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: arpyvan

ROP aware

Description

ROP aware is an app that helps to raise awareness about Retinopathy of Prematurity (ROP) and track diagnosed disease. It allows the users to create appointment reminders with attach hospitals and doctors.

Intended User

The target users of this app are parents with premature born children that are in the risk group of ROP diagnosis.

Features

- Provides information about premature birth risks and ROP
- Shows a calendar with upcoming appointments
- Allows user to create new appointment with specified doctor in specified hospital
- Allows user to save a doctor and hospital as favorite for easy access
- Shows notification about upcoming appointment
- Provides a widget with upcoming appointments list

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.

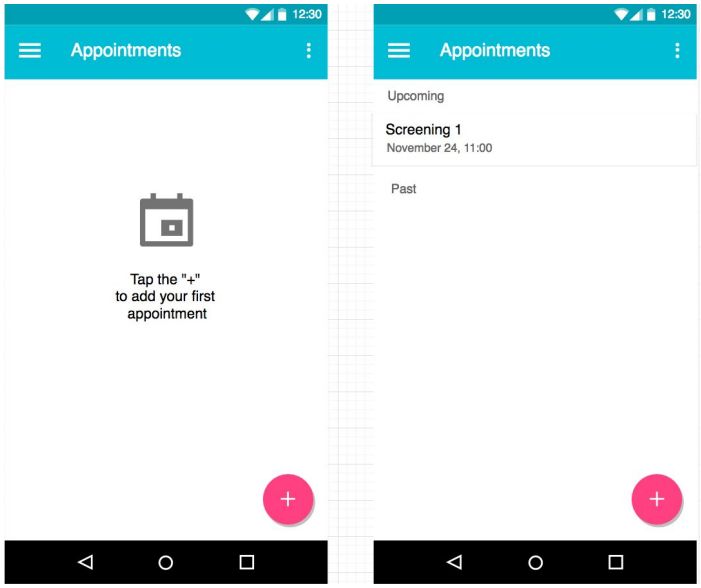
Screen 1

Main screen of the app is the guide page. Here is a grid view of all available tutorials about ROP. Each tutorial has a cover image and a title. Already finished tutorials are marked with as done.
The list of tutorials are fetched via content provider.

Screen 2

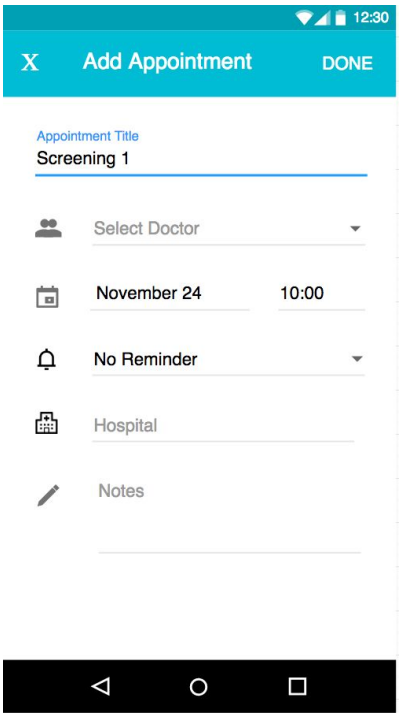
Selecting a guide in main screen opens tutorial detail page. It is a scrollable view with tutorial cover image, title and text.

Screen 3



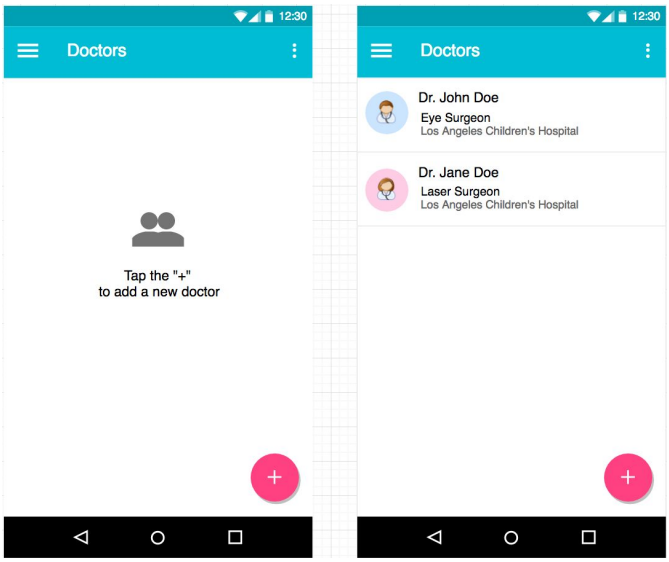
Appointments screen displays all created appointments for the user. If there is no appointment yet, it will display corresponding image. Existing appointments will be displayed in 2 separate blocks for past and upcoming dates. Floating action button will open a modal for adding new appointment.

