Deliverable 3

FINAL DOCUMENT LAYOUT

The username and password for app:

Username: r.nissan2010@gmail.com, gbinay24@gmail.com,
Password: 2010nissan, Mobile2022

1.Document Title

Smart Bike Docking System

2.Team Name

Group 6

3.Project Name

The Docky

4.Student's names and IDS

Team Member Names (Please Print)	Signatures	Student ID
Project Leader:	R. Nathan	n01425273

Nathan Ryan		
Semen Dyakonov	D. Semen	n01391812
Nissan Rayappu	R.Nissan	n01435235
Binay Pawan Garlapati	G.Binay pawan	n01368870

5.Table of Contents

6. Project Description

1. The final vision of the Docky system is to have a completely connected network of bike docks around the city. We want it to be accessible to everyone with a simple lock and unlock feature. The Docky app will streamline payments and bike sharing with friends with a simple and easy-to-navigate app.

2. Firstly, we need to buy a solenoid that will push a metal bar in between the bicycle wheel spokes/frame. This prevents the bike from moving and the solenoid won't move until activated by the mobile app. Next, we need a weight sensor to detect if the bike moves out of the spot. This will send a notification to the users' phones telling them their bike has moved out of the docking station. Also, we will need some LED lights. A red one to show that the bike is locked in the dock and a green one if the dock is available. Lastly, we will have a led display that will show how long the bike has been there and the username of the person who parked it there. The app is connected to the Dock and is essential to the function of the circuit. The user input will be sent to the database and the docky system will take the user inputs to move the locks and change the LED display.

3. The app will open with a login screen and the user will input their information to connect the username to the Docky locks. Then the next screen will be a lock and unlock button which will be deactivated until the user goes to the third page. This third page has a GPS location of the bike dock they're at and will also tell the user the status of their bikes. Finally, the 4 pages will be a payment plan/hourly payment page for the user to input credit card info and other online banking options

4. We have been guided to do one fragment per group member. The first fragment is about the login page where you will need to put your credentials. The next fragment is about GPS, so it will display the map which depends on the user's location and will show the user the closest lock

spots. The third one is about lock and unlock buttons. And the last one is the payment page where the user will need to put his card information to pay for the bike locker.

5. Using firebase, we will incorporate user id's and passwords. All information will be stored in our Cloud Database and to retrieve it, you need to verify your identity. After that we will connect our database to Android App and Raspberry PI so that they can work in real-time. When the user try to unlock or lock his bike, our app will automatically make a request to the database, and depending on the answer it will send a signal to our PI. Also, it will store time information about how long the bike was locked and after that our app will make a request to the database and based on the answer will give to our users the right price based on the rate. So our database will store login information and lock time.

7. Signatures Of Team Members

1. G. Binay Pawan
2. R.Nissan
3. R.Nathan
4. D.Semens

8. Members of project and participation

Name	**ID**	**Signature**	**Effort**
Semen Dyakonov	N01391812	S.D.	100%
Nissan Rayappu	N01435235	N.R	100%
Binay Garlapati	N01368870	B.P	100%
Nathan Ryan	N01425273	N.R	100%

9. GitHub Repo link and strategy

https://github.com/SemenDyakonov1391812/TheDocky

10. Verify

11. Each member must have a minimum of 10 commits. Counted starting Oct. 22.

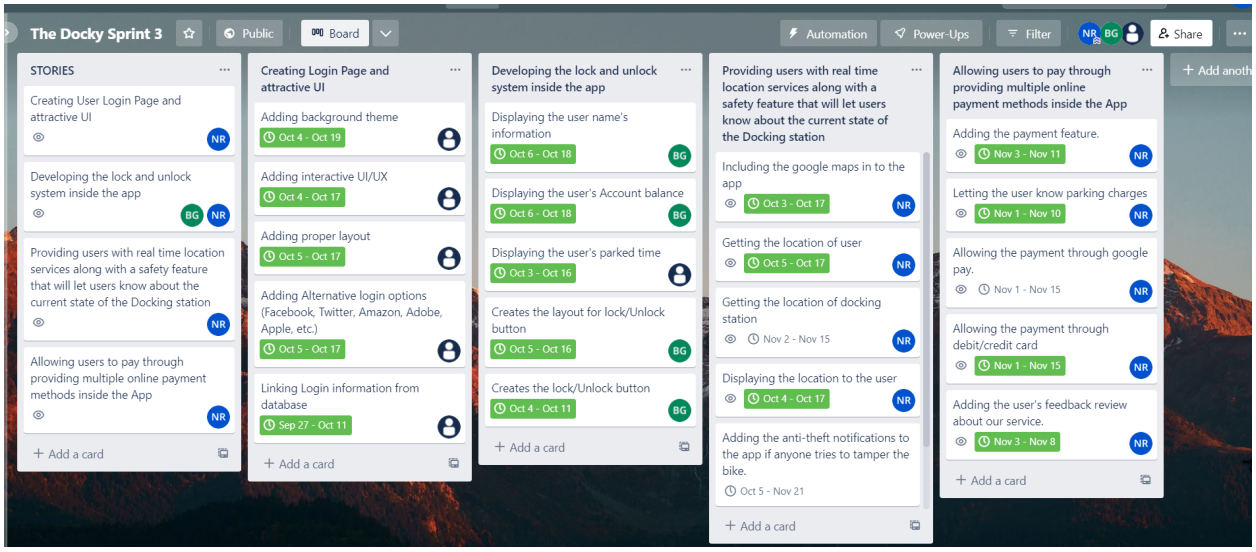
12. Marks deducted for members who are not meeting the minimum 10 commits (i.e. if only 8 commits, 20% percent deducted, …etc.).

13. Sprint goals, list sprint goals.

- Improve customer retention by 20% by implementing a feedback review system.
- Focus on payment fragment and review fragment in this sprint.

14. You must have completed a minimum of 4 stories for Sprint 3. Dashboard should clearly show that. Each story must have minimum of 5 tasks.

