Deliverable 3

CENG-322 TEAM PROJECT

Team Name: Hermes Logistics

Project Name: PetasosExpress

Team Number: 1

Team Members:

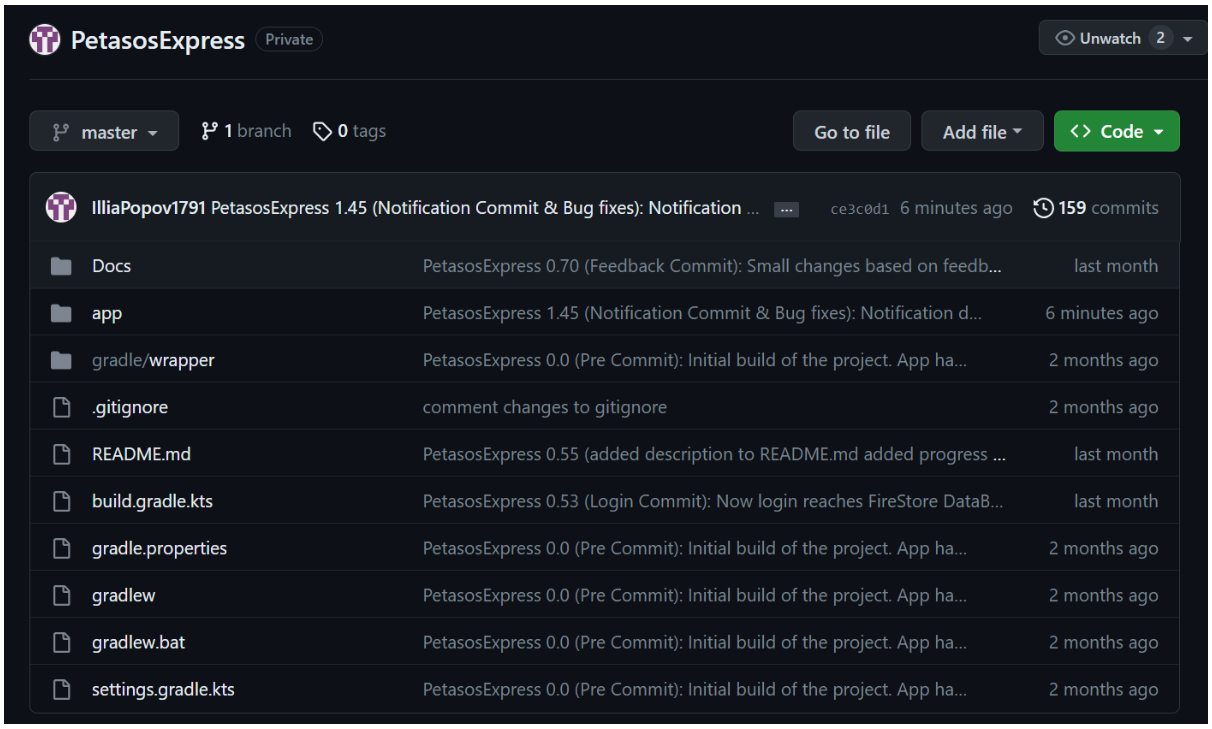
- Illia Myrza Popov (n01421791) - Distance and GPS sensors
- Ahmad Aljawish (n01375348) - Balance Sensor
- William Margalik (n01479878) - Motor Sensor
- Dylan Ashton (n01442206) - Proximity Sensor

Members Info and Participation:

Name	ID	Signature	Effort
Illia Popov	n01421791	IlliaPopov	100%
Ahmad Aljawish	n01375348	AhmadALjawish	100%
Dylan Ashton	n01442206	DylanAshton	100%
William Margalik	n01479878	WilliamMargalik	100%

GitHub Repository Links:

GitHub Repository: https://github.com/IlliaPopov1791/PetasosExpress

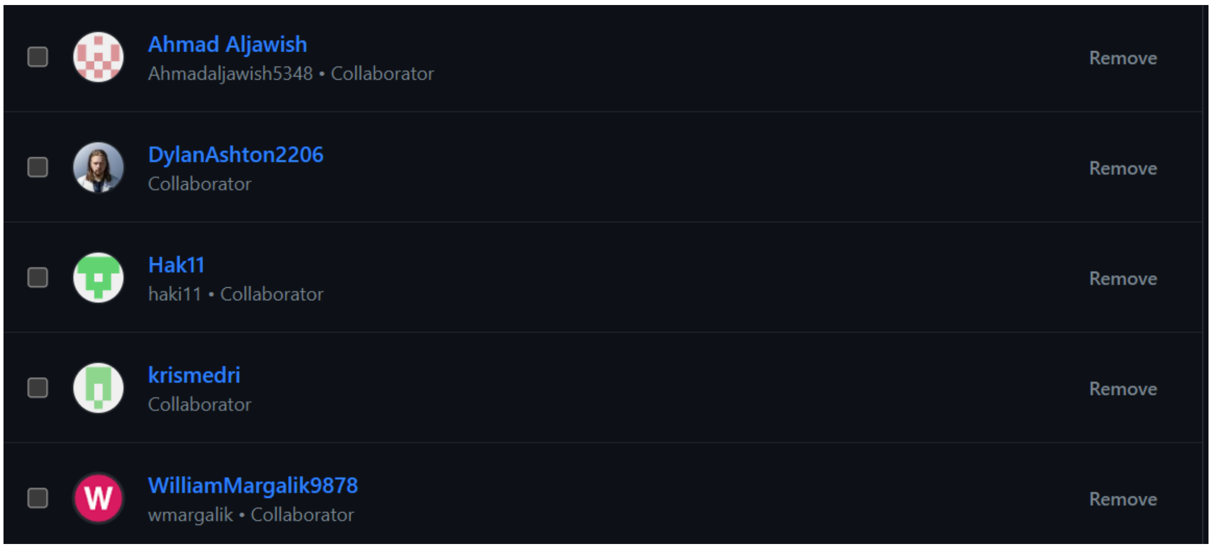


IlliaPopov1791 PetasosExpress 1.45 (Notification Commit & Bug fixes): Notification ... ce3c0d1 6 minutes ago 159 commits

Docs	PetasosExpress 0.70 (Feedback Commit): Small changes based on feedb...	last month
app	PetasosExpress 1.45 (Notification Commit & Bug fixes): Notification d...	6 minutes ago
gradle/wrapper	PetasosExpress 0.0 (Pre Commit): Initial build of the project. App ha...	2 months ago
.gitignore	comment changes to gitignore	2 months ago
README.md	PetasosExpress 0.55 (added description to README.md added progress ...	last month
build.gradle.kts	PetasosExpress 0.53 (Login Commit): Now login reaches FireStore DataB...	last month
gradle.properties	PetasosExpress 0.0 (Pre Commit): Initial build of the project. App ha...	2 months ago
gradlew	PetasosExpress 0.0 (Pre Commit): Initial build of the project. App ha...	2 months ago
gradlew.bat	PetasosExpress 0.0 (Pre Commit): Initial build of the project. App ha...	2 months ago
settings.gradle.kts	PetasosExpress 0.0 (Pre Commit): Initial build of the project. App ha...	2 months ago

GitHub Invitation Confirmation:

Repository Invites of Software Project and Hardware Production professors, and all team members (Taken by IlliaPopov1791):



