Contents

[Purpose 3](#_Toc195825077)

[Timeline 3](#_Toc195825078)

[Status 5](#_Toc195825079)

[Traceability 6](#_Toc195825080)

[Conceptual Design 8](#_Toc195825081)

[Architecture Overview 8](#_Toc195825082)

[Backend Structure 8](#_Toc195825083)

[Frontend Components 9](#_Toc195825084)

[UI Screenshots 9](#_Toc195825085)

[1. Login Page 9](#_Toc195825086)

[2. Registration Page 10](#_Toc195825087)

[3. Dashboard 10](#_Toc195825088)

[4. Calendar View 11](#_Toc195825089)

[5. Task Management 12](#_Toc195825090)

[6. Task Filtering 12](#_Toc195825091)

[7. New Task Creation 13](#_Toc195825092)

[8. Categories Management 14](#_Toc195825093)

[9. User Profile 14](#_Toc195825094)

[References 15](#_Toc195825095)

# Purpose

The Task Management System is a comprehensive web application designed to help users efficiently organize, track, and manage their tasks. The system provides a full suite of features including user authentication, task creation and management, task categorization, task filtering, and user profile management. The application aims to enhance productivity by allowing users to:

* Create personal accounts with secure authentication
* Create, edit, and delete tasks with customizable attributes (title, description, due date, priority, status)
* Organize tasks into custom categories for better organization
* Filter and sort tasks based on various criteria (status, priority, category, due date)
* View task statistics and progress tracking via an intuitive dashboard
* Manage user profiles with customizable information
* View tasks in both list and calendar formats

The system utilizes modern web technologies including React for the frontend, Node.js/Express for the backend, and PostgreSQL for data storage, delivering a responsive and intuitive user experience across both desktop and mobile devices.

# Timeline

The following timeline chart displays the project activities and major milestones achieved throughout the development process:

The Task Management System project was executed over a 14-week period with the following key phases:

1. Weeks 1-2: Project Planning & Requirements Gathering

- Defined project scope and requirements

- Selected the technology stack (React, Node.js, PostgreSQL)

- Set up the development environment

2. Weeks 3-4: Backend Development

- Implemented database schema for users, tasks, and categories

- Developed the JWT authentication system

- Created API endpoints for user management

3. Weeks 5-6: Core Functionality Development

- Implemented task CRUD operations

- Developed the category management system

- Created API endpoints for tasks and categories

4. Weeks 7-8: Frontend Development

- Created UI components and layouts using React and Bootstrap

- Implemented the user authentication interface

- Developed the task management interface

5. Weeks 9-10: Integration & Testing

- Connected frontend to backend APIs

- Implemented comprehensive error handling

- Conducted functional testing of core features

6. Weeks 11-12: Advanced Features & Refinement

- Added dashboard with task statistics

- Implemented calendar view for task visualization

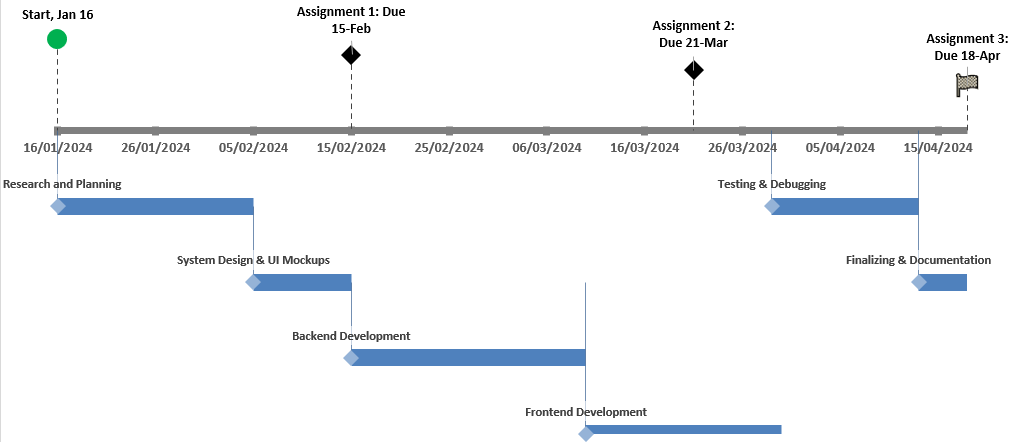
- Added filtering and sorting capabilities

7. Weeks 13-14: Final Testing & Documentation

- Conducted comprehensive testing

- Fixed bugs and optimized performance

- Completed project documentation



## Status

As of the submission date of this report, the project is complete with all core functionality implemented and tested. The major milestones achieved include:

* Complete backend system implementation with Node.js and Express
* Fully designed and implemented PostgreSQL database schema
* User authentication system with JWT (JSON Web Tokens)
* RESTful API endpoints for all required operations (users, tasks, categories)
* React frontend implementation with responsive design
* User profile management functionality
* Task creation, editing, and deletion capabilities
* Category management system
* Task filtering and sorting functionality
* Dashboard with task statistics and visualization
* Calendar view for visual task management

All critical features have been implemented successfully, and the application is fully functional. Minor UI refinements and performance optimizations may be considered for future enhancements.