Screen 4



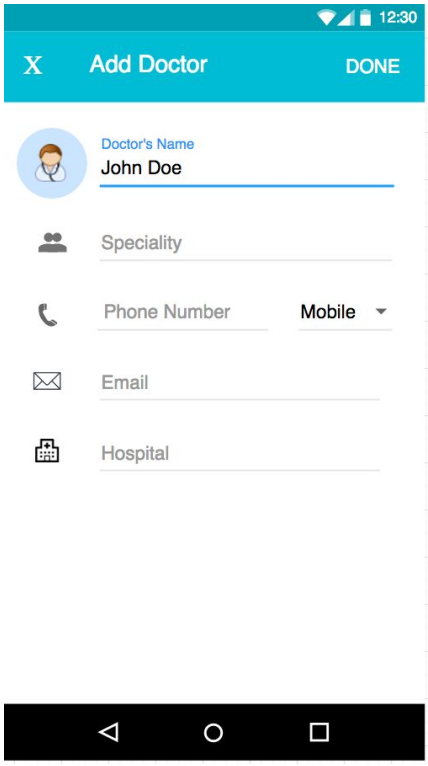
Add new appointment screen is opened as a modal and contains a form for adding new record. It consists of usual input fields as well as a date/time picker and select boxes.
The 'X' button closes the modal and the 'Done' button saves the new appointment.

Screen 5



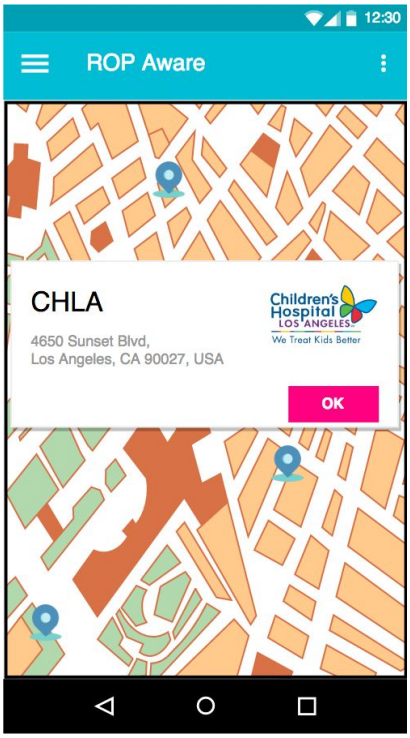
Doctors screen displays all saved doctors if there are any. For each doctor it displays an image, the name, field of study and workplace.
Floating action button opens a modal screen for adding a new doctor.

Screen 6



Add new doctor screen is opened as a modal and contains a form for adding new record. It consists of usual input fields with select boxes. Tapping on image opens an overlay to select from some image options.
The 'X' button closes the modal and the 'Done' button saves the new doctor.

Screen 7



Nearby hospitals screen displays a map centered near user location. It displays pins of places that are somehow related to ROP treatment. Tapping a pin opens an info view containing place details such as name, address and logo.

Widget

The app collection widget displays the list of upcoming appointments with the opportunity to call the doctor if phone number is specified.
It also has a “+” button that opens the app on “Add appointment” screen.

Key Considerations

How will your app handle data persistence?

The app will use Firebase API to store data on cloud and Content Providers in order to persist local database. Firebase is used for handling all user data like authentication and appointments list. Content Provider is responsible for Tutorials section data management. The app uses SyncAdapter in order to periodically retrieve and cache tutorials.

Describe any corner cases in the UX.

The app follows all general Material Design concepts in its UI and navigation flow. In cases when the user wants to add new data to existing list (like adding new doctor) it will open the “Add” activity as a modal view with ‘X’ that will close the modal. All pages are opened with transition animations.

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

The app will use the following libraries:

- Glide for displaying and caching images
- Firebase for data API
- ButterKnife for working with views

Describe how you will implement Google Play Services.

The app will use Google Places service for displaying nearby hospitals and Google AddsMob for displaying advertisement.

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

- Setup initial project
- Configure 2 separate build variants
- Include all needed libraries
- Implement app design theme

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Implement drawer menu
- Build UI for tutorials list activity
- Build UI for tutorial detail activity
- Build UI for appointments list activity
- Build UI for add new appointment activity
- Build UI for doctors list activity
- Build UI for add new doctor activity
- Build UI for hospitals map activity

Task 3: Implement Content Provider

- Create CP classes
- Implement CP requests
- Implement Tutorials data sync with CP

Task 4: Implement Firebase API

- Implement device authentication
- Implement appointment management requests
- Implement doctor management requests
- Implement hospitals management requests

Task 5: Generate views with real data

- Implement tutorials activity functionality
- Implement appointments activity functionality
- Implement doctors activity functionality
- Implement hospitals activity functionality

Task 6: Implement app widget

- Implement widget UI
- Implement widget functionality

Task 7: Implement app notification

- Implement upcoming appointment notifications

Task 8: Test and finalize the app

- Test edge cases
- Fix errors
- Prepare the app for Google Play upload

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**"