Contents

[1. INTRODUCTION 3](#_Toc198497823)

[1.1 App Purpose 3](#_Toc198497824)

[1.2 App Scope 3](#_Toc198497825)

[1.3 Goals of the Team 3](#_Toc198497826)

[1.4 Process Model 3](#_Toc198497827)

[1.5 Team Organization 3](#_Toc198497828)

[2. RESEARCH 3](#_Toc198497829)

[2.1 Market Research: 3](#_Toc198497830)

[2.1.1 Industry Trends 3](#_Toc198497831)

[2.1.2 Competitive Analysis 3](#_Toc198497832)

[2.1.3 Target Audience 3](#_Toc198497833)

[2.2 Technical Research: 3](#_Toc198497834)

[2.2.1 Technology Stack 3](#_Toc198497835)

[3. DESCRIPTION 3](#_Toc198497836)

[3.1 Home Page: 3](#_Toc198497837)

[3.2 Dashboard: 3](#_Toc198497838)

[3.3 Registration: 3](#_Toc198497839)

[3.4 Login: 3](#_Toc198497840)

[3.5 Logout: 3](#_Toc198497841)

[3.6 User Management: 3](#_Toc198497842)

[4. Requirements 3](#_Toc198497843)

[4.1 Functional requirements: 3](#_Toc198497844)

[4.1.1 Worker Home 3](#_Toc198497845)

[4.1.2 TaskList 3](#_Toc198497846)

[4.1.3 Task Details 3](#_Toc198497847)

[4.1.4 MySubmission 3](#_Toc198497848)

[4.1.5 WithDrawals 3](#_Toc198497849)

[4.1.6 Task-Creator-Home 3](#_Toc198497850)

[4.1.7 Add new Tasks 3](#_Toc198497851)

[4.1.8 My Task’s 3](#_Toc198497852)

[4.1.9 Purchase Coin 3](#_Toc198497853)

[4.1.10 Admin-Home 3](#_Toc198497854)

[4.1.11 Manage Users 3](#_Toc198497855)

[4.1.12 Manage Tasks 3](#_Toc198497856)

[4.2performance requirement 3](#_Toc198497857)

[4.2.1 Page Load Speed 3](#_Toc198497858)

[4.2.2 Responsiveness 3](#_Toc198497859)

[4.2.3 Scalability 3](#_Toc198497860)

[4.2.4 Server and Backend Performance 3](#_Toc198497861)

[4.3 Software Requirements 3](#_Toc198497862)

[4.3.1 Development Environment 3](#_Toc198497863)

[4.3.2 Frontend Requirements 3](#_Toc198497864)

[4.3.3 Backend Requirements 3](#_Toc198497865)

[4.3.4 Additional Tools 4](#_Toc198497866)

[4.4 Hardware Requirements 4](#_Toc198497867)

[4.4.1 For Development 4](#_Toc198497868)

[4.4.2 For Hosting 4](#_Toc198497869)

[5. SYSTEM ANALYSIS AND MODELING 4](#_Toc198497870)

[5.1 Functional Modeling 4](#_Toc198497871)

[5.1.1 Level 0 of Data Flow Diagram 4](#_Toc198497872)

[5.1.2 Level 1 of Data Flow Diagram 5](#_Toc198497873)

# 1. INTRODUCTION

This blogging web application is designed to provide an intuitive, interactive, and user-friendly platform for individuals to create, share, and read blogs across various topics. Built with **React** for the frontend and **Django** 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 the blogging web app is to create a space where users can express their thoughts, share knowledge, and engage with others through written content. The app aims to provide a clean and seamless experience for authors to publish posts and for readers to discover articles based on categories, trending topics, and personalized recommendations.

## 1.2 App Scope

The app will include the following key features:

1. **Search Functionality**: Users can search for specific articles by keywords, tags, or categories.
2. **Home Page**: Displays categorized articles, trending content, and recommended blogs based on user interests.
3. **Dashboard**: Users can manage their blog posts, track views, comments, and interactions with other users.
4. **Authentication**: Secure login and logout panel for user registration, login, and session management.
5. **Interactive Features**: Users can leave comments on articles, interact with other readers, and engage with authors.

