

Report on
FEEDBACK LOOP

By
SHIVAM SHARMA (2300290140170)
Session: 2024-2025 (III Semester)

Under the supervision of

Dr. VIPIN KUMAR
ASSOCIATE PROFESSOR

KIET Group of Institutions, Delhi-NCR, Ghaziabad



DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR,
GHAZIABAD-201206
(BATCH 2023-2025)

DECLARATION

We hereby declare that the work presented in this report entitled “**FEEDBACK LOOP**”, was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

We have given due credit to the original authors/sources for all the words, ideas, diagrams graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of my work is plagiarized, and the experiments and results reported in there port are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

SHIVAM SHARMA

(2300290140170)

ACKNOWLEDGEMENT

At the outset, we would like to thank our guide and advisor, **Dr. VIPIN KUMAR Associate Professor**, for giving us an opportunity to work on this challenging topic and providing us ample and valuable guidance throughout the Project.

Without his encouragement and constant guidance, we would not have been able to finish this project. He has been always a source of inspiration and motivator for innovative ideas during the entire span of this work.

We are grateful **Dr. Arun Kumar Tripathi, Head Department of Computer Applications, KIET Group of Institutions, Ghaziabad** for providing all the necessary resources to carry out this Project work.

We will be failing in our duty if we don't acknowledge the people behind this work to give us moral and psychological support. Our special thanks to our parents for their endless care and constant support.

SHIVAM SHARMA

(2300290140170)

ABSTRACT

Feedback Loop is designed to be your central hub for seamless project collaboration. Think of it as the ultimate space where you can manage, track, and enhance every aspect of your project without the complexity.

Whether you're a project manager, team member, or stakeholder, Feedback Loop brings everyone together on the same page. It provides all the essential tools for efficient planning, task assignment, progress tracking, document storage, and team communication.

By consolidating all these features into one platform, Feedback Loop ensures that everyone stays organized, informed, and aligned, making it easier to navigate projects from start to finish. It's not just about simplifying workflows—it's about fostering stronger collaboration and driving project success through effective feedback and continuous improvement.

CERTIFICATE

Certified that **Shivam Sharma** has carried out the project work having **“FEEDBACK LOOP” (Mini-Project - KCA353)** for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU) Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate to anybody else from this or any University/Institution.

Mr. Vipin Kumar

Associate Professor

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

Dr. Arun Kumar Tripathi

Dean

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

TABLE OF CONTENTS

Declaration	ii
Certificate	iii
Abstract	iv
Acknowledgements	v
Table of Contents	vi
List of Figures	vii
1. Introduction	1-3
1.1 Overview	1
1.2 Project Description	2
1.3 Project Scope	3
1.4 Hardware/Software used in Project	3
2. Feasibility Study	4-6
2.1 Economical Feasibility	4
2.2 Technical Feasibility	5
2.3 Operational Feasibility	6
3. Database Design	7-15
3.1 SDLC	8
3.2 Use Case Diagram	10
3.3 ER Diagram	12

4 Project Screenshot & Coding	21-25
4.1 Home Page	16
4.2 Sign-In Page	17
4.3 Log In	17
4.4 About page	18
4.5 Tag page	19
4.6 User Page	20
4.7 Profile Page	21
4.8 Question Section	22
4.9 Answer section	23
Coding	24-49
1 Frontend	24-42
1.1 AskQuestion.jsx	24-27
1.2 Auth.jsx	28-32
1.3 Hom.jsx	33
1.4 DisplayAnswer.jsx	34-35
1.5 QuestionDetail.jsx	36-42
2 Backend	43-
2.1 Answer.js	43-45
2.2 Question.js	45-48
2.3 Index.js	48-49
5 Testing	50-51
6 Conclusion and Future Scope	52-54
7 References	55

CHAPTER-1

INTRODUCTION

1.1 Overview

In today's world, sharing ideas is crucial for new breakthroughs. Feedback Loop stands out as a game-changing platform for people looking to get helpful input and work together. This new system links users with a group of experts and peers letting them share ideas and get personalized tips. By encouraging meaningful talks and promoting knowledge sharing, Feedback Loop helps users fine-tune their thoughts widen their views, and grow through the wisdom of the crowd.

1.2 Project Description

In today's quick-moving world where teamwork and helpful criticism drive new ideas, the way we polish our thoughts often lacks well-organized and useful input. Feedback Loop tackles this problem by setting up a lively platform that links people who share ideas with a wide range of experts and peers. This system uses advanced formulas to study ideas that users submit and how they interact, allowing for custom-made input that matches personal aims and creative visions.

By bringing in live talks, ways to share knowledge and tools to track progress, Feedback Loop offers a complete setting to come up with and improve ideas. The platform not only boosts individual concepts but also creates an environment where people can learn from each other and offer support.

Feedback Loop is a complete solution for those who want to take their ideas to the next level. It helps with coming up with new projects, making existing ideas better, and learning from experienced pros. Committed to encouraging new ideas and inclusivity, Feedback Loop is a must-have tool for creators, business owners, and thinkers looking to change the way feedback and collaboration work.

1.3 Project Scope

The scope of this project goes beyond just being a feedback platform; it aims to create a comprehensive system that not only encourages the exchange of ideas but also adapts to the changing needs and aspirations of its users. By analyzing user interactions and understanding their goals, the system is crafted to offer insightful, personalized, and context-aware suggestions. Feedback Loop also focuses on promoting active engagement, using user feedback to enhance its processes and support ongoing improvement. This ensures that the platform evolves alongside its users, transforming into a dynamic ecosystem for collaboration, creativity, and knowledge-sharing.

Hardware: Windows 10 or 11

Backend

Developed using Node.js, the backend leverages frameworks like Express.js to manage the web application's core functionality and API Integration.

Frontend

User Interface: The frontend of Feedback Loop is developed using React.js, ensuring dynamic and responsive user experience. The use combination HTML, CSS, and JavaScript creates a visually appealing and intuitive interface. Allowing users to navigate seamlessly through the platform.

IDE (Integrated Development Environment): Visual Studio Code (VS Code) is the primary IDE used for the development of Feedback.

Browser: Chrome, Edge

CHAPTER-2

FEASIBILITY STUDY

A feasibility study is crucial for evaluating whether a project is worth the investment of substantial resources and time. Here is an assessment of the Feedback Loop system based on three important aspects: Economic, Technical, and Operational.

2.1 Economical Feasibility

Development Costs: Includes hiring developers, designing the platform, And integrating feedback algorithms and machine Learning tools. Cloud services like AWS, Google Cloud will also incur ongoing costs.

Hardware/Software Costs: Monthly cloud hosting, software licenses and to integration with third-party tools (e.g., analytics, notifications).

Maintenance Costs: Includes bug fixes, system updates, server to be Management and ensuring security.

2.2 Technical Feasibility

System Architecture:

Frontend (Vercelli): Use React.js for a dynamic user interface. Vercel Handles Automatic scaling and fast deployment.

Backend (Netlify): Use Netlify serverless functions with Node.js and Express.js to process API requests (e.g ., feedback Submission) . Functions scale automatically with demand.

Database: MongoDB Atlas stores user data and feedback, integrate seamlessly with the backend.

Technology Stack:

- **Frontend:** React.js (or Next.js), Axios for API calls.
- **Backend:** Node.js, Express.js, Netlify serverless functions for API handling.
- **Database:** MongoDB Atlas for storing user data.

2.3 Operational Feasibility

User Acceptance: Based on feedback, assess how well users perceive the value of the platform in improving their ideas and whether they would adopt it. Users are likely to appreciate the personalized feedback and expert insights that Feedback Loop offers.

Operational Impact: Feedback Loop seamlessly integrates into existing

workflows for creators and teams. Users can easily submit ideas, receive feedback, and track without improvements without disrupting their processes.

CHAPTER-3

DESIGN AND PLANNING

3.1 Software development life cycle model prototype model

The Prototype model requires that before carrying out the development software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possibly exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product, the processing needs, and the output requirement, the prototyping model may be employed.