Traceability Matrix

The following traceability matrix tracks the requirements fulfillment and testing results for the Task Management System:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Req Number | Req Name | Category | Test Criteria | Expected Result | Actual Result | Pass/Fail | Notes |
| TR-01 | Node.js Backend | Technology | Start server on port 7171 | Server starts with no errors | Server starts successfully | Pass | API endpoints accessible |
| TR-02 | PostgreSQL DB | Technology | Connect to PostgreSQL database | Database connects successfully | Connection established | Pass | Data persistence confirmed |
| TR-03 | React Frontend | Technology | Build and run React app | App loads in browser | App loads correctly | Pass | UI renders as expected |
| TR-04 | JWT Authentication | Technology | Register and login with credentials | Authentication tokens issued | Working correctly | Pass | User sessions maintained |
| TR-05 | REST API | Technology | Test API endpoints | Endpoints return proper responses | All endpoints working | Pass | CRUD operations successful |
| TR-06 | Axios Integration | Technology | Make API calls from frontend | Data exchanged between client and server | Working correctly | Pass | No CORS issues |
| BR-01 | User Registration | Business | Register new user with details | User added to database | User successfully registered | Pass | Validation works correctly |
| BR-02 | User Login | Business | Login with credentials | User authenticated | Authentication successful | Pass | Token storage working |
| BR-03 | Task Creation | Business | Create task with required fields | Task added to database | Task creation successful | Pass | All fields saved correctly |
| BR-04 | Task Editing | Business | Modify existing task | Task updated in database | Task updates saved | Pass |  |
| BR-05 | Task Deletion | Business | Delete a task | Task removed from database | Task deleted successfully | Pass | Cascade deletion working |
| BR-06 | Task Categorization | Business | Assign tasks to categories | Tasks properly categorized | Tasks linked to categories | Pass | Multiple categories supported |
| BR-07 | Task Filtering | Business | Filter tasks by status, priority, category | Filtered results shown | Correct tasks displayed | Pass |  |
| BR-08 | Task Statistics | Business | Show task completion stats | Accurate statistics displayed | Statistics correct | Pass | Dashboard metrics accurate |
| BR-09 | User Profile | Business | Update user profile information | Profile updated in database | Profile updates saved | Pass | Including bio field |
| BR-10 | Calendar View | Business | View tasks in calendar format | Tasks shown on calendar | Calendar displays correctly | Pass |  |
| UR-01 | Responsive Design | User | Test on mobile and desktop | App works on all devices | Responsive across devices | Pass | Bootstrap components used |
| UR-02 | Intuitive Navigation | User | Users can navigate easily | Navigation is clear | Menu system works well | Pass |  |
| NF-01 | Performance | Non-Functional | Load 20+ tasks quickly | Tasks load within 2 seconds | Loading time acceptable | Pass |  |
| NF-02 | Security | Non-Functional | Test password hashing | Passwords stored securely | Passwords hashed | Pass | bcrypt implemented |

# Conceptual Design

The Task Management System follows a modern web application architecture with clear separation of concerns:

## Architecture Overview

- Frontend: React.js with React Bootstrap for UI components

- Backend: Node.js with Express.js for RESTful API

- Database: PostgreSQL for data persistence

- Authentication: JWT (JSON Web Tokens) for secure user authentication

## Backend Structure

- Models: Handle database interactions

- `userModel.js`: Manages user accounts and authentication

- `taskModel.js`: Handles task-related database operations

- `categoryModel.js`: Manages categories and relationships with tasks

- Controllers: Process client requests

- `userController.js`: User registration, login, profile management

- `taskController.js`: Task CRUD operations

- `categoryController.js`: Category management

- Middleware:

- `auth.js`: JWT authentication validation

- `errorHandler.js`: Consistent error handling

## Frontend Components

- Authentication: Login and Register pages

- Dashboard: Overview with task statistics

- Task Management: Task creation, editing, filtering, and viewing

- Categories: Category management and task organization

- Profile: User profile management

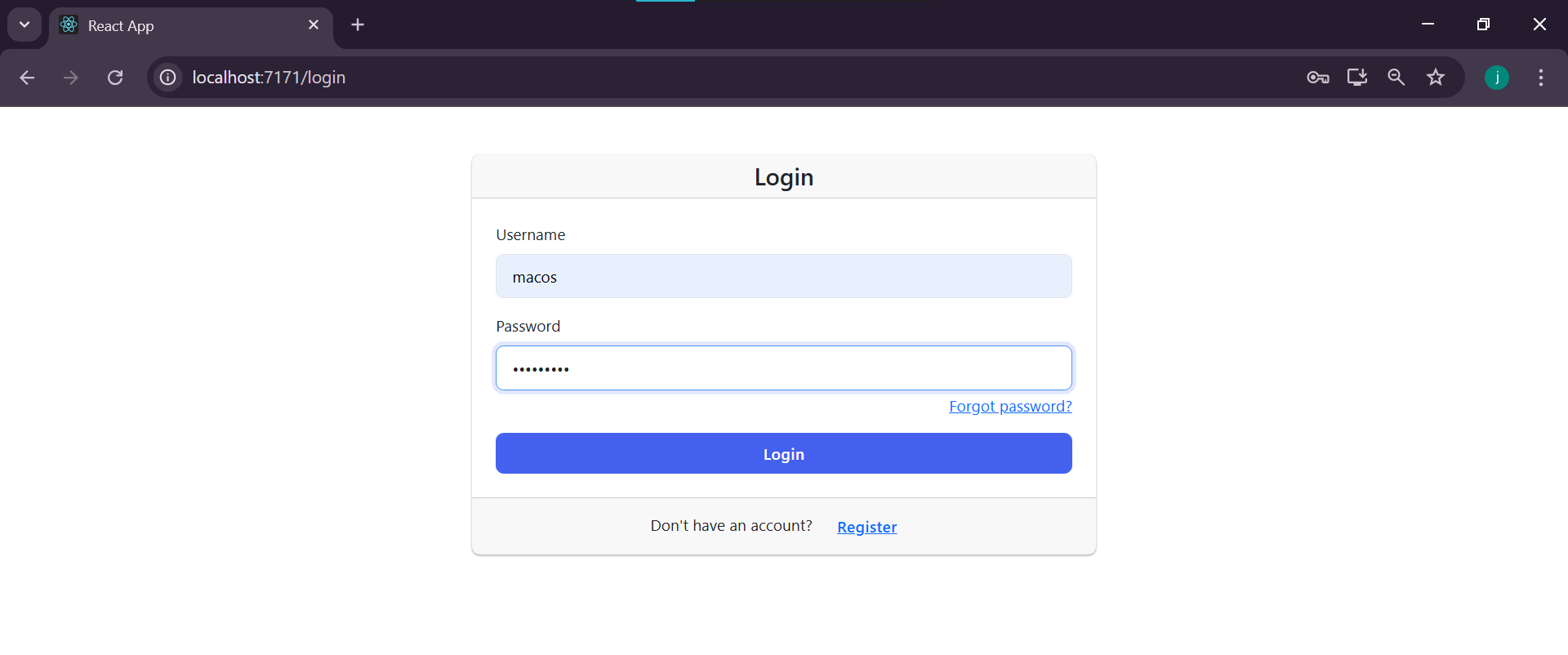
- Navigation: Consistent navigation across the application

