How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: jdippena

Jaco Dippenaar

Description

Shadowroller supplements Shadowrun 5th Edition by providing easy access to dice rolls, common rolls, probability, and other mechanics. With Shadowroller it is easier to forget the minutia of the dice and play more fluently with less hassle. You can

- Perform simple and extended tests
- Remember previous rolls
- View probabilities for rolls with and without edge
- Create easy-to-access dice pools that sync between your devices

Intended User

This app has a very targeted user base: players of Shadowrun (the tabletop role playing game). Most of the mechanics from 5th edition should transfer over to 4th edition as well.

Features

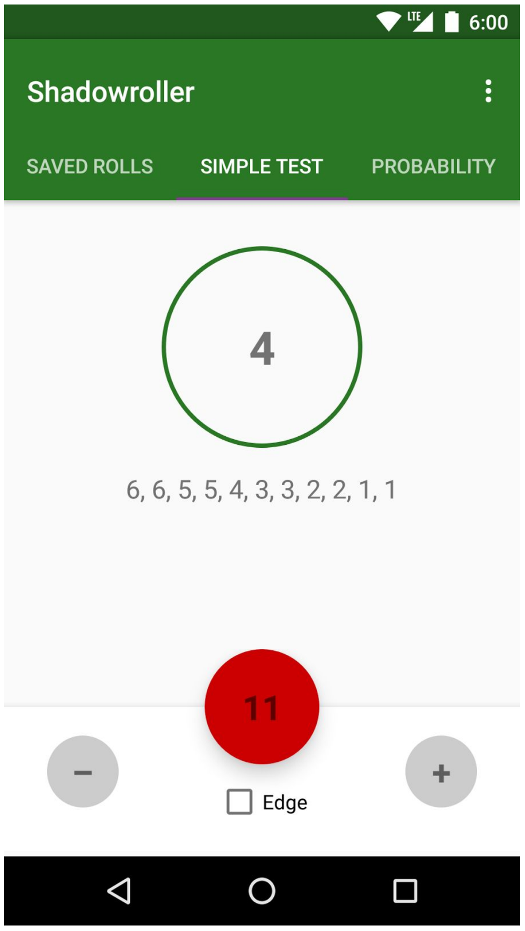
- Roll dice and determine number of “hits” and degree of success for the test
- Store the history of rolls made (with option to clear the list) in a database (possibly synced between devices)
- Store user-defined dice pools and sync between devices
- Show pre-calculated probabilities for rolls with or without edge

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

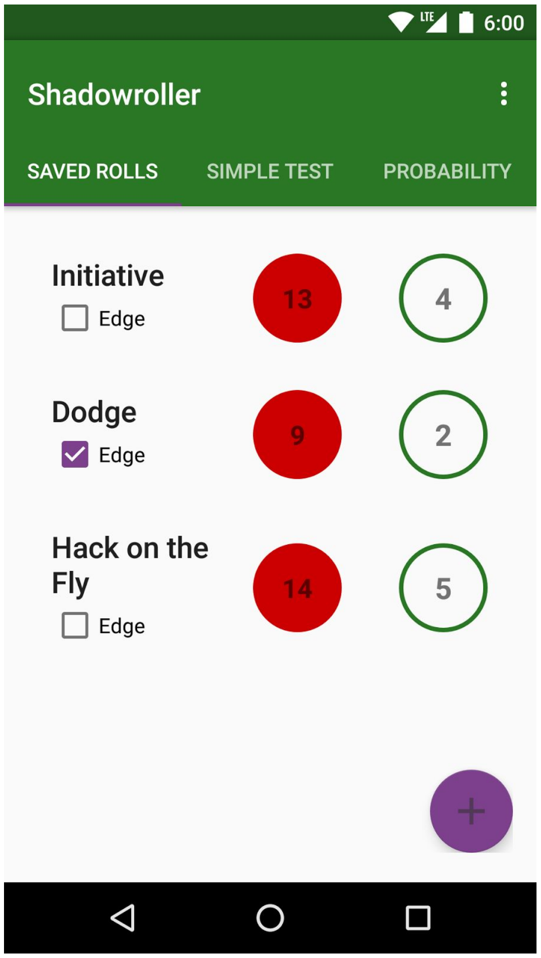
1. Simple/Extended Test
2. History
3. Probability
4. Saved Rolls

