**CMPE 408 project one report**

**Group 2**

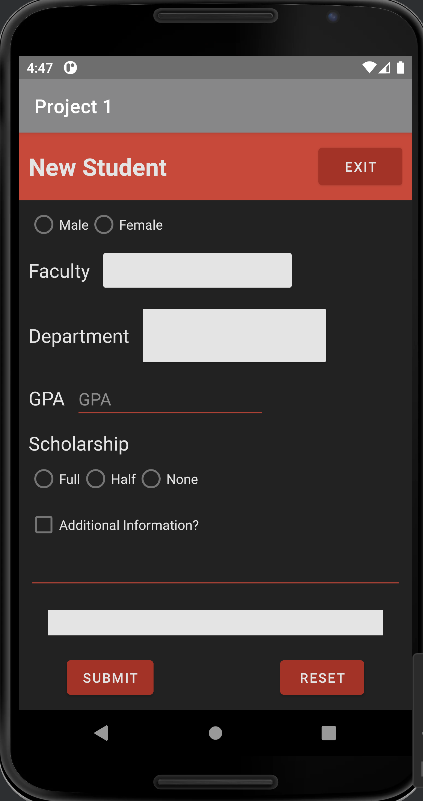
Mohammad Hamed 120200155

Hamza Abuzahra 120200003

Introduction:

Before starting the project the team members discussed how to split the tasks equally and evenly between the team. Due to the questions' short nature, we could not simultaneously work on a question. Thus we resorted to splitting the tasks down the middle. And work independently, until we are done with the said question. GitHub was used to manage the merging and the pulling of the project in between the questions. We later met up online to work together on the styling of the application. The styling of the application went through multiple iterations, each getting us closer to the end goal, but with each iteration, we were still not satisfied with the styling of the app. Until we finally reached the desired look of the app

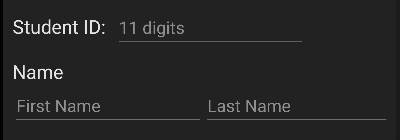
Application layout and styling:



It was decided to use a dark theme in order to provide an elegant and sleek theme for the user, a black background contrasted by the white text which filled the application. Moreover, the user was guided to input data by using white spinner boxes to indicate a spinner and by the red-accented fields for text inputs. The Buttons are a dark orange which complements the other colors on the page, the application theme is consistent and sleek. With visible text input fields which stand out to the user.

**Components of the application:**

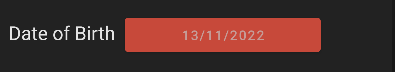
Student Id and name:



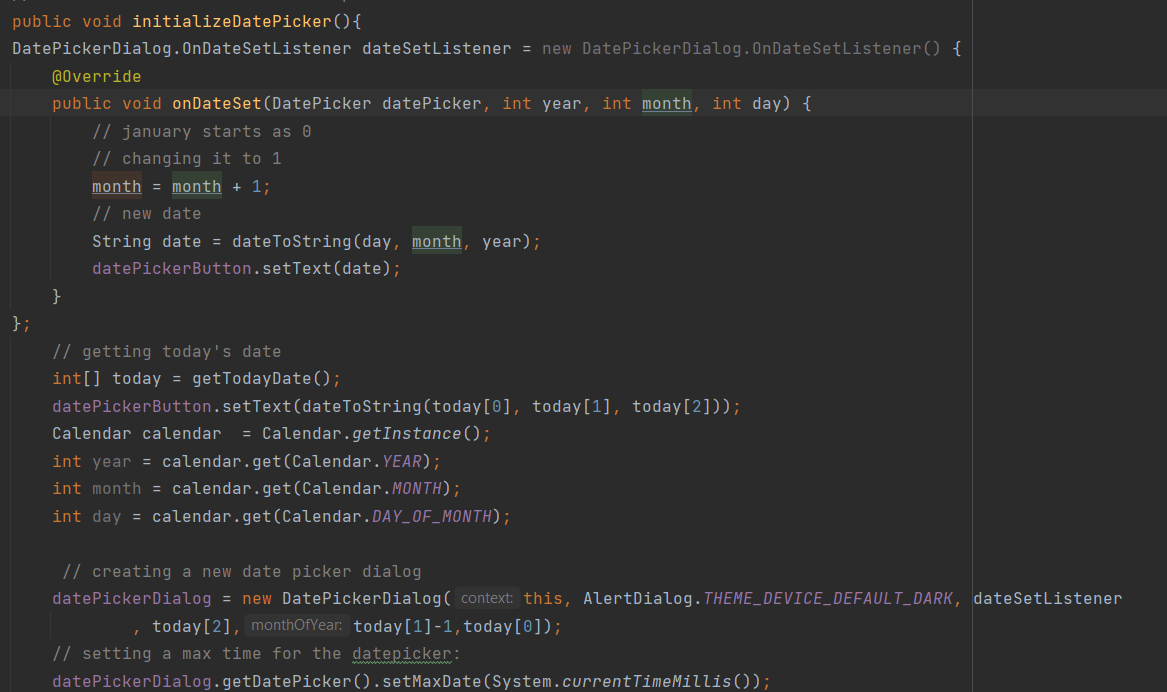
The student Id has a simple Linear Layout which is set to horizontal to place the student id TextView side by side with the EditText. The text size of the student Id is 20sp with a margin of 10dp. The EditText input is set to have maxLength 11, and the input type is number. so the student cant input any number longer than 11.

