Contents

[1. INTRODUCTION 3](#_Toc185105225)

[1.1 App Purpose 3](#_Toc185105226)

[1.2 App Scope 3](#_Toc185105227)

[1.3 Goals of the Team 3](#_Toc185105228)

[1.4 Process Model 4](#_Toc185105229)

[1.5 Team Organization 4](#_Toc185105230)

[2. RESEARCH 4](#_Toc185105231)

[2.1 Market Research: 4](#_Toc185105232)

[2.1.1 Industry Trends 4](#_Toc185105233)

[2.1.2 Competitive Analysis 5](#_Toc185105234)

[2.1.3 Target Audience 5](#_Toc185105235)

[2.2 Technical Research: 5](#_Toc185105236)

[2.2.1 Technology Stack 5](#_Toc185105237)

[3. DESCRIPTION 6](#_Toc185105238)

[3.1 Home Page: 6](#_Toc185105239)

[3.2 Dashboard: 7](#_Toc185105240)

[3.3 Registration: 7](#_Toc185105241)

[3.4 Login: 7](#_Toc185105242)

[3.5 Logout: 7](#_Toc185105243)

[3.6 User Management: 8](#_Toc185105244)

[4. Requirements 8](#_Toc185105245)

[4.1Functional requirements: 8](#_Toc185105246)

[4.1.1 Daily medicine: 8](#_Toc185105247)

[4.1.2 Categories: 8](#_Toc185105248)

[4.1.3 Healthcare medicine : 8](#_Toc185105249)

[4.1.4 Sign up/ Register: 9](#_Toc185105250)

[4.1.5 Post-details: 9](#_Toc185105251)

[4.1.6 Contact page: 9](#_Toc185105252)

[4.1.7 Dashboard: 9](#_Toc185105253)

[4.1.8 Post medicine: 9](#_Toc185105254)

[4.2 Performance requirement 9](#_Toc185105256)

[4.2.1 Page Load Speed 9](#_Toc185105257)

[4.2.2 Responsiveness 9](#_Toc185105258)

[4.2.3 Scalability 9](#_Toc185105259)

[4.2.4 Server and Backend Performance 10](#_Toc185105260)

[4.3 Software Requirements 10](#_Toc185105261)

[4.3.1 Development Environment 10](#_Toc185105262)

[4.3.2 Frontend Requirements 10](#_Toc185105263)

[4.3.3 Backend Requirements 10](#_Toc185105264)

[4.3.4 Additional Tools 10](#_Toc185105265)

[4.3.5. Third-Party Services 11](#_Toc185105266)

[4.4 Hardware Requirements 11](#_Toc185105267)

[4.4.1 For Development 11](#_Toc185105268)

[4.4.2 For Hosting 11](#_Toc185105269)

[5. SYSTEM ANALYSIS AND MODELING 11](#_Toc185105270)

[5.1 Functional Modeling 11](#_Toc185105271)

[5.1.1 Level 0 of Data Flow Diagram 11](#_Toc185105272)

[5.1.2 Level 1 of Data Flow Diagram 12](#_Toc185105273)

# 1. INTRODUCTION

Medi-Shop is an innovative web application designed to provide an intuitive, interactive, and user-friendly platform for accessing a wide range of healthcare products and services. Built with **React** for the frontend and **Express** for the backend, this app offers modern web experiences and a robust architecture to support scalability and efficient content management.

## 1.1 App Purpose

## The primary purpose of Medi-Shop is to create a seamless and reliable platform where users can access, purchase, and manage their healthcare needs effortlessly. Medi-Shop aims to provide a clean and user-friendly experience, enabling customers to browse a wide range of medical products, including prescription medications, over-the-counter remedies, and wellness essentials.

## 1.2 App Scope

The app will include the following key features:

1. **Search Functionality**: Users can search for specific medicine by keywords, tags, or categories.
2. **Home Page**: Displays categorized medicines, daily medicines, and healthcare medicines based on user interests.
3. **Dashboard**: Admin can upload, edit, delete medicines whenever they need to do.
4. **Authentication**: Secure login and logout panel for user registration, login, and session management.

## 1.3 Goals of the Team

The primary goals of the development team are:

* To build a user-centric platform that makes webpage accessible and enjoyable for both authors and buyers.
* To ensure a responsive and seamless user interface (UI) and user experience (UX) across all devices.
* To implement robust backend systems for secure user authentication, smooth medicine post management, and efficient data storage and retrieval.
* To integrate features like search, filtering, and trending articles, making content discovery easy and intuitive.

