

Kennesaw State University
College of Computer and Software Engineering

DEPARTMENT OF COMPUTER SCIENCE

CS 4850 - Section 01 - Fall 2023

SP-32: EVENT TRACKER APP

KSU EVENT TRACKER

Team members:

Angie Djonkep / Kerian Ebot / Steeve Mocto / Warren Fongang

Professor: *Sharon Perry*

Website Link: https://sp-32-event-tracker-app.github.io/Event_Tracker_Website/

GitHub Link: <https://github.com/SP-32-Event-Tracker-App>

Friday, December 1, 2023



Table of contents

1. Introduction	Page 3
2. Project Overview	Page 3
3. Requirements	Page 3
4. Design	Page 4
4.1. Web Scraper	Page 4
4.2. Database	Page 5
4.3. Event Management	Page 6
4.4. Notification System	Page 8
4.5. API Component	Page 8
5. Project Plan	Page 9

6. Version Control	Page 10
7. Architectural Drawings	Page 10
8. Challenges, Assumptions, and Risk Assessments	Page 11
8.1. Challenges	Page 11
8.2. Lessons Learned	Page 12
9. Test Plan and Test Report	Page 12
10. Conclusion	Page 14
11. Appendix	Page 15

1. INTRODUCTION

The Event Tracker App is a mobile application developed using Flutter, integrated with a MySQL database hosted on AWS. The primary goal of the app is to provide users with a convenient tool to track and manage events sourced from a designated website in this case “OWLLIFE.” The KSU (Kennesaw State University) Event Tracker is an innovative mobile application designed to enhance the campus experience by simplifying event discovery and management. This report provides a comprehensive project overview, covering requirements, design, project plan, version control, architectural drawings, challenges, assumptions, risk assessments, and testing.

2. PROJECT OVERVIEW

The KSU Event Tracker aims to revolutionize how students engage with campus life, providing a streamlined event tracking process, personalized notifications, and insights into event participation. The project includes a mobile app, a desktop app for event creation, and a website highlighting the work. This project is aimed at providing a user-friendly mobile app for event tracking and implementing a notification system to keep users informed about upcoming events.

The app's scope will not only deliver pre-existing events to users but will also enable administrators or any other authorized users to add, modify and delete events easily. Additionally, it offers a notification system to alert users about upcoming events. The project's scope covers the development and deployment of the app along with its associated database.

3. REQUIREMENTS

The requirements document outlines the goals, assumptions, and functional requirements of the KSU Event Tracker. It emphasizes creating a user-friendly mobile app, enhancing the campus community experience, and providing management with an overview of popular events.

Some Functional requirements:

- **Launch Page**
 - Login button redirects to the login page.
 - Register button redirects to create account page.
- **Login Page**
 - Login with username.
 - Password.
 - Recover password (optional).
- **Create Account Page**
 - Email.
 - Password.
 - First Name.
 - Last Name.
 - Phone.
- **Event Creation and Management Page (Admin Only)**
 - Authorized event organizers should create, edit, and delete events.
 - Event details should include title, date, time, location, description, and category, benefits (if they apply).
- **Event Search and Filter Pages**
 - Users must search for events by keywords using a search bar.
 - Filter options for narrowing down search results must be provided such as date, location, and category, benefit.
- **Event Details Page**
 - Users must view comprehensive event details, including organizers, number of attendees, benefits, and associated media.

- **Event RSVP and Registration Pages**
 - Users must be able to RSVP or register for events.
 - Event organizers should manage attendee lists.
- **Notifications Page**
 - Users should receive notifications for upcoming events, RSVP confirmations, and updates.
 - Users should be able to customize notification preferences.
 - Users should opt in or out of notifications.
- **Event Sharing Page**
 - Users should share event details on social media and messaging apps.

4. DESIGN

The design document presents the architecture of the KSU Event Tracker, detailing the login process, event management, data retrieval from external sources, and notification customization. Firebase authentication manages login, API calls handle data interactions, and Firebase notifications personalize event notifications.

4.1. WEB SCRAPER

Initially, the web scraper encountered challenges while extracting events from the RSS feed due to inconsistent attribute availability and varying event formats. To overcome this, we shifted our approach to reverse-engineering how the website itself displays events. Upon investigation, we discovered an accessible API without the need for authentication. Leveraging these API calls through the developer console, we successfully obtained the required data after rigorous testing of different attributes and routes. Subsequently, we developed a script to automate daily data retrieval, ensuring real-time updates. Prior to database insertion, we implemented a mechanism to remove outdated events. This proactive approach prevents displaying past events and guarantees the latest information is consistently refreshed. Overall, this streamlined process enhances the accuracy and reliability of our event data collection and management system.

