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: kzaf

TriviApp

Description

TriviApp is an application for unlimited hours of fun and quizzing. Play among millions of triviapp players and sharpen your brain by scoring the answers. TriviApp helps you to expand your knowledge while having fun and challenging your friends in multiple difficulty levels.
Create an account to keep track of your achievements and scores.

The app will be written solely in Java language.

Intended User

This app is intended for everyone who is looking to have fun and at the same time wants to expand knowledge. It is a great app for kids and grownups.

Features

- User login using firebase auth
- Select category
- Select difficulty level
- Select type (multiple choice, true/false)
- Fetch quizzes from an API (Open trivia DB)
- Track score

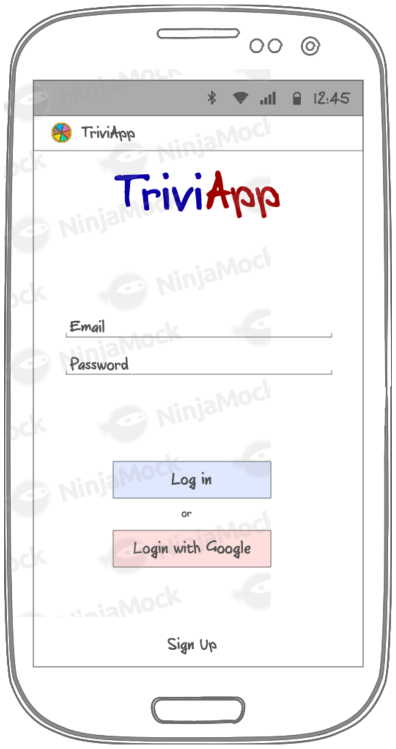
User Interface Mocks

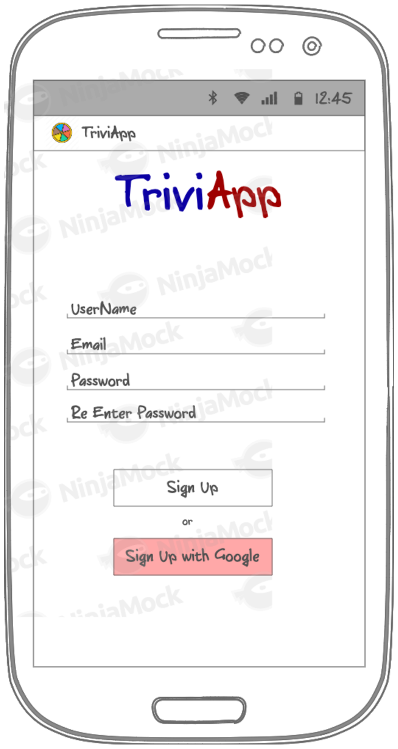
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1

