NATIONAL UNIVERSITY

School of Information Technology

Finance Tracker

March 11, 2024

APPLICATIONS DEVELOPMENT

AND EMERGING TECHNOLOGIES

FINAL PROJECT

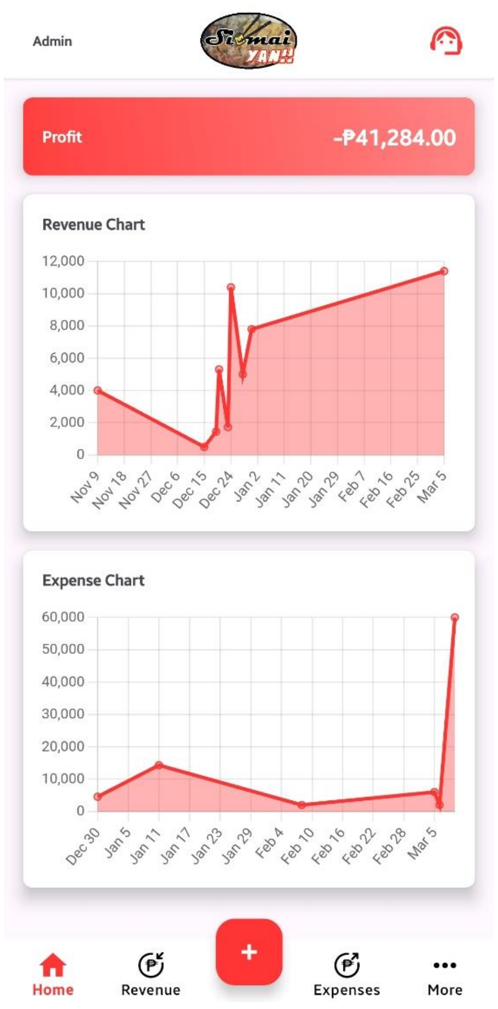
Salles, Francis James E.

INF224

TABLE OF CONTENTS

USER INTERFACE DESIGN

Simple Design



The app uses the simplest design possible to allows the user to navigate the app easily, focus on what's essential, remove clutter and unnecessary elements. This helps the interface to be less overwhelming and easier for beginners to understand.

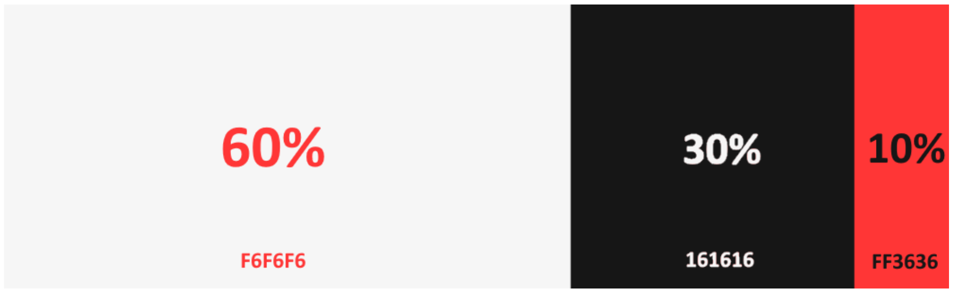
60-30-10 Color Rule

This application utilizes the 60-30-10 color guideline when it comes to deciding on what color are used for each component. It helps to create a balanced and aesthetically pleasing color scheme which highlights important elements and guide user towards specific functions.

The 60% or the dominant color in this application is white. It covers majority of the background and main UI elements of the application. It sets the overall tone and feel of the application.

