HAPPENHUB

A MINI PROJECT REPORT

Submitted by

Juluri Sravan Kumar(B210825)

Yashoda Adarsh(B210010)

Guguloth Sammaiah(B211376)

Of

Bachelor of Technology

Under the guidance of

Mr. R Rahul

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
BASAR, NIRMAL (DIST.),
TELANGANA - 504107

HAPPENHUB - Event Aggregator

Project Report submitted to

Rajiv Gandhi University of Knowledge Technologies, Basar

for the partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

by

Juluri Sravan Kumar(B210825)

Yashoda Adarsh(B210010)

Guguloth Sammaiah(B211376)

Under the Guidance of

Mr. R Rahul

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR

OCTOBER 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR

CERTIFICATE

This is to certify that the Project Report entitled **HappenHub** submitted by **Juluri Sravan Kumar(B210825), Yashoda Adarsh(B210010), Guguloth Sammaiah (B211376)**, Department of Computer Science and Engineering, Rajiv Gandhi University Of Knowledge Technologies, Basar; for partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering; is a bonafide record of the work and investigations carried out by him/her/them under my supervision and guidance.

PROJECT SUPERVISOR	HEAD OF DEPARTMENT
Mr. R Rahul	**Dr.B.Venkat Raman**
Assistant Professor	**Assistant Professor**

EXTERNAL EXAMINER

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR

DECLARATION

We hereby declare that the work which is being presented in this project entitled, "**HappenHub**" submitted to **RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, BASAR** in the partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING**, is an authentic record of our own work carried out under the supervision of **"Mr. R Rahul"**, Assistant Professor in Department of Computer Science And Engineering, RGUKT, Basar.
The matter embodied in this project report has not been submitted by us for the award of any other degree.

Place: Basar
Date :

Juluri Sravan Kumar(B210825)
Yashoda Adarsh(B210010)
Guguloth Sammaiah(B211376)

Acknowledgement

We would like to express my sincere gratitude to my advisor, **Mr. R Rahul**, whose knowledge and guidance has motivated me to achieve my goals possible. She has consistently been a source of motivation, encouragement, and inspiration. The time I have spent working under her supervision has truly been a pleasure.

I thank HOD **Dr.B.Venkat Raman** for his effort and guidance and all senior faculty members of the CSE Department for their help during my course. Thanks to programmers and non-teaching staff of the CSE Department.

We would like to express my sincere gratitude to my Project Co-ordinator **Mrs.B. Latha Reddy**, whose knowledge and guidance has motivated me to achieve goals I never thought possible. She has consistently been a source of motivation, encouragement, and inspiration. The time I have spent working under his supervision has truly been a pleasure.

I thank my Vice-Chancellor, **Prof. A. Govardhan**, and Management for providing excellent facilities to carry out my project work.

Finally, special thanks to my parents for their support and encouragement throughout my life and this course. Thanks to all my friends and well-wishers for their constant support.

Juluri Sravan Kumar(B210825)
Yashoda Adarsh(B210010)
Guguloth Sammaiah(B211376)

Abstract

In today's digital era, discovering relevant opportunities like jobs, internships, and hackathons efficiently is a growing challenge due to scattered information across multiple platforms and manual search processes. **HappenHub – Event Discovery Platform** is a comprehensive microservices-based solution designed to aggregate, process, and deliver personalized event information to users through automated scraping, intelligent processing, and recommendation systems.

HappenHub enables seamless discovery and management of opportunities through a multi-service architecture. The platform scrapes data from external sources, processes and standardizes it, provides fast search capabilities, generates personalized recommendations, and allows users to manage wishlists. Real-time notifications and secure authentication ensure transparency and user engagement.

The system is developed using a **microservices architecture** with **Spring Boot (Java)** for most services, **Python with LangChain** for data processing, and **Apache Kafka** for asynchronous communication. The frontend uses **React.js with Redux Toolkit** for state management and **Tailwind CSS** for responsive design. Authentication is secured using **JWT tokens**, while **MySQL** serves as the primary database for transactional data.

HappenHub eliminates manual search processes, improves discoverability, and enhances user experience through intelligent recommendations and fast search. By integrating automated scraping, data processing, and personalized feeds, the system significantly reduces search effort and fosters an efficient event discovery ecosystem. This project demonstrates how modern distributed technologies can effectively transform opportunity discovery into a secure, scalable, and intelligent solution.

Contents

List of Figures

List of Abbreviations and Symbols Used

API	Application Programming Interface
JWT	JSON Web Token
Kafka	Apache Kafka Message Broker
MySQL	MySQL Relational Database
REST	Representational State Transfer
JSON	JavaScript Object Notation
AI	Artificial Intelligence
NLP	Natural Language Processing
ML	Machine Learning
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
SMTP	Simple Mail Transfer Protocol
CRUD	Create, Read, Update, Delete
UI	User Interface
UX	User Experience
DB	Database

Chapter 1

Introduction

1.1 INTRODUCTION

In the modern digital era, discovering relevant opportunities like jobs, internships, and hackathons efficiently has become a key challenge for users. Information is scattered across multiple platforms, search processes are manual and time-consuming, and personalization is often lacking. To overcome these challenges, HappenHub has been developed as an Event Discovery Platform that aggregates, processes, and delivers personalized event information through a microservices-based system.

HappenHub is a distributed web application designed to serve users seeking opportunities. The platform automatically scrapes event data from external sources, processes and standardizes it, provides fast search and filtering capabilities, generates personalized recommendations based on user preferences, and allows wishlist management. This ensures efficient discovery and engagement with relevant opportunities.

1.2 MOTIVATION

Opportunity discovery within educational and professional contexts often faces multiple challenges such as scattered information across platforms, manual search processes, lack of personalization, and difficulty in tracking relevant opportunities. Traditional methods rely on browsing multiple websites, social media, and email notifications, leading to inefficiency, missed opportunities, and information overload.

