1. Project Overview

Full-Stack Task Manager Application

This is a full-stack task manager application built with React (frontend) and Node.js + Express (backend), connected to MongoDB Atlas.

It allows users to:-

- Register and log in securely

- Manage personal tasks with due dates and completion status

- Change password securely

- Reset forgotten passwords via email (Mailtrap integration)

- View their profile information

- Enjoy a responsive and clean UI with smooth user experience

The app demonstrates core concepts of authentication, authorization, state management, RESTful APIs, and scalable frontend architecture using React components, routing, and lazy loading.

This project is structured for clarity, performance, and future extensibility.

2. Features

User Authentication

- Register new users with name, email, and password
- Login using JWT authentication
- Forgot password flow with reset email via Mailtrap
- Change password securely while logged in

User Profile

- View personal information including user ID, email, and name
- Navigate to change password and back to dashboard

To-Do Task Management

- Add new tasks with title and due date
- Edit task title and due date
- Mark tasks as Complete/Incomplete
- Delete tasks
- Highlight tasks due today
- Sort tasks by due date, title, or status
- Filter by:-
    - Specific date
    - Date range
    - Status (All, Completed, Incomplete)

Pagination

- Paginated task list with 5 tasks per page
- Navigation controls for “Prev” and “Next”

Optimizations

- Lazy loading of major routes (Dashboard, Profile, Login, Register etc.)
- Clean state management with `useState` and `useEffect`
- Component extraction for reusability and scalability
- Form validation and graceful error handling

Tech & Tools

- Frontend: React + Vite + Tailwind CSS
- Backend: Node.js + Express
- Database: MongoDB Atlas (Cloud, AWS-backed)
- Version Control: Git + GitHub
- Code Formatting: Prettier
- Mailtrap for email testing

Cloud Integration

- Uses MongoDB Atlas (cloud-hosted MongoDB)
- Database is hosted on AWS cloud infrastructure
- Accessible from anywhere with proper credentials and security

3. Tech Stack & Tools Used

Layer	Technology/Tool	Notes
Frontend	React	Single Page Application with React Router
	Tailwind CSS	Utility-first CSS framework for fast, responsive UI
	Axios	For communicating with backend APIs
Backend	Node.js + Express	REST API server with secure endpoints
	JWT (jsonwebtoken)	Authentication + authorization
	Bcrypt.js	Secure password hashing
	dotenv	Manage environment variables
Database	MongoDB Atlas (Cloud)	Cloud-hosted database (MongoDB) running on AWS
Email Service	Mailtrap	Email testing for password reset functionality
Version Control	Git + GitHub	Code versioning, commits, branches
Formatting	Prettier	Automatically formats all frontend code (npx prettier --write .)
API Testing	Postman	Used to test backend APIs independently
Code Editor	Visual Studio Code (VS Code)	With extensions like ESLint, Prettier, Tailwind IntelliSense

4. Project Structure

Frontend (client/)

Built with React and Tailwind CSS, bootstrapped via Vite.

client/

- components/
  - TaskItem.jsx — Reusable component for rendering individual tasks
- pages/
  - ChangePassword.jsx — UI + logic for updating password (after login)
  - Dashboard.jsx — Main app page with task list and filters

o ForgotPassword.jsx	Sends reset link to user’s email via Mailtrap
o Login.jsx	User login page (auth + redirect)
o Register.jsx	User registration with name, email, password
o ResetPassword.jsx	Handles password reset using token from email
o UserProfile.jsx	Displays logged-in user’s email, ID, name
• App.jsx	Root component containing route layout
• main.jsx	React entry point + route setup
• App.css / index.css	Global styles
• tailwind.config.js	Tailwind CSS configuration
• vite.config.js / postcss.config.js	Build and processing config for Vite + PostCSS

Backend (backend/)

Built with Node.js, Express, MongoDB Atlas, and JWT authentication.

