Deliverable 3

Team Name: SmartGate

Project Name: Automated Barrier Gate System

Group Number: 2

Team Members: Richard Pancham (student ID: n01373454)

Jaspreet Heer (student ID: n01315290)

Table Of Contents:

Brief Description of Project:

Our project aims to create an automated barrier gate system that combines IR beam sensors, a luminosity sensor, and a mobile application to make vehicle entry and exit easier. The main goal is to book a parking spot and pay for it within the application then use the application to automatically raise or lower the barrier gate accordingly.

Name	ID	Signature	Effort
Richard Pancham	n01373454	Richard	95%
Jaspreet Heer	n01315290		95%

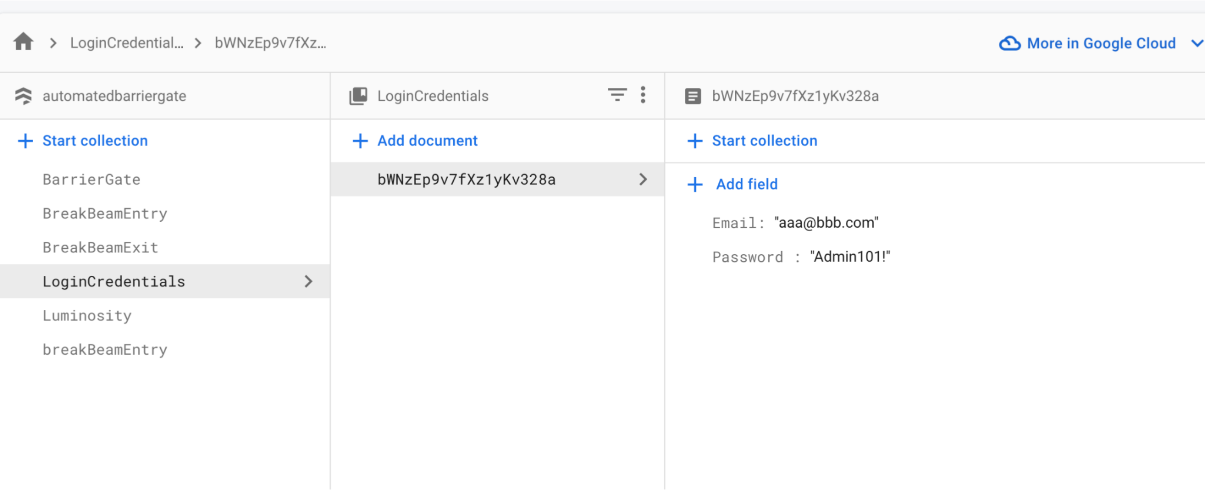
GitHub Repo link

https://github.com/RichardPancham/AutomatedBarrierGate

Login Credentials:

Email: aaa@bbb.com Password: Admin101!

screenshot showing the account in the DB:



Email: "aaa@bbb.com"

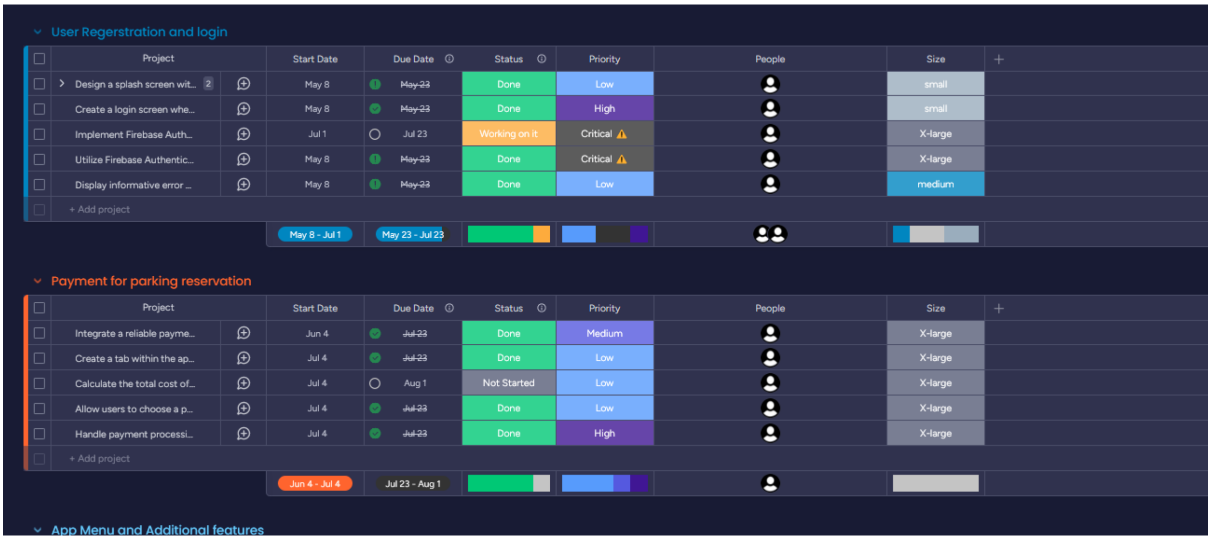
Password : "Admin101!"

