International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Social Media Application Using Reactjs

A NATHISH KUMAR
Mrs. A. Nandhini Professor - MCA

NEHRU COLLEGE OF MANAGEMENT

Thirumalayampalayam Coimbatore-105.

CHAPTER 1 INTRODUCTION

This chapter gives an overview about the aim, objectives, background and operation environment of the system

1.1 OVERVIEW OF THE PROJECT

Snapgram is a full-stack social media web application designed to provide users with an interactive platform to share posts, engage with content, and connect with others. Inspired by popular social media platforms like Instagram, Snapgram allows users to create accounts, upload images, like and comment on posts, and follow or unfollow other users. The project is built using React.js, Vite, Tailwind CSS, Node.js, Express.js, MongoDB, and Cloudinary, ensuring a smooth and modern user experience.

The frontend of Snapgram is developed using React.js with Vite, offering a fast and optimized user interface. Styling is handled by Tailwind CSS, ensuring a responsive and visually appealing design. The backend is powered by Node.js and Express.js, managing all API requests and user interactions efficiently. Data is stored in MongoDB, a NoSQL database, which handles user profiles, posts, and social interactions. Additionally, Cloudinary is integrated to manage and optimize image uploads, ensuring quick loading times and efficient media handling.

Snapgram provides essential social media features such as user authentication, post creation, profile management, likes and comments, and a follow/unfollow system. Users can create an account, log in securely using JWT-based authentication, and interact with content in real time. The application follows RESTful API principles, making it scalable and maintainable.

The project is designed to serve as a learning resource for full-stack developers interested in building modern MERN stack applications. By working with Snapgram, developers can gain hands-on experience with user authentication, API development, database management, and front-end optimization. The project also demonstrates best practices for performance optimization, utilizing Vite for fast builds and Cloudinary for media storage.

Snapgram is a great foundation for anyone looking to build and extend a social media application, making it an ideal project for those exploring full-stack web development.

CHAPTER 2 SYSTEM ANALYSIS

In this chapter, we will discuss and analyze the development process of the Snapgram social media application, including software requirement specifications (SRS) and a comparison between the existing and proposed systems. The functional and non-functional requirements are included in the SRS to provide a comprehensive description of system requirements before development. Additionally, the existing vs. proposed system analysis highlights how the new system improves efficiency, usability, and security compared to traditional social media platforms.

2.1 FEASIBILITY STUDY

1. Technical Feasibility:

The technical feasibility of implementing Snapgram using React.js and Node.js is high, ensuring efficient development and scalability. The following aspects contribute to the technical feasibility:

- **Development Tools** – Utilization of modern development frameworks and libraries for optimized performance.
- **Scalability** – The system is designed to handle a large user base efficiently.
- **Database Integration** – Uses MongoDB for fast and flexible data storage.
- **Security** – Implements JWT authentication and encryption for secure user data.
- **Web Service Integration** – RESTful APIs facilitate seamless backend communication.
- **User Authentication and Authorization** – Includes OAuth and multi-factor authentication (MFA).
- **Customization** – Users can personalize profiles, settings, and notifications.
- **Third-Party Libraries and Frameworks** – Uses libraries like Redux for state management.
- **Testing and Debugging** – Implemented using Jest and Cypress for efficient testing.
- **Documentation and Community Support** – Leverages a strong developer community.
- **Web Hosting and Deployment** – Uses Vercel and AWS for reliable deployment.
- **Mobile Optimization** – Ensures a responsive interface compatible with mobile devices.

2. Operational Feasibility:

Operational feasibility evaluates whether Snapgram can be successfully integrated into users' daily routines. Below are key factors considered:

- **User Experience & Interface** – A seamless, intuitive UI designed for engagement.
- **User Training & Onboarding** – Provides guides and tutorials for new users.
- **Data Migration** – Facilitates secure transfer from existing social media platforms.
- **Integration with External Services** – Allows users to share content on other platforms.
- **Workflow Integration** – Designed for real-time interactions with minimal delays.
- **Security & Compliance** – Ensures GDPR and CCPA compliance for data privacy.

- **Performance Optimization** – Uses caching and load balancing for efficiency.
- **User Feedback Mechanism** – Enables continuous improvements through feedback.
- **Accessibility** – Designed for users with disabilities using WCAG guidelines.

3. Economic Feasibility:

Economic feasibility determines the cost-effectiveness of Snapgram. The following aspects are considered:

- **Cost Estimation:**
  - Development Costs
  - Infrastructure and Hosting Costs
  - API and Licensing Fees
  - Maintenance and Support Costs
  - Marketing and User Acquisition
- **Benefits Assessment:**
  - Improved User Engagement
  - Monetization Opportunities (Ad Revenue, Premium Features)
  - Enhanced Content Sharing & Interaction
  - Cost Reduction in Infrastructure Maintenance
- **Return on Investment (ROI) & Revenue Model:**
  - Subscription Plans for Premium Users
  - Ad Placement & Sponsored Content
  - Creator Monetization Features
  - Digital Marketplace for User Transactions

2.2 EXISTING SYSTEM

Most existing social media platforms provide basic functionalities such as:

- User profiles with bio, photos, and friend lists.
- A news feed that displays posts based on an engagement algorithm.
- Messaging and chat functionalities with basic encryption.
- Content sharing (images, videos, and links) with likes and comments.
- Limited privacy settings to control visibility.
- Basic moderation tools for handling inappropriate content.
- Minimal monetization options for content creators.

While these platforms serve a large user base, they have drawbacks such as:

- Privacy concerns and data misuse.

- Lack of personalization in content recommendations.
- Limited revenue opportunities for users.
- Inadequate moderation leading to harmful content spread.
- Slow adoption of new technologies like AI and blockchain for security.

2.3 PROPOSED SYSTEM

The **Snapgram** platform enhances traditional social media experiences by introducing:

- **Personalized Content Recommendations:** Uses **AI-driven algorithms** to curate a user's news feed based on real interests.
- **Advanced Privacy Controls:** Allows users to **customize audience visibility** for different types of content.
- **Improved Community Features:** Offers **interactive groups** with better moderation and engagement tools.
- **AI-Based Content Moderation:** Detects and removes **hate speech, fake news, and harmful content** automatically.
- **Content Monetization:** Users can earn through **ad revenue sharing, in-app purchases, and creator subscriptions**.
- **Enhanced Security:** Implements **end-to-end encryption for messages** and ensures compliance with **privacy laws**.
- **Real-Time Interactions:** Introduces **live streaming, voice/video calls, and interactive polls**.
- **User Well-Being Features:** Encourages **healthy screen time habits** with reminders and positive interaction nudges.

