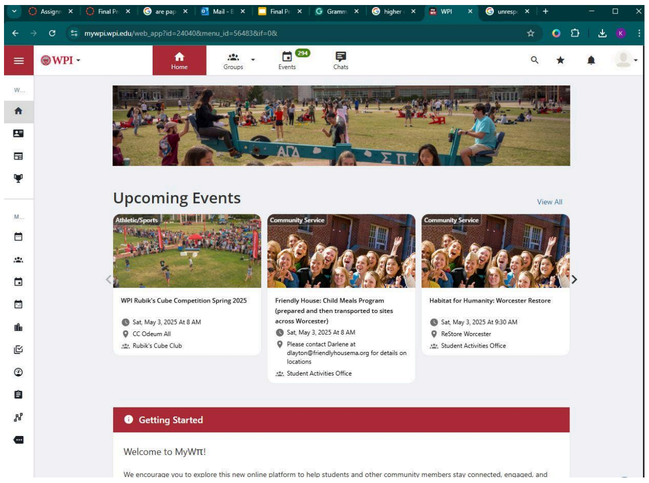
WPI Group Finder App - Final Project CS 528

Edward Fantasia, Kathryn Butziger, Nick Heineman, Omkar Tikar
Worcester Polytechnic Institute

ABSTRACT
In this paper, we describe our motivation and processes for creating our Android application "WPI Group Finder," which fulfills the requirements of our CS 528 class. Our app solves a real problem Worcester Polytechnic Institute (WPI) students face each day and demonstrates our knowledge of mobile development through the usage of devices and sensors on the phone, various modules and libraries, and third-party development tools.

1. INTRODUCTION
For undergraduate and graduate students in any higher education program, maintaining a social life during their studies is often critical to their mental health. Many programs of study can be stressful, demanding, and time-consuming. Without extracurriculars and other opportunities to connect with peers outside of the classroom, students may find themselves burnt out, isolated, and lonely. Outside the academic setting, many activities, clubs, and organizations exist to help students de-stress and feel connected to their campus community. However, with the number of clubs and events on campus, it can be difficult for students to find and keep track of the ones that interest them. WPI has an online tool, MyWPI[1], to solve this problem. MyWPI allows students to find clubs, events, and connect with other students. However, many students shy away from using MyWPI because the interface is overly-complicated and it generally has a slow response time. For our final group project, we wanted to create an app that both keeps students informed about campus activities and addresses the shortcomings of the existing MyWPI system. We hope our work can empower and enable our fellow students to try new activities on campus and form new connections.

2. RELATED WORK
MyWPI was our initial frame of reference when planning for our app and what it should accomplish. MyWPI is a secure website only accessible to WPI students and club advisors. After logging in, it gives the user access to one of its key features, a feed of events happening on campus. Users may search through events by specifying certain fields such as the group that is hosting the event, the type of group, the category of the event, and the event type. Once the user has found an event of interest, they may begin to register for an event. On the registration page they can see who is hosting the event, the other attendees, the location of the event, the time of the event, and any other details of the event. There is an interactable map displaying the location of the event. If they register, they are added to the list of attendees and are visible to the event or club organizer. Another key feature of MyWPI is the "Groups" tab. Here, users may search through all groups and organizations on campus. There is a text input field where they can search for a specific group by name, or they can casually browse through all groups. Groups can also be searched by category. A user can click on a club to pull up its home page. The group's home page includes the group leaders, other members, and past events. Users can send a request to join a group, which is then sent to a group leader for approval. There is also a chat feature on the website where a user can message each other, usually for event or group membership questions. Overall, the aim of this website is simply to allow students to find clubs and stay informed on the latest events. However, as shown in Figure 1, its interface is crowded. As a user it is not immediately obvious how to register for a group or sign up for a specific kind of event, which can be annoying and overwhelming. Additionally, when an event is clicked on, the response time can be very slow. These factors often discourage students from checking MyWPI, so they use other methods of communication to send out club meeting times and other information.



WPI Rubik's Cube Competition Spring 2025
Sat, May 3, 2025 At 8 AM
CC Odeum All
Rubik's Cube Club

Friendly House: Child Meals Program (prepared and then transported to sites across Worcester)
Sat, May 3, 2025 At 8 AM
Please contact Darlene at dlayton@friendlyhousema.org for details on locations
Student Activities Office

Habitat for Humanity: Worcester Restore
Sat, May 3, 2025 At 9:30 AM
ReStore Worcester
Student Activities Office

Welcome to MyWπ!

