Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Department of Computer Science and Engineering

UCS2611 – Internet Programming Lab

Mini Project Report

CAMS - Consultancy Management System

Tharun Senthuravel - 3122 22 5001 150

Vineeth U - 3122 22 5001 160

Vishal K - 3122 22 5001 161

1. Problem Statement and Objective

The aim of this project is to develop **CAMS(Consultancy Management System)**, a web-based platform that facilitates the acquisition and management of consultancy project data. The existing process of handling such information is often manual, inefficient, and scattered across various sources. This leads to difficulties in organizing, retrieving, and communicating consultancy-related information, placing a considerable administrative burden on faculty and coordinators.

CAMS(Consultancy Management System) is designed to address these challenges by providing a centralized and secure interface where faculty can input, update, and manage consultancy project details. The platform includes key features such as:

- Secure management of consultancy data
- Timely automated notifications
- Exportable, filtered reports
- Tools for effective tracking of project progress and billing documentation

This solution aims to streamline project handling, improve transparency, reduce manual workload, and ensure more effective monitoring and documentation of consultancy activities.

2. Introduction

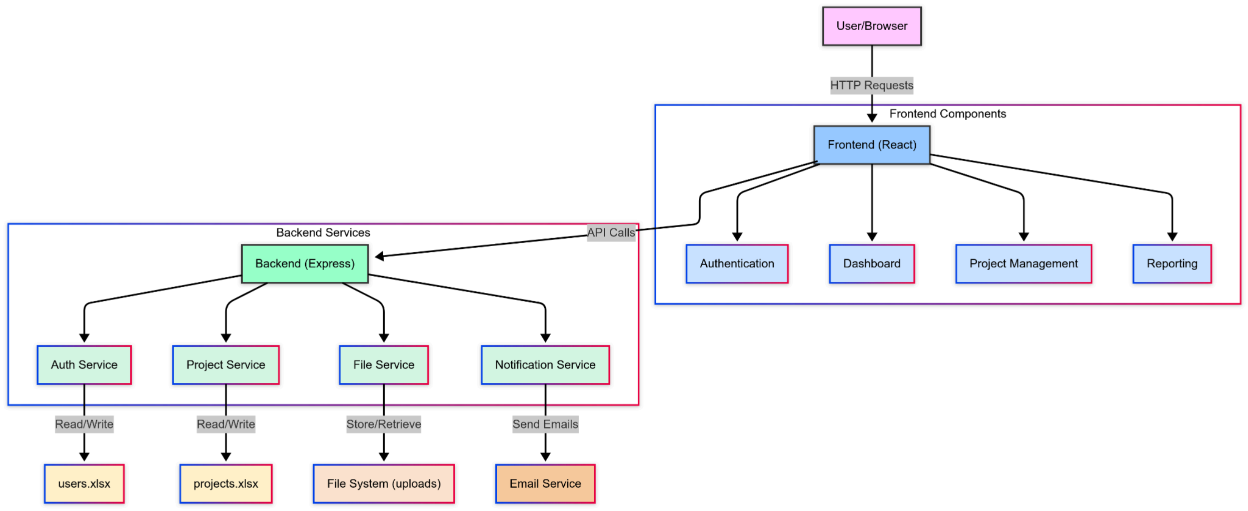
CAMS (Consultancy Management System) is a comprehensive web-based application developed to streamline the management of consultancy projects within academic institutions. Consultancy projects often involve collaboration between faculty members and industry partners, generating valuable opportunities for knowledge exchange, research, and institutional development. However, managing these projects manually can lead to inefficiencies, disorganization, and communication gaps.

CAMS addresses these challenges by providing a centralized and user-friendly platform that enables faculty to effectively track, manage, and document their consultancy engagements. The system supports the entire lifecycle of a consultancy project—from initiation and progress monitoring to final reporting and billing. Faculty members can securely log in to input project details, update progress, and access real-time information on ongoing activities.

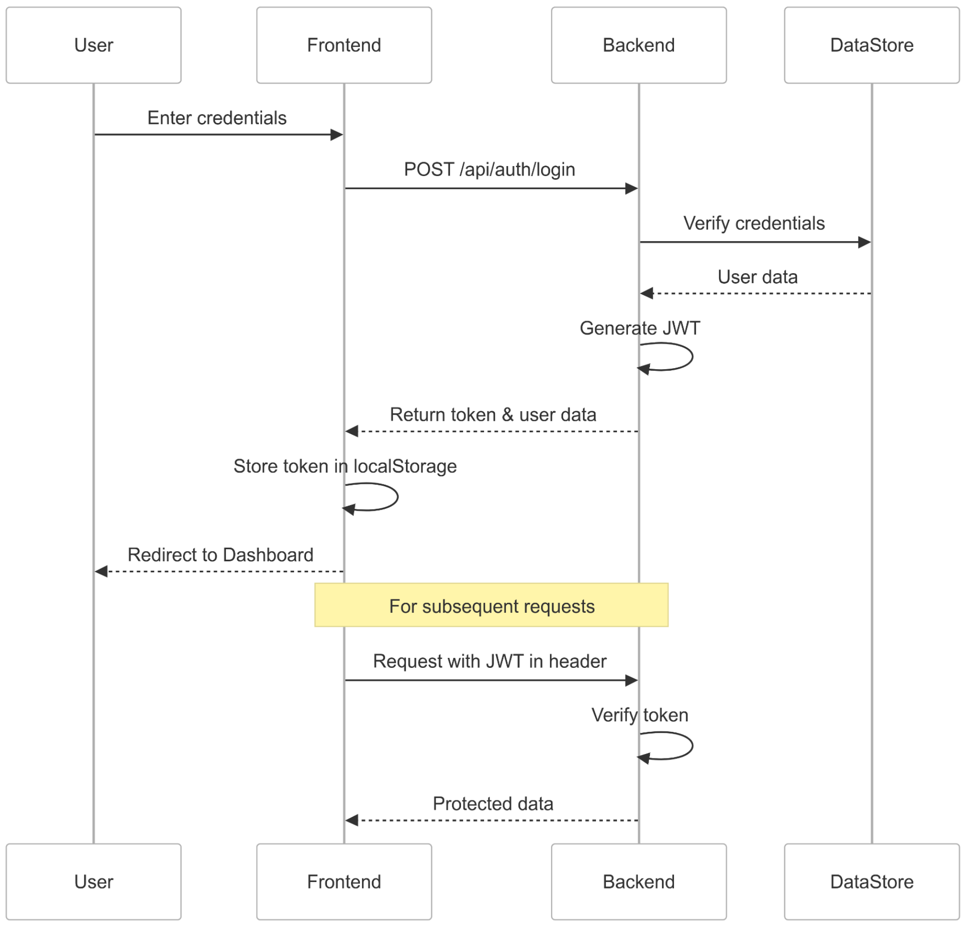
Additionally, CAMS incorporates features such as automated notifications, document uploads, and the ability to export filtered data reports. These capabilities reduce the administrative burden on faculty and coordinators, improve data accuracy, and enhance overall project transparency. By centralizing all consultancy-related activities, CAMS ensures that academic institutions can better manage their external collaborations while maintaining organized records for institutional reporting and audits.

Ultimately, CAMS empowers academic institutions to foster stronger partnerships with industry, streamline internal processes, and promote a more professional approach to consultancy project management.

