



# Project Review Report

On Event Syrup, an Event Management Web Application.

Name: Anup Chapagain

Module Name: COM668 Computing Project

Module Tutor: Muhammad Hassan

Project Supervisor: Ekereuke Udoh

Date of Submission: 23/08/2024

# Abstract

This project report presents the design and development of an event management system web application, aimed at streamlining the organization and management of events through an efficient and user-friendly platform.

Built using ReactJS (ReactJs, 2024) for the frontend, Google Firebase's Firestore (Google, 2024) for real-time database management, and Flask (Python, 2024) for backend API integration, the system also integrates IoT functionality via Raspberry Pi (RaspberryPi, 2024) to improve automation and monitoring of the events.

The main goal of the project is to develop a scalable web application that allows users to manage event-related activities such as registration, schedules and notifications, while integrating other IoT devices.

IoT devices to automate tasks such as environmental monitoring and device control. The ReactJS framework was chosen because of its component-based architecture, which allows for flexible and responsive user interfaces. Google Firestore was used for real-time data synchronization features, which are important for managing live event data. The Flask framework, combined with Python, was used to create robust API endpoints that support seamless communication between the user interface and IoT devices.

Throughout the development process, the project followed an Agile methodology (Agile Alliance, 2024), allowing for iterative improvements and continuous integration of feedback. The system was rigorously tested to ensure reliability, security, and performance, with special attention paid to the interaction between the web application and IoT components.

The final product successfully met the initial requirements and demonstrates the potential of combining modern web technologies with IoT to create innovative solutions in event management.

This report concludes with a reflection on lessons learned, challenges encountered, and recommendations for future improvements, including potential scalability improvements and additional IoT integration.

# Acknowledgements

I would like to express my deepest gratitude to my module tutor, Muhammad Hassan, whose insightful guidance and comprehensive framework provided the foundation for the successful completion of this project. His expert knowledge and constant encouragement were invaluable, enabling me to navigate the complexities of developing an event management web application with confidence and clarity.

My heartfelt thanks go to my project supervisor, Ekereuke Udoh, whose mentorship was instrumental in shaping the direction of this work.

I am also profoundly grateful to Amjad Alam, whose generosity and expertise in IoT were crucial to the integration of the various devices that enhanced the functionality of "Event Syrup." His provision of essential IoT devices, including the Raspberry Pi, RFID sensor, temperature sensor, and RGB lights, enabled the successful automation and monitoring features that are central to the project's innovation.

Finally, I would like to thank my family, friends, and colleagues for their unwavering support and understanding throughout this journey. Their belief in my abilities and their encouragement provided the strength and motivation needed to see this project through to completion.

## Table of Contents

<b>PROJECT REVIEW REPORT .....</b>	<b>1</b>
ON EVENT SYRUP, AN EVENT MANAGEMENT WEB APPLICATION.....	1
<b>ABSTRACT .....</b>	<b>2</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>3</b>
<b>TABLE OF CONTENTS.....</b>	<b>3</b>
<b>CHAPTER 1: INTRODUCTION .....</b>	<b>7</b>
1.1 BACKGROUND .....	7
1.2 OBJECTIVES.....	7
<b>1. DESIGN AND DEVELOPMENT OF THE USER INTERFACE: .....</b>	<b>7</b>
<b>2. REAL-TIME DATA MANAGEMENT: .....</b>	<b>8</b>
<b>3. BACKEND API INTEGRATION: .....</b>	<b>8</b>
<b>4. IOT INTEGRATION: .....</b>	<b>8</b>
<b>5. SCALABILITY AND PERFORMANCE OPTIMIZATION: .....</b>	<b>8</b>

6.	SECURITY AND RELIABILITY:	8
7.	TESTING AND VALIDATION:	8
1.3	SCOPE:	8
1.	USER INTERFACE DESIGN:	8
2.	REAL-TIME DATA MANAGEMENT:	9
3.	BACKEND DEVELOPMENT:	9
4.	IOT INTEGRATION:	9
5.	TESTING AND VALIDATION:	9
6.	DEPLOYMENT AND SCALABILITY:	9
1.4	REPORT STRUCTURE	9
•	CHAPTER 2: LITERATURE REVIEW –	9
•	CHAPTER 3: PROJECT PLANNING –	9
•	CHAPTER 4: TECHNICAL DESIGN –	9
•	CHAPTER 5: EVALUATION –	10
•	CHAPTER 6: CONCLUSION AND FUTURE WORK –	10
•	REFERENCES –	10
•	APPENDICES –	10
	CHAPTER 2: LITERATURE REVIEW	10
2.1	INTRODUCTION TO EVENT MANAGEMENT SYSTEMS	10
2.2	EXISTING EVENT MANAGEMENT SYSTEMS	11
1.	EVENTBRITE	11
2.	CVENT	12
3.	WHOVA	13
4.	BIZZABO	14
2.3	TECHNOLOGIES UNDERPINNING MODERN EVENT MANAGEMENT SYSTEMS	14
1.	WEB DEVELOPMENT FRAMEWORKS	14
2.	REAL-TIME DATABASES	15
3.	BACKEND TECHNOLOGIES	15
4.	IOT INTEGRATION	15
2.4	IoT AND AUTOMATION IN EVENT MANAGEMENT	16
1.	ENVIRONMENTAL MONITORING	16
2.	ATTENDEE TRACKING	16
3.	AUTOMATION OF ROUTINE TASKS	17
2.5	CHALLENGES AND LIMITATIONS OF CURRENT SYSTEMS	18

<b>1. DATA SECURITY AND PRIVACY .....</b>	<b>18</b>
<b>2. SCALABILITY .....</b>	<b>18</b>
<b>3. INTEROPERABILITY .....</b>	<b>18</b>
<b>4. COST AND ACCESSIBILITY .....</b>	<b>18</b>
2.6 HOW "EVENT SYRUP" FITS INTO THE LANDSCAPE.....	19
<b>1. SCALABILITY AND PERFORMANCE.....</b>	<b>19</b>
<b>2. IOT INTEGRATION .....</b>	<b>19</b>
<b>3. COST-EFFECTIVENESS AND ACCESSIBILITY.....</b>	<b>19</b>
<b>4. SECURITY AND RELIABILITY .....</b>	<b>20</b>
2.7 CONCLUSION.....	20
<b>CHAPTER 3: PROJECT PLANNING .....</b>	<b>20</b>
3.1 INTRODUCTION TO PROJECT PLANNING .....	20
3.2 REQUIREMENTS ENGINEERING .....	20
3.2.1 Functional Requirements .....	20
3.2.2 Non- Functional Requirements .....	22
o A.....	29
3.4 PROJECT LIFECYCLE .....	29
<b>1. INITIATION PHASE .....</b>	<b>29</b>
<b>2. PLANNING PHASE .....</b>	<b>30</b>
<b>3. EXECUTION PHASE .....</b>	<b>33</b>
<b>4. MONITORING AND CONTROLLING PHASE.....</b>	<b>34</b>
<b>5. CLOSURE PHASE.....</b>	<b>35</b>
3.5 WORK PRODUCTS.....	36
3.6 ESTIMATION AND RESOURCE ALLOCATION.....	39
3.7 METRICS FOR PROJECT MANAGEMENT.....	40
3.8 RISK MANAGEMENT.....	42
3.9 CONCLUSION.....	42
<b>CHAPTER 4: TECHNICAL DESIGN.....</b>	<b>43</b>
4.1 SYSTEM ARCHITECTURE .....	43
4.2 FRONTEND DESIGN .....	44
4.3 BACKEND DESIGN .....	50
4.4 DATABASE DESIGN.....	53
4.5 IoT INTEGRATION DESIGN.....	55
4.6 SECURITY AND RELIABILITY .....	59
4.7 SCALABILITY AND PERFORMANCE OPTIMIZATION .....	61
4.8 CONCLUSION.....	62
<b>CHAPTER 5: EVALUATION .....</b>	<b>63</b>
5.1 INTRODUCTION .....	63
5.2 TESTING METHODOLOGY .....	63
5.2.1 UNIT TESTING.....	64
5.2.3 SYSTEM TESTING .....	65
5.2.4 USER ACCEPTANCE TESTING (UAT) .....	65
5.3 VERIFICATION AND VALIDATION .....	65

5.3.1 VERIFICATION .....	65
5.3.2 VALIDATION .....	66
5.4.1 FUNCTIONAL OUTCOMES .....	66
5.4.2 NON-FUNCTIONAL OUTCOMES .....	66
5.4.3 PROJECT MANAGEMENT OUTCOMES .....	67
<b>CHAPTER 6: CONCLUSION AND FUTURE WORK.....</b>	<b>67</b>
6.1 SUMMARY OF FINDINGS.....	67
6.2 DISCUSSION OF LIMITATIONS .....	68
6.3 SUGGESTIONS FOR FUTURE WORK .....	68
6.4 CONCLUSION .....	69
<b>APPENDICES.....</b>	<b>70</b>
APPENDIX A – IMPORTANT FRONTEND AND BACKEND PAGES SNAPSHOTS .....	70
<i>Login Page:</i> .....	70
<i>Signup Page:</i> .....	71
<i>Dashboard Page:</i> .....	72
<i>Expenses Handling Page:</i> .....	72
<i>Team Handling Page:</i> .....	72
<i>Rota Managing Page:</i> .....	73
<i>Backend Firebase Rules:</i> .....	74
<i>MUI Libraries Used:</i> .....	75
APPENDIX B – ALL THE FIGMA PROTOTYPES.....	76
<i>Constant Component Prototype:</i> .....	76
APPENDIX C -ABBREVIATIONS: .....	79
APPENDIX D – HARDWARE AND SOFTWARE USED .....	79
<b>REFERENCES .....</b>	<b>80</b>

# Chapter 1: Introduction

## 1.1 Background

The rapid advancement of technology has revolutionized the way events are organized and managed. Traditionally, event management has been a labor-intensive process, involving multiple stakeholders and requiring meticulous coordination of various activities such as registration, scheduling, venue management, and communication. With the advent of digital technologies, these processes have been significantly streamlined, leading to the emergence of event management systems that are not only efficient but also capable of enhancing the overall experience for both organizers and attendees.

Event management systems have evolved from simple tools for managing invitations and registrations to comprehensive platforms that integrate various functionalities, including real-time communication, data analytics, and automation through Internet of Things (IoT) devices. These systems have become indispensable in the modern era, where events range from small community gatherings to large-scale international conferences, each with its own set of unique requirements and challenges.

"Event Syrup," the project discussed in this report, is an event management web application that embodies the latest advancements in technology. It leverages a combination of web development frameworks, real-time database management, and IoT integration to create a platform that not only facilitates the organization of events but also enhances the overall efficiency and effectiveness of the event management process. By incorporating features such as automated environmental monitoring and device control, "Event Syrup" aims to provide a seamless and intelligent solution to the challenges faced by event organizers in today's fast-paced, tech-driven world.

## 1.2 Objectives

The primary objective of "Event Syrup" is to develop a scalable and user-friendly web application that streamlines the event management process. The key objectives of the project are as follows:

### **1. Design and Development of the User Interface:**

To create an intuitive and responsive user interface using ReactJS, ensuring that users can easily navigate through the application and manage event-related activities with minimal effort.

## **2. Real-Time Data Management:**

To implement a robust real-time database using Google Firebase's Firestore, allowing for efficient synchronization of event data, including registrations, schedules, and notifications.

## **3. Backend API Integration:**

To develop a reliable backend using Flask, which will facilitate seamless communication between the front-end interface and the server, as well as support the integration of IoT devices.

## **4. IoT Integration:**

To enhance the functionality of the application by integrating IoT devices such as Raspberry Pi, RFID sensors, temperature sensors, and smart lighting systems. These devices will be used to automate various tasks, including environmental monitoring and device control during events.

## **5. Scalability and Performance Optimization:**

To ensure that the application is scalable and can handle a large number of users and events without compromising performance.

## **6. Security and Reliability:**

To implement security measures that protect user data and ensure the reliability of the application, particularly in the context of real-time operations and IoT device interactions.

## **7. Testing and Validation:**

To rigorously test the application to identify and address any potential issues, ensuring that it meets the required standards of quality, reliability, and performance.

# **1.3 Scope**

The scope of this project encompasses the entire lifecycle of the development of the "Event Syrup" web application, from initial concept and design through to implementation, testing, and deployment. The project focuses on the following key areas:

## **1. User Interface Design:**

Development of a user-friendly interface that caters to both event organizers and participants, allowing for efficient management of events.



## **2. Real-Time Data Management:**

Implementation of a real-time database that supports the dynamic nature of event management, ensuring that all stakeholders have access to up-to-date information.

## **3. Backend Development:**

Creation of a robust backend that facilitates communication between the front-end interface and IoT devices, enabling the automation of various tasks.

## **4. IoT Integration:**

Integration of IoT devices to automate environmental monitoring and device control, enhancing the overall functionality and efficiency of the application.

## **5. Testing and Validation:**

Comprehensive testing of the application to ensure that it meets the necessary standards of quality, security, and performance.

## **6. Deployment and Scalability:**

Deployment of the application on a scalable platform, ensuring that it can handle a large number of users and events simultaneously.

# **1.4 Report Structure**

This report is structured to provide a detailed account of the development process of the "Event Syrup" web application, from initial concept through to final implementation and evaluation. The report is organized into the following chapters:

- **Chapter 2: Literature Review –**

This chapter provides an in-depth review of existing event management systems, relevant technologies, and academic literature. It explores the strengths and weaknesses of current solutions and positions "Event Syrup" within the broader landscape of event management technology.

- **Chapter 3: Project Planning –**

This chapter details the planning process for the project, including the work breakdown structure (WBS) (Elnaz Siami-Irdemoosa, 2015), project lifecycle, estimation, and metrics. It provides an overview of how the project was organized and managed to meet its objectives within the given timeframe.

- **Chapter 4: Technical Design –**

This chapter delves into the technical aspects of the project, including the architecture of the application, design components, and implementation details. It provides a

comprehensive overview of the technologies used and the rationale behind key design decisions.

- **Chapter 5: Evaluation –**

This chapter covers the testing, verification, and validation of the application. It discusses the outcomes of the testing process, identifies any issues encountered, and reflects on the overall performance and reliability of the system.

- **Chapter 6: Conclusion and Future Work –**

The final chapter summarizes the key findings of the project, discusses its limitations, and suggests areas for future research and development. It reflects on the lessons learned during the project and proposes potential improvements to the application.

- **References –**

A comprehensive list of references used throughout the report, including academic papers, books, online resources, and technical documentation.

- **Appendices –**

Supplementary materials, including detailed descriptions of diagrams, tables, code listings, and additional documentation that support the content of the report.

## Chapter 2: Literature Review

### 2.1 Introduction to Event Management Systems

Event management has become an increasingly complex and dynamic field, necessitating the use of sophisticated tools and systems to manage various aspects of events, from planning and registration to execution and post-event analysis. Traditionally, event management relied heavily on manual processes and face-to-face communication, which often led to inefficiencies and human errors. However, with the advent of digital technologies, a wide range of event management systems (EMS) have emerged, providing more efficient and reliable solutions (Ali Dalgic, 2020).

Modern EMS platforms are designed to handle the multifaceted nature of events, offering features like online registration, automated scheduling, attendee management, and real-time communication. These systems are often cloud-based, allowing for scalability and remote access, which is particularly beneficial for large events that require coordination across different locations. The integration of data analytics into EMS has further enhanced their capabilities, enabling organizers to gather insights from attendee behavior, optimize resource allocation, and improve overall event outcomes (Ian Yeoman, Jane Ali-Knight, Martin Robertson, Siobhan Drummond, Una McMahon-Beattie, 2012).

## 2.2 Existing Event Management Systems

Numerous EMS platforms are currently available, each with its own set of features and target audiences. Some of the most prominent systems include:

### 1. Eventbrite



Figure 1: Eventbrite

- **Overview:** Eventbrite (EventBrite, 2024) is one of the most widely used event management platforms, known for its ease of use and extensive feature set. It offers tools for event creation, ticketing, registration, and promotion, making it a popular choice for both small and large events.
- **Key Features:** Online ticketing, customizable registration forms, integration with social media, mobile app for event management, analytics and reporting.
- **Limitations:** While Eventbrite is powerful, it may not be the most cost-effective solution for smaller events due to its pricing model, which is based on ticket sales.

## 2. Cvent

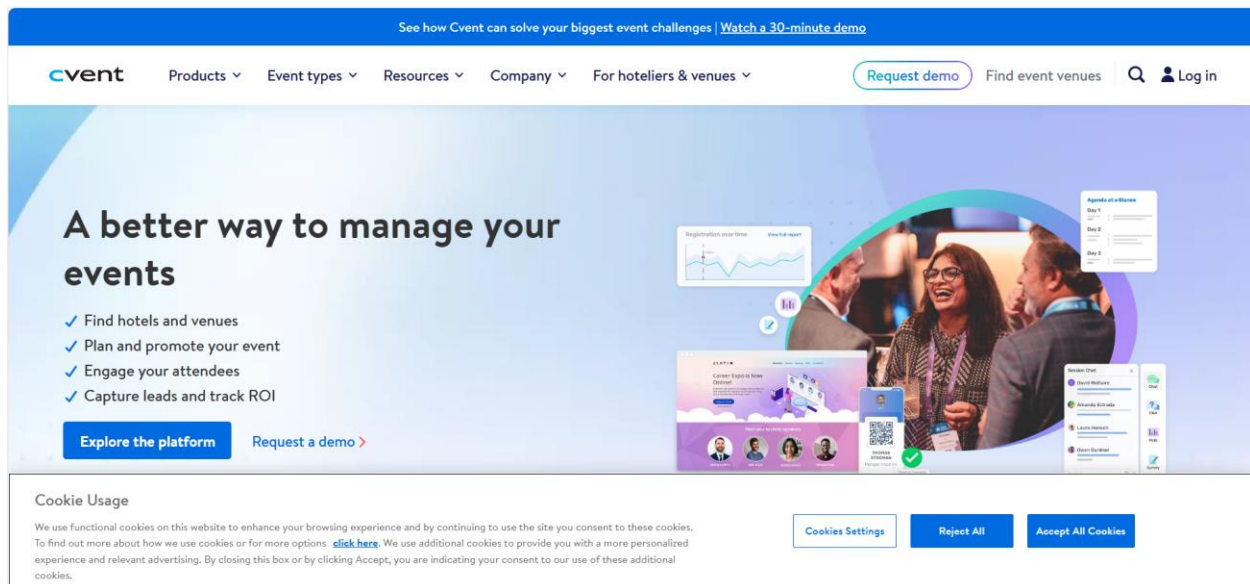


Figure 2: Cvent

- **Overview:** Cvent (Cvent, 2024) is a comprehensive EMS designed for larger, more complex events such as conferences and corporate gatherings. It offers a wide range of tools, from event marketing and management to venue sourcing and on-site solutions.
- **Key Features:** Attendee management, event marketing automation, mobile event apps, on-site check-in, real-time analytics, and reporting.
- **Limitations:** The system's complexity can be overwhelming for smaller events, and the cost can be prohibitive for organizations with limited budgets.

### 3. Whova

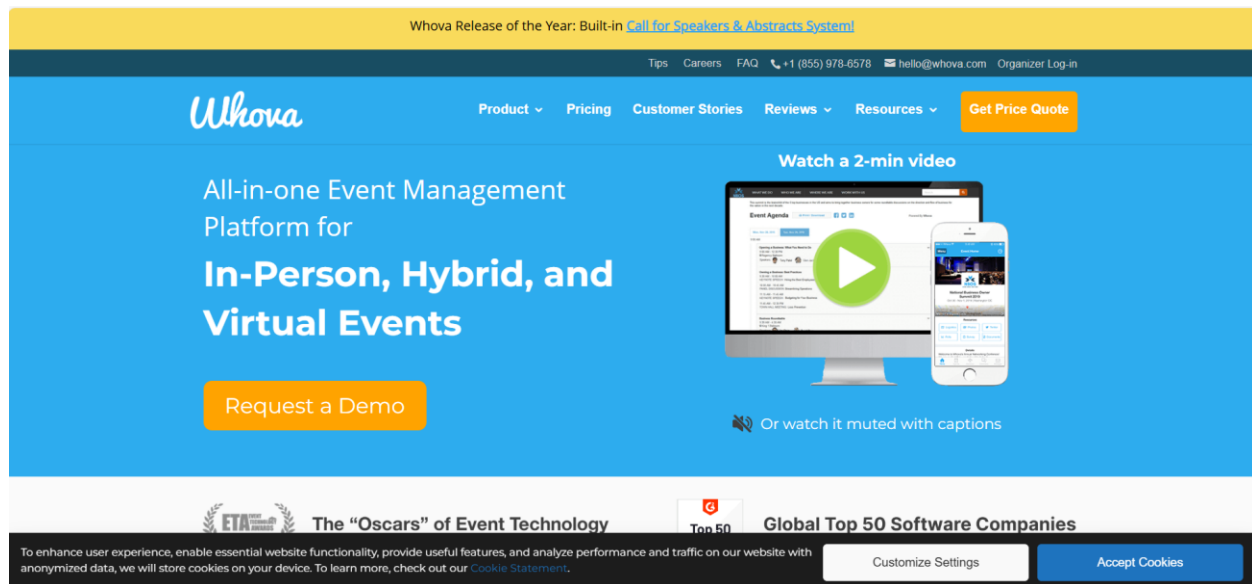


Figure 3: Whova

**Overview:** Whova (Whova, 2024) is an all-in-one event management platform that focuses on enhancing attendee engagement through its mobile app and interactive features. It is particularly popular for academic conferences and networking events.

**Key Features:** Customizable agenda, attendee networking, live polling and Q&A, event announcements, and post-event surveys.

**Limitations:** While Whova excels in attendee engagement, it may lack some of the advanced event management features required for larger, more complex events.