## 1.3 Goals of the Team

The primary goals of the development team are:

* To build a user-centric platform that makes blogging accessible and enjoyable for both authors and readers.
* 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 blog 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 **Django**.
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 **Django** 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 blogging industry has evolved significantly over the past decade, with content platforms becoming central to personal expression, business marketing, and professional networking.
* As of 2023, over **600 million blogs** exist globally, with billions of active readers engaging in content ranging from technology and lifestyle to education and entertainment.
* The demand for personalized and categorized content is increasing, as users prefer platforms that recommend blogs based on their interests.
* Mobile accessibility is a critical factor; over 55% of blog readers access content via mobile devices.
* Monetization through advertisements, affiliate marketing, and sponsored content continues to drive blogging as a profitable business for many content creators.

### **2.1.2 Competitive Analysis**

* **Competitors:** Popular platforms such as WordPress, Medium, Blogger, and Ghost dominate the market. These platforms provide ease of use, scalability, and powerful content management tools.
* **Strengths of Competitors:**
  + WordPress: Highly customizable with plugins and themes.
  + Medium: Focused on simplicity and clean reading experiences.
  + Ghost: A modern, fast platform optimized for creators.
* **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 writers and bloggers who want an easy-to-use platform for sharing their content.
  + Readers interested in curated content across multiple categories.
* **Secondary Audience:**
  + Businesses and organizations looking to use blogs for marketing and branding.
  + Educators and students creating academic content or personal blogs.

## **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:** Python
* **Framework:** Django
  + Django 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:** PostgreSQL
  + An open-source, relational database system that offers advanced features like JSON support, indexing, and scalability.
  + Compatible with Django’s ORM (Object Relational Mapping).

**Additional Tools:**

* **State Management:** Zustland (for managing global state in the app).
* **Styling:** Bootstrap CSS.
* **APIs:** Django REST Framework (DRF) 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 Technology, Lifestyle, Education, and more.
* **Trending Articles:** Highlights the most popular and frequently viewed blogs, offering users a quick glance at current trends.
* **Personalized Recommendations:** Suggestions based on the user’s interests and reading 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:

* **Blog Management:** Create, edit, and delete blog posts.
* **Analytics:** View engagement metrics such as the number of views, likes, and comments on their articles.
* **Drafts:** Save articles 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 Blogging Web App, ensuring a user-friendly experience for both readers and content creators while maintaining robust administrative tools for smooth platform operation.

# 4. Requirements

## 4.1 Functional requirements:

### 4.1.1 Worker Home

States

Worker will see his available coin, Total Submission (Count of all submission made by

worker), Total Earning (sum of payable\_amoun of the worker where status is approved)

Approved Submission

Task-Creator will see all the submission made by him where the status is “approved” in a table

format from submission collection with following information

● task\_title

● payable\_amount

● creator\_name

● status

### 4.1.2 TaskList

In This Route, Worker will see All the tasks where the task\_count is greater than 0 with

Following information

● task\_title

● creator\_name

● completion\_date

● payable\_amount

● task\_quantity

● ViewDetails Button

Data will be in card format. By clicking view Details navigate workers to the task details route.

### 4.1.3 Task Details

show all the information of the Task and a submission form in this Route.

The Submission form will contain 1 input field(text-area) name submission\_Details. After

submitting the form insert following informations in the submission Collection

● task\_id

● task\_title

● task\_detail

● task\_img\_url

● payable\_amount

● worker\_email

● submission\_details

● worker\_name

● creator\_name

● creator\_email

● current\_date

● status (pending)

### 4.1.4 MySubmission

Show all the submissions information from submissionCollection where the workerEmail

matched with the current worker Email . show data in a tabular form.

### 4.1.5 WithDrawals

20 Coins = 1Dollar.

This Route will show how much money a worker can withdraw. Suppose if a worker has 300

coins then he can withdraw a maximum 15 dollars.

MaximumWithdrawal Amount

Show user his Maximum WithDraw Amount (dollar)

WithDrawal Form

● CoinToWithDraw (Number)

● withdraw\_amount (Number) (Not editable. It will change when the coin to withdraw

field changes. 20 coin = 1 dollar)

● Select Payment System (DropDown) (Baksh, Rocket, Nagad)

● AccountNumber

● WithDrawButton

if The Amount is greater than maximum withDraw account then reject his request and show

him an alert.