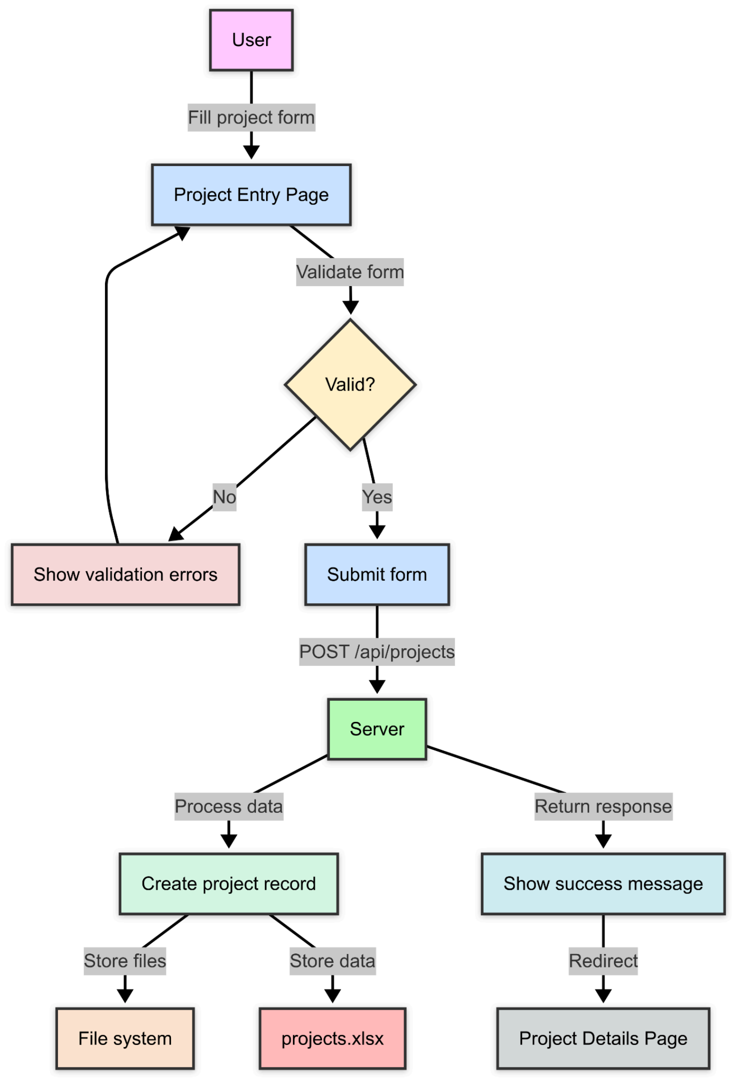
CAMS System Architecture



User Authentication Flow



Project Creation Flow



---

3. Requirements

Software Requirements

Frontend:

- HTML, CSS, JavaScript
- React.js
- Tailwind CSS (for styling)
- React Router (for navigation)
- Axios (for API requests)
- React Hot Toast (for notifications)

Backend:

- Node.js with Express.js
- JSON Web Tokens (JWT) for authentication
- Multer (for handling file uploads)
- XLSX (for Excel file generation and data storage)
- Nodemailer (for sending email notifications)
- Node-cron (for scheduled tasks)

Storage:

- Excel sheets (.xlsx) for storing and retrieving data

Hardware Requirements

- A system with a modern web browser (Chrome, Firefox, etc.)

- Node.js (v14 or higher) installed
- Stable internet connection for testing email notifications

Installation and Setup Instructions

Prerequisites

- Node.js (v14 or higher)
- npm or yarn package manager

Steps to Setup the Project

1. **Clone the Repository**

- git clone https://github.com/yourusername/consultease.git
- cd consultease

2. **Install Backend Dependencies**

- cd server
- npm install

3. **Install Frontend Dependencies**

- cd ../client
- npm install

4. **Configure Environment Variables**

   Create a .env file in the server directory with the following contents:

- PORT=5000

- JWT_SECRET=your_jwt_secret_key
- EMAIL_USER=your_email@gmail.com
- EMAIL_PASS=your_email_password

5. **Start Development Servers**

Backend:

- cd server
- npm run dev

Frontend:

- cd ../client
- npm start

6. **Access the Application**

- Frontend: http://localhost:3000

- Backend API: http://localhost:5000

---

4. Code Flow Diagram

CAMS

```
├── client               # Frontend React application
│   ├── public           # Public assets
│   └── src
│       ├── components      # Reusable UI components
│       ├── context       # React context providers
│       ├── pages         # Page components
│       └── App.js          # Main application component
└── server                  # Backend Node.js application
    ├── data               # Data storage (Excel files)
    ├── uploads              # Uploaded files storage
    ├── corsMiddleware.js    # CORS configuration
    └── server.js            # Main server file
```

---

5. Functionality with Explanation

User Authentication

The system features a secure login and registration mechanism tailored specifically for faculty members, using their college email addresses. This ensures that only authorized personnel can access and manage consultancy projects, maintaining the confidentiality and integrity of sensitive project data. The authentication process leverages best practices such as password hashing and JSON Web Tokens (JWT) for secure session management.

Dashboard Overview

The dashboard provides a visual summary of key project statistics and recent activities. It allows faculty members to quickly view important metrics such as the number of active consultancy projects, project milestones, and upcoming deadlines. The dashboard serves as a central hub for navigating through the system and ensures that users are always informed about the status of their projects.

Project Management

CAMS allows faculty members to create, view, and manage consultancy projects with ease. Users can input essential project details, track ongoing progress, and update project statuses. The system provides an intuitive interface for managing both ongoing and completed projects, streamlining workflows and ensuring all project information is centralized and easily accessible.

Financial Tracking

The platform includes robust financial tracking features to monitor the financial health of each consultancy project. Faculty members can track sanctioned amounts, monitor received payments, and keep an eye on outstanding balances. This ensures that all financial aspects of the project are properly documented and accounted for, reducing the risk of errors or missed payments.

Document Management

CAMS facilitates the uploading and secure storage of essential project documents. Faculty members can upload project agreements, contracts, and proof of bill settlements, ensuring that all necessary documentation is easily accessible and well-organized. The system provides a simple interface to upload, store, and retrieve these documents, enhancing transparency and efficiency in managing project paperwork.

Reporting

The system allows faculty members to generate Excel-based reports with customizable filters, enabling easy extraction of key data from the consultancy projects. These reports can be tailored to include specific time periods, project statuses, or financial details, making it easy to analyze trends and assess project performance. The ability to export data ensures that reports can be shared with other stakeholders or used for official documentation.

Notification System

CAMS integrates an automated email notification system that keeps faculty members informed about project milestones and important updates. Notifications are triggered for various events, such as the creation of a new project or when a project is nearing its completion. This helps ensure that faculty members are always aware of project deadlines, upcoming tasks, and critical actions that require attention.

5. Usage

User Registration and Login

To begin using the system, register with a valid college email address (ending with .edu or .ac.in). After registration, you can log in using your credentials. A test account has been automatically created with the following credentials for demonstration purposes:

- **Email**: test@college.edu
- **Password**: password123

Authentication Flow

The authentication system works as follows:

1. User registers or logs in with credentials

2. Server validates credentials and generates JWT token

3. Token is returned to client and stored in localStorage

4. Token is included in Authorization header for subsequent requests

5. Server middleware validates token for protected routes

6. On token expiration or logout, token is removed from localStorage

Managing Projects

Creating a New Project

To create a new project, navigate to the "New Project" section from the dashboard or the navigation menu. Once there, fill in all required project details, such as the industry name, project title, and financial information. Additionally, you will be required to upload necessary documents, with the project agreement document being mandatory. After completing the form, click on the "Submit" button to create and store the project.

Viewing Projects

To view your projects, go to the "Projects" page, where you will see a list of all the projects you have created or are managing. You can use various filters to narrow down the list by academic year, project amount, faculty name, or industry. By clicking on any project from the list, you will be able to view its full details, including progress, financials, and attached documents.

Downloading Reports

You can generate and download project reports from the "Projects" page. Simply apply the desired filters to narrow down the projects you wish to include in the report. Once the filters are applied, click the "Download Excel" button to generate an Excel report of the filtered data, which can be saved and shared as needed.

---

6. API Endpoints

Authentication

- **POST /api/auth/register**
  Register a new user
- **POST /api/auth/login**
  Login and get authentication token
- **GET /api/auth/verify**
  Verify authentication token

Projects

- **POST /api/projects**
  Create a new project

- **GET /api/projects**
  Get all projects (with optional filters)

- **GET /api/projects/:id**
  Get a specific project by ID

- **GET /api/projects/recent**
  Get recent projects

- **GET /api/projects/stats**
  Get project statistics

- **GET /api/projects/:id/download/:fileType**
  Download project documents

- **GET /api/projects/download**
  Download projects as Excel file

Data Storage

ConsultEase uses Excel files for data storage:

- **users.xlsx** – Stores user information
- **projects.xlsx** – Stores project details

Uploaded files are stored in the **uploads** directory.

7. User Roles and Workflow

CAMS currently operates without strict role-based access but supports two primary user types:

- Faculty Members: Main users who create, manage, and track their consultancy projects.
- Administrators: Have access to all project data for monitoring and oversight (future role expansion planned).

Typical Workflow

1. Faculty registers using their institutional email.
2. After login, they access a personalized dashboard.
3. They create and manage projects with detailed inputs.
4. Users can filter, view, and search through projects.
5. Documents can be downloaded, and reports exported.
6. Notifications are sent for deadlines and updates.

8. Deployment Considerations

To prepare CAMS for production, several upgrades are recommended:

1. Database Migration

Switch from Excel files to a database like MongoDB or PostgreSQL to ensure scalability, integrity, and advanced querying.

