### **Project Tracker - Service Charter Document**

### **Purpose**

### To provide a centralized project management platform for teams. The platform helps:

### Track tasks status

### Monitor progress

### Manage deadlines

### Verify completed work

### Improve collaboration

### Manage documentation

### **Objective**

### Simplify project management for cross-functional teams.

### Enable users to create, assign, and track tasks with real-time updates.

### Provide tools for progress monitoring (dashboards, timelines and virtual meetings).

### **Demand / Opportunity**

### Responds to the increasing demand for simple yet effective project tracking tools in small-team environments.

### Fills the gap left by overly complex tools or those not designed for student needs or basic needs.

### Helps organizations and teams move away from inefficient solutions like spreadsheets or informal chats.

### **Business Requirement**

#### **4.1 User Authentication**

### Login system based on email and password and Google OAuth through Firebase Authentication for session handling.

#### **4.2 Project & Task Management**

### CRUD operations for projects, tasks, and milestones with proper role-based permissions.

#### **4.3 Progress Tracking**

### Dashboards, charts, and Gantt timelines to monitor task and milestone statuses.

#### **4.4 Real-Time Notifications**

### Email or in-app notifications for task updates, deadlines, or completion.

#### **4.5 Role Assignment**

### Users assigned as project creators, project completers; ability to comment and give feedback.

### **4.6 Real-time Communication and collaboration**

### Users can set up virtual meets to discuss progress and prevent miscommunication.

### **4.7 Download Project Report**

### Users can download detailed reports containing project information, milestones, tasks, team members and deadlines.

### 

### **Technical Requirement**

#### **5.1 User Authentication**

### API routes for login, registration, and session handling using Firebase Authentication.

### Password encryption and role-based access.

#### **5.2 Project Management**

### Backend logic for project/task CRUD operations.

### Role assignment and deadline tracking.

#### **5.3 Progress Visualization**

### Charts, timelines using libraries like Chart.js or D3.js.

#### **5.4 Notification**

### Integration with NodeMailer for real-time and email notifications.

### 

#### **5.5 Collaboration and communication**

### Integration with Google Calendar to automatically schedule virtual meetings.

### **Technological Requirement**

#### **6.1 Frontend**

### React.js

### Tailwind CSS

### shadcn/ui

#### **6.2 Backend**

### Node.js with Express.js

#### **6.3 Database**

### PostgreSQL

### Firestore

#### **6.4 Authentication**

### Firebase Authentication

#### **6.5 Virtual meetings**

### Google Meet through Google Calendar (.ics invites)

#### **6.6 Tools**

### Visual Studio Code

### Git & GitHub

### Postman

### AWS Services ( S3)

### Railway

### **Stakeholders**

|  |  |  |
| --- | --- | --- |
| **Stakeholder** | **Name** | **Count** |
| Teams/Users | NA | - |
| Developers | Dhanwantari, Viraj, Om, Shripad, Parth | 5 |
| Testers | Srivaths, Hanumant, Prajwal, Avishkar | 4 |
| Cloud Service Provider | AWS, Railway | 1 |
| Third-parties | React, Node.js, Tailwind Developers, etc. | - |

### 

### **Resources Needed**

#### **8.1 Documentation**

### React.js

### Tailwind CSS

### PostgreSQL

### Express.js

* Firebase

### Cloud Account

#### **8.2 Human Resource\***

|  |  |  |
| --- | --- | --- |
| **Role** | **Count** | **Names** |
| Frontend Developers | 2 | Parth, Dhanwantari |
| Backend Developers | 3 | Viraj, Shripad, Om |
| API Integration | 2 | Om, Dhanwantari, Viraj |
| DB Designer | 2 | Shripad, Parth |
| Documentation | 2 | Parth, Dhanwantari, Om |
| Testers | 4 | Srivaths, Hanumant, Prajwal, Avishkar |