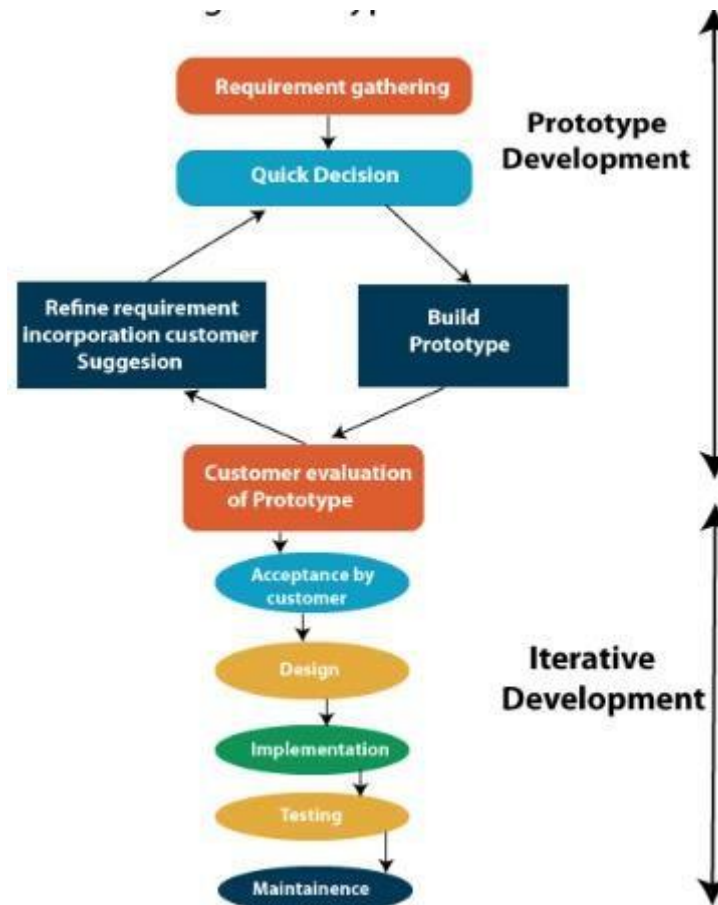


Fig 3.1 Prototype Model

3.1.1 Requirement Gathering:

In the initial phase, the primary focus is on gathering detailed requirements for the Feedback Loop platform. This involves engaging with a range of stakeholders including potential users (such as idea creators, entrepreneurs, and professionals) and experts who will provide feedback. Various methods, such as surveys, interviews, and focus groups, are employed to understand user needs and expectations.

The key requirements include:

User Registration & Profiles: Users should be able to register and create Profiles, providing details about their expertise or ideas.

Idea Submission: Users should be able to submit their ideas for feedback, including uploading relevant documents or files.

Feedback Mechanism: The system needs to allow users to receive feedback

from experts or peers.

Dashboard: Users need an intuitive dashboard to manage submitted ideas, feedback, and interactions.

Security and Privacy: Ensuring that the platform is secure and user data is protected.

3.1.2 System Design

The System Design phase is all about creating the architecture of the Platform to ensure it meets both functional and non-functional Requirements.

Technical level and will cover:

Architecture Design: The Feedback Loop will be structured using a client Server model, where the React.js frontend interacts with the Node.js. Backend through to the Restful APIs, User data, ideas, feedback, and Interactions will be stored in a MongoDB database.

The backend will utilize Express.js to set up a server and manage API requests for the features such as user authentication, idea Submission, feedback retrieval. The Security measures like JWT Authentication and data validation will be implemented to guarantee that only authorized users can access certain data.

Database Design: The database schema will be organized and manage user Profiles, ideas, feedback, and notifications. MongoDB Atlas will be employed for scalable and cloud-based data storage, providing Flexibility and high availability.

UI/UX Design: Comprehensive wireframes and user interface mockups will be developed and ensure a smooth and intuitive experience for users.

This will include designing screens for:

- 1- User registration/login
- 2- Idea submission and feedback pages
- 3- Real-time messaging
- 4- Dashboards to monitor feedback and idea progress.

3.1.3 Development

In this phase, Development follows an agile approach. The Frontend will be built with React.js, while the backend uses Node.js, Express.js, MongoDB. Key features like to be use Authentication with JWT , idea and Submission, feedback posting, and to the real-time communication will be implemented.

3.1.4 Testing

Testing encompasses unit testing, integration testing, system testing, and User acceptance testing (UAT). Additionally, performance and security Testing will guarantee that the platform remains stable, scalable, and Secure.

3.1.5 Deployment

Once the smart parking system has been thoroughly tested and Deemed ready for use I deployed in the target environment. This involves installing and configuring the necessary hardware and software component , ensuring compatibility with the existing Infrastructure and training staff on how to use and manage the System.

3.1.6 Maintenance and Support

After deployment, Feedback Loop will need ongoing maintenance and Support to keep it running smoothly. This involves fixing bugs, optimizing Performance, and regularly updating the software to enhance features and Security. Additionally, technical support will be available to help users and Administrators with any issues they may face. The maintenance phase is the Crucial for keeping the platform relevant by responding to user feedbackchanging needs. Throughout the Software Development Life Cycle (SDLC),it important adhere best practices in project management, documentation , and Communication ensure the successful development and implementationof Feedback Loop. There will be a strong focus on security and data Privacy,making sure that sensitive user information is safeguarded that all regulatoryrequirements are fulfilled to prevent unauthorized Access.

3.2 USE CASE DIAGRAM:

Use-case diagrams model the behavior of a system and help to Capture the requirements of the system. Use-case diagrams Describe the high-level functions and scope of a system. These Diagrams also identify the interactions between the system and its Actors.

A use case diagram is used to represent the dynamic behavior of a System. It encapsulates the system's functionality by incorporating Use cases, actors, and their relationships. It models the tasks, Services, and functions required by a system / subsystem of an Application. It depicts the high level functionality of a system and also tells how the user handles a system.

Purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

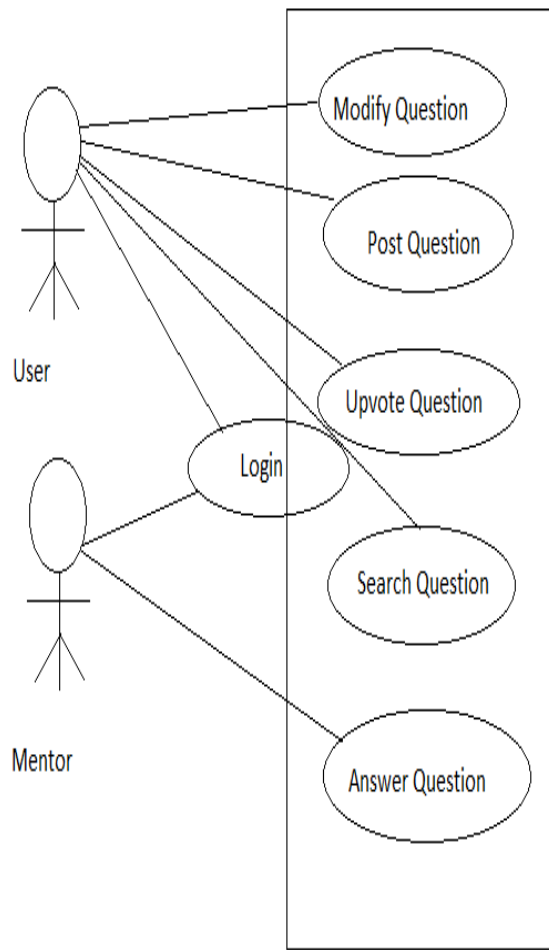


Fig3.2 Use Case Diagram

3.3 ER(Entity Relationship)Diagram

1-ER model stands for an Entity-Relationship model. It is a high-level data model . This model is used to define the data elements relation for a specified system.

2-It develop conceptual design for the database.It also develop simple and design view of data.

3-In ER modeling , the data base structure portrayed as a diagram call an entity-relationship diagram.

