Use Case Stories

Our use case stories are on GitHub as “milestones”. For now I’ve laid out each use story as a sprint. Each milestone has associated “issues” which are the tasks for that story. We also have a taskboard on GitHub (via ZenHub plugin) with all of these issues / tasks. Screenshots of how this looks on GitHub are included in the same directory as this file. These stories are listed in order of priority, however the ones with #X haven’t been given a firm order.

# Sprint #1 User Login (10 points)

Create the base app, have an SQL database, and preload the data somehow. Have a login screen that takes a username and password and states success or failure.

# Sprint #2 Create / Delete / Edit Accounts (10 points)

Once we can log into the app with preloaded data, we will have a basic employee start screen. For HR employees, views that list all employees and allow editing of employee information.

# Sprint #X Create / Edit Events (5 points)

Allow events (the big planning ones, not personal calendar items) to be created and have employees assigned to them. Implement editing event info and stats after creation.

# Sprint #X Create / Edit Teams (5 points)

This will essentially be assigning people to events with various roles and creating links / teams amongst these people.

# Sprint #X Calendars (10 points)

Implement the calendar classes and associated views. Have conflict checking in place. Allow users to invite other employees to things on their personal calendars. We might have invites show up across calendars although he TA said it was acceptable to have calendars be their own little islands essentially.

# Sprint #X Permissions (5 points)

This is optional. Implement permissions checks throughout the login and database methods to allow / disallow tasks based on the user's role in the company. So a normal employee can no longer delete employees, etc.

# Sprint #X Messaging (1 point)

Implement some form of messaging within the app between employees. This could be as simple as having it use a phone number for an employee and open up the SMS app with a thread for that number.

Tasks for First 2 Stories / Sprints

These are the tasks for the first 2 stories we plan to implement. Screenshots of this are also included for how they look on the actual taskboard on GitHub. The tasks are presented in order of priority for each sprint, although some of them can and will be occurring simultaneously by different team members. The task number is the issue number on GitHub.

# Sprint #1 User Login (10 points)

## Create Base App #8

Should be pretty simple. Choose a basic template in Android Studio and set the API levels. Create the project. Run it in the emulator. Push to GitHub. Make sure everyone can pull and run it.

## SQLLite Database #9

Follow this tutorial. This story is finished when we have a database and can run some sample code to test it in some way to know that it's "active" and “working".

## Database Test Data #10

Hardcode in (either by having the DB pre-populated or having code run every time on initial launch to insert data) a master login. This will be like an admin or root user.

## Login Activity / View #11

Create an Activity class and associated view for the login. A user will be presented with a screen that has a username and password field with a submit button. On submission it will check against the DB and present a Toast that says success or failure.

## Test Successful Login #12

This is just us physically using a device or emulator and entering the correct info, which should get us a "success" message and entering incorrect info which should get us a "failure" message.

# Sprint #2 Create / Delete / Edit Accounts (10 points)

## HR Activity / View with List of Employees #13

Create an Activity class and associated view that will display a list of all employees in the database. This will be a ListView and each list item can be clicked to proceed to another Activity with details and options for that employee.

## HR Activity / View to Edit Individual Employee Info #14

Create an Activity and corresponding view to display an individual employee's info and edit the info or delete the employee. This will be the same as the Activity / view for normal employees except that it allow the changing of employee roles and deleting an employee completely. We may use the same files for both of these and just test for who's viewing.

## Employee Activity / View to Edit Info #15

See issue #14. Essentially the same except fewer abilities. We may use same code for issue #14 & #15.

## Database Controllers / Methods to Handle Editing Data #16

Add methods to SQLLite classes to handle editing employee info, this may be one method with options or multiple methods.

## Database Method / Controller to Delete Employee #17

Add method to SQLLite code to handle removing a user from the database. Must handle removing employee properly from all tables in DB and there may be additional checks for this to prevent things from failing.