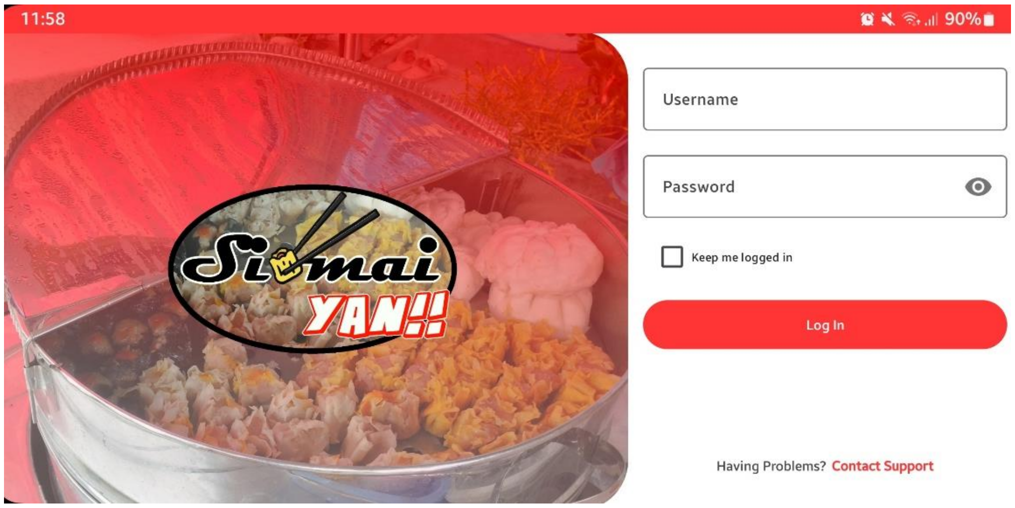
The 30% is the secondary color, this color supports the dominant color and adds more visual interest to the design. In this app, it is used in typography, icons, labels, and title.

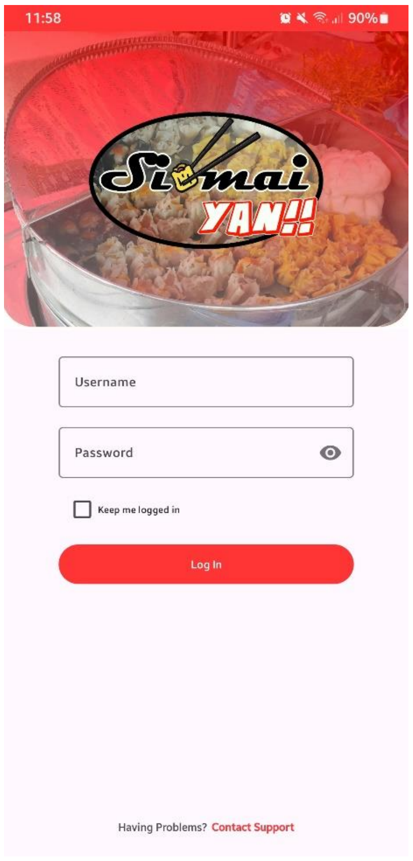
Lastly, the 10% is the accent color of the app which in this case is red. It helps highlights specific sections of the design like buttons, call-to-action elements, or any element that needs emphasis in a design.