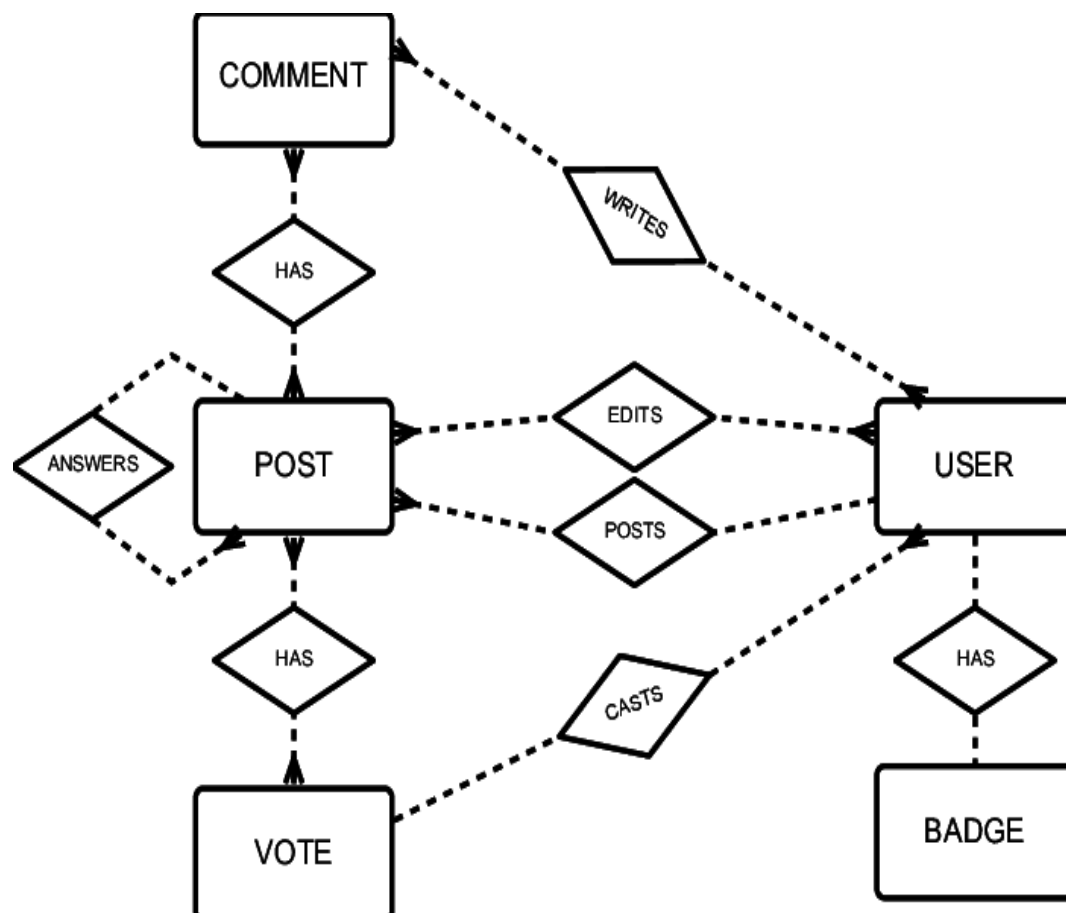


Fig3.3 ER Diagram

Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored. The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called a data flow graph or bubble chart.

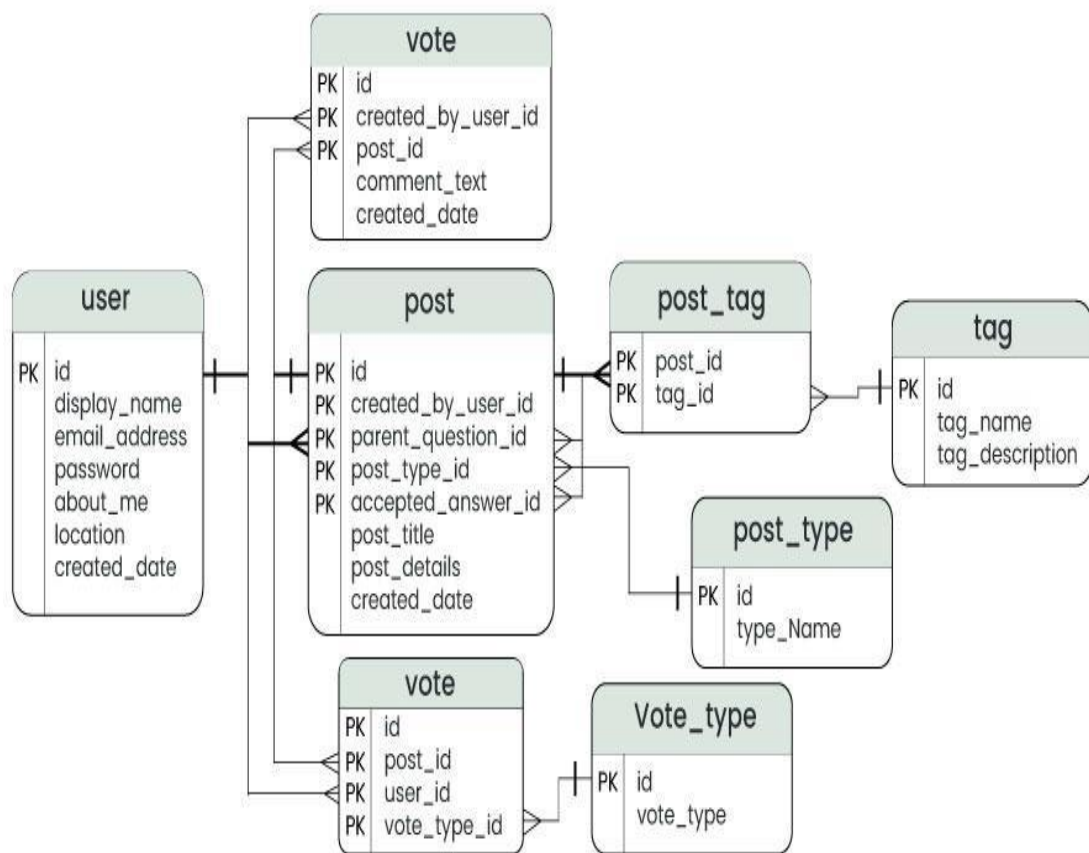


Fig3.4 Data Flow Diagram

CHAPTER-4

PROJECT

SCREENSHOTS & CODDING

4.1 Home page

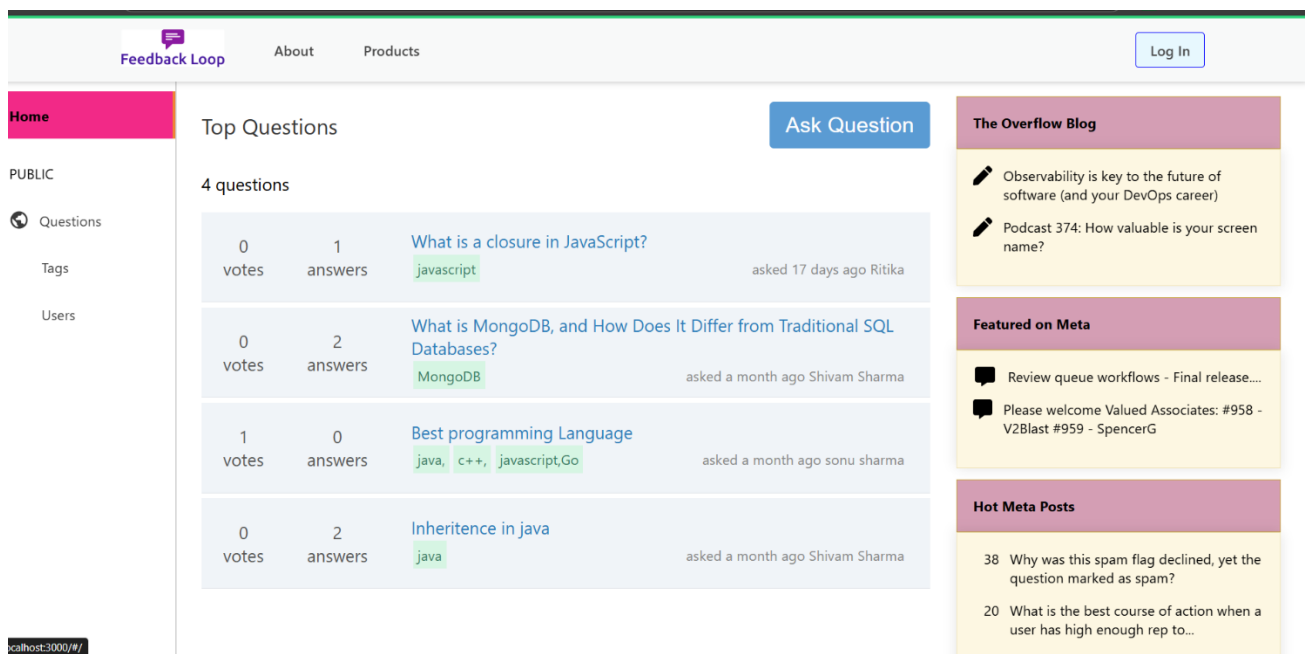
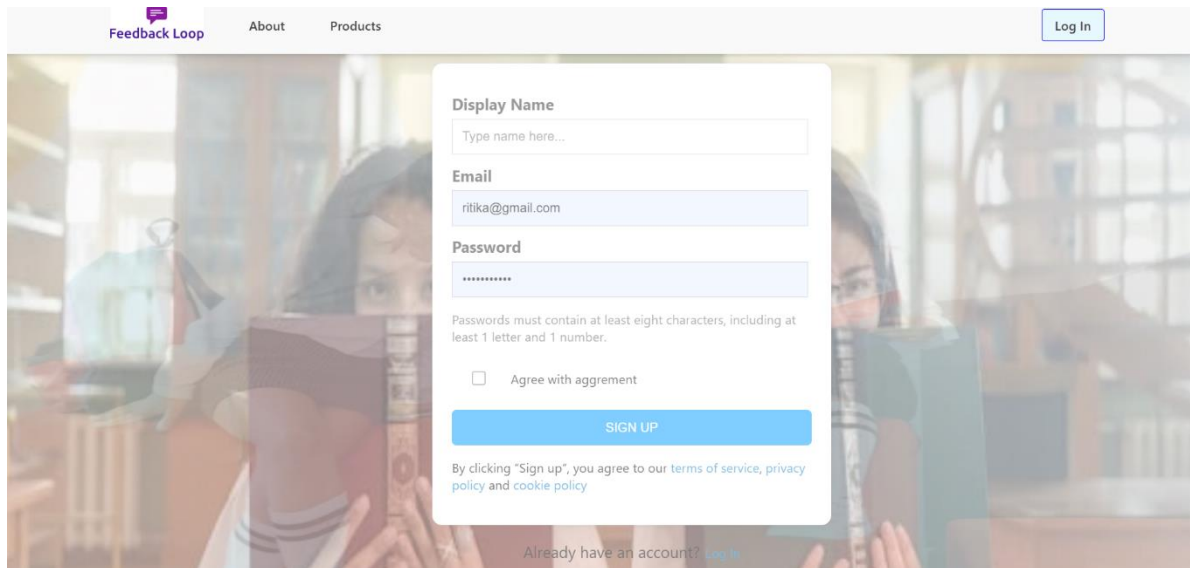