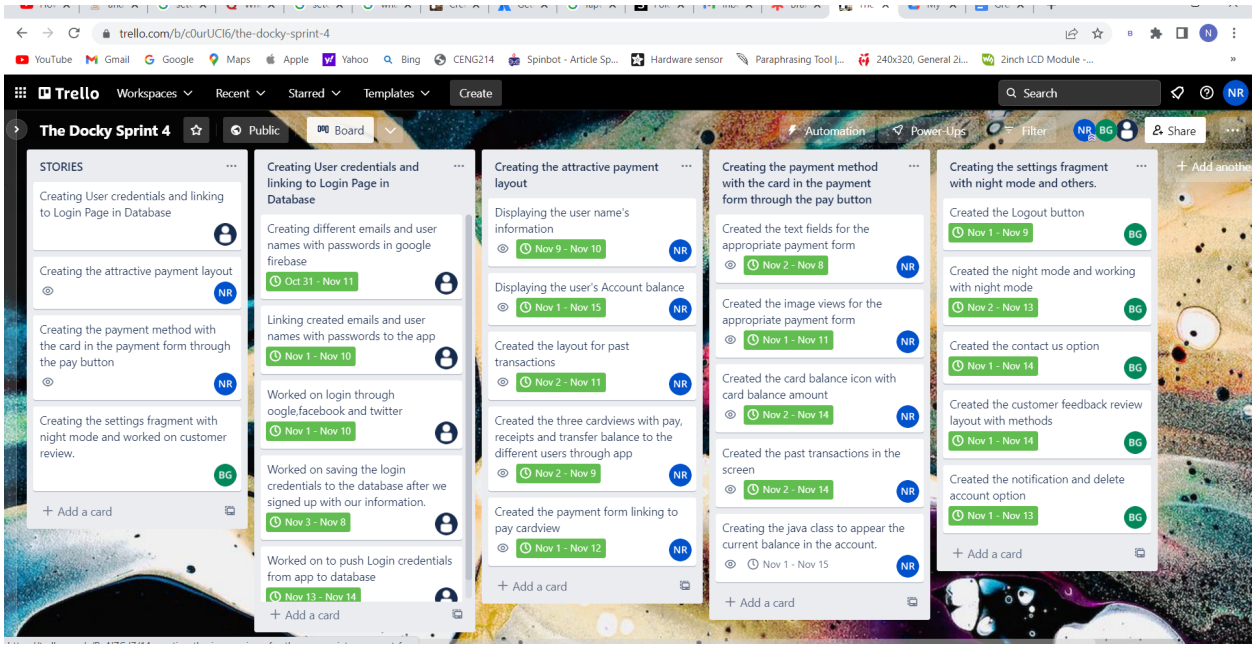
https://trello.com/b/Ezxeql16/the-docky-sprint-3



15. Update the Sprint dashboard, showing Sprint 3 with closed
tasks and tasks you did not complete.

16. Update the Sprint dashboard, showing Sprint 4 with minimum of 4 stories. Each story must have minimum of 5 tasks.

https://trello.com/b/c0urUCl6/the-docky-sprint-4



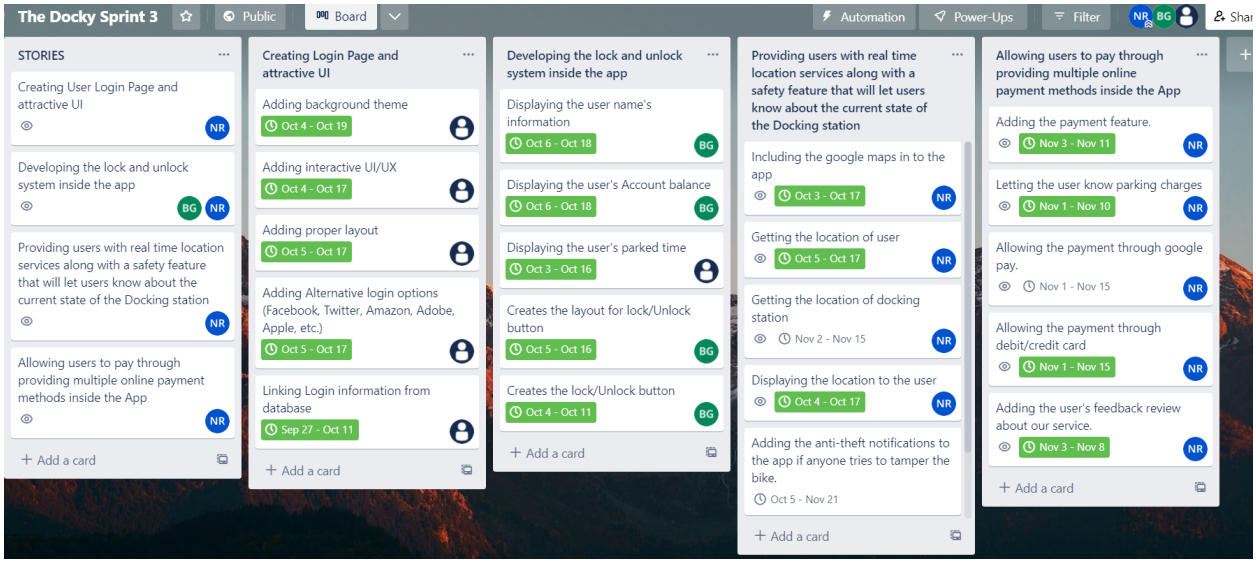
17. Dashboard, should clearly show task, owner, status, start date, end date, and size.

18. Must use a tool such as Trello or Monday. Word, Excel, …
etc. will not be accepted.

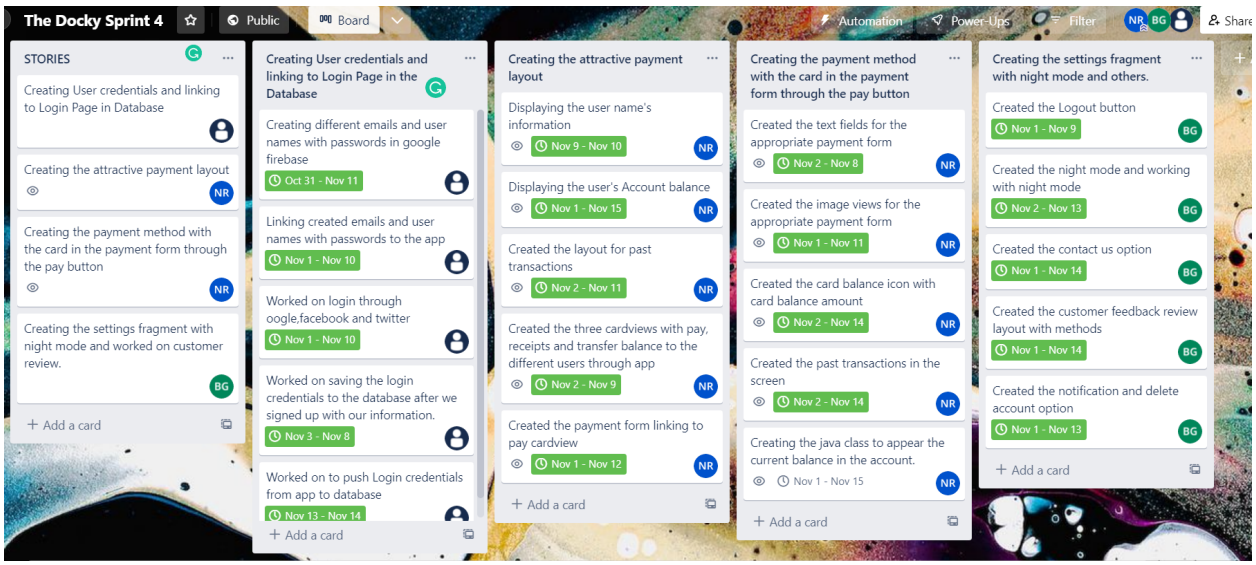
19. Please try to use Scrum and not Kanban.

