A Compact and Secure Anti-Theft Bike Mobile Application for iPhone Operating System (iOS) users

Software Engineering (COEN 6311)

Pranav Jha
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
jha_k.pranav@live.com

Jayapriya Muthuramasamy
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
jayapriya054@gmail.com

***Abstract*—Bike theft is a persistent issue that concerns many individuals. Despite taking precautions, bikes can still be stolen, and recovering them becomes incredibly challenging. To address this problem, we propose an anti-theft solution specifically designed for iPhone Operating System (iOS) users. Our solution comprises both hardware and software components. The hardware components consist of a compact module that can be easily installed on bikes. This module includes a Global Positioning System (GPS) tracker, an accelerometer, a gyroscopic sensor, and a microcontroller. These components work together to provide effective tracking capabilities. Complementing the hardware, we have developed an iOS application that can be downloaded from the iOS app store and installed on iPhones or iPads. The iOS application interacts with the hardware components through Firebase cloud services, ensuring seamless communication. Using the iOS application, users can track their bikes in real-time. The application synchronizes with the hardware components installed on the bike at regular intervals via Firebase cloud. Furthermore, users receive push message alerts regarding the parked location of their bike and any changes in its orientation. Additionally, the iOS application provides access to the bike's tracking history, allowing users to view historical data for up to a year. Users can customize the time interval for recording recent GPS locations, ranging from 1 minute to 1 hour, based on their preferences. By combining the hardware and software components, our anti-theft solution provides iOS users with an efficient and reliable means to track and monitor their bikes. This solution aims to enhance bike security and increase the chances of recovery in case of theft.**

***Keywords*—Anti-Theft, Bike, GPS, Alert, iOS, Cloud.**

DELIVERABLE 1-5

I. TASK 1: ADDITION OF LOCATIONS HISTORY

A. Use Case: Past Routes

The overall use case is shown in Fig. 1. The past route use case is presented in Fig. 2(a) which can be accessed from the home screen of our app as shown in Fig. 2(b). User can see the past route list using this service as shown in Fig. 4(a) and details about a particular trip among those listed routes which can be seen in Fig. 4(b). Depending on the time interval set by the user which can be 1 minute, 5 minutes up to an hour, the history of past routes of the bike will be recorded and saved in the app for an year at max. The user story associated to this use case is presented in Table I.

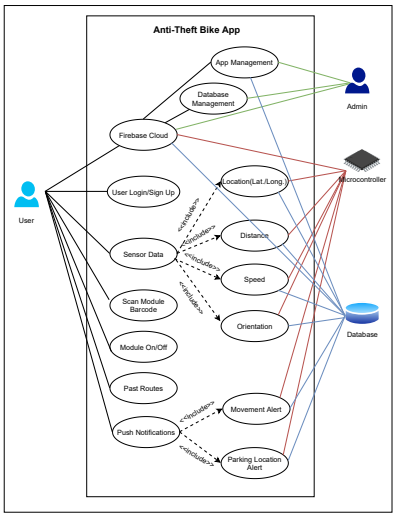
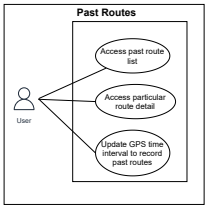
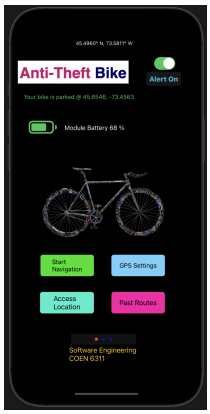


Figure 1: Overall use case diagram



(a) Past routes use case.