backend/

• middlewares/	
o authMiddleware.js	Verifies JWT tokens before accessing protected routes
• models/	
o Task.js	Task schema (title, due date, isCompleted, user ref)
o User.js	User schema (name, email, password, resetToken)
• routes/	
o authRoutes.js	Handles register, login, forgot/reset/change password
o taskRoutes.js	CRUD operations for tasks (GET, POST, PUT, DELETE)
o userRoutes.js	Fetches authenticated user info (ID, name, email)
• utils/	
o sendEmail.js	Sends password reset email via Mailtrap
• server.js	Main Express app entry point, connects to MongoDB
• .env	Environment variables (e.g., DB URI, JWT secret)
• .env.example	Template for setting up .env

5. Project Setup

Here’s how to clone, install, and run both the frontend and backend locally.

Prerequisites

- Node.js (v16+)
- MongoDB Atlas or local MongoDB
- Mailtrap account for email testing (or any test SMTP server)
- Git (optional but recommended)

- **Clone the Repository**

  ```
  git clone <your-repo-url>
  cd your-project-folder
  ```

- **Backend Setup**

  ```
  cd backend
  ```

- Install dependencies

  ```
  npm install
  ```

- Create .env file

  ```
  cp .env.example .env
  ```

- Update the following in .env

  ```
  MONGODB_URI=<Your MongoDB URI>
  JWT_SECRET=<YourSecretKey>
  CLIENT_URL=http://localhost:5173
  MAIL_HOST=smtp.mailtrap.io
  MAIL_PORT=2525
  MAIL_USER=<YourMailtrapUsername>
  MAIL_PASS=<YourMailtrapPassword>
  ```

- Run the server

  npm run dev

  Server will run on http://localhost:5000.

  Frontend Setup

  cd client

- Install dependencies

  npm install

- Run the app

  npm run dev

  Frontend runs on http://localhost:5173/login

Important Notes

- Backend connects to MongoDB Atlas (hosted on AWS).
- Auth-protected routes use JWT stored in localStorage.
- Tailwind CSS handles styling and responsiveness.
- Make sure backend and frontend are running simultaneously.

6. Environment Variables

To ensure the application works correctly, both the backend and frontend require certain environment variables.

Backend (/backend)

Create a file named .env in the root of the backend folder (you can copy .env.example if available).

Required Environment Variables:

```
PORT=5000

MONGO_URI=your-mongodb-atlas-uri

JWT_SECRET=your-jwt-secret-key
```

```
CLIENT_URL=http://localhost:5173

# Mailtrap SMTP details for forgot-password email
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USER=your-mailtrap-username
MAIL_PASS=your-mailtrap-password
```

If you're using MongoDB Atlas, make sure to whitelist your IP and allow connections from anywhere (0.0.0.0/0) during development.

Frontend (/client)

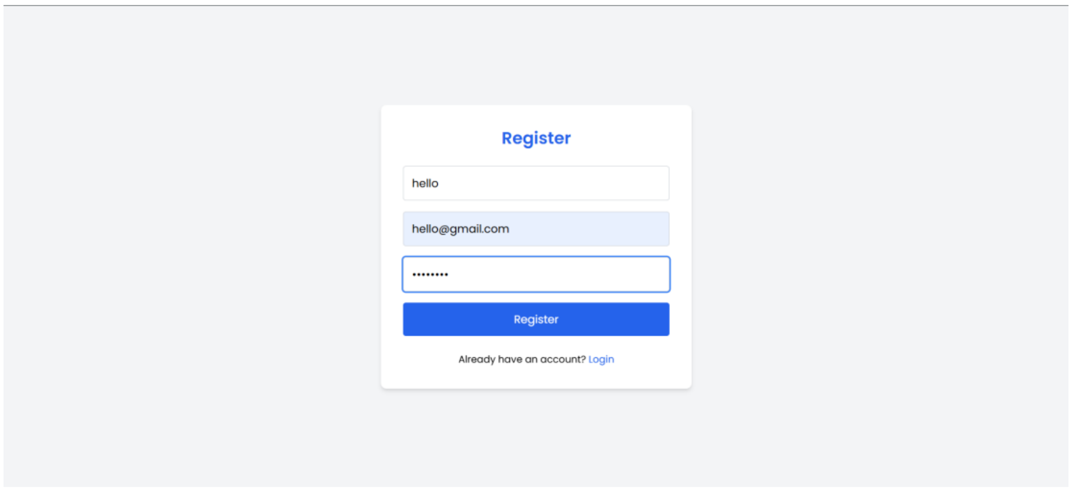
No .env file is currently required for the frontend, as the API base URL is hardcoded (http://localhost:5000) in Axios requests.