20. Take a screenshot showing clearly the stories and breakdown of tasks with all details for sprint 3 and 4.

Sprint 3



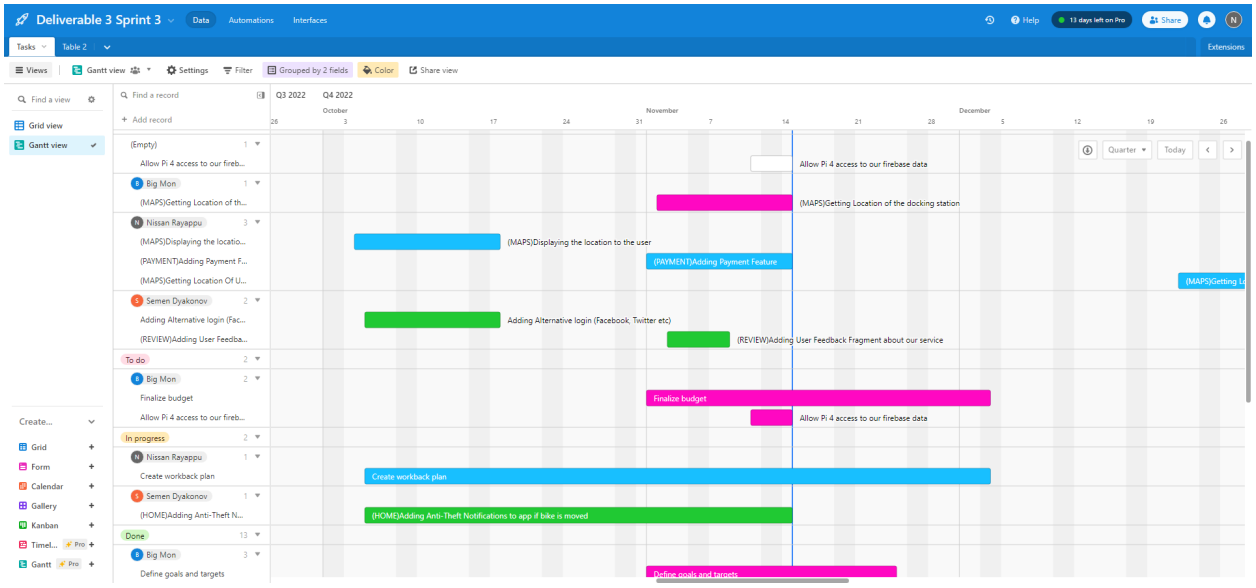
Sprint 4



21. Create gantt chart showing main milestones and work progress. You can use https://try.airtable.com/gantt2 or any other tool. Do the research and select the proper tool. Excel or word will not be accepted.

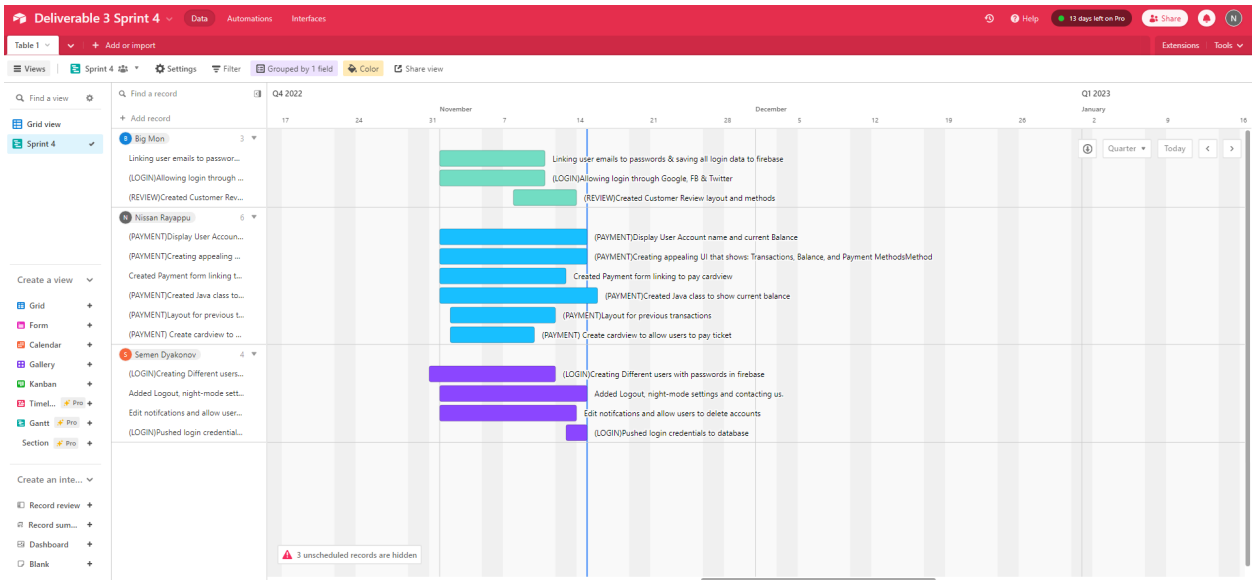
22. The gantt chart must show the work for a minimum of 10 components and the time lines. Focus on main components and not every small task.

Sprint 3



https://airtable.com/appBa4lq9AaEekPJu/tbljwdD8phCbJ1X0v/viwvjYNe2lLqGiNcz?blocks=hide

Sprint 4



https://airtable.com/appmNlzj6pqNPsgm5/tbl7sN9RFAVySx4l4/viwSB9sDU5n0ll7gl?blocks=hide

23. Add a column to show the person assigned. An example of a Gantt chart:

24. Take a screenshot of the gantt chart.

25. Updated records of the daily stand-ups outcome, use any tool you like, include a screenshot into the document. Table include

date, outcome and who missed the daily standup.

