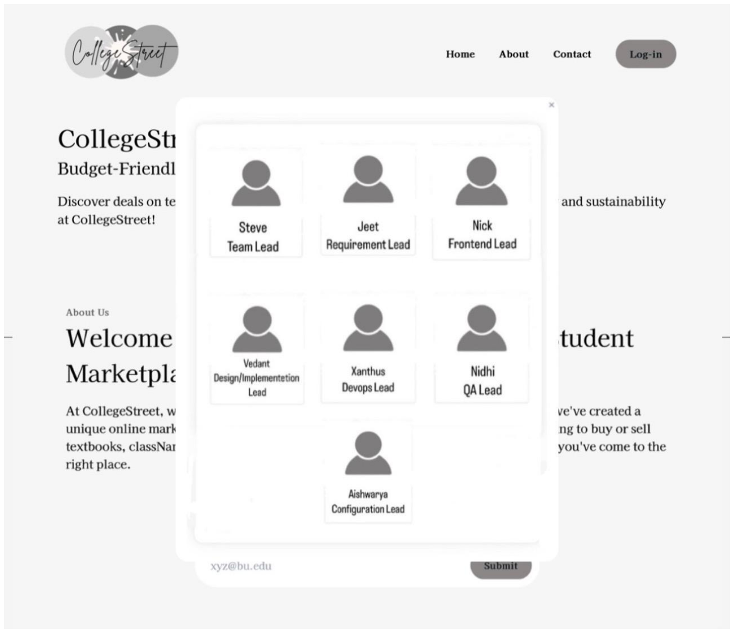
Verdant

V5 - #E6F9F3	V30 - #85E0C2	V50 - #33CC99
V20 - #ADEBD6	V40 - #5CD6AD	V60 - #29A37A

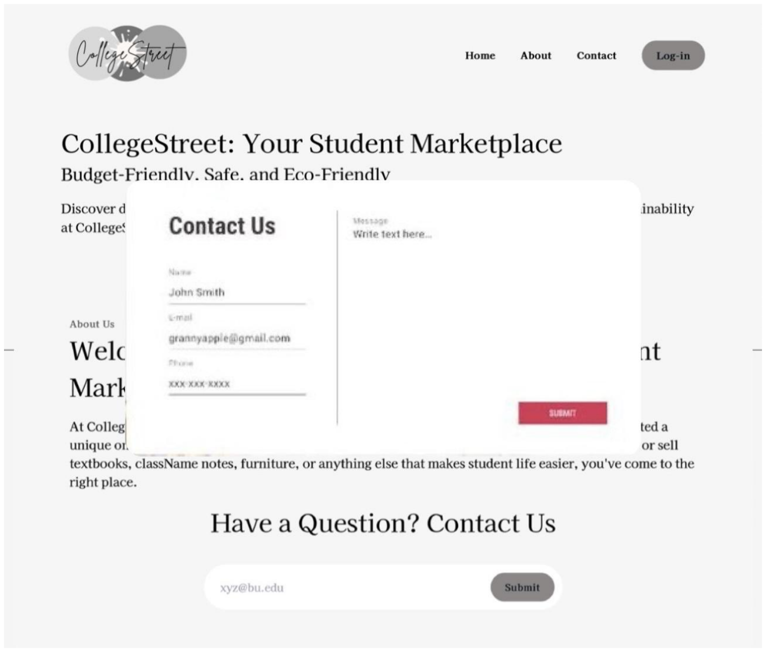




About Page Popup UI Design:



Contact US Page Popup UI Design:



Home About Contact Log-in

CollegeStreet: Your Student Marketplace

Budget-Friendlv. Safe. and Eco-Friendlv

Discover d … inability
at CollegeS

About Us

Welc … nt
Mark

At Colleg … ted a
unique or … or sell
textbooks, className notes, furniture, or anything else that makes student life easier, you've come to the right place.

Have a Question? Contact Us

xyz@bu.edu Submit

Contact Us

Name
John Smith

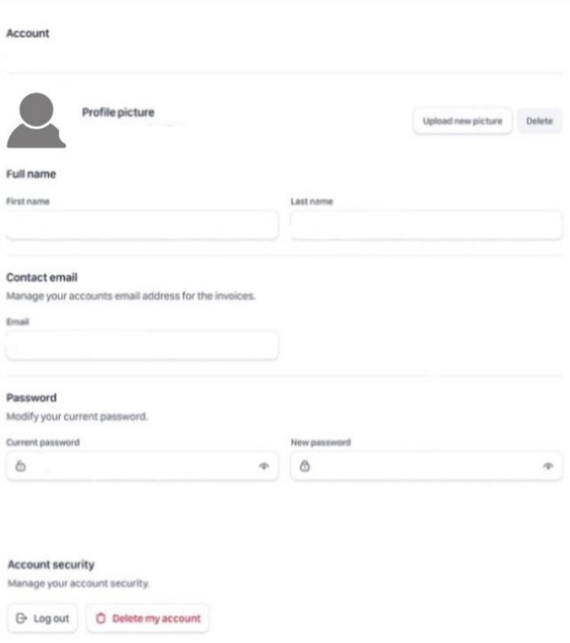
E-mail
grannyapple@gmail.com

Phone
xxx-xxx-xxxx

Message
Write text here...