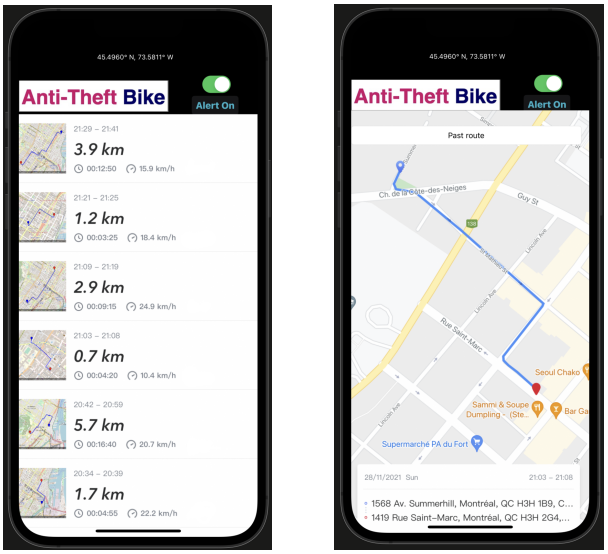


(b) Home screen

Figure 2: Past routes service addition to the app

B. Activity Diagram

The activity diagram for location history access by the user is presented below in Fig. 3. The user can access this service using the 'Past Routes'. This new change in our system from 'Assignment-1' [1] would be able to satisfy the corresponding request for change by the user which is "*In addition to above description, history of the bike tracking can be queried by the user given a time span specified by the user. The history is kept maximum for a year. The user can choose the time interval between 1 minute, 5 minutes, to 1 hour to record the GPS locations of the bike.*" [2]. The user will start by login and/or sign up to use this service inside the app. Then he will be redirected to the app's home screen where he will have options to either update the GPS settings required to record the past trip for 1 minute, 5 minutes up to 1 hour or he can choose to see the past route list as shown in the Fig. 4(a) and details about a certain trip from those mentioned in the list as shown in Fig. 4(b). The app will keep these details saved for an year and whenever the user wants to see these details, he can access these by going to home screen and clicking the 'Past routes' button. Inside app, these details will be categorized by date, time, day and month which will make it easier for the user to see these details within the span of one year.