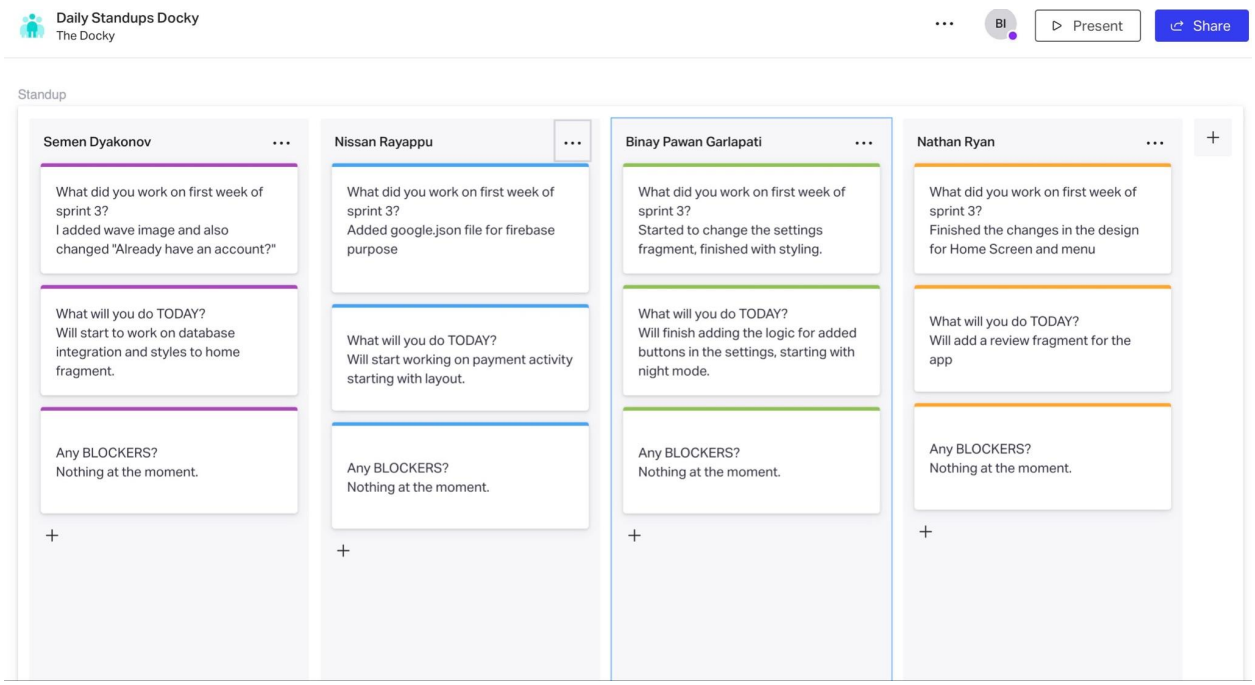
26. Daily standup should answer three questions:

A. **What did you do yesterday?**
B. **What will you do today?**
C. **What (if anything) is blocking your progress?**

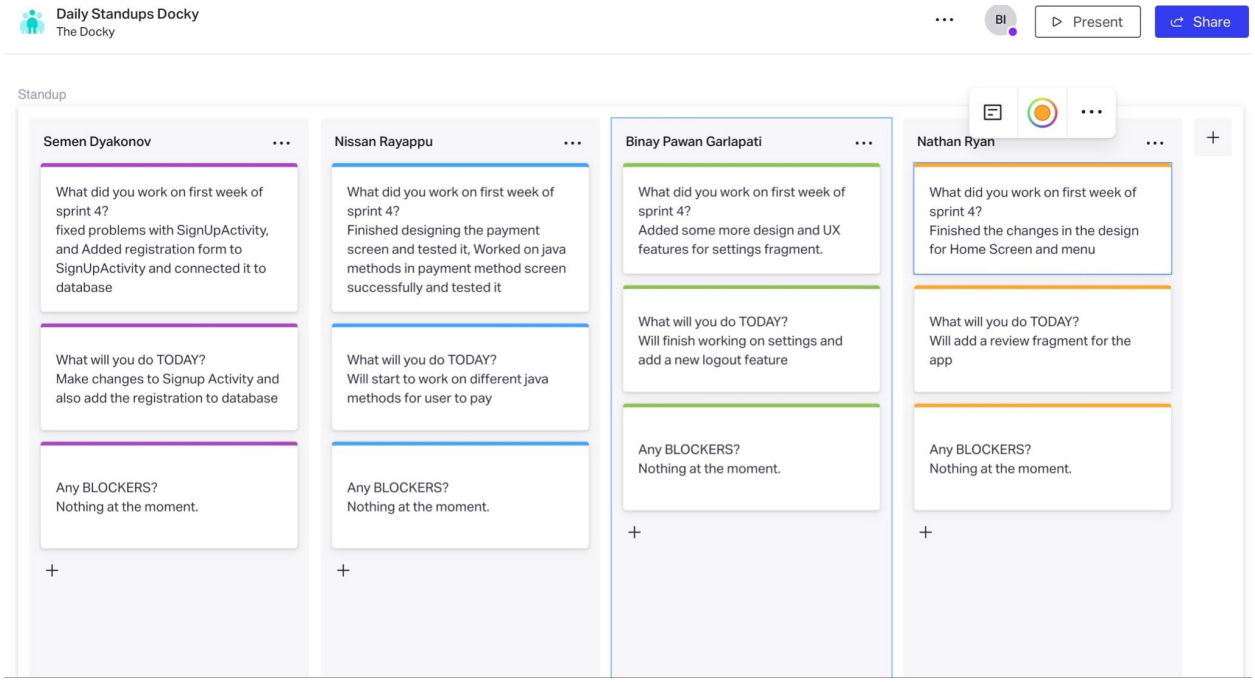
27. Record a minimum of 3 meetings.