7. Testing & Usage Instructions

This section provides a step-by-step guide for testing all core features of the application.

1. User Registration

- Navigate to /register
- Fill in **name**, **email**, and **password**
- Submit to create a new user
- You'll be redirected to the login page on success



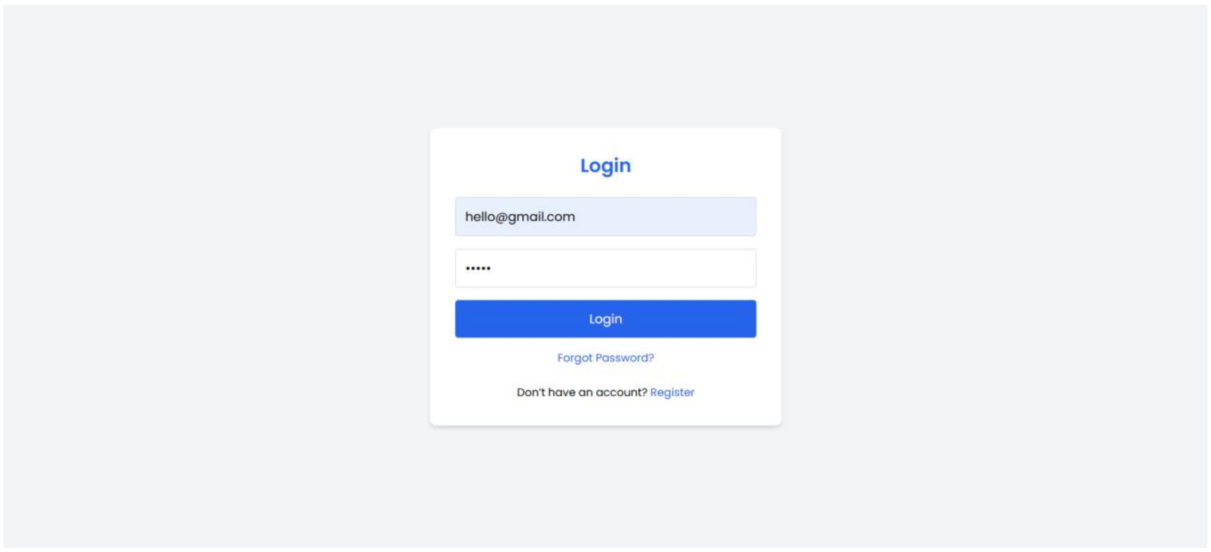
hello

hello@gmail.com

••••••••

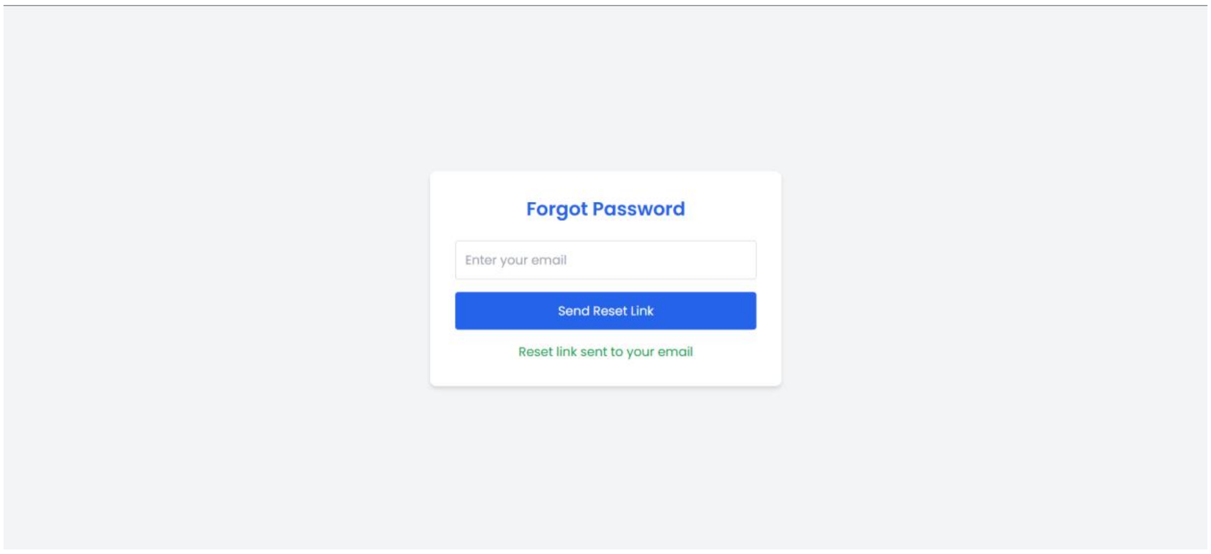
