Deliverable 4

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

Project Description:

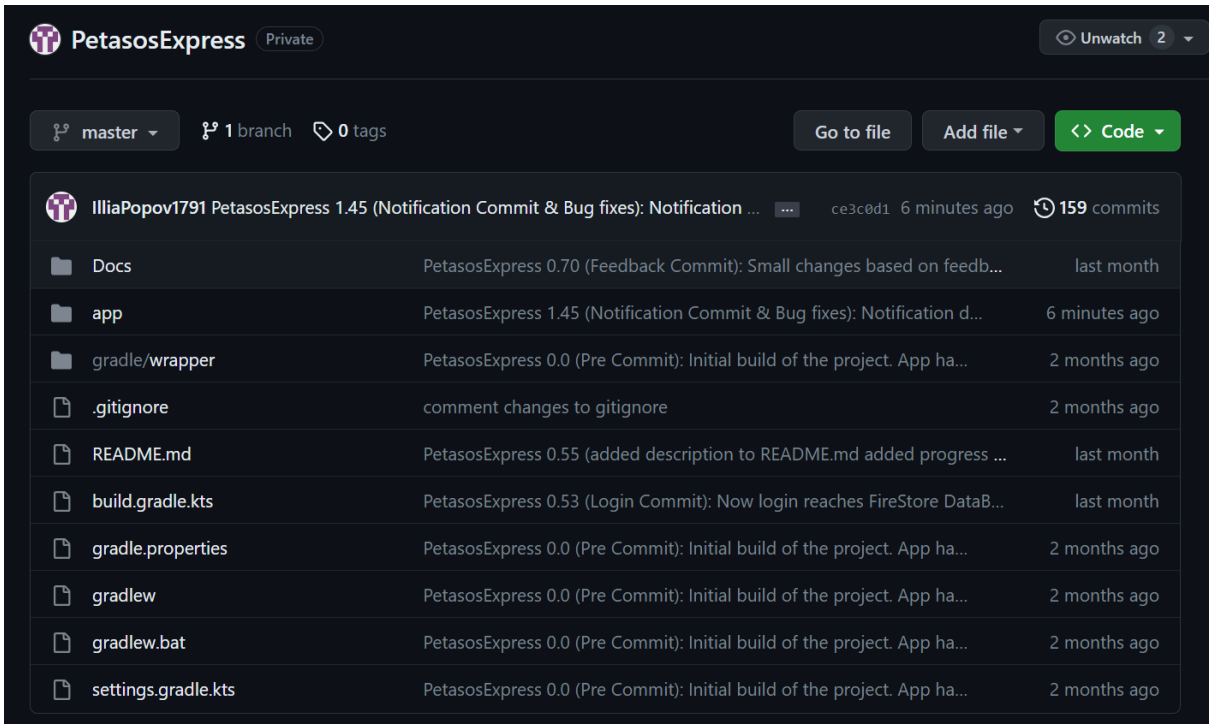
The primary goal of this project is to develop PetasosExpress, a state-of-the-art IoT-enabled Smart Delivery Robot System. This project will encompass the integration of advanced sensors and robotics with a Raspberry Pi 4, the development of a user-friendly Android application, and the establishment of a robust backend server system for efficient route management, data analysis, and user interaction.

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%

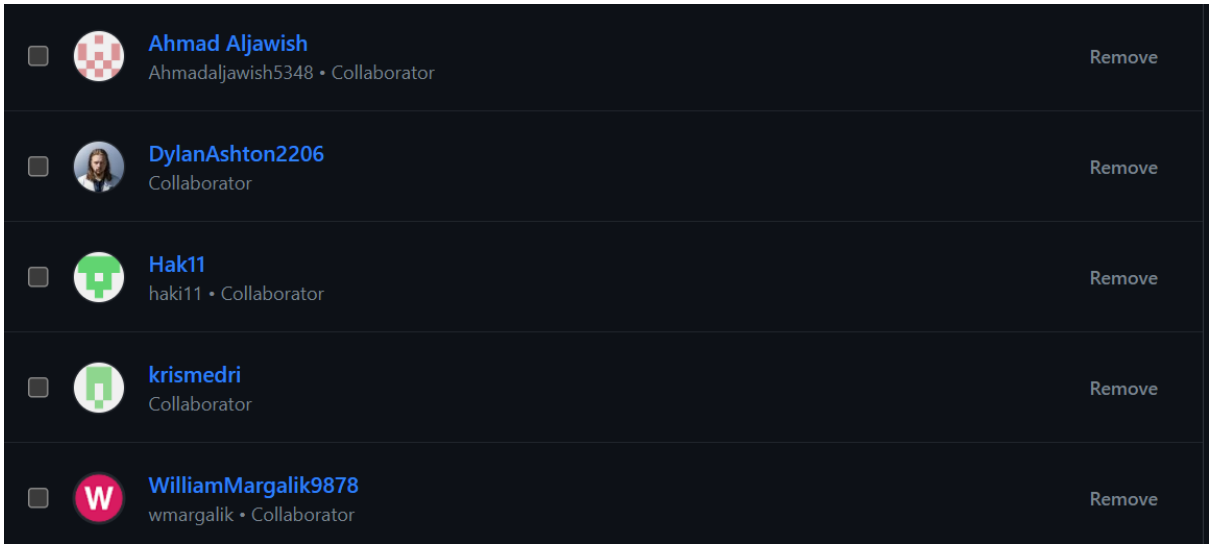
GitHub Repository Links:

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



GitHub Invitation Confirmation:

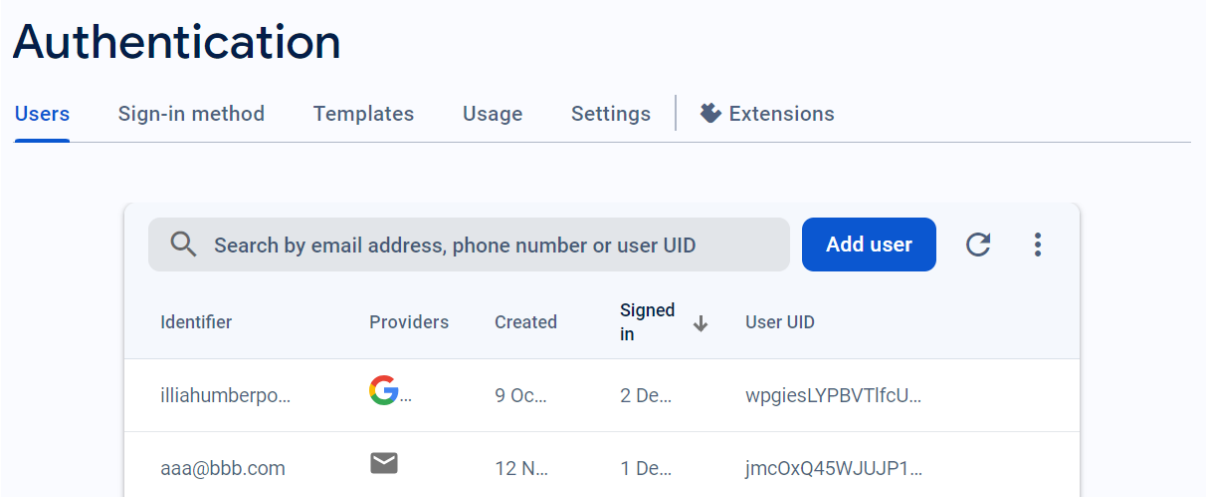
Repository Invites of Software Project and Hardware Production professors, and all team members (Taken by IlliaPopov1791):

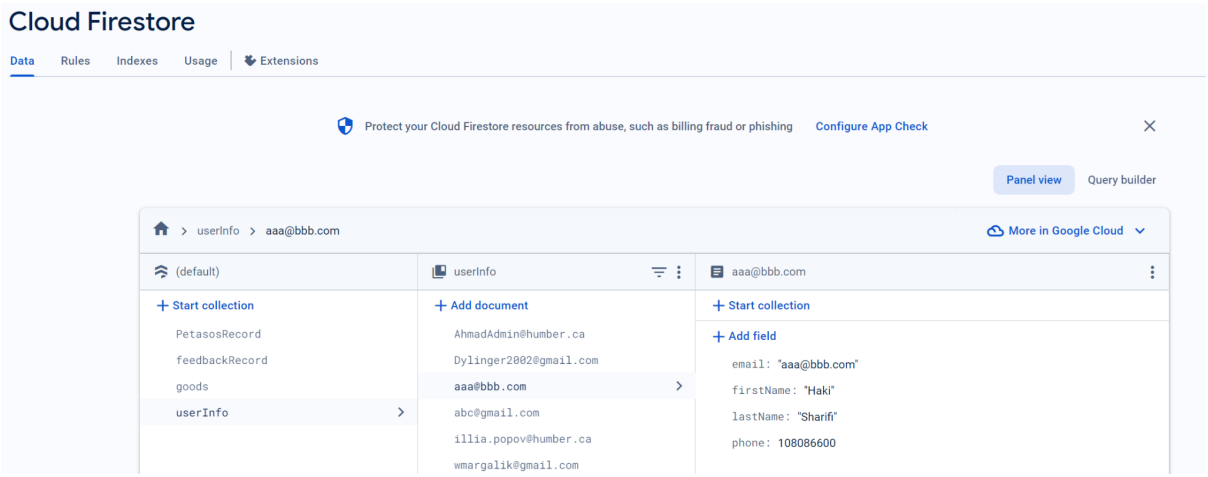


Created account in the DB with requested credentials:

Admin Credential:

Email: aaa@bbb.com Password: Admin101!





Sprint Goals:

- ☑ The sprint goals for our team for deliverable 4 are as follows:
- ☑ Implement testing classes using Junit4, Roblectric and Espresso.
- ☑ Create a functioning floating action button.
- ☑ Implement Offline User Credentials Storage and Synchronization Code refactoring and cleanup.
- ☑ Creating an assigning system for each order to have one unique robot assigned.
- ☑ Changing sensor data fetching process so users will see data only from the robot assigned to their oldest order.
- ☑ Implementing the “queue” system in case of lack of free robots.
- ☑ Polishing Existing Features.

