How to Use this Template
1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: KirkB1693

Baseball by the Numbers

Description

Baseball by the Numbers is a baseball simulation app. It will allow the user to simulate individual games or seasons. The players will be generated when a user starts a new career and new players will be generated for a draft in between seasons. There will also be a free agent pool. Each team will have a salary cap and need to stay under the cap with each transaction.

Intended User

This is an app for baseball simulation fans.

Features

The main features of Baseball by the Numbers are:
- Create your own league
- Run your own team
- Player data will be saved over time
- Your team record will be saved over time

User Interface Mocks

Screen 1 : Main Screen

This is the main screen for the app. It lists the team/city the user chose and details about the next scheduled game. There are buttons to edit the current lineup (links to "Lineup Screen") or let the coach (AI) set the lineup. The next game can then be started (link to the game screen) or simmed. If sim is chosen the game will be simmed and the screen updated with the new next game info. If the user is using the app for the first time they will be directed to the "New League Setup" screen.

Screen 2 : New League Setup
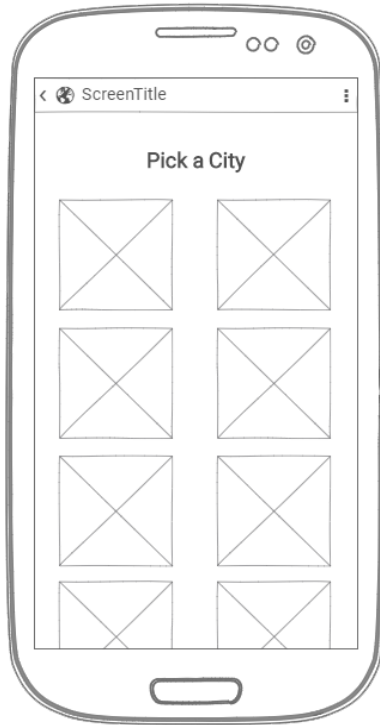
The user will be shown a list of cities they can use as their team city. When they click on one they will be taken to the "New League Options" screen.

Screen 3 : New League Options

The user will be asked to enter a new Coach/GM Name and Team Name. Through the use of dropdown lists (spinners) the user will be asked to customize their league. Once finished the user is able to start the season by pressing the "Start Season" button. Clicking the "Start Season" button will take the user back to the main screen and allow them to play/sim games as setup is now finished.

Screen 4 : Game Screen
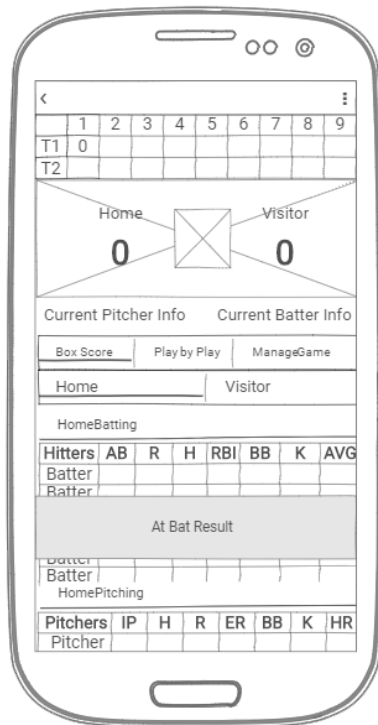
The user will be shown the game in progress. The screen will update after each at bat with a toast showing the result of the at bat. If the user chooses to manage the game there will be options for pinch hitting, substituting pitchers and a pause/play game button.

Screen 5 : Lineup Screen
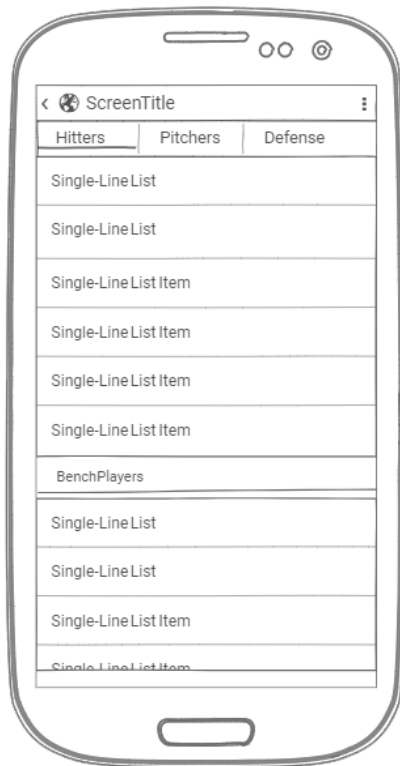
This screen allows the user to set a lineup for the batters. The pitching tab will allow the user to set the rotation of starting pitchers and order the relievers. The defense tab will allow the user to set where the batters are playing in the field. To change the lineup ideally it would be drag and drop. Clicking on a player will take the user to the "Player Screen"

