Project Report: Medley Pharma – Online Form Builder

Table of Contents

1. Introduction.....1
    a. 1 Project Overview.....1
    b. 2 Purpose.....1
2. Ideation Phase.....2
    a. 1 Problem Statement.....2
    b. 2 Empathy Map Canvas.....2
    c. 3 Brainstorming.....3
3. Requirement Analysis.....4
    a. 1 Customer Journey Map.....4
    b. 2 Solution Requirements.....4
    c. 3 Data Flow Diagram.....5
    d. 4 Technology Stack.....5
4. Project Design.....6
    a. 1 Problem-Solution Fit.....6
    b. 2 Proposed Solution.....6
    c. 3 Solution Architecture.....7
5. Project Planning & Scheduling.....7
    a. 1 Project Planning.....7
6. Functional and Performance Testing.....8

a. 1 Performance Testing.....8
7. Results.....9
a. 1 Output Screenshots.....9
8. Advantages & Disadvantages.....9
9. Conclusion.....9
10. Future Scope.....10
11. Appendix.....10

1. INTRODUCTION

Project Title: Medley Pharma - Online Form Builder

Team Members and Roles:

NAME	Reg. No.	Roles
Darshnik Rohal	22BAI10387	Backend
Anam Saeed	22BCE11045	Frontend
Devanshi Singh	22BCE11433	Documentation
Vairag Akbari	22BCE11402	Documentation

2. PROJECT OVERVIEW

Purpose:

Medley Pharma aims to provide a versatile and reliable solution for creating, managing, and analyzing online forms. It addresses the need for a user-friendly, customizable, and secure form-building tool that can cater to various industries, including healthcare, education, and market research.

Features:

- **Drag-and-Drop Interface**: Easy form creation without coding.
- **Customizable UI**: Branding options for logos, colors, and fonts.
- **Secure Submissions**: SSL encryption and CAPTCHA verification.
- **Data Analytics**: Visual reports and data export options.
- **Third-Party Integrations**: Automate workflows with tools like Zapier or Salesforce.

3. ARCHITECTURE

Frontend:

The frontend is built using **React.js**, leveraging its component-based architecture for a modular and maintainable UI. Key features include:

- **Responsive Design**: Ensures compatibility across devices.
- **Real-Time Preview**: Instantly reflects changes made to forms.
- **Interactive Elements**: Supports drag-and-drop form fields and conditional logic.

Backend:

The backend is developed using **Node.js** with **Express.js** as the framework. It handles:

- **API Endpoints**: Manages form data, submissions, and analytics.
- **Security Measures**: Implements authentication and authorization using JSON Web Tokens (JWT).
- **Third-Party Integrations**: Handles API calls for services like email notifications and CRM integrations.

Database:

The database is designed using **MongoDB**, chosen for its flexibility and scalability. It stores:

- **Form Definitions**: Metadata about each form, including fields and layout.

- **Form Submissions**: Collected data from users, securely stored and accessible for analytics.
- **User Data**: Information about users, including login credentials and access permissions.

4. SETUP INSTRUCTIONS

Prerequisites

Before setting up the project, ensure you have the following software dependencies installed:

1. **Node.js** (v16 or later): Required for running the backend server and managing dependencies[1].
2. **npm** (Node Package Manager): Comes bundled with Node.js for installing packages[1].
3. **MongoDB**: A NoSQL database for storing form data and user information[2][3].
4. **Git**: Version control system to clone the repository.
5. **Code Editor**: Recommended editors like Visual Studio Code or Sublime Text.
6. **Dotenv Package**: For managing environment variables securely[4].

