

A  
Project Report  
on  
**Boxico: Connect Boxers & Trainers**  
**(Your Social Hub for Boxing)**

Developed by  
**Hanshvee Patil - IT111**

BTech-IT, Sem VI

Guided By  
**Prof. Sunil K. Vithlani**  
**Department of Information Technology**  
**Dharmsinh Desai University**



**Department of Information Technology**  
**Faculty of Technology, Dharmsinh Desai University College**

**Road, Nadiad – 387001**

**April - 2022**

**DHARMSINH DESAI UNIVERSITY**  
**NADIAD-387001, GUJARAT**



**CERTIFICATE**

This is to certify that the project carried out in the subject of System Design Practice, entitled “**Boxico: Connect Boxers & Trainers - Your Social Hub for Boxing**” and recorded in this report is a bonafide report of work of

1) Hanshvee Patil                      Roll No. IT111                      ID No: 21ITUOS100

of the Department of Information Technology, semester VI . They were involved in Project work during the academic year 2023 – 2024.

Prof. Sunil K. Vithlani  
(Project Guide)  
Department of Information Technology,  
Faculty of Technology,  
Dharmsinh Desai University, Nadiad  
Date:

## CANDIDATE'S DECLARATION



We declare that the pre-final semester report entitled “**Boxico: Connect Boxers & Trainers - Your Social Hub for Boxing**” is our own work conducted under the supervision of the guide **Prof. Sunil K. Vithlani**.

We further declare that to the best of our knowledge the report for B.Tech. VI semester does not contain part of the work which has been submitted either in this or any other university without proper citation.

Candidate's Signature  
Hanshvee Aditya Patil  
Student ID:21ITUOS100

## ACKNOWLEDGMENT

On the very outset of this report, we would like to extend our sincere & heartfelt obligations towards all the personages who have directly or indirectly helped in this endeavor. Without their active guidance, help, cooperation & encouragement, we wouldn't have made the headway in the project.

We are ineffably indebted to our instructor **Prof. Sunil K. Vithlani** for his guidance, encouragement and conscientiousness to accomplish this project. We are extremely thankful for his support.

We would also like to express our gratitude to our Head, **Dr. Vipul Dabhi** along with Dharmsinh Desai University for giving us this opportunity.

Last but not least, we would also like to acknowledge with a deep sense of reverence, our gratitude towards our family members who have always supported us morally. Any omission in this brief acknowledgement does not mean lack of gratitude.

Thank You,  
Hanshvee Patil

## TABLE OF CONTENT

1. Introduction.....	
Project Details: Broad specifications of the work entrusted to you... ..	
Purpose .....	
Scope .....	
Objective .....	
Technology and Literature Review .....	
2. Project Management... ..	
Feasibility Study.....	
Technical feasibility... ..	
Time Schedule feasibility.....	
Operational feasibility... ..	
Implementation feasibility.....	
Project Planning... ..	
Project Development Approach and Justification... ..	
Project Plan.....	
Roles and Responsibilities.....	
3. System Requirements Study... ..	
Study of Current System... ..	
Problems and Weaknesses of Current System... ..	
User Characteristics .....	
Hardware and Software Requirements... ..	
Constraints.....	
4. System Analysis.....	
Requirements of New System... ..	
Functional Requirements.....	
Non-functional Requirements... ..	

5. System Design...	.....
Use Case Diagram.....	.....
Class Diagram...	.....
Sequence Diagram.....	.....
Data Flow Diagram .....	.....
Component Diagram...	.....
Deployment Diagram...	.....
6. Implementation Planning.....	.....
Implementation Environment...	.....
Program/Modules Specification.....	.....
Coding Snippets...	.....
7. Testing.....	.....
Testing Plan.....	.....
Testing Strategy...	.....
Testing Methods .....	.....
Test Cases.....	.....
8. User Manual.....	.....
9. Limitation and Future Enhancement.....	.....
10. Conclusion and Discussion...	.....
Conclusions .....	.....
Discussion...	.....

# 1. INTRODUCTION

## PROJECT DETAILS

Boxico serves as the ultimate platform for connecting boxers with trainers, streamlining their boxing journey. Boxico makes it easy for boxers to discover and connect with suitable trainers. Users can access essential details to make informed decisions about their training choices. With Boxico, the boxing community can seamlessly navigate their training needs and goals.

## PURPOSE

The purpose of the "Boxers Community" project is to establish a specialised social media platform dedicated to boxing enthusiasts, leveraging the functionalities of the MERN (MongoDB, Express.js, React.js, Node.js) stack. The platform aims to create a vibrant and engaging space where boxers can connect, share experiences, find trainers, and engage in real-time communication.

Utilising MongoDB for flexible data management, Express.js for efficient backend operations, React.js for dynamic user interfaces, and Node.js for real-time communication, "Boxers Community" seeks to redefine social networking within the boxing community. By offering a user-friendly interface and robust features, the platform aims to foster connections, facilitate group interactions, and provide a collaborative environment.

## SCOPE

"Boxers Community" anticipates catering to a diverse boxing community, providing a platform for enthusiasts, amateur boxers, professional fighters, trainers, and boxing aficionados. The platform's scope includes facilitating connections, sharing training insights, discovering trainers, and engaging in real-time conversations related to boxing. As user adoption grows, the platform expects increased user engagement, supporting various discussions and collaborations within the boxing community. "Boxers Community" aims to offer an extensive range of channels, promoting inclusive interactions among users from diverse boxing backgrounds and interests.

## OBJECTIVE

Boxico consists of two primary modules, catering specifically to boxers:

**Trainer Registration Module:** This module allows trainers to register their profiles, specifying their expertise and availability.

**Boxer Module:** This module is exclusively designed for boxers, enabling them to search for and connect with registered trainers based on their preferences and training needs.

## TECHNOLOGY AND LITERATURE REVIEW

The MERN stack—MongoDB, Express.js, React.js, and Node.js—offers a robust foundation for modern web development:

- MongoDB: A flexible NoSQL database for efficient data storage.
- Express.js: A minimalist Node.js framework for building web APIs.
- React.js: A JavaScript library for building dynamic user interfaces.
- Node.js: A server-side JavaScript runtime for scalable network applications.

### Literature Review:

Numerous studies and articles highlight the advantages of using the MERN stack for web development projects:

**Scalability and Performance:** Research has shown that MongoDB's flexible data model and horizontal scaling capabilities contribute to improved scalability and performance of web applications, making it a preferred choice for startups and enterprises alike.

**Developer Productivity:** Express.js and Node.js streamline the development process by providing a simple and consistent programming model, reducing development time and effort. React.js's component-based architecture further enhances developer productivity by promoting code reuse and modularization.

**User Experience:** React.js's virtual DOM and efficient rendering algorithms contribute to a smoother and faster user experience, leading to higher user satisfaction and engagement levels. This can be crucial for applications like Boxico, where user interaction plays a significant role in the success of the platform.

**Community Support and Ecosystem:** The MERN stack benefits from a vibrant and active community of developers, contributing to a rich ecosystem of libraries, frameworks, and tools. This ensures continuous innovation and support, providing developers with access to resources and solutions to address various challenges encountered during the development process.



## **2. PROJECT MANAGEMENT**

### **FEASIBILITY STUDY**

#### **Technical Feasibility**

After considering our project functionalities and technical needs we decided to go with the following technologies :

1. MongoDB: A flexible NoSQL database for efficient data storage.
2. Express.js: A minimalist Node.js framework for building web APIs.
3. React.js: A JavaScript library for building dynamic user interfaces.
4. Node.js: A server-side JavaScript runtime for scalable network applications.

All the above mentioned technologies are open sources therefore it is feasible to use with the knowledge of basics in each of them it would be easy for us to manage the project too.

#### **Time Schedule Feasibility**

The plan for completing our project, Boxico, unfolds as follows:

Initially, all requirements were meticulously gathered and analyzed by the end of December 2023. Following this, I crafted the Software Requirements Specification (SRS) document to outline project objectives and functionalities. Subsequently, I prioritized frontend development, ensuring an intuitive and user-friendly interface to facilitate seamless interactions. This was complemented by the creation of essential diagrams to visualize system architecture and workflow. The coding and unit testing phase spanned 4 weeks, ensuring the reliability and functionality of each component. An additional 2 weeks were dedicated to rigorous system and integration testing to ensure a robust and error-free application. As the sole member of the development team, I successfully completed the project within the estimated timeframe. I am now prepared to demonstrate Boxico during the submission week, accompanied by a comprehensive report detailing its features and accomplishments.

#### **Operational Feasibility**

The user experience of Boxico, our social media app tailored for boxers and trainers, is designed to be intuitive and user-friendly. Its simple yet engaging interface ensures that users can easily navigate the platform without requiring any specialized training. Technical performance is a key focus, ensuring that the system reliably delivers the right information to users, whether it's connecting boxers with trainers or providing pertinent details about training sessions. Leveraging the MERN stack, Boxico is organized to efficiently handle user interactions, ensuring timely access to relevant information through seamless integration with

intranet services.

