ToDoList System Design Specification

Table of Contents

1. Introduction

The **ToDoList Web Application** will help users manage tasks efficiently. It will provide features such as user authentication, task creation, editing, deletion, sorting, and reminders for task deadlines. The goal is to deliver a responsive, secure, and user-friendly web solution.

2. Scope

- **Target Users**: Individuals or professionals managing tasks and deadlines.
- **Features**:
  - User authentication (login, signup, password reset).
  - CRUD operations for tasks.
  - Task sorting and filtering.
  - Search functionality.
  - Notifications for task deadlines.
  - Profile management with image uploads.
- **Limitations**:
  - Focuses on individual task management, not team collaboration.
  - Requires internet connectivity.

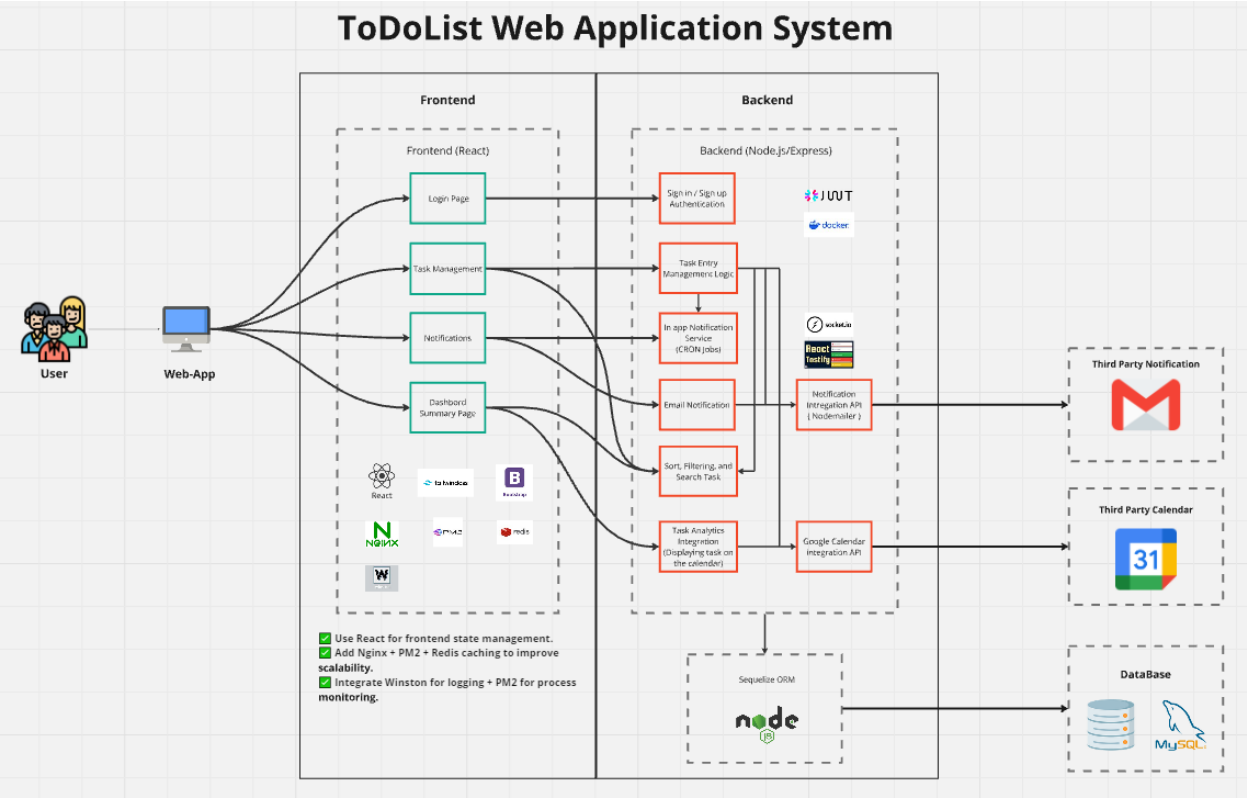
3. System Overview

The **web application** will follow a **client-server architecture**:

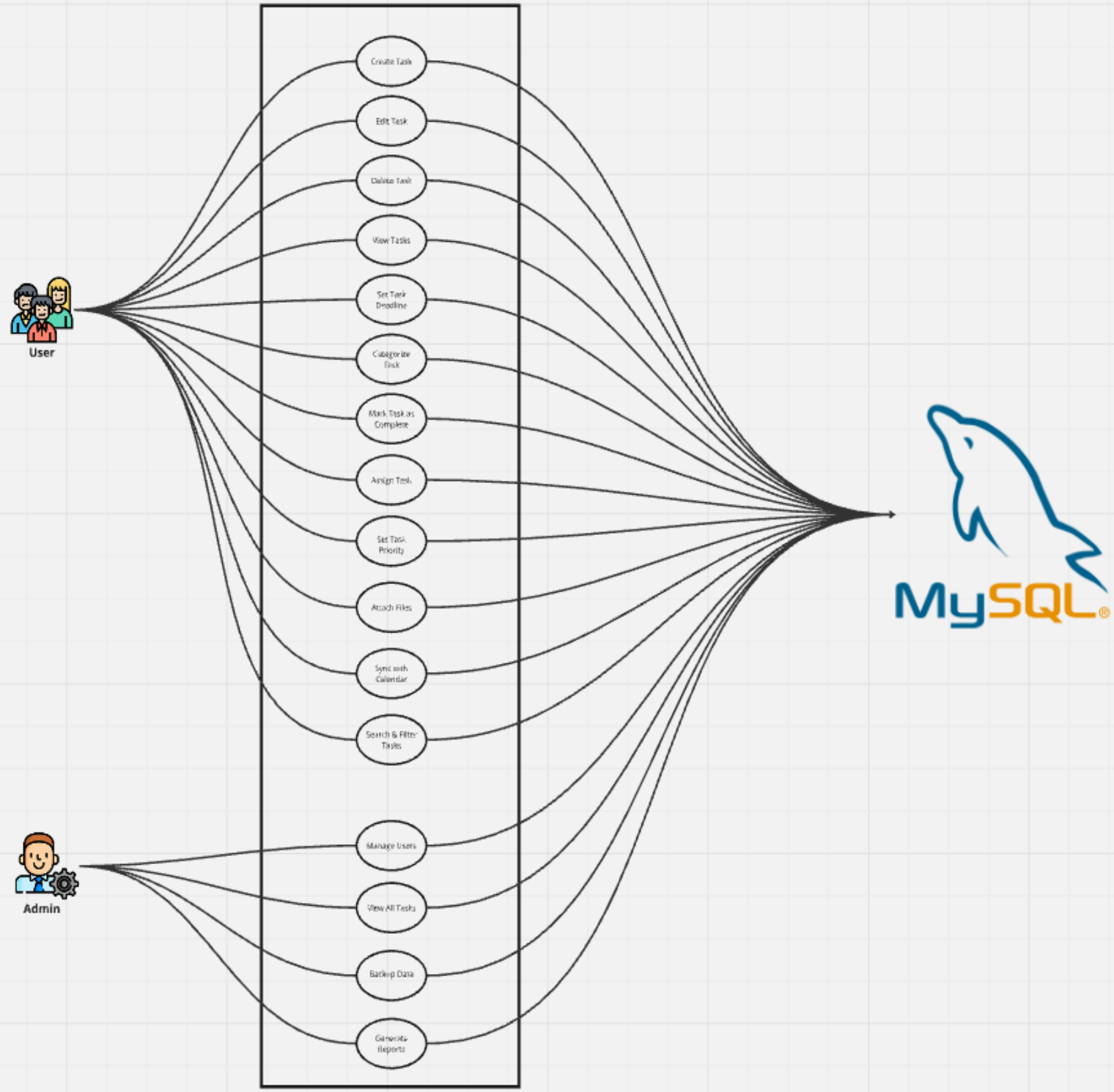
- **Frontend**: Built using React (or similar), focusing on responsiveness and usability.
- **Backend**: RESTful API (Node.js/Express or Django) for business logic.
- **Database**: PostgreSQL for relational data or MongoDB for flexibility.
- **Notification Service**: Push/email notifications for task reminders.

