**PRICEWATCH**

**A PROJECT REPORT**

**for**

**Project KCA451**

**Session 2023-24**

**Submitted by**

**Akhil Singh Chauhan**

**2200290140020**

**Submitted in partial fulfillment of the**

**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

### Ms. Divya Singhal

**Assistant Professor**



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**(JUNE 2024)**

**CERTIFICATE**

Certified that **Akhil Singh Chauhan 2200290140020** has carried out the project work having “**PRICEWATCH**” (**Project KCA451**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:**

**Akhil Singh Chauhan**

**2200290140020**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

**Ms. Divya Singhal Dr. Arun Tripathi**

**Assistant Professor Head**

**Department of Computer Applications Department of Computer Applications KIET Group of Institutions, Ghaziabad KIET Group of Institutions Ghaziabad**

**ABSTRACT**

This project aims to replicate the fundamental features of Pastebin.com, a widely used web-based text-sharing platform. The implemented system provides users with the ability to create, share, and manage text snippets with advanced functionalities such as syntax highlighting, privacy settings, and expiration options. Users can register accounts, allowing for secure paste management and editing capabilities. The platform supports various programming languages through syntax highlighting libraries, enhancing the readability of shared code snippets. Privacy settings enable users to choose between public, unlisted, and private paste visibility, while an expiration management system ensures the automatic removal of pastes after specified time intervals. The user interface is designed to be intuitive, offering a seamless experience for paste creation, editing, and viewing. Additional features include search functionality, API integration, notifications, analytics, and legal considerations. The project utilizes modern web technologies, employing a robust backend framework, a reliable database system, and a responsive frontend framework. The replication adheres to security standards, implementing measures such as rate limiting to prevent abuse. This project serves as a comprehensive exploration of web development principles, encompassing user authentication, data management, and feature-rich user interfaces.

**ACKNOWLEDGEMENTS**

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Ms. Divya Singhal** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Akhil Singh Chauhan**

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Certificate | | ii |
|  | Abstract | | iii |
|  | Acknowledgements | | iv |
|  | Table of Contents | | v |
| **1** | **Introduction** | | **6-11** |
|  | 1.1 | Background and Motivation | 6 |
|  | 1.2 | Objectives and Goals | 7 |
|  | 1.3 | Scope and Limitations | 9 |
| **2** | **Literature Overview** | | **12-16** |
|  | 2.1 | Review of Existing Text-Sharing Platforms | 13 |
|  | 2.2 | Overview of Technologies and Frameworks | 14 |
|  | 2.3 | Relevant Literature on Web Development | 15 |
| **3** | **Requirement Analysis** | | **17-20** |
|  | 3.1 | Introduction | 17 |
|  | 3.2 | Functional Requirements | 17 |
|  | 3.3 | Non Functional Requirements | 18 |
|  | 3.4 | Use Case Diagram | 19 |
| **4** | **System Design** | | **21-24** |
|  | 4.1 | Architecture and System Overview | 21 |
|  | 4.2 | Database Design | 22 |
|  | 4.3 | ER Diagram | 23 |
|  | 4.4 | Data Flow Diagram | 23 |
| **5** | **Implementation** | | **25-26** |
|  | 5.1 | Tech Stack | 25 |
|  | 5.2 | Challenges | 26 |
| **6** | **Testing** | | **27-29** |
|  | 6.1 | Testing Methodologies | 27 |
|  | 6.2 | Test Cases | 29 |
| **7** | **Conclusion** | | **30-31** |
| **8** | **References** | | **32** |

**CHAPTER 1**

**INTRODUCTION**

Online shopping has revolutionized the way consumers purchase goods, offering unparalleled convenience and access to a vast array of products. Among the leading e-commerce platforms, Amazon stands out for its extensive product range and competitive pricing. However, the dynamic nature of online prices, influenced by factors such as sales, discounts, and market trends, presents a significant challenge for consumers aiming to make cost-effective purchases. Manually tracking these price fluctuations is not only tedious but also inefficient.

Pricewatch emerges as a solution to this problem, offering an automated, user-friendly web application designed to help consumers monitor price changes on Amazon. By leveraging advanced web technologies such as Next.js for the front end, Puppeteer for web scraping, MongoDB for data storage, Bright Data for IP rotation, and cron jobs for scheduling, Pricewatch provides a seamless experience that empowers users to make informed purchasing decisions effortlessly.

The development of Pricewatch is motivated by a desire to simplify the online shopping experience and assist consumers in making informed purchasing decisions. By integrating cutting-edge technologies and user-centric design principles, Pricewatch aims to revolutionize the way users track prices and secure the best deals on Amazon..

**1.1 Background and Motivation**

In recent years, the proliferation of online shopping platforms has transformed the retail landscape, offering consumers unprecedented access to a vast array of products at their fingertips. Among these platforms, Amazon has emerged as a dominant force, renowned for its extensive product catalog, competitive pricing, and convenient shopping experience. However, with the sheer volume of products available and the dynamic nature of online prices, consumers face challenges in navigating the digital marketplace.

Traditionally, consumers relied on manual methods to track price changes, such as periodically revisiting product pages or utilizing price tracking tools. However, these methods are labor-intensive, time-consuming, and often ineffective in capturing real-time fluctuations. As a result, consumers may miss out on potential savings or fail to capitalize on favorable price drops.Recognizing these challenges, the Pricewatch project was conceived with the goal of revolutionizing the way consumers monitor prices on Amazon. By harnessing the power of technology and automation, Pricewatch aims to provide users with a streamlined solution for tracking price changes in real-time, thereby empowering them to make informed purchasing decisions with confidence and ease.

The landscape of online collaboration and information sharing has undergone profound transformations, and at the heart of this evolution lies the venerable Pastebin.com. Established in 2002, Pastebin.com has not merely been a platform; it has been a catalyst for a paradigm shift in how users share, discuss, and collaborate on textual information. Its emergence marked a crucial juncture in the internet's history, providing users with a streamlined mechanism to share succinct textual snippets, ranging from code snippets to general text.

**1.2 Purpose and Scope**

The primary purpose of the Pricewatch project is to develop a sophisticated and user-friendly web application that revolutionizes the way consumers track and monitor price changes on Amazon. The scope of the project encompasses a comprehensive set of functionalities aimed at providing users with a seamless and efficient price tracking experience.

At the core of Pricewatch is its ability to gather real-time price data from Amazon product pages. Leveraging advanced web scraping techniques, the application systematically retrieves pricing information for a wide range of products across various categories. This ensures that users have access to the most up-to-date and accurate price data available. Once the price data is collected, Pricewatch processes and stores it in a structured database, ensuring efficient organization and retrieval. By structuring the data in a logical manner, the application can quickly respond to user queries and requests, providing a seamless browsing experience. Pricewatch boasts a user-friendly interface that allows users to customize their price tracking experience according to their preferences. Users can easily search for products, set up notifications for specific items or price thresholds, and visualize price trends over time. The interface is designed to be intuitive and accessible, catering to users of all technical levels.