4.2. DATABASE

4.2.1. DATABASE DESIGN

- DBMS Selection:

Our initial task involved selecting the suitable Database Management System (DBMS) for this project. As a result, we opted for MYSQL as the database management system. The choice was driven by its well-established reputation for reliability, ease of use, and strong community support. MySQL's compatibility with Flutter and its ability to handle complex queries made it an ideal fit for the data storage requirements of the Event Tracker App. Furthermore, more than 90% of the group members had beforehand experience in MYSQL and this facilitated our choice.

- Actors:

After selecting the appropriate DBMS, we had to identify the different actors and their interactions with the event tracker app. These actors are:

- **Admin/Event Coordinator:** The person or team responsible for managing and inputting event information into the system.
- **School:** Represents the educational institution hosting the events.
- **Student:** The primary user of the mobile application who wants to receive event notifications.

- Entities, Attributes, and Relationships:

The next step in the design of the database is to identify all the entities, their attributes, and the possible relationships between them. In the following lines, we shall explore the different entities, their attributes, and relationships.

Entities are real world objects whose information (attributes) can be classified. In the event tracker application, the entities, and their attributes we could identify were:

ENTITY	ATTRIBUTES
Student	StudentID, Name, Email, Notification
Event	EventID , Title, Description, Picture, Date, Location.
Notification	EventID, NotificationID, StudentID, email, message, sent.
Registration	StudentID , EventID , RegistrationID

The link between entities, attributes, and the relationships between them can be illustrated more easily and comprehensively with the help of an ER-Diagram as shown below:

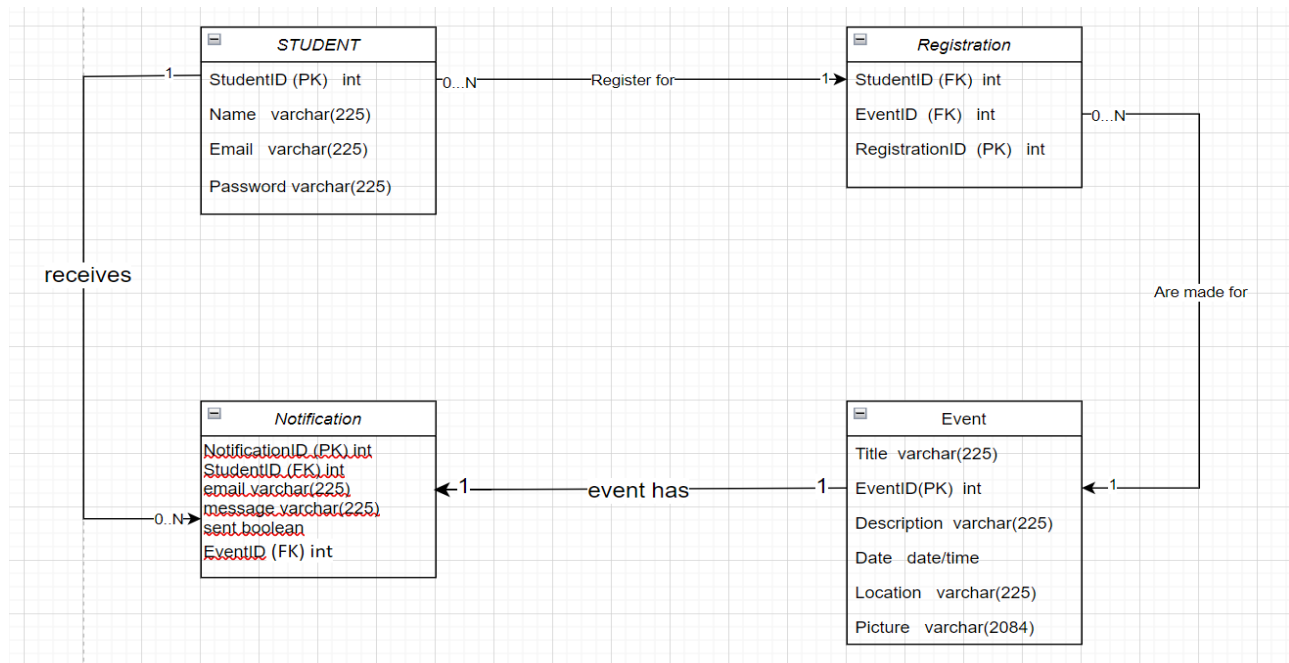


FIGURE 1: ER-DIAGRAM

4.2.2. DATABASE IMPLEMENTATION

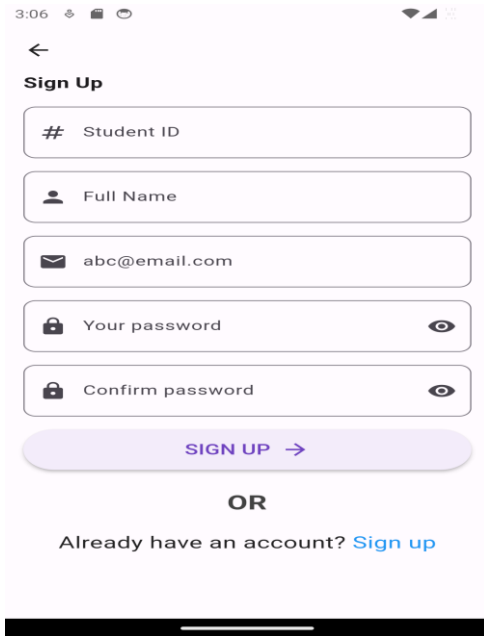
To implement the database, we used tools like workbench which is a software tool or interface that provides a unified environment for database developers and administrators. It typically includes features for designing database schemas, writing, and testing queries, and managing database structures. After having built the database's structure we need to populate the database. Data is collected from the designated website through a scraping mechanism. The collected data undergoes preprocessing before being stored in the MySQL database. The database ensures a structured storage format, facilitating efficient retrieval by the app.

4.3. EVENT MANAGEMENT

The management of the app involves a set of Python and Fast API written APIs powered by the Unicorn web server. These APIs perform various tasks, such as querying the database for user registration and authentication via Firebase, utilizing token-based authentication. Deployment is done on an Amazon Web Services EC2 instance, ensuring accessibility for all group members. For an in-depth view, you can explore the API.

4.3.1. USER INTERFACE

The user interface is designed with a focus on simplicity and functionality. Users can easily add, edit, and delete events. Intuitive categorization and tagging features enhance the organizational aspects of the app. The images below show how the app presents itself.



3:06

←

Sign Up

Student ID

Full Name

abc@email.com

Your password

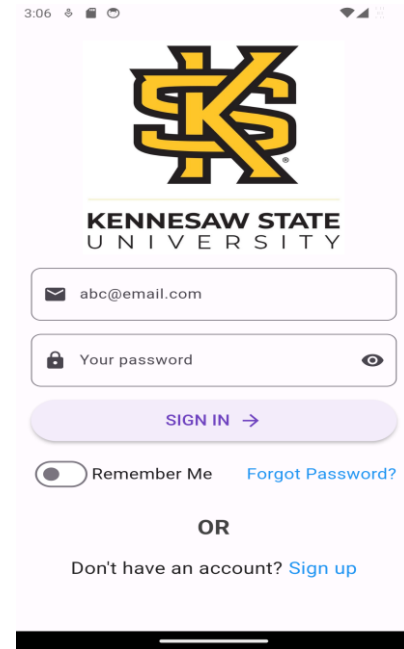
Confirm password

SIGN UP →

OR

Already have an account? [Sign up](#)