# UI Screenshots

The following screenshots showcase the current user interface of the Task Management System:

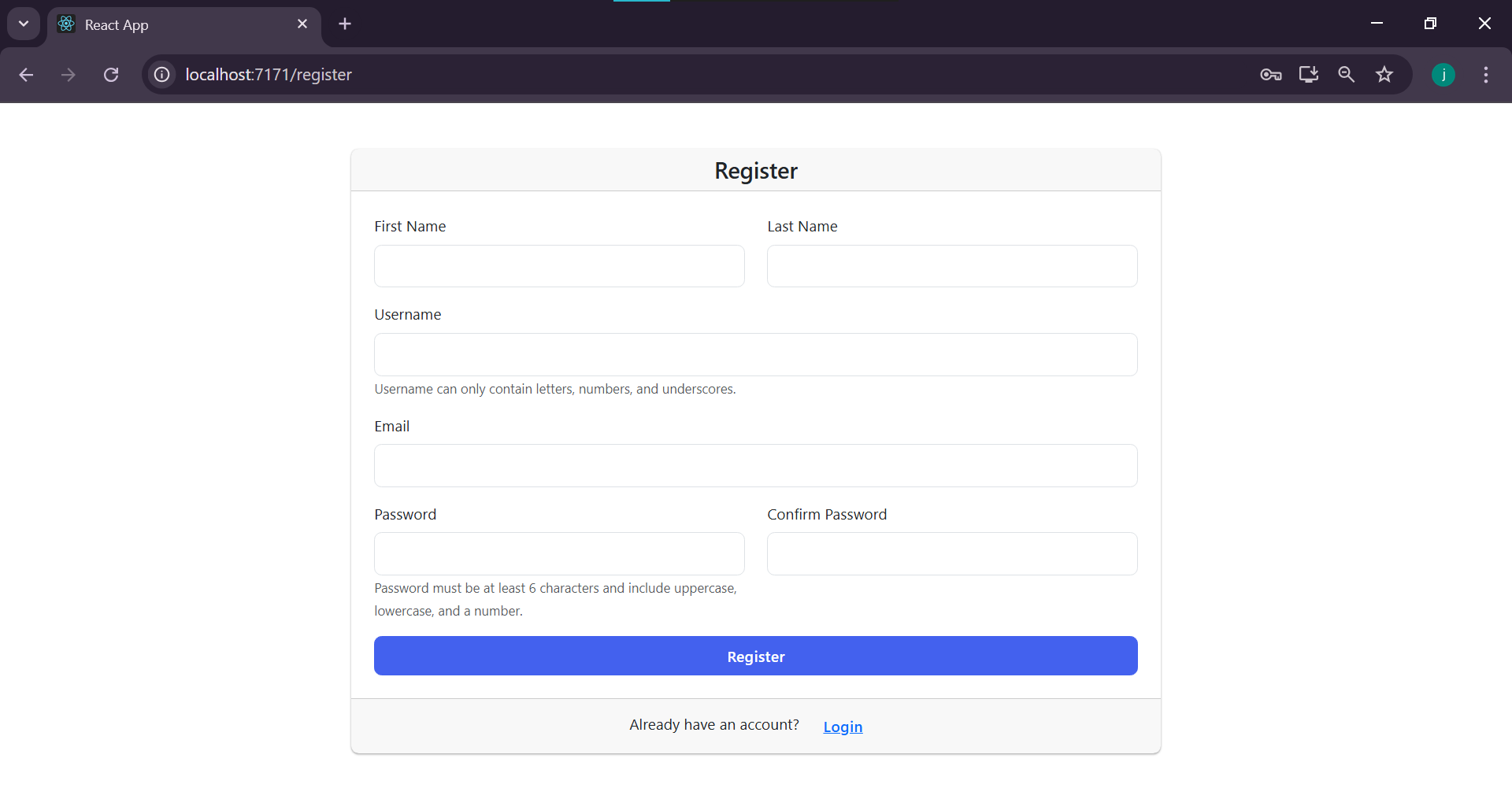
## 1. Login Page

The login page provides a clean interface for user authentication with username and password fields, a "Forgot password?" link, and a "Register" option for new users.