The need for a smart, centralized, and personalized digital platform is crucial in today's data-driven environment. With increasing numbers of jobs, internships, and hackathons posted across various sources, users require a robust system that can automate data aggregation, intelligent processing, and personalized recommendations.

HappenHub aims to bridge this gap by providing a modern, scalable, and secure event discovery solution using distributed technologies. It leverages microservices architecture for modularity, Kafka for asynchronous processing, and machine learning for recommendations, ensuring a smooth experience for users seeking opportunities.

1.3 PROBLEM DEFINITION

Currently, users seeking opportunities often face the following issues in event discovery:

- **Scattered information:** Event data is distributed across multiple platforms without centralized access.
- **Manual search processes:** Users must manually browse and search multiple websites for relevant opportunities.
- **Lack of personalization:** No intelligent recommendations based on user preferences and history.
- **Data inconsistency:** Different platforms use varying formats for dates, salaries, and descriptions.
- **Limited search capabilities:** Basic text search without advanced filtering or full-text search.
- **No wishlist management:** Users cannot save and track interesting opportunities.

Thus, there is a strong need for a web-based application that can provide an integrated environment for discovering and managing opportunities effectively and securely.

1.4 OBJECTIVE OF THE PROJECT

The primary objective of **HappenHub** is to develop a comprehensive, microservices-based event discovery platform that aggregates, processes, and delivers personalized opportunity information. The key objectives include:

1. To provide a **centralized discovery system** for jobs, internships, and hackathons.
2. To implement **automated data scraping** from external sources using Selenium.
3. To enable **intelligent data processing** and standardization using Python and LangChain.
4. To implement a **fast search service** with full-text search and filtering capabilities.

5. To generate **personalized recommendations** based on user preferences and history.
6. To offer **wishlist management** for saving and tracking interesting opportunities.
7. To maintain a **responsive and visually appealing frontend** using React.js and Redux.
8. To ensure **asynchronous communication** between services using Apache Kafka.

In summary, HappenHub serves as a unified digital platform to enhance opportunity discovery, personalization, and user experience while ensuring scalability, reliability, and intelligent processing.

Chapter 2

Analysis

2.1 USER REQUIREMENT ANALYSIS

HappenHub is designed as an event discovery platform where users can discover, search, and manage opportunities. To ensure efficiency and clarity, it is essential to analyze both the **functional** and **non-functional** requirements of the system.

2.1.1 Functional Requirements

1. User Authentication Requirements

- Users should be able to register and login securely using email and password.
- JWT-based authentication for session management.
- Profile management with preferences for event types (jobs, internships, hackathons).
- Secure password storage and validation.

2. Data Scraping Requirements

- Automated scraping from external sources (Internshala, DevPost) using Selenium.
- Scheduled daily execution at 12 AM.
- Raw data publishing to Kafka topics.
- Manual trigger APIs for on-demand scraping.

3. Data Processing Requirements

- Consume raw data from Kafka and standardize formats.
- Date parsing and conversion to consistent formats.

- Currency conversion (USD to INR) and salary normalization.
- Description shortening and text cleaning.
- Publish processed data to event-data Kafka topic.

4. Search and Discovery Requirements

- Full-text search across event fields.
- Filtering by event type, location, and date.
- Pagination for large result sets.
- Fast indexing and querying capabilities.

5. Recommendation Requirements

- Personalized feed generation based on user preferences.
- Consume user data and event data from Kafka.
- Machine learning models for recommendation generation.
- Paginated API responses.

6. Wishlist Management Requirements

- Add/remove events to/from user wishlist.
- Retrieve user's saved events.
- Email notifications for expiring events.
- Integration with event service for data retrieval.

7. System Requirements (Cross-Functional)

- Asynchronous communication via Apache Kafka.
- MySQL databases for each service (usersdb, eventsdb, etc.).

- RESTful APIs for frontend communication.
- Responsive React.js frontend with Redux state management.
- Docker containerization for deployment.

2.1.2 Non-Functional Requirements

- **Performance:** The system should handle multiple concurrent users efficiently.
- **Security:** Data transmission and authentication should use HTTPS and encrypted tokens.
- **Scalability:** Architecture should support easy integration of new modules (e.g., payment gateway).
- **Usability:** Interface should be intuitive with minimal learning curve.
- **Availability:** The application should maintain 99% uptime for hosted services.
- **Maintainability:** Modular code structure allows easy debugging and enhancements.
- **Compatibility:** Application should work across major browsers and devices.

2.1.3 Hardware and Software Requirements

Hardware Requirements

- Processor: Intel i5 or higher
- RAM: Minimum 16 GB
- Storage: Minimum 1 TB SSD
- Internet connection: Required for hosting and API access

Software Requirements

- Operating System: Windows / Linux / macOS
- Frontend: React.js with Redux Toolkit and Tailwind CSS
- Backend: Spring Boot (Java), Python with LangChain

- Message Broker: Apache Kafka
- Database: MySQL
- Development Tools: IntelliJ IDEA, VS Code, Postman, Git
- Version Control: GitHub
- Containerization: Docker and Docker Compose

Chapter 3

System Analysis

3.1 EXISTING SYSTEM

Currently, users seeking opportunities like jobs, internships, and hackathons face significant challenges in discovering relevant information. The traditional approach relies on manual browsing across multiple platforms, leading to several inefficiencies:

- **Scattered Information Sources:** Opportunity data is distributed across numerous websites without centralized access.
- **Manual Search Processes:** Users must manually visit and search multiple platforms for relevant opportunities.
- **Lack of Personalization:** No intelligent recommendations based on user preferences or history.
- **Data Inconsistency:** Different platforms use varying formats for dates, salaries, and descriptions.
- **Limited Search Capabilities:** Basic keyword search without advanced filtering or full-text search.
- **No Tracking Mechanisms:** Users cannot save or track interesting opportunities.
- **Information Overload:** Users are overwhelmed by irrelevant results and duplicate listings.

Thus, the existing approach is time-consuming, inefficient, and prone to missed opportunities. It lacks automation, intelligence, and user-centric features.

