PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

BANGALORE

Report On

"StartUp Ayush Portal"

Team details

Sl. no.	Roll no.	Name
2.	20211CSG0021	Pawan P
3.	20211CSG0022	Rishith R Rai
5.	20211CSG0031	Ganashree P

School of Computer Science ,
Presidency University, Bengaluru

Under the Supervision of
Dr. Saravana Kumar (Associate Professor)

School of Computer Science and Engineering, Presidency University.

Abstract

The Startup AYUSH Portal is a comprehensive digital platform designed to foster innovation, collaboration, and entrepreneurship in the AYUSH (Ayurveda, Yoga & Naturopathy, Unani, Siddha, and Homeopathy) sector. This one-stop platform connects startups, investors, incubators, accelerators, government agencies, and the public, enabling seamless interaction in a dynamic ecosystem.

The portal provides a collaborative environment for networking, mentorship, funding opportunities, and resource sharing. Entrepreneurs in the AYUSH domain can showcase their innovations, access government schemes, and connect with key stakeholders globally. Additionally, the platform offers virtual events, educational resources, and incubation support, ensuring the sustainable growth of AYUSH-based startups.

By integrating technology with traditional healthcare systems, the Startup AYUSH Portal aims to accelerate the global expansion of AYUSH practices, promote wellness-based startups, and support research and development in the field. Through a user-friendly digital interface, it bridges the gap between innovation and investment, making AYUSH-based entrepreneurship more accessible and impactful.

Introduction

The Startup Ayush Portal is a cutting-edge, digital platform designed to revolutionise the Ayush(Ayurveda, Yoga & Naturopathy, Unani, Siddha, and Homeopathy) ecosystem by fostering innovation, collaboration, and growth. Built on the MERN stack (MongoDB, Express.js, React.js, and Node.js), this portal leverages modern web technologies to deliver a seamless, scalable, and user-friendly experience for all stakeholders, including startups, investors, incubators, accelerators, government agencies, and public users. The Ayush sector, with its deep roots in traditional medicine and wellness practices, has immense potential to contribute to global healthcare and economic development.

However, startups in these sectors often face unique challenges, such as limited access to funding, mentorship, regulatory guidance, and global markets. Existing platforms are either generic or lack the specialised features needed to address these challenges, creating a gap in the ecosystem. The Startup Ayush Portal addresses these gaps by providing a dedicated, one-stop platform tailored to the needs of the Ayush community. By harnessing the power of the MERN stack, the portal offers a robust, dynamic, and interactive environment where stakeholders can connect, collaborate, and thrive. Key features include stakeholder dashboards, a centralized resource repository, virtual networking tools, a global showcase platform, and data-driven analytics, all designed to empower Ayush startups in their entrepreneurial journey.

The use of the MERN stack ensures that the portal is not only highly performant and scalable but also equipped with modern UI/UX capabilities to enhance user engagement. MongoDB provides a flexible and scalable database solution, Express.js enables efficient backend development, React.js ensures a responsive and interactive frontend, and Node.js facilitates real-time communication and seamless integration of features.

By bringing together the entire Ayush startup community on a single platform, the Startup Ayush Portal aims to create a vibrant, interconnected ecosystem that drives innovation, supports collaboration, and unlocks the full potential of the Ayush sector. Whether it's connecting startups with investors, providing access to mentorship and resources, or showcasing innovations to a global audience, this portal poised become the cornerstone of the Ayush startup ecosystem, both in India and worldwide.

Algorithm Details

1.

Objective:

- The main goal of these routes is to develop a modular and structured RESTful API for the Startup AYUSH Portal, supporting user authentication, discussions, funding opportunities, mentorship connections, and profile management. These routes manage requests concerning user interactions, funding applications, mentorship searches, and user profile updates in a secure and efficient manner.

Key Components:

1. **User Authentication:**
    - Handles user registration and login using JWT tokens for secure access. Passwords are hashed before storing.
2. **Discussion Management:**
    - Fetches and displays discussion topics from the database for user engagement.
3. **Funding System:**
    - Allows users to view funding opportunities, apply for funding, and check their application status.
4. **Mentorship Program:**
    - Provides mentor listings, retrieves mentor details, and supports searching for mentors based on expertise.
5. **User Profile Management:**
    - Enables users to update their profiles, change passwords, and upload profile images with authentication.
6. **Middleware for Security:**
    - Ensures only authenticated users can access protected routes, verifying identities before processing requests.