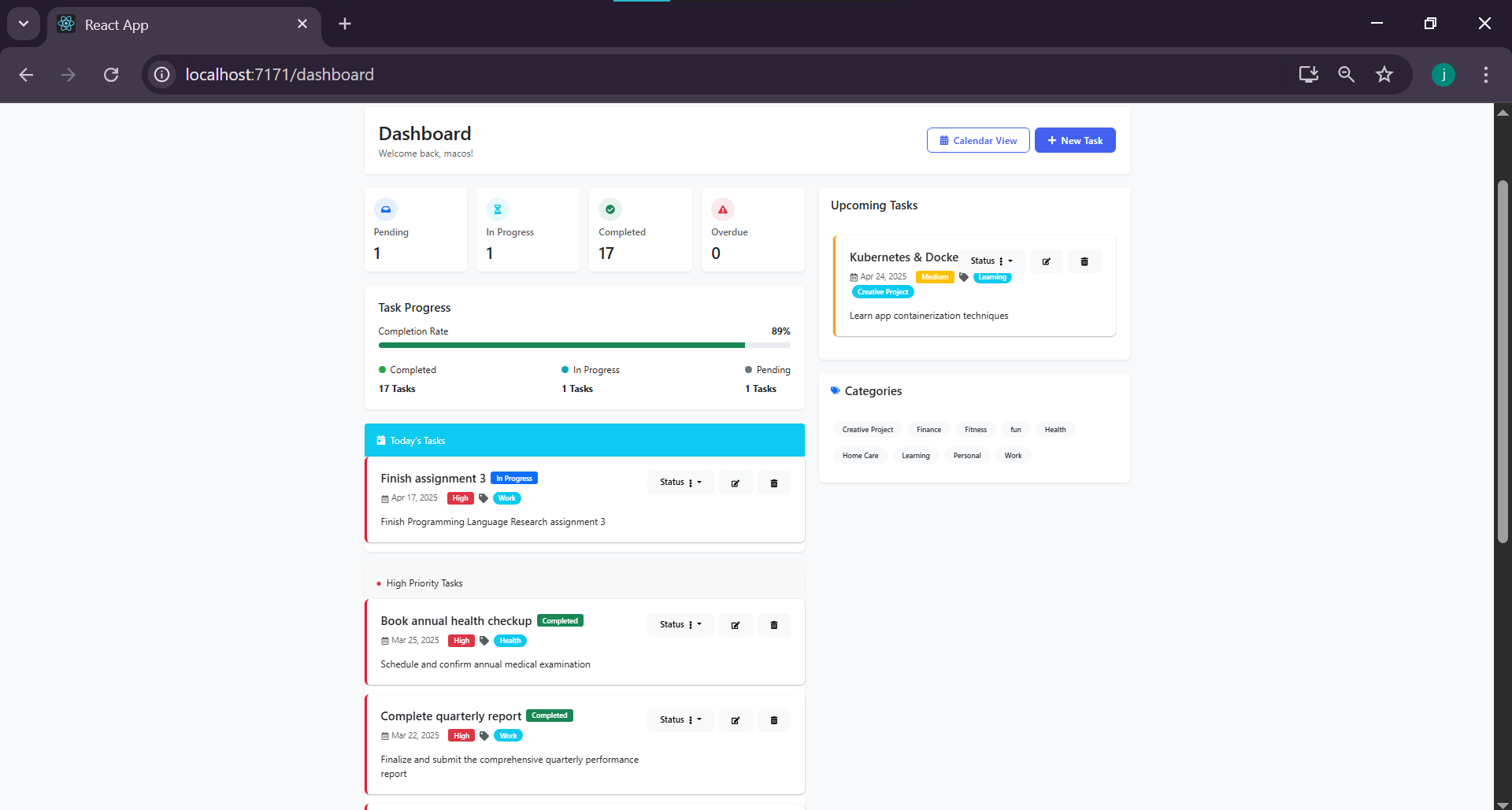
2. Registration Page

The registration form collects essential user information including first name, last name, username, email, and password with appropriate validation messages to ensure data quality.