3.2 PROPOSED SYSTEM

The proposed system, **HappenHub**, introduces a comprehensive microservices-based platform to revolutionize opportunity discovery. It automates data aggregation, intelligent processing, and personalized delivery through distributed architecture.

Features of the Proposed System

- **Automated Data Scraping:** Scheduled scraping from external sources using Selenium with manual trigger options.
- **Intelligent Data Processing:** Python-based processing with LangChain for data cleaning, standardization, and enrichment.
- **Fast Search and Filtering:** Full-text search with advanced filtering by type, location, salary, and date.
- **Personalized Recommendations:** ML-based recommendation engine generating customized opportunity feeds.
- **Wishlist Management:** User ability to save, track, and receive notifications for saved opportunities.
- **Asynchronous Architecture:** Kafka-based communication ensuring scalability and fault tolerance.
- **Responsive Frontend:** Modern React.js interface with Redux state management.

Advantages of the Proposed System

- Eliminates manual search processes and reduces discovery time.
- Provides intelligent, personalized recommendations.
- Ensures data consistency through automated processing.
- Offers fast, comprehensive search capabilities.
- Enables wishlist management and tracking.
- Provides scalable, fault-tolerant architecture.
- Delivers modern, responsive user experience.

The proposed system therefore ensures an intelligent, efficient, and user-centric solution for opportunity discovery and management.

3.3 SOFTWARE USED

The following technologies are used in the development of **HappenHub**:

Frontend Technologies

- **React.js with Redux Toolkit:** For building the Single Page Application (SPA) with state management and dynamic routing.
- **Tailwind CSS:** Utility-first CSS framework for responsive design.
- **React Router DOM:** For routing and handling protected routes.
- **Axios:** For handling HTTP requests to backend APIs.

Backend Technologies

- **Spring Boot (Java):** Primary framework for most microservices with robust backend services.
- **Python with LangChain:** Used for data processing, AI/NLP tasks, and complex logic.
- **Apache Kafka:** High-throughput message broker for asynchronous communication between services.
- **JWT Authentication:** Implements secure login and session handling.
- **MySQL:** Relational database for transactional data storage.

Database Technologies

- **MySQL:** Stores all user, event, and transactional data in a relational format.
- **JPA/Hibernate:** ORM library for schema management and database operations.

Development Tools

- **IntelliJ IDEA:** Primary IDE for Java development.
- **Visual Studio Code:** For frontend and Python development.

- **Postman:** Used for testing REST API endpoints.
- **Docker and Docker Compose:** For containerization and local development environment.
- **Git and GitHub:** For version control and team collaboration.

Chapter 4

System Design

System design is the blueprint of the proposed system. It explains the overall structure, interaction between system components, and data movement across modules. In **HappenHub**, system design includes use case modeling, class diagrams, sequence diagrams, and data flow diagrams. These collectively help in understanding the workflow and dependencies between different user roles and modules.

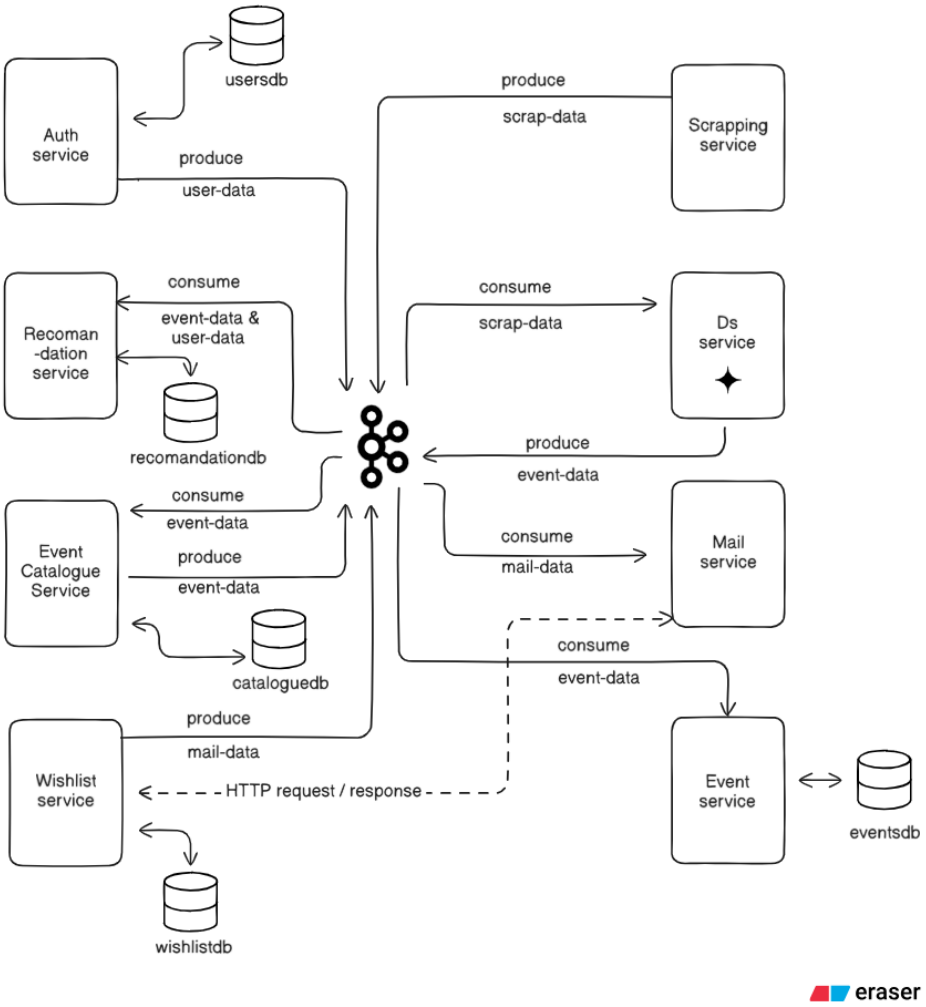

Figure 4.1: Backend System Design

4.1 UML DIAGRAMS

Unified Modeling Language (UML) diagrams are used to describe, visualize, and document the structure and behavior of the system. They represent how different entities interact within the HappenHub architecture.

