CS 3354 Software Engineering
Final Project Deliverable 2

FestGuide

Fatima Khalid, Adrian Martin, Kamal Al Shawa, Jasmine Mathers, Rida Zameer,
Mauricio Rodriguez Rios, Krishna Chilukuri

1. Delegation of Tasks

- **Kamal:** Architectural Design, Competitor Analysis
- **Mauricio:** Use Case Diagrams, Estimated costs of hardware and software
- **Adrian:** Functional and Non-Functional Requirements, GitHub Management, Personnel Costs Analysis
- **Rida:** Class Diagrams, GitHub management, Software Testing
- **Jasmine:** Sequence Diagrams, Project Scheduling and Cost Analysis
- **Fatima:** Overall Project Management, Presentation Slides, UX/UI
- **Krishna:** Software Model Process, Project Review and Conclusions

2. Project Deliverable 1 Content

3. Final project draft description with instructor feedback:

Title of our project

FestGuide

Group Members' first and last names

Fatima Khalid
Adrian Martin
Kamal Al Shawa
Jasmine Mathers
Rida Zameer
Mauricio Rodriguez Rios
Krishna Chilukuri

What you'll be doing

An application that helps organizations communicate information about festivals and other events. It would include important points of interest, and the ability to handle transactions among other things helpful for the people attending.

Detailed description of motivation and expected use cases

Festivals and events are a crucial part of culture and society today. Attending such events is a detailed task that requires organization, communication, and solid navigation. Our motivation in creating this app is to streamline the process of hosting and attending events. The app would allow organizations to create their events while also allowing other users to access the information about these events. Meaning, the app would host event creation while also hosting event viewing on the same platform.

List of tasks to be delegated to each person (1 sentence per person)

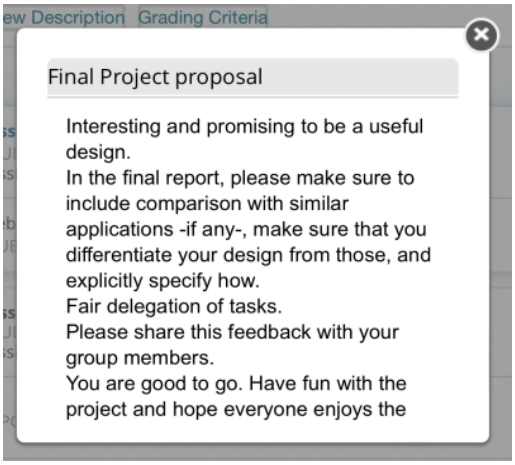
Kamal - UML Design
Mauricio- Use Case diagram

Adrian- Functional requirements, task 1.5
Rida - Class diagram, task 1.3 (add all team members & TA to Github)
Jasmine - Sequence Diagram
Fatima Khalid - Overall project Management, task 1.4

Scholar paper?

Not interested

Instructor Feedback:



Based on instructor feedback, we are confident that our project will be a functional and practical one. To improve the quality of our design based on the feedback, we will compare FestGuide with competitors. It is important to clearly specify what makes FestGuide different from other similar applications, because there is no need for redundancy in the software market. We will be including these detailed descriptions when presenting our final project to the audience. We have also **slightly revised** the delegation of tasks since the original document plan since we felt different areas of the project required more attention and we are aiming for a sound design.

4. Set up GitHub repository

https://github.com/RidaZameer07/CS3354-FestGuide

1.1 Everyone has made a GitHub
1.2 Repo created and named
1.3 Added team members and TA
1.4 Did first commit (ReadME)
1.5 Added Project scope file
1.6 Added all other project related files

5. Delegation of Tasks (initial submission)

Fatima Khalid:

- Revising final project draft proposal based on Instructor feedback
- Setting up basic GitHub repo

Adrian Martin:

- Software requirements: Functional and non-functional
- Initial GitHub commit

Kamal Al Shawa:

- Architectural design

Jasmine Mathers:

- Sequence diagram

Rida Zameer:

- Adding GitHub collaborators
- Class diagram

Mauricio Rodriguez Rios:

- Use case diagram

Krishna Chilukuri:

- Software Model Process

6. Which software model process is employed and why?

1) Iterative Development: Given the goal of our app we have the iterative approach of scrum allows the team to deliver increasing value to users with each sprint.

2) Flexibility: As the user and market trends evolve with the times, scrum allows the team to act on feedback and make changes accordingly.

3) Collaboration: The success of an app depends on the collaboration with the event organizers and attendees to understand what is needed and should be included.

4) Risk Mitigation: By breaking the project into smaller parts, scrum can help prevent risks especially with a complex project like this.

5) Transparency: Being transparent helps stakeholders understand the progress made and the failures that have occurred, so that informed decisions can be made.