Sprint goals:

Our main goal for this task is to develop a fragment dedicated to reviews, where users can share their thoughts and feedback ,and incorporate a user login system where user id and password is stored in the firebase, ensure that the main menu functions are fully implemented, allowing users to navigate seamlessly through the various features of the system and lastly, incorporate a payment validation system.

Monday link :

https://view.monday.com/4777913677-423f7f1770f4602ee247e0f42ee02f42?r=use1



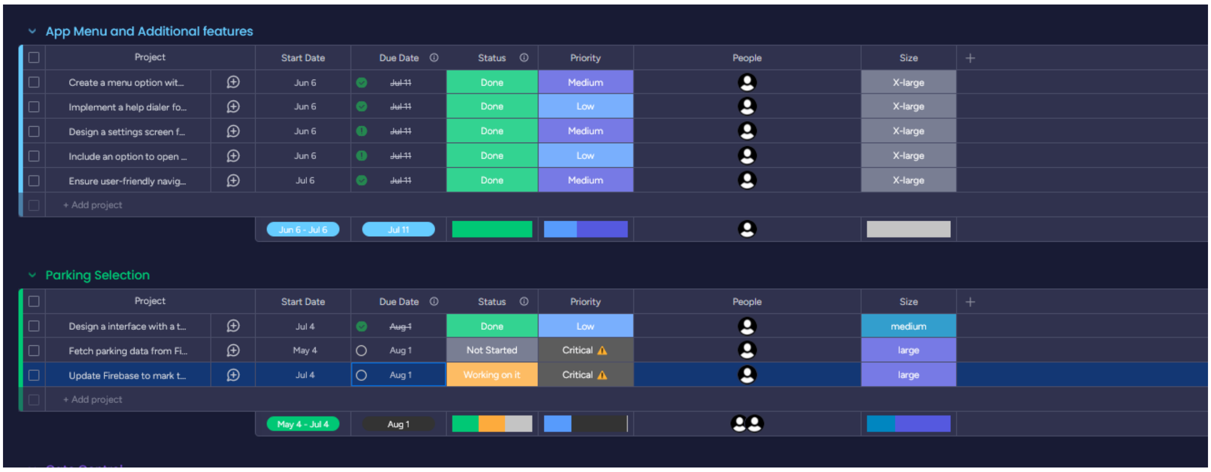
User Regerstration and login

□	Project	Start Date	Due Date	Status	Priority	People	Size
□	Design a splash screen wit… 2	May 8	! ~~May 23~~	Done	Low		small
□	Create a login screen whe…	May 8	✓ ~~May 23~~	Done	High		small
□	Implement Firebase Auth…	Jul 1	○ Jul 23	Working on it	Critical ⚠		X-large
□	Utilize Firebase Authentic…	May 8	! ~~May 23~~	Done	Critical ⚠		X-large
□	Display informative error …	May 8	! ~~May 23~~	Done	Low		medium
□	+ Add project						
		May 8 - Jul 1	May 23 - Jul 23				

Payment for parking reservation

□	Project	Start Date	Due Date	Status	Priority	People	Size
□	Integrate a reliable payme…	Jun 4	✓ ~~Jul 23~~	Done	Medium		X-large
□	Create a tab within the ap…	Jul 4	✓ ~~Jul 23~~	Done	Low		X-large
□	Calculate the total cost of…	Jul 4	○ Aug 1	Not Started	Low		X-large
□	Allow users to choose a p…	Jul 4	✓ ~~Jul 23~~	Done	Low		X-large
□	Handle payment processi…	Jul 4	✓ ~~Jul 23~~	Done	High		X-large
□	+ Add project						
		Jun 4 - Jul 4	Jul 23 - Aug 1				

