To avoid the archaic time consuming check-in and group management methods commonly used today, we’ve developed an application which streamlines the process, whilst providing additional organization and security. To provide our users with a custom and safe experience, all individuals who utilize our app must register directly with our application or using a common social media account. Rather than managing separate social media authentication responses ourselves, we are leveraging Firebase Cloud Message, which will authenticate users using any federated identity providers, and return a single user authentication token, known as a JSON Web Token (JWT), which we send to our database using an HTTPS request from the client. Once a user has attempted a sign-in we verify the user authentication token in our backend, to ensure it is up-to-date and corresponds to an existing user in our system. Our backend is currently hosted on the provided KU MySQL database.

Following backend authentication, users will be directed to their dashboard which allows for immediate access to recent post and check-ins for groups they are associated with. Users can instantly check-in for active role calls that are visible on the dashboard, or may acquire additional information by selecting a desired notification. Selecting a check-in notification will open an overlay which will provide a description set by a group administrator, a pinned location on a map, and a check-in button – if the user is within the specified range. We feel that providing a map location allows for clear proximity visualization, allowing users to more easily recognize how close they are to any given check-in location. Mapping is done using the Google Maps API which is a part of Google Play Services. This comes with added pictorial flexibility and innate Google Maps features, such as pinning locations, adding overlays to maps, and providing navigation. Additionally, we seek to provide group administrators with an added sense security allowing them to check their members’ location, given that they are within the specified radius of the check-in point, and that the member has provided permission for their location to be visible. We feel this addresses use cases such as class fieldtrips, where groups may be on the move or in large spaces, where line of sight may be obscured to group members.

A key feature that our app introduces to the market is the idea of a mobile check-in. Many existing location based pinning or check-in services fail to address cases like the one above. By utilizing the Google Location Services API we provide administrators the option to bind the check-in location directly to their device, in addition to providing static location based check-in. While creating a check-in event admins will also be given the ability to specify a check-in radius as well as an event description, which are both visible to the member, as mentioned previously. Upon finalizing the event the client app will send the event information upstream, via an HTTPS request, to a PHP script running on our backend server. All appropriate changes will be made to database, at which point a push notification will be initiated to notify all group members of the check-in event. Push notifications will be completed using a call to Firebase’s REST API from the server, providing the event description, radius, check-in type, check-in location, and all Instance IDs for group members. An Instance ID is simply an identifier used by Firebase to differentiate each individual instance of our app. The Instance ID is sent and verified when the user signs into the app, at the same time that we send the JSON Web Token. To clarify, the JWT corresponds to an individual user and is used to verify that specific user whenever and wherever they login, regardless of device. The Instance ID on the other hand specifically correlates to an instance of our app running on a single device. Once a push notification has been received by the client, the appropriate aforementioned check-in activity will be opened. Administrators are then able to view who has and has not checked in, as well as a running total of check ins for the group.

For easy group management we have provided a ‘Groups’ page which is accessible form the main dashboard. The groups page contains all groups that a particular user is involved in, differentiating them by admin/member status. Selecting a group will provide the user with a list of events, similar to the main dashboard, however only containing events from the selected group. Additionally, a settings menu will be provided allowing a user to exit the group, modify their visibility settings, and initiate an event, if they are an admin. We hope to provide users with an extra level of customization and security if they wish for their location to never be shared to the group administrators. Furthermore, location information is only shared to group administrators given that the member has checked in to the check-in event in question and is within the specified radius. These circumstances ensure that location information is only sent under conditions in which the user would want their location information visible. In reference to the app as a whole, when it comes to accessing private information, mainly user location, permission checks will be completed at runtime, as is standard in Android Application Development. Requests cannot be completed unless the user is currently providing the necessary permissions, which they can modify at any point within Android Application Settings.