(a) Past routes for last one hour (b) Past route

Figure 4: History of past bike locations

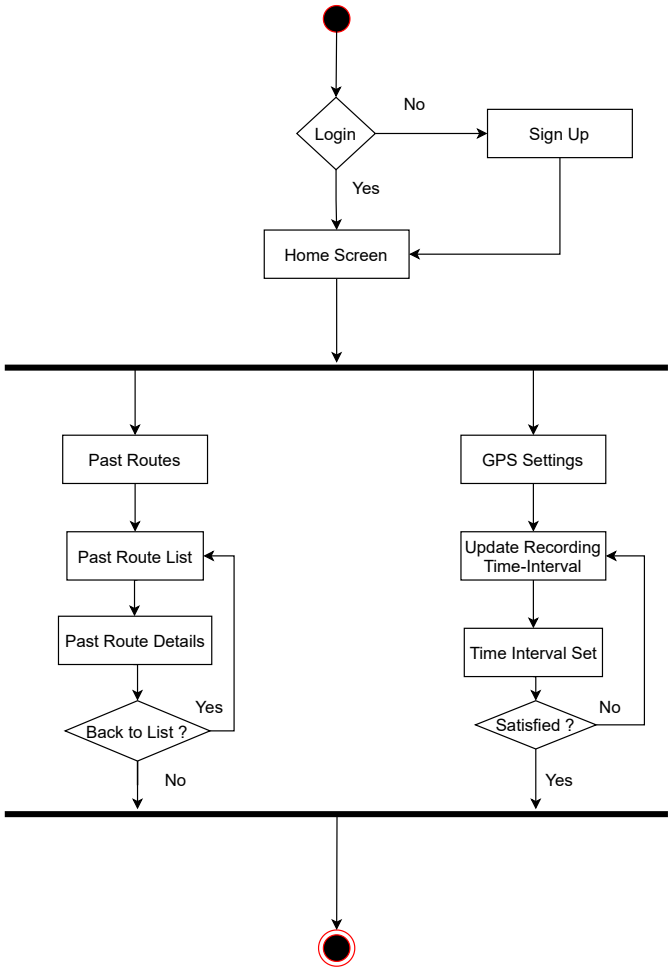


Figure 3: Activity diagram for 'past routes' use case

Table I: User story for past routes use case

Use Case	**GPS Locations History** Scenario: User would be able to track the history of the bike's GPS locations which will be saved in the app to view for maximum an year.
Summary:	This use case allows a login user to see the history of recorded GPS locations of bike for a specified time interval in the app for maximum an year.
Basic Flow:	1. The use case starts when a login user wants to use the app to see the past routes recorded inside the app for a particular time interval set by the user. 2. The user can see the past routes saved in the app. 3. The user can also see the details about a particular trip from that list.
Alternative Flow:	Step 1: If there are no recorded list of past routes, the user has not used his bike for an year and in order to see the history, he must use his bike for a few rides and then start with Step 1 in this use case story.
Preconditions:	The app is connected to the hardware module installed on the bike.
Postconditions:	The user can access the past route locations of the bike for one year.
Business Rules:	For using the services, user must purchase the "Anti-Theft Bike" iOS app.

II. Task 2: Context View Modeling

The context view modeling UML diagrams are presented below in following subsections. We have used 'Class Diagram' to solve our task. For comparison, these diagrams are classified mainly into (1) Class Diagram with History, and (2) Class Diagram without History.

Class Diagram without History: Fig. 5 [1] represents the class diagram without history for the 'Anti Theft Bike' app where the attributes and the associated functions/methods of each classes are clearly explained. Here, '+' represents the access modifier used is public, and '−' and '#' stands for private and protected access modifiers, respectively. Here, Microcontroller is the parent class with attribute 'Sensor ID' and the function of the module is to receive the sensor data namely location, distance, speed and orientation from the derived classes GPS Tracker, Accelerometer and Gyroscope, and transmit the same to the firebase cloud using the functions +movementAlert() and +ParkingLocationAlert(). The corresponding function of each derived class is mentioned in the Fig. 5. The user would be able to perform functions such as login, scanning barcode, receive push notifications and accessing the sensor data from the firebase cloud. The notation 1 to 1..∗ is used to demonstrate that the push notification and the sensor details can be transmitted n number of times to the firebase cloud which uses APNs Key as a token to send notifications to the users. Admin takes care of managing the iOS App and it's associated database where, user policy update, app update and managing the existing data are the main functions of the class 'Database management' [1].

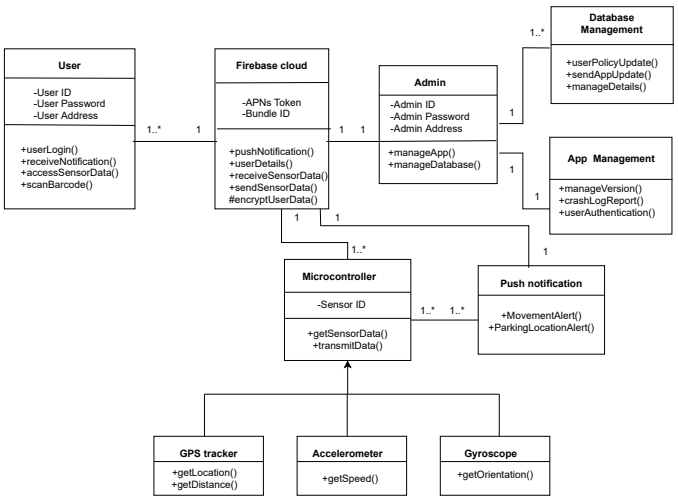


Figure 5: Class diagram without history

III. Task 3: Context View Modeling

Class Diagram with History:

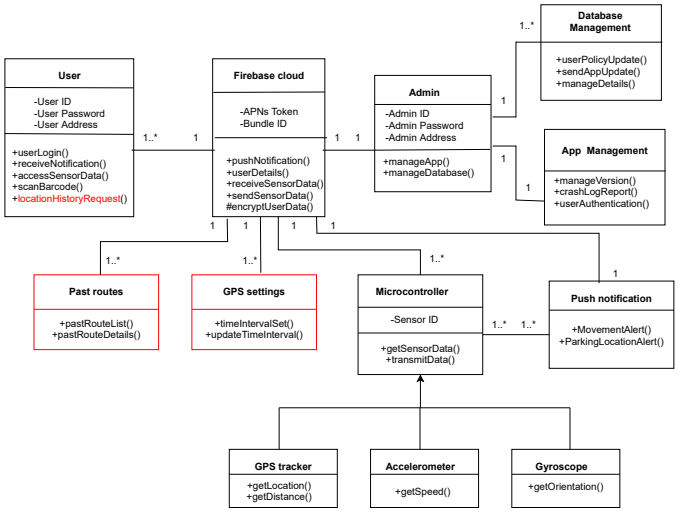


Figure 6: Updated class diagram with history

Class diagram with history as per the new requirement is shown in the Fig. 6. The user class is updated with an extra function named 'locationHistoryRequest()'as a part of the additional requirement. So, whenever the user wants to request the location history, he can access two classes namely 'Past routes' and 'GPS settings' (added as a part of additional requirement). In the Past route class, +pastRouteList() and +pastRouteDetails() are the two available functions with which the user can access the past route list easily. GPS setting class consists of two functions namely +timeIntervalSet() and +updateTimeInterval(). The function +timeIntervalSet() helps the user to set the time for 1 minute, 5 minutes up to 1 hour based on his needs and accordingly he can access the past route locations. If the user is not satisfied with the current time settings, he can access the function +updateTimeInterval() to update the time for which the location history is required. The classes Past routes and the GPS settings are connected to the Firebase using one to multiple function representing that the user can have multiple access to the functions specified under these classes.

Review: The process to handle the requested requirement in Task 1 [2] was presented in Section I using the activity diagram and III using the context view modeling UML diagram. The above-mentioned defined process is capable of handling the required request. The app will allow users to track their bike history and details about a particular trip from the list of all routes taken among the history list. When the user attempts to login the iOS Anti-theft bike mobile app, the app will check if the provided user detail is already existing in the Firebase cloud or not. If authenticated, the user will be able to login the iOS Application, on the other hand, if the authentication fails, then he has to sign up to access the app. Once the user is logged in, he gets to see Past routes and GPS setting options (newly added in Assignment 2) available on the app home page. Using past route option, the user can request for the past route details of the bike. The 'GPS settings' option will allow

user to update the time interval, which is set for 5 minutes by default, to record the past routes for 1 minute, 5 minutes up to 1 hour.

