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

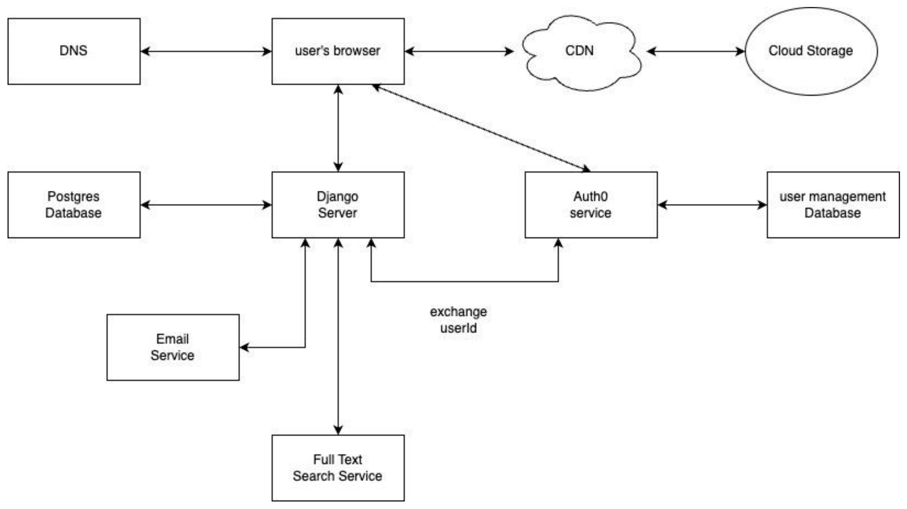
Revision history

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

- ## Introduction

  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



Software Architecture Overview: College-Street Project

User Interaction:

Users interact with the system via a web browser that communicates with the Django server.
Django Server:

Acts as the central server, handling application logic and data processing.
Implements the MVC (Model-View-Controller) design pattern.
Model: Manages data and business logic.
View: Handles the user interface.
Controller: Manages input handling and coordination.
Email Service:

Integrated with the Django server for sending verification and password reset emails.
Authentication and Authorization:

Initial Implementation: Initially used Auth0 for secure authentication and authorization.
Current System: Transitioned to an in-house Django-based authentication system, replacing Auth0.
Full-Text Search Service:

Connected to the Django server, allowing users to perform comprehensive searches through the product database.
Content Delivery Network (CDN):

