FRANKFURT UNIVERSITY OF APPLIED SCIENCES

GROUP PROJECT REPORT

Ubugs Watch

Author:
Dao The Hien
Dang Hoang Anh Khoi
Nguyen Truc Linh
Truong Quang Minh
Pham Phi Long

Supervisor:
DR. PETER THOMA
DR. ROBERT LOKAICZYK

Ubugs Watch
Computer Science

February 27, 2025

Declaration of Authorship

We hereby certify that the "Ubugs Watch" project report we are submitting is entirely our own original work except where otherwise indicated. We did not submit this work anywhere else before. We are aware of the University's regulations concerning plagiarism, including those regulations concerning disciplinary actions that may result from plagiarism. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Signed: Pham Phi Long - 1520590

Signed: Dao The Hien - 1515365

Signed: Dang Hoang Anh Khoi - 1518337

Signed: Nguyen Truc Linh - 1517327

Signed: Truong Quang Minh - 1516812

Date:
February 27, 2025

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Abstract

Faculty 2
Computer Science

Project Report

Ubugs Watch

by Dao The Hien
Dang Hoang Anh Khoi
Nguyen Truc Linh
Truong Quang Minh
Pham Phi Long

The "Ubugs Watch" project, under the supervision of Dr. Peter Thoma and Dr. Robert Lokaiczyk, focuses on implementing and analyzing smartwatch features using Unified Modeling Language (UML). The research aims to provide a comprehensive UML diagram that shows elements like timer, stopwatch, gps tracking, weather display, fitness tracking, sleep tracking, calling application, and—above all—sightless support which is very helpful for blinded people. By using UML to standardize smartwatch system design, the project hopes to increase stakeholder engagement and provide the wearable technology community with meaningful information. The resultant image serves as a vital tool for understanding, enhancing, and expanding the capabilities of smartwatches in the quickly evolving technological landscape.…

Acknowledgements

We express our sincere gratitude to Dr. Peter Thoma and Dr. Robert Lokaiczyk for their assistance throughout the development of our smartwatch project. We appreciate you giving us access to the licensing server for Magic Systems of Systems Architect (Magic Draw) and the Balsamiq Wireframes tool at this momentous completion moment. Our abilities to create UI prototypes and diagrams for the project was substantially aided by this help.

Contents

Chapter 1

Introduction

1.1 Project Description

The purpose of this paper is to present and document the development of our own smartwatch features and especially sightless support dedicated for blinded people. This comprises requirement analysis, UML diagram creation, and, UI prototype design where possible.

In order to represent our functions more detail, we will first use Trello board to generate some basic ideas of implementation. After that, we will create the relevant diagrams and UI prototype for each function. There are 9 functions that we have carried out. 8 of them will be evenly distributed to 4 members and the special sightless support will be given to 1 person.

1.2 Ideas

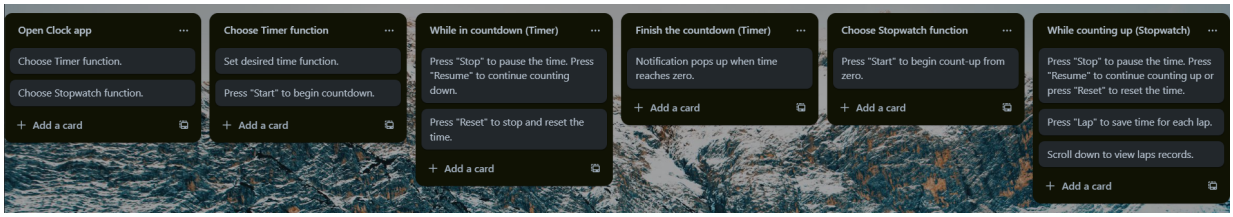


FIGURE 1.1: Timer and Stopwatch

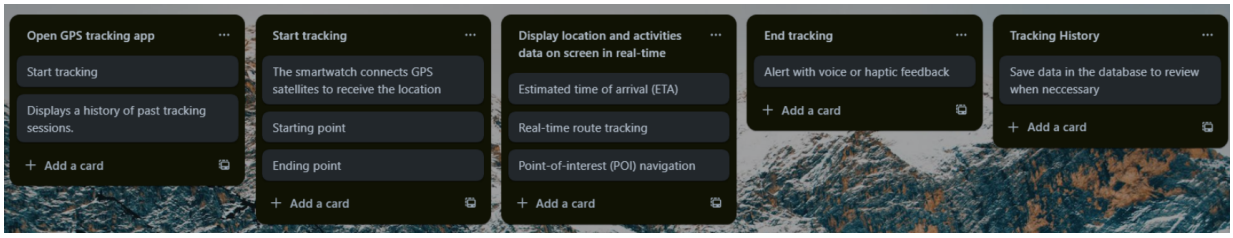


FIGURE 1.2: GPS Tracking

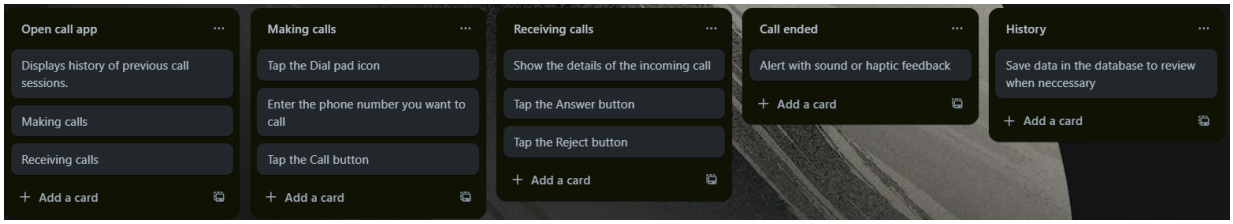


FIGURE 1.3: Calling Function

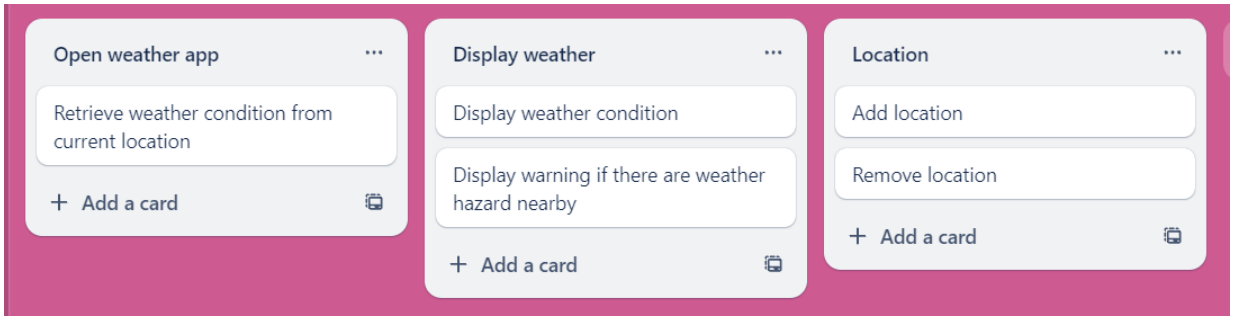


FIGURE 1.4: Weather Display

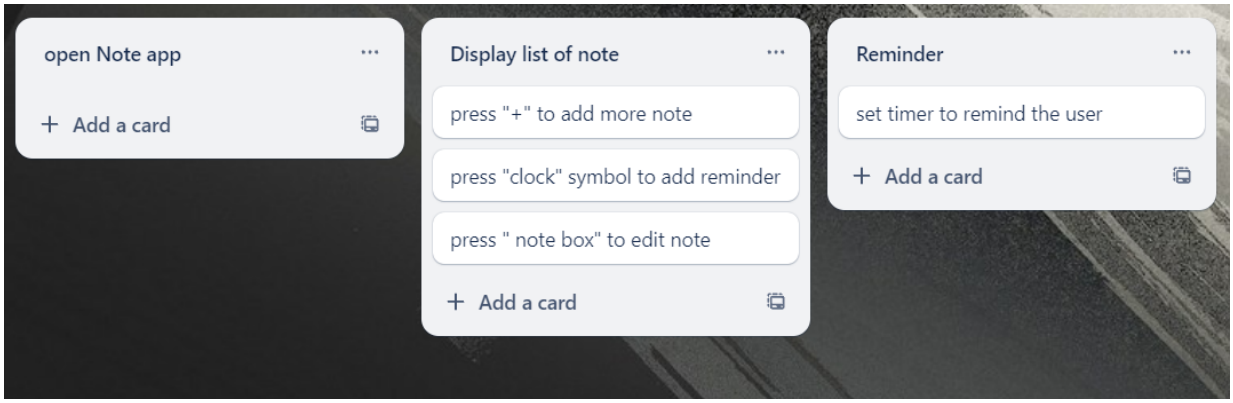


FIGURE 1.5: Note

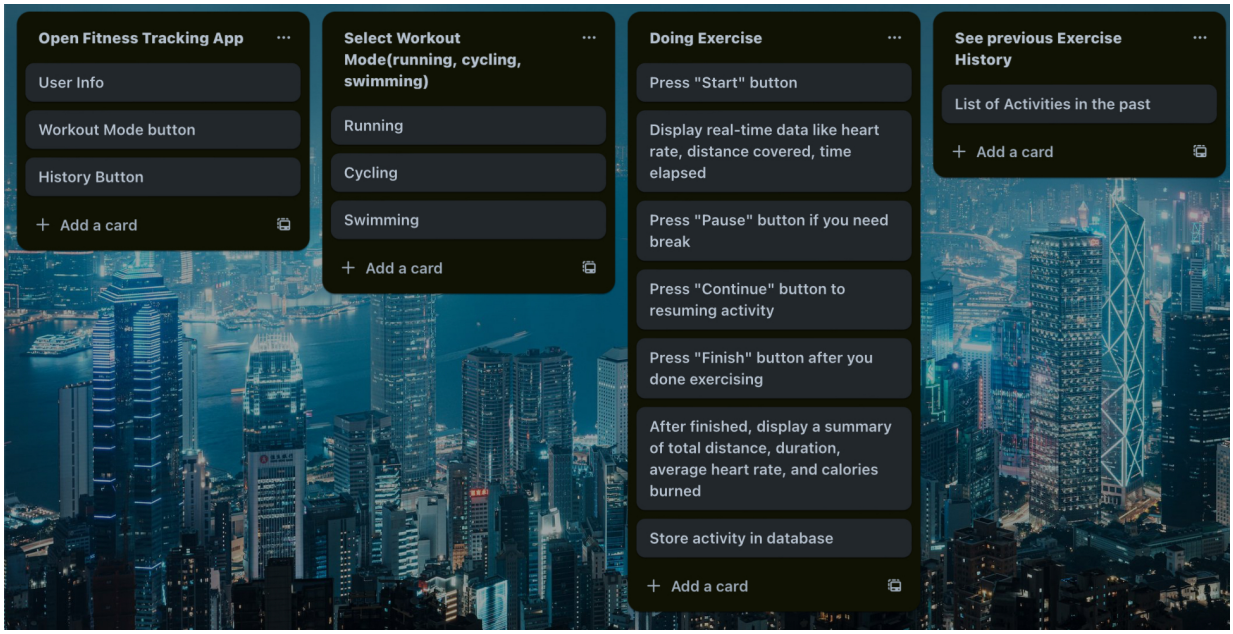


FIGURE 1.6: Fitness Tracking

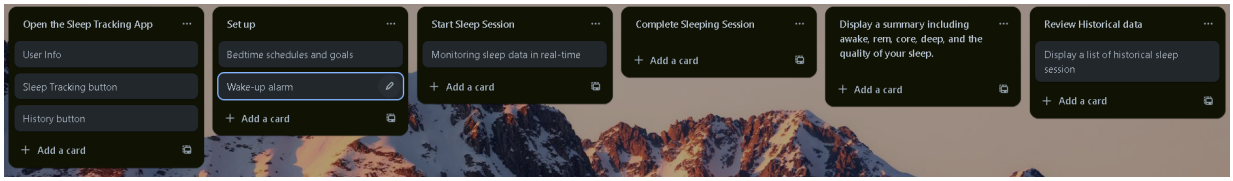


FIGURE 1.7: Sleep Tracking

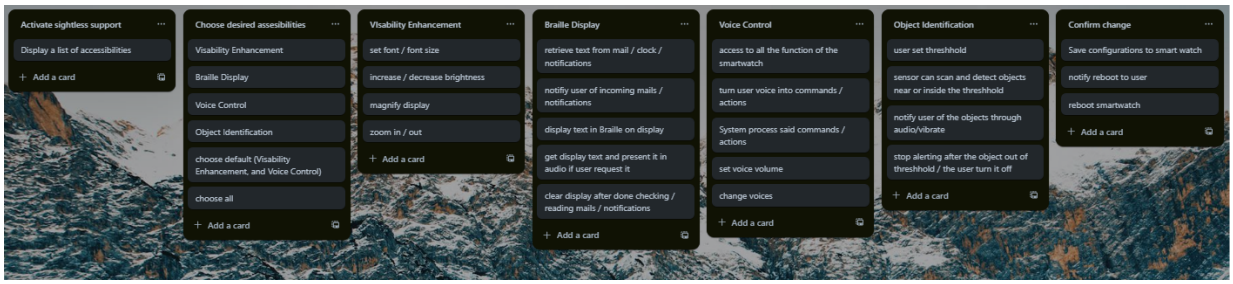


FIGURE 1.8: Sightless Support

1.3 Implementations

1.3.1 Timer and Stopwatch

At the beginning, we decided to add two simple features that any smartwatch must have which help user manage time effectiveness, boost up their productivity and measure efficiently in doing outdoor activities.