In addition to its core functionalities, Pricewatch plans to implement user account creation and login features in future iterations of the application. This will enable users to create personalized profiles, save their tracking preferences, and receive tailored recommendations based on their browsing history and preferences. By incorporating user accounts, Pricewatch aims to enhance user engagement and loyalty, providing a more immersive and personalized shopping experience.

While the initial focus of Pricewatch is on tracking prices on Amazon, the project has the potential to expand its scope to include other e-commerce platforms in the future. By integrating with additional platforms, Pricewatch can offer users a more comprehensive and holistic solution for tracking prices across a wide range of online retailers.

**1.3 Target Audience**

Pricewatch caters to a diverse range of consumers who engage in online shopping on Amazon. This includes bargain hunters, individuals who actively seek out the best deals and discounts to maximize savings. Bargain hunters are often price-sensitive and appreciate tools that help them find the lowest prices on the products they want to purchase. Additionally, Pricewatch serves frequent online shoppers, regular Amazon customers who make frequent purchases and are interested in optimizing their shopping experience. These users may have a wide range of shopping needs and preferences and benefit from tools that help them track prices across multiple products and categories. Moreover, Pricewatch appeals to tech-savvy users, individuals who are comfortable with technology and appreciate the benefits of automation and innovation in simplifying complex tasks. Tech-savvy users may be early adopters of new technologies and are likely to appreciate the advanced features and functionalities offered by Pricewatch.

Furthermore, the platform targets e-commerce enthusiasts and analysts, those interested in tracking market trends, price fluctuations, and consumer behavior on e-commerce platforms. E-commerce enthusiasts may use Pricewatch to conduct research, analyze pricing patterns, and gain insights into the online shopping landscape.Lastly, budget-conscious consumers represent another segment of Pricewatch's target audience. These individuals are mindful of their spending and seek to maximize the value of their purchases. Budget-conscious consumers may use Pricewatch to track prices on essential items, compare prices across different sellers, and identify opportunities to save money on their purchases.

By catering to the diverse needs and preferences of these target audiences, Pricewatch aims to become an indispensable tool for anyone looking to make smarter purchasing decisions and maximize value while shopping on Amazon.

.

**1.4 Motivation**

The motivation behind the Pricewatch project is multifaceted and driven by a combination of consumer needs, technological innovation, and a desire to enhance the online shopping experience. At its core, Pricewatch is motivated by the overarching goal of simplifying the online shopping journey for consumers and empowering them to make informed purchasing decisions. By automating the price tracking process, Pricewatch aims to save users valuable time and effort, allowing them to focus on finding the best deals and maximizing their savings.

Moreover, the project is motivated by a desire to leverage cutting-edge technologies to develop a sophisticated and robust application. With advancements in web scraping, data processing, and user interface design, Pricewatch seeks to create a seamless user experience that delivers accurate, real-time price information to users. By harnessing the power of technology, Pricewatch aims to stay ahead of the curve and provide users with a competitive edge in the digital marketplace Furthermore, the planned addition of user account creation and login features motivates the project by providing opportunities for further customization and personalization. By allowing users to create personalized profiles, save their tracking preferences, and receive tailored recommendations, Pricewatch aims to enhance user engagement and loyalty, ultimately becoming a trusted companion for online shoppers.

Additionally, Pricewatch is motivated by a commitment to continuous improvement and innovation. By staying abreast of emerging trends and technologies in the e-commerce space, Pricewatch aims to evolve and adapt to meet the changing needs and preferences of its users. Whether through the addition of new features, integration with additional e-commerce platforms, or optimization of existing functionalities, Pricewatch remains dedicated to providing users with a best-in-class shopping experience.

Overall, the motivation behind the Pricewatch project lies in its potential to simplify the online shopping experience, empower consumers, and harness technology to deliver value in the digital marketplace. By addressing consumer needs, leveraging technological innovation, and fostering a culture of continuous improvement, Pricewatch strives to become a trusted and indispensable tool for online shoppers worldwide.

**CHAPTER 2**

**LITERATURE REVIEW**

The literature surrounding web scraping and price tracking solutions provides valuable insights into the evolving landscape of e-commerce and data extraction techniques. Various studies and articles explore the challenges and opportunities inherent in web scraping, shedding light on the methods, tools, and ethical considerations involved in extracting data from online sources. Price tracking solutions, in particular, have garnered significant attention due to the increasing importance of pricing information in consumer decision-making and market analysis.

Researchers have examined the effectiveness of different web scraping techniques, ranging from traditional screen scraping methods to more advanced browser automation tools like Puppeteer and Selenium. Comparative analyses of these techniques highlight their respective strengths and limitations in terms of performance, scalability, and ease of use. Additionally, studies explore the impact of anti-scraping measures deployed by websites, such as rate limiting, CAPTCHA challenges, and IP blocking, on the success of scraping operations.

Moreover, the literature offers insights into the diverse applications of web scraping in price tracking and market analysis. Researchers have investigated the use of scraped data for monitoring price fluctuations, identifying pricing strategies employed by competitors, and predicting market trends. Case studies and real-world examples demonstrate how price tracking solutions empower consumers to make informed purchasing decisions by providing access to timely and accurate pricing information across various online retailers.

Ethical and legal considerations surrounding web scraping are also a prominent topic of discussion in the literature. Scholars examine the ethical implications of scraping data from websites without permission, addressing concerns related to data privacy, intellectual property rights, and fair use. Legal analyses explore the legal frameworks governing web scraping activities, including copyright laws, terms of service agreements, and data protection regulations, to provide guidance for developers and researchers navigating the complex legal landscape.

**2.1 Existing Price Tracking Solutions**

In today's dynamic e-commerce landscape, consumers have access to a plethora of tools and platforms aimed at assisting them in tracking prices of products across various online retailers. These solutions encompass a wide range of functionalities, from simple browser extensions to comprehensive price comparison websites. Browser extensions such as CamelCamelCamel, Honey, and PriceBlink offer users the ability to track price changes on specific products directly within their web browsers. These extensions typically display historical price data, provide price drop alerts, and offer browser integration for seamless use during online shopping sessions. Standalone applications like PriceWatcher and Keepa provide similar functionalities in a standalone application format, catering to users who prefer a dedicated interface for price tracking.

Price comparison websites, on the other hand, aggregate price data from multiple retailers and present it to users in a unified interface, allowing for easy comparison of prices across different sellers. Websites such as PriceGrabber, Google Shopping, and Shopzilla are popular examples of price comparison platforms that provide users with access to a vast array of products and sellers. These platforms typically offer advanced filtering options, user reviews, and additional features such as price alerts and price history charts to aid users in making informed purchasing decisions.

A comparative analysis of these existing solutions reveals varying levels of functionality, accuracy, and user experience. Factors such as data freshness, notification options, and compatibility with different e-commerce platforms influence users' preferences and choices when selecting a price tracking solution. For example, while some tools may offer real-time price tracking and customizable alerts, others may be limited in scope or prone to inaccuracies due to delays in data updates or incomplete product coverage. Additionally, user interface design, ease of use, and integration with popular e-commerce platforms play crucial roles in determining the overall user satisfaction and adoption rates of these solutions.

