A compact and secure Anti-Theft Bike mobile application for iPhone Operating System (iOS) users

Software Engineering (COEN 6311)
Assignment-1

Pranav Jha
Student ID: 40081750
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
jha_k.pranav@live.com

Jayapriya Muthuramasamy
Student ID: 40184587
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
jayapriya054@gmail.com

***Abstract*—Bike theft has always been an issue to be concerned about. Despite of being cautious, one may get their bike stolen and it becomes extremely difficult or almost impossible to track it once it happened. Targeting the iPhone Operating System (iOS) users, we propose an anti-theft solution to this problem which has hardware as well as software components. Hardware components consist of a Global Positioning System (GPS) tracker, an accelerometer, a gyroscopic sensor and a microcontroller, all are embedded in a compact hardware module and can easily be installed on the bikes whereas software component consists of an iOS application which can be downloaded from the iOS app store and installed on iPhone/iPad. The iOS application interacts with the hardware components using Firebase cloud services. Users can track their bike using the iOS application that synchronizes with the hardware components installed on the bike on a regular interval through firebase cloud. User also receives push message alerts about the parked location of the bike, and also if there are any changes in the orientation of the parked bike.**

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

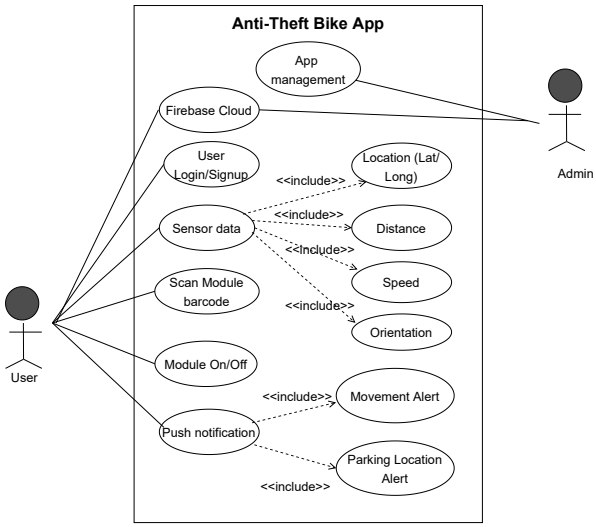

Figure 1: Overall use case diagram

DELIVERABLE 1-5

I. USE CASE

A. Use Cases

The overall use case is shown in Fig. 1 The details about the use cases [1] along with the corresponding use case diagrams for our 'Anti-Theft Bike' app are listed below:

1) **Log in/Sign up:** An user can log in or sign up using this use case which is shown in Fig. 2. The prototype of our application shows the login screen in Fig. 3. The user stories for this use case is presented in the Tab. I and Tab. II [2]. User can use a registered username and password to login into the app or use his Apple or Facebook account to login directly, for which the links are available in the app.

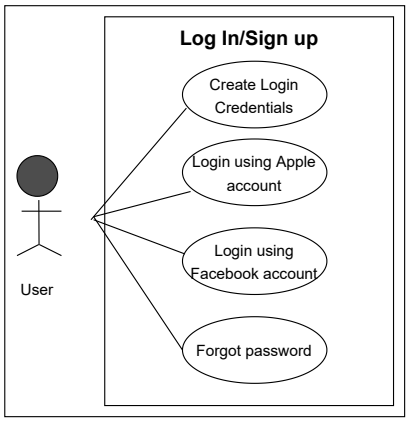

Figure 2: Log in/Sign up use case diagram

Table I

Use Case	Login
Summary:	This use case allows a user to login into the 'Anti-Theft Bike' app and can access the app functionalities.
Basic Flow:	1. The use case starts when a user indicates to login in the app through the option available on the screen. 2. The app requests the username and password. 3. The user provides username and password or choose a login options available in the app screen. 4. The app verifies the username and password against all registered users. 5. The app allows login session and displays the scan barcode section in order to connect to the hardware module.
Alternative Flows:	Step 4: If username is invalid, the use case goes back to step 2. Step 4: If the password is invalid the system requests that the user re-enter the password. When the user enters another password, the use case continues with step 4 using the original username and new password.
Preconditions:	The user is registered in the app.
Postconditions:	The user can now connect the app to the hardware module through the barcode provided.
Business Rules:	User must purchase the app with full payment at the iOS App store.