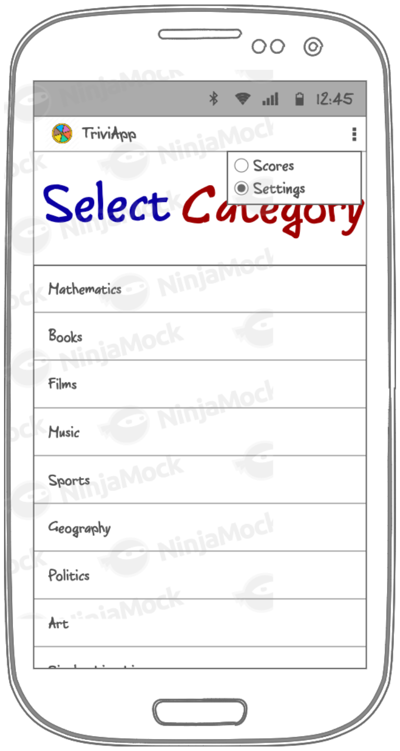
Splash screen

Screen 2 & 3





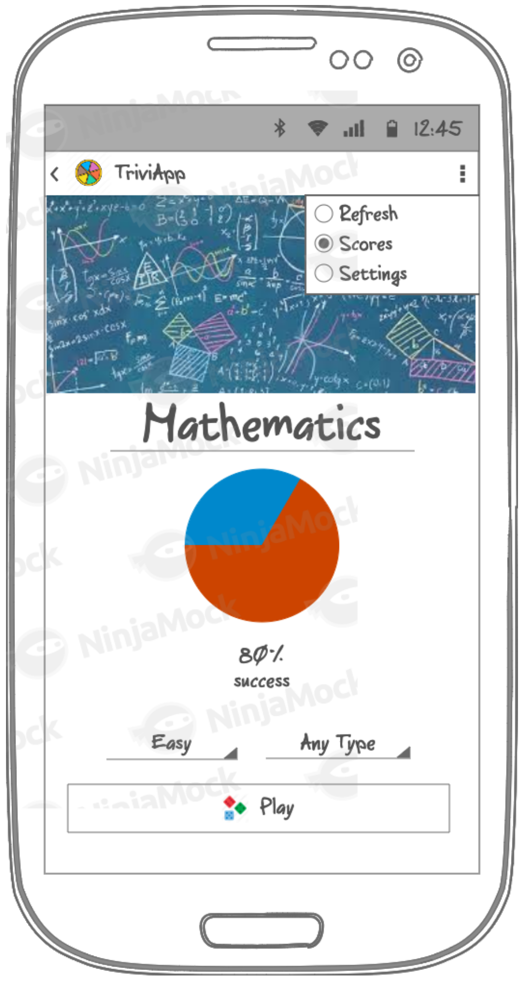
Login and SignUp screens.

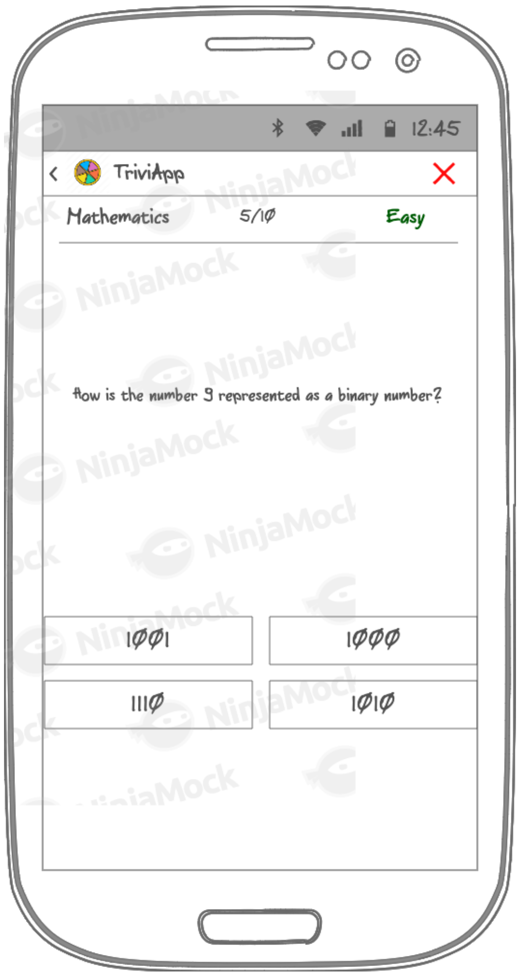


Screen 4

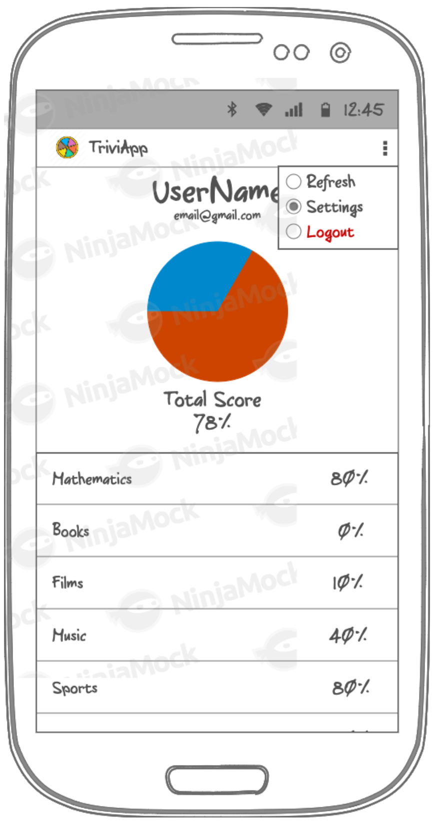
Select Category screen. This will be shown after the user has successfully logged in.

