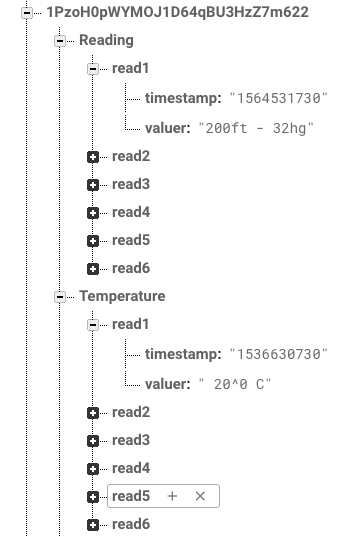
**SAG-DRONING**

Our team specialized in creating android applications that can be used for professional or personal practice. The leader or Head of our team is Gursehaj Harika. He is very positive and workaholic Leader who likes to get the team together and complete the assigned task successfully, expert in removing bugs from applications, head for authentication and securities for SAG-droning. Arman Velani is one of the important People on the Team. He is an Expert in understanding the concept, problems and gathering solutions. He played a key role in development of the SAG Droning App with an eye as sharp as an eagle, Arman velani finds small mis alignments in the UI that an average person would’ve never even guessed were there. The last team member in our group was Shubham Sharma. He was considered as the resource manager for our team. He is an expert in developing xml pages and also understands the concepts of Sql Database, he creates UI designs like it’s his second nature with seamless integration for all the components. Unfortunately Shubham had some troubles related to his Theoretical studies and is not in our team anymore. But, in his membrane we kept the name as SAG droning which stretches to ‘Shubham Arman Gursehaj Droning’.

Our Android Application is based on the idea of safe drone flights. Most of the drones that have all the sensors for safety of the drones and of people are very expensive. They range from $1500-$4000. Most people cannot afford that so they try to look for an alternative drones that cost about $200-$400. These drones can fly but they cut of the costings by removing all the sensors. Our team aimed at creating a device that can be attached to any drones in order to increase the safety measurements

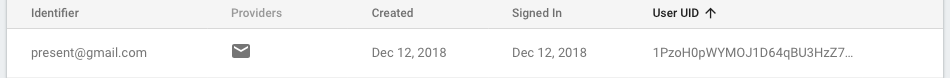
The SAG droning application connects with the hardware and helps the controller to keep a track on the Drone. The app allows user to register with own email, username and password. If the user is already registered he can login and save users preferences. Once logged in the App connects to the Hardware device (SAG attach) via NoSQL database called firebase. All the user information and readings from sensors are stored in the database. The user can look back into their data or save them on their local device. Our application also has an Account page where you can manage your account, change your password or delete your account. The app has dual authentication, online and offline. When user fails to login after multiple attempts to login they are redirected to contact us page where they can choose to call us or email us for further process. The readings of all the sensors are shown on the main home page using a drop down list. The user can also choose to look at the logs of a specific Sensor using the menu on the left. All the readings are stored on the device using shared preferences in case of a network failure.

When it comes to the database, the readings are stored in “**Reading**” under child “**read**” with a number at the end which will be auto incremented every time a new reading is stored. There are two parts of these “**read**” objects i.e. timestamp and valuer. The “**timestamp**” is in *EPOOCH format* that stores the time when the data was uploaded on the database which is converted in real time on the application before being displayed. The “**valuer**” stores the *real time readings* from the sensor as they are detected and read and uploaded to the database. The child “**Temperature**” has the same structure. The only difference is in the readings and the way the readings show up. In “**Reading**” the data stored in child “**valuer**” is a string “200ft – 32hg” which means the sensor shows information in feet as well as in hg that are the true values coming in from the sensor just in case a purist wants the real readings but in “**Temperature**” the child “**valuer**” has values in ^C that is used to temperature around the sensor in degree Celsius.



For User Information,

A user has to create an account and once this account is created successfully i.e. the user’s email did not exist previously and nor is the password weak enough to be ignored by the database, the application takes the user’s email and password and runs it stores the user’s email and password and saves it under the authentication section in the database. For the offline authentication, the email and password is run through a function that compares the strings of the user’s email and password and grants access to the user



Once the user is created and registered. Their name, email, password (For testing purpose), the product ID and their user ID are all the values that are stored in the database. The UID is unique and special to a user, no two UID’s in the database are alike which makes them the best key to use in order to make it easier and the searching up of a user under child “**userinfo**” that is under “**user**” faster.