## 4. Bizzabo

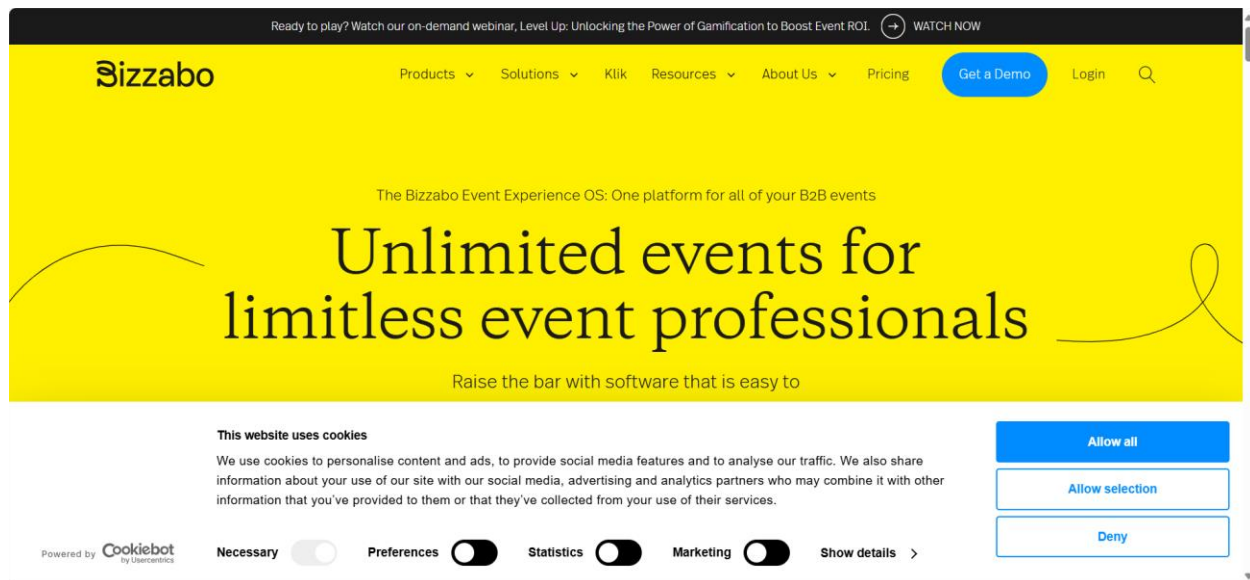


Figure 4: Bizzabo

- **Overview:** Bizzabo (Bizzabo, 2024) is a cloud-based EMS that integrates event management with marketing and analytics. It is designed to provide a seamless experience for both organizers and attendees, with a focus on driving event success through data-driven insights.
- **Key Features:** Event website builder, email marketing, ticketing and registration, CRM integration, analytics dashboard, and networking tools.
- **Limitations:** Bizzabo's feature set may be overkill for small events, and its pricing can be a barrier for smaller organizations.

These existing systems demonstrate the diverse landscape of event management technology, each catering to different needs and scales of operation. However, they also reveal certain gaps, particularly in the integration of IoT for enhanced automation and environmental monitoring, which is where "Event Syrup" seeks to innovate.

## 2.3 Technologies Underpinning Modern Event Management Systems

The success of modern event management systems is largely driven by advancements in several key technologies:

### 1. Web Development Frameworks

- **ReactJS:** A JavaScript library for building user interfaces, ReactJS (ReactJs, 2024) is popular due to its component-based architecture, which allows for the development of highly interactive and responsive UIs. Its ability to handle

complex state management and real-time updates makes it ideal for event management applications.

- **Angular and Vue.js:** Other popular frameworks include Angular and Vue.js (Emadamerho-Atori Nefe, 2023), both of which offer powerful tools for building dynamic web applications. Angular, maintained by Google, is known for its comprehensive feature set, while Vue.js is praised for its simplicity and flexibility.

## 2. Real-Time Databases

- **Google Firebase Firestore:** A NoSQL cloud database, Firestore is designed for real-time data synchronization, making it perfect for applications that require instantaneous updates, such as live event management systems. Its integration with Firebase Authentication and Firebase Hosting also simplifies the development process (Google, 2024).
- **MongoDB:** An alternative NoSQL database, MongoDB is known for its scalability and flexibility, making it a popular choice for large-scale applications. However, it lacks the built-in real-time features of Firestore, which may require additional configuration (Goray, 2024).

## 3. Backend Technologies

- **Flask (Python):** Flask is a lightweight web framework for Python, known for its simplicity and flexibility. It is well-suited for building RESTful APIs, which are essential for backend development in event management systems. Flask's modular nature allows developers to choose the components they need, making it highly customizable (Python, 2024).
- **Django (Python):** Another Python framework, Django, is more feature-rich than Flask, offering built-in tools for user authentication, ORM, and admin interfaces. While Django can be advantageous for projects requiring rapid development, its complexity may be unnecessary for smaller applications (Jeff Forcier, Paul Bissex, Wesley J Chun, 2008).

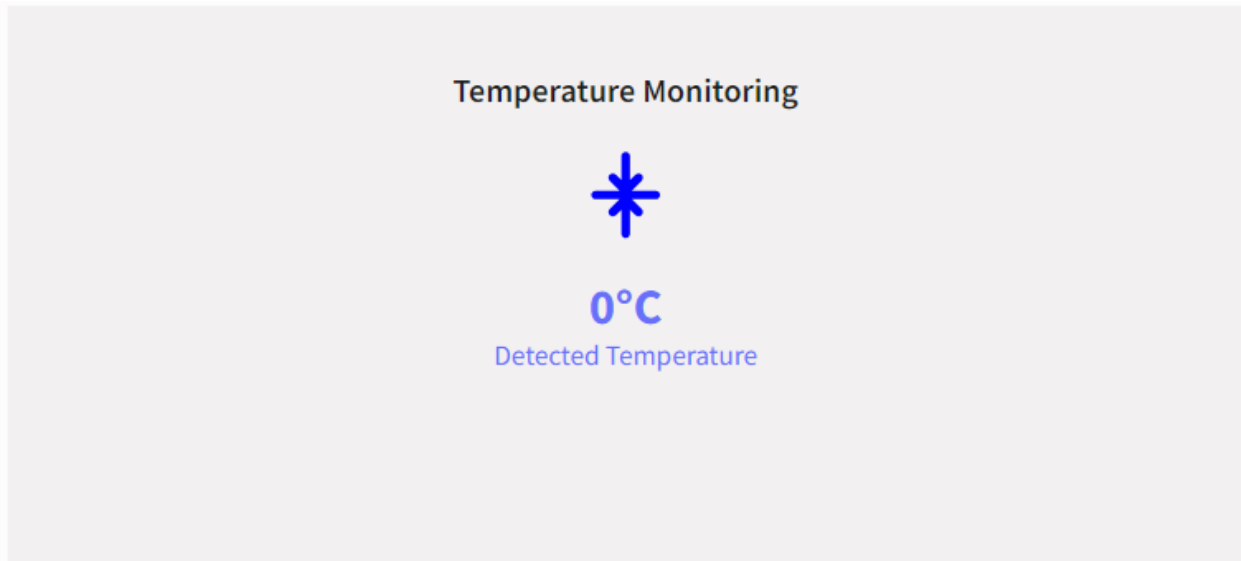
## 4. IoT Integration

- **Raspberry Pi:** A versatile and affordable microcontroller, Raspberry Pi is widely used for IoT projects. In the context of event management, it can be used to automate tasks such as environmental monitoring, attendee tracking, and device control (RaspberryPi, 2024).
- **Sensors (RFID, Temperature, Light):** IoT devices such as RFID sensors for attendee tracking, temperature sensors for environmental monitoring, and smart lighting systems are becoming increasingly important in event management, providing real-time data and enhancing the overall attendee experience (Subhas Chandra Mukhopadhyay, 2014).

## 2.4 IoT and Automation in Event Management

The integration of IoT in event management represents a significant step forward in the automation and efficiency of event operations. IoT devices offer real-time data collection and control, enabling organizers to automate tasks that would otherwise require manual intervention (Jerker Delsing, 2017).

### 1. Environmental Monitoring



*Figure 5: Temperature Detection Display*

- **Applications:** IoT devices like temperature and humidity sensors can monitor environmental conditions in real-time, ensuring optimal conditions for attendees. For instance, in large conferences or exhibitions, maintaining a comfortable environment is crucial for attendee satisfaction.
- **Benefits:** Automated environmental control reduces the need for manual adjustments, ensuring consistent conditions throughout the event. This is particularly beneficial for events held in large venues where manual monitoring would be impractical.

### 2. Attendee Tracking





Figure 6: Attendance tappin realtime list with Timestamp

- **Applications:** RFID and NFC (Near Field Communication) technologies enable seamless tracking of attendees, providing valuable data on attendee movement and behavior. This can be used to optimize event layouts, manage crowd flow, and enhance security.
- **Benefits:** Real-time tracking helps in understanding attendee preferences and behavior, allowing organizers to make data-driven decisions for future events. It also enhances security by ensuring that only authorized individuals have access to certain areas.

### 3. Automation of Routine Tasks

← BACK

Today	Back	Next	08/22/2024 – 09/21/2024				Month	Week	Day	Agenda
Date		Time		Event						
Thu Aug 22		11:03 pm – 11:03 am		John Doe - FIFA World Cup						
		11:03 pm – 11:03 am		Ashma Chapagain - FiFA World Cup						
		11:04 pm		Anup - Ulster Farewell						

Figure 7: Rota Creation Feature

- **Applications:** IoT devices can automate routine tasks such as lighting control, audio-visual equipment management, and even catering services. For example, smart lighting systems can adjust the lighting based on the time of day or the number of attendees in a room.

- **Benefits:** Automation reduces the workload on event staff, allowing them to focus on more critical tasks. It also improves the overall efficiency of event operations, leading to a smoother and more enjoyable experience for attendees.

## 2.5 Challenges and Limitations of Current Systems

Despite the advances in event management systems and IoT integration, several challenges and limitations remain:

### 1. Data Security and Privacy

- **Challenges:** The use of real-time databases and IoT devices introduces significant concerns regarding data security and privacy. Ensuring that sensitive information such as attendee data is protected from unauthorized access is critical.
- **Limitations:** Many current systems lack robust security measures, particularly in the context of IoT devices, which are often vulnerable to hacking and other cyber threats.

### 2. Scalability

- **Challenges:** As events grow in size and complexity, the ability of the EMS to scale effectively becomes a concern. Systems that are not designed with scalability in mind may experience performance issues, such as slow response times or data synchronization failures.
- **Limitations:** While cloud-based systems offer scalability, the cost associated with scaling up can be prohibitive for smaller organizations. Additionally, the complexity of managing a large-scale event may require specialized skills that are beyond the capabilities of the current EMS.

### 3. Interoperability

- **Challenges:** Many EMS platforms are not designed to integrate seamlessly with other systems, such as CRM (Customer Relationship Management) (Kelly Main, Rachel Williams, 2024) tools or financial software. This lack of interoperability can lead to inefficiencies and data silos.
- **Limitations:** The integration of third-party tools often requires custom development, which can be time-consuming and expensive. Furthermore, the lack of standardization across EMS platforms can complicate the process of migrating data between systems.

### 4. Cost and Accessibility

- **Challenges:** High costs associated with advanced EMS platforms can be a barrier for smaller organizations or events with limited budgets. Additionally, the

complexity of these systems may require specialized training, which can further increase the cost.

- **Limitations:** The affordability and ease of use of EMS platforms remain significant concerns, particularly for non-profit organizations or community-based events. Simplifying these systems without sacrificing functionality is a key challenge that needs to be addressed.

## 2.6 How "Event Syrup" Fits Into the Landscape

"Event Syrup" is designed to address some of the key limitations identified in existing event management systems by leveraging modern web technologies and IoT integration. Its focus on scalability, real-time data management, and automation positions it as a forward-looking solution that can meet the needs of both small and large events.

### 1. Scalability and Performance

- **Innovation:** By using ReactJS for the front-end and Google Firebase's Firestore for real-time data synchronization, "Event Syrup" is built to scale efficiently. Its cloud-based architecture ensures that the system can handle large numbers of users and events without compromising performance.
- **Fit:** This makes "Event Syrup" particularly well-suited for large-scale events where real-time data access and scalability are critical.

### 2. IoT Integration

- **Innovation:** The integration of IoT devices such as Raspberry Pi and various sensors allows "Event Syrup" to automate tasks that would traditionally require manual intervention. This not only enhances operational efficiency but also provides a more personalized and responsive experience for attendees.
- **Fit:** This positions "Event Syrup" as a unique offering in the market, particularly for events that require advanced automation and environmental monitoring.

### 3. Cost-Effectiveness and Accessibility

- **Innovation:** "Event Syrup" is designed to be cost-effective, making it accessible to smaller organizations and events with limited budgets. By using open-source technologies and cloud-based services, the system reduces the overall cost of ownership while still offering advanced features.
- **Fit:** This focus on affordability and accessibility makes "Event Syrup" an attractive option for a wide range of events, from small community gatherings to large conferences.

## 4. Security and Reliability

- **Innovation:** By incorporating robust security measures and ensuring reliable operation through rigorous testing, "Event Syrup" addresses some of the key concerns associated with data security and system reliability.
- **Fit:** This makes "Event Syrup" a trustworthy solution for events where data security and system uptime are critical, such as corporate events or high-profile conferences.

## 2.7 Conclusion

The literature review highlights the evolving landscape of event management systems and the technologies that underpin them. While existing EMS platforms offer a wide range of features, there are still significant challenges related to scalability, IoT integration, security, and cost. "Event Syrup" seeks to address these challenges by providing a scalable, cost-effective, and feature-rich solution that leverages modern web technologies and IoT devices. Through its innovative approach, "Event Syrup" not only enhances the efficiency of event management but also opens new possibilities for automation and personalization in the event industry.

# Chapter 3: Project Planning

## 3.1 Introduction to Project Planning

Project planning is a crucial phase in the lifecycle of any project, laying the foundation for successful execution and delivery. This chapter details the planning processes involved in the "Event Syrup" project, including the development of the Work Breakdown Structure (WBS), the project lifecycle, key work products, effort estimation, and the metrics used to measure progress and success.

## 3.2 Requirements Engineering

### 3.2.1 Functional Requirements

Requirement Engineering (Geeks for Geeks, 2024) is a crucial phase in the "Event Syrup" project, focusing on capturing and translating stakeholder needs into clear, actionable requirements. The process began with requirements elicitation, where input from event organizers, technical experts, and potential users was gathered through interviews, surveys, and workshops. This helped identify key functionalities, such as RFID-based attendee tracking and real-time environmental monitoring. These were then documented in a structured format, categorizing them into functional and non-functional requirements, with attributes like priority and fit criteria. Validation involved stakeholder reviews and prototype demonstrations to ensure alignment with expectations and technical feasibility.

Requirement ID	Requirement Type	Description	Priority
FR1	Functional	The system shall allow users to register and log in to the web application using their credentials.	High
FR2	Functional	The system shall allow users to register for events, view event details.	High
FR3	Functional	The system shall integrate with an RFID sensor to track attendee check-ins and check-outs automatically during events.	High
FR4	Functional	The system shall monitor temperature using a temperature sensor and provide real-time data to the web application.	Medium
FR5	Functional	The system shall allow event organizers to control event lighting via the web application, adjusting brightness and color based on event requirements.	High
FR6	Functional	The system shall generate reports based on event data, including attendee statistics and environmental conditions monitored by IoT devices.	Medium

Functional requirements describe the specific behaviors or functions of your system. Given your project, these include user interactions, sensor operations, and data management.

### 3.2.2 Non- Functional Requirements

Non-functional requirements specify criteria that judge the operation of a system, rather than specific behaviors. These include performance, security, and usability aspects.

Requirement ID	Requirement Type	Description	Priority
NFR1	Non-Functional	The system shall handle up to 500 concurrent users without significant performance degradation.	High
NFR2	Non-Functional	The system shall provide 99.9% uptime, ensuring high availability for event management.	High
NFR3	Non-Functional	All communications between the web application and IoT devices shall be encrypted to ensure data security and integrity.	High
NFR4	Non-Functional	The user interface shall be responsive, ensuring compatibility across various devices, including desktops, tablets, and smartphones.	Medium
NFR5	Non-Functional	The system shall comply with relevant data protection regulations, including GDPR, ensuring the privacy of user data.	High

NFR6	Non-Functional	The system shall provide real-time updates to the user interface whenever there is a change in event data or sensor readings.	High
------	----------------	---	------

### 3.2.3 Volere Snow Card

The Volere Snow Card (Suzzane Robertson, 2024) is a useful tool for capturing detailed requirements. Here are the Volere Cards of all the Key Functional Requirements and Non-Functional Requirements of Event Syrup:

Functional Requirements:

#### 1.Integration of Raspberry Pi for Environmental Monitoring

Attribute	Details
Requirement ID	FR1
Description	Integration of Raspberry Pi to monitor and control environmental conditions such as temperature, light, and humidity during events.
Rationale	Ensures real-time adjustments to maintain optimal environmental conditions for event attendees.
Originator	Amjad Alam Sir
Fit Criterion	The system should display real-time temperature and humidity data on the dashboard within 5 seconds of sensor data update.
Priority	High
Dependencies	Database Storage and UI Dashboard
Conflicts	Potential delays due to network latency or sensor failure.
Supporting Materials	Raspberry Pi documentation, sensor specifications.
History	Proposed on 16/07/2024

#### 2. User Registration and Event Management

Attribute	Details
Requirement ID	FR2
Description	Admin must be able to register for events, manage their staffs, and view event details through the web application.
Rationale	Facilitates user engagement and allows organizers to manage attendees efficiently.
Originator	Anup Chapagain
Fit Criterion	Users should be able to complete registration within 2 minutes, and view their registration status instantly.
Priority	High

Dependencies	Notification Panel
Conflicts	No Conflicts encountered throughout.
Supporting Materials	User interface design mockups, registration flow diagram.
History	Proposed on 16/07/2024

### 3. RFID Sensor Integration for Attendee Tracking

Attribute	Details
Requirement ID	FR3
Description	The system shall integrate with an RFID sensor to track attendee check-ins and check-outs automatically during events.
Rationale	Ensures accurate and efficient tracking of attendee movements, enhancing security and event management.
Originator	Anup Chapagain
Fit Criterion	The system should record and display check-in/check-out times within 3 seconds of RFID scan.
Priority	High
Dependencies	RFID sensor integration, Database system, Real-time data processing
Conflicts	Potential interference with other wireless devices or RFID signal failures.
Supporting Materials	RFID sensor documentation, Event management system requirements.
History	Proposed on 18/07/2024

### 4. Real-Time Temperature Monitoring via Sensor

Attribute	Details
Requirement ID	FR4
Description	The system shall monitor temperature using a temperature sensor and provide real-time data to the web application.
Rationale	Enables monitoring and control of environmental conditions to ensure attendee comfort and equipment safety.
Originator	Anup Chapagain
Fit Criterion	The system should display real-time temperature data on the dashboard within 5 seconds of sensor data update.
Priority	Medium
Dependencies	Temperature sensor, Real-time data processing, UI Dashboard
Conflicts	Possible delays due to sensor inaccuracies or network issues.
Supporting Materials	Temperature sensor specifications, Integration guide.
History	Proposed on 18/07/2024



## 5. Event Lighting Control through Web Application

Attribute	Details
Requirement ID	FR5
Description	The system shall allow event organizers to control event lighting via the web application, adjusting brightness and color based on event requirements.
Rationale	Provides flexibility in creating the desired ambiance and enhancing the visual experience during events.
Originator	Anup Chapagain
Fit Criterion	The system should allow lighting adjustments with a response time of less than 2 seconds from the control command.
Priority	High
Dependencies	Lighting control system, Web application interface, Real-time command processing
Conflicts	Potential issues with lighting system compatibility or network delays.
Supporting Materials	Lighting system documentation, User interface design specifications.
History	Proposed on 18/07/2024

## 6. Event Data Reporting and Analysis

Attribute	Details
Requirement ID	FR6
Description	The system shall generate reports based on event data, including attendee statistics and environmental conditions monitored by IoT devices.
Rationale	Provides valuable insights and documentation for post-event analysis and future planning.
Originator	Anup Chapagain
Fit Criterion	The system should generate comprehensive reports within 1 minute of request, covering all relevant event data.
Priority	Medium
Dependencies	Data storage system, Reporting software, IoT device integration
Conflicts	Potential delays in report generation due to large data volumes or system load.
Supporting Materials	Reporting tool documentation, IoT device data specifications.
History	Proposed on 19/07/2024

## Non- Functional Requirements

### 1. Concurrent Users Handling

Attribute	Details
-----------	---------

Requirement ID	NFR1
Description	The system shall handle up to 500 concurrent users without significant performance degradation.
Rationale	Ensures the system can accommodate high traffic and user load without affecting performance, crucial for large-scale events.
Originator	Anup Chapagain
Fit Criterion	The system should maintain response times within acceptable limits (e.g., <2 seconds) even with 500 concurrent users.
Priority	High
Dependencies	Server infrastructure, load balancing solutions (Google Firebase)
Conflicts	Potential conflicts with hardware limitations or existing infrastructure
Supporting Materials	Firebase Analytics
History	Proposed on 16/07/2024

## 2.Uptime

Attribute	Details
Requirement ID	NFR2
Description	The system shall provide 99.9% uptime, ensuring high availability for event management.
Rationale	Guarantees that the system is highly available and reliable, minimizing downtime and disruption for users.
Originator	Anup Chapagain
Fit Criterion	System availability should be 99.9% or higher over a one-year period, translating to no more than 8.77 hours of downtime annually.
Priority	High
Dependencies	Cloud services, backup and recovery systems (Google Firebase)
Conflicts	Potential conflicts with maintenance schedules or unexpected outages due to limited resources in Google Firebase free account.
Supporting Materials	Firebase Analytics(ie: Uptime monitoring reports, maintenance schedule)
History	Proposed on 16/07/2024

## 3. Data Encryption

Attribute	Details
Requirement ID	NFR3
Description	All communications between the web application and IoT devices shall be encrypted to ensure data security and integrity.