Sprint Dashboard:

Epic 4: Story 12: Implement Offline Data Storage and Synchronization 4 Tasks

Task	Person	Status	Priority	Size	Timeline	Date of comple...
Task 1: Relocate unnecessary for the server data to Shared Preferences	IP	Done	Low	Small	Nov 15 - 19	Nov 19
Task 2: Implement Online Data Synchronization	IP	Done	Medium	Medium	Nov 15 - 19	Nov 19
Task 3: Handle Data Conflicts	IP	Done	Medium	Small	Nov 15 - 19	Nov 19
Task 4: Ensure Smooth Offline-Online Transitions	IP	Done	Low	Small	Nov 15 - 19	Nov 19
+ Add task						
					Nov 15 - 19	

Epic 4: Story 13: Create a Floating Action Button

Task	Person	Status	Priority	Size	Timeline	Date of comple...
Task 1: Create a Floating Action Button in the targeted classes	DA	Done	Medium	Small	Nov 21 - 29	Nov 28
Task 2: Redesign targeted classes' portrait layouts to include FAB	DA	Done	Medium	Small	Nov 21 - 29	Nov 28
Task 3: Redesign targeted classes' landscape layouts to include FAB	DA	Done	Low	Medium	Nov 21 - 29	Nov 28
Task 4: Implement functionality of the Floating Action Button	DA	Done	High	Small	Nov 21 - 29	Nov 28
Task 5: Ensure that FAB works a intended on all screens	DA	Done	Medium	Small	Nov 21 - 29	Nov 28
+ Add task						

Epic 4: Story 14: Modify the feedback screen 4 Tasks

Task	Person	Status	Priority	Size	Timeline	Date of comple...
Task 1: Implement delay for submission of the feedback	DA	Done	Low	Small	Nov 21 - 30	Nov 26
Task 2: Add the progress bar to represent submission process in the desi...	DA	Done	Low	Small	Nov 21 - 30	Dec 26
Task 3: Implement the progression on the progress bar	DA	Done	Low	Small	Nov 21 - 30	Dec 30
Task 4: Restrict Feedback Submission to once in 24 hours	DA	Done	Low	Small	Nov 21 - 30	Nov 26
+ Add task						
					Nov 21 - 30	

Epic 4: Story 15: Payment and Order Tracking Implementation 10 Tasks

Task	Person	Status	Priority	Size	Timeline	Date of comple...
Task 1: Creating portrait layout for Product Screen	IP	Done	Medium	Medium	Nov 21 - Dec	Nov 23
Task 2: Creating landscape layout for Product Screen	AA	Done	Medium	Medium	Nov 21 - Dec	Nov 29
Task 3: Implementing fetching of the correct data for the Product screen	IP	Done	Medium	Medium	Nov 21 - Dec	Nov 23
Task 4: Designing portrait layout for Cart Screen	IP	Done	Medium	Medium	Nov 21 - Dec	Nov 25
Task 5: Designing landscape layout for Cart Screen	AA	Done	Medium	Medium	Nov 21 - Dec	Nov 30
Task 6: Implement Storing and Displaying Cart content	IP	Done	Medium	Medium	Nov 21 - Dec	Dec 25
Task 7: Developing portrait UI for Payment Screen	IP	Done	Medium	Medium	Nov 21 - Dec	Dec 21
Task 8: Developing landscape UI for Payment Screen	AA	Done	Medium	Medium	Nov 21 - Dec	Dec 2
Task 9: Implementing Order placing	IP	Done	Medium	Medium	Nov 21 - Dec	Dec 27
Task 10: Changing sensors code to work with assigned Delivery Robot O...	IP	Done	Medium	Medium	Nov 21 - Dec	Dec 27
+ Add task						
					Nov 21 - D...	

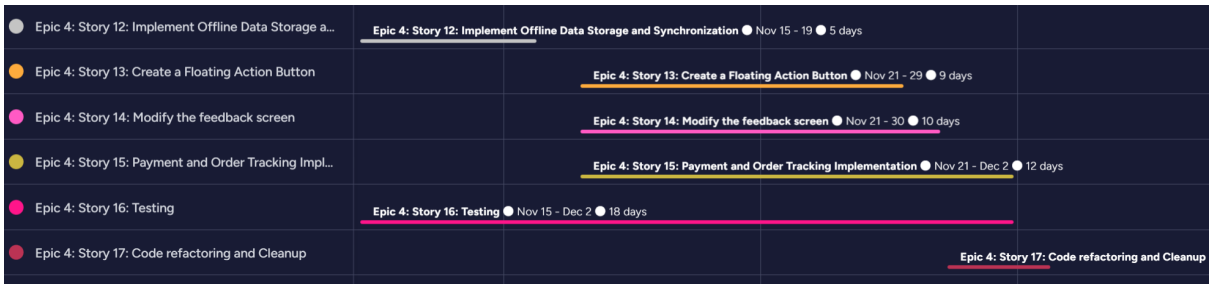
Epic 4: Story 16: Testing 5 Tasks

Task	Person	Status	Priority	Size	Timeline	Date of comple...
Task 1: Junit4 testing for General Verification Processes	AA	Done	Medium	Medium	Nov 15 - Dec	Dec 1
Task 2: Junit4 testing for Product	AA	Done	Medium	Medium	Nov 15 - Dec	Dec 2
Task 3: Robletric Testing	WM	Done	Medium	Medium	Nov 15 - Dec	Dec 2
Task 4: Espresso Testing	WM	Done	Medium	Medium	Nov 15 - Dec	Dec 1
Task 5: Validate all tests pass and function as expected	WM AA	Done	Medium	Medium	Nov 15 - Dec	Dec 2
+ Add task						
					Nov 15 - D...	