Table II

Use Case	Sign Up
Summary:	The user must register a username and password in order to access the app services
Basic Flow:	1. The use case starts when a user wants to register in the app. 2. The app requests a username and password. 3. The user enters a username and password. 4. The app checks that the username does not duplicate any existing registered usernames. 5. The app requests for address, phone number and email address. 6. The user enters the information. 7. The app determines the user's location and stores all user information. 8. The app executes use case Register Preferences. 9. The app starts a login session and displays the scan barcode section.
Alternative Flows:	Step 4: If the username duplicates an existing username, a message is displayed and the use case repeats step 2. Step 5: If the user does not enter a required field, the app displays a message and the use case goes back to step 4.
Preconditions:	The user is not registered in the app.
Postconditions:	The user can now access the services provided by the app.
Business Rules:	User must purchase the app with full payment at the iOS App store.

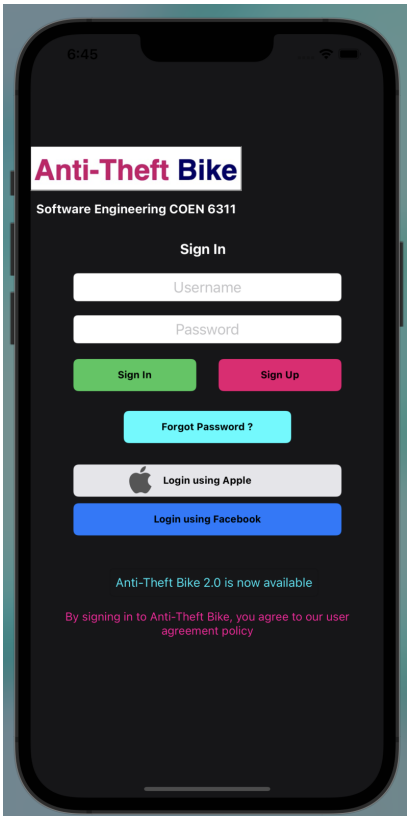

Figure 3: Login and Sign up screen

2) **Hardware module set up:** When a user buys the 'Anti-Theft Bike' app from iOS app store, he will be sent a hardware module which can be installed on the bike very easily. The hardware module comes with a barcode which will be used to connect it to the app in order to send sensor's data. Once the user is logged in to the app, he would be able to connect the hardware module just by scanning the barcode provided with it. After this step, he can access data and perform functions in the app. The user story associated with this use case is presented in Tab. III

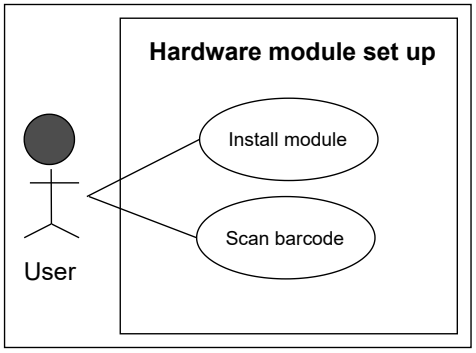

Figure 4: Hardware module setup use case diagram

Table III

Use Case	**Hardware module set up** Scenario: User would be able to connect the hardware module to the app by scanning the barcode on the hardware module.
Summary:	This use case allows a login user to connect to the hardware module.
Basic Flow:	1. The use case starts when a user install the hardware module to his bike. 2. The user indicates to scan the barcode to connect with the hardware module installed on the bike.
Alternative Flow:	Step 2: If the scan is not successful the use case returns to step 2.
Preconditions:	The user is logged in to the app.
Postconditions:	The user is connected to the hardware module.
Business Rules:	For using the services, user must purchase the“Anti-Theft Bike” app.

3) **Receive push notifications:**
The use case is shown in Fig. 5 using which a user can receive push notifications about the movement of the bike when it's parked. As shown in Fig. 6, 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 or provide a link to track the user's bike. User will also receive the notification whenever he stops or parks his bike. As shown in Fig. 7, the message will be displayed as a push message with the location information such as address and coordinates. It will also ask if the location is secure or user has parked only for some time. 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 home screen of the app as shown in Fig. 12. The user story associated to this use case is presented in detail in Tab. IV.