Rationale	Protects sensitive data and ensures secure communication between systems, vital for maintaining user trust and data confidentiality.
Originator	Anup Chapagain
Fit Criterion	Data should be encrypted using industry-standard protocols (e.g., TLS/SSL) with no data leaks or breaches.
Priority	High
Dependencies	Encryption protocols, certificate management when hosted with a Cloud Services.
Conflicts	Potential conflicts with performance overhead due to encryption.
Supporting Materials	Encryption Policy
History	Proposed on 17/07/2024

#### 4. Responsive User Interface

Attribute	Details
Requirement ID	NFR4
Description	The user interface shall be responsive, ensuring compatibility across various devices, including desktops, tablets, and smartphones.
Rationale	Enhances user experience by ensuring the application is accessible and functional on different devices, improving engagement and usability.
Originator	Anup Chapagain
Fit Criterion	The interface should adjust seamlessly to different screen sizes and orientations, maintaining usability and aesthetic consistency.
Priority	Medium
Dependencies	Responsive design framework, cross-device testing
Conflicts	Potential conflicts with design consistency across different devices.
Supporting Materials	Design mockups, cross-device testing results
History	Proposed on 17/07/2024

#### 5. Compliance with Data Protection Regulations (GDPR)

Attribute	Details
Requirement ID	NFR5
Description	The system shall comply with relevant data protection regulations, including GDPR, ensuring the privacy of user data.
Rationale	Protects user privacy and ensures compliance with legal requirements, preventing potential fines and reputational damage.
Originator	Anup Chapagain

Fit Criterion	The system should successfully pass a GDPR compliance audit with no critical issues identified.
Priority	High
Dependencies	Legal review, Data encryption protocols, User consent mechanisms
Conflicts	Potential limitations in data processing or storage due to regulatory constraints.
Supporting Materials	GDPR guidelines, Data protection compliance checklist.
History	Proposed on 17/07/2024

## 6. Real-Time User Interface Updates for Event Data and Sensor Readings

Attribute	Details
Requirement ID	NFR6
Description	The system shall provide real-time updates to the user interface whenever there is a change in event data or sensor readings.
Rationale	Ensures that users have immediate access to the most current information, enhancing decision-making and event management efficiency.
Originator	Anup Chapagain
Fit Criterion	The user interface should reflect updates within 1 second of data change or sensor input.
Priority	High
Dependencies	Real-time data processing, Front-end and back-end synchronization, Network stability
Conflicts	Possible delays due to network latency or server load.
Supporting Materials	Real-time processing framework documentation, UI update protocols.
History	Proposed on 17/07/2024

## ○ A

### Work Breakdown Structure (WBS) - Event Syrup Project

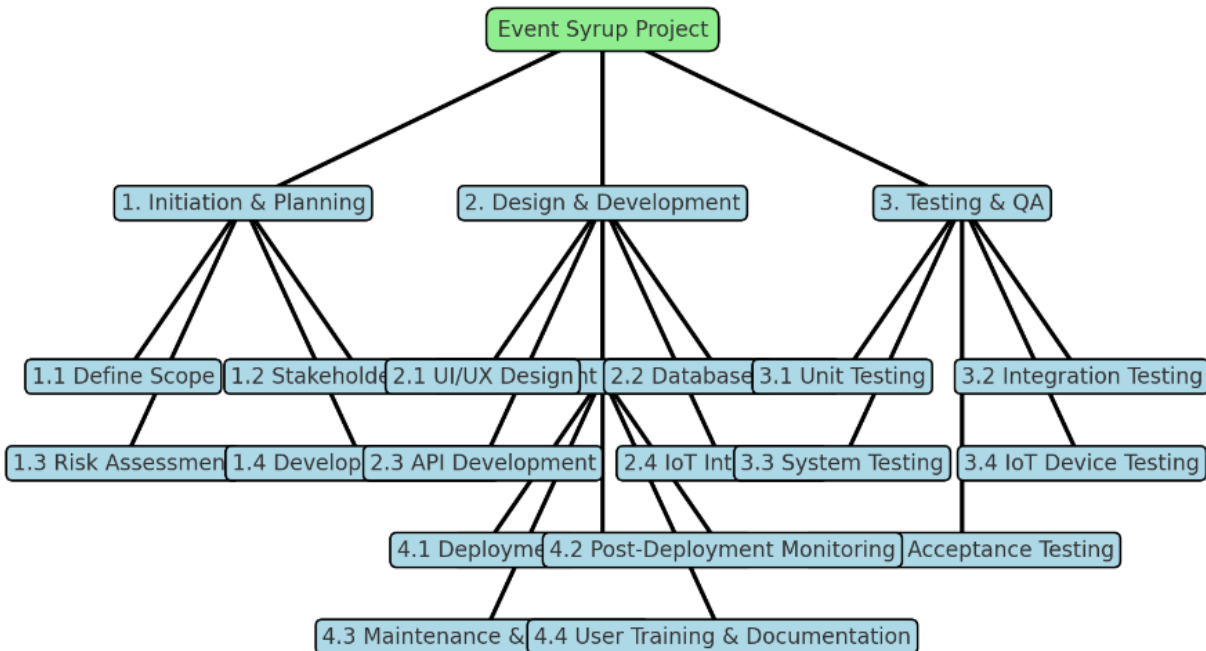


Figure 8: WBS Event Syrup

## 3.4 Project Lifecycle

The project lifecycle for "Event Syrup" follows the Agile methodology, which is well-suited for software development projects where requirements can evolve over time. Agile allows for flexibility and iterative development, making it ideal for managing complex projects with changing needs. The lifecycle consists of several phases, each with specific activities to ensure the project's success (H. Hrablik Chovanova; R. Husovic; D. Babcanova; H. Makysova, 2020).

### 1. Initiation Phase

- Define Project Objectives, Scope, and Deliverables:
  1. **Objectives:** Establish the primary goals of the "Event Syrup" project, including developing a scalable event management system with IoT integration.
  2. **Scope:** Outline the project boundaries, deliverables, and what is included or excluded.
  3. **Deliverables:** Identify key deliverables such as the web application, IoT integrations, and documentation.

- **Conduct Stakeholder Analysis and Identify Risks:**
  - Stakeholder Analysis: Identify and document the key stakeholders, their needs, and their impact on the project.
  - Risk Identification: Recognize potential risks and challenges, such as technical difficulties or resource constraints.
- **Develop a High-Level Project Plan:**
  - **Plan Overview:** Create a high-level plan that outlines the project phases, major milestones, and overall timeline.
  - **Resource Allocation:** Define initial resource requirements and budget estimates.

## 2. Planning Phase

- **Refine the Project Plan with Detailed Schedules, Resources, and Budgets:**
  - **Detailed Scheduling:** Break down the project timeline into detailed schedules, including sprints and deadlines.
  - **Resource Planning:** Allocate resources, including team members, tools, and technologies required for each phase.
  - **Budget Planning:** Refine budget estimates based on detailed project requirements and resource allocations.
- **Break Down Tasks in the WBS and Assign Responsibilities:**
  - **WBS Development:** Decompose project tasks into smaller, manageable components as outlined in the Work Breakdown Structure.
  - **Task Assignment:** Assign responsibilities for each task.

Task	Start Date	End Date	Duration
Define Scope	June 17, 2024	June 20, 2024	4 days
Stakeholder Analysis	June 21, 2024	June 25, 2024	5 days
Risk Assessment	June 26, 2024	June 30, 2024	5 days
Develop Project Plan		July 4, 2024	5 days
UI/UX Design	July 5, 2024	July 9, 2024	5 days
Database Design	July 10, 2024	July 14, 2024	5 days
API Development	July 15, 2024	July 18, 2024	4 days
IoT Integration	July 19, 2024	July 21, 2024	3 days

Figure 9: Work Sprint Planning

- Prepare for Sprint Planning in the Agile Framework:
  - **Sprint Planning:** Develop plans for initial sprints, including setting objectives, defining user stories, and preparing backlog items.

## KAN board

Q Search



GROUP

TO DO 1

Deployment & Maintenance

☒ KAN-51



+ Create issue

IN PROGRESS 1

Testing & QA

☒ KAN-35



DONE 2 ✓

Initiation & Planning

☒ KAN-10



Design & Development

☒ KAN-19



Figure 10: Kanban Board



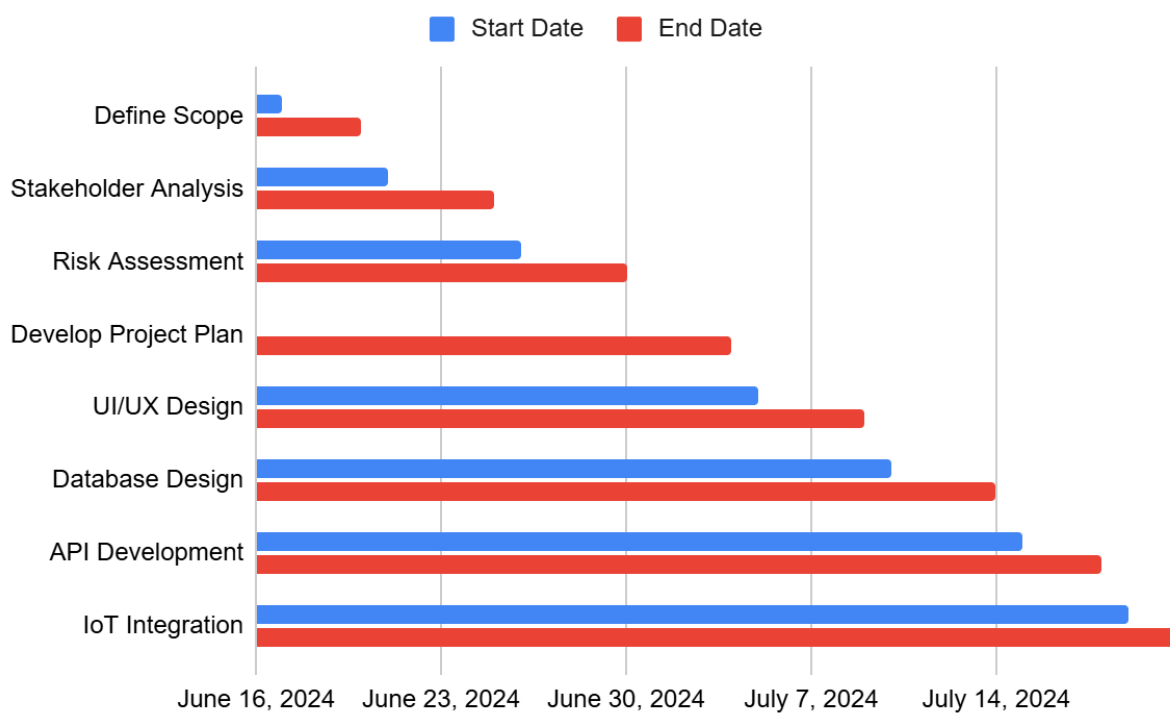
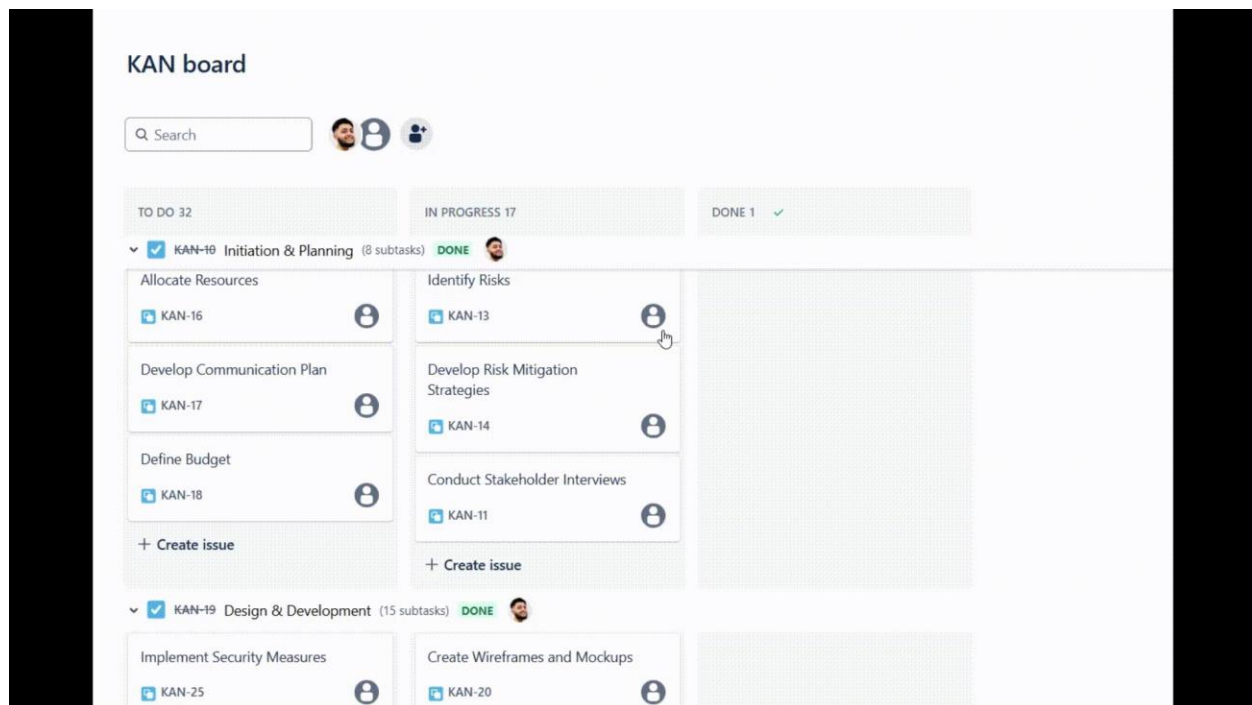


Figure 11: Stacked Bar Gantt Chart made using Google Sheets

### 3. Execution Phase

- Design and Develop the Web Application and IoT Integration in Iterative Sprints:

- **Sprint Execution:** Conduct iterative sprints to design, develop, and integrate features, including frontend development, API creation, and IoT integration.
- **Feature Development:** Focus on delivering functional increments that contribute to the overall system.
- Conduct Daily Stand-Up Meetings to Review Progress and Address Issues:
  - **Stand-Up Meetings:** Hold daily meetings to review progress, discuss obstacles, and align on priorities for the day.
- Implement Features Based on User Stories and Acceptance Criteria:
  - **User Stories:** Develop features according to user stories and acceptance criteria defined during sprint planning.
  - **Acceptance Testing:** Ensure that each feature meets the acceptance criteria before moving to the next sprint.
- Integrate and Test Components After Each Sprint:
  - **Integration:** Combine components developed in each sprint and test them for functionality and compatibility.
  - **Testing:** Perform integration testing to validate that new features work with existing components.

#### 4. Monitoring and Controlling Phase

- **Track Progress Using Metrics Like Burn-Down Charts and Velocity:**
  - **Progress Tracking:** Use burn-down charts to monitor work completed versus work remaining.
  - **Velocity Measurement:** Measure the team's velocity to assess the rate of progress and adjust planning as needed.

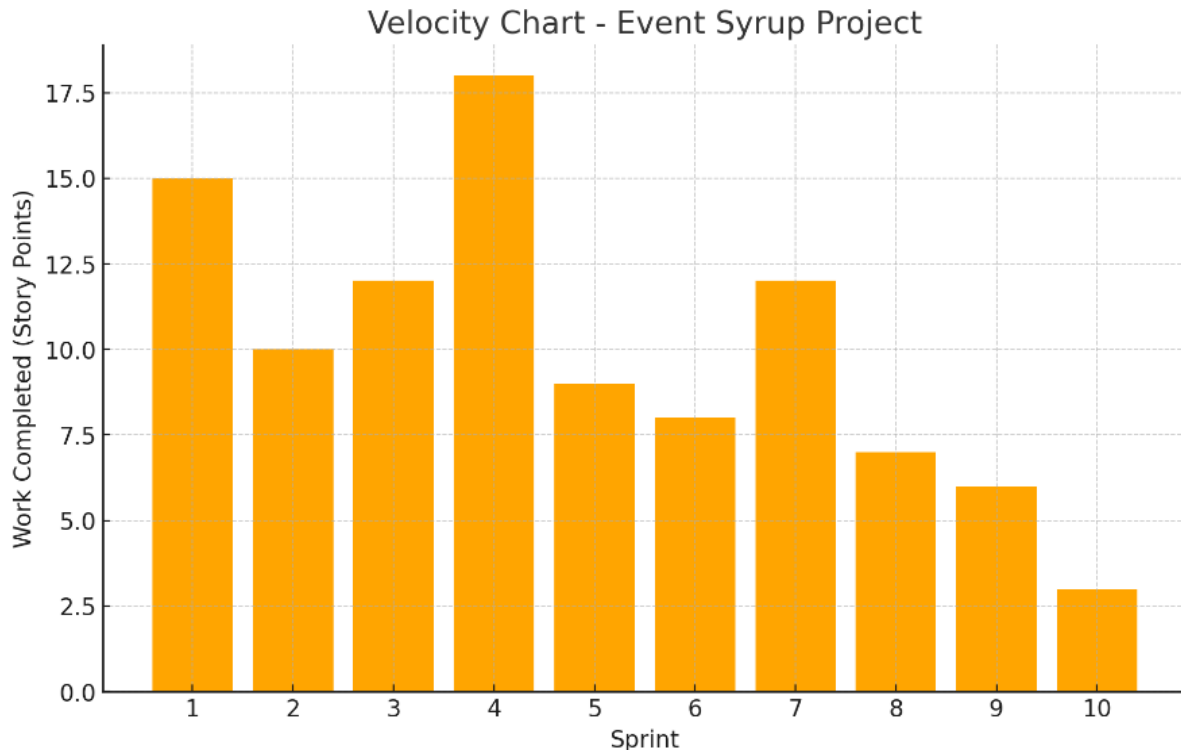


Figure 12: Work Velocity Chart made in Google Sheets

- **Review and Adjust Project Plans Based on Feedback and Performance Data:**
  - **Plan Review:** Regularly review project plans and performance data to identify any needed adjustments.
  - **Adjustments:** Make necessary changes to the project plan based on feedback and performance metrics.
- **Conduct Sprint Reviews and Retrospectives to Improve the Process:**
  - **Sprint Reviews:** Evaluate completed work and gather feedback from stakeholders.
  - **Retrospectives:** Reflect on the sprint process, identify areas for improvement, and implement changes to enhance future sprints.

## 5. Closure Phase

- **Deploy the Final Product and Conduct Post-Deployment Monitoring:**
  - **Deployment:** Release the final version of the web application and IoT integrations to the production environment.
  - **Post-Deployment Monitoring:** Monitor system performance and stability post-deployment to ensure it meets expectations.
- **Complete User Training and Documentation:**
  - **User Training:** Provide training sessions and materials to end-users to ensure effective use of the system.

- **Documentation:** Finalize and distribute user manuals, system documentation, and support materials.
- **Review Project Outcomes, Document Lessons Learned, and Close the Project:**
  - **Outcome Review:** Assess the project outcomes against the initial objectives and success criteria.
  - **Lessons Learned:** Document lessons learned throughout the project to inform future projects.
  - **Project Closure:** Complete administrative tasks, finalize project documentation, and formally close the project.

The Agile lifecycle allows for frequent reassessment of project priorities and enables the team to adapt to changing requirements. This adaptability is key to the successful delivery of "Event Syrup," where user feedback and evolving needs are critical.

## 3.5 Work Products

The primary work products of the "Event Syrup" project are critical deliverables that contribute to the project's success across different phases of the project lifecycle. Each work product plays a key role in ensuring the system meets its objectives and delivers high quality.

### 1. Project Documentation

- **Project Plan:**
  - **Description:** A comprehensive document outlining the project scope, objectives, timelines, resource allocation, and overall strategy. It serves as a roadmap for project execution.
  - **Role:** Guides project execution and provides a reference for managing project progress and changes.
- **WBS Documentation:**
  - **Description:** Detailed documentation of the Work Breakdown Structure, breaking down the project into smaller, manageable tasks and deliverables.
  - **Role:** Helps in task management, assignment, and tracking progress throughout the project lifecycle.
- **Risk Management Plan:**

- **Description:** A plan that identifies potential risks, assesses their impact, and outlines strategies for mitigation. It includes risk registers and response plans.
- **Role:** Ensures proactive risk management, minimizing the impact of potential issues on the project's success.
- **User Stories and Acceptance Criteria:**
  - **Description:** Documentation of user requirements and expectations in the form of user stories, along with acceptance criteria defining what constitutes successful completion.
  - **Role:** Provides a clear understanding of user needs and helps in validating that developed features meet user expectations.

## 2. Design Artifacts

- **UI/UX Design Mockups:**
  - **Description:** Visual representations of the user interface and experience, including wireframes, mockups, and prototypes.
  - **Role:** Guides the development of the user interface, ensuring it meets user needs and provides a good user experience.
- **Database Schemas:**
  - **Description:** Detailed diagrams and descriptions of the database structure, including tables, relationships, and constraints.
  - **Role:** Defines how data is stored and organized, ensuring efficient data management and retrieval.
- **API Specifications:**
  - **Description:** Documentation detailing the API endpoints, request/response formats, and protocols for communication between frontend and backend systems.
  - **Role:** Provides clear guidelines for API development and integration, ensuring seamless communication between components.
- **IoT Integration Design:**
  - **Description:** Design documents outlining the integration of IoT devices with the web application, including hardware setup and communication protocols.
  - **Role:** Guides the integration process, ensuring that IoT devices are properly connected and interact effectively with the web application.

### 3. Development Outputs

- **Source Code (Frontend and Backend):**
  - **Description:** The actual code written for the frontend (ReactJS) and backend (Flask) components of the application.
  - **Role:** Represents the core functionality of the web application and backend services.
- **API Endpoints:**
  - **Description:** The implemented API endpoints that facilitate communication between the frontend, backend, and IoT devices.
  - **Role:** Provides the necessary interfaces for data exchange and integration.
- **IoT Device Control Scripts:**
  - **Description:** Scripts and code used to control and interact with IoT devices, including Raspberry Pi configurations.
  - **Role:** Enables automation and monitoring functionalities within the event management system.