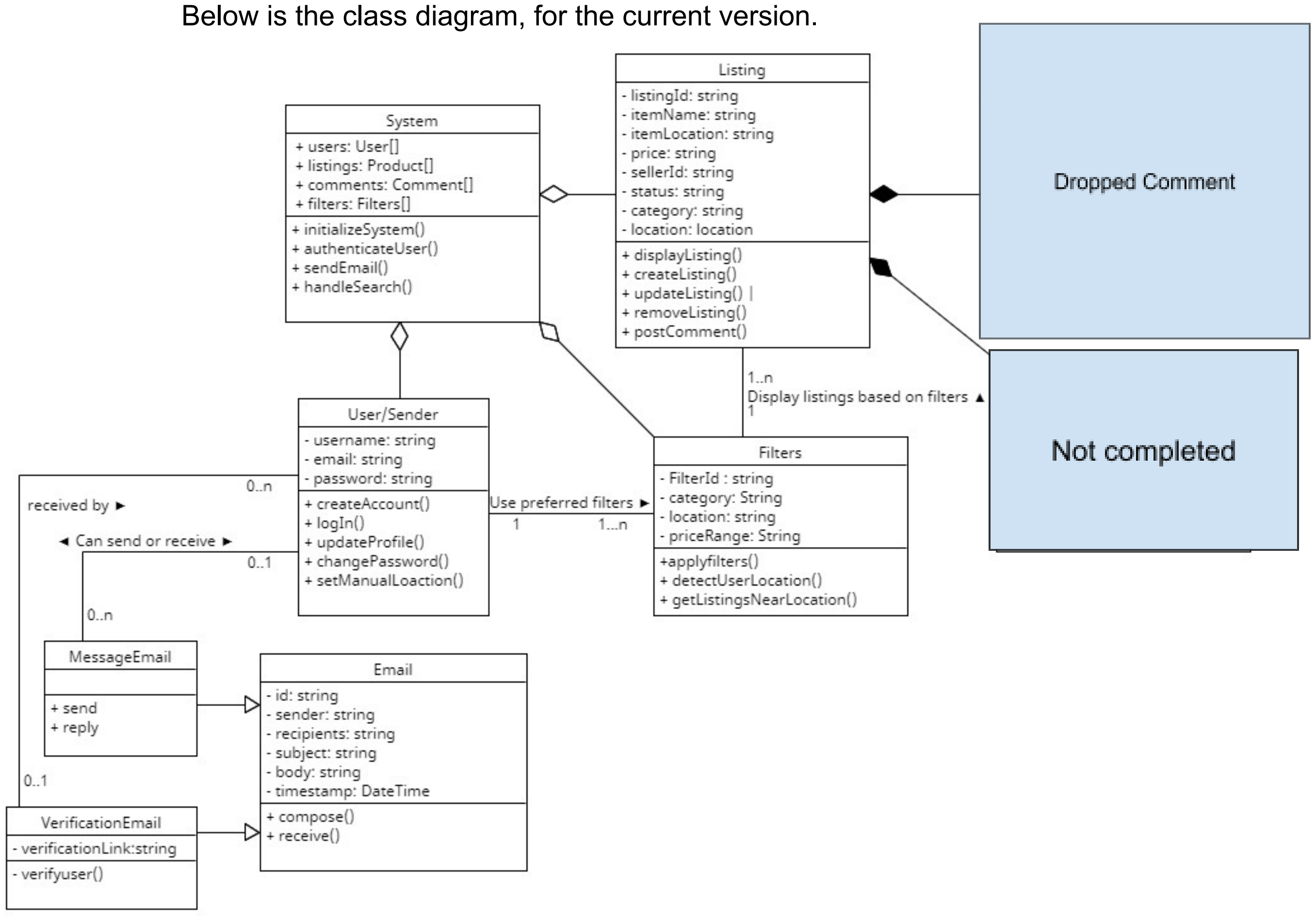
Used for processing product images, which are then uploaded to the Cloudinary cloud network.
Data Flow and Dependencies:

The SA diagram includes arrows to represent the direction of data flow and the dependencies among different components.
Client-Side Development:

Client-side code is developed in ReactJS.
Focuses on UI/UX design, with ReactJS internally using the Flux design pattern, though not strictly adhered to in this project.

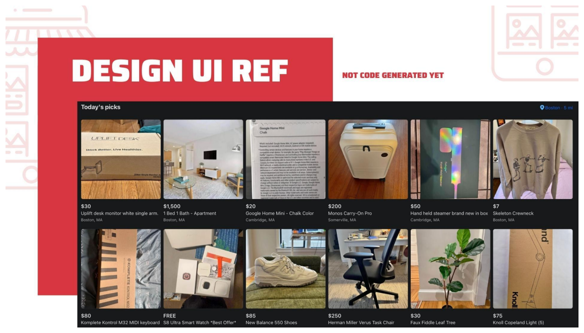
● Class Diagram

Below is the class diagram, 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.



