**Introduction**

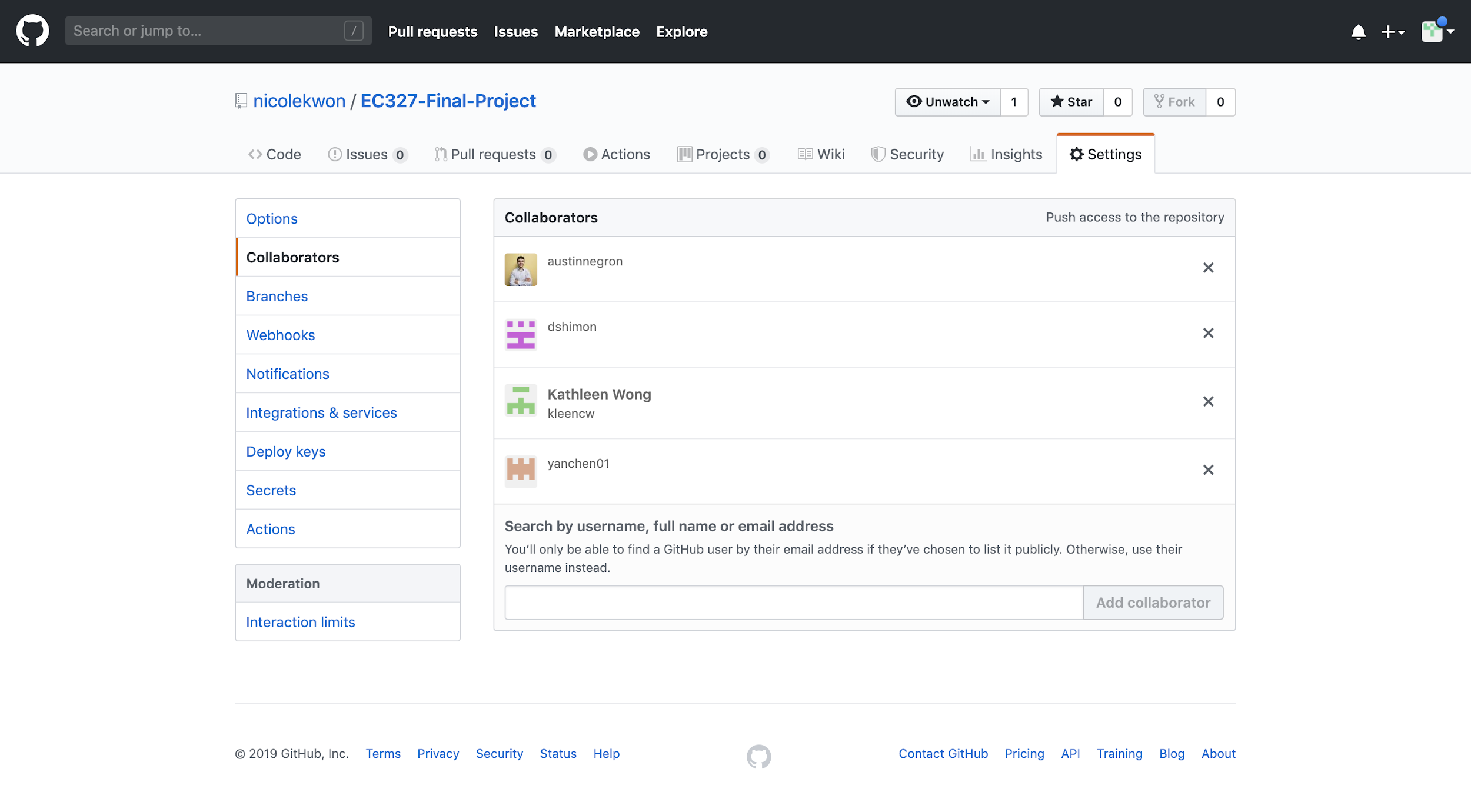
With the digital age currently happening, there is an ever-growing reliance and increase of technology use such as e-commerce. Our original idea involves creating an online marketplace for specifically university students where they can buy, rent, and sell their textbooks, stationery, and other school-related products. Although there are Facebook groups and other websites such as Craiglist that allow for students to do these things, it is disorganized and is not individualized for a specific university. By creating this Android app, the process for both parties of buying/renting or selling/renting would be much more quick and convenient. There will also be a feature to bid the products placed on the platform if the product is being sold.