CHAPTER 3 SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

- Processor: 12th Gen Intel(R) Core(TM) i5 1240P 1.70 GHz
- Hard Disk: 512GB SSD
- RAM: 16GB
- Operating System: Windows 11

3.2 SOFTWARE REQUIREMENTS

- Front End: HTML, CSS, React.js
- Back End: Node.js
- Database: MongoDB

3.3 SOFTWARE TOOLS USED

- IDE/Workbench: Visual Studio Code
- Version Control: Git & GitHub
- Package Manager: npm / yarn
- API Testing Tool: Postman
- Deployment Tools: Vercel (Frontend), AWS (Backend), MongoDB Atlas

CHAPTER 4 SYSTEM DESIGN

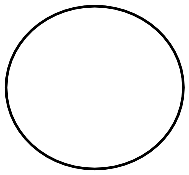
4.1 DESIGN TERMINOLOGY:

DFD Symbols

In a DFD there are four symbols

A Square defines a source or destination of system data

An arrow identifies data flow in motion. It is a pipeline through which in format flows

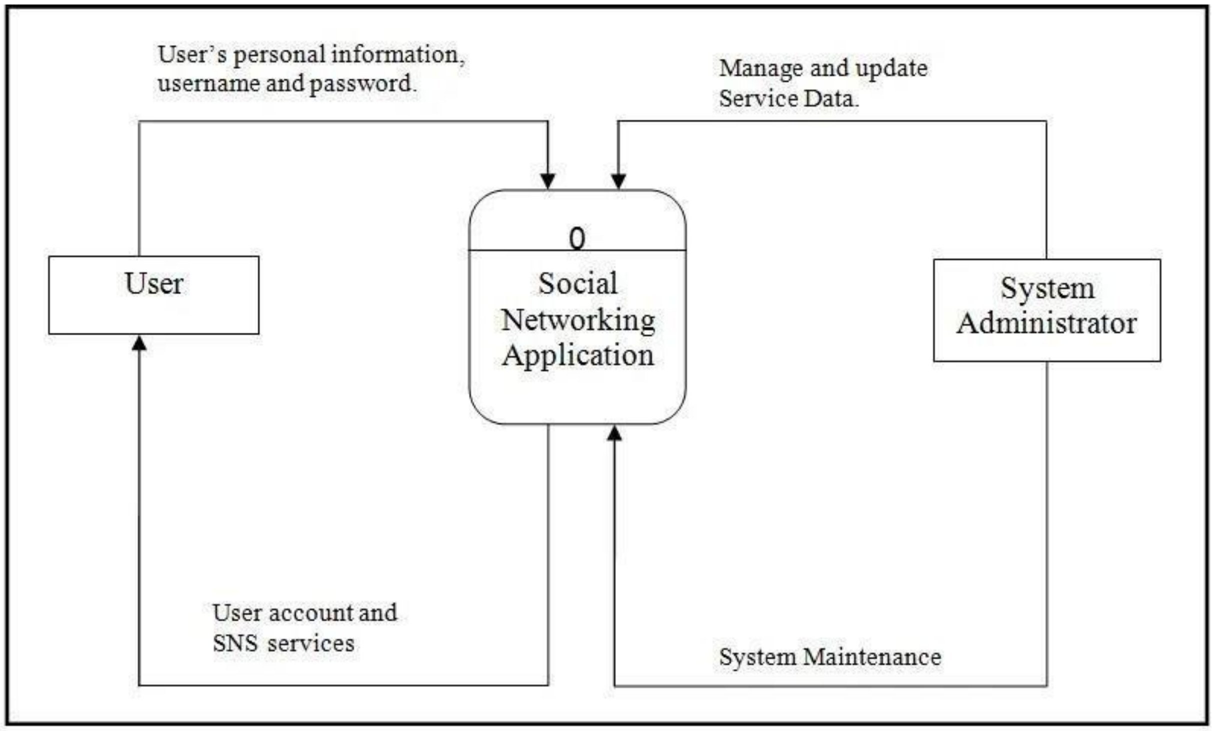


A circle or a bubble represents a process that transform incoming data into going data flows

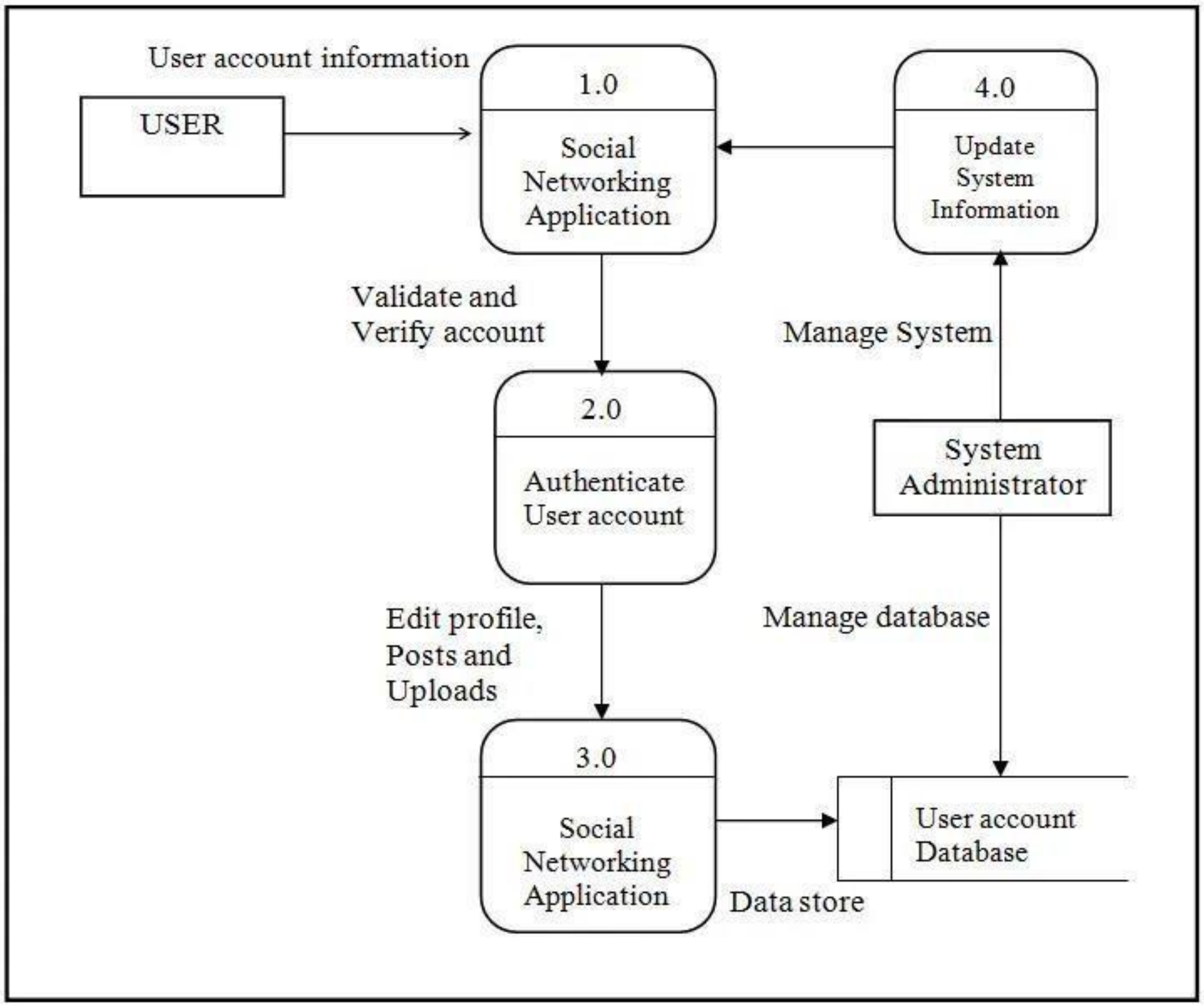
An open rectangle is a data source or data at rest or a temporary of data constructing a DFD