else, insert all the information in the withdrawCollection with following info

● worker\_email

● worker\_name

● withdraw\_coin

● withdraw\_amount

● payment\_system

● withdraw\_time

### 4.1.6 Task-Creator-Home

States

Task-Creator will see the his available coin, pending Task(sum of all task\_quantity of his

added Tasks), total payment paid by user

Task To Review

Task-Creator will see all Review requests of his tasks where the status is “pending” in a table

format from submission collection with following information

● worker\_name and worker\_email

● task\_title

● payable\_amount

● ViewSubmission Button( will open a modal and show the submission detail)

● Actionable Buttons

○ ApproveButton

○ Reject Button

\*\*\* On Clicking Approve Button increase payable amount coin for the workers and change the

SubmissionStatus to “approve” to the submission collection

\*\*\* On Clicking Reject Button change the status to “rejected” to the submission collection

### 4.1.7 Add new Tasks

This section will contain a Form with following input fields

● task\_title

● task\_detail

● task\_quantity (number)

● payable\_amount (per Task) (number)

● completion\_date

● submission\_info

● task\_image\_url ( implement imageBB for uploading if want to get challenge mark )

● AddTask

On Clicking Add Task

\*\*\*check if (task\_quantity\* payable\_amount) is greater than users available count then

through an alert “Not available Coin. Purchase Coin”.

Else, it will add the following input field information and creator\_email, creator\_name,

current\_time to the Task Collection of system Database and reduce (task\_quantity\*

payable\_amount) from users' available coins

### 4.1.8 My Task’s

In this section the user will show all the tasks he added in descending order based on Time in a table format.

● ShowThe task\_title , task\_count, payable\_amount, update and delete button

● onClicking update users can update the Title , TaskDetail and submission Count.

● onClicking Delete, delete the task from task Collection. And Increase the

(task\_quantity\* payable\_amount) coin in his available coin

### 4.1.9 Purchase Coin

From This Route User can purchase coins. Implement a stripe-based payment system on this

route.

Payment info

Show users 4 card with following info

● 10coins=1dollar.

● 100coins=9dollars.

● 500coin= 19dollars

● 1000coin=39dollar

On Clicking a specific card redirects the user to pay a specific amount.

After successful payment, add the payment info into paymentCollection and increase

TaskCreator Coin.

### 4.1.10 Admin-Home

States

admin will see the count of total users, total coin, total payments

Withdraw request

Admin will see all withdrawal requests from withdrawCollection made by users in a table

format with following information

● worker\_name

● withdraw\_coin

● Withdrawamount

● PaymentNumber

● Payment\_system

● withdraw\_time

● PaymentSuccessButton

After clicking the payment success button, data will be deleted from the withdrawal collection.

And the user coin will be deducted by withdraw\_coin from withdrawal collection

### 4.1.11 Manage Users

The section will show a table of all users who have the role “worker” with display\_name,

user\_email, photo\_url, role, coin and some actionable button

● remove (will delete user from the database)

○ Byclicking Remove user will be deleted from the server.

● UpdateRole (Dropdown field. On change it will change the role of user)

○ Admin

○ Task-Creator

○ Worker

### 4.1.12 Manage Tasks

User will see The TaskList in a table format with following information

● task\_title

● TaskCreator Name

● TaskCount

● Coin\_Needed

● Availability

● ViewTaskIcon(onClicking It will open Modal and show Tasks details )

● Delete Task ( By clicking Task will be deleted from database

## 4.2performance 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:**
  + **MongoDB**.
* **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:** Express
* **Authentication & Authorization:**, Firebase Auth

### 4.3.4 Additional Tools

* **Version Control:** Git and platforms like GitHub or GitLab.
* **Deployment Tools**: Vercel, firebase

## 4.4 Hardware Requirements

### 4.4.1 For Development

* **Processor:** Quad-core or higher (e.g., Intel i5/i7 or AMD Ryzen 5/7).
* **RAM:** Minimum 8GB (16GB recommended for smooth multitasking with IDEs, local servers, etc.).
* **Storage:** SSD with at least 256GB space for projects and tools.
* **Operating System:** Windows, macOS, or Linux.

### 4.4.2 For Hosting

* **Small to Medium Blog**
  + **CPU:** Single-core 2GHz or better.
  + **RAM:** 1GB (2GB recommended).
  + **Storage:** 10GB SSD (expand based on content size).
* **High Traffic Blog**
  + **CPU:** Multi-core (4 vCPUs or more).
  + **RAM:** 8GB or higher.
  + **Storage:** SSD with 100GB or more.
  + **Example:** AWS EC2 t3. medium or a similar VPS with scalable options.