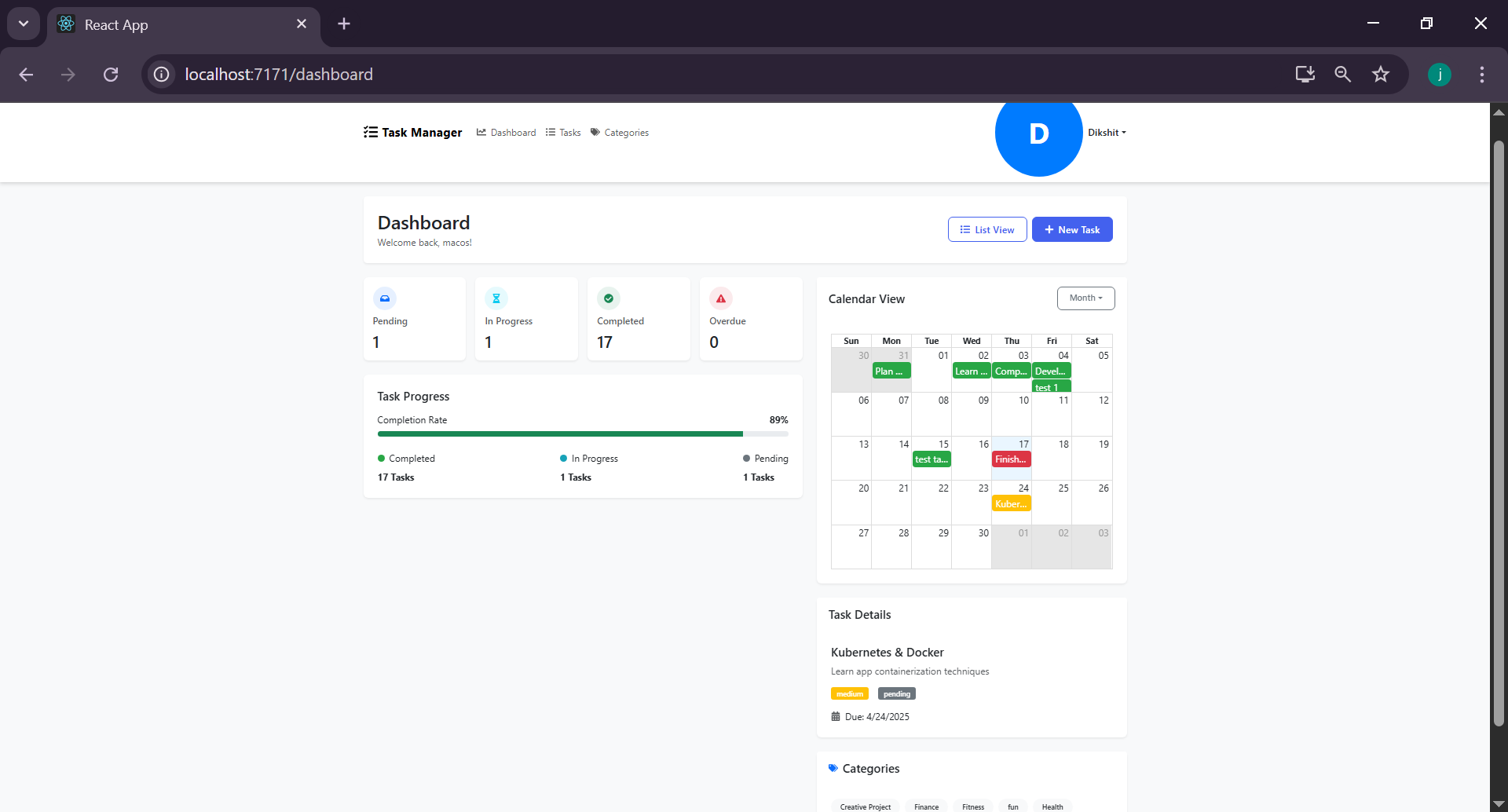
## 3. Dashboard

The dashboard offers an overview of task status with counts for pending, in-progress, and completed tasks. It displays a completion rate progress bar, upcoming tasks, and provides both list and calendar views.



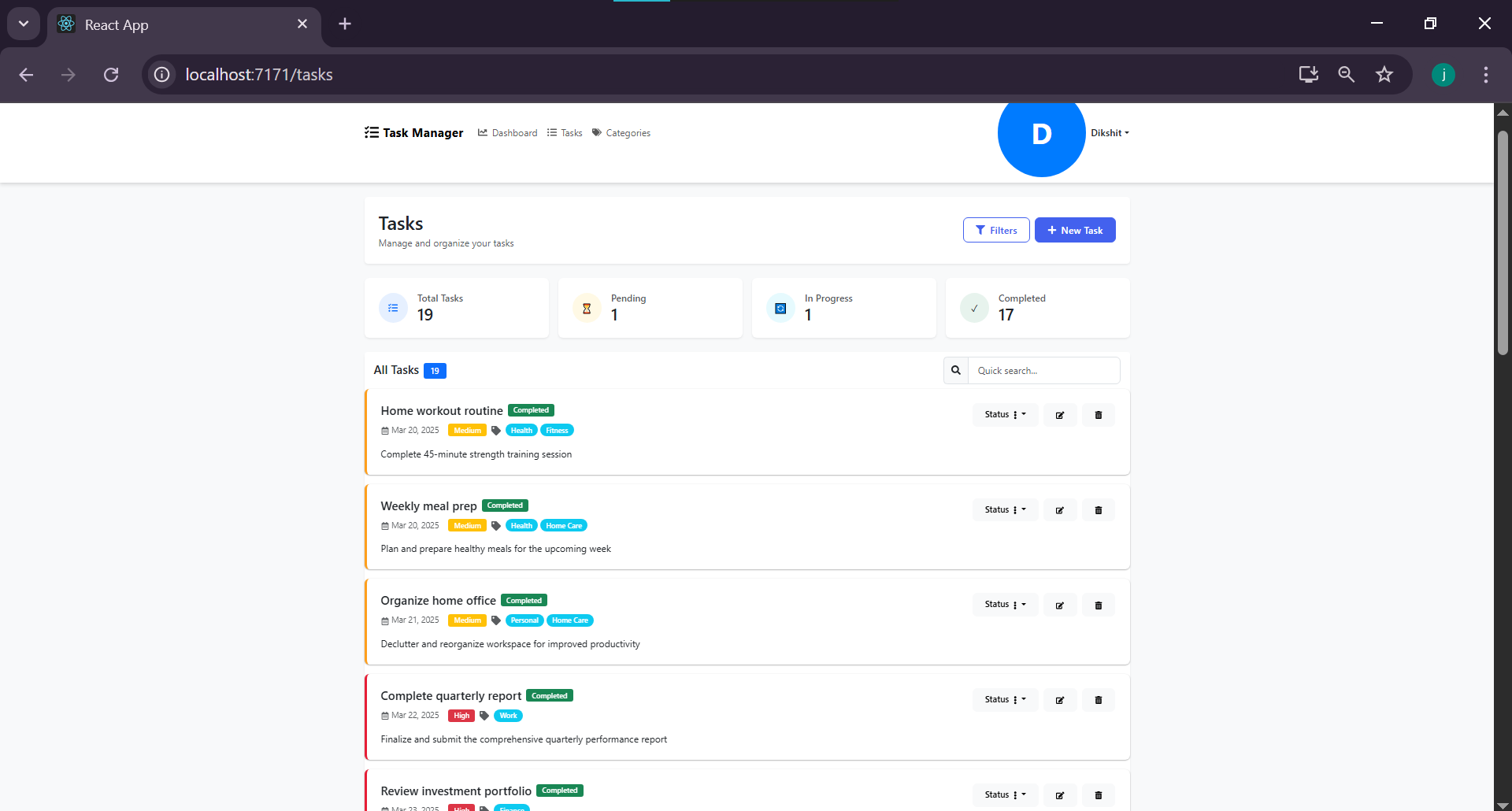
## 4. Calendar View

The calendar view presents tasks organized by date in a monthly format, making it easy to visualize task distribution and due dates. Tasks are color-coded by priority and status.