2. User Login

- Navigate to /login
- Enter your registered **email** and **password**
- On success, you'll be redirected to the dashboard

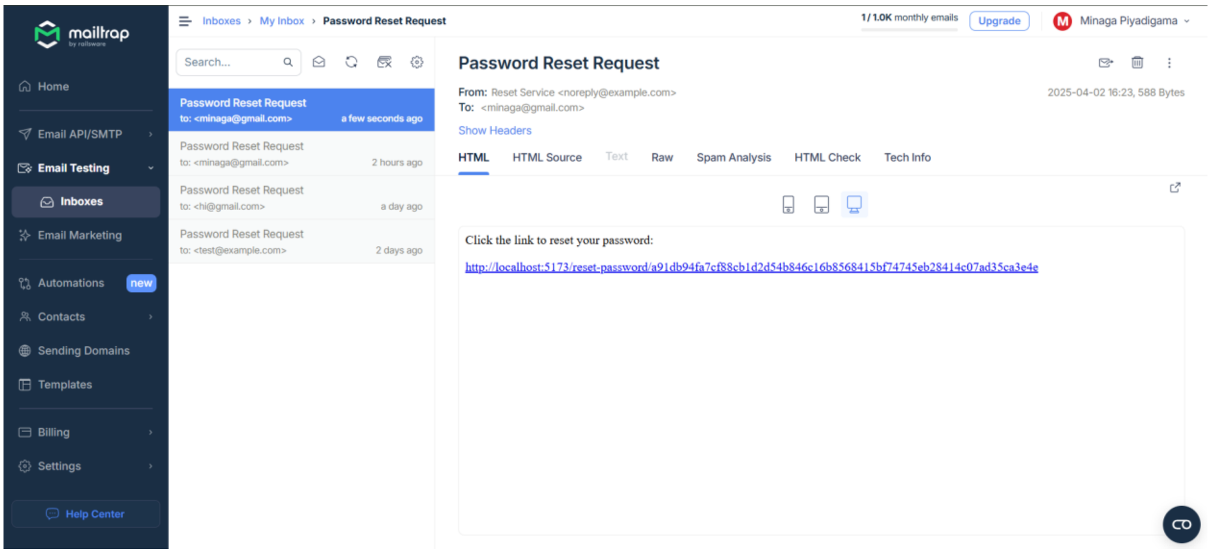


3. Forgot Password Flow

- On the login page, click "Forgot Password?"
- Enter your email to receive a reset link (sent via Mailtrap)
- Open Mailtrap -> find the reset email -> click the link
- Set a new password to complete the reset process