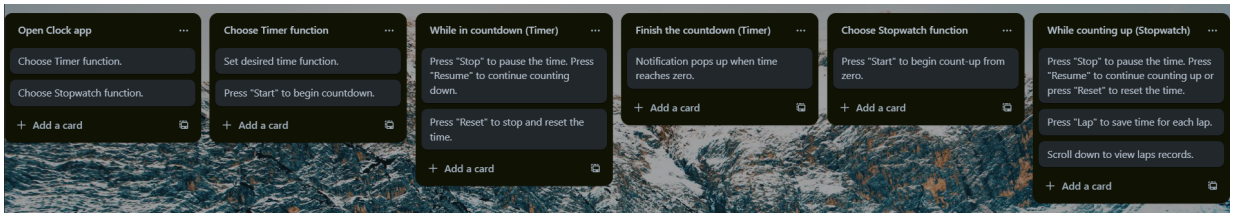


FIGURE 1.9: Timer and Stopwatch

1.3.2 GPS Tracking

In order to move around easily, we also choose to implement another obvious and necessary functionalities of a smartwatch which is route tracking and directions.

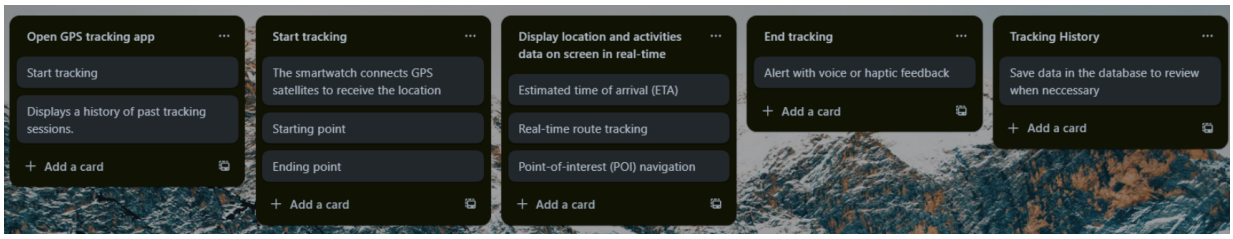


FIGURE 1.10: GPS Tracking

1.3.3 Calling application

If users want to contact someone, our watch gives users an application to call.

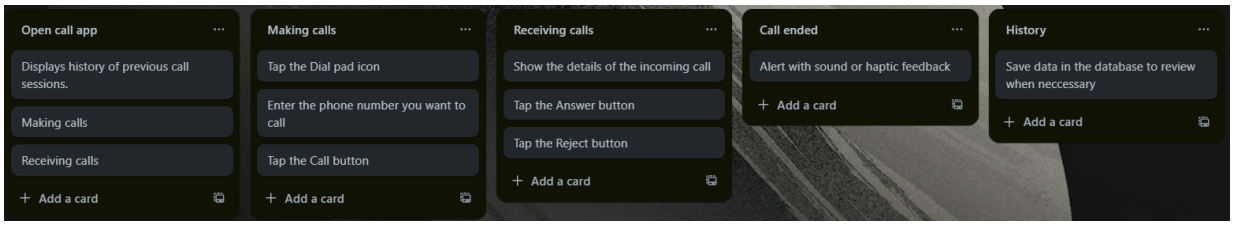


FIGURE 1.11: Calling Function

1.3.4 Weather Display

Talking about weather, our app has a simple UI and user-friendly design. Providing with accurate updates weather info, including alerts about potential hazards. With this function, user can prepare appropriately before going anywhere.

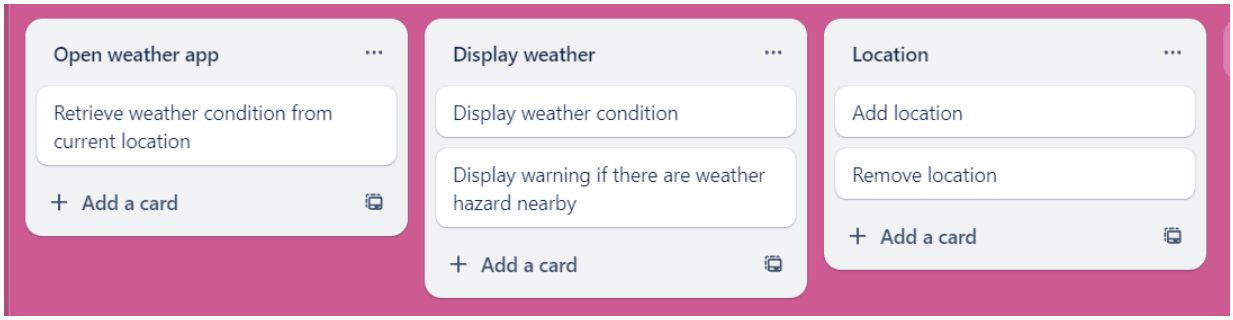


FIGURE 1.12: Weather Display

1.3.5 Note

The Note App is a user-friendly smartwatch application that help users to take note with ease and reminders with real-time alerts.

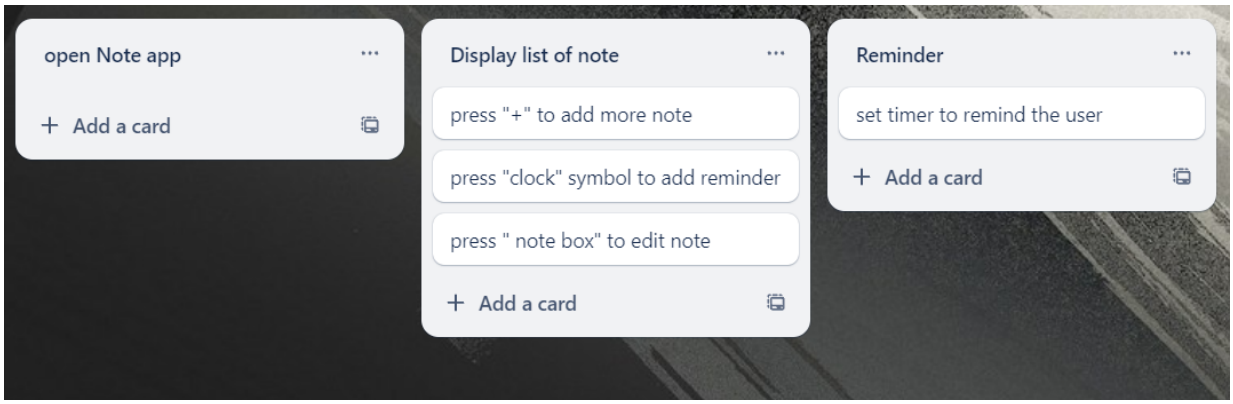


FIGURE 1.13: Note

1.3.6 Fitness Tracking

On the topic of exercising to improve health condition, we also think it is relevant that a smartwatch should need an app to record everything when users doing exercise so that they can keep track of the physical progress.

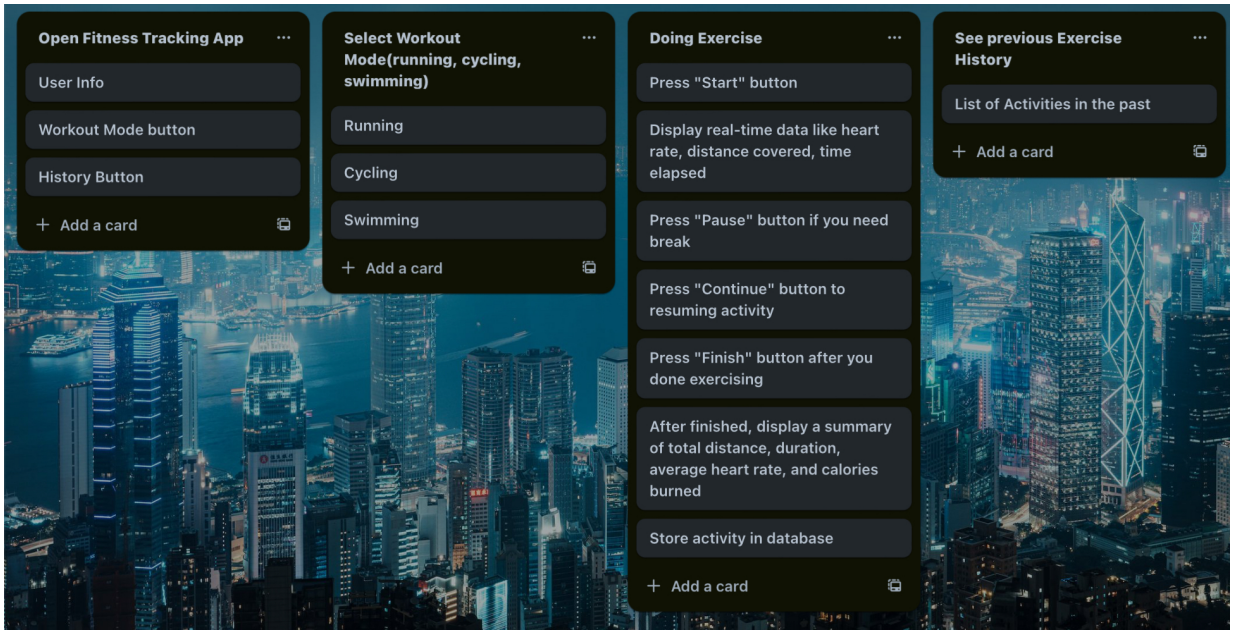


FIGURE 1.14: Fitness Tracking

1.3.7 Sleep Tracking

In order for users to get a better sleep, our smartwatch offers a dedicated sleep tracking feature which can improve their sleep quality overtime.

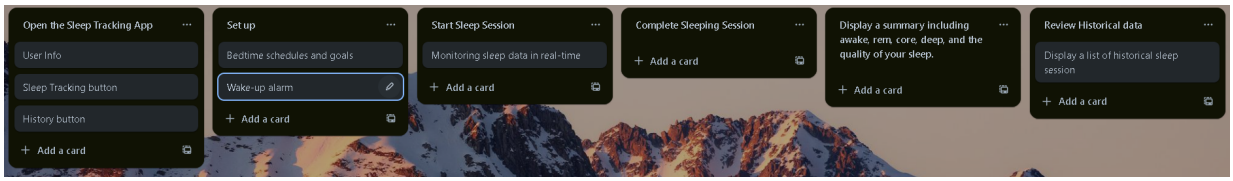


FIGURE 1.15: Sleep Tracking

1.3.8 Sightless Support

Other than essential features , our smartwatch also aim to aid people with eye-related health problems. Our goal is to help our users to better understand their surrounding environment, even if they have visual disadvantages.