We encourage you to explore this new online platform to help students and other community members stay connected, engaged, and

Figure 1: MyWPI Frontend

3. SMARTPHONE FEATURES
3.1 Facial Detection
Our app includes a verification process for signing up and creating an account. Before official sign up, the user is prompted to take a photo of their face which is then scanned by ML Kit[2] face detection to determine if there is at least one face in the image. It will visually display the box of the user location and prominent facial features as proof of success. If the verification fails, the user

[1] MyWPI. https://mywpi.wpi.edu/home_login

[2] ML Kit. https://developers.google.com/ml-kit

will be prompted to either redo the photo, or go back to the login screen if they cannot complete the requirement.

3.2 Step Counter

Our app includes a step counter as a bonus function. By connecting to Android's SensorManager Application Programming Interface (API)[3] and step counter built into the Android device we are able to obtain and display the user's steps for that day on their user page. The step counter is a live feed of the steps with minimal delay. Many students are sedentary because they spend most of their day doing work. The step counter will encourage students to move if they open the app and see a low step count. On the other hand, by walking around campus and participating in events, students' step counts will be higher, and it is rewarding to see the high step count number as a result. This is a way of showing the student that they've been active (or inactive) on campus today.

3.3 Map and Location

Our app includes an interactable map of the user's current location, in a similar style to MyWPI, and the user's current address. This is accomplished through the use of Google Maps API[4], for the map, and Geocoder API[5], to determine the user's location. Google Maps API uses location permissions on the phone to determine the user's location and display it as a red pin on the map. Additionally, it returns the user's latitude and longitude, which are used by the Geocoder API to determine their address. Both Map and Location are displayed on the same page to help the user determine where they are. Students can use this if they get lost on campus or if they need to find where a club or event is meeting.

4. IMPLEMENTATION

4.1 Implementation Plan

When first starting development on our app, we decided to create a schedule for when different pieces of implementation should be complete. We decided that the first steps in development should be to establish a code repository on GitHub and a cloud-based relational MySQL database on Amazon Web Services (AWS)[6]. The remote repository was crucial as it is important for all team members to have access to the most recent as well as and all previous versions of the code base. We decided to use GitHub due to all team members having prior experience using GitHub, its industry-wide popularity. We also required a database to hold data pertaining to our application, such as users, events, clubs, etc while also being easily accessible to all team members. We decided to use AWS due to most team members having prior experience using AWS which would lead to smoother development. We also decided to create a development schedule of sprints each week for four weeks to ensure development would be as fluid as possible with plenty of time to work out bugs within the code. Our team decided a schedule of four weekly sprints would be best to allow for plenty of time for implementation of all required functionality.

4.2 Database Implementation

As discussed in the previous section, we used AWS to create our cloud-based relational database due to most team members having prior experience with the platform. AWS works by creating functions called lambda functions that connect to the database connected with said lambda and using SQL statements to create, update, or delete table rows. These lambda functions are then connected to API endpoints which are then accessed with request bodies from our application to retrieve necessary data to display within our application. When designing our database, we first started with a diagram of all tables we would need to finish our application. Once done with said diagram, we came to the agreement that the tables we needed to properly store relational data include the following: user, club, event, membership, and registration. With the users table, users can be created and signed in on the front end with lambda functions in AWS creating said users and editing them if needed. The club table records clubs created by users. The event table records events created by users and club leaders and whether the event is associated with a club or not. The membership table records what clubs users have joined and whether or not they are active. Finally, the registration table is used to record what events users have joined. A visual representation of the database is seen in Figure 2.

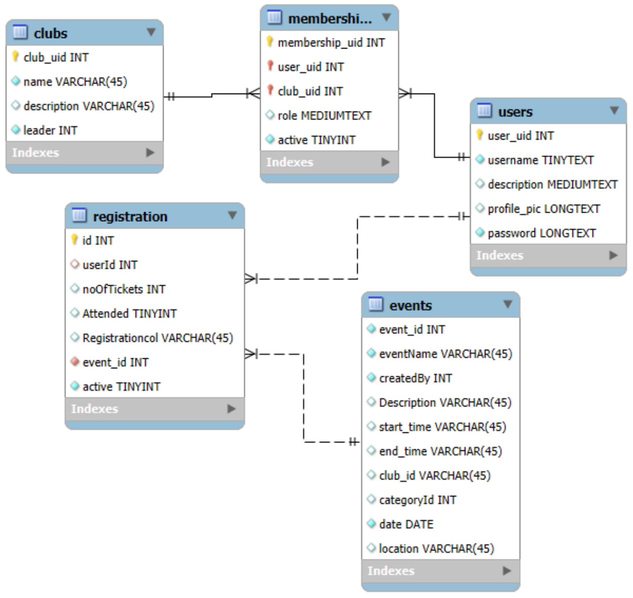


Figure 2: WPI Group Finder Database Structure

4.3 Frontend Structure

The structure of the project was important to plan and stick with from the start, to make sure even with multiple people making changes the system would make sense as we progressed. Thus we built the project around the NavController[7] framework. The "MainActivity" class in Kotlin was used purely as a place to track all the paths between pages in the system. Then, for each distinct