Ideally, we were going to store the user’s email and password once they have logged In, in the shared preferences but that makes our application vulnerable to hacks and theft of user’s information. So, for our offline functionality the data that comes from the database is stored in shared preferences and is then loaded from there. Whenever the user tries to log in using the offline feature it will load the last updated readings. If it’s the first the application is run and the user tires to log in using the offline function then random generated values are presented (this feature is yet to be implemented, the code is there for random values but the function to detect if it’s the first time is yet to be created).

Our team is working hard on coming up with some ideas on how to implement our hardware code. According to our teams join research, we found that python is a great language that can be used on both the hardware side as well as software side and since we have already created out application. Using a python code to collect the readings from the database and uploading them to the database is easier than ever.

The hardware design that will complement the application design is supposed to be a sleek and minimalistic, we have two prototype designs already in works. These designs are in testing phase, so size of the design was not the core idea when they were being designed. The main idea behind them was to make sure the sensor stays intact when attached on the drone using a Raspberry Pi 3.

[](https://user-images.githubusercontent.com/43188450/49106925-70e38880-f252-11e8-94d2-0dfc12e6c991.jpeg)The First prototype has a multi layered design with option to add multiple sensors and have a modular take. This design helps us to add more sensors in near future which could be used as add-on paid features in our application as well as for the hardware. The upside to this design is that if one part fails it could easily be replaced for a newer part. A newer model of an existing part is released it can be switched on the other hand, the flaw with this design is that if the user has this attached to their drone and they crash land their drone, this case has tons of pieces that come to make it and an impact from a reasonable height will break it in pieces and could lead to a damaged part or the whole the whole product could fail.

[](https://user-images.githubusercontent.com/43185907/48949567-60f03f80-ef06-11e8-90fd-74825a6f81f5.jpeg)The Second design has a straighter approach, reduces the number of pieces required and has open space inside which means it could include a wider array of sensors but at the cost of zero modularity. The number of open spaces where dust particles could enter and risk the reliability of the product are significantly reduces. If the user wishes to use just one or two sensors rather than the whole array of sensors in our product, with this design they would not have the luxury to get exactly what they want.

Test cases play a very significant role in software design and in the process of creating a fully functional application. The cases that we designed for our application stretch our application to its limits and we expect our application to have at least 90% of sucess rate. For our login page that is supposed to be the first page the user starts to interact with has 4 cases.

**1.** Typing in the wrong email and password and pressing the online button to test if the database authenticates or no.

**2.**Typing in the right email and password combination along with the online button to activate online authentication to check if the application gets expected results.

**3**.Typing in the wrong password and correct email and use the offline feature to check if an error comes along when wrong password and email combination is entered along with a 3 incorrect password limit that is in place in order to help the user if they forgot their email or password. After the limit is reached at application must go the other page where a calling option and an email is given to contact out support team.

**4**.Typing in the correct email and password and checking if the authentication actually works when right combination is entered.

Once these test cases are competed and passed. Our application will undergo another set of test case that check its functionality. These tests will check if the application retrieves the data correctly from the database and if the navigation bar works as intended and if activities that show all the information from a specific sensor correctly.

1.Clicking on the select senor option to see if the readings are showing up once clicked.

2.Cllicking on the 3 lines on the top right corner to check if the navigation bar appears along with its option and click on the middle which should open up an activity which will only work if the navigation bar worked as intended.

3.By clicking on the barometric pressure, an activity should open up that has all the readings from the barometric pressure sensor that were stored in the database.

4.Lastly at the bottom of the navigation drawer a logout option is available which when pressed should send a sign-out signal to the database.

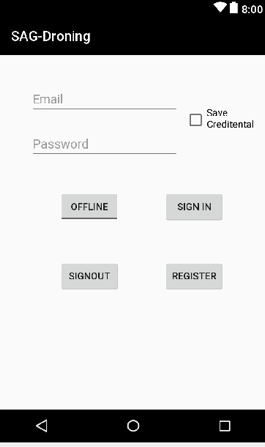
These test cases should give us an idea to our team, how our application will work in real life.

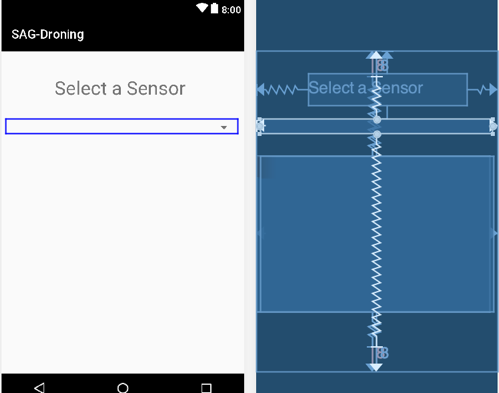
SAG- Droning is a fusion of multiple components working together seamlessly and creating a user interface that works in a manner in which it seems like all the components were designed just for this application.