App Menu and Additional features



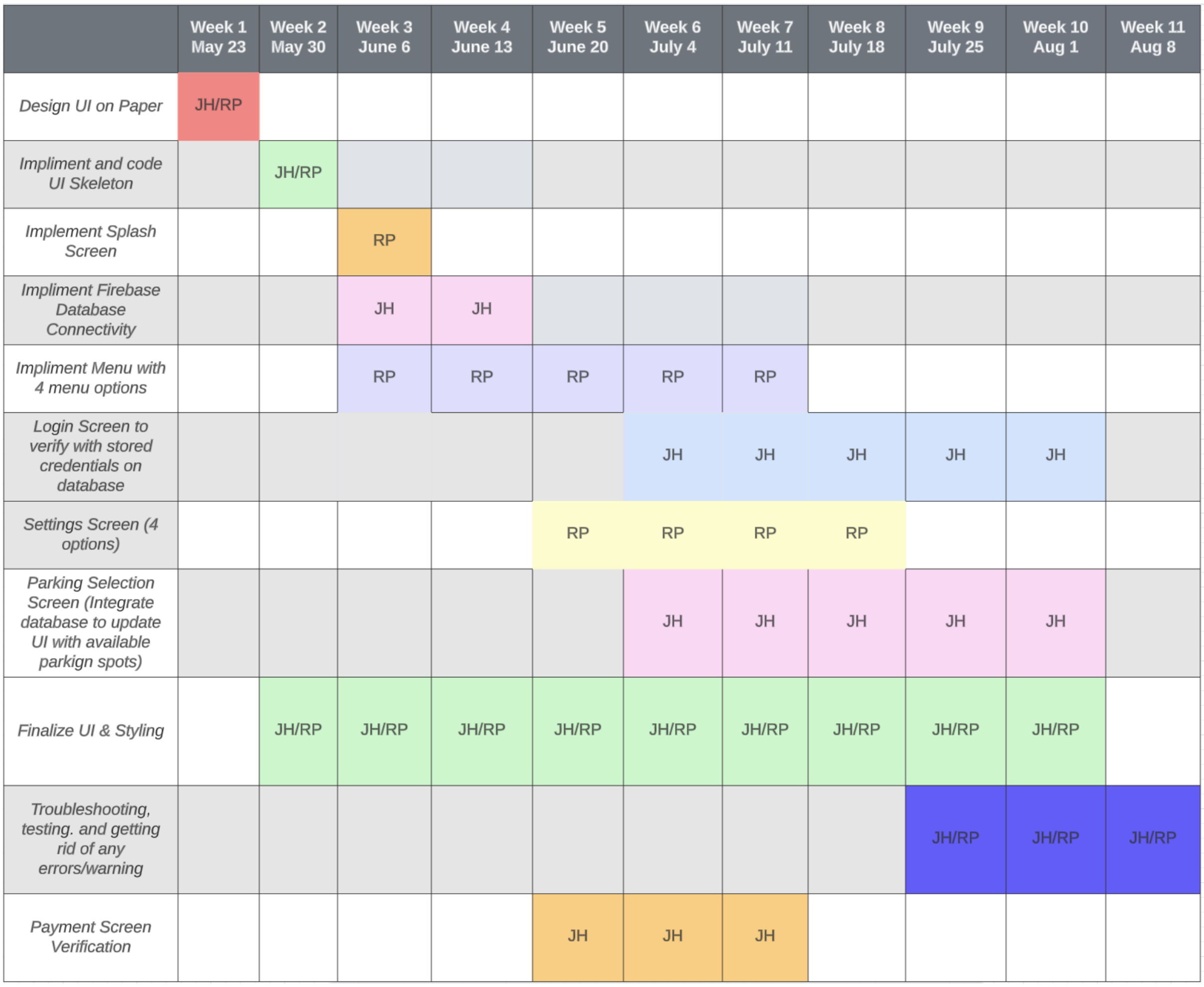
App Menu and Additional features

□	Project	Start Date	Due Date	Status	Priority	People	Size
□	Create a menu option wit…	Jun 6	✓ ~~Jul 11~~	Done	Medium		X-large
□	Implement a help dialer fo…	Jun 6	✓ ~~Jul 11~~	Done	Low		X-large
□	Design a settings screen f…	Jun 6	! ~~Jul 11~~	Done	Medium		X-large
□	Include an option to open …	Jun 6	! ~~Jul 11~~	Done	Low		X-large
□	Ensure user-friendly navig…	Jul 6	✓ ~~Jul 11~~	Done	Medium		X-large
□	+ Add project						
		Jun 6 - Jul 6	Jul 11				

Parking Selection

□	Project	Start Date	Due Date	Status	Priority	People	Size
□	Design a interface with a t…	Jul 4	✓ ~~Aug 1~~	Done	Low		medium
□	Fetch parking data from Fi…	May 4	○ Aug 1	Not Started	Critical ⚠		large
□	Update Firebase to mark t…	Jul 4	○ Aug 1	Working on it	Critical ⚠		large
□	+ Add project						
		May 4 - Jul 4	Aug 1				

Gantt chart:

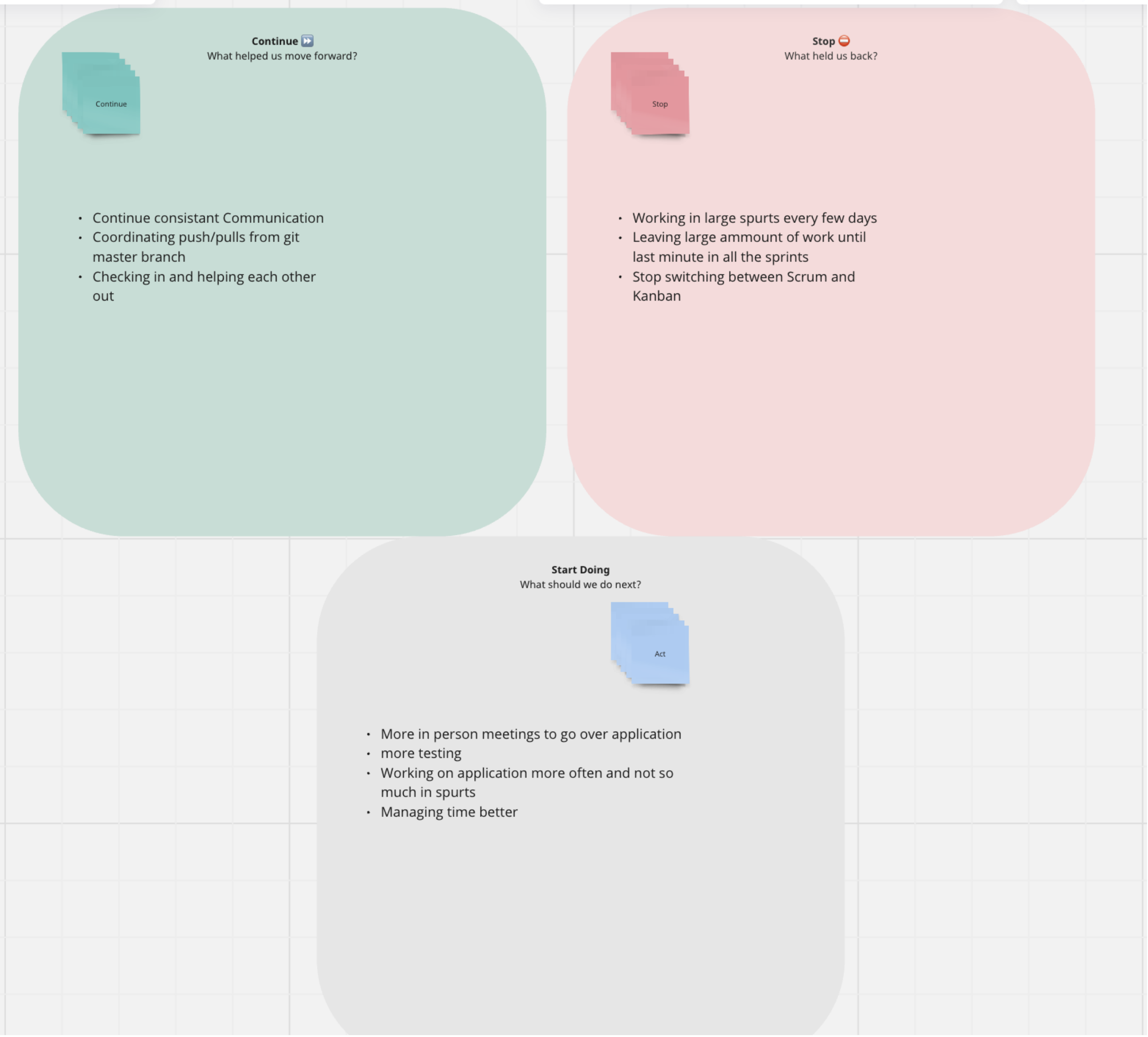


	Week 1 May 23	Week 2 May 30	Week 3 June 6	Week 4 June 13	Week 5 June 20	Week 6 July 4	Week 7 July 11	Week 8 July 18	Week 9 July 25	Week 10 Aug 1	Week 11 Aug 8
Design UI on Paper	JH/RP										
Impliment and code UI Skeleton		JH/RP									
Implement Splash Screen			RP								
Impliment Firebase Database Connectivity			JH	JH							
Impliment Menu with 4 menu options			RP	RP	RP	RP	RP				
Login Screen to verify with stored credentials on database						JH	JH	JH	JH	JH	
Settings Screen (4 options)					RP	RP	RP	RP			
Parking Selection Screen (Integrate database to update UI with available parkign spots)						JH	JH	JH	JH	JH	
Finalize UI & Styling		JH/RP	JH/RP	JH/RP	JH/RP	JH/RP	JH/RP	JH/RP	JH/RP	JH/RP	
Troubleshooting, testing. and getting rid of any errors/warning									JH/RP	JH/RP	JH/RP