Ahmad Aljawish
Ahmadaljawish5348 • Collaborator
Remove

DylanAshton2206
Collaborator
Remove

Hak11
haki11 • Collaborator
Remove

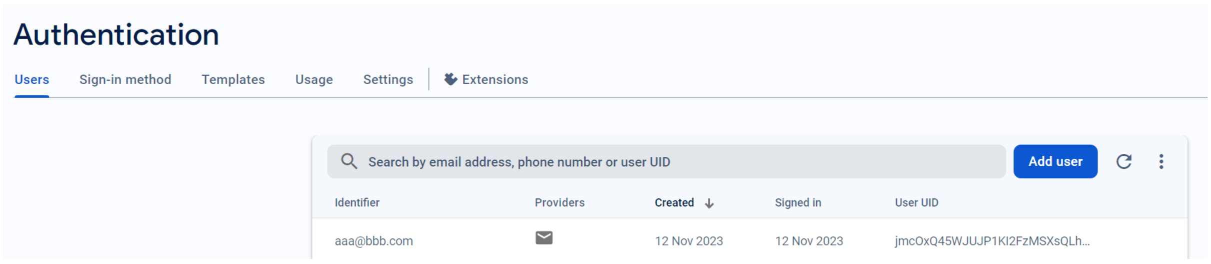
krismedri
Collaborator
Remove

WilliamMargalik9878
wmargalik • Collaborator
Remove

Created account in the DB with requested credentials:

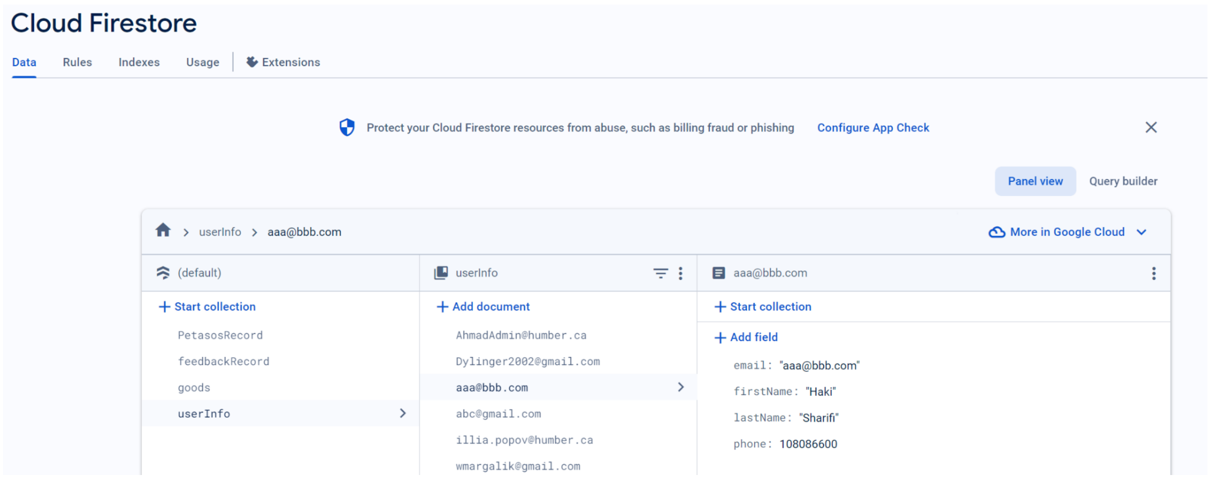
Admin Credential:

Email: aaa@bbb.com Password: Admin101!



Authentication

Identifier	Providers	Created	Signed in	User UID
aaa@bbb.com	✉	12 Nov 2023	12 Nov 2023	jmcOxQ45WJUJP1KI2FzMSXsQLh...



Cloud Firestore

(default)
+ Start collection
PetasosRecord
feedbackRecord
goods
userInfo

userInfo
+ Add document
AhmadAdmin@humber.ca
Dylinger2002@gmail.com
aaa@bbb.com
abc@gmail.com
illia.popov@humber.ca
wmargalik@gmail.com

aaa@bbb.com
+ Start collection
+ Add field
email: "aaa@bbb.com"
firstName: "Haki"
lastName: "Sharifi"
phone: 108086600

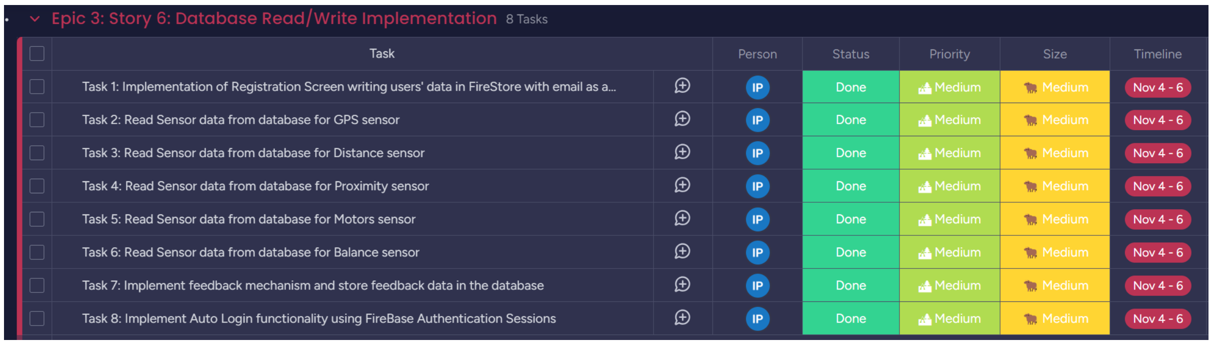
Sprint Goals

List of Sprint Goals for Deliverable 3

The sprint goals for Hermes Logistics team for deliverable 3 are as follows: -.

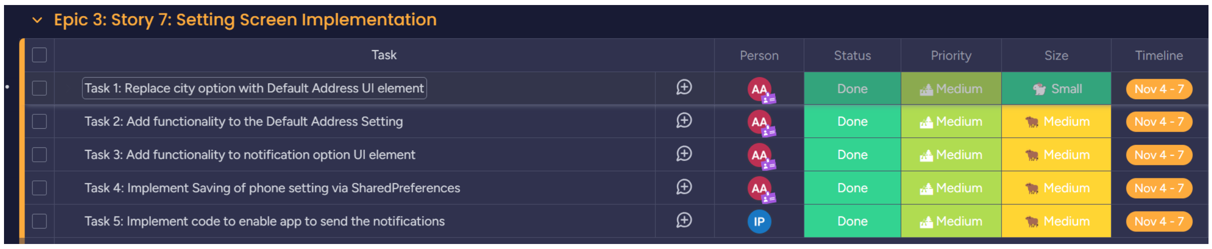
- Implementation of the functionality of Settings Screen.
- Implement storing the settings preferred by users using shared preference.
- Implement Reading data from and Writing data to the Database(Sensors, Registration, Feedback, Account Management screen, etc).
- Implement runtime permissions.
- Implementing a functional Feedback page using Firestore Database.
- Implementation of the functional Search Engine.
- Merge sensor screens keeping their functionality.

Sprint Dashboard:



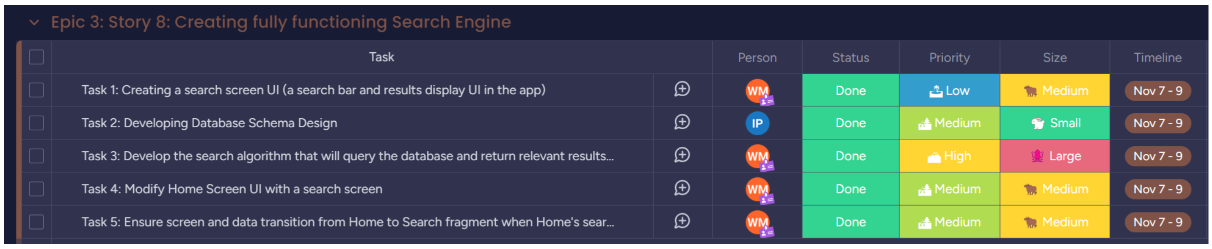
Epic 3: Story 6: Database Read/Write Implementation 8 Tasks