 Our application starts with a **Splash screen**. With a timer. We have another Splash screen to show the user’s name along with a welcome back text.

There is an **ImageView** used in order to show the logo

On Our Login page there are multiple components that we used,

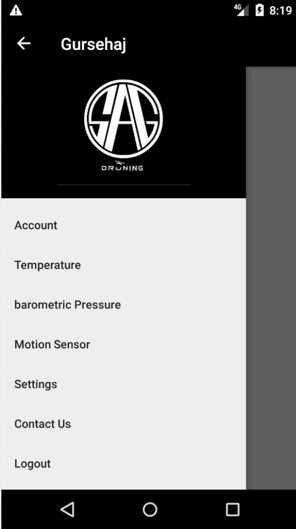
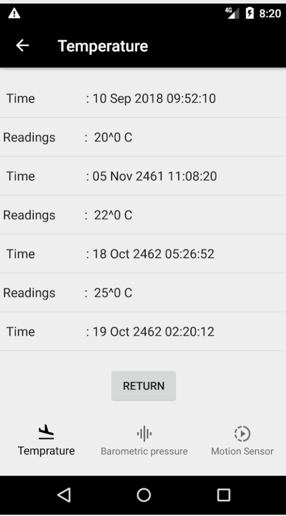
there are two **EditText** boxes that are used to get the user’s name or password. There is a **CheckBox** that is used to save user’s password and email if checked. The password is saved using the **Shared Preferences.** There are 3 **buttons** that are used to do multiple things as their name states. There is a **Toggle Button** that is implemented in order to make it the application work online or offline.

The home page has a couple of new Components used,

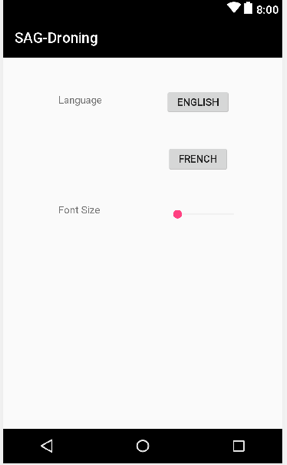
first is a **TextView** that is there to guide the user as to what can be done on this page.

Second, there are 3 **Cardview** on top of each there that can change their visibilities as per the user’s choice.

Third, there is a **Listview** that shows the most recent reading from the database.



There is a **Navigation Drawer** as well as a **Bottom Navigation Bar** for easier navigation

At last there is a **SeekBar** that is used to increase the font size if the user wishes to do it so

Overall the whole team was dedicated in doing all the tasks. The proposal was created by Arman. Shubham was focusing on the mock Designs from the application. Gursehaj at the same time was working on the Software Requirement specification document. We all were working as a team by distributing the work among each individual. The next step, Gursehaj and Arman were working on the research about how to use android studio and its functions. In the meantime Shubham was busy creating the xml files and labelling everything on the xml files. Once the xml files were ready Gursehaj started to create the code for login and registration page. He also worked on the online authentication and offline authentication. Arman researched on Database creation and connecting it with the App. He also figured out the code on reading the values from the database and displaying it in the App. Gursehaj and Arman together worked on Pages like contact us, side Menu, App Logo and Account page. Unfortunately, some of the work here was assigned to Shubham as well but because he left the Semester the Project was completed by the remaining team members.

In conclusion, the application was able to meet the criteria’s that were given to us at the start of the semester. SAG-Droning is still in beta phase but we are on track as per our schedule into making it a full fledged application. i.e. SAG-Droning V1.0.

References: -

In order to use these components,

For Navigation bar,

<https://www.youtube.com/watch?v=AS92bq3XxkA&t=129s>

For Navigation Drawer,

<https://www.youtube.com/watch?v=fGcMLu1GJEc&t=281s>

For Cardviews,

<https://www.youtube.com/watch?v=VUPM387qyrw>

For Shared Preferences,

<https://developer.android.com/training/data-storage/shared-preferences>

for all small enquires and questions our team had while developing this application,

<https://developer.android.com/docs/>

Prototype Design1(Arman Velani) –

<https://user-images.githubusercontent.com/43188450/49106925-70e38880-f252-11e8-94d2-0dfc12e6c991.jpeg>

Prototype Design2(Gursehaj Harika) –

<https://user-images.githubusercontent.com/43185907/48949567-60f03f80-ef06-11e8-90fd-74825a6f81f5.jpeg>

**GitHub Link for the Application**

<https://github.com/GursehajHarika/SAG-Droning>