Fig 4.1 Home Page

4.2 Sign in

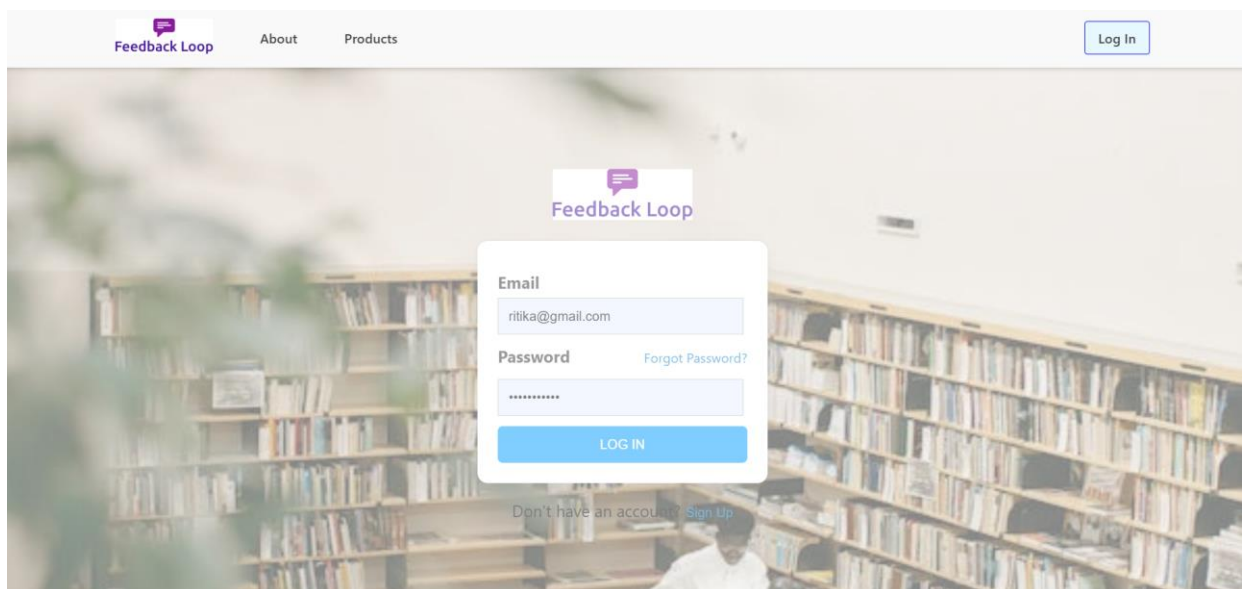


The image shows a web application interface for 'Feedback Loop'. The background is a blurred photo of a library with bookshelves and people. A white sign-up form is centered on the screen. The form has a header 'Feedback Loop' with a speech bubble icon. Below the header, there are navigation links 'About' and 'Products'. In the top right corner, there is a 'Log In' button. The sign-up form itself contains the following elements:

- Display Name:** A text input field with the placeholder 'Type name here...'.
- Email:** A text input field containing 'ritika@gmail.com'.
- Password:** A password input field with masked characters '*****'.
- Password Requirements:** A note stating 'Passwords must contain at least eight characters, including at least 1 letter and 1 number.'
- Agreement:** A checkbox labeled 'Agree with aggrement'.
- SIGN UP:** A blue button.
- Footer:** Text stating 'By clicking "Sign up", you agree to our [terms of service](#), [privacy policy](#) and [cookie policy](#)'.
- Link:** Text at the bottom saying 'Already have an account? [Log In](#)'.

Fig 4.2 Sign in

4.3 Log In



The image shows the same web application interface as Fig 4.2, but with a log-in form overlay. The background is a blurred photo of a library. The log-in form is centered and contains the following elements:

- Header:** 'Feedback Loop' with a speech bubble icon.
- Navigation:** 'About' and 'Products' links.
- Log In Button:** A blue button in the top right corner.
- Email:** A text input field containing 'ritika@gmail.com'.
- Password:** A password input field with masked characters '*****'.
- Forgot Password?:** A link next to the password field.
- LOG IN:** A blue button.
- Footer:** Text at the bottom saying 'Don't have an account? [Sign Up](#)'.