However, due to the time constraint, the group decided to focus solely on Boston University students and implementing a buy, sell, and shopping cart feature for textbooks. There is no renting or bidding feature, but if there was more time, the group would have done so. The group still managed to create a functioning app that allows the user to register, log in, sell their books, and buy other people’s books. We made sure to make it aesthetically pleasing and geared towards students through a clean and minimalist design. In order to make the app appealing and convenient for Boston University students, they can buy books specific to a particular college.

**What Needs to be Implemented**

The group still needs to update the code for the buying and shopping cart fragments. The linked lists to display the different books available and their images to buy from the Firebase have not been implemented yet, and the group must focus on doing so over the next few days.The group will also try to implement a way to add up the prices of the books that the user is interested in buying to output a total. This will be possible as the information for the prices of the book are on the Firebase. Although the next button on the shopping cart page reroutes to a check out fragment, this fragment has nothing implemented, and the group will need to work on the frontend and the backend to allow the user to input their information to buy the specific book. However, it is not feasible with the given time constraint and the tools we have to set up a successful transaction mechanism through credit card information for the user to buy the book. The next few days will also be focused on discussing what other possible transaction methods can happen.

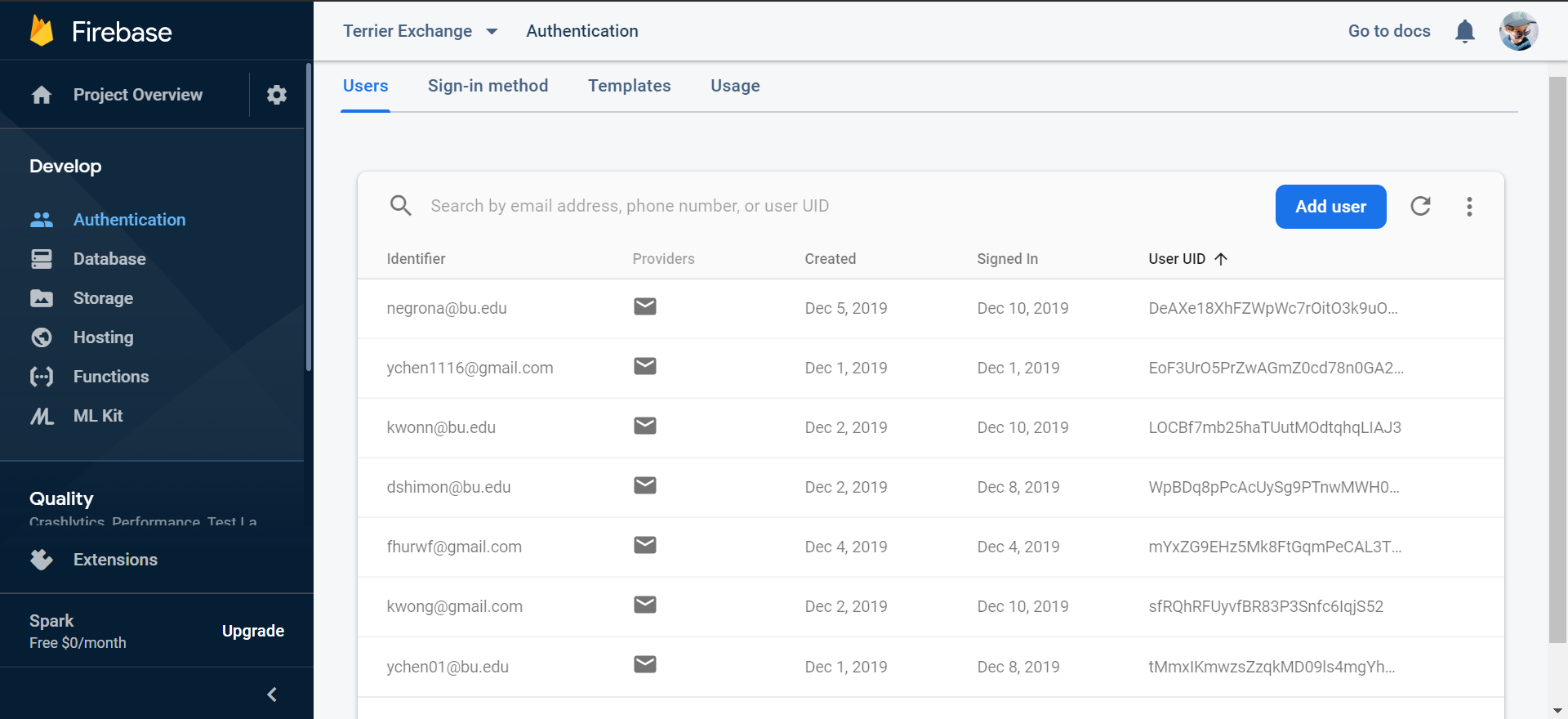
**Step 1: The Github repository was made and shared with everyone. This repository is for creating an Android app that acts as an online marketplace for specifically university students where they can buy, rent, and sell their textbooks, stationery, and other school-related products.**