FIGURE 2: LOGIN PAGE



3:06

Kennesaw State University

abc@email.com

Your password

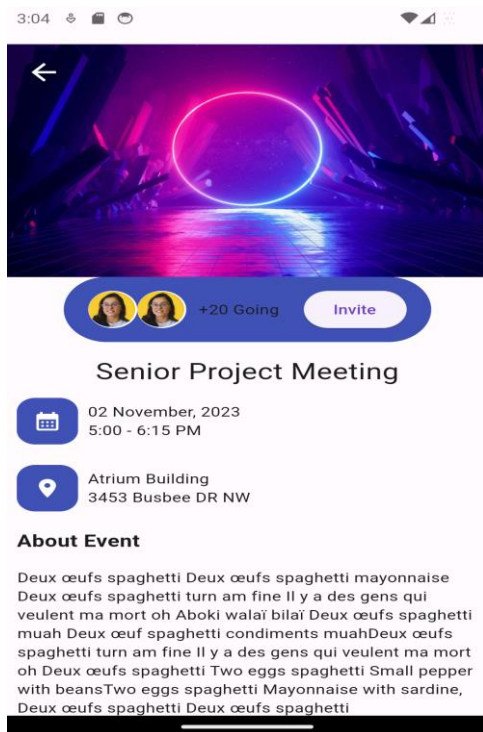
SIGN IN →

☐ Remember Me [Forgot Password?](#)

OR

Don't have an account? [Sign up](#)

FIGURE 3: LOGIN PAGE



3:04

←

+20 Going **Invite**

Senior Project Meeting

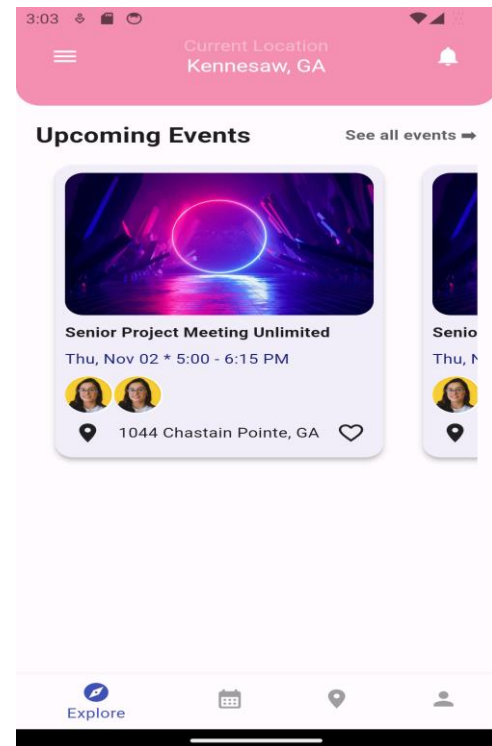
02 November, 2023
5:00 - 6:15 PM

Atrium Building
3453 Busbee DR NW

About Event

Deux œufs spaghetti Deux œufs spaghetti mayonnaise
Deux œufs spaghetti turn am fine Il y a des gens qui
veulent ma mort oh Aboki walaï bilaï Deux œufs spaghetti
muah Deux œuf spaghetti condiments muahDeux œufs
spaghetti turn am fine Il y a des gens qui veulent ma mort
oh Deux œufs spaghetti Two eggs spaghetti Small pepper
with beansTwo eggs spaghetti Mayonnaise with sardine,
Deux œufs spaghetti Deux œufs spaghetti

FIGURE 4: EVENT PAGE



3:03

Current Location
Kennesaw, GA

Upcoming Events [See all events →](#)

Senior Project Meeting Unlimited
Thu, Nov 02 * 5:00 - 6:15 PM
1044 Chastain Pointe, GA

Senio
Thu, N

Explore

FIGURE 5: HOME PAGE

4.3.2. FUNCTIONALITY

The app facilitates seamless event management, allowing users to perform CRUD operations on events. Categories and tags enable users to filter and search for events efficiently. Integration with the MySQL database ensures real-time synchronization of event data.

4.3.3. INTEGRATION WITH DATABASE

The app communicates with the MySQL database through APIs, ensuring secure and efficient data retrieval. The integration enables a smooth user experience by providing accurate and up-to-date event information.

4.4. NOTIFICATION SYSTEM

4.4.1. OVERVIEW

Notifications play a crucial role in keeping users informed. The app incorporates a notification system to send timely reminders and updates about upcoming events.

4.4.2. IMPLEMENTATION

The notification system is integrated using Flutter's native notification features. AWS Simple Notification Service (SNS) is employed for sending push notifications, ensuring reliable and scalable delivery.

4.4.3. AWS INTAGRATION

The MySQL database is hosted on AWS using the free tier. AWS services, including S3 for storage and SNS for notifications, contribute to the overall robustness and reliability of the app.