4.2 DATA FLOW DIAGRAM

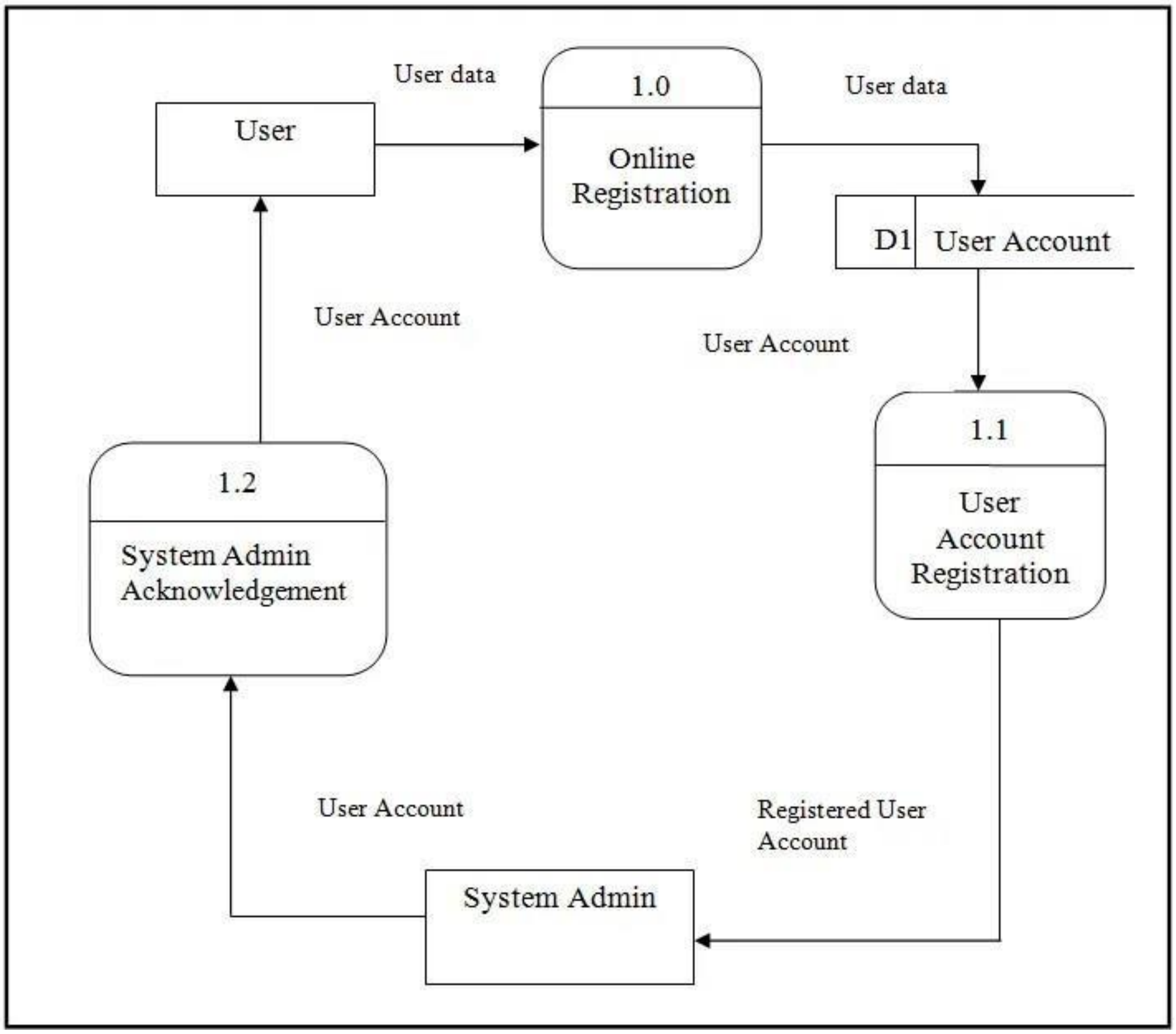
LEVEL 0 :


User's personal information, username and password.
Manage and update Service Data.
User
0
Social Networking Application
System Administrator
User account and SNS services
System Maintenance