### **Implementation Feasibility**

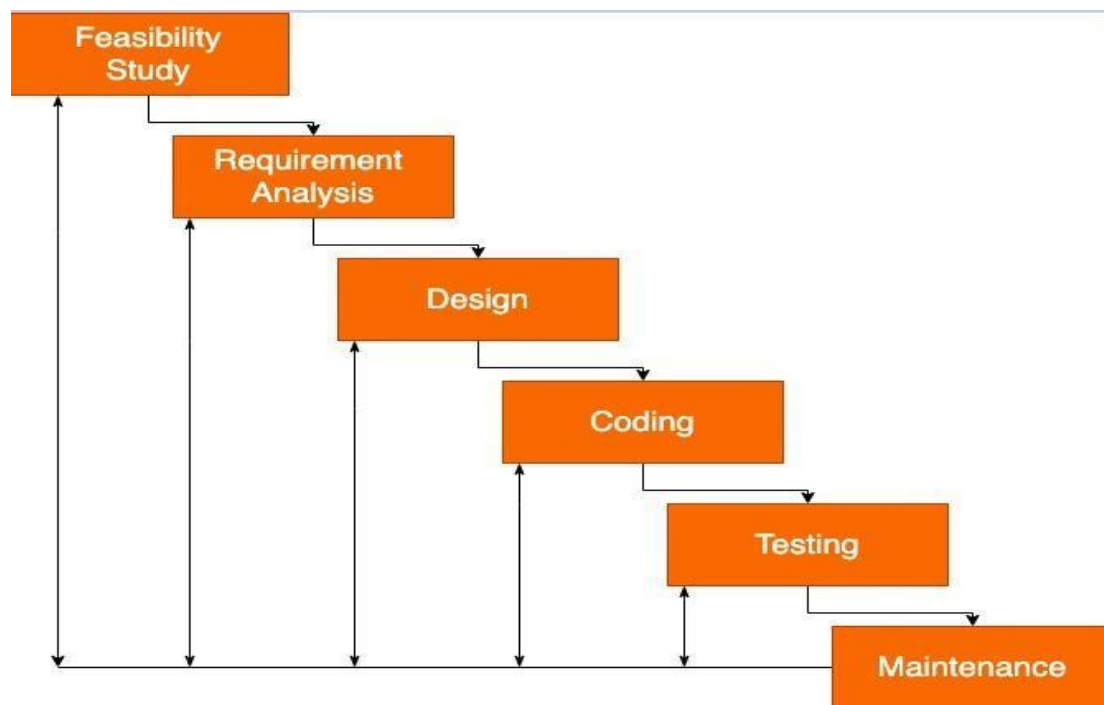
The Functional requirements of our project will be easily fulfilled by the technologies we have mentioned. All the technologies we are using are easily supported by all the Computer System. Hence, the implementation is feasible.

# PROJECT PLANNING

## Project Development Approach and Justification

The software development approach we used for our project : Iterative Waterfall Model This includes the stages like :

1. Feasibility Study
2. Requirement Analysis
3. Design
4. Coding
5. Testing
6. Maintenance



We choose this approach because it gives us the liberty to get back to a stage if it needs some redo. Our entire approach was trial and error and learn as you need. By choosing the iterative waterfall we get the liberty of flexibility and the risk management can be done easily.

- **Project Plan**

<b>Sr. No</b>	<b>Documents</b>	<b>Timeline</b>
<b>1</b>	Project Definition, Submission and approval	<b>December</b>
<b>2</b>	System Design	<b>January</b>
<b>3</b>	Implementation	<b>January</b>
<b>4</b>	Testing	<b>March</b>
<b>5</b>	Documentation	<b>March</b>
<b>6</b>	Submission	<b>April</b>

- **Roles and Responsibility**

<b>Name</b>	<b>Analysis</b>	<b>Design</b>	<b>Coding</b>	<b>Testing</b>	<b>Documentation</b>	<b>Maintenance</b>
<b>Hanshvee</b>	✓	✓	✓	✓	✓	✓

### **3. SYSTEM STUDY REQUIREMENT**

#### **STUDY OF CURRENT SYSTEM**

currently, we are using face recognition technology in criminal detection, Airport, Mobile face-lock, and traffic signals. Also, we are using a biometric fingerprint scanner for taking attendance but that is time-consuming that's why our approach is to make fast attendance system and make the classroom smarter.

#### **PROBLEMS AND WEAKNESS OF CURRENT SYSTEM**

Communication Constraint: The inability for boxers and trainers to chat directly limits personalized interaction and real-time communication, potentially hindering the effectiveness of their training relationship.

Email Verification Issue: Users may encounter difficulties with email verification, which could impede their ability to register or access the platform fully, leading to user frustration and potential dropout rates.

#### **USER CHARACTERISTICS**

- Admin : Manages database and platform activities.
- Trainer : Creates and manages training programs. Guides and supports boxers.
- Boxer : Provides feedback and raises concerns.

#### **HARDWARE AND SOFTWARE REQUIREMENTS**

Software Requirements:

- MERN stack (MongoDB, Express.js, React.js, Node.js)
- Internet connection for accessing online resources, updates, and communication functionalities.

Hardware Requirements:

- Standard devices capable of running the required software.

#### **CONSTRAINTS**

Only registered boxers and coaches can create accounts and post content, limited to text and photos, ensuring authenticity and relevance of shared information.

# **1. SYSTEM ANALYSIS**

## **• FUNCTIONAL REQUIREMENTS**

### **R.1 Registering a user**

**R.1.1:** Users can select a "Register" button on the platform's homepage.

**R.1.2:** Users can fill in required details such as username, email, password, boxing experience, and profile picture.

**R.1.3:** Users receive a confirmation email or message to verify their registration.

### **R.2 Logging in a user**

**R.2.1:** Users can select a "Login" button on the platform's homepage.

**R.2.2:** Users can fill in their registered email/username and password to access their accounts.

### **R.3 Creating and Sharing Posts**

**R.3.1:** Users can access a "Create Post" feature from their profile or the homepage.

**R.3.2:** Users can upload images/videos, add captions, include hashtags, and share posts with followers.

### **R.4 Interacting with Posts**

**R.4.1:** Users can view posts from others in their news feed.

**R.4.2:** Users can like and comment within the platform.

### **R.5 Searching for Boxers or Posts**

**R.5.1:** Users can use the search bar to find specific boxers or posts.

**R.5.2:** Users can enter keywords, usernames, or hashtags to fetch relevant profiles or content.

### **R.6 Admin Tools for Content Moderation**

**R.6.1:** Admins can access tools to moderate user-generated content, flag inappropriate posts, and manage user accounts.

### **R.7 Finding Trainers**

**R.7.1:** Users can access a "Find Trainers" section either from the homepage or navigation menu.

**R.7.2:** Users can search for trainers based on criteria such as location, expertise, experience level, specialty, or availability.