Task	Person	Status	Priority	Size	Timeline
Task 1: Implementation of Registration Screen writing users' data in FireStore with email as a...	IP	Done	Medium	Medium	Nov 4 - 6
Task 2: Read Sensor data from database for GPS sensor	IP	Done	Medium	Medium	Nov 4 - 6
Task 3: Read Sensor data from database for Distance sensor	IP	Done	Medium	Medium	Nov 4 - 6
Task 4: Read Sensor data from database for Proximity sensor	IP	Done	Medium	Medium	Nov 4 - 6
Task 5: Read Sensor data from database for Motors sensor	IP	Done	Medium	Medium	Nov 4 - 6
Task 6: Read Sensor data from database for Balance sensor	IP	Done	Medium	Medium	Nov 4 - 6
Task 7: Implement feedback mechanism and store feedback data in the database	IP	Done	Medium	Medium	Nov 4 - 6
Task 8: Implement Auto Login functionality using FireBase Authentication Sessions	IP	Done	Medium	Medium	Nov 4 - 6



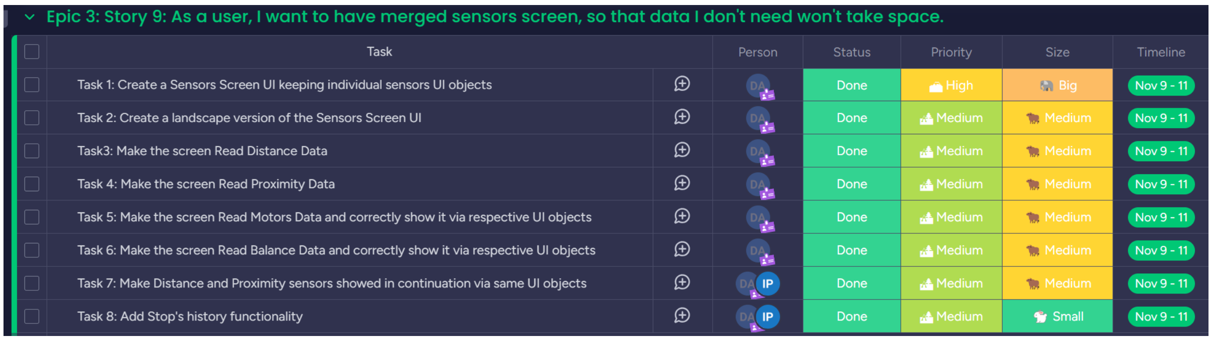
Epic 3: Story 7: Setting Screen Implementation

Task	Person	Status	Priority	Size	Timeline
Task 1: Replace city option with Default Address UI element	AA	Done	Medium	Small	Nov 4 - 7
Task 2: Add functionality to the Default Address Setting	AA	Done	Medium	Medium	Nov 4 - 7
Task 3: Add functionality to notification option UI element	AA	Done	Medium	Medium	Nov 4 - 7
Task 4: Implement Saving of phone setting via SharedPreferences	AA	Done	Medium	Medium	Nov 4 - 7
Task 5: Implement code to enable app to send the notifications	IP	Done	Medium	Medium	Nov 4 - 7



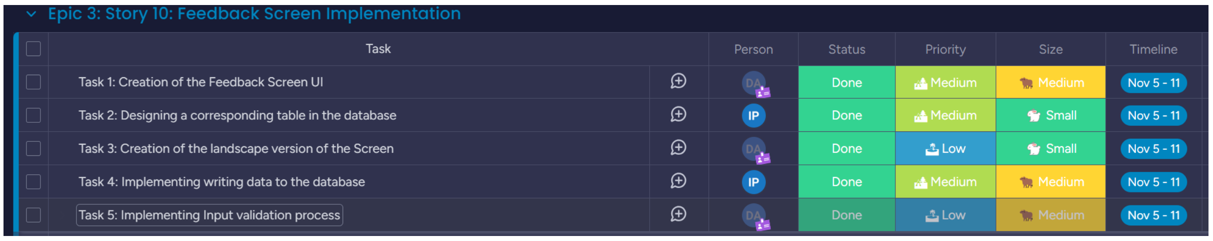
Epic 3: Story 8: Creating fully functioning Search Engine

Task	Person	Status	Priority	Size	Timeline
Task 1: Creating a search screen UI (a search bar and results display UI in the app)	WM	Done	Low	Medium	Nov 7 - 9
Task 2: Developing Database Schema Design	IP	Done	Medium	Small	Nov 7 - 9
Task 3: Develop the search algorithm that will query the database and return relevant results...	WM	Done	High	Large	Nov 7 - 9
Task 4: Modify Home Screen UI with a search screen	WM	Done	Medium	Medium	Nov 7 - 9
Task 5: Ensure screen and data transition from Home to Search fragment when Home's sear...	WM	Done	Medium	Medium	Nov 7 - 9



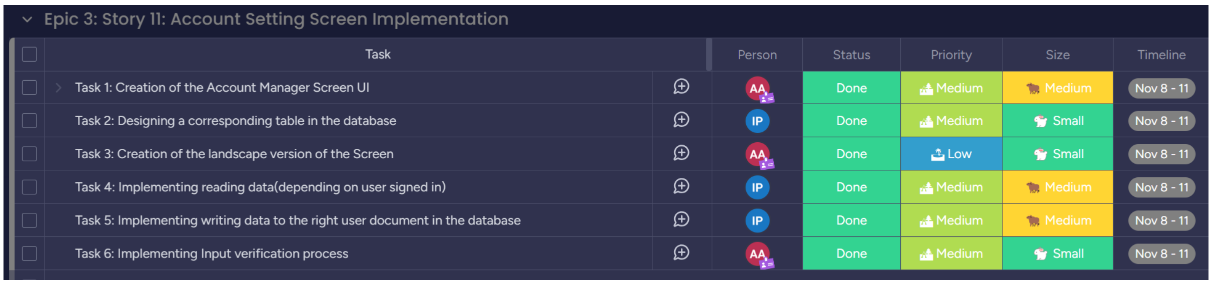
Epic 3: Story 9: As a user, I want to have merged sensors screen, so that data I don't need won't take space.

Task	Person	Status	Priority	Size	Timeline
Task 1: Create a Sensors Screen UI keeping individual sensors UI objects	DA	Done	High	Big	Nov 9 - 11
Task 2: Create a landscape version of the Sensors Screen UI	DA	Done	Medium	Medium	Nov 9 - 11
Task3: Make the screen Read Distance Data	DA	Done	Medium	Medium	Nov 9 - 11
Task 4: Make the screen Read Proximity Data	DA	Done	Medium	Medium	Nov 9 - 11
Task 5: Make the screen Read Motors Data and correctly show it via respective UI objects	DA	Done	Medium	Medium	Nov 9 - 11
Task 6: Make the screen Read Balance Data and correctly show it via respective UI objects	DA	Done	Medium	Medium	Nov 9 - 11
Task 7: Make Distance and Proximity sensors showed in continuation via same UI objects	DA IP	Done	Medium	Medium	Nov 9 - 11
Task 8: Add Stop's history functionality	DA IP	Done	Medium	Small	Nov 9 - 11



Epic 3: Story 10: Feedback Screen Implementation

Task	Person	Status	Priority	Size	Timeline
Task 1: Creation of the Feedback Screen UI	DA	Done	Medium	Medium	Nov 5 - 11
Task 2: Designing a corresponding table in the database	IP	Done	Medium	Small	Nov 5 - 11
Task 3: Creation of the landscape version of the Screen	DA	Done	Low	Small	Nov 5 - 11
Task 4: Implementing writing data to the database	IP	Done	Medium	Medium	Nov 5 - 11
Task 5: Implementing Input validation process	DA	Done	Low	Medium	Nov 5 - 11