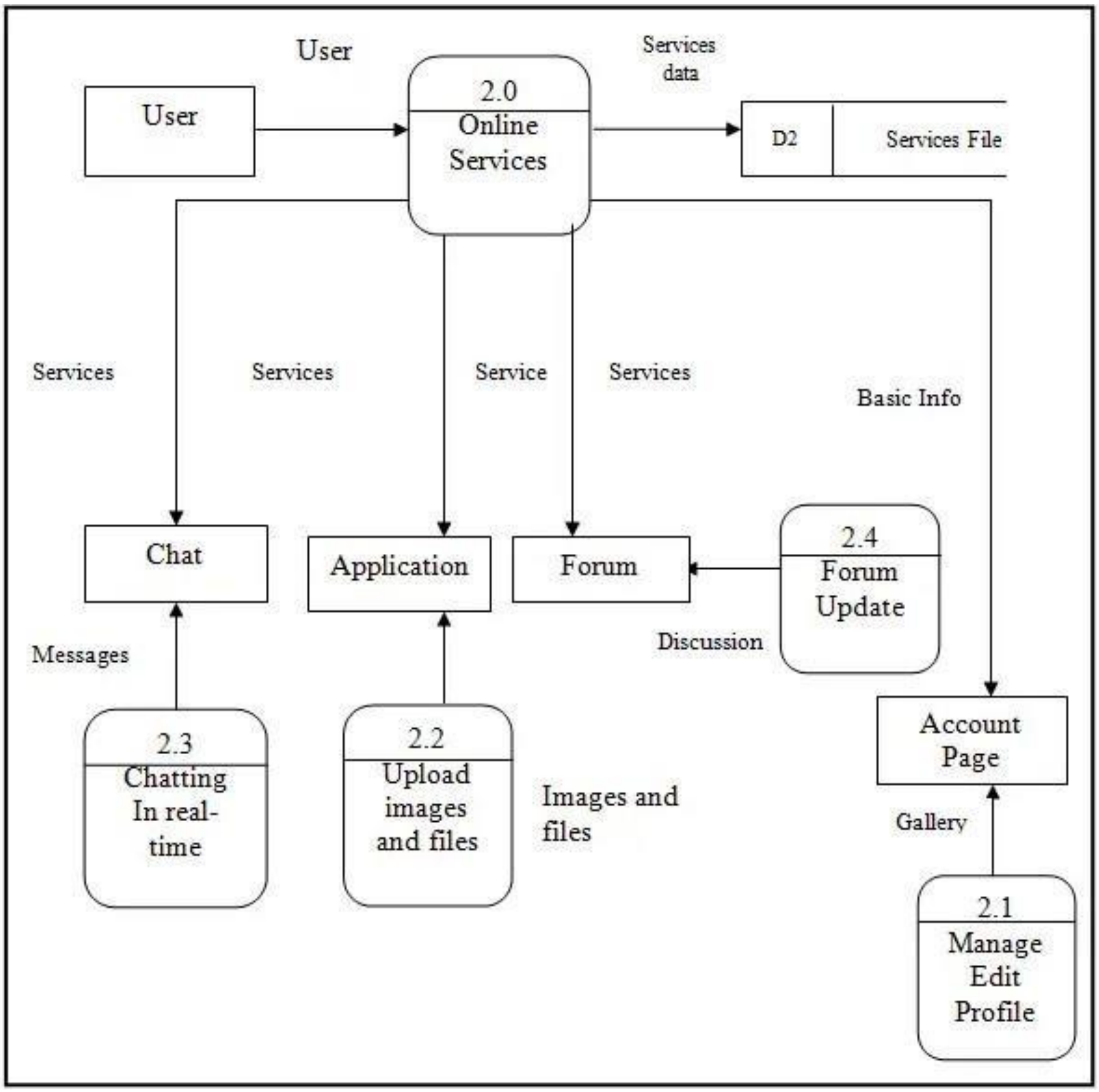
LEVEL 1 :


User account information
USER
1.0
Social Networking Application
4.0
Update System Information
Validate and Verify account
Manage System
2.0
Authenticate User account
System Administrator
Edit profile, Posts and Uploads
Manage database
3.0
Social Networking Application
User account Database
Data store

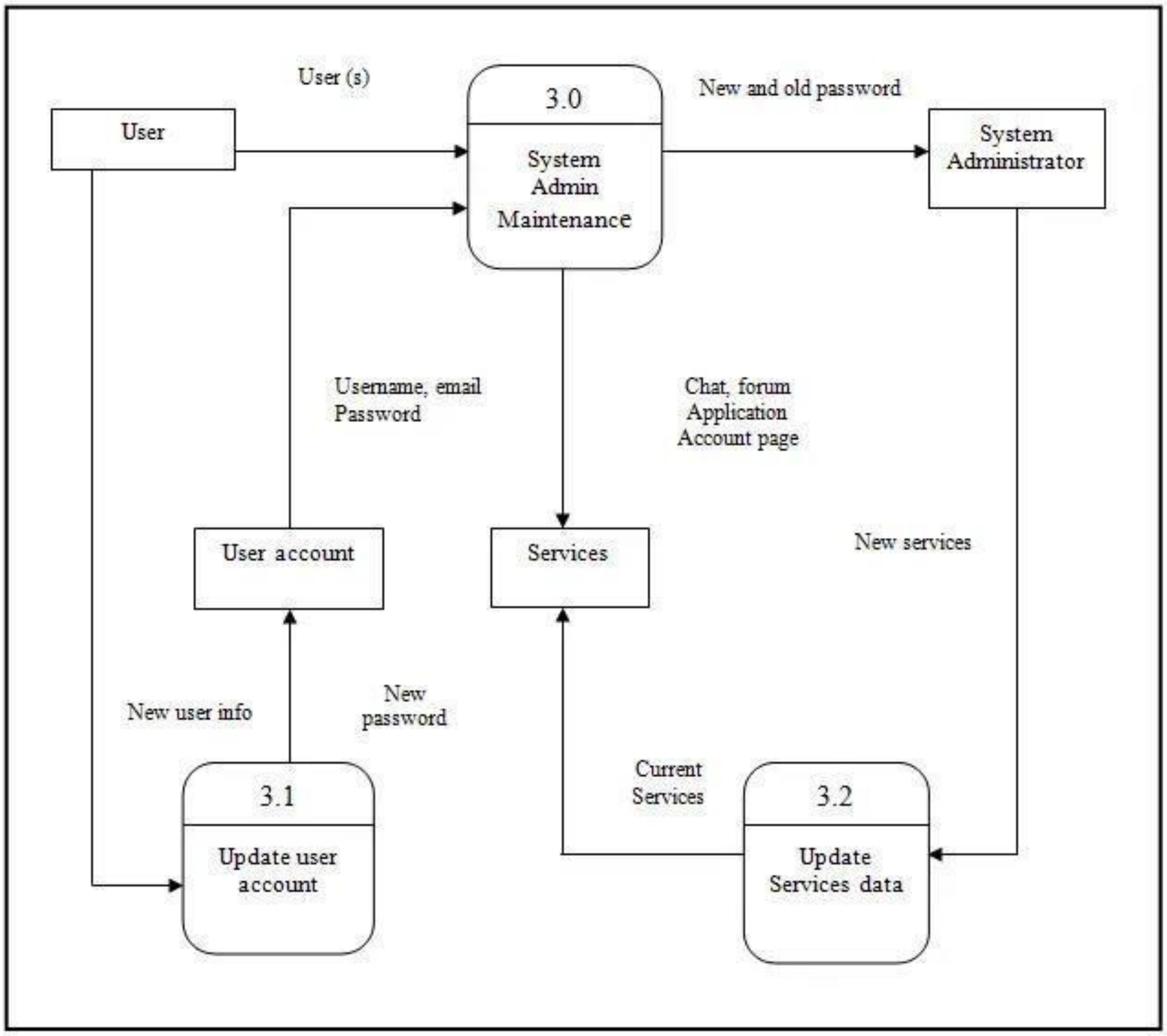
LEVEL 2 :