Screen 5 & 6





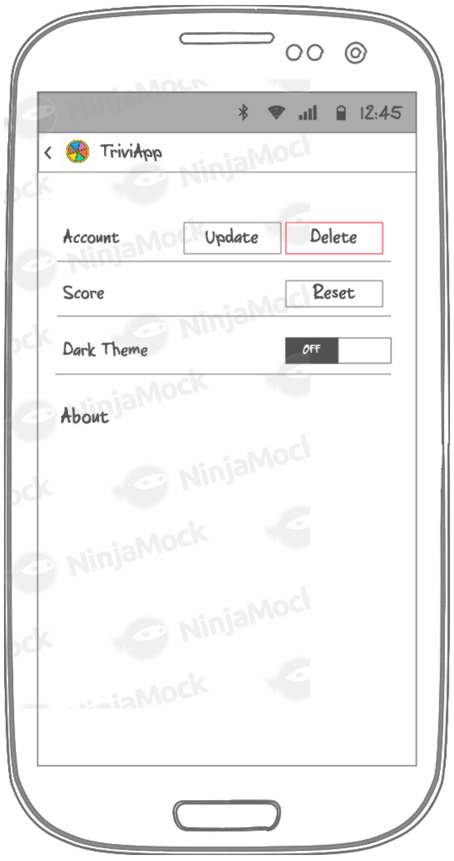
(5) Selected category screen, shows details about the category score and the quiz preparation.
(6) Gameplay screen where the actual quiz is played.



Screen 7

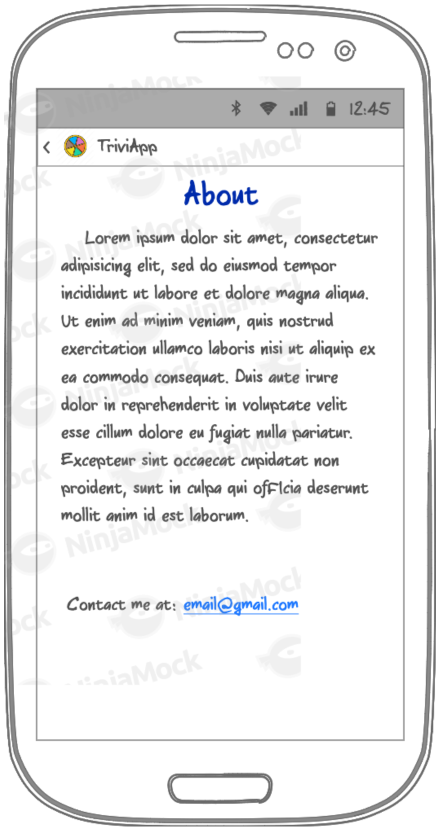
Profile or Scorers Activity. Will show the profile details along with the total score and the score of each category of the logged user.

Screen 8



Settings screen

Screen 9



TriviApp

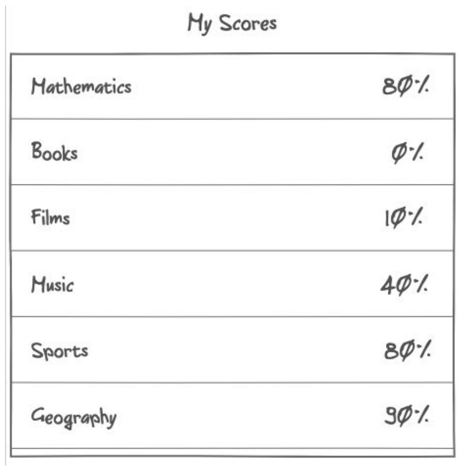
About

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui offIcia deserunt mollit anim id est laborum.

Contact me at: email@gmail.com

About page to display info about the app

Screen 10



My Scores

Mathematics	8Ø%
Books	Ø%
Films	1Ø%
Music	4Ø%
Sports	8Ø%
Geography	9Ø%

The widget will display the categories list with the scores.

Key Considerations

How will your app handle data persistence?