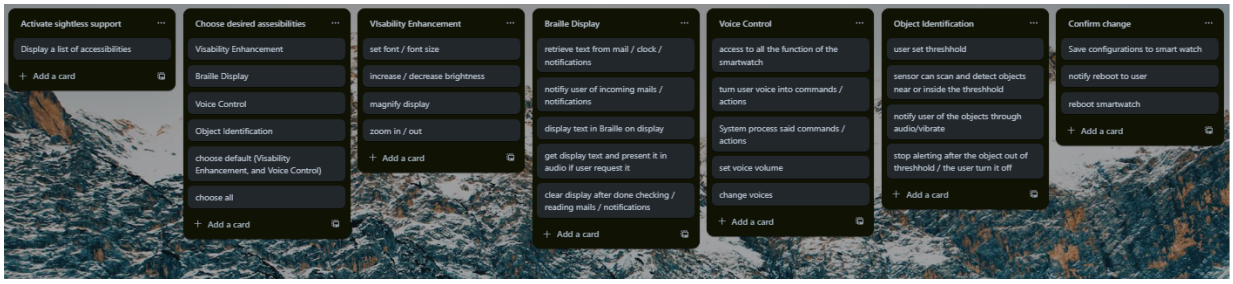


FIGURE 1.16: Sightless Support

This is the list of all functions that our team has added into the project.

Chapter 2

Backlog item

	Title	Actor	Category	Sprint	Member
Timer and Stopwatch	Timer	User	Functional	2	Dang Hoang Anh Khoi
	Stopwatch	User	Functional	4	Dang Hoang Anh Khoi
GPS Tracking	GPS Tracking	User	Functional	2	Dao The Hien
	Review Past Activities	User	Functional	2	Dao The Hien
Calling Application	Calling Application	User	Functional	4	Dao The Hien
	Review Call History	User	Functional	4	Dao The Hien
Weather Display	Weather Report	User	Functional	2	Truong Quang Minh
	Adjust Location	User	Functional	2	Truong Quang Minh
Note	Note	User	Functional	4	Truong Quang Minh
	Reminder	User	Functional	4	Truong Quang Minh

Fitness Tracking	Fitness Tracking	User	Functional	2	Nguyen Truc Linh
	Review Past Fitness Activities	User	Functional	2	Nguyen Truc Linh
Sleep Tracking	Sleep Tracking	User	Functional	4	Nguyen Truc Linh
	Review historical sleep data	User	Functional	4	Nguyen Truc Linh
Sightless Support	Activate Sightless Support	User	Functional	2	Pham Phi Long
	Default Setting	User	Functional	2	Pham Phi Long
	Deactivate Sightless Support	User	Functional	2	Pham Phi Long
Visibility Enhancement	Set Brightness	User	Functional	3	Pham Phi Long
	Set Font	User	Functional	3	Pham Phi Long
	Magnify	User	Functional	3	Pham Phi Long
Braille Display	Enable Braille Display	User	Functional	3	Pham Phi Long
Object Identification	Set Threshold	User	Functional	3	Pham Phi Long
Voice Control	Enable Voice Control	User	Functional	3	Pham Phi Long
	Perform Request	User	Functional	3	Pham Phi Long

Chapter 3

Timer

The Timer function is a versatile tool that simplifies time management. It can be used to create countdowns that are particular to our activities, such as cooking, working exercise, or concentrated work times. It is a feature that is simple to use and increases the smartwatches' overall effectiveness and adaptability for a variety of everyday activities. It operates in the background and has an user friendly interface.

3.1 Requirement Analysis

3.1.1 Snow cards

Requirements Type: Functional
For Whom: User
User Satisfaction: High
User Dissatisfaction: Low
Description: The "Ubugs Watch" timer feature improves users' overall experience by offering a practical and effective method of managing time. People may create customized countdowns using this function, which has easy-to-use settings and accurate timing capabilities. Whether it's for cooking, working out, or managing productivity, the timer feature makes time saving become easier and more flexible. It is an essential component of the smartwatch and increases its versatility by assisting users with a range of daily tasks thanks to its background operation and user-friendliness.

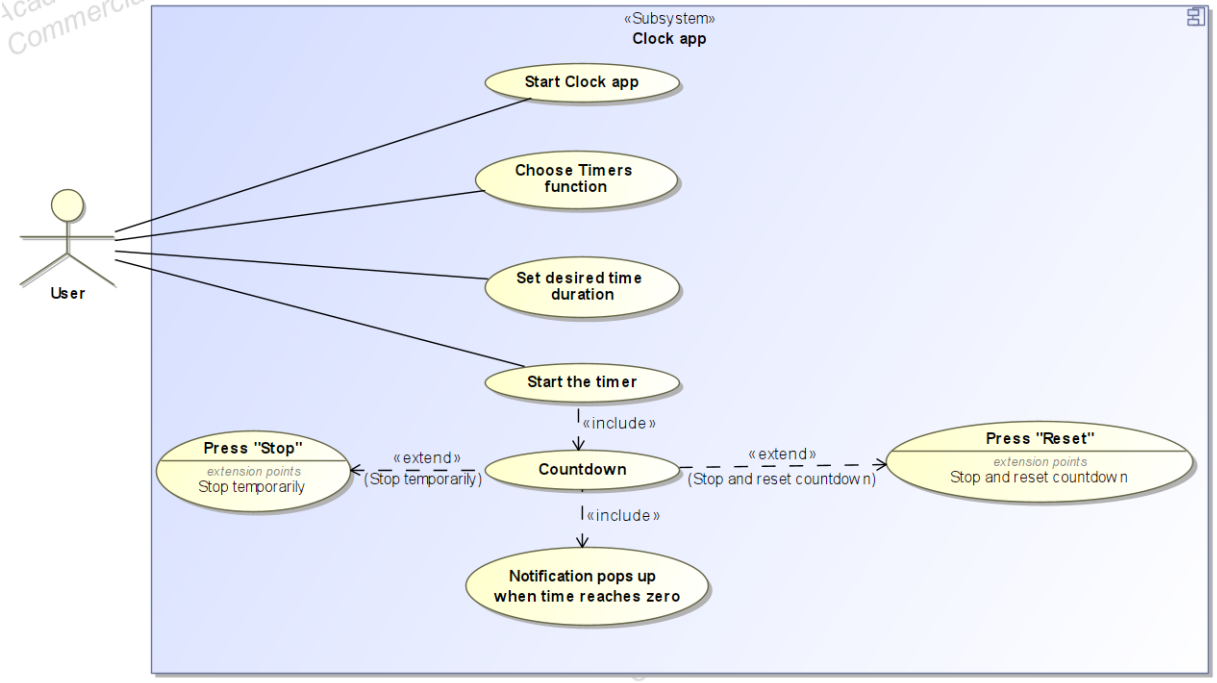
3.1.2 Use Case Analysis

Name	Timer
ID	1a
Description	The "Ubugs Watch" timer feature improves users' overall experience by offering a practical and effective method of managing time. People may create customized countdowns using this function, which has easy-to-use settings and accurate timing capabilities. Whether it's for cooking, working out, or managing productivity, the timer feature makes time saving become easier and more flexible. It is an essential component of the smartwatch and increases its versatility by assisting users with a range of daily tasks thanks to its background operation and user-friendliness.
Trigger	For time counting down: When the app is chosen, and Timer function is selected
Pre-conditions	The app is chosen, and Timer function is selected. User set desired time duration and press "Start".
Post-conditions	A notification and alarm appear if the time reaches zero.
Basic Flow	-
Description	This is the main scenarios when the users utilize the Timer function.

Actions	0. Users open Clock App. 1. Users then choose Timer function. 2. Users set desired countdown duration. 3. Users press "Start" to begin countdown. 4. If users want to break, they can press "Stop" to stop countdown temporarily and then press "Resume" to continue. 5. If users want to reset countdown, they can press "Reset" and then can press "Start" to start the countdown again. 6. After the countdown finishes, there will be a notification pops-up with an alarm rings.