4.5. API COMPONENT

The API component serves as the bridge between the user interface and the underlying data, facilitating seamless interaction with the Event Tracker system. Key functionalities include user registration, login, and comprehensive event management. The API design adheres to industry best practices, employing RESTful principles for clear and predictable endpoints. Robust security measures, including encryption and authentication mechanisms, are implemented to safeguard user data and ensure the integrity of the application. This section explores the intricacies of API design, detailing the endpoints for various operations, and sheds light on the security considerations that underpin the overall reliability and trustworthiness of the Event Tracker application.

4.5.1. DESKTOP APPLICATION COMPONENT

The Java and JavaFX-based desktop application component enhances admin experience by providing a dedicated platform for managing events. This component abstracts SQL queries, simplifying the process of creating, updating, and deleting events. JavaFX's rich user interface elements facilitate an intuitive and user-friendly environment, enabling users to interact with the application effortlessly. The desktop application also empowers users with the ability to create custom events, adding a layer of personalization to the event management process. By leveraging Java's robust capabilities and JavaFX's compelling UI (User Interface) components, this component ensures a seamless and efficient experience for users engaging with the Event Tracker application on desktop platforms.

4.5.2. MOBILE APPLICATION COMPONENT

The mobile application component, developed using Flutter and Dart, represents a key aspect of the Event Tracker's commitment to cross-platform accessibility and a unified user experience. Flutter's widget-based architecture enables the creation of a visually appealing and responsive interface across iOS and Android devices. Dart, with its just-in-time compilation, ensures high-performance execution on mobile platforms. The user interface design is carefully crafted to prioritize usability and aesthetics, providing an intuitive navigation flow for users to explore upcoming events, register, and manage their participation effortlessly. This section delves into the specifics of Flutter's role in crafting the mobile app, the challenges encountered in ensuring consistency across platforms, and the overall impact on user engagement and satisfaction.

5. PROJECT PLAN

The project plan includes milestones, meeting schedules, collaboration and communication plans, and task planning. Key milestones involve database design, event creation app development, web scraping for KSU events, mobile app prototypes, mobile app development, website creation, and the final report.

#1 - By 9/17/2023: Database design

- This milestone involves designing the database structure for our project. It includes defining tables, relationships, data types, and constraints to store and manage data efficiently.

#2 - By 9/24/2023: Event Creation App

- This achievement signifies the development of an application that allows users to create and manage events. Users can input event details, such as title, date, location, and description.

#3 - By 9/30/2023: Web scraper for KSU Events

- The web scraper milestone focuses on creating a web scraping script in Python that collects data about events happening at KSU from various sources. The scraped data can be used for various purposes, such as displaying events on the platform or for an API

#4 - By 10/7/2023: Mobile App Prototype

- Before developing the final mobile app, this milestone involves creating a prototype or mockup of the app's user interface and functionality. It serves as a visual representation to gather feedback and refine the app's design.

#5 - By 10/21/2023: Mobile App

- In this stage, the actual mobile app is developed based on the prototype. The app will have features like authentication, event browsing, notifications registration, and any other functionality required.

#6 - By 10/24/2023: Website

- This phase involves the development of a website that highlights the work done throughout the project and the course. It will also feature links to the various repositories and the mobile app.

#7 – By 10/27/2023: Final Report

- The final report milestone is typically associated with documenting the project's overall progress, achievements, challenges, and outcomes. It may include technical details, user guides, and future recommendations. This report provides a comprehensive overview of the entire project.

6. VERSION CONTROL

In our version control strategy, Git and GitHub play a significant role. We have established four repositories, each dedicated to a distinct project. The initial repository hosts a desktop application facilitating the creation and management of custom events, offering direct interaction without manual SQL queries. The second repository encompasses a web scraper and API application, including scripts for website data extraction, sanitization, and database updates. The third repository comprises the website, presenting various profiles and links. The fourth repository contains the mobile app code written in Flutter and Dart.