Screen 6 : Player Screen
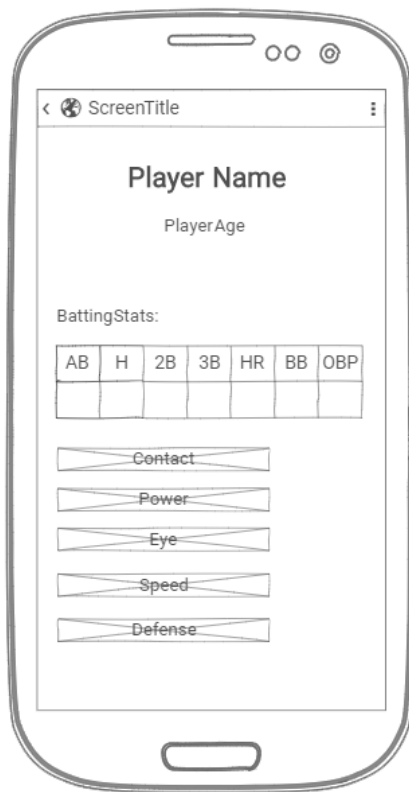
This screen will give details about the selected player including their season (and possibly career) statistics. It will also show a visual representation of their ratings in several categories. Stats shown and ratings will change based on if the player is a hitter or a pitcher.

Key Considerations

How will your app handle data persistence?

Player and team/league data will be stored in a database (or multiple databases) using Room. If time permits I might setup the ability to load real players from an online source using firebase, but as this is a large project initially the teams will all use randomly generated players.

Describe any edge or corner cases in the UX.

Once a game has started there will need to be a record of the game and the users place in it so that if the user backs out or closes the app it is not left unfinished. My solution would be to check for any in progress games when the app starts or upon return to the main screen and then either give the users the option to continue the game in progress or sim the remainder of the game.

Another edge case could develop if career statistics are tracked and/or additional players are generated at the end of each season. If the user plays a lot of seasons the database could grow very large and cause memory/storage issues. I would solve that by limiting the number of seasons that could be played before having to reset the database (start from scratch).

Describe any libraries you'll be using and share your reasoning for including them.

Picasso for image caching.
Dagger or the Android DataBinding Library for databinding.
JRand for random data generation (names, numbers for determining at bat outcomes, initial player qualities/stats, etc...)
CardView for displaying list data.

Describe how you will implement Google Play Services or other external services.

I'm not currently planning on using services, but I might implement banner ads in a free version which would need Google Play Services for the ad generation. Firebase could also be implemented to allow downloading players/teams from specific years at the point of League creation.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

Build a java library to handle new team generation

- Generate new players based on mlb averages with upper and lower bounds.
- Use the JRand library to handle name and statistic generation
- Generate a fixed number of position players, starting pitchers, and relief pitchers.
- Return a full team of players

Implement schedule generation
- Pass a list of teams, # of leagues, # of divisions and create a new database using Room with the schedule data for the season.

Implement a league database of players
- Use Room to setup and manage the player database for the league.
- Ensure that database has enough data columns to support future simulation options and changes to the game simulator not just bare minimum for current version.

Implement the results of an at bat
- Pass a game state (where are runners and how many outs), the batter and the pitcher to the AtBat module and return the result of the at bat as a new game state.
- Update runs scored, if needed, based on the game state change.

Implement a game simulation AI
- Pass in two teams and return the result of the game.
- Update all player stats based on the individual at bats in the game

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:
- Build UI for Main Screen (MainActivity)
- Build UI for New League Setup (NewLeagueSetupActivity)
- Build UI for New League Options (NewLeagueOptionsActivity)
- Build UI for Game Play (GamePlayActivity)
- Build UI for Lineup Screen (LineupActivity)
- Build UI for Player Detail Screen (PlayerDetailActivity)

Task 3: Test for Errors

Check for errors in the database(s) as they grow in size
- Implement error checking and set limits to prevent errors
- Check on and report errors correctly in code and handle them appropriately to prevent a crash.

Task 4: Implement Free/Paid Versions

Implement a free version with ads
- Create new layouts for a free version with banner ads (and maybe interstitial if it doesn't interfere to much with gameplay flow)
- Paid version (no ads) should already be complete and working at this point

Task 5: Clean up all screens/activities

Check all screens, implement transitions where appropriate
- Ensure all screens follow Material Design Principles
- Make sure screen navigation is intuitive and clear

Make sure all constants are stored in values/strings folder

Add as many tasks as you need to complete your app.

Submission Instructions
- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
  - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:
- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"