Sprint 4

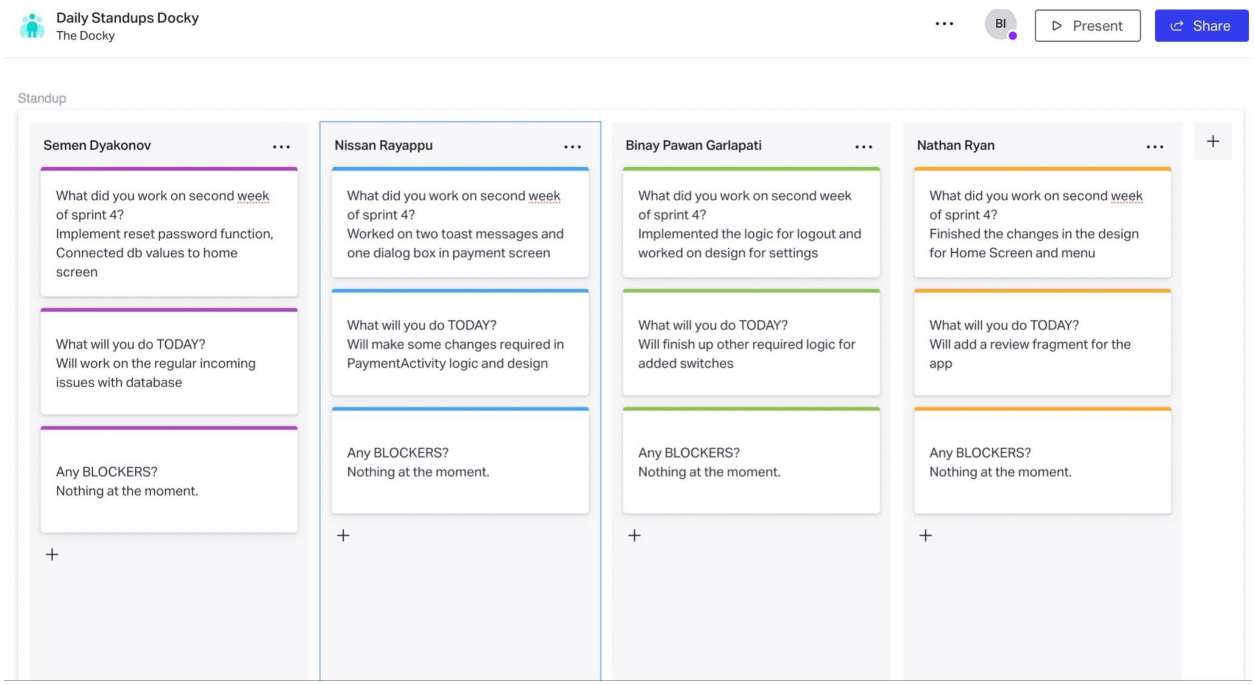
26/10/22



03/11/22:



10/11/22:

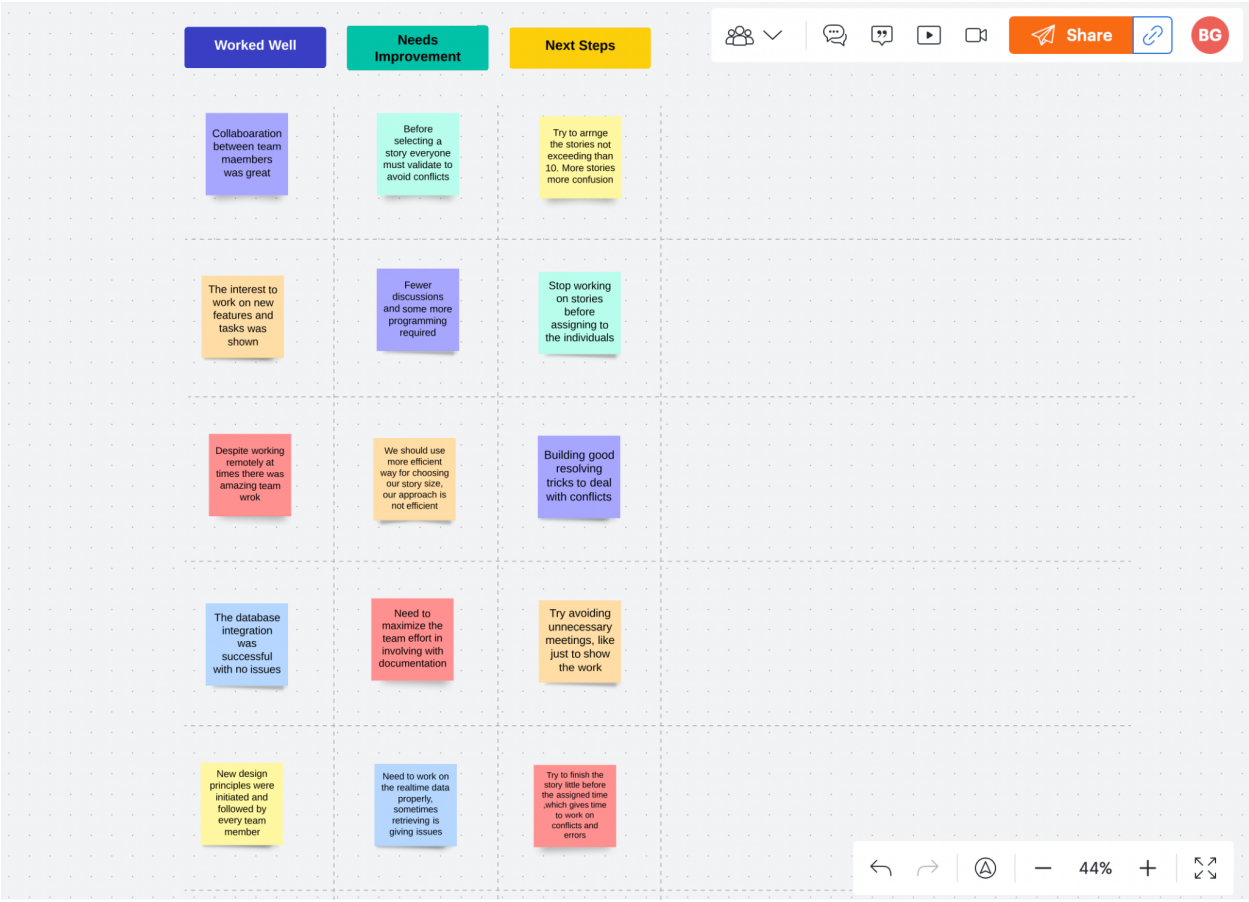


28. Use a tool to record your Sprint Retrospective (i.e. https://lucidspark.com/landing/create/online-sticky-notes

https://miro.com/

1. Who missed the meeting, marks will be deducted for missing

the retro.



https://lucid.app/lucidspark/ef2b46d3-61f4-488f-8b25-7fe09ed8553f/edit?viewport_loc=2023%2C2359%2C3296%2C1804%2C0_0&invitationId=inv_f58e0617-b846-4a47-86ec-1d2fd28e7c44