4. Architecture

- **Frontend**:
  - React or Angular for a responsive and interactive UI.
  - Axios or Fetch API for communication with the backend.
- **Backend**:
  - Node.js/Express or Django REST Framework.
  - Implements API for task management, authentication, and notifications.
- **Database**:
  - PostgreSQL for structured relational data or MongoDB for JSON-like flexibility.
- **Other Services**:
  - Elasticsearch for search functionality.
  - AWS S3/Cloudinary for profile image storage.



UML Diagrams



5. System Components

Frontend

- **UI Pages**:
  - Authentication (Login, Signup, Password Reset).
  - Dashboard (Task List, Sorting/Filtering options).
  - Notifications (Upcoming and overdue reminders).
  - Profile Management (Edit profile, upload image).
- **Libraries**:
  - Material-UI, TailwindCSS, or Bootstrap for styling. (Optional)
  - Framer Motion for animations. (Optional)

Backend

- **Core Modules**:
  - Authentication (JWT-based).
  - Task Management (CRUD operations).
  - Notification Service (Scheduled tasks).
  - Search Service (Indexing tasks).
- **Dependencies**:
  - Express/Django for API.
  - Sequelize/Prisma (PostgreSQL) or Mongoose (MongoDB) for ORM.

Database

- **Schema**:
  - **Users**:
    - id, email, password_hash, name, profile_picture.
  - **Tasks**:
    - id, user_id, title, description, due_date, priority, status.
  - **Notifications**:
    - id, user_id, task_id, reminder_time.

Search and Notifications

- **Search**:
  - Full-text search on task titles and filtering based on priority/due date.
- **Notifications**:
  - Use CRON jobs or Node.js libraries (Agenda.js) to schedule reminders.

6. Functional Requirements

- **User Authentication**:
  - Register, log in, log out, and reset passwords.
- **Task Management**:
  - Add, edit, delete, complete/incomplete status.
- **Sorting/Filtering**:
  - Sort tasks by due date or priority.

- **Search**:
  - Quick search by title or keyword.
- **Notifications**:
  - Email/push notifications for upcoming deadlines.
- **Profile Management**:
  - Edit profile and upload profile images.