SUBMIT

User Profile Page Popup UI Design: (Code in progress)



Account

Profile picture

Upload new picture Delete

Full name

First name

Last name

Contact email

Manage your accounts email address for the invoices.

Email

Password

Modify your current password.

Current password

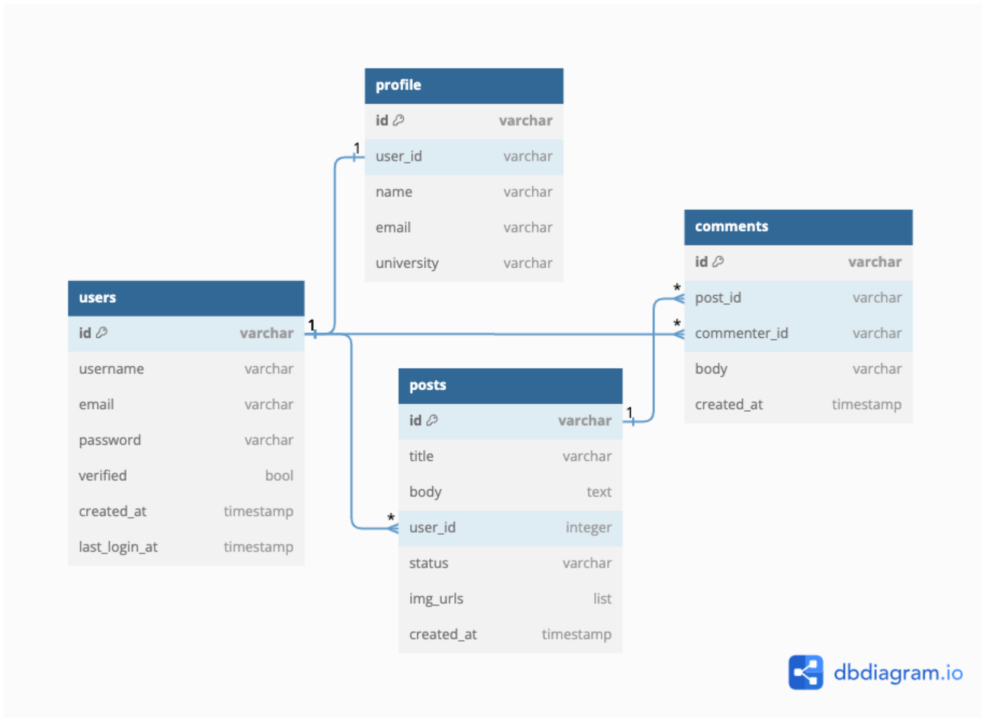
New password

Account security

Manage your account security.

Log out Delete my account

- Database Design



- Security Design

We will skip this section for now, because we do not have a security design for any part of the design except for the privacy information design for the user to encrypt user information such username and password.

User Authentication

Registration

Users provide an edu email address and create a password.

Email Verification

An email verification link is sent to the user's edu email address. User verifies the email using provided link in the email to activate the account.

Password

Passwords are securely stored in the system using cryptographic hashing.

Data Security

Password

Strong passwords are enforced.

Data Sorage

Secure database.

Data Transmission

Secure connection HTTPS.

Session Management

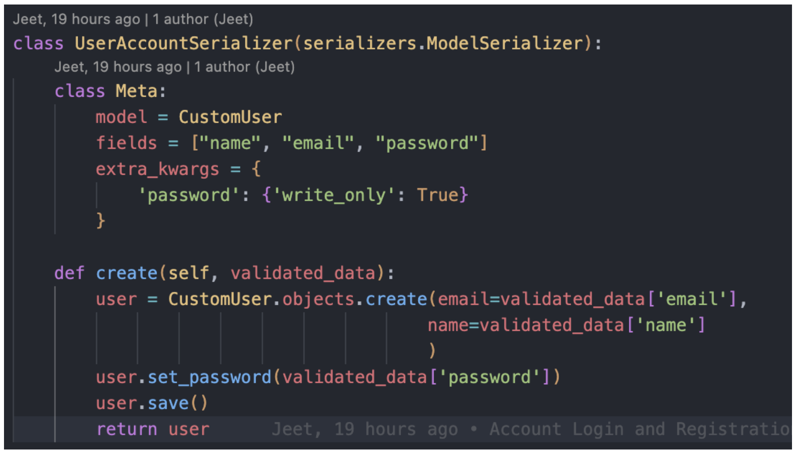
jwtToken.