IV. Task 4: Behaviour View Modeling

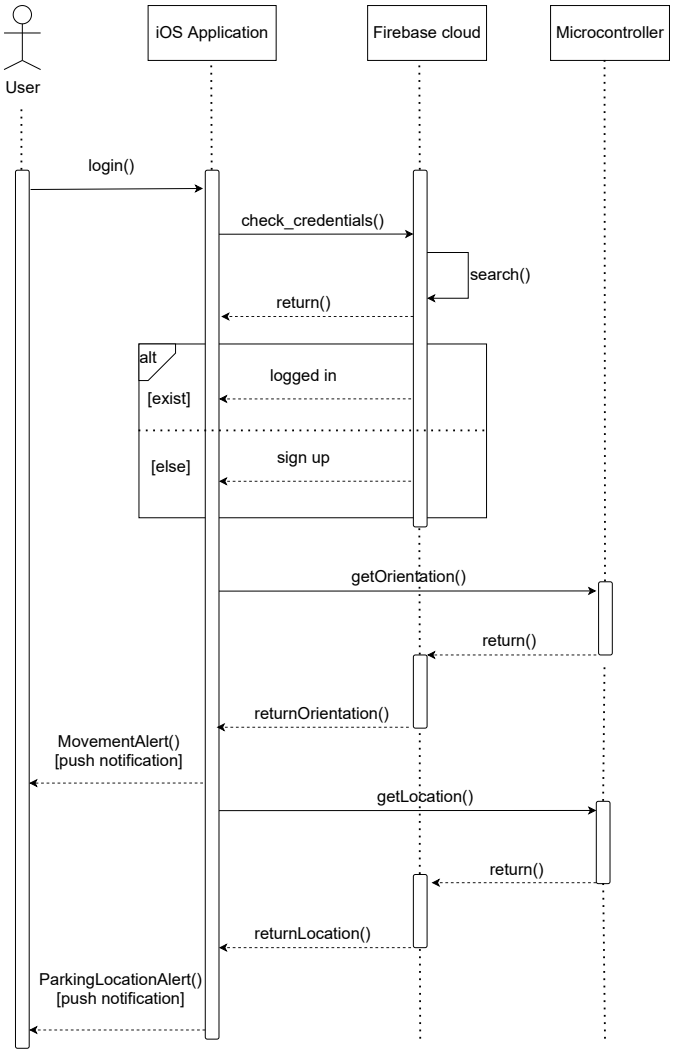
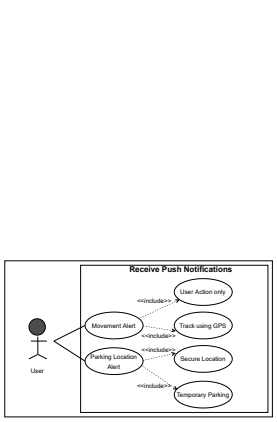


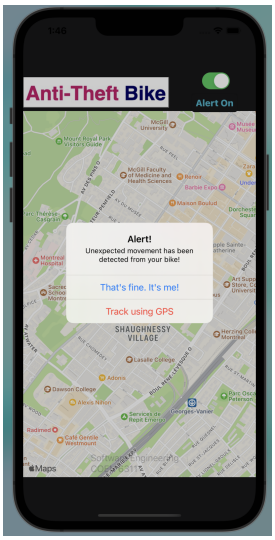
Figure 7: Sequence diagram for push notification use case



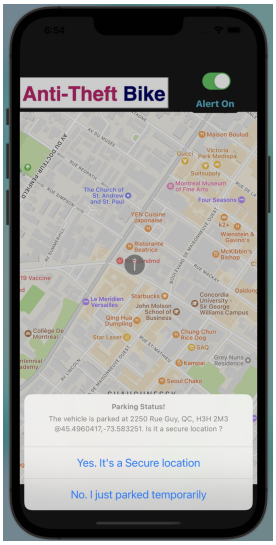
(a) Receive push notifications use case diagram.

(b) Push notification activity diagram.

Figure 8: Push notifications



(a) Push notification alert for unexpected movement in the bike's parked state.



(b) Push notification alert with GPS coordinates for the bike's parked location.

Figure 9: Push notifications

Fig. 7 represents the 'Sequence UML Diagram' to show the behavioural view modeling for the event driven flow of push notifications use case in Fig. 8(a) [1] and it's associated activity diagram and user story in 8(b) and Table II, respectively [1], which has been used in our app to alert the user for any form of suspicious activities about his bike. It is the interaction diagram which reflects the order of different tasks with associated group of objects where the order in which the interaction takes place is considered to be of a significant factor. Here, the sequence diagram is modelled for the task where users will be notified about the changes in the location or in the parked state of his bike via push notifications. The user can track his bike using the message prompt with a link to go for the GPS tracking immediately as shown in Fig. 9(a).

The use case and the activity diagram associated to this use case is shown in Fig. 8(a) [1] and Fig. 8(b), respectively using which a user can receive push notifications about the movement of the bike when it's parked. As shown in Fig. 9(a) [1], a push notification with message "Unexpected movement has been detected from your bike!" will be displayed to the user. The message will ask if this action is done by the user himself or provide a link to track the user's bike.