7. Non-functional Requirements

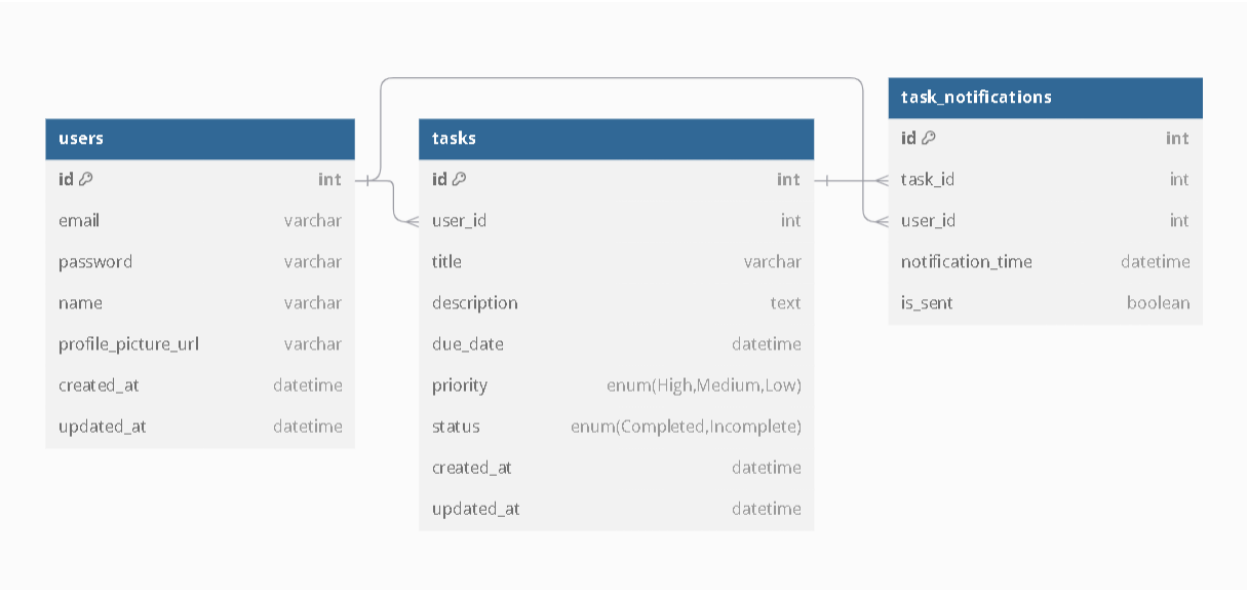
- **Scalability**:
  - Support multiple concurrent users with a scalable backend.
- **Performance**:
  - Ensure task queries execute in under 1 second.
- **Security**:
  - Encrypt passwords with bcrypt.
  - Use HTTPS for all data exchanges.
- **Usability**:
  - Responsive UI optimized for mobile and desktop users.

8. Database Design

1. **Users Table**
    a. Stores user information for authentication and profile management.
    b. **Columns**:
        - id (Primary Key)
        - email
        - password (hashed)
        - name
        - profile_picture_url (optional)
        - created_at
        - updated_at
2. **Tasks Table**
    a. Stores the tasks created by users.
    b. **Columns**:
        - id (Primary Key)
        - user_id (Foreign Key referencing Users)
        - title
        - description
        - due_date
        - priority (e.g., High, Medium, Low)
        - status (e.g., Completed, Incomplete)
        - created_at
        - updated_at
3. **Task Notifications Table**
    a. Tracks notifications/reminders for tasks.

b. **Columns**:

- id (Primary Key)
- task_id (Foreign Key referencing Tasks)
- user_id (Foreign Key referencing Users)
- notification_time (datetime for when the notification will be triggered)
- is_sent (boolean to mark whether the notification has been sent)



9. API Design

Authentication

- POST /api/register: Register new users.
- POST /api/login: Authenticate users and return JWT.
- POST /api/reset-password: Reset user password.

Tasks

- GET /api/tasks: Fetch all tasks for a user.

- POST /api/tasks: Create a new task.
- PUT /api/tasks/{id}: Update task details.
- DELETE /api/tasks/{id}: Delete a task.

Notifications

- GET /api/notifications: Get user reminders.

Profile

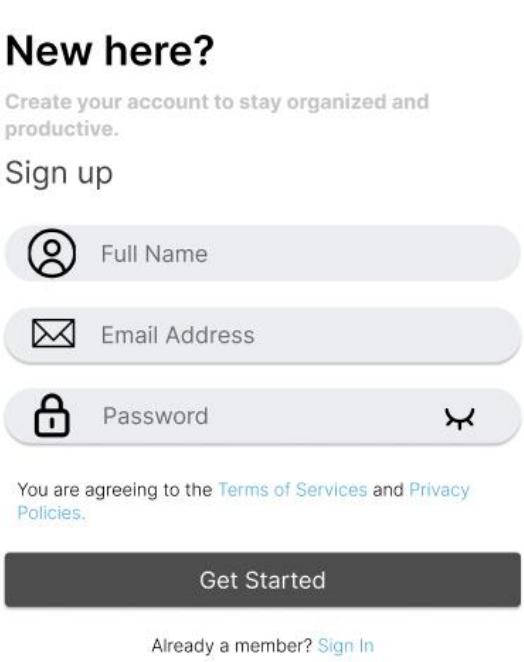
- GET /api/profile: Get user profile details.
- PUT /api/profile: Update profile details.