**2.2 Web Scraping Techniques**

`Web scraping, a fundamental technique in data extraction, empowers developers and researchers to retrieve valuable information from websites. Its applications span various domains, including price tracking, market analysis, and content aggregation. Web scraping involves programmatically accessing web pages, extracting relevant data, and structuring it for analysis or storage. Among the plethora of techniques available, Puppeteer stands out as a prominent tool in the web scraping arsenal.

Puppeteer, a Node.js library developed by Google, offers a high-level API for controlling headless Chrome or Chromium browsers. Its versatility and ease of use make it a preferred choice for developers embarking on web scraping projects. By automating web browsing tasks, Puppeteer enables seamless navigation through web pages, form submissions, and data extraction. Its headless mode, which operates without a graphical user interface, ensures efficiency and reliability in scraping operations.

Despite its advantages, ethical and legal considerations loom large in the realm of web scraping. Developers must navigate a complex landscape of terms of service, copyright laws, and website policies to ensure compliance. Scraping data without explicit permission from website owners may result in legal repercussions, such as copyright infringement or breach of contract. To mitigate these risks, developers must exercise caution, respect robots.txt directives, and adopt responsible scraping practices.

Furthermore, developers should prioritize data privacy and security when handling scraped data. Sensitive information obtained through scraping, such as personal details or proprietary content, requires careful handling to prevent unauthorized access or misuse. Implementing encryption, access controls, and data anonymization techniques can safeguard against data breaches and privacy violations.

In addition to legal and ethical considerations, developers must contend with technical challenges inherent in web scraping. Websites may employ anti-scraping measures, such as rate limiting, CAPTCHA challenges, or IP blocking, to deter automated access. To circumvent these obstacles, developers leverage various strategies, including IP rotation, proxy servers, and CAPTCHA solving services. IP rotation involves dynamically switching between different IP addresses to evade detection and avoid being blacklisted by target websites. Proxy servers route scraping requests through intermediary servers with diverse IP addresses, masking the origin of the requests and enhancing anonymity. CAPTCHA solving services automate the process of solving CAPTCHA challenges encountered during scraping, enabling uninterrupted data extraction.

Despite these challenges, web scraping remains a powerful tool for extracting valuable insights from the vast expanse of the web. By leveraging tools like Puppeteer and adhering to ethical guidelines, developers can unlock a wealth of data to fuel innovation and discovery across diverse domains.

**2.3 Data Storage and Management**

Effective data storage and management are imperative components of any web scraping endeavor, ensuring that collected data is organized, accessible, and secure. MongoDB, a popular NoSQL database, emerges as a favored choice among developers for its flexibility and scalability. MongoDB's document-oriented approach allows for the storage of structured and unstructured data in a format conducive to efficient querying and analysis. Its support for horizontal scalability and replica sets enables seamless scaling of databases to accommodate growing data volumes and user demand.

In the context of web scraping, best practices for data handling are paramount to maintaining data integrity and reliability. Data normalization, a fundamental technique in database design, reduces redundancy and improves data consistency by organizing data into logical tables and eliminating duplicate entries. Indexing enhances query performance by creating data structures that facilitate rapid data retrieval, especially for frequently accessed fields or criteria. Additionally, backup procedures safeguard against data loss or corruption, offering peace of mind in the event of system failures or disasters.

Furthermore, data privacy and security are critical considerations in the storage and management of scraped data. Personal information and sensitive data obtained through scraping must be handled with care to prevent unauthorized access or misuse. Encryption techniques, such as SSL/TLS encryption for data in transit and encryption-at-rest for data at rest, provide robust safeguards against data breaches and unauthorized access. Access controls and authentication mechanisms restrict access to sensitive data, ensuring that only authorized users can view or modify data.

Moreover, compliance with regulatory requirements, such as GDPR and CCPA, is essential when handling scraped data, particularly personal or sensitive information. Developers must adhere to legal guidelines governing data collection, processing, and storage to protect user privacy and avoid legal repercussions. This entails obtaining explicit consent from users before collecting personal data, providing transparency regarding data usage and retention practices, and honoring user rights to access, rectify, or delete their data.

In summary, effective data storage and management are foundational elements of successful web scraping projects, enabling developers to harness the full potential of collected data while safeguarding against privacy breaches and legal risks. By leveraging robust database technologies like MongoDB and implementing best practices for data handling, developers can ensure the reliability, security, and compliance of their scraping operations, ultimately empowering them to derive valuable insights from the web with confidence and integrity.

**2.4 Data Storage and Management**

IP rotation and proxy management are indispensable strategies employed by web scrapers to circumvent obstacles and enhance the effectiveness of data extraction processes. These techniques play a crucial role in mitigating the risk of detection and blocking by target websites, enabling web scrapers to operate smoothly and efficiently in dynamic online environments.

At the heart of IP rotation lies the concept of dynamically switching between different IP addresses during scraping operations. By rotating IP addresses, web scrapers can distribute scraping requests across multiple endpoints, reducing the likelihood of triggering rate limits, CAPTCHA challenges, or IP bans imposed by target websites. This proactive approach helps maintain a low profile and minimizes the risk of detection, allowing scrapers to access and extract data without interruptions or disruptions.

Proxy servers serve as intermediaries between web scrapers and target websites, masking the true origin of scraping requests and enhancing anonymity. These servers route scraping traffic through diverse IP addresses, making it challenging for websites to trace and block scraping activity based on individual IP addresses. Proxy rotation further enhances this strategy by periodically switching between different proxy servers, diversifying the pool of IP addresses available to scrapers and further obscuring their digital footprint.

In addition to IP rotation and proxy management, web scrapers leverage techniques such as user agent rotation and CAPTCHA solving to overcome anti-scraping measures deployed by target websites. User agent rotation involves periodically changing the HTTP user agent header sent with scraping requests, mimicking the behavior of different web browsers and devices. This helps prevent websites from identifying and blocking scraping activity based on user agent signatures, enhancing the stealth and resilience of scraping operations.

CAPTCHA solving services automate the process of solving CAPTCHA challenges encountered during scraping, enabling scrapers to bypass CAPTCHA protections and continue extracting data seamlessly. These services leverage machine learning algorithms and human solvers to analyze and solve CAPTCHA puzzles, enabling scrapers to navigate through CAPTCHA-protected pages without human intervention.

Overall, IP rotation and proxy management are essential components of the web scraping toolkit, enabling developers to navigate the complexities of the online landscape and extract valuable data with efficiency and reliability. By adopting these strategies and leveraging complementary techniques, web scrapers can overcome obstacles, evade detection, and unlock insights from the web with confidence and effectiveness.

**2.5 Scheduling and Automation**

Scheduling and automation are integral facets of web scraping operations, facilitating the efficient and reliable extraction of data from target websites. These processes streamline repetitive tasks, ensuring that scraping tasks are executed at predetermined intervals without manual intervention.