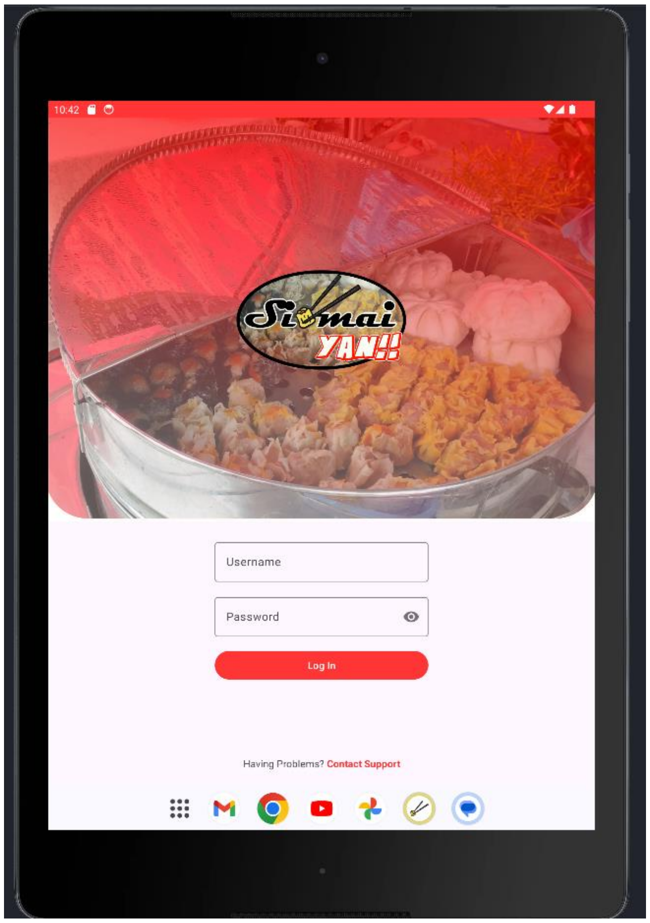


Responsive Application

The application is responsive which means the application is designed to automatically adjust its layout and functionality based on the orientation of the device and the device it's being accessed from. This ensures a smooth user experience regardless of whether someone is using a tablet, a television, or a mobile phone.







FEATURES AND FUNCTIONALITIES

1. Two types of User
   - The application has two types of users: Employees and Admin
     - Employees can use the app with only limited features. They can track their own performance and total income earned throughout their employment.
     - Admin can use all the features of the application which includes adding the revenue share of the employees. Admin is also the only one who can create, delete, and manage the account of employees.
2. Online Database
   - This application utilizes an online database that stores different data that the application tracks. This way, data can be retrieved and modified on different phones.
3. Offline Support
   - With the use of Firestore Database, admins can view, add, delete, and modify different data even offline. Firestore stores a copy of the database on your phone as a cache that allows you to view the data in the database even if offline. When you add a data while offline, it stores it in the cache which will be put on the cloud once you are connected to the internet.
4. Revenue Tracking
   - This application allows the admins to track the revenue of each transaction with the employees when assigned in the food cart. The app stores the following information:
     - Date
     - Location
     - Seller
     - Revenue
   - This helps the business to monitor the money being earned by the business in different dates, location, and seller.
5. Expense Tracking
   - The application also allows the owners to track the expenses the business make. These includes buying raw supplies from retailers, buying appliances for better customer service and efficient processing of products.
   - This helps the business owners to see how much they are spending for the business and check if they are earning more money or already losing money.
6. Employee’s Share Tracking