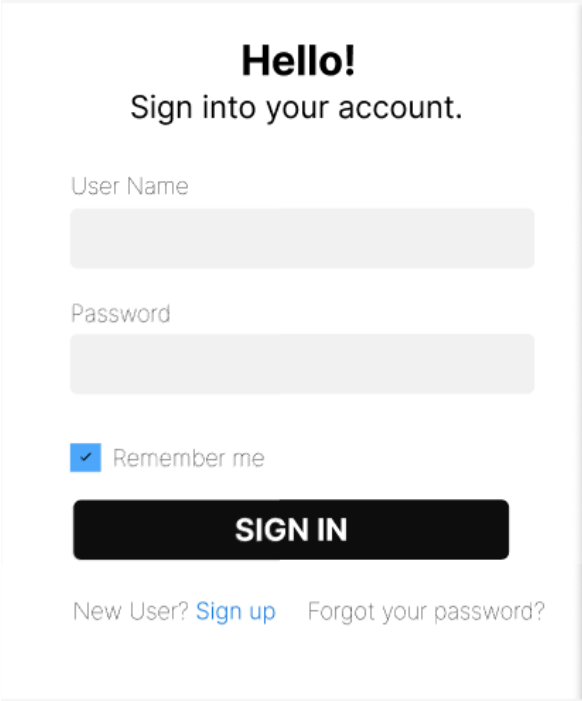
10. User Interface Design

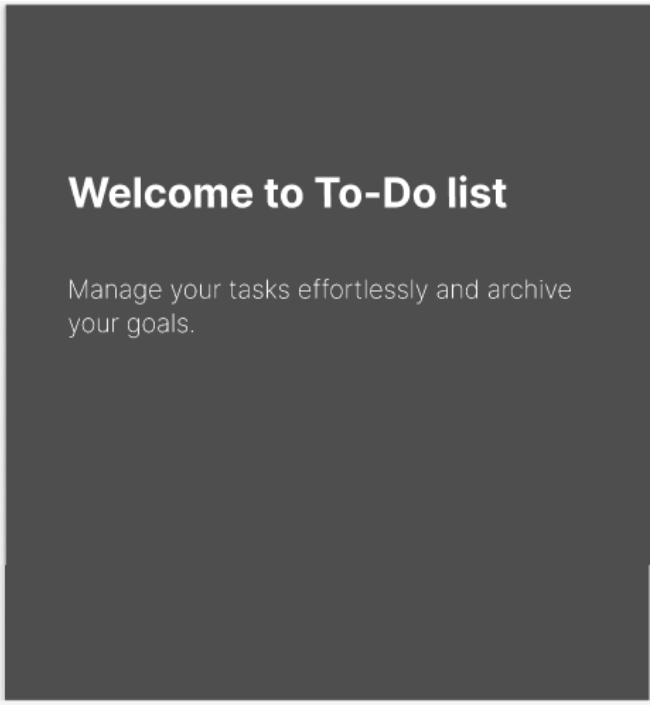
- **Login/Signup Page**:
  - Input fields for email/password.
  - Links for password reset.
- **Dashboard / Home Page**:
  - List of tasks with sorting and filtering options.
  - Buttons for adding, editing, and deleting tasks.
- **Category Page:**
  - Displays task categories, each containing related tasks.
  - Sorting and filtering options for better task organization.
  - Tasks grouped by status (e.g., Completed, In Progress,).
  - Buttons for adding, editing, deleting tasks, and managing categories.

- **Profile Page**:
    - Editable fields for user details.
    - Profile picture upload.

Sign In, Sign Up, and Forgot Password







Verification

Enter Verification Code

— — — —

If you didn't receive a code, Resend!