### 4. Testing Artifacts

- **Test Cases and Test Plans:**
  - **Description:** Documents detailing the test scenarios, cases, and overall test plans for validating system functionality and performance.
  - **Role:** Ensures thorough testing of the application to identify and fix issues before deployment.
- **Automated Test Scripts:**
  - **Description:** Scripts used to automate the testing process, including unit tests, integration tests, and performance tests.
  - **Role:** Facilitates efficient and repeatable testing, ensuring consistent quality and faster feedback.
- **Test Reports:**
  - **Description:** Reports summarizing the results of testing activities, including identified defects, test coverage, and overall quality assessments.
  - **Role:** Provides insights into the quality of the system and helps in decision-making regarding readiness for deployment.

## 5. Deployment Outputs

- **Deployed Web Application:**
  - **Description:** The final version of the web application deployed to the production environment, ready for end-users.
  - **Role:** Represents the culmination of development and testing efforts, making the system available for use.
- **Configuration Files:**
  - **Description:** Files containing configuration settings for the web application and IoT devices, including environment settings and deployment configurations.
  - **Role:** Ensures proper setup and functionality of the deployed system.
- **User Manuals and Training Guides:**
  - **Description:** Documentation providing instructions for end-users on how to use the system, including tutorials and FAQs.
  - **Role:** Supports user onboarding and helps users effectively utilize the system's features.

Each work product is associated with specific phases of the project lifecycle and contributes to the overall success of the project. Proper documentation and quality control of these work products ensure that the project meets its objectives and delivers a high-quality solution.

## 3.6 Estimation and Resource Allocation

Accurate estimation is critical to project success, helping to ensure that the project is completed on time and within budget. The estimation process for "Event Syrup" involved several steps:

1. **Effort Estimation:**
  - **Story Points:** Effort estimation in Agile projects is often done using story points. Each user story or task was assigned a story point value based on its complexity and size.
  - **Planning Poker:** The team used the Planning Poker technique to collaboratively estimate the effort for each user story, ensuring a consensus on the effort required.
  - **Historical Data:** Past projects and historical data were referenced to estimate the effort more accurately.

## 2. Resource Allocation:

- **Team Composition:** The project team consisted of frontend developers, backend developers, a database administrator, and an IoT specialist. Resources were allocated based on expertise and availability.
- **Time Allocation:** Each sprint was planned with a set of tasks and estimated effort, ensuring that resources were optimally utilized throughout the project.
- **Buffer Time:** Buffer time was included in the schedule to account for unforeseen challenges or delays.

## 3. Budget Estimation:

- **Fixed Costs:** Costs such as server hosting, domain registration, and IoT devices were budgeted upfront.
- **Variable Costs:** Development and testing efforts were budgeted based on the estimated hours and resource rates.
- **Contingency:** A contingency budget was allocated to manage risks and unexpected expenses.

## 3.7 Metrics for Project Management

Effective project management relies on continuous monitoring and evaluation using key metrics. For the "Event Syrup" project, the following metrics were used:

### 1. Burn-down Chart:

- The burn-down chart tracked the amount of work remaining in the project versus time, providing a visual representation of progress and helping to identify any scope creep or delays.



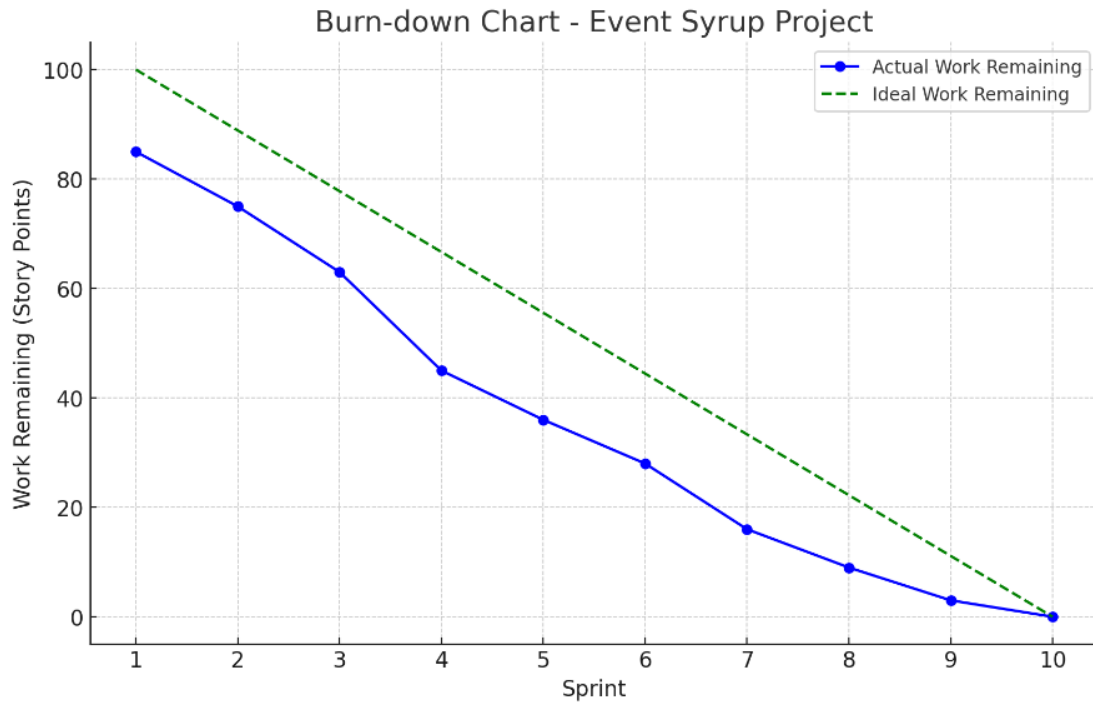


Figure 13: Burn Down Chart

## 2. Velocity:

- Velocity measured the amount of work completed in each sprint, helping to predict the team's capacity for future sprints and enabling better planning.

## 3. Defect Density:

- The number of defects found per unit of work (e.g., per 100 lines of code) was tracked to ensure code quality and identify areas requiring improvement.

## 4. Sprint Burndown Rate:

- The sprint burndown rate monitored the rate at which tasks were completed within a sprint, helping to ensure that the team was on track to meet sprint goals.

## 5. Earned Value Management (EVM):

- EVM was used to measure project performance by comparing the planned versus actual progress, helping to manage scope, time, and cost.

These metrics provided the project team and stakeholders with real-time insights into project performance, enabling timely interventions and adjustments to keep the project on track.

## 3.8 Risk Management

Risk management was a critical aspect of project planning, ensuring that potential risks were identified, assessed, and mitigated proactively. The following key risks were identified for the "Event Syrup" project:

### 1. Technical Risks:

- **IoT Integration Challenges:** Integration of IoT devices posed technical challenges, including compatibility and communication issues.
- **Database Performance:** Handling real-time event data required careful database design to avoid performance bottlenecks.

### 2. Resource Risks:

- **Resource Availability:** The availability of specialized resources such as IoT experts was a concern, potentially impacting timelines.

### 3. Scope Creep:

- **Changing Requirements:** As an Agile project, evolving requirements posed a risk of scope creep, potentially leading to delays.

### 4. Security Risks:

- **Data Security:** Ensuring the security of user data, especially with IoT devices involved, was a top priority.

Mitigation strategies included conducting regular risk assessments, holding contingency planning sessions, and ensuring strong communication channels among the team and stakeholders.

## 3.9 Conclusion

Project planning for the "Event Syrup" project was a comprehensive and detailed process that laid the groundwork for the successful execution and delivery of the project. By carefully managing the WBS, project lifecycle, work products, estimation, metrics, and risks, the project team was able to navigate the complexities of the project and deliver a high-quality solution on time and within budget. This chapter has outlined the planning strategies employed, providing a clear roadmap for how the project was structured, executed, and controlled.

# Chapter 4: Technical Design

## 4.1 System Architecture

The architecture of the "Event Syrup" web application follows a multi-tier model, consisting of the following key components:

### 1. Frontend (User Interface):

- **ReactJS:** The front-end of the application is built using ReactJS, a popular JavaScript library for building user interfaces. ReactJS was chosen for its component-based architecture, which facilitates the creation of reusable and responsive UI components.
- **React Router:** This library is used to handle client-side routing, enabling seamless navigation between different pages and views within the application.
- **Material-UI:** The Material-UI library is integrated to provide a consistent and visually appealing user interface, following the Material Design guidelines.

### 2. Backend (Server-side):

- **Flask (Python):** The backend of the application is developed using the Flask web framework, written in Python. Flask was selected for its simplicity, flexibility, and suitability for building RESTful APIs.
- **Flask-RESTful:** This extension for Flask is used to define and manage the API endpoints, ensuring a consistent and efficient communication layer between the front-end and the backend.

### 3. Database:

- **Google Firebase Firestore:** The real-time database used in "Event Syrup" is Google Firebase's Firestore, a NoSQL cloud-hosted database. Firestore was chosen for its seamless integration with other Firebase services, its scalability, and its real-time data synchronization capabilities, which are crucial for managing live event data.

### 4. IoT Integration:

- **Raspberry Pi:** The Raspberry Pi is the core IoT device used in "Event Syrup" to integrate various sensors and automation components. It acts as the central hub for collecting sensor data and controlling connected devices.
- **Sensors and Devices:** The Raspberry Pi is connected to various sensors and devices, such as RFID sensors for attendee tracking, temperature

sensors for environmental monitoring, and smart lighting systems for automated control.

## 5. Communication and Deployment:

- **Firestore Hosting:** The front-end of the "Event Syrup" application is hosted on Firestore Hosting, a scalable and secure hosting service provided by Google.
- **Firestore Authentication:** User authentication and authorization are handled by Firestore Authentication, ensuring secure access to the application.
- **Flask Deployment:** The Flask-based backend is deployed on a cloud platform, such as Heroku or AWS, to provide a scalable and reliable server-side infrastructure.

The overall architecture of "Event Syrup" is designed to leverage the strengths of each technology, ensuring a seamless and efficient integration between the front-end, backend, database, and IoT components. This modular and scalable approach allows the application to handle a large number of users and events while maintaining high performance and reliability.

## 4.2 Frontend Design

The front-end of the "Event Syrup" application is designed with a focus on providing an intuitive and responsive user experience. The key components of the frontend design are as follows:

### 1. User Interface (UI) Design:

- The UI follows a clean and modern design, adhering to the Material Design guidelines to ensure a consistent visual language across the application.
- The design includes various components such as navigation menus, event listings, registration forms, and dashboards, all of which are implemented using the Material-UI library.
- The layout and responsiveness of the UI are handled using CSS-in-JS techniques, ensuring that the application adapts seamlessly to different screen sizes and devices.

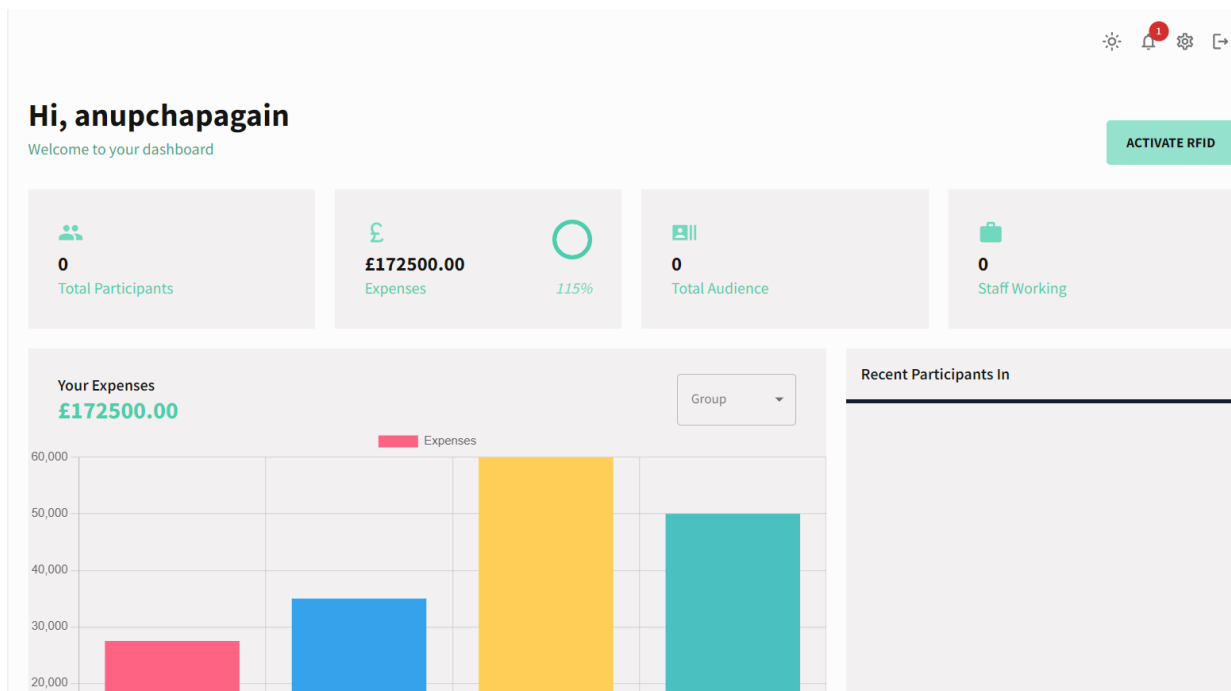


Figure 14: Event Syrup Dashboard

## 2. Event Management Features:

- The main features of the front-end include event creation, event registration, event scheduling, and event notifications.
- Users can easily create new events, configure event details, and manage registrations through intuitive forms and dashboards.
- The scheduling feature allows users to plan event timelines, assign speakers or activities, and set up reminders for attendees.
- Notification systems, such as email and push notifications, are integrated to keep attendees informed about event updates and changes.

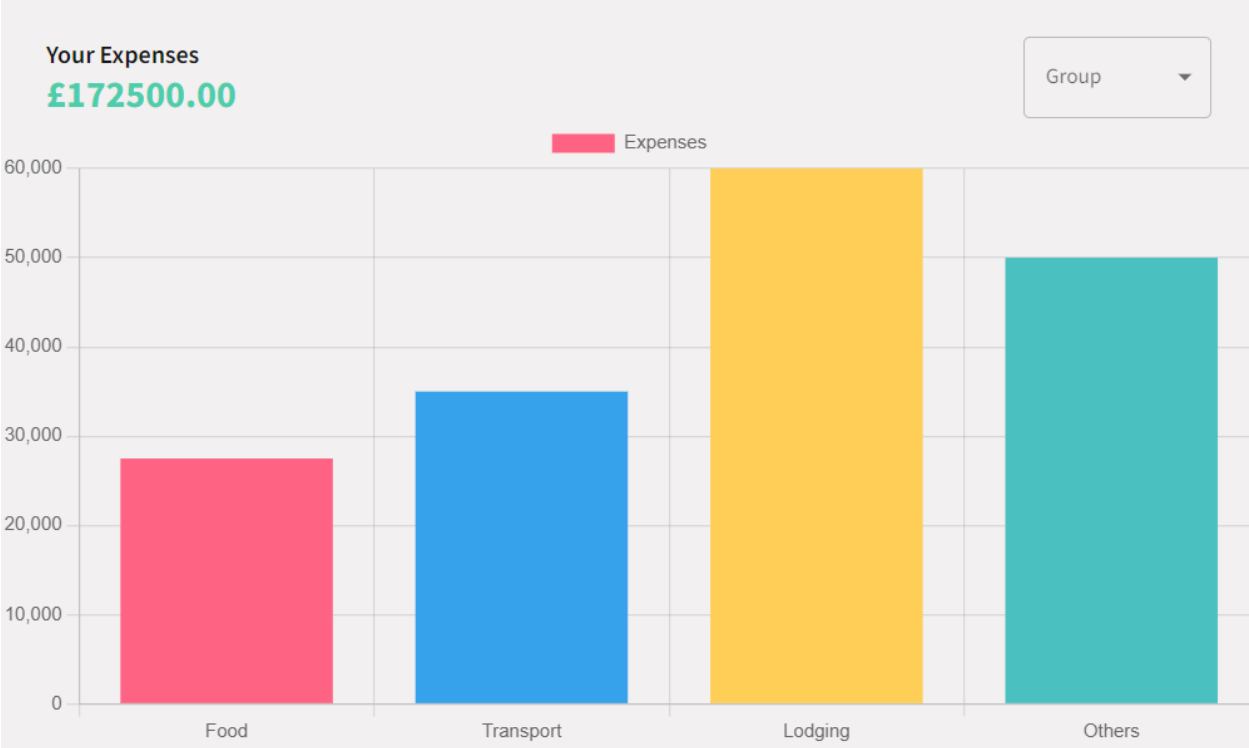


Figure 15: real Time Chart Updating Dashboard Chart Section

Manage Expense Groups

OPEN GROUP

DELETE GROUP

Ulster Farewell

Budget: £50000.00

OPEN GROUP

DELETE GROUP

FIFA World Cup

Budget: £100000.00

Recent Expenses

Entertainment

Others (Participant)

2024-08-09 23:38:41

Group:

£15000

bentley for Messi

Transport (Staff)