Epic 4: Story 17: Code refactoring and Cleanup 5 Tasks

Task	Person	Status	Priority	Size	Timeline	Date of comple...
Task 1: Removing all Commented-Out Code	WM	Done	Medium	Medium	Dec 1 - 3	Dec 3
Task 2: Implement Proper Code Formatting	WM	Done	Medium	Medium	Dec 1 - 3	Dec 3
Task 3: Implement Proper Naming convention	WM AA	Done	Medium	Medium	Dec 1 - 3	Dec 3
Task 4: Refactoring the App	AA WM	Done	Medium	Big	Dec 1 - 3	Dec 3
Task 5: Verifying Functionality after the refactoring	AA	Done	High	Medium	Dec 1 - 3	Dec 3

Gantt Chart:



Daily Standups:

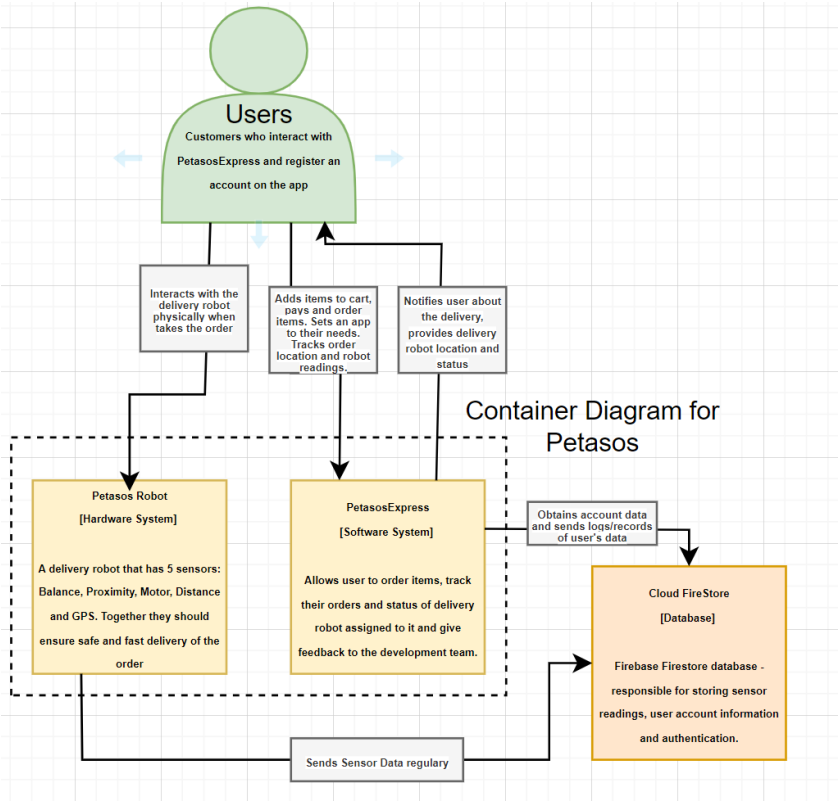
Nov.15	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Manage Account screen and Merged screen function	Account Settings screen with both portrait and landscape	Worked on Sensor screen landscape	Fixed the landscape orientation bug that was not happening for the Home Screen.
	What will you work on today	Start planning tasks for the sprint. Assigned some tasks to people.	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.21 -22	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Sprint planning	Started implementing unit testing	Added delay to feedback screen	Start planning of the sprint
	What will you work on today	Create UI for payment and product screen. Develop the process of product screen fetching firebase data	Creating testclass for appSettings using robolectric	Create progress bar for the delay And fully implement feedback sharedpref 24 hour timer	Configure build settings and unnecessary files: Removed files that were wasting extra space.
	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