At the core of scheduling and automation lies the concept of cron jobs, a time-based job scheduler commonly used in Unix-like operating systems. Cron jobs allow developers to schedule tasks to run periodically, such as data scraping scripts, data processing routines, or system maintenance tasks. By defining the schedule for each task using cron syntax, developers can automate the execution of scraping operations according to specific time intervals, such as hourly, daily, or weekly.

Cron jobs offer simplicity and reliability, making them well-suited for automating routine tasks in web scraping applications. Developers can define cron jobs to initiate scraping scripts at regular intervals, ensuring that price data is collected and updated consistently without manual intervention. This automation reduces the burden on developers and ensures that scraping operations run smoothly and reliably, even in the absence of human oversight.

While cron jobs provide a robust solution for scheduling and automation, alternative approaches such as serverless computing and third-party automation services offer additional flexibility and scalability. Serverless computing platforms like AWS Lambda and Google Cloud Functions allow developers to deploy and run code without managing servers or infrastructure. This serverless architecture is ideal for event-driven tasks like web scraping, where resources are provisioned dynamically based on demand, reducing costs and complexity.

Third-party automation services provide pre-built solutions for web scraping and data extraction, offering developers ready-to-use tools and APIs for integrating scraping functionality into their applications. These services offer a range of features, including scheduling, data extraction, and data processing, allowing developers to focus on building their scraping logic without worrying about infrastructure or scalability.

In summary, scheduling and automation play a pivotal role in the success of web scraping operations, enabling developers to automate repetitive tasks and ensure the consistent and reliable extraction of data from target websites. Whether through cron jobs, serverless computing, or third-party automation services, developers have a range of options to choose from when designing their scraping workflows, each offering its own advantages and considerations. By leveraging these tools effectively, developers can streamline their scraping operations and unlock valuable insights from the web with efficiency and ease.

**CHAPTER 3**

**REQUIREMENT ANALYSIS**

**3.1. Introduction**

The requirement analysis phase is a critical precursor to the development of our text-sharing platform. Through a meticulous examination of functional and non-functional aspects, this analysis aims to define the project's scope, functionalities, and performance characteristics. The requirements outlined here serve as the foundation upon which the development team will build a robust and user-centric platform.

**3.2 Functional Requirements**

**3.2.1 User Authentication**

Implement a secure and user-friendly authentication system.

* User registration and login.
* Secure login with JWT Authentication.

**3.2.2 Paste Creation and Editing**

Enable users to create, edit, and manage text snippets.

* Syntax highlighting for various programming languages.
* Rich-text editing capabilities for general text snippets.
* Version history and revision tracking for collaborative editing.
* Organizational features such as folders or categories.

**3.2.3 Privacy Settings**

Provide users with control over the visibility of their pastes.

* Public, private visibility options for pastes.
* Customizable expiration options for time-limited sharing.

**3.2.4 Search and Discovery**

Enhance user experience through efficient content discovery.

* Search functionality based on paste id
* Shows Users paste as well as most recent public pastes

**3.2.5 Sharing**:

Allow users to share pastes among their friends.

* Share code with links or with paste id
* Can also share the snippet with copy button for easy sharing.

**3.3 Non-Functional Requirements**

**3.3.1 Performance**

Ensure responsive and efficient platform performance.

* Page load times within a specified threshold.
* Support for a scalable number of concurrent users.

**3.3.2 Security**

Prioritize the security of user data and interactions.

* Encryption of data in transit using HTTPS.
* Implementation of secure coding practices to mitigate common vulnerabilities.

**3.3.3 Scalability**

Design the platform to scale with increasing user demands.

* Architecture supporting horizontal scalability.
* Efficient database indexing and query optimization.

**3.3.4 Usability**

Ensure an intuitive and user-friendly interface.

* Accessibility features for diverse user needs.
* Consistent and cohesive design elements.

**3.3.5 Availability**

Minimize downtime and ensure platform availability.

* Implementing redundant servers and failover mechanisms.
* Regular maintenance windows communicated in advance.

This requirement analysis provides a comprehensive overview of both functional and non-functional aspects, ensuring a clear and detailed understanding of the platform's scope and performance expectations. These requirements will guide the development team in creating a robust, secure, and user-centric text-sharing platform.

**3.4 Use Case Diagram**

Below is the use case diagram representing all the possible interactions of the user with the system.

In our web application, we can have two types of user, an authenticated user and a user that is not logged in.

According to which, that user may have different ways to interact with the system. As you can see in the Use case Diagram, A logged in user can perform all the operations while, a user that is not logged in can only perform, certain functions, namely creating a paste, viewing a paste, searching for a paste, and registering.

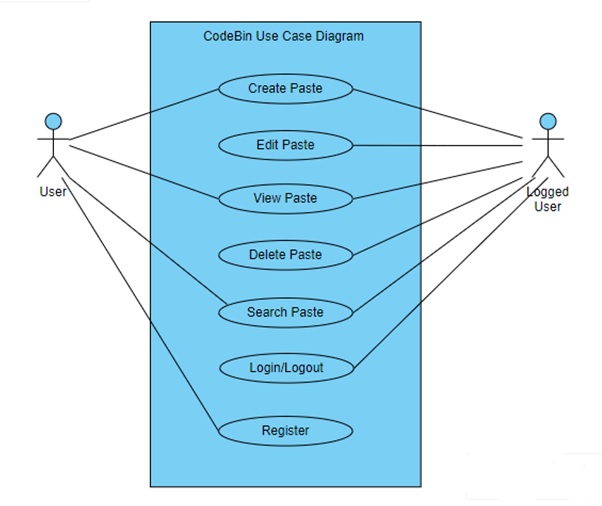


Fig 3.1 Use Case Diagram

**CHAPTER 4**

**SYSTEM DESIGN**

In the system design phase, we delve into the architectural blueprint and detailed schematics of our text-sharing platform built on the MERN (MongoDB, Express.js, React, Node.js) stack. This phase is pivotal for ensuring a scalable, efficient, and cohesive system that aligns with the project's objectives.

**4.1 Architecture and System Overview**

**4.1.1 Architecture**

The platform adopts a client-server architecture. Clients interact with the server through a RESTful API for data exchange. The server-side application, built with Node.js and Express.js, serves as the backend logic handler.

**4.1.2 System Overview**

Client is built using React, constitute the frontend of the application. Node.js, along with Express.js, manages server-side logic and API endpoints. MongoDB, a NoSQL database, stores and retrieves data efficiently.

**4.1.3 Key Components**

**Client-Side (React)**

Components for paste creation, editing, and viewing. User authentication and profile management interfaces. Search and discovery components for efficient content exploration.

**Server-Side (Node.js and Express.js)**

RESTful API handling for CRUD operations on pastes and user-related functionalities. Middleware for authentication, error handling, and data validation. Integration with the MongoDB database for data storage.