1. Use Case Diagram

The use case diagram for **HappenHub** illustrates the interaction between different user roles (Admin, Organisation, Volunteer, and Participant) and the system.

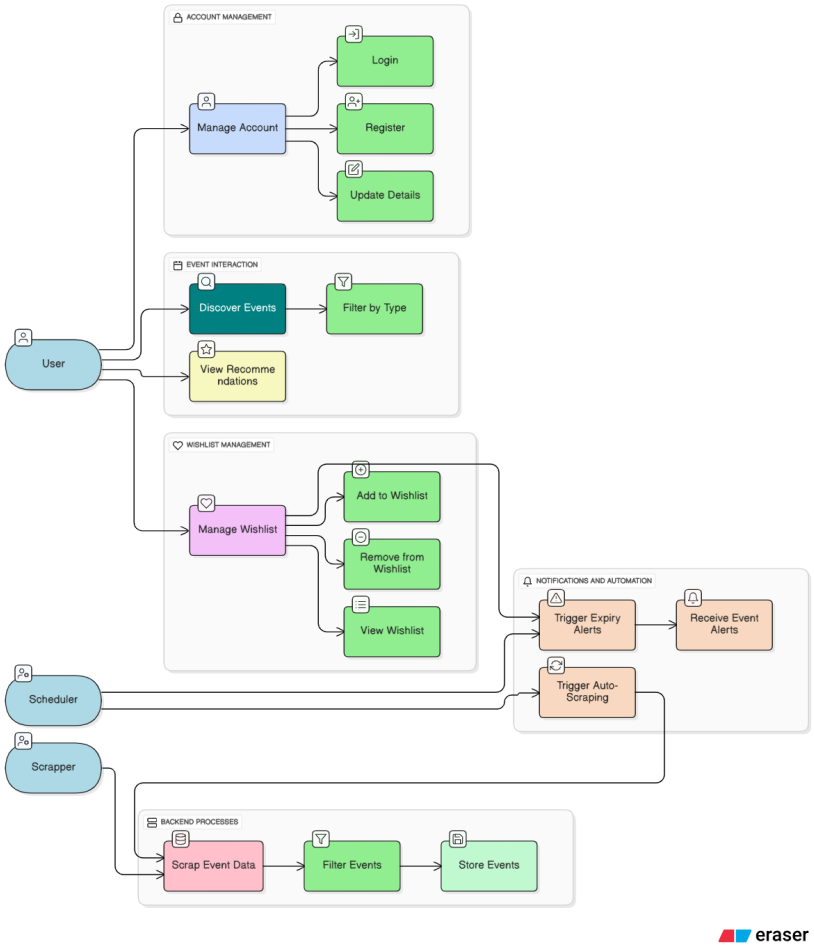

Figure 4.2: Use Case Diagram of HappenHub

2. Class Diagram

The class diagram shows the main entities and their relationships within the system. It helps in database schema design and object-oriented implementation.