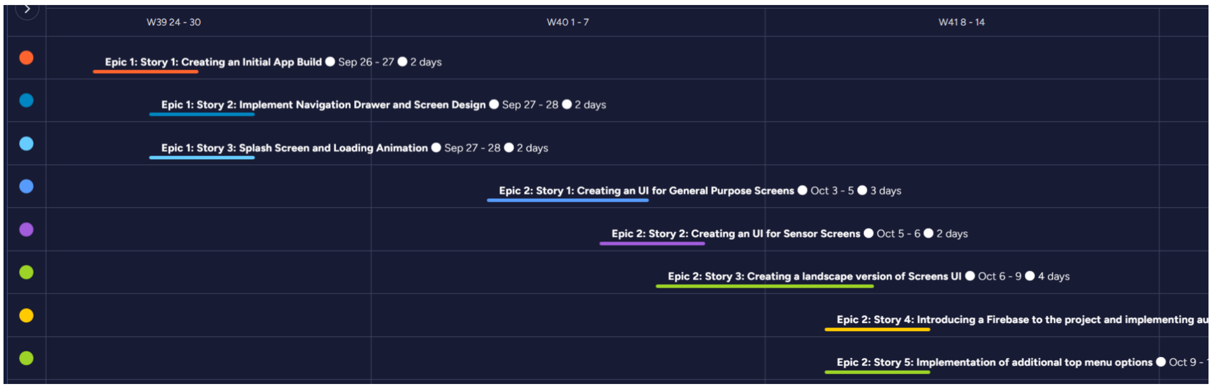


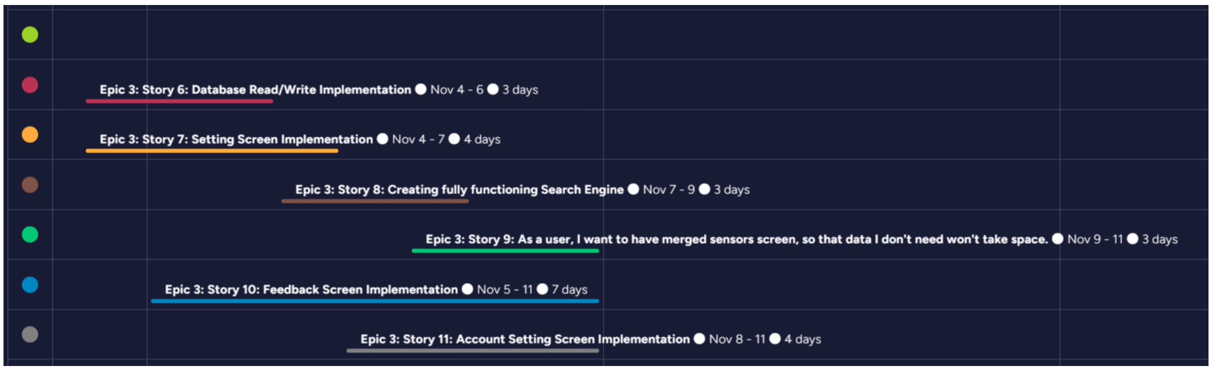
Epic 3: Story 11: Account Setting Screen Implementation

Task	Person	Status	Priority	Size	Timeline
Task 1: Creation of the Account Manager Screen UI	AA	Done	Medium	Medium	Nov 8 - 11
Task 2: Designing a corresponding table in the database	IP	Done	Medium	Small	Nov 8 - 11
Task 3: Creation of the landscape version of the Screen	AA	Done	Low	Small	Nov 8 - 11
Task 4: Implementing reading data(depending on user signed in)	IP	Done	Medium	Medium	Nov 8 - 11
Task 5: Implementing writing data to the right user document in the database	IP	Done	Medium	Medium	Nov 8 - 11
Task 6: Implementing Input verification process	AA	Done	Medium	Small	Nov 8 - 11

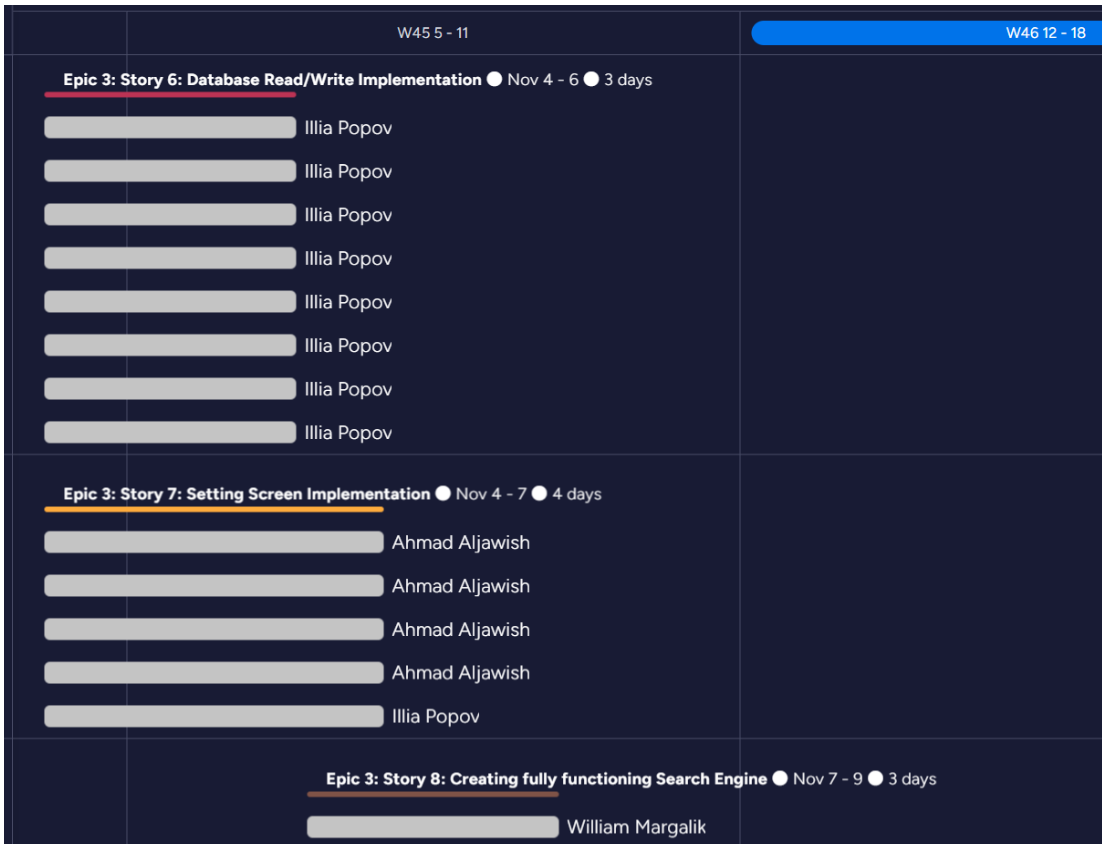
Gantt Chart:

General:





Detailed:



W45 5 - 11

W46 12 - 18

Epic 3: Story 6: Database Read/Write Implementation ● Nov 4 - 6 ● 3 days

Illia Popov

Illia Popov

Illia Popov

Illia Popov

Illia Popov

Illia Popov

Illia Popov

Illia Popov

Epic 3: Story 7: Setting Screen Implementation ● Nov 4 - 7 ● 4 days

Ahmad Aljawish

Ahmad Aljawish

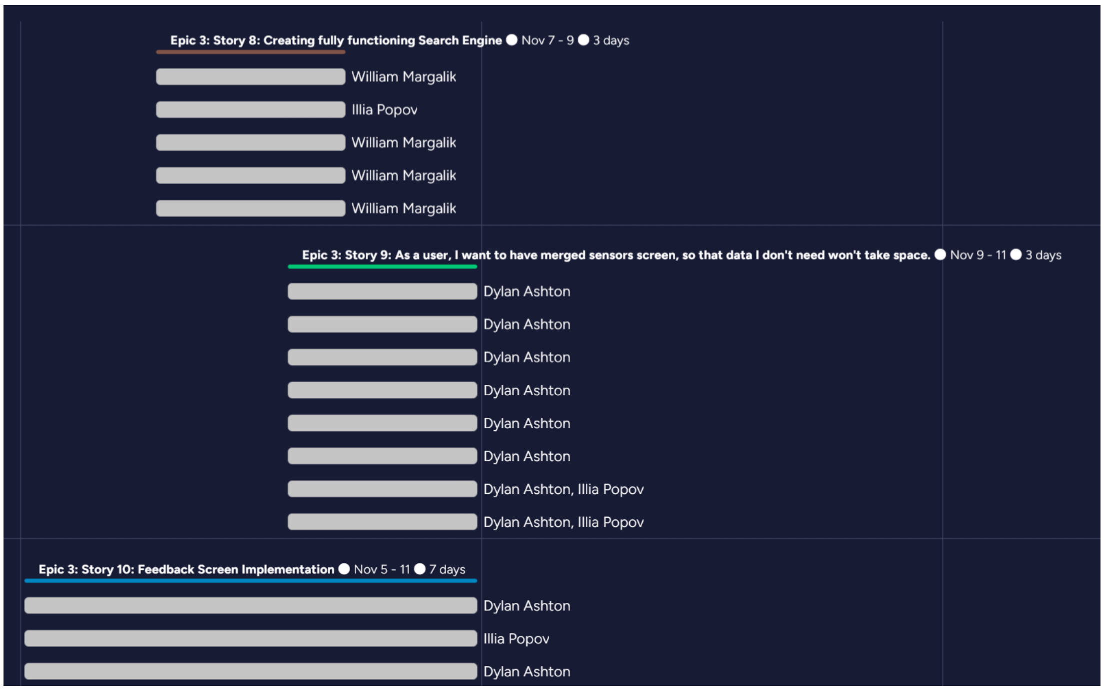
Ahmad Aljawish

Ahmad Aljawish

Illia Popov

Epic 3: Story 8: Creating fully functioning Search Engine ● Nov 7 - 9 ● 3 days

William Margalik



Epic 3: Story 8: Creating fully functioning Search Engine ● Nov 7 - 9 ● 3 days

William Margalik

Illia Popov

William Margalik

William Margalik

William Margalik

Epic 3: Story 9: As a user, I want to have merged sensors screen, so that data I don't need won't take space. ● Nov 9 - 11 ● 3 days

Dylan Ashton

Dylan Ashton

Dylan Ashton

Dylan Ashton

Dylan Ashton

Dylan Ashton

Dylan Ashton, Illia Popov

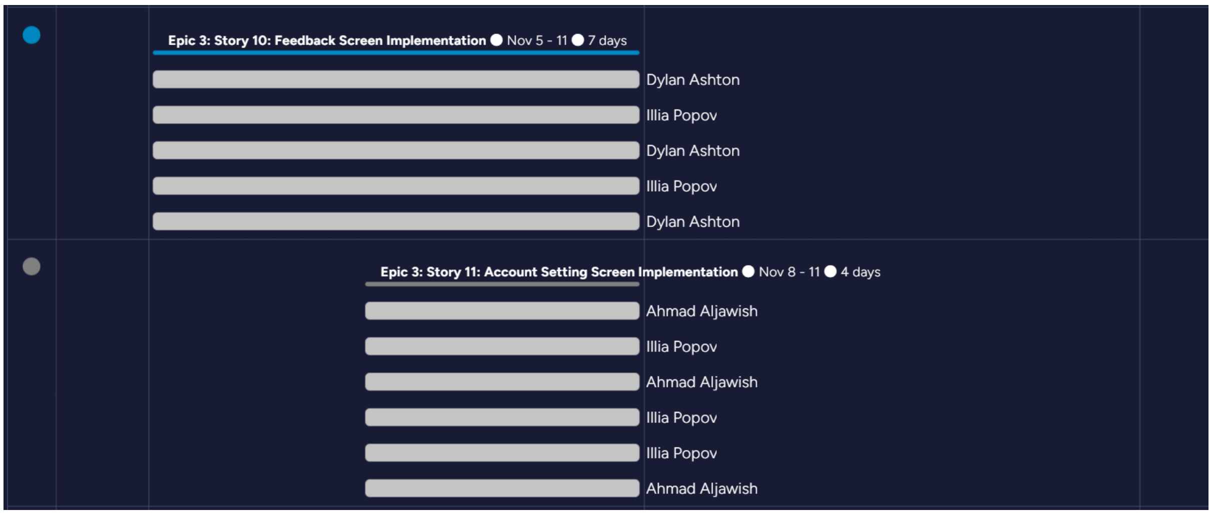
Dylan Ashton, Illia Popov

Epic 3: Story 10: Feedback Screen Implementation ● Nov 5 - 11 ● 7 days

Dylan Ashton

Illia Popov

Dylan Ashton


Epic 3: Story 10: Feedback Screen Implementation ● Nov 5 - 11 ● 7 days

Dylan Ashton
Illia Popov
Dylan Ashton
Illia Popov
Dylan Ashton

Epic 3: Story 11: Account Setting Screen Implementation ● Nov 8 - 11 ● 4 days

Ahmad Aljawish
Illia Popov
Ahmad Aljawish
Illia Popov
Illia Popov
Ahmad Aljawish

Daily Standup:

Nov.03	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Made changes to the setting screen based on the feedback from the product owner	Read the feedback from deliverable 2	Landscape designs and UI uniformity changes	Brainstormed ideas for how to fully implement a functional search engine.
	What will you work on today	Start planning of the sprint	Start planning of the sprint	Start planning of the sprint	Start planning of the sprint
	Are there any roadblocks stopping you?	No blocker at the moment.	No blocker at the moment.	No blocker at the moment.	No blocker at the moment.

Nov.04	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Finished planning of the sprint	Finished planning of the sprint	Finished planning of the sprint	Finished planning of the sprint
	What will you work on today	Working on Auto Login function and redesigning Registration	Made the Settings UI more user-friendly	Feedback Page UI, portrait and landscape	Implemented UI for search screen fragments
	Are there	No blocker at	No blocker at	No blocker at	No blocker at

	any roadblocks stopping you?	the moment.	the moment	the moment.	the moment.

