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: AustinJZeller

Addition Learner

Description

This app is a simple brain exercise that will teach the user their addition. For users that don't know their addition skills fast enough, this application will train users to know their addition like the back of their hand, and for others who know their addition well, this will be a great brain warm up for them as well.

Intended User

This application is intended for users in Elementary School.

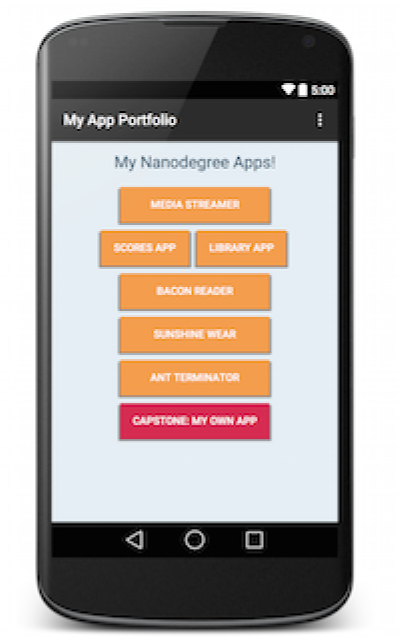
Features

- Uses a timer to pressure users
- Has a score in the top right corner to encourage users to achieve the highest score
- Random number generator to generate new addition samples

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.

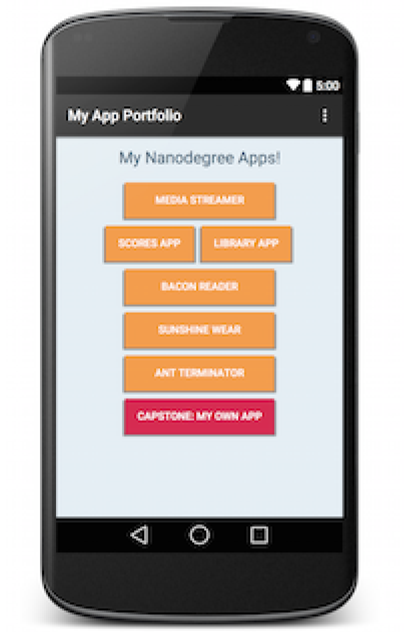
Screen 1



Replace the above image with your own mock [click on the above image, then navigate to Insert → Image…]

This is the start screen, it has a “Go!”button, so the user can start whenever they are ready. Once they press the “Go!” button, they will be taken to screen 2 to start the addition game.

Screen 2



Replace the above image with your own mock [click on the above image, then navigate to Insert → Image…]

This screen has the functionality of the game, showing a timer of 30 seconds in the upper left corner, a score correct out of wrong in the upper right corner, a random addition sum centered below which is generated by a random number generator and below that, 4 connected squares with random numbers in them, only one of the numbers is correct, the other 3 numbers are wrong, if a user clicks a correct number, their score will go up and a message below at the bottom will say “Correct!”, if they guess a wrong number, their score will stay the same until the score goes up and they will get a message at the bottom saying “Incorrect!”. At the end when the timer goes off, their score will be displayed at the bottom and a “Play Again” button will appear allowing the user to play the game again.

Key Considerations

How will your app handle data persistence?

My Application won't have to use data persistence, for it is an Education app that randomly generates addition questions to help Elementary school students become more proficient in addition.

Describe any corner cases in the UX.

Instead of there being a back button, the user is prompted with a “Play Again” button to restart the game, this allows the user to restart the game immediately instead of having to go back and press “Go!” to start the game. Although when the user closes the application, and opens it back up again, the “Go!” button will be prompted the them again, for them to play the game.

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

No Libraries were used for this application. This application was built intended for Education Purposes to help Elementary School Kids their addition as a simple offline game.

Describe how you will implement Google Play Services.

No Google Play Service APIs were used for this application. This application was built intended for Education Purposes to help Elementary School Kids their addition as a simple offline game.

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

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.

You may want to list the subtasks. For example:

- Create two activities to showcase two different screens (A start screen and game screen)
- Show and Hide two to three different UI Elements within the app at different time
- Configure a Timer to make the game more difficult
- Configure a scorekeeper to encourage users to want to play more for a better score
- Use a random number generator for the addition questions and answers
- Use colors of choice to color the background of each of the answers.
- Have a play again button pop up at the bottom of the screen as soon as the timer runs out.
- Show the' score the user achieved in big text size at the bottom after the timer runs out.

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:

- UI for MainActivity (1st screen)
    - Blackish grey colored background
    - A Button colored in green with the words “Go” or “Start”
- UI for game screen (2nd screen)
    - A textview for a timer location in the upper left corner of the screen counting down from 30 seconds
    - A texview for a score of the amount correct out of wrong in the upper right portion of the screen
    - A textview that holds an addition question, centered in the middle that prompts the user a new random addition question every time they submit an answer.
    - 4 textviews in the middle that act as buttons for answer choices for the user.
    - A button called “Play Again” hidden, so it is not shown until it's game over

Task 6: Add a button on the first screen for the user to start the game

- This button will say “start”
- Once the user presses this button, the game screen will appear for the user to start the game

Task 3: Add 4 squares with numbers inside that act as answers for the user

- 4 connected squares centered in the middle, with random numbers within each square
- All of the 4 connected squares generate random numbers in each square, and always in different places, so that the answer is not in the same place every time.
- Each of the 4 random squares will each have a random background color indicating to the user that the numbers are clickable answers

Task 4: Add a textview for the score to the upper right side

- A Score of the amount correct out of wrong
- Each time a user gets an answer corrected, their score will increase by 1
- Each time a user gets an answer wrong their score will decrease by 1

Task 5: **Add a “Play Again” button**

- This button will appear after the timer has run out, when the game is running, it will have an alpha of 0
- Placed at the bottom of the screen
- When the user clicks the “Play Again” button, the timer and game itself will restart for the user to play again

Task 6: **Add a textview for the timer in the upper left corner**

- Timer will have a maximum of 30 seconds
- Adds a challenge for the user to complete as many questions as possible
- Once timer runs out, it will not restart until the user hits the “play again” button or closes out the app and restarts it to start the game again.

Task 7: **Add a textview at the bottom of the screen**

- This textview will say “Your score: ?/?” showing the user their score they achieved
- This shows the number of questions they got right out of the ones they got wrong.

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**”