Final Project Report
Event Attendance Management System

SEG 2105[A] - Intro To Software Engineering
Fall 2024
University of Ottawa

Course Coordinator:
Hussein Al Osman

Teaching assistants:
Ammar Rashed - arasi005@uottawa.ca
Hassan Awad - hawad104@uottawa.ca
Subhashri Mohan - smoha300@uottawa.ca
Nima Meghdadi - nmegh082@uottawa.ca

Group: 6

Rebecca Beagley: 300380734
Maeve Joys Evans: 300367434
Brooke MacQuarrie: 300350709
Christina Young Pow:300370843

Submission date: 04/12/2024
Pages: 10

Introduction

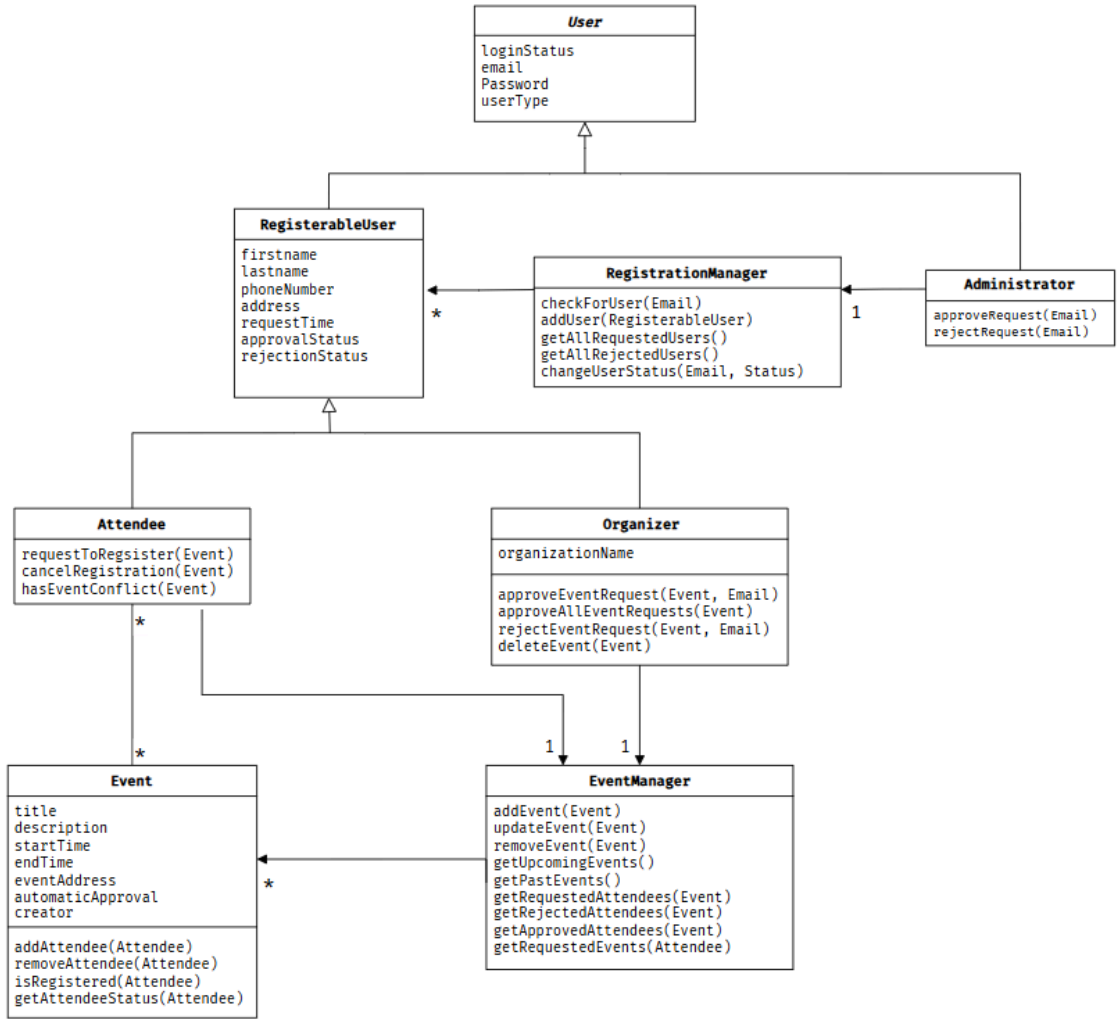
The last few months have been spent creating, fine tuning, and testing the Event Attendance Management System for SEG2105[A]. The completion of this project has resulted in both the growth of the application and a significant increase of experience in software development. There were four deliverables that were completed, each focusing on a different part of the project. This allowed for pre-existing knowledge and techniques to be applied, while simultaneously introducing new concepts and skills essential for bringing the application to life. Although there were many moments of confusion, the development process prompted many meaningful learning opportunities.