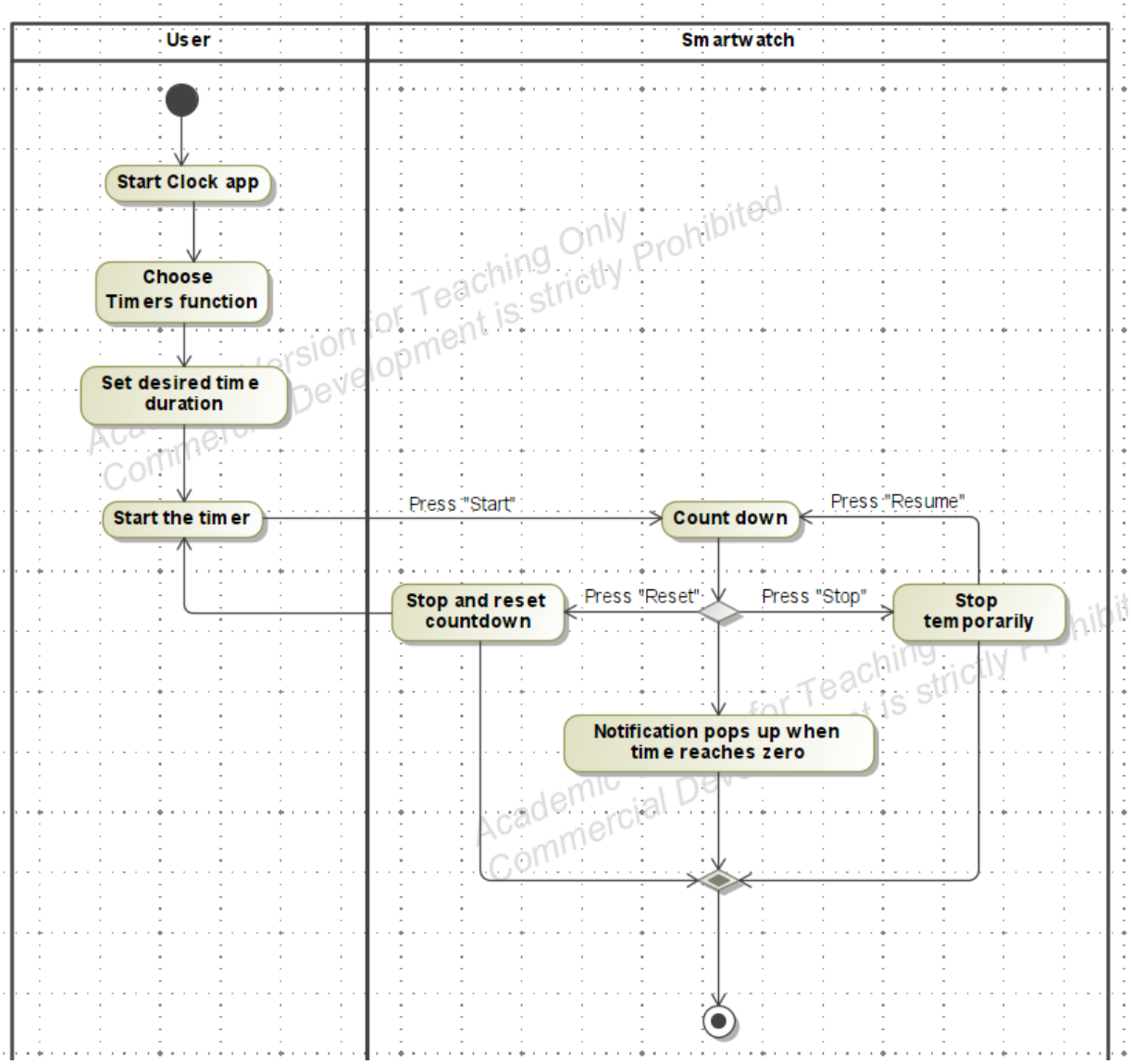
3.2 UML diagrams

3.2.1 Use Case Diagram



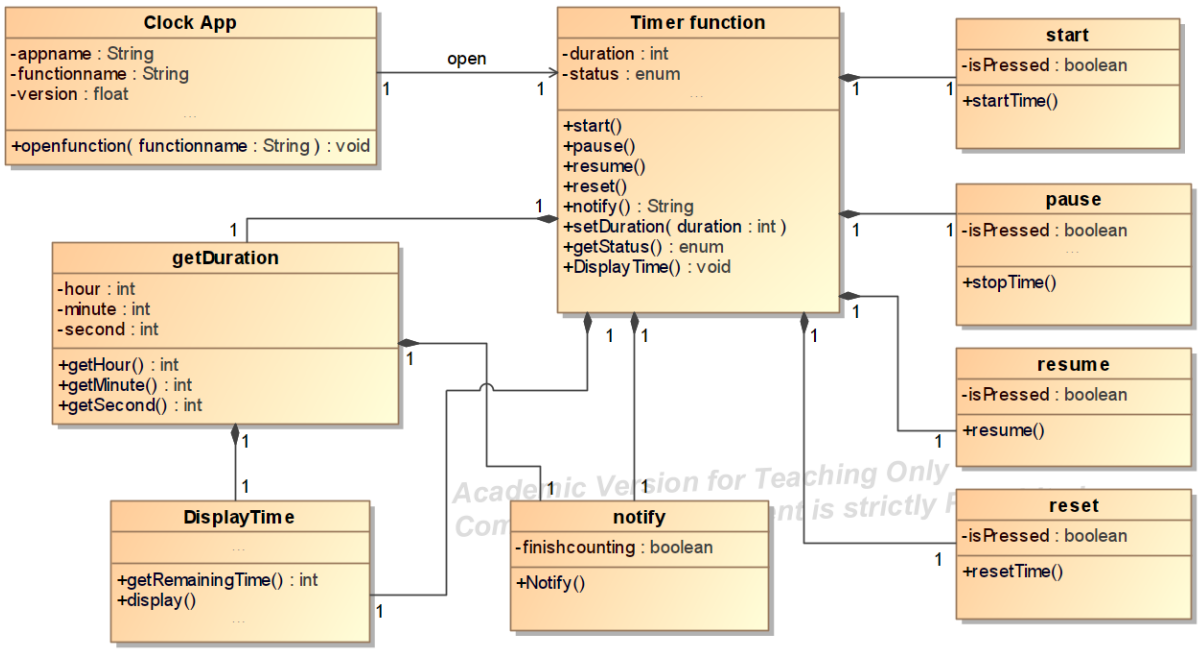
The use case diagram of Timer function describes many ways that actors and the system interact to accomplish particular time management functions. Actors, representing entities external to the system, could include users interacting with the timer. Use cases may involve actions like setting the timer, starting or pausing it, and receiving notifications when the timer expires. Relationships and associations between actors and use cases illustrate the communication paths such as starting Clock app, choosing Timer function, notifying, ... Additionally, the diagram may incorporate system boundaries and include any relevant extensions or variations of the Timer functionality like starting, pausing or resetting countdown. In general, the use case diagram offers an overview of how actor interacts with the system to accomplish tasks of the function.

3.2.2 Activity Diagram



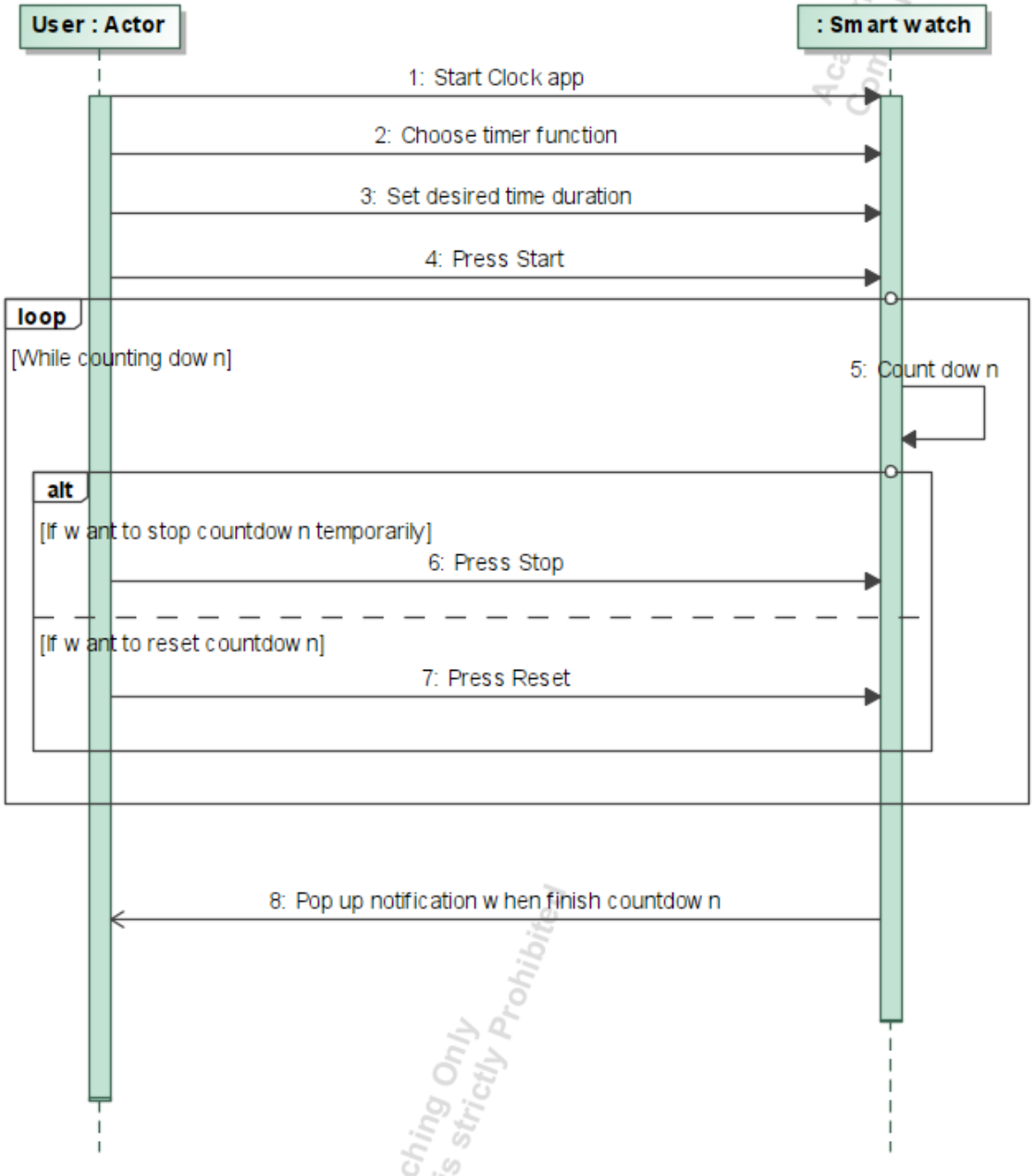
The activity diagram of Timer function describe the process of how the function works and it is divided into two fields that show which part of process is done by user or the system. It may begin with an initial node representing opening Clock apps, follows by activities such as access the Timer function, set duration, start the timer and finally ends with show notification when time reaches zero. Decision points is included to describe conditions while counting down such as pausing or resetting the timer. The figure presents the logical order of actions and decision points throughout the operation of timer's function.

3.2.3 Class Diagram



The diagram shows the relationships between classes, with arrows indicating a one-to-one compositions. The "Clock App" class represents the clock application, featuring properties like app name, functions in the Clock app, and the version of the app, along with an operation to open specific functions. The main class, “Timer function”, includes attributes for duration and status of the countdown. It also possesses other function classes that serve the changing counting status, display time and notify when time reaches zero. The methods and attributes of each class are designed to encapsulate the functionality of a clock app’s timer. For instance, the Timer function class has methods to start, pause, resume, and reset a timer, and also to set its duration and display the time. Together, these classes and their relationships form a clear blueprint of the Timer function system, outlining the structure and behavior of its components.

3.2.4 Sequence Diagram



The sequence diagram for the Timer function illustrates the dynamic interactions between objects during the execution of various operations. It begins with a message from the user to smartwatch to start the Clock app. After accessing Timer function and set desired time function, user presses "Start" to begin countdown. This triggers a sequence of conditional messages exchanged between smartwatch and user while system is on counting down, like pressing Stop or Reset, and finally end with the message from smart watch to user to notify that the countdown has finished. The diagram visually portrays the chronological order of events, providing a step-by-step depiction of

how different objects collaborate and communicate in realizing the functionalities of the Timer function.

3.2.5 UI Prototype

This is the user interface of smart watch when using Timer function. The first screen describes what happens when user starts the Timer function, there will be a stopping clock which user can scroll up and down to adjust and set desired time duration. User then clicks “Start” button and the second screen of the UI will come up, which shows that the clock is counting down. At this time, user can either reset the countdown by pressing “Reset” and get back to first screen or pause the countdown by pressing “Stop”. When the countdown stops temporarily, the “Stop” button will be replaced with the “Resume” button which user can press to continue the countdown. Overall, this user interface is very easy to use and adapt.

Chapter 4

Stopwatch

The stopwatch is an essential tool for keeping track of time. It may be used to quantify elapsed time during outdoor activities such as running laps, training sessions, or race timing. Its intuitive buttons with clear display makes it simple to operate, increasing the stopwatch's total utility and flexibility for daily activities.

4.1 Requirement Analysis

4.1.1 Snow cards

Requirements Type: Functional
For Whom: User
User Satisfaction: High
User Dissatisfaction: Low
Description: The stopwatch feature on the "Ubugs Watch" enhances user experience by providing a convenient and efficient means of measure the time. Users may simply start, stop, and reset the stopwatch, which allows for precise time measurement. Additionally, users may save laps can see the data to keep track of their productivity. Whether using timing laps during sports, fitness or doing exercises, the stopwatch function simplifies time tracking, making it more convenient and adaptable. As an essential component of the smartwatch, it increases adaptability by assisting users in various daily activities with smooth functioning and simplicity of use.

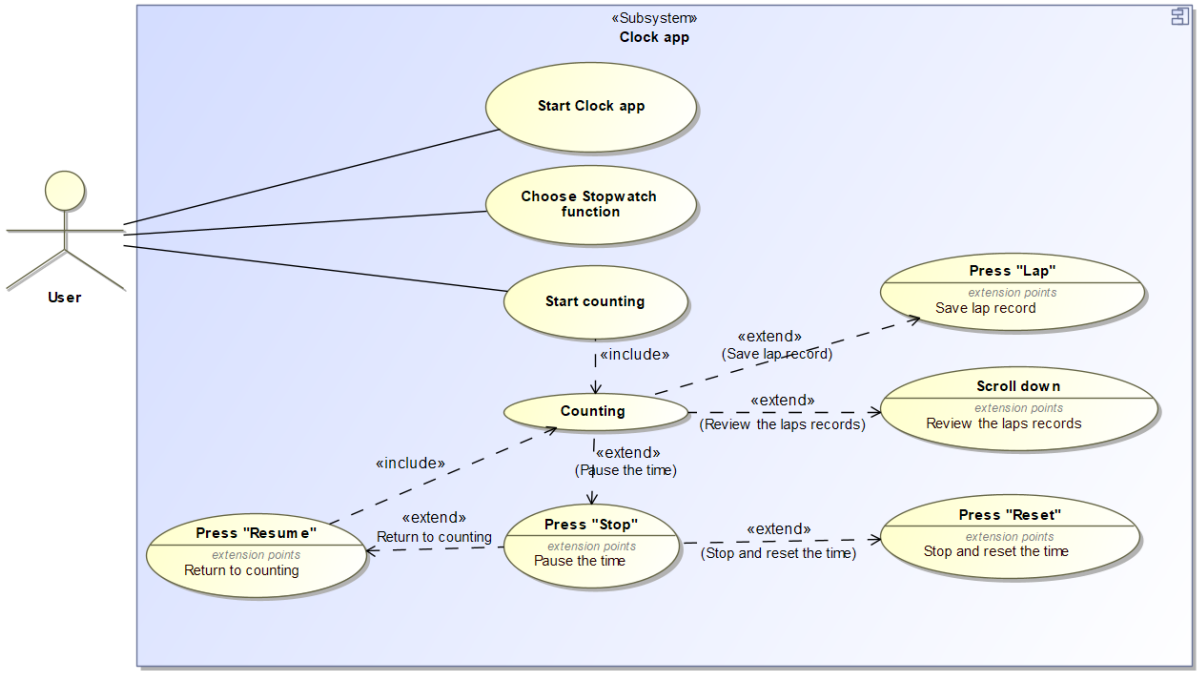
4.1.2 Use Case Analysis

Name	Stopwatch
ID	1b
Description	The stopwatch feature on the "Ubugs Watch" enhances user experience by providing a convenient and efficient means of measure the time. Users may simply start, stop, and reset the stopwatch, which allows for precise time measurement. Additionally, users may save laps can see the data to keep track of their productivity. Whether using timing laps during sports, fitness or doing exercises, the stopwatch function simplifies time tracking, making it more convenient and adaptable. As an essential component of the smartwatch, it increases adaptability by assisting users in various daily activities with smooth functioning and simplicity of use.
Trigger	For time tracking: When the app is chosen, and Stopwatch function is selected
Pre-conditions	The app is chosen, and Stopwatch function is selected. User press "Start".
Post-conditions	Scroll down to view the laps records
Basic Flow	-
Description	This is the main scenarios when the users utilize the Stopwatch function.