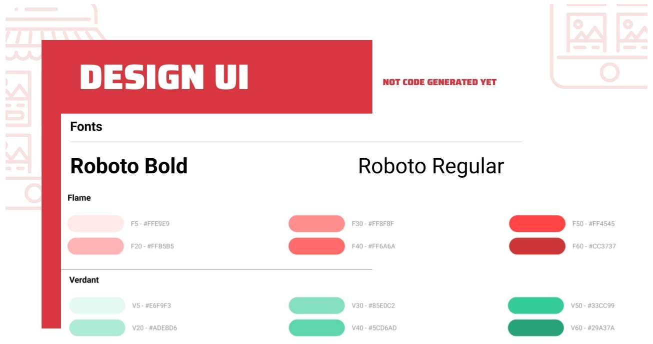
DESIGN UI REF

NOT CODE GENERATED YET

Today's picks

Boston · 5 mi

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



DESIGN UI

NOT CODE GENERATED YET

Fonts

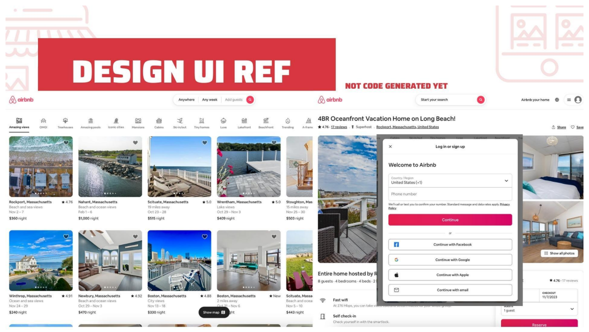
Roboto Bold   Roboto Regular

Flame

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

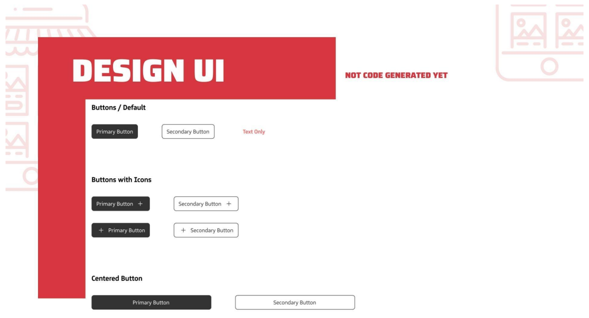
Verdant

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



DESIGN UI REF

NOT CODE GENERATED YET



DESIGN UI

NOT CODE GENERATED YET

Buttons / Default

Primary Button   Secondary Button   Text Only

Buttons with Icons

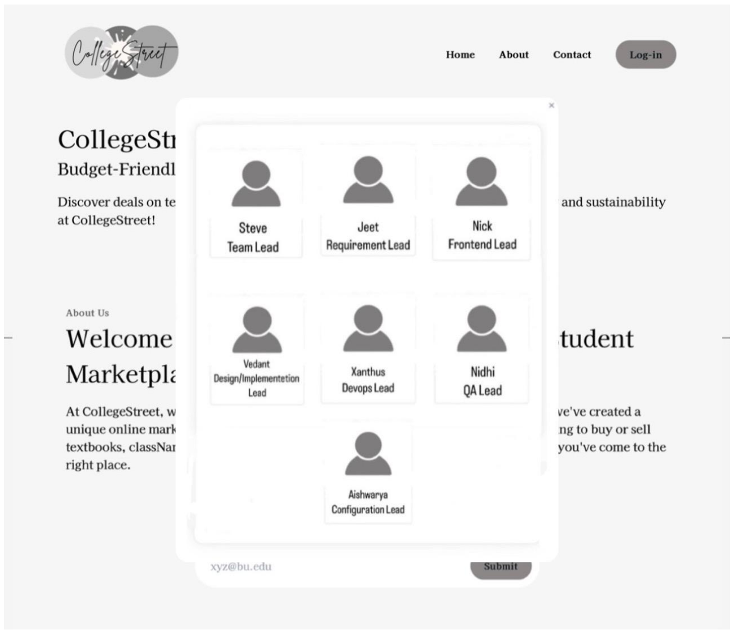
Primary Button +   Secondary Button +

+ Primary Button   \+ Secondary Button

Centered Button

Primary Button   Secondary Button

About Page Popup UI Design:



CollegeSt
Budget-Friendl

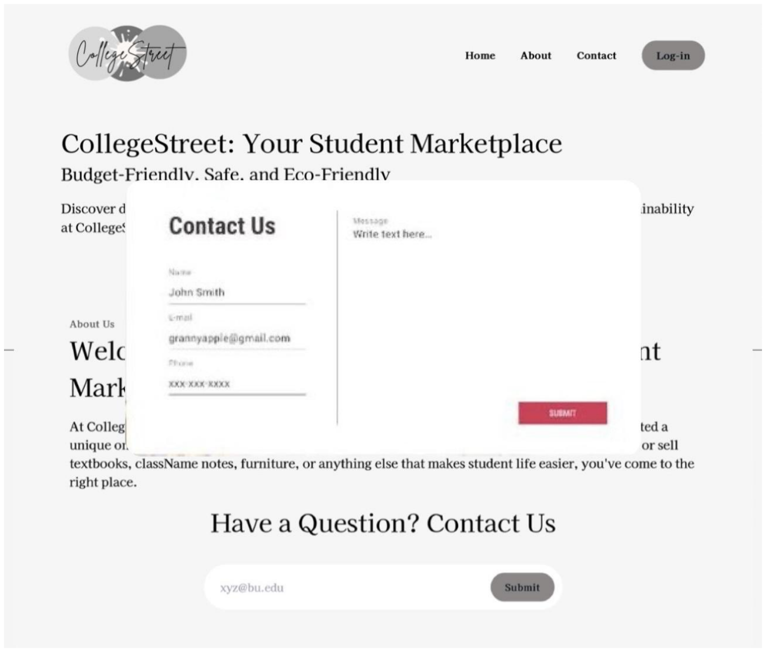
Discover deals on te … and sustainability at CollegeStreet!

Welcome … tudent Marketpl

At CollegeStreet, w … ve've created a unique online mark … ng to buy or sell textbooks, classNar … you've come to the right place.

xyz@bu.edu

Contact US Page Popup UI Design:



CollegeStreet: Your Student Marketplace
Budget-Friendly, Safe, and Eco-Friendly

Discover d … inability at College

Contact Us

Name: John Smith
E-mail: grannyapple@gmail.com
Phone: xxx-xxx-xxxx
Message: Write text here...

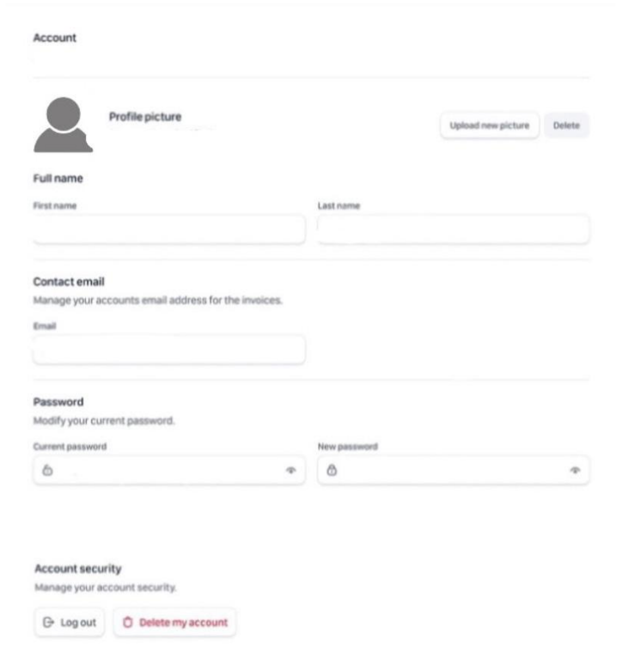
Welc … nt
Mark

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

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



Account

Profile picture

Full name

First name

Last name

Contact email

Manage your accounts email address for the invoices.

Email

Password

Modify your current password.