**Database (MongoDB)**

Collections for users, pastes. Indexing for efficient querying and data retrieval. Adherence to MongoDB best practices for scalability.

**4.2 Database Design and Schema**

**4.2.1 MongoDB Collections**

**Users Collection**:User details such as username, email, hashed password, and authentication tokens. User preferences and profile information.



Fig 4.1 MongoDB User Collection

**Pastes Collection:** Paste content, metadata, and visibility settings. Version history and revision information for collaborative editing.



Fig 4.2 MongoDB Paste Collection

**4.3 ER Diagram**

ER diagram to show the relation between a registered user and the pastes he creates.

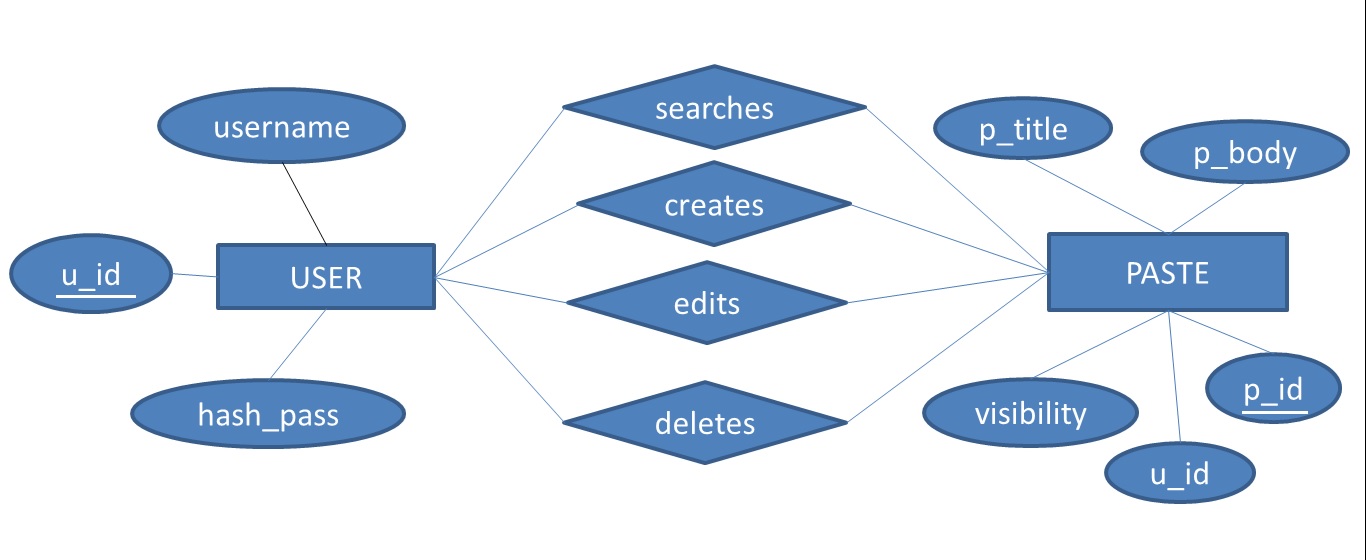


Fig 4.3 ER Diagram for CodeBin

**4.4 Data Flow Diagram**

**4.4.1 User Authentication Flow**

User initiates authentication by providing credentials. Client sends a request to the server for authentication. Server validates credentials, generates a JWT, and sends it back to the client. Subsequent requests from the client include the JWT for authentication.

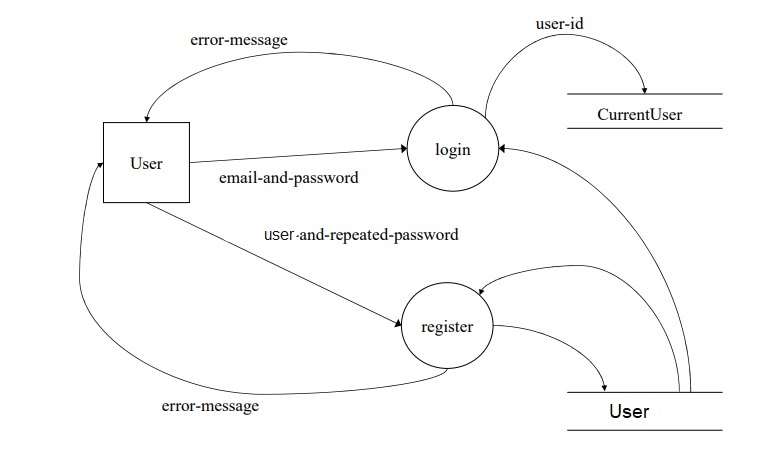
****

Fig 4.4 DFD Level 0: Showing User Registration

**4.4.2 Paste Creation and Editing Flow**

User creates or edits a paste through the frontend interface. Client sends a request to the server's API endpoint for paste creation or update. Server processes the request, updates the database, and returns the result to the client.