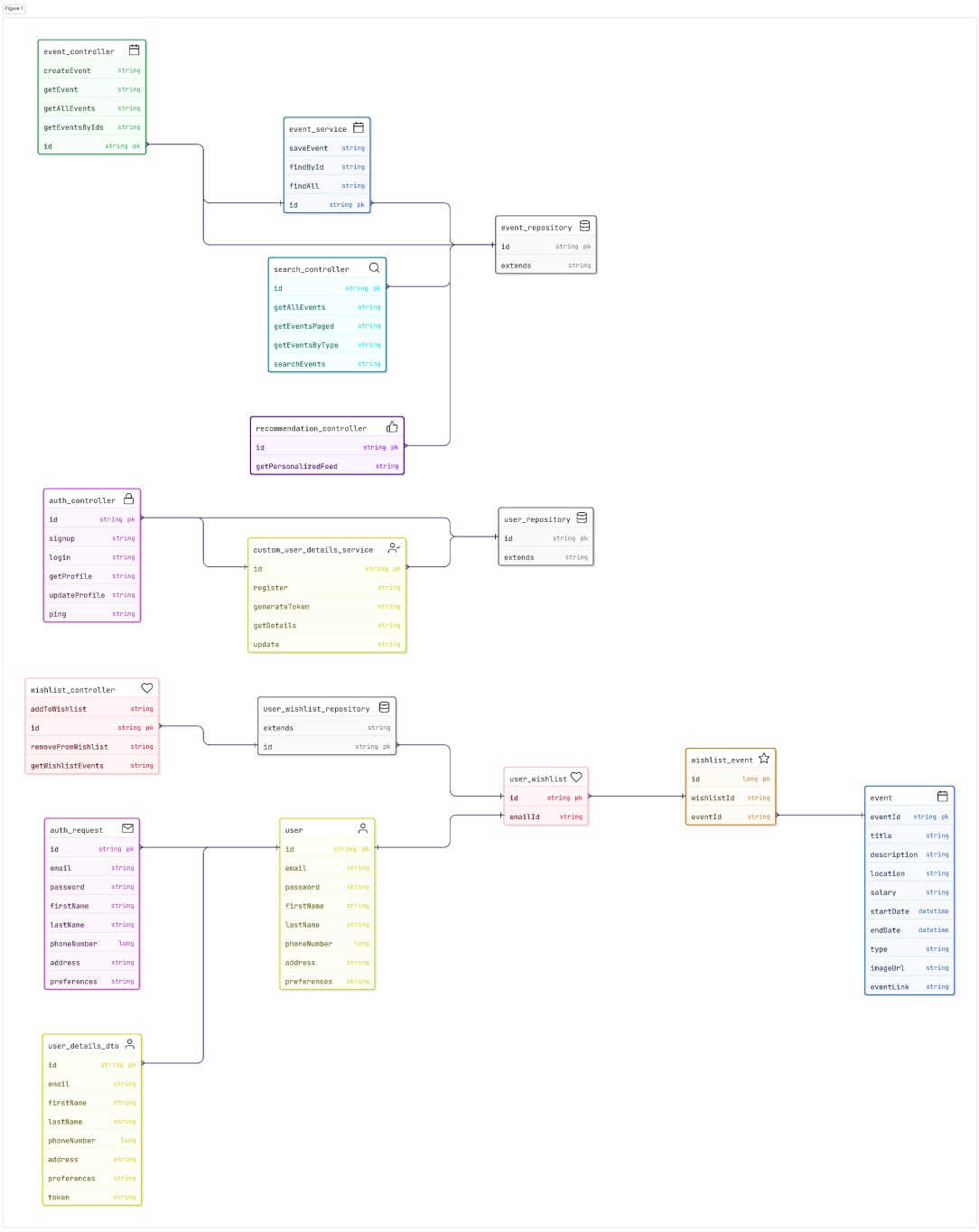

Figure 4.3: Sequence Diagram

3. Sequence Diagram

The sequence diagram demonstrates the order of interactions between system components for a particular operation, such as user registration or event creation.

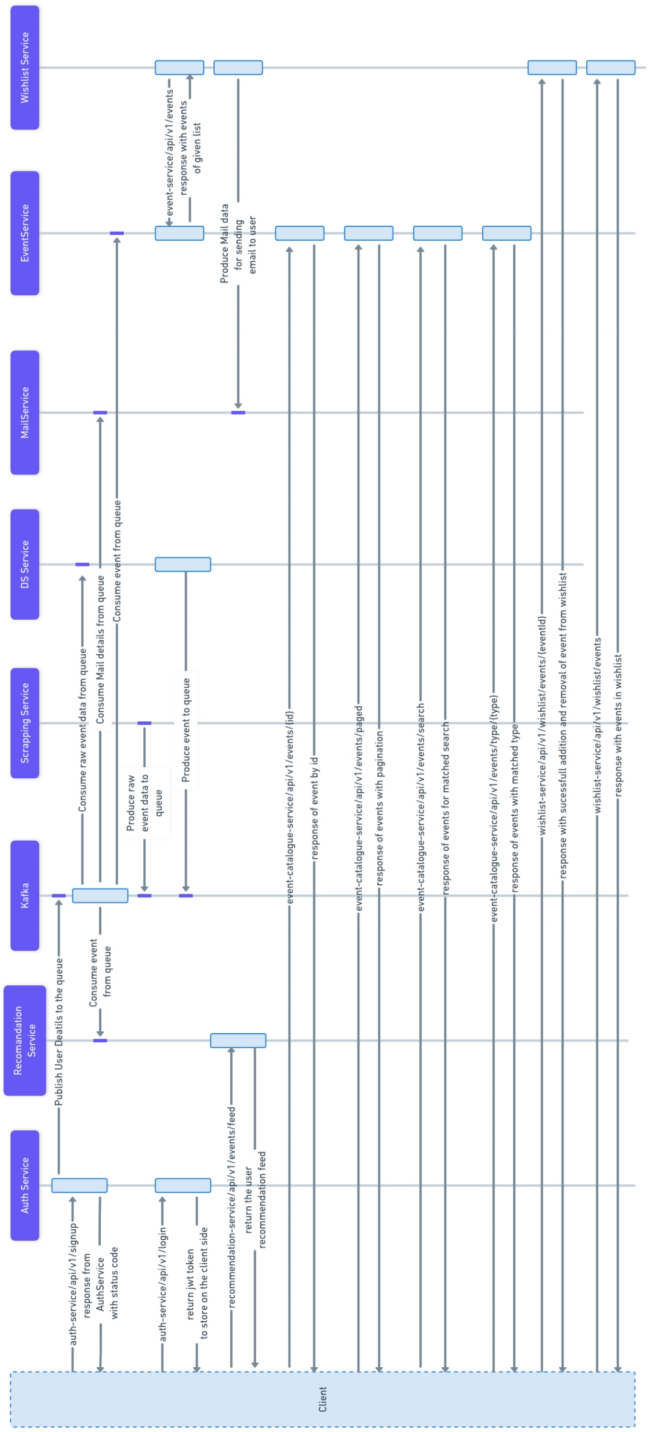

Figure 4.4: Sequence Diagram

4.2 DATA FLOW

The Data Flow Diagram (DFD) shows how information moves through the HappenHub system and how inputs are transformed into outputs through different processes.

1. **Input:** User credentials (registration/login), user preferences, search queries.
2. **Processing:**
    - Authentication and authorization using JWT (Auth Service).
    - Automated data scraping from external sites (Scraping Service).
    - Data cleaning, standardization, and enrichment (DS Service).
    - Asynchronous message passing via Kafka topics.
    - Data indexing and querying (Search Service).
    - Recommendation generation (Recommendation Service).
3. **Output:**
    - Personalized event feeds.
    - Search results with filters.
    - Wishlist management and email notifications.
    - Secure user session (JWT token).

4. Data Storage and Access

- All records are stored in **MySQL Databases**. Each microservice manages its own database (e.g., ‘usersdb‘, ‘eventsdb‘, ‘wishlistdb‘) to ensure loose coupling.
- CRUD operations (Create, Read, Update, Delete) are performed through **Spring Data JPA** and Hibernate repositories within each service.
- Asynchronous data flow between services is handled by **Apache Kafka**, which acts as a durable log for events (e.g., ‘scrap-data‘, ‘event-data‘).
- Each transaction is logged within its service and traceable via correlation IDs for audit purposes.

The system design thus provides a comprehensive view of HappenHub’s workflow and ensures efficient interaction between all functional components.

Chapter 5

Implementation

Implementation refers to the actual construction and deployment of the software system as per the design and specifications. The HappenHub platform is implemented using the MERN (MongoDB, Express.js, React.js, Node.js) stack with secure authentication, modular architecture, and responsive frontend design. This chapter details the structure of the project, its implementation flow, and descriptions of all major modules.