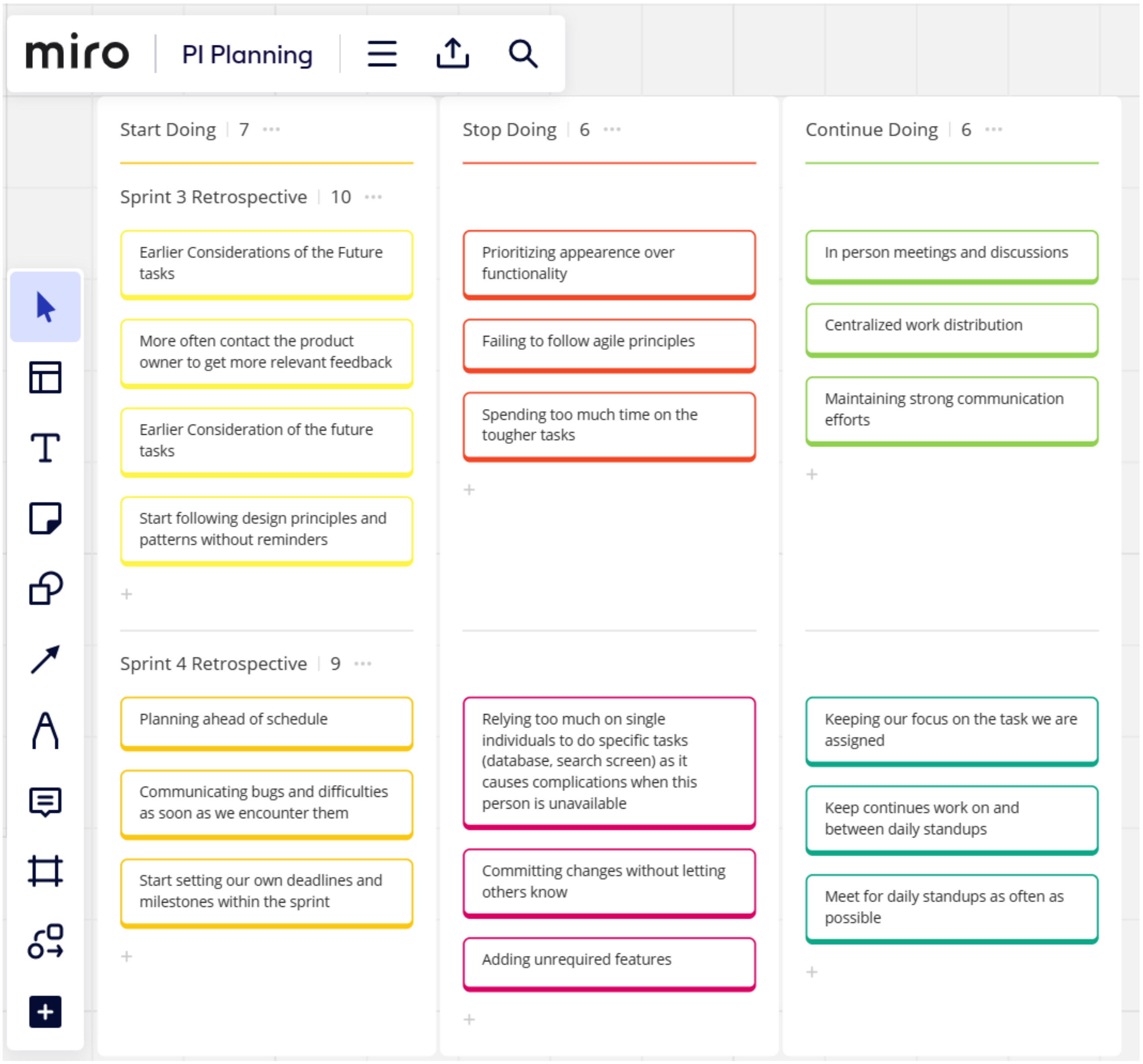
Nov.07 -08	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Finished implementatio n of the Auto Login and Registration	Finished implementatio n of all the setting screen	Worked on functionality and UI of Proximity Screen	Brainstormed how the UI fragment would look like by the end of the week.
	What will you work on today	Working on the sensors and feedback pages communicatio n (reading data from and writing data to) with the Database	Working on saving user selection from the settings screen when the app is restarted using SharedPrefere nces	Logic of switching between status image based on data from Firebase, and making progressbar update based on this	Created a search bar for the search fragment_sear ch_screen.xml
	Are there any roadblocks stopping you?	No blocker at the moment.	No blocker at the moment	No blocker at the moment.	Just some code restricting the bar from functioning smoothly without bugs from home screen to search fragment.

Nov.10	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Finished implementatio n of the app and Firestore database communicatio n	Finished all functionality with settings screen and tested to make sure Shared preference is working properly.	Merged all sensor java logic(except GPS) into sensor screen to improve user experience	Implementing a function where once the user clicks enter it transfers to the search screen fragment.
	What will you work on today	Working on fixing bugs and making	Worked on UI changes for Sensor Screen	Creating UI for the new SensorScreen	Implemented a code when once the

		search engine use data from the database instead of string(used previously for testing)	and FeedbackScree n	Including most of proximity,dist, balance and motor screens	database is set up for it, it will display search results onto the screen from the search bar requests.
	Are there any roadblocks stopping you?	Waiting on William to finish semi-functiona l Searching screen demo	Waiting for Dylan to create the new UI to update Balance Sensor	Struggling with merge without losing functionality	Had some code issues with the search not displaying on the list, but debugged and finished it.

Nov.11	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Finished incorporation our database with our search engine	Finished doing all the UI updates for Balance sensor and Settings Screen	Worked on making sure merged sensorScreen is working	Made a functional custom search bar to be able to search items retrieved from the database.
	What will you work on today	Working on Manage Account page and non-database related functionality of the merged sensor screen. Also, redesigned database and paths app use to read data for sensors. In addition, added notification functionality.	Creating an Account Settings Screen with a portrait and landscape UI. Making it functional to retrieve data from the database to allow the user to change their info using the app.	Creating landscape xml For sensor screen, edited menu and main java to replace previous screens with combined version. Made changes to Sensor Screen java code to better align with coding principles	Fixed the landscape orientation bug that was not happening for the Home Screen.
	Are there any roadblocks stopping you?	No blocker at the moment	No blocker at the moment	No blocker at the moment	No blocker at the moment

Screenshot Showing retrospective of Sprint 3 and demo of Sprint 4:



miro | PI Planning

Start Doing | 7

Sprint 3 Retrospective | 10

- Earlier Considerations of the Future tasks
- More often contact the product owner to get more relevant feedback
- Earlier Consideration of the future tasks
- Start following design principles and patterns without reminders

Sprint 4 Retrospective | 9

- Planning ahead of schedule
- Communicating bugs and difficulties as soon as we encounter them
- Start setting our own deadlines and milestones within the sprint