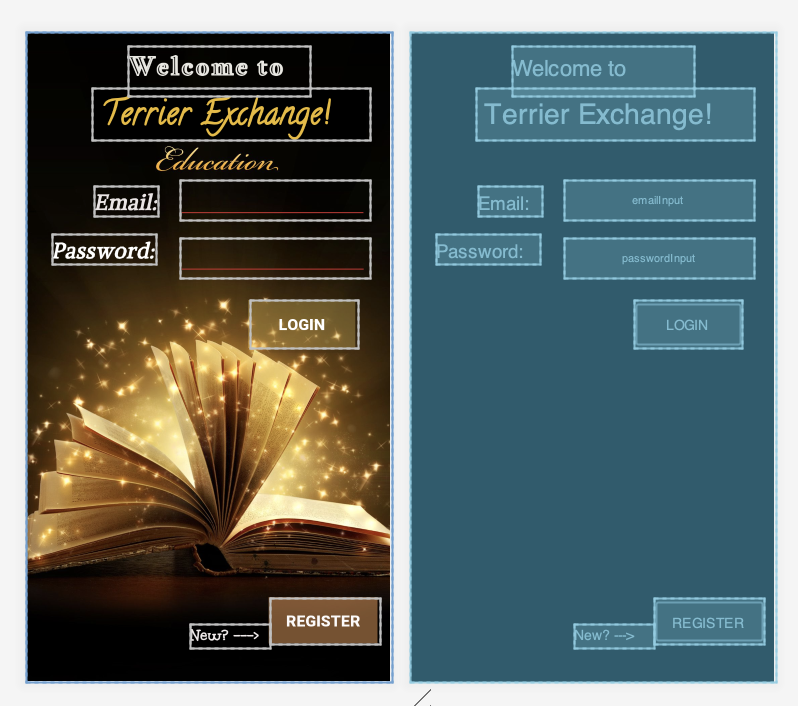
* The group additionally allocated the team roles in order to facilitate a group dynamic with each other.
  + Project Lead: Austin
  + Specification Lead: Daniel
  + Interface Lead: Kathleen
  + Technical Lead: Yan
  + Documentation Lead: Nicole

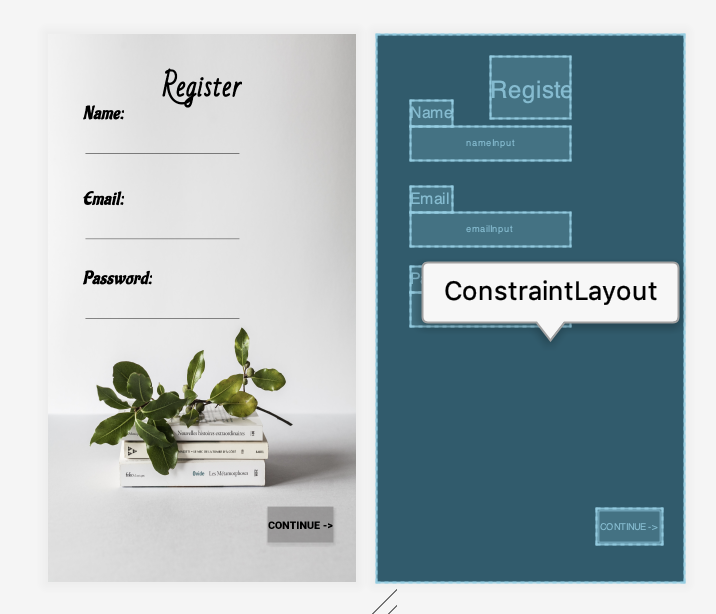
**Step 2: The frontend and backend code for the login page design were created. This was done so by creating a main activity page that acts as the login page. There is a button underneath the fields where the user can input their login information that will reroute the page to the user’s dashboard. There is also a register button on the bottom that allows the user to create an account. The accounts will be saved in an online firebase in order for the account to not be deleted once the user exits the app. The group then added two activity pages: one for the register page, and another for the dashboard once the user logs in. These buttons reroute to those respective activity pages.**

* The group found and uploaded photos onto Android Studio for the background for both the login page and the register page. These photos took a bit to find as we wanted to make it obvious that this is an app that focuses on education and university students.
* Google’s Firebase Authentication provides backend services, easy to use SDKs, and ready-made UI libraries to authenticate the users of the app. The group was able to implement authentication of logging in through FirebaseUI Auth, which is a drop-in auth solution that handles UI flows for signing in users with email addresses and passwords, and registering through Firebase SDK Authentication, which provides methods to create and manage users that use their email addresses and passwords to sign in.