7. Software Requirements
    a. 5-7 functional requirements

Users can add and edit event information such as name, date, time, and location to ensure accuracy and relevance.
Users can search for events based on categories, dates, or locations to find events easily.
Users can purchase event tickets within the app, providing convenience and an easy transaction experience.
The main page displays upcoming events, enhancing user engagement and encouraging participation.
Users can access event details which include date, time, location, and description for informed decision-making.
Users can share events on external platforms which helps facilitate event promotion and increases visibility among potential attendees.
The app includes a FAQ section addressing common event attendance questions, and providing support and guidance to users

b. One Nonfunctional Requirement For Each Type

Security Requirements: The use of user authentication and authorization mechanisms to protect user data. Also the Encryption of sensitive data such as payment information and user credentials.

Regulatory Requirements: Compliance with data protection regulations such as GDPR or CPRA to ensure user privacy.

Ethical Requirements: Being transparent in data usage and handling by ensuring user consent for data processing.

Legislative Requirements: Compliance with the local laws and regulations related to online ticket sales and event organization.

Operational Requirements: Scalability to accommodate an increasing number of users and events over time. Also, regular maintenance and updates to keep the system running smoothly.

Development Requirements: Adherence to coding standards and best practices to ensure code quality and maintainability.

Environmental Requirements: Minimization of resource consumption to reduce the environmental impact of the software.

Safety/Security Requirements: Measures to protect user safety during physical events, such as emergency contact information and venue safety guidelines. Also implementing security measures to prevent unwanted access to event-related data.

Accounting Requirements: Integration with accounting systems for accurate tracking of financial transactions and revenue generation from ticket sales.

Product Requirements: Compatibility with various devices to reach a wide user base and a well-made user interface design for easy navigation and interaction.

Organizational Requirements: App users must have a verified account to purchase tickets or create an event.

External Requirements: Following the regulations regarding the sale of tickets online and the management of events, ensuring that the platform is legally compliant and approved for use by regulatory bodies.

Efficiency Requirements: Optimized code for efficient use of system resources and minimization of key functionalities.

Dependability Requirements: Reliable error handling and logging to facilitate troubleshooting and debugging.

Performance Requirements: Fast response times for user interactions such as event searches, ticket purchases, and page loading.

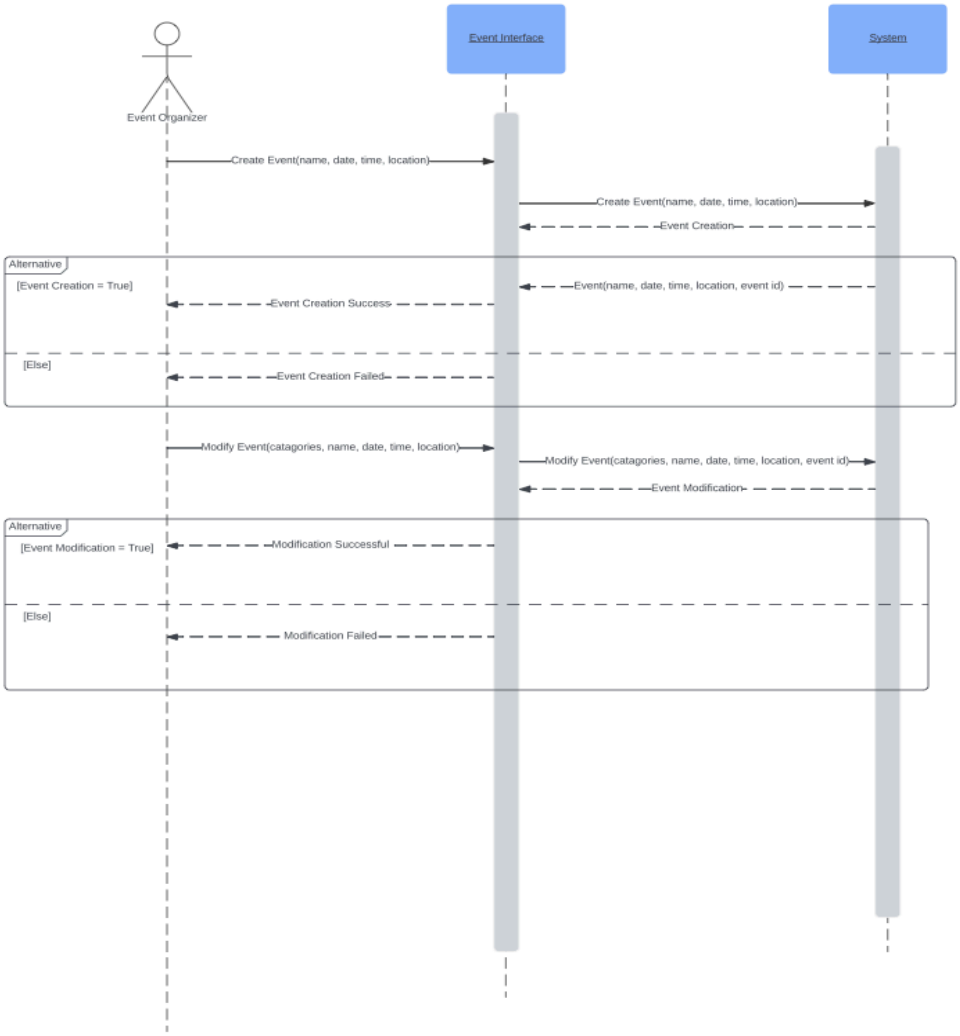
Space Requirements: At least 500.555MB daily storage requirement for event data, user profiles, and multimedia content.

