EventHub

Event Management System

Final Report

Version 1.0

Date: 31/05/2024

Name:

Ravini Kuruppu

Abstract/ Executive summary

This report addresses the inefficiencies in managing and communicating about events within organizations. By developing a centralized Event Management System, EventHub improves event coordination, enhances member engagement, and ensures timely transmission of event information. This report provides a comprehensive overview of the project, including its goals, methodologies, key outcomes, and future implications.

Table of Content

1. Introduction

    1.1. background of the application domain/ problem 4

    1.2. motivation for the selected system development 4

    1.3. importance and main purpose of the system 4

    1.4. overview/ summary of the system and used approach and outcome 4

2. Literature Review 5

3. System Models

    3.1. System Requirement 5

    3.2. System Design 8

    3.3. Database Design 11

4. System Implementation

    4.1. Implementation Procedure 12

    4.2. Materials 13

    4.3. Main Interfaces 14

5. System Testing and Analysis

    5.1. Testing approach 16

6. Challenges and Problems 18

7. Conclusion and Future Work 18

Reference 19

1. Introduction

1.1 background of the application domain/ problem

In today's fast-paced environment, organizations such as universities, large and medium-scale companies, clubs, and societies face the challenge of effectively managing and communicating about numerous events annually. Current practices often rely on fragmented communication channels like emails, WhatsApp, Microsoft Teams, and social media platforms. This approach is inefficient and increases the risk of important events being overlooked. To address the issue, the proposal is for the development of a dedicated Event Management System (EMS) aimed at enhancing event management and communication, ensuring that all members are well-informed about upcoming events.

1.2 motivation for the selected system development

The motivation behind the development of this Event Management System is to improve event management processes, enhance member engagement, and mitigate the challenges associated with traditional event communication methods. The existing methods of event notification through disparate channels such as emails, social media, and messaging apps are inefficient and prone to information loss. This leads to missed opportunities for members to attend valuable events, redundant communication efforts by event organizers, and lack of a centralized platform for event information and management. Additionally, there is a significant market opportunity for EMS to be marketed to educational institutions, large and medium-scale companies, and clubs and societies such as the Lions Club. By targeting this vast audience, the product can achieve high market value and broad adoption, addressing the needs of diverse organizations and enhancing their event management capabilities.

1.3 importance and main purpose of the system

The primary purpose of EventHub lies in its role as a centralized platform facilitating efficient management and communication of events within organizations. By offering a unified space for event coordination and transmission, EventHub ensures that members remain informed and engaged with upcoming activities. This approach not only enhances organizational efficiency but also increases a sense of community and participation among members, ultimately contributing to the overall success of events and initiatives

1.4 overview/ summary of the system and used approach and outcome

EventHub represents a comprehensive solution to the challenges of event management and communication within organizations. Employing a user-centric approach, EventHub offers features such as event listing, favoriting, and sorting, accessible through intuitive interfaces on both web and mobile platforms. Leveraging modern technologies like Flutter and Node.js, the system ensures seamless performance and scalability. The outcome is a versatile platform that enhances organizational efficiency and member engagement, revolutionizing the way events are managed and communicated.

2. Literature Review

Event management systems (EMS) are essential tools that facilitate the organization, scheduling, and communication of events within various types of organizations. The theoretical foundation of EMS emphasizes enhancing efficiency through automation and centralized information management. Key aspects include user engagement, real-time notifications, and data analytics to improve event planning and execution.

Several existing systems have paved the way for modern EMS solutions. *Eventbrite*, a popular platform, offers comprehensive event management tools including ticketing, promotion, and attendee tracking (Eventbrite, 2020). However, it is primarily designed for public events and lacks tailored features for internal organizational use. Similarly, *Meetup* focuses on community events and provides social networking capabilities, but its functionality is often limited by a subscription model (Meetup, 2021).

EventHub distinguishes itself by catering specifically to the needs of educational institutions and large organizations, providing a unified solution for internal event management. Unlike Eventbrite and Meetup, EventHub focuses on private events, offering features such as keyword-based event sorting, favoriting, and real-time notifications. Moreover, its integration with both web and mobile platforms using Flutter ensures accessibility and user engagement across devices. The use of PostgreSQL for database management and Amazon S3 for file storage.