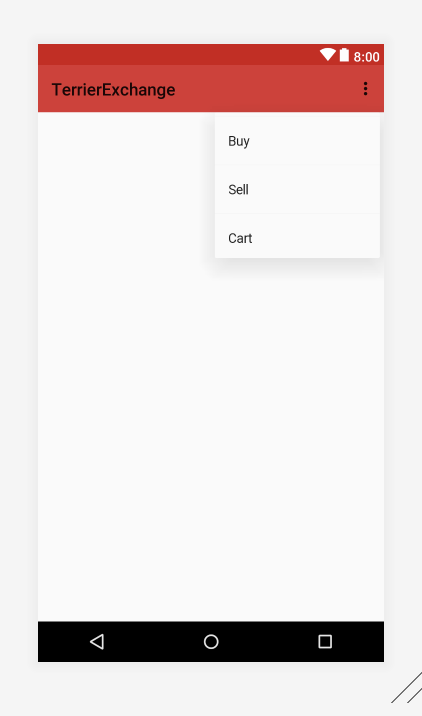
* Main Activity → This page sets up the first page that the user sees when opening the app. It provides the user with the option to login or to register an account.
  + “activity\_main.xml” → The frontend of this activity page involves setting up the background, the buttons to login and register, and the text fields for the user to input their email and password. In addition, the group labelled everything so that the user knows how to navigate the page. The position, shadow, and text parameters were edited for all the text views created to look the way it does in the photos below. The group included edit texts to allow the user to input their email and password, with the login button underneath. On the bottom of the page, there is another textview and it points to the register button.



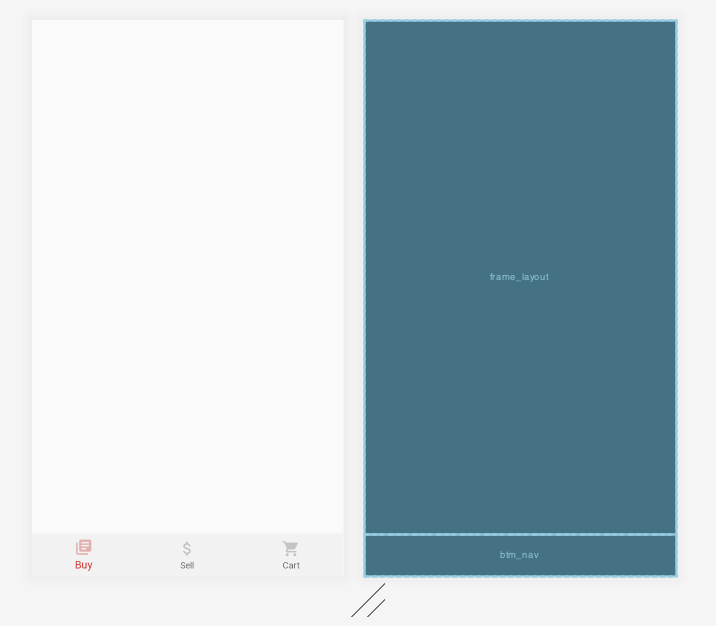
* + “MainActivity.java” → The backend of this activity page involves declaring the firebase and the buttons to reroute the login page to either the register page or the dashboard once you log in. In the public class, the objects for the email input, the password input, the login button, the registration button, and the authentication from the firebase were initialized through the findViewById function.
    - If the user clicks on the login button, this will set an onclick event and will take the email and password as a string and trim any spaces afterwards. There is input validation to check if the user forgot to enter an email, a password, or both. Finally, there is a firebase authentication where it checks to see if the email and password matches with any of the account information stored on the database.
    - If the user clicks on the registration button, this will set an onclick event and creates an intent that will start activity and reroute the user to the registration page.
* Registration Page → This page makes it possible for the user to register an account if he or she did not make one already. It will allow for the user to then use this account being made to log in and proceed with buying or selling what they want. However, because of the time constraint, the group is thinking of focusing on textbooks for now.
  + “activity\_register\_page.xml” → The frontend of this activity page involves setting up the background, the button to continue, and the text fields for the user to input their name, email, and password. Once again, the group labelled everything so that the user knows how to navigate the page. The position, the shadow, and text parameters were edited for all the text views to look the way it does in the photos below. The group included edit texts to allow the user input their name, email, and password, with the continue button underneath. 
  + “RegistrationPage.java” → The backend of this activity page involves declaring the firebase, the edit texts for the input of name, email, and password, and the register button that will then send the information inputted to the database online in the public class. In the protected onCreate function, the objects for the name input, the email input, the password input, and the register button were initialized through the findViewById function. The firebase authentication objects fAuth and fStore were created through the getInstance function.
    - If the user already has an account based on the input, the page will reroute back to the login page without creating another duplicate account. This was checked through the getCurrentUser function, so if it does not return null, it will redirect to the login page.
    - If the user clicks on the register button, this will set an onclick event and take the email, password, and name inputs as strings. There is input validation to check if the user forgot to enter an email, a password, or both. If the password is not greater than 6 characters long, the account will not be made for security purposes and an error message will appear. Finally, there is a firebase authentication that will register the name, email, and password that the user inputted as an account.
      * If the task is successful, there will be a message through the Toast function that outputs that the user created an account.
      * If the task is not successful, there will be a message through the Toast function that outputs that there is an error.