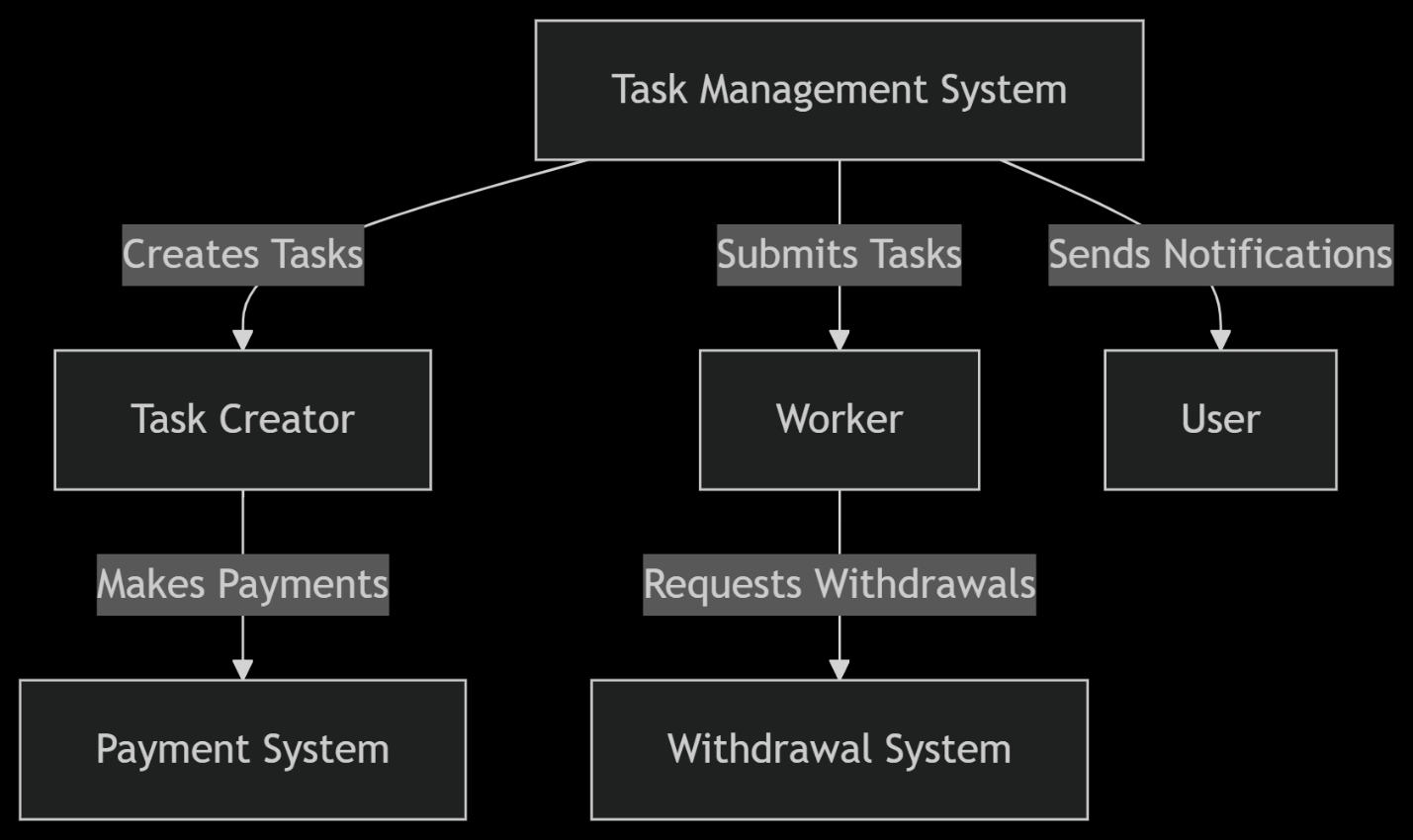
# 5. SYSTEM ANALYSIS AND MODELING

## 5.1 Functional Modeling

### 5.1.1 Level 0 of Data Flow Diagram

Level 0 of data flow diagram shows the system as a single process and its interactions with external entities.

**0-LEVEL-DFD**



### 5.1.2 Level 1 of Data Flow Diagram

Level 1 of data flow diagram breaks down the system into major sub-processes and data flows.

**1-LEVEL-DFD**