Screen 1



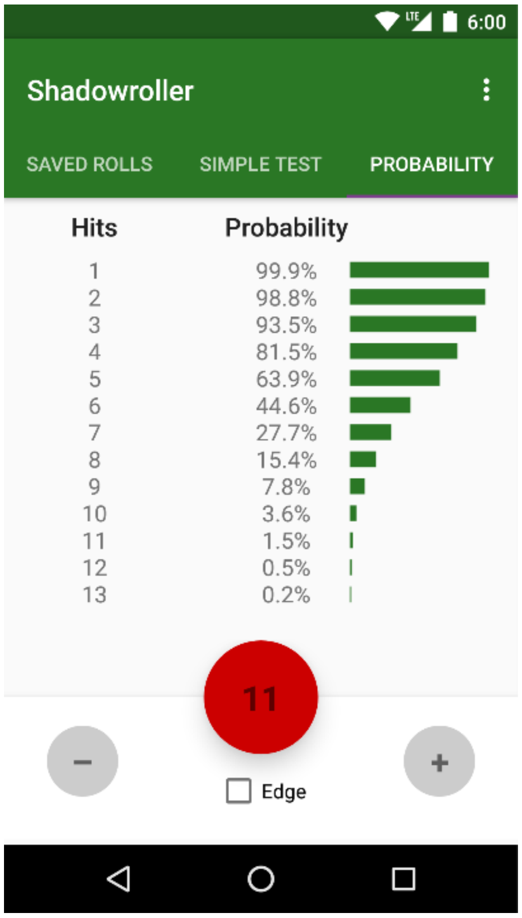
This is the screen that provides the main functionality of the app. It's fairly simple in design and in function: the user can adjust the number of dice and roll them to produce a number of "hits" (the number of 5's and 6's, in this case 4). There will be another method to select dice as well. The green circle will animate after each roll and expand in proportion to the number of hits scored.

Screen 2



Here the user can save certain commonly used dice rolls, roll them, and see an abridged output. The FAB will open a dialog where the number of dice and name can be selected.