Installation

Follow these steps to set up the Medley Pharma project on your local machine:

Step 1: Clone the Repository

1. Open a terminal or command prompt.
2. Run the following command to clone the project repository:

```
git clone https://github.com/your-username/medley-pharma.git
```

3. Navigate into the project directory:

```
cd medley-pharma
```

Step 2: Install Dependencies

1. Install backend dependencies:

```
cd server
npm install
```

2. Install frontend dependencies:

```
cd ../client
npm install
```

Step 3: Set Up Environment Variables

1. Create a `.env` file in both `server` and `client` directories.
2. Add the following variables to the `.env` file in the server directory:

```
DB_URI=mongodb://localhost:27017/medley-pharma
JWT_SECRET=your_jwt_secret_key
PORT=5000
```

3. Add any necessary frontend environment variables in the client `.env` file.

Step 4: Start MongoDB Server

1. Ensure MongoDB is installed on your system.
2. Start MongoDB locally by running:

```
mongod --dbpath=/data/db
```

3. Verify that MongoDB is running successfully.

Step 5: Run Backend and Frontend Servers

1. Start the backend server:

```
cd server
npm start
```

2. Start the frontend server:

```
cd ../client
npm start
```

Notes:

- Ensure that both servers are running on their respective ports (e.g., backend: `http://localhost:5000`, frontend: `http://localhost:3000`).
- If you encounter issues, check that all dependencies are installed correctly and MongoDB is running.

1. https://www.mongodb.com/docs/drivers/node/current/quick-start/download-and-install/
2. https://blog.nextideatech.com/how-to-integrate-mongodb-with-your-node-js-application-4-easy-steps/
3. https://www.geeksforgeeks.org/how-to-connect-mongodb-database-in-a-node-js-applications/
4. https://www.mongodb.com/resources/languages/mongodb-and-npm-tutorial

5. FOLDER STRUCTURE

Client (React Frontend)

The React frontend is structured as follows:

```
client/
├── node_modules/        # Node.js dependencies
├── public/              # Static assets (HTML, images, etc.)
├── src/                 # Source code
│   ├── components/      # Reusable React components
│   │   ├── FormBuilder/   # Form builder components
│   │   ├── FormDisplay/   # Components to display forms
│   │   └── ...
│   ├── pages/           # React pages/views
│   │   ├── Home.js      # Home page
│   │   ├── CreateForm.js  # Page to create new forms
│   │   └── ...
│   ├── App.js           # Main application component
│   ├── index.js         # Entry point for the React app
│   └── styles/          # CSS or styled-components
├── .env                 # Environment variables
├── package.json         # Frontend dependencies and scripts
├── README.md            # Documentation
└── ...
```

Server (Node.js Backend)

The Node.js backend is organized in this structure:

```
server/
├── node_modules/          # Node.js dependencies
├── models/              # MongoDB data models
│   ├── Form.js          # Form model
│   ├── User.js          # User model
│   └── ...
├── routes/              # API routes
│   ├── formRoutes.js    # Form-related API endpoints
│   ├── userRoutes.js    # User authentication and profile endpoints
│   └── ...
├── config/              # Configuration files
│   ├── database.js      # MongoDB connection
│   └── ...
├── middleware/          # Custom middleware functions
│   ├── authMiddleware.js # Authentication middleware
│   └── ...
├── .env                 # Environment variables
├── server.js            # Main server file
├── package.json         # Backend dependencies and scripts
└── README.md            # Documentation
```

6. RUNNING THE APPLICATION

To start the Medley Pharma application locally, follow these steps:

Frontend

1. Open a terminal and navigate to the `client` directory:

   ```
   cd client
   ```

2. Start the React development server:

   ```
   npm start
   ```

This command starts the frontend server, typically on `http://localhost:3000`.

Backend

1. Open a separate terminal and navigate to the `server` directory:

```
cd server
```

2. Start the Node.js backend server:

```
npm start
```

This command starts the backend server, typically on `http://localhost:5000`.

With both servers running, the Medley Pharma application should be accessible in your web browser.