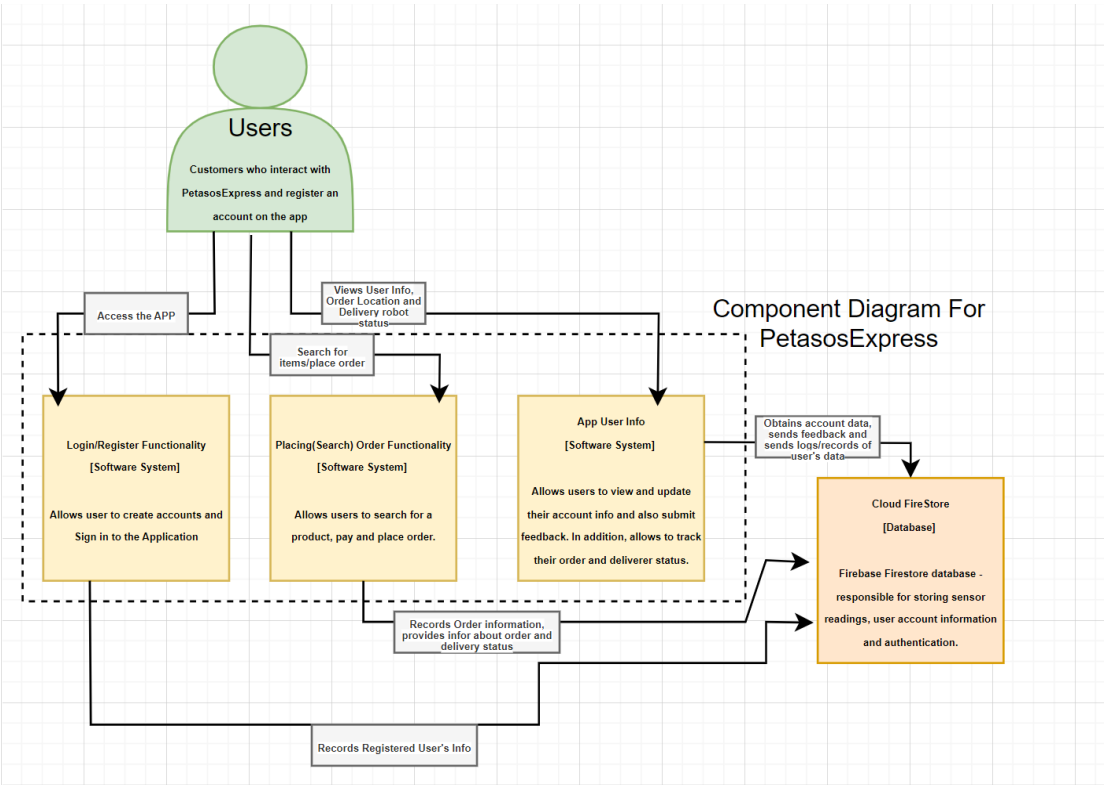
Nov.28	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Develop the process of assigning a specific delivery robot to the client's order. Made sensors show only the assigned robot data.	Testing the Cases in the robolectric test class that's testing the Settings Screen	Added Floating action button to bring user from (search/home/ product) screen to the cart screen when pressed	Configure build settings and unnecessary files: Removed files that were wasting extra space.
	What will you work on today	Working on the "Login with Google" login option. Making changes to the sensors' data fetching functions.	Added landscape screen for Product screen	Improving the looks of feedback Screen Landscape and portrait	Added dependencies for the espresso for androidx
	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.29 Dec.01	Questions	Illia	Ahmad	Dylan	William
	What did you work on yesterday?	Adding a functional login with the Google option	Added landscape screen for Product screen	Visually Improved app settings screen (landscape and portrait)	Added dependencies for the espresso for androidx
	What will you work on today	Testing the Home UI with Espresso. Redesigning Validation process	Added landscape for Cart Screen and added test cases for junit	Visual improvements to sensor screen and account manage	Cleanup/ Organise code
	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

Container Diagram:



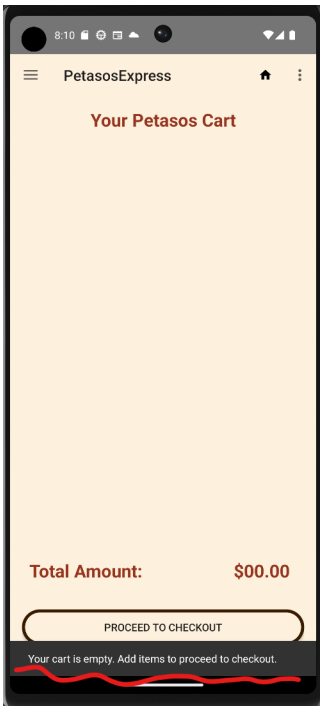
Component Diagram:



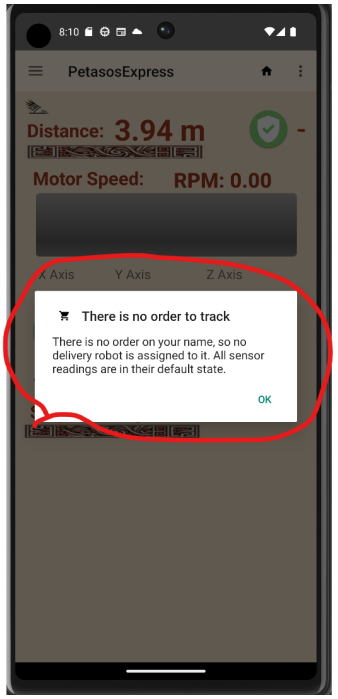
User Notification Functionality:

Toast:

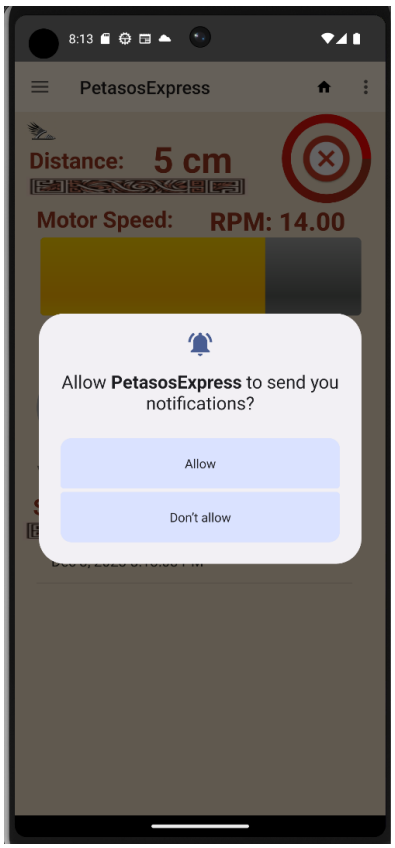
Snackbar:

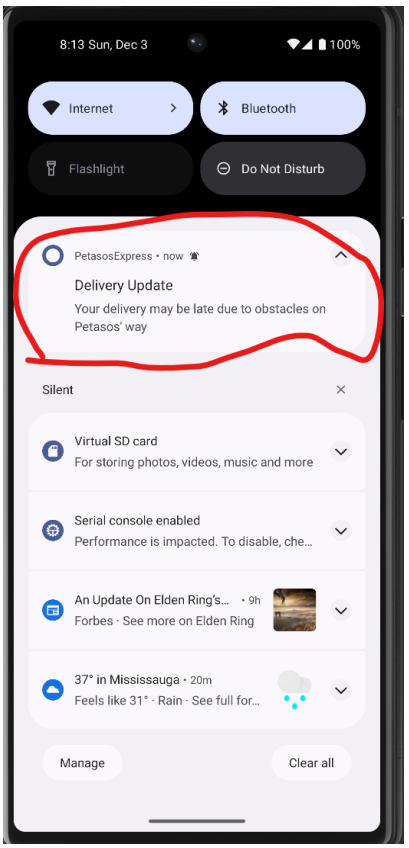


Alert Dialog:



Notifications (Requires Permissions):

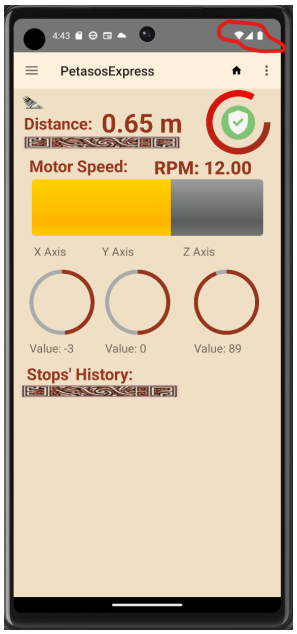




Offline Functionality:

Since the App is using Firestore: it supports offline data persistence by default. This means that Firestore has an offline feature that allows the application to continue functioning even when it temporarily loses network connectivity.

If a user loses an internet connection, they will stay logged into the app and can still view their data.



Example shows the sensors' screen before and after losing the connection. Data is still showing after the fragments were changed.

Work on feedback Provided:

- Goals' scale is more reasonable
- Avoid hardcoding numbers
- Implement Login with Google

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). Below is the code showing implementation of the CALL_PHONE runtime permission:

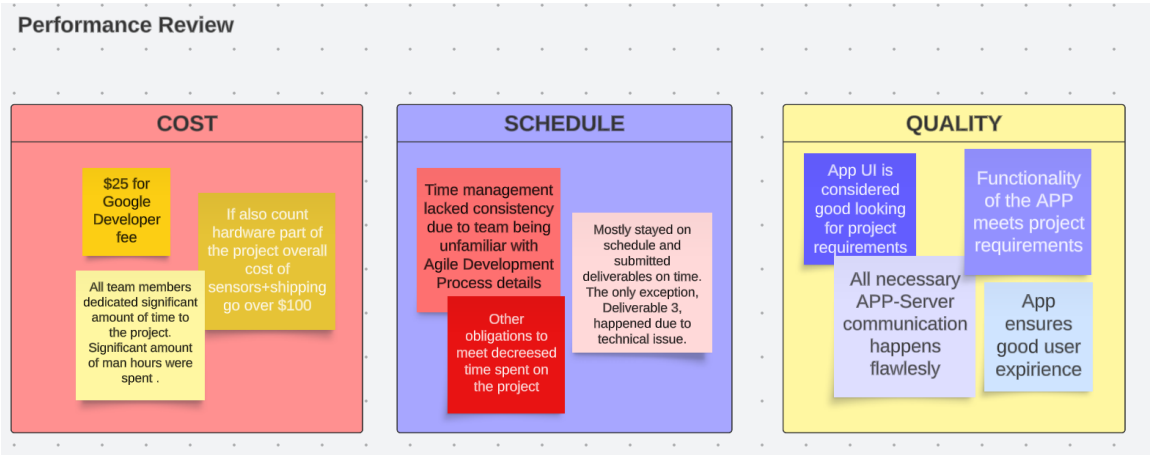
```
private void initiateCall() {
  if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CALL_PHONE}, 123);
    return;
  }
  callPhoneNumber();
}
private void callPhoneNumber() {
  Intent callIntent = new Intent(Intent.ACTION_CALL);
  callIntent.setData(Uri.parse(getString(R.string.tel_1234567890))); // Adjust with your phone number
  startActivity(callIntent);
}
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
  super.onRequestPermissionsResult(requestCode, permissions, grantResults);
  if (requestCode == 123) {
    if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
      Snackbar.make(drawerLayout, R.string.permission_granted, Snackbar.LENGTH_SHORT).show();
      callPhoneNumber();
    } else {
      Snackbar.make(drawerLayout, R.string.permission_denied, Snackbar.LENGTH_SHORT).show();
    }
  }
```

[redacted]