Actions	0. Users open Clock App. 1. Users then choose Stopwatch function. 2. Users press "Start" to begin time elapsed. 3. While counting, user can press "Lap" to save lap's time. The lap is then reset to begin counting new lap. 4. If users want to break, they can press "Stop" to stop counting temporarily. 5. While stopping, user can press "Resume" to continue counting time. Other wise, user can press "Reset" to reset the time. 6. In any time, user can scroll down to view all laps records.

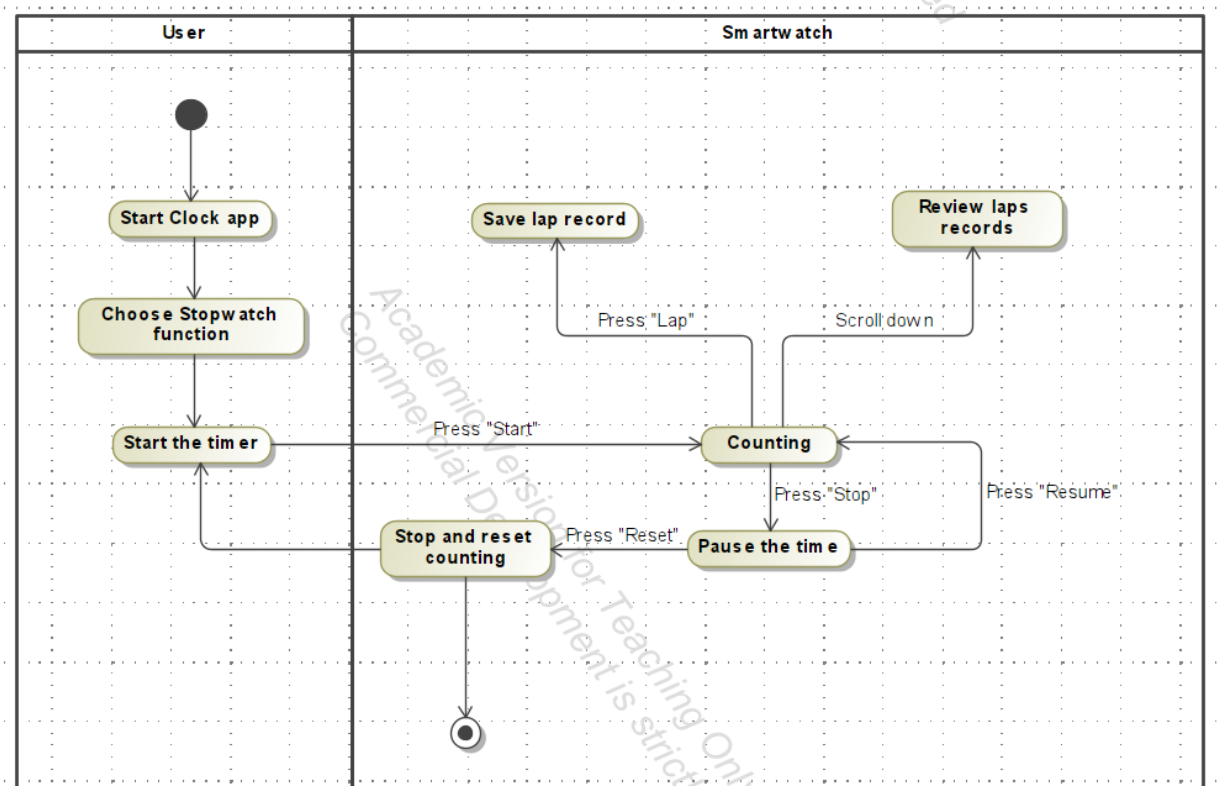
4.2 UML diagrams

4.2.1 Use Case Diagram



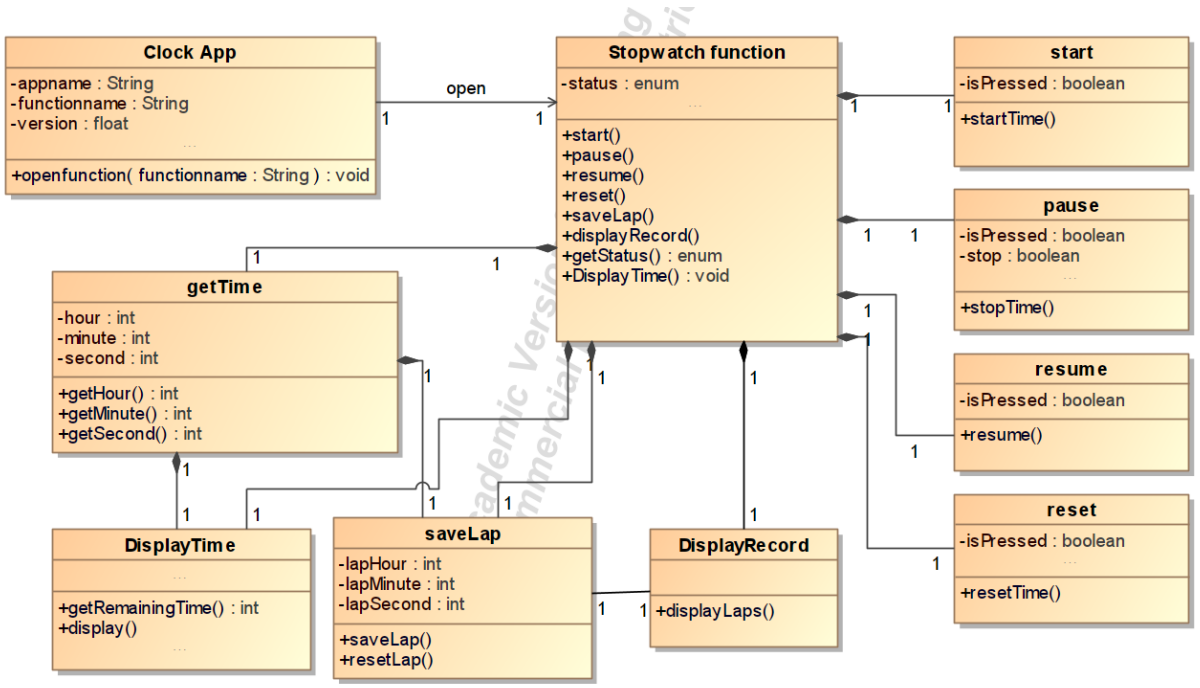
The use case diagram of Stopwatch function describes many ways that actors and the system interact to accomplish particular time tracking functions. Actors, representing entities external to the system, could include users interacting with the function. Use cases may involve actions like pressing start, stop, resume and reset button, saving lap and scrolling down to view the laps. Relationships and associations between actors and use cases illustrate the communication paths such as starting Clock app, choosing Stopwatch function, reviewing the laps, ... Additionally, the diagram may incorporate system boundaries and include any relevant extensions or variations of the stopwatch functionality like starting, pausing, resuming, resetting counting or saving laps. In general, the use case diagram offers an overview of how actor interacts with the system to accomplish tasks of the function.

4.2.2 Activity Diagram



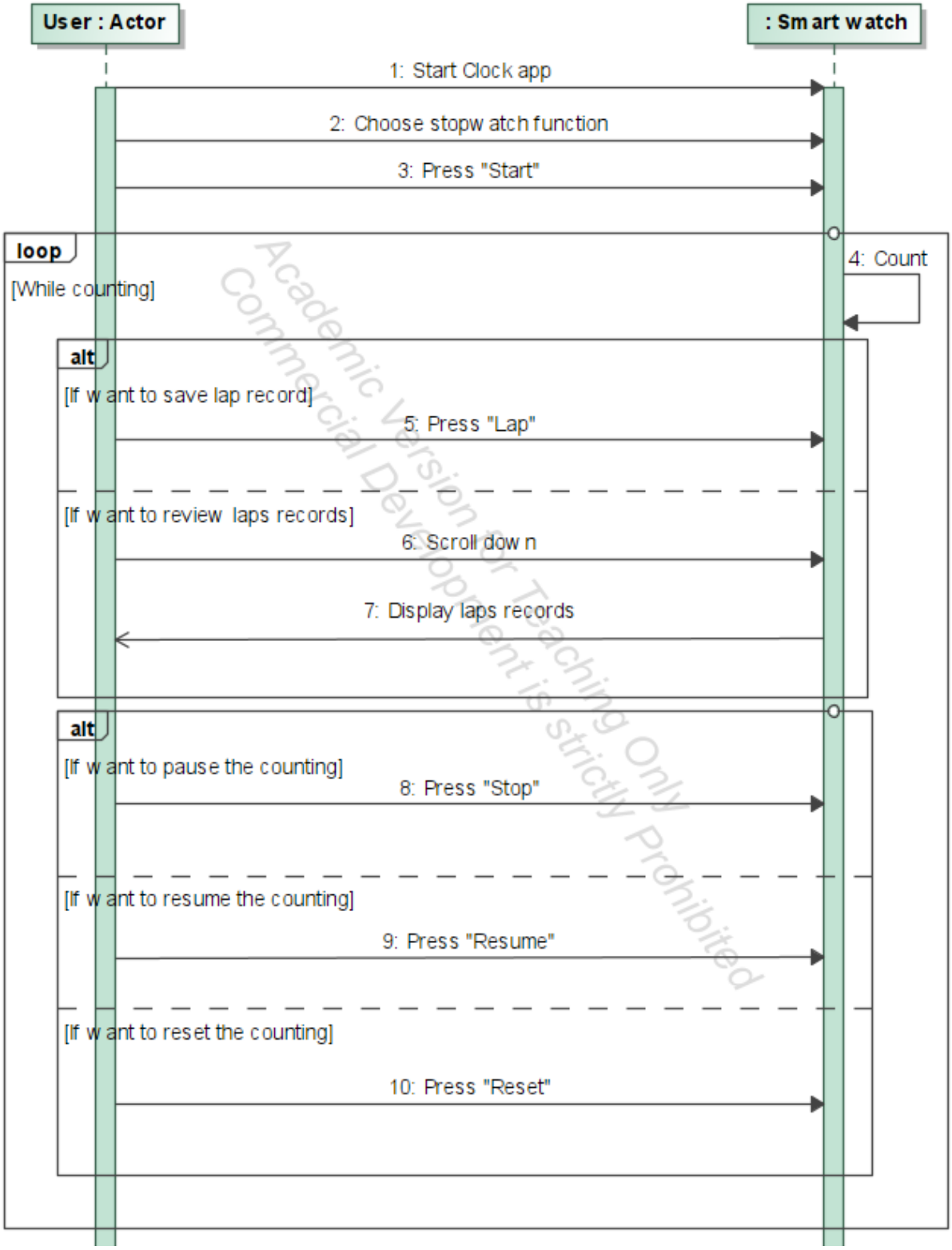
The activity diagram of Stopwatch function describes the process of how the function works and it is divided into two fields that show which part of process is done by user or the system. It may begin with an initial node representing opening Clock apps, follows by activities such as access the Stopwatch function, start the time and finally ends with optional stop and reset the counting. The figure presents the logical order of actions and decision points throughout the operation of Stopwatch function.