3. System Models

3.1 System Requirement

Functional Requirements

Event management system have several core functionalities aimed at simplify event management and enhancing user engagement. The system includes robust user management features, allowing users to register, create profiles, and manage roles. Admins can effortlessly create, edit, and delete events, while users can view, sort, and filter upcoming events by keywords or types. Users can mark events as favorites, register for events directly, and receive notifications via email or alerts. Additionally, the system offers search and filter options to find events quickly and allows users to provide feedback and rate events, enabling admins to refine future events based on this input.

Non-Functional Requirements

EventHub is designed with several non-functional requirements to ensure its reliability, performance, and usability. The system can handle many concurrent users, with event lists and details loading within few seconds. It is scalable, designed to accommodate increasing numbers of users and events without compromising performance. Security is prioritized with strong encryption for all user data, robust authentication, and authorization mechanisms. Usability is enhanced with an intuitive interface, allowing users to perform key tasks within three clicks. It is compatible with major web browsers and mobile operating systems, ensuring an uninterrupted user experience. This combination ensures that EventHub is a robust, efficient, and user-centric event management solution.

Use case diagram

This diagram and description highlight the primary interactions and functionalities within the EventHub system, providing a clear overview of user roles and their actions.

In this use case diagram:

- **System Admin:** Handles user and role management.
- **Admin:** Manages event creation, editing, and deletion, Handle User Registration for events
- **User:** Views and interacts with events, including marking favorites, registering, and receiving notifications.

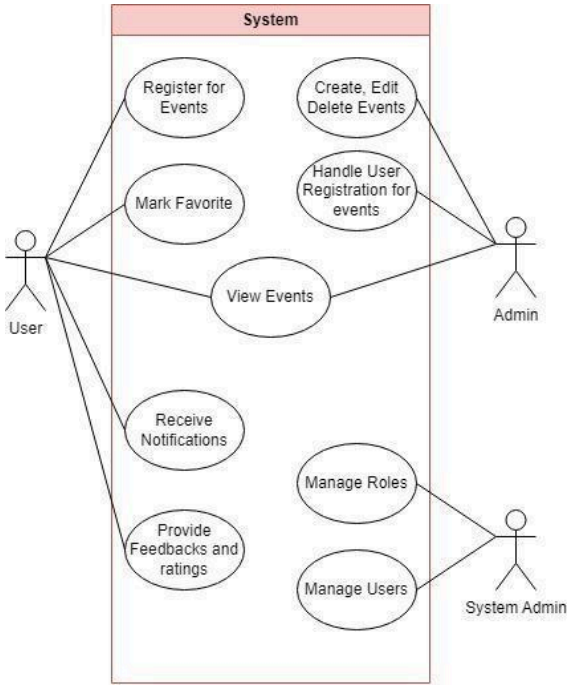

Fig 1-Use case Diagram

3.2 System Design

The system architecture of EventHub is designed to ensure scalability, reliability, and performance. It follows a three-tier architecture comprising the presentation layer, application layer, and data layer.

1. **Presentation Layer:**
    - **Frontend:** Built using Flutter, this layer includes the web and mobile interfaces through which users interact with the system. It communicates with the backend via RESTful APIs.
    - **Components:** User interface elements, state management (using Provider or Riverpod), and navigation.
2. **Application Layer:**
    - **Backend:** Developed using Node.js with Express.js, this layer handles the business logic and processes requests from the presentation layer.
    - **Components:** API endpoints, authentication (JWT), event management logic, and notification services.
3. **Data Layer:**
    - **Database:** PostgreSQL stores all the data related to users, events, and interactions.
    - **File Storage:** Amazon S3 for storing event images and other media files.
    - **Components:** Database schemas, data access objects (DAOs), and cloud storage configurations.

The class diagram provides a logical view of the system, detailing the key classes, attributes, and associations within EventHub.