5.1 PROJECT STRUCTURE

The entire system follows a microservices architecture with separate services for different business functions. The frontend communicates with multiple backend services via RESTful APIs.

```
happenhub/
+-- Backend/
|   +-- AuthService/          # User authentication
|   +-- DsService/            # Data processing, AI tasks
|   +-- EventService/         # Core event data management
|   +-- MailService/          # Email notifications
|   +-- RecomandationService/# Personalized recommendations
|   +-- ScrappingService/     # External data scraping
|   +-- SearchService/        # Fast search and indexing
|   +-- WishlistService/      # User favorites management
|   +-- docker-compose.yml    # Service orchestration
|
+-- FrontEnd/
|   +-- src/
|   |   +-- components/       # Reusable UI components
|   |   +-- pages/            # Page components
|   |   +-- redux/            # State management
|   |   +-- api/              # API configurations
|   |   +-- hooks/            # Custom React hooks
```

```
|   |   +-- App.jsx          # Main application component
|   +-- public/
|   +-- package.json
|   +-- tailwind.config.js
|
+-- README.md
```

This microservices structure ensures better scalability, fault isolation, and independent deployment. Each service owns its data and communicates asynchronously via Kafka where appropriate.

5.2 DESCRIPTION OF IMPLEMENTATION

Frontend Implementation

- The frontend is built using **React.js** with **Redux Toolkit** for state management.
- Styling is managed through **Tailwind CSS**, ensuring a modern, responsive interface.
- Navigation between routes (Home, Login, Dashboard, Explore, etc.) is handled using **React Router DOM**.
- Reusable UI components such as event cards, navigation bars, and forms are placed under the `src/components` directory.
- **Axios** is used for all API communications with multiple backend services.
- Animations and smooth transitions are created using **Framer Motion**.
- Alerts and loading effects are implemented using **React Hot Toast**.

Backend Implementation

- The backend follows a **microservices architecture** with 8 independent services.
- Most services are implemented using **Spring Boot (Java)** for robust, scalable backend services.
- **DS Service** uses **Python with LangChain** for AI/NLP tasks and complex data processing.

- **Apache Kafka** serves as the central message broker for asynchronous communication.
- Each service maintains its own **MySQL database** for data isolation and autonomy.
- **JWT authentication** is implemented across services for secure user sessions.
- **Docker** containerization enables easy deployment and scaling.

5.3 MODULE DESCRIPTION

The HappenHub system is divided into multiple microservices, each designed to fulfill a specific business function in the platform's workflow.

5.3.1 Auth Service

Purpose: Manages user authentication, registration, and profile management.

Features:

- User registration and login with JWT authentication.
- Profile management with user preferences.
- Password security and validation.
- User data publishing to Kafka for recommendations.

Implementation Highlights:

- Spring Boot service with JPA for database operations.
- JWT token generation and validation.
- Kafka producer for user-data events.

5.3.2 Scraping Service

Purpose: Extracts opportunity data from external sources.

Features:

- Automated scraping from Internshala and DevPost using Selenium.
- Scheduled daily execution at 12 AM.
- Manual trigger APIs for on-demand scraping.
- Raw data publishing to Kafka scrap-data topic.

Implementation Highlights:

- Spring Boot service with Selenium WebDriver.
- Scheduled tasks using Spring Scheduler.
- Kafka producer for raw event data.

5.3.3 DS Service (Data Science)

Purpose: Processes and standardizes scraped data using AI/NLP.

Features:

- Consumes raw data from Kafka scrap-data topic.
- Date parsing and standardization.
- Currency conversion (USD to INR).
- Description shortening and text cleaning.
- Publishes processed data to event-data topic.

Implementation Highlights:

- Python service with LangChain for NLP tasks.
- Kafka consumer and producer.
- AI-powered text processing and data enrichment.

5.3.4 Event Service

Purpose: Manages core event data storage and CRUD operations.

Features:

- Consumes processed event data from Kafka.
- Stores events in eventsdb MySQL database.
- Provides APIs for event retrieval and management.
- Supports bulk operations and data integrity.

Implementation Highlights:

- Spring Boot service with JPA repositories.
- Kafka consumer for event-data topic.
- RESTful APIs for event operations.

5.3.5 Search Service

Purpose: Provides fast search and filtering capabilities.

Features:

- Consumes event data and maintains search index.
- Full-text search across event fields.
- Filtering by type, location, salary, and date.
- Pagination support for large result sets.

Implementation Highlights:

- Spring Boot with JPA and custom query methods.
- LIKE-based SQL search for flexibility.
- RESTful APIs with pagination.

5.3.6 Recommendation Service

Purpose: Generates personalized opportunity feeds.

Features:

- Consumes event-data and user-data from Kafka.
- Generates personalized recommendations based on preferences.
- Machine learning models for feed generation.
- Paginated API responses for frontend consumption.

Implementation Highlights:

- Spring Boot service with recommendation algorithms.
- Kafka consumers for multiple topics.
- ML-based personalization engine.

5.3.7 Wishlist Service

Purpose: Manages user favorites and notifications.

Features:

- Add/remove events to/from user wishlist.
- Retrieve user's saved opportunities.
- Email notifications for expiring events.
- Daily scheduled checks for reminders.

Implementation Highlights:

- Spring Boot with JPA for wishlist storage.
- Integration with Mail Service via Kafka.
- Scheduled tasks for notification triggers.

5.3.8 Mail Service

Purpose: Handles email notifications and communications.

Features:

- Consumes mail-data from Kafka.
- Sends transactional emails (reminders, updates).
- Template-based email generation.
- SMTP integration for reliable delivery.

Implementation Highlights:

- Spring Boot with JavaMailSender.
- Kafka consumer for mail-data topic.
- Email template processing.

5.4 SCREENSHOTS

The following figures represent various user interfaces and dashboards of the Happen-Hub platform. (Insert real screenshots when available.)