- **Non-Functional Requirements**

**NFR.1 Performance Requirements:**

**NFR.2 Safety Requirements:**

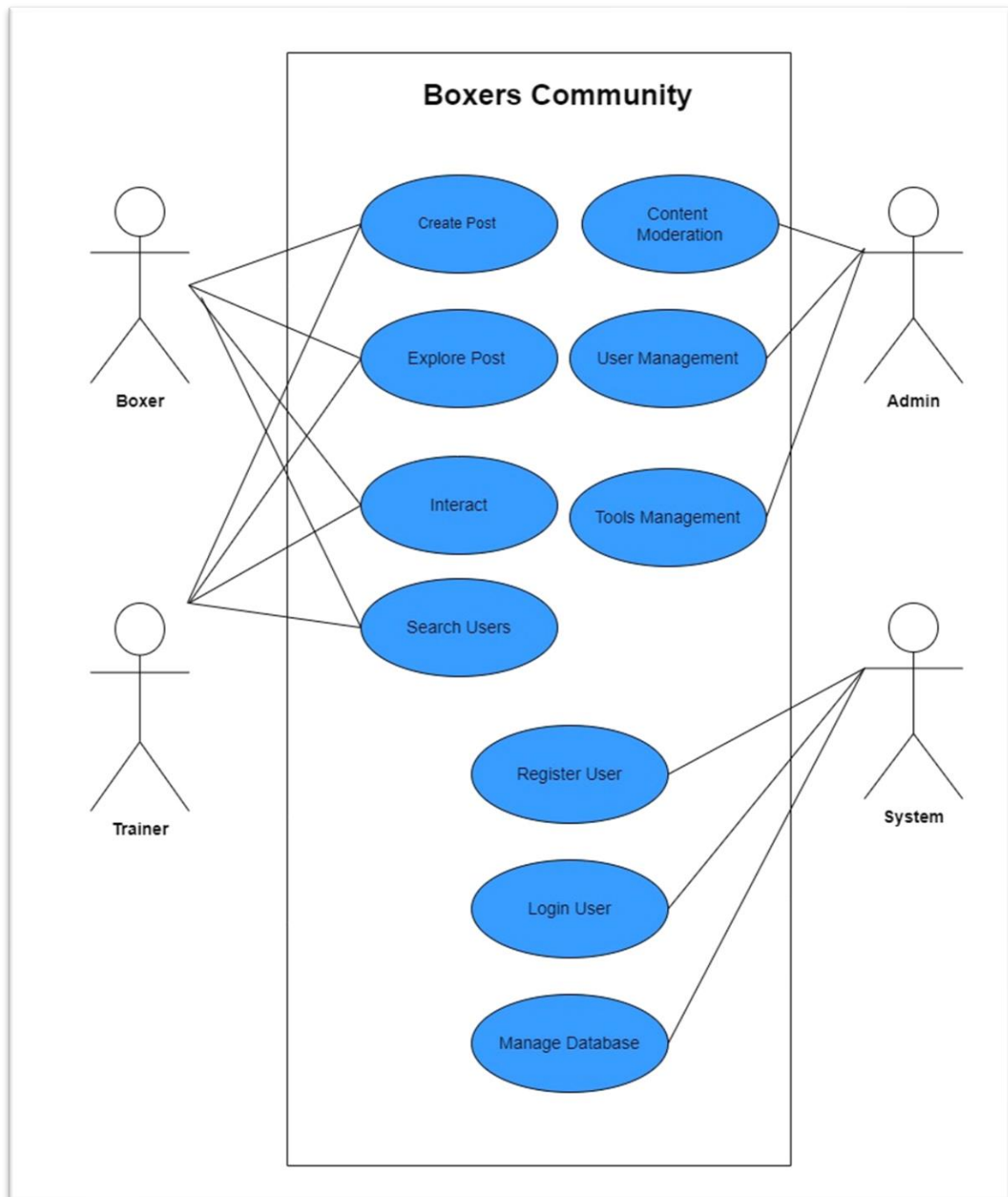
**NFR.3 Security Requirements:**

**NFR.4 Software Quality Attributes:**

**NFR.5 Business Rules:**

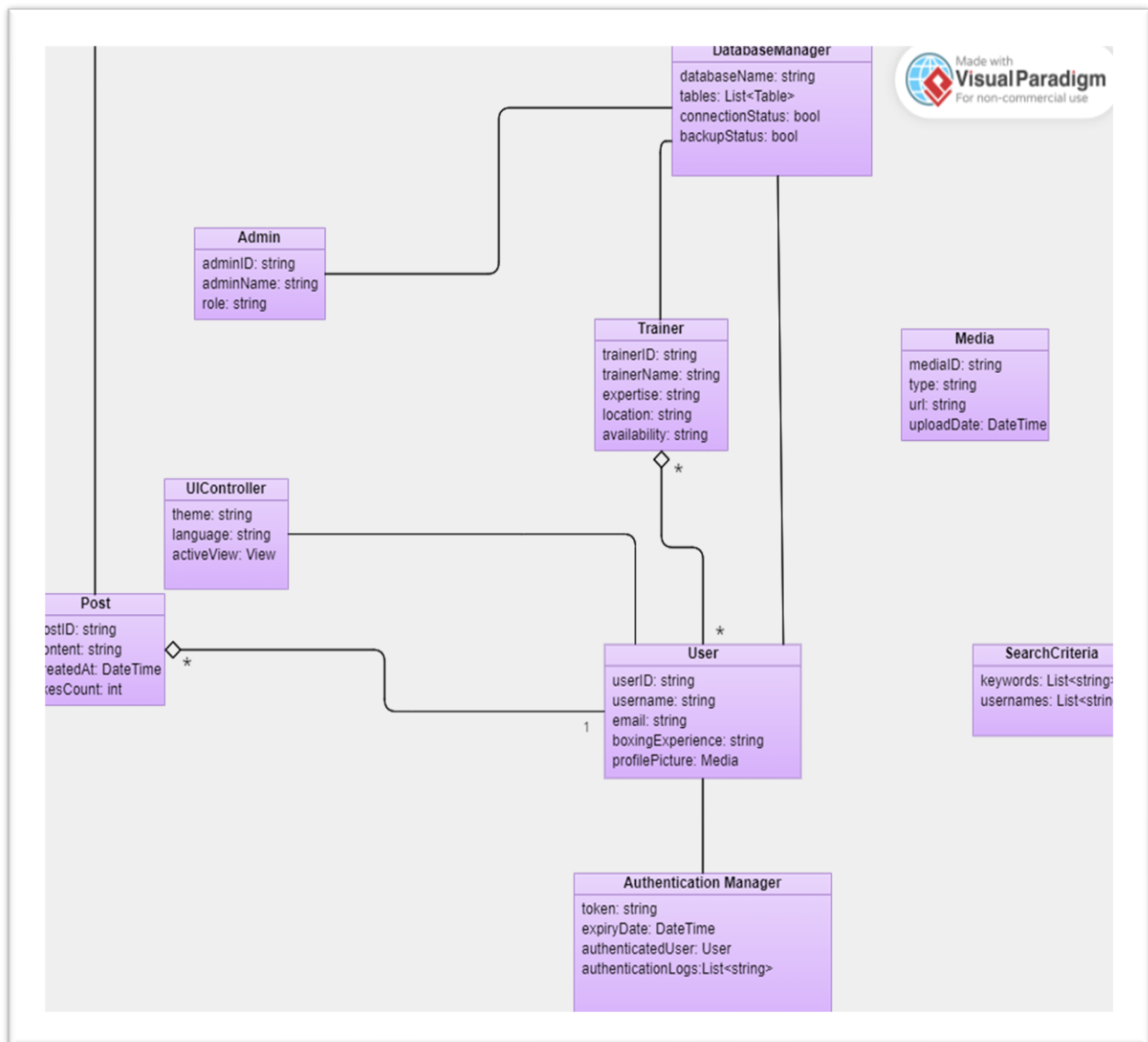
## 2. SYSTEM DESIGN

- **USE CASE DIAGRAM**





- **CLASS DIAGRAM**



## 6. IMPLEMENTATION PLANNING

### IMPLEMENTATION ENVIRONMENT:

- **Version Control Integration:** Utilize Git for version control, integrating it with Visual Studio Code for easy management of changes.
- **Linting and Formatting:** Use ESLint with Prettier to ensure consistent code style and catch errors early in development. You can configure Visual Studio Code to automatically format code on save.
- **Debugging:** Set up debugging configurations in Visual Studio Code for both frontend and backend to easily debug your application.
- **Environment Variables:** Utilize the dotenv package to manage environment variables across your project. This allows you to securely store sensitive information like API keys or database credentials.
- **Frontend Development:** Use react-scripts for frontend development. You can start the development server using npm start and build the production-ready version of your frontend using npm run build.
- **Backend Development:** Use nodemon to automatically restart the server when changes are made to backend files. Start the backend server using npm start.
- **Routing:** Utilize react-router-dom for client-side routing in the frontend, allowing for a single-page application experience.
- **State Management:** Use react-redux and @reduxjs/toolkit for state management in the frontend, enabling efficient data flow and global state management.
- **API Integration:** Implement APIs in the backend using Express.js. Utilize axios or fetch in the frontend to interact with these APIs.
- **Database Connectivity:** Connect your backend to MongoDB using mongoose. Implement models and schemas to define the structure of your data.
- **Authentication and Authorization:** Implement user authentication and authorization using JSON Web Tokens (JWT) and the jsonwebtoken package.
- **File Uploads:** Implement file uploads using multer and multer-gridfs-storage for storing files in MongoDB GridFS.
- **Security:** Utilize packages like helmet and cors to enhance the security of your Express.js server by setting various HTTP headers and handling cross-origin resource sharing.

- **Logging:** Implement logging in your backend using morgan to track HTTP requests and responses for debugging and monitoring purposes.
- **Testing:** Set up testing frameworks such as Jest and React Testing Library for unit and integration testing of both frontend and backend components

## **PROGRAM / MODULES SPECIFICATION:**

- **Frontend Models:**

1. **Login Page:**  
Input fields for email and password.  
Validation for email and password fields.  
Submit button to authenticate the user.  
Error message display for invalid credentials.
2. **Signup Page:**  
Input fields for first name, last name, email, password, occupation, location, and profile photo upload.  
Validation for all input fields.  
Submit button to register the user.  
Error message display for invalid input or existing email.  
Success message upon successful registration.
3. **Homepage:**  
Divided into three sections: left, right, and center.  
Left section displays user profile details.  
Center section displays a feed of posts from different users.  
Right section displays sponsorship images and friends.  
Pagination or infinite scroll for navigating through posts. Like, comment, and share functionalities for each post.  
Option to create a new post.
4. **Navbar:**  
Dark/light mode toggle button.  
User profile name with a dropdown menu.  
Dropdown menu includes options such as profile page, settings, and logout.
5. **Profile Page:**  
Display user profile details such as name, occupation, location, and profile photo.  
Option to edit profile details.  
Display user's posts with options to edit or delete each post.  
Option to upload a new profile photo.  
Back button to navigate to the homepage.

- **Backend Models:**

- 1. User Model:

- Fields: first name, last name, email, password (hashed), occupation, location, profile photo.

- Validation for unique email and required fields.

- Methods for user authentication and password hashing.

- 2. Post Model:

- Fields: user reference (to link with the user who created the post), content, timestamp.

- Validation for required fields.

- Methods for CRUD operations on posts.

## **CODING SNIPPETS**

While writing our code we took the utmost care to follow the basic coding standards while writing a Python code like,

- Following proper naming conventions of local variables, global variables, constants and functions.
- Doing proper indentation.
- Proper error handling.
- Adding comments for better understanding.
- Proper Syntax

Here are some of the code snippets written during implementation,  
**Front End Snippets**

## Frontend

Components:

HomePage

```
mern-social-media-master > client > src > scenes > homePage > index.jsx > ...
1  import { Box, useMediaQuery } from "@mui/material";
2  import { useSelector } from "react-redux";
3  import Navbar from "scenes/navbar";
4  import UserWidget from "scenes/widgets/UserWidget";
5  import MyPostWidget from "scenes/widgets/MyPostWidget";
6  import PostsWidget from "scenes/widgets/PostsWidget";
7  import AdvertWidget from "scenes/widgets/AdvertWidget";
8  import FriendListWidget from "scenes/widgets/FriendListWidget";
9
10 const HomePage = () => {
11   const isNonMobileScreens = useMediaQuery("(min-width:1000px)");
12   const { _id, picturePath } = useSelector((state) => state.user);
13
14   return (
15     <Box>
16       <Navbar />
17       <Box
18         width="100%"
19         padding="2rem 6%"
20         display={isNonMobileScreens ? "flex" : "block"}
21         gap="0.5rem"
22         justifyContent="space-between"
23       >
24         <Box flexBasis={isNonMobileScreens ? "26%" : undefined}>
25           <UserWidget userId={_id} picturePath={picturePath} />
26         </Box>
27         <Box
28           flexBasis={isNonMobileScreens ? "42%" : undefined}
29           mt={isNonMobileScreens ? undefined : "2rem"}
30         >
31           <MyPostWidget picturePath={picturePath} />
32           <PostsWidget userId={_id} />
33         </Box>
34         {isNonMobileScreens && (
35           <Box flexBasis="26%">
36             <AdvertWidget />
37             <Box m="2rem 0" />
38             <FriendListWidget userId={_id} />
39           </Box>
40         )}
41       </Box>
42     </Box>
43   );
44 };
45 export default HomePage;
```

Login Page:

//form.jsx

```
mem-social-media-master / client / src / scenes / loginPage / Form.jsx / ...
1  import { useState } from "react";
2  import {
3    Box,
4    Button,
5    TextField,
6    useMediaQuery,
7    Typography,
8    useTheme,
9  } from "@mui/material";
10 import EditOutlinedIcon from "@mui/icons-material/EditOutlined";
11 import { Formik } from "formik";
12 import * as yup from "yup";
13 import { useNavigate } from "react-router-dom";
14 import { useDispatch } from "react-redux";
15 import { setLogin } from "state";
16 import Dropzone from "react-dropzone";
17 import FlexBetween from "components/FlexBetween";
18
19 const registerSchema = yup.object().shape({
20   firstName: yup.string().required("required"),
21   lastName: yup.string().required("required"),
22   email: yup.string().email("invalid email").required("required"),
23   password: yup.string().required("required"),
24   location: yup.string().required("required"),
25   occupation: yup.string().required("required"),
26   picture: yup.string().required("required"),
27 });
28
29 const loginSchema = yup.object().shape({
30   email: yup.string().email("invalid email").required("required"),
31   password: yup.string().required("required"),
32 });
33
34 const initialValuesRegister = {
35   firstName: "",
36   lastName: "",
37   email: "",
38   password: "",
39   location: "",
40   occupation: "",
41   picture: "",
42 };
43
44 const initialValuesLogin = {
45   email: "",
```

mern-social-media-master > client > src > scenes > loginPage > Form.jsx > ...

```
44 const initialValuesLogin = {
45   email: "",
46   password: "",
47 };
48
49 const Form = () => {
50   const [pageType, setPageType] = useState("login");
51   const { palette } = useTheme();
52   const dispatch = useDispatch();
53   const navigate = useNavigate();
54   const isNonMobile = useMediaQuery("(min-width:600px)");
55   const islogin = pageType === "login";
56   const isRegister = pageType === "register";
57
58   const register = async (values, onSubmitProps) => {
59     // this allows us to send form info with image
60     const formData = new FormData();
61     for (let value in values) {
62       formData.append(value, values[value]);
63     }
64     formData.append("picturePath", values.picture.name);
65
66     const savedUserResponse = await fetch(
67       "http://localhost:3001/auth/register",
68       {
69         method: "POST",
70         body: formData,
71       }
72     );
73     const savedUser = await savedUserResponse.json();
74     onSubmitProps.resetForm();
75
76     if (savedUser) {
77       setPageType("login");
78     }
79   };
80
81   const login = async (values, onSubmitProps) => {
82     const loggedInResponse = await fetch("http://localhost:3001/auth/login", {
83       method: "POST",
84       headers: { "Content-Type": "application/json" },
85       body: JSON.stringify(values),
86     });
87     const loggedIn = await loggedInResponse.json();
88     onSubmitProps.resetForm();
```



```

mern-social-media-master > client > src > scenes > loginPage > Form.jsx > ...
49  const Form = () => {
81  const login = async (values, onSubmitProps) => {
88      onSubmitProps.resetForm();
89      if (loggedIn) {
90          dispatch(
91              setLogin({
92                  user: loggedIn.user,
93                  token: loggedIn.token,
94              })
95          );
96          navigate("/home");
97      }
98  };
99
100  const handleFormSubmit = async (values, onSubmitProps) => {
101      if (isLogin) await login(values, onSubmitProps);
102      if (isRegister) await register(values, onSubmitProps);
103  };
104
105  return (
106      <Formik
107          onSubmit={handleFormSubmit}
108          initialValues={isLogin ? initialValuesLogin : initialValuesRegister}
109          validationSchema={isLogin ? loginSchema : registerSchema}
110      >
111          {{
112              values,
113              errors,
114              touched,
115              handleBlur,
116              handleChange,
117              handleSubmit,
118              setFieldValue,
119              resetForm,
120          }} => (
121              <form onSubmit={handleSubmit}>
122                  <Box
123                      display="grid"
124                      gap="30px"
125                      gridTemplateColumns="repeat(4, minmax(0, 1fr))"
126                      sx={{
127                          "& > div": { gridColumn: isNonMobile ? undefined : "span 4" },
128                      }}
129                  >
130                      {isRegister && (

```

mern-social-media-master > client > src > scenes > loginPage > Form.jsx > ...

```
49   const Form = () => {
111     {{{
131     <>
132       <TextField
133         label="First Name"
134         onBlur={handleBlur}
135         onChange={handleChange}
136         value={values.firstName}
137         name="firstName"
138         error={
139           Boolean(touched.firstName) && Boolean(errors.firstName)
140         }
141         helperText={touched.firstName && errors.firstName}
142         sx={{ gridColumn: "span 2" }}
143       />
144       <TextField
145         label="Last Name"
146         onBlur={handleBlur}
147         onChange={handleChange}
148         value={values.lastName}
149         name="lastName"
150         error={Boolean(touched.lastName) && Boolean(errors.lastName)}
151         helperText={touched.lastName && errors.lastName}
152         sx={{ gridColumn: "span 2" }}
153       />
154       <TextField
155         label="Location"
156         onBlur={handleBlur}
157         onChange={handleChange}
158         value={values.location}
159         name="location"
160         error={Boolean(touched.location) && Boolean(errors.location)}
161         helperText={touched.location && errors.location}
162         sx={{ gridColumn: "span 4" }}
163       />
164       <TextField
165         label="Occupation"
166         onBlur={handleBlur}
167         onChange={handleChange}
168         value={values.occupation}
169         name="occupation"
170         error={
171           Boolean(touched.occupation) && Boolean(errors.occupation)
172         }
173         helperText={touched.occupation && errors.occupation}
```

mem-social-media-master > client > src > scenes > loginPage > Form.jsx > ...

```
49   const Form = () => {
111     {{
173       helperText={touched.occupation && errors.occupation}
174       sx={{ gridColumn: "span 4" }}
175     } />
176     <Box
177       gridColumn="span 4"
178       border={`1px solid ${palette.neutral.medium}`}
179       borderRadius="5px"
180       p="1rem"
181     >
182       <Dropzone
183         acceptedFiles=".jpg,.jpeg,.png"
184         multiple={false}
185         onDrop={(acceptedFiles) =>
186           setFieldValue("picture", acceptedFiles[0])
187         }
188       >
189         {{({ getRootProps, getInputProps }) => (
190           <Box
191             {...getRootProps()}
192             border={`2px dashed ${palette.primary.main}`}
193             p="1rem"
194             sx={{ "&:hover": { cursor: "pointer" } }}
195           >
196             <input {...getInputProps()} />
197             {!values.picture ? (
198               <p>Add Picture Here</p>
199             ) : (
200               <FlexBetween>
201                 <Typography>{values.picture.name}</Typography>
202                 <EditOutlinedIcon />
203               </FlexBetween>
204             )}
205           </Box>
206         )}
207       </Dropzone>
208     </Box>
209   </>
210 }}
211
212   <TextField
213     label="Email"
214     onBlur={handleBlur}
215     onChange={handleChange}
```

```

mern-social-media-master > client > src > scenes > loginPage > Form.jsx > ...
  49  const Form = () => {
111    {{{
215      onChange={handleChange}
216      value={values.email}
217      name="email"
218      error={Boolean(touched.email) && Boolean(errors.email)}
219      helperText={touched.email && errors.email}
220      sx={{ gridColumn: "span 4" }}
221    />
222    <TextField
223      label="Password"
224      type="password"
225      onBlur={handleBlur}
226      onChange={handleChange}
227      value={values.password}
228      name="password"
229      error={Boolean(touched.password) && Boolean(errors.password)}
230      helperText={touched.password && errors.password}
231      sx={{ gridColumn: "span 4" }}
232    />
233  </Box>
234
235  { /* BUTTONS */ }
236  <Box>
237    <Button
238      fullWidth
239      type="submit"
240      sx={{
241        m: "2rem 0",
242        p: "1rem",
243        backgroundColor: palette.primary.main,
244        color: palette.background.alt,
245        "&:hover": { color: palette.primary.main },
246      }}
247    >
248      {isLogin ? "LOGIN" : "REGISTER"}
249    </Button>
250    <Typography
251      onClick={() => {
252        setPageType(isLogin ? "register" : "login");
253        resetForm();
254      }}
255      sx={{
256        textDecoration: "underline",
257        color: palette.primary.main,

```

```

257         color: palette.primary.main,
258         "&:hover": {
259             cursor: "pointer",
260             color: palette.primary.light,
261         },
262     }}
263     >
264     {isLogin
265       ? "Don't have an account? Sign Up here."
266       : "Already have an account? Login here."}
267     </Typography>
268   </Box>
269 </form>
270 )}
271 </Formik>
272 );
273 };
274
275 export default Form;
276

```

//index.jsx

```
mern-social-media-master > client > src > scenes > loginPage > index.jsx > ...
1  import { Box, Typography, useTheme, useMediaQuery } from "@mui/material";
2  import Form from "../Form";
3
4  const LoginPage = () => {
5    const theme = useTheme();
6    const isNonMobileScreens = useMediaQuery("(min-width: 1000px)");
7    return (
8      <Box>
9        <Box
10          width="100%"
11          backgroundColor={theme.palette.background.alt}
12          p="1rem 6%"
13          textAlign="center"
14        >
15          <Typography fontWeight="bold" fontSize="32px" color="primary">
16            Boxico
17          </Typography>
18        </Box>
19
20        <Box
21          width={isNonMobileScreens ? "50%" : "93%"}
22          p="2rem"
23          m="2rem auto"
24          borderRadius="1.5rem"
25          backgroundColor={theme.palette.background.alt}
26        >
27          <Typography fontWeight="500" variant="h5" sx={{ mb: "1.5rem" }}>
28            Welcome to Boxico, where boxers connect with their trainers and take charge of their own training journey!
29          </Typography>
30          <Form />
31        </Box>
32      </Box>
33    );
34  };
35
36  export default LoginPage;
37
```