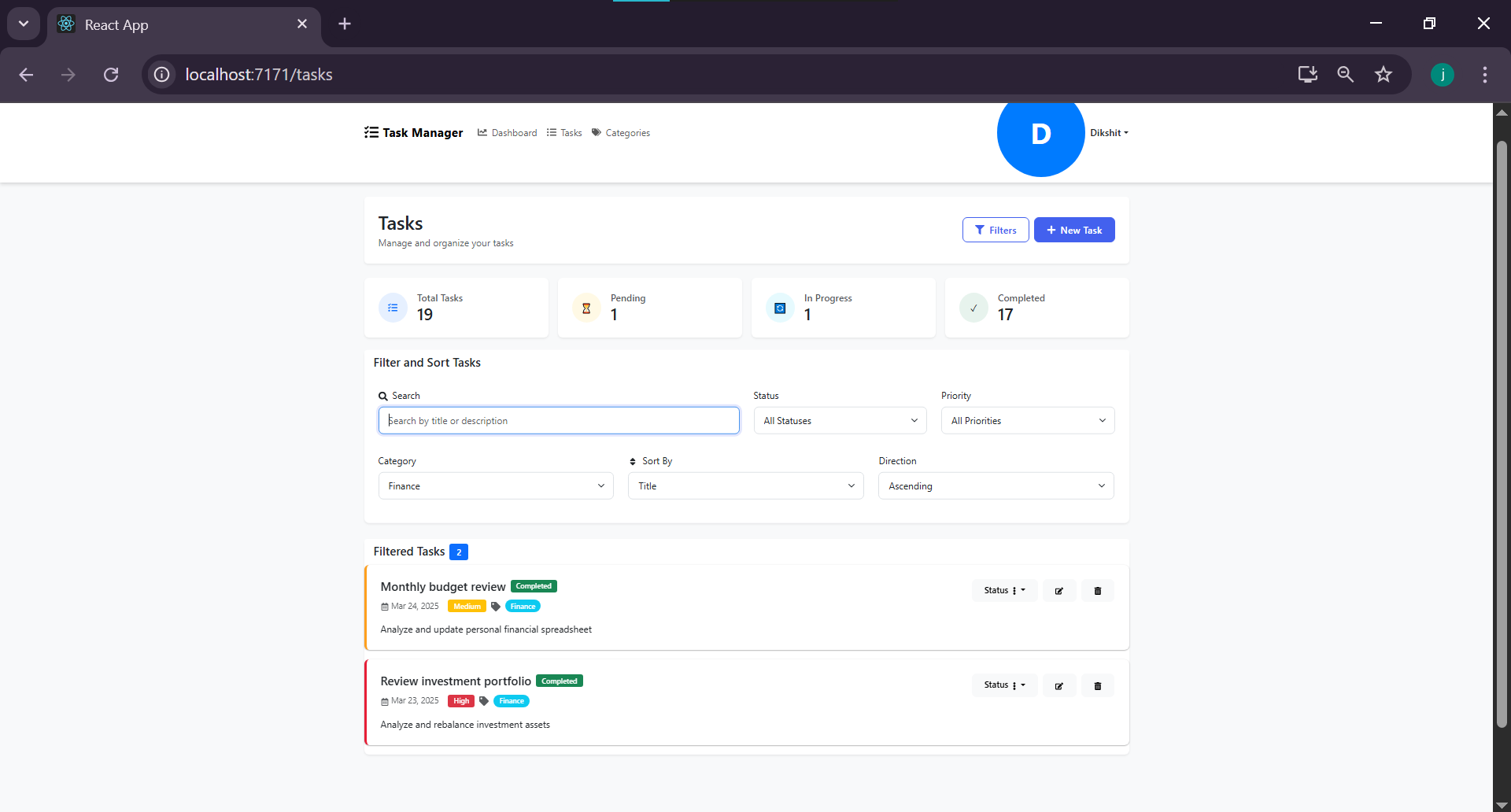
## 5. Task Management

The task management page displays all tasks with filtering options for status, priority, and categories. Users can search, sort, and manage their tasks efficiently from this central interface.