7. **Database Integration:**
    - Stores and retrieves user details, discussions, funding applications, and mentor profiles efficiently.

Process:

- Users can access discussion topics and participate in conversations. When a request is made, the system fetches discussions from the database and returns them as a JSON response. This data is displayed on the frontend, allowing users to browse and engage in relevant discussions.
- Users can explore and apply for funding opportunities through the portal. When applying, they submit an application that is validated and stored in the database after checking eligibility. They can later track their funding status (Pending, Approved, or Rejected) via a request to the server.
- The mentorship system helps users find and connect with mentors. Users can request a list of mentors, search for specific ones by expertise or name, and retrieve mentor profiles. The system filters mentors based on search criteria and returns relevant results, making it easier to find guidance.
- Users can change their passwords and update profile details. When changing passwords, the system verifies the old password, hashes the new one, and updates it in the database. Users can also upload profile images using multer.js, and the updated information is securely stored.
- Authentication middleware (authMiddleware.js) ensures only authorized users can access certain features. The system verifies JWT tokens in every request, preventing unauthorized access. Database queries handle user data, discussions, funding applications, and mentor profiles, ensuring efficiency and security.

Source Code:

```
import express from "express";
import { register, login } from "../controllers/authController.js";

const router = express.Router();

router.post("/register", register);
router.post("/login", login);

export default router;
```

```
import express from "express";
import { getDiscussions } from "../controllers/discussion.js";

const router = express.Router();
router.get("/", getDiscussions);
export default router;
```

```
import express from "express";
import {
  getAllFunding,
  applyForFunding,
  getUserFundingStatus,
} from "../controllers/funding.js";

const router = express.Router();

// Route to get all available funding opportunities
router.get("/", getAllFunding);

// Route to apply for a funding opportunity
router.post("/apply", applyForFunding);

// Route to get user-specific funding application status
router.get("/status/:userId", getUserFundingStatus);

export default router;
```

```
import express from "express";
import {
  getAllMentors,
  getMentorById,
  searchMentors,
} from "../controllers/mentor.js";

const router = express.Router();

// Route to get all mentors
router.get("/", getAllMentors);
```

```
// Route to get a mentor by ID
router.get("/:id", getMentorById);

// Route to search mentors by name or expertise
router.get("/search", searchMentors);

export default router;
```

```
import express from "express";
import { changePassword, editProfile } from "../controllers/userController.js";
import authMiddleware from "../middlewares/authMiddleware.js";
import upload from "../middlewares/multer.js";

const router = express.Router();

// Change Password Route
router.put("/change-password", authMiddleware, changePassword);

// Edit Profile Route (with Image Upload)
router.put(
  "/edit-profile",
  authMiddleware,
  upload.single("image"),
  editProfile
);

export default router;
```

2.

Objective:

- The aim of these server-side models is to create a structured and efficient database system for the Startup AYUSH Portal so that it can manage users, discussions, funding applications, and mentors with ease. These models provide data consistency, integrity, and relationships between entities, so that users can register, engage in discussions, apply for funding, and be connected with mentors. Through precise schemas for each entity, the system provides safe storage of data, effective retrieval, and expandability, serving as a dynamic and interactive web platform for AYUSH entrepreneurs and stakeholders.

Key Components:

1. **User Model:**
    - The User model stores details like username, email, password, profile image, and bio while ensuring unique usernames and emails. A default profile image and bio are assigned to new users. It also uses timestamps to track account creation and updates.

2. **Discussion Model:**
    - The Discussion model manages discussion topics with title and description fields. It allows users to post and retrieve discussions, promoting engagement and knowledge sharing on the platform.

3. **Funding Model:**
    - The Funding model stores funding title, amount, and eligibility criteria, along with the application status (Pending, Approved, Rejected). Each application is linked to a specific user using a reference to the User model. Timestamps track when applications are submitted or updated.

4. **Mentor Model:**
    - The Mentor model stores details about mentors, their expertise, and contact information while ensuring unique mentor contacts. Timestamps track profile updates, helping manage mentor-mentee connections efficiently.