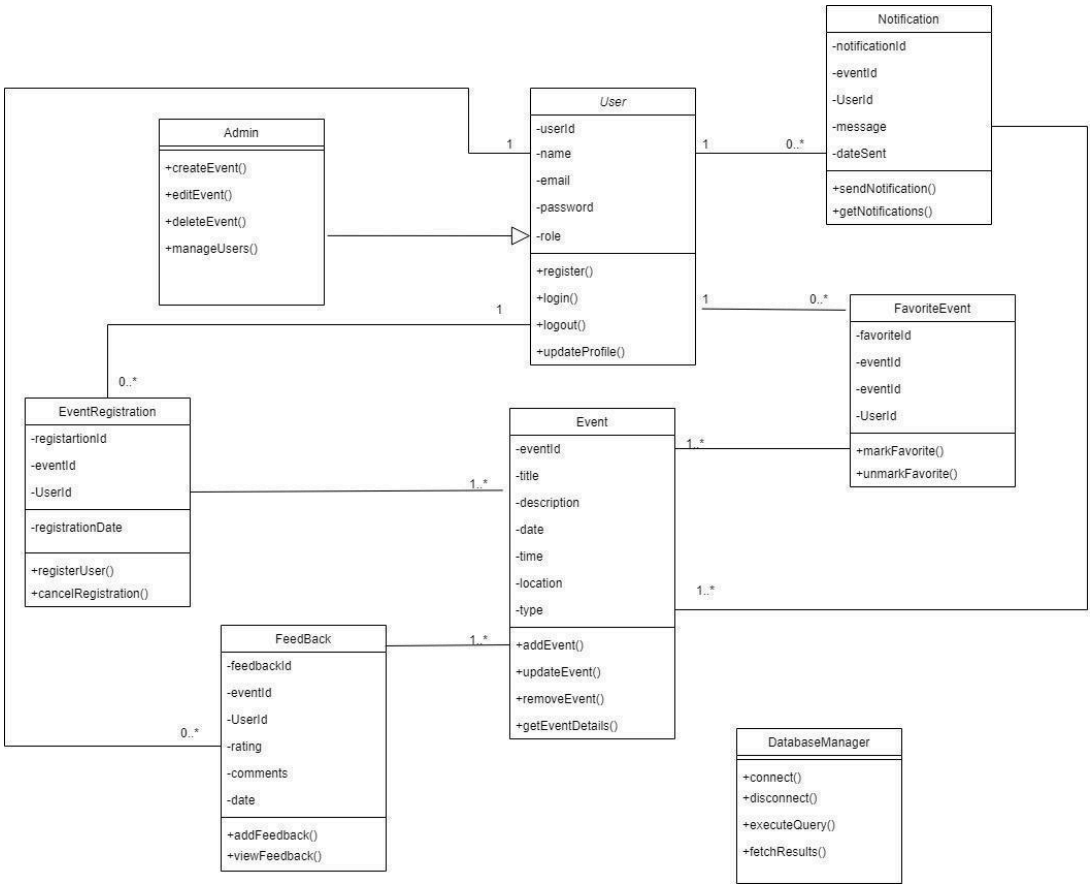

Fig 2 - Class diagram

3.3 **Database Design**

The database design for EventHub is crucial for efficiently storing and retrieving information related to users, events, registrations, and notifications. Below is an overview of the database schema and an Entity-Relationship (ER) diagram to illustrate the storage structure.