Navbar:

```
mern-social-media-master > client > src > scenes > navbar > index.jsx > ...
1  import { useState } from "react";
2  import {
3    Box,
4    IconButton,
5    InputBase,
6    Typography,
7    Select,
8    MenuItem,
9    FormControl,
10   useTheme,
11   useMediaQuery,
12 } from "@mui/material";
13 import {
14   Search,
15   Message,
16   DarkMode,
17   LightMode,
18   Notifications,
19   Help,
20   Menu,
21   Close,
22 } from "@mui/icons-material";
23 import { useDispatch, useSelector } from "react-redux";
24 import { setMode, setLogout } from "state";
25 import { useNavigate } from "react-router-dom";
26 import FlexBetween from "components/FlexBetween";
27
28 const Navbar = () => {
29   const [isMobileMenuToggled, setIsMobileMenuToggled] = useState(false);
30   const dispatch = useDispatch();
31   const navigate = useNavigate();
32   const user = useSelector((state) => state.user);
33   const isNonMobileScreens = useMediaQuery("(min-width: 1000px)");
34
35   const theme = useTheme();
36   const neutrallight = theme.palette.neutral.light;
37   const dark = theme.palette.neutral.dark;
38   const background = theme.palette.background.default;
39   const primaryLight = theme.palette.primary.light;
40   const alt = theme.palette.background.alt;
41
42   const fullName = `${user.firstName} ${user.lastName}`;
43
44   return (
45     <FlexBetween padding="1rem 6%" backgroundColor={alt}>
```

mern-social-media-master > client > src > scenes > navbar > index.jsx > ...

```
28  const Navbar = () => {
44    return (
45      <FlexBetween padding="1rem 6%" backgroundColor={alt}>
46        <FlexBetween gap="1.75rem">
47          <Typography
48            fontWeight="bold"
49            fontSize="clamp(1rem, 2rem, 2.25rem)"
50            color="primary"
51            onClick={() => navigate("/home")}
52            sx={{
53              "&:hover": {
54                color: primaryLight,
55                cursor: "pointer",
56              },
57            }}
58          >
59            Boxico
60          </Typography>
61          {isNonMobileScreens && (
62            <FlexBetween
63              backgroundColor={neutrallight}
64              borderRadius="9px"
65              gap="3rem"
66              padding="0.1rem 1.5rem"
67            >
68              <InputBase placeholder="Search..." />
69              <IconButton>
70                <Search />
71              </IconButton>
72            </FlexBetween>
73          )}
74        </FlexBetween>
75
76        {/* DESKTOP NAV */}
77        {isNonMobileScreens ? (
78          <FlexBetween gap="2rem">
79            <IconButton onClick={() => dispatch(setMode())}>
80              {theme.palette.mode === "dark" ? (
81                <DarkMode sx={{ fontSize: "25px" }} />
82              ) : (
83                <LightMode sx={{ color: dark, fontSize: "25px" }} />
84              )}
85            </IconButton>
86            <Message sx={{ fontSize: "25px" }} />
87            <Notifications sx={{ fontSize: "25px" }} />

```



```

mern-social-media-master > client > src > scenes > navbar > index.jsx > ...
28  const Navbar = () => {
87      <Notifications sx={{ fontSize: "25px" }} />
88      <Help sx={{ fontSize: "25px" }} />
89      <FormControl variant="standard" value={fullName}>
90          <Select
91              value={fullName}
92              sx={{
93                  backgroundColor: neutrallight,
94                  width: "150px",
95                  borderRadius: "0.25rem",
96                  p: "0.25rem 1rem",
97                  "& .MuiSvgIcon-root": {
98                      pr: "0.25rem",
99                      width: "3rem",
100                  },
101                  "& .MuiSelect-select:focus": {
102                      backgroundColor: neutrallight,
103                  },
104              }}
105              input={<InputBase />}
106          >
107              <MenuItem value={fullName}>
108                  <Typography>{fullName}</Typography>
109              </MenuItem>
110              <MenuItem onClick={() => dispatch(setLogout())}>Log Out</MenuItem>
111          </Select>
112      </FormControl>
113  </FlexBetween>
114  ) : (
115      <IconButton
116          onClick={() => setIsMobileMenuToggled(!isMobileMenuToggled)}
117      >
118          <Menu />
119      </IconButton>
120  )}
121
122  {/* MOBILE NAV */}
123  {!isNonMobileScreens && isMobileMenuToggled && (
124      <Box
125          position="fixed"
126          right="0"
127          bottom="0"
128          height="100%"
129          zIndex="10"
130          maxWidth="500px"

```

```

mern-social-media-master > client > src > scenes > navbar > index.jsx > ...
28  const Navbar = () => {
130      maxWidth="500px"
131      minWidth="300px"
132      backgroundColor={background}
133  >
134      {/* CLOSE ICON */}
135      <Box display="flex" justifyContent="flex-end" p="1rem">
136          <IconButton
137              onClick={() => setIsMobileMenuToggled(!isMobileMenuToggled)}
138          >
139              <Close />
140          </IconButton>
141      </Box>
142
143      {/* MENU ITEMS */}
144      <FlexBetween
145          display="flex"
146          flexDirection="column"
147          justifyContent="center"
148          alignItems="center"
149          gap="3rem"
150      >
151          <IconButton
152              onClick={() => dispatch(setMode())}
153              sx={{ fontSize: "25px" }}
154          >
155              {theme.palette.mode === "dark" ? (
156                  <DarkMode sx={{ fontSize: "25px" }} />
157              ) : (
158                  <LightMode sx={{ color: dark, fontSize: "25px" }} />
159              )}
160          </IconButton>
161          <Message sx={{ fontSize: "25px" }} />
162          <Notifications sx={{ fontSize: "25px" }} />
163          <Help sx={{ fontSize: "25px" }} />
164          <FormControl variant="standard" value={fullName}>
165              <Select
166                  value={fullName}
167                  sx={{
168                      backgroundColor: neutrallight,
169                      width: "150px",
170                      borderRadius: "0.25rem",
171                      p: "0.25rem 1rem",
172                      "& .MuiSvgIcon-root": {
173                          pr: "0.25rem",

```

```

mern-social-media-master > client > src > scenes > navbar > index.jsx > ...
28  const Navbar = () => {
162      <Help sx={{ fontSize: "25px" }} />
163      <FormControl variant="standard" value={fullName}>
164      <Select
165        value={fullName}
166        sx={{
167          backgroundColor: neutrallight,
168          width: "150px",
169          borderRadius: "0.25rem",
170          p: "0.25rem 1rem",
171          "& .MuiSvgIcon-root": {
172            pr: "0.25rem",
173            width: "3rem",
174          },
175          "& .MuiSelect-select:focus": {
176            backgroundColor: neutrallight,
177          },
178        }}
179      >
180      <MenuItem value={fullName}>
181        <Typography>{fullName}</Typography>
182      </MenuItem>
183      <MenuItem onClick={() => dispatch(setLogout())}>
184        Log Out
185      </MenuItem>
186    </Select>
187  </FormControl>
188  </FlexBetween>
189  </Box>
190  )}
191  </FlexBetween>
192  );
193  };
194
195  export default Navbar;
196
197
198

```

## Profile Page:

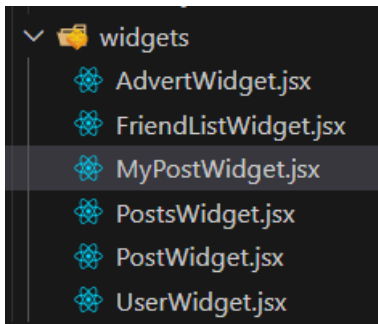
```
mern-social-media-master > client > src > scenes > profilePage > index.jsx > ...
1  import { Box, useMediaQuery } from "@mui/material";
2  import { useEffect, useState } from "react";
3  import { useSelector } from "react-redux";
4  import { useParams } from "react-router-dom";
5  import Navbar from "scenes/navbar";
6  import FriendListWidget from "scenes/widgets/FriendListWidget";
7  import MyPostWidget from "scenes/widgets/MyPostWidget";
8  import PostsWidget from "scenes/widgets/PostsWidget";
9  import UserWidget from "scenes/widgets/UserWidget";
10
11  const ProfilePage = () => {
12    const [user, setUser] = useState(null);
13    const { userId } = useParams();
14    const token = useSelector((state) => state.token);
15    const isNonMobileScreens = useMediaQuery("(min-width:1000px)");
16
17    const getUser = async () => {
18      const response = await fetch(`http://localhost:3001/users/${userId}`, {
19        method: "GET",
20        headers: { Authorization: `Bearer ${token}` },
21      });
22      const data = await response.json();
23      setUser(data);
24    };
25
26    useEffect(() => {
27      getUser();
28    }, []); // eslint-disable-line react-hooks/exhaustive-deps
29
30    if (!user) return null;
31
32    return (
33      <Box>
34        <Navbar />
35        <Box
36          width="100%"
37          padding="2rem 6%"
38          display={isNonMobileScreens ? "flex" : "block"}
39          gap="2rem"
40          justifyContent="center"
41        >
42          <Box flexBasis={isNonMobileScreens ? "26%" : undefined}>
43            <UserWidget userId={userId} picturePath={user.picturePath} />
44            <Box m="2rem 0" />
45            <FriendListWidget userId={userId} />
46          </Box>
47          <Box
48            flexBasis={isNonMobileScreens ? "42%" : undefined}
49            mt={isNonMobileScreens ? undefined : "2rem"}
50          >
51            <MyPostWidget picturePath={user.picturePath} />
52            <Box m="2rem 0" />
53            <PostsWidget userId={userId} isProfile />
54          </Box>
55        </Box>
56      </Box>
57    );
58  };
59
60  export default ProfilePage;
```

```

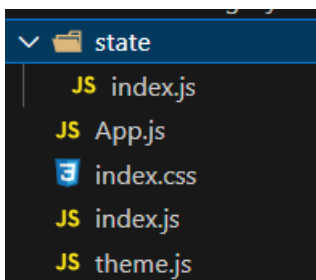
36     width="100%"
37     padding="2rem 6%"
38     display={isNonMobileScreens ? "flex" : "block"}
39     gap="2rem"
40     justifyContent="center"
41   >
42     <Box flexBasis={isNonMobileScreens ? "26%" : undefined}>
43       <UserWidget userId={userId} picturePath={user.picturePath} />
44       <Box m="2rem 0" />
45       <FriendListWidget userId={userId} />
46     </Box>
47     <Box
48       flexBasis={isNonMobileScreens ? "42%" : undefined}
49       mt={isNonMobileScreens ? undefined : "2rem"}
50     >
51       <MyPostWidget picturePath={user.picturePath} />
52       <Box m="2rem 0" />
53       <PostsWidget userId={userId} isProfile />
54     </Box>
55   </Box>
56 </Box>
57 );
58 };
59
60 export default ProfilePage;
61