Reset link sent to your email



Password Reset Request
to: <minaga@gmail.com> a few seconds ago

Password Reset Request
to: <minaga@gmail.com> 2 hours ago

Password Reset Request
to: <hi@gmail.com> a day ago

Password Reset Request
to: <test@example.com> 2 days ago

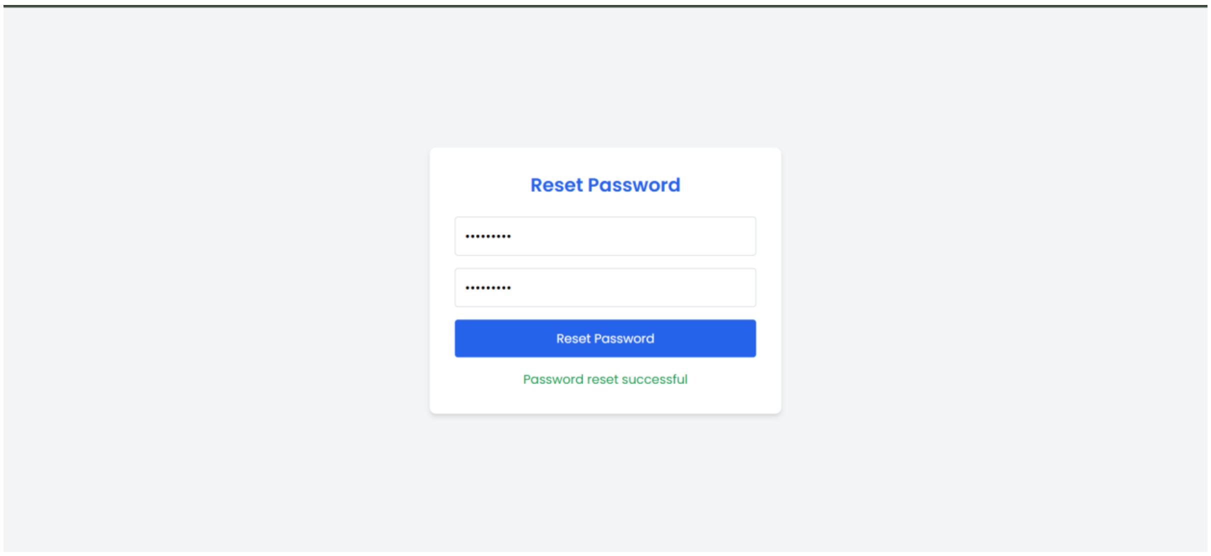
Password Reset Request

From: Reset Service <noreply@example.com>
To: <minaga@gmail.com>

2025-04-02 16:23, 588 Bytes

Click the link to reset your password:

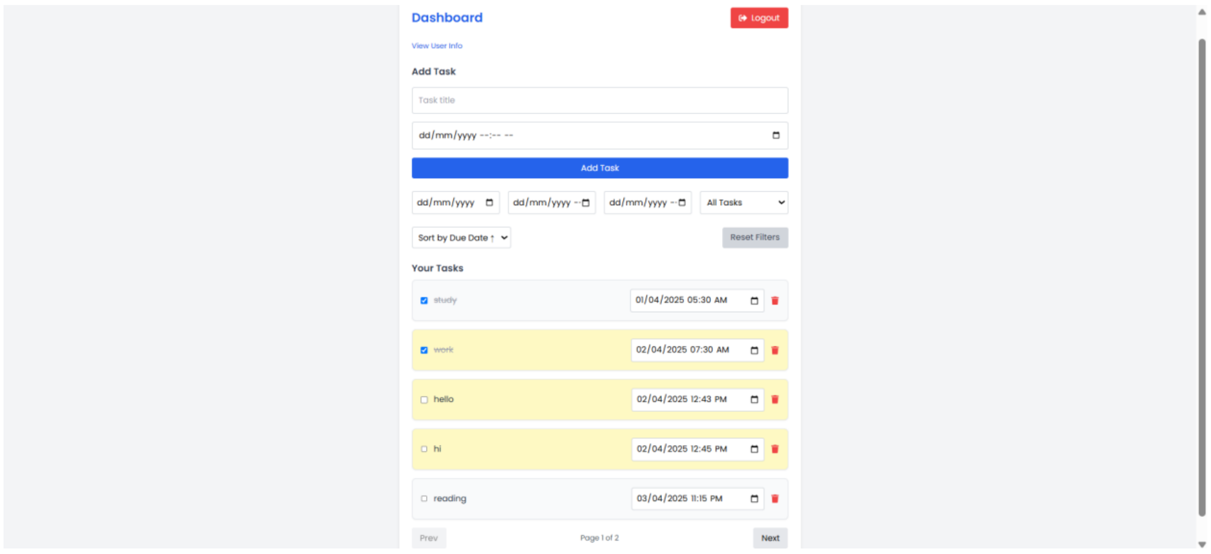
http://localhost:5173/reset-password/a91db94fa7cf88cb1d2d54b846c16b8568415bf74745eb28414c07ad35ca3e4e



4. Dashboard Features

- After login, you will land on the /dashboard
- Add tasks with **title** and **due date**
- Tasks will show:
  - Checkbox to mark complete/incomplete
  - Inline editing of task title
  - Delete button
  - Inline due date editing
- Filter tasks by:
  - A specific date
  - Date range (From, To)
  - Status (Completed / Incomplete)

- Sort tasks by:
    - Due date
    - Title
    - Status
- Highlight today’s tasks in yellow
- Pagination: View 5 tasks per page with Next/Prev controls