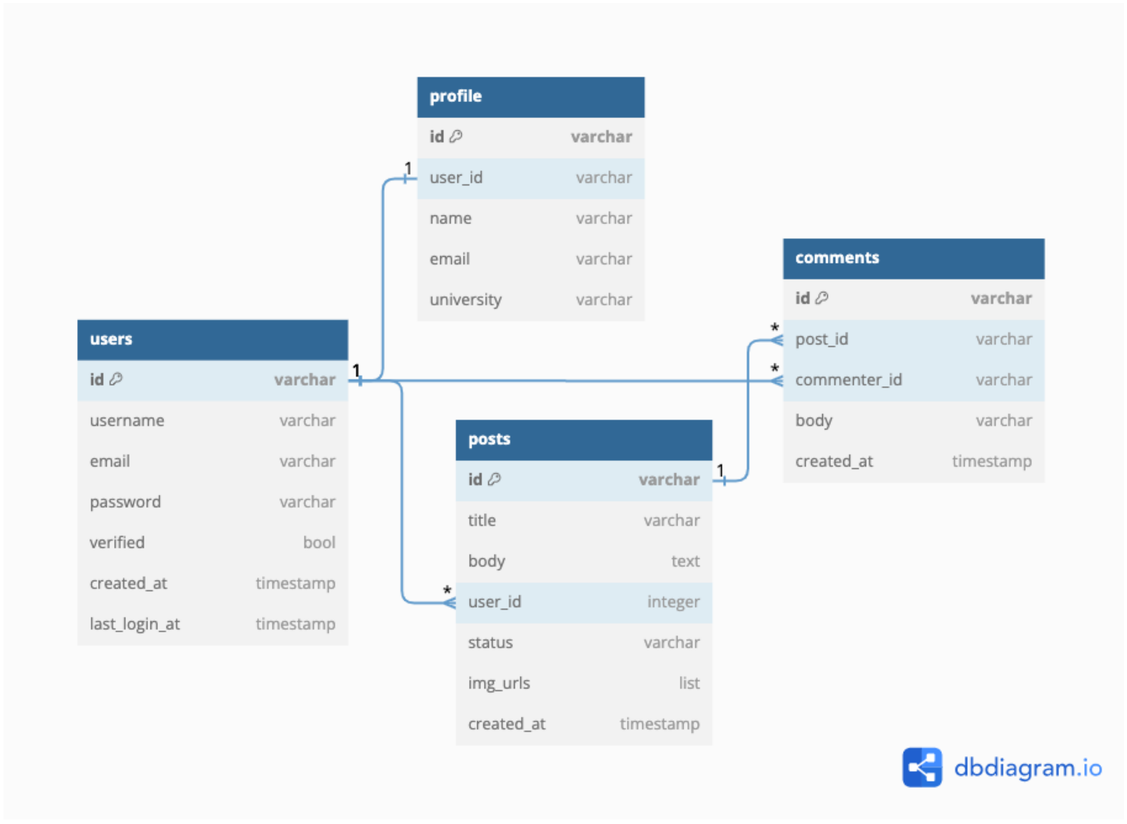
Current password

New password

Account security

Manage your account security.

- Database Design



profile

id	varchar
user_id	varchar
name	varchar
email	varchar
university	varchar

users

id	varchar
username	varchar
email	varchar
password	varchar
verified	bool
created_at	timestamp
last_login_at	timestamp

posts

id	varchar
title	varchar
body	text
user_id	integer
status	varchar
img_urls	list
created_at	timestamp

comments

id	varchar
post_id	varchar
commenter_id	varchar
body	varchar
created_at	timestamp

- ## 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

Django Rest Framework (DRF) handles hashing internally.

Data Security

Password

Strong passwords are enforced.

- (?=.*[a-z]): at least one lowercase letter
- (?=.*[A-Z]): at least one uppercase letter
- (?=.*\d): at least one digit
- (?=.*[@$!%*?&]): at least one special character
- [A-Za-z\d@$!%*?&]{8,12}: a total length of 8 to 12 characters consisting of the allowed character set

Data Sorage

Django Rest Framework (DRF) handles secure database.

Data Transmission

Secure connection HTTPS.

Session Management

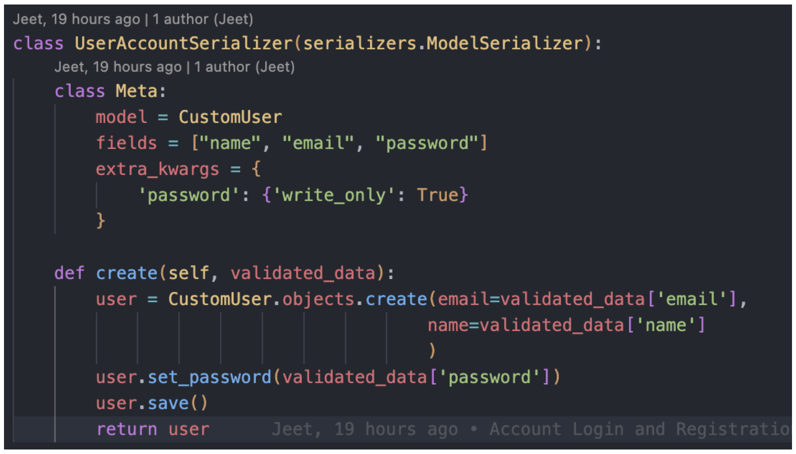
jwtToken.