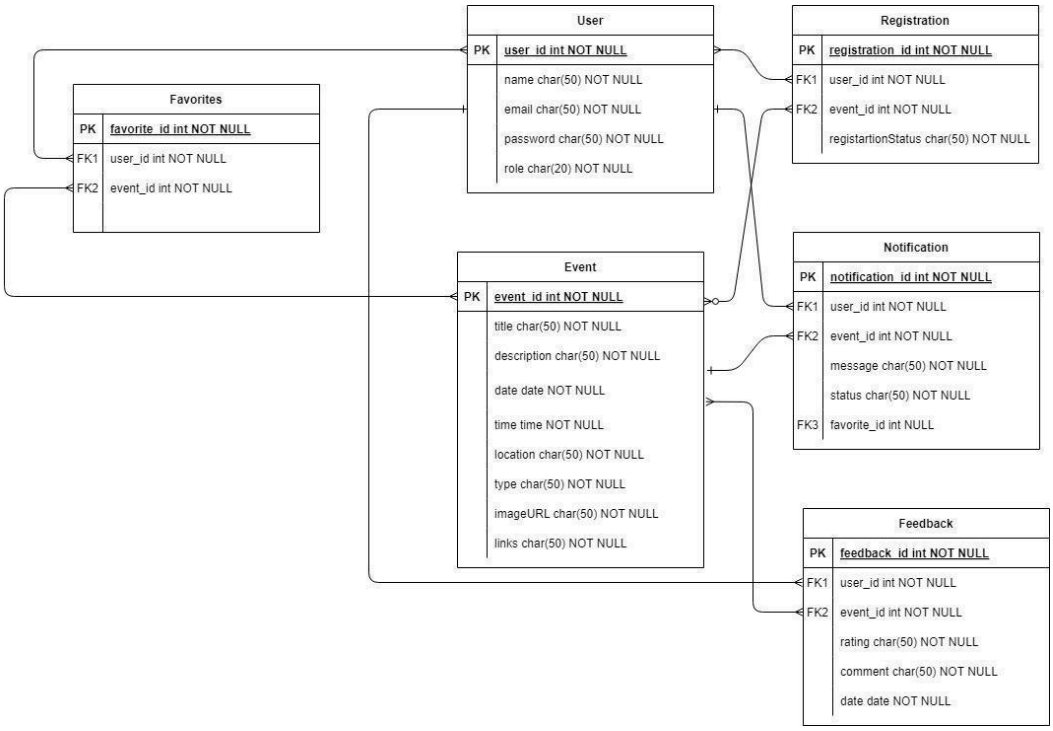

Fig 3 – ER diagram for the database

The database schema is designed to efficiently manage the relationships and data integrity within EventHub. The use of foreign keys ensures referential integrity between related tables, such as users, events, and registrations. Additionally, storing images in Amazon S3 and referencing them via URLs in the database helps to optimize storage and retrieval performance.

4. System Implementation

4.1 Implementation Procedure

After finalizing the software design specification for EventHub, all the business logics and architectures were finalized, identifying key services to be implemented. The system is divided into two main parts: a mobile application and a web application. Different technologies were employed to meet the requirements efficiently.

For the frontend, EventHub used Flutter, a versatile framework, to develop both web and mobile applications. Flutter's component-based approach enabled the development team to build modular, interactive, and responsive elements, ensuring a smooth user experience across platforms. Dart was used as the core programming language, supported by state management tools like Provider or Riverpod to maintain a consistent application state. Styling was handled using Flutter's built-in widget system, ensuring that EventHub was not only functional but also visually appealing.

EventHub adopted a three-tier architecture consisting of the presentation layer, business logic layer, and data layer. For the backend, Node.js and Express.js played key roles. Node.js provided a runtime environment for server-side JavaScript, while Express.js served as a framework for building APIs. These technologies allowed for efficient request handling, routing, and server-side logic, ensuring the smooth functioning of the application.

Document and media storage in EventHub was facilitated using Amazon S3, a cloud-based solution for secure and scalable file storage. This ensured that event images and other media files were efficiently stored and easily retrievable.
The mobile and web applications of EventHub were developed using Flutter, which allowed for a single codebase to be used across multiple platforms. Flutter also handled the notification service, using Firebase Cloud Messaging (FCM) to manage and deliver notifications to users, enhancing communication and engagement.
In the testing phase, a combination of testing tools and frameworks was used. For the frontend, Flutter's built-in testing tools were employed for unit, widget, and integration testing. For the backend, also unit and integration testing was done.
Deployment for EventHub involved the use of GitHub Actions for continuous integration and continuous deployment (CI/CD), ensuring seamless updates and maintenance. The web application was deployed using Firebase Hosting, while the mobile applications were published on the Google Play Store. The backend was deployed on AWS for scalability and reliability.
This implementation procedure, utilizing modern technologies and efficient tools, ensured that EventHub was a robust, scalable, and user-friendly event management system.

4.2 **Materials**

In the implementation of EventHub, several existing materials were utilized to ensure the system's functionality and effectiveness. Initial event data, including titles, descriptions, dates, times, locations, and types, were sourced from past university or organizational event records, providing a comprehensive dataset for populating the application and testing its functionalities. Event images and media files were gathered from previous event marketing materials to ensure a visually appealing user experience. User information, such as names, emails, and roles, was compiled from existing directories and databases within the organization to create user profiles and simulate various user interactions during testing. Historical user feedback and ratings from past events were also collected to inform the feedback and rating system design. Email templates and notification messages used in prior event communications were adapted for the system's notification functionalities, ensuring consistency in communication. Technical documentation on best practices for Flutter, Node.js, and PostgreSQL was referenced to ensure the system was built following industry standards, while existing API documentation from integrated services like Amazon S3 and Firebase Cloud Messaging was used for effective implementation and integration.