User data
User
1.0
Online Registration
User data
D1 User Account
User Account
User Account
1.1
User Account Registration
1.2
System Admin Acknowledgement
User Account
Registered User Account
System Admin

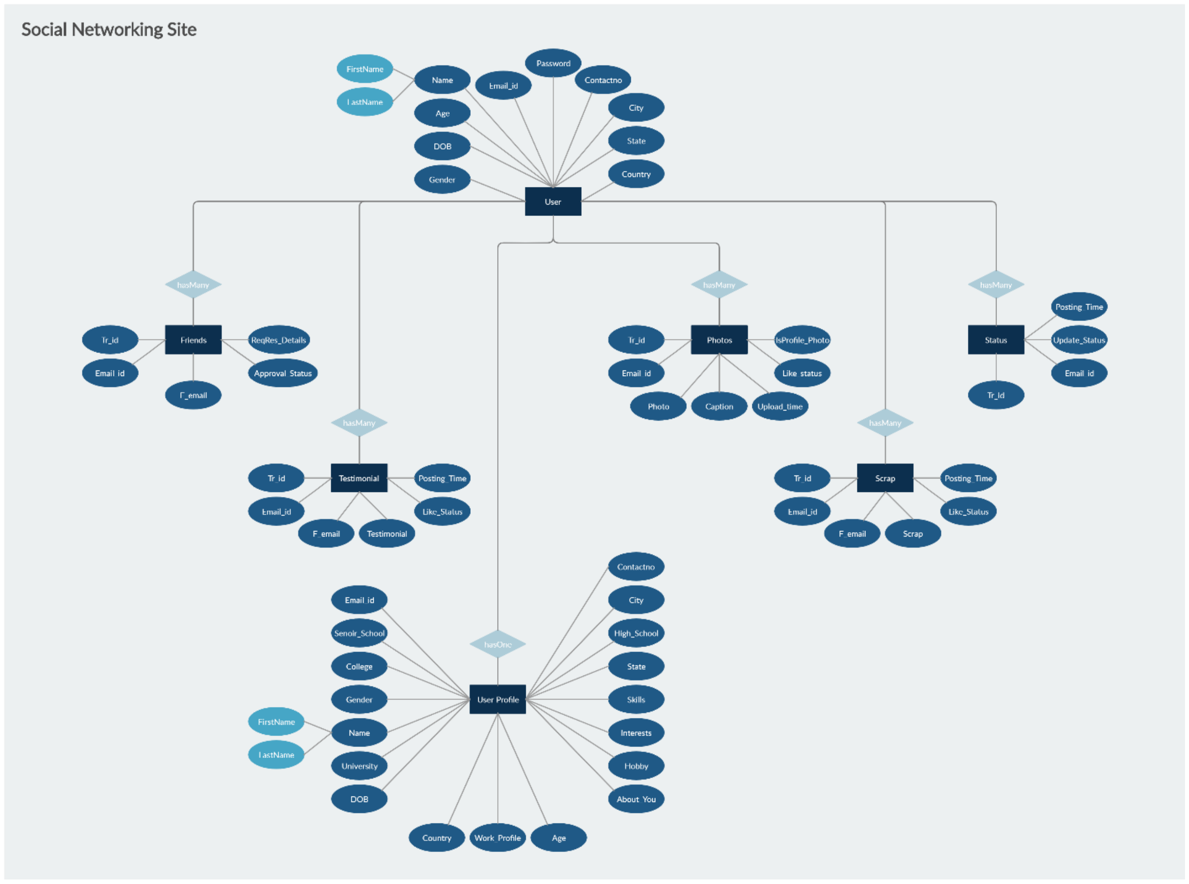
LEVEL 3 :


User
User
2.0
Online Services
Services data
D2
Services File
Services
Services
Service
Services
Basic Info
Chat
Application
Forum
2.4
Forum Update
Discussion
Messages
Account Page
2.3
Chatting In real-time
2.2
Upload images and files
Images and files
Gallery
2.1
Manage Edit Profile

LEVEL 4 :


User (s)
3.0
System Admin Maintenance
New and old password
User
System Administrator
Username, email Password
Chat, forum Application Account page
User account
Services
New services
New user info
New password
Current Services
3.1
Update user account
3.2
Update Services data

ER DIAGRAM


Social Networking Site
FirstName
LastName
Name
Email_id
Password
Contactno
Age
City
DOB
State
Gender
Country
User
hasMany
Friends
Fr_id
Email id
ReqRes_Details
Approval Status
F_email
hasMany
Photos
Fr_id
Email id
IsProfile_Photo
Like status
Photo
Caption
Upload_time
hasMany
Status
Posting Time
Update_Status
Email id
Fr_Id
hasMany
Testimonial
Fr id
Email_id
F email
Testimonial
Posting_Time
Like_Status
hasMany
Scrap
Fr id
Email_id
F email
Scrap
Posting_Time
Like_Status
hasOne
User Profile
Email id
Senoir_School
College
Gender
FirstName
LastName
Name
University
DOB
Country
Work Profile
Age
Contactno
City
High_School
State
Skills
Interests
Hobby
About You

4.3 DATABASE DESIGN

Table name : User Table

Column Name	Data Type	Description
User ID	INT	Unique identifier for each user
Username	VARCHAR(50)	Unique username for login
Password	VARCHAR(255)	Securely hashed password
Email	VARCHAR(100)	Email address of the user
Full Name	VARCHAR(100)	Full Name of the user
Bio	TEXT	User's bio or description
Profile Picture	VARCHAR(255)	URL or file path of the user's profile picture
Cover Photo	VARCHAR(255)	URL or file path of the user's cover photo
Date Of Birth	DATE	User's date of birth
Location	VARCHAR(100)	User's current location
Website	VARCHAR(255)	User's personal website
Joined Date	DATETIME	Date and Time when the user joined the platform
Last Login	DATETIME	Date and Time of the user's last login
Is Active	BOOLEAN	Flag indicating if the user's account is currently active
Is Verified	BOOLEAN	Flag indicating if the user's account is verified

CHAPTER 5 SYSTEM DEVELOPMENT

System Modules Specification

Snapgram consists of multiple functional modules, each responsible for specific tasks. These modules ensure seamless interaction, content sharing, and security while enhancing the overall user experience.