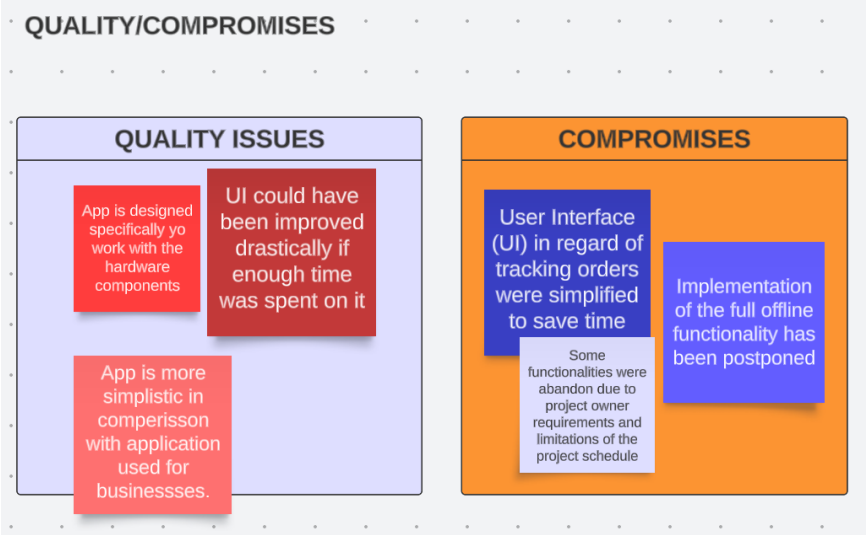
This approach respects user privacy by asking for permissions as needed and also provides the user with control over what the app can do with their data or device features.

Post-Mortem & Project Review Meeting:

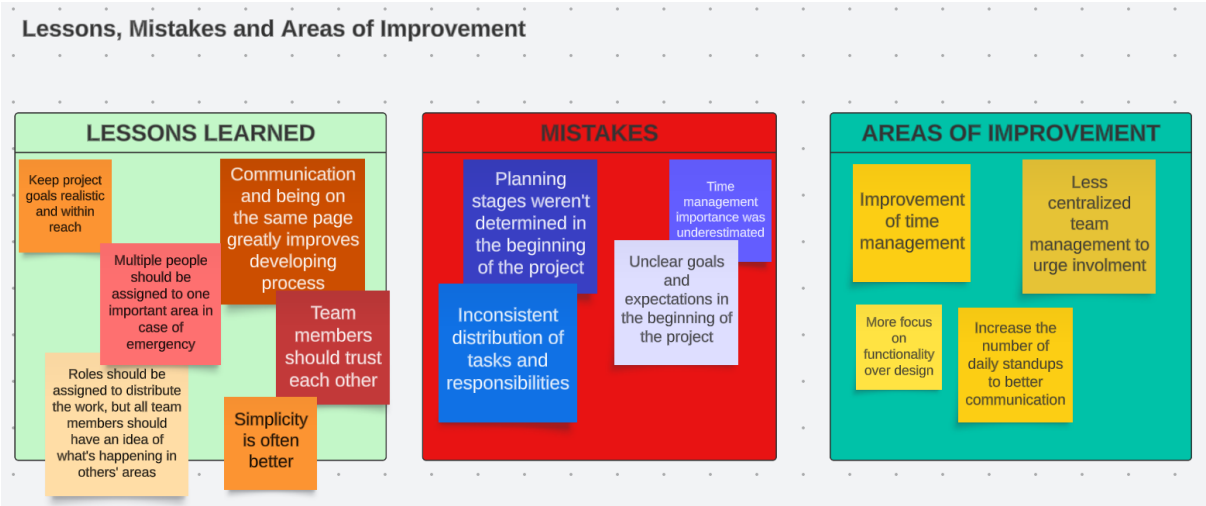
Performance Review:



Quality/Compromise:



Lessons, Mistakes and Areas of Improvement:



Addressing Technical Debt:

In our project, our initial priority was to establish the required functionality. After achieving this, we shifted our focus to improving and reorganising the code.

- Throughout the development process, we adhered to various design patterns and principles.
- We also maintained a close watch on technical debt, allocating specific times to address it.
- Our adoption of agile methodologies helped us keep technical debt under control. This approach ensured that issues and bugs identified in earlier sprints, based on user feedback and testing outcomes, were resolved at the beginning of each new sprint.
- As the project neared completion, we developed automated test cases.

Areas of Refactoring:

Area 1 - Validation for Feedback (Used as example), Login and Registration:

Before:

```diff
@@ -104,27 +104,36 @@ private void submitFeedback() {
            String comment = editTextComment.getText().toString().trim();
            float rating = ratingBar.getRating();

-           if (name.isEmpty() || phone.isEmpty() || email.isEmpty()) {
-               DisplayToast(getString(R.string.please_fill_in_all_fields));
                return;
            }

-           if (!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
                DisplayToast(getString(R.string.invalid_email_format));
                return;
            }

-           String cleanPhone = phone.replaceAll("[^0-9]", "");
-           if (cleanPhone.length() != 10) {
                DisplayToast(getString(R.string.invalid_phone_format));
                return;
            }
```