4.2.3 Class Diagram



The diagram shows the relationships between classes, with arrows representing a one-to-one compositions. The "Clock App" class represents the clock application, and it includes attributes like as the app's name, functions, and version, as well as an action to open certain functions. The primary class, "Stopwatch Function", includes attributes for status of the counting. It also includes various function classes that change the counting state, display the time, and add and display records when the user presses the Lap button to save the lap record. The methods and attributes of each class are designed to encapsulate the functionality of a clock app's stopwatch. For instance, the Stopwatch Function class includes methods to start, pause, resume, reset time, save and show laps and display the time.

4.2.4 Sequence Diagram



Actors, representing entities external to the system, could include users interacting with the stopwatch. Use cases may include operations such as pushing the start, stop, resume, and reset buttons, storing laps, and scrolling down to see them. Relationships and associations between actors and use cases illustrate the communication paths such as starting Clock app, choosing Stopwatch function, reviewing the laps, ... Additionally, the diagram may incorporate system boundaries and include any relevant extensions or variations of the stopwatch functionality like starting, pausing, resuming, resetting counting or saving laps. In general, the use case diagram offers an overview of how actor interacts with the system to accomplish tasks of the function.

4.2.5 UI Prototype

This is the user interface of smart watch when using Stopwatch function. The first screen describes what happens when user starts the Stopwatch function, there will be a clock which user can click "Start" button and the second screen of the UI will come up, which shows that the time is running. At this time, the user may either pause the counting by hitting "Stop" and get to the third screen, or save the lap record by pressing "Lap". When the time stops, the "Stop" button is replaced by the "Resume" button which user may push to continue the elapsing and the "Lap" button will be replaced by the "Reset" button which user can press to reset the time. Additionally, user may scroll up and down to view the records for each saved lap. Overall, this user interface is very simple to use and adapt.

Chapter 5

GPS Tracking

GPS tracking application is a common and necessary function that can help users with navigation when they need to get to any location. The smartwatch will records the event and display them to users.

5.1 Requirement Analysis

5.1.1 Snow cards

Requirements Type: Functional
For Whom: Customer
User Satisfaction: High
User Dissatisfaction: Low
Description: The GPS Tracking feature inside the "Ubugs Watch" smartwatch gives customers a course monitoring and steering enjoy. It gathers real-time information using algorithms and correct satellite connectivity, then makes use of analytics to direct users to their favored locations. Estimated time of arrival (ETA), real-time route tracking, and point-of-interest (POI) navigation may also all be computed. To make it simpler for customers to observe the path at the same time as touring, all relevant data will also be proven at the display. Additionally, the information may be updated often due to the satellite link. In addition, the smartwatch establishes a easy reference to a database, giving customers access to a wealth of historical records that allows them to replay past trips.

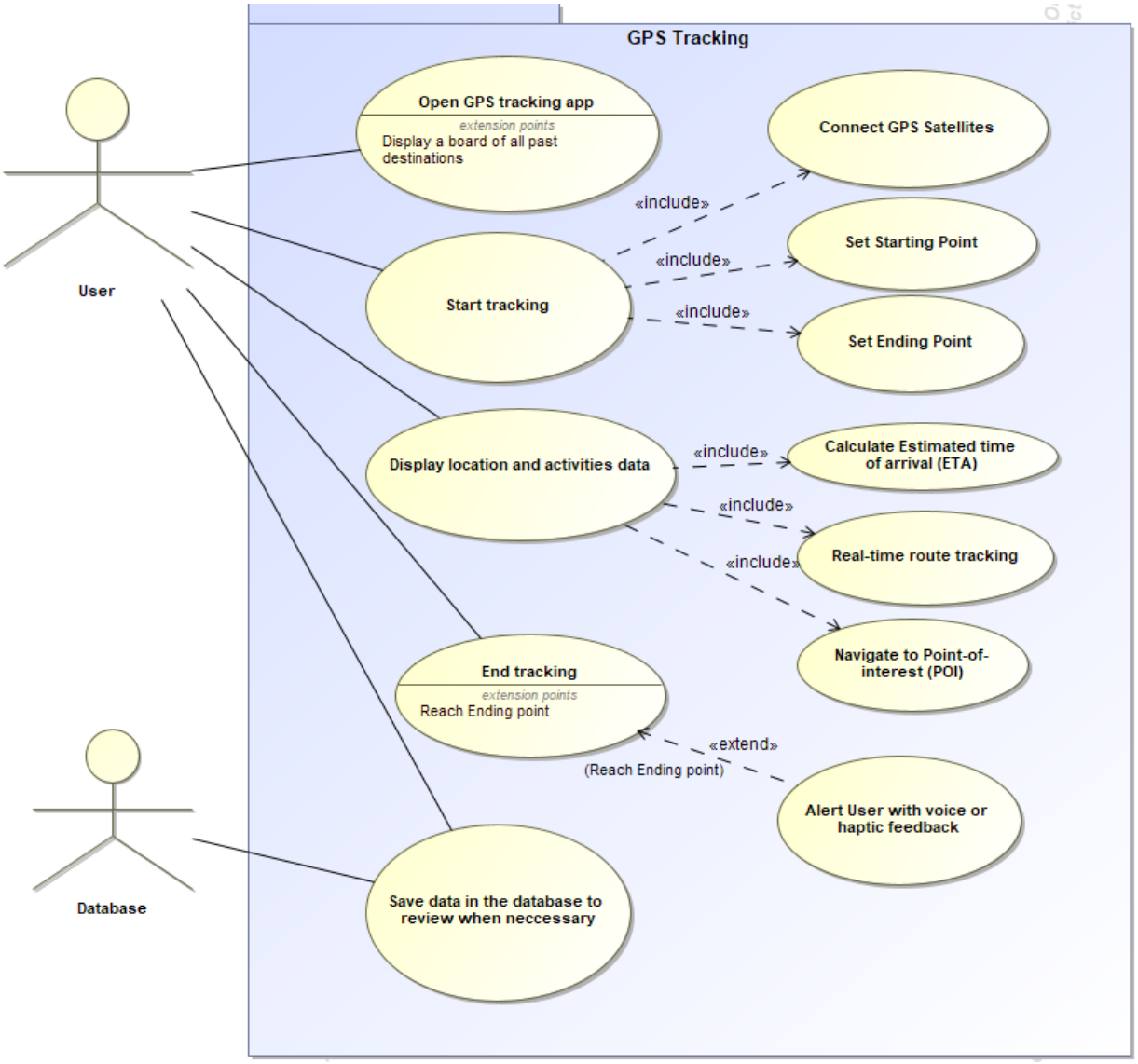
5.1.2 Use Case Analysis

Name	GPS Tracking
ID	2
Description	The GPS Tracking feature inside the "Ubugs Watch" smartwatch gives customers a course monitoring and steering enjoy. It gathers real-time information using algorithms and correct satellite connectivity, then makes use of analytics to direct users to their favored locations. Estimated time of arrival (ETA), real-time route tracking, and point-of-interest (POI) navigation may also all be computed. To make it simpler for customers to observe the path at the same time as touring, all relevant data will also be proven at the display. Additionally, the information may be updated often due to the satellite link. In addition, the smartwatch establishes a easy reference to a database, giving customers access to a wealth of historical records that allows them to replay past trips.
Trigger	For start tracking: After the smartwatch is worn, the app is chosen, the app has connected to the satellite and specific "Start tracking" is select
Pre-conditions	For GPS tracking: After the smartwatch is worn and the app is chosen. For real-time data monitoring: After user choose specific new tracking mode, the app has connected to the satellite and press "Start".
Post-conditions	Sends alerts via voice or haptic feedback when reaching a Point-of-interest.
Basic Flow	-

Description	This is the main scenarios when the user choose to track.
Actions	0. Users open the GPS Tracking App. 1. They then press "Start tracking" button. 2. After pressing "Start tracking", The watch will connect to satellites to get real-time location data and calculate Estimated Time of Arrival (ETA), real-time route tracking, Point of Interest (POI) navigation 3. The app notifies the user with either haptic or audio feedback when they reach at their destination. Alternatively they might choose to "end tracking" to terminate the tracking session.
Alternative Flow	-
Description	User want to see past activities
Actions	0. Users open the GPS Tracking App. 1. Then they press the history button 2. A list of past tracking sessions is displayed

5.2 UML diagrams

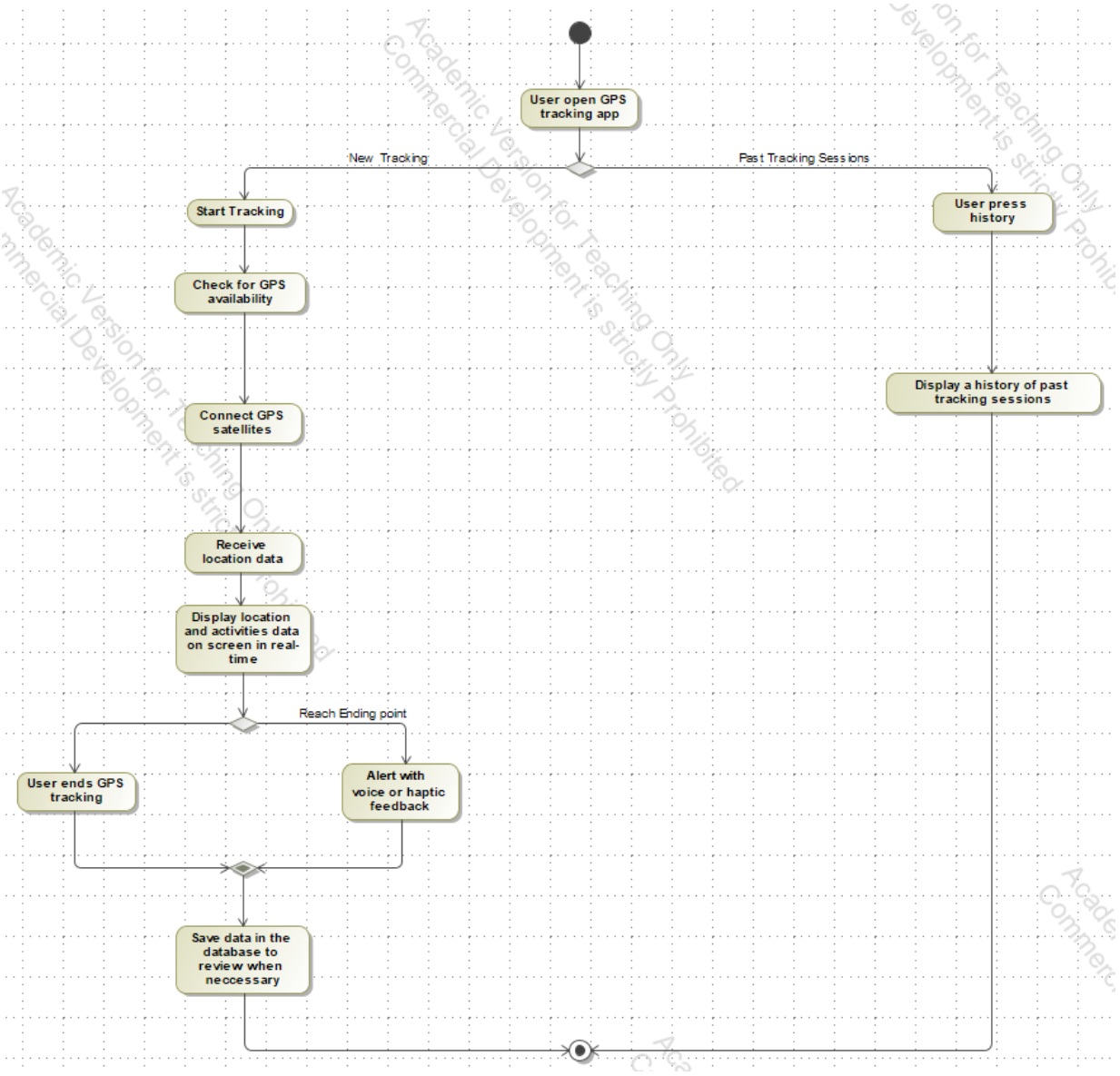
5.2.1 Use Case Diagram



In this UML use case diagram, we have a look at the capability of a GPS monitoring. The primary actor is the 'User,' but there's additionally an external 'Database' device engaged. Consumers initiate tracking by launching the GPS tracking app and deciding on or entering their selected locations. This begins the hyperlink with GPS satellites. Once related, the person may additionally start monitoring, consisting of specifying the start and finishing locations for their direction. The 'include' arrows advise that placing starting and finishing factors is an essential stage inside the process of setting out monitoring. Tracking permits users to look at place and activity facts in real time. The system determines the estimated time of arrival (ETA) for every point of interest (POI) alongside the course. Real-time course tracking updates customers on their present day region and progress to their aim. When users arrive at a factor of interest or destination,

they get speech or haptic feedback indicators. The 'extend' arrow from 'End Tracking' to 'Alert User with Voice or Haptic Feedback' denotes an non-obligatory step that expands the importance of ending monitoring. Before achieving the finish point, users can choose to stop tracking. All information accumulated in the course of the complete adventure can be stored in a database for future reference.