1. **User Management Module**
   - **Registration:** Users can create accounts with unique usernames, emails, and passwords.
   - **Login/Logout:** Secure authentication system using JWT tokens.
   - **Profile Management:** Users can edit their profiles, update personal details, and upload profile pictures.
   - **Privacy Settings:** Users can control who can see their content and manage blocked accounts.
2. **Content Sharing Module**
   - **Post Creation:** Users can share posts, including text, images, and videos.
   - **Post Editing/Deleting:** Users can modify or remove their posts.
   - **Comments & Replies:** Allows users to engage with posts through comments and threaded replies.
   - **Like & Reactions:** Users can express their opinions through likes and emoji-based reactions.
3. **Social Interaction Module**
   - **Friend/Follow System:** Users can follow or send friend requests to connect with others.
   - **Messaging System:** Real-time private messaging between users.
   - **Notifications:** Users receive alerts for likes, comments, messages, and friend requests.
4. **Search and Discovery Module**
   - **User Search:** Users can search for profiles based on name, username, or email.
   - **Content Discovery:** Personalized recommendations based on interests, trending topics, and user interactions.

5. **Security and Privacy Module**
   - **Access Control:** Implements role-based permissions and authentication mechanisms.
   - **Data Protection:** Secure storage with encrypted passwords and HTTPS communication.
   - **Threat Protection:** Prevents SQL injection, Cross-Site Scripting (XSS), and other vulnerabilities.
6. **Analytics and Reporting Module**
   - **Usage Statistics:** Tracks metrics like user engagement, number of posts, and interactions.
   - **User Reports:** Users can report inappropriate content or abuse for review.
   - **Admin Dashboard:** Displays insights on platform usage and trends.
7. **Administration Module**
   - **User Management:** Admins can manage user roles, permissions, and account status.
   - **Content Moderation:** Tools for reviewing and removing flagged posts, comments, and accounts violating community guidelines.

CHAPTER 6 SYSTEM TESTING

6.1 TESTING METHODOLOGY

System testing was done by giving different training and testing datasets. This test was done to evaluate whether the system was predicting accurate result or not. During the phase of the development of the system our system was tested time and again.

The series of testing conducted are as follows:

UNIT TESTING

In unit testing, we designed the whole system in modularized pattern and each module was tested. Till we get the accurate output from the individual module we worked on the same module.

INTEGRATION TESTING

After constructing individual modules all the modules were merged and a complete system was made. Then the system was tested whether the prediction given by training dataset to testing set was correct or not. We tried to meet the accuracy as higher as much as we can get. After spending a couple of days in integration testing the average accuracy of our system was 91%.

ALPHA TESTING

Alpha testing is the first stage of software engineering which is considered as a simulated or actual operational testing done by the individual member of the project. Alpha testing is conducted by the project developers, in context of our project.

BETA TESTING

Beta testing comes continuously after alpha testing which is considered as a form of external user acceptance testing. The beta version of the program is developed to and provided to limited audience. This is the final test process in the case of this project. In this system the beta testing is done by our colleagues and the project supervisor.

6.2 TESTING IN DEVELOPED SOFTWARE

All the project is developed successfully and tested completely on base of the requirements mentioned in the DFD and the module details. Unit testing is conducted while development regression testing is conducted while merging module. System testing is conducted while all project ready. Regression testing is done while improve the performance of the system.

CHAPTER 7 SOFTWARE MAINTENANCE

1.1 SYSTEM MAINTENACE

Maintenance means restoring something to its original conditions. Enhancement means adding, modifying the code to support the changes in the user specification. System maintenance conforms the system to its original requirements and enhancement adds to system capability by incorporating new requirements.

Thus, maintenance changes the existing system, enhancement adds features to the existing system, and development replaces the existing system. It is an important part of system development that includes the activities which corrects errors in system design and implementation, updates the documents, and tests the data.

Any maintenance activity comprises the following four key stages:

- Help Desk: A preliminary analysis of the change request submitted by the user through a formal change request will be done, and if the problem is sensible, it is accepted.

• Analysis: Managerial and technical analysis of the problems are undertaken to investigate the cost factors and other alternative solutions.

• Implementation: The maintenance team implements the chosen change/ solution as well as tested by them. All infected components are to be identified and brought in to the scope of the change. Unit test, integration test, user oriented functional acceptance tests and regression test strategies are provided.

• Release: The changes are released to the customer, with a release note and appropriate documentation giving details of the changes.

Maintenance Types

Once the system is fully implemented and starts operating, the maintenance phase begins. The following are the different types of maintenance activities:

- **Corrective Maintenance**: This type of maintenance is to rectify design, coding and implementation problems detected after the implementation of the System. This kind of problem generally surfaces immediately after the system is implemented. This type of problem needs immediate attention as it hampers the day to day work of the end user. Proper planning and interaction with the end user during system development process can minimize Corrective Maintenance. In spite of the all these kinds of maintenance, these constitute more than 60 percent of total maintenance effort. Corrective maintenance is very much undesirable. Care should be taken to see that normal business operations are not disturbed because of it. (180) System Implementation and Maintenance
- **Adaptive Maintenance**: Adaptive maintenance is required because business operates on a social environment and need of the organization changes as organization ventures in to new areas, or as government regulation policy changes, etc. Maintenance of the software to adapt to this kind of changes is called adaptive maintenance. Unlike corrective maintenance, this kind of activity adds value to the information system and affects a small part of the organization. This activity is not as urgent as corrective maintenance as these changes are gradual and allow sufficient time to the system group to make changes to the software.
- **Perfective Maintenance**: Adding new functionalities and features to the software to make it more versatile and user oriented is the characteristics of Perfective Maintenance. Some times, changes are made to improve performance of the software. In some sense, this maintenance can be thought of as a new development activity.