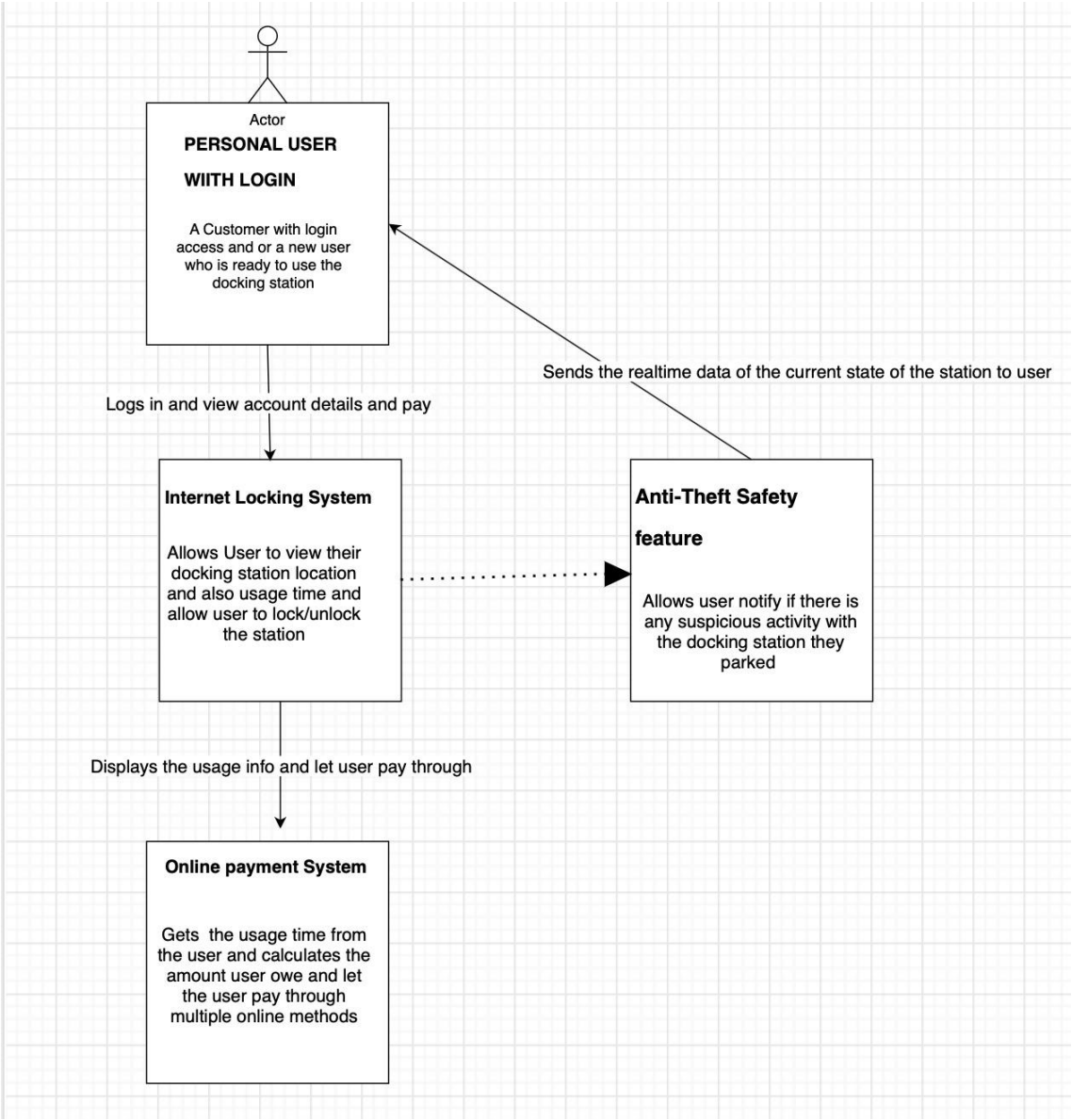
1. Answer the questions below, a minimum of 3 for each of the following:

1. **Start doing.**
2. **Stop doing.**
3. **Continue doing.**

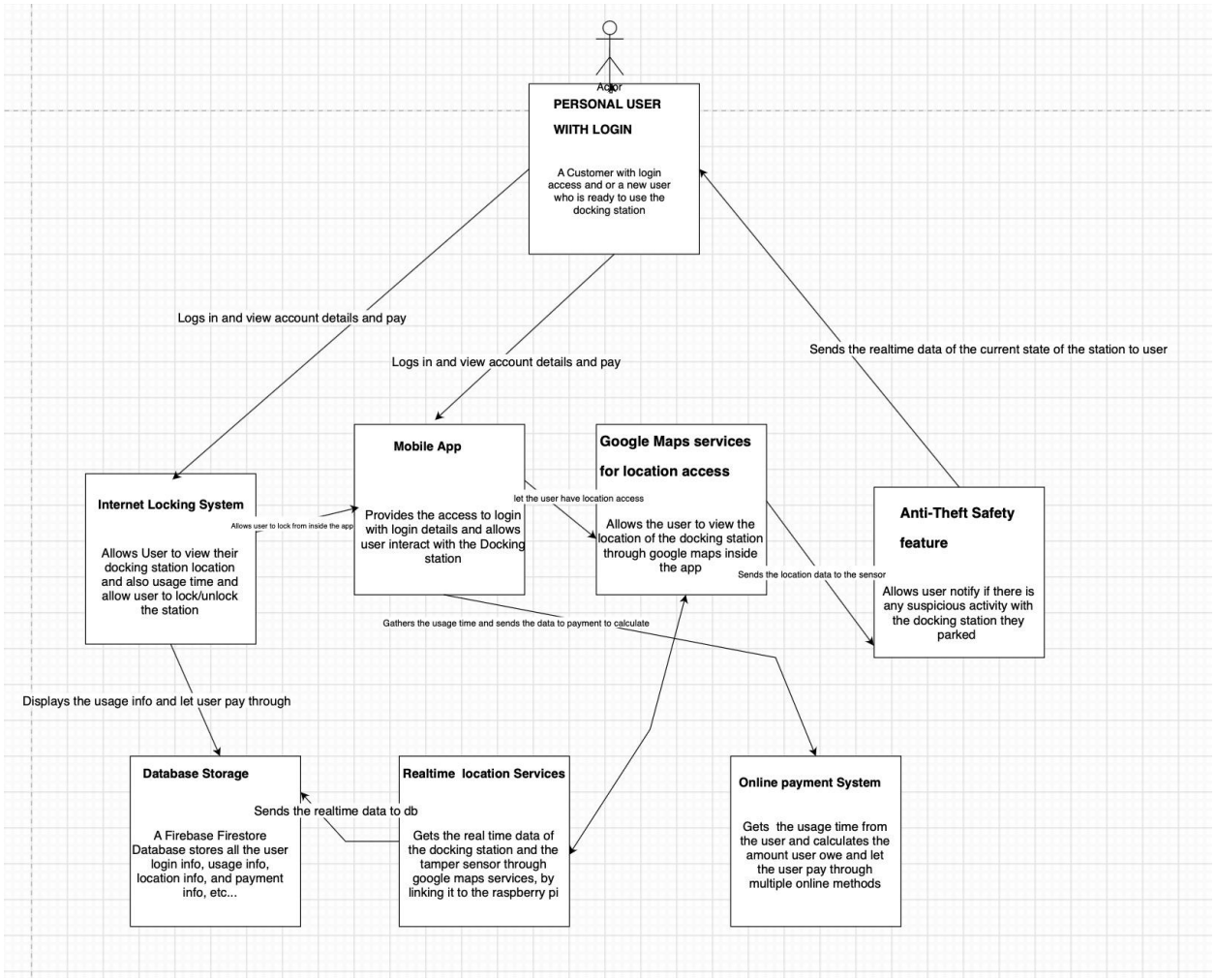
2. Take a screenshot.
3. Must use online tool for the Retrospective. No mark will be given if no tool used.