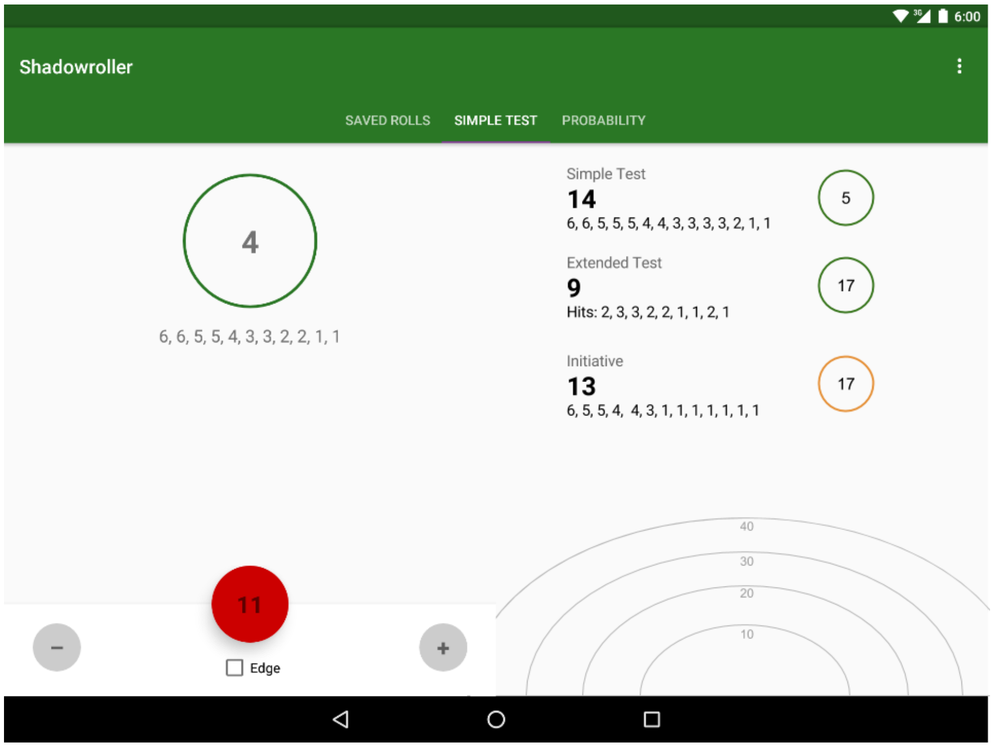
Screen 3



Hits	Probability
1	99.9%
2	98.8%
3	93.5%
4	81.5%
5	63.9%
6	44.6%
7	27.7%
8	15.4%
9	7.8%
10	3.6%
11	1.5%
12	0.5%
13	0.2%

Here the probability distribution for achieving “n or more hits” is displayed. This would allow users to make more informed decisions regarding character improvement.

Screen 4



The left of the screen will emulate the phone app, but the right side will present the history view (accessible via overflow menu in the phone app) and the dice selector (accessible by touching or swiping the bottom panel). The history view gives the name of the test and whether or not a “glitch” occurred. The dice selector works by allowing the user to draw their finger over the concentric ellipses to a certain value.

Key Considerations

How will your app handle data persistence?

The app will build a content provider with the use of third party library. The data will then be synced to the cloud so that roll history and saved rolls will be accessible across devices.

Describe any corner cases in the UX.

The green circle in the simple test screen will respond differently if a “glitch” or “critical glitch” is encountered (i.e., turn red or display an exclamation point). There is also another mode of rolling dice called an Extended Test, which has a number of different options. There will be a custom method of selecting dice that doesn’t depend on the plus and minus buttons (the bottom section will slide up to reveal a selection UI). Other considerations would be for the tablet version of the app and responding to network events. For example, data will need to be carefully managed if there are conflicts between devices due to lack of connectivity.

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

There is an appropriately named SimpleContentProvider library created by MIT Media Experience Lab which should vastly streamline the process of dealing with data. (https://github.com/mitmel/SimpleContentProvider)

Next Steps: Required Tasks

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

Task 1: Project Setup

I already created the basic outline of the project to make the UI mocks above, so the basic framework is already done. I will include the SimpleContentProvider library to integrate immediately with any data generation. I will also flesh out the rest of the framework by adding a settings activity and a fragment class for the Extended Test.

Task 2: Implement UI for Each Activity and Fragment

- Make UI flexible across device screen sizes and make buttons in Simple Test functional
- Do the same for Extended Test
- Ditto for the probability screen, but also create simple custom view to display probability distributions
- Create dialog fragment for adding saved rolls
- Create custom view for dice selection

Task 3: Build Tablet UI

- Take the UI I built for phones and extend it to larger screens, tweaking and adapting it where needed

Task 4: Create Free and Paid build variants

- Create two separate build variants

- Include Admob to serve ads in the free version of the app

Task 5: Include Analytics integration

- Research the Analytics API
- Import the necessary libraries
- Configure frontend for the API

Task 6: Implement Data Sync to the Cloud

- Create Google App Engine project to sync data to
- Create Sync Adapter and batch requests to be sent to the server, taking API level into account (i.e., using JobScheduler for API > 21)

Task 7: Create a Widget

- Design the layout for the widget consistent with the theme of the app
- Write code to integrate it into the app's data layer and

Task 7: Finishing Touches

- Add any setting options to allow the user to customize the experience

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

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"