```

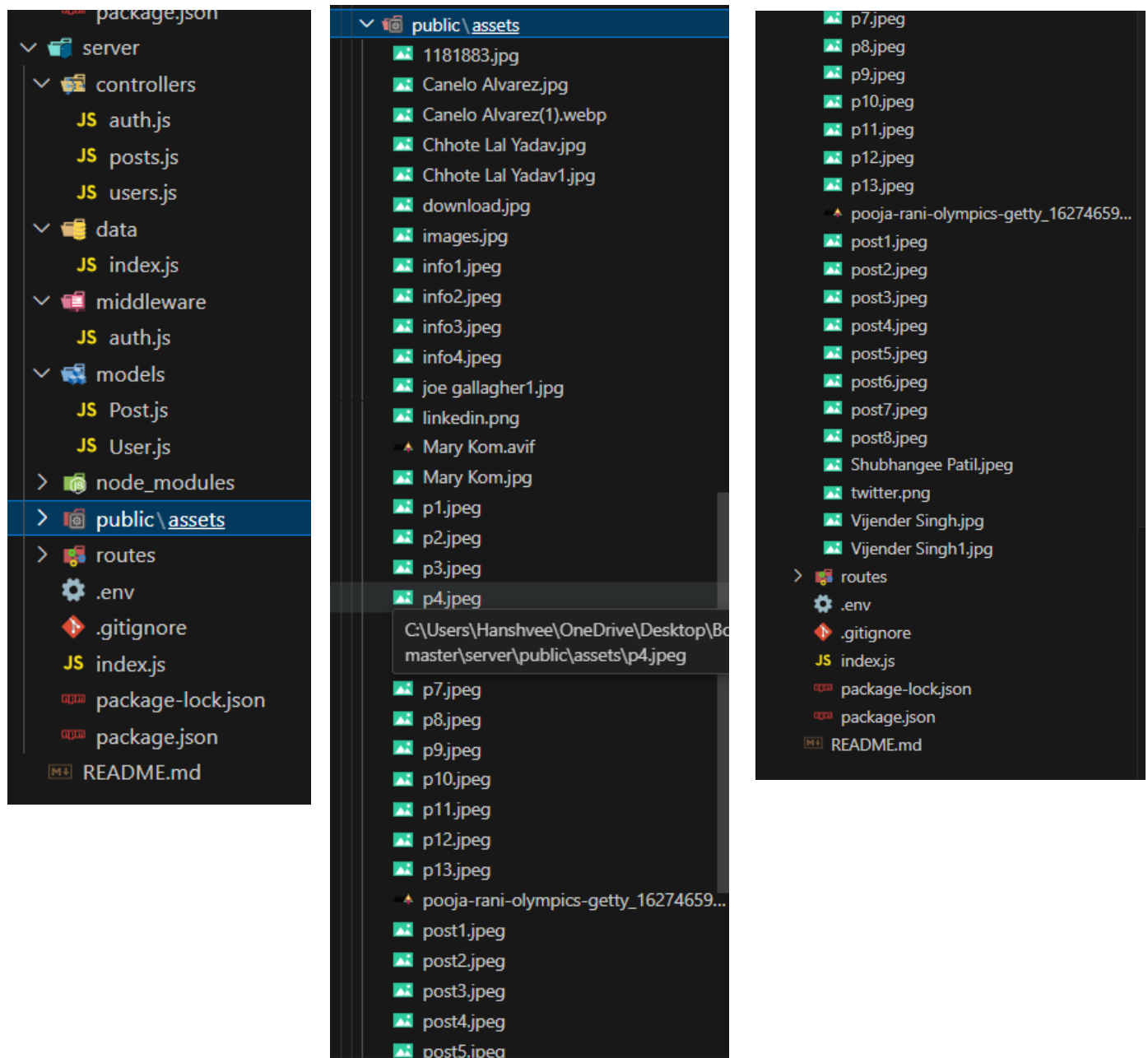
## Widgets:



## State:



## Backend File Structure:



# 1. TESTING

## TESTING PLAN

The method of testing used in this project is White Box testing as in White Box testing, the tester has the complete knowledge and understanding of the code.

Internal program logic exercised using – white box test case design Techniques.

Software requirement are exercised using – block box test case Design Techniques.

In both cases the intent is to find the maximum number of errors with the Minimum amount of effort and time.

## TESTING STRATEGY

The development process repeats this testing sub-process a number of times for the following phases.

a) Unit Testing.

b) Integration Testing

RESTful APIs: Express.js in the backend provides APIs.

HTTP Requests: React in the frontend makes requests to these APIs using Axios or Fetch.

JSON: Data exchanged between frontend and backend is typically in JSON format.

Database Integration: MongoDB with Mongoose for database operations.

State Management: Redux manages application state.

Middleware: Express.js middleware for handling requests.

CORS: CORS middleware in Express.js for cross-origin requests.

## TESTING METHOD

There are two kinds of testing mainly Blackbox and Whitebox testing:

- In black-box testing a software item is viewed as a blackbox, without knowledge of its internal structure or behavior. Possible input conditions based on the specifications (and possible sequences of input conditions), are presented as test cases.
- In white-box testing, knowledge of internal structure and logic is explored. Test cases are presented such that possible paths of control flow through the software item are traced.

Hence more defects than black-box testing are likely to be found. Unit test is white-box oriented. The unit testing implemented in every module of student attendance management system . by giving correct manual input to the system the data are stored in database and retrieved.

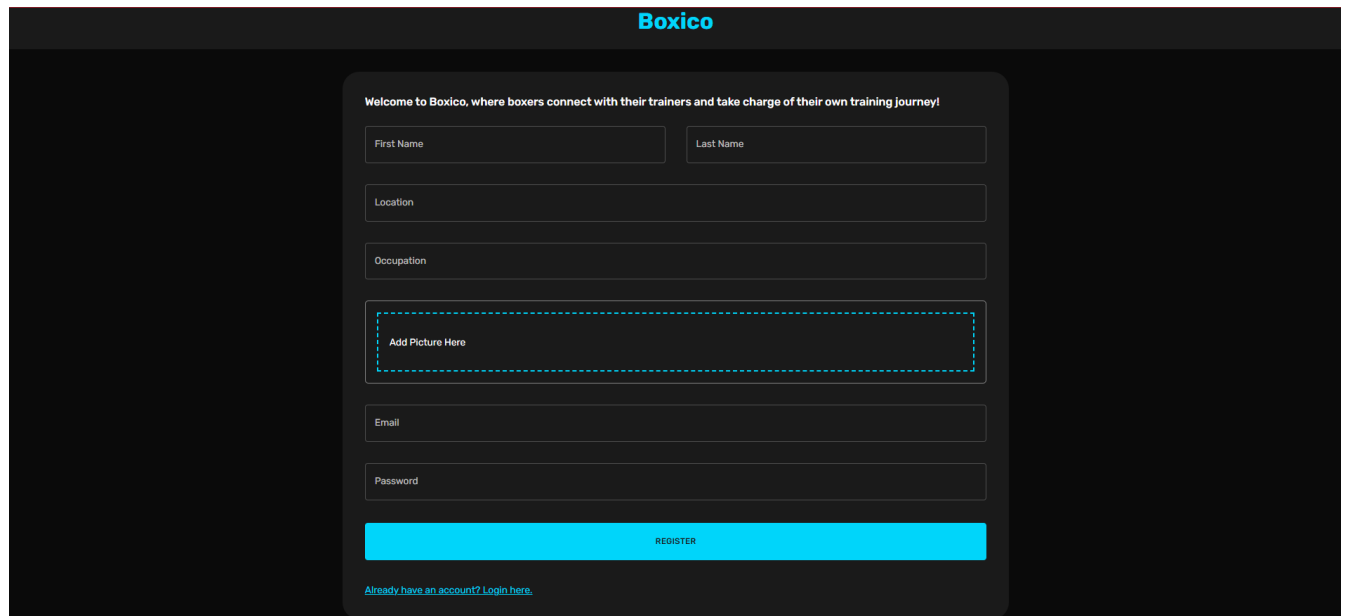


## 2. TEST CASES

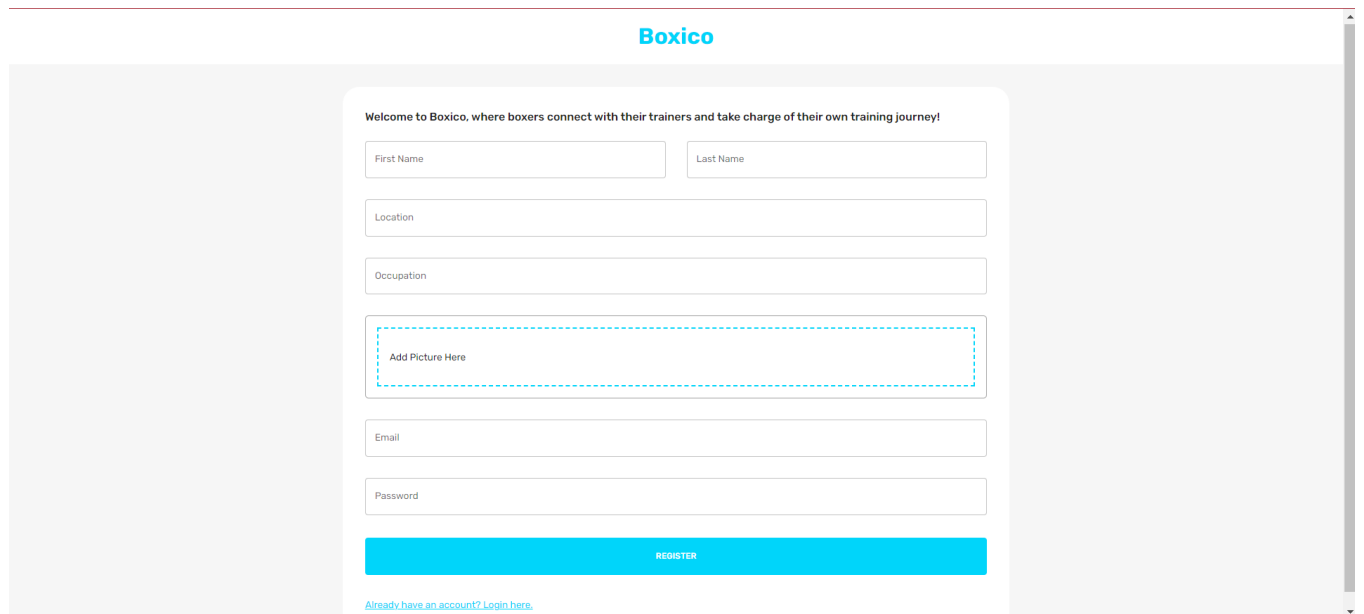
- User Registration:  
Input: Valid user details.  
Expected: Successful account creation.
- Login Functionality:  
Input: Correct credentials.  
Expected: Successful login.
- Posting a Training Session:  
Input: Training session details.  
Expected: Session posted and visible.
- Search Functionality:  
Input: Search query.  
Expected: Relevant results displayed.
- Follow/Unfollow Users:  
Input: Follow/unfollow action.  
Expected: Follower count updated.
- Like/Unlike Training Session:  
Input: Like/unlike action.  
Expected: Like count updated.
- Edit Profile Information:  
Input: Profile updates.  
Expected: Information updated.