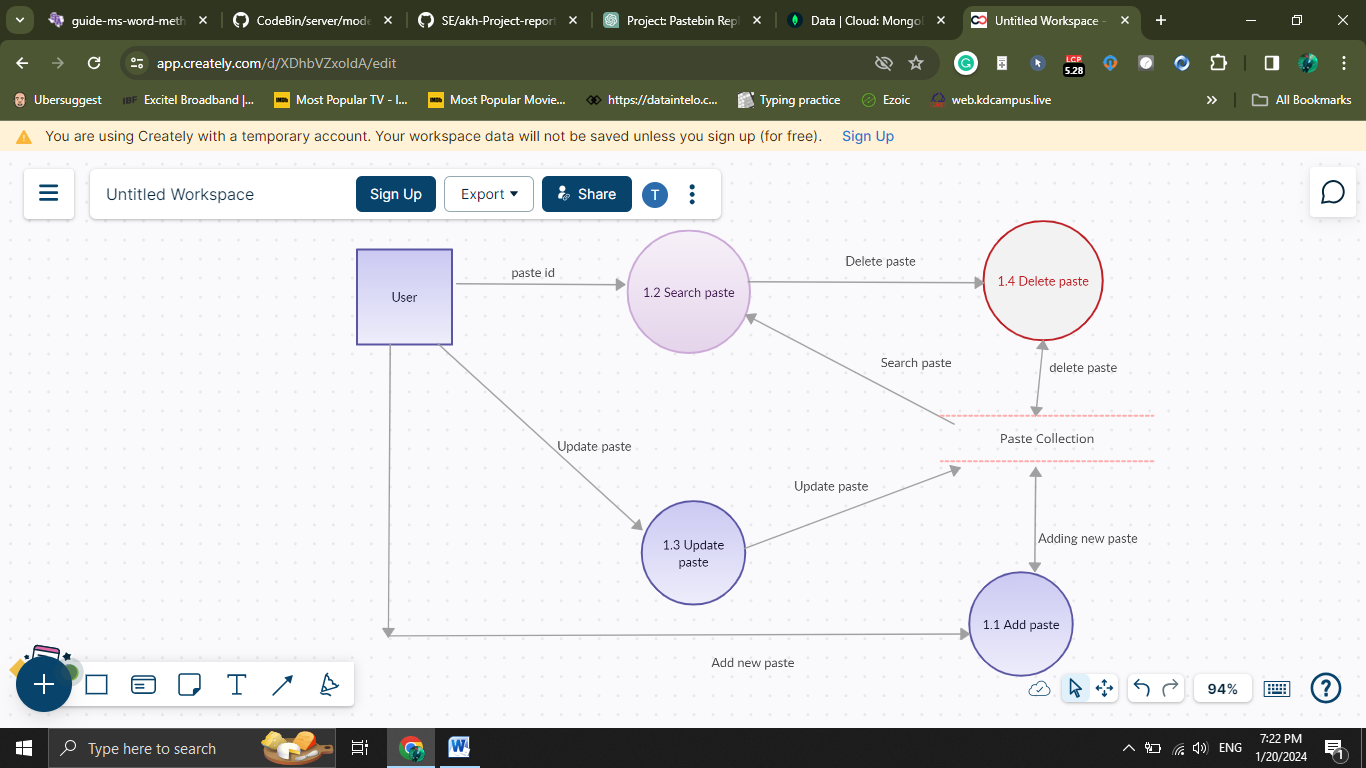


Fig 4.5 DFL Level 1: Showing how Paste is managed

**CHAPTER 5**

**IMPLEMENTATION**

In the implementation phase of the PriceWatch project, the focus is on translating the conceptual design into a fully functional software application. This phase involves a meticulous series of tasks aimed at setting up the development environment, developing both frontend and backend components, implementing web scraping functionality, configuring IP rotation services, and automating recurring tasks using cron job scheduling.

Setting up the development environment for PriceWatch involves selecting and configuring tools, frameworks, and libraries that are best suited for the project's requirements. Developers ensure seamless collaboration by setting up code editors, version control systems, and development environments tailored to the project's needs, fostering an efficient workflow throughout the development process.

Frontend development plays a crucial role in crafting an engaging and intuitive user interface for PriceWatch users. Leveraging technologies like Next.js, developers design and implement key components such as product listings, price history charts, and user notification settings. The goal is to create a seamless browsing experience that meets the expectations of users while ensuring ease of navigation and interaction.

Simultaneously, backend development efforts focus on establishing the core functionality and logic of the application. This includes setting up server environments, defining API endpoints, and integrating with databases like MongoDB for storing and retrieving price tracking data. The backend infrastructure is designed to be scalable, secure, and performant, enabling seamless communication between frontend and backend components.

Web scraping functionality is a critical aspect of PriceWatch, enabling the automated extraction of price data from target websites such as Amazon. Using tools like Puppeteer, developers create scraping scripts that navigate through web pages, extract relevant price information, and store it in the application's database. Techniques for handling pagination, rate limits, and dynamic content loading are implemented to ensure reliable and efficient scraping operations.

Integration of IP rotation services like Bright Data enhances the reliability and resilience of the scraping infrastructure by dynamically rotating IP addresses. Developers configure IP rotation settings to mitigate the risk of detection and blocking by target websites, ensuring continuous and uninterrupted data collection for PriceWatch users.

Furthermore, cron job scheduling is utilized to automate recurring tasks such as data scraping and database maintenance. Developers configure cron jobs to execute scraping scripts at predefined intervals, ensuring that price data is collected and updated regularly without manual intervention. This automated approach streamlines the data collection process and enhances overall efficiency.

**5.1 Development Environment**

In the implementation phase of PriceWatch, configuring the development environment is essential for facilitating seamless development and deployment of the application. Developers focus on selecting and setting up the necessary tools and frameworks tailored to the project's requirements, ensuring compatibility and efficiency throughout the development process. With Next.js chosen as the framework for serverless functions, developers streamline the setup process to maximize productivity and optimize resource utilization. The initial step in configuring the development environment involves selecting a suitable code editor and version control system. Developers commonly opt for popular editors like Visual Studio Code or Sublime Text, which offer robust features and support for various programming languages. Additionally, version control systems like Git are utilized to track changes, collaborate with team members, and manage code repositories effectively.

Once the code editor and version control system are in place, developers proceed to set up the development environment to support Next.js for serverless functions. Next.js offers a comprehensive platform for building server-rendered React applications with minimal setup and configuration. With built-in support for serverless functions, Next.js simplifies the process of deploying server-side logic to cloud platforms like Vercel or AWS Lambda. Integration with additional tools and services further enhances the development environment's capabilities. While Express.js may not be utilized for server-side routing and middleware in PriceWatch, developers may still leverage tools like ESLint for code linting, Webpack for bundling assets, and npm or Yarn for package management. These tools help enforce coding standards, optimize build performance, and manage project dependencies effectively.

Furthermore, developers focus on configuring the development environment to seamlessly integrate with external services and APIs. This may include setting up environment variables, configuring API endpoints, and integrating with third-party services like MongoDB for data storage. By ensuring smooth integration with external dependencies, developers lay the groundwork for building a robust and scalable price tracking application.

In summary, configuring the development environment for PriceWatch involves selecting and setting up the necessary tools and frameworks to support Next.js for serverless functions. By leveraging Next.js' built-in capabilities for server-side rendering and serverless functions, developers streamline the development process and optimize resource utilization. Through meticulous planning and attention to detail, the development team ensures a seamless and efficient workflow for building and deploying the price tracking application.A JavaScript library for building user interfaces.

**5.2 Frontend**

In the Frontend Development phase of PriceWatch, the focus is on implementing the Next.js framework to create an intuitive and responsive user interface. Next.js, with its built-in features for server-side rendering and routing, provides a solid foundation for building dynamic web applications. Developers leverage Next.js to design and implement key components that enhance user experience and functionality without compromising performance.

The implementation of Next.js begins with setting up the project structure and configuring routing to define the navigation flow of the application. Developers create pages for different sections of PriceWatch, such as product listings, price history charts, and user settings. Each page is associated with a corresponding URL route, allowing users to navigate seamlessly between different sections of the application.

Next, developers design and implement UI components to display relevant information and enable user interactions. Components such as product cards, price charts, and notification settings are carefully crafted to provide users with a visually appealing and intuitive interface. Responsive design principles are applied to ensure that the application looks and functions seamlessly across various devices and screen sizes.

Integration with external APIs and services further enhances the functionality of the frontend. Developers leverage APIs to retrieve real-time price data from external sources, such as Amazon or other e-commerce platforms. This data is then dynamically rendered on the frontend, allowing users to view up-to-date pricing information for their desired products.

Additionally, developers implement client-side logic using JavaScript to enhance interactivity and responsiveness. This includes features such as filtering and sorting product listings, updating price charts in real-time, and managing user preferences and notifications. Client-side validation is also implemented to ensure data integrity and provide feedback to users when interacting with forms or input fields.