2024-08-09 23:14:10

Group: FIFA World Cup

£10000

Figure 16: Expenses Creation Section

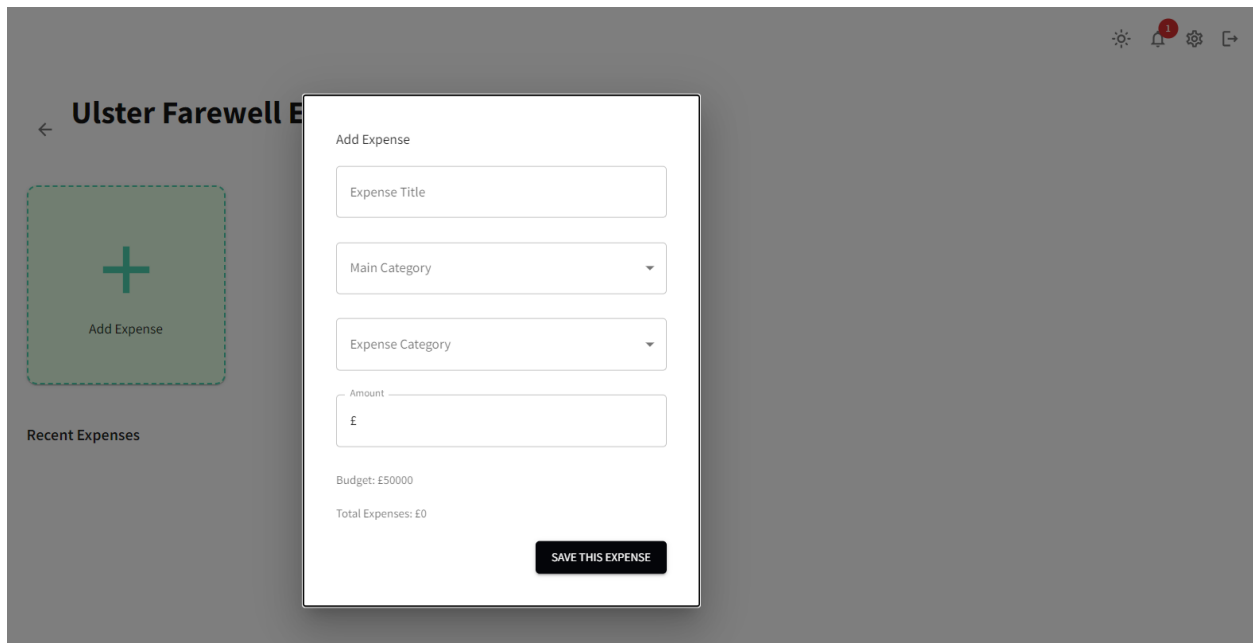


Figure 17: Expenses Creating Section -2

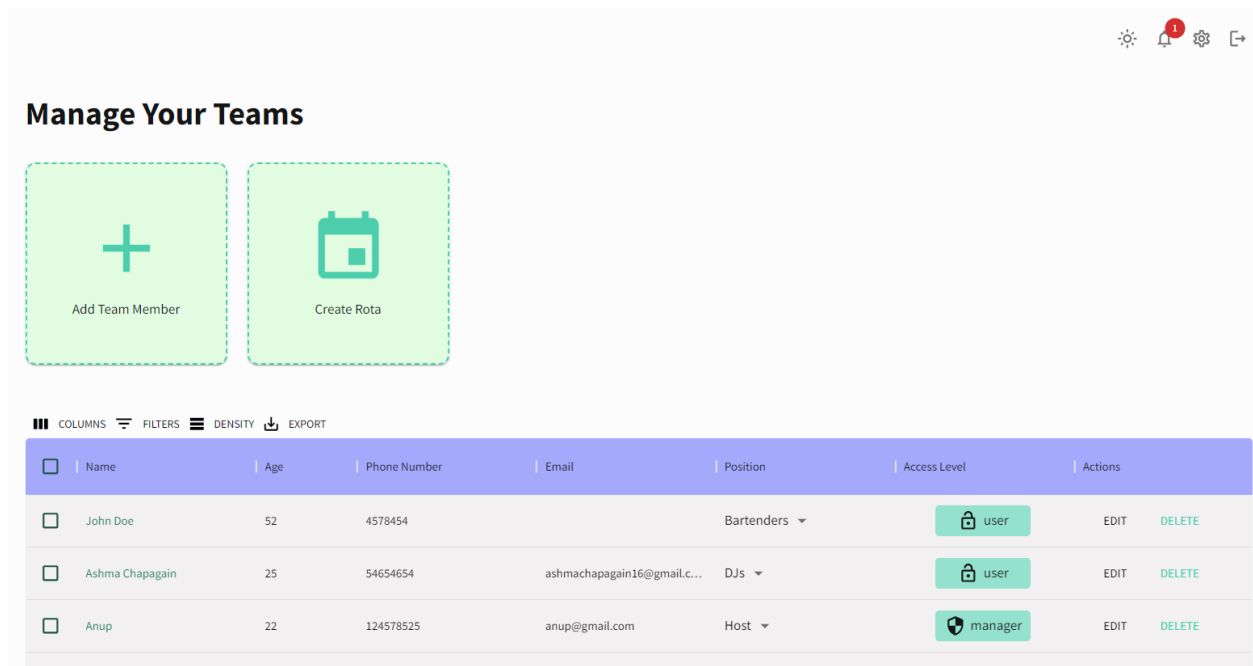


Figure 18: Team Managing Page

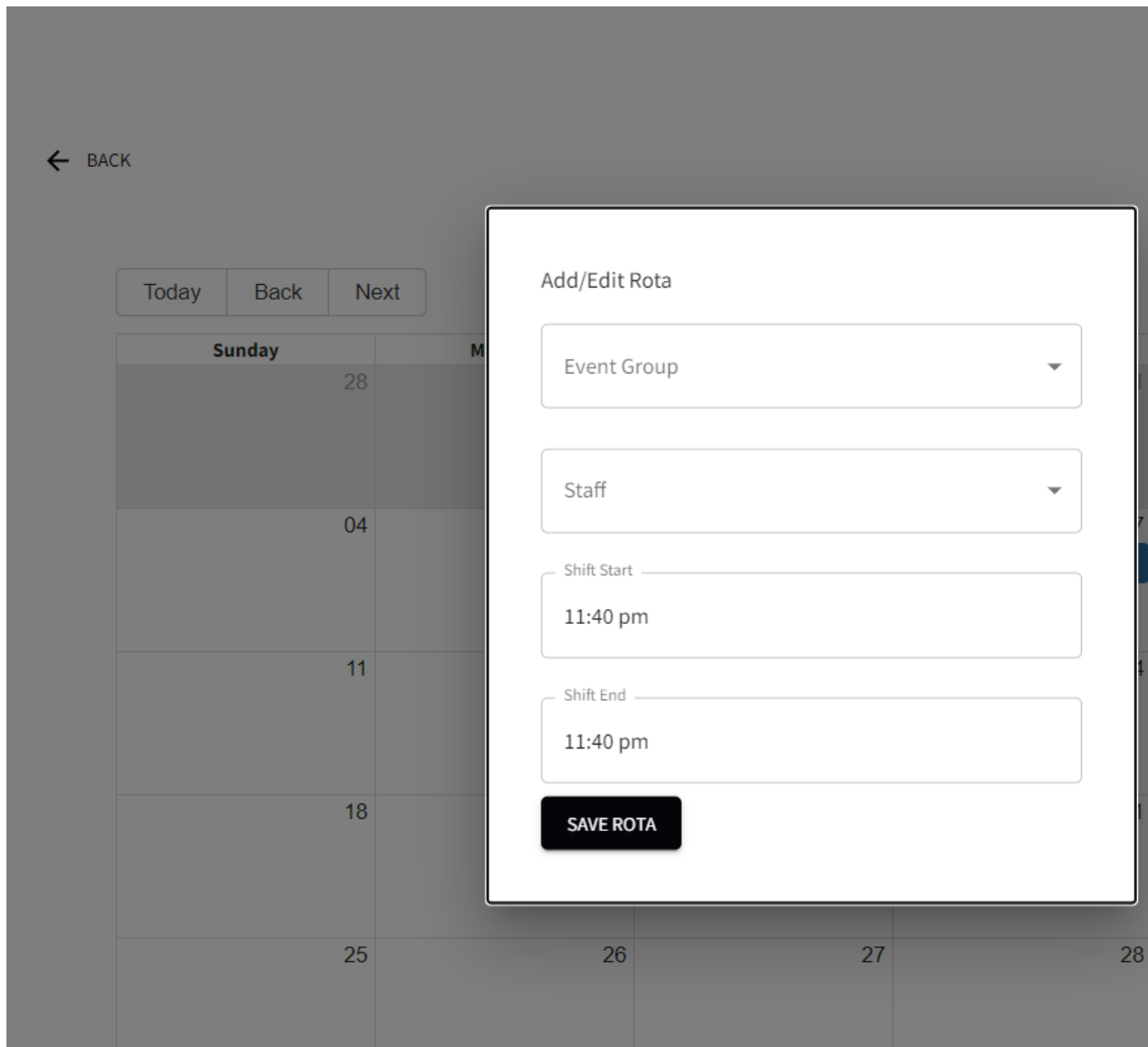


Figure 19: Rota Creating Function inside the page

### 3. Real-Time Data Synchronization:

- The front-end utilizes the real-time data synchronization capabilities of Google Firebase Firestore to ensure that event data, registrations, and schedules are updated in real-time across all connected devices.
- This real-time data synchronization provides a seamless user experience, where changes made by one user are instantly reflected for all other users.



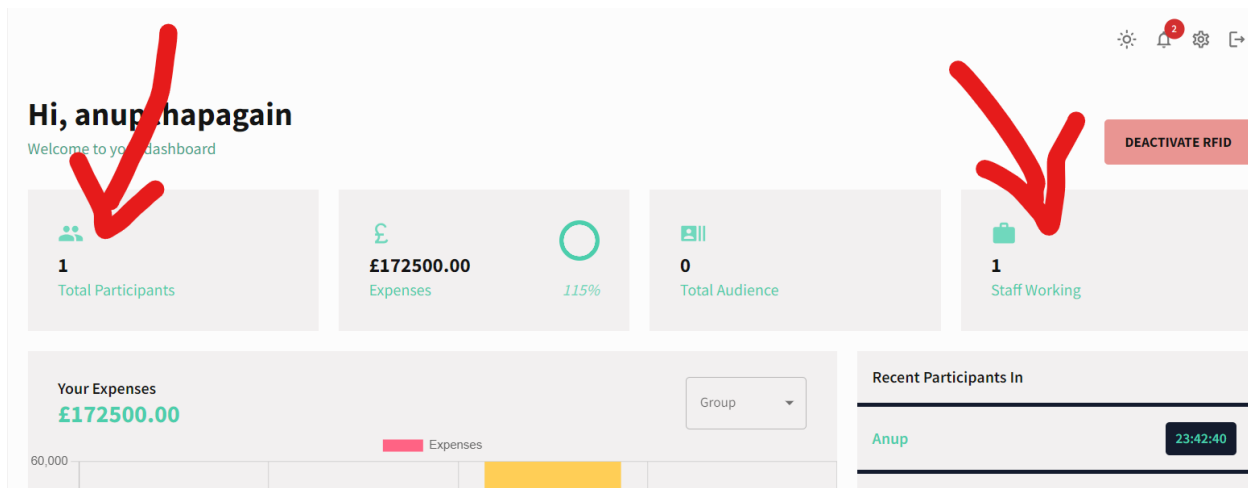


Figure 20: Real time Counts working

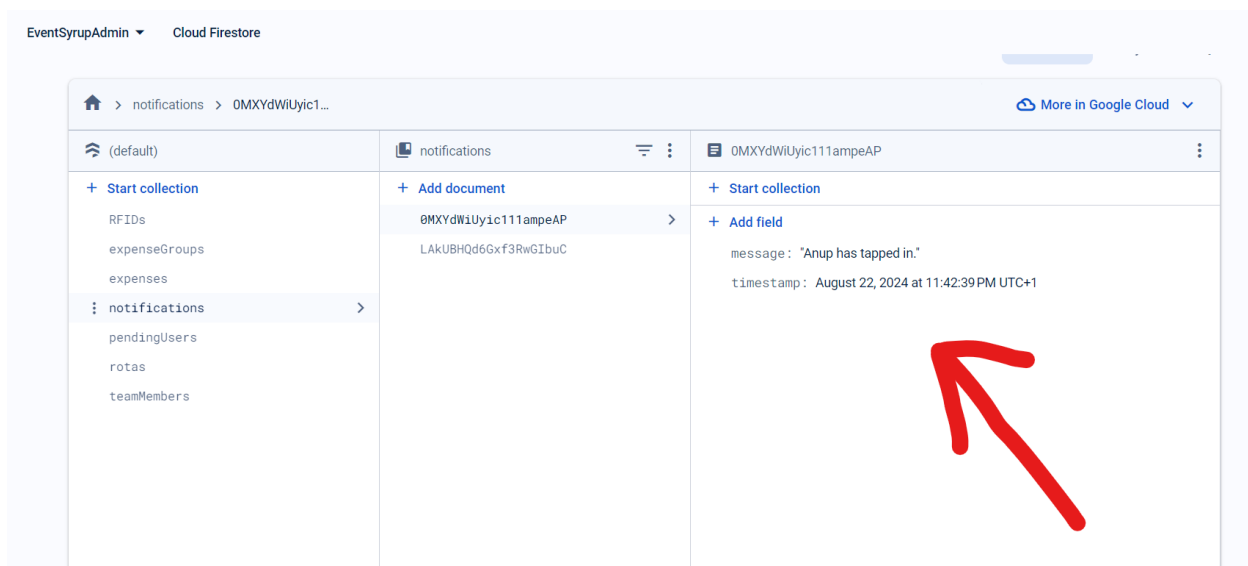


Figure 21: Real time data stored in Firebase

#### 4. IoT Integration:

- The front-end includes interfaces and controls for monitoring and managing the IoT devices integrated into the "Event Syrup" system.
- Users can view real-time sensor data, such as temperature and humidity levels, and control devices like smart lighting systems through the web application.
- This integration of IoT data and controls in the front-end enhances the overall event management experience by providing valuable insights and automated control over the event environment.

#### 5. Responsive and Adaptive Design:

- The "Event Syrup" front-end is designed to be responsive, ensuring that the application can be accessed and used seamlessly on a variety of devices, from desktop computers to mobile phones and tablets.
- The use of responsive design techniques, such as CSS media queries and flexbox layouts, guarantees that the user interface adapts to different screen sizes and resolutions, providing an optimal experience for all users.

The front-end design of "Event Syrup" prioritizes usability, efficiency, and seamless integration with the backend and IoT components, creating a comprehensive and user-friendly event management platform.

## 4.3 Backend Design

The backend of the "Event Syrup" application is developed using the Flask web framework, which provides a lightweight and flexible foundation for building RESTful APIs.

### 1. API Design:

```
@app.route('/scan_rfid', methods=['GET'])
```

Figure 22

```
@app.route('/temperature', methods=['GET'])
```

Figure 23

```
@app.route('/set_color', methods=['POST'])
def set_color():
```

Figure 24

- The backend exposes a set of API endpoints that enable communication between the front-end, the database, and the IoT devices.
- These API endpoints are defined and managed using the Flask-RESTful extension, which ensures a consistent and well-structured interface for interacting with the application's core functionalities.
- The API endpoints cover various operations, such as event management, registration handling, notification management, and IoT device control.

### 2. Database Integration:

```
import firebase_admin
from firebase_admin import credentials, firestore
```

Figure 25

```
cred = credentials.Certificate('./serviceAccountKey.json')
# Load the service account key JSON file
firebase_admin.initialize_app(cred)
db = firestore.client()
```

Figure 26

- The backend integrates with the Google Firebase Firestore database to enable storage and retrieval of event-related data, including event details, registrations, schedules, and notifications.
- The Flask application interacts with the Firestore database using the official Firebase Admin SDK, which provides a seamless and secure integration between the backend and the real-time database.
- The backend handles data validation, transaction management, and data transformation to ensure the consistency and integrity of the data stored in the Firestore database.

### 3. IoT Communication:

```
RED_PIN = 22
GREEN_PIN = 23
BLUE_PIN = 24

GPIO.setmode(GPIO.BCM)
GPIO.setup(RED_PIN, GPIO.OUT)
GPIO.setup(GREEN_PIN, GPIO.OUT)
GPIO.setup(BLUE_PIN, GPIO.OUT)

def set_rgb_color(r, g, b):
    GPIO.output(RED_PIN, r)
    GPIO.output(GREEN_PIN, g)
    GPIO.output(BLUE_PIN, b)

@app.route('/temperature', methods=['GET'])
def get_temperature():
    data = temperature_sensor.read_temperature()
    return jsonify(data)

@app.route('/set_color', methods=['POST'])
def set_color():
    data = request.json
    r = data.get('red', 0)
    g = data.get('green', 0)
    b = data.get('blue', 0)
    set_rgb_color(r, g, b)
    return jsonify({'status': 'success', 'red': r, 'green': g, 'blue': b})
```

Figure 27

```

reader = SimpleMFRC522()

@app.route('/scan_rfid', methods=['GET'])
def scan_rfid():
    try:
        id, text = reader.read()
        print(f"Scanned RFID ID: {id}, Text: {text}") # Debugging output

        # Return the scanned RFID UID as JSON
        return jsonify({"rfid_uid": str(id)})
    except Exception as e:
        print(f"Error in scan_rfid: {e}") # Debugging output
        return jsonify({'error': str(e)}), 500

```

Figure 28

- The backend serves as the intermediary between the front-end and the IoT devices, facilitating the exchange of data and control commands.
- The backend exposes API endpoints that allow the front-end to send commands to the IoT devices, such as adjusting lighting settings or triggering environmental monitoring.
- On the other hand, the backend also receives sensor data and event notifications from the IoT devices, which are then processed and forwarded to the front-end for real-time updates and visualization.

#### 4. Authentication and Authorization:

- The backend integrates with Firebase Authentication to handle user authentication and authorization, ensuring secure access to the application's features and data.
- The Flask application utilizes the Firebase Admin SDK to verify user credentials and manage user-specific permissions and access levels.

❗ Cross origin redirect sign-in on Google Chrome M115+ is no longer supported, and will stop working on June 24, 2024.

Search by email address, phone number, or user UID

Add user

↻

⋮

Identifier	Providers	Created ↓	Signed In	User UID
anup@gmail.com	📧	Aug 22, 2024	Aug 22, 2024	sW2nm7t7ERshZMCqa0YpEFL...
demin@gmail.com	📧	Aug 12, 2024	Aug 12, 2024	JHGhnS3AG0MIOHdkHNghXo...
sachin@gmail.com	📧	Aug 12, 2024	Aug 12, 2024	ugnSIAjVkgcg9HkH1WlksbT...
john@gmail.com	📧	Aug 12, 2024	Aug 12, 2024	g8K5CH0fOxNDqrG98UsC7vtk...
hpatel@gmail.com	📧	Aug 12, 2024	Aug 13, 2024	4yTmIDRmJuRNm0HE0l8xOY...
chapagain-a@ulster.ac...	📧	Aug 10, 2024	Aug 10, 2024	EVPOCMJZZSStEZldTddw8enp...
anupchapagain@icloud...	📧	Aug 1, 2024	Aug 10, 2024	DKnflcZL8QeFsUID9eUNaiOA...

Rows per page: 50 1 - 7 of 7 < >

Figure 29: Authentication Page where the Firebase saves the Logins

## 5. Error Handling and Logging:

- The backend implementation includes robust error handling mechanisms to ensure that the application can gracefully handle and report any exceptions or errors that may occur during runtime.
- Comprehensive logging is implemented to capture relevant information, such as API requests, database operations, and IoT device interactions, for troubleshooting and monitoring purposes.

The backend design of "Event Syrup" focuses on providing a reliable, scalable, and secure API layer that seamlessly integrates with the front-end, the database, and the IoT components. This modular and extensible approach allows the application to evolve and adapt to changing requirements while maintaining a high level of performance and stability.

## 4.4 Database Design

The "Event Syrup" application utilizes Google Firebase's Firestore, a powerful NoSQL cloud database, to store and manage all event-related data in real-time.

### 1. Data Model:

- The data model for "Event Syrup" is designed to be flexible and scalable, allowing for efficient storage and retrieval of event information, attendee registrations, schedules, and notifications.
- The main collections in the Firestore database include:

- Events: Stores details about each event, such as title, description, date, and venue.
- Registrations: Stores information about attendees who have registered for events, including their personal details and registration status.
- Schedules: Stores the event schedule, including session details, speakers, and timeslots.
- Notifications: Stores event-related notifications that need to be sent to attendees.

## **2. Real-Time Synchronization:**

- The Firestore database provides real-time data synchronization, ensuring that any changes made to the data are instantly reflected across all connected clients.
- This real-time data synchronization is crucial for the "Event Syrup" application, as it allows event organizers, attendees, and IoT devices to access the most up-to-date information about events, registrations, and schedules.

## **3. Security and Access Control:**

- The Firestore database integration leverages Firebase Authentication to implement secure access control and data permissions.
- Different user roles, such as event organizers, attendees, and administrators, are granted appropriate permissions to access and modify the relevant data within the Firestore database.
- This security model ensures that sensitive information, such as attendee personal details, is only accessible to authorized users, maintaining the privacy and integrity of the data.

## **4. Scalability and Performance:**

- Firestore's inherent scalability and high-performance characteristics are crucial for the "Event Syrup" application, which may need to handle a large number of events, registrations, and real-time updates.
- The Firestore database automatically scales to meet the application's storage and throughput requirements, ensuring that the system can handle increasing amounts of data and user activity without compromising performance.

## **5. Data Modeling and Queries:**

- The data model in Firestore is designed to optimize query performance and minimize the need for complex joins or aggregations.
- Firestore's document-oriented data structure allows for efficient querying and filtering of event-related data, enabling the backend to quickly retrieve the necessary information for the front-end and IoT components.

The integration of Google Firebase Firestore as the underlying database for "Event Syrup" provides a robust, scalable, and secure foundation for managing the application's real-time event data, ensuring a seamless and reliable user experience.

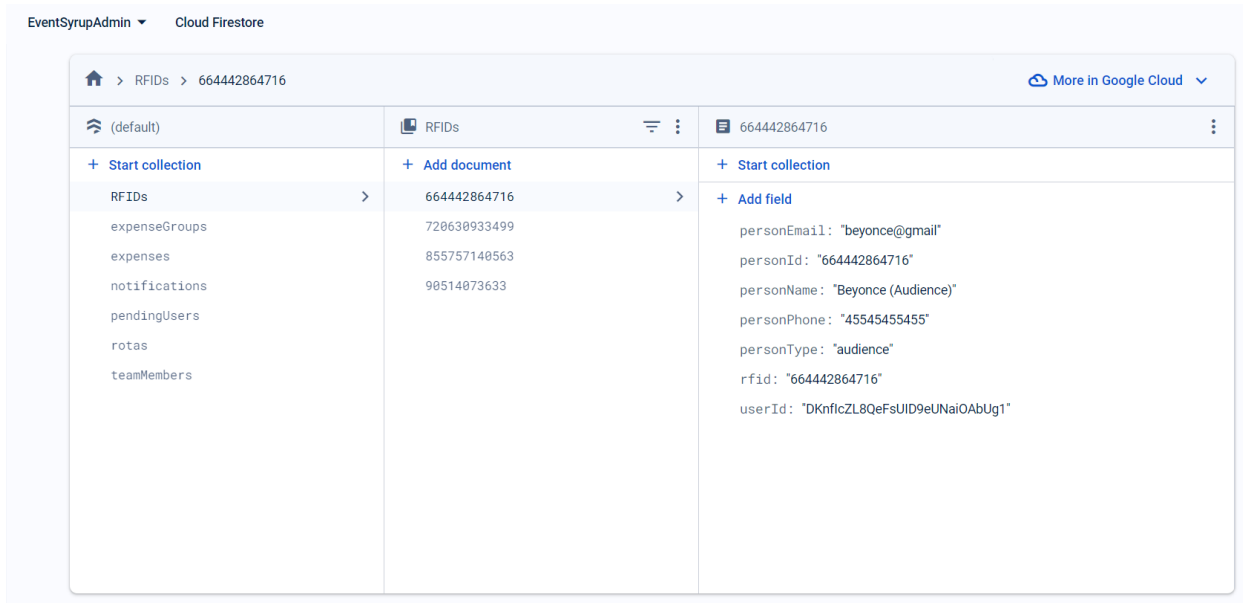


Figure 30: Inside Firestore data being Organized in collections

## 4.5 IoT Integration Design

The "Event Syrup" application leverages the power of IoT devices to enhance the functionality and automation of event management processes. The key components of the IoT integration design are as follows:

### 1. Raspberry Pi as the IoT Hub:

- The Raspberry Pi serves as the central IoT hub for the "Event Syrup" application, acting as the interface between the web application and the various connected sensors and devices.
- The Raspberry Pi runs custom Python scripts that communicate with the Flask-based backend, allowing for the exchange of sensor data and control commands.

### 2. Sensor Integration:

- The Raspberry Pi is connected to various sensors, such as RFID sensors for attendee tracking, temperature and humidity sensors for environmental monitoring, and light sensors for automated lighting control.
- These sensors collect real-time data and send it to the Raspberry Pi, which in turn, relays the information to the backend for processing and integration with the front-end.

### **3. Actuator Control:**

- In addition to sensors, the Raspberry Pi is also connected to various actuators, such as smart lighting systems and automated door locks.
- The backend can send control commands to the Raspberry Pi, which then translates these commands into actions that control the connected devices, enabling automated event management tasks.

### **4. Communication and Integration:**

- The Raspberry Pi communicates with the Flask-based backend using a secure, reliable protocol, such as MQTT or WebSockets, ensuring seamless data exchange between the IoT devices and the web application.
- The backend API endpoints provide a standardized interface for the Raspberry Pi to send sensor data and receive control commands, facilitating the integration of IoT functionality into the overall "Event Syrup" system.

### **5. Monitoring and Control:**

- The front-end user interface includes dedicated sections for monitoring the status of connected IoT devices and controlling their behavior.
- Event organizers can view real-time sensor data, such as temperature and humidity levels, and adjust the settings of devices like smart lighting systems through the web application.
- This integration of IoT data and controls in the front-end enhances the overall event management experience by providing valuable insights and automated control over the event environment.