During each of the deliverables, the design process began with a group meeting where the requirements for the deliverable were divided up and expanded upon in terms of implementation ideas and strategies. Any questions that arose were clarified. This process was completed in the beginning of each deliverable period, allowing adequate time for submission before the deadline. After this initial meeting, each team member completed their tasks up until the deadline, asking others for help when necessary. By the end of the fourth deliverable the application was in full working order.

This report will cover the structure of the application and the operations of the team. This includes UML diagrams that were drafted to plan out the structure and interactions, the organized division of work, screenshots of the final working application, and finally, elaboration upon the many lessons learned. These elements will be detailed in full to provide an outline from start to finish of Group 6’s developmental process.

UML Class Diagram

This is the system domain model UML diagram for the Event Attendance Management System application. It demonstrates how the user hierarchy operates and how the users interact with the system.



Contribution Tables

Here is a link to the team's project documentation where we kept track of contributions during the project creation:
https://docs.google.com/spreadsheets/d/11OeuiUcL1EEY7I-cwVr_xk0qHBf3KwypaoFXJuQErBI/edit?usp=sharing

Below are the tables with the summaries of each group member's contributions for each of the four deliverables.

Deliverable 1:

Rebecca	• Github classroom setup • Account Creation and signup page UI
Maeve	• Demo video recording and submission • Welcome Messages • Log off button and functionality • Login Page UI
Brooke	• UML Class Diagram • APK Submission • Firebase integration and DatabaseManager Class
Christina	• Input utilities for field validation for the signup page and login page

Deliverable 2:

Rebecca	• APK Submission • Demo video recording and submission • rejectRequest method for the administrator • Login failure popups
Maeve	• approveRequest method for the administrator
Brooke	• UML class diagram • Requested/Rejected Lists for administrator page UI • Registration manager class to handle rejections and requests in database
Christina	• Reject and Accept buttons for the administrator page UI • User information subpage UI

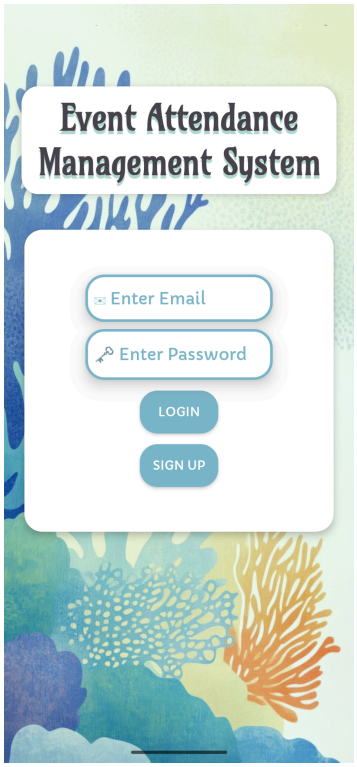
Deliverable 3:

Rebecca	• Date validation • Event manager class to handle events and attendees registered to them in the database
Maeve	• Create event page UI • Date and time inputs for event creation • Create event field validation
Brooke	• UML class diagram • APK Submission and demo video submission • List of upcoming and past events • CircleCI integration and test units
Christina	• Organizer approve and reject event request methods • Delete event method • Approve all event request methods • Event class

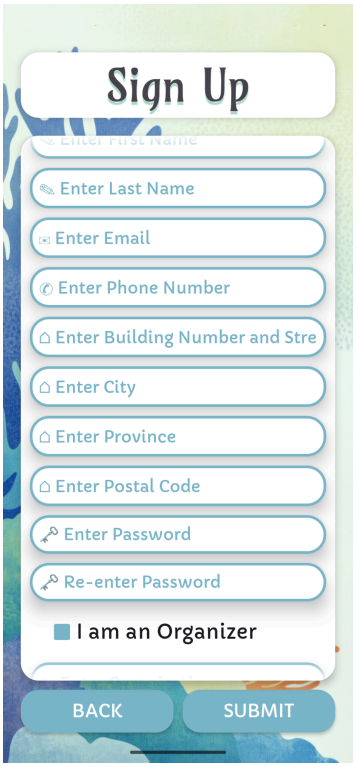
Deliverable 4:

Rebecca	• Final report introduction • Ability to search for events by title and description • Attendee event reminder notifications
Maeve	• requestToRegister method for Attendee class • cancelRegistration method for Attendee class • event conflict detection
Brooke	• UML class diagram • Updated whole application UI to look nicer • Attendee available and requested events
Christina	• Subpage to view event information • Subpage to view attendee information for organizers • Demo video recording and submission