The name is split into two parts First name and Last Name both are simple Edit Text inputs, which only accept text as an input. The styling in this part of the application is set to be simple and coherent. The inputs have a red accent underline which contrasts well with the black background

Date of birth date picker:



The date of birth has a linear layout set to horizontal to fit the Text view side by side with the date picker button.

Initialization of the date picker

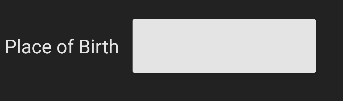
The date picker is initiated in the function shown above.

First of all, a data picker dialog listener is set to the data picker dialog

In the constructor override, the month is incremented as the months start from 0 for January in java. So January returns a 0 instead of a 1, thus we increment the month. After that, we convert the date to a string using a simple helper function which returns a string when passed the day, month, and year. And finally, we set the date picker button, (the orange button) to have the text displayed on it be the text made in the constructor.

To create a new date picker dialog we use the new DatePickerDialog which takes an instance of itself (this), the alert dialog theme, the dataSetListener, and finally the day month, and year. Finally, there is an important line of code at the end of the snippet provided above which sets the maximum date to be the date of today as we can't possibly have future birthdates as inputs.

Place of Birth spinner:



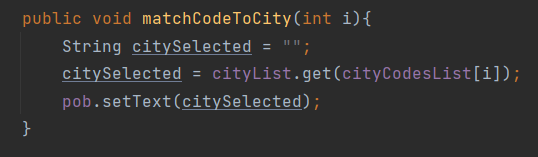
The place of birth is styled in the same manner as the other components above. But the place of birth spinner is set to be of a white color which contrasts the black background.

Place of birth code analysis:



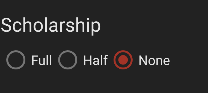
The data displayed in the spinner comes from the city codes list array. The array contains some codes that we gathered from the internet. The match code to the city algorithm solution was a simple hashtable with 2 strings which were initialized in the initializeCityList:

The code to match the city to the list is all in matchCodeToCity function which takes in i as the input of the onItemSelected default overridden method



Get the city name associated with the city code using the cityList.get() method and set the text to be displayed in a hidden textView which pops up after this operation, pob.

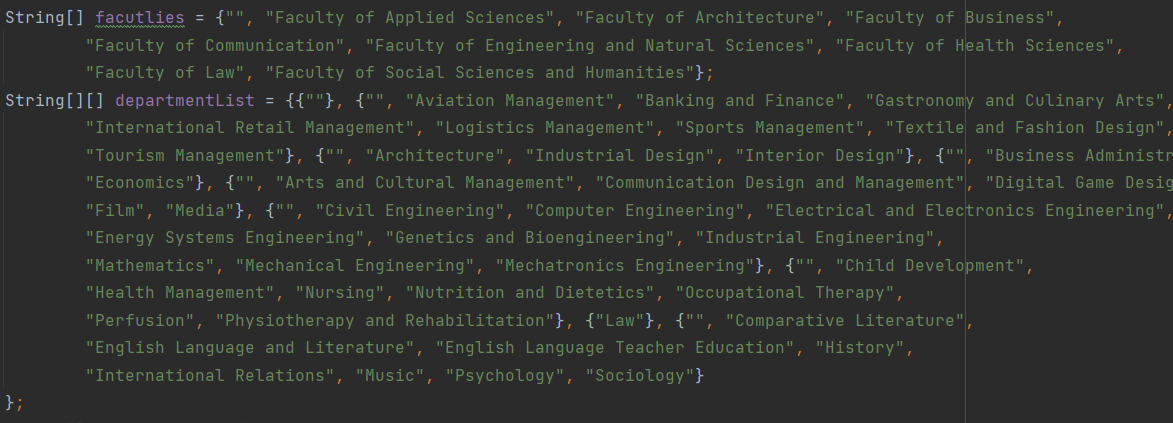
Gender and scholarship Components



The gender component has simple radio buttons which toggle between male and female, so you can't be a male and female at the same time. The same goes for the scholarship component, you can choose one of the three options, either full, half, or none. But only one of them

Faculty And Department Components:



The faculty and department components are both spinners. The spinners take their values from arrays predefined in the java code

The spinner displays the data from the array, through the use of a spinner adapter. And the data that is selected is then stored to be later displayed in the strings XML.

Additional information Component:  
The additional information component is a simple radio button that when clicked enables the input to an EditText to store the user’s additional information. If the user does not click on the radio button then the user cannot input additional information

Submit and Reset Components:

The submit and reset components allow the user to reset the information that was just entered by setting all the texts to empty and all the radio buttons to false.

The submit button on the other hand when clicked saves all the information gathered from the different buttons, spinners, and text fields into variables, and checks if all the fields are not empty. If so a toast message is displayed and a strings.xml is constructed, and the data is displayed to the user in a textView.