TEAM-2 DOCUMENTATION

NGO: St. Bhatevara Foundation

TEAM MEMBERS

1. DAKSH RAHEJA
2. JYOTHI SAMJYOTHA
3. NAMITHA NERELLA
4. K MOHANA
5. RISHIKTEJ REDDY
6. SIDDESHWARI ADEPU
7. NARAYANA

SDLC PHASES

Phase 1: RESEARCH

1. **Project Scope and Objectives**
    - Project's purpose: Create a digitalised system to manage scholarships, student data, and facilitate communication between students and alumni.
    - Objectives:
        - Digitize the scholarship application process for freshers and renewed students in 3 levels.
        - Create a database for storing and visualizing student, alumni and ngo's data.

- Generate reports based on stored data.
- Facilitate networking and mentoring between current students and alumni.
- Track student progress and performance.

2. **Funding Sources**
    - Donations and grants from supporters.
    - Partnerships with other NGOs.
    - Sponsorship from corporations or educational institutions.
3. **Benefits**
    - Increased efficiency in managing scholarships.
    - Better data management and reporting.
    - Enhanced communication between students and alumni.
    - Reduced manual labor and errors.

Phase 2: Requirements Analysis

1.Information Required:

- **Partner NGOs**: Gather information on their current process for identifying and submitting student profiles.
- **Volunteer:** Gather the volunteer information and tracking their approval process
- **Students and Alumni**: Collect insights on their needs for networking, accessing information, and interacting with the NGO.

2.Document Analysis

- Review existing documents i.e, application forms, Excel sheets, and reports to understand current data structures and workflows.

3.Functional Requirements

- **Scholarship Application and Approval**
    1. **Application Submission**: Allow students to submit their profiles through a web portal.
    2. **Validation Process**: Implement a multi-step validation process where profiles are first validated by partner NGOs, volunteers and then by the foundation's committee.
    3. **Approval Workflow**: Define and automate the workflow for verifying and approving scholarships, including notifications to applicants and partner NGOs.
- **Student Data Management**
    1. **Profile Management**: Maintain detailed profiles for students, including personal information, academic performance tracking,attendance and additional vocational courses statistics and scholarship details.
    2. **Alumni Data**: Track alumni information, including current occupation and contact details.
- **Reporting and Data Visualization**
    1. **Custom Reports**: Generate reports on various metrics such as scholarship distribution, student performance and attendance, approval status, number of students, enrolled courses percentages.
    2. **Dashboards**: Provide visual dashboards for quick insights into key data points
- **Networking and Mentoring**
    1. **Alumni Network**: Create a platform for current students to connect with alumni for mentorship and career guidance.
    2. **Community Building**: Implement discussion forums for students and alumni to interact and share knowledge.

4.Non- Functional Requirements

1. **Usability :** The system should be simple and intuitive for everyone to use.
2. **Control Access**: Make sure users can only see what they're allowed to see.
3. **Performance**: The system should work well even as the number of users and data grows.
4. **Portability**: The system should work on different platforms with minimal changes.
5. **Maintainability**: Design the system in a way that makes updates and maintenance simple.

Phase 3:Design

- **Components**:
  - **User Interface (UI)**: Frontend of the system.
  - **Application Logic (Backend)**: Server-side processing.
  - **Database**: Storage of all data.
  - **Reporting Service**: Generates reports and visualizations.
  - **Integration APIs**: Interfaces for external systems.

- **Detailed Architecture:**

  1. **Frontend**
  - **Technology**: React.js
  - **Components**:
    - **Forms**: Registration, login, courses and verification forms
    - **Dashboards**: Students, NGO's , volunteers, trustees and alumni dashboards.
    - **Tables**: Displaying student data

  2 .**Backend**:

  - **Technology**: Node.js with Express

    - **APIs**: RESTful APIs for frontend-backend communication.
    - **Modules**:
        - **User Management**: Handles user roles and permissions.
        - **Scholarship Management**: Manages applications, validations, and approvals.
        - **Reporting**: Generates and manages reports.
        - **Alumni Network**: Facilitates communication between students and alumni.
- **Database**:
    - **Technology**: MongoDB
    - **Schema Design**: Tables for students, scholarships, users, applications, trainings, alumni, reports.
- **Flow Diagrams**:
    - Visualize user navigation through the system.
    - **Examples**: User registration and login, submitting a scholarship application, viewing application status, generating a report.
- **Reports**:
    - Types of reports needed (e.g., scholarship distribution, student performance, student attendance).
- **Data Visualization**:
    - Charts, graphs, and dashboards for data visualization.
- **Libraries:**:
    - Chart.js for data visualization.
- **Integration**:
    - Ensure dynamic data fetching and display.

Phase 4:Implementation

Frontend Development

1. **Setup Project Structure**:
    - Initialize the project using create-react-app.
    - Set up folder structure for components, services, and assets.
2. **Develop Components**:
    - Create reusable components (e.g., forms, buttons, tables).
    - Implement pages (e.g., login, dashboard, application form).
3. **API Integration**:
    - Set up services to handle API calls.
    - Integrate frontend components with backend APIs.
4. **Styling**:
    - Use CSS/SASS for styling.
    - Ensure responsiveness across devices.

Backend Development

1. **Setup Project Structure**:
    - Initialize the Node.js project.
    - Set up folder structure for controllers, models, routes, and services.
2. **Develop APIs**:
    - Implement RESTful APIs for user management, scholarship management, reporting, and alumni network.
3. **Database Integration**:
    - Set up for MONGODB integration.
    - Implement data models and migrations.
4. **Authentication and Authorization**:
    - Implement JWT-based authentication.
    - Set up role-based access control.

Future Phases:Cloud Deployment and Maintenance

In the future, ongoing testing, deployment, and maintenance will ensure the scholarship management system remains reliable and effective. Regular updates and improvements will be implemented to enhance functionality, security, and user experience, supported by thorough testing and seamless deployment practices.

DATABASE SCHEMA

USER SCHEMA:

```
const UserSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  username: {
    type: String,
    required: true,
    unique: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  role: {
    type: String,
    enum: ['student', 'alumni', 'volunteer', 'trustee', 'NGO'],
    required: true,
  },
});
```

NOTIFICATION SCHEMA:

```
const NotificationSchema = new Schema({
  title: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  date: {
    type: Date,
    default: Date.now,
  }
});
```

SCHOLARSHIP SCHEMA:

```
const ScholarshipRequestSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  course: {
    type: String,
    required: true,
  },
  amount: {
    type: Number,
    required: true,
  },
  collegeName: {
    type: String,
    required: true,
  },
  bankAccountNumber: {
    type: String,
    required: true,
  },
  incomeStatement: {
    type: String,
    required: true,
  },
  marksheet10: {
    type: String,
    required: true,
  },
  marksheet12: {
    type: String,
    required: true,
  },
  approvedBy: {
    type: [String],
    enum: ['NGO', 'volunteer', 'trustee'],
    default: [],
  },
});
```

```
const RenewalRequestSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  amount: {
    type: Number,
    required: true,
  },
  incomeStatement: {
    type: String,
    required: true,
  },
  receipt: {
    type: String,
    required: true,
  },
  collegeMarksheet: {
    type: String,
    required: true,
  },
  approvedBy: {
    type: [String],
    enum: ['NGO', 'volunteer', 'trustee'],
    default: [],
  },
});
```

UML DIAGRAMS