HTTPS/REST-API Connection

ReactJS using 'axios' for making HTTP requests.

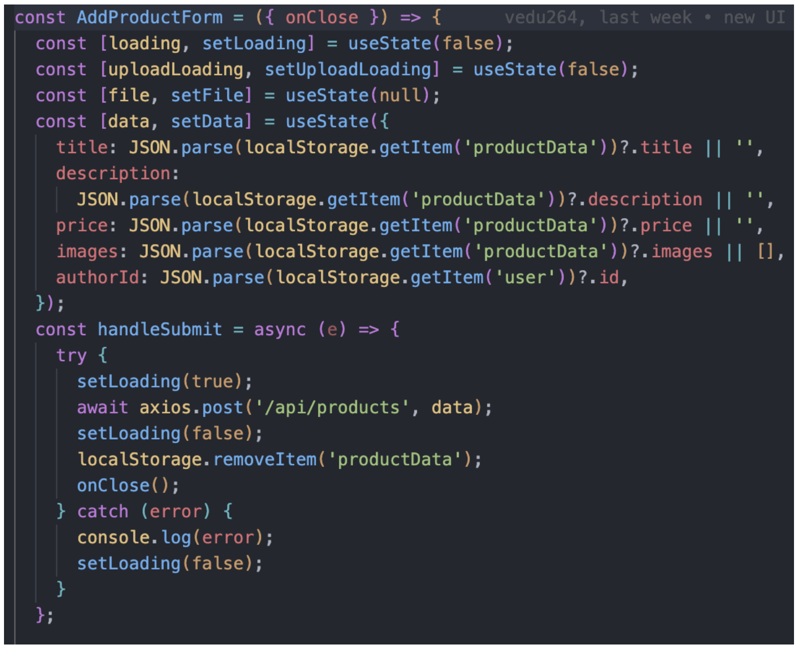
- Business Logic and/or Key Algorithms

User Authentication



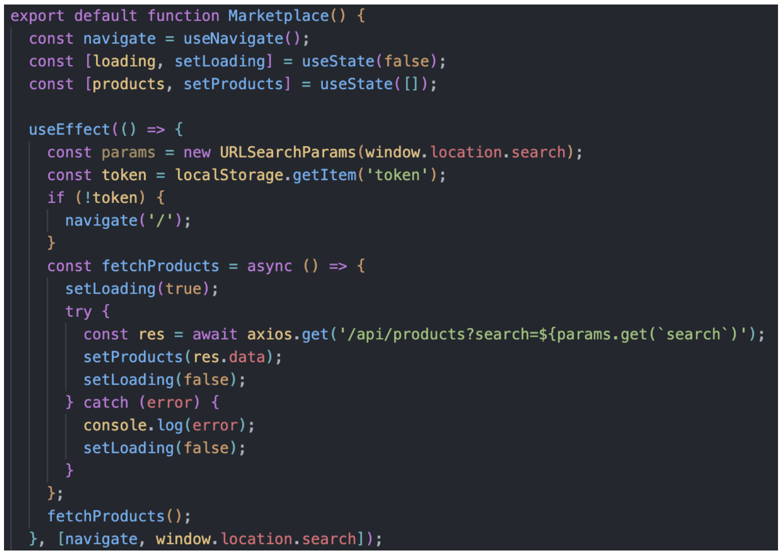
```
Jeet, 19 hours ago | 1 author (Jeet)
class UserAccountSerializer(serializers.ModelSerializer):
    Jeet, 19 hours ago | 1 author (Jeet)
    class Meta:
        model = CustomUser
        fields = ["name", "email", "password"]
        extra_kwargs = {
            'password': {'write_only': True}
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

    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

  Component Composition: Building complex UIs by combining smaller, reusable components.

  Hooks: Utilizing React Hooks like useState and useEffect for state management and side effects in functional components.

  Functional Programming: Emphasizing the use of functional components and possibly pure functions.

  Container/Presentational Components: Separating logic (containers) from UI rendering (presentational components).

  State Hoisting: Managing shared state by lifting it to common ancestor components.

- ## Any Additional Topics you would like to include.

- ## References

- ## Glossary