- **Preventive Maintenance**: The activity done to prevent system failure by bringing changes to software for the system safe future and easy maintenance comes under preventive maintenance. This reduces the need of corrective maintenance. Preventive maintenance could increase the volume of transactions that can be handled by the system. Preventive maintenance is done when the system is least used or not used at all. This does not add value to the system, but certainly lowers the cost of corrective maintenance.

CHAPTER 8

LIMITATIONS, FUTURE SCOPE & CONCLUSION

8.1 LIMITATIONS

Developing a social media application project entails confronting various limitations and challenges that can impact its design, functionality, and overall success. Technical limitations, such as scalability issues, performance bottlenecks, and security vulnerabilities, pose significant hurdles. Ensuring the application can handle a growing user base, increasing data volumes, and maintaining robust security measures requires careful planning and resource allocation. Additionally, legal and regulatory constraints, such as privacy regulations and content guidelines, impose compliance requirements that must be meticulously addressed to avoid legal repercussions.

Resource limitations, including budget constraints, time constraints, and availability of skilled personnel, can restrict the project's scope and progress. Balancing the project's objectives with the available resources demands strategic decision-making and prioritization. Moreover, user engagement and growth limitations, such as competition from established platforms, user acquisition challenges, and monetization strategies, can impact the project's viability and sustainability.

Ethical and societal considerations, such as addressing misinformation, fostering a healthy online environment, and safeguarding user well-being, present complex challenges that require careful navigation. Maintaining user trust and confidence while promoting responsible platform usage is essential for long-term success. Furthermore, infrastructure and maintenance limitations, such as managing server infrastructure, releasing software updates, and providing customer support, require ongoing investment and attention to ensure the application's reliability and performance.

Successfully navigating these limitations demands a comprehensive understanding of the project's objectives, stakeholders, and constraints. By proactively addressing these challenges and leveraging available resources effectively, social media application projects can overcome limitations and achieve their goals in a competitive and dynamic digital landscape.

8.2 FUTURE SCOPE

The future scope of social media applications is expansive, offering a plethora of opportunities for innovation and advancement. One significant avenue for development lies in the integration of artificial intelligence (AI) and machine learning (ML) technologies. By harnessing AI and ML algorithms, social media platforms can deliver more personalized experiences, improve content moderation, and enhance user engagement by providing tailored content recommendations and detecting and preventing abusive behavior.

Another promising area for future development is the integration of augmented reality (AR) and virtual reality (VR) technologies. By incorporating AR and VR capabilities, social media platforms can create immersive experiences, virtual environments, and interactive storytelling possibilities, revolutionizing the way users interact and engage with each other and with content.

Blockchain technology also holds considerable promise for the future of social media applications. By leveraging blockchain, platforms can explore decentralized social networks, tokenization of user engagement, and enhanced security and transparency for user data management, fostering trust and empowering users with greater control over their data.

Furthermore, the integration of voice and conversational interfaces presents an exciting opportunity to enhance user experiences and engagement. By implementing voice recognition and conversational interfaces, social media platforms can enable hands-free interactions, voice- controlled commands, and more natural and intuitive user engagement.

Additionally, future social media applications may focus on niche and specialized communities, catering to specific interests, industries, or demographics. By creating tailored experiences for unique community needs and preferences, platforms can foster deeper connections and engagement among users.

As social media continues to evolve, prioritizing data privacy and user control will be paramount. Platforms must implement robust privacy controls, data ownership models, and transparent data policies to protect user privacy and build trust.

In summary, the future of social media applications is rich with possibilities, from AI- driven personalization to immersive AR/VR experiences, blockchain-enabled decentralization, and beyond. By embracing these future scopes and staying attuned to emerging technologies and user preferences, social media platforms can continue to innovate and thrive in an ever-changing digital landscape.

8.3 CONCLUSION

In conclusion, developing a social media application project presents a myriad of challenges and limitations across technical, legal, resource, user engagement, ethical, and infrastructure aspects. These challenges encompass scalability issues, performance bottlenecks, and security vulnerabilities on the technical front, while legal and regulatory constraints demand compliance with privacy regulations and content guidelines. Resource limitations, including budget constraints and availability of skilled personnel, can impact the project's scope and progress. Moreover, user engagement and growth limitations pose challenges in terms of competition, user acquisition, and monetization strategies.

Ethical considerations, such as addressing misinformation and fostering a healthy online environment, are crucial for maintaining user trust and confidence. Infrastructure and maintenance limitations, including managing server infrastructure and providing customer support, require ongoing investment and attention to ensure reliability and performance.

Despite these challenges, navigating the complexities of social media application development requires strategic planning, proactive mitigation strategies, and a commitment to innovation and user-centric design. By addressing these limitations with diligence and creativity, social media application projects can overcome obstacles and capitalize on opportunities for growth and success in an ever-evolving digital landscape.

CHAPTER 9 BIBILIOGRAPHY

9.1 REFERENCES

When working on a project for a social media network, it's essential to draw inspiration from existing platforms while ensuring that your project has its unique features and innovations. Here are some references you can explore:

- https://www.facebook.com
- https://twitter.com
- https://www.instagram.com
- https://github.com/adrianhajdin (Reference to project source)
- https://developer.mozilla.org (For web development documentation)
- https://www.w3schools.com (For frontend and backend technology references)
- https://www.npmjs.com (For managing dependencies and libraries)