Usability Requirements: A user-friendly interface to end hance user satisfaction and engagement

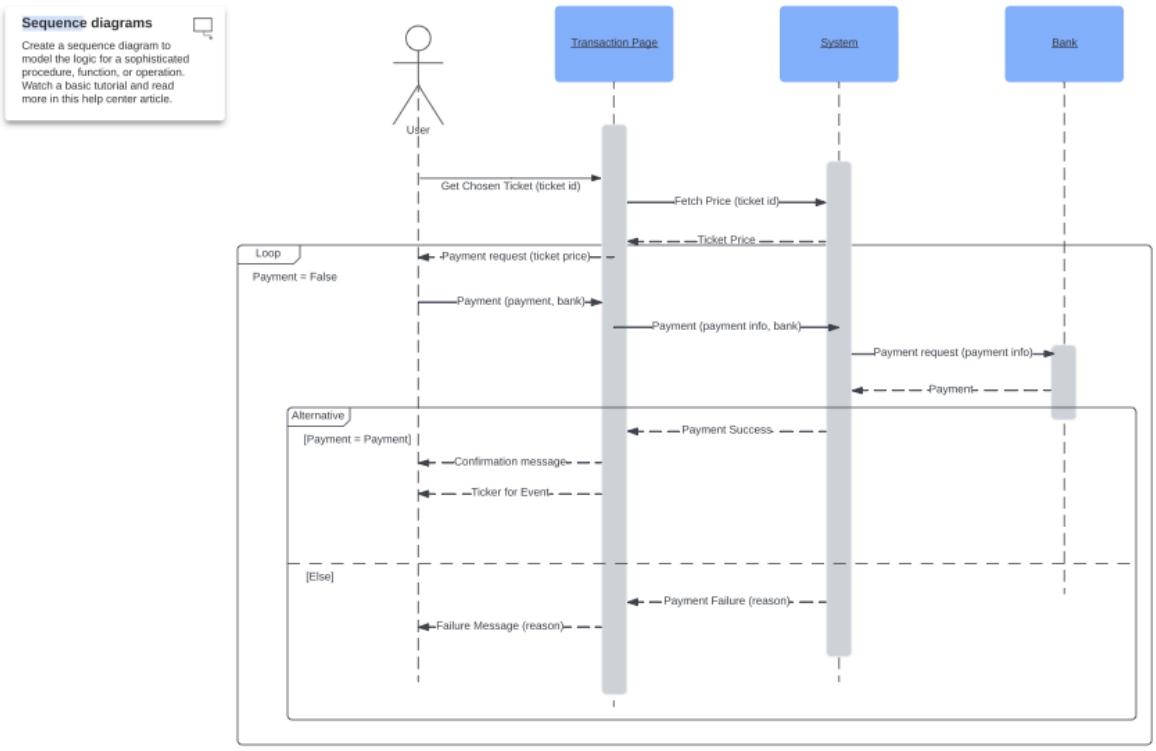
8. Use Case diagram

9. Sequence diagram

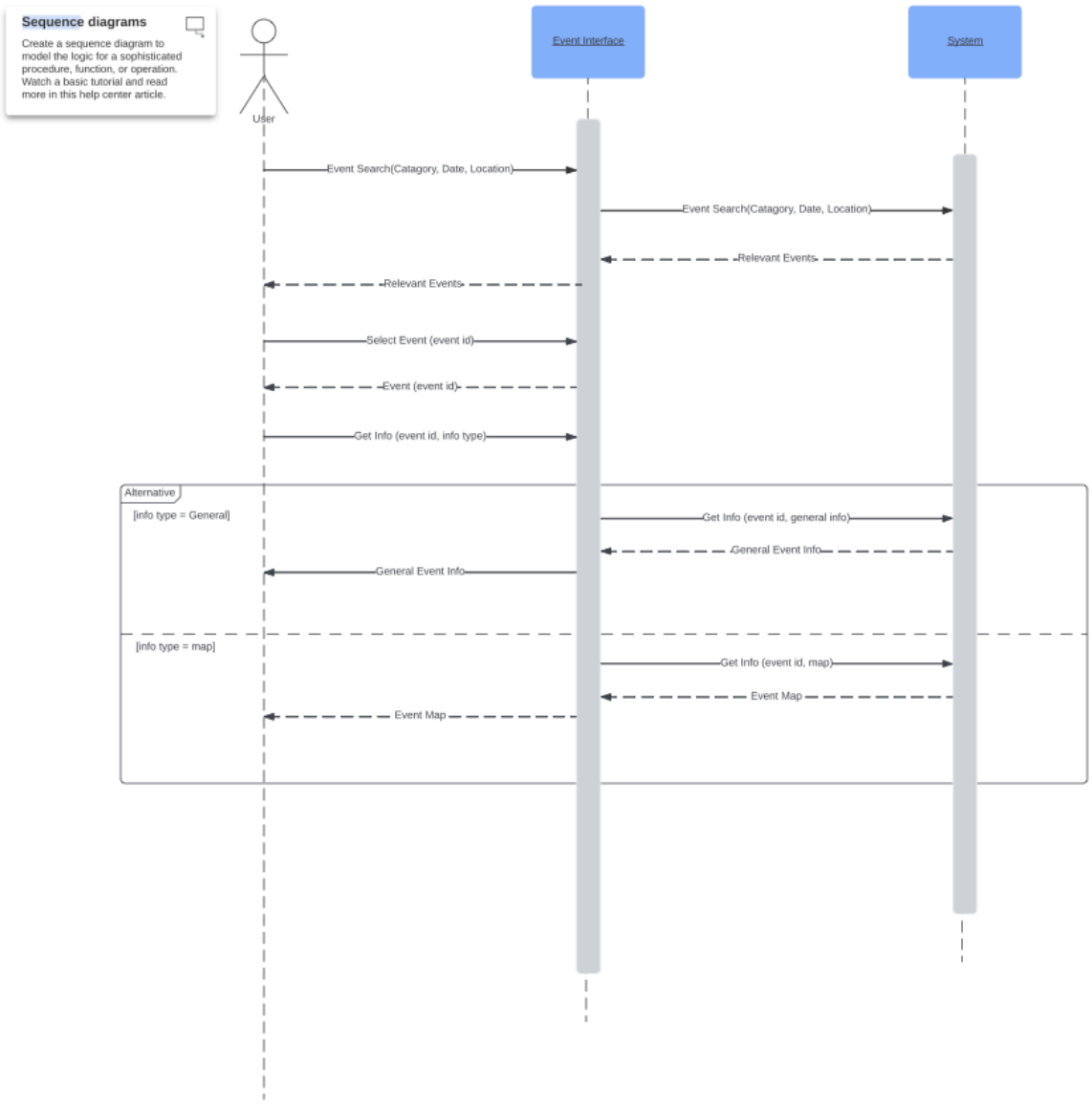
Sequence Diagram 1.1



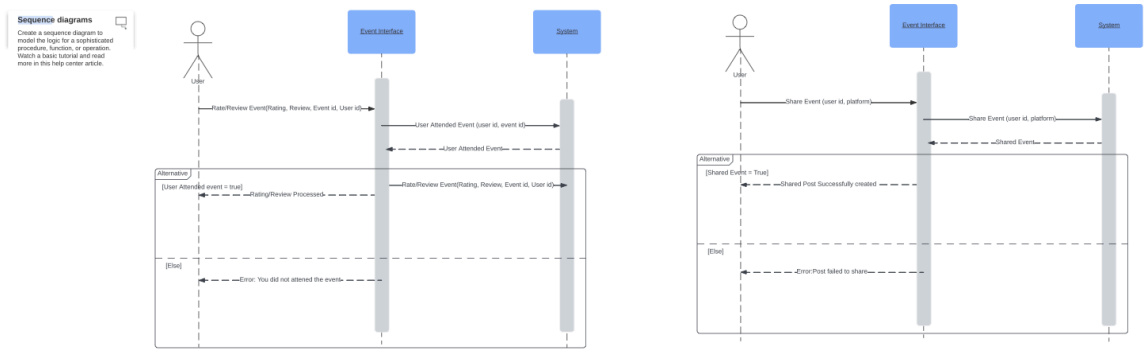
Sequence Diagram 1.2 and 3.1



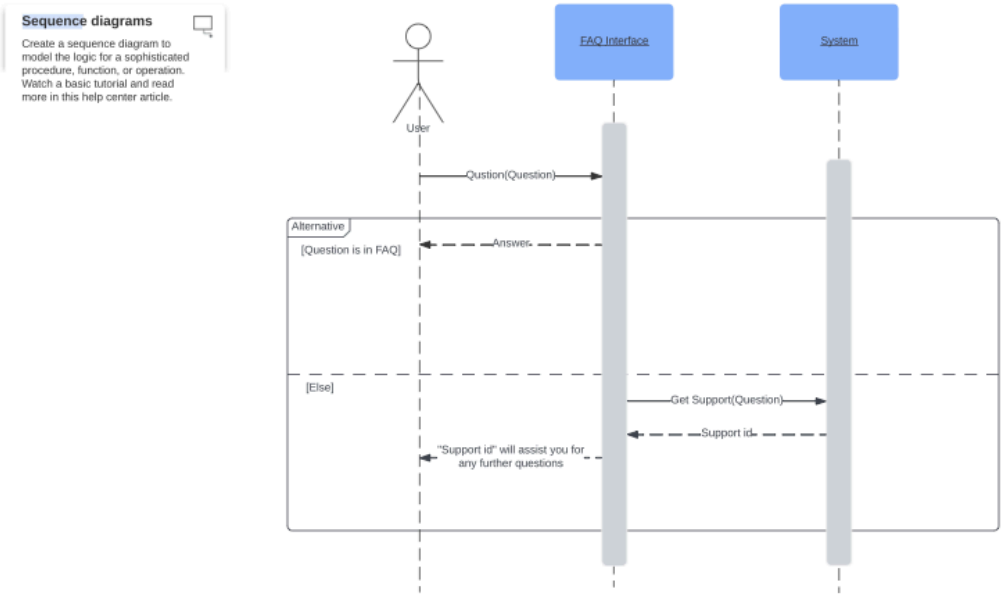
Sequence Diagram 2.1 and 2.2



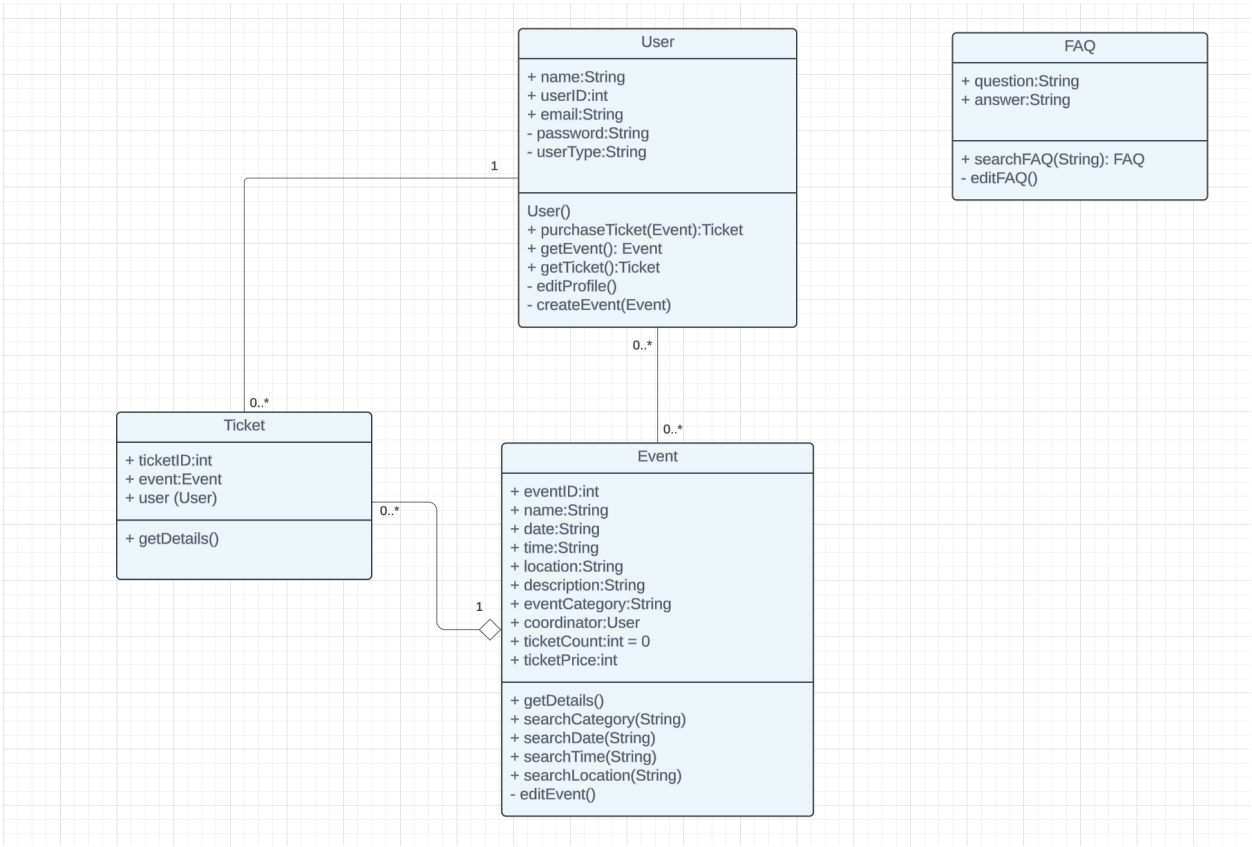
Sequence Diagram 3.2



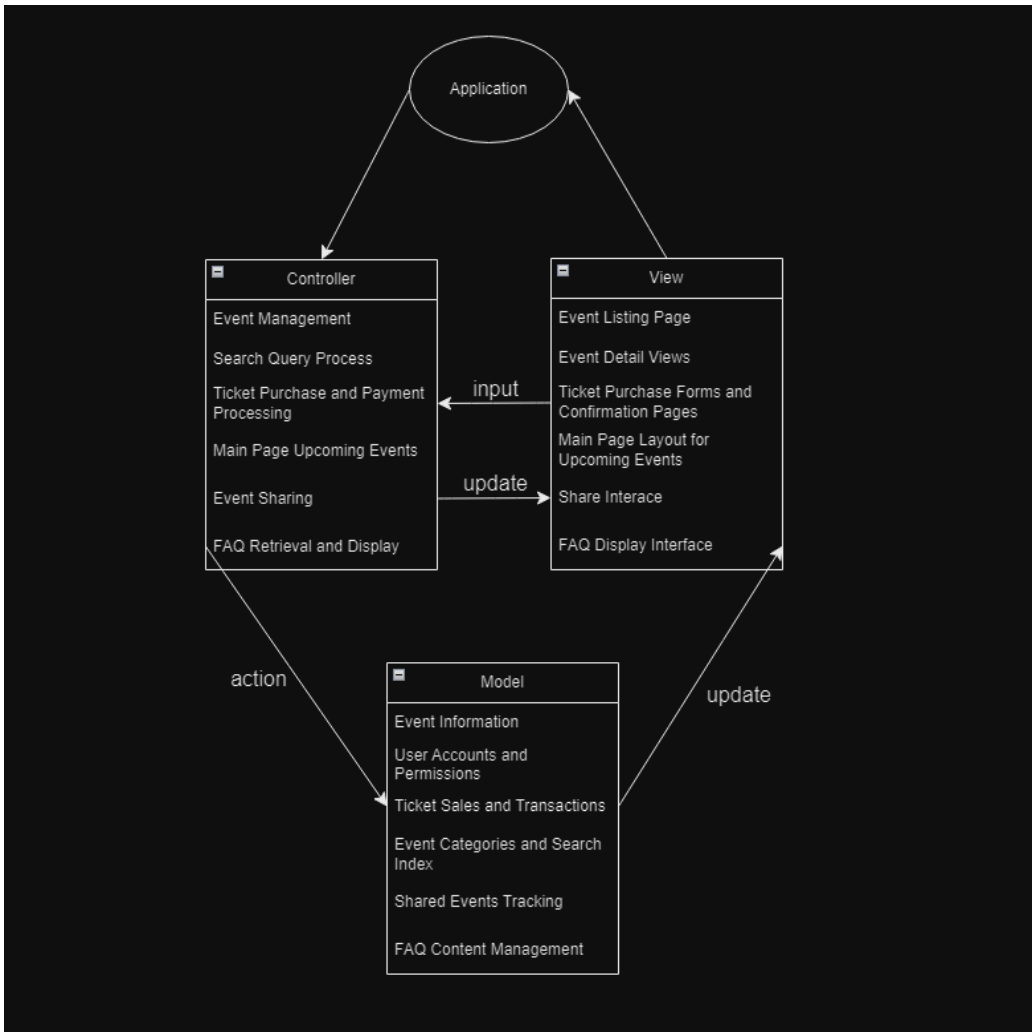
Sequence Diagram 4.1



10. Class diagram



11. Architectural design - MVC pattern



3. Project Scheduling, Cost, Effort/Pricing Estimation, Project duration, staffing

3.1 Cost, Effort, and Pricing Estimation

We decided to use Function Point (FP) to estimate the cost and effort of this application.
EI: Add Event (average), Modify Event (simple), Purchase Ticket (complex) = 3+4+6 = 13 FP total.
EO: Create Ticket (simple), Send Confirmation (simple), Display Purchase Summary (simple) = 4*3 = 12 FP total.