Process:

- When a new user registers, the system checks for unique usernames and emails before storing their credentials. The password is hashed for security, and a default profile image and bio are assigned if not provided. The system records the timestamp of user creation, allowing tracking of profile updates. When a user logs in, their email and password are verified before granting access.

- When a user creates a discussion, they submit a title and description, which are stored in the database. The system retrieves and displays these discussions

whenever requested, allowing users to browse and engage with different topics. The structured format ensures that discussions are easily searchable and accessible.

- When a funding opportunity is created, the system stores details such as title, amount, and eligibility criteria. Users can apply for funding, linking their application to their user ID. The system assigns a default “Pending” status, which can later be updated to “Approved” or “Rejected”. Timestamps track when applications are submitted or modified, ensuring accurate record-keeping.

- When a mentor is added, their name, expertise, and contact details are stored. The system ensures that each mentor’s contact information is unique to prevent duplication. Users can search for mentors based on their expertise, making it easier to find guidance. Timestamps track when a mentor profile is created or updated.

Source Code:

```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema(
  {
   username: { type: String, required: true, unique: true },
   email: { type: String, required: true, unique: true },
   password: { type: String, required: true },
   imageUrl: {
    type: String,
    default:

   "https://res.cloudinary.com/dkwxc1anx/image/upload/v1742282816/default_pr
   ofile_picture_birsvl.jpg",
   },
   bio: {
    type: String,
    default: "Hi I'm the most passionate founder you'll ever meet",
   },
  },
  { timestamps: true }
);

const User = mongoose.model("User", userSchema);
export default User;
import mongoose from "mongoose";
```

```
const discussionSchema = new mongoose.Schema({
  title: { type: String, required: true },
  description: { type: String, required: true },
});

const Discussion = mongoose.model("Discussion", discussionSchema);

export default Discussion;
import mongoose from "mongoose";

const fundingSchema = new mongoose.Schema(
  {
    title: { type: String, required: true },
    amount: { type: String, required: true },
    eligibility: { type: String, required: true },
    status: {
      type: String,
      enum: ["Pending", "Approved", "Rejected"],
      default: "Pending",
    },
        applicant: { type: mongoose.Schema.Types.ObjectId, ref: "User" }, // Assuming a User model exists
  },
  { timestamps: true }
);

const Funding = mongoose.model("Funding", fundingSchema);
export default Funding;
import mongoose from "mongoose";

const mentorSchema = new mongoose.Schema(
  {
    name: { type: String, required: true },
    expertise: { type: String, required: true },
    contact: { type: String, required: true, unique: true },
  },
  { timestamps: true }
);

const Mentor = mongoose.model("Mentor", mentorSchema);
export default Mentor;
```

3.

Objective:

- The server-side controllers are responsible for the core operations of the Startup AYUSH Portal, providing secure and efficient user interactions. The controllers are responsible for user authentication (registration, login, password management), retrieval of discussions, funding requests, mentor management, and updating the user profile. By using MongoDB for data storage, bcrypt for password hashing, JWT for authentication, and Cloudinary for image uploads, the system provides data security, scalability, and seamless user experience. These controllers are intermediaries between frontend and database, processing the requests and returning proper responses, rendering the platform dynamic and interactive.

Key Components:

1. **User Authentication & Security:**
    - The system manages user registration, login, and password security using bcrypt to hash passwords and JWT to generate secure authentication tokens. This ensures that user credentials are safely stored and protected while allowing secure access to the platform.
2. **Discussion Management:**
    - The discussion controller is responsible for retrieving and displaying discussions from the database. It allows users to view ongoing conversations and interact with relevant topics, ensuring seamless engagement within the community.
3. **Funding Application Handling:**
    - Users can browse available funding opportunities, submit applications, and track their application status. The system maintains an organized structure by linking applications to user accounts, ensuring transparency and accessibility.

4. **Mentor Management:**
    - The mentor controller helps users find, search, and connect with industry experts. Users can browse the list of available mentors, filter them by expertise, and retrieve detailed mentor information, facilitating networking and guidance.
5. **Profile Management & Image Uploads:**
    - Users can update their profile details, including username, email, bio, and profile pictures. The system integrates Cloudinary for secure image uploads, allowing users to personalize their profiles and maintain a professional online presence.
6. **Error Handling & Response Management:**
    - The controllers include robust error handling mechanisms, ensuring that errors are properly managed and meaningful responses are sent to users. This improves system stability and user experience by preventing crashes and providing clear feedback.