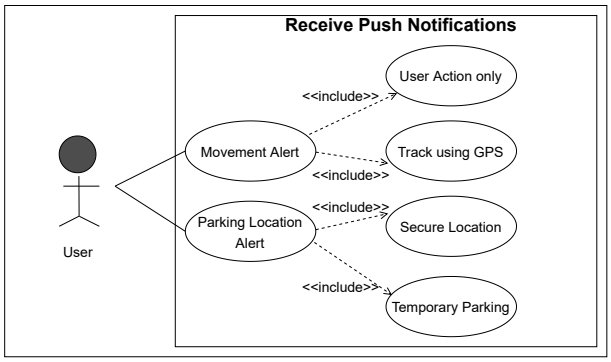

Figure 5: Receive push notifications use case diagram

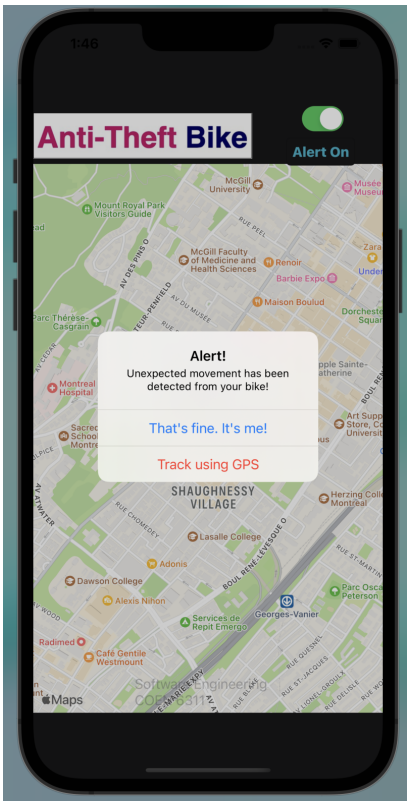

Figure 6: Push notification alert for unexpected movement in the bike's parked state.

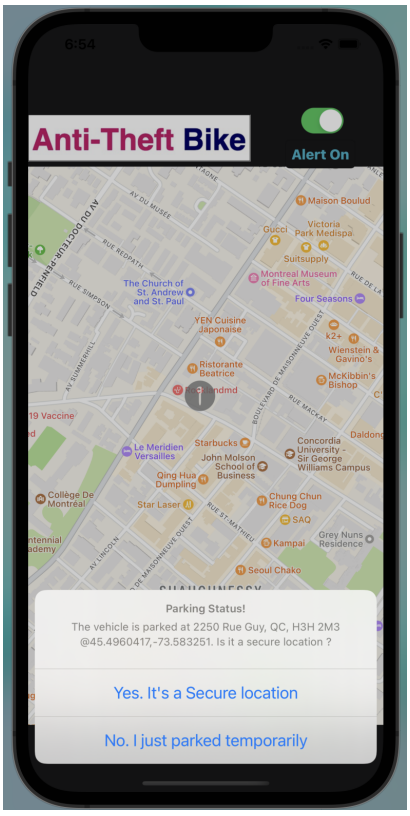

Figure 7: Push notification alert with GPS coordinates for the bike's parked location.

Table IV

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.

Table V

Use Case	**Access the GPS** Scenario: User would be able to track his bike if any unexpected movement has been reported through "Push Notification" use case.
Summary:	This use case allows a login user to use the GPS services in the app.
Basic Flow:	1. The use case starts when a login user wants to use the app. 2. The user can see the parked location of the bike connected to the hardware module. 3. The user can track his bike if it is stolen. 4. The user can navigate using the map view in the app.
Alternative Flow:	Step 2: If the bike is parked somewhere, the user will receive the parking location coordinates using 'Receive push notifications' use case and would respond to the message prompt on the screen asking if that location needs to be added to the secure location list or not.
Preconditions:	The app is connected to the hardware module installed on the bike.
Postconditions:	The user can access the location or start navigation in the app map view.
Business Rules:	For using the services, user must purchase the "Anti-Theft Bike" iOS app.