### 

### **PESTEL Analysis**

#### **9.1 Political**

### No direct impact; follows industrial policies.

#### **9.2 Economic**

### Cost-efficient, scalable. Reduces project delays.

#### **9.3 Social**

### Promotes structured collaboration and accountability in teams.

#### **9.4 Technological**

### Built on modern tech stack; supports scalability, extensibility, and cross-device use.

#### **9.5 Legal**

### Compliant with IT Act 2000 and GDPR policies for user data protection.

#### **9.6 Environmental**

### No environmental impact; hosted on cloud infrastructure.

### **Risk Analysis**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Description** | **Mitigation Strategy** |
| Data Breaches | Sensitive data can be exposed. | Use Firebase Authentication, encrypt sensitive info, log access. |
| Downtime | Service unavailability due to AWS, DB or API outages. | Use multi-AZ deployment and load balancing. |
| Compliance | GDPR or institutional regulation violations. | Follow legal protocols; seek data protection officer’s review. |
| High Cost | Budget overrun due to third-party services. | Optimize code, use serverless APIs wherever possible. |
| UI/UX Confusion | Users might find it hard to navigate initially. | Conduct usability testing and regular UX audits. |
| Security Weakness | Improper access control or outdated tokens. | Use Firebase Authentication with expiration and secure role checks in backend APIs. |
| Blank data exports | The exported PDF file might be empty due to missing data or export failure. | Add data validation before export; show users a warning if data is missing. |
| Mobile Performance Issues | Poor mobile experience | Use mobile-first approach, optimize for 3G networks. |
| Database Scalability | PostgreSQL performance degrades with increased concurrent users. | Implement connection pooling, database indexing, query optimization, and horizontal scaling plans. |
| Cross-Platform Compatibility | Application doesn't work consistently across different devices/browsers. | Implement comprehensive browser testing, use progressive enhancement, maintain device compatibility matrix. |
| API Rate Limiting | Third-party services (Firebase, cloud providers) impose unexpected rate limits. | Implement exponential backoff, request queuing, monitor API usage, and have fallback mechanisms. |

### 

### **11. Project Timeline / Milestone Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phase** | **Timeline** | **Milestone** | **Tasks** | **Duration** |
| **Phase 1: Project Foundation** | **Week 1** | **Planning** | **• Project Setup & Environment****• Technical Architecture Design****• Database Schema Design****• API Endpoints Planning** | **4 days** |
|  |  | **Wireframing** | **• User Flow Diagram****• Low-fidelity Wireframes** | **3 day** |
| **Phase 2: Infrastructure Setup** | **Week 1** | **Database & Infrastructure** | **• PostgreSQL Database Setup****• Prisma ORM Configuration****• AWS S3 Bucket Configuration****• Environment Configuration** | **5 days** |
|  |  | **Authentication System** | **• Firebase Admin SDK Setup****• Authentication Middleware** | **2 day** |
| **Phase 3: Backend Development** | **Week 2** | **Basic API Structure** | **• Express Server Setup****• User Management API** | **3 day** |
|  |  | **Backend API Development** | **• Project CRUD Operations APIs****• User Management APIs****• Member Invitation APIs****• Meeting Scheduling APIs****• Project Details Export APIs****• Task Completion Review API** | **4 days** |
| **Phase 4: Frontend Development** | **Week 3** | **Core React Setup** | **• React App Initialization****• Routing & Navigation Setup** | **3 days** |
|  |  | **Authentication Components** | **• Firebase Client Setup****• Login/Register Components** | **3 day** |
|  |  | **Core UI Components** | **• Dashboard Layout****• Project Management Interface****• Member Invitation Interface****• Timeline Interface****• Task Completion - Proof Submission Interface****• Task Completion - Verification Interface** | **7 days** |
| **Phase 5: Integration & Features** | **Week 4-5** | **Frontend-Backend Integration** | **• Authentication Integration****• Project Management Integration****• Task Management Integration****• User Management Integration****• Member Invitation Integration****• Meeting Scheduling Integration****• Project Details Export Integration****• Task Completion Review Integration** | **2 days** |
| **Phase 6: Quality Assurance** | **Week 6** | **Testing** | **• Unit Testing - API Endpoints****• Authentication Testing****• Unit Testing - UI Components****• Integration Testing - UI & Database****• End-to-End Testing****• Cross-browser Testing** | **3 days** |
| **Phase 7: Launch Preparation** | **Week 6** | **Deployment Setup** | **• Production Environment Setup****• CI/CD Pipeline Configuration** | **2 day** |