Process:

- When a user registers, the system first checks if the provided email is already in use. If the email is unique, the password is securely hashed using bcrypt before being stored in the database. Upon successful registration, a confirmation response is sent. During login, the system verifies the credentials, and if correct, generates a JWT token for secure authentication.
- The discussion controller retrieves all discussion topics from the database when a request is made. This ensures that users can access and participate in ongoing discussions within the platform. If an error occurs during data retrieval, the system responds with an error message. This helps maintain a smooth and engaging interaction for users.
- Users can view available funding opportunities retrieved from the database. When a user applies, the system validates the request and creates a new funding record linked to the applicant, setting the status to “Pending.” Users can also check their application status, and the system fetches funding records

based on their user ID. This ensures transparency in the funding process and allows users to track their applications.

- The system retrieves a list of available mentors from the database, allowing users to connect with experts. Users can search for mentors by name or expertise using a filtering mechanism. They can also view detailed mentor profiles using their unique ID. This feature helps startups gain valuable guidance from industry professionals.

- Users can update their profile information, such as username, email, and bio. If a profile picture is uploaded, it is processed and stored securely using Cloudinary. After a successful update, the system returns the modified user profile data as confirmation. This ensures users can maintain a professional and updated presence on the platform.

- When a user changes their password, the system verifies their identity for security. The new password is then hashed using bcrypt before being updated in the database. Once the update is successful, a response is sent confirming the change. This helps enhance security and prevents unauthorized access to user accounts.

- Each controller includes error-handling mechanisms to catch potential issues. If an error occurs, a detailed response is sent to inform users about the problem. This helps prevent crashes and ensures a smooth user experience. Effective error handling keeps the application stable and responsive.

