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: krizk93

TriviApp

Description

This app is a fun and entertaining trivia app, where the user is asked several multiple choice questions, and gets points on each correct answer. The user can choose from different categories and try to answer the questions correctly before time runs out.

Intended User

Anyone who would like to test their knowledge in a fun game

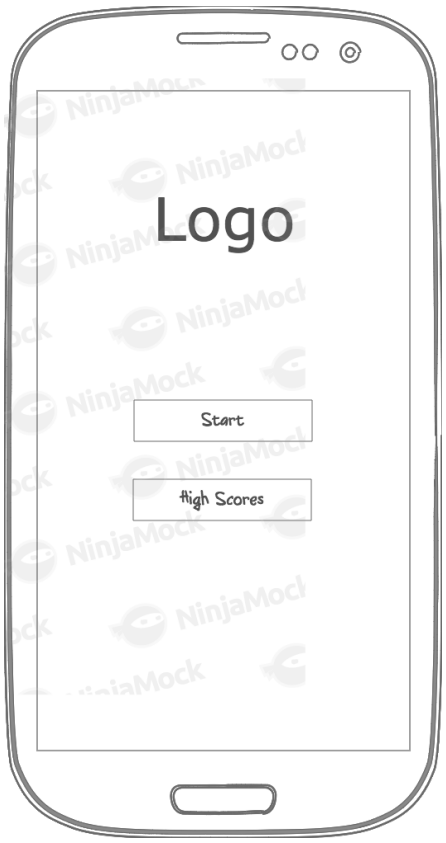
Features

- Display multiple choice questions
- Allows user to answer questions within a time frame
- Saves user high scores

User Interface Mocks

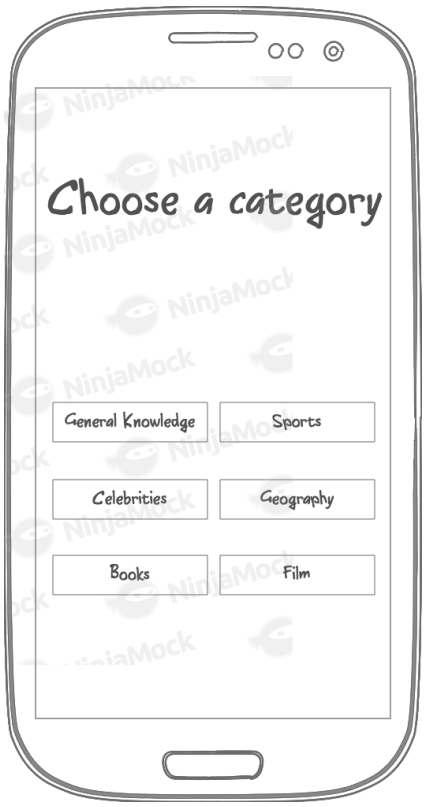
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1



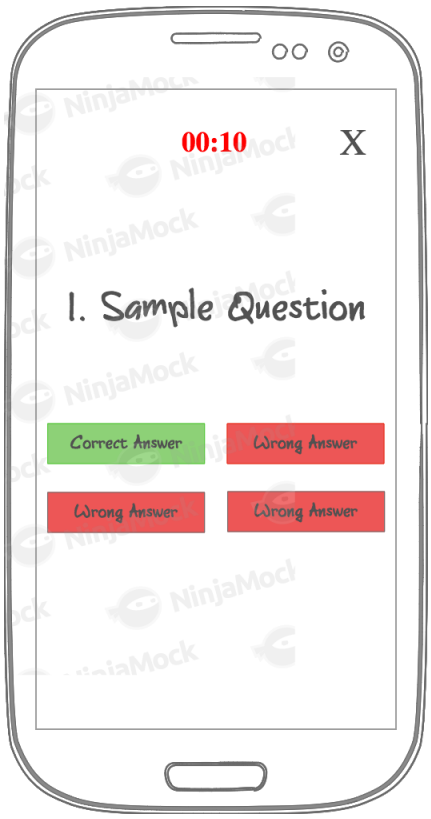
The first welcome screen displays the app logo and a "Start"button to start the trivia quiz, and a "high scores" button to display the last saved high score.

Screen 2



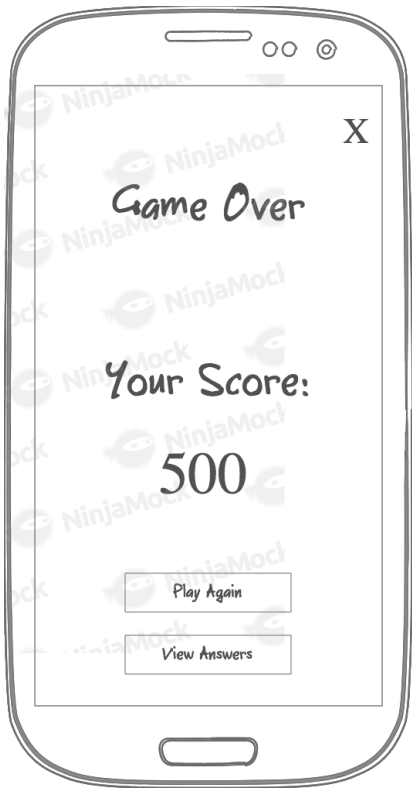
Once the user clicks on "Start" the app displays the category chooser screen, asking the user to choose a category.