EQ: FAQs (simple), Check Ticket/Seating Availability (simple), View Event Details (simple), Search Events (average) = 3*3 + 4 = 13 FP total.
ILF: User Account Information (average), Event Details (average), Transaction Information (average), Ticket Information (average) = 10*4 = 40 FP total.
ELF: Event Calendar Interface (Simple), Payment Gateway Interface (Average), Customer Database Interface (Average), Venue Information Interface (Average) = 5 + 7*3 = 26 FP total.
GFP = 13+12+13+40+26 = 104
PCA = 0.65+0.01*((3*4)+(11*3)) = 1.10
FP = 104*1.10 = 114.4
Assuming 5 FP per person per week
114.4/4 = 22.88 ~= 23 person weeks
Assuming a team size of 7
Project Duration
23/7 = 3 weeks and 2 days.

3.2 Project Scheduling

Start Date: June 3rd
End Date: July 2nd
The calculated time to complete this project was 3 weeks and 2 days. We decided to add an extra week to the schedule to give leeway for unexpected issues. Weekends are not counted in our schedule. The number of working hours per day is 8 hours per person.

3.3 Estimated Cost of Hardware Products

It is important to note that the app will host User Data and Transactional data. The app will also need reliable backup and redundancy controls. To meet hardware needs, it has been decided to use Oracle Cloud Infrastructure(OCI), so as to not actually host most hardware On-premise, which takes care of a majority of hardware needs. This incurs a cost, however. OCI offers Virtual machines support, Database automation(replaces Database Administrator in many tasks), and data recovery. Overall, if we keep the

package as basic as possible, this would come to a cost of about $3500 a month, but this easily goes up to $15000+ if more support options are checked.

3.4 Estimated Cost of Software Products

In general, most of the UI's and database programming can be done using free software. A techstack that includes software like Python/Django, HTML/CSS/JS (and their related frameworks); all of which are free, open-source software. The costs really start to arise when considering database hosting and