Payment Screen Verification					JH	JH	JH				

Daily Standup:

Date:	Outcome
2023-07-05	Progress made researching database integration for hardware and application prior to meeting. Discussed settings screen options during the meeting.
2023-07-09	Progress made on creating deliverable 3 document and updating gantt chart with accordingly adjusted dates. Login screen verification will be worked on today.
2023-07-12	Progress made on settings screen. Planned how to integrate parking spots in database.

Sprint Retrospective:



- Continue consistant Communication
- Coordinating push/pulls from git master branch
- Checking in and helping each other out

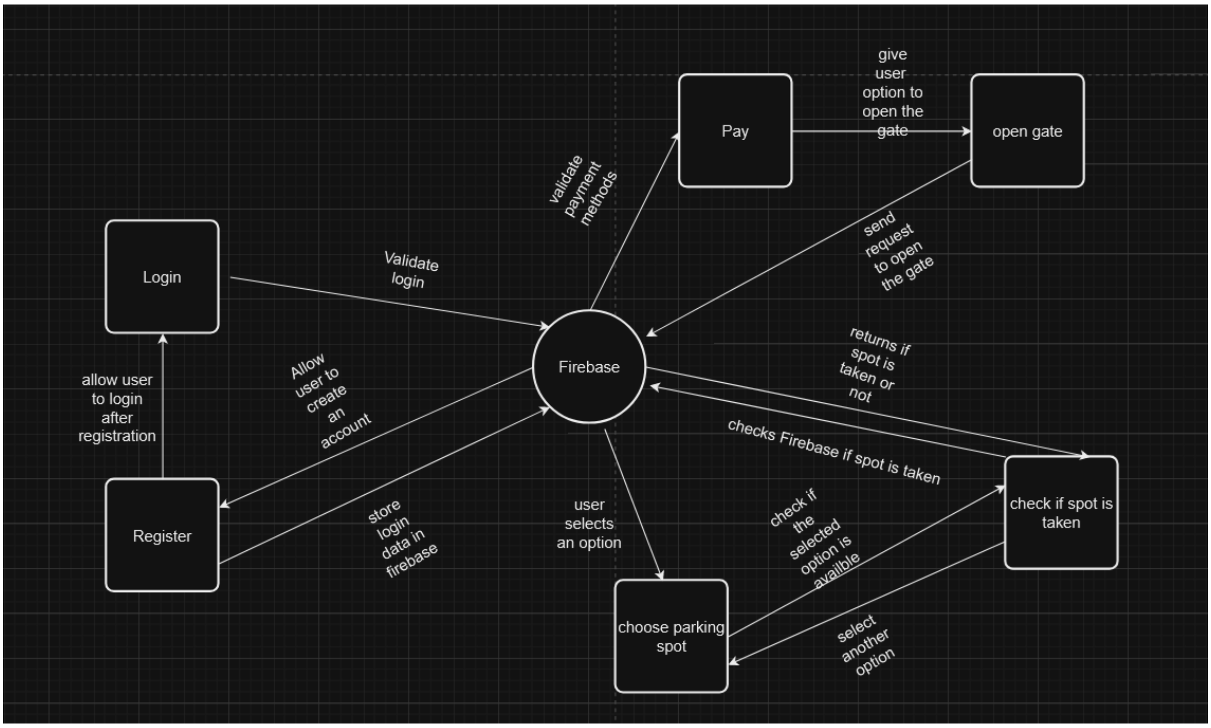
- Working in large spurts every few days
- Leaving large ammount of work until last minute in all the sprints
- Stop switching between Scrum and Kanban