2. Authentication Enhancements

Add refresh tokens, password reset, and 2FA for better security and user control.

3. File Storage

Use cloud storage (e.g., AWS S3) for better scalability and implement virus scanning for uploads.

4. Security Hardening

Enable HTTPS, add rate limiting, sanitize inputs, and perform regular security audits.

5. Performance Optimization

Use caching, optimize APIs, and consider server-side rendering for faster frontend performance.

---

9. Potential Future Enhancements

1. Advanced Role Management

Introduce roles like Department Head, Admin Staff, and Auditors for controlled access.

2. Collaboration Tools

Enable comments, task tracking, and team management for better collaboration.

3. System Integrations

Connect CAMS to ERP systems, calendars, and financial software for seamless operations.

4. Analytics & Insights

Add trend analysis, department metrics, and faculty dashboards for actionable insights.

5. Mobile Application

Develop native iOS/Android apps with push notifications for real-time updates on the go.

---

10. Key Code Snippets

Here are the most important code snippets from our CAMS project:

1. Authentication System

JWT Token Generation (Backend)

JavaScript

```
// server.js - Login endpoint
app.post("/api/auth/login", async (req, res) => {
  try {
    const { email, password } = req.body;

    // Validate input
    if (!email || !password) {
      return res.status(400).json({ message: "Email and password are required"
});
    }

    // Read users from Excel
    const users = readExcelFile(usersFilePath);

    // Find user by email
    const user = users.find((user) => user.email === email);

    if (!user) {
      return res.status(401).json({ message: "Invalid email or password" });
    }

    // Compare passwords
    const isPasswordValid = await bcrypt.compare(password, user.password);

    if (!isPasswordValid) {
      return res.status(401).json({ message: "Invalid email or password" });
    }

    // Create JWT token
    const token = jwt.sign(
      { id: user.id, email: user.email, name: user.name },
      JWT_SECRET,
      { expiresIn: "24h" }
    );

    // Return user info (excluding password) and token
    const { password: _, ...userWithoutPassword } = user;

    res.json({
      message: "Login successful",
      token,
      user: userWithoutPassword,
    });
  } catch (error) {
    console.error("Login error:", error);
    res.status(500).json({ message: "Server error during login" });
  }
});
```

Authentication Middleware (Backend)

JavaScript

```javascript
// server.js - Authentication middleware

const authenticateToken = (req, res, next) => {

  const authHeader = req.headers["authorization"];

  const token = authHeader && authHeader.split(" ")[1];



  if (!token) {

    return res.status(401).json({ message: "Authentication required" });

  }



  jwt.verify(token, JWT_SECRET, (err, user) => {

    if (err) {

      return res.status(403).json({ message: "Invalid or expired token" });

    }



    req.user = user;
    next();
  });
};
```

Authentication Context (Frontend)

```javascript
// AuthContext.js - Authentication context provider
import { createContext, useState, useContext, useEffect } from "react";
import axios from "axios";

const AuthContext = createContext();

export const useAuth = () => useContext(AuthContext);

// Base URL for API requests
const API_BASE_URL = "http://localhost:5000";

export const AuthProvider = ({ children }) => {
  const [currentUser, setCurrentUser] = useState(null);
  const [isAuthenticated, setIsAuthenticated] = useState(false);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    // Check if user is logged in on component mount
    const checkAuthStatus = async () => {
      const token = localStorage.getItem("token");

      if (token) {
        try {
          // Set default auth header for all requests
          axios.defaults.headers.common["Authorization"] = `Bearer ${token}`;

          // Verify token with backend
          const response = await axios.get(`${API_BASE_URL}/api/auth/verify`);
          setCurrentUser(response.data.user);
          setIsAuthenticated(true);
        } catch (error) {
          console.error("Auth verification failed:", error);
          localStorage.removeItem("token");
          delete axios.defaults.headers.common["Authorization"];
        }
      }

      setLoading(false);
    };

    checkAuthStatus();
  }, []);

  const login = async (email, password) => {
    try {
      const response = await axios.post(`${API_BASE_URL}/api/auth/login`, {
        email,
        password,
      });

      const { token, user } = response.data;

      localStorage.setItem("token", token);
      axios.defaults.headers.common["Authorization"] = `Bearer ${token}`;

      setCurrentUser(user);
      setIsAuthenticated(true);

      return { success: true };
    } catch (error) {
      console.error("Login error:", error);
      return {
        success: false,
        message: error.response?.data?.message || "Login failed",
      };
    }
  };

  // Other auth methods...
```

```
  return (
    <AuthContext.Provider value={{ currentUser, isAuthenticated, loading,
login, register, logout }}>
      {!loading && children}
    </AuthContext.Provider>
  );
};
```

2. Excel-Based Data Storage

```javascript
// server.js - Excel file operations
const readExcelFile = (filePath) => {
  try {
    const workbook = xlsx.readFile(filePath);
    const sheetName = workbook.SheetNames[0];
    const worksheet = workbook.Sheets[sheetName];
    return xlsx.utils.sheet_to_json(worksheet);
  } catch (error) {
    console.error(`Error reading Excel file ${filePath}:`, error);
    return [];
  }
};

const writeExcelFile = (filePath, data) => {
  try {
    const workbook = xlsx.utils.book_new();
    const worksheet = xlsx.utils.json_to_sheet(data);
    xlsx.utils.book_append_sheet(workbook, worksheet, "Sheet1");
    xlsx.writeFile(workbook, filePath);
    return true;
  } catch (error) {
    console.error(`Error writing Excel file ${filePath}:`, error);
    return false;
  }
};

// Create Excel files if they don't exist
```

```
const createExcelFileIfNotExists = (filePath, headers) => {
  try {
    if (!fs.existsSync(filePath)) {
      console.log(`Creating new Excel file at ${filePath}`);
      const workbook = xlsx.utils.book_new();
      const worksheet = xlsx.utils.aoa_to_sheet([headers]);
      xlsx.utils.book_append_sheet(workbook, worksheet, "Sheet1");
      xlsx.writeFile(workbook, filePath);
      console.log(`Excel file created successfully at ${filePath}`);
    }
  } catch (error) {
    console.error(`Error creating Excel file at ${filePath}:`, error);
    // Create an empty directory structure if it doesn't exist
    const dir = path.dirname(filePath);
    if (!fs.existsSync(dir)) {
      fs.mkdirSync(dir, { recursive: true });
      console.log(`Created directory: ${dir}`);
    }
  }
};
```

3. File Upload Handling

JavaScript

```javascript
// server.js - File upload configuration
// Configure multer for file uploads
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, uploadsDir);
  },
  filename: (req, file, cb) => {
    const uniqueSuffix = Date.now() + "-" + Math.round(Math.random() * 1e9);
    cb(null, file.fieldname + "-" + uniqueSuffix +
path.extname(file.originalname));
  },
});

const upload = multer({
  storage,
  fileFilter: (req, file, cb) => {
    // Accept only PDF files
    if (file.mimetype === "application/pdf") {
      cb(null, true);
    } else {
      cb(new Error("Only PDF files are allowed!"), false);
    }
  },
  limits: {
    fileSize: 5 * 1024 * 1024, // 5MB limit
  },
});

// Project creation endpoint with file upload
app.post(
  "/api/projects",
  authenticateToken,
  upload.fields([
    { name: "agreementDocument", maxCount: 1 },
    { name: "billSettlementProof", maxCount: 1 },
  ]),
  (req, res) => {
    try {
      const projectData = req.body;
      const files = req.files;

      // Read existing projects
      const projects = readExcelFile(projectsFilePath);

      // Create new project
      const newProject = {
        id: Date.now().toString(),
        ...projectData,
        agreementDocument: files.agreementDocument ?
files.agreementDocument[0].filename : null,
        billSettlementProof: files.billSettlementProof ?
files.billSettlementProof[0].filename : null,
        createdAt: new Date().toISOString(),
        userId: req.user.id,
      };

      // Add project to Excel file
      projects.push(newProject);
      writeExcelFile(projectsFilePath, projects);

      res.status(201).json({ message: "Project added successfully", id:
newProject.id });
    } catch (error) {
      console.error("Error adding project:", error);
      res.status(500).json({ message: "Server error while adding project" });
    }
  }
);
```