7. API DOCUMENTATION

Here are some example endpoints exposed by the backend, along with their descriptions:

Endpoint	Method	Description	Parameters (Request Body)	Example Response
`/api/forms`	POST	Creates a new form.	`{ "title": "Form Title", "fields": [{...}] }`	`{ "success": true, "formId": "uniqueFormId" }`
`/api/forms/:formId`	GET	Retrieves a specific form by ID.	None	`{ "formId": "uniqueFormId", "title": "Form Title", "fields": [{...}] }`
`/api/forms/:formId`	PUT	Updates an existing form.	`{ "title": "Updated Title", "fields": [{...}] }`	`{ "success": true, "message": "Form updated successfully" }`
`/api/forms/:formId`	DELETE	Deletes a form by ID.	None	`{ "success": true, "message": "Form deleted successfully" }`
`/api/submissions/:formId`	POST	Submits a new form submission.	`{ "responses": [{ "fieldId": "field1", "value": "User Response" }] }`	`{ "success": true, "submissionId": "uniqueSubmissionId" }`
`/api/submissions/:formId`	GET	Retrieves all submissions for a specific form.	None	`[{ "submissionId": "uniqueSubmissionId", "responses": [{...}] }]`
`/api/users/register`	POST	Registers a new user.	`{ "username": "newuser", "password": "password123" }`	`{ "success": true, "message": "User registered successfully" }`

/api/users/login	POST	Logs in an existing user.	{ "username": "existinguser", "password": "password123" }	{ "success": true, "token": "jwtToken" }
/api/users/profile	GET	Retrieves the profile information for the currently authenticated user.	Requires JWT token in the Authorization header.	{ "userId": "uniqueUserId", "username": "existinguser", "email": "user@example.com" }

8. AUTHENTICATION

Authentication and authorization are handled using JSON Web Tokens (JWT). The process is as follows:

1. **User Registration**:
    - When a new user registers, their credentials (username and password) are validated and stored in the database.
    - A JWT is generated and sent back to the user upon successful registration.
2. **User Login**:
    - When a user logs in, the provided credentials are authenticated against the stored credentials.
    - Upon successful authentication, a JWT is generated and sent back to the user.
3. **Authorization**:
    - Protected routes require a valid JWT in the `Authorization` header (e.g., `Authorization: Bearer <token>`).
    - Middleware verifies the JWT and authorizes the user to access the protected resource.
    - Tokens have an expiration time to enhance security.

9. USER INTERFACE

Medley Pharma features a user-friendly interface designed for ease of use and customization. Key UI elements include:

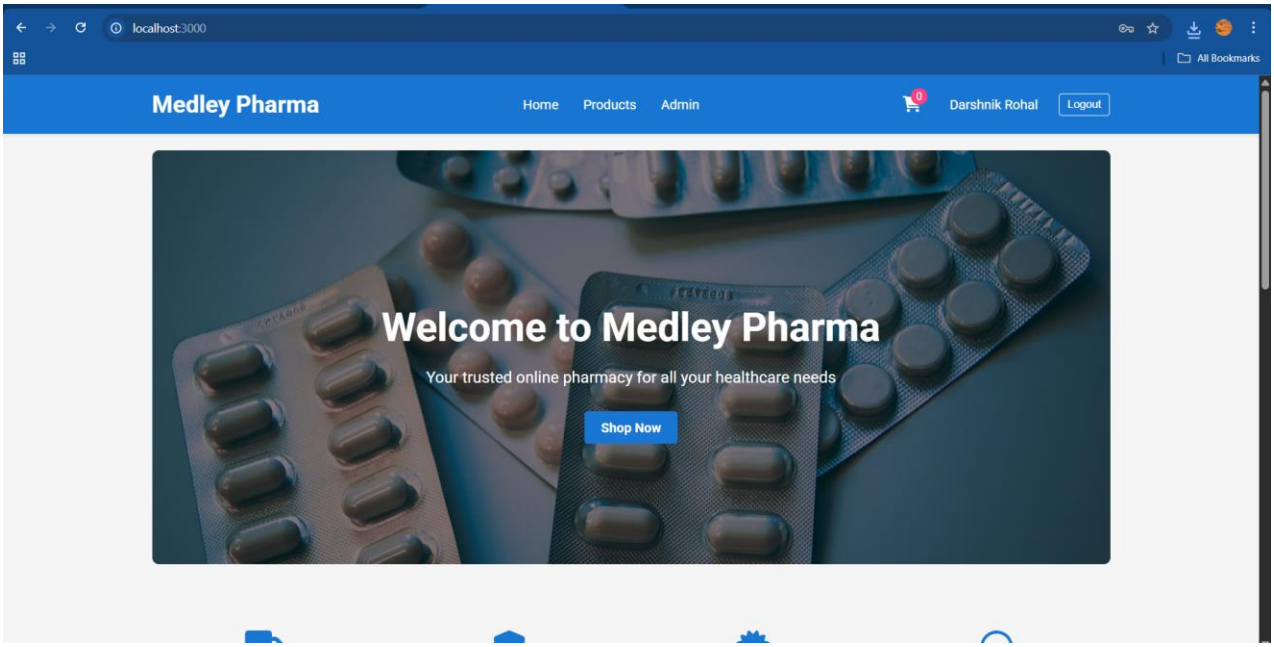
- **Form Builder**: A drag-and-drop interface for creating forms.
- **Form Preview**: Real-time preview of the form as it's being built.
- **Customization Options**: Branding settings to adjust colors, fonts, and logos.
- **Data Visualization**: Charts and graphs for analyzing form submissions.$$*Insert screenshots or GIFs showcasing different UI features here*]