4.3 The Algorithm

Notification Scheduling Algorithm:

- **Purpose:** To schedule and send notifications to users about upcoming events they are interested in or registered for.
- **Pseudocode:**

```
function scheduleNotifications(user, events):
    for event in events:
        if event.date - currentDate <= notificationThreshold:
            sendNotification(user, event)

function sendNotification(user, event):
    message = "Reminder: Upcoming event '" + event.title + "' on " + event.date
    NotificationService.send(user.email, message)
```

Event Sorting and Filtering Algorithm:

- **Purpose:** To enable users to sort and filter events based on various criteria such as date, type, and popularity.
- **Pseudocode:**

```
function sortAndFilterEvents(events, criteria):
    if criteria.type:
        filteredEvents = filter(events, event => event.type == criteria.type)
    else:
        filteredEvents = events

    if criteria.date:
        filteredEvents = filter(filteredEvents, event => event.date == criteria.date)

    if criteria.popularity:
        sortedEvents = sort(filteredEvents, (a, b) => b.popularity - a.popularity)
    else:
        sortedEvents = filteredEvents

    return sortedEvents
```

4.4 **Main Interfaces**

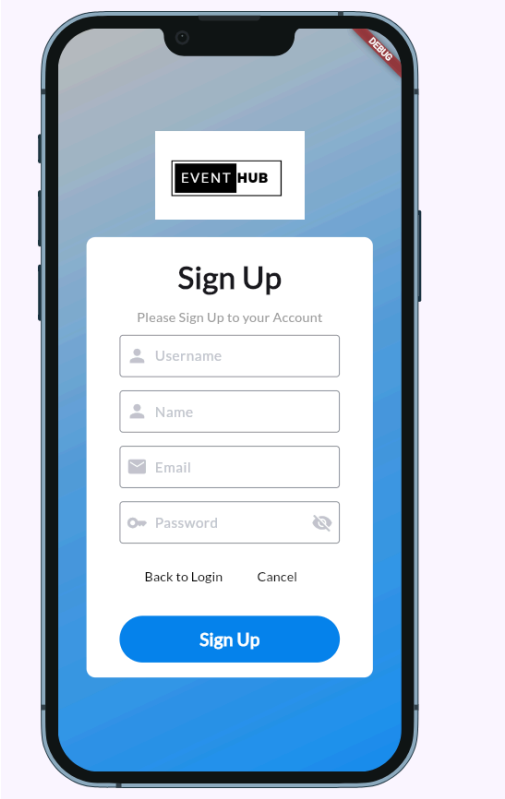

Fig 4 -Signup Screen

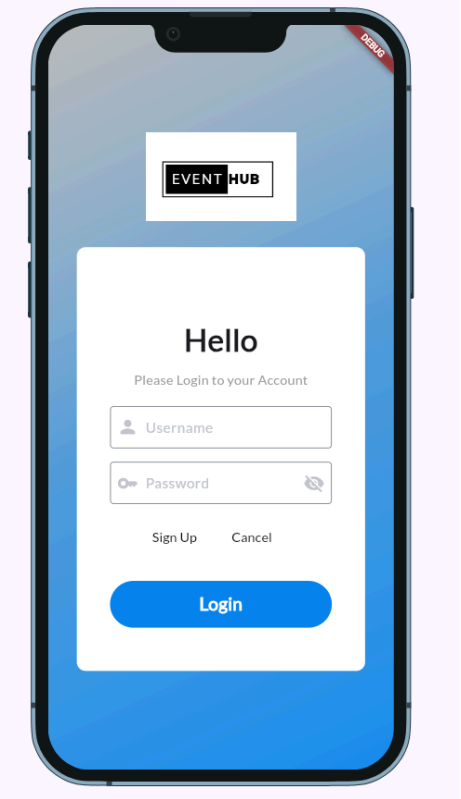

Fig 5- login screen

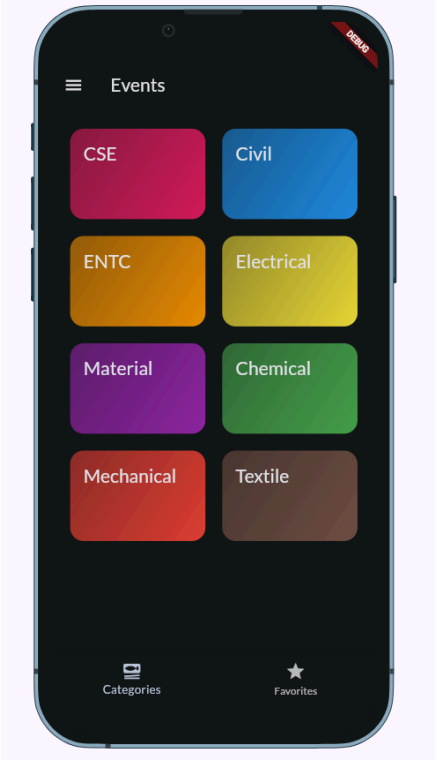

Fig 6 -Categories Screen

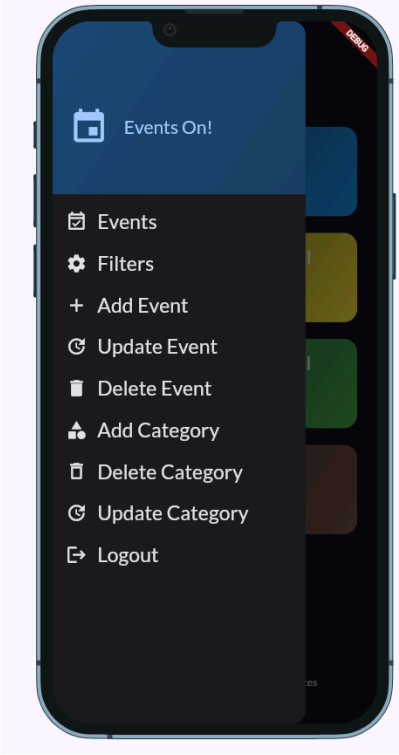

Fig 7 -Functions

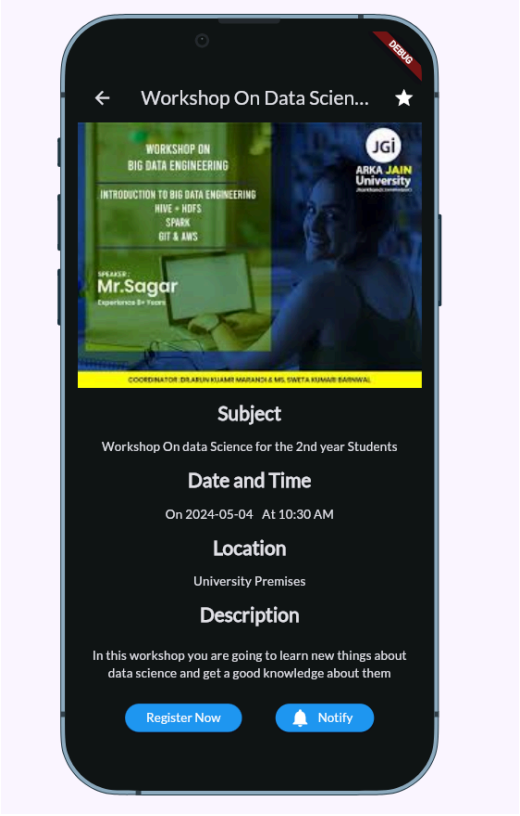

Fig 8 -Event Screen

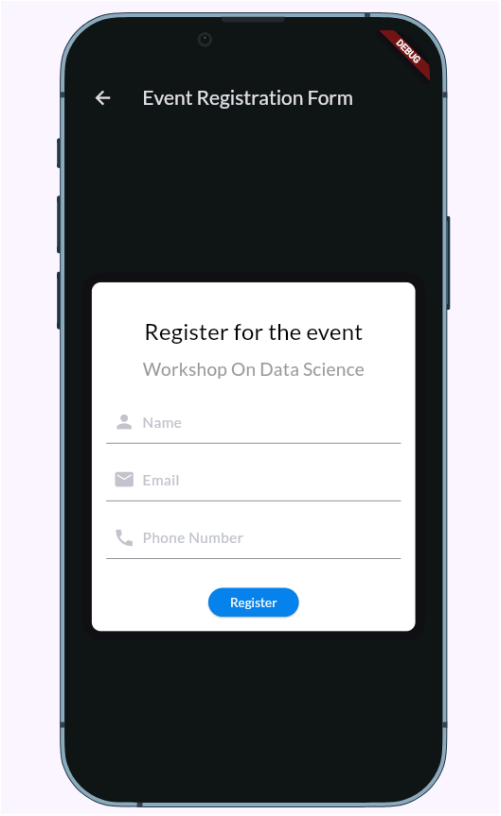

Fig 9 -Event Registration Screen

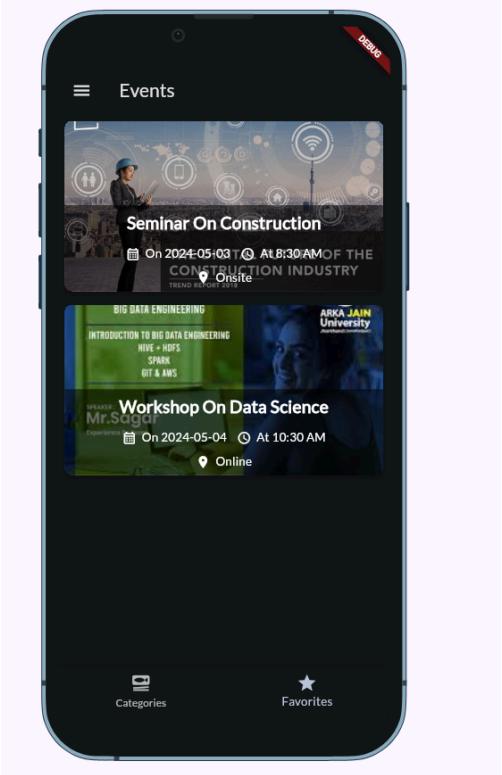

Fig 10 -List Of favorite Events

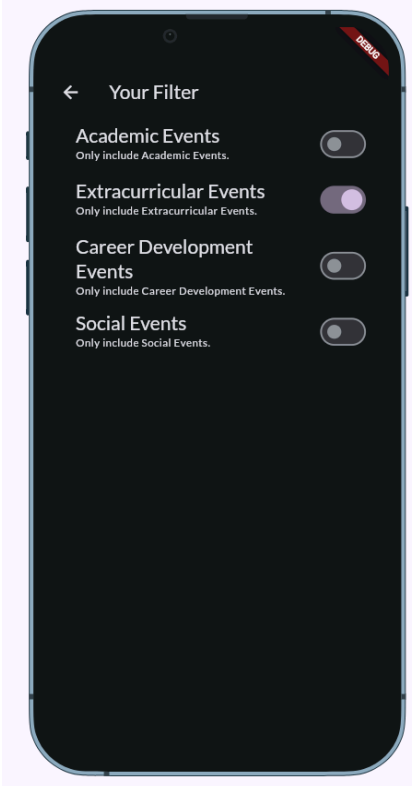

Fig 11 -Filter Screen

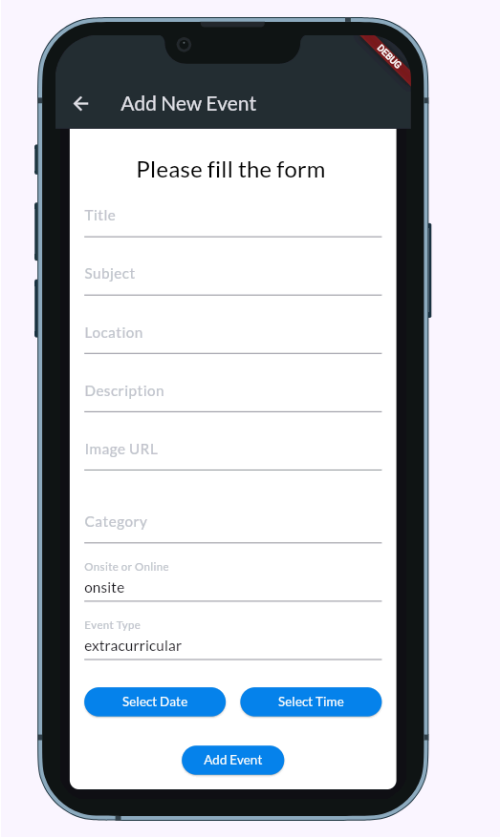

Fig 12 -Event Registration Screen

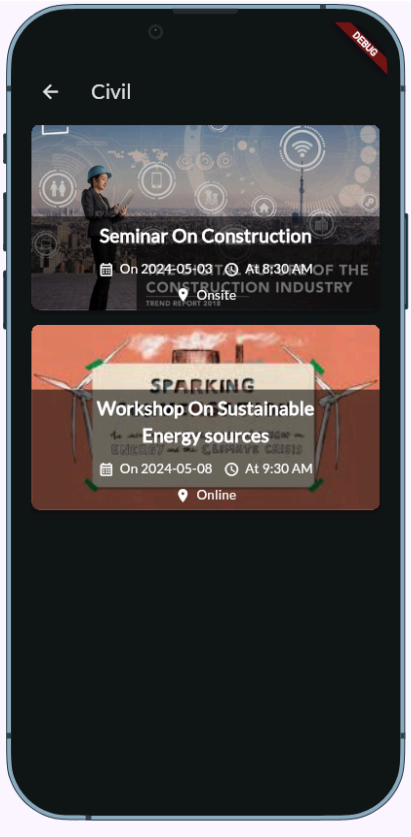

Fig 13 -Event list in each category