4. Project Creation Form (Frontend)

```javascript
// ProjectEntry.js - Project creation form (partial)
const ProjectEntry = () => {
  const navigate = useNavigate();
  const [isLoading, setIsLoading] = useState(false);

  const [formData, setFormData] = useState({
    industryName: "",
    duration: "",
    title: "",
    principalInvestigator: "",
    coPrincipalInvestigator: "",
    academicYear: "",
    amountSanctioned: "",
    amountReceived: "",
    billSettlementDetails: "",
    studentDetails: "",
    summary: "",
  });

  const [files, setFiles] = useState({
    billSettlementProof: null,
    agreementDocument: null,
  });
```

```
const handleChange = (e) => {
  const { name, value } = e.target;
  setFormData((prev) => ({
    ...prev,
    [name]: value,
  }));
};

const handleFileChange = (e) => {
  const { name, files } = e.target;
  setFiles((prev) => ({
    ...prev,
    [name]: files[0],
  }));
};

const handleSubmit = async (e) => {
  e.preventDefault();

  // Basic validation
  if (
    !formData.industryName ||
    !formData.title ||
    !formData.principalInvestigator ||
    !formData.academicYear ||
    !formData.amountSanctioned
  ) {
    toast.error("Please fill in all required fields");
    return;
  }

  // File validation
  if (!files.agreementDocument) {
    toast.error("Please upload the signed agreement document");
    return;
  }

  setIsLoading(true);

  try {
    // Create form data for file upload
    const projectFormData = new FormData();
```

```
      // Add all text fields
      Object.keys(formData).forEach((key) => {
        projectFormData.append(key, formData[key]);
      });

      // Add files
      if (files.billSettlementProof) {
        projectFormData.append("billSettlementProof",
files.billSettlementProof);
      }

      if (files.agreementDocument) {
        projectFormData.append("agreementDocument", files.agreementDocument);
      }

      // Submit the form
      const response = await axios.post(`${API_BASE_URL}/api/projects`,
projectFormData, {
        headers: {
          "Content-Type": "multipart/form-data",
        },
      });

      toast.success("Project added successfully!");
      navigate(`/project/${response.data.id}`);
    } catch (error) {
      console.error("Error adding project:", error);
      toast.error(error.response?.data?.message || "Failed to add project");
    } finally {
      setIsLoading(false);
    }
  };

  // Form JSX...
};
```

5. Dashboard Implementation (Frontend)

JavaScript
```
// Dashboard.js - Dashboard component with API data fetching
const Dashboard = () => {
  const { currentUser } = useAuth();
  const [stats, setStats] = useState({
    totalProjects: 0,
    activeProjects: 0,
    completedProjects: 0,
    totalAmount: 0,
  });
  const [recentProjects, setRecentProjects] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchDashboardData = async () => {
      try {
        const [statsResponse, projectsResponse] = await Promise.all([
          axios.get(`${API_BASE_URL}/api/projects/stats`),
          axios.get(`${API_BASE_URL}/api/projects/recent`),
        ]);

        setStats(statsResponse.data);
        setRecentProjects(projectsResponse.data);
      } catch (error) {
        console.error("Error fetching dashboard data:", error);
        toast.error("Failed to load dashboard data");
      } finally {
        setLoading(false);
      }
    };

    fetchDashboardData();
  }, []);

  if (loading) {
    return (
      <div className="flex justify-center items-center h-64">
        <div className="animate-spin rounded-full h-12 w-12 border-t-2
border-b-2 border-blue-500"></div>
      </div>
    );
  }

  // Dashboard JSX...
};
```

6. Protected Route Implementation (Frontend)

JavaScript
```javascript
// App.js - Protected route implementation
// Protected route component
const ProtectedRoute = ({ children }) => {
  const { isAuthenticated } = useAuth();

  if (!isAuthenticated) {
    return <Navigate to="/login" replace />;
  }

  return children;
};

function App() {
  return (
    <AuthProvider>
      <Router>
        <div className="min-h-screen bg-gray-50">
          <Toaster position="top-right" />
          <Navbar />
          <div className="container mx-auto px-4 py-8">
            <Routes>
              <Route path="/login" element={<Login />} />
              <Route path="/register" element={<Register />} />
              <Route
                path="/"
                element={
                  <ProtectedRoute>
                    <Dashboard />
                  </ProtectedRoute>
                }
              />
              <Route
                path="/project/new"
                element={
                  <ProtectedRoute>
                    <ProjectEntry />
                  </ProtectedRoute>
                }
              />
              {/* Other protected routes... */}
              <Route path="*" element={<NotFound />} />
            </Routes>
          </div>
        </div>
      </Router>
    </AuthProvider>
  );
}
```

7. Project Listing with Filters (Frontend)

JavaScript

```javascript
// ProjectList.js - Project listing with filtering functionality
const ProjectList = () => {
  const [projects, setProjects] = useState([]);
  const [loading, setLoading] = useState(true);
  const [filters, setFilters] = useState({
    academicYear: "",
    amountThreshold: "",
    facultyName: "",
    industryName: "",
  });

  // Fetch projects function
  const fetchProjects = async () => {
    try {
      setLoading(true);
      const response = await axios.get(`${API_BASE_URL}/api/projects`);
      setProjects(response.data);
    } catch (error) {
      console.error("Error fetching projects:", error);
      toast.error("Failed to load projects");
    } finally {
      setLoading(false);
    }
  };

  // Apply filters function
  const applyFilters = async () => {
    try {
      setLoading(true);

      // Build query string from filters
      const queryParams = new URLSearchParams();

      if (filters.academicYear) {
        queryParams.append("academicYear", filters.academicYear);
      }

      if (filters.amountThreshold) {
        queryParams.append("amountThreshold", filters.amountThreshold);
      }

      if (filters.facultyName) {
        queryParams.append("facultyName", filters.facultyName);
      }

      if (filters.industryName) {
        queryParams.append("industryName", filters.industryName);
      }

      const response = await
axios.get(`${API_BASE_URL}/api/projects?${queryParams.toString()}`);
      setProjects(response.data);
    } catch (error) {
      console.error("Error applying filters:", error);
      toast.error("Failed to filter projects");
    } finally {
      setLoading(false);
    }
  };

  // Component JSX...
};
```

8. Excel Export Functionality (Backend)

JavaScript
```
// server.js - Excel export endpoint
app.get("/api/projects/download", authenticateToken, (req, res) => {
  try {
    const { academicYear, amountThreshold, facultyName, industryName } =
req.query;

    // Read projects from Excel
    let projects = readExcelFile(projectsFilePath);

    // Apply filters if provided
    if (academicYear) {
      projects = projects.filter((project) => project.academicYear ===
academicYear);
    }

    if (amountThreshold) {
      const threshold = Number.parseInt(amountThreshold);
      projects = projects.filter((project) =>
Number.parseInt(project.amountSanctioned) >= threshold);
    }

    if (facultyName) {
      const name = facultyName.toLowerCase();
      projects = projects.filter(
        (project) =>
          project.principalInvestigator.toLowerCase().includes(name) ||
          (project.coPrincipalInvestigator &&
project.coPrincipalInvestigator.toLowerCase().includes(name)),
      );
    }

    if (industryName) {
      const name = industryName.toLowerCase();
      projects = projects.filter((project) =>
project.industryName.toLowerCase().includes(name));
    }
    // Create a new workbook for download
    const workbook = xlsx.utils.book_new();

    // Remove file paths from the data
    const projectsForExport = projects.map((project) => {
      const { agreementDocument, billSettlementProof, ...rest } = project;
      return {
        ...rest,
        hasAgreementDocument: agreementDocument ? "Yes" : "No",
        hasBillSettlementProof: billSettlementProof ? "Yes" : "No",
      };
    });

    // Add the data to the workbook
    const worksheet = xlsx.utils.json_to_sheet(projectsForExport);
    xlsx.utils.book_append_sheet(workbook, worksheet, "Projects");

    // Create a temporary file
    const tempFilePath = path.join(dataDir, "temp-export.xlsx");
    xlsx.writeFile(workbook, tempFilePath);

    // Send the file
    res.download(tempFilePath, `consultancy-projects-${new
Date().toISOString().split("T")[0]}.xlsx`, (err) => {
      // Delete the temporary file after download
      if (fs.existsSync(tempFilePath)) {
        fs.unlinkSync(tempFilePath);
      }
    });
  } catch (error) {
    console.error("Error downloading projects Excel:", error);
    res.status(500).json({ message: "Server error while downloading projects"
});
  }
});
```