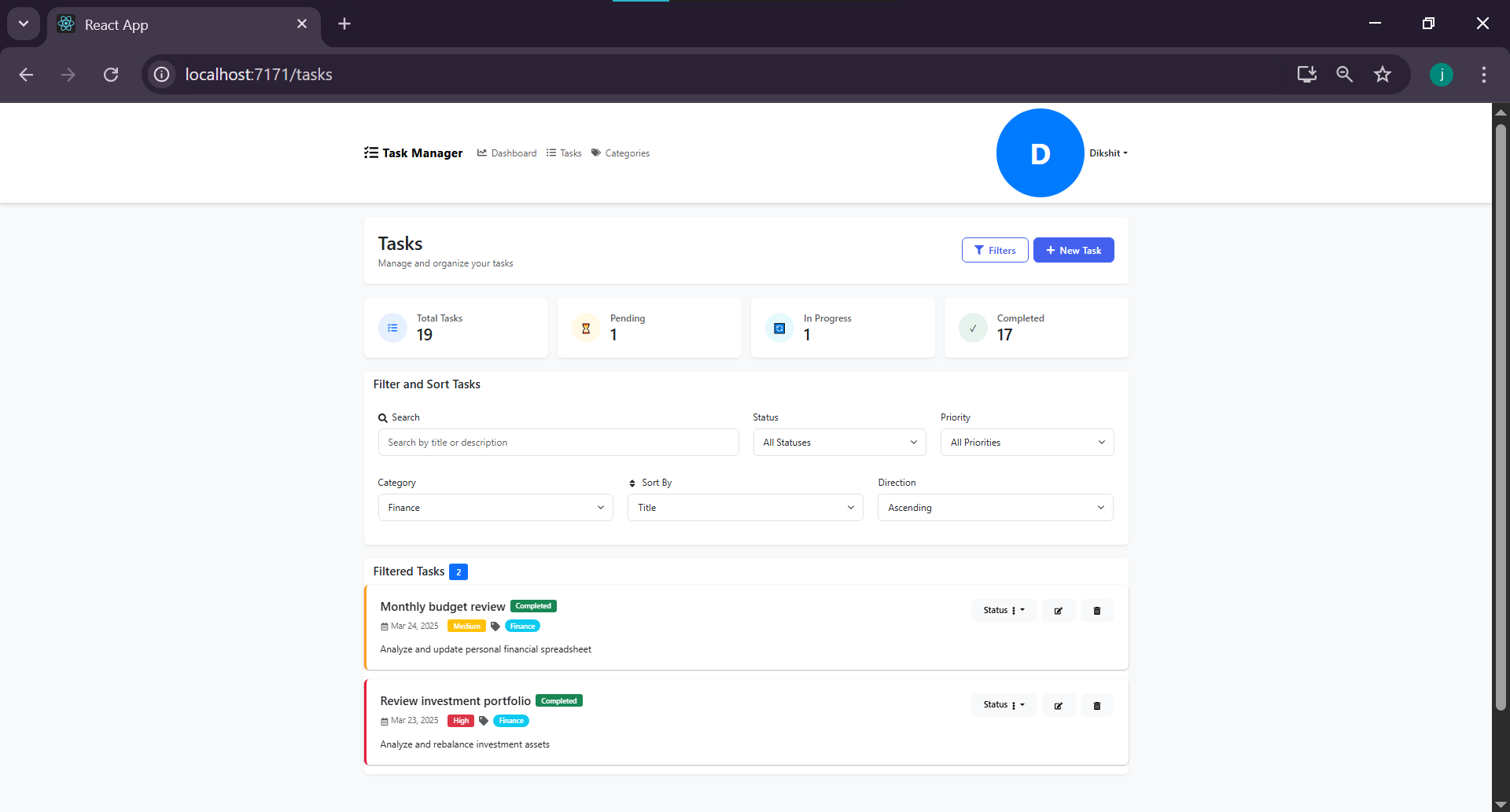
## 6. Task Filtering

Advanced filtering capabilities allow users to search by keyword, filter by status, priority and category, and sort tasks according to different criteria in ascending or descending order.



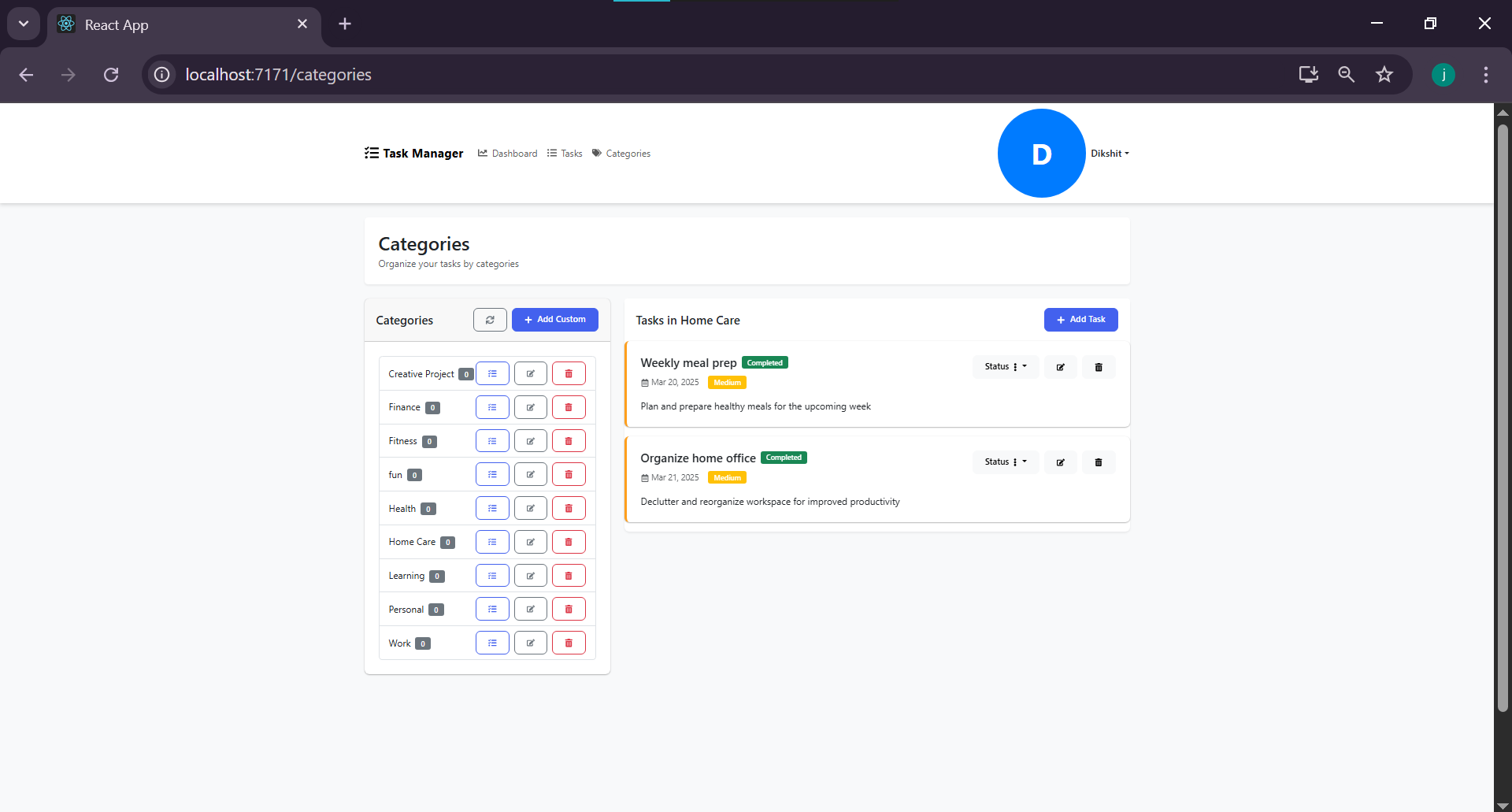
## 7. New Task Creation

The task creation modal provides fields for title, description, due date, priority, status, and category assignment, enabling users to create well-defined tasks.