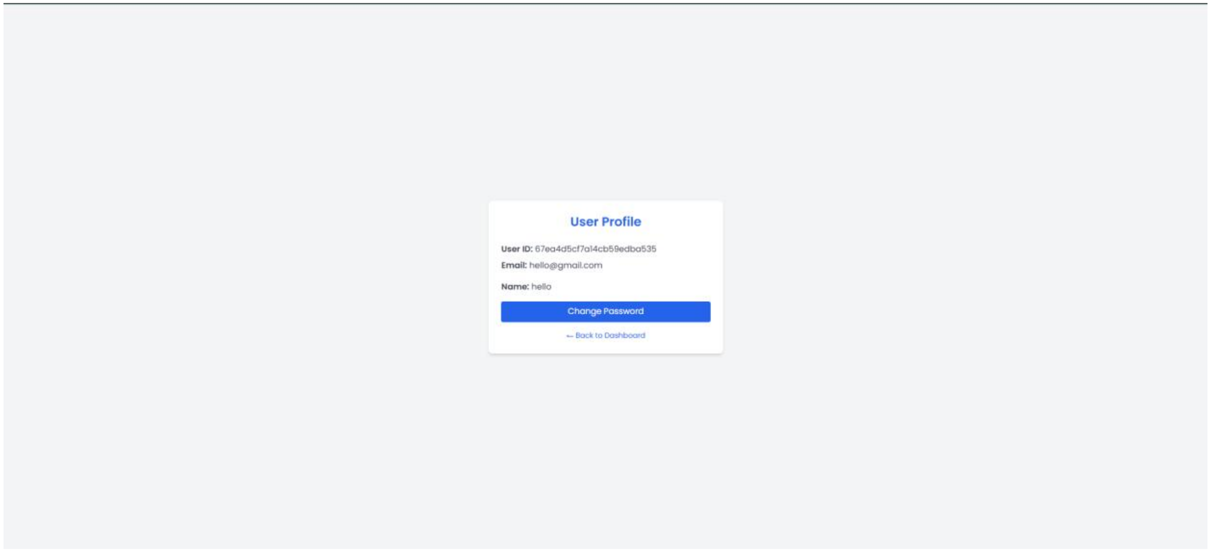
Your Tasks

✓ ~~study~~	01/04/2025 05:30 AM
✓ ~~work~~	02/04/2025 07:30 AM
hello	02/04/2025 12:43 PM
hi	02/04/2025 12:45 PM
reading	03/04/2025 11:15 PM

Page 1 of 2

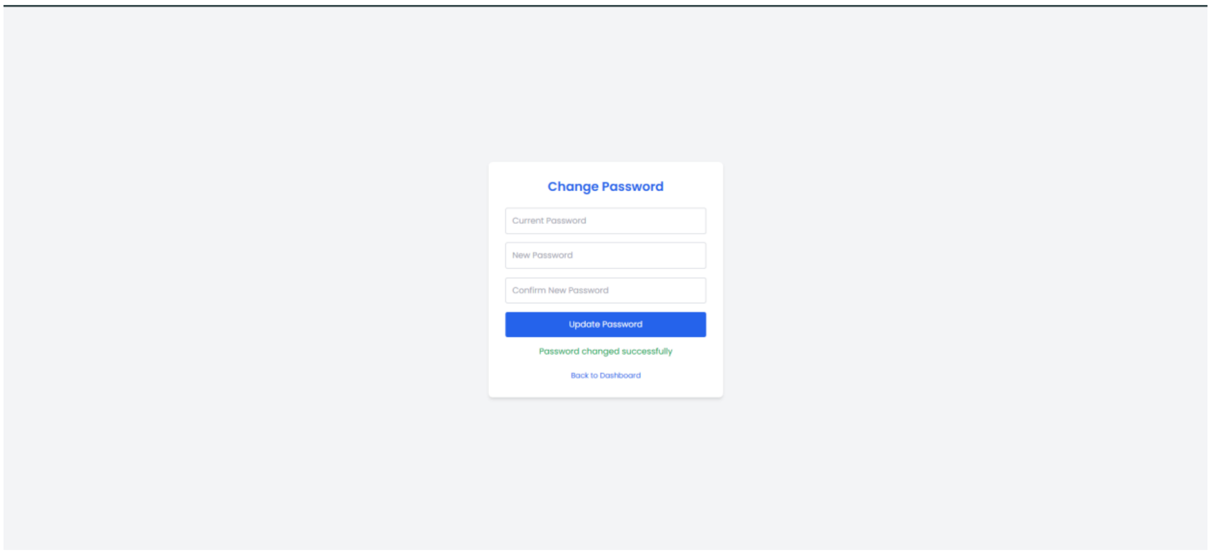
5. View Profile & Change Password

- On dashboard, click “View User Info” to go to /profile
- Shows your user ID, name, and email
- Click “Change Password” to go to /change-password
- Requires current password + new password to update