5. System Testing and Analysis

5.1 Testing approach

In EventHub, a comprehensive approach to system testing was adopted to ensure its reliability, performance, and security. Various testing techniques, including unit testing, integration testing, and end-to-end testing, were employed to verify the functionality of both frontend and backend components. Unit testing was conducted using Flutter's built-in testing tools for the frontend and Mocha and Chai for the backend, ensuring that individual units of code behaved as expected. Integration testing focused on testing the interaction between different modules and services, while end-to-end testing validated the system's functionality from the user's perspective. Performance and security considerations were also addressed, with thorough testing conducted to identify and mitigate potential performance bottlenecks and security vulnerabilities. Additionally, aspects related to performance, such as load testing and stress testing, were evaluated to ensure the system could handle the expected workload efficiently. Security testing involved assessing the system for vulnerabilities, implementing encryption mechanisms, and enforcing secure authentication and authorization protocols to safeguard user data. Overall, the testing and analysis phase played a crucial role in validating the functionality, reliability, and security of EventHub, ensuring a seamless user experience and mitigating potential risks.

6. Challenges and problems

Developing EventHub involved overcoming several challenges to ensure a robust and user-friendly system. Integrating with existing organizational systems posed complexities due to differing data formats, which were addressed by developing custom APIs. Scalability was a significant concern, handled by utilizing cloud services and horizontal scaling strategies. Ensuring data security and privacy required implementing strong encryption and robust