4) **Access the GPS services:** User can access the location of his bike using this use case which is hown in Fig. 8. He can also be able to navigate using the GPS inside the app map view and can select different routing options such as navigation routes for cars, bikes or walking. If user's bike is stolen, he can track the bike using this service. He will also get location updates from 'Receive push notifications' use case whenever the bike is stopped or parked. The user story associated to this use case is presented in the Tab. V.

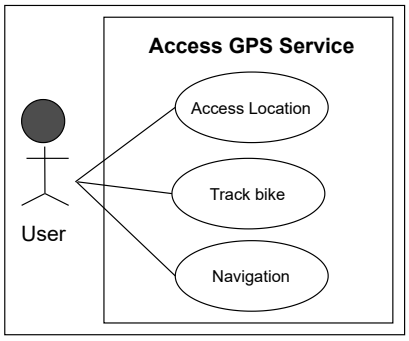

Figure 8: Access the GPS services use case.

II. Main Requirements

The main requirements consist of following requirements along with it's sub requirements:

1) **Login/Sign Up:** User can login or sign up in the app.
   - User can login using registered username and password.
   - App shall provide alternate methods to login such as using his apple or facebook account if a user does not want to create an account.
   - User can login using his Facebook account.
   - User should be able to connect the app to the hardware module: The hardware module along with a barcode is couriered to the user when a user purchases the app on the iOS app store.
2) **Push Notification:** App shall notify the user via push messages if any movement is detected in the bike's parked state or send the location of the bike with GPS coordinates whenever a bike is parked or stopped.
   - User can activate or deactivate the alert.
   - User can mark a parked location as a secure location where if parked again in the future, won't be notified by the app.
3) **GPS Location Services:** User can track the location of the bike at any point of time he decided to do so using the "Anti-Theft Bike" app.
   - User can navigate using the app's map view screen.
   - User can mark a parked location as a secure location.

III. System Design

The system design for our app is classified mainly in two categories (1) Architecture Diagram and (2) Class Diagram, the details about which are mentioned below:

A. Architecture Diagram

The architecture diagram of the complete system can be divided into (i) The high level architecture diagram as shown

in Fig. 9 and (ii) The system level architecture diagram as shown in Fig. 10.

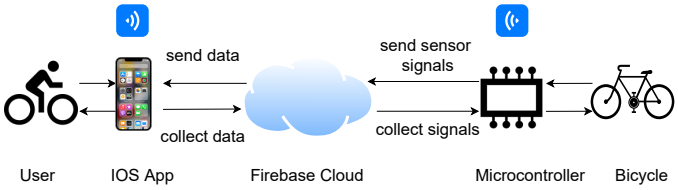

Figure 9: High level architecture diagram.

1) High Level Architecture Diagram: The high level architecture diagram [3] of 'Anti-Theft Bike' app has been provided in 9. The system has three components: an iOS mobile application, a firebase cloud service, and a hardware module. The hardware module has a microcontroller, gyroscope, GPS and accelerometer. The user needs to attach the hardware module to the bike. It fetches data from the bike and updates the firebase cloud in real-time. The user can receive that information from the firebase cloud using our 'Anti-Theft Bike' app installed in the user's iPhone/iPad.

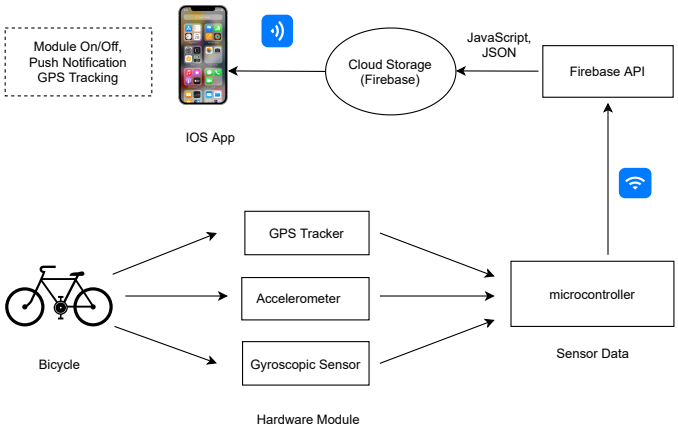

Figure 10: System level architecture diagram.