Screen 3



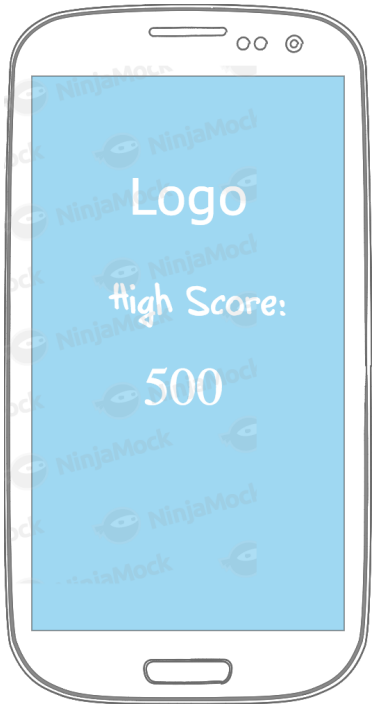
After choosing a category, the app fetches 10 questions from the API from that category, but with varying difficulties (3 easy, 4 medium and 3 difficult). The user has to choose between 4 multiple choice questions by tapping on the button displaying the correct answer before the time runs out. (around 20-30s per question… but I have to test it to find the correct time interval). Choosing the wrong answer, or failing to choose an answer before the timer runs out will end the game. After the 10 answers are solved successfully, the user is prompted to choose a new category and then fetches 10 more questions.

Screen 4



When the quiz ends, the score is displayed. If it is a new high score, the high score gets updated. The user has the option to view the answers to the questions he solved, currently stored using the Content Provider. When the user clicks "play again" he is redirected to the category chooser screen, and the score is reset and previous answers are deleted from the database.

Screen 5: Widget



The app contains a Widget, displaying the app logo, and the current high score. Clicking on the widget launches the WelcomeActivity (Screen 1)

Key Considerations

- The app will be written solely in the Java Programming Language. The app also uses stable versions of Android Studio (3.1.3) and Gradle (3.1.3)
- All strings are stored in Strings.xml
- The app supports RTL layout switching in all activities
- The app includes accessibility support

How will your app handle data persistence?

The app will use a Content Provider to store the user's answered questions after one trivia game is finished and give the user the option to view his answers. It will also save the user's high score (maybe in shared preference)

Describe any edge or corner cases in the UX.

- At any time during the questions activity (screen 3) the user can click on the close button to cancel the quiz and be redirected to the first screen. All progress is lost and saved answers for this game session are deleted.

- Clicking the back button will not allow the user to go back to a previously answered question (screen 3, for example if the user answered question 2 and now is in question 3, he cannot go back to question 2)
- Clicking on the close button in screen 4 will also redirect to screen 1 and all progress is reset.
- When app is offline a toast message appears to show that there is no internet connection

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

The app will use Retrofit (2.4.0) to get a JSON response from the API containing the trivia questions and answers.

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

The app will use admob to add interstitial ads (between maybe every 10 questions) in the free version.
It will also implement Firebase Analytics to get insights on how users interact with the app, and particularly to know which of the different trivia categories are being most selected.

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

- Configure Retrofit library
- Add Dependencies
- Add Firebase Project
- Create Data Model Class

Task 2: Implement UI for Each Activity and Fragment

- Build UI for WelcomeActivity
- Build UI for CategoryChooserActivity
- Build UI for QuestionsActivity and QuestionFragment
- Build UI for GameOverActivity

Task 3: Retrieve Data from API

- Use Retrofit library to make API calls (asynchronous request) and retrieve the JSON containing the trivia questions (or make synchronous requests using retrofit inside an AsyncTask ??)
- After the user chooses a category, the app retrieves 10 questions from that category using api call: (3 API calls are needed: first call: get 3 easy questions, second call: get 4 medium questions, and third call: get 3 difficult questions, then the results are appended and stored in a database).

Task 4: Implement Data Persistence Model

- Create Database model to store trivia questions and the user's answer once he answers a question.(the questions are stored in the database after the API request, then when a user answers a question, the database entry is edited to mark the user's chosen answer, and at the end of the game, the user can query the database and show his previously answered questions. After that, when navigating to the Main Screen again, the database entries are deleted.)
- Implement Content Provider to manage CRUD.

Task 5: Implement Countdown Timer

- Create Timer Class and implement logic to be used in the QuestionsActivity

Task 6: Implement Google Play Services

- Create Free and Paid build variants
- Add Admob implementation to free version
- Add Google Analytics to keep track custom events (when user selects a category in CategoryChooserActivity)

Task 7: Implement App Widget

- Create app widget containing the user's score, and when clicked, it opens the Welcome Screen

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**"