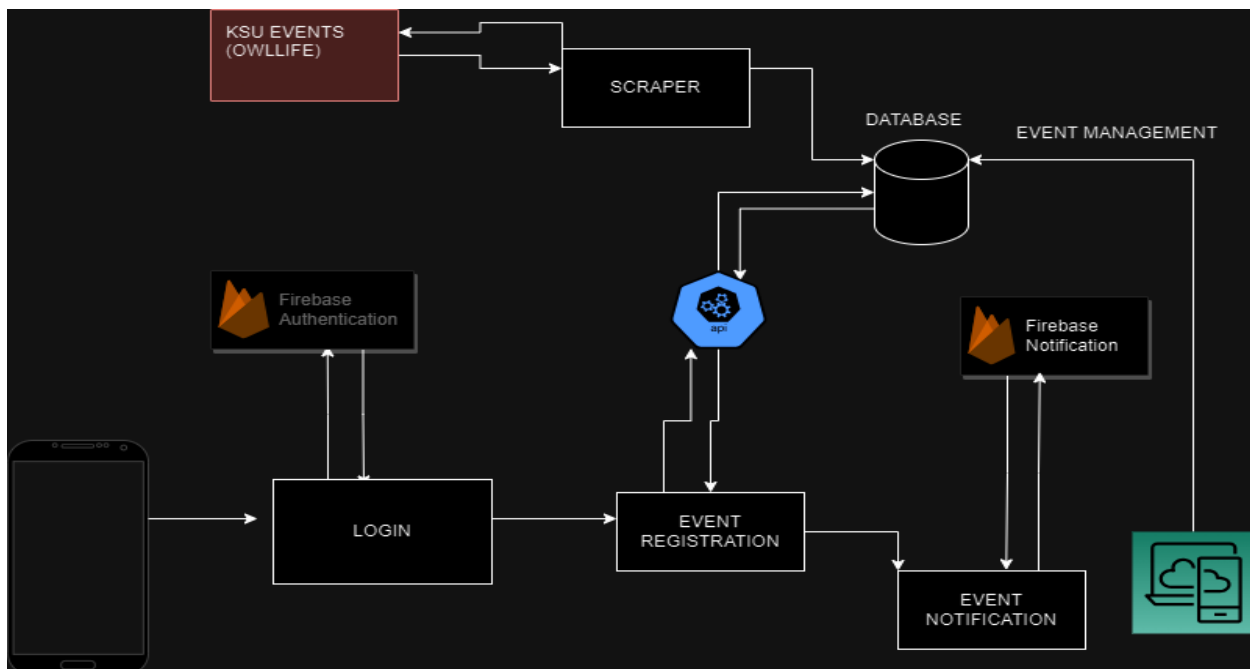
For each repository, we employ a main branch and create feature branches for specific additions. Pull requests are made, and after code changes or reviews, approval is contingent on code functionality, successful unit tests, and code readability and maintainability. We use VS Code with the Git extension or Git Desktop for a more visually appealing and user-friendly interface, facilitating seamless tracking of changes across branches.

7. ARCHITECTURAL DRAWINGS

Architectural drawings illustrate the interaction between various components of the KSU Event Tracker, including login, registration, event creation, data retrieval, and notifications.

Before building any app. We need to understand the unique design architecture that are involved for the app. There are a lot of things to consider whenever you are using. A mobile application and a mobile Framework such as Flutter.

Initially, data originates from our custom desktop app, allowing admins to create events. The remaining data is pulled directly from the live website, scripted and written to the database via Python. Upon user login, registration is possible by providing information such as username, password, first name, last name, phone number, and student ID, generating a JSON Web Token for resource access verification. Alternatively, the mobile app interfaces with Flutter and Firebase for login. Firebase issues a token after validating email and password. The API checks token validity. Users access pages like the main page, trending events, and profiles through API calls. Cached Python APIs are hosted on an EC2 instance with an Elastic IP address. User profile pictures are stored in Amazon S3, and local SQLite manages user-supported events in the Flutter app. Push notifications from Firebase enable event reminders through a publish-and-subscribe model using a message queue.



8. CHALLENGES AND LESSONS

This section discusses the challenges faced during the project, assumptions made, and an assessment of potential risks.

8.1. CHALLENGES

Overcoming website changes: Adapting to changes in the structure of the source website required dynamic scraping mechanisms.

AWS configuration: Configuring and optimizing AWS services presented challenges in the initial stages.

Api Configuration: Building adequate APIs was a hassle as most members had little or no experience in API implementation.

Daily Standups: Meetings were most often held online and not following the initial schedule as members had different and conflicting timetables.

Tutorials: It was difficult to find a reliable source of tutorials to implement components or code sections that applied to our project.

8.2. LESSONS LEARNED

Flexibility is key: Building adaptability into the scraping mechanism allowed for easier handling of website changes.

Thorough AWS exploration: A deeper exploration of AWS services upfront could have mitigated initial configuration challenges

Setting Up AWS Free-Tier: Creating an account in AWS and creating an instance of our database is not enough. One must make sure that the IP address that enables connection to the database is the same as that on the local computer, else connection will not be possible.