**Step 3: The frontend and the backend of the dashboard were created. This was done so by creating an activity page that acts as the dashboard. There is a navigation bar on the bottom of the dashboard that redirects the user to either buy, sell, or check out what is in the shopping cart. This was done so through creating a menu in the resource folder that contains the frontend to the navigation bar. In addition, the group added three pages that the user can access through the navigation bar: a buy fragment, a sell fragment, and a shopping cart fragment.**

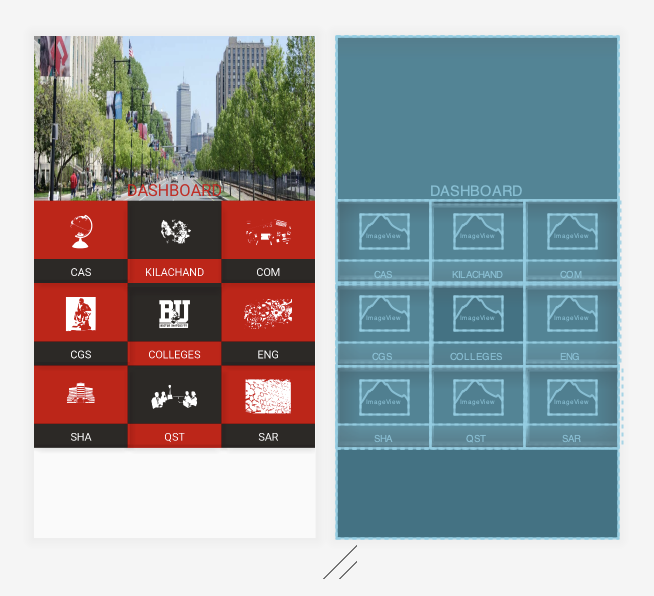
* Navigation Bar
  + “bottom\_nav.xml” → This was set up through an item menu that sets up an id for the buy, sell, and shopping cart fragment. The icons for buy, sell, and cart are vector assets that were set up in the drawable folder.