Fig 4.3 Log In

4.4 About Page:

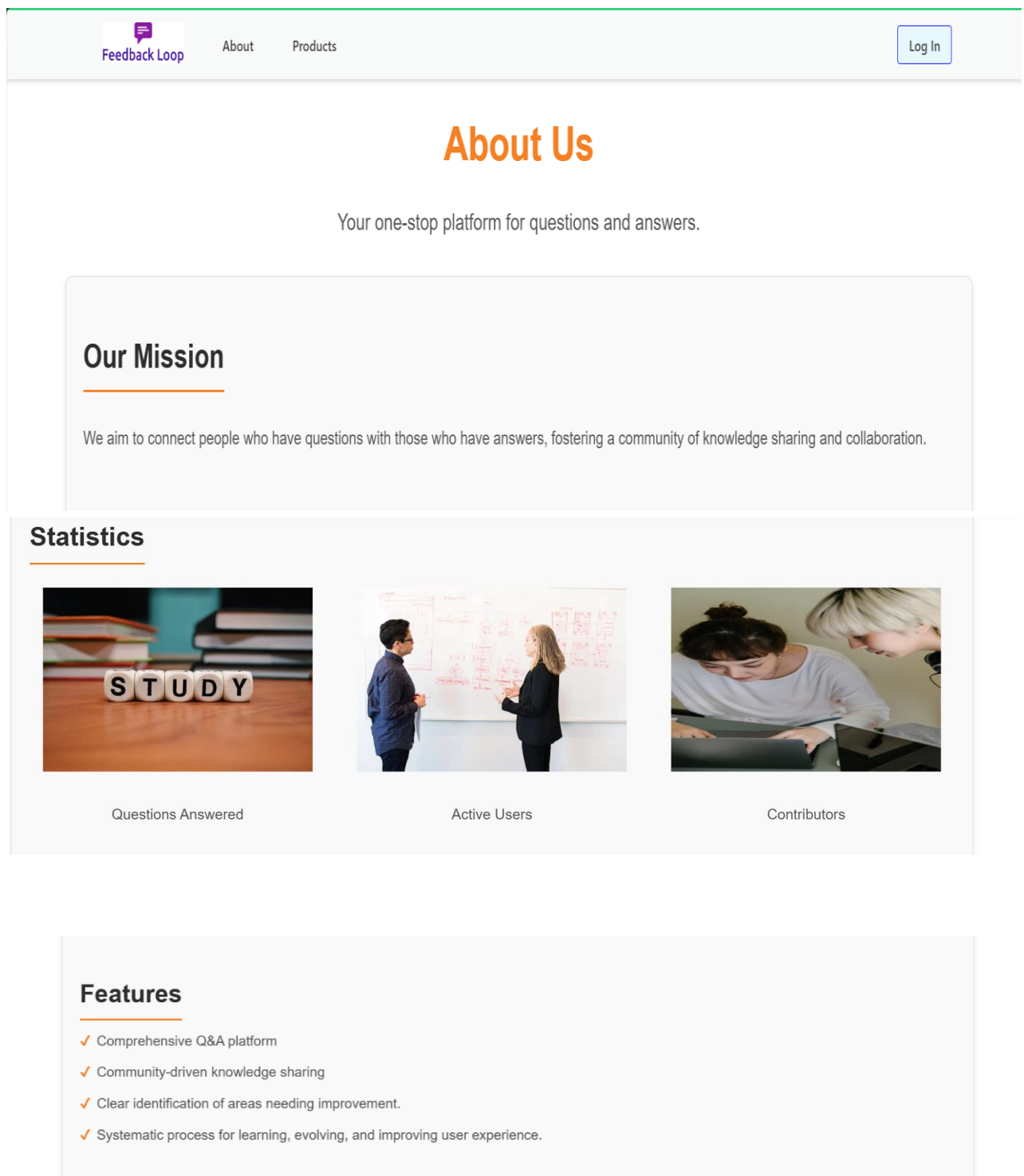


Fig 4.4 About Us

4.5 Tags page

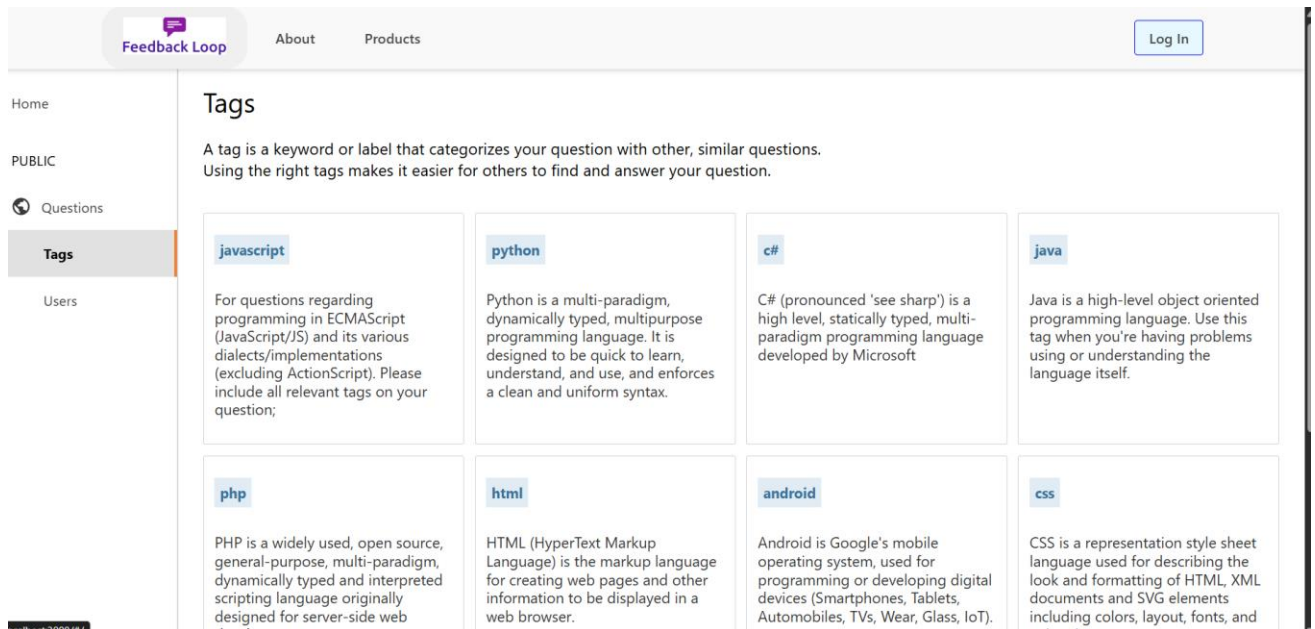


Fig 4.5 Tags Page

4.6 User page

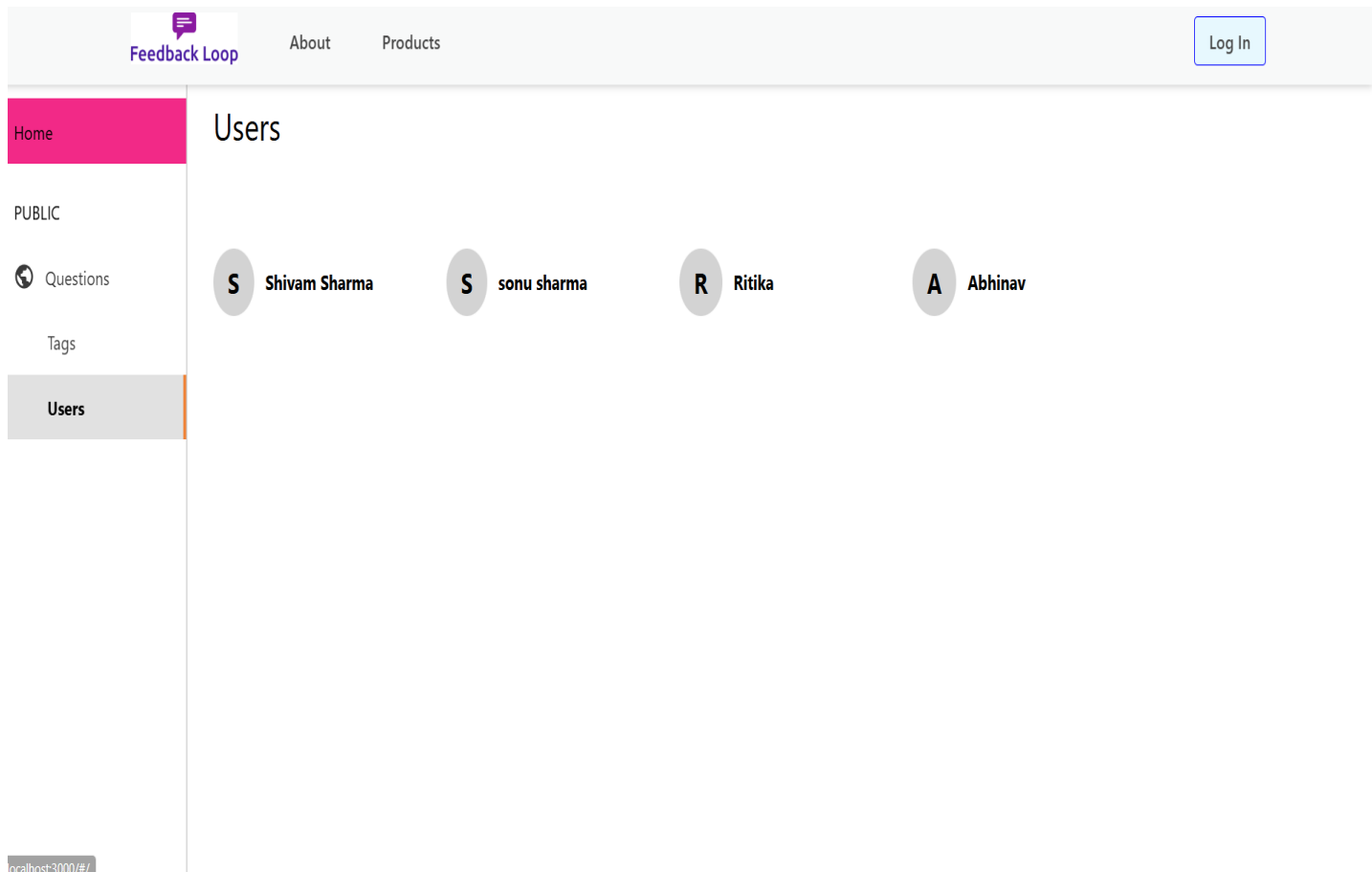




Fig 4.6 User Page

4.7 Profile Page



Shivam Sharma
 Joined a month ago

Tags watched

- java
- html
- javascript
- css

About

I am a MCA student from KIET

Fig 4.7 Profile Page

4.8 Question Section

Feedback Loop

About

Products

R

Log Out

Ask a Public Question

Title
Be specific and imagine you're asking a question to another person

Body
Include all the information someone would need to answer your question

Tags
Add up to 5 tags to describe what your question is about

Review Your Question

Fig 4.8 Question Section

4.9 Answer Section

Feedback Loop

AboutProducts

RLog Out

Home

PUBLIC

Questions

Tags

Users

What is a closure in JavaScript?