29. Using C4 Model, show “System Context Diagram”. Use a tool to draw your diagram, hand drawing will not be accepted, i.e. you can use https://app.diagrams.net/

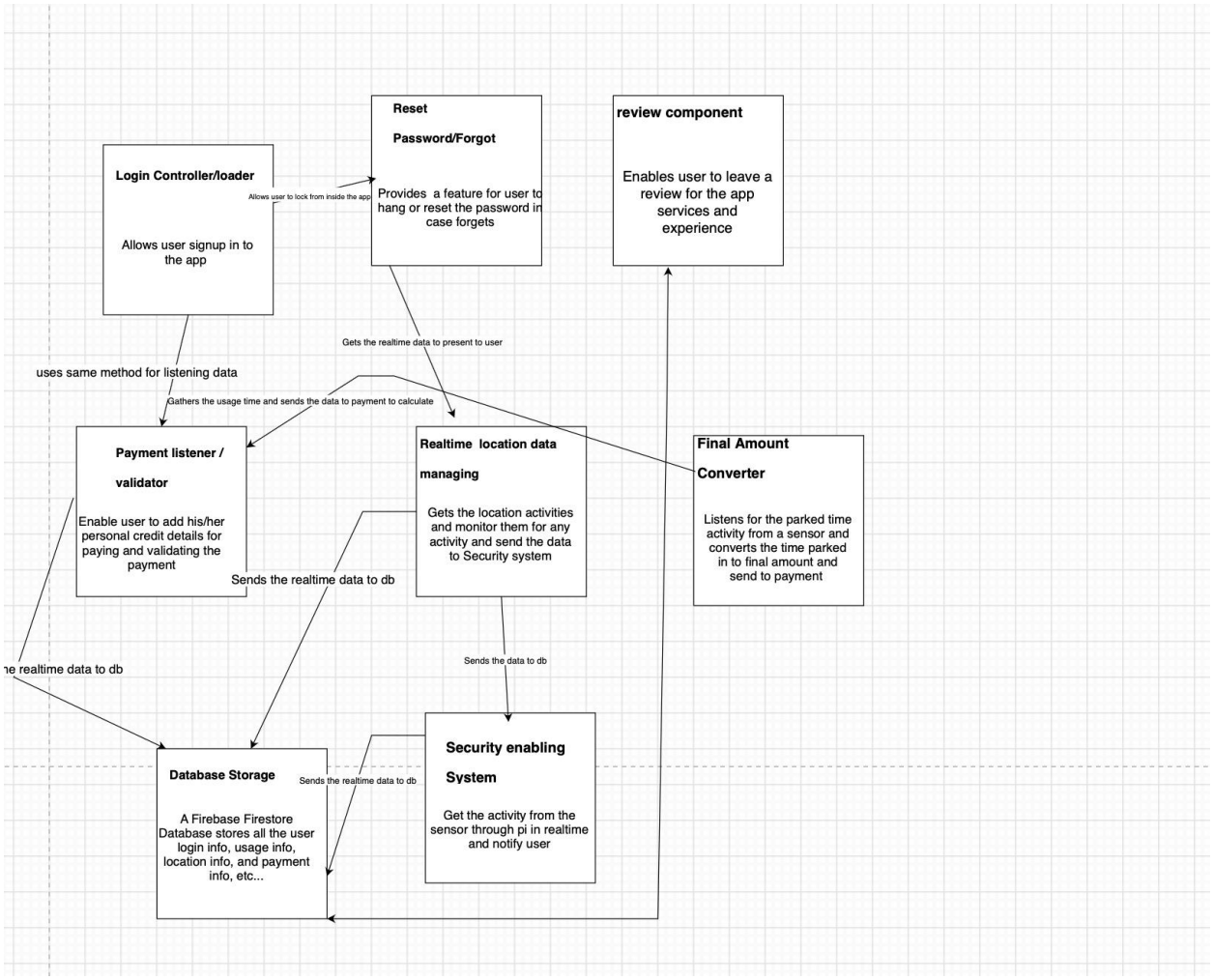
LEVEL1:
SYSTEM CONTEXT:



LEVEL 2:
SYSTEM CONTAINER:



LEVEL 3:
SYSTEM COMPONENT:



30. Document two different design principles used in the code.

The two different design principles we used in this project are:

DRY (Don’t Repeat Yourself):

According to this idea, any tiny bit of information (or piece of code) can only appear exactly once throughout the entire system. We can develop scalable, maintainable, and reusable code thanks to this.
For example we only created single different activities or java classes for every feature or functionality and used the pieces again and again across our different logical java functions.

KISS (Keep it simple, Stupid):

According to this idea, excessive complexity should be avoided and every small piece of software should be kept as simple as possible. This makes it easier for us to write maintainable code.
I personally prefer this principle as it is way more fun to code following this. For example we have not included any complicated animations or graphics in our application to avoid abrupt crashes or malfunctionality.

31. Document two different design patterns used in the code.

First design pattern used in the code:
The Singleton Design Pattern, which is creational.
One single instance of a class should be created, and there should only be one global access point to that object, according to the Singleton Design Design, a Creational pattern. Calendar is a well-known instance of a class like this in Java, but you cannot create an instance of it. In order to obtain the desired object, it also employs its own getInstance()method.

There are many ways to implement this pattern. One of the ways are: Eager Beaver

```
public class EagerSingleton {
	// create an instance of the class.
	private static EagerSingleton instance = new EagerSingleton();

	// private constructor, so it cannot be instantiated outside this class.
	private EagerSingleton() {  }

	// get the only instance of the object created.
	public static EagerSingleton getInstance() {
		return instance;
	}
}
```

Since the variable instance is created outside of any method, this kind of instantiation takes place during class loading. If the client programme doesn't use this class at all, this has a significant disadvantage. If this class isn't being used, the Lazy Instantiation serves as a backup plan.