9. Notification System (Backend)

JavaScript

```javascript
// server.js - Email notification system
// Setup notification system
const sendNotifications = async () => {
  try {
    // Read projects and users from Excel
    const projects = readExcelFile(projectsFilePath);
    const users = readExcelFile(usersFilePath);

    // Create a transporter
    const transporter = nodemailer.createTransport({
      service: "gmail",
      auth: {
        user: process.env.EMAIL_USER || "your-email@gmail.com",
        pass: process.env.EMAIL_PASS || "your-password",
      },
    });

    // Get current date
    const currentDate = new Date();

    // Find projects that need notifications
    const newProjects = projects.filter((project) => {
      const creationDate = new Date(project.createdAt);
      const daysSinceCreation = Math.floor((currentDate - creationDate) / (1000
* 60 * 60 * 24));
      return daysSinceCreation <= 15; // Projects created within the last 15
days
    });

    // Find projects nearing completion based on duration
    const projectsNearingCompletion = projects.filter((project) => {
      if (!project.duration) return false;

      const creationDate = new Date(project.createdAt);
      const durationInDays = Number.parseInt(project.duration) * 30; // Convert
months to days
      const completionDate = new Date(creationDate.getTime() + durationInDays *
24 * 60 * 60 * 1000);
      const daysUntilCompletion = Math.floor((completionDate - currentDate) /
(1000 * 60 * 60 * 24));

      return daysUntilCompletion > 0 && daysUntilCompletion <= 15; // Projects
completing within 15 days
```

```
    });

    // Send notifications to users
    for (const user of users) {
      // Skip if no email
      if (!user.email) continue;

      let emailContent = "";

      // Add new projects to email
      if (newProjects.length > 0) {
        emailContent += "<h2>New Consultancy Projects</h2>";
        emailContent += "<ul>";
        for (const project of newProjects) {
          emailContent += `<li><strong>${project.title}</strong> -
${project.industryName}</li>`;
        }
        emailContent += "</ul>";
      }

      // Add projects nearing completion to email
      if (projectsNearingCompletion.length > 0) {
        emailContent += "<h2>Projects Nearing Completion</h2>";
        emailContent += "<ul>";
        for (const project of projectsNearingCompletion) {
          emailContent += `<li><strong>${project.title}</strong> -
${project.industryName}</li>`;
        }
        emailContent += "</ul>";
      }

      // Skip if no content
      if (!emailContent) continue;

      // Send email
      await transporter.sendMail({
        from: process.env.EMAIL_USER || "your-email@gmail.com",
        to: user.email,
        subject: "CAMS: Project Notifications",
        html: `
          <h1>CAMS Notifications</h1>
          <p>Hello ${user.name},</p>
          <p>Here are your consultancy project updates:</p>
        ${emailContent}
        <p>Login to the system for more details.</p>
        <p>Regards,<br>CAMS Team</p>
      `,
    });
  }

  console.log("Notifications sent successfully");
 } catch (error) {
  console.error("Error sending notifications:", error);
 }
};

// Schedule notifications to run every 15 days
cron.schedule("0 0 */15 * *", () => {
 console.log("Running scheduled notifications");
 sendNotifications();
});
```

10. Project Details View (Frontend)

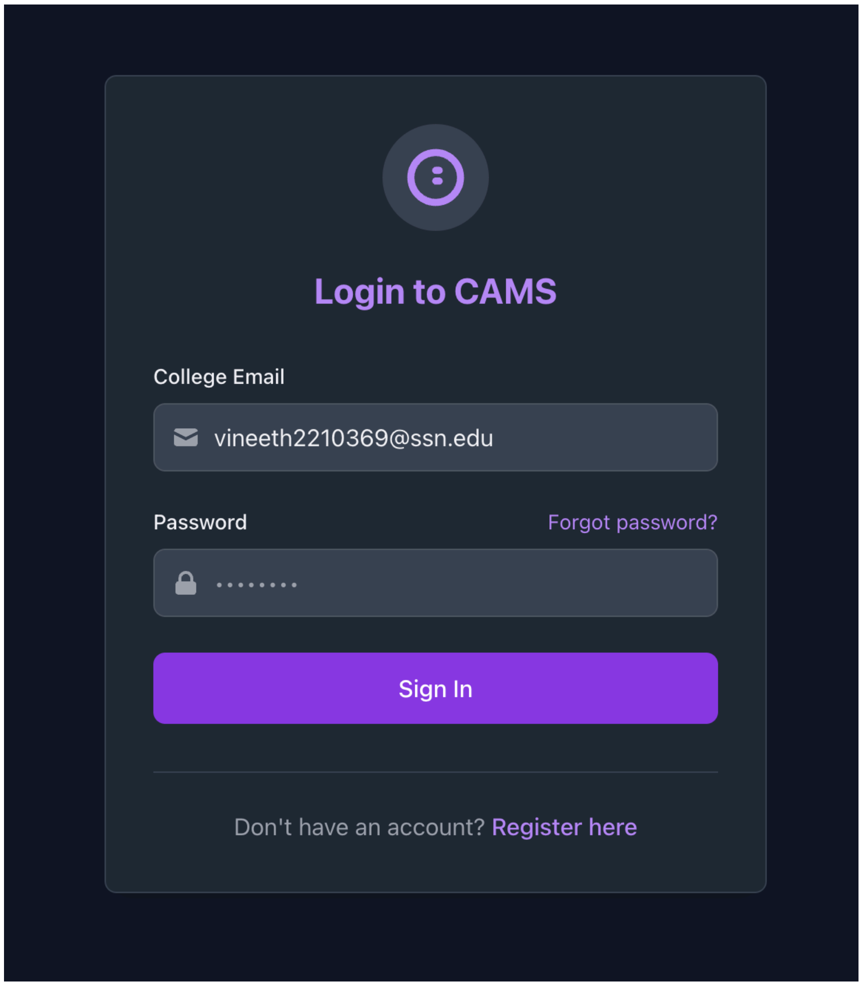
JavaScript

```
// ProjectDetails.js - Project details component
const ProjectDetails = () => {
  const { id } = useParams();
  const navigate = useNavigate();
  const [project, setProject] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchProjectDetails = async () => {
      try {
        const response = await axios.get(`${API_BASE_URL}/api/projects/${id}`);
        setProject(response.data);
      } catch (error) {
        console.error("Error fetching project details:", error);
        toast.error("Failed to load project details");
        navigate("/projects");
      } finally {
        setLoading(false);
      }
    };

    fetchProjectDetails();
  }, [id, navigate]);

  const handleDownloadFile = async (fileType) => {
    try {
      const response = await
axios.get(`${API_BASE_URL}/api/projects/${id}/download/${fileType}`, {
        responseType: "blob",
      });

      // Create a download link and trigger the download
      const url = window.URL.createObjectURL(new Blob([response.data]));
      const link = document.createElement("a");
      link.href = url;
      link.setAttribute("download", `${fileType}-${project.title}.pdf`);
      document.body.appendChild(link);
      link.click();
      link.remove();

      toast.success("File downloaded successfully");
    } catch (error) {
      console.error(`Error downloading ${fileType}:`, error);
      toast.error(`Failed to download ${fileType}`);
    }
  };

  // Component JSX...
};
```

These code snippets showcase the core functionality of our CAMS project. They highlight the key architectural decisions, data flow, and implementation details that make our system work.