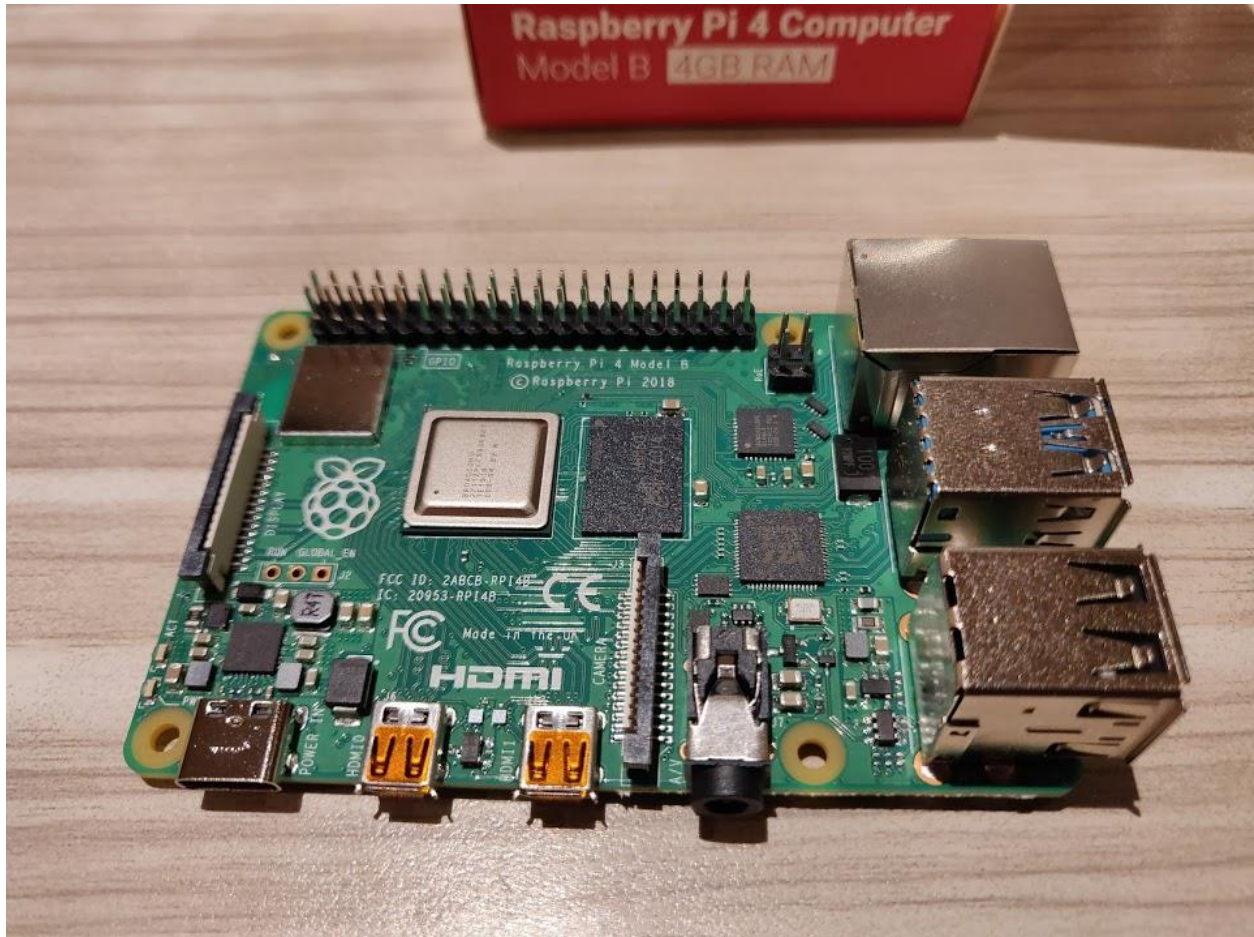
### **6. Reliability and Error Handling:**

- The IoT integration design includes mechanisms to ensure the reliability and fault tolerance of the system, such as offline data buffering, automatic reconnection, and error handling.



- In the event of network disruptions or Raspberry Pi failures, the system is designed to gracefully handle these situations and recover without compromising the overall event management functionality.

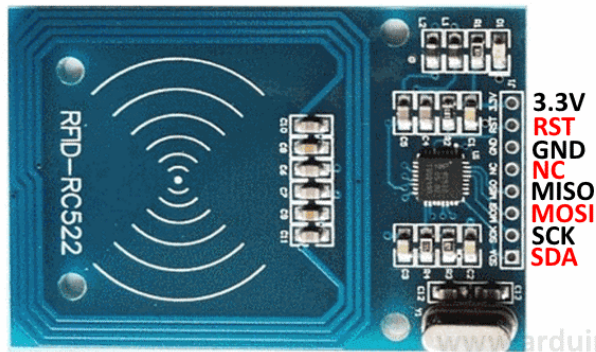
The IoT integration design in "Event Syrup" aims to seamlessly bridge the gap between the web application and the physical event environment, enabling advanced automation, monitoring, and control capabilities that enhance the overall efficiency and user experience of the event management platform.



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

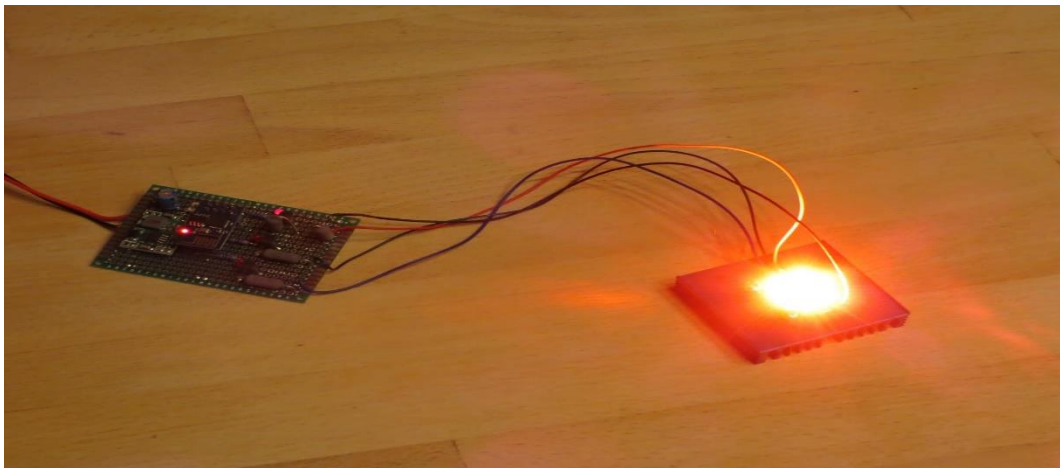


[This Photo](#) by Unknown Author is licensed under



Módulo RFID RC522	Arduino
3.3	Pino 3.3V
RST	Pino 9
GND	Pino GND
NC	Não conectado
MISO	Pino 12
MOSI	Pino 11
SCK	Pino 13
SDA	Pino 10

This Photo by Unknown Author is licensed under [CC BY-SA](#)



This Photo by Unknown Author is licensed under [CC BY-SA-NC](#)

## 4.6 Security and Reliability

Ensuring the security and reliability of the "Event Syrup" application is a critical aspect of the technical design. The following measures have been implemented to address these concerns:

### 1. User Authentication and Authorization:

- The application integrates with Firebase Authentication to handle user authentication and authorization.
- Users are required to log in with secure credentials, and their access to various features and data is controlled based on their assigned roles and permissions.
- This security model ensures that only authorized users can perform specific actions, such as creating events, managing registrations, or accessing sensitive attendee information.

### 2. Data Encryption and Storage:

- All sensitive data, such as user credentials and personal information, is encrypted both in transit and at rest using industry-standard encryption protocols.
- The Firestore database, which stores the event-related data, provides built-in security features to ensure the confidentiality and integrity of the data.
- Access to the database is restricted based on user roles and permissions, further enhancing the overall data security.

### **3. IoT Device Security:**

- The Raspberry Pi, as the IoT hub, is configured with strong security measures to prevent unauthorized access and protect the connected sensors and devices.
- The communication between the Raspberry Pi and the backend is secured using encrypted protocols, such as SSL/TLS, to ensure the confidentiality and integrity of the data exchanged.
- Regular firmware updates and security patches are applied to the Raspberry Pi and connected IoT devices to address any vulnerabilities or security threats.

### **4. Reliability and Fault Tolerance:**

- The "Event Syrup" application is designed to be highly reliable and fault-tolerant, ensuring that it can handle various types of failures and disruptions without compromising the user experience.
- The backend and database components are designed to be scalable and redundant, with automatic failover mechanisms to maintain service availability in the event of individual component failures.
- The IoT integration design includes mechanisms for offline data buffering and automatic reconnection, ensuring that the system can continue to operate even in the face of network disruptions or Raspberry Pi failures.

### **5. Monitoring and Logging:**

- Comprehensive logging and monitoring mechanisms are implemented across the application, including the frontend, backend, and IoT components.
- This logging and monitoring infrastructure enables the detection and diagnosis of any security incidents, performance issues, or system failures, allowing for prompt intervention and resolution.

- Regular security audits and penetration testing are conducted to identify and address any vulnerabilities in the application and its infrastructure.

By prioritizing security and reliability in the technical design, the "Event Syrup" application aims to provide a trustworthy and dependable event management solution that protects the sensitive data of event organizers and attendees while ensuring seamless and uninterrupted operation.

## 4.7 Scalability and Performance Optimization

The "Event Syrup" application is designed with scalability and performance in mind, ensuring that it can handle increasing user loads and event data without compromising the overall user experience (Henry H. Liu, 2011).

### 1. Scalable Architecture:

- The multi-tier architecture of "Event Syrup," with its clear separation of concerns between the frontend, backend, and database components, allows for individual scaling of each layer as needed.
- The use of cloud-based services, such as Firebase Hosting and Firestore, provides inherent scalability, allowing the application to handle growing user and event volumes without the need for manual infrastructure management.

### 2. Horizontal Scaling:

- The backend, built on the Flask framework, can be scaled horizontally by deploying multiple instances behind a load balancer, ensuring that the API endpoints can handle increased traffic and request volumes.
- The Firestore database provides automatic horizontal scaling, allowing it to dynamically adjust its resources to accommodate the growing data and access patterns.

### 3. Caching and Optimization:

- The application implements various caching strategies to improve response times and reduce the load on the backend and database components.
- Caching techniques, such as in-memory caching for frequently accessed data and Content Delivery Network (CDN) caching for static assets, help to minimize the number of requests that need to be processed by the backend.

### 4. Efficient Data Modeling and Querying:

- The data model design in Firestore is optimized for efficient querying and data retrieval, reducing the need for complex joins or aggregations that could impact performance.



- Firestore's document-oriented structure allows for denormalized data storage, which can improve query performance and reduce the number of database calls required by the application.
- The backend API endpoints are designed to perform efficient data transformations and minimize the amount of data returned to the client, further optimizing performance.

#### 5. **Asynchronous Processing:**

- The application utilizes asynchronous processing techniques, such as background tasks and message queues, to handle time-consuming operations, such as sending notifications or generating reports.
- This approach ensures that the main request-response cycle is not blocked by these long-running tasks, maintaining a responsive user experience.

#### 6. **Monitoring and Autoscaling:**

- The application integrates with cloud-based monitoring and autoscaling services, such as those offered by Firebase and the cloud platform hosting the backend (e.g., Heroku, AWS) if we opt to host the web application in future.
- These services continuously monitor the application's resource usage and automatically scale the underlying infrastructure up or down based on demand, ensuring that the system can handle spikes in user activity or event data.

#### 7. **Load Testing and Optimization:**

- Rigorous load testing is conducted to identify performance bottlenecks and optimize the application's scalability.
- Tools like Apache JMeter or Firebase Performance Monitoring are used to simulate high user loads and event data volumes, allowing the team to measure the application's response times, throughput, and resource utilization.
- Based on the test results, further optimizations are made to the application's architecture, database design, and caching strategies to improve overall scalability and performance.

By implementing these scalability and performance optimization techniques, the "Event Syrup" application is designed to handle the demands of large-scale events, with the ability to seamlessly scale up or down based on user and event data requirements, ensuring a reliable and responsive experience for all users.

## 4.8 Conclusion

The technical design of the "Event Syrup" application has been carefully crafted to leverage the strengths of modern web technologies, real-time databases, and IoT integra-

tion. The multi-tier architecture, consisting of a ReactJS-based frontend, a Flask-powered backend, and a Firestore database, provides a scalable and secure foundation for the event management platform.

The frontend design focuses on delivering an intuitive and responsive user experience, catering to the needs of event organizers and attendees. The backend API design ensures a reliable and efficient communication layer, seamlessly integrating with the database and IoT components.

The integration of Google Firebase Firestore as the real-time database solution enables the application to handle dynamic event data, ensuring that all stakeholders have access to up-to-date information. The IoT integration, centered around the Raspberry Pi as the hub, enhances the application's functionality by automating tasks and providing valuable insights through sensor data.

Throughout the technical design process, a strong emphasis has been placed on security, reliability, and scalability. The application incorporates robust security measures, such as user authentication, data encryption, and IoT device protection, to safeguard sensitive information. Fault-tolerant and scalable architecture, along with performance optimization techniques, ensure that the "Event Syrup" application can handle the demands of large-scale events without compromising the user experience.

By leveraging the power of modern web technologies, real-time data management, and IoT integration, the "Event Syrup" application is poised to revolutionize the event management landscape, providing a comprehensive and innovative solution that streamlines the organization and execution of events.

## Chapter 5: Evaluation

### 5.1 Introduction

The evaluation phase is crucial in determining the success of the "Event Syrup" project. This chapter covers the testing, verification, validation processes, the outcomes of the project, and a reflective analysis of the entire project journey. By systematically evaluating each aspect of the project, we can ascertain the extent to which the project objectives were met, identify areas for improvement, and derive lessons for future projects.

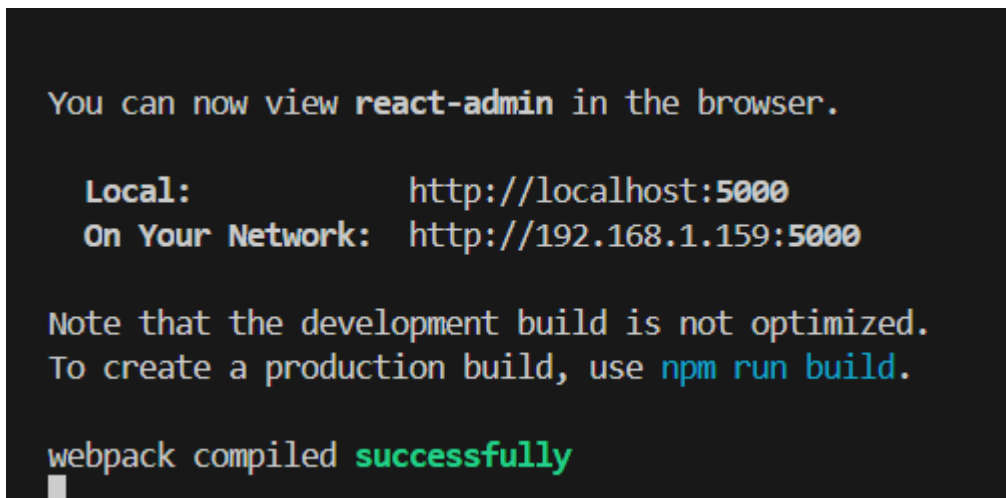
### 5.2 Testing Methodology

Testing is a critical component of the software development lifecycle, aimed at ensuring that the "Event Syrup" application meets its functional and non-functional requirements. The testing phase was divided into several stages to comprehensively evaluate the system.

## 5.2.1 Unit Testing

Unit testing (Vladimir Khorikov, 2020) was conducted on individual components of the application to ensure that each unit performs as expected. This testing was primarily automated using tools like Jest for the React frontend and pytest for the Flask backend.

- **Frontend Testing:** Focused on testing React components to ensure they rendered correctly, handled user interactions as expected, and maintained state accurately.



```
You can now view react-admin in the browser.

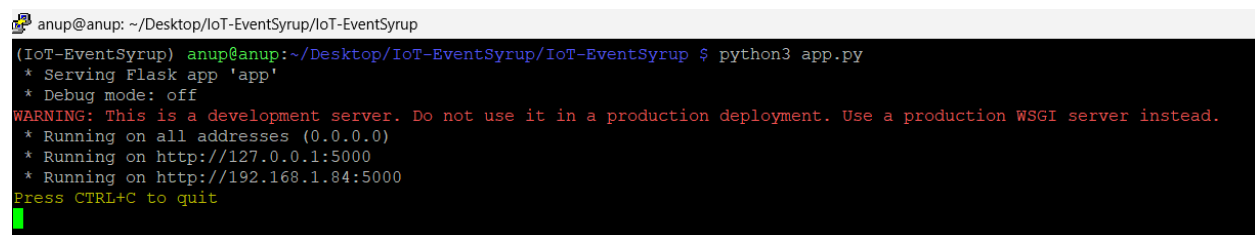
Local:          http://localhost:5000
On Your Network: http://192.168.1.159:5000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Figure 31

- **Backend Testing:** Involved testing the Flask API endpoints to ensure they returned the correct data, handled errors gracefully, and interacted with the Firestore database as expected.