2) System Level Architecture Diagram: As shown in Fig. 10, the system of 'Anti-Theft Bike' app is basically an iOS app which can be installed on iPhones/iPads and a compact hardware module which can be installed or fitted on the bike of the user. When a bike is parked, the alert is activated and gyroscope sensor would be reading the movement data which for a parked or stopped state is zero. Now, if there is any changes in this 'zero' movement measurement by the gyroscope, it will send the readings to microcontroller and as the reading are non zero, microcontroller will send a notification alert in form of message to the firebase cloud and the same message will be received at the user's iPhone/iPad. The user can respond to the message for any suspicious activity.

The GPS is another sensor installed on the hardware device and When a user wants to navigate through the city, he can use the GPS navigation service which will be accessed inside the app's map view screen.

The accelerometer is basically to measure the proper acceleration of the bike and the data will be used while navigating using GPS. If user is accelerating beyond certain limit, he will be notified to reduce the velocity as the acceleration is a rate of change of velocity over time. This can be considered as a safety feature of the app but we are not considering it adding to our app for now. The data is stored in the JSON database which will provide it to the app when requested. To enable sending push notifications the app reuires an Apple Push Notification (APNs) key which will be used as a token by Firebase cloud messaging service to send push notifications to the app identified by the App ID.

B. Class Diagram

Fig. 11 represents the class diagram 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. As show in the Fig. 11, Microcontroller hardware module 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 in using function +movementAlert() and +ParkingLocationAlert(). The corresponding function of each derived class is mentioned in the Fig. 11. The user would be able to perform functions such as login, scanning barcode, receive push notifications and access 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 to 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 the associated database. User policy update, app update and managing the existing data are the main functions of the class 'App/ Database management'.

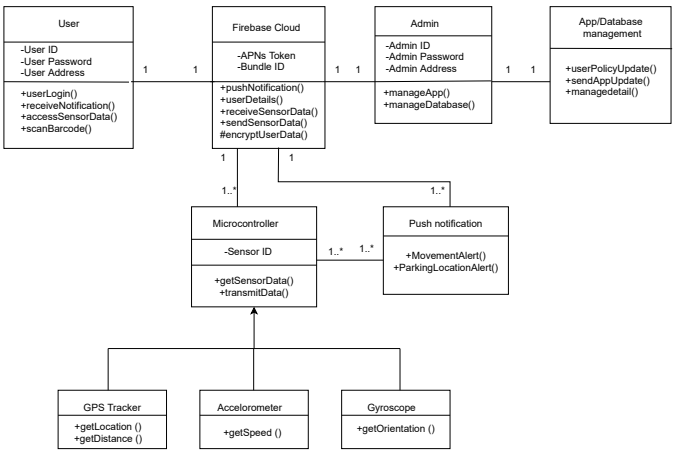

Figure 11: Class diagram

IV. Incremental Process

The Activity diagram is illustrated for the "GPS Location Service", which is one of the main requirements of 'Anti Theft Bike' app mentioned in section. II. Once the user is Logged in to the app, he would be directed to the home screen where

he can access the 'Scan Barcode' option in order to connect to the Hardware module. Once this is done, the hardware module will be activated and the user can access the GPS location coordinates of the bike from the app's home screen as shown in Fig. 12. The user can also navigate in the map view section of the app by 'Start Navigation' button provided in the app's home screen. The location of the bike can be seen as a blue circled dot as shown in Fig. 14 and can be used whenever user wants to track his bike. Location and navigation parameters will be generated in real-time and will be sent to the app using firebase cloud. These data can be accessed by the user whenever requested through app. If the bike is parked in a random location, the GPS sends the current location coordinates as a push notification to the user to check if the bike is going to be parked temporarily or if the location can be considered as a secure location as shown in fig.7. If the user chooses to add the location as a secure location, he will not receive further notifications for the same location as the location gets saved in the secured location list in the database. On the other hand, if the bicycle is parked temporarily, user will keep receiving the same push notification if he comes for the same location again in the future.