The user also receives push notification with the location coordinates whenever he finishes a trip and park his bike where he will be prompted to choose to add the location in the 'secure location list' or 'temporary parking' as shown in Fig. 9(b) [1]. If user selects it as a secure location, it will be updated in the database as 'secure location' and if any location from this

secure location is chosen to park in future, the system will not send a notification or alert message to the user. A user can stop receiving push notifications just by clicking alert on/off button on the app's home screen. The user story associated to this use case is presented in detail in Table II [1].

Table II

Use Case	**Receive push notifications** Scenario: User will get push notification alert for unexpected movement of bike and parked location.
Summary:	This use case allows a login user to use the push notification services.
Basic Flow:	1. The use case starts when a user connects the hardware module to his bike. 2. The alert to receive push notifications is set to 'on' by default. 3. User can activate or deactivate the alert. 4. The system will send push notification alert to the user in the app for any movement while parked. 5. The system will send push notification alert with GPS coordinates to the user in the app when the bike is parked.
Alternative Flow:	Step 2: If the bike is not moving for over five minutes after the last trip, the app will activate the alert.
Preconditions:	The app is connected to the hardware module.
Postconditions:	The user can switch on/off the alert service in the app.
Business Rules:	For using the services, user must purchase the "Anti-Theft Bike" app.

V. TASK 5: OBSERVER DESIGN PATTERN

We have applied observer design pattern to program the code to implement the scenario "*The companion application will notify users of these changes through push notifications. The application will allow users to track their bike in the event immediate intervention is not possible.*" [2], which will be submitted in a zip file along with this pdf.

The observer design pattern, as applied to the requirement mentioned in Task 5 [2], is depicted in Fig. 10 through a UML structural model. This pattern facilitates the use of push notifications to inform users about relevant changes in the parked state of their bikes.

In the design, an observable object is capable of attaching or detaching observers and notifying them whenever there is a change in the parked state of a bike. The notifications are sent as push messages to all iPhone/iPad devices connected to the observer's user account credentials. It is possible for an observer to have multiple iOS devices with the "Anti-Theft Bike" app installed, all logged in using the same username and password.

When a push notification is received, it prompts an interactive message on the user's app screen. This message asks if the user wants to track the bike using GPS, providing an immediate intervention option.

The observer design pattern ensures effective communication between the observable (the bike's parked state) and the observers (the iOS devices of the user). This allows for real-time updates and user interaction based on the received push notifications.

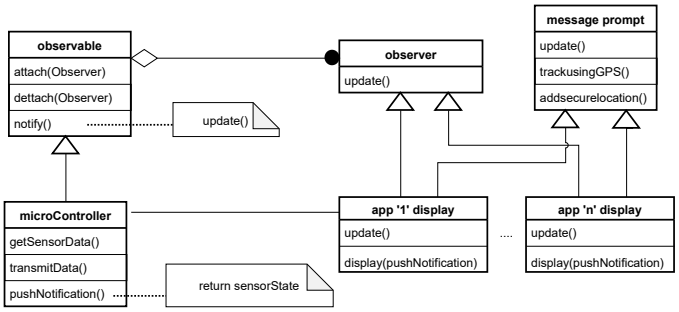


Figure 10: Observer view pattern for our app

REFERENCES

[1] P. Jha and J. Muthuramasamy. "A compact and secure anti-theft bike mobile application for iphone operating system (iOS) users," Assignment-1, *Software Engineering (COEN 6311)*. [Online]. Available: https://www.overleaf.com/3147613129bnxfhvfdnsrh

[2] B. Goodarzi. "Assignment-2," *Software Engineering (COEN 6311)*. [Online]. Available: https://moodle.concordia.ca/moodle/pluginfile.php/5172809/mod_resource/content/1/Assignment%202.pdf