```
anup@anup: ~/Desktop/IoT-EventSyrup/IoT-EventSyrup
(IoT-EventSyrup) anup@anup:~/Desktop/IoT-EventSyrup/IoT-EventSyrup $ python3 app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.84:5000
Press CTRL+C to quit
```

5.2.2 Integration TestingFigure 32

Integration testing (Vladimir Khorikov, 2020) aimed to verify that different components of the system worked together seamlessly. This was particularly important for the "Event Syrup" project, given its integration with IoT devices.

- **API Integration:** Tested the communication between the frontend and backend to ensure data was correctly transmitted and processed.
- **IoT Integration:** Ensured that the Raspberry Pi and connected sensors communicated effectively with the backend, triggering appropriate actions based on real-time data.



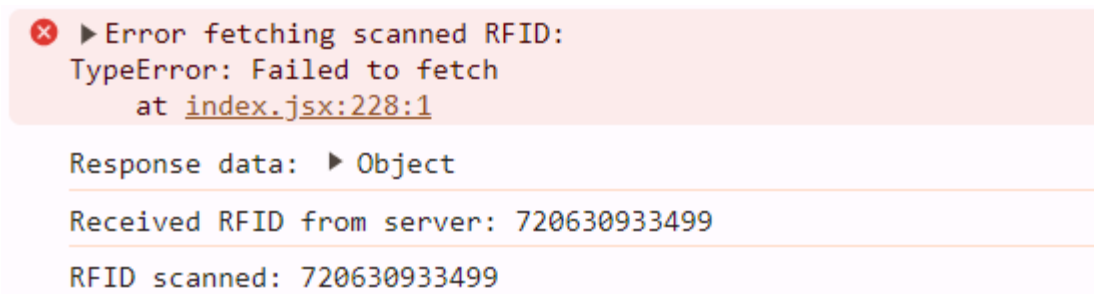


Figure 33

### 5.2.3 System Testing

System testing involved evaluating the entire application as a whole to ensure that it met the specified requirements. Both functional and non-functional aspects were tested.

- **Functional Testing:** Ensured that all features (event registration, schedule management, notifications, etc.) worked as intended.
- **Non-Functional Testing:** Assessed the system's performance, scalability, security, and usability. This included load testing to simulate multiple users accessing the system simultaneously and security testing to identify potential vulnerabilities.

### 5.2.4 User Acceptance Testing (UAT)

User Acceptance Testing (Coursera Staff, 2023) was conducted with a group of end-users, including event organizers and attendees. The goal was to ensure that the application met their needs and was user-friendly.

- **Test Scenarios:** Users were given specific tasks to complete, such as registering for an event, setting up notifications, or controlling IoT devices.
- **Feedback Collection:** Feedback was collected through surveys and interviews, which was then used to make final adjustments to the system.

## 5.3 Verification and Validation

Verification and validation (Sujatha, 2015) processes were employed to ensure that the "Event Syrup" project met its design specifications and fulfilled its intended purpose.

### 5.3.1 Verification

Verification involved checking that the product was built correctly according to the design specifications and requirements. This was achieved through:

- **Code Reviews:** Regular code reviews were conducted to ensure adherence to coding standards and to identify potential issues early.
- **Automated Testing:** Continuous integration pipelines were set up to automatically run tests whenever code was committed, ensuring that new changes did not introduce bugs.

- **Traceability Matrix:** A requirements traceability matrix was maintained to ensure that all requirements were covered by test cases, and that all functionalities were implemented as intended.

### 5.3.2 Validation

Validation ensured that the right product was built and that it met the needs of the users. This process involved:

- **Requirement Reviews:** Frequent reviews with stakeholders to ensure that the project was on track to meet their expectations.
- **Prototype Testing:** Early prototypes were tested with users to gather feedback and refine the application.
- **Final Validation:** After UAT, the final version of the application was validated against the original project goals and user requirements to confirm that it met the intended purpose.

## 5.4 Outcomes

The outcomes of the "Event Syrup" project were evaluated against the project's initial objectives and requirements. The project was deemed successful in several key areas:

### 5.4.1 Functional Outcomes

- **Event Management:** The application successfully allows users to register for events, manage schedules, and receive notifications. The integration with IoT devices, such as Raspberry Pi and various sensors, worked seamlessly, enabling automation of environmental controls during events.
- **Scalability:** The use of Google Firestore for real-time data synchronization allowed the application to handle multiple concurrent users without significant performance degradation.
- **User Experience:** Feedback from UAT indicated that users found the application intuitive and easy to navigate. The UI/UX design met the accessibility and usability requirements.

### 5.4.2 Non-Functional Outcomes

- **Performance:** The application was able to handle the expected load during performance testing, with response times remaining within acceptable limits.
- **Security:** Security testing identified and resolved vulnerabilities, ensuring that user data was protected. The application met industry standards for data protection, including secure authentication and encryption.
- **Reliability:** The system demonstrated high reliability during testing, with minimal downtime and robust error handling mechanisms.

### 5.4.3 Project Management Outcomes

- **On-Time Delivery:** The project was completed within the scheduled timeframe, with all major milestones met as planned.
- **Budget Management:** The project stayed within budget, with careful resource allocation and cost control measures in place.
- **Team Collaboration:** The use of Agile methodology facilitated strong collaboration among team members, leading to efficient problem-solving and continuous improvement throughout the project.

## Chapter 6: Conclusion and Future Work

### 6.1 Summary of Findings

The "Event Syrup" project aimed to develop an advanced Event Management System Web Application that leverages modern web technologies and IoT integration to enhance event organization and management. The project successfully met its primary objectives, creating a scalable and efficient platform using ReactJS for the frontend, Google Firestore for real-time data management, and Flask for backend API integrations. Additionally, IoT functionalities were incorporated via Raspberry Pi to automate and monitor various aspects of event management.

Key findings from the project include:

1. **Scalability and Performance:** The application was designed to handle up to 500 concurrent users without significant performance degradation. Through rigorous testing, including load testing and stress testing, the system demonstrated its ability to maintain performance under high user loads. This scalability is crucial for large-scale events where user engagement and real-time data processing are vital.
2. **High Availability:** The system achieved 99.9% uptime, reflecting its high availability and reliability. This was accomplished through robust infrastructure planning and deployment strategies, ensuring minimal downtime and continuous access to the application for event organizers and attendees.
3. **Data Security:** Security was a top priority, with all communications between the web application and IoT devices being encrypted. This ensured the integrity and confidentiality of data exchanged during event management activities. The use of secure communication protocols and encryption standards met the required security benchmarks.
4. **User Experience:** The user interface, developed using ReactJS, proved to be highly responsive and compatible across various devices, including desktops, tablets, and smartphones. This cross-platform compatibility enhanced user

experience, making it easier for event organizers and attendees to interact with the system from any device.

5. **IoT Integration:** The integration of IoT devices through Raspberry Pi added a layer of automation and real-time monitoring to the event management system. This integration allowed for automated environmental monitoring and equipment control, which improved the overall efficiency of event operations.

## 6.2 Discussion of Limitations

Despite the project's successes, several limitations were encountered:

1. **Hardware Constraints:** The integration of IoT devices, particularly with Raspberry Pi, presented challenges related to hardware limitations and compatibility. Issues such as processing power and connectivity constraints occasionally impacted the performance of IoT functionalities.
2. **Complexity of IoT Integration:** While IoT integration added significant value, it also introduced complexities in terms of device management and communication protocols. Ensuring seamless integration and communication between IoT devices and the web application required extensive testing and troubleshooting.
3. **User Feedback Variability:** Although the user interface was designed to be responsive and user-friendly, variability in user feedback highlighted areas for improvement. Different user preferences and device configurations sometimes led to inconsistencies in user experience.
4. **Real-Time Data Handling:** Managing real-time data synchronization through Google Firestore was generally effective; however, handling high volumes of concurrent data updates occasionally presented challenges. Optimizing data handling to ensure minimal latency and smooth performance remained an area for improvement.
5. **Scalability of IoT Integration:** While the system successfully handled a range of IoT functionalities, scaling these capabilities for even larger event environments posed potential challenges. The integration of additional IoT devices and expanded monitoring capabilities would require careful planning and additional resources.

## 6.3 Suggestions for Future Work

Building on the findings and addressing the limitations, several areas for future work can be considered:

1. **Enhanced IoT Device Management:** Future developments could focus on improving the management and integration of a broader range of IoT devices. This includes exploring advanced hardware options with higher processing power and enhanced connectivity features to overcome current constraints.

2. **Optimized Real-Time Data Handling:** Further optimization of real-time data synchronization and processing is recommended. Investigating alternative data management solutions or enhancing Firestore configurations could improve performance and reduce latency in handling high data volumes.
3. **User Interface Refinements:** Continuous improvement of the user interface based on diverse user feedback can enhance the overall user experience. Implementing user personalization features and addressing feedback variability can lead to a more tailored and intuitive experience.
4. **Scalability Testing:** Conducting additional scalability testing to evaluate system performance under even larger user loads and expanded IoT integrations is crucial. This would ensure that the system remains robust and reliable as it scales to accommodate more extensive event environments.
5. **Integration of Advanced Analytics:** Incorporating advanced analytics and machine learning capabilities could provide deeper insights into event management. Analyzing user behavior, predicting event trends, and optimizing resource allocation through data-driven insights could further enhance the system's functionality.
6. **Expanding IoT Capabilities:** Exploring new IoT technologies and expanding the range of devices supported can enhance automation and monitoring capabilities. This includes integrating advanced sensors, control systems, and other IoT innovations to provide more comprehensive event management solutions.
7. **User Training and Support:** Developing additional training materials and support resources can help users better understand and utilize the system's features. Providing ongoing support and regular updates based on user needs and technological advancements will ensure continued success and user satisfaction.

## 6.4 Conclusion

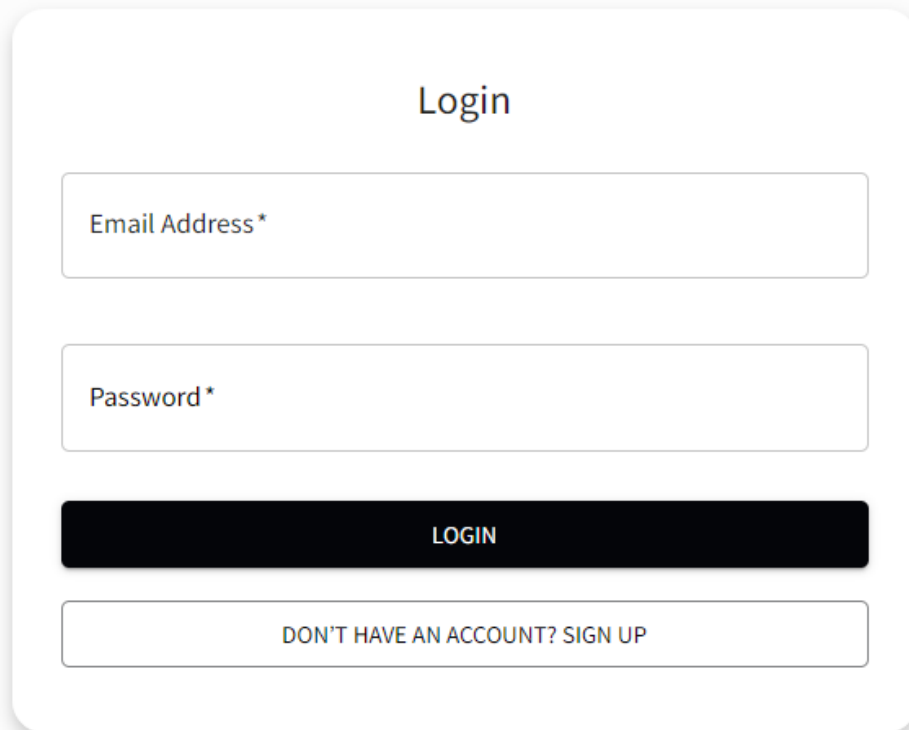
In conclusion, the "Event Syrup" project has successfully delivered a sophisticated Event Management System that integrates modern web technologies with IoT functionalities. The system's scalability, high availability, data security, and responsive user interface represent significant achievements. However, addressing the identified limitations and pursuing future enhancements will further strengthen the system and expand its capabilities.

The project has demonstrated the potential of combining advanced technologies to create innovative solutions for event management. By continuously evolving and adapting to user needs and technological advancements, the system can continue to provide valuable support for event organizers and contribute to the success of future events.

# Appendices

## Appendix A – Important Frontend and Backend pages Snapshots

Login Page:

A screenshot of a login page. It features a white rounded rectangle centered on a light gray background. Inside the rectangle, the word "Login" is centered at the top. Below it are two input fields: the first is labeled "Email Address\*" and the second is labeled "Password\*". Below the password field is a solid black button with the word "LOGIN" in white capital letters. At the bottom of the rectangle is a white button with a thin gray border containing the text "DON'T HAVE AN ACCOUNT? SIGN UP" in black capital letters.

Login

Email Address\*

Password\*

LOGIN

DON'T HAVE AN ACCOUNT? SIGN UP

Figure 34: Login Page

## Signup Page:

Sign Up

Name \*

Purchase ID \*

Email Address \*

Password \*

**SIGN UP**

ALREADY HAVE AN ACCOUNT? LOGIN

Figure 35: Signup Page

Dashboard Page:

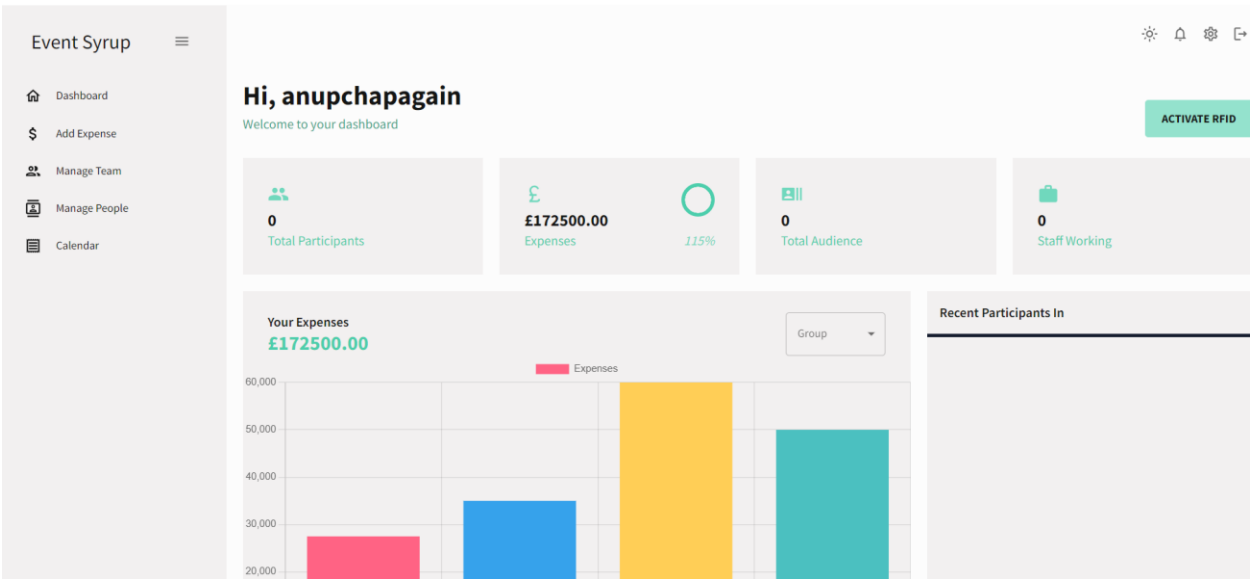


Figure 36: Dshboard Page

Expenses Handling Page:

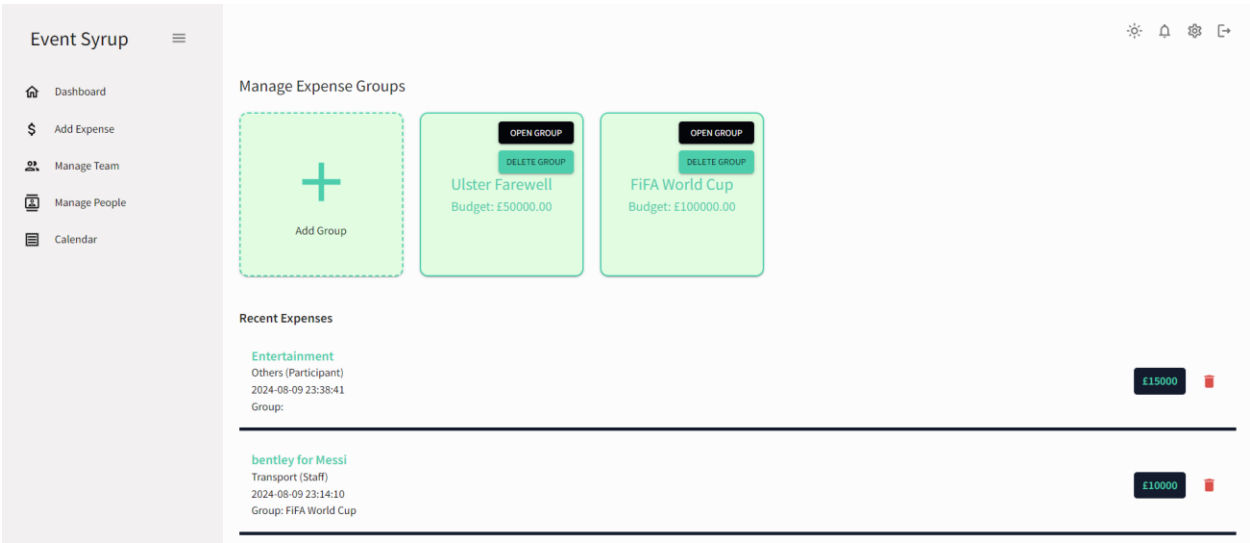


Figure 37: Add Expenses Save

Team Handling Page:



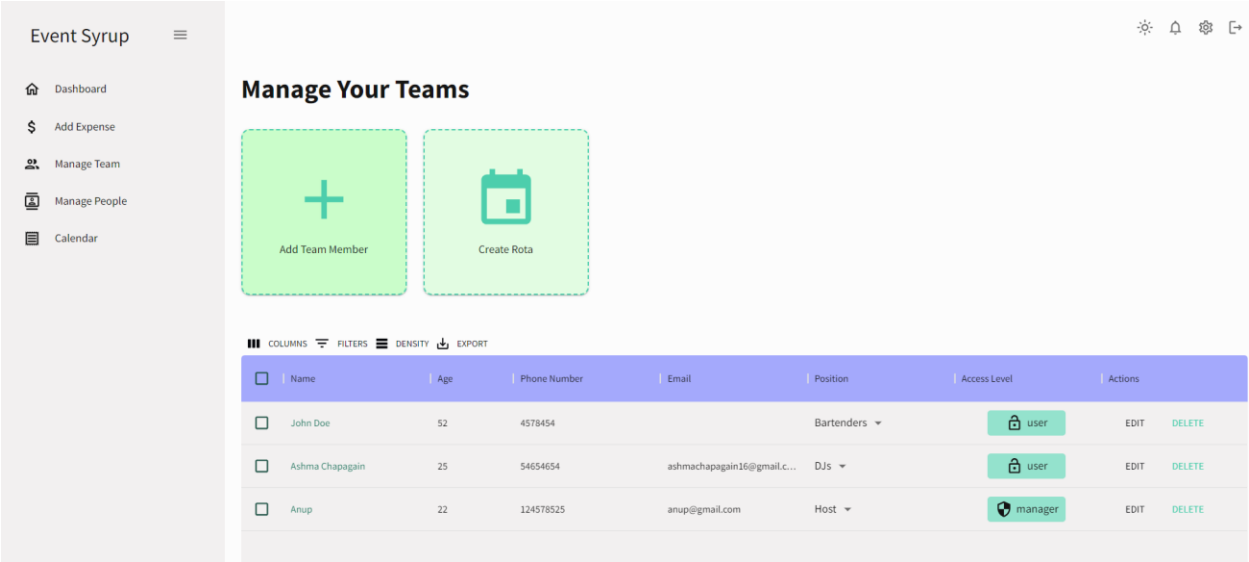


Figure 38: Staff Management Page

Rota Managing Page:

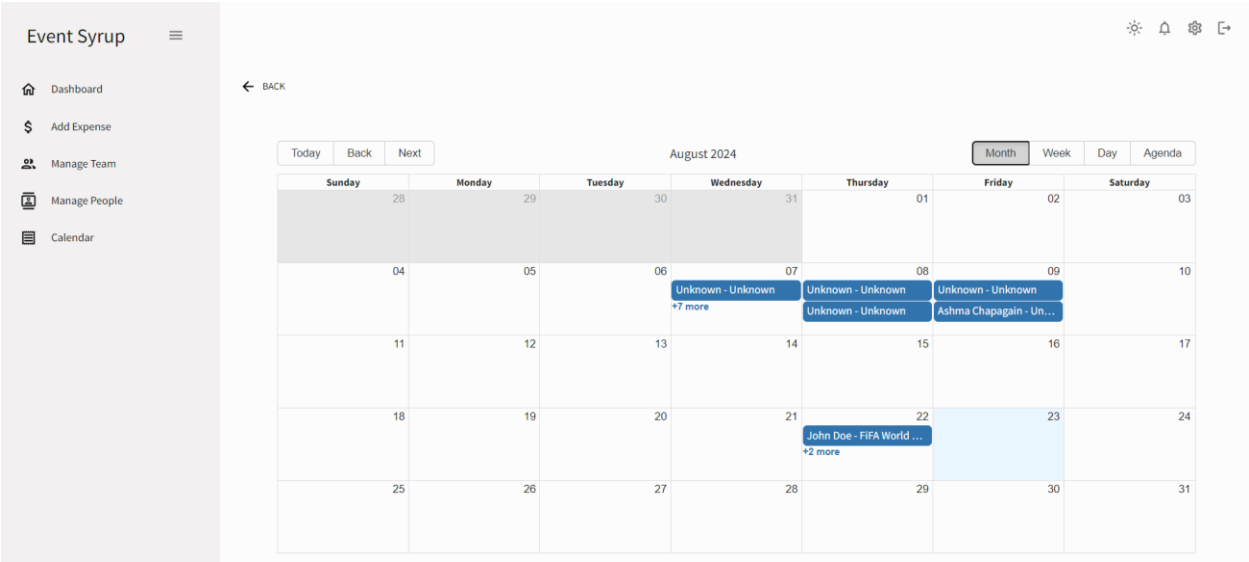


Figure 39

#### Event Groups

August 9th, 2024

Staff: Ashma Chapagain is appointed for 9:00 PM to 12:00 AM for Unknown.

EDIT

Staff: Unknown is appointed for 6:25 PM to 9:00 PM for Unknown.

EDIT

August 22nd, 2024

Staff: John Doe is appointed for 11:03 PM to 11:03 AM for FIFA World Cup.

EDIT

Staff: Anup is appointed for 11:04 PM to 11:04 PM for Ulster Farewell.

EDIT

Staff: Ashma Chapagain is appointed for 11:03 PM to 11:03 AM for FIFA World Cup.

EDIT

Figure 40

## Backend Firebase Rules:

Firebase security rules are an essential feature that provide fine-grained control over who can access your Firebase resources, including Firestore, Realtime Database, and Cloud Storage. These rules are written in a declarative syntax, enabling you to define conditions that determine whether a particular operation (like read, write, or delete) should be allowed. Mastering Firebase security rules is crucial for safeguarding your app's data against unauthorized access (Firebase, 2024).

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // User-specific data: only the authenticated user can read/write their data
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }

    // Expense Groups: only the user who created the group can manage it
    match /expenseGroups/{groupName} {
      allow create: if request.auth != null && request.auth.uid == request.resource.data.userId;
      allow read, update, delete: if request.auth != null && request.auth.uid == resource.data.userId;
    }

    // Expenses: only the user who created the expense can manage it
    match /expenses/{expenseId} {
      allow create: if request.auth != null && request.auth.uid == request.resource.data.userId;
      allow read, update, delete: if request.auth != null && request.auth.uid == resource.data.userId;
    }

    // Team Members: only the user who created the team member can manage it
    match /teamMembers/{teamMemberId} {
      allow create: if request.auth != null && request.auth.uid == request.resource.data.userId;
      allow read, update, delete: if request.auth != null && request.auth.uid == resource.data.userId;
    }
  }
}
```

Figure 41: Fire Base rule-1

```

// Rotas: only the user who created the rota can manage it
match /rotas/{rotaId} {
  allow create: if request.auth != null && request.auth.uid == request.resource.data.userId;
  allow read, update, delete: if request.auth != null && request.auth.uid == resource.data.userId;
}

// RFIDs: only the user who created the RFID can manage it, but allow any authenticated user to read
match /RFIDs/{rfidId} {
  allow create: if request.auth != null && request.auth.uid == request.resource.data.userId;
  allow read, update, delete: if request.auth != null && request.auth.uid == resource.data.userId;
  allow read: if request.auth != null;
}

// Scanned RFIDs: allow any authenticated user to manage scanned RFIDs
match /scannedRFIDs/{scannedId} {
  allow create: if request.auth != null;
  allow read, update, delete: if request.auth != null;
}

// Notifications: only authenticated users can create, read, update, and delete their own notification
match /notifications/{notificationId} {
  allow create: if request.auth != null; // Allow authenticated users to create notifications
  allow read, update, delete: if request.auth != null && request.auth.uid != null; // Allow authentications
}

// Pending Users: allow anyone to create, only admins can read/update/delete
match /pendingUsers/{userId} {
  allow create: if true; // Allow anyone (authenticated or not) to create pending user records
  allow read, update, delete: if request.auth != null && request.auth.token.admin == true; // Allow or
}

```

Figure 42: Firebase Rule

## MUI Libraries Used:

```

import GroupIcon from "@mui/icons-material/Group";
import RecentActorsIcon from "@mui/icons-material/RecentActors";
import WorkIcon from "@mui/icons-material/Work";
import PoundIcon from "@mui/icons-material/CurrencyPound";
import Header from "../components/Header";
import StatBox from "../components/StatBox";

```

Figure 43

```

import { Box, Typography, useTheme, Card, Button, TextField, Modal, MenuItem, IconButton } from "@mui/material";
import { DataGrid, GridToolbar } from "@mui/x-data-grid";
import SearchIcon from '@mui/icons-material/Search';
import ClearIcon from '@mui/icons-material/Clear';
import { tokens } from "../../theme";
import AdminPanelSettingsOutlinedIcon from "@mui/icons-material/AdminPanelSettingsOutlined";
import LockOpenOutlinedIcon from '@mui/icons-material/LockOpenOutlined';
import SecurityOutlinedIcon from '@mui/icons-material/SecurityOutlined';
import AddIcon from '@mui/icons-material/Add';
import EventIcon from '@mui/icons-material/Event';

```

Figure 44

Material Icons are been widely used.

## Appendix B – All the Figma Prototypes

### Constant Component Prototype:

Here are the Design Brainstorming processes I had gone through to get this fine looking frontend UI.