[3] SensorManagerAPI. https://developer.android.com/reference/android/hardware/SensorManager

[4] Google Maps API. https://developers.google.com/maps

[5] Geocoding API. https://developers.google.com/maps/documentation/geocoding/overview

[6] Amazon Web Services. https://aws.amazon.com/

[7] NavController. https://developer.android.com/guide/navigation/navcontroller

screen in the app, we created a new page. This resulted in varying sizes of page, but we decided that it made the most sense for a project with this many screens (10+ ultimately). After adding each screen, we would simply add a path into NavController with a keyword that would allow other screens to navigate to it. This design structure proved very effective because it is very git friendly. Thus, even though we had four people working on different pages at different times, we almost never had any merge conflicts, because we could keep each section in its proper place and avoid overwriting each other's code. A visual representation of the frontend structure can be seen in Figure 3.

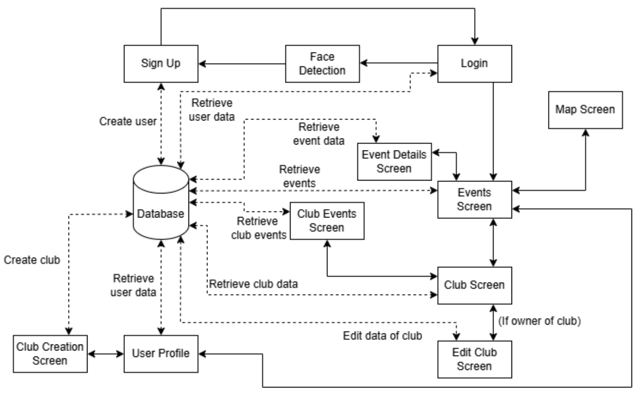


Figure 3: WPI Group Finder Frontend Structure

4.4 Functionality Summary
Users create an account which is stored with the password encrypted in the database and can then be logged into at a later date. Users can then create a club or an event. Clubs can be created by anyone, and only club leaders can associate their created event with a club they are a leader of. All users can create events, and, as mentioned before, only club leaders can associate an event with their club. Created events are then displayed in an event feed, where users can press on an event that has not started yet and register for said event. Users can also click on the club associated with the event and join said club, where they can view the club profile by clicking on the club in their joined clubs section.

4.5 Testing
To test WPI Group Finder, we used manual unit testing. In order to unit test, we individually tested components of the app in "sections" of app functionality before creating connections between said sections. For example, we had "sections" such as user functionality and event functionality. Once we finished both sections of functionality, we connected them within the frontend and after testing for bugs within each "section." This same methodology was used when testing lambda functions, where once lambda functions for a "section" were complete, we would test for any errors in said lambda functions before staging the API within AWS.

5. FUTURE OF WPI GROUP FINDER
Although this app includes all functionality we desired, there are features we would like to implement and support if we continue to develop this application.

5.1 Implement S3 Bucket Functionality
With the implementation of AWS S3 bucket[8], we would be able to support custom user profile pictures as well as custom club profile pictures. This would allow for more user personality to be expressed within the application. We utilized S3 bucket for a temporary profile picture but due to time constraints we were only able to support the temporary profile picture.

5.2 Utilize MyWPI API For Event Creation
Early on in the project's lifespan, we were given the idea by Professor Agu to utilize the MyWPI API to populate the events table within our database. This sounded like a great idea and we all desired to implement it into our project, however, we were unable to fully realize this idea due to time constraints. In the future, however, this would be great to implement to keep our application competitive with MyWPI.

6. ACKNOWLEDGMENTS
We would like to thank Professor Agu for his teachings and support throughout the course, Mr. Inekwe for his role as teaching assistant, and the institution of Worcester Polytechnic Institute for providing the opportunity to take this course.

7. REFERENCES
[1] Worcester Polytechnic Institute. MyWPI. https://mywpi.wpi.edu/home_login.

[2] Google, Inc. ML Kit. https://developers.google.com/ml-kit.

[3] Google, Inc. SensorManagerAPI. https://developer.android.com/reference/android/hardware/SensorManager.

[4] Google, Inc. Google Maps API. https://developers.google.com/maps.

[5] Google. Geocoding API. https://developers.google.com/maps/documentation/geocoding/overview

[6] Amazon, Inc. Amazon Web Services. https://aws.amazon.com/

[7] Google, Inc. NavController. https://developer.android.com/guide/navigation/navcontroller

[8] Amazon, Inc. Amazon Web Services S3 Bucket. https://aws.amazon.com/s3/

[8] Amazon Web Services S3 Bucket. https://aws.amazon.com/s3/