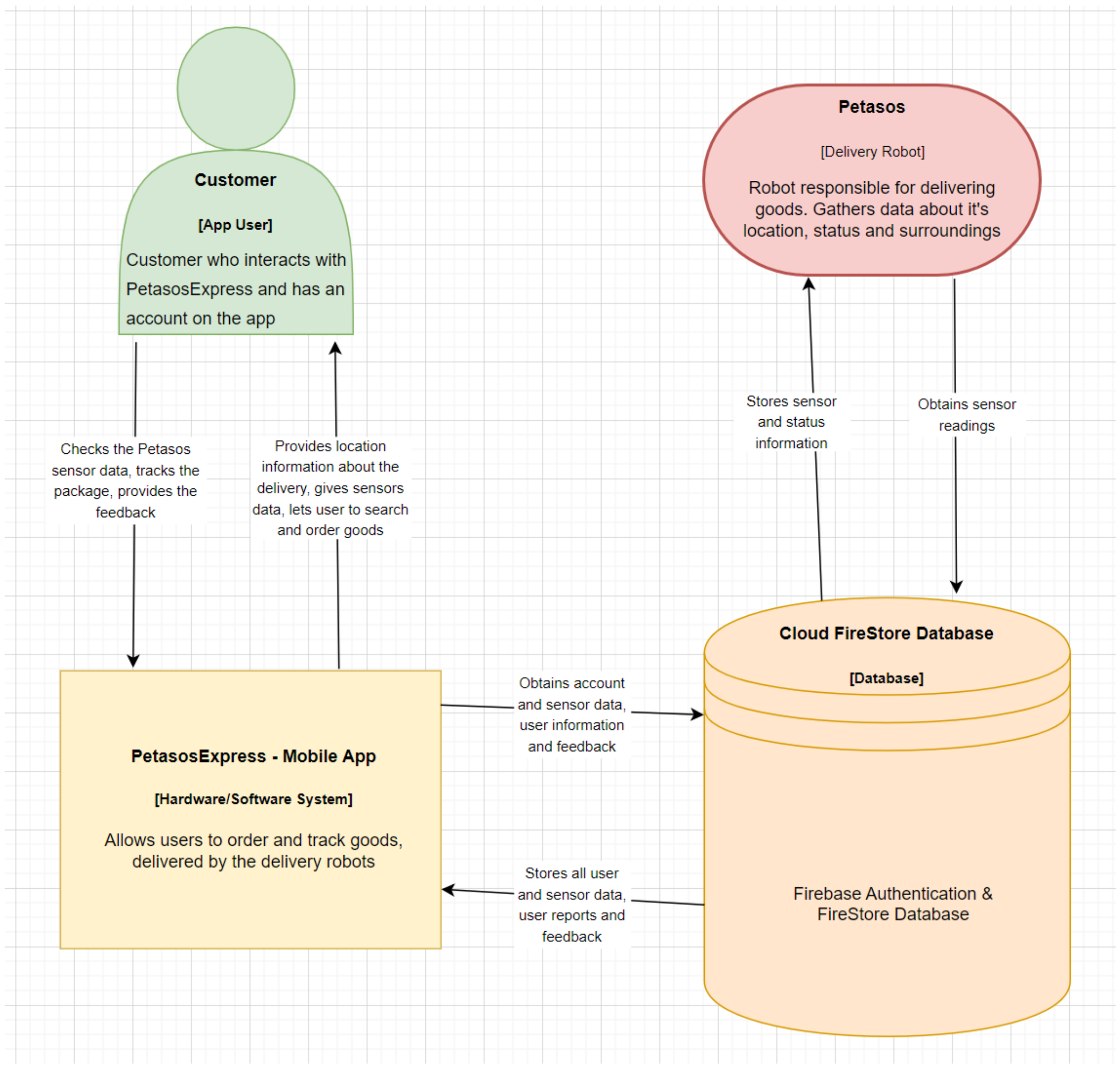
Stop Doing | 6

- Prioritizing appearence over functionality
- Failing to follow agile principles
- Spending too much time on the tougher tasks
- Relying too much on single individuals to do specific tasks (database, search screen) as it causes complications when this person is unavailable
- Committing changes without letting others know
- Adding unrequired features

Continue Doing | 6

- In person meetings and discussions
- Centralized work distribution
- Maintaining strong communication efforts
- Keeping our focus on the task we are assigned
- Keep continues work on and between daily standups
- Meet for daily standups as often as possible

C4 Model, showing "System Context Diagram":


Customer
[App User]
Customer who interacts with PetasosExpress and has an account on the app
Petasos
[Delivery Robot]
Robot responsible for delivering goods. Gathers data about it's location, status and surroundings
Checks the Petasos sensor data, tracks the package, provides the feedback
Provides location information about the delivery, gives sensors data, lets user to search and order goods
Stores sensor and status information
Obtains sensor readings
PetasosExpress - Mobile App
[Hardware/Software System]
Allows users to order and track goods, delivered by the delivery robots
Obtains account and sensor data, user information and feedback
Stores all user and sensor data, user reports and feedback
Cloud FireStore Database
[Database]
Firebase Authentication & FireStore Database

Design Patterns & Principles:

Design Patterns:

Strategy Pattern: Retrieved code from MainActivity.java within the method configureNavigationView().

```
navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
   @SuppressLint({"NonConstantResourceId", "ResourceType"})
   @Override
   public boolean onNavigationItemSelected(@NonNull MenuItem item) {
       int itemId = item.getItemId();
       FragmentManager fragmentManager = getSupportFragmentManager();
       Fragment fragmentToLoad = null;
       String fragmentTag = "";

       if (itemId == R.id.home_screen) {
           fragmentToLoad = new Home();
           fragmentTag = getString(R.string.home_tag);
       } else if (itemId == R.id.gps_sensor) {
           fragmentToLoad = new SensorGPS();
           fragmentTag = getString(R.string.gps_tag);
       }  else if (itemId == R.id.sensor_screen) {
           fragmentToLoad = new SensorScreen();
           fragmentTag = getString(R.string.proximity_tag);
       }  else if (itemId == R.id.AppSettings) {
           fragmentToLoad = new AppSettings();
           fragmentTag = getString(R.string.settings_tag);
       }
       else if (itemId == R.id.FeedbackScreen) {
           fragmentToLoad = new FeedbackScreen();
           fragmentTag = getString(R.string.feedback);
       }
       else if (itemId == R.id.AccountSettings) {
           fragmentToLoad = new ManageAccount();
           fragmentTag = getString(R.string.ManageAccount);
       }

       if (fragmentToLoad != null) {
           fragmentManager.beginTransaction().replace(R.id.main_frame_layout,
fragmentToLoad).commit();
           saveCurrentFragment(fragmentTag); // Save the current fragment's
tag
       }
```

```
        drawerLayout.closeDrawer(GravityCompat.START);
        return true;
    }

});
```

Explanation:

The Strategy pattern is used in the code to encapsulate the algorithm behind what happens when a navigation item is selected. Each if condition within the onNavigationItemSelected method checks the item ID and dynamically sets the Fragment that should be displayed. This is an implementation of the Strategy pattern as the actual fragment displayed (fragmentToLoad) can vary at runtime depending on the user's choice.

Overall, the Strategy pattern enhances the code's flexibility and adaptability to change.

Observer Pattern: Retrieved code from MainActivity.java within the method configureNavigationView() and setupBalanceSensor() in SensorScreen.java.

```
private void setupBalanceSensor() {
    DocumentReference docRef = db.collection("PetasosRecord")
            .document("Toronto")
            .collection("Petasos001")
            .document("Balance");
    docRef.addSnapshotListener(new EventListener<DocumentSnapshot>() {
        @Override
        public void onEvent(@Nullable DocumentSnapshot snapshot,
@Nullable FirebaseFirestoreException e) {
            if (e != null) {
                xAxisValue.setText(R.string.server_error);
                yAxisValue.setText(R.string.server_error);
                zAxisValue.setText(R.string.server_error);
                return;
            }

            if ((snapshot != null && snapshot.exists() && isAdded())) {
                Number xAxis = snapshot.getLong("X-axis");
                Number yAxis = snapshot.getLong("Y-axis");
                Number zAxis = snapshot.getLong("Z-axis");

                updateAxis(xAxisProgressBar, xAxisValue, xAxis);
                updateAxis(yAxisProgressBar, yAxisValue, yAxis);
                updateAxis(zAxisProgressBar, zAxisValue, zAxis);
            } else {
                xAxisValue.setText(R.string.no_data);
                yAxisValue.setText(R.string.no_data);
                zAxisValue.setText(R.string.no_data);
            }
        }
    });
```

Explanation:
The Observer pattern is used in this code (instance of this above) to establish a subscription mechanism allowing multiple objects to listen and react to events or changes happening in another object. Here, Firestore's DocumentReference acts as the Subject, and the EventListener acts as the Observer. When the balance sensor's data changes, the DocumentReference will notify all attached EventListeners by invoking onEvent(). These listeners react to the event by updating the UI components such as TextView and ProgressBar with the new data.

Design Principles:

Single Purpose (S from SOLID): Retrieved code from SensorScreen.java within the sendNotification() method.

```
private void sendNotification() {
   String channelId = "delivery_notifications";
   String channelName = "Delivery Notifications";
   String notificationTitle = "Delivery Update";
   String notificationText = "Your delivery may be late due to obstacles on
Petasos' way";
   SharedPreferences settings =
getActivity().getSharedPreferences(AppSettings.PREFS_NAME, 0);
   boolean areNotificationsEnabled =
settings.getBoolean(AppSettings.NOTIFICATIONS_KEY, true);

   if (!areNotificationsEnabled) {
       //Exit if notifications are disabled
       return;
   }
   // Proceed and create the NotificationChannel (required for API 26+)
   if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
       NotificationChannel channel = new NotificationChannel(channelId,
channelName, NotificationManager.IMPORTANCE_DEFAULT);
       NotificationManager notificationManager =
getContext().getSystemService(NotificationManager.class);
       if (notificationManager != null) {
           notificationManager.createNotificationChannel(channel);
```

```
        }
    }

    // Build the notification
    NotificationCompat.Builder builder = new
NotificationCompat.Builder(getContext(), channelId)
            .setSmallIcon(R.mipmap.ic_launcher)
            .setContentTitle(notificationTitle)
            .setContentText(notificationText)
            .setPriority(NotificationCompat.PRIORITY_DEFAULT);

    // Show the notification
    NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(getContext());
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        if (ContextCompat.checkSelfPermission(requireContext(),
Manifest.permission.POST_NOTIFICATIONS) == PackageManager.PERMISSION_GRANTED)
{
            notificationManager.notify(1, builder.build());
        } else {
            requestPermissions(new
String[]{Manifest.permission.POST_NOTIFICATIONS},
NOTIFICATION_PERMISSION_REQUEST_CODE);
        }
    } else {
        notificationManager.notify(1, builder.build());
    }
}
```

Explanation:

The sendNotification() method used as an example adheres to the Single Responsibility Principle. Its sole responsibility is to manage the display of notifications to the user. The method checks if notifications are enabled, builds the notification with the necessary parameters, and then displays it. It does not involve itself with other concerns such as data retrieval, UI updates, or business logic; it simply handles the notification aspect. initializeBalanceSensor(), setupBalanceSensor(), updateAxis() and others were designed in a similar manner.