authentication mechanisms. Designing an intuitive user interface involved extensive usability testing and feedback incorporation. Achieving cross-platform compatibility was facilitated by using Flutter, which allowed for consistent performance across web and mobile platforms. Implementing real-time notifications through Firebase Cloud Messaging ensured users remained informed about events. Efficient database management was achieved using PostgreSQL, with careful schema design for complex queries. Performance testing with tools like JMeter helped simulate real-world conditions, identifying and resolving potential bottlenecks. Lastly, maintaining and updating the system seamlessly was ensured by adopting continuous integration and deployment practices, allowing for smooth updates without user disruption.

6. Conclusion and Future Work

In conclusion, the development of EventHub has addressed the inefficiencies in traditional event management systems by providing a user-friendly and efficient platform for managing and accessing event information. The system's implementation has demonstrated its effectiveness in simplifying event organization processes, improving user engagement, and enhancing communication within organizations. By leveraging modern technologies and best practices, EventHub offers a robust and scalable solution for organizations of varying sizes and domains.

Moving forward, several avenues for future enhancements and research directions are identified. Firstly, incorporating advanced recommendation algorithms based on user preferences and behavior patterns could further personalize the event discovery experience, increasing user engagement and satisfaction. Additionally, integrating machine learning techniques for predictive analytics could enable organizations to anticipate user preferences and optimize event planning strategies accordingly. Furthermore, expanding the system's capabilities to support virtual and hybrid events, along with enhanced multimedia content management features, would cater to the evolving needs of remote and distributed work environments.