Verify

Forgot Password

Enter email address

example@gmail.com

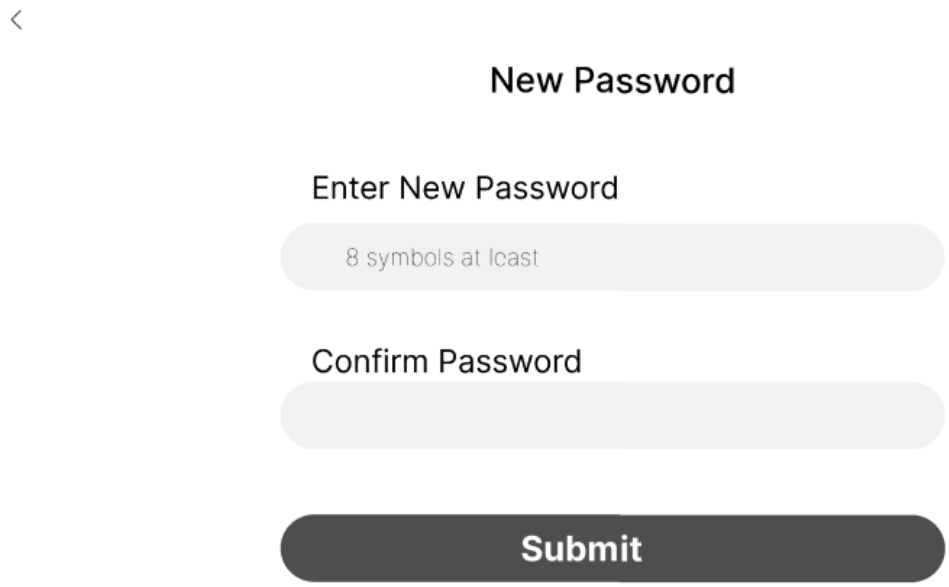
Back to Sign In

Send

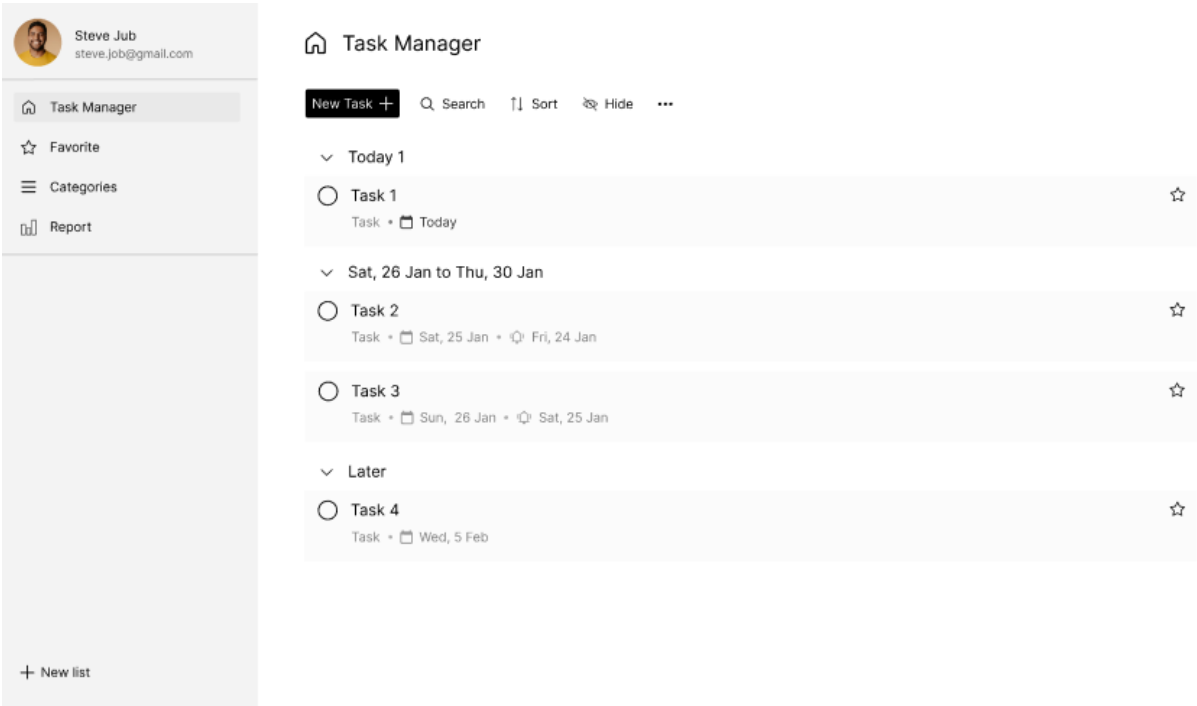
Or

Don't have an account?