- More in person meetings to go over application
- more testing
- Working on application more often and not so much in spurts
- Managing time better

System Context Diagram

Link for System Context Diagram:

https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1#G1xyF7p19Dz_Bs3zbBefpedZ1egUUX6dtE



Pay
give user option to open the gate
open gate
validate payment methods
Login
Validate login
send request to open the gate
Firebase
returns if spot is taken or not
Allow user to create an account
allow user to login after registration
checks Firebase if spot is taken
Register
store login data in firebase
user selects an option
check if the selected option is availble
check if spot is taken
choose parking spot
select another option

two different design principles

32. Document two different design principles used in the code.
Copy the code you used, and add your explanations.
CENG-322 5

Design Principle 1: Single Responsibility Principle (SRP) The code fragment that demonstrates the Single Responsibility Principle is the `switchFragment()` method:

```
private void switchFragment(Fragment fragment) {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.fragmentContainer, fragment);
    fragmentTransaction.commit();
}
```

This method has a single responsibility, which is to switch the current fragment in the `fragmentContainer` view. It encapsulates the logic of fragment switching and ensures that each method is responsible for one specific task.

Design Principle 2: Open/Closed Principle (OCP) The code fragment that demonstrates the Open/Closed Principle is the `onOptionsItemSelected()` method:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.settings:
            switchFragment(settingsFragment);
            return true;
        case R.id.web:
            // Handle menu item 2 click
            return true;
        case R.id.dialer:
            // Handle menu item 3 click
            Intent intent = new Intent(Intent.ACTION_DIAL);
            String phoneNumber = "1234567890"; // Replace with the
desired phone number
            Uri phoneUri = Uri.parse("tel:" + phoneNumber);
            intent.setData(phoneUri);
            startActivity(intent);
            return true;
        case R.id.about:
            return true;
        case R.id.reviews:
            switchFragment(reviewsFragment);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

The code follows the Open/Closed Principle. This means that it's open for adding new features but closed for changing existing ones. The onOptionsItemSelected() method handles different menu options without needing to change the code every time a new option is added. It's flexible and allows for easy updates without breaking existing functionality.

two different design patterns

33. Document two different design patterns used in the code.
Copy the code you used, and add your explanation.

Code Fragment 1: Observer Pattern

java

```
bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item)
{
        switch (item.getItemId()) {
            case R.id.payment:
                switchFragment(paymentFragment);
                return true;
            case R.id.location:
                switchFragment(locationFragment);
                return true;
            case R.id.open_barrier:
                switchFragment(barrierFragment);
                return true;
        }
        return false;
    }
});
```

This code sets a listener for the bottom navigation view. When an item is selected, the corresponding fragment is switched and displayed. The `onNavigationItemSelected()` method acts as an observer, responding to item selection events. It determines the selected item and triggers the appropriate action, such as switching to the payment fragment or location fragment. This makes it easy to handle navigation events and update the UI accordingly.