the possible integration of third-parties for the management of billing. MySQL by Oracle is a good option for our database needs, bundled together with Heatwave to improve the query speeds and provide extra analytics; it has a cost of about $72 a month. Stripe is a third party service that provides APIs to process payments within our architecture. The most basic package, which covers all of the typical card types (like Visa, mastercard, etc…). In exchange, a cost of 2.9% + 30 cents per successful charge. Stripe offers a lot of options for UIs, along with accepting a great variety of payment options and great uptime, which could make it a powerful ally to develop our app.

3.5 Estimated Cost of Personnel

Developing this application would approximately take 5-10 developers to complete
Number of software developers- 7
Average software developer hourly salary (Tx)- $41.79
Daily pay (one software developer)- $41.79 * 8 hours = $334.32
Gross pay (one software developer)) - $334.32 * 23 days = $7689.36
Total gross pay- $53,825.52
Training cost after installation- $1250 per employee
Total training cost after installation- $8,750
Total estimated cost of personnel- $62,575.52

4. Software Test Plan

Description:

We will focus on testing the **purchaseTicket(Event event)** method found in the User class, as listed in the project's class diagram (section 10 of the project proposal deliverable). This method allows users of type 'attendee' to purchase a ticket for a valid event listed within the system.

This method is responsible for making sure the user is able to 'purchase' a ticket for a specific event upon request. The method should validate that the user is of type attendee, and should return an error message otherwise (to indicate that the ticket purchase was invalid). The method should also increment the ticketCount for the event

after a User purchases a ticket, and should link the ticket to the user by returning an object of the ‘Ticket’ type.

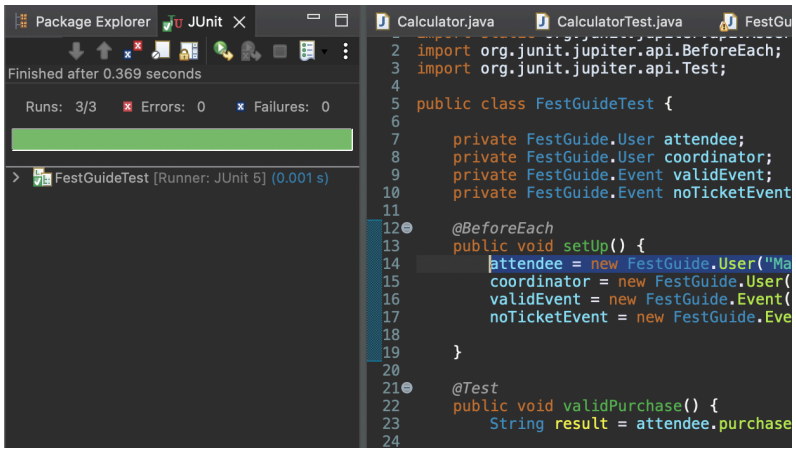
The test cases we will use are:

1. **Valid Ticket Purchase :**
    - Description: This test case simulates a successful ticket purchase scenario.
    - Input: A valid Event object with available tickets, and a valid User object of type ‘attendee’
    - Expected Outcome: ticketCount should be incremented by one in the Event object, TicketsAvailable should be decremented, and a message should return to confirm successful purchase.
2. **Insufficient Tickets:**
    - Description: This test case checks the behavior of a function when no remaining tickets are left.
    - Input: valid User object with ‘attendee’ type, Event object with ticketsAvailable = 0 value.
    - Expected Outcome: The method should return a error message to the user
3. **Invalid User Type:**
    - Description: This test case verifies that only user types of type ‘attendee’ are authorized to purchase tickets.
    - Input: invalid User object with userType value that is not ‘attendee’, valid Event object.
    - Expected Outcome: The method should return with a error message to the user.

Our testing method:

- We used **Java** as our programming language of choice for this test, and used **JUnit** as our testing automation framework.
- Our testing method involves giving in an input and expecting a specific output message (Either a successful purchase message, or an error message. This closely resembles **black-box testing, or functional testing** (functional testing verifies if the system's features behave as intended according to the requirements.)
- The test cases also test the changes of the Objects created in the program, such as changes to the Event object’s integer values. This overlaps with **white-box testing**.

- **Screenshot of 3 successful runs, (zip file included in submission)**



5. Comparison of Work with Similar Designs

FestGuide positions itself within the event management platform space by offering features that are comparable to some of the prominent names in the industry like Eventbrite, Explara, and Cvent. Similar to Eventbrite, FestGuide provides functionalities for event creation, promotion, and ticket sales but with added capabilities for enhanced customization and integration with external platforms for marketing and visibility. This positions FestGuide as not just a tool for ticketing but a more integrated solution for event organizers seeking to extend their reach across multiple digital platforms [3].

Explara offers a comprehensive suite that includes not just ticketing but also attendee engagement tools and post-event analytics, similar to what FestGuide aims to provide. However, FestGuide differentiates itself by focusing more on user-driven event management where attendees also have the power to update event details, offering a more collaborative approach compared to Explara's more centralized control model. FestGuide's emphasis on user engagement and data accuracy could appeal to a market segment that values community-driven event updates and participatory event shaping [4].

Cvent, known for its robust feature set catering to large-scale professional event planners, offers advanced analytics and vast integration options. In contrast, FestGuide may appeal more to community-driven and smaller scale event organizers due to its more accessible interface and simpler setup process. While Cvent focuses on delivering a high-powered, comprehensive suite of tools for event professionals, FestGuide might carve out a niche among users who require a straightforward, flexible platform that supports active participant involvement in both the planning and the promotional phases of event management [5].

6. Conclusion

Our planning process ensures that we can execute this project successfully, by using function point analysis ensures that we can put in enough resources into this project, we also added another week to give us sufficient breathing room in case of unforeseen circumstances or delays.

We have done a thorough estimation of personnel costs including salaries, training expenses, and other relevant factors. Our testing plan ensures that every aspect of the project runs smoothly, especially the ticket purchasing function. The use of Junit increases efficiency and reliability of the testing procedures.

In comparison with similar apps, FestGuide has the upper hand in the event management platform market, highlighting its strengths in user driven event management and community engagement. In the future we will be including more quantitative data or customer engagement to boost its persuasiveness.

Overall, as a student team there is room for improvement in area regarding cost breakdowns and comparative analysis. Going forward we will have continual monitoring to ensure this app is a success.

7. References

[1] Software engineer salary in Texas, https://www.indeed.com/career/software-engineer/salaries/TX (accessed Apr. 19, 2024).

[2] Classiebit.com, “The Ultimate Guide to Event Management Software Cost: Classiebit Software,” classiebit.com, https://classiebit.com/blogs/understanding-the-operational-costs-of-event-ticketing-software-a-breakdown (accessed Apr. 18, 2024).

[3] L. Davey, “The Best Event Management Software: 10 Apps to Plan and Manage Your Business Events,” Zapier, https://zapier.com/blog/best-event-management-software/. (accessed Apr. 19, 2024).

[4] “16 Best Event Management Software,” Scoro, https://www.scoro.com/blog/best-event-management-software/. (accessed Apr. 19, 2024).

[5] “Event Management Platform Comparison,” Cvent, https://www.cvent.com/en/event-management-software (accessed Apr. 19, 2024).

8. Presentation Slides

Attached in submission.

9. Optional Implementation

(chose to not do)

10. GitHub Requirement

Attached in submission.