▲

closure in javascript

▼

0

javascript

ShareDelete

asked 17 days ago

R Ritika

1 Answers

A closure is the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment). When you create a closure, you gain access to an outer function's scope from an inner function. Closures are automatically created every time a function is defined in JavaScript.

Share

answered 16 days ago

A Abhinav

Your Answer

Post Your Answer

Browse other Question tagged `javascript` or [ask your own question](#).

The Overflow Blog

- Observability is key to the future of software (and your DevOps career)
- Podcast 374: How valuable is your screen name?

Featured on Meta

- Review queue workflows - Final release...
- Please welcome Valued Associates: #958 - V2Blast #959 - SpencerG

Hot Meta Posts

- 38 Why was this spam flag declined, yet the question marked as spam?
- 20 What is the best course of action when a user has high enough rep to...
- 14 Is a link to the "How to ask" help page a useful comment?

Watched tags

`c``css``express``firebase``html``java`

Fig. 4.9 Answer Section

CODING:

FRONTEND

(AskQuestion.js)

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import { useDispatch, useSelector } from "react-redux";
import { askQuestion } from "../../actions/question";
import "../AskQuestion.css";

const AskQuestion = () => {
  const [questionTitle, setQuestionTitle] = useState("");
  const [questionBody, setQuestionBody] = useState("");
  const [questionTags, setQuestionTags] = useState("");
  const User = useSelector((state) => state.currentUserReducer);
  const navigate = useNavigate();
  const dispatch = useDispatch();
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log({ questionTitle, questionBody, questionTags });
    dispatch(
      askQuestion(
        {
          questionTitle,
          questionBody,
```

```

    questionTags,
    userPosted: User.result.name,
  },
  navigate
)
);
};

const handleEnter = (e) => {
  if (e.key === "Enter") {
    setQuestionBody(questionBody + "\n");
  }
};

return (
  <div className="ask-question">
    <div className="ask-ques-container">
      <h1>Ask a Public Question</h1>
      <form className="askquesform" onSubmit={handleSubmit}>
        <div className="ask-form-container">
          <label htmlFor="ask-ques-title">
            <h4>Title</h4>
            <p>
              Be specific and imagine you're asking a question to another
              person
            </p>
            <input
              type="text"

```



```
id="ask-ques-title"
onChange={(e) => {
  setQuestionTitle(e.target.value);
}}
placeholder="e.g. Is there an R function for finding the index of an
element in a vector?"
```

```
/>
```

```
</label>
```

```
<label htmlFor="ask-ques-body">
```

```
<h4>Body</h4>
```

```
<p>
```

Include all the information someone would need to answer your question

```
</p>
```

```
<textarea
```

```
name=""
```

```
id="ask-ques-body"
```

```
onChange={(e) => {
```

```
  setQuestionBody(e.target.value);
```

```
}}

```

```
cols="30"
```

```
rows="10"
```

```
onKeyDown={handleEnter}
```

```
></textarea>
```

```
</label>
```

```
<label htmlFor="ask-ques-tags">
```

```

    <h4>Tags</h4>
    <p>Add up to 5 tags to describe what your question is about</p>
    <input
      type="text"
      id="ask-ques-tags"
      onChange={(e) => {
        setQuestionTags(e.target.value.split(" "));
      }}
      placeholder="e.g. (xml typescript wordpress)"
    />
  </label>
</div>
<input
  type="submit"
  value="Review Your Question"
  className="review-btn"
/>
</form>
</div>
</div>
);
};

export default AskQuestion;

```

(Auth.jsx)

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import { useDispatch } from "react-redux";
import AboutAuth from "../AboutAuth";
import logo from "../../assets/logo.png";
import { signup, login } from "../../actions/auth";
import "../Auth.css";
const Auth = () => {
  const [isSignup, setIsSignup] = useState(false);
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const handleSubmit = (e) => {
    e.preventDefault();
    if (!email && !password) {
      alert("Enter email and password");
    }
    if (isSignup) {
      if (!name) {
        alert("Enter a name to continue");
      }
      dispatch(signup({ name, email, password }, navigate));
      console.log({ name, email, password });
    }
  }
}
```

```

    } else {
      dispatch(login({ email, password }, navigate));
    }
  };
const handleSwitch = () => {
  setIsSignup(!isSignup);
};
return (
  <section className="auth-section">
    {isSignup && <AboutAuth />}
    <div className="auth-container-2">
      { !isSignup && (
        <img src={logo} alt="feedback" className="login-logo" />
      )}
      <form onSubmit={handleSubmit}>
        {isSignup && (
          <label htmlFor="name">
            <h4>Display Name</h4>
            <input
              type="text"
              id="name"
              name="name"
              placeholder="Type name here..."
              onChange={(e) => {
                setName(e.target.value);
              }}
            />
          )}
        }
      </form>
    </div>
  </section>
)

```

```

    />
  </label>
)}
<label htmlFor="email">
  <h4>Email</h4>
  <input
    type="email"
    name="email"
    id="email"
    placeholder="Type email here..."
    onChange={ (e) => {
      setEmail(e.target.value);
    }}
  />
</label>
<label htmlFor="password">
  <div style={{ display: "flex", justifyContent: "space-between" }}>
    <h4>Password</h4>
    { !isSignup && (
      <p
        className="forgot-password"
        style={{ color: "#007ac6", fontSize: "13px" }}
      >
        Forgot Password?
      </p>
    )}
  </div>

```

```

</div>
<input
  type="password"
  name="password"
  id="password"
  placeholder="Type password here..."
  onChange={ (e) => {
    setPassword(e.target.value);
  }}
/>
{isSignup && (
  <p style={{ color: "#666767", fontSize: "13px" }}>
    Passwords must contain at least eight characters, including at
    least 1 letter and 1 number.
  </p>
)}
</label>
{isSignup && (
  <label htmlFor="check">
    <input type="checkbox" id="check" />
    <p style={{ fontSize: "13px" }}>
      Agree with agreement
    </p>
  </label>
)}
<button type="submit" className="auth-btn">

```

```

        {isSignup ? "SIGN UP" : "LOG IN"}
    </button>
    {isSignup && (
        <p style={{ color: "black", fontSize: "13px" }}>
            By clicking “Sign up”, you agree to our
            <span style={{ color: "#007ac6" }}> terms of service</span>,
            <span style={{ color: "#007ac6" }}> privacy policy</span> and
            <span style={{ color: "#007ac6" }}> cookie policy</span>
        </p>
    )}
</form>
<p>
    {isSignup ? "Already have an account?" : "Don't have an account?"}
    <button
        type="button"
        className="handle-switch-btn"
        onClick={handleSwitch}
    >
        {isSignup ? "Log In" : "Sign Up"}
    </button>
</p>
</div>
</section>
);
};

```

```
export default Auth;
```

(Home.jsx)

```
import React from "react";
import LeftSidebar from "../../components/LeftSidebar/LeftSidebar";
import HomeMainbar from "../../components/HomeMainbar/HomeMainbar";
import RightSidebar from "../../components/RightSidebar/RightSidebar.jsx";
import "../../App.css";
const Home = () => {
  return (
    <div className="home-container-1">
      <LeftSidebar />
      <div className="home-container-2">
        <HomeMainbar />
        <RightSidebar />
      </div>
    </div>
  );
};
export default Home;
```