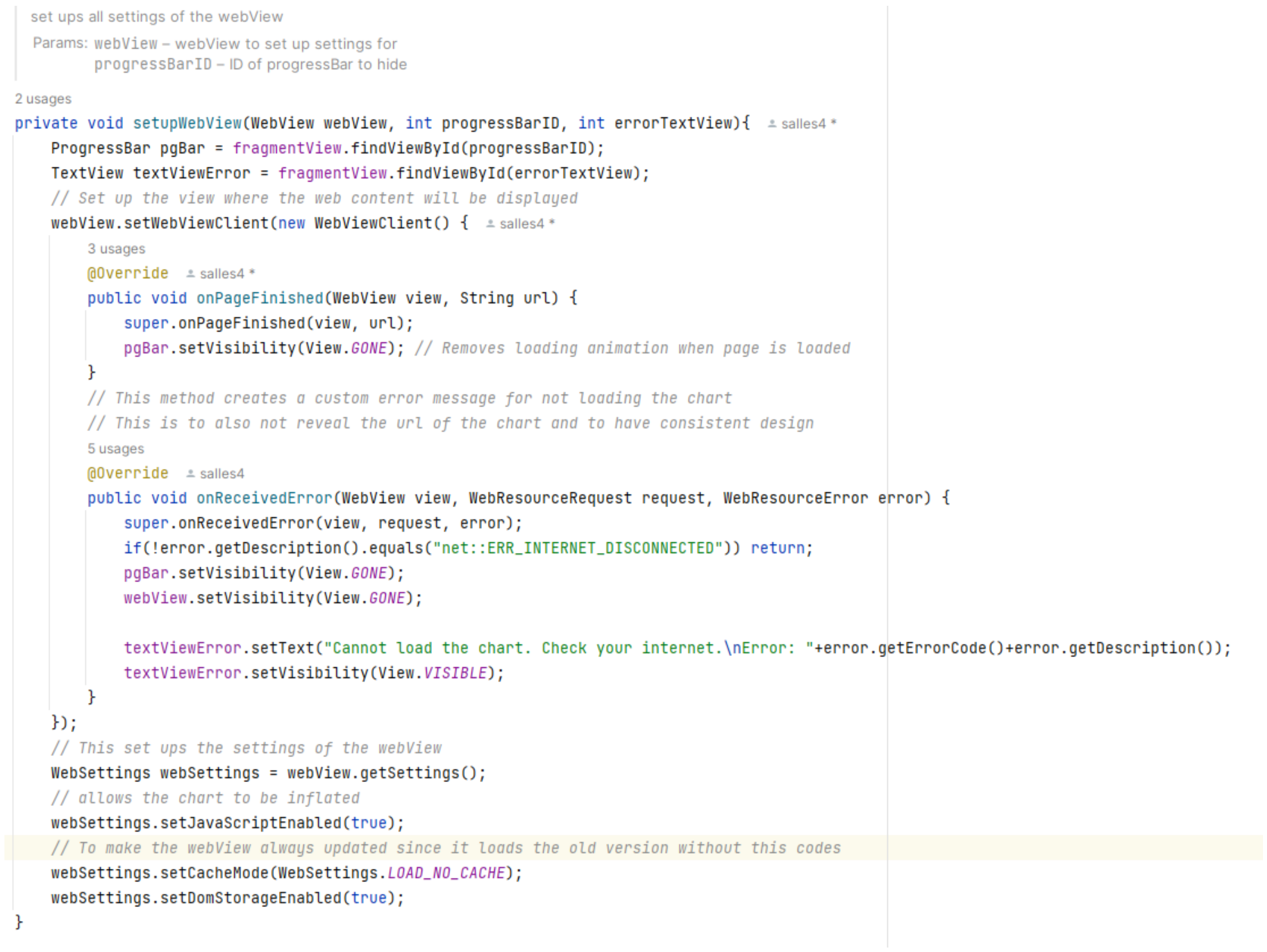
- Tracking the employee’s share on the revenue helps the admin to monitor how much an employee makes in a transaction and it keeps track a history of their earnings to see if they are selling the products well.

7. Add Data
   - A tracker is nothing without data. In this application, the admin can add data through the application which will be saved in the cloud. When adding data, the application strictly makes sure that the data provided are intended and correct by checking if it meets all the requirement of specific data field.
8. Managing Data
   - Admins can also edit and delete these data through the app which then will also be updated in the database.
9. Charts
   - The app features a line chart that displays the performance of the business. This helps the business to easily see how much the business earned or spent in different times.
10. Adapts to Data
    - This application adapts to data changes, which means the app can handle when there is a new data or when data are modified or deleted. An example is when an employee is added, its data will be added to suggestions when assigning an employee, it will also create a chart that contains their performance. Like most of the applications that are being used these days.
11. Total Business Profit
    - As the app track the revenue and expenses, the app displays the total profit by deducting the expenses to the revenue. This helps the business owners to see if the business is doing well or they are losing money.
12. Keep Me Logged In
    - Once checked, you no longer need to enter your credentials again next time you open the app. However, when you log out, you have to use your credentials again to log in.
13. Contact Support
    - Contacting tech support is one click away with this button. This directs them to the messenger of the tech support that will help you with the application or data management.
14. Contact Employee
    - The app also features an easy way to manage and contact employees with the phone number or Facebook messenger.

15. Image Slider
    - This displays different images related to the business, it can either be an ad, news, or updates.
16. Splash Screen
    - Splash screen helps the app to retrieve all data first from the database and directs you to log in page if you are not logged in. If you are logged in and selected “Keep Me Logged In”, it will redirect you to your logged in account.
17. Filter
    - This allows the user to filter listed data with only specified filter. For example, if the user selected a location and a seller, the revenue list will update, and it will only display transactions with that location and seller.

