Task Management System – Technical Documentation

---

1. Overview

This Task Management System allows users to register, login, create, view, update, delete, and filter tasks. Each task is associated with a user and includes timestamps and status indicators.

---

2. Tech Stack

- Frontend: React, TypeScript, Redux Toolkit, React Bootstrap, Axios, React Router
- Backend: Node.js, Express.js, TypeScript, MongoDB (Mongoose)
- Auth: JWT (Access & Refresh tokens)
- Validation: Joi
- UI: Bootstrap, React Toastify
- State Persistence: redux-persist

---

3. Features

- User Registration & Login
- JWT-based Authentication (Access/Refresh Tokens)
- Task CRUD (Create, Read, Update, Delete)
- Task Filters (status-based, search, pagination)
- Logout and session management
- Persist login on page refresh using redux-persist
- Responsive UI with toast notifications

---

4. Backend Architecture

- Models:
  - User (username, email, password, timestamps, isLoggedIn)
  - Task (title, description, status, userId, timestamps)
  - RefreshToken (userId, token, expiry)
- Middlewares:
  - authMiddleware – Validates JWT and injects userId into req
- Services:
  - TokenService – Generates, verifies, saves, deletes tokens
  - ErrorHandler – Custom error messages
  - ResponseWrapper – Standardized HTTP responses

---

5. API Endpoints

Method	Route	Description	Protected
POST	/api/register	Register a new user	No

Method	Route	Description	Protected
GET	/api/login	Login user and return tokens	No
POST	/api/refresh-token	Refresh access token	No
GET	/api/me	Get authenticated user profile	Yes
POST	/api/logout	Logout user and revoke token	Yes
POST	/api/createTask	Create new task	Yes
GET	/api/getTasks	Get tasks (with filters)	Yes
GET	/api/getTaskById/:id	Get task by ID	Yes
PUT	/api/updateTask/:id	Update task	Yes
DELETE	/api/deleteTask/:id	Delete task	Yes

---

6. Authentication Flow

- On login, backend returns:
  - accessToken (expires in 1 day)
  - refreshToken (expires in 1 year, stored in DB)
  - Authenticated routes require accessToken in Authorization header.
- On token expiry:
  - Frontend catches 401/403
  - Sends refresh token to /api/refresh-token
  - Receives new access token and retries original request
- Logout:
  - Deletes refresh token
  - Updates isLoggedIn in user record

7. Frontend Overview

- Pages:
  - Login
  - SignUp
  - Dashboard (Task List)
  - Task Card View

- Features:
  - Task search by title/email
  - Pagination
  - Filter by status (Todo, InProgress, Completed)
  - View task detail and update status inline
  - Secure routing with ProtectedRoute wrapper

8. Redux State Management

- Slice: auth
  - accessToken
  - refreshToken
  - user
- Persisted using redux-persist to survive page reloads.
- Actions:
  - signInSuccess
  - signOutSuccess

9. Security Considerations

a) Passwords hashed with bcrypt
b) JWTs signed with secret keys
c) Refresh tokens stored securely in DB
d) Protected API routes with auth middleware

e) Input validation with Joi
f) HTTP-only cookie storage optional for further enhancement

10. Deployment & Environment Setup

.env example:

```
PORT=5000
MONGO_URI=mongodb://localhost:27017/task-manager
ACCESS_TOKEN_SECRET=your_access_secret
REFRESH_TOKEN_SECRET=your_refresh_secret
```

Start backend:

```
npm run dev
```

Start frontend:

```
npm run dev
```

11. Future Enhancements

- Role-based access (Admin/User)
- Multi-device refresh token handling
- Audit logs for task edits
- Task attachments (file uploads)
- Email notifications for due tasks
- Real-time updates with Socket.io