To optimize performance, developers leverage Next.js' built-in features for code splitting and prefetching. Code splitting allows the application to load only the necessary JavaScript code for each page, reducing initial page load times and improving overall performance. Prefetching automatically fetches data for subsequent pages in the background, ensuring a smooth and uninterrupted browsing experience for users.

**5.3 Backend**

In the Backend Development phase of PriceWatch, meticulous attention is given to setting up the server environment and integrating with MongoDB to establish a robust backend infrastructure. This phase involves configuring the server environment, defining API endpoints, and implementing data storage and retrieval mechanisms to support the application's functionality.

The first step in backend development is setting up the server environment to host the PriceWatch application. Developers select a suitable hosting provider or cloud platform, such as AWS, Google Cloud, or Heroku, and configure the server environment to meet the application's requirements. This includes provisioning server resources, installing necessary software dependencies, and configuring security settings to ensure the integrity and availability of the application.

Integration with MongoDB, a popular NoSQL database, is essential for storing and retrieving price tracking data in PriceWatch. Developers create a MongoDB database instance and configure the connection settings to establish a secure and reliable connection between the application and the database server. This includes specifying connection strings, authentication credentials, and database access permissions to ensure data integrity and confidentiality.

Once the database connection is established, developers define API endpoints to enable communication between the frontend and backend components of the application. These endpoints serve as interfaces for performing CRUD (Create, Read, Update, Delete) operations on the database, allowing users to interact with price tracking data stored in MongoDB. Developers implement request handling logic to process incoming requests, validate input data, and perform database operations accordingly.

Data models are defined to represent the structure and relationships of the data stored in MongoDB. Developers create schemas to define the properties and constraints of each data entity, such as products, price history records, and user preferences. This ensures consistency and integrity of the data stored in the database, enabling efficient querying and retrieval of relevant information.

To optimize performance and scalability, developers implement indexing and query optimization techniques in MongoDB. Indexes are created on frequently queried fields to accelerate data retrieval and improve query performance. Additionally, developers optimize queries to minimize resource usage and reduce query execution times, ensuring optimal performance even under heavy load conditions. Backend Development phase of PriceWatch involves setting up the server environment and integrating with MongoDB to establish a robust backend infrastructure. By configuring the server environment, defining API endpoints, and implementing data storage and retrieval mechanisms, developers ensure seamless communication between the frontend and backend components of the application, enabling users to interact with price tracking data effectively and efficiently.

**5.4 Web Scraping**

In the Web Scraping phase of PriceWatch, developers harness the capabilities of Puppeteer to automate the extraction of price data from target websites such as Amazon. Puppeteer, a Node.js library, empowers developers to control headless Chrome or Chromium browsers, facilitating the simulation of user interactions and data extraction from web pages programmatically. Script examples are crafted to showcase the methodology behind scraping scripts using Puppeteer's API. These examples elucidate the process of launching a headless browser instance, navigating to the target URL, and selecting specific elements containing price data. Techniques involving CSS selectors or XPath expressions are employed to locate and extract product prices from product listings or detail pages.

Pagination poses a challenge in web scraping, necessitating the implementation of logic to traverse through paginated pages and extract price data systematically. Developers engineer solutions that involve navigating through pagination links, scrolling down pages to load additional content dynamically, and strategically waiting for content to load before extracting data.

Moreover, strategies are devised to handle rate limits imposed by target websites to prevent misuse of resources. These strategies encompass implementing delays between requests to emulate human behavior, restricting the number of concurrent requests to avoid server overload, and employing retry mechanisms with exponential backoff to mitigate transient network errors. Advanced scraping techniques are also employed, utilizing Puppeteer's features for request interception and manipulation. Developers intercept network requests to circumvent CAPTCHA challenges, navigate anti-scraping mechanisms, and manipulate user agents or IP addresses to evade detection and blocking by target websites.

To ensure robustness and efficiency, developers implement comprehensive error handling and logging mechanisms. These mechanisms capture and manage exceptions, log errors, warnings, and debugging information, aiding in troubleshooting and diagnosing issues throughout the development and production phases.

In essence, the Web Scraping phase of PriceWatch revolves around leveraging Puppeteer's capabilities to automate data extraction from target websites efficiently. Through meticulous script crafting, thoughtful handling of pagination and rate limits, and the implementation of advanced scraping techniques, developers ensure the reliable and accurate collection of price data to empower PriceWatch users with valuable insights.

**5.5 IP Rotation**

In the realm of PriceWatch, orchestrating the integration of IP rotation with Bright Data stands as a pivotal strategy in fortifying the scraping infrastructure's resilience and reliability. This fusion of technologies necessitates a methodical approach, weaving through intricate steps and nuanced configuration details aimed at orchestrating the seamless rotation of IP addresses. The initiation of this integration voyage commences with the acquisition of access credentials from Bright Data's ecosystem. Developers embark on a journey of registering for a Bright Data account, securing API keys or authentication tokens that serve as the gateway to the platform's array of functionalities. Subsequently, the integration fabric is woven as developers meticulously infuse the scraping scripts with the ability to communicate with Bright Data's IP rotation service. This involves a careful dance of configuring proxy settings within the scraping infrastructure, ensuring a secure and seamless connection to Bright Data's extensive network of proxies.

Moreover, the integration journey delves into the realm of dynamic IP address rotation, a symphony orchestrated to thwart the detection and blockade efforts of target websites. Developers craft algorithms that orchestrate the rotation of IP addresses at strategic intervals, a virtuoso performance aimed at preserving anonymity and continuity in data extraction endeavors.

As the integration unfolds, developers navigate the labyrinth of configuration settings within Bright Data's platform, fine-tuning parameters to optimize performance and ensure compliance with target website policies. This entails a delicate balancing act of specifying geographic regions, refining request headers, and adjusting request rates to harmonize with the rhythms of the digital landscape. Beyond the realm of configuration, vigilant monitoring and iterative refinement emerge as guiding principles in the integration odyssey. Developers stand vigilant, scrutinizing scraping performance metrics and error logs, ever-ready to adapt and optimize in response to evolving challenges and opportunities.

In essence, the integration of IP rotation with Bright Data in the PriceWatch ecosystem transcends mere technical integration; it embodies a symphony of meticulous orchestration, harmonizing technology, strategy, and vigilance to safeguard the integrity and efficiency of data collection endeavors.

**5.6 Cron Job Scheduling**

In the PriceWatch ecosystem, the orchestration of cron job scheduling stands as a cornerstone in automating recurring tasks, facilitating the seamless execution of critical operations such as data scraping and database maintenance. This endeavor unfolds through a meticulous process, interweaving intricate steps and configurations aimed at orchestrating the rhythmic execution of tasks with precision and efficiency.

At the heart of this orchestration lies the configuration of cron jobs within the PriceWatch environment. Developers delve into the intricacies of cron syntax, crafting precise schedules that dictate when each task should be executed. This involves specifying minute, hour, day of the month, month, and day of the week parameters, orchestrating a symphony of time-based triggers that govern the cadence of task execution. With schedules defined, developers proceed to integrate cron jobs into the PriceWatch application, embedding logic within the codebase to trigger the execution of tasks at predefined intervals. This integration imbues the application with the ability to autonomously initiate scraping scripts, database cleanup routines, and other essential operations without manual intervention, fostering a seamless and hands-free workflow.