HTTPS/REST-API Connection

Update on iteration 3

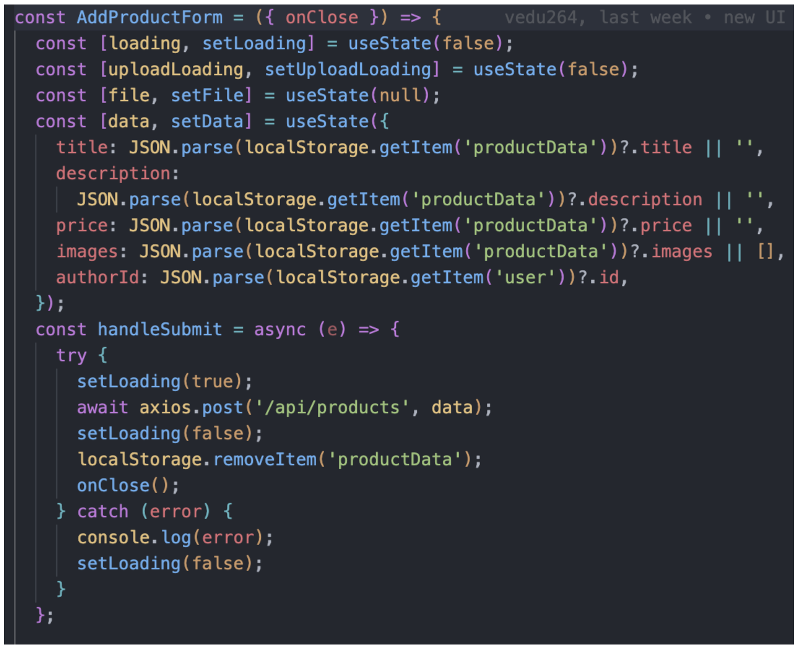
- Business Logic and/or Key Algorithms

User Authentication



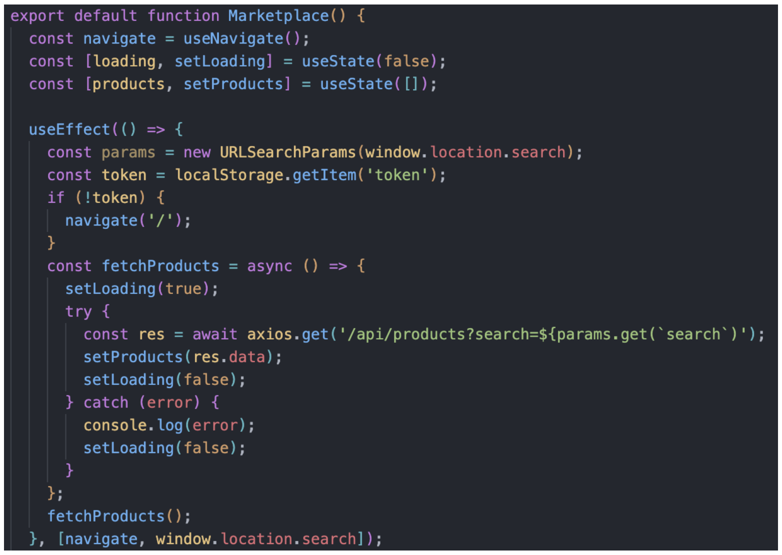
```
Jeet, 19 hours ago | 1 author (Jeet)
class UserAccountSerializer(serializers.ModelSerializer):
    Jeet, 19 hours ago | 1 author (Jeet)
    class Meta:
        model = CustomUser
        fields = ["name", "email", "password"]
        extra_kwargs = {
            'password': {'write_only': True}
        }

    def create(self, validated_data):
        user = CustomUser.objects.create(email=validated_data['email'],
                                         name=validated_data['name']
                                         )
        user.set_password(validated_data['password'])
        user.save()
        return user      Jeet, 19 hours ago • Account Login and Registratio
```

Publish a Post



```
const AddProductForm = ({ onClose }) => {      vedu264, last week • new UI
  const [loading, setLoading] = useState(false);
  const [uploadLoading, setUploadLoading] = useState(false);
  const [file, setFile] = useState(null);
  const [data, setData] = useState({
    title: JSON.parse(localStorage.getItem('productData'))?.title || '',
    description:
      JSON.parse(localStorage.getItem('productData'))?.description || '',
    price: JSON.parse(localStorage.getItem('productData'))?.price || '',
    images: JSON.parse(localStorage.getItem('productData'))?.images || [],
    authorId: JSON.parse(localStorage.getItem('user'))?.id,
  });
  const handleSubmit = async (e) => {
    try {
      setLoading(true);
      await axios.post('/api/products', data);
      setLoading(false);
      localStorage.removeItem('productData');
      onClose();
    } catch (error) {
      console.log(error);
      setLoading(false);
    }
  };
```

Display Published Posts



```
export default function Marketplace() {
  const navigate = useNavigate();
  const [loading, setLoading] = useState(false);
  const [products, setProducts] = useState([]);

  useEffect(() => {
    const params = new URLSearchParams(window.location.search);
    const token = localStorage.getItem('token');
    if (!token) {
      navigate('/');
    }
    const fetchProducts = async () => {
      setLoading(true);
      try {
        const res = await axios.get('/api/products?search=${params.get(`search`)}');
        setProducts(res.data);
        setLoading(false);
      } catch (error) {
        console.log(error);
        setLoading(false);
      }
    };
    fetchProducts();
  }, [navigate, window.location.search]);
```

- Design Patterns

  Update on iteration 3.

- Any Additional Topics you would like to include.

- References

- Glossary