App Screenshots



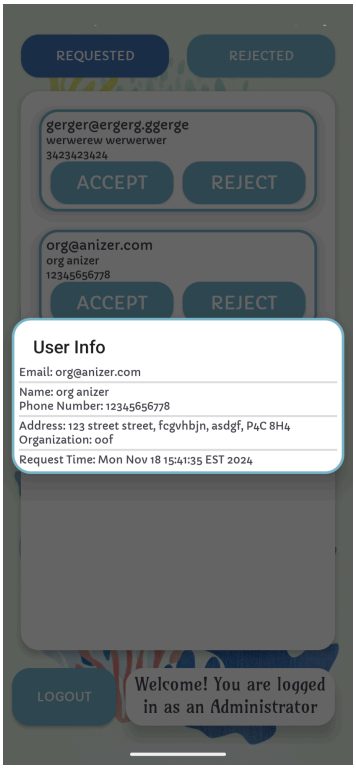
Login Page



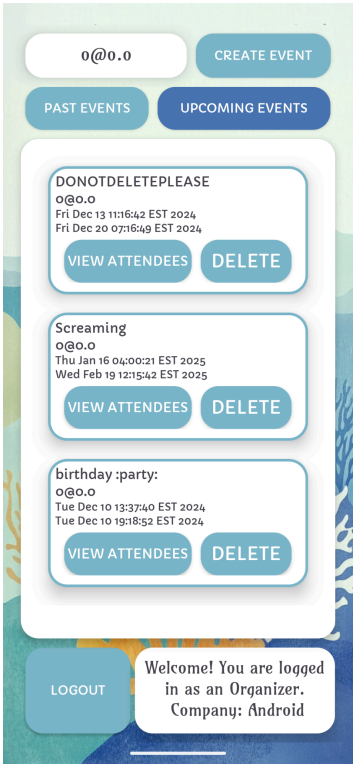
Sign Up Page

Administrator Requested Page

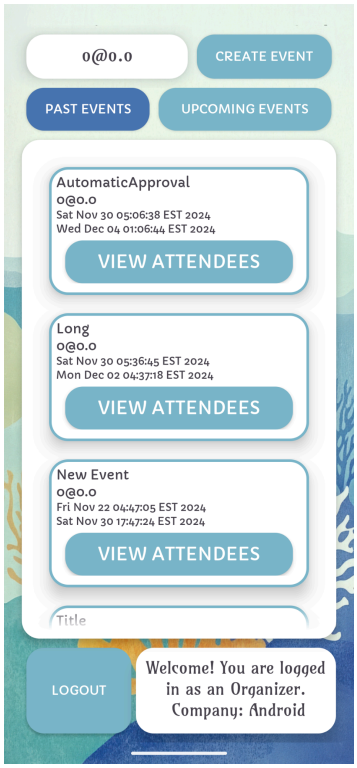
Administrator Rejected Page



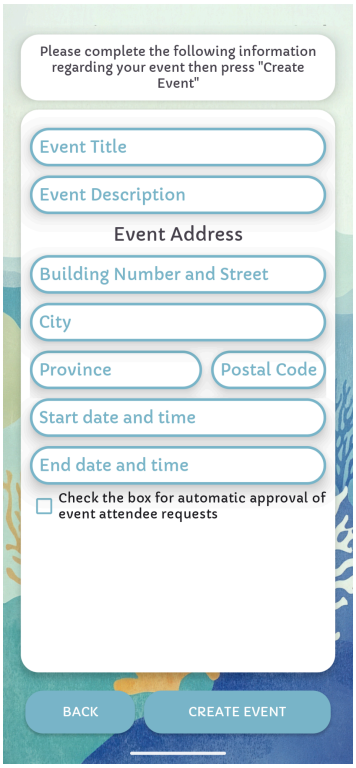
User Info Subpage



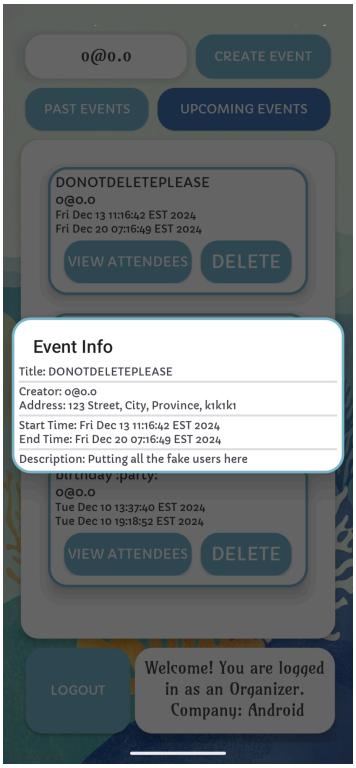
Organizer Upcoming Events



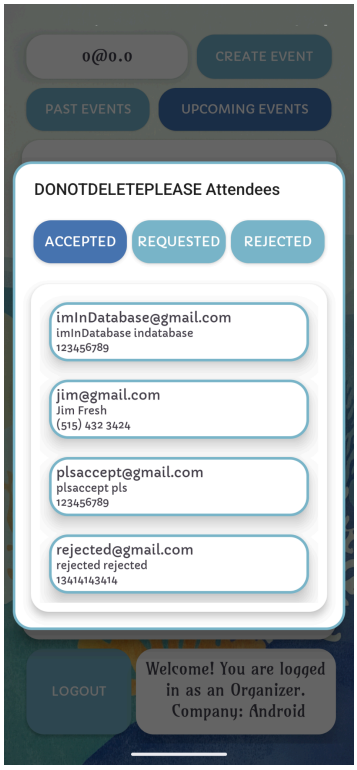
Organizer Past Events



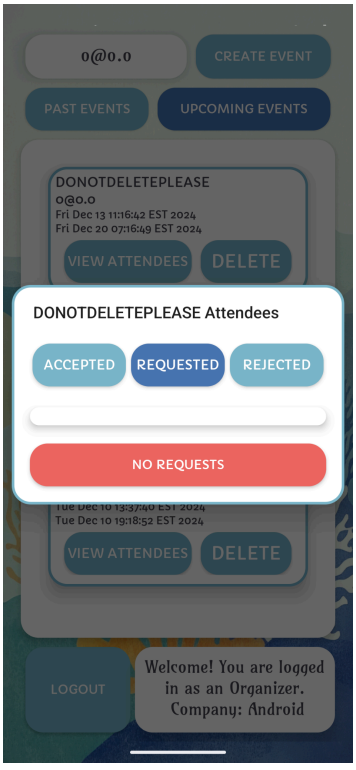
Create Event Page



Event Info Subpage

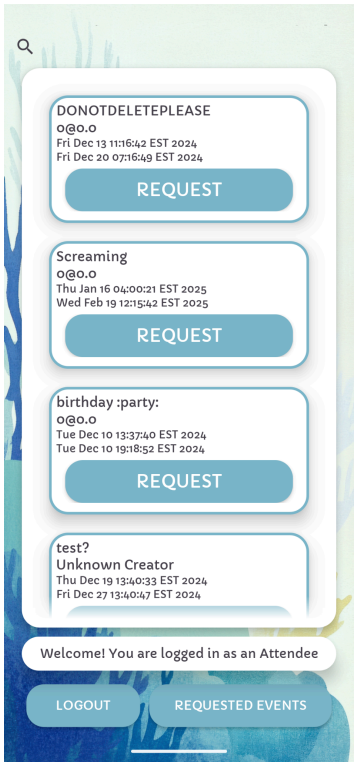


Accepted Attendees Subpage

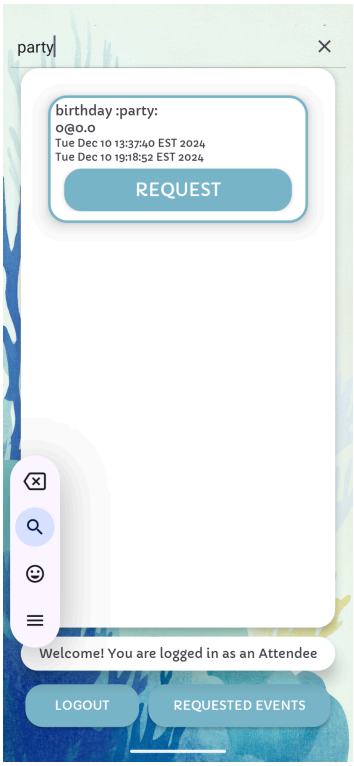


Empty Request/Reject Subpage

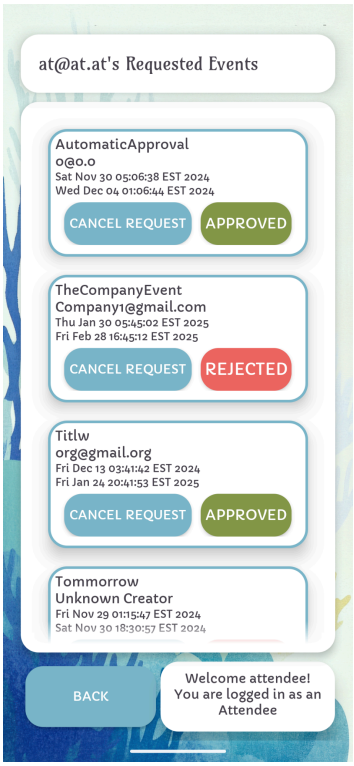
Request/Reject Subpage



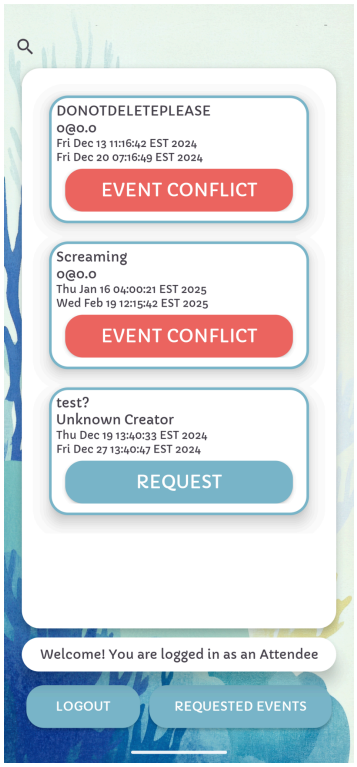
Attendee Available Events



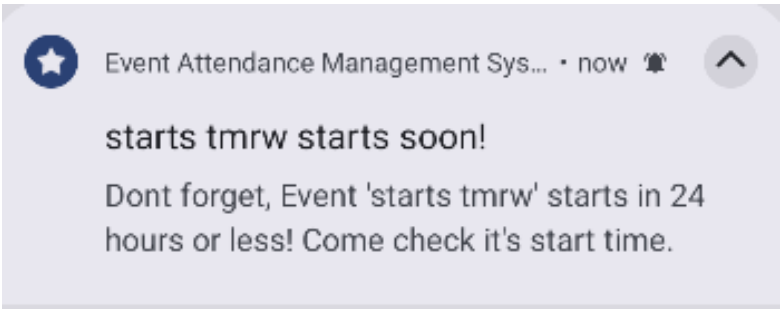
Search Available Events



Requested Events Page



Event Conflict Notifications



Upcoming Event Push Notifications

Lessons Learned

Organized Software Development & Planning
This project was a journey from start to finish. From the very beginning, we decided to take an organized approach by dividing up the tasks and working together as a group to get things done. Overall, it introduced us to Git and its many functions, taught us the importance of clear communication, and of course, stressed the importance of proper time management. Inevitably, we quickly ran into issues as it was our first time working on a bigger project as a group. Although there were many roadblocks related to these