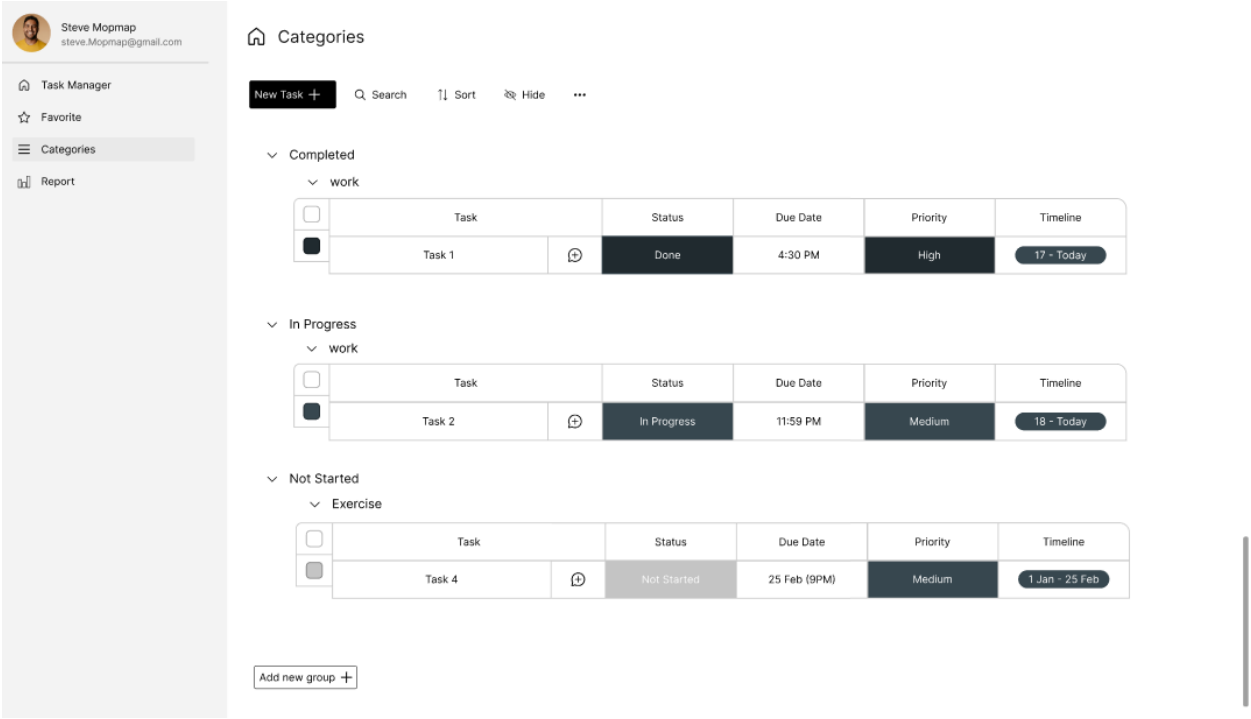
Sign Up


New Password
Enter New Password
8 symbols at least
Confirm Password
Submit

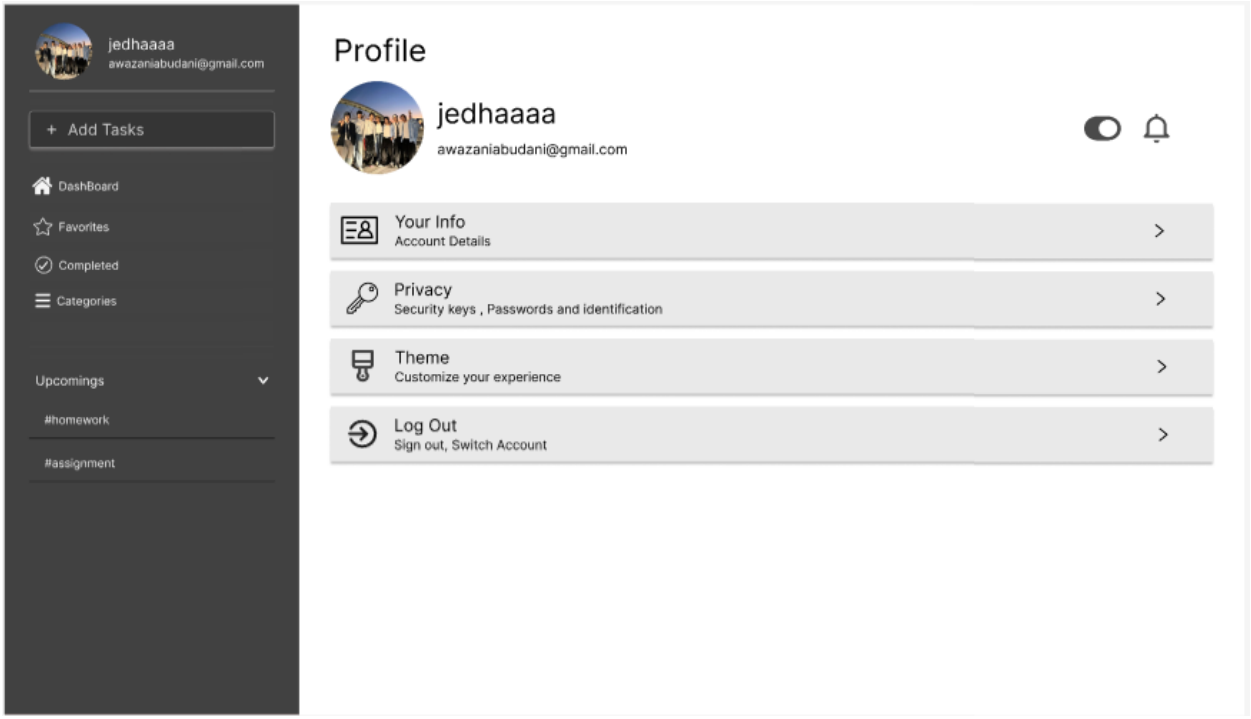
Home Page


Steve Jub
steve.job@gmail.com
Task Manager
Favorite
Categories
Report
New list
Task Manager
New Task
Search
Sort
Hide
Today 1
Task 1
Task • Today
Sat, 26 Jan to Thu, 30 Jan
Task 2
Task • Sat, 25 Jan • Fri, 24 Jan
Task 3
Task • Sun, 26 Jan • Sat, 25 Jan
Later
Task 4
Task • Wed, 5 Feb

Category Page



Profile Page



11. Security Considerations

- Use bcrypt for password encryption.
- Validate all inputs to prevent SQL Injection and XSS attacks.
- Secure APIs with JWT and implement role-based access control.

12. Deployment and Maintenance

- **Frontend**:
  - Deploy on **Vercel** or **Netlify**.
- **Backend**:
  - Host on **AWS EC2**, **Heroku**, or **Render**.
- **Database**:
  - Use managed services like **AWS RDS** or **MongoDB Atlas**.
- **Monitoring**:
  - Use tools like **New Relic** or **Sentry** for real-time performance monitoring.
- **Maintenance**:
  - Regular updates for libraries.
  - Database backups.