Quiz data will be fetched from API and will be saved in SQLite using Room. Scores and profile data will be saved in Firebase.

1) App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.

2) App includes support for accessibility. That includes content descriptions, navigation using a D-pad, and, if applicable, non-audio versions of audio cues.

Describe any edge or corner cases in the UX.

When the user is pressing the back button, it will return to the category screen. By pressing twice, the app will exit.
If the user is in the middle of the game, a message will be popped up to ask him if he wanted to leave the game. If the user is in the Login Activity, the back press will exit the app.
If there is no internet connection a proper message will pop.
If the user rotates the phone, the app will handle the data properly.

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

Toms Handwritten fonts will be used in the entire app.
Picasso for images. (V. 2.71828)
Room (V. 2.1.0), LiveData and ViewModel (lifecycle components V 1.1.1) for storing data locally (SQLite).
Retrofit2 (V. 2.6.0) with Gson (V. 2.8.5) converter for the API fetching.
Butterknife for handling the views. (V. 10.1.0)
MPAndroidChart for pie chart. (V. 3.1.0)
Firebase for Auth and LiveDB. (firebase-core V. 17.0.1)
Dynamic-Toasts for fancy colored toasts. (V. 2.3.0)

All libraries will utilize stable release versions.

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

Maybe AdMob.

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

App requires some libraries

- Setup gradle dependencies (libraries)
- Build basic activity with fragments

Task 2: Implement UI for Each Activity and Fragment

Build UI for splash Activity to display logo for couple of seconds.
Build IU for Login and Sign Up Activities
Build IU for Game Activity that has:

- UI for Select Category Activity.
- UI for Selected Category Details.
- UI for Gameplay Activity.

Build UI for Profile Activity.
There will be preferences UI as well.
UI for the About page.

Task 3: Implement Splash Screen

Create an empty Activity with the app logo that stays there for 2 seconds.

Task 4: Implement Firebase Auth

Setup Main Screen:

- For logging in (if already have an account)
- For signing up (ability to sign up with google auth)
- Create Session to keep the user signed.

Task 5: Implement Firebase database and SQLite

The user information along with each category score and summary score will be stored in Firebase. The scores will be calculated every time the user opens a related page and in case there is no internet connection, the latest scores will be shown that are stored locally.
There will be an SQLite DB where it will store the user info an the calculated score everytime.

It will perform short duration, on-demand requests, so the app will use an AsyncTask.

Task 6: Implement Preferences (Shared Preferences)

Preferences will be an Activity that will have the app and account settings:

- Account settings option (Delete, Update)
- Score reset option
- Dark Theme toggle (Maby)
- About page

Task 7: Implement Categories screen

Categories Activity is a typical activity that will have:

- A label
- A list with categories.

After selecting a category, it will prompt to the Gameplay screen
The menu options are:

- Scores (Prompts the profile screen with the scores)
- Setting that prompts to the settings screen

Task 8: Implement Selected Category screen

Category Detail Activity shows the preparation screen for the quiz. Contains:

- An image for the selected category.
- The last score with a pie diagram.
- 2 spinners to select difficulty level and question type (true/false or multiple choice)
- A button to start

The menu options are:

- Refresh (to refresh the score)
- Scores (Prompts the profile screen with the scores)
- Setting that prompts to the settings screen

Task 9: Implement Quiz screen

Quiz Activity is the "main" activity that have the questions and it will contain:

- The selected category and the level on top.
- The number of question (out of ten -default number-)
- A cancel option on AppBar that stops the game (no score counts when cancel)
- 4 or 2 buttons according to the question type.
- By pressing a button, the correct answer will be revealed and the score will count.

Task 10: Implement Profile screen

Profile Activity displays the username and the total score:

- The profile username an email.
- The total score of all categories.
- The score percentage for each category so far.

The menu options are:

- Refresh (to refresh the scores)
- Setting that prompts to the settings screen
- Share, to share progress with friends
- Logout option to destroy the Session and return to the Login screen

Task 11: Implement About screen

A simple Activity that displays information about the app and the developer.

Task 12: Implement Widget

Create a 3X3 widget for list with scores.

- Create widget layout
- Create data provider, service
- Add it to Manifest

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**”