11. Output Screenshots

LOGIN



Login to CAMS

College Email

vineeth2210369@ssn.edu

Password

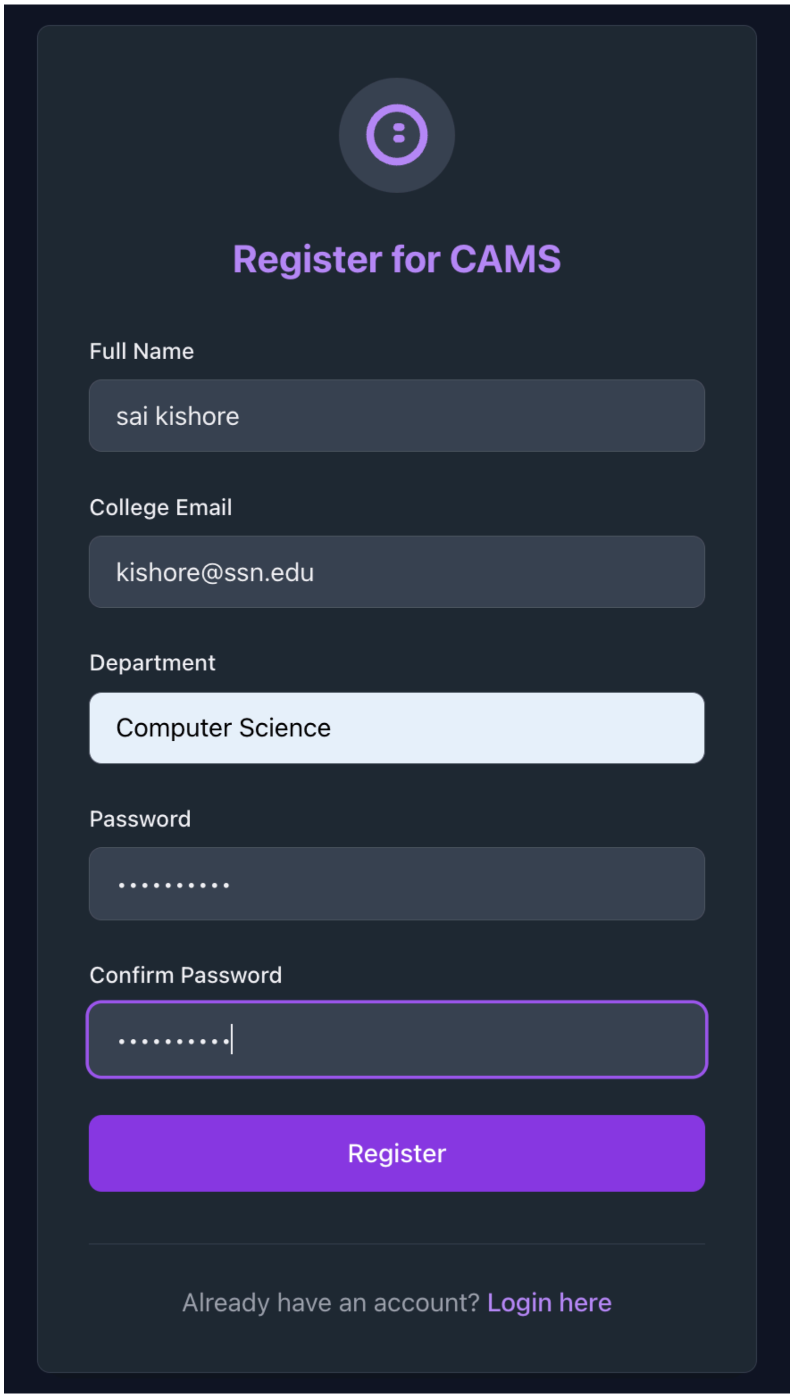
Forgot password?

••••••••

Sign In

Don't have an account? Register here

SIGNUP



Register for CAMS

Full Name: sai kishore

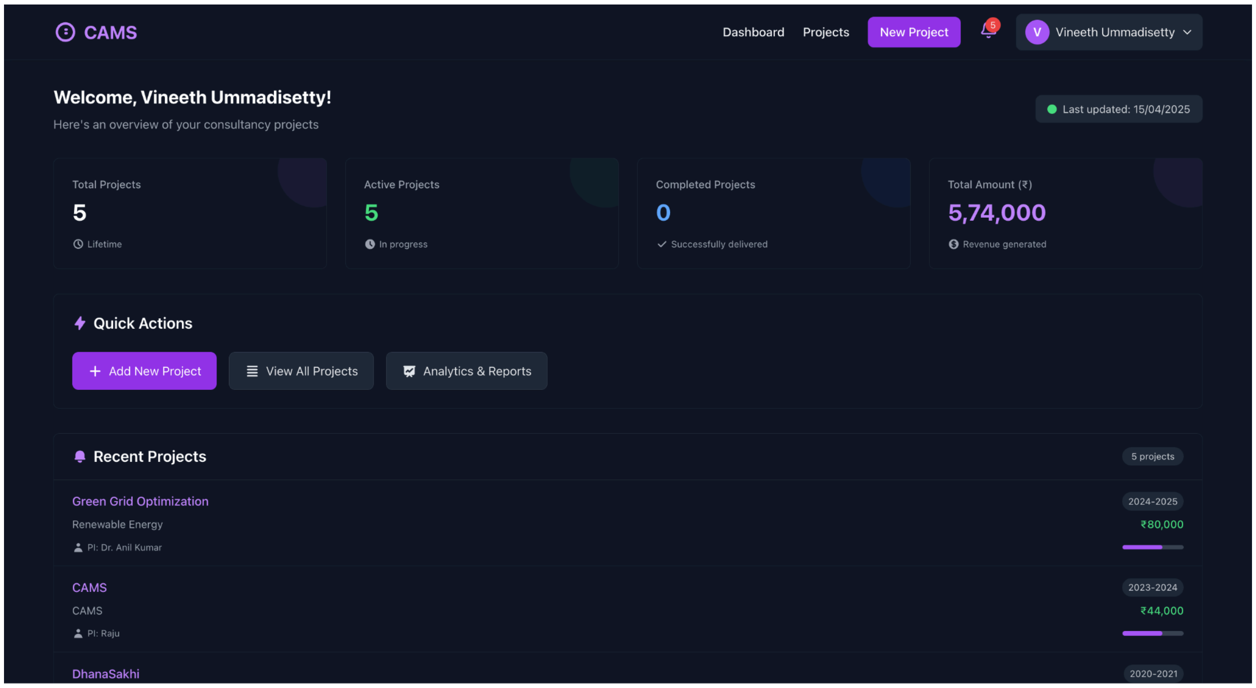
College Email: kishore@ssn.edu

Department: Computer Science

Password: ••••••••••

Confirm Password: ••••••••••

DASHBOARD



Welcome, Vineeth Ummadisetty!

Here's an overview of your consultancy projects

Last updated: 15/04/2025

Total Projects	Active Projects	Completed Projects	Total Amount (₹)
5	5	0	5,74,000
Lifetime	In progress	Successfully delivered	Revenue generated