## 1.4 Process Model

The development process follows an **Agile methodology** with the following stages:

1. **Requirement Gathering**: Identifying user needs, app functionalities, and technical specifications.
2. **Design & Prototyping**: Creating wireframes, mockups, and prototypes for both the frontend and backend.
3. **Development & Implementation**: Dividing tasks into sprints, building components and APIs iteratively using **React** and **Express**.
4. **Testing & Debugging**: Conducting unit, integration, and user acceptance testing (UAT) to ensure quality and stability.
5. **Deployment**: Preparing the app for deployment on cloud platforms (e.g., Vercel, Heroku) and making it publicly accessible.
6. **Maintenance & Updates**: Continuously improving the app based on user feedback and monitoring its performance.

## 1.5 Team Organization

The development team is organized into specialized roles:

* **Frontend Developer(s)**: Focus on implementing the user interface using **React**, integrating with the backend, and ensuring the UI is responsive and visually appealing.
* **Backend Developer(s)**: Work on setting up and maintaining the **Express** backend, developing APIs for data management, user authentication, and integrating with the frontend.
* **Quality Assurance (QA) Engineer**: Responsible for testing the app across different devices and ensuring all features work as expected.
* **Project Manager**: Oversees the entire development process, manages timelines, ensures the team follows the Agile process, and communicates with stakeholders.

# 2. RESEARCH

## **2.1 Market Research:**

### **2.1.1 Industry Trends**

* The pharmacy industry has evolved significantly over the past decade, with content platforms becoming central to personal expression, business marketing, and professional networking.
* As of 2024, over **tens of thousands of medicines** exist globally. The exact number is challenging to determine due to variations in classifications, approvals, and the existence of generic versions of the same drug.
* The demand for medicine is increasing, as users prefer platforms that recommend medicines based on their interests.
* Mobile accessibility is a critical factor; over 55% of medical items access content via mobile devices.
* Monetization through advertisements, affiliate marketing, and sponsored content continues to drive Medi-shop as a profitable business for many pharmacies.

### **2.1.2 Competitive Analysis**

* **Competitors:** Popular platforms such as PharmEasy, Netmeds and MediBuddy dominate the market. These platforms provide ease of use, scalability, and powerful content management tools.
* **Opportunities:** By offering a unique feature set such as a more interactive UI, advanced search capabilities, and trending content sections, our blogging app can carve out its niche.

### **2.1.3 Target Audience**

* **Primary Audience:**
  + Aspiring patient who wants an easy-to-use platform for their daily medicines.
  + Patients are interested in curated content across multiple categories.
* **Secondary Audience:**
  + Businesses and organizations looking to use medicines for marketing and branding.
  + Pharmacies and medicine companies are creating web application for covering more market.

## **2.2 Technical Research:**

### **2.2.1 Technology Stack**

**Frontend:**

* **Language:** JavaScript
* **Framework:** React.js
  + Popular for its component-based architecture, reusable UI components, and a vast ecosystem of libraries.
  + React’s virtual DOM ensures faster updates and a responsive user experience.
  + Strong community support and compatibility with mobile development through React Native.

**Backend:**

* **Language:** JavaScript
* **Framework:** Express
  + Express provides a robust, high-level framework suitable for rapid development and clean, pragmatic design.
  + Features such as built-in authentication, database management, and RESTful API support streamline backend development.
  + Scalability and security are major advantages.

**Database:**

* **Choice:** MongoDB
  + An open-source, non-relational database system that offers advanced features like JSON support, indexing, and scalability.
  + Compatible with Express’s ORM (Object Relational Mapping).

**Additional Tools:**

* **State Management:** Redux (for managing global state in the app).
* **Styling:** Tailwind CSS.
* **APIs:** Express REST Framework (ERF) for exposing backend functionality to the frontend.
* **Hosting:**
  + Frontend: Vercel or Netlify (optimized for React.js deployments).
  + Backend: Vercel.

# 3. DESCRIPTION

## 3.1 Home Page:

The Home Page serves as the primary interface for users, providing access to all essential features of the blogging web app. It includes:

* **Category-Based Content:** A well-organized display of articles categorized into various topics such as daily, healthcare, children and more.
* **Daily Medicines:** Highlights the most popular and frequently viewed medicines, offering users a quick glance at current trends.
* **Personalized Recommendations:** Suggestions based on the user’s interests and search history.
* **Search Bar:** Allows users to quickly locate articles by entering keywords or tags.

## 3.2 Dashboard:

The Dashboard is a personalized space for registered users, providing tools to manage their content and activities. Key functionalities include:

* **Medicine Management:** Create, edit, and delete blog posts.
* **Drafts:** Save medicines as drafts for later editing and publishing.
* **Content Moderation:** For admin users, the ability to review and moderate user-generated content and comments.

## 3.3 Registration:

The Registration feature enables new users to sign up and create an account on the platform. Key elements include:

* **User-Friendly Form:** A simple and intuitive form requiring basic details such as username, email, and password.
* **Email Verification:** Sends a confirmation email to verify the user’s identity.
* **Error Handling:** Provides clear feedback for invalid inputs or already registered email addresses.