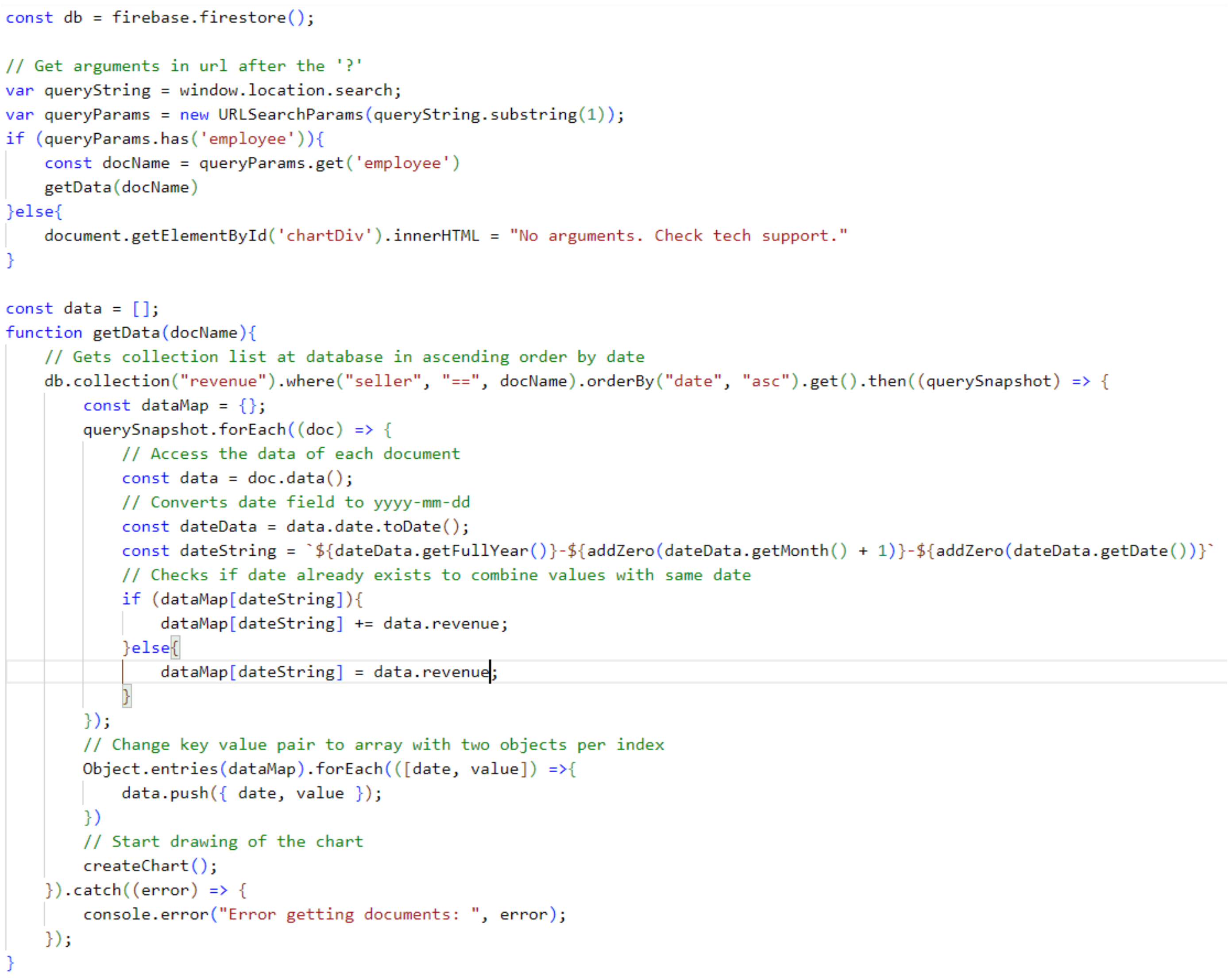
CODE EXAMPLES AND EXPLANATIONS

- Code snippet of creating a WebView for the chart. It features custom error message and a loading indicator.



```
set ups all settings of the webView
Params: webView – webView to set up settings for
        progressBarID – ID of progressBar to hide

2 usages
private void setupWebView(WebView webView, int progressBarID, int errorTextView){  salles4 *
    ProgressBar pgBar = fragmentView.findViewById(progressBarID);
    TextView textViewError = fragmentView.findViewById(errorTextView);
    // Set up the view where the web content will be displayed
    webView.setWebViewClient(new WebViewClient() {  salles4 *
        3 usages
        @Override  salles4 *
        public void onPageFinished(WebView view, String url) {
            super.onPageFinished(view, url);
            pgBar.setVisibility(View.GONE); // Removes loading animation when page is loaded
        }
        // This method creates a custom error message for not loading the chart
        // This is to also not reveal the url of the chart and to have consistent design
        5 usages
        @Override  salles4
        public void onReceivedError(WebView view, WebResourceRequest request, WebResourceError error) {
            super.onReceivedError(view, request, error);
            if(!error.getDescription().equals("net::ERR_INTERNET_DISCONNECTED")) return;
            pgBar.setVisibility(View.GONE);
            webView.setVisibility(View.GONE);

            textViewError.setText("Cannot load the chart. Check your internet.\nError: "+error.getErrorCode()+error.getDescription());
            textViewError.setVisibility(View.VISIBLE);
        }
    });
    // This set ups the settings of the webView
    WebSettings webSettings = webView.getSettings();
    // allows the chart to be inflated
    webSettings.setJavaScriptEnabled(true);
    // To make the webView always updated since it loads the old version without this codes