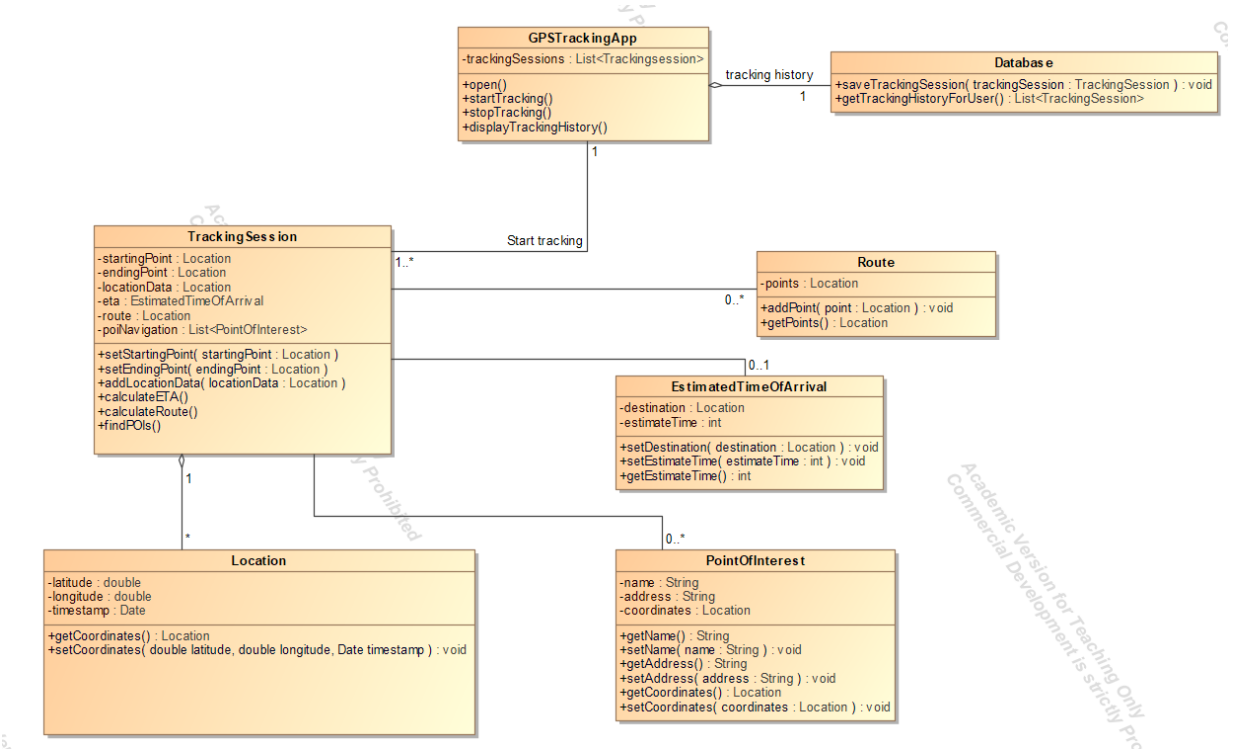
5.2.2 Activity Diagram



In my Activity Diagram, it represents the consumer's adventure from beginning the app to accomplishing their purpose. Open the GPS monitoring app. When the consumer launches the app, he or she is given the selection of beginning a new monitoring consultation or viewing previous monitoring sessions. If the person chooses to check the history, the app will provide a list of preceding tracking classes, inclusive of date, duration, and direction. Start a new monitoring session. The person chooses the "start

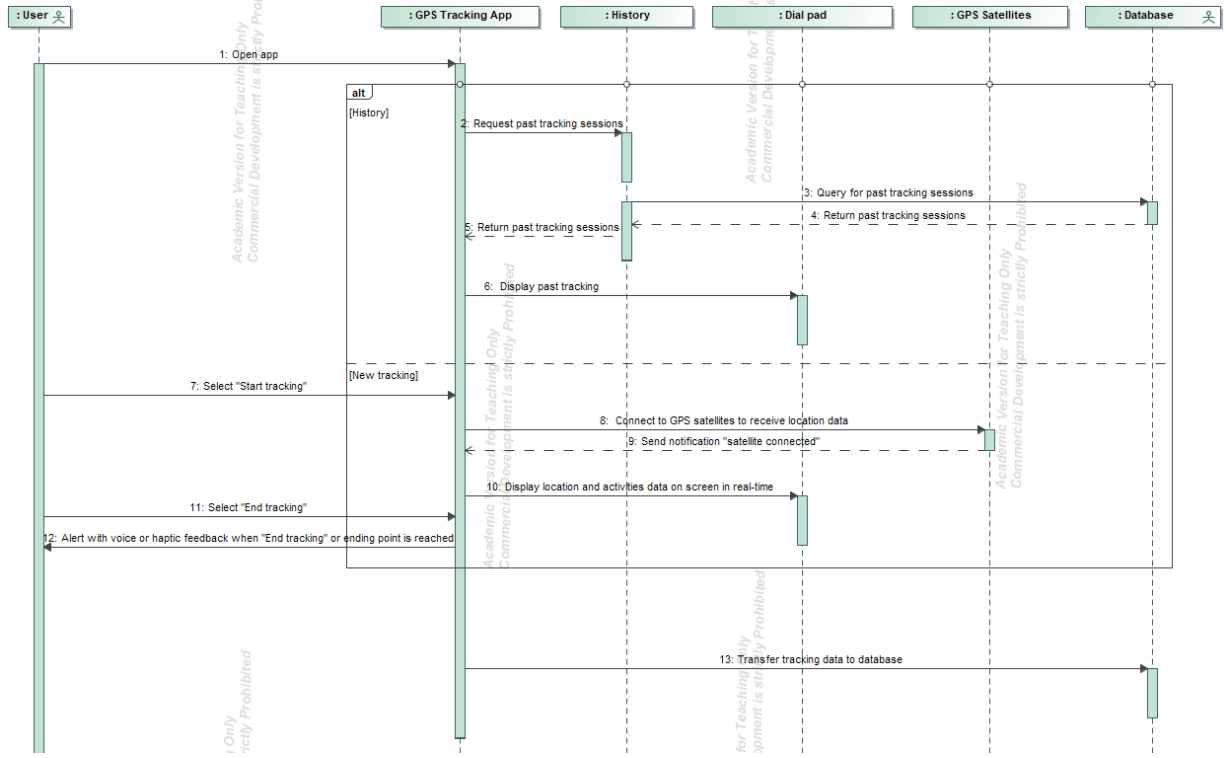
tracking" choice, and the program tests for GPS availability. If GPS is available, the app makes use of GPS satellites to acquire vicinity information. Receive and display area facts: The app gets place records from satellites and suggests it on the screen in actual time. Other facts that the app can offer include the estimated time of arrival (ETA), real-time direction tracking, and factor-of-hobby (POI) navigation. End tracking consultation: When the consumer arrives at their vacation spot, the app supplies an alert via voice or haptic comments. Alternatively, users can near the monitoring consultation by hitting the "end tracking" alternative. The software program will store the monitoring facts within the database for next evaluation.

5.2.3 Class Diagram



In my Class Diagram, the first one is the User class which includes location. Users can open the GPS Tracking app, which has the operation to open, startTracking, endTracking, and displayTrackingHistory, it also links to trackingSession when the user wants to start new tracking. The Database class includes saveTrackingSession and getTrackingHistoryForUser to store or display when needed by the user. The primary class that defines our application is TrackingSession which is the operation to setStartingPoint, setEndingPoint and it has three subclasses: Route, EstimatedTimeOfArrival, PointOfInterest to be able to calculate estimated time of arrival (ETA), point-of-interest (POI) navigation and real-time route tracking, which keeps users updated on their present location and progress to their goal. In addition, TrackingSession class will receive location data from Location class with latitude, longitude,and timestamp to provide more information to three subclasses. Lastly, all TrackingSession are stored in Database.

5.2.4 Sequence Diagram



In my Sequence Diagram, the customers will first open the GPS Tracking app. After that, they will have an option to choose, whether they want to start new tracking or see the past activities. If the users decide to start new tracking, they will first press the "Start tracking" button. Next, the app will connect to GPS satellites to receive location data, this does not have a reply message because this is often an asynchronous process. Once connected to the satellite, the application will collect information and display it on the screen in real-time. When the user arrives at their location, the app provides an alert with voice or haptic feedback. In addition, users can close the tracking session by pressing the "end tracking" option. And GPS tracking app will Transfer tracking data to database. On the other hand, if the users want to see the history board of all tracking history in the past, they can choose to open the History. It will request the history data from the database. After retrieving the data needed, it will display a list of past tracking sessions that the customers have done.

5.2.5 UI Prototype

The first UI Prototype is the home page of the app with two options for the user to choose include: Start tracking and History. The second one is the display of real-time location data including time of arrival, real-time route, and POI navigation. And the third one is the notifications when the user has reached the destination, the screen will display "Arrived" and at the same time, the app delivers an alert via voice or haptic feedback.

Chapter 6

Calling Application

The calling application is a popular and necessary function that helps users call anyone they want to contact in their contacts or phone number. This is an almost mandatory application on all smart watches.

6.1 Requirement Analysis

6.1.1 Snow cards

Requirements Type: Functional
For Whom: Customer
User Satisfaction: High
User Dissatisfaction: Low
Description: The Calling function on the smart watch "Ubugs Watch" enables users to call family, contacts, or new contacts. Users may make new calls, receive calls from others, and see their call history. It has algorithms that include particular functionality for those who are blind. They may use their voice to operate the above capabilities, such as reading the phone number instead of manually entering it and instructing the watch to accept or cancel calls. Today, all smart watches include a calling feature.