Source Code:

```
import User from "../models/User.js";
import bcrypt from "bcryptjs";
import jwt from "jsonwebtoken";
const register = async (req, res) => {
  try {
    const { username, email, password } = req.body;
    // Check if user already exists
    const existingUser = await User.findOne({ email });
```

```
    if (existingUser)
      return res.status(400).json({ message: "Email already in use" });
    // Hash the password
    const hashedPassword = await bcrypt.hash(password, 10);
    // Create new user
    const newUser = new User({ username, email, password: hashedPassword });
    await newUser.save();
    res.status(201).json({ message: "User registered successfully" });
  } catch (error) {
    res.status(500).json({ message: "Server error", error });
  }};
const login = async (req, res) => {
  try {
    const { email, password } = req.body;
    // Find user by email
    const user = await User.findOne({ email });
    if (!user) return res.status(400).json({ message: "Invalid credentials" });
    // Compare passwords
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch)
      return res.status(400).json({ message: "Invalid credentials" });
    // Generate JWT
    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
      expiresIn: "1h",
    });
    res.status(200).json({
      token,
      user: { id: user._id, username: user.username, email: user.email },
    });
  } catch (error) {
    res.status(500).json({ message: "Server error", error });
  }};
export { register, login };
```

```
import Discussion from "../models/discussion.js";
export const getDiscussions = async (req, res) => {
  try {
    const discussions = await Discussion.find();
    res.json(discussions);
  } catch (error) {
    res.status(500).json({ message: "Server error" });
  }};
import Funding from "../models/funding.js";
// Get all funding opportunities
export const getAllFunding = async (req, res) => {
  try {
    const fundings = await Funding.find();
    res.status(200).json(fundings);
  } catch (error) {
    res
      .status(500)
      .json({ message: "Error fetching funding opportunities", error });
  }};
// Apply for a funding opportunity
export const applyForFunding = async (req, res) => {
  try {
    const { title, applicantId } = req.body;
    const funding = await Funding.findOne({ title });
    if (!funding) {
      return res.status(404).json({ message: "Funding opportunity not found" });
    }
    // Create new funding application
    const newApplication = new Funding({
      title: funding.title,
      amount: funding.amount,
      eligibility: funding.eligibility,
      applicant: applicantId,
```

```
      status: "Pending",
    });
    await newApplication.save();
    res
      .status(201)
      .json({ message: "Application submitted successfully", newApplication });
  } catch (error) {
    res.status(500).json({ message: "Error applying for funding", error });
  }};
// Get funding applications by user
export const getUserFundingStatus = async (req, res) => {
  try {
    const { userId } = req.params;
    const applications = await Funding.find({ applicant: userId });
    if (!applications.length) {
      return res
        .status(404)
        .json({ message: "No funding applications found for this user" });
    }
    res.status(200).json(applications);
  } catch (error) {
    res
      .status(500)
      .json({ message: "Error fetching application status", error });
  }};
import Mentor from "../models/mentor.js";
// Get all mentors
export const getAllMentors = async (req, res) => {
  try {
    const mentors = await Mentor.find();
    res.status(200).json(mentors);
  } catch (error) {
    res.status(500).json({ message: "Error fetching mentors", error });
```

```
  }};
// Get a specific mentor by ID
export const getMentorById = async (req, res) => {
  try {
    const { id } = req.params;
    const mentor = await Mentor.findById(id);
    if (!mentor) {
      return res.status(404).json({ message: "Mentor not found" });
    }
    res.status(200).json(mentor);
  } catch (error) {
    res.status(500).json({ message: "Error fetching mentor", error });
  }};
// Search mentors by name or expertise
export const searchMentors = async (req, res) => {
  try {
    const { query } = req.query;
    const mentors = await Mentor.find({
      $or: [
        { name: { $regex: query, $options: "i" } },
        { expertise: { $regex: query, $options: "i" } },
      ], });
    res.status(200).json(mentors);
  } catch (error) {
    res.status(500).json({ message: "Error searching mentors", error });
  }};
import User from "../models/User.js";
import cloudinary from "../config/cloudinary.js";
import bcrypt from "bcryptjs";
//change password
export const changePassword = async (req, res) => {
  try {
    const newPassword = req.body;
```

```
    const user = await User.findById(req.user.id);
    if (!user) return res.status(404).json({ message: "User not found" });
    // Hash new password and save
    const salt = await bcrypt.genSalt(10);
    user.password = await bcrypt.hash(newPassword.toString(), salt);
    await user.save();
    res.status(200).json({ message: "Password changed successfully" });
  } catch (error) {
    console.log(error);
    res.status(500).json({ message: "Server error", error: error.message });
  }};
// Edit Profile (Supports Image Upload)
export const editProfile = async (req, res) => {
  try {
    const { username, email, bio } = req.body;
    let imageUrl;
    // Upload image to Cloudinary if provided
    if (req.file) {
      const uploadResult = await cloudinary.uploader.upload_stream(
        { folder: "profile_pictures" }, // Cloudinary folder
        async (error, result) => {
          if (error)
            return res
              .status(500)
              .json({ message: "Image upload failed", error });
          imageUrl = result.secure_url;
          // Update user profile
          const updatedFields = {};
          if (username) updatedFields.username = username;
          if (email) updatedFields.email = email;
          if (bio) updatedFields.bio = bio;
          if (imageUrl) updatedFields.imageUrl = imageUrl;
          const user = await User.findByIdAndUpdate(
```

```
          req.user.id,
          updatedFields,
          { new: true,
            runValidators: true,
          });
        if (!user) return res.status(404).json({ message: "User not found" });
        res
          .status(200)
          .json({ message: "Profile updated successfully", user });
      });
    uploadResult.end(req.file.buffer); // Pass file buffer to Cloudinary
  } else {
    // If no image, update profile without uploading
    const updatedFields = { username, email, bio };
    const user = await User.findByIdAndUpdate(req.user.id, updatedFields, {
      new: true,
      runValidators: true,
    });
    if (!user) return res.status(404).json({ message: "User not found" });
    res.status(200).json({ message: "Profile updated successfully", user });
  }} catch (error) {
  res.status(500).json({ message: "Server error", error: error.message });
}};
```

4.

Objective:

- The objective of this client-side React.js application is to create an interactive and user-friendly platform for AYUSH startups to connect with mentors, investors, and funding opportunities. It provides essential features such as user authentication (login/register), funding application tracking, mentorship search, and community discussions to support startups in their growth journey. The app includes a collapsible sidebar for easy navigation, dynamic filtering for funding applications, and a search functionality for mentors. Built with React Router and Tailwind CSS, it ensures a responsive, modern, and seamless user experience for startup founders and entrepreneurs.