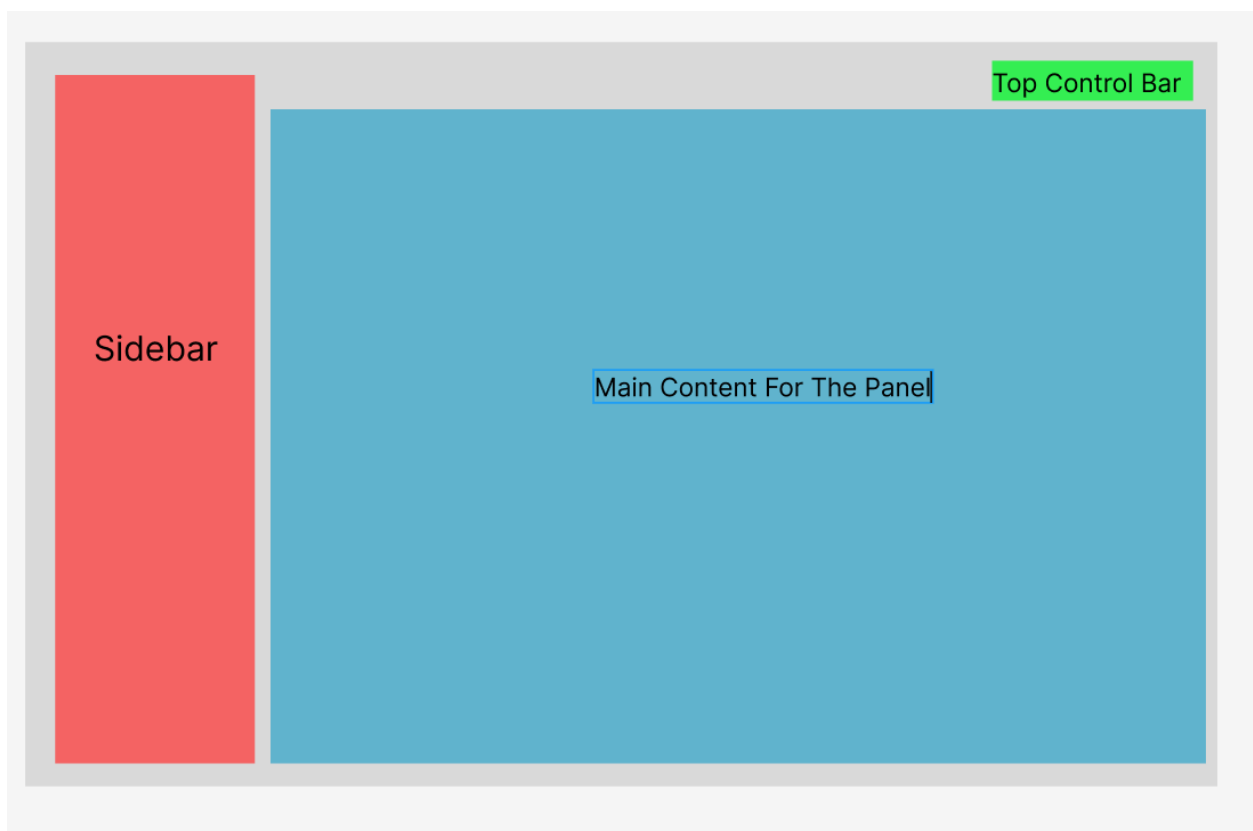
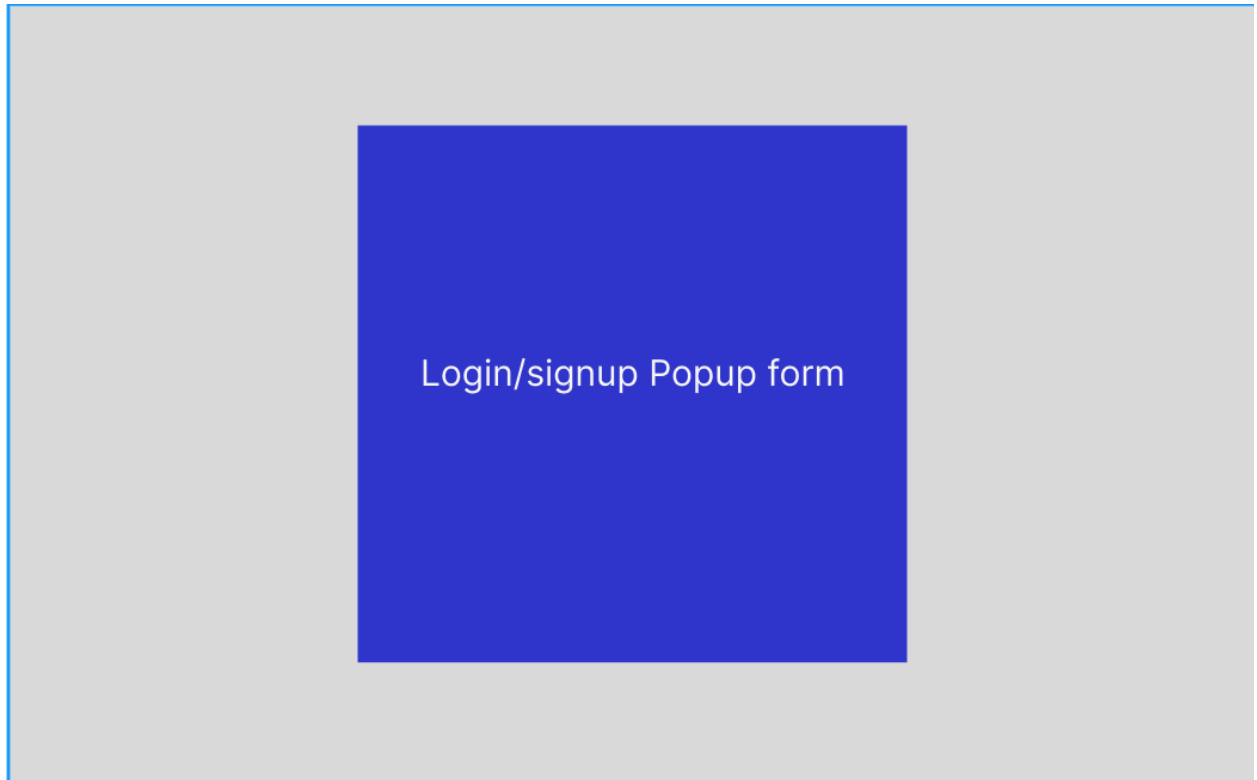


Figure 45Main Prototype



*Figure 46: Login Prototype*

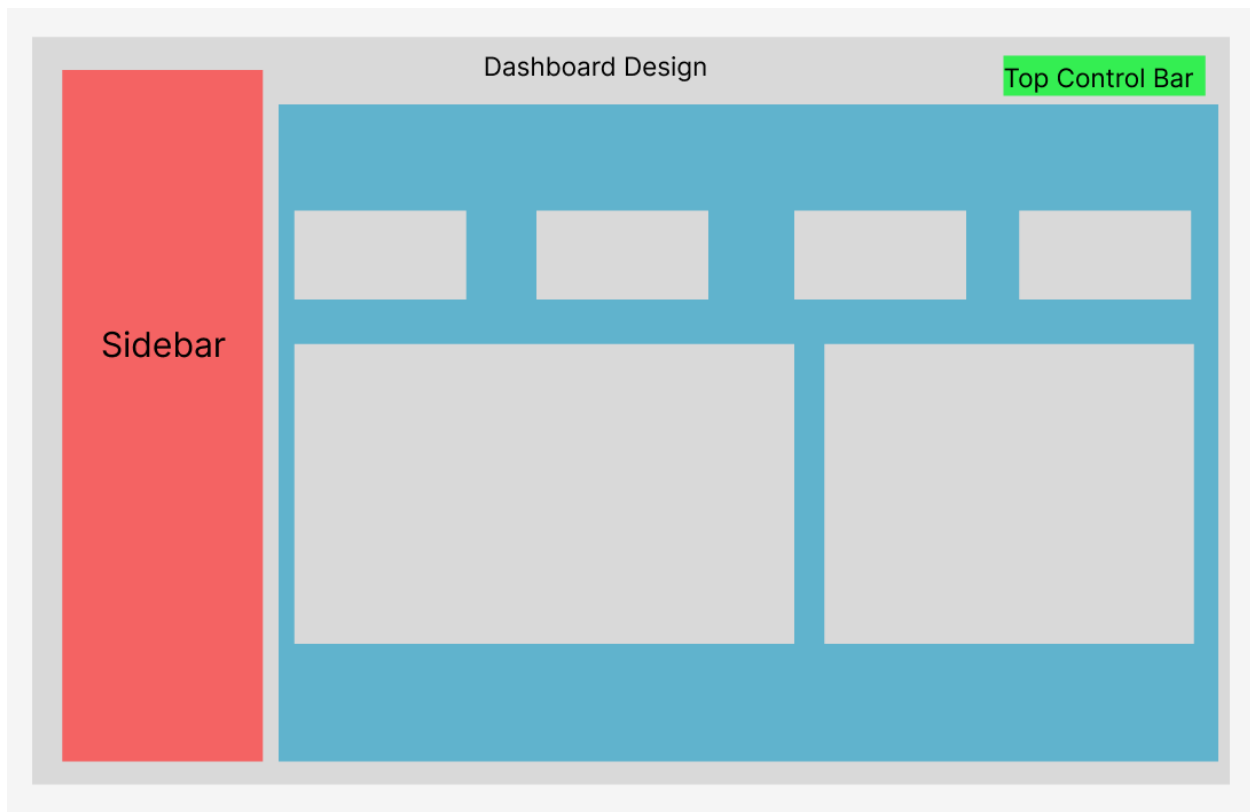


Figure 47: Dashboard Design Prototype

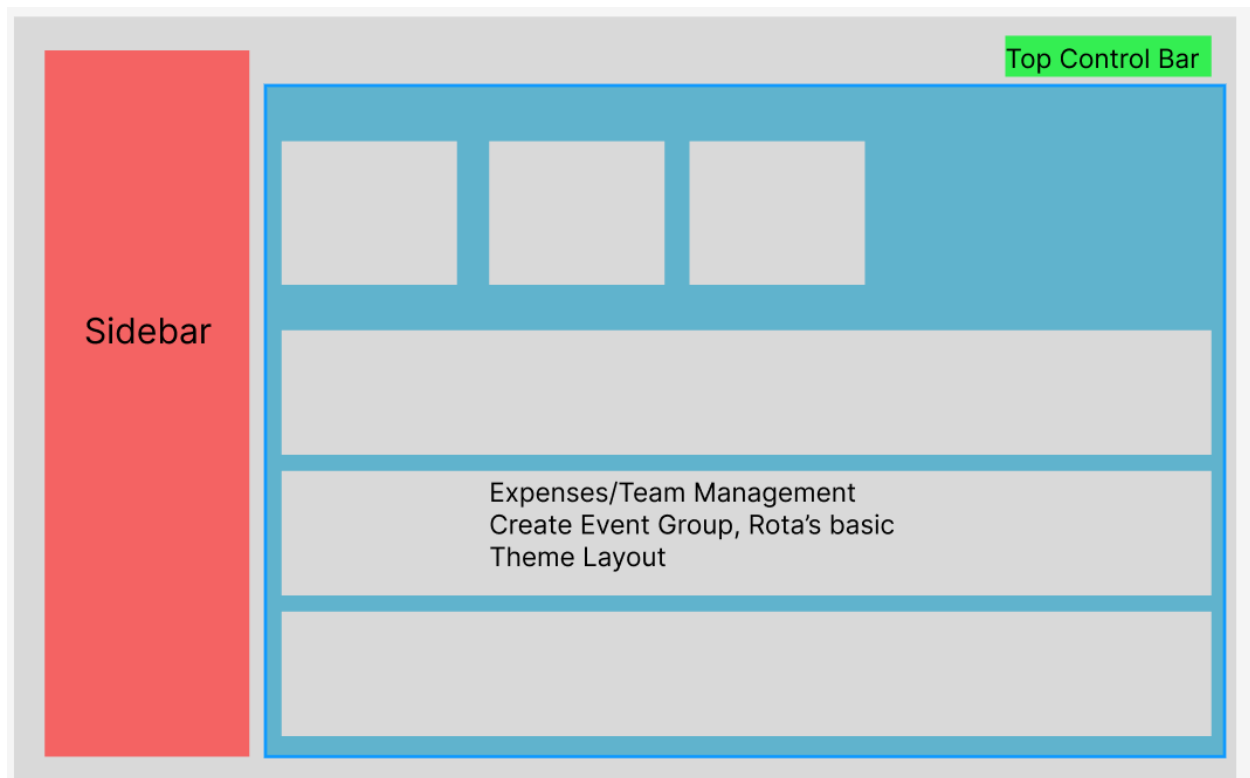


Figure 48: All the other functionality pages basic frame

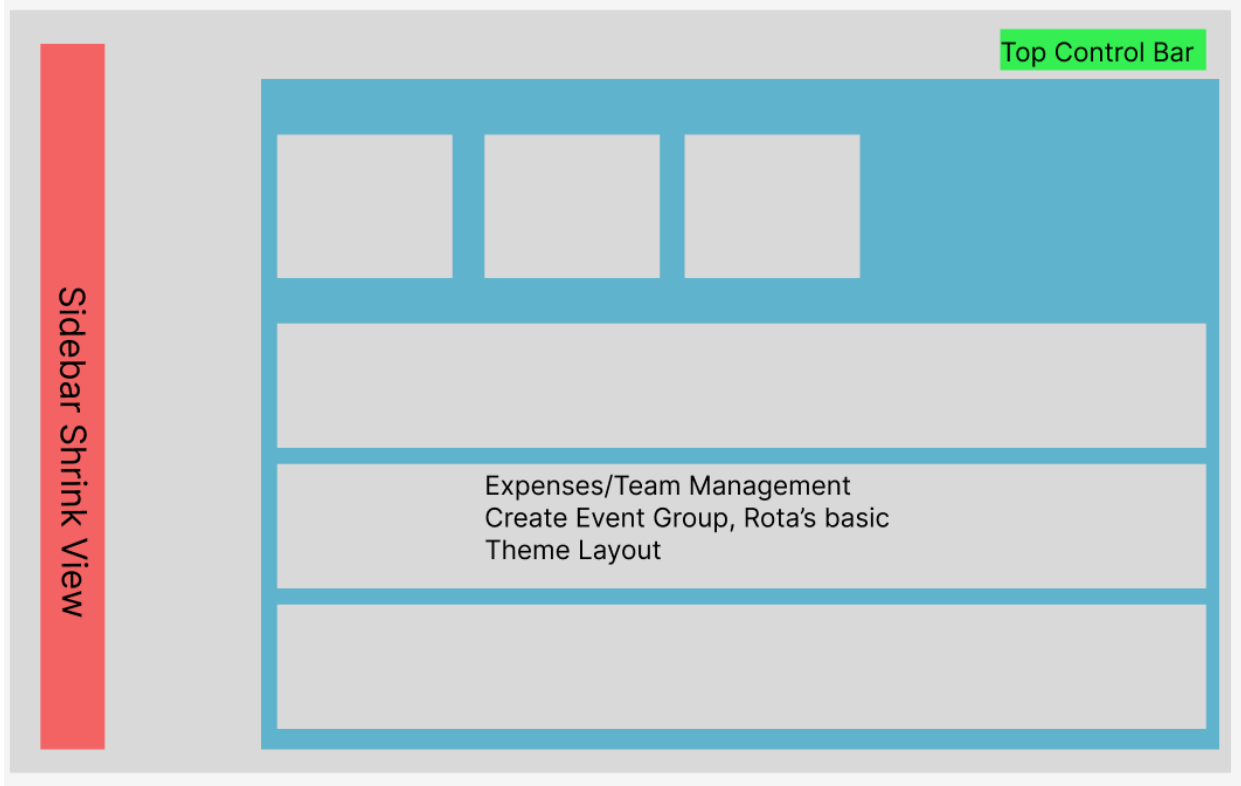


Figure 49: Sidebar Shrinking enabled looks check

## Appendix C -Abbreviations:

<b>API</b>	Application Programing Interface
<b>RFID</b>	Radio Frequency Identification
<b>EMS</b>	Event Management System
<b>SQL/NO SQL</b>	Structured Query Language / No Structured Query Language
<b>UI</b>	User Interface
<b>UX</b>	User Experience
<b>IoT</b>	Internet of Things
<b>UAT</b>	User Acceptance Testing
<b>CDN</b>	Content Delivery Network

## Appendix D – Hardware and Software Used

<b>Hardware Used</b>	<ul style="list-style-type: none"> <li>• RaspberryPi 4 Model B</li> <li>• DHT 11 Temperature Sensor</li> <li>• RFID-RC522 Sensor</li> <li>• RGB LED Light</li> <li>• Laptop</li> </ul>
----------------------	--

<b>Software Used</b>	<ul style="list-style-type: none"> <li>• Visual Studio Code</li> <li>• RaspberryPi Imager</li> <li>• PuTTY</li> <li>• Real VNC Viewer</li> <li>• Google Chrome and Microsoft Edge Browser</li> </ul>
<b>GitHub Project Access</b>	<a href="https://github.com/Anupcchapagain/EventSyrup-admin">Anupcchapagain/EventSyrup-admin (github.com)</a>

## References

Ali Dalgic, D.T.K.B.S.B. (2020) *Impact of ICTs on Event Management and Marketing*. In: Ali Dalgic, D.T.K.B.S.B., ed. IGI Global.

Bizzabo (2024) *Bizzabo*. Available from: [https://www.bizzabo.com/?utm\\_source=bing&utm\\_medium=paidsearch&utm\\_campaign=demo-request-bing&utm\\_content=brand-exact-uk&utm\\_term=brand-exact-uk&msclkid=8f2b2330ae75142b2a9e45f7000094b2](https://www.bizzabo.com/?utm_source=bing&utm_medium=paidsearch&utm_campaign=demo-request-bing&utm_content=brand-exact-uk&utm_term=brand-exact-uk&msclkid=8f2b2330ae75142b2a9e45f7000094b2) [Accessed 11 July 2024].

Coursera Staff (2023) *What Is User Acceptance Testing (UAT)?* Available from: <https://www.coursera.org/articles/what-is-user-acceptance-testing> [Accessed 16 July 2024].

Cvent (2024) *Cvent*. Available from: <https://www.cvent.com/uk> [Accessed 10 July 2024].

Elnaz Siامي-Irdemoosa, S.R.D.M.S.Z. (2015) Work breakdown structure (WBS) development for underground construction. *Automation in Construction*, 16 July, 58 (0926-5805), pp.85-94. <https://doi.org/10.1016/j.autcon.2015.07.016>. Available from: <https://www.sciencedirect.com/science/article/pii/S0926580515001594> [Accessed 14 July 2024].

Emadamerho-Atori Nefe (2023) *Angular vs. React vs. Vue.js: Comparing performance*. Available from: <https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/> [Accessed 08 April 2024].

EventBrite (2024) *Eventbrite*. Available from: <https://www.eventbrite.co.uk/> [Accessed 10 July 2024].

Firebase (2024) *Firebase Security Rules*. Available from: <https://firebase.google.com/docs/rules/> [Accessed 16 July 2024].

Geeks for Geeks (2024) *Requirements Engineering Process in Software Engineering*. Available from: <https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/> [Accessed 13 July 2024].



Google (2024) *Google Firebase*. Available from: <https://firebase.google.com/> [Accessed 07 August 2024].

Goray, S. (2024) *All About MongoDB NoSQL Database: Top Advantages and Disadvantages*. Available from: <https://webandcrafts.com/blog/mongodb-advantages-and-disadvantages> [Accessed 09 April 2024].

H. Hrablik Chovanova; R. Husovic; D. Babcanova; H. Makysova (2020) *Agile Project Management — What is It?* Available from: <https://ieeexplore.ieee.org/abstract/document/9379181> [Accessed 14 July 2024].

Henry H. Liu (2011) *Software Performance and Scalability : A Quantitative Approach*. John Wiley & Sons.

Ian Yeoman, Jane Ali-Knight, Martin Robertson, Siobhan Drummond, Una McMahon-Beattie (2012) *Festival and Events Management*. In: Ian Yeoman, J.A.-K.M.R.S.D.U.M.-B., ed. Taylor & Francis.

Jeff Forcier, Paul Bissex, Wesley J Chun (2008) *Python Web Development with Django*. Addison-Wesley Professional.

Jerker Delsing (2017) *IoT Automation : Arrowhead Framework*. CRC Press.

Kelly Main, Rachel Williams (2024) *Best CRM Software Of 2024*. Available from: <https://www.forbes.com/advisor/business/software/best-crm-software/> [Accessed 13 July 2024].

Python (2024) *Flask 3.0.3*. Available from: <https://pypi.org/project/Flask/> [Accessed 07 August 2024].

RaspberryPi (2024) *Computing for Everybody*. Available from: <https://www.raspberrypi.com/> [Accessed 07 August 2024].

ReactJs (2024) *React*. Available from: <https://www.react.dev> [Accessed 7 August 2024].

Subhas Chandra Mukhopadhyay (2014) *Internet of Things : Challenges and Opportunities*. Springer International Publishing.

Sujatha, P..S.G.V..R.A.S..&.S. (2015) The Role of Software Verification and Validation in Software Development Process. *Taylor and Francis*, 26 March, pp.23-26. <https://doi.org/10.1080/02564602.2001.11416938> Available from: <https://www.tandfonline.com/doi/abs/10.1080/02564602.2001.11416938> [Accessed 16 July 2024].

Suzzane Robertson (2024) *Volere – the Evolution of Successful Requirements Techniques*. Available from: [https://www.volere.org/wp-content/uploads/2019/07/2019\\_volere\\_history.pdf](https://www.volere.org/wp-content/uploads/2019/07/2019_volere_history.pdf) [Accessed 14 July 2024].

Vladimir Khorikov (2020) *Integration Testing. Unit Testing Principles, Practices, and Patterns*. Manning.

Vladimir Khorikov (2020) What is Unit Test? *Unit Testing Principles, Practices, and Patterns*. Manning. Ch. 2.

Whova (2024) *Whova*. Available from: <https://whova.com/> [Accessed 10 July 2024].