things, each problem was taken into consideration right after, ensuring that we could prevent the issues from occurring again. Our understanding of Git improved, we clearly laid out tasks and were not afraid to ask questions when confused, and finally, set early deadlines to prevent last minute issues and stress. Overall, these software development learning experiences will be carried with us to future projects.

Debugging Techniques
Running into errors was inevitable. But, since this was our first bigger project involving many different classes, it was often difficult to identify the cause of the issue. Android Studio's Logcat was particularly useful in these situations. Whenever the program ran, Logcat stored messages documenting warnings, errors, or simply just general information about the process. Most of these were automatic, thus whenever a gradle build or internet related issue would occur, the error would be highlighted and described in red, making it much easier to identify and fix the issue. Additionally, by incorporating 'Log' into our code, we could add additional Log statements that appear in the Logcat. This way, if there's a logic error, the steps taken by the code can be followed and through further analysis of the specific log messages describing what is going on behind the scenes, the issue can be solved.

Incorporation of Design Patterns
As class went on and design patterns were introduced, things began to click in our heads about ways to incorporate them into our code, improving our pre-existing implementations. One example that comes to mind is the use of a singleton to store the user information one time, rather than repeatedly calling on the database from different pages to access the information. Overall, having an opportunity to incorporate design patterns into code really is different from just learning about them, thus this was an important experience.

Using APIs
Up until this point, most in-class programming assignments have not included any sort of API integration, so this app was our introduction to this whole new world. We learned how to use Android's API to bring our application to life visually, using activities, adapters, and more. Additionally, in the backend, we learned how to use Firebase's API to create a database for all

user and event information. The learning process prompted us to consult other resources, often teaching ourselves how to bring our ideas to life. Ultimately, this introduction to API integration will surely follow us into our future projects to come.

Integrated Testing

By incorporating continuous integration into our project repository, each commit automatically ran tests that we created. This made it easier to catch an accidental change that resulted in an error, rather than having this small change going unnoticed, causing issues later on. Additionally, this was a lot more time efficient than going through the application and manually testing every aspect each and every time. Although manual testing was important and thoroughly done, this addition did help.