Key Components:

1. **Homepage:**
    - The homepage introduces the platform, highlighting its purpose and benefits. It features a hero section, key features, and testimonials to engage users and encourage exploration.
2. **Sidebar:**
    - A collapsible navigation menu provides quick access to sections like Dashboard, Funding, Mentorship, and Community. Users can toggle it for better navigation, especially on smaller screens.
3. **Page Layout:**
    - A reusable layout ensures consistency across pages, integrating the sidebar, title, and content area. This structure simplifies maintenance and enhances the user experience.
4. **Funding Applications:**
    - Users can track funding applications with statuses like Pending, Approved, or Rejected. A filtering system helps them organize and review applications efficiently.

5. **Community Discussions:**
    - A space for startups to engage, ask questions, and share knowledge. Users can browse recent discussions on topics like funding, networking, and growth strategies.
6. **Mentorship Search:**
    - Users can search for mentors by name or expertise and contact them directly for guidance. This feature supports networking and knowledge-sharing for startups.
7. **User Authentication:**
    - A simple system for login and registration, allowing users to create accounts and access personalized resources like funding and mentorship.
8. **State Management:**
    - Various interactive features use state management, including sidebar toggling, search input, and authentication mode switching, making the platform dynamic and user-friendly.

Process:

- When users visit the platform, they see the homepage with an overview, key features, and testimonials. The sidebar navigation helps them move between sections like Dashboard, Funding, Mentorship, and Community. They can toggle the sidebar open or closed for better accessibility.
- New users can register by providing their details, while existing users can log in to access personalized features. Once logged in, users can explore funding opportunities and track their applications. The funding page shows applications as Pending, Approved, or Rejected, with filters for easy tracking.
- In the community section, users can join discussions, ask questions, and share insights. The mentorship search feature helps users find experts, supporting networking and guidance. This allows startups to connect with the right people for advice and growth.

- The platform offers interactive features for a smooth experience. Users can toggle the sidebar, filter applications, search mentors, and switch between login/register modes. These features use state management for better usability. When finished, users can log out using the sidebar button, ensuring security.