    webSettings.setCacheMode(WebSettings.LOAD_NO_CACHE);
    webSettings.setDomStorageEnabled(true);
}
```

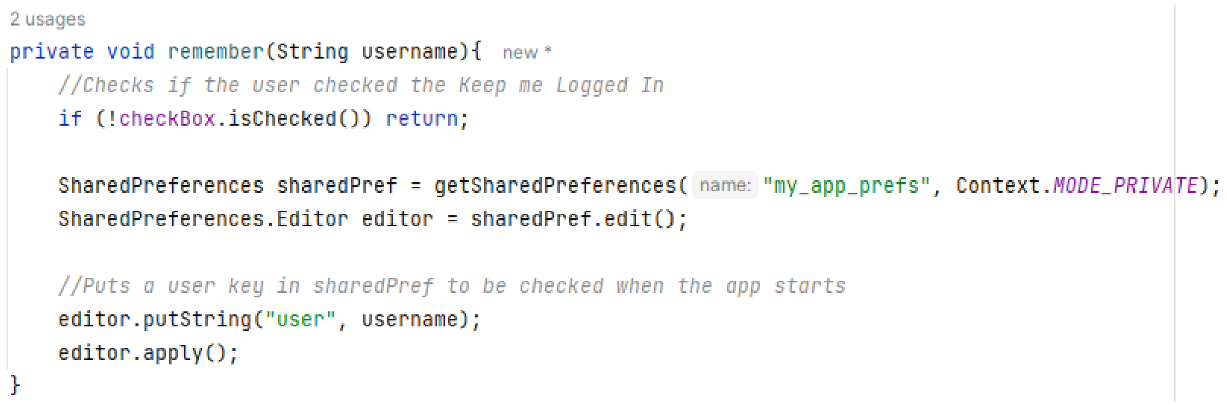
- The following code snippet is a JavaScript code of retrieving data from the database storing it in a key value pairs and array to be used in the chart.



```javascript
const db = firebase.firestore();

// Get arguments in url after the '?'
var queryString = window.location.search;
var queryParams = new URLSearchParams(queryString.substring(1));
if (queryParams.has('employee')){
    const docName = queryParams.get('employee')
    getData(docName)
}else{
    document.getElementById('chartDiv').innerHTML = "No arguments. Check tech support."
}

const data = [];
function getData(docName){
    // Gets collection list at database in ascending order by date
    db.collection("revenue").where("seller", "==", docName).orderBy("date", "asc").get().then((querySnapshot) => {
        const dataMap = {};
        querySnapshot.forEach((doc) => {
            // Access the data of each document
            const data = doc.data();
            // Converts date field to yyyy-mm-dd
            const dateData = data.date.toDate();
            const dateString = `${dateData.getFullYear()}-${addZero(dateData.getMonth() + 1)}-${addZero(dateData.getDate())}`
            // Checks if date already exists to combine values with same date
            if (dataMap[dateString]){
                dataMap[dateString] += data.revenue;
            }else{
                dataMap[dateString] = data.revenue;
            }
        });
        // Change key value pair to array with two objects per index
        Object.entries(dataMap).forEach(([date, value]) =>{
            data.push({ date, value });
        })
        // Start drawing of the chart
        createChart();
    }).catch((error) => {
        console.error("Error getting documents: ", error);
    });
}
```