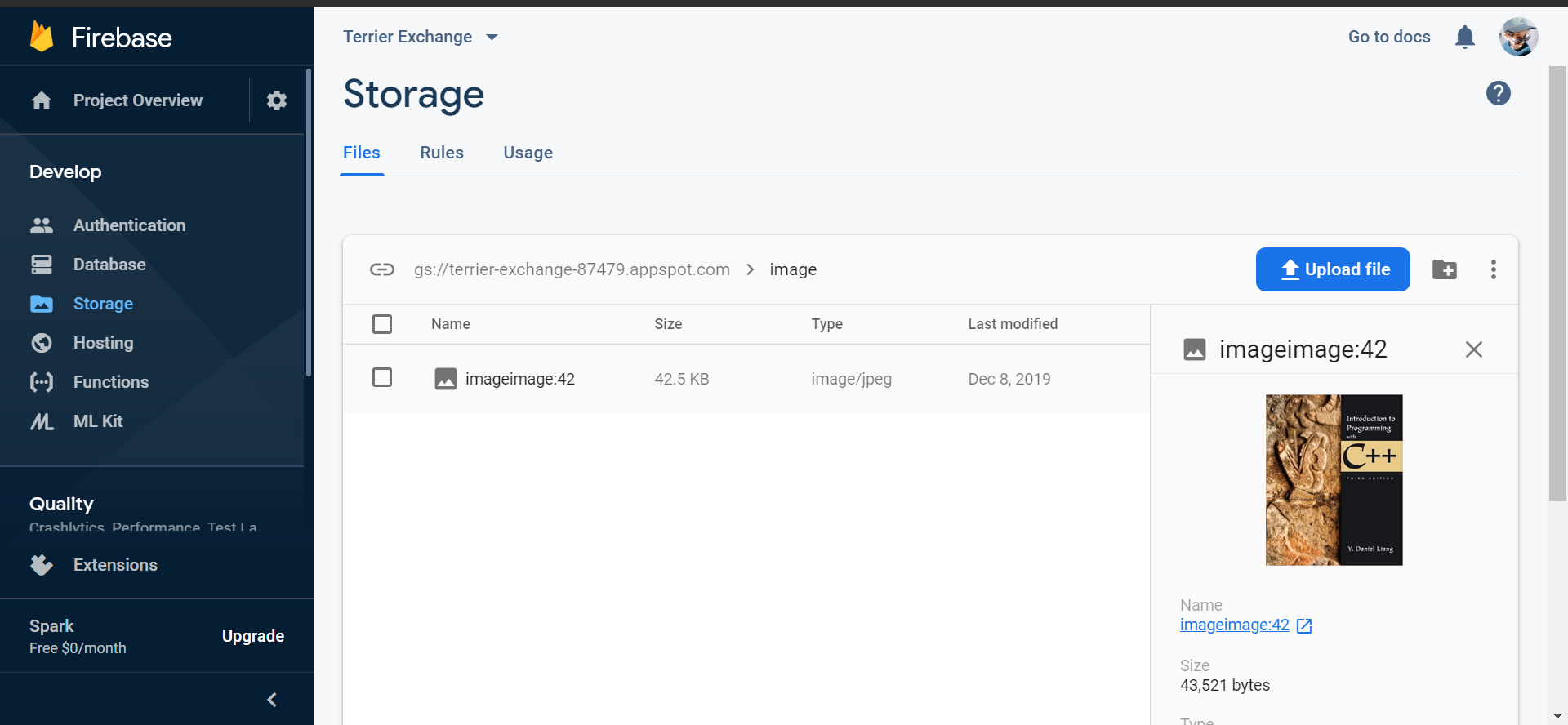
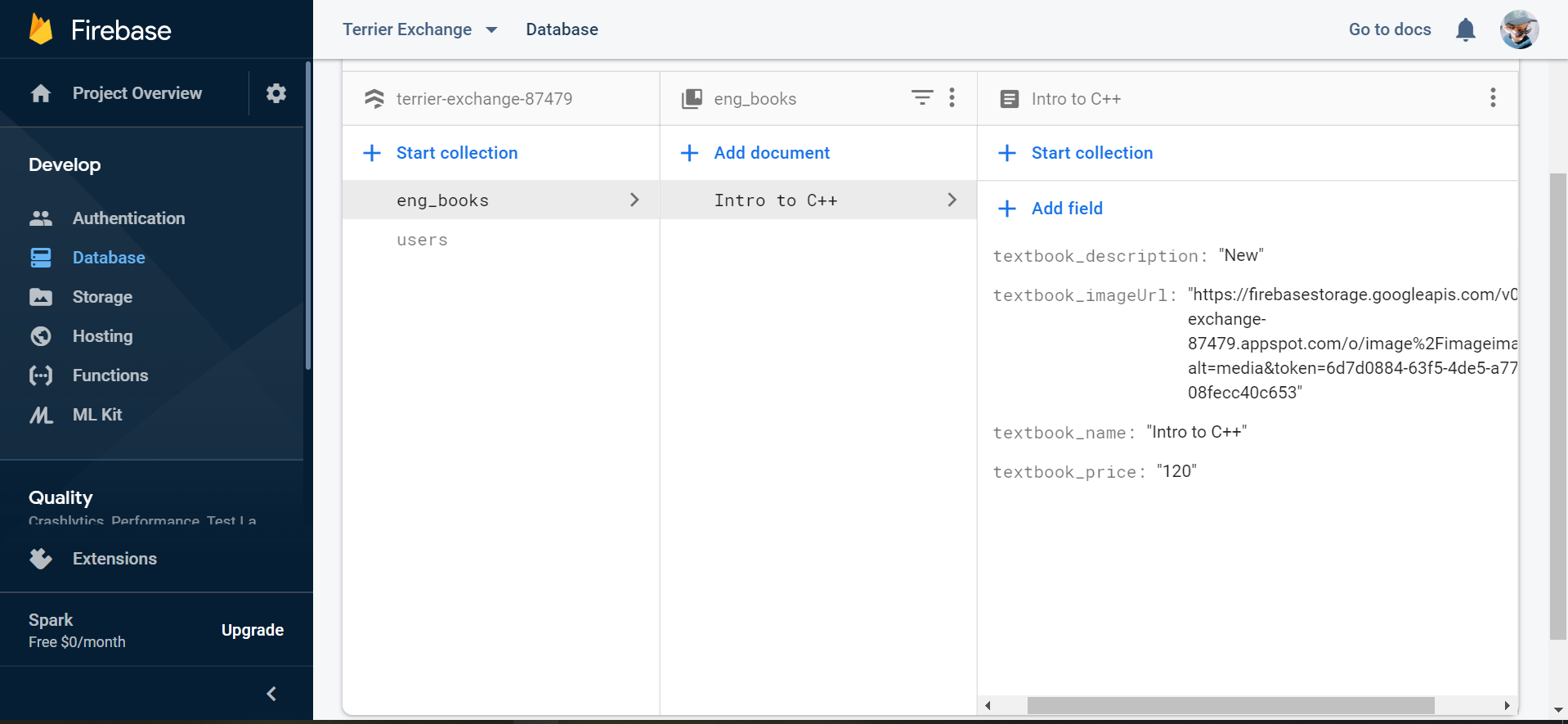
* Dashboard → This page is shown after the user successfully logs in, but in the dashboard specifically, it is an empty frame layout that sets up the navigation bar. It is first automatically set to the buy fragment. On the bottom of the page, the user has the choice to buy, sell, or look at what is in the shopping cart.
  + “acitivity\_dashboard.xml” → The frontend of this activity page involves setting up the navigation bar to be aligned on the bottom of the screen and a frame layout that will allow for rerouting the fragments.