Furthermore, developers implement mechanisms for error handling and logging within cron job scripts, ensuring robustness and reliability in task execution. This involves capturing and logging errors, warnings, and debugging information to facilitate troubleshooting and diagnosis in the event of anomalies or failures. As the orchestration unfolds, developers monitor the execution of cron jobs vigilantly, scrutinizing logs and performance metrics to gauge the efficacy of task scheduling and execution. They stand ready to adapt and optimize schedules and scripts in response to evolving requirements, fine-tuning parameters to maximize efficiency and minimize resource utilization. Moreover, developers leverage the flexibility of cron job scheduling to orchestrate complex workflows and dependencies between tasks. This may involve chaining together multiple tasks in sequence, defining dependencies between tasks, and implementing logic to handle edge cases and contingencies gracefully.

In essence, the orchestration of cron job scheduling within the PriceWatch ecosystem transcends mere task automation; it embodies a symphony of precision, efficiency, and reliability, orchestrating the rhythmic execution of critical operations with finesse and mastery. Through meticulous configuration, integration, and monitoring, developers empower PriceWatch with the ability to navigate the temporal landscape seamlessly, ensuring the continuity and efficiency of data-driven endeavors.nscen

**CHAPTER 6**

**TESTING**

In the testing phase, the goal is to ensure the reliability, functionality, and performance of the text-sharing platform. This involves employing various testing methodologies to identify and address potential issues, bugs, and ensure the platform meets the specified requirements.

**6.1 Testing Methodologies**

**6.1.1 Unit Testing**

Verify individual components and functions. Utilizing testing frameworks like Jest for React components and Mocha for backend functions. Test cases covered different scenarios and edge cases.

**6.1.2 Integration Testing**

Evaluate interactions between components. We used tools like Postman for backend API testing. This ensured smooth communication between the frontend and backend.

**6.1.3 End-to-End (E2E) Testing**

Validate entire application flows. Tested critical user journeys, including authentication, paste creation, and commenting.

**6.1.4 Performance Testing**

Assess system responsiveness and scalability. Performance Testing is achieved using tools like Apache JMeter or Artillery to simulate user loads. Evaluated and optimize database query performance.

**6.1.5 Security Testing**

Identify and address security vulnerabilities. This is done by conducting security audits and penetration testing. Focus on common issues like SQL injection and secure data transmission.

**6.1.6 Usability Testing**

Evaluate user interface and experience. Conducted usability tests with real users for feedback on design and navigation which aims to enhance user-friendliness.

**6.1.7 Regression Testing**

Ensure new changes don't impact existing functionalities. We setup automated regression testing using tools like Selenium which runs tests with each deployment to catch potential issues.

**6.1.8 Exploratory Testing:**

Identify unexpected issues. This is done by the testers (beta/alpha) who explore the application freely and report any bugs/issues. These bugs/issues are then documented and reported to the admin for improvement and then later fixed in the new updates.

**6.2 Test Cases**

Table 1: Showing Test Cases with their description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Test Steps** | **Expected Result** | **Pass/Fail** |
| TC\_001 | User Registration | 1. Navigate to the Registration Page | Registration form is displayed | Pass |
|  |  | 2. Enter valid user details | User is registered Successfully. | Pass |
|  |  | 3. Submit the Registration form. | User is redirected to the login page. | Pass |
| TC\_002 | User Login | 1. Navigate to the login page. | Login form is displayed | Pass |
|  |  | 2. Enter Valid login credentials. | User is logged in and redirected to the dashboard | Pass |
|  |  | 3. Submit the login form. | User can now see his own pastes | Pass |
| TC\_003 | Paste Creation | 1. Navigate to create Paste. | Paste creation form is displayed. | Pass |
|  |  | 2. Enter Paste content and set the visibility. | Paste is successfully created. | Pass |
| TC\_004 | Editing a Paste | 1. Navigate to edit Page form by clicking edit button. | Paste edit form is displayed with current content | Pass |
|  |  | 2. Modify the content and submit the form | Paste is successfully updated | Pass |
| TC\_005 | Search Functionality | 1. Enter the Paste Id | Search result is displayed | Pass |
|  |  | 2. If incorrect Paste Id | 404 page is displayed | Pass |

**CHAPTER 7**

**CONCLUSION**

The development and testing phases of our text-sharing platform CodeBin have provided valuable insights into the creation of a robust and user-friendly application. This endeavor aimed to replicate the core functionalities of popular platforms like Pastebin.com, focusing on user registration, paste creation, collaboration features, and effective search functionality.

**7.1 Achievements and Highlights**

**User Registration and Authentication**: The user registration and authentication processes have been successfully implemented, ensuring secure access to the platform.

**Paste Creation / Editing / Deleting**: Users can seamlessly create, edit, and manage their pastes, fostering a collaborative environment for content sharing.

**Search Functionality**: The search functionality enables efficient content discovery, enhancing the overall user experience.

**7.2 Challenges Overcome**

**Security Measures:** Robust security measures, including secure authentication and data encryption, were implemented to safeguard user information.

**7.3 Testing and Quality Assurance**

Rigorous testing methodologies, including unit testing, integration testing, and end-to-end testing, were employed to ensure the reliability and functionality of the platform. Performance testing identified potential areas for optimization, contributing to a smoother user experience.

**7.4 Future Enhancements**

Consideration for real-time collaboration features, such as live editing and instant commenting, could be explored in future iterations.

Further optimization for scalability, particularly in managing large datasets and concurrent user loads, will be a focus for ongoing improvements.

**7.5 Lessons Learned**

The project emphasized the importance of iterative development and continuous testing, allowing for early detection and resolution of potential issues.

Collaboration and effective communication among team members played a pivotal role in overcoming challenges and ensuring project success.

**CHAPTER 8**

**REFERENCES**

**1. MongoDB - NoSQL Database**

**Website**: <https://www.mongodb.com/>

**2. Mongoose** - MongoDB Object Data Modeling (ODM) Library for Node.js

**Website**: <https://mongoosejs.com/>

**3**. **React.js** - JavaScript Library for Building User Interfaces

**Website**: <https://reactjs.org/>

**4. Node.js -** JavaScript Runtime for Server-Side Development

**Website**: <https://nodejs.org/>

**5. Express.js** - Web Application Framework for Node.js

**Website**: <https://expressjs.com/>

**6. CORS (Cross-Origin Resource Sharing)** - Middleware for Express.js

**GitHub Repository**: <https://github.com/expressjs/cors>

**7. JSON Web Tokens (JWT)** - Secure Authentication Mechanism

**Website**: <https://jwt.io/>

**8.** **Bcrypt** - Password Hashing Library

**GitHub Repository:** <https://github.com/kelektiv/node.bcrypt.js>