Project My Basecamp:

My basecamp project aims to replicate the basic functionality of the widely-used project management and team collaboration tool called Basecamp launched in 2004. Basecamp offers a centralized platform for teams to organize their work, communicate effectively, and track progress on projects.

Project Plan:

- Requirements
- System Design
- Database Design
- Project Plan
- Testing
- Deployment

Requirements:

Functional Requirements:

- User should be able to create, edit and delete projects, add team members
- User should be able to create a new user, add team member, delete a user
- User should be able to log in and log out
- User authentication and authorization. Ability to add and remove the admin permission from a user.
- Task management: creating, assigning, and tracking tasks within projects.
- File sharing: upload, download, and manage files associated with projects.

Non-Functional Requirements:

- User-friendly interface for ease of navigation and task management
- Fast response times under the load of multiple users
- Ability to scale up for more projects and users as needed.
- Secure handling of data with proper authentication and authorization
- High availability and minimal downtime

System Constraints:

- Limited development resources and time constraints
- Single-page application architecture due to lightweight nature

Assumptions and Dependencies:

- Assumes stable internet connectivity for real-time features
- Dependent on third-party libraries for certain functionalities.

Hardware interface:

- No specific hardware interface requirements

Memory Constraints:

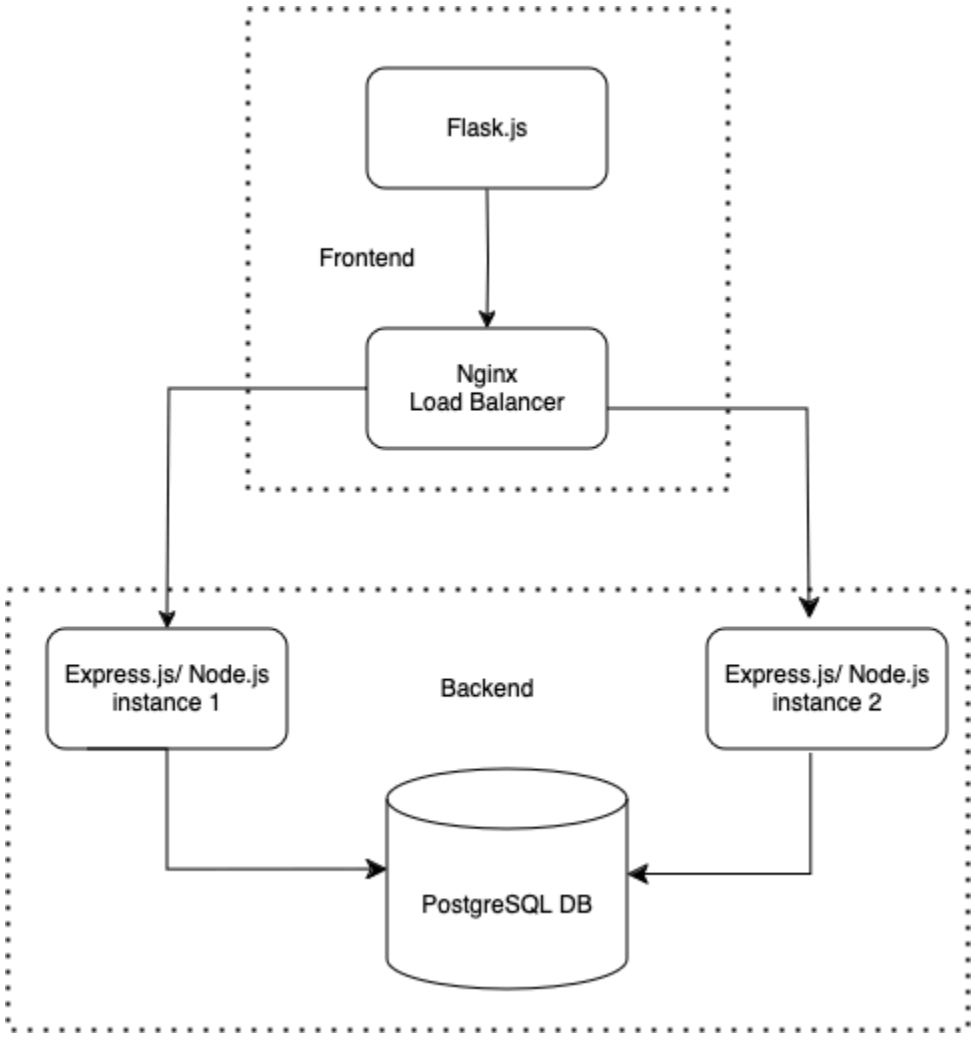
- Optimize memory usage for efficient performance on devices with limited resources.

Interfaces:

- Web-based interface with responsive design for desktop

Diagram:

- A high-level component diagram to visualize the system architecture and major components.



System Design:

Architecture and Design:

- Client-Server model application architecture using React for frontend and Node.js for backend.

Data Flow:

- Frontend interacts with the backend REST API to fetch and update data.

Components/ Modules/ Interfaces:

- Frontend: React components for UI elements handling user registration, login, profile management etc
- Backend: Express.js for REST API endpoints.

Algorithms and Data Structures:

- Sorting and Filtering: Algorithms for task scheduling, prioritization and searching
- Data Structures: Tables for users, projects, tasks, files etc. Use of lists for task management and trees for project breakdown.

APIs:

- RESTful API for client-server communication

Languages/ Frameworks/ Tools:

- JavaScripts: Main programming language
- React.js: Frontend framework
- Redux: state management
- Nginx: Load Balancer
- Node.js: Backend runtime environment
- Express.js: Backend framework

Database:

- PostgreSQL for relational database management. Chosen for its robustness and support for complex queries.

Database Design:

- **Structure:** Tables for Users, Projects, Tasks. Messages, Files with appropriate relationships and indexes.
- **Entity-Relationship Diagram:** Visual representation of entities and their relationships
- **Schema Details:** Detail attributes for each table including data types.

Project Plan:

Project Management Tool: Trello
Project Timeline: 3 months
Meeting Plan: Weekly progress updates and issue resolution

Milestones:

- Milestone 1: Backend Development
- Milestone 2: Frontend Development
- Milestone 3: Testing and bug Fixing
- Milestone 4: Deployment and launch

Task List:

- Backend Development Tasks: API endpoints, database schema setup
- Frontend Development Tasks: UI design, client-server communication
- Testing tasks: Unit testing, Integration testing, End-to-end testing

Project Risks: Potential performance issues due to its complexity

Test Plan:

Testing Strategy:

- Unit testing: For individual components
- Integration testing: To ensure components work together as expected
- System Testing: To ensure entire application works as expected

Test Scenarios:

- User authentication and authorization
- Project creation and management
- Task creation and assignment
- File sharing and collaboration

Test Cases:

- Detailed steps to validate each scenario

Test Data:

- Sample data for testing different scenarios

Expected Result:

- Defined outcomes for each test case

Deployment:

Deployment Plan:

- Docker for containerization
- AWS for hosting
-

Deployment Procedures:

- Build Docker images
- Deploy container to AWS
- Build CI/CD pipeline

System Requirements:

- Linux-based server with sufficient CPU, memory, and storage resources