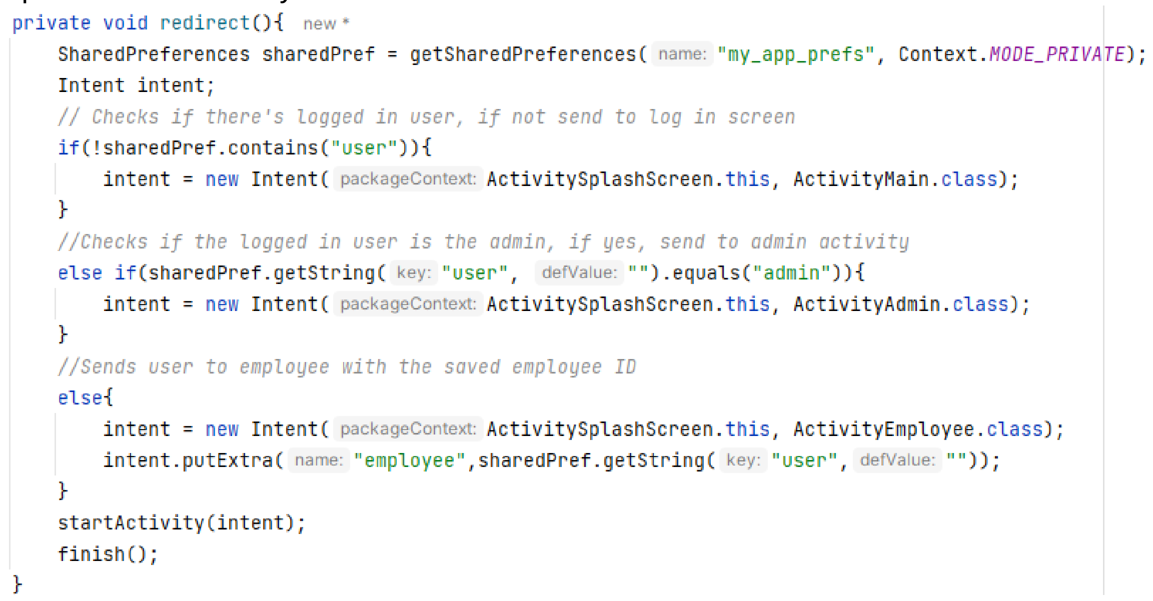
- The following code snippets are for the “Keep me Logged In” feature:

Log In Activity:



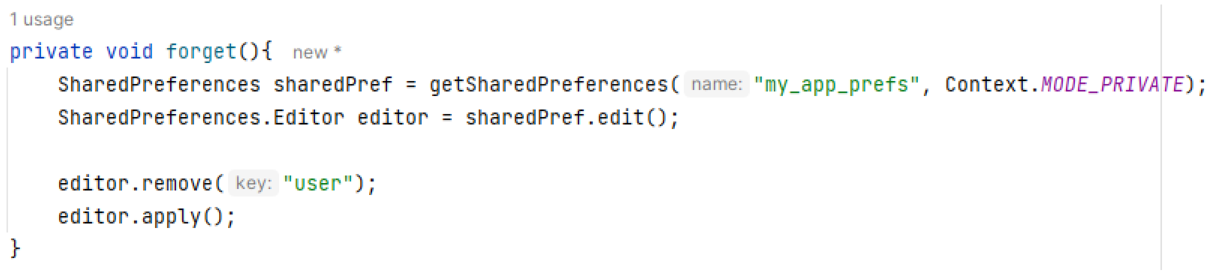
```
2 usages
private void remember(String username){  new *
    //Checks if the user checked the Keep me Logged In
    if (!checkBox.isChecked()) return;

    SharedPreferences sharedPref = getSharedPreferences( name: "my_app_prefs", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPref.edit();

    //Puts a user key in sharedPref to be checked when the app starts
    editor.putString("user", username);
    editor.apply();
}
```

Splash Screen Activity:



```
private void redirect(){  new *
    SharedPreferences sharedPref = getSharedPreferences( name: "my_app_prefs", Context.MODE_PRIVATE);
    Intent intent;
    // Checks if there's logged in user, if not send to log in screen
    if(!sharedPref.contains("user")){
        intent = new Intent( packageContext: ActivitySplashScreen.this, ActivityMain.class);
    }
    //Checks if the logged in user is the admin, if yes, send to admin activity
    else if(sharedPref.getString( key: "user",  defValue: "").equals("admin")){
        intent = new Intent( packageContext: ActivitySplashScreen.this, ActivityAdmin.class);
    }
    //Sends user to employee with the saved employee ID
    else{
        intent = new Intent( packageContext: ActivitySplashScreen.this, ActivityEmployee.class);
        intent.putExtra( name: "employee",sharedPref.getString( key: "user", defValue: ""));
    }
    startActivity(intent);
    finish();
}
```

When the user log outs:



```
1 usage
private void forget(){  new *
    SharedPreferences sharedPref = getSharedPreferences( name: "my_app_prefs", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPref.edit();

    editor.remove( key: "user");
    editor.apply();
}
```

CHALLENGES AND SOLUTIONS

1. Time Constraint
   - One of the major challenges faced during creating this app is the lack of time and focus on creating the app. With 5 other subjects that also needs to be focused on, I had to remove some of the features that I want to add. This also led to less efficient code on some areas and less user interface polish.
2. Constant Change of Mind
   - Being indecisive and constant change of design led to wasting a lot of time. Even though the development of the application started early, the constant change of components as new features is being learnt led to time being used on unnecessary overhaul of the initial concept. For example, in previous concept I use the layout inflator and scroll view to create the list but in the later date I changed it into recycler view.
3. Charts
   - Last year during the presentation of the Inventory System for the final project of Objected-Oriented Programming. One of the professor suggested to include a chart to show visual presentation of the data in the SQLite. This term, I implemented it in this application but not so easily because android charts does not fit the application design. That’s why I used Chart JS a JavaScript library that is very customizable.
4. Image Fit
   - Since one of features of this app is the responsive design, fitting an image in Android Studio the way you wanted is hard. You have little control on how the image will behave. That’s why using vector graphics is better for icons since it is customizable and can be easily updated.

CONCLUSION

The **Siomai 'Yan! Finance Tracker** app helps the business owners to keep track of their sales and allows them to manage different type of data efficiently, record financial transactions, monitor business performance and a lot more. With the use of an online database, admins and employees can log in anytime and anywhere through the app even on different phones. For the future, the app will continue to get updates like product calculator, money counter, search feature, and a lot more. For the employee's side of the app, there are plans of allowing them to customize their profile more like having a profile picture or changing your username. Adding a math mini game is also in the plans to enhance money counting skills.

REFERENCES

For storing different types of data in the application:

Cloud Firestore

For hosting the site and displaying the charts:

GitHub Pages Chart.js JS

For extra features:

android-gif-drawable Image Slide Show

For animation and editing: