

# AUTOPIA: Automotive Care System

Tan Xin Li

*School of Computing*

*Asia Pacific University of Technology  
and Innovation (APU)*  
Kuala Lumpur, Malaysia

TP057927@mail.apu.edu.my

Amad Arshad

*School of Computing*

*Asia Pacific University of Technology  
and Innovation (APU)*  
Kuala Lumpur, Malaysia

amad.arshad@apu.edu.my

Reshiwaran Jegatheswaran

*School of Technology*

*Asia Pacific University of Technology  
and Innovation (APU)*  
Kuala Lumpur, Malaysia

reshiwaran@apu.edu.my

**Abstract**— Currently in Malaysia, the public awareness of vehicle care remained low; besides, there exists difficulties for vehicle owners to contact and place appointments with workshops for routine vehicle servicing. The software development method selected is Rapid Application Development (RAD), considering there is only a sole developer to this project and the given development time is limited. Then, to ensure the title proposed has demands, the developer conducted two requirements gathering activities with workshops and the public to dive deeper into this topic. It was found that this application is deemed useful by most of the respondents while they had provided valuable opinions to the developer to enhance the project too. After evaluating the responses, the developer had finally fixated the project scope and its features.

**Keywords**—*Automotive, Care System, management system, queuing system, Appointment management system*

## I. INTRODUCTION

Automotive industries are becoming a well-established industry as the usage of automotive vehicles in Malaysia is increasing drastically. Malaysian are purchasing brand new and used Automotive vehicles for personal use. Purchasing a brand-new vehicle would effortless the job of sending the vehicle for service. Servicing a used vehicle would require a lot more effort compared to a brand-new vehicle. With this proposed solution, Malaysian who purchased used vehicle would no longer need to spend time on picking their workshop or making appointment.

### A. Background Study

The automotive industry is a broad sector that includes various complicated processes such as manufacturing, trading, and servicing of motor vehicles – the transportation tools to assist users travelling from one place to another easier and faster (Predictive Analytics Today, 2021). As time goes by, the industry received intense attention and grew tremendously globally due to it revolutionizing the traditional travelling method which was once energy and time-consuming.

Based on Müller's (2021) statements, Malaysia's automotive industry has always been thriving because it holds the third ranking among all ASEAN countries for a few years. For the past two years, although the negative impact of Covid-19 on automotive sales for the past two years is unarguably huge, the industry has slowly resumed to normal since the ease of MCO in August 2021. Automotive-related business activities such as car manufacturing, washing, and trading are allowed (Tan, 2021). Following the government's vital announcement, automakers pace up the vehicle manufacturing process to serve the expected spiking market demand. It is predicted that the demand for automobiles will increase rapidly due to the restriction being lifted. Additionally, Datuk Aishah Ahmad, the chairman of the Malaysian Automotive Association (MAA) mentioned that the authority promotes

incentive plans consistently to encourage the national automotive market's growth (The Malaysian Reserve, 2021).

Currently, the main communication methods are via phone calling or text messaging (Mohd Sam & Farhana Baharin, 2018). The manual communication methods may cause many issues including inconvenience, time wastage, inefficiency issues and more.

Besides, the customer does not have an appropriate way to keep track of the service histories. It could be a problem when he or she wishes to trace back the services that had been completed in the past. Therefore, the proposed solution – Autopia: Automotive Care System is aimed to provide helpful functionalities to the targeted users – automotive mechanic workshops and automobile owners to overcome the gap identified. It is expected to allow the automobile owners to book appointments, view vehicle service histories, communicate with mechanic workshops and more. Therefore, the system platform will be deployed on Android as a mobile application for both user groups mentioned.

### B. Problem Statement

Majority of the automotive mechanic workshops in Malaysia are still practicing manual appointments booking systems such as via phone calls, messages, or walk-ins (Mohd Sam & Farhana Baharin, 2018). Some problems posed by manual booking systems are unpredictability, a time constraint for appointment placement, long waiting time and queue, and lack of service history records. A car workshop owner had reflected the situation of being unable to predict the number of customers visiting daily (Selan, 2019). There are times when the shop is full of customers, yet it might be empty during other days. This may result in losses when there is sudden traffic of customers as the mechanics might not be able to cater for the spiking demand, forcing them to turn the customers away (Qtrac, n.d.). Turning customers away not only causes an impact on the business' sales and revenues, yet it might increase the perceived waiting time of the respective mechanic workshop among its customers (Worlitz et al., 2020). The manual booking systems are unable to assist unpredictability issue. Furthermore, a manual booking system via telephone calls also creates a barrier for the consumers to place appointments due to time constraints.

The consequence of long queues is the reduction of customer satisfaction as much as 50% (Tšernov, n.d.). Besides, considering the current situation of the Covid-19 pandemic, queuing is one of the “taboos” as it raises the risk of cross-contagion and difficulty to retain physical distancing (Perlman & Yechiali, 2020). Additionally, the manual booking system is also unhelpful in keeping track of service history records. Over 77% of respondents in research stated that service records are usually not well-retained when they visited automotive workshops (Abdul Wahab et al., 2017).

Neither of the manual booking methods could store the information about appointment or services' details properly into a database, which creates difficulties for the vehicle owners to trace the records in the future.

### C. Rationale

The result of this project is an automotive care system that assists vehicle owners to manage their vehicle maintenance tasks with automotive mechanic workshops. It is mainly built to improve the problems of low awareness and willingness among Malaysians to perform regular vehicle maintenance and the downsides of using the manual appointment booking system. With the reminder and notification function provided by the proposed system to inform the users about the upcoming maintenance dates, it could be helpful to increase the awareness and willingness of vehicle owners to perform regular vehicle maintenance.

It reduces the hassle where vehicle owners no longer need to place appointments using manual methods such as phone calling and messaging. The users can browse through all the mechanic workshops available on the system then decide on which workshop that they would prefer to place appointments. Digitalizing appointment booking prevents users from issues including frustrating long queues, lack of service records, and time constraints to place appointments as mentioned in Problem Context. Besides, the system will be providing a rating and review system for users with experience to provide comments about the automotive mechanic workshops.

### D. Potential Benefits

Through the preparation of the proposed system, two types of potential benefits are obtained, which are tangible benefits and intangible benefits. Below are the lists of tangible benefits and intangible benefits.

#### 1) Tangible Benefits

The tangible benefits that are obtained throughout the preparation of the proposed system are as follows:

- Helps vehicle owners to reduce high repair costs in the event of vehicle breakdown due to long term poor vehicle care.
- Reduces the costs of paper works for automotive service history records purposes.
- Helps vehicle owners to estimate the expenditures by receiving quotations from the automotive mechanic workshops before proceeding to perform the actual services.
- Allows automobile owners to keep track of their vehicle service expenses to avoid unnecessary overspending.
- Increases revenue for the local mechanic workshops by attracting more customers through the system's ability to provide more exposure and visibility.
- Retains a vehicle's value for future reselling purposes as Richardson (2009) mentioned vehicles with complete service history are usually sold second handed at a higher price.

#### 2) Intangible Benefits

- Increases the automotive mechanic workshops' productivity in managing customers' appointment requests.

- Raises awareness and willingness among Malaysian automobile owners regarding the importance of regular vehicle maintenance.

- Enhances the customer experience in terms of the reduced queue waiting time and more flexibility in appointment placement time selection.

- Allows users to review past vehicle service history records as an insight.

- Enhances user experience and satisfaction about the appointment booking procedure with the automotive mechanic workshops.

### E. Aim

To build a system that includes automotive mechanic workshops and vehicle owners in facilitating automotive care services via a digital appointment booking system.

### F. Objectives

- To investigate the current issues of the automotive industry in Malaysia, especially regarding the manual appointment booking system that is widely adopted by the automotive mechanic workshops.
- To raise the awareness and willingness among Malaysian vehicle owners to perform regular vehicle maintenance through the adoption of service reminder notifications in the system proposed.
- To develop a system that facilitates digitalized service appointment booking between the vehicle owners and automotive mechanic workshops.
- To build a system that lets users to record automobile service history details to keep track of their vehicles' information.
- To develop a system that assists the local automotive mechanic workshops in gaining more exposure
- To evaluate different scenarios in registering medical appointments of the system.

### G. Deliverables

This project is expected to produce an automotive care system that intends to facilitate the appointment booking processes for vehicle servicing between vehicle owners and automotive mechanic workshops. The system may be focusing on functions such as service appointment booking, service reminder notifications, service history tracker and more. It is mainly to bridge the gap that currently, the local automotive workshops are still utilizing manual booking ways such as phone calling or text messaging to place appointments with the workshops.

### H. Nature of Challenges

First and foremost, developing the system is one of the toughest challenges the developer will face. Throughout the process of development, the developer must conduct careful programming and debugging tasks to ensure that the system is functioning with minimal to zero issues. An underdeveloped system tends to have uncountable logical issues; thus, it impacts the user experience negatively.

Besides that, the anticipated challenge is the possible difficulty to obtain positive user involvement and engagement during the data-gathering phase. Based on the title proposed,

this project requires user participation from target groups such as automobile owners and automotive mechanic workshops' representatives. The main reason collecting user input for the investigation is considered challenging is because the participation is fully voluntary based

Moreover, selection of system development environment, programming languages, frameworks, application programming interfaces (APIs), and databases is challenging as they directly impact the system quality.

Finally, the time constraint imposed for this project's development phase is relatively limited and rigid. Four months period is allocated for the project's system development. Based on Moazed's (n.d.) arguments, this time frame is considered sufficient to develop a minimal viable product (MVP) under the assumption that there are a few developers assigned into both front-end and back-end teams.

## II. SYSTEM ARCHITECTURE

### A. System Design

#### 1) System Architectural Diagram

The backend system architecture is designed to utilize a mixed approach including REST API built using AWS API Gateway, Lambda, S3, RDS, with ASP .Net Core (C# and Entity Framework Core) stack, and Firebase services from Google Cloud. The reason to mix such use cases was due to REST API had been proven to be poor at real time use cases like chat, which is needed for the project.

Based on the diagram in Fig 1., once users initiate any data fetching requests from the client-side mobile application, the request will pass through OkHttp and Retrofit libraries which are responsible to process the API interactions into simpler form – Plain Old Java Objects (POJOs) to prevent the developers from writing complicated low-level codes. The processes utilized the HTTPS REST architecture.

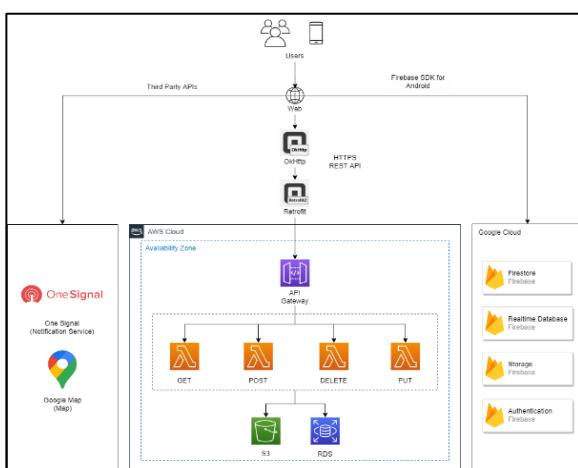


Fig. 1. System architectural diagram

Besides, a few third-party APIs had also been integrated to the project's structure. The first API is OneSignal, which is used heavily in this project to support notification services. It plays an important role due to Autopia's nature for being a CRM tool to remind customers about the maintenance schedules, as well as other activities like an appointment request's status updates, new messages from other users and more. The main reason to select OneSignal is due to it is free, which contributes favourability to the project's budgets management. Besides, OneSignal is a professional notification services API; therefore, the integration with

Android mobile application is smooth and requires lesser codes.

#### 2) Client Mobile Application Architectural Design

The client mobile application is built using Android Studio in Kotlin language. Instead of employing Activity-Based architecture where majority of the screens are using Activity from Android, this project is adopting Single-Activity Architecture with Fragments. Activity-based architecture is an approach that tends to be practiced by Android developers earlier; however, it suffered from navigational issues. For example, users will witness total screen swapping every time they press to navigate to a new screen, which does not present a smooth navigational flow.

Worse, it tends to suffer from transition animation issues such as unusual blinking will be seen on the action bar upon navigation (Ghosh, n.d.). Another major constraint in Activity-Based Architecture is data retention and passing problems, as switching to a new activity depicts that the data is not retained within a similar scope anymore. Sabag (2018) said that the data passing between activities might require data object types to be Parcelable, a more error-prone method, and the assistance from Services and Content Providers classes. Therefore, it increased the risk and level of inconvenience.

#### 3) User Case Diagram

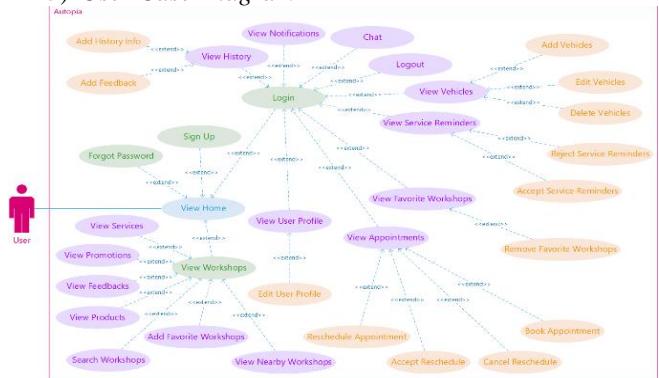


Fig. 2. Use case Diagram (User/Vehicle Owner)

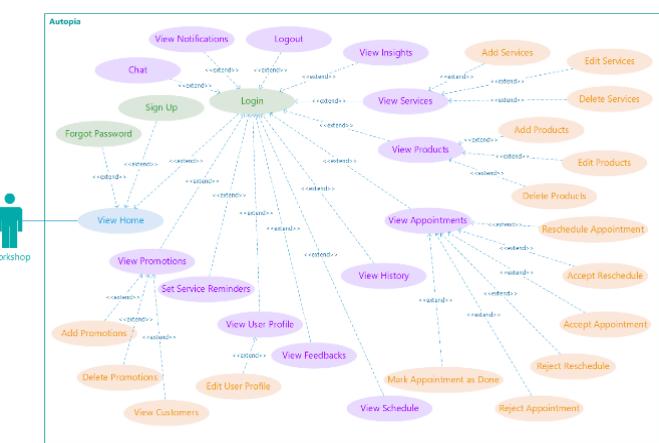


Fig. 3. Use Case Diagram (Workshop)

## III. INTERFACE DESIGN

Interface design is a crucial chapter where an application's appearance is sketched with all the important components and features considered. To adhere to the design principles and guidelines, the developer had considered the six design principles from Donald Norman's book – The Design of

Everyday Things (1998). The six principles proposed by Norman included visibility, feedback, mapping, affordance, constraints, and consistency. All principles could be witnessed in each of the screen designed. Before implementation the it is important for the software to be tested. The main ideology to test the software before being implemented is to test its reliability and quality. Discovering a lack in either one comportment would result on improvement.

#### A. Interface Login Screen

Login Screen is common for users and workshops to instruct the system in initiating a user account's session. The screen is designed to contain of email and password input fields for the users to enter valid credentials. A login button is provided for users to press after filling in the email and password fields to start the processes. Besides, a forgot password button is also designed to assist the users who forgot the original password and wish to retrieve the user account by resetting it. As for the users who have yet to register a user account on Autopia, the sign-up button is designed to navigate the user to the Sign-Up Screen. The interface is designed to be minimal with necessary elements only to prevent cognitive overload and confusion. Fig 4. shows the Interface login screen.

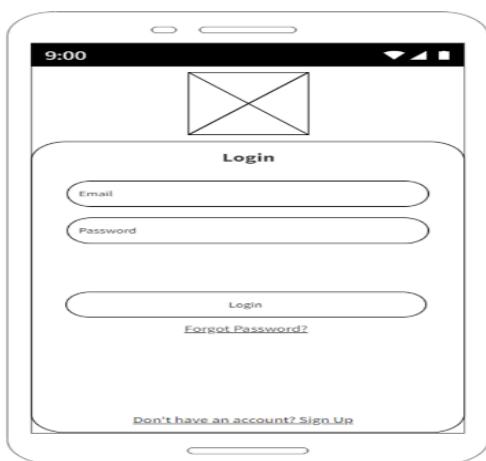


Fig. 4. Interface login screen

#### B. Interface for Home Screen

Home Screen is designed to show the most important information of Autopia such as the workshops available on the platform. This is to display the most important information at the most prominent space so that the users can view them easily. The screen is also designed to be equipped with top and bottom navigation bars. On the top navigation bar, a hamburger menu icon is showed which will trigger the opening of a navigation drawer. Besides, the screen name will also be displayed. On the bottom navigation bar, the design contains of important items for navigation such as "Home", "Appointment", "History", "Chat", and "Profile", where "Home" is the default selection. Upon clicking on each bottom navigation item, the system will navigate the users to the corresponding screen. Fig 5. shows the Interface for homescreen.

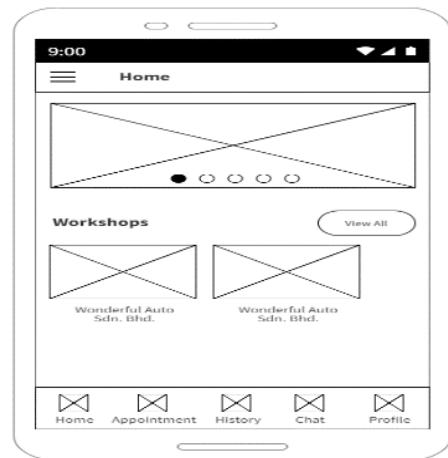


Fig. 5. Interface for homescreen

#### C. Interface for Workshop Screen

Workshop Screen shows the details of a workshop upon users clicking on the workshop card from Home Screen or Workshops Screen. The basic information of the selected workshop such as the profile image, information, services offered and more will be showed to the users clearly. A tab navigation bar is included in this screen to allow the users to swap between screens according to the information that they wish to browse. Essentially, there are three icons located above the name of the workshop. The icons represent Chat, Heart, and Appointment functions respectively. Upon clicking on these icons, the system performs certain functions or navigates users to the corresponding screens. Fig 6. shows the Interface for workshop screen implementation.

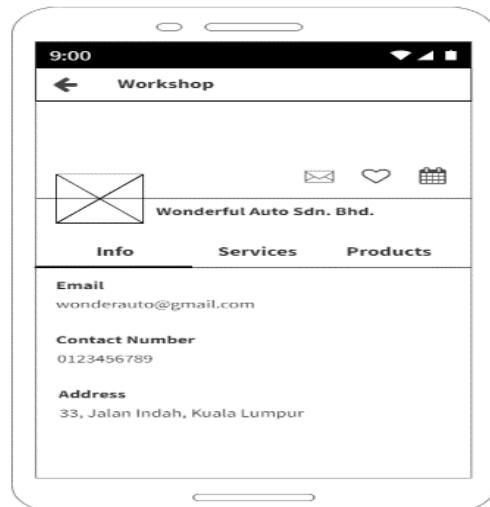


Fig. 6. Interface for Workshop Screen Implementation

## IV. IMPLEMENTATION

#### A. Screenshot of Homepage

In this screen, there is a carousel showing all the latest events in images. Users can swipe or press on the indicator dots to navigate between the images. When user is logged in, the system will greet the user based on username. Following greetings, the Workshops section displays all the workshops registered on Autopia. Clicking the images of each workshop will lead user to each workshop's info screen. Fig 7. shows the screenshot of homepage.

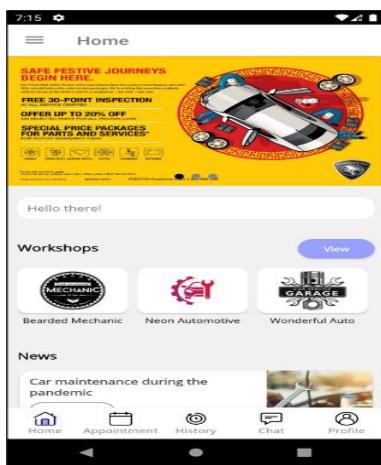


Fig. 7. Screenshot of homepage

#### B. Screenshot for Login

The users are required to fill in email and password with accurate credentials for the system to generate user session. If the inputs are invalid, a toast message will be displayed to notify the users that the login process had failed. Fig 8. shows the screenshot of login.

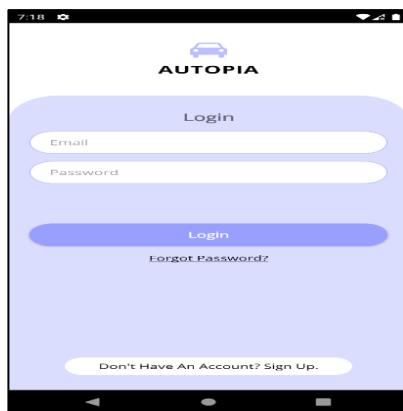


Fig. 8. Screenshot for login

#### C. Screenshot for Search Workshop

Workshops screen shows all the available workshops registered on Autopia with their name, description, address, and logo shown. Fig 9. shows the screenshot of search workshop.



Fig. 9. Screenshot for search workshop

Upon clicking on each of these workshop cards, the users will be navigated to the specific workshop's details screen. Apart from the main content, the screen also facilitates search by keyword and search by location.

#### D. Screenshot for Workshop information

Workshop Info screen is the detail screen when user selects a workshop. It shows all the information regarding the workshop, including its general information, services, products, and promotions offered, and feedbacks that had been submitted by other clients. In this screen, signed in users may also proceed to request for an appointment, chat, or add the workshop to their favourite list. Fig 10. shows the screenshot of for workshop information.

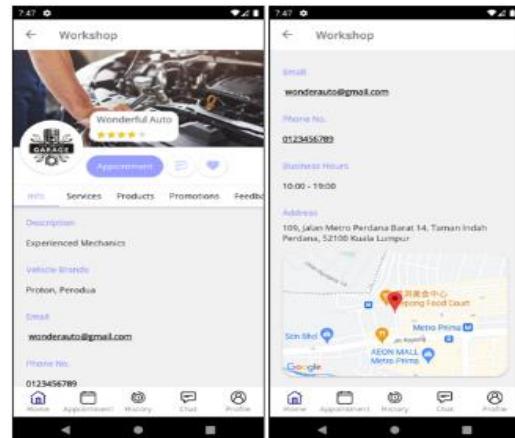


Fig. 10. Screenshot for workshop information

#### E. Screenshot for Appointment Booking

This screen shows all the information, including workshop's basic information, service requested, vehicle selected, client's name, contact number, quotation, description, appointment date and time. It is to allow users to make confirmation before they officially request for an appointment. Once all the information had been confirmed, the user may proceed to press the "Request Appointment" button. Fig 11. shows the screenshot for appointment booking.



Fig. 11. Screenshot for appointment booking

#### F. Screenshot for Hospital Staff Dashboard Page

All the appointment-related screens from the workshop's point of view. However, since the functionalities are similar to the clients' features and they had been explained before, this section will only show the screenshots to prevent duplication. Fig 12. shows the screenshot for hospital staff dashboard page.

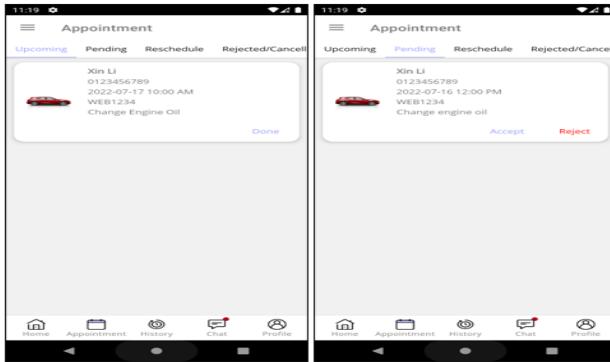


Fig. 12. Screenshot for hospital staff dashboard page

#### G. Screenshot of Feedback (Workshop)

Feedbacks screen displays all the previous feedbacks that had been given by the clients to the workshops. It shows the client's profile image, name, ratings, and comments. Upon clicking on the feedback card, workshops will be navigated to the Feedback Details Screen with all the client inputs filled for the workshops to view clearly. Fig 13. shows the screenshot of feedback (workshop).

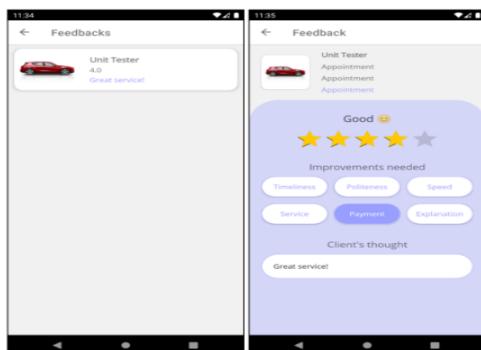


Fig. 13. Screenshot of feedback (workshop)

#### H. Screenshot of Notification

Autopia is also equipped with the ability of sending and receiving notifications upon activities like chatting, appointment requesting or status changing, new promotion events and more. Fig 14. shows the screenshot of notification.

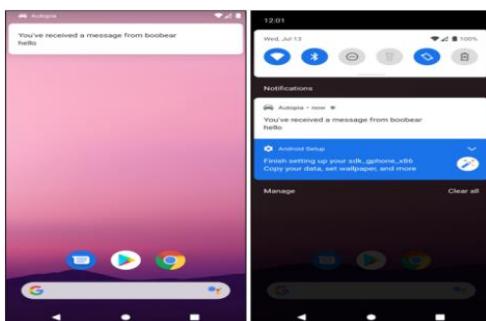


Fig. 14. Screenshot of notification

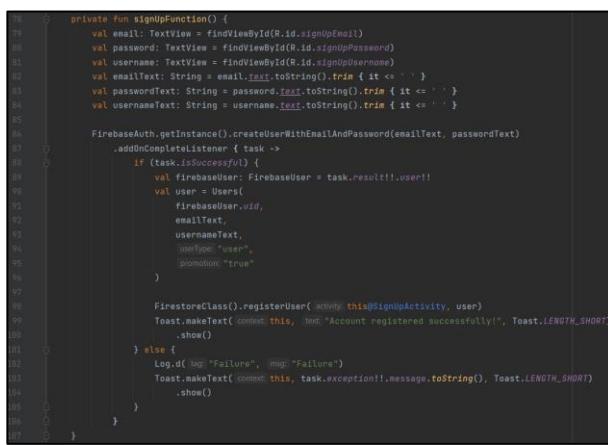
#### V. SAMPLE CODE

##### A. Smaple Code written for Login

The login function utilizes the Firebase Authentication sign-in method which takes in the user's email and password. This function will be launched when users press the login button on Login Screen, and it will only be accessible when both email and password input fields had been filled. Line 68 creates the instance of Firebase Authentication and then calls the pre-made sign-in method by Firebase which takes in email and password. Firebase will then use the email and password for validation purposes and return a task result upon completion. The developer then used the returned task to check if it is successful. If the task is successful, the system should show a toast message to inform the user that he or she had been signed in successfully. Then, the following codes are to check whether the user is a client or workshop based on the data returned by Firebase Firestore using the user's ID. Firebase Firestore is the database used to store user-related information. This checking is to ensure the system navigates the user to the correct home screen. Besides, line 89 registers the user with One Signal, the third-party API used in this project to support the notification feature. Upon user sign-in, it is crucial to update the user-ID with the API to notify it that the user session is active and should be receiving any notifications if applicable. Fig 15. shows sample code written for login.

```

43     private fun loginFunction() {
44         val email: TextView = findViewById(R.id.loginEmail)
45         val password: TextView = findViewById(R.id.loginPassword)
46         val emailText: String = email.editText.toString().trim { it <= ' ' }
47         val passwordText: String = password.text.toString().trim { it <= ' ' }
48
49         FirebaseAuth.getInstance().signInWithEmailAndPassword(emailText, passwordText)
50             .addOnCompleteListener { task: Task<AuthResult>? -->
51                 if (task.isSuccessful) {
52                     task.result?.let { showToast(it, "Login successful!", Toast.LENGTH_SHORT) }
53
54                     val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
55                     if (user != null) {
56                         FirebaseFirestore.getInstance().collection(Constants.Users)
57                             .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
58                             if (document != null) {
59                                 if (document.get("type").toString() == "workshop") {
60                                     if (document.get("name").toString() == "Autopia") {
61                                         if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
62                                             val intent = Intent(
63                                                 packageContext, this@LoginActivity,
64                                                 WorkshopInfoActivity::class.java
65                                             )
66                                             startActivity(intent)
67                                         } else {
68                                             OneSignalUser.setUser(user.uid)
69                                             val intent = Intent(
70                                                 packageContext, this@LoginActivity,
71                                                 WorkshopNavigationDrawerActivity::class.java
72                                             )
73                                             startActivity(intent)
74                                         }
75                                     }
76                                 }
77                             }
78                         }
79                     }
80                 }
81             }
82         }
83     }
84
85     private fun showToast(text: String, title: String, duration: Int) {
86         val toast = Toast.makeText(packageContext, text, duration)
87         toast.setTitle(title)
88         toast.show()
89     }
90
91     private fun registerUser() {
92         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
93         if (user != null) {
94             OneSignal.setUserId(user.uid)
95             val intent = Intent(
96                 packageContext, this@LoginActivity,
97                 WorkshopNavigationDrawerActivity::class.java
98             )
99             startActivity(intent)
100         }
101     }
102
103     private fun showToast(text: String, title: String, duration: Int) {
104         val toast = Toast.makeText(packageContext, text, duration)
105         toast.setTitle(title)
106         toast.show()
107     }
108
109     private fun checkUserType() {
110         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
111         if (user != null) {
112             FirebaseFirestore.getInstance().collection(Constants.Users)
113                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
114                     if (document != null) {
115                         if (document.get("type").toString() == "workshop") {
116                             if (document.get("name").toString() == "Autopia") {
117                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
118                                     val intent = Intent(
119                                         packageContext, this@LoginActivity,
120                                         WorkshopInfoActivity::class.java
121                                     )
122                                     startActivity(intent)
123                                 }
124                             }
125                         }
126                     }
127                 }
128             }
129         }
130     }
131
132     private fun checkUserType() {
133         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
134         if (user != null) {
135             FirebaseFirestore.getInstance().collection(Constants.Users)
136                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
137                     if (document != null) {
138                         if (document.get("type").toString() == "client") {
139                             if (document.get("name").toString() == "Autopia") {
140                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
141                                     val intent = Intent(
142                                         packageContext, this@LoginActivity,
143                                         ClientInfoActivity::class.java
144                                     )
145                                     startActivity(intent)
146                                 }
147                             }
148                         }
149                     }
150                 }
151             }
152         }
153     }
154
155     private fun checkUserType() {
156         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
157         if (user != null) {
158             FirebaseFirestore.getInstance().collection(Constants.Users)
159                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
160                     if (document != null) {
161                         if (document.get("type").toString() == "client") {
162                             if (document.get("name").toString() == "Autopia") {
163                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
164                                     val intent = Intent(
165                                         packageContext, this@LoginActivity,
166                                         ClientInfoActivity::class.java
167                                     )
168                                     startActivity(intent)
169                                 }
170                             }
171                         }
172                     }
173                 }
174             }
175         }
176     }
177
178     private fun checkUserType() {
179         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
180         if (user != null) {
181             FirebaseFirestore.getInstance().collection(Constants.Users)
182                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
183                     if (document != null) {
184                         if (document.get("type").toString() == "client") {
185                             if (document.get("name").toString() == "Autopia") {
186                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
187                                     val intent = Intent(
188                                         packageContext, this@LoginActivity,
189                                         ClientInfoActivity::class.java
190                                     )
191                                     startActivity(intent)
192                                 }
193                             }
194                         }
195                     }
196                 }
197             }
198         }
199     }
200
201     private fun checkUserType() {
202         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
203         if (user != null) {
204             FirebaseFirestore.getInstance().collection(Constants.Users)
205                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
206                     if (document != null) {
207                         if (document.get("type").toString() == "client") {
208                             if (document.get("name").toString() == "Autopia") {
209                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
210                                     val intent = Intent(
211                                         packageContext, this@LoginActivity,
212                                         ClientInfoActivity::class.java
213                                     )
214                                     startActivity(intent)
215                                 }
216                             }
217                         }
218                     }
219                 }
220             }
221         }
222     }
223
224     private fun checkUserType() {
225         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
226         if (user != null) {
227             FirebaseFirestore.getInstance().collection(Constants.Users)
228                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
229                     if (document != null) {
230                         if (document.get("type").toString() == "client") {
231                             if (document.get("name").toString() == "Autopia") {
232                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
233                                     val intent = Intent(
234                                         packageContext, this@LoginActivity,
235                                         ClientInfoActivity::class.java
236                                     )
237                                     startActivity(intent)
238                                 }
239                             }
240                         }
241                     }
242                 }
243             }
244         }
245     }
246
247     private fun checkUserType() {
248         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
249         if (user != null) {
250             FirebaseFirestore.getInstance().collection(Constants.Users)
251                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
252                     if (document != null) {
253                         if (document.get("type").toString() == "client") {
254                             if (document.get("name").toString() == "Autopia") {
255                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
256                                     val intent = Intent(
257                                         packageContext, this@LoginActivity,
258                                         ClientInfoActivity::class.java
259                                     )
260                                     startActivity(intent)
261                                 }
262                             }
263                         }
264                     }
265                 }
266             }
267         }
268     }
269
270     private fun checkUserType() {
271         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
272         if (user != null) {
273             FirebaseFirestore.getInstance().collection(Constants.Users)
274                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
275                     if (document != null) {
276                         if (document.get("type").toString() == "client") {
277                             if (document.get("name").toString() == "Autopia") {
278                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
279                                     val intent = Intent(
280                                         packageContext, this@LoginActivity,
281                                         ClientInfoActivity::class.java
282                                     )
283                                     startActivity(intent)
284                                 }
285                             }
286                         }
287                     }
288                 }
289             }
290         }
291     }
292
293     private fun checkUserType() {
294         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
295         if (user != null) {
296             FirebaseFirestore.getInstance().collection(Constants.Users)
297                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
298                     if (document != null) {
299                         if (document.get("type").toString() == "client") {
300                             if (document.get("name").toString() == "Autopia") {
301                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
302                                     val intent = Intent(
303                                         packageContext, this@LoginActivity,
304                                         ClientInfoActivity::class.java
305                                     )
306                                     startActivity(intent)
307                                 }
308                             }
309                         }
310                     }
311                 }
312             }
313         }
314     }
315
316     private fun checkUserType() {
317         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
318         if (user != null) {
319             FirebaseFirestore.getInstance().collection(Constants.Users)
320                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
321                     if (document != null) {
322                         if (document.get("type").toString() == "client") {
323                             if (document.get("name").toString() == "Autopia") {
324                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
325                                     val intent = Intent(
326                                         packageContext, this@LoginActivity,
327                                         ClientInfoActivity::class.java
328                                     )
329                                     startActivity(intent)
330                                 }
331                             }
332                         }
333                     }
334                 }
335             }
336         }
337     }
338
339     private fun checkUserType() {
340         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
341         if (user != null) {
342             FirebaseFirestore.getInstance().collection(Constants.Users)
343                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
344                     if (document != null) {
345                         if (document.get("type").toString() == "client") {
346                             if (document.get("name").toString() == "Autopia") {
347                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
348                                     val intent = Intent(
349                                         packageContext, this@LoginActivity,
350                                         ClientInfoActivity::class.java
351                                     )
352                                     startActivity(intent)
353                                 }
354                             }
355                         }
356                     }
357                 }
358             }
359         }
360     }
361
362     private fun checkUserType() {
363         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
364         if (user != null) {
365             FirebaseFirestore.getInstance().collection(Constants.Users)
366                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
367                     if (document != null) {
368                         if (document.get("type").toString() == "client") {
369                             if (document.get("name").toString() == "Autopia") {
370                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
371                                     val intent = Intent(
372                                         packageContext, this@LoginActivity,
373                                         ClientInfoActivity::class.java
374                                     )
375                                     startActivity(intent)
376                                 }
377                             }
378                         }
379                     }
380                 }
381             }
382         }
383     }
384
385     private fun checkUserType() {
386         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
387         if (user != null) {
388             FirebaseFirestore.getInstance().collection(Constants.Users)
389                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
390                     if (document != null) {
391                         if (document.get("type").toString() == "client") {
392                             if (document.get("name").toString() == "Autopia") {
393                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
394                                     val intent = Intent(
395                                         packageContext, this@LoginActivity,
396                                         ClientInfoActivity::class.java
397                                     )
398                                     startActivity(intent)
399                                 }
400                             }
401                         }
402                     }
403                 }
404             }
405         }
406     }
407
408     private fun checkUserType() {
409         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
410         if (user != null) {
411             FirebaseFirestore.getInstance().collection(Constants.Users)
412                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
413                     if (document != null) {
414                         if (document.get("type").toString() == "client") {
415                             if (document.get("name").toString() == "Autopia") {
416                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
417                                     val intent = Intent(
418                                         packageContext, this@LoginActivity,
419                                         ClientInfoActivity::class.java
420                                     )
421                                     startActivity(intent)
422                                 }
423                             }
424                         }
425                     }
426                 }
427             }
428         }
429     }
430
431     private fun checkUserType() {
432         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
433         if (user != null) {
434             FirebaseFirestore.getInstance().collection(Constants.Users)
435                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
436                     if (document != null) {
437                         if (document.get("type").toString() == "client") {
438                             if (document.get("name").toString() == "Autopia") {
439                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
440                                     val intent = Intent(
441                                         packageContext, this@LoginActivity,
442                                         ClientInfoActivity::class.java
443                                     )
444                                     startActivity(intent)
445                                 }
446                             }
447                         }
448                     }
449                 }
450             }
451         }
452     }
453
454     private fun checkUserType() {
455         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
456         if (user != null) {
457             FirebaseFirestore.getInstance().collection(Constants.Users)
458                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
459                     if (document != null) {
460                         if (document.get("type").toString() == "client") {
461                             if (document.get("name").toString() == "Autopia") {
462                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
463                                     val intent = Intent(
464                                         packageContext, this@LoginActivity,
465                                         ClientInfoActivity::class.java
466                                     )
467                                     startActivity(intent)
468                                 }
469                             }
470                         }
471                     }
472                 }
473             }
474         }
475     }
476
477     private fun checkUserType() {
478         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
479         if (user != null) {
480             FirebaseFirestore.getInstance().collection(Constants.Users)
481                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
482                     if (document != null) {
483                         if (document.get("type").toString() == "client") {
484                             if (document.get("name").toString() == "Autopia") {
485                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
486                                     val intent = Intent(
487                                         packageContext, this@LoginActivity,
488                                         ClientInfoActivity::class.java
489                                     )
490                                     startActivity(intent)
491                                 }
492                             }
493                         }
494                     }
495                 }
496             }
497         }
498     }
499
500     private fun checkUserType() {
501         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
502         if (user != null) {
503             FirebaseFirestore.getInstance().collection(Constants.Users)
504                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
505                     if (document != null) {
506                         if (document.get("type").toString() == "client") {
507                             if (document.get("name").toString() == "Autopia") {
508                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
509                                     val intent = Intent(
510                                         packageContext, this@LoginActivity,
511                                         ClientInfoActivity::class.java
512                                     )
513                                     startActivity(intent)
514                                 }
515                             }
516                         }
517                     }
518                 }
519             }
520         }
521     }
522
523     private fun checkUserType() {
524         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
525         if (user != null) {
526             FirebaseFirestore.getInstance().collection(Constants.Users)
527                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
528                     if (document != null) {
529                         if (document.get("type").toString() == "client") {
530                             if (document.get("name").toString() == "Autopia") {
531                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
532                                     val intent = Intent(
533                                         packageContext, this@LoginActivity,
534                                         ClientInfoActivity::class.java
535                                     )
536                                     startActivity(intent)
537                                 }
538                             }
539                         }
540                     }
541                 }
542             }
543         }
544     }
545
546     private fun checkUserType() {
547         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
548         if (user != null) {
549             FirebaseFirestore.getInstance().collection(Constants.Users)
550                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
551                     if (document != null) {
552                         if (document.get("type").toString() == "client") {
553                             if (document.get("name").toString() == "Autopia") {
554                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
555                                     val intent = Intent(
556                                         packageContext, this@LoginActivity,
557                                         ClientInfoActivity::class.java
558                                     )
559                                     startActivity(intent)
560                                 }
561                             }
562                         }
563                     }
564                 }
565             }
566         }
567     }
568
569     private fun checkUserType() {
570         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
571         if (user != null) {
572             FirebaseFirestore.getInstance().collection(Constants.Users)
573                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
574                     if (document != null) {
575                         if (document.get("type").toString() == "client") {
576                             if (document.get("name").toString() == "Autopia") {
577                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
578                                     val intent = Intent(
579                                         packageContext, this@LoginActivity,
580                                         ClientInfoActivity::class.java
581                                     )
582                                     startActivity(intent)
583                                 }
584                             }
585                         }
586                     }
587                 }
588             }
589         }
590     }
591
592     private fun checkUserType() {
593         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
594         if (user != null) {
595             FirebaseFirestore.getInstance().collection(Constants.Users)
596                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
597                     if (document != null) {
598                         if (document.get("type").toString() == "client") {
599                             if (document.get("name").toString() == "Autopia") {
600                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
601                                     val intent = Intent(
602                                         packageContext, this@LoginActivity,
603                                         ClientInfoActivity::class.java
604                                     )
605                                     startActivity(intent)
606                                 }
607                             }
608                         }
609                     }
610                 }
611             }
612         }
613     }
614
615     private fun checkUserType() {
616         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
617         if (user != null) {
618             FirebaseFirestore.getInstance().collection(Constants.Users)
619                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
620                     if (document != null) {
621                         if (document.get("type").toString() == "client") {
622                             if (document.get("name").toString() == "Autopia") {
623                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
624                                     val intent = Intent(
625                                         packageContext, this@LoginActivity,
626                                         ClientInfoActivity::class.java
627                                     )
628                                     startActivity(intent)
629                                 }
630                             }
631                         }
632                     }
633                 }
634             }
635         }
636     }
637
638     private fun checkUserType() {
639         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
640         if (user != null) {
641             FirebaseFirestore.getInstance().collection(Constants.Users)
642                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
643                     if (document != null) {
644                         if (document.get("type").toString() == "client") {
645                             if (document.get("name").toString() == "Autopia") {
646                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
647                                     val intent = Intent(
648                                         packageContext, this@LoginActivity,
649                                         ClientInfoActivity::class.java
650                                     )
651                                     startActivity(intent)
652                                 }
653                             }
654                         }
655                     }
656                 }
657             }
658         }
659     }
660
661     private fun checkUserType() {
662         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
663         if (user != null) {
664             FirebaseFirestore.getInstance().collection(Constants.Users)
665                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
666                     if (document != null) {
667                         if (document.get("type").toString() == "client") {
668                             if (document.get("name").toString() == "Autopia") {
669                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
670                                     val intent = Intent(
671                                         packageContext, this@LoginActivity,
672                                         ClientInfoActivity::class.java
673                                     )
674                                     startActivity(intent)
675                                 }
676                             }
677                         }
678                     }
679                 }
680             }
681         }
682     }
683
684     private fun checkUserType() {
685         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
686         if (user != null) {
687             FirebaseFirestore.getInstance().collection(Constants.Users)
688                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
689                     if (document != null) {
690                         if (document.get("type").toString() == "client") {
691                             if (document.get("name").toString() == "Autopia") {
692                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
693                                     val intent = Intent(
694                                         packageContext, this@LoginActivity,
695                                         ClientInfoActivity::class.java
696                                     )
697                                     startActivity(intent)
698                                 }
699                             }
700                         }
701                     }
702                 }
703             }
704         }
705     }
706
707     private fun checkUserType() {
708         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
709         if (user != null) {
710             FirebaseFirestore.getInstance().collection(Constants.Users)
711                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
712                     if (document != null) {
713                         if (document.get("type").toString() == "client") {
714                             if (document.get("name").toString() == "Autopia") {
715                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
716                                     val intent = Intent(
717                                         packageContext, this@LoginActivity,
718                                         ClientInfoActivity::class.java
719                                     )
720                                     startActivity(intent)
721                                 }
722                             }
723                         }
724                     }
725                 }
726             }
727         }
728     }
729
730     private fun checkUserType() {
731         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
732         if (user != null) {
733             FirebaseFirestore.getInstance().collection(Constants.Users)
734                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
735                     if (document != null) {
736                         if (document.get("type").toString() == "client") {
737                             if (document.get("name").toString() == "Autopia") {
738                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
739                                     val intent = Intent(
740                                         packageContext, this@LoginActivity,
741                                         ClientInfoActivity::class.java
742                                     )
743                                     startActivity(intent)
744                                 }
745                             }
746                         }
747                     }
748                 }
749             }
750         }
751     }
752
753     private fun checkUserType() {
754         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
755         if (user != null) {
756             FirebaseFirestore.getInstance().collection(Constants.Users)
757                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
758                     if (document != null) {
759                         if (document.get("type").toString() == "client") {
760                             if (document.get("name").toString() == "Autopia") {
761                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
762                                     val intent = Intent(
763                                         packageContext, this@LoginActivity,
764                                         ClientInfoActivity::class.java
765                                     )
766                                     startActivity(intent)
767                                 }
768                             }
769                         }
770                     }
771                 }
772             }
773         }
774     }
775
776     private fun checkUserType() {
777         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
778         if (user != null) {
779             FirebaseFirestore.getInstance().collection(Constants.Users)
780                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
781                     if (document != null) {
782                         if (document.get("type").toString() == "client") {
783                             if (document.get("name").toString() == "Autopia") {
784                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
785                                     val intent = Intent(
786                                         packageContext, this@LoginActivity,
787                                         ClientInfoActivity::class.java
788                                     )
789                                     startActivity(intent)
790                                 }
791                             }
792                         }
793                     }
794                 }
795             }
796         }
797     }
798
799     private fun checkUserType() {
800         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
801         if (user != null) {
802             FirebaseFirestore.getInstance().collection(Constants.Users)
803                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
804                     if (document != null) {
805                         if (document.get("type").toString() == "client") {
806                             if (document.get("name").toString() == "Autopia") {
807                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
808                                     val intent = Intent(
809                                         packageContext, this@LoginActivity,
810                                         ClientInfoActivity::class.java
811                                     )
812                                     startActivity(intent)
813                                 }
814                             }
815                         }
816                     }
817                 }
818             }
819         }
820     }
821
822     private fun checkUserType() {
823         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
824         if (user != null) {
825             FirebaseFirestore.getInstance().collection(Constants.Users)
826                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
827                     if (document != null) {
828                         if (document.get("type").toString() == "client") {
829                             if (document.get("name").toString() == "Autopia") {
830                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
831                                     val intent = Intent(
832                                         packageContext, this@LoginActivity,
833                                         ClientInfoActivity::class.java
834                                     )
835                                     startActivity(intent)
836                                 }
837                             }
838                         }
839                     }
840                 }
841             }
842         }
843     }
844
845     private fun checkUserType() {
846         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
847         if (user != null) {
848             FirebaseFirestore.getInstance().collection(Constants.Users)
849                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
850                     if (document != null) {
851                         if (document.get("type").toString() == "client") {
852                             if (document.get("name").toString() == "Autopia") {
853                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
854                                     val intent = Intent(
855                                         packageContext, this@LoginActivity,
856                                         ClientInfoActivity::class.java
857                                     )
858                                     startActivity(intent)
859                                 }
860                             }
861                         }
862                     }
863                 }
864             }
865         }
866     }
867
868     private fun checkUserType() {
869         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
870         if (user != null) {
871             FirebaseFirestore.getInstance().collection(Constants.Users)
872                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
873                     if (document != null) {
874                         if (document.get("type").toString() == "client") {
875                             if (document.get("name").toString() == "Autopia") {
876                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
877                                     val intent = Intent(
878                                         packageContext, this@LoginActivity,
879                                         ClientInfoActivity::class.java
880                                     )
881                                     startActivity(intent)
882                                 }
883                             }
884                         }
885                     }
886                 }
887             }
888         }
889     }
890
891     private fun checkUserType() {
892         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
893         if (user != null) {
894             FirebaseFirestore.getInstance().collection(Constants.Users)
895                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
896                     if (document != null) {
897                         if (document.get("type").toString() == "client") {
898                             if (document.get("name").toString() == "Autopia") {
899                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
900                                     val intent = Intent(
901                                         packageContext, this@LoginActivity,
902                                         ClientInfoActivity::class.java
903                                     )
904                                     startActivity(intent)
905                                 }
906                             }
907                         }
908                     }
909                 }
910             }
911         }
912     }
913
914     private fun checkUserType() {
915         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
916         if (user != null) {
917             FirebaseFirestore.getInstance().collection(Constants.Users)
918                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
919                     if (document != null) {
920                         if (document.get("type").toString() == "client") {
921                             if (document.get("name").toString() == "Autopia") {
922                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
923                                     val intent = Intent(
924                                         packageContext, this@LoginActivity,
925                                         ClientInfoActivity::class.java
926                                     )
927                                     startActivity(intent)
928                                 }
929                             }
930                         }
931                     }
932                 }
933             }
934         }
935     }
936
937     private fun checkUserType() {
938         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
939         if (user != null) {
940             FirebaseFirestore.getInstance().collection(Constants.Users)
941                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
942                     if (document != null) {
943                         if (document.get("type").toString() == "client") {
944                             if (document.get("name").toString() == "Autopia") {
945                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
946                                     val intent = Intent(
947                                         packageContext, this@LoginActivity,
948                                         ClientInfoActivity::class.java
949                                     )
950                                     startActivity(intent)
951                                 }
952                             }
953                         }
954                     }
955                 }
956             }
957         }
958     }
959
960     private fun checkUserType() {
961         val user: FirebaseUser = FirebaseAuth.getInstance().currentUser
962         if (user != null) {
963             FirebaseFirestore.getInstance().collection(Constants.Users)
964                 .document(user.uid).addSnapshotListener { document, error: FirebaseFirestoreException? -->
965                     if (document != null) {
966                         if (document.get("type").toString() == "client") {
967                             if (document.get("name").toString() == "Autopia") {
968                                 if (document.get("address").toString() == "" || document.get("description").toString() == "" || document.get("contactNumber").toString() == "") {
969                                     val intent = Intent(
970                                         packageContext, this@LoginActivity,
971                                         ClientInfoActivity::class.java
972                                     )
973                                     startActivity(intent)
974                                 }
975
```



```

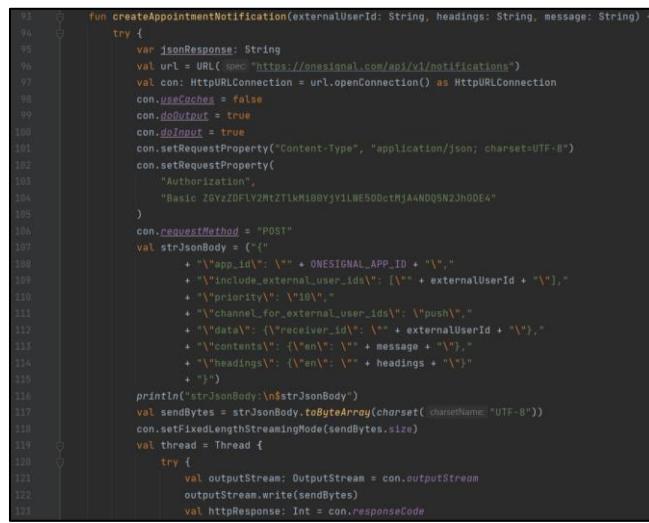
78     private fun signUpFunction() {
79         val email: TextView = findViewById(R.id.signupEmail)
80         val password: TextView = findViewById(R.id.signupPassword)
81         val username: TextView = findViewById(R.id.signupUsername)
82         val emailText: String = email.text.toString().trim { it <= ' ' }
83         val passwordText: String = password.text.toString().trim { it <= ' ' }
84         val usernameText: String = username.text.toString().trim { it <= ' ' }
85
86         FirebaseAuth.getInstance().createUserWithEmailAndPassword(emailText, passwordText)
87             .addOnCompleteListener { task ->
88                 if (task.isSuccessful) {
89                     val firebaseUser: FirebaseUser = task.result!!.user!!
90                     val user = Users(
91                         firebaseUser.uid,
92                         emailText,
93                         usernameText,
94                         userType = "user",
95                         promotion = "true"
96                     )
97
98                     FirestoreClass().registerUser(activity, this@SignUpActivity, user)
99                     Toast.makeText(context, "Account registered successfully!", Toast.LENGTH_SHORT)
100                     .show()
101                 } else {
102                     Log.e("Failure", "msg: " + task.exception)
103                     Toast.makeText(context, task.exception!!.message.toString(), Toast.LENGTH_SHORT)
104                     .show()
105                 }
106             }
107         }
108     }

```

Fig. 16. Sample code written for Sign Up

### C. Sample Codes Written for Notification

Notification is one of the essential components for appointment scheduling applications. This function will be called when any updates related to appointment matters happen. It takes in parameters like external user ID (the receiver's user ID), headings, and message of the notification. For safe coding, the developer wrapped the following codes in try and catch code block. Firstly, the system creates a string object which contains the URL link to One Signal's API. It then creates connection to the API with properties like headers, API credentials, request method, and its body. The body is appended as string with important information like the One Signal application ID, notification receiver IDs, contents, headings and more. Then, using the threading technique, the system executes the processes to send the POST HTTP request to the One Signal API without blocking the main UI thread. Afterwards, to facilitate the activity feed feature, the system is instructed to post the related notification information to the REST API hosted by the developer for display usages. Fig 17. shows sample codes written for notification.



```

93     fun createAppointmentNotification(externalUserId: String, headings: String, message: String) {
94         try {
95             var jsonResponse: String
96             val url = URL("https://onesignal.com/api/v1/notifications")
97             val con: HttpURLConnection = url.openConnection() as HttpURLConnection
98             con.useCaches = false
99             con.doOutput = true
100            con.doInput = true
101            con.setRequestProperty("Content-Type", "application/json; charset=UTF-8")
102            con.setRequestProperty(
103                "Authorization",
104                "Basic ZGVzZDFLYzttIKhB0YjYLWE50ctMjA4NDQ5N2JhODE4"
105            )
106            con.requestMethod = "POST"
107            val strJsonBody = ("{" +
108                + "\"app_id\": " + ONE_SIGNAL_APP_ID + "\",\n" +
109                + "\"include_external_user_ids\": [" + externalUserId + "],\n" +
110                + "\"priority\": \"100\","
111                + "\"channel_for_external_user_ids\": \"push\","
112                + "\"data\": {\"receiver_id\": " + externalUserId + "},"
113                + "\"contents\": {\"en\": " + message + "},"
114                + "\"headings\": {\"en\": " + headings + "}"
115                + "}")
116
117            println("strJsonBody: $strJsonBody")
118            val sendBytes = strJsonBody.toByteArray(charset("UTF-8"))
119            con.setFixedLengthStreamingMode(sendBytes.size)
120            val thread = Thread {
121                try {
122                    val outputStream: OutputStream = con.outputStream
123                    outputStream.write(sendBytes)
124                    val httpResponse: Int = con.responseCode
125                }
126            }
127        }
128    }

```

Fig. 17. Sample codes written for notification

## VI. CONCLUSION

### A. Critical Evaluation

Autopia could bring benefits to both workshops and vehicle owners. For workshops, this application mainly assists in helping the local automotive workshop owners to build and

maintain stronger customer relationships as it has features like chat, appointment management, and promotions. As mentioned in the Domain Research, in Malaysia, lack of proper platforms for workshops to communicate with customers is one of the major gaps this project is focusing on. Besides, the chat feature helps workshops to build stronger relationships with their clients as unique and personalized one-to-one experience could be gained by each client. The system is capable of broadcasting new promotion events uploaded by workshops to the clients who had turned on the receiving notification status.

### B. Conclusion & Reflection

The developer would conclude the system developed had met the intention of solving lack of communication and appointment booking platform between automotive workshops and vehicle owners. The core of Autopia at the end of development is the appointment module with functions like request, accept, reject, cancel, reschedule appointments, auto-detect no shows, and mark appointments as "done". These are basic functionalities required for an appointment booking application, which would make the digitalized processes more convenient for both sides. Besides, another objective – to build a platform which could store service histories had also been met as Autopia can track past or completed appointments. It helps users to track their histories easier than the most popular approach selected based on the questionnaire's responses – stickers on vehicles' windshields given by workshops. This is because storing the histories in database would eliminate risks of fading words on stickers or falling stickers.

Therefore, this is the most time-consuming part for both development and debugging processes. Other coding tasks like setting device's alarm, sending and receiving notifications for users with certain IDs specifically, restraining date and time picker, auto no-show detection, auto appointment time clashing detection and more are also some of the difficult parts of this project.

## REFERENCES

Predictive Analytics Today. (2021). All about automotive industry: Segments, value chain and competitive advantage. <https://www.predictiveanalyticstoday.com/what-is-automotive-industry-top-software-in-automotive-industry/>

Tan, D. (2021). Car factories can operate; showroom visits, car wash allowed for fully vaccinated in Phase 1 PPN states. Paul Tan. <https://paultan.org/2021/08/15/car-factories-can-operate-showroom-visits-car-wash-allowed-for-fully-vaccinated-in-phase-1-ppn-states/>

The Malaysian Reserve. (2021). Automotive market likely to rebound in 2021. <https://themalaysianreserve.com/2021/01/22/automotive-market-likely-to-rebound-in-2021/>

Mohd Sam, M., & Farhana Baharin, S. (2018). The factors which influencing users' behavioral intention towards using online booking system for car service at car service centre in Malacca. COJ Electronics and Communications, ISSN 2640-9739, Volume 1, Issue 4. <https://crimsonpublishers.com/cojec/fulltext/COJEC.000517.php>

Selan, S. (2021). Back street car mechanics just scraping by as drivers take the high road. Malaysia Now. <https://www.malaysianow.com/news/2021/04/12/back-street-car-mechanics-just-scraping-by-as-drivers-take-the-high-road/>

Qtrac. (n.d.). Walk-in appointments vs. appointment scheduling: How Qtrac can help with both. Qtrac. <https://qtrac.com/blog/walk-in-appointments-vs-appointment-scheduling/>

Worlitz, J., Hettling, L., Woll, R., & Linh, D. (2020). Perceived Waiting Time and Waiting Satisfaction: a Systematic Literature Review. Brandenburg University of Technology, 22nd Quality Manage and Organisational Develop Conference. [https://www.researchgate.net/publication/338336943\\_Perceived\\_Wait](https://www.researchgate.net/publication/338336943_Perceived_Wait)

[ing Time and Waiting Satisfaction a Systematic Literature Review](#)

Tsernov, K. (n.d.). 4 ideas to reduce customer service wait times. Qminder. <https://www.qminder.com/blog/queue-management/reduce-wait-time-with-enjoyable-queues/>

Perlman, Y., & Yechiali, U. (2020). Reducing the risk of infection - The COVID-19 queuing game. *Safety science*, 132, 104987. <https://doi.org/10.1016/j.ssci.2020.104987>

Abdul Wahab, M. A. F., Mohd Jawi, Z., Abdul Hamid, I., Solah, M. S., Mohd Latif, M. H., Md Isa, M. H., Abdul Khalid, M. S., Ariffin, A. H., & Hamzah, A. (2017). Automotive consumerism in Malaysia with regard to car maintenance. *Journal of the Society of Automotive Engineers Malaysia (JSAEM)*, Vol. 1, Issue 2, pp. 137-153. [https://www.researchgate.net/publication/318283971\\_Automotive\\_Consumerism\\_in\\_Malaysia\\_with\\_Regard\\_to\\_Car\\_Maintenance](https://www.researchgate.net/publication/318283971_Automotive_Consumerism_in_Malaysia_with_Regard_to_Car_Maintenance)

Moazed, A. (n.d.). How long does it take to build an iOS or android mobile app? Applico Inc. <https://www.applcoinc.com/blog/long-take-build-ios-android-mobile-app/>

Jie, M. L. Z. (2022). EnNos Mobile Application for Automated Malware Tracking on Android. *Journal of Applied Technology and Innovation*, 6(3), 24-28. [https://dif7uuuh3zqcp5.cloudfront.net/wp-content/uploads/sites/11/2022/06/09143334/Volume6\\_Issue3\\_Paper5\\_2022.pdf](https://dif7uuuh3zqcp5.cloudfront.net/wp-content/uploads/sites/11/2022/06/09143334/Volume6_Issue3_Paper5_2022.pdf)

Ghosh, S. (n.d.). Why single activity architecture is preferred over multiple activity architecture in Android app development? Web Skitters Academy. <https://www.webskittersacademy.in/why-single-activity-architecture-android-app-development/#:~:text=Instead%20of%20using%20multiple%20activities,the%20stability%20of%20the%20application.>

Sabag, R. (2018). What problems exist with multi activities. Pro Android Dev. <https://proandroiddev.com/what-problems-exist-with-multi-activities-4ea1a335d85a>

Abdul Wahab, M. A. F., Mohd Jawi, Z., Abdul Hamid, I., Solah, M. S., Mohd Latif, M. H., Md Isa, M. H., Abdul Khalid, M. S., Ariffin, A. H., & Hamzah, A. (2017). Automotive consumerism in Malaysia with regard to car maintenance. *Journal of the Society of Automotive Engineers Malaysia (JSAEM)*, Vol. 1, Issue 2, pp. 137-153. [https://www.researchgate.net/publication/318283971\\_Automotive\\_Consumerism\\_in\\_Malaysia\\_with\\_Regard\\_to\\_Car\\_Maintenance](https://www.researchgate.net/publication/318283971_Automotive_Consumerism_in_Malaysia_with_Regard_to_Car_Maintenance)

Baijouk, M. (2021). Software testing for reliability and quality improvement. *Journal of Applied Technology and Innovation*, 5(2), 40-46. [https://jati.sites.apipi.edu.my/files/2021/03/Volume5\\_Issue2\\_Paper7\\_2021.pdf](https://jati.sites.apipi.edu.my/files/2021/03/Volume5_Issue2_Paper7_2021.pdf)