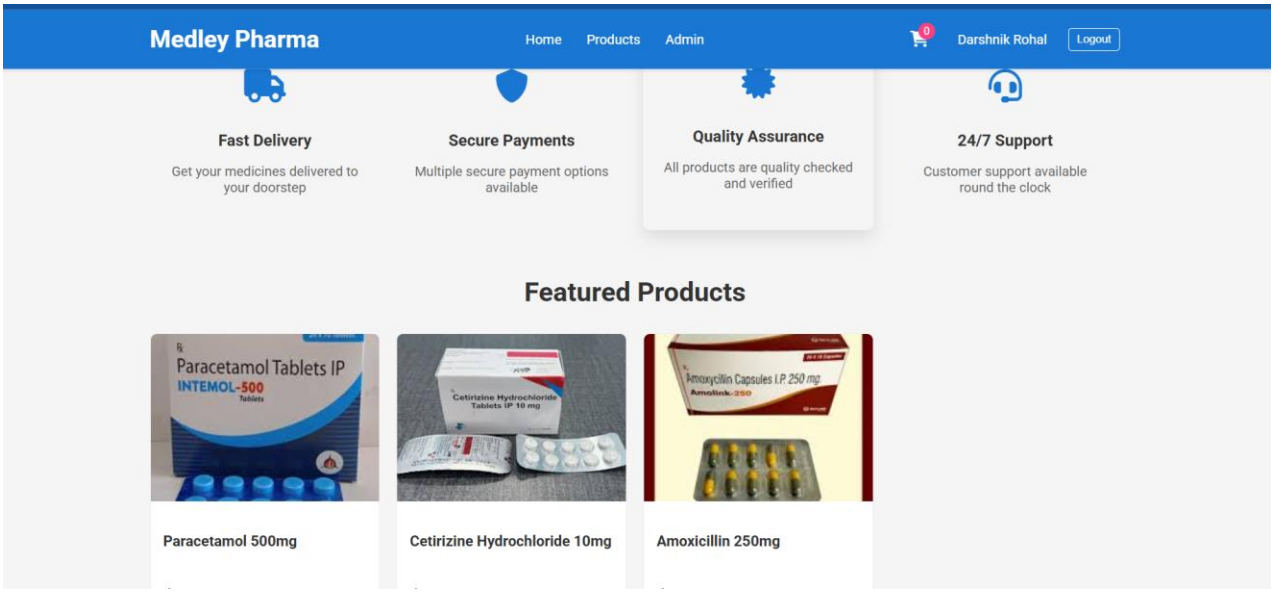
10. TESTING

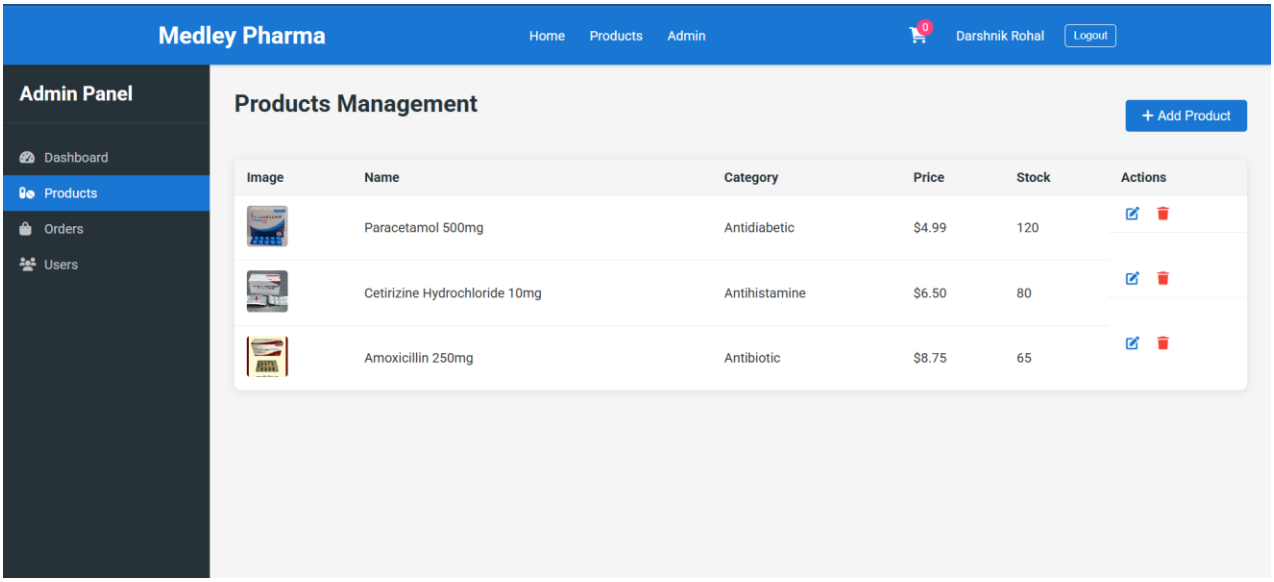
The testing strategy includes:

- **Unit Tests**: Testing individual components and functions in isolation.
    - Tools: Jest, Mocha.
- **Integration Tests**: Testing interactions between different parts of the system.
    - Tools: Jest, Mocha.
- **End-to-End Tests**: Testing the entire application flow from user interaction to backend processing.
    - Tools: Cypress, Selenium.
- **Performance Tests**: Evaluating the application's performance under different loads.
    - Tools: Apache JMeter, LoadView.
- **Security Tests**: Identifying and addressing potential security vulnerabilities.
    - Tools: OWASP ZAP, Nessus.

11. SCREENSHOTS OR DEMO


localhost:3000
All Bookmarks
Medley Pharma
Home
Products
Admin
0
Darshnik Rohal
Logout
Welcome to Medley Pharma
Your trusted online pharmacy for all your healthcare needs
Shop Now


Medley Pharma
Home
Products
Admin
0
Darshnik Rohal
Logout
Fast Delivery
Get your medicines delivered to your doorstep
Secure Payments
Multiple secure payment options available
Quality Assurance
All products are quality checked and verified
24/7 Support
Customer support available round the clock
Featured Products
Paracetamol Tablets IP
INTEMOL-500
Cetirizine Hydrochloride Tablets IP 10 mg
Amoxycillin Capsules I.P. 250 mg
Amolink-250
Paracetamol 500mg
Cetirizine Hydrochloride 10mg
Amoxicillin 250mg



Demo Link: https://drive.google.com/file/d/140f65Ppcd6F7IZUjgXRHEBuqSWwjVcSX/view?usp=drive_link

12. KNOWN ISSUES

Current known issues include:

- Offline functionality is limited and may not work reliably in all scenarios.
- Third-party integrations may require additional setup and configuration.
- Compatibility issues with older browsers.

13. FUTURE ENHANCEMENTS

Potential future enhancements include:

- Enhanced offline form functionality.
- Mobile app for on-the-go form management.
- Dynamic field suggestions using AI.
- More advanced analytics and reporting features.
- Integration with more third-party services.