Source Code:

```
import { useState } from "react";
import { Link } from "react-router-dom";
import { Menu, X, LogOut, User, DollarSign, Users, MessageSquare, Briefcase,
FileText, CheckCircle, XCircle, Clock, Filter, Key } from "lucide-react";

const Sidebar = ({ isOpen, toggle }) => (
  <aside className={`fixed inset-y-0 left-0 bg-[#736F72] text-white p-6
transition-all ${isOpen ? "w-64" : "w-16"}`}>
    <button onClick={toggle} className="absolute top-4 right-[-40px] bg-[#736F72]
p-2 rounded-full shadow-md md:block hidden">{isOpen ? <X /> : <Menu
/>}</button>
    {[['/dashboard', <User />, "Dashboard"], ['/funding', <DollarSign />, "Funding"],
['/mentors', <Users />, "Mentorship"], ['/community', <MessageSquare />,
"Community"], ['/fundingsub', <Briefcase />, "Funds & Status"], ['/applications',
<FileText />, "Applications"]].map(([to, icon, label]) => (
      <Link key={to} to={to} className="flex items-center space-x-2
hover:text-[#C3BABA] transition">{icon} {isOpen &&
<span>{label}</span>}</Link>
    ))}
    <button className="flex items-center gap-2 text-white hover:text-[#C3BABA]
transition mt-auto"><LogOut size={20} />{isOpen && <span>Log
Out</span>}</button>
  </aside>
);

const Page = ({ title, children }) => {
  const [isSidebarOpen, setSidebar] = useState(false);
  return (
    <div className="min-h-screen flex bg-[#E9E3E6]">
```

```
    <Sidebar isOpen={isSidebarOpen} toggle={() => setSidebar(!isSidebarOpen)} />
    <main className="flex-1 p-8">
     <button onClick={() => setSidebar(true)} className="md:hidden
text-[#736F72] text-3xl"><Menu /></button>
     <div className="bg-[#C3BABA] p-6 rounded-xl shadow-md text-white
mt-4">
      <h1 className="text-3xl font-bold">{title}</h1>
      {children}
     </div>
    </main>
   </div>
 );};

const Applications = () => {
 const [filter, setFilter] = useState("All"), apps = [{ id: 1, title: "Startup Seed Grant",
status: "Pending" }, { id: 2, title: "Growth Accelerator Fund", status: "Approved" }, {
id: 3, title: "Innovation Challenge Grant", status: "Rejected" }];
 return (
  <Page title="Funding Applications">
   <div className="mt-4 flex items-center space-x-4"><Filter
className="text-white" />
    {["All", "Pending", "Approved", "Rejected"].map(s => <button key={s}
onClick={() => setFilter(s)} className={`px-3 py-1 rounded-lg ${filter === s ?
"bg-[#736F72] text-white" : "bg-gray-200 text-[#736F72]"}`}>{s}</button>)}
   </div>
   <div className="mt-6 space-y-4">
    {apps.filter(app => filter === "All" || app.status === filter).map(({ id, title,
status }) => (
     <div key={id} className="bg-[#E9E3E6] p-4 rounded-lg flex justify-between
items-center shadow">
      <h2 className="text-lg font-semibold text-[#736F72]">{title}</h2>
      <span className="flex items-center gap-2 text-[#9A8F97]">{status ===
"Approved" ? <CheckCircle className="text-green-500" /> : status === "Rejected"
```

```
? <XCircle className="text-red-500" /> : <Clock className="text-yellow-500"
/>}{status}</span>
          </div>
        ))}
      </div>
    </Page>
  );};

const Community = () => (
  <Page title="Community Discussions">
    <div className="mt-8 bg-white p-6 rounded-xl shadow-md">
      <h2 className="text-xl font-semibold text-[#736F72]">Recent
Discussions</h2>
      <ul className="mt-4 space-y-3 text-[#9A8F97]">{["💬 How to secure
funding?", "🔍 Scaling a tech company", "🚀 Networking strategies", "📈 Market
research techniques", "🛠️ Tools for startups"].map((d, i) => <li
key={i}>{d}</li>)}</ul>
    </div>
  </Page>
);

const mentors = [{ name: "Dr. Ayesha Patel", expertise: "Business Strategy", contact:
"ayesha@email.com" }, { name: "John Carter", expertise: "Marketing & Growth",
contact: "john@email.com" }, { name: "Sophia Lin", expertise: "Product
Development", contact: "sophia@email.com" }, { name: "Raj Mehta", expertise:
"Funding & Investment", contact: "raj@email.com" }];

const App = () => {
  const [isSidebarOpen, setSidebar] = useState(false), [isLogin, setLogin] =
useState(true), [searchTerm, setSearch] = useState(""), [profile] = useState({ name:
"John Doe", email: "johndoe@example.com", bio: "Entrepreneur", profilePic:
"https://via.placeholder.com/100" });
  const filteredMentors = mentors.filter(({ name, expertise }) =>
name.toLowerCase().includes(searchTerm.toLowerCase()) ||
expertise.toLowerCase().includes(searchTerm.toLowerCase()));
```

```
  return (
    <div className="min-h-screen flex bg-[#E9E3E6]">
      <Sidebar isOpen={isSidebarOpen} toggle={() => setSidebar(!isSidebarOpen)} />
      <main className="flex-1 p-8">
        <button onClick={() => setSidebar(true)} className="md:hidden
text-[#736F72] text-3xl"><Menu /></button>
        <h1 className="text-3xl font-bold text-[#736F72]">{isLogin ? "Login" :
"Register"}</h1>
        <form className="space-y-4 bg-white p-8 shadow-lg rounded-lg">
          {!isLogin && <input type="text" placeholder="Full Name"
className="input" />}
          <input type="email" placeholder="Email" className="input" />
          <input type="password" placeholder="Password" className="input" />
          <button className="w-full py-2 text-white bg-[#736F72] rounded-lg
hover:bg-[#9A8F97]">{isLogin ? "Login" : "Register"}</button>
        </form>
        <p className="text-center text-[#736F72]">{isLogin ? "Don't have an
account?" : "Already have an account?"} <button onClick={() =>
setLogin(!isLogin)} className="text-[#9A8F97] underline">{isLogin ? "Sign up" :
"Login"}</button></p>
        <h2 className="text-2xl font-bold text-[#736F72] mt-8">Find a Mentor</h2>
        <input type="text" placeholder="Search mentors..." className="mt-6 p-3
rounded-lg shadow-md w-full md:w-1/2" value={searchTerm} onChange={e =>
setSearch(e.target.value)} />
        <div className="mt-8 grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3
gap-6">{filteredMentors.map(({ name, expertise, contact }, i) => (
          <div key={i} className="bg-white p-6 rounded-lg shadow-md"><User
className="text-[#736F72] w-10 h-10" /><h2 className="text-xl font-semibold
text-[#736F72]">{name}</h2><p className="text-[#9A8F97]">{expertise}</p><a
href={`mailto:${contact}`}className="text-[#C3BABA]
hover:text-[#9A8F97]">Contact Mentor</a></div>
        ))}</div>
      </main>
```

```
    </div>
  );};

export { Applications, Community };
export default App;
```