### 3. USER MANUAL

Sign Up :



The image shows a dark-themed sign-up form for 'Boxico'. The form is centered on a dark background. At the top, the 'Boxico' logo is displayed in a light blue color. Below the logo, a welcome message reads: 'Welcome to Boxico, where boxers connect with their trainers and take charge of their own training journey!'. The form contains several input fields: 'First Name' and 'Last Name' (side-by-side), 'Location', 'Occupation', 'Email', and 'Password'. A dashed blue box labeled 'Add Picture Here' is positioned above the 'Email' field. A prominent blue 'REGISTER' button is located at the bottom of the form. Below the button, a link states: 'Already have an account? [Login here.](#)'.



The image shows a light-themed sign-up form for 'Boxico'. The form is centered on a light gray background. At the top, the 'Boxico' logo is displayed in a blue color. Below the logo, a welcome message reads: 'Welcome to Boxico, where boxers connect with their trainers and take charge of their own training journey!'. The form contains several input fields: 'First Name' and 'Last Name' (side-by-side), 'Location', 'Occupation', 'Email', and 'Password'. A dashed blue box labeled 'Add Picture Here' is positioned above the 'Email' field. A prominent blue 'REGISTER' button is located at the bottom of the form. Below the button, a link states: 'Already have an account? [Login here.](#)'.

Login:

---

Boxico

Welcome to Boxico, where boxers connect with their trainers and take charge of their own training journey!

LOGIN

[Don't have an account? Sign Up here.](#)

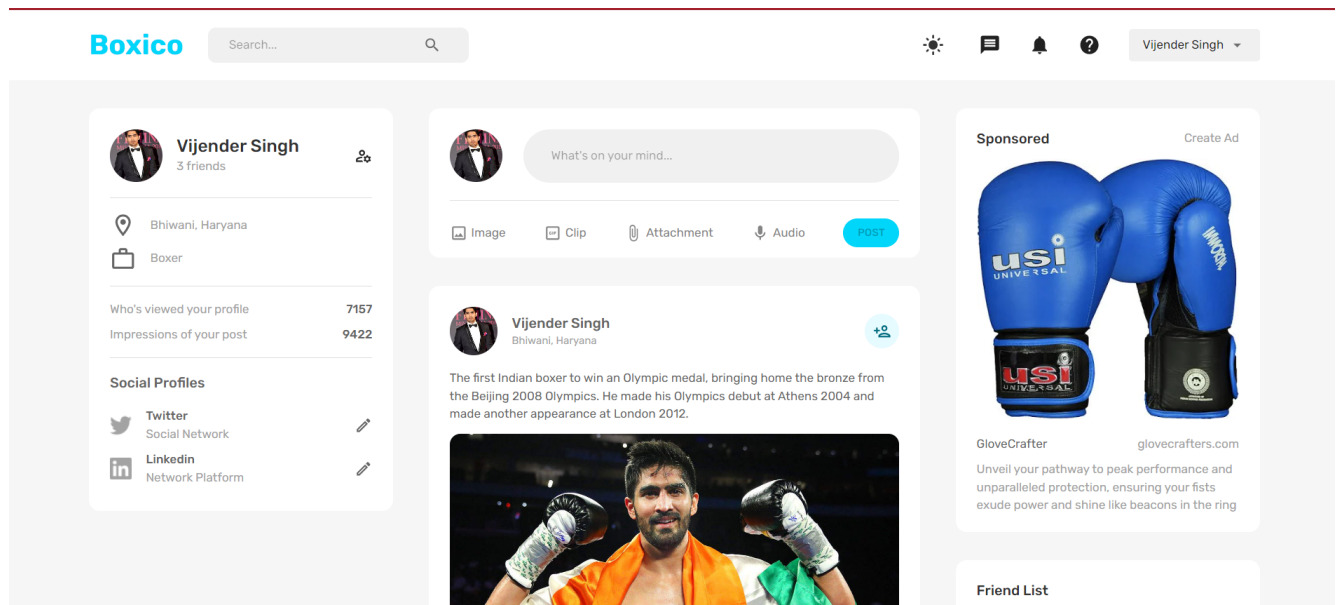
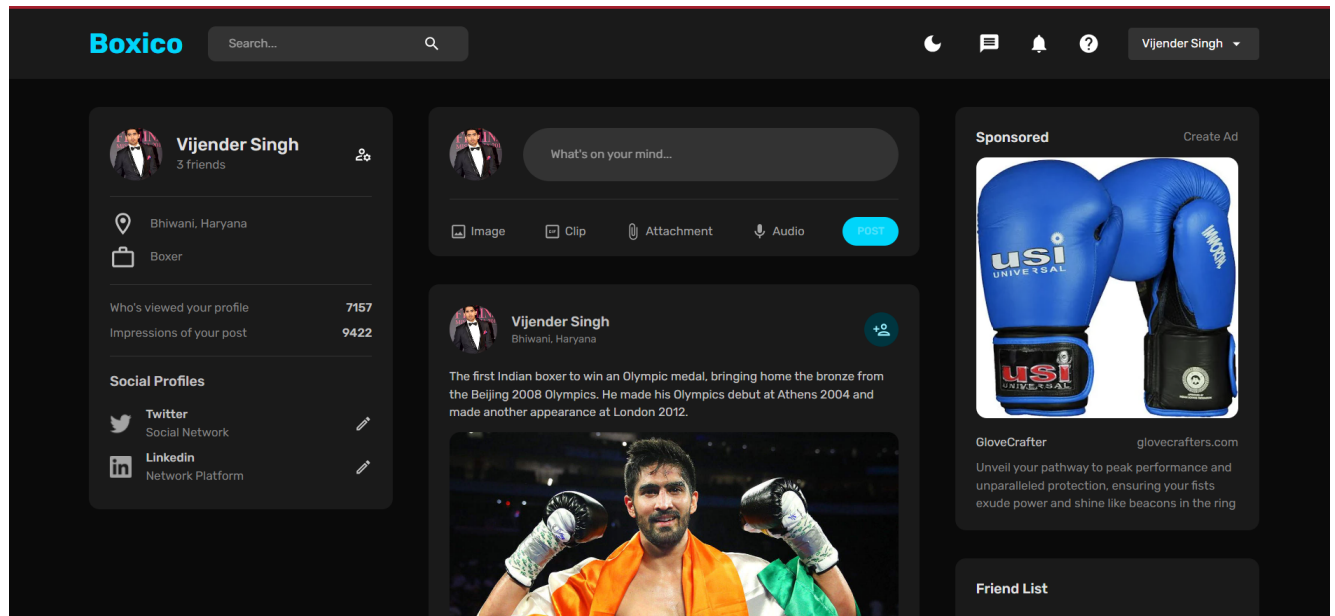
Boxico

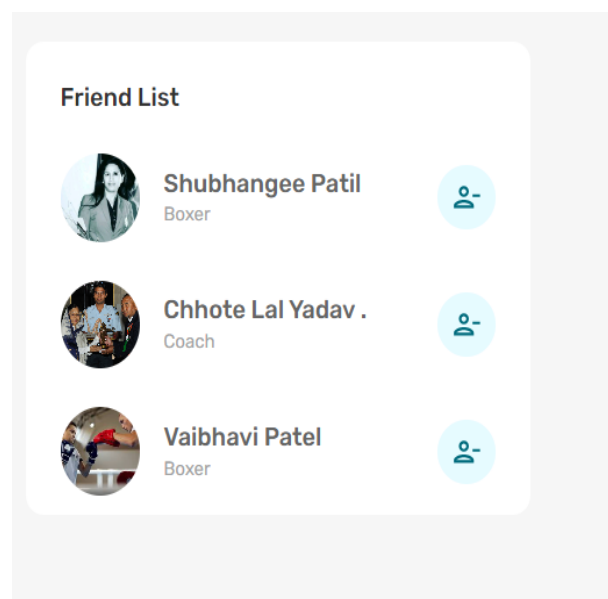
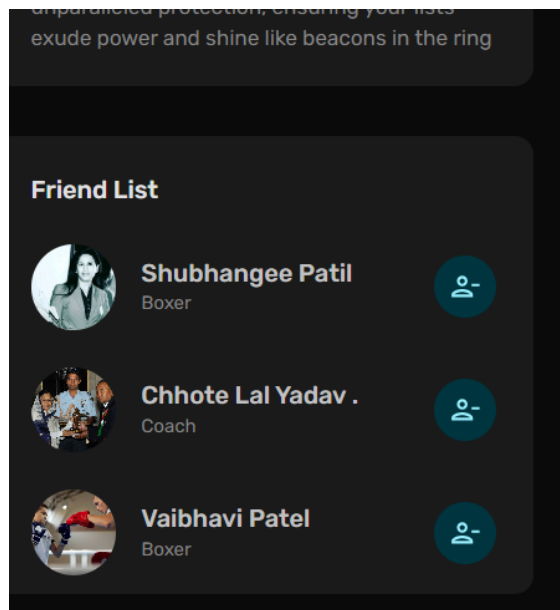
Welcome to Boxico, where boxers connect with their trainers and take charge of their own training journey!