Quick Actions

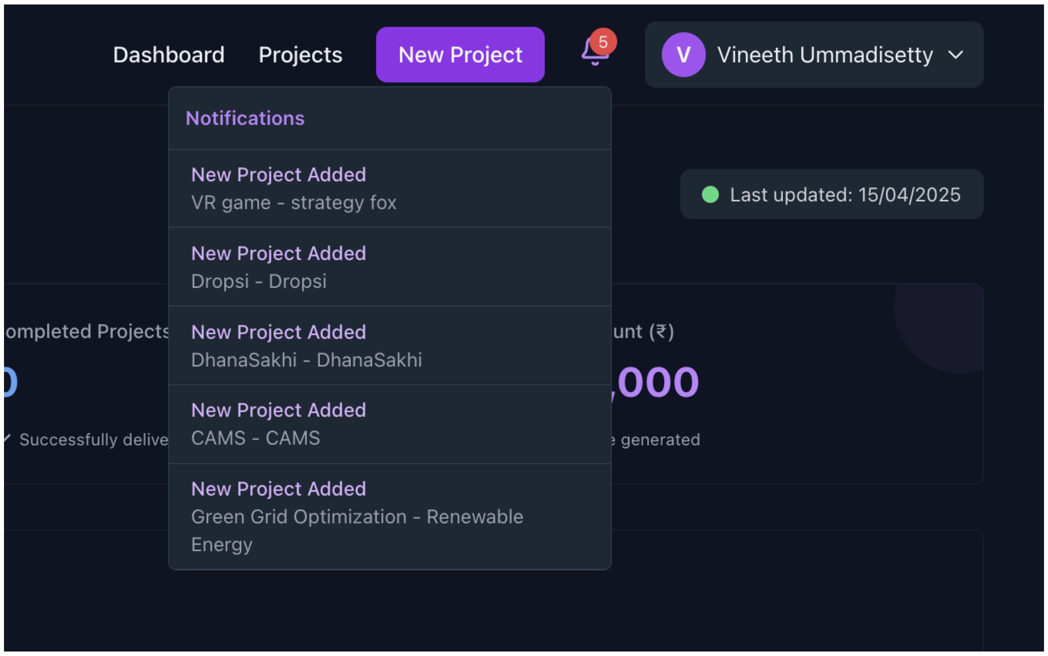
Recent Projects — 5 projects

Green Grid Optimization
Renewable Energy
PI: Dr. Anil Kumar
2024-2025
₹80,000

CAMS
CAMS
PI: Raju
2023-2024
₹44,000

DhanaSakhi
2020-2021

NOTIFICATION ALERT



Notifications

New Project Added
VR game - strategy fox

New Project Added
Dropsi - Dropsi

New Project Added
DhanaSakhi - DhanaSakhi

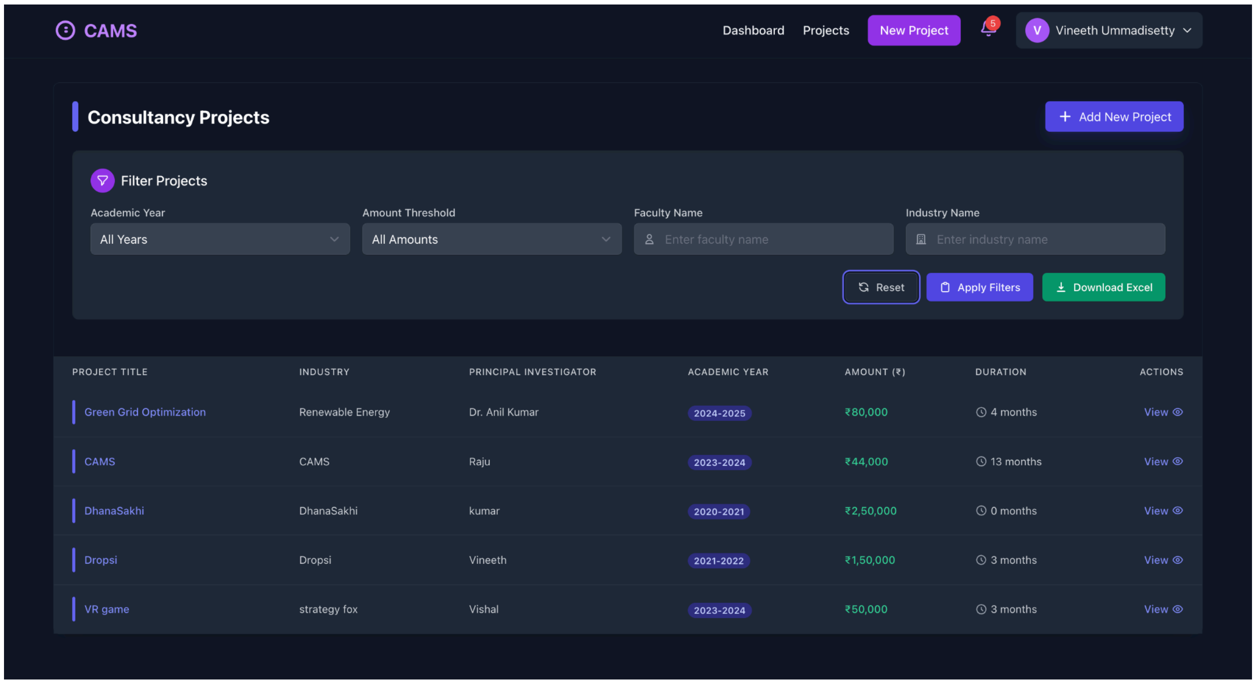
New Project Added
CAMS - CAMS

New Project Added
Green Grid Optimization - Renewable Energy

Last updated: 15/04/2025

ompleted Projects
0
Successfully delive
unt (₹)
,000
generated

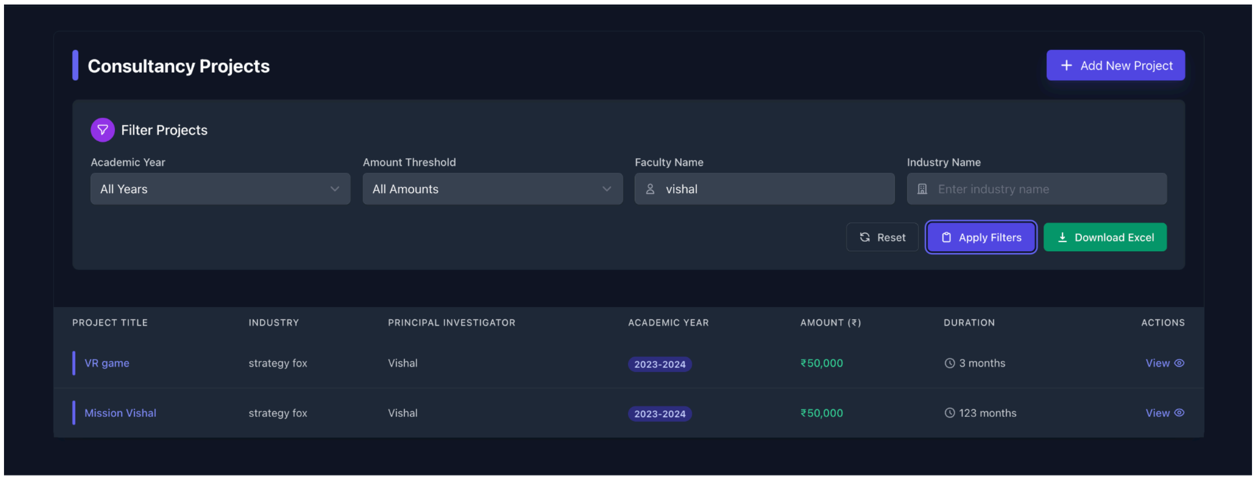
PROJECT LIST



Consultancy Projects

Academic Year: All Years
Amount Threshold: All Amounts
Faculty Name: Enter faculty name
Industry Name: Enter industry name

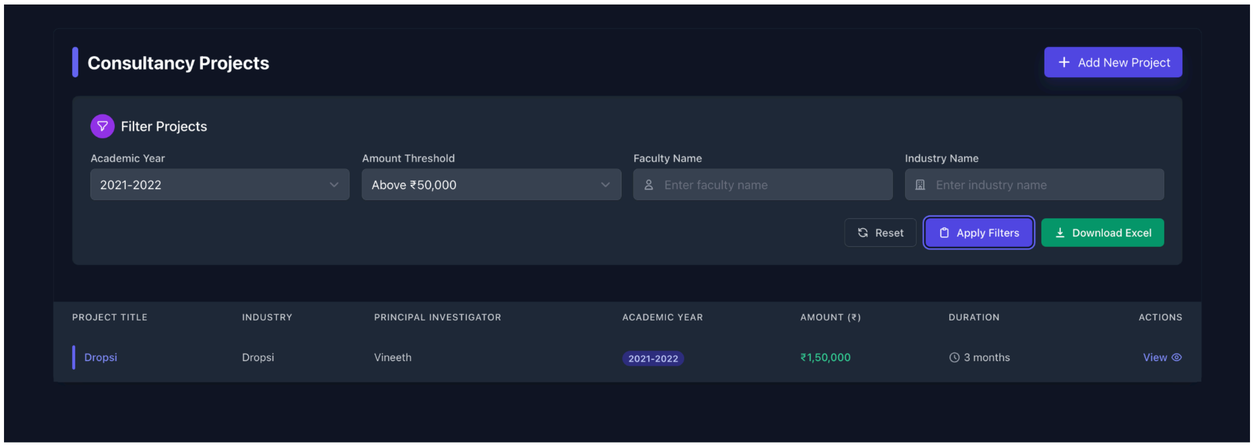
PROJECT TITLE	INDUSTRY	PRINCIPAL INVESTIGATOR	ACADEMIC YEAR	AMOUNT (₹)	DURATION	ACTIONS
Green Grid Optimization	Renewable Energy	Dr. Anil Kumar	2024-2025	₹80,000	4 months	View
CAMS	CAMS	Raju	2023-2024	₹44,000	13 months	View
DhanaSakhi	DhanaSakhi	kumar	2020-2021	₹2,50,000	0 months	View
Dropsi	Dropsi	Vineeth	2021-2022	₹1,50,000	3 months	View
VR game	strategy fox	Vishal	2023-2024	₹50,000	3 months	View



Consultancy Projects

Academic Year: All Years
Amount Threshold: All Amounts
Faculty Name: vishal
Industry Name: Enter industry name

PROJECT TITLE	INDUSTRY	PRINCIPAL INVESTIGATOR	ACADEMIC YEAR	AMOUNT (₹)	DURATION	ACTIONS
VR game	strategy fox	Vishal	2023-2024	₹50,000	3 months	View
Mission Vishal	strategy fox	Vishal	2023-2024	₹50,000	123 months	View