Another way for implementing this pattern: Lazy Days

```
public class LazySingleton {
```

```
        // initialize the instance as null.
        private static LazySingleton instance = null;

        // private constructor, so it cannot be instantiated outside this class.
        private LazySingleton() {  }

        // check if the instance is null, and if so, create the object.
        public static LazySingleton getInstance() {
                if (instance == null) {
                        instance = new LazySingleton();
                }
                return instance;
        }
}
```

Little has changed from the implementation described above. The static variable is initially declared null and is only instantiated within the getInstance() method if and only if the instance variable is still null at the time of the check, which is one of the main differences between the two.

The Second Design Pattern is:
The Command Design Pattern for Behavior :

The interaction between classes and objects is the emphasis of a behavioral design pattern. The command pattern's primary goal is to encourage more loose connection between the parties involved (read: classes).

Two (or more) classes that interact with one another do so through coupling. When these classes interact, it is desirable if they do not strongly rely on one another. Loose coupling, that. So classes that are coupled but use each other the least would be a better way to define loose coupling. When requests were to be made without the sender or recipient being aware of them, this pattern became necessary.

Technically speaking, the process goes as follows: you create a concrete command that implements the Command interface, ask the receiver to perform a task, and send the command to the invoker. It is the invoker who has the timing necessary to issue this command. The only person who knows what to do when given a precise command or order is the chef. As a result, when the invoker's execute method is called, the command objects' execute method is called on the receiver as well, performing the appropriate tasks.

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

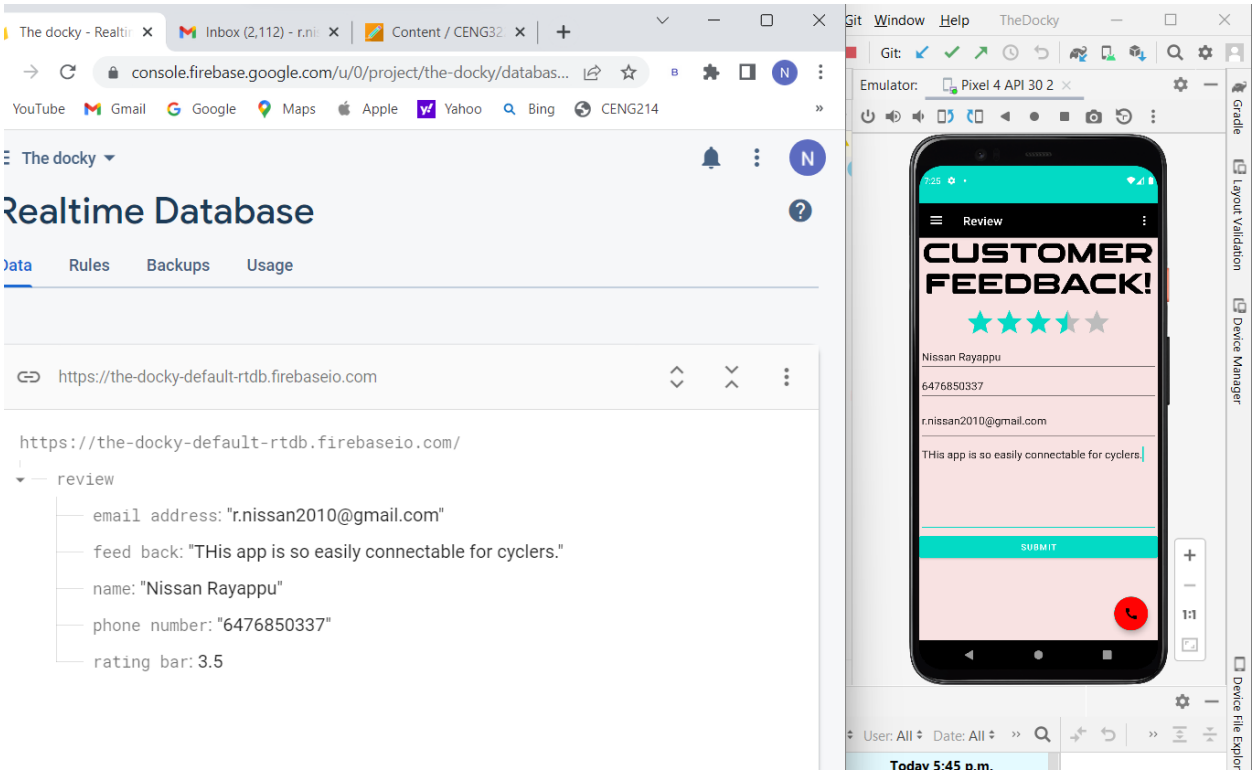
Many features have been added since deliverable2.

- Login Functionality was created with the proper ui/ux
- Registraton form was completed which was prepared in deliverable2
- Created the database in the firebase and integrated it with the application.
- Coding for reading the data, retrieving the data, and saving the data was done
- Added payement feature with attractive ui/ux
- Implemented coding functionality for all the java classes in the payment activity.
- Completed the payment form and also coding for user payment method is completed
- Completed code implementation for settings fragment and also implemented a new runtime permission
- Added some buttons and layouts for later implementation and created few other new fragments for future use.
- Code portion for Reset Password functionality was also completed
- Implementation of logout functionality along with code was done.
- Customer Review Page was also added with a creative ui/ux
- Many new Changes were made in design wise for the app

33. Document what runtime permission you have implemented

We have implemented a new runtime permission where user is allowed to make calls through the application.
When clicked on call, the device call feature would enable and will allow the user to make a call to whomever they wish.

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


The docky - Realti
Inbox (2,112) - r.ni
Content / CENG32
console.firebase.google.com/u/0/project/the-docky/databas...
YouTube
Gmail
Google
Maps
Apple
Yahoo
Bing
CENG214
The docky
Realtime Database
Data
Rules
Backups
Usage
https://the-docky-default-rtdb.firebaseio.com
https://the-docky-default-rtdb.firebaseio.com/
review
email address: "r.nissan2010@gmail.com"
feed back: "THis app is so easily connectable for cyclers."
name: "Nissan Rayappu"
phone number: "6476850337"
rating bar: 3.5
Git
Window
Help
TheDocky
Emulator:
Pixel 4 API 30 2
Review
CUSTOMER FEEDBACK!
Nissan Rayappu
6476850337
r.nissan2010@gmail.com
THis app is so easily connectable for cyclers.
SUBMIT
Gradle
Layout Validation
Device Manager
Device File Explor
User: All
Date: All
Today 5:45 p.m.