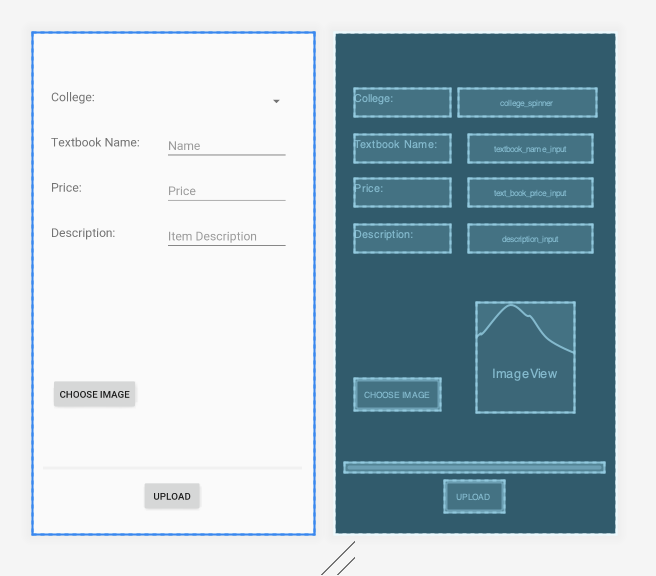


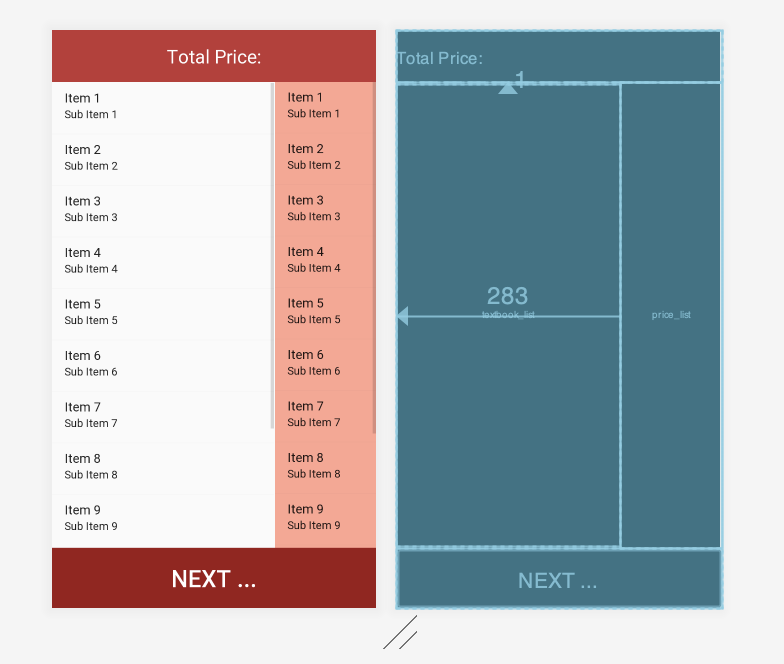
* + “Dashboard.java” → The backend of this activity page involves declaring the firebase in the public Dashboard class. In the protected onCreate function, the firebase and the user id were initialized through the getInstance() and getCurrentUser() functions. The navigation bar declared and initialized a BottomNavigationView object through findViewById() and the id of what the user selected in the navigation bar through getItemId().
    - If the user clicks on the buy menu item, the page will reroute to the buy fragment. It uses a FragmentTransaction object that will replace the frame\_layout in the frontend of the dashboard.
    - If the user clicks on the sell menu item, the page will reroute to the sell fragment. It uses a FragmentTransaction object that will replace the frame\_layout in the frontend of the dashboard.
    - If the user clicks on the shopping cart menu item, the page will reroute to the cart fragment. It uses a FragmentTransaction object that will replace the frame\_layout in the frontend of the dashboard.
* Buy → This page is first shown in the dashboard. There are nine image views that represent the different colleges and eight buttons that reroute to fragments of the respective college chosen by the user.
  + “fragment\_buy.xml” → The frontend of this fragment sets up the background, the image views for the icons, the text views, and the buttons that reroute to specific buy fragments of each college (ex. BuyCasFragment). It was difficult trying to organize this page and setting up the 3x3 grid with the icons and buttons constrained in the right places. The placement of these objects was done so through hard coding in the frontend.



* + “BuyFragment.java” → The backend of this fragment involves declaring and initializing a View object that inflates the layout and buttons for every respect college. Following this, there is an onclick event depending on which button is clicked and reroutes to another fragment. It uses a FragmentTransaction object that will replace the frame\_layout in the frontend of the dashboard.
* Sell → This page is shown when clicked on the bottom navigation bar. It allows for the user to input information about the textbook that they want to sell and uploading it onto the firebase.
  + Cloud Storage from Google’s Firebase Authentication was utilized to store and serve user-generated content, and in this case, the images that users will upload to sell their textbooks. It is a powerful, simple, and cost-effective object storage service, with the Firebase SDKs allowing for the user to upload a photo and storing these photos in a Google Cloud Storage bucket. It also stores the information that the user inputs about the textbook’s description, name, and price.



* + “fragment\_sell.xml” → The frontend of this fragment sets up the spinner that acts as a drop down menu for every BU college, text fields to input the name of the textbook, the price, and the description, a button to choose an image, and a button to upload. 
  + “SellFragment.java” → The backend of this fragment involves declaring the firebase objects, an image view of the photo uploaded, a progress bar, and a string of the image url in the public class. Following this, the public onCreateView inflates the layout for this fragment. It declares and initializes the objects of the spinner for the dropdown menu; the edit text of the book name, the book description, and the book price; the buttons for the upload and image; the activity to upload an image; and a progress bar for processing the upload of an image through findViewById(). The image store was also declared and initialized.
    - An array adapter was declared and initialized for the spinner to implement a drop down menu through setDropDownViewResource() and setAdapter().
    - Onclick events were made for imageBtn and uploadBtn.
      * For imageBtn, the image view was declared and openFileChooser() was called. This function creates a new intent and starts the activity to pick an image.
      * For uploadBtn, a hashmap was created that sorts through what the user chose for the college relating to the textbook. The information for the textbook name, price, description, and image url are sent to the Firebase. If this process is successful, a message is shown through a toast that says “item added.” If this process is unsuccessful, a message is shown through a toast that outputs an error.
* Shopping Cart → This page is shown when the user wants to look at what textbooks they are interested in buying and checking out. Although not implemented yet, the shopping cart should show the user the price of each book and the total price of all the books they added. There is a next button on the bottom of the page that allows the user to go onto the checkout page.
  + “fragment\_cart.xml” → The frontend of this page sets up a text view labelling total price, linked lists for the textbook names and their respective prices, and a button that is labelled next. This button will reroute to a checkout page.



* + “CartFragment.java” → The backend of this page inflates the layout for this fragment. Following this, the public onCreateView inflates the layout for this fragment. It initializes and declares the object for the next button through findViewById() and uses a FragmentTransaction object that will replace the frame\_layout in the frontend of the dashboard to the checkout fragment.