6.1.2 Use Case Analysis

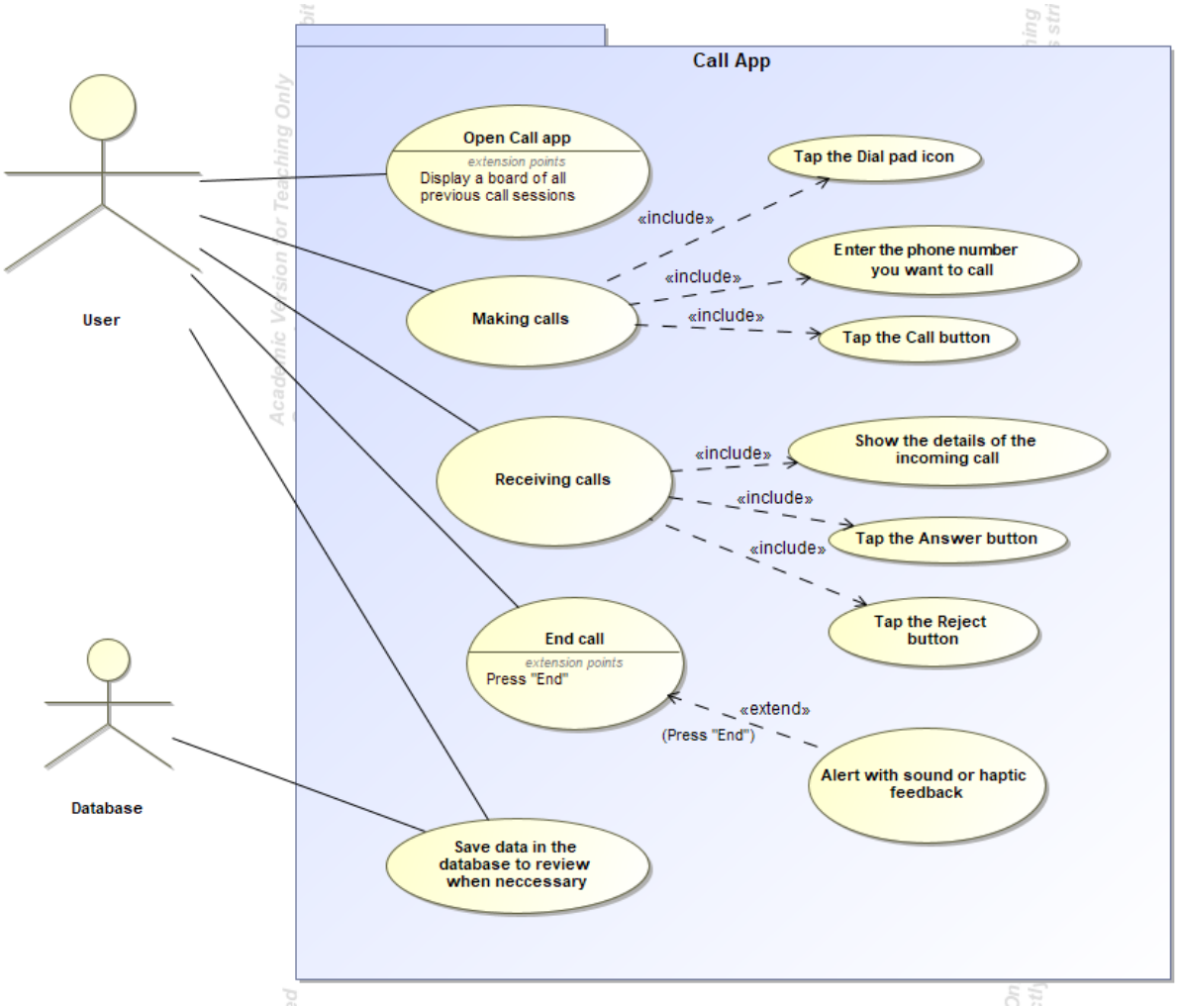
Name	Calling
ID	3
Description	The Calling function on the smart watch "Ubugs Watch" enables users to call family, contacts, or new contacts. Users may make new calls, receive calls from others, and see their call history. It has algorithms that include particular functionality for those who are blind. They may use their voice to operate the above capabilities, such as reading the phone number instead of manually entering it and instructing the watch to accept or cancel calls. Today, all smart watches include a calling feature.
Trigger	For start calling: After the smartwatch is worn, the app is chosen, the app prompts the user to input the phone number of the person they wish to call into the D-pad or choose it from the contact. They can push "Call" to initiate the call.
Pre-conditions	For Calling: After the smartwatch is worn and the app is chosen. For making call: After user choose specific new call, press "Call" and the application connects to the recipient's phone number.
Post-conditions	Send alert with sound or haptic feedback when the call is ended
Basic Flow	-
Description	This is the main scenarios when the user choose to call.

Actions	0. Users open the Calling App. 1. They then press "Dialpad" button or "Contact" button. 2. After you have entered the phone number or selected the phone number from the contact. press "Call" and the application connects to the recipient's phone number. 3. The app notifies the user with either haptic or audio feedback when the call is ended.
Alternative Flow	-
Description	User want to see call history
Actions	0. Users open the Calling App. 1. Then they press the "Call history" button 2. A list of past call sessions is displayed
Alternative Flow	-
Description	The user wants to accept an incoming call

Actions	0. Users open the Calling App. 1. The app show the details of the incoming call 2. Press the "Answer" button. The application connects to the caller. 3. The app notifies the user with either haptic or audio feedback when the call is ended.
Alternative Flow	-
Description	The user wants to reject an incoming call
Actions	0. Users open the Calling App. 1. The app show the details of the incoming call 2. Press the "Reject" button. 3. The application will return the user to the home screen.

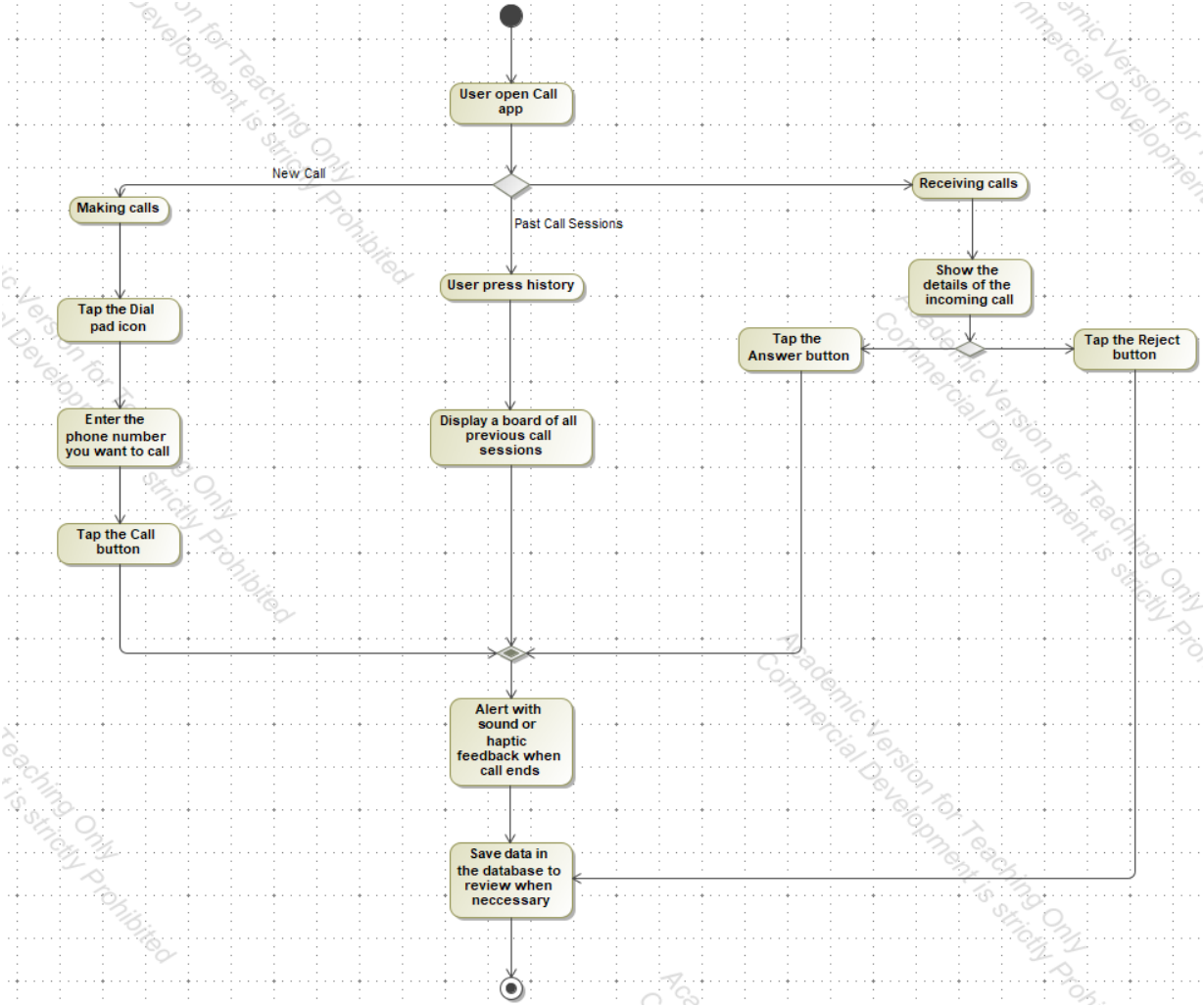
6.2 UML diagrams

6.2.1 Use Case Diagram



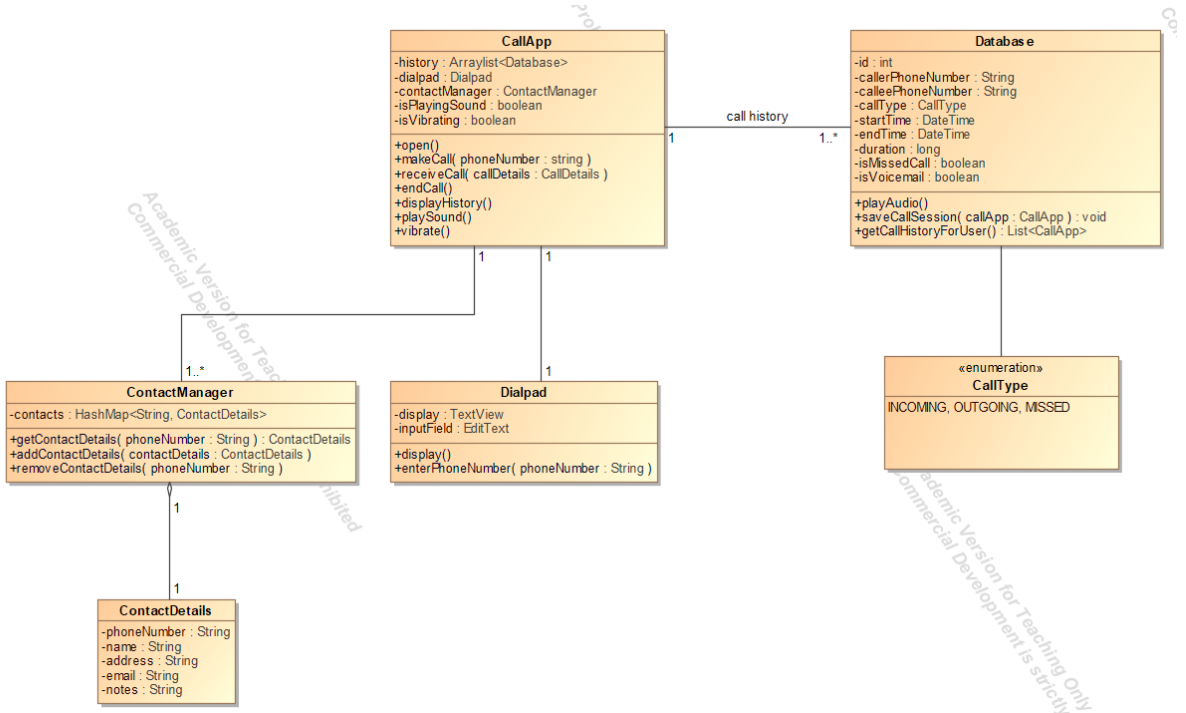
In this UML use case diagram, we look at the functionality of a Calling. The number one actor is the 'User,' but there may be also an outside 'Database' engaged. The person may additionally begin the app, with a purpose to show a listing of latest call sessions, after which press the dial pad icon to input a phone number or selected the phone number from the touch and region a call. The 'include' arrows recommend that Making calls and Receiving calls is an essential degree within the technique of starting up calling. The user might also accept incoming calls, which show the caller's facts, and reply or reject them. The consumer can prevent a name by way of clicking the give up button, to be able to offer an alarm with sound or haptic feedback. The software additionally retains the statistics from each call in a database, which the user may get admission to each time required.

6.2.2 Activity Diagram