Open/Closed Principle (OCP) from SOLID: Retrieved code from SensorScreen.java within the setupBalanceSensor(), setupMotorSensor(), and setupRangeSensors() methods.

```java
private void setupBalanceSensor() {
   DocumentReference docRef = db.collection("PetasosRecord")
           .document("Toronto")
           .collection("Petasos001")
           .document("Balance");
   docRef.addSnapshotListener(new EventListener<DocumentSnapshot>() {
       @Override
       public void onEvent(@Nullable DocumentSnapshot snapshot, @Nullable
FirebaseFirestoreException e) {
           if (e != null) {
               xAxisValue.setText(R.string.server_error);
               yAxisValue.setText(R.string.server_error);
               zAxisValue.setText(R.string.server_error);
               return;
           }

           if ((snapshot != null && snapshot.exists() && isAdded())) {
               Number xAxis = snapshot.getLong("X-axis");
               Number yAxis = snapshot.getLong("Y-axis");
               Number zAxis = snapshot.getLong("Z-axis");

               updateAxis(xAxisProgressBar, xAxisValue, xAxis);
               updateAxis(yAxisProgressBar, yAxisValue, yAxis);
               updateAxis(zAxisProgressBar, zAxisValue, zAxis);
           } else {
               xAxisValue.setText(R.string.no_data);
               yAxisValue.setText(R.string.no_data);
               zAxisValue.setText(R.string.no_data);
           }
       }
   });
}
```

Explanation:

The methods setupBalanceSensor(), setupMotorSensor(), and setupRangeSensors() in SensorScreen.java adhere to the Open/Closed Principle. They are designed to listen for updates in the Firestore database and reflect these changes in the UI without modifying the methods themselves. If a new sensor type needs to be tracked, we can extend the functionality by adding a new setup method following the existing pattern without changing the existing methods.

Progress Since Deliverable 2:

• Settings screen: New features included such as Default Address, functional Enable Notifications, etc. Also, now Shared Preferences are used to save user inputs and settings between sessions.

• The feedback screen has been created with the usage of the cloud database for easy data retrieval and display.

• Login Screen: Users can now use the option "Remember Me" to Automatically login in the App.

• Registration Screen: A registration screen is now fully functional and corresponds to Product Owner requirements.

• Home & Search Screen: fully functional search engine was implemented in the application. Products can be searched by name, category or producer. Added functionality to buttons on the Home screen to search for products of the specific types.

• Account Management Screen: Account management screen was added to change records FireStore Database gets when users register.

• Sensor Screen: all sensor screens except GPS were merged into one for better user experience(less useless data). Sensor screens read sensors' data from the database and react accordingly(Change in UI objects, notifications, etc).

• Sensor Screen: all sensor screens except GPS were merged into one for better user experience(less useless data). Sensor screens read sensors' data from the database and react accordingly(Change in UI objects, notifications, etc).

• Home Screen: cosmetic changes in the screen design.

• GPS Screen: now reads the data and displays package location.

Runtime Permissions Implemented:

In the PetasosExpress App, runtime permissions are an important feature that respects user privacy and control, especially when dealing with sensitive capabilities like making phone calls and sending notifications.
Here’s how we manage these permissions:

1. Permission Declaration in Manifest:
The `android.permission.CALL_PHONE` and `android.permission.POST_NOTIFICATIONS` permissions are declared in the Android Manifest, signalling to the system which permissions may be requested during runtime.

2. Runtime Permission Request:
The app checks for permission before performing phone calls and requests it using a system dialog if not already granted, allowing users to grant or deny explicitly.

3. Handling User Response:
The `onRequestPermissionsResult()` callback processes the user's decision, enabling the app to act accordingly, either by proceeding with the action (if granted) or abstaining (if denied).

This approach respects user privacy by asking for permissions as needed and also provides the user with control over what the app can do with their data or device features.

Main Functionalities Implemented:

- Search Engine (Allows users to look for specific products of companies we have partnerships with).
- Delivery Tracking via GPS.

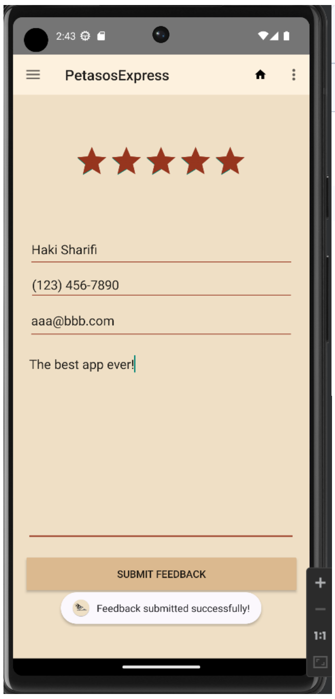
Main Functionalities Implemented Partially:

- User’s Account Registration, Authentication and Management(Ensured that users can create accounts, log in securely, and manage their profiles.)

Main Functionalities Planned to be implemented next Deliverable:

- User’s Account Registration, Authentication and Management.
- Order Placement.
- Payment Processing (Demo/Just a schema).

Customer Feedback Screen stored in the Firestore:



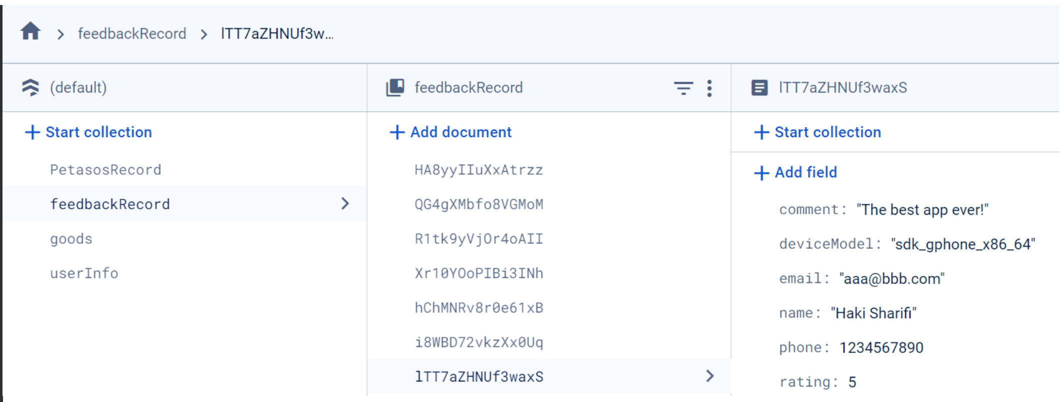
Haki Sharifi

(123) 456-7890

aaa@bbb.com

The best app ever!

Feedback submitted successfully!



PetasosRecord
feedbackRecord
goods
userInfo

HA8yyIIuXxAtrzz
QG4gXMbfo8VGMoM
R1tk9yVj0r4oAII
Xr10YOoPIBi3INh
hChMNRv8r0e61xB
i8WBD72vkzXx0Uq
lTT7aZHNUf3waxS

```
comment: "The best app ever!"
deviceModel: "sdk_gphone_x86_64"
email: "aaa@bbb.com"
name: "Haki Sharifi"
phone: 1234567890
rating: 5
```