(DisplayAnswer.jsx)

```
import React from "react";
import { Link, useParams } from "react-router-dom";
import { useSelector, useDispatch } from "react-redux";
import moment from "moment";
```



```

import Avatar from "../../components/Avatar/Avatar";
import { deleteAnswer } from "../../actions/question";
const DisplayAnswer = ({ question, handleShare }) => {
  const { id } = useParams();
  const dispatch = useDispatch();
  const User = useSelector((state) => state.currentUserReducer);
  const handleDelete = (answerId, noOfAnswers) => {
    dispatch(deleteAnswer(id, answerId, noOfAnswers - 1));
  };
  return (
    <div>
      {question.answer.map((ans) => (
        <div className="display-ans" key={ans._id}>
          <p>{ans.answerBody}</p>
          <div className="question-actions-user">
            <div>
              <button type="button" onClick={handleShare}>
                Share
              </button>
              {User?.result?._id === ans?.userId && (
                <button
                  type="button"
                  onClick={() => handleDelete(ans._id, question.noOfAnswers)}
                >
                  Delete
                </button>
              )}
            </div>
          </div>
        </div>
      )}
    </div>
  );
}

```

```

    })
</div>
<div>
  <p>answered { moment(ans.answeredOn).fromNow() }</p>
  <Link
    to={` /Users/${ ans.userId }`}
    className="user-link"
    style={{ color: "#0086d8" }}
  >
    <Avatar
      backgroundColor="lightgreen"
      px="8px"
      py="5px"
      borderRadius="4px"
    >
      { ans.userAnswered.charAt(0).toUpperCase() }
    </Avatar>
  <div>{ ans.userAnswered }</div>
</Link>
</div>
</div>
</div>
  )))
</div>
);
};

```

```
export default DisplayAnswer;
```

(QuestionDetail.jsx)

```
import React, { useState } from "react";
import { useParams, Link, useNavigate, useLocation } from "react-
router-dom";
import { useSelector, useDispatch } from "react-redux";
import moment from "moment";
import copy from "copy-to-clipboard";
import DisplayAnswer from "../DisplayAnswer";
import Avatar from "../../components/Avatar/Avatar";
import upvote from "../../assets/sort-up.svg";
import downvote from "../../assets/sort-down.svg";
import {
  postAnswer,
  deleteQuestion,
  voteQuestion,
} from "../../actions/question";
import "../Questions.css";
const QuestionsDetails = () => {
  const { id } = useParams();
  const User = useSelector((state) => state.currentUserReducer);
  const questionsList = useSelector((state) => state.questionsReducer);
  const [Answer, setAnswer] = useState("");
  const Navigate = useNavigate();
```

```

const dispatch = useDispatch();
const location = useLocation();
const url = "http://localhost:3000/";
const handlePostAns = (e, answerLength) => {
  e.preventDefault();
  if (User === null) {
    alert("Login or Signup to answer a question");
    Navigate("/Auth");
  } else {
    if (Answer === "") {
      alert("Enter an answer before submitting");
    } else {
      dispatch(
        postAnswer({
          id,
          noOfAnswers: answerLength + 1,
          answerBody: Answer,
          userAnswered: User.result.name,
        })
      );
    }
  }
};

const handleShare = () => {
  copy(url + location.pathname);
  alert("Copied url : " + url + location.pathname);
};

```

```

};

const handleDelete = () => {
  dispatch(deleteQuestion(id, Navigate));
};

const handleUpVote = () => {
  dispatch(voteQuestion(id, "upVote"));
};

const handleDownVote = () => {
  dispatch(voteQuestion(id, "downVote"));
};

return (
  <div className="question-details-page">
    {questionsList.data === null ? (
      <h1>Loading...</h1>
    ) : (
      <div>
        {questionsList.data
          .filter((question) => question._id === id)
          .map((question) => (
            <div key={question._id}>
              <section className="question-details-container">
                <h1>{question.questionTitle}</h1>
                <div className="question-details-container-2">
                  <div className="question-votes">
                    <img

```

```

src={upvote}
alt=""
width="18"
className="votes-icon"
onClick={handleUpVote}
/>
<p>{question.upVote.length - question.downVote.length}
</p>
<img
src={downvote}
alt=""
width="18"
className="votes-icon"
onClick={handleDownVote}
/>
</div>
<div style={{ width: "100%" }}>
<p className="question-body">{question.questionBody}</p>
<div className="question-details-tags">
  {question.questionTags.map((tag) => (
    <p key={tag}>{tag}</p>
  ))}
</div>
</div>
<div className="question-actions-user">
  <div>

```

```

<button type="button" onClick={handleShare}>
  Share
</button>

{User?.result?._id === question?.userId && (
  <button type="button" onClick={handleDelete}>
    Delete
  </button>
)}
</div>
<div>
  <p>asked {moment(question.askedOn).fromNow()}</p>
  <Link
    to={`/${Users}/${question.userId}`}
    className="user-link"
    style={{ color: "#0086d8" }}
  >
    <Avatar
      backgroundColor="orange"
      px="8px"
      py="5px"
      borderRadius="4px"
    >
      {question.userPosted.charAt(0).toUpperCase()}
    </Avatar>
  </div>{question.userPosted}</div>
</Link>

```

```

        </div>
    </div>
</div>
</section>
{question.noOfAnswers !== 0 && (
    <section>
        <h3>{question.noOfAnswers} Answers</h3>
        <DisplayAnswer
            key={question._id}
            question={question}
            handleShare={handleShare}
        />
    </section>
)}
<section className="post-ans-container">
    <h3>Your Answer</h3>
    <form
        onSubmit={ (e) => {
            handlePostAns(e, question.answer.length);
        }}
    >
        <textarea
            name=""
            id=""
            cols="30"
            rows="10"

```



```

      onChange={(e) => setAnswer(e.target.value)}
    ></textarea>

    <br />

    <input
      type="Submit"
      className="post-ans-btn"
      value="Post Your Answer"
    />
  </form>

  <p>
    Browse other Question tagged
    {question.questionTags.map((tag) => (
      <Link to="/Tags" key={tag} className="ans-tags">
        {" "}
        {tag} {" "}
      </Link>
    ))} {" "}
    or
    <Link
      to="/AskQuestion"
      style={{ textDecoration: "none", color: "#009dff" }}
    >
      {" "}
      ask your own question.
    </Link>
  </p>

```

```

        </section>
      </div>
    )))}
  </>
  )}
</div>

);
};

export default QuestionsDetails;

```

Backend:

(Answer.js)

```

import mongoose from "mongoose";
import Questions from "../models/questions.js";
export const postAnswer = async (req, res) => {
  const { id: _id } = req.params;
  const { noOfAnswers, answerBody, userAnswered } = req.body;
  const userId = req.userId;
  if (!mongoose.Types.ObjectId.isValid(_id)) {
    return res.status(404).send("question unavailable...");
  }
  updateNoOfQuestions(_id, noOfAnswers);
  try {

```

```

    const updatedQuestion = await Questions.findByIdAndUpdate(_id, {
      $addToSet: { answer: [{ answerBody, userAnswered, userId }] },
    });
    res.status(200).json(updatedQuestion);
  } catch (error) {
    res.status(400).json("error in updating");
  }
};

const updateNoOfQuestions = async (_id, noOfAnswers) => {
  try {
    await Questions.findByIdAndUpdate(_id, {
      $set: { noOfAnswers: noOfAnswers },
    });
  } catch (error) {
    console.log(error);
  }
};

export const deleteAnswer = async (req, res) => {
  const { id: _id } = req.params;
  const { answerId, noOfAnswers } = req.body;
  if (!mongoose.Types.ObjectId.isValid(_id)) {
    return res.status(404).send("Question unavailable...");
  }
  if (!mongoose.Types.ObjectId.isValid(answerId)) {
    return res.status(404).send("Answer unavailable...");
  }
}

```

```

updateNoOfQuestions(_id, noOfAnswers);
try {
  await Questions.updateOne(
    { _id },
    { $pull: { answer: { _id: answerId } } }
  );
  res.status(200).json({ message: "Successfully deleted..." });
} catch (error) {
  res.status(405).json(error);
}
};

```

(Question.js)

```

import mongoose from "mongoose";
import Questions from "../models/questions.js";
export const AskQuestion = async (req, res) => {
  const postQuestionData = req.body;
  const userId = req.userId;
  const postQuestion = new Questions({ ...postQuestionData, userId });
  try {
    await postQuestion.save();
    res.status(200).json("Posted a question successfully");
  } catch (error) {
    console.log(error);
    res.status(409).json("Couldn't post a new question");
  }
}

```

```
};
```

```
export const getAllQuestions = async (req, res) => {  
  try {  
    const questionList = await Questions.find().sort("-askedOn");  
    res.status(200).json(questionList);  
  } catch (error) {  
    res.status(404).json({ message: error.message });  
  }  
};
```

```
export const voteQuestion = async (req, res) => {  
  const { id: _id } = req.params;  
  const { value } = req.body;  
  const userId = req.userId;  
  if (!mongoose.Types.ObjectId.isValid(_id)) {  
    return res.status(404).send("question unavailable...");  
  }  
  try {  
    const question = await Questions.findById(_id);  
    const upIndex = question.upVote.findIndex((id) => id === String(userId));  
    const downIndex = question.downVote.findIndex(  
      (id) => id === String(userId)  
    );  
    if (value === "upVote") {  
      if (downIndex !== -1) {  
        question.downVote = question.downVote.filter(  

```

```

        (id) => id !== String(userId)
    );
}
if (upIndex === -1) {
    question.upVote.push(userId);
} else {
    question.upVote = question.upVote.filter((id) => id !== String(userId));
}
} else if (value === "downVote") {
    if (upIndex !== -1) {
        question.upVote = question.upVote.filter((id) => id !== String(userId));
    }
    if (downIndex === -1) {
        question.downVote.push(userId);
    } else {
        question.downVote = question.downVote.filter(
            (id) => id !== String(userId)
        );
    }
}
await Questions.findByIdAndUpdate(_id, question);
res.status(200).json({ message: "voted successfully..." });
} catch (error) {
    res.status(404).json({ message: "id not found" });
}
};

```

```

export const deleteQuestion = async (req, res) => {
  const { id: _id } = req.params;
  if (!mongoose.Types.ObjectId.isValid(_id)) {
    return res.status(404).send("question unavailable...");
  }
  try {
    await Questions.findByIdAndRemove(_id);
    res.status(200).json({ message: "successfully deleted..." });
  } catch (error) {
    res.status(404).json({ message: error.message });
  }
};

```

(Index.js)

```

import cors from "cors";
import dotenv from "dotenv";
import express from "express";
import mongoose from "mongoose";
import userRoutes from "./routes/users.js";
import answerRoutes from "./routes/Answers.js";
import questionRoutes from "./routes/Questions.js";
const app = express();
dotenv.config();
app.use(express.json({ limit: "30mb", extended: true }));

```

```
app.use(express.urlencoded({ limit: "30mb", extended: true }));
app.use(cors());
app.get("/", (req, res) => {
  res.send("This is a Feedback loop");
});
app.use("/user", userRoutes);
app.use("/answer", answerRoutes);
app.use("/questions", questionRoutes);
const PORT = process.env.PORT || 5000;
const DATABASE_URL = process.env.DATABASE_URL;
mongoose.set("strictQuery", true);
mongoose.connect(DATABASE_URL, {useNewUrlParser:true,
useUnifiedTopology: true })
  .then(() =>
    app.listen(PORT, () => {
      console.log(`server running on port ${PORT}`);
    })
  )
  .catch((err) => console.log(err.message));
```


CHAPTER 5

TESTING

5.1 Unit Testing

This involves testing individual components or units of the system to ensure they function correctly in isolation. For *Feedback Loop*, unit testing might involve testing components that handle feedback submissions, user authentication, or the retrieval of user interaction data.

5.2 Integration Testing

Integration testing verifies that different components of the system work together as expected. In the context of *Feedback Loop*, integration testing might involve testing the interaction between the feedback submission module, the user profile display module, and the database.

5.3 Functional Testing

Functional testing examines whether the system meets the specified functional requirements. For *Feedback Loop*, this might involve testing whether the feedback is submitted, categorized, and displayed correctly, and whether users can interact with feedback seamlessly.

5.4 Regression Testing

Regression testing ensures that recent code changes have not adversely affected existing functionalities. It involves re-testing existing features of the system after changes have been made. In *Feedback Loop*, regression testing would ensure that updates to feedback categorization or user interface enhancements do not disrupt other functionalities like authentication or data retrieval.

5.5 User Acceptance Testing (UAT)

UAT involves testing the system with real users to ensure that it meets their needs and expectations. In the case of *Feedback Loop*, UAT might involve gathering feedback from users about the usability of the feedback submission process and the overall system experience.

5.6 Performance Testing

Performance testing evaluates the system's responsiveness, scalability, and stability under various load conditions. For *Feedback Loop*, performance testing might involve measuring how quickly feedback is processed and displayed, especially during periods of high user activity.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Conclusion

The *Feedback Loop* project emerges as a transformative platform for facilitating meaningful interactions and knowledge sharing among users. By integrating cutting-edge technologies and user-centric design, it ensures a seamless and engaging experience for all participants.

The project offers several key benefits:

1. Effective Feedback Management:

With streamlined submission and categorization, *Feedback Loop* ensures that users can easily share and access feedback, enabling more constructive and actionable insights.

2. User-Centric Design:

A responsive and intuitive interface makes it effortless for users to navigate the system, interact with feedback, and track their contributions or responses.

3. Community Engagement:

Interactive features, such as discussions and responses to feedback, foster collaboration and promote active participation, creating a thriving ecosystem for idea exchange.

4. Continuous Improvement:

Regular updates and enhancements to the system ensure it adapts to evolving user needs, incorporating new features and refining existing functionalities for a better experience.

Future Outlook

1. Enhanced Feedback Analysis:

Advanced analytical tools can be integrated to extract deeper insights from user feedback, such as identifying trends, sentiments, and actionable recommendations, thereby improving the system's utility.

2. Integration with External Platforms:

Connecting *Feedback Loop* with external collaboration tools and social platforms can foster broader participation, enabling users to share feedback and ideas seamlessly across multiple channels.

3. Accessibility:

Incorporating features like screen reader compatibility, adjustable UI

settings, and multilingual support can ensure the platform remains inclusive and accessible to a diverse user base.

4. **Scalability:**

Expanding the platform to accommodate different industries or community groups can increase its impact, making it a versatile tool for feedback and collaboration in varied contexts.

In essence, *Feedback Loop* serves as an invaluable tool for fostering collaboration and communication. By continuously evolving with technological advancements and user needs, the platform is well-positioned to remain a reliable and innovative solution for years to come. Its future enhancements aim to empower users further, fostering a dynamic ecosystem of engagement and growth.

CHAPTER-7

BIBLIOGRAPHY

- <https://www.mindinventory.com/blog/mean-stack-vs-mern-stack/>
- <https://www.mongodb.com/mern-stack>
- <https://www.ijert.org/research/performance-optimization-using-mern-stack-on-web-application-IJERTV10IS060239.pdf>
- "MERN Stack for Web Development – A Comprehensive Guide" by K. Gopalan, V. Vijayakumar, and R. Vasantha. This paper was presented at the 2020 International Conference on Smart Electronics and Communication (ICOSEC)
- "A Secure and Efficient Authentication System Using JSON Web Tokens (JWT)" by A. Kumar, S. Sharma, and R. Singh. This paper discusses the design and implementation of a secure authentication system using JWT and was published in the IEEE Access journal
- "Implementing JWT in RESTful Web Services" by R. Gupta, A. Sharma, and N. Kumar. This paper provides a detailed guide on implementing JWT in RESTful web services and was published in the IEEE Journal on Web Services.
- MERN Stack Full Tutorial & Project by Dave Gray. This video provides a detailed guide on building a MERN stack application from scratch, including setting up the server, database, routes, models, and more
- Node.js and Express.js - Full Course by freeCodeCamp.org. This comprehensive course teaches you how to use Node.js and Express.js to build a complex REST API and a MERN stack application.
- <https://react.dev/>
- <https://developer.mozilla.org/en-US/docs/Glossary/Node.js?retiredLocale=hu&language=hu>
- <https://www.geeksforgeeks.org/>