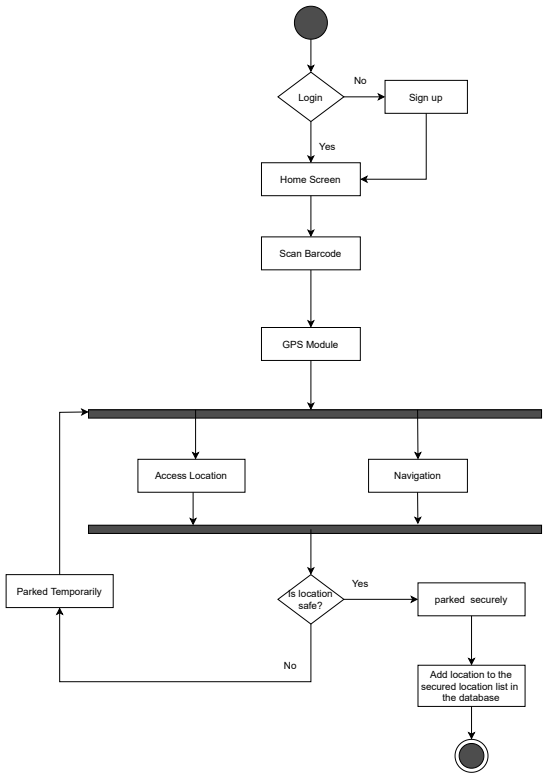

Figure 13: Activity diagram

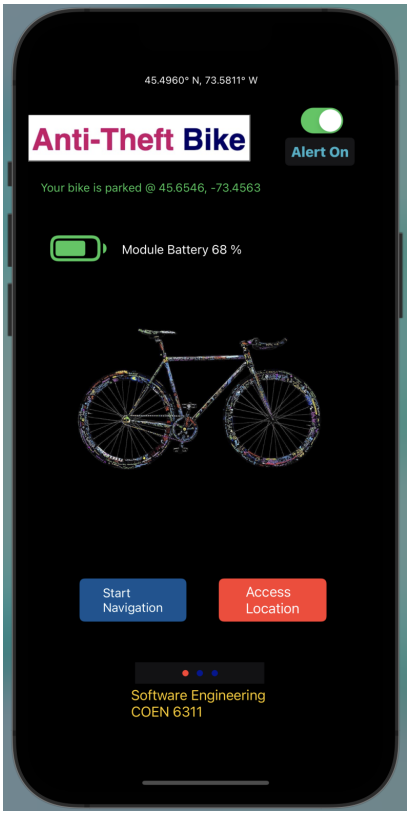

Figure 12: Home screen of the 'Anti-Theft Bike' app

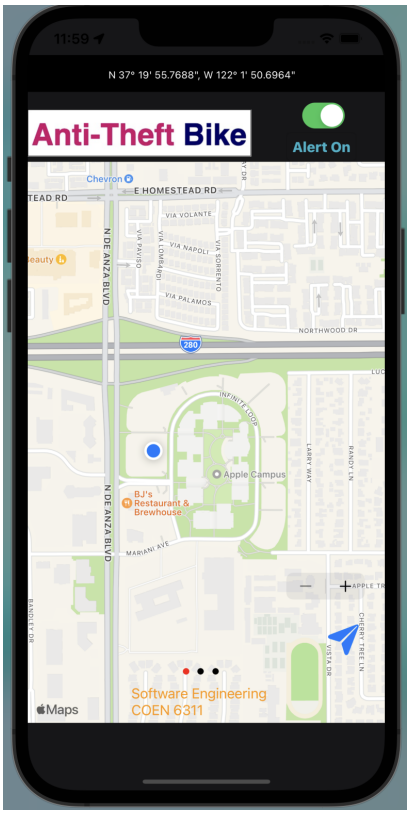

Figure 14: Bike's location in the blue circled dot provided by the GPS

V. Extreme Programming

We are defining two task cards for our selected requirement in section IV as (i) Location access and (ii) Navigation. We are choosing our first task card for prototype and unit testing. Our task is designed using the following agile practices [4]:

1) Location access:

- **Planning:** The higher level features of our project is defined well ahead in time when we received the assignment and for every feature such as GPS location, push notifications, navigation we divided our time to complete each of them within two to three days of efforts.
- **Simple Design:**
  Our design for the 'Anti-Theft Bike' app is simple and anyone who uses iOS devices, can easily access the features such as 'access location' or 'navigation'. The prototype of our application is presented in Fig. 15. We used Xcode 13.1 to design our app in swift language developed by Apple Inc. and used the iPhone 13 Pro max simulator in it to simulate our app. We installed CocoaPods which made managing our project much simpler by easily adding, removing and updating libraries. Also, firebase allowed us to synchronize and store data quickly in real time. We have provided the source code for our project in a zip file. The screenshots for our task card is provided in Fig. 16, Fig. 17, Fig. 18 and 19.