### **Total Project Duration: 6 weeks**

### 

### 

### 

### 

### 

### 

### 

### **12. RACI Chart**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ACTIVITY / MEMBERS** | **Parth** | **Dhanwantari** | **Viraj** | **Om** | **Shripad** | **Jayesh Sir** | **Anuradha Mam** |
| **PROJECT FOUNDATION** | | | | | | | |
| **Project Setup & Environment** | I | C | C | R | A | C | I |
| **Technical Architecture Design** | R | I | C | C | A | C | I |
| **Database Schema Design** | R | C | A | C | I | C | I |
| **API Endpoints Planning** | C | I | R | A | C | C | I |
| **User Flow Diagram** | A | R | C | I | C | C | I |
| **Low-fidelity Wireframes** | C | A | C | R | I | C | I |
| **INFRASTRUCTURE SETUP** | | | | | | | |
| **PostgreSQL Database Setup** | I | C | C | R | A | C | I |
| **Prisma ORM Configuration** | R | I | C | A | C | C | I |
| **Firebase Admin SDK Setup** | C | I | A | R | C | C | I |
| **Authentication Middleware** | C | I | R | A | C | C | I |
| **BACKEND DEVELOPMENT** | | | | | | | |
| **Express Server Setup** | C | C | A | R | I | C | I |
| **User Management API** | R | I | A | R | C | C | I |
| **Project CRUD Operations APIs** | A | C | I | C | R | C | I |
| **Member Invitation APIs** | C | R | A | I | C | C | I |
| **Task Completion Review API** | C | I | R | A | C | C | I |
| **API Testing** | I | R | C | A | C | I | I |
| **FRONTEND DEVELOPMENT** | | | | | | | |
| **React App Initialization** | A | C | C | I | R | C | I |
| **Routing & Navigation Setup** | R | A | C | I | I | C | I |
| **Firebase Client Setup** | C | C | R | A | I | C | I |
| **Login/Register Components** | A | R | I | C | C | C | I |
| **Dashboard Layout** | R | C | A | I | C | C | I |
| **Project Management Interface** | C | A | R | C | I | C | I |
| **Member Invitation Interface** | C | C | A | I | R | C | I |
| **Task Completion Interfaces** | A | C | C | R | I | C | I |
| **INTEGRATION & FEATURES** | | | | | | | |
| **Authentication Integration** | R | I | A | C | C | C | I |
| **Project Management Integration** | I | R | C | C | A | C | I |
| **Task Management Integration** | A | C | R | I | C | C | I |
| **User Management Integration** | C | A | C | R | I | C | I |
| **Member Invitation Integration** | A | I | C | C | R | C | I |
| **Task Completion Review Integration** | I | C | A | R | C | C | I |
| **QUALITY ASSURANCE** | | | | | | | |
| **Authentication Testing** | C | R | I | A | C | I | I |
| **Integration Testing - UI & Database** | C | C | A | R | I | I | I |
| **Basic End-to-End Testing** | R | C | C | I | A | I | I |
| **LAUNCH PREPARATION** | | | | | | | |
| **Production Environment Setup** | I | C | A | C | R | C | I |
| **Manual Deployment** | C | I | R | A | C | C | I |

### 

### 

### **13. Budget**

|  |  |
| --- | --- |
| **Service** | **Monthly Cost (INR)** |
| AWS EC2 | 567.07 |
| AWS S3 + CloudWatch | 2000.00 |
| Development Cost | 20000 \* 5 = 100000 |
| Total Estimated Cost | 102567.07 INR |