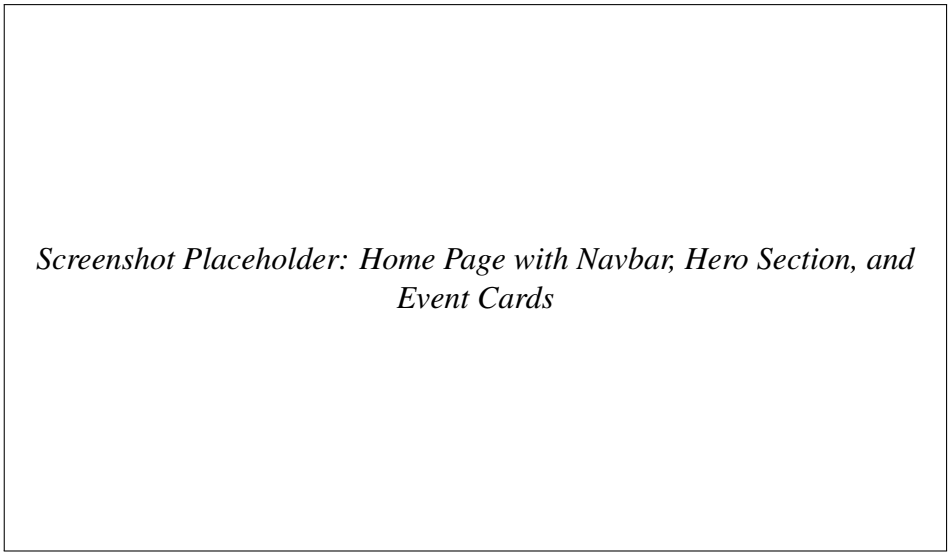

Figure 5.1: Home Page Interface

Screenshot Placeholder: Organisation Dashboard with Event Stats and Notifications

Figure 5.2: Organisation Dashboard Interface

Screenshot Placeholder: Participant Registration and OTP Verification Screen

Figure 5.3: Participant Registration Screen

All modules together ensure a seamless flow of data and functionality between users, events, and the backend, thereby achieving the objective of an intelligent and interactive event management platform.

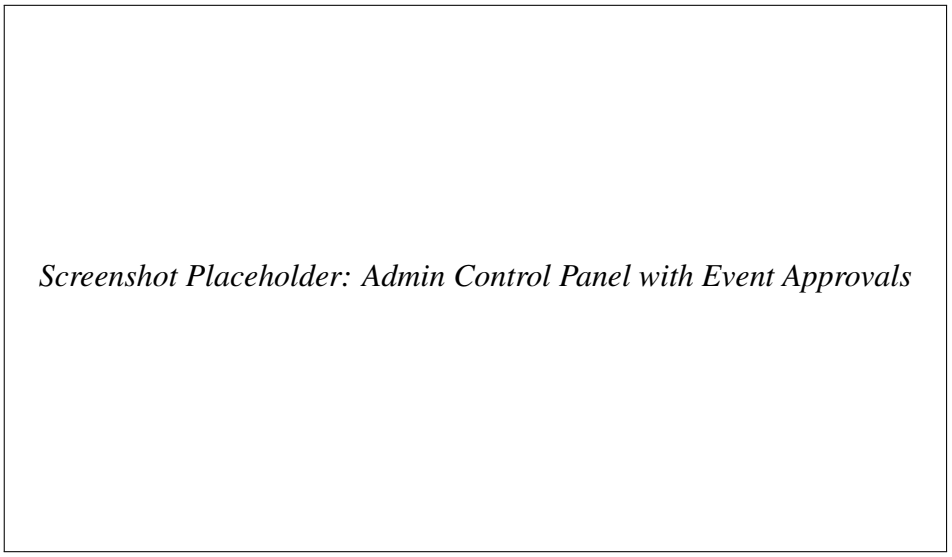

Figure 5.4: Admin Dashboard Interface

Chapter 6

Testing

Software testing is a critical phase in software development that ensures the reliability, accuracy, and stability of the application. For **HappenHub**, testing was performed at multiple levels — system testing, unit testing, security testing, and integration testing — to verify that every microservice works correctly and interacts seamlessly with others.

6.1 SYSTEM TESTING

System testing validates the complete and integrated application to ensure that all functionalities perform as intended. It focuses on verifying both functional and non-functional requirements under realistic conditions.

Objectives of System Testing

- To ensure that the entire HappenHub application meets the specified user requirements.
- To verify that all microservices (Auth, Event, Search, etc.) interact without conflicts.
- To validate end-to-end data flows, such as data scraping, processing, and display.
- To confirm proper data flow between frontend, backend services, **MySQL databases, and Kafka topics**.

System Testing Scenarios

1. **User Registration:** Verify that new users can register successfully via the Auth Service.
2. **Login Authentication:** Test valid and invalid credential combinations and JWT token generation.

3. **Data Pipeline:** Trigger scraping, verify raw data in Kafka, verify processed data in the Event Service database, and verify visibility in the Search Service.
4. **Search and Filtering:** Validate that search results match various filter combinations.
5. **Wishlist Functionality:** Test adding/removing from wishlist and receiving email notifications for expiring events.
6. **Recommendation Feed:** Ensure the recommendation service provides a personalized (not random) feed.

System testing confirmed that the HappenHub platform successfully integrates all microservices with minimal latency, providing accurate responses and reliable behavior under normal and high load conditions.

6.2 UNIT TESTING

Unit testing involves verifying individual modules or components of the system to ensure they perform their specific functions as expected.

Tools Used:

- **JUnit 5 & Mockito (Java):** For testing Spring Boot service layers, controllers, and repositories.
- **PyTest (Python):** For testing the data transformation logic in the DS Service.
- **React Testing Library (Frontend):** For testing React components and user interactions.
- **Embedded Kafka/MySQL:** Used to mock message brokers and databases during automated backend testing.

Sample Unit Test Cases

- **Frontend Component Tests:**
  - Verify that Navbar renders correct links based on authentication status (from Redux store).

  - Ensure EventCard component displays event title, date, and details correctly.
  - Validate form field inputs for email and password in login/registration components.