LOGIN

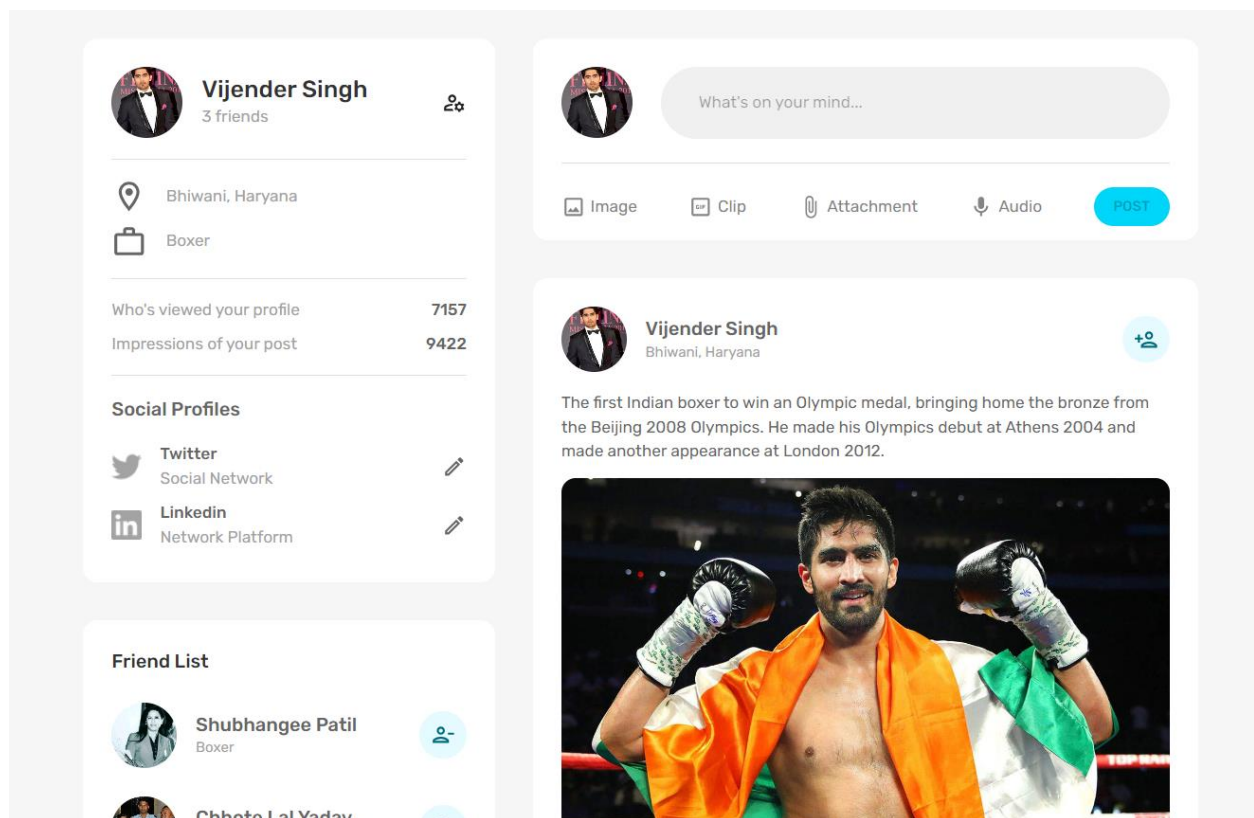
[Don't have an account? Sign Up here.](#)

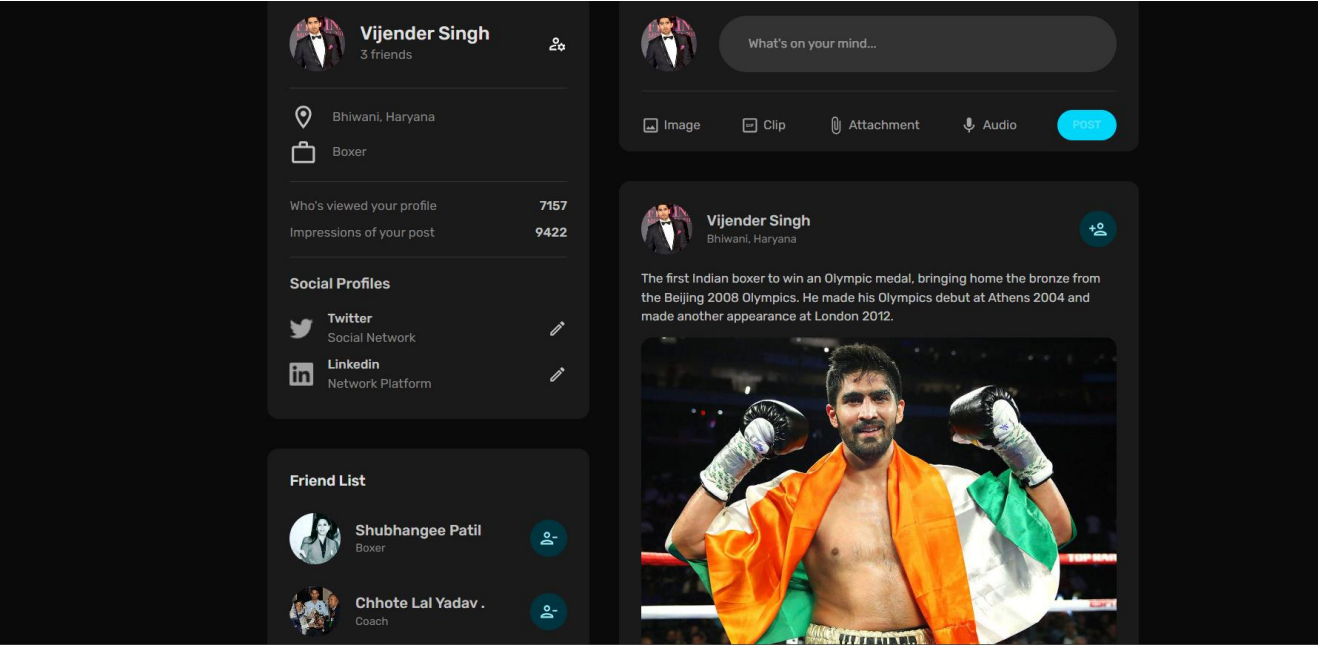
## Home Page:





## Profile:





## **4. LIMITATIONS AND FUTURE ENHANCEMENTS**

### **LIMITATIONS:**

#### Email Verification:

Without email verification, there may be a risk of unauthorized access or the creation of fake accounts. Users may not receive important notifications or updates related to their accounts. It could be challenging to ensure the authenticity of user accounts, potentially leading to issues with trust and security within the platform.

#### Update Profile:

Without the ability for users to update their profiles, they may experience frustration if their information changes over time (e.g., change of location, occupation, profile picture). Lack of profile update functionality limits user control over their personal information and may result in outdated or inaccurate user profiles.

#### Chat:

The absence of chat functionality restricts real-time communication between users, potentially hindering collaboration, networking, and engagement within the platform. Users may resort to external communication channels (e.g., email, social media) for direct interaction, leading to fragmentation and reduced user engagement within the platform itself.

### **FUTURE ENHANCEMENTS:**

#### Email Verification:

Automated Verification: Implement automated email verification upon registration.  
Resend Option: Allow users to request a verification email resend.  
Custom Templates: Provide customizable verification email templates.

#### Update Profile:

User-Friendly Editing: Develop an intuitive interface for profile editing.  
Validation Checks: Implement validation and error handling for profile updates.  
Version History: Consider adding a version history feature for profiles.

#### Chat:

Real-Time Messaging: Integrate real-time chat functionality for users.  
Notification System: Implement notifications for new chat messages.  
Multimedia Sharing: Allow users to share multimedia content within chats.  
Privacy Controls: Provide options for chat privacy settings.  
Search and Archive: Include search and archiving features for chat conversations.

## 5. CONCLUSION AND DISCUSSION

### CONCLUSION

"Boxico" is a social media platform designed for boxers and coaches. Initial functionalities are in place, but future enhancements will focus on providing a more engaging user experience. These enhancements include implementing features like automated email verification, user-friendly profile editing, and real-time chat functionality. Overall, Boxico aims to facilitate communication and collaboration within the boxing community.

### DISCUSSION

#### Self-Analysis of Project Viabilities:

In self-analysis, the project is recognized for its strong UI design, which enhances user experience. However, there is a recognized need for additional features to further enrich the platform's functionality and meet user expectations. Moving forward, focusing on implementing these features will contribute to enhancing the overall value and usability of the project.

#### Resources

- <https://app.diagrams.net/#G1OAMviTSyOWCj51II0QLIK9617MW-h45K>
- <https://online.visual-paradigm.com/w/ltaxjmby/diagrams/#diagram:workspace=ltaxjmby&proj=0&id=5&type=ClassDiagram>
- <https://github.com/>
- [https://in.linkedin.com/?src=go-pa&trk=sem-ga\\_campid.14650114788\\_asid.150089839322\\_crid.656569072777\\_kw.linkedin%20com\\_d.c\\_tid.kw\\_d-2246447702\\_n.g\\_mt.e\\_geo.1007753&mcid=6844056167778418689&cid=&gad\\_source=1&gclid=CjwKCAjwte-vBhBFEiwAQsv\\_xYJFZqHBamPCFyNsYU55sRvJMtlBjfVtKZpWRwxNWH1TxHYv4uxmNxoCzOEQA\\_vD\\_BwE&gclsrc=aw.ds](https://in.linkedin.com/?src=go-pa&trk=sem-ga_campid.14650114788_asid.150089839322_crid.656569072777_kw.linkedin%20com_d.c_tid.kw_d-2246447702_n.g_mt.e_geo.1007753&mcid=6844056167778418689&cid=&gad_source=1&gclid=CjwKCAjwte-vBhBFEiwAQsv_xYJFZqHBamPCFyNsYU55sRvJMtlBjfVtKZpWRwxNWH1TxHYv4uxmNxoCzOEQA_vD_BwE&gclsrc=aw.ds)