9. TEST PLAN AND TEST REPORT

To ensure the robustness of our app, we employ four distinct testing approaches: UI testing, manual testing, unit testing, and integration testing. Manual testing involves app usage, clicking, and assessing responsiveness. UI testing focuses on widget representation. Unit testing utilizes frameworks like JUnit, Mockito, and Dart's built-in testing module to verify code functionality in various scenarios. Integration testing, facilitated by the Selenium package, assesses code interaction. Selenium is adapted for mobile app testing as a web application, running on browsers like Chrome. For manual testing, two beta testers explore diverse scenarios, offering valuable feedback on registration, event participation, password recovery, viewing future and past events, social interactions, and security considerations. This comprehensive testing strategy ensures a thorough evaluation of the app's performance and user experience.

- Unit Testing with JUnit and Mockito:

Unit testing is a critical aspect of ensuring the reliability and correctness of the individual components within the Event Tracker application. JUnit, a widely used testing framework for Java, was employed to create and run unit tests for the Java and JavaFX desktop application. Mockito, a mocking framework, played a key role in isolating and testing individual units of code by simulating dependencies. Through a systematic approach, unit tests were designed to cover various scenarios, edge cases, and expected behaviors of functions within the application. This rigorous unit testing process significantly contributed to the stability of the desktop application.

- API Testing with PyTest and HTTPX:

API testing is essential for verifying the functionality and reliability of the backend services. PyTest, a testing framework for Python, was utilized to create comprehensive test suites for the API component. HTTPX, an HTTP client for Python, facilitated making HTTP requests and responses during testing. The test cases covered aspects such as user registration, authentication, and various CRUD (Create, Read, Update, Delete) operations for events. By systematically validating the API endpoints, the test plan ensured that the application's backend met the expected standards of performance and functionality.

- Manual Testing and User Acceptance Testing:

In addition to automated testing, manual testing played a crucial role in the verification process. Manual testing involved real users interacting with the application to identify usability issues, visual inconsistencies, and unexpected behaviors. User acceptance testing (UAT) involved collaborating with beta testers and end-users to validate that the application met the requirements and expectations. This hands-on testing approach allowed for a holistic assessment of the user experience and provided valuable insights into areas that automated tests might not cover.

- UI Testing with Selenium:

User interface (UI) testing is integral to ensuring that the application's front-end functions are as intended. Selenium, a powerful open-source tool for automating web browsers, was employed to create UI tests for the web-based components of the Event Tracker application. Selenium scripts were designed to simulate user interactions such as clicks, form submissions, and navigation. The UI testing plan covered critical user workflows, including event registration and account management, providing confidence in the application's front-end functionality and responsiveness.

For UI testing, we just made sure that our components and our widget looked exactly just like the mockups we had in the design. Sometimes it is difficult to replicate the design. Using appropriate performance and we are trying to see if the rendering was not overloading with the API calls. So that is what we did with the UI testing.

The test plan outlines strategies for testing the mobile app, desktop app, and website. The test report provides insights into the testing process, results, and any identified issues.

- Unit Testing:

Unit testing focuses on testing individual Dart functions and classes in isolation. We use the built-in testing framework in Dart, the **test** package. We develop test cases for each Dart function and class, covering various input scenarios.

- Integration Testing:

Integration testing focuses on verifying the collaboration and interaction between different Flutter widgets and modules. We develop test cases that cover the integration points between different widgets and modules, and we include scenarios that involve data flow and communication between modules.

- UI Testing:

UI testing will focus on validating the Flutter UI for functionality, responsiveness, and visual correctness. We develop widget tests that cover UI interactions, navigation, and visual elements and we include scenarios for different screen sizes and resolutions.

10. CONCLUSION

In conclusion, the KSU Event Tracker project aims to create a dynamic and user-friendly platform for campus event management. The Event Tracker App successfully meets its objectives, providing users with an intuitive platform for event management. The combination of Flutter, MySQL, collaboration of team members, adherence to the project plan, AWS, effective version control, and rigorous testing contribute to the success of this innovative application.

APPENDIX

Project plan:

(will be added)

Requirement Document:

(will be added)

Design Document:

(will be added)

Source Code: Will attach a zip file containing the source code exported from GitHub.