## 3.4 Login:

The Login feature allows users to access their accounts securely. Key aspects include:

* **Authentication:** Verifies user credentials against the stored database.
* **Remember Me Option:** Saves user login state for convenience on trusted devices.
* **Error Messages:** Displays messages for incorrect username or password inputs

## 3.5 Logout:

The Logout feature ensures user sessions can be terminated securely. Features include:

* **Session Termination:** Ends the active session and clears authentication tokens.
* **Redirect to Home Page:** After logging out, users are redirected to the Home Page.

3.6 User Management:

The UserManagement system encompasses functionalities for both regular users and admin users. Features include:

* **Profile Management:** Allows users to update their personal information, including profile picture, bio, and password.
* **Roles and Permissions:** Differentiates between regular users and administrators, with specific privileges for each.
* **Content Moderation Tools:** For administrators, tools to monitor and manage inappropriate content or user behavior.

This detailed breakdown of features highlights the essential components of the Medi-Shop Web App, ensuring a user-friendly experience for both patient and seller while maintaining robust administrative tools for smooth platform operation.

4. Requirements

4.1Functional requirements:

4.1.1 Daily Medicines:

The **Daily Medicines** section on Medi-Shop is designed for users seeking quick and convenient access to commonly used medications for everyday health needs

4.1.2 Categories:

There will be a category section. There will be many categories.  And an article under them. Clicking them it redirects to the article under that particular category

4.1.3 Healthcare medicines:

This section is sorted based on the healthcare.

4.1.4 Sign up/ Register:

For sign up or register a user must provide his name, email and password. For any missing input field, it doesn’t allow user to register

4.1.5 Post-details:

the part contains the medicine’s name, posting date, and other details of the post with that content. It also allows a logged in user to comment in the details.

4.1.6 Contact page:

It contains the information about the website like email address, office address

4.1.7 Dashboard:

It will have posts, add posts, notifications and comments buttons. The dashboard page will have the count of all his posts, total view count, total likes and his total bookmarks. There will be details of all posted by author by himself in a tabular format.

4.1.8 Post medicine:

The post medicine is shown in tabular format and there will be a search field and a sort button. Edit and delete any post options are available here

4.2 Performance requirement

4.2.1 Page Load Speed

* Time to First Byte (TTFB): Should be under 200ms.
* Full Page Load: Aim for under 3 seconds on a 4G connection.
* Core Web Vitals:
  + Largest Contentful Paint (LCP): < 2.5 seconds.
  + First Input Delay (FID): < 100ms.
  + Cumulative Layout Shift (CLS): < 0.1.

4.2.2 Responsiveness

* 1. **Device Compatibility:** Optimize for mobile, tablet, and desktop views.
  2. **Media Optimization:** Use responsive images, lazy loading, and modern formats like WebP

4.2.3 Scalability

1. Handle increased traffic without significant performance degradation.
2. **Concurrent Users:** Define expected active users (e.g., 1000 concurrent users for a mid-tier blog).
3. **Requests per Second (RPS):** Measure API endpoints' capacity (e.g., at least 50 RPS per API).

4.2.4 Server and Backend Performance

* 1. Database Query Response: Queries should respond within 100ms.
  2. Use caching mechanisms like Redis or in-memory caching for frequently accessed content.
  3. Optimize backend APIs for minimal latency.

4.3 Software Requirements

4.3.1 Development Environment

* **Programming Language:** JavaScript,
* **Frameworks:** React
* **Database:**
  + **Relational:** MongoDB.
* **Web Server:**
  + Nginx or Apache for serving static content and managing requests.
  + Node.js if using server-side rendering (e.g., Next.js).
* **Operating System:** Linux (Ubuntu, CentOS, etc.) for the server environment.

4.3.2 Frontend Requirements

* HTML, CSS, JavaScript: Basic tools for building responsive and interactive UIs.
* **CSS Frameworks:** tailwind
* **Bundlers:** Webpack, Vite, or similar for efficient frontend build processes.

4.3.3 Backend Requirements

* **API Frameworks:** REST Framework, Flask, etc.
* **Authentication & Authorization:** Use libraries like Passport.js, Firebase Auth, or Auth0.
* **Caching Systems:** Redis or Memcached for improving response times.

4.3.4 Additional Tools

* **Version Control:** Git and platforms like GitHub or GitLab.
* **Deployment Tools**: Docker, Kubernetes, or simple CI/CD pipelines via GitHub Actions or Jenkins.
* **Monitoring Tools:** Prometheus, Grafana, or New Relic for server and app performance monitoring.

4.3.5. Third-Party Services

* **Content Delivery Network (CDN):** Cloudflare, AWS CloudFront.
* **Analytics Tools:** Google Analytics, Hotjar.