After:

```java
if (!GeneralValidationUtils.isValidName(name)) {
    DisplayToast(getString(R.string.please_fill_the_name_field));
    return;
}

if (!GeneralValidationUtils.isValidEmail(email)) {
    DisplayToast("Please enter a valid email address.");
    return;
}

if (!GeneralValidationUtils.isValidPhone(phone)) {
    DisplayToast("Invalid phone format");
    return;
}

if (!GeneralValidationUtils.isValidComment(comment)) {
    DisplayToast(getString(R.string.please_leave_a_comment));
    return;
}

if (!GeneralValidationUtils.isValidRating(rating)) {
    DisplayToast(getString(R.string.rating_must_be_between_1_0_and_5_0));
    return;
}
```

```java
//Method to validate email
3 usages  IlliaPopov1791
protected static boolean isValidEmail(String email) {
    return android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches();
}

//Method to validate phone number
4 usages  IlliaPopov1791
protected static boolean isValidPhone(String phone) {
    String cleanPhone = phone.replaceAll( regex: "[^0-9]",  replacement: "");
    String phonePattern = "^[0-9]{10}$";
    return cleanPhone.matches(phonePattern);
}

//Method to validate password
2 usages  IlliaPopov1791
protected static boolean isValidPassword(String password) {
    String passwordPattern = "^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=!])(?=\\S+$).{6,}$";
    return password.matches(passwordPattern);
}

3 usages  IlliaPopov1791
public static boolean isValidRating(float rating) { return rating >= 1.0f && rating <= 5.0f; }

3 usages  IlliaPopov1791
public static boolean isValidComment(String comment) {
    return comment != null && !comment.trim().isEmpty();
}
2 usages  ahmadaljawish5348
public static boolean isValidName(String name) {
    return name != null && !name.trim().isEmpty();
}
}
```

Comments:

The notable changes between the original and refactored code involve the relocation and centralization of validation logic:

- **Consolidation of Validation Logic**: In the initial code, validation for emails, phone numbers, passwords, ratings, comments, and names was dispersed across multiple classes, leading to redundancy and potential inconsistencies. The refactored code introduces a new class, GeneralValidationUtils, within the ca.hermeslogistics.itservices.petasosexpress package. This class centralises all validation methods, such as isValidEmail, isValidPhone, isValidPassword, isValidRating, isValidComment, and isValidName. Each method is designed to handle specific input validations and returns a boolean indicating the validity of the input.

- **Enhancement in Code Reusability and Maintenance:** By moving the validation logic into the GeneralValidationUtils class, the code becomes more reusable. Different classes can now invoke these methods to validate different types of inputs without duplicating the logic. This approach significantly improves the maintenance of the code. Any changes or enhancements in validation logic need to be made only in the GeneralValidationUtils class, rather than in multiple places across the codebase.

- **Improved Code Organization and Readability:** The refactoring results in a cleaner and more organised code structure. Validation methods are neatly grouped together, making it easier for developers to locate and understand the validation rules.

Area 2 - Separation of Concerns in Search Screen:

Before:

```
private void fetchItems(String initialQuery) {
    db.collection( collectionPath: "goods").get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            // Clear lists
            productList.clear();
            fullItemList.clear();

            // Parse Firestore documents
            for (QueryDocumentSnapshot document : task.getResult()) {
                String name = document.getString( field: "name");
                Long id = document.getLong( field: "id");
                Double price = document.getDouble( field: "price");
                String producer = document.getString( field: "producer");
                String type = document.getString( field: "type");

                // Check for null values
                if (name != null && price != null) {
                    Product product = new Product(name, id.intValue(), price, producer, type);
                    fullItemList.add(product);
                    productList.add(product);
                }
            }

            // Notify adapter
            productAdapter.notifyDataSetChanged();

            // Filter items based on initial query
            if (!initialQuery.isEmpty()) {
                filterAdapter(initialQuery);
            }
        } else {
            Log.e( tag: "FetchItems",  msg: "Error fetching documents", task.getException());
        }
    });
}
```

After:

```java
private void fetchItems(String initialQuery) {
    db.collection( collectionPath: "goods").get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            productList.clear();
            fullItemList.clear();

            for (QueryDocumentSnapshot document : task.getResult()) {
                Product product = parseDocument(document);
                if (product != null) {
                    fullItemList.add(product);
                    productList.add(product);
                }
            }

            productAdapter.notifyDataSetChanged();

            if (!initialQuery.isEmpty()) {
                filterAdapter(initialQuery);
            }
        } else {
            Log.e( tag: "FetchItems",  msg: "Error fetching documents", task.getException());
        }
    });
}

// Parse a Firestore document into a Product object
1 usage   wmargalik *
private Product parseDocument(QueryDocumentSnapshot document) {
    String name = document.getString( field: "name");
    Long id = document.getLong( field: "id");
    Double price = document.getDouble( field: "price");
    String producer = document.getString( field: "producer");
    String type = document.getString( field: "type");

    if (name != null && id != null && price != null) {
        return new Product(name, id.intValue(), price, producer, type);
    }
    return null;
}
```

Comments:

- **Enhanced Readability and Maintainability:** The separation of document parsing logic into parseDocument simplifies the fetchItems method, making the codebase more modular and easier to understand. This separation aids in maintainability, as modifications in parsing logic are isolated from the data fetching logic.
- **Adherence to Single Responsibility Principle(S from SOLID)**: parseDocument adheres to the Single Responsibility Principle by exclusively managing the creation of Product objects from Firestore documents. This makes the code more robust, facilitating easier testing and future modifications.
- **Scalability and Reusability:** The isolated parsing logic in parseDocument can be reused or extended if similar functionality is needed elsewhere in the application. This promotes scalability and reduces potential duplication of logic across the codebase.

Suggestions to the instructor for future projects:

- **Continuous Feedback:** Incorporate continuous feedback loops with the professor to imitate a product owner's presence on daily stand-ups. Maybe use a discussion board on Blackboard so professors can increase their involvement.
- **Scrum Ceremonies:** ensure students practise Agile ceremonies like daily stand-ups, sprint planning, and sprint reviews **before the first deliverable**, as many people have a vague understanding of the process at that point and try wrong approaches.