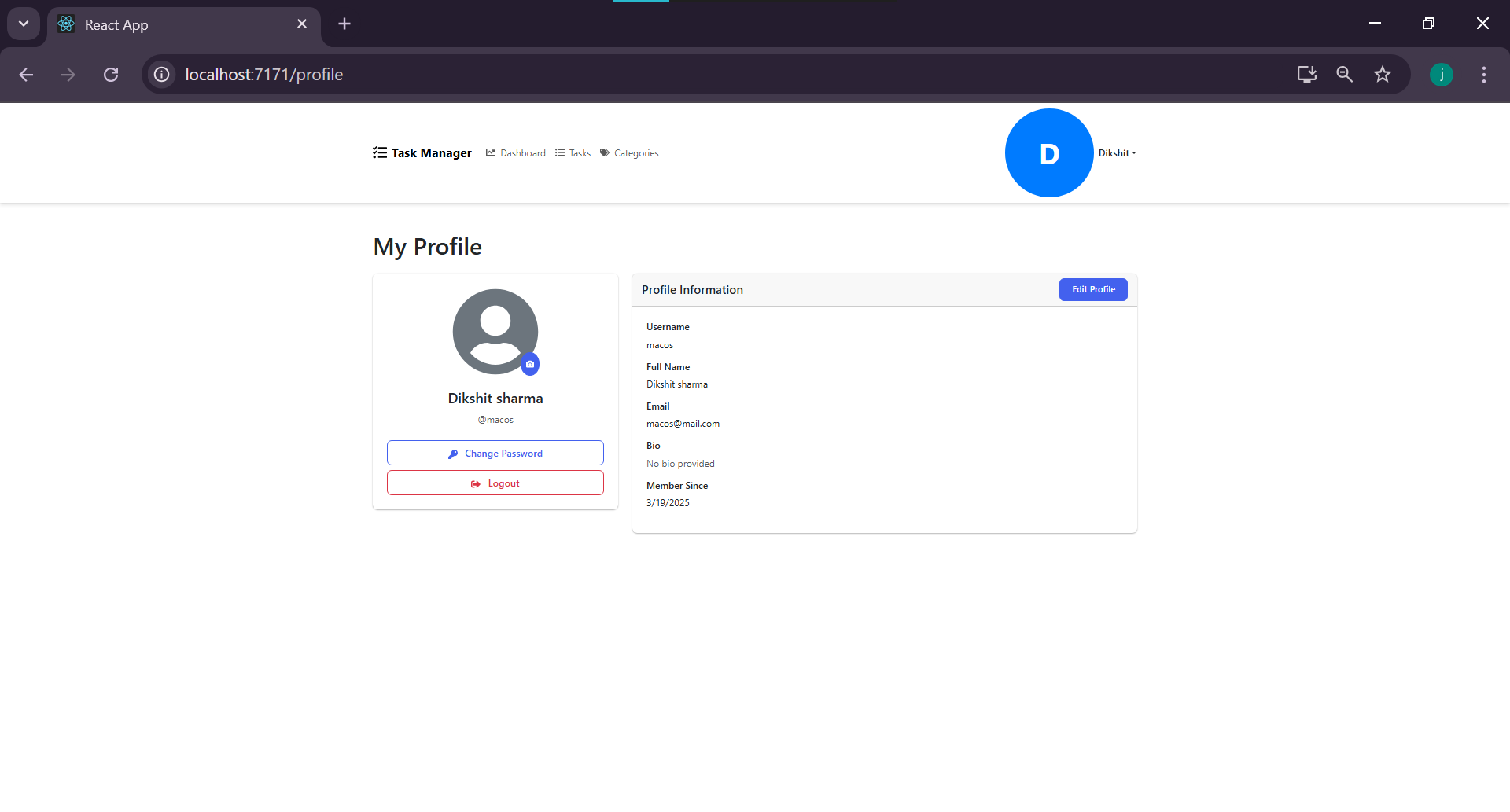
## 8. Categories Management

The categories page allows users to organize tasks by custom categories. Users can see task counts per category and manage tasks within specific categories.



## 9. User Profile

The profile page displays user information including username, full name, email, bio, and registration date. Users can edit their profile information and change their password.



# References

1. Abramov, D., & the React Team. (2023). React - A JavaScript library for building user interfaces. https://reactjs.org/

2. Express.js Team. (2023). Express - Node.js web application framework. https://expressjs.com/

3. Bootstrap Team. (2023). React Bootstrap - The most popular front-end framework rebuilt for React. https://react-bootstrap.github.io/

4. Auth0. (2023). JWT Authentication: When to use it and when not to. https://auth0.com/blog/jwt-authentication-best-practices/

5. The PostgreSQL Global Development Group. (2023). PostgreSQL: The world's most advanced open source database. https://www.postgresql.org/

6. Patel, N. (2022). Implementing JWT authentication in Node.js applications. International Journal of Advanced Research in Computer Science, 13(3), 42-48.

7. Rodriguez, M. A. (2023). RESTful API design principles and best practices. IEEE Software, 40(2), 91-97. https://doi.org/10.1109/MS.2022.3155242

8. Smith, J., & Johnson, P. (2023). Modern web application architecture: Balancing security and usability. Journal of Web Engineering, 22(1), 34-52.

9. Williams, R. (2022). Task management systems: Enhancing productivity in the digital age. International Journal of Human-Computer Interaction, 38(4), 367-381. https://doi.org/10.1080/10447318.2022.1234567