Consultancy Projects

Add New Project

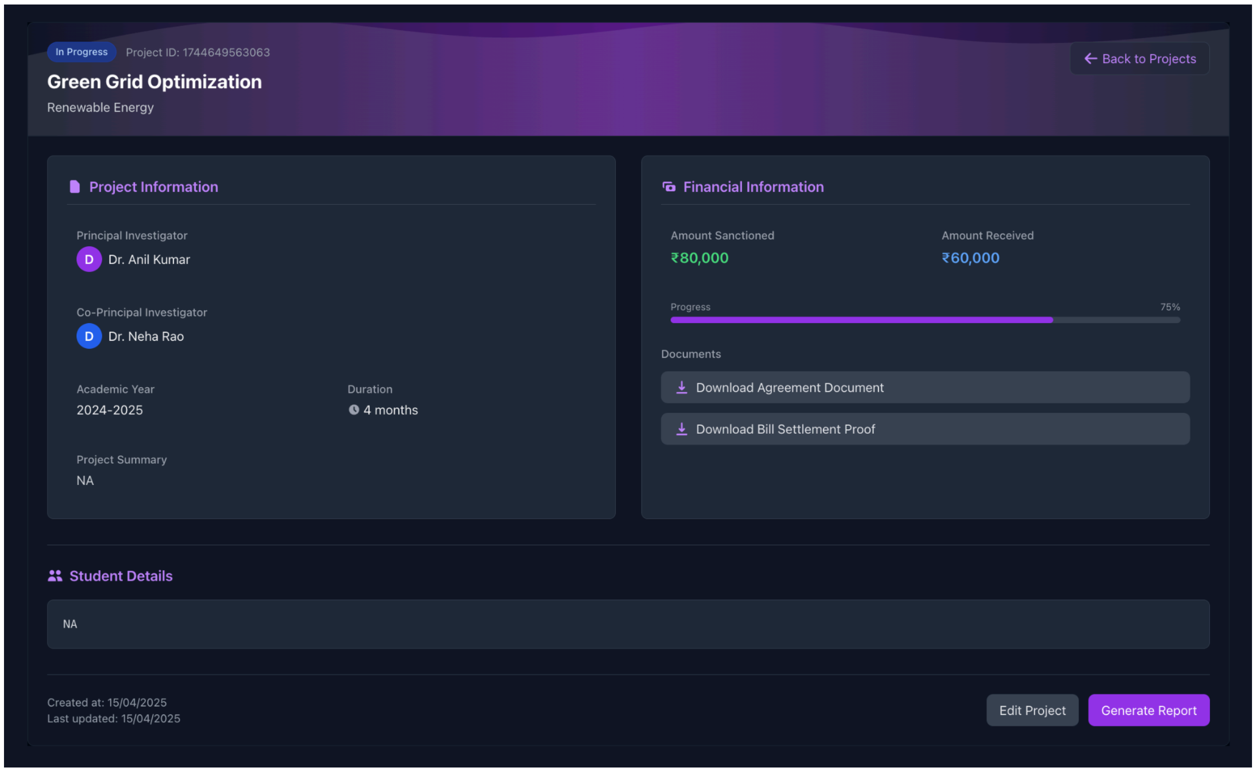
Filter Projects

Academic Year	Amount Threshold	Faculty Name	Industry Name
2021-2022	Above ₹50,000	Enter faculty name	Enter industry name

Reset Apply Filters Download Excel

PROJECT TITLE	INDUSTRY	PRINCIPAL INVESTIGATOR	ACADEMIC YEAR	AMOUNT (₹)	DURATION	ACTIONS
Dropsi	Dropsi	Vineeth	2021-2022	₹1,50,000	3 months	View

PROJECT VIEW



In Progress Project ID: 1744649563063

Green Grid Optimization

Renewable Energy

Back to Projects

Project Information

Principal Investigator
D Dr. Anil Kumar

Co-Principal Investigator
D Dr. Neha Rao

Academic Year
2024-2025

Duration
4 months

Project Summary
NA

Financial Information

Amount Sanctioned
₹80,000

Amount Received
₹60,000

Progress 75%

Documents

Download Agreement Document

Download Bill Settlement Proof

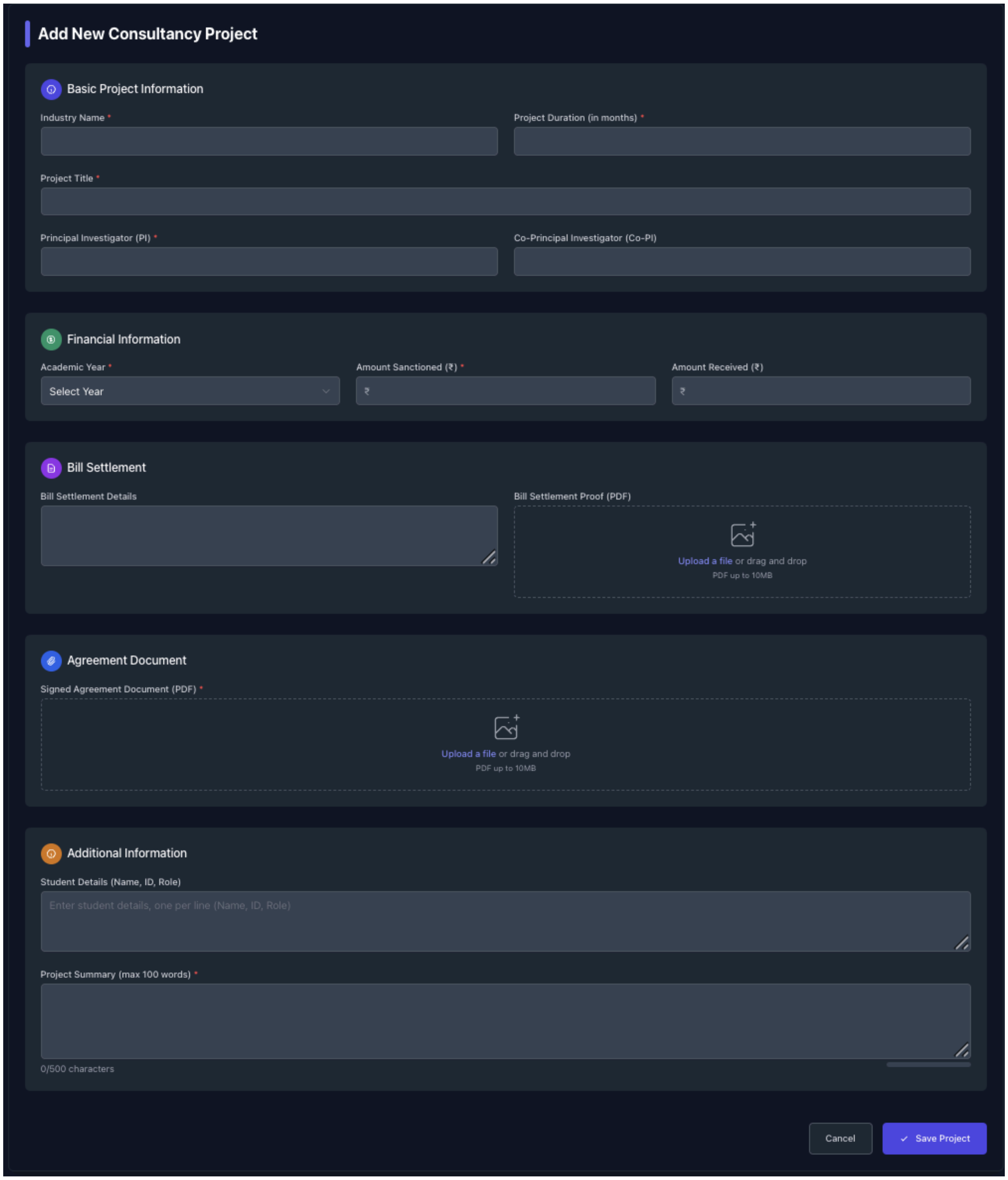
Student Details

NA

Created at: 15/04/2025
Last updated: 15/04/2025

Edit Project Generate Report

PROJECT ENTRY



Add New Consultancy Project

Basic Project Information

Industry Name *

Project Duration (in months) *

Project Title *

Principal Investigator (PI) *

Co-Principal Investigator (Co-PI)

Financial Information

Academic Year *

Select Year

Amount Sanctioned (₹) *

₹

Amount Received (₹)

₹

Bill Settlement

Bill Settlement Details

Bill Settlement Proof (PDF)

Upload a file or drag and drop

PDF up to 10MB

Agreement Document

Signed Agreement Document (PDF) *

Upload a file or drag and drop

PDF up to 10MB

Additional Information

Student Details (Name, ID, Role)

Enter student details, one per line (Name, ID, Role)

Project Summary (max 100 words) *

0/500 characters

12. Learning Outcomes

- Gained practical experience in building a **full-stack web application** using the MERN stack.
- Learned how to handle **Excel-based data storage** using Node.js libraries like xlsx and exceljs.
- Understood how to integrate **file uploads** and manage documents within a web application.
- Implemented a **cron-based notification system** to improve user engagement and reminders.
- Developed skills in building user authentication and **access-controlled interfaces**.
- Enhanced understanding of organizing code structure and component-based architecture using **React**.