Code Fragment 2: Factory Method Pattern

```
paymentFragment = new PaymentFragment();
locationFragment = new LocationFragment();
barrierFragment = new BarrierOpenAndClose();
settingsFragment = new SettingsFragment();
reviewsFragment = new ReviewsFragment();
```

The code creates different fragments by directly making objects of their respective classes. However, using a factory method would provide more flexibility and make it easier to add new fragment types without changing the existing code. This improves the application's adaptability and maintainability.

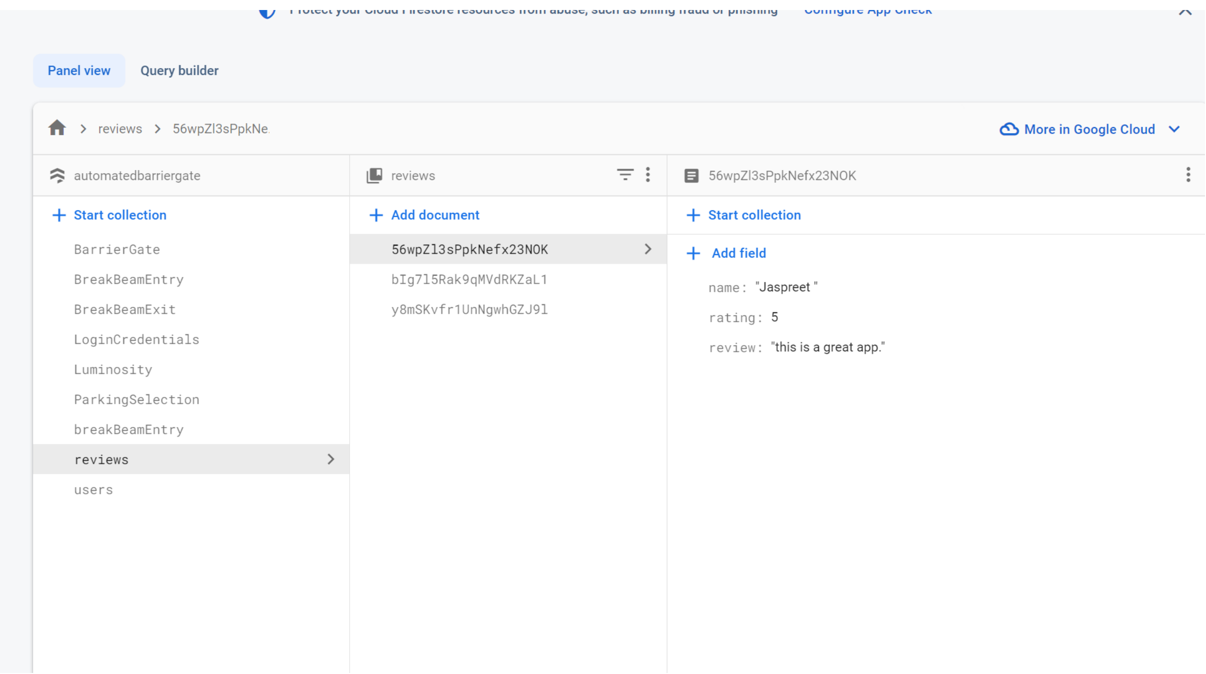
34. Coding work progress since deliverable 2. What additional features/functionality added since deliverable 2.

Since our last deliverable, we’ve made a solid amount of progress in regards to the coding of our application. We have completely revamped the parking selection screen to show a 2d overview of the parking lot. We have completed setting up the menu bar options as well as the settings screen. A reviews screen has been added allowing the users to submit a review of tha application. The biggest feature added has been the login screen functionality allowing users to sign in with credentials which will be verified in the database. Payment screen verification has also been added.

35. Document what runtime permission you have implemented

In our application, we currently have the access fine location and access coarse location runtime permissions implemented. We are working to implement android.permission.CALL_PHONE allowing our phone/support menu option to call directly to our support line.

36. Provide screenshot how the data from the Customer Review Screen stored in the DB



BarrierGate
BreakBeamEntry
BreakBeamExit
LoginCredentials
Luminosity
ParkingSelection
breakBeamEntry
reviews
users

56wpZl3sPpkNefx23NOK
bIg7l5Rak9qMVdRKZaL1
y8mSKvfr1UnNgwhGZJ9l

name: "Jaspreet "
rating: 5
review: "this is a great app."