User Profile

User ID: 67ea4d5cf7a14cb59edba535
Email: hello@gmail.com
Name: hello



Change Password

Current Password

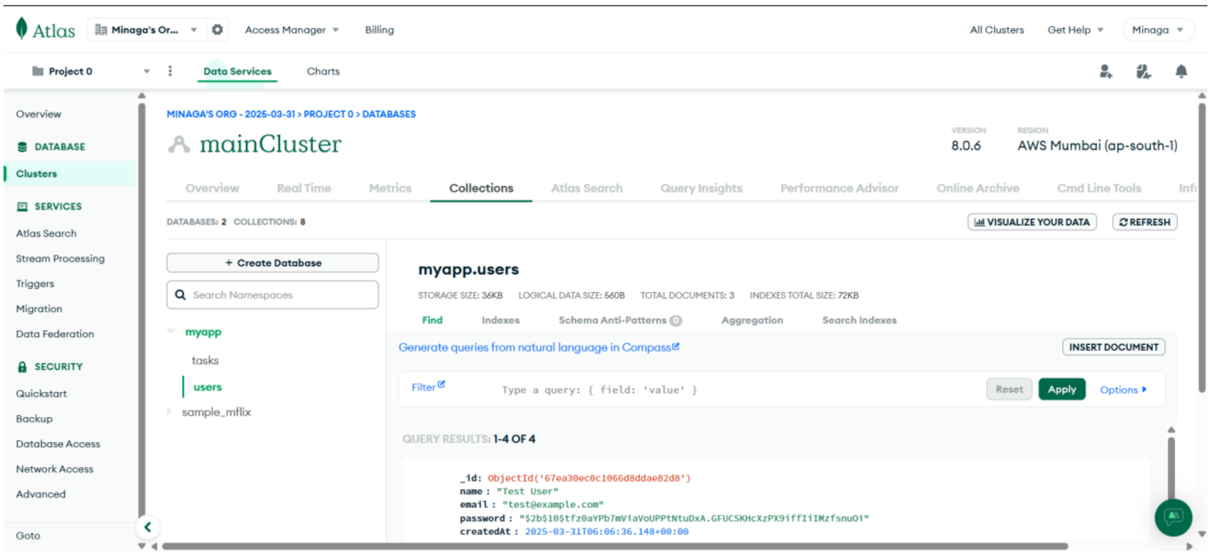
New Password

Confirm New Password

Update Password

Password changed successfully

Back to Dashboard



myapp.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 560B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 72KB

QUERY RESULTS: 1-4 OF 4

```
_id: ObjectId('67ea30ec0c1066d8ddae82d8')
name : "Test User"
email : "test@example.com"
password : "$2b$10$tfz0aYPb7mViaVoUPPtNtuDxA.GFUCSKHcXzPX9iffIiIMzfsnuOi"
createdAt : 2025-03-31T06:06:36.148+00:00
```

6. Logout

- Click Logout in the top right of the dashboard
- Token is cleared from local storage
- You're redirected to login

8. Additional Notes

Git & Version Control

- Git was initialized and used during the final phase of development to organize and submit the project.
- The full project was committed and pushed to GitHub using:
  - A single repository with separate branches for frontend and backend
  - Proper file structuring, .gitignore, and .env exclusion
- Branches:
  - frontend – Contains all React frontend code
  - backend – Contains Node.js + Express server and MongoDB logic
- The final project can be submitted as:
  - A GitHub link (preferred)
  - Or a zipped folder (with .git folder if needed)

Cloud Technologies

- **MongoDB Atlas** (cloud-hosted database on AWS)
  - Used to store users, tasks, reset tokens
  - Easy to scale and deploy for real-world usage
- Optional deployment to **Render / Vercel / Netlify** if needed later

Optimization Techniques

- **Pagination** added to improve frontend performance
- **Lazy loading** used for large route components (Dashboard, Profile, etc.)
- React state kept **minimal and clean** to avoid unnecessary re-renders
- Data fetching handled inside useEffect with token-based authentication

Testing Coverage

- All core user flows tested:
  - Register / Login / Logout
  - Forgot + Reset password

    - Dashboard CRUD
    - Sorting, filtering, pagination
- Manual testing done on:
    - Chrome
    - Firefox
    - Edge

Code Style & Formatting

- Used **Prettier** to auto-format all frontend files:

```
npx prettier --write .
```

- Ensures consistent indentation, spacing, and syntax across the codebase

Future Enhancements (Optional)

- Add **backend-side pagination** and filtering for better scalability
- Add **profile editing** (name, avatar, etc.)
- Deploy app to a public URL with **CI/CD integration**
- Replace Mailtrap with **production-ready email service** (e.g., SendGrid)