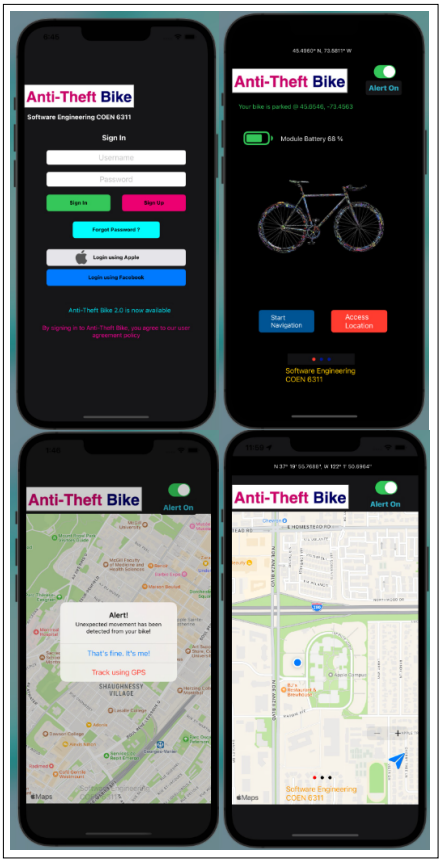

Figure 15: Prototype of our app

- **Unit Testing:** For testing of our design, we need the actual hardware module without which it is not possible to get the actual readings to generate the results on the simulator. But the use case, the user story associated to this use case and the activity diagram is given in the Tab. V, Fig. 8 and Fig. 13, respectively. Also, the details about the task is presented in section IV.
- **Pair Programming:** We worked as a team. We are two members and the name and student ID is mentioned at the top of the page. Working as a team is always recommended for earliest possible code inspections, earliest possible brainstorming and also, peer pressure reinforces discipline.
- **Collective Ownership:** This is good for programmers which can be interchanged at times and needs. The team can maintain it's speed for the project and can change anything, anytime, without any kind of delay.
- **Sustainable Place:** As tired programmers tend to make mistakes quite often, it is advisable to stay fresh, healthy, positive, and effective and in this way extreme programming can be used for more efficient workplace outputs and also for the long run. We followed this and kept ourselves motivated throughout the assignment project.

```swift
//
//  ViewController.swift
//  MapLocation
//
//  Created by Pranav K. Jha on 2021-10-30.
//

import UIKit
import MapKit
import CoreLocation

class ViewController: UIViewController, CLLocationManagerDelegate {


    @IBOutlet var mapView: MKMapView!

    let manager = CLLocationManager()

    override func viewDidLoad() {
        super.viewDidLoad()

    }


    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        manager.desiredAccuracy = kCLLocationAccuracyBest // Battery drain
        manager.delegate = self
        manager.requestWhenInUseAuthorization()
        manager.startUpdatingHeading()
    }

    func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
        if let location = locations.first {
            manager.stopUpdatingLocation()

            render(location)
        }
    }


    func render(_ location: CLLocation){

        let coordinate = CLLocationCoordinate2D(latitude: location.coordinate.latitude, longitude: location.coordinate.longitude)

        let span = MKCoordinateSpan(latitudeDelta: 0.1, longitudeDelta: 0.1)

        let region = MKCoordinateRegion(center: coordinate, span: span)

        mapView.setRegion(region, animated: true)

        let pin = MKPointAnnotation()
        pin.coordinate = coordinate
        mapView.addAnnotation(pin)

    }
}
```