Moreover, enhancing accessibility features to accommodate diverse user needs, such as support for assistive technologies and multilingual interfaces, would promote inclusivity and accessibility. Continuous refinement of security measures and compliance with data protection regulations will remain crucial to safeguarding user privacy and data integrity. Additionally, conducting user feedback sessions and usability studies to gather insights for further refinement and iteration of the user interface and overall user experience will be imperative.

Overall, the development of EventHub marks a significant step towards modernizing event management practices, and ongoing efforts to innovate and refine the system will ensure its continued relevance and effectiveness in facilitating seamless event organization and engagement.

Reference

[1] Eventbrite. (2020). Eventbrite: Create and Host Successful Events. [online] Available at: https://www.eventbrite.com [Accessed on 31 May. 2024]

[2] Meetup. (2021). Meetup: Find and Create Local Events. [online] Available at: https://www.meetup.com [Accessed on 31 May. 2024]

[3] flutter.dev (2023) A community open to everyone. [online] Available at: https://flutter.dev/community[Accessed on 31 Oct. 2023]

[4] nodejs.org (2023) About Node.js. [online] Available at: https://nodejs.org/en/about [Accessed on 31 Oct. 2023]

[5] dev.mysql.com (2023) MySQL Documentation. [online] Available at: https://dev.mysql.com/doc [Accessed on 31 Oct. 2023]

[6] jwt.io (2023) What is JSON Web Token? [online] Available at: https://jwt.io/introduction [Accessed on 31 Oct. 2023]

[7] learning.postman.com (2023) Overview. [online] Available at: https://learning.postman.com/docs/introduction/overview[Accessed on 31 Oct. 2023]