- **Backend Service Tests:**
  - Test `AuthService` for correct JWT generation and password hashing.
  - Test `DsService` for correct date parsing and currency conversion logic.
  - Test `EventService` Kafka consumer for correct **database insertion**.
  - Test `SearchService` API for correct pagination and filtering logic.

Each component and API endpoint was tested independently to ensure reliability before system integration.

6.3 SECURITY TESTING

Since HappenHub handles sensitive user data and authentication credentials, robust security testing was carried out to detect and prevent vulnerabilities.

Security Aspects Tested

- **Authentication:** Verified that JWT tokens are correctly issued, validated, and expired.
- **Password Protection:** Confirmed that passwords are hashed using **Spring Security (bcrypt)** before storage.
- **Input Validation:** Ensured that all API inputs are sanitized to prevent SQL injection attacks (handled by JPA) and XSS.
- **Session Management:** Tested for proper handling of expired tokens and invalid sessions.
- **Cross-Site Scripting (XSS):** Verified that all user-generated inputs are escaped before rendering on the frontend.
- **Cross-Origin Resource Sharing (CORS):** Configured correctly in Spring Boot to allow only trusted frontend domains.

Results:

- No major vulnerabilities were detected.
- All authentication flows were validated successfully.
- Unauthorized access attempts to service APIs were correctly blocked.

6.4 INTEGRATION TESTING

Integration testing was conducted after successful unit testing to verify interactions between microservices and ensure the system works as a cohesive whole.

Objectives of Integration Testing

- To validate proper communication between the frontend (React) and backend (Spring Boot REST APIs).
- To verify data consistency between API requests and **MySQL responses**.
- To ensure the Kafka-based data pipeline (Scraping → DS → Event) operates correctly.

Integration Test Cases

- Register a new user (Auth Service) → login → access protected route (Event Service).
- Trigger Scraping Service → check Kafka ‘scrap-data‘ topic → check DS Service processing → check Kafka ‘event-data‘ topic → verify data in Event Service DB.
- Add event to wishlist (Wishlist Service) → verify API call to Mail Service (or Kafka topic) for notification.

Integration Testing Results

- Frontend and backend synchronization verified successfully.
- API response times were within acceptable limits.

- Asynchronous data pipeline processed records without data loss or inconsistency.

Testing at every stage confirmed that the HappenHub platform is secure, reliable, and functionally complete. All major workflows were validated successfully, ensuring readiness for deployment.

Chapter 7

Conclusion and Future Scope

7.1 CONCLUSION

The **HappenHub – Event Discovery Platform** successfully demonstrates the development of a scalable, intelligent, and distributed system that revolutionizes opportunity discovery through automated data aggregation and personalized recommendations. By implementing a microservices architecture with asynchronous communication, HappenHub bridges the gap between scattered information sources and intelligent user-centric discovery.

The platform enables:

- Automated data scraping from external sources with intelligent processing.
- Fast, comprehensive search and filtering capabilities.
- Personalized recommendation feeds based on user preferences.
- Wishlist management for tracking interesting opportunities.
- Asynchronous, fault-tolerant communication between services.

The use of modern distributed technologies such as **Spring Boot, Python with LangChain, Apache Kafka, MySQL, React.js with Redux, and Docker** ensures high performance, scalability, and maintainability. Integration of **microservices architecture** for modularity, **Kafka** for asynchronous processing, and **AI-powered data processing** make the system robust and intelligent.

Comprehensive testing—including unit, system, integration, and security testing—validated the reliability and efficiency of each microservice. As a result, HappenHub achieves its goal of providing an intelligent event discovery solution that significantly reduces search effort and enhances user experience.

7.2 FUTURE SCOPE

While the current version of HappenHub fulfills the essential requirements of an event discovery platform, there are several opportunities for future enhancements and scalability:

- **Mobile Application:** Develop a cross-platform mobile app using React Native to provide real-time access to opportunity updates and notifications.
- **Advanced ML Models:** Implement more sophisticated recommendation algorithms using collaborative filtering and deep learning.
- **Real-time Notifications:** Add WebSocket support for instant notifications about new opportunities matching user preferences.
- **Data Analytics Dashboard:** Create comprehensive analytics for user behavior, popular opportunities, and platform performance.
- **Multi-Source Integration:** Expand scraping capabilities to include more platforms and APIs for broader opportunity coverage.
- **Blockchain Integration:** Implement blockchain for secure, verifiable credentials and application tracking.
- **Multi-Language Support:** Add localization and NLP processing for multiple languages.
- **API Marketplace:** Create a marketplace for third-party integrations and custom recommendation models.

The proposed enhancements would not only extend the system's capabilities but also make HappenHub a comprehensive platform

References

[1] Spring Boot Documentation, *Spring Boot Reference Documentation*, Available at: https://spring.io/projects/spring-boot

[2] ReactJS Documentation, *React – A JavaScript library for building user interfaces*, Available at: https://react.dev/

[3] Apache Kafka Documentation, *Apache Kafka – A distributed event streaming platform*, Available at: https://kafka.apache.org/

[4] Python Documentation, *Python 3 Documentation*, Available at: https://docs.python.org/3/

[5] MySQL Documentation, *MySQL Reference Manual*, Available at: https://dev.mysql.com/doc/

[6] LangChain Documentation, *LangChain – Develop applications with LLMs*, Available at: https://www.langchain.com/

[7] Tailwind CSS Documentation, *A utility-first CSS framework for rapid UI development*, Available at: https://tailwindcss.com/

[8] JWT (JSON Web Token) Documentation, *Open standard for secure token-based authentication*, Available at: https://jwt.io/

[9] Docker Documentation, *Docker Overview*, Available at: https://docs.docker.com/

[10] Fowler, M. (2014). *Microservices*. Available at: https://martinfowler.com/articles/microservices.html

[11] Pressman, R. S., (2019). *Software Engineering: A Practitioner's Approach*, 8th Edition, McGraw Hill Education.