Figure 16: View controller swift code for map location in Xcode 13.1

```swift
//
//  ViewController.swift
//  Accelerometer
//
//  Created by Pranav K. Jha on 2021-10-30.
//
import CoreMotion
import UIKit

class ViewController: UIViewController {

    let manager = CMMotionManager()

    override func viewDidLoad() {
        super.viewDidLoad()

        manager.startGyroUpdates()
        manager.startAccelerometerUpdates()

        manager.accelerometerUpdateInterval = 3

        manager.startDeviceMotionUpdates()

        if let gyroData = self.manager.gyroData {
            gyroData.rotationRate.x
            gyroData.rotationRate.y
            gyroData.rotationRate.z
        }

        Timer.scheduledTimer(withTimeInterval: 1, repeats: true) {_ in

            if let data = self.manager.accelerometerData{

            data.acceleration.x
            data.acceleration.y
            data.acceleration.z

            }

        }

    }


}
```

Figure 17: View controller swift code for accelerometer and gyroscope readings in Xcode 13.1

```swift
//
//  Created by Pranav K. Jha on 2021-10-30.
//

import UIKit
import MapKit
import CoreLocation

class ViewController: UIViewController, CLLocationManagerDelegate {


    @IBAction func didTapButton() {
        showActionSheet()
            // showAlert()

    }

    func showAlert() {
        let alert = UIAlertController(title: "Alert!", message: "Unexpected movement has been detected from your bike!", preferredStyle: .alert)

        alert.addAction(UIAlertAction(title: "That's fine. It's me!", style: .default, handler: {action in
            print("tapped Dismiss")
        }))


        alert.addAction(UIAlertAction(title: "Track using GPS", style: .destructive, handler: {action in
            print("tapped Dismiss")
        }))

        present(alert, animated: true)
    }

    func showActionSheet() {

        let actionsheet = UIAlertController(title: "Parking Status!", message: "The vehicle is parked at 2250 Rue Guy, QC, H3H 2M3
            045.4960417,-73.583251. Is it a secure location ?", preferredStyle: .actionSheet)


        actionsheet.addAction(UIAlertAction(title: "Yes. It's a Secure location", style: .default, handler: {action in
            print("tapped Dismiss")
        }))


        actionsheet.addAction(UIAlertAction(title: "No. I just parked temporarily", style: .default, handler: {action in
            print("tapped Dismiss")
        }))
        //actionsheet.addAction(UIAlertAction(title: "GPS: Track your bike", style: .destructive, handler: {action in
            // print("tapped Dismiss")
        //}))

        present(actionsheet, animated: true)
    }
```

Figure 18: View controller swift code for secure location in Xcode 13.1 part I

```swift
    @IBOutlet var mapView: MKMapView!

    let manager = CLLocationManager()

    override func viewDidLoad() {
        super.viewDidLoad()

    }


    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        manager.desiredAccuracy = kCLLocationAccuracyBest // Battery drain
        manager.delegate = self
        manager.requestWhenInUseAuthorization()
        manager.startUpdatingHeading()
    }

    func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
        if let location = locations.first {
            manager.stopUpdatingLocation()

            render(location)
        }
    }


    func render(_ location: CLLocation){

        let coordinate = CLLocationCoordinate2D(latitude: location.coordinate.latitude, longitude: location.coordinate.longitude)

        let span = MKCoordinateSpan(latitudeDelta: 45.4893, longitudeDelta: -78.6353)

        let region = MKCoordinateRegion(center: coordinate, span: span)

        mapView.setRegion(region, animated: true)

        let pin = MKPointAnnotation()
        pin.coordinate = coordinate
        mapView.addAnnotation(pin)

    }
}
```

Figure 19: View controller swift code for secure location in Xcode 13.1 part II

REFERENCES

[1] A. Mishra, J. Elijah, J. Y. Maipan-uku, and A. Awal, "An android based intelligent anti-theft and tracking system for automobiles," *Asian Journal of Computer Science Engineering*, vol. 3, no. 3, pp. 47–54, 2018.

[2] M. B. Lohse, C. C. Caldwell, D. D. Dreese, and H. H. Hitchcock. "A new web-based sales system for solar based energy, inc. (sbe)," *SBE Sales System*. [Online]. Available: https://web.cse.ohio-state.edu/~bair.41/616/Project/Example_Document/Req_Doc_Example.html

[3] M. R. A. Sarker, K. T. Alam, and T. Rahman, "Dichokro: An anti-theft system for two wheelers," in *2019 22nd International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2019, pp. 1–5.

[4] K. R. Larson. "Extreme programming and scrum - getting started with agile software development," *Advanced Technologies Integration, Inc.* [Online]. Available: www.atico.com