Conclusion

This project followed a structured four-phase timeline to ensure efficient research, development, and documentation. Beginning with title finalization and literature survey, the project progressed through algorithm design, partial implementation, and full integration before reaching the final phase of report submission and viva preparation. Each phase was planned with clear objectives and milestones, allowing systematic progress.

By adhering to the set timeline, the project achieved complete implementation, rigorous testing, and well-documented reporting within the scheduled timeframe. The final outcome is a functional system, supported by a comprehensive research report, ready for presentation and potential publication.

Overall, this structured approach not only ensured timely completion but also enhanced problem-solving skills, research capabilities, and technical expertise. The project serves as a valuable learning experience, reinforcing the importance of proper planning, execution, and documentation in software development and academic research.

Project Timeline

The project will be executed in four phases to ensure systematic progress toward final implementation and report submission.

Phase 1: Research & Planning (Feb 1 - Feb 15)
This phase involves finalizing the title, conducting a literature survey, defining the problem statement, and planning the methodology. The abstract and initial project timeline will be prepared.

Phase 2: Algorithm Design & Partial Implementation (Feb 16 - Mar 10)
The core algorithms and system architecture will be designed, followed by the development of 50% of the implementation. Initial testing and debugging will ensure smooth progress.

Phase 3: Full Implementation & Report Drafting (Mar 11 - Mar 31)
This phase includes completing the remaining implementation, integration testing, and performance optimization. The first draft of the project report will be prepared and reviewed.

Phase 4: Report Submission & Viva Preparation (Apr 1 - Apr 15)
The final project report will be submitted, followed by mock viva preparation. The final presentation will be created, with potential for publications if applicable.

Key Milestones:
Feb 15: Objectives & research completed
Mar 10: 50% implementation done
Mar 31: Full implementation & report draft ready
Apr 15: Final report submission & viva preparation

References

1. A. Gawer, "Bridging differing perspectives on technological platforms," Strategic Management Journal, vol. 35, no. 10, pp. 1319–1337, 2014.

2. E. Stam, "Entrepreneurial ecosystems and regional policy," Research Policy, vol.44, no. 4, pp. 805–816, 2015.

3. Ministry of AYUSH, Government of India, "AYUSH Sector: Opportunities andChallenges," 2021.

4. R. Kumar and A. Sharma, "Entrepreneurship in the AYUSH sector: A study of challenges and opportunities," Journal of Ayurveda and Integrative Medicine, vol.9, no. 4, pp. 308–314, 2018.

5. P. Thakkar, "Building scalable web applications with the MERN stack," Journal of Web Development, vol. 5, no. 2, pp. 45–60, 2020.

6. NITI Aayog, "Strengthening the startup ecosystem in India," 2018.

7. B. Patwardhan, D. Warude, P. Pushpangadan, and N. Bhatt, "Ayurveda and traditional Chinese medicine: A comparative overview," Evidence-Based Complementary and Alternative Medicine, vol. 2, no. 4, pp. 465–473, 2005.

8. F. Provost and T. Fawcett, Data Science for Business. O'Reilly Media, 2013.

9. S. M. Hackett and D. M. Dilts, "A systematic review of business incubation research," The Journal of Technology Transfer, vol. 29, no. 1, pp. 55–82, 2004.

10.K. Chodorow, MongoDB: The Definitive Guide. O'Reilly Media, 2013.