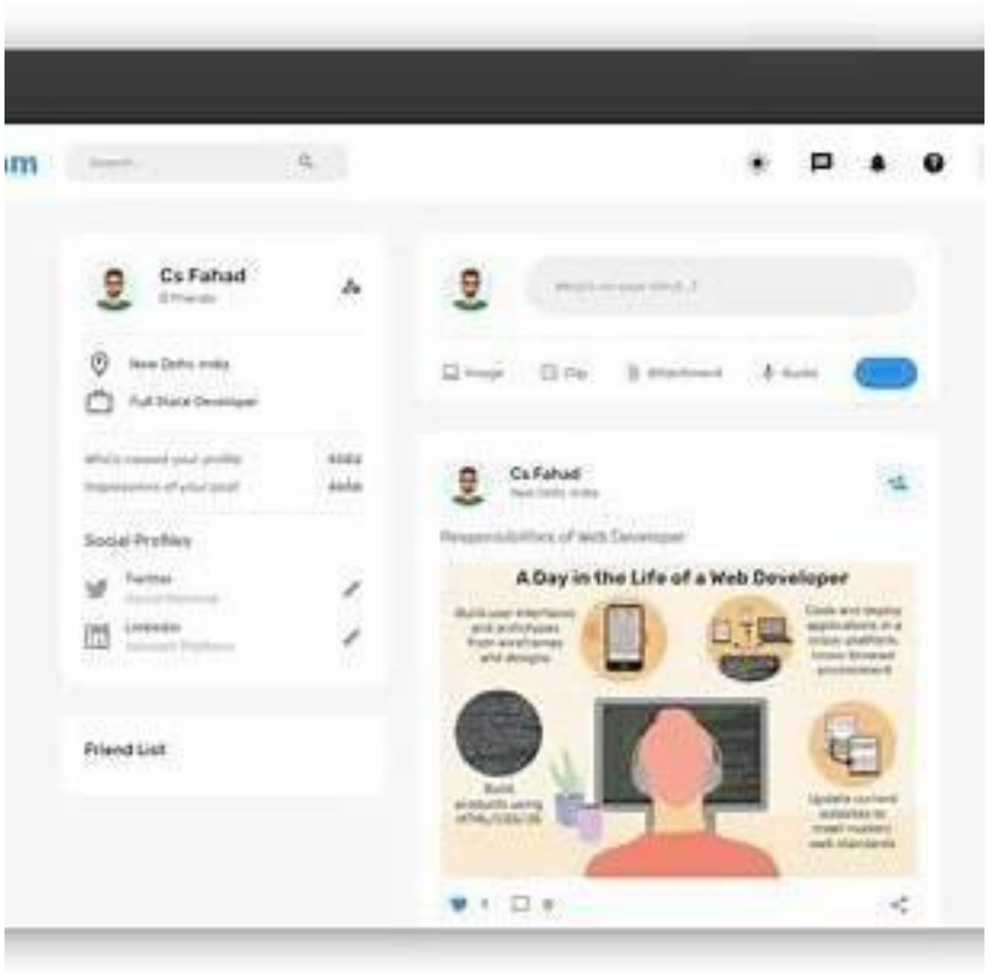
For more reference :

- https://www.google.co.in
- https://www.youtube.com
- https://chat.openai.com

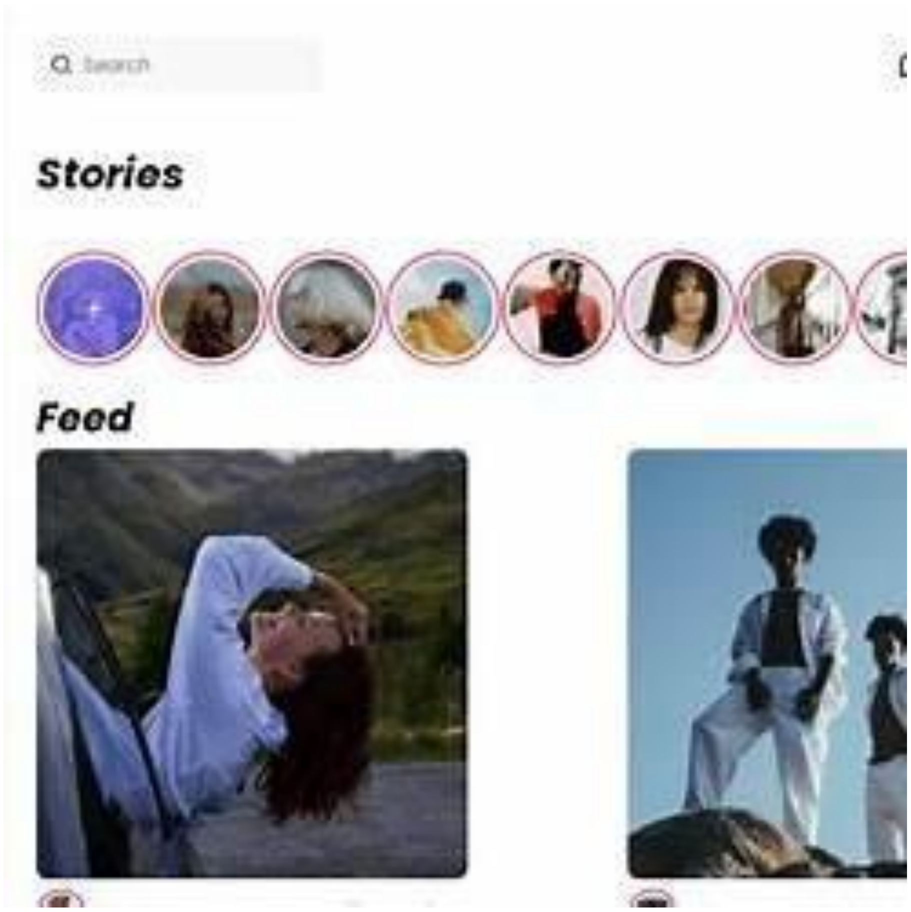
CHAPTER 10

APPENDICES

10.1 SCREENSHOTS



Screenshot 1



Screenshot 2

Screenshot 3

10.2 SAMPLE SOURCE CODE

Server .js

```
import express from "express"; import mongoose from "mongoose"; import cors from "cors";
import dotenv from "dotenv";

dotenv.config();
const app = express();

app.use(express.json()); app.use(cors());

mongoose
  .connect(process.env.MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log("MongoDB Connected"))
  .catch((err) => console.log(err));

  const postSchema = new mongoose.Schema({ user: String,
  content: String, image: String,
  createdAt: { type: Date, default: Date.now },
});

const Post = mongoose.model("Post", postSchema); app.get("/posts", async (req, res) => {
  const posts = await Post.find(); res.json(posts);
});

  app.post("/posts", async (req, res) => { const newPost = new Post(req.body); await newPost.save();
   res.json(newPost);
});

app.listen(5000, () => console.log("Server running on port 5000"));
```

App.js

```
import React, { useEffect, useState } from "react"; import axios from "axios";

const App = () => {
  const [posts, setPosts] = useState([]); const [content, setContent] = useState("");

  useEffect(() => {
    axios.get("http://localhost:5000/posts").then((res) => setPosts(res.data));
  }, []);

    const addPost = () => { axios
      .post("http://localhost:5000/posts", { user: "Sona", content })
      .then((res) => setPosts([...posts, res.data]));
  };

  return (
    <div>
      <h1>Snapgram</h1>
<input value={content} onChange={(e) => setContent(e.target.value)} placeholder="Write something..." />
      <button onClick={addPost}>Post</button>

      {posts.map((post) => (
        <div key={post._id}>
          <h3>{post.user}</h3>
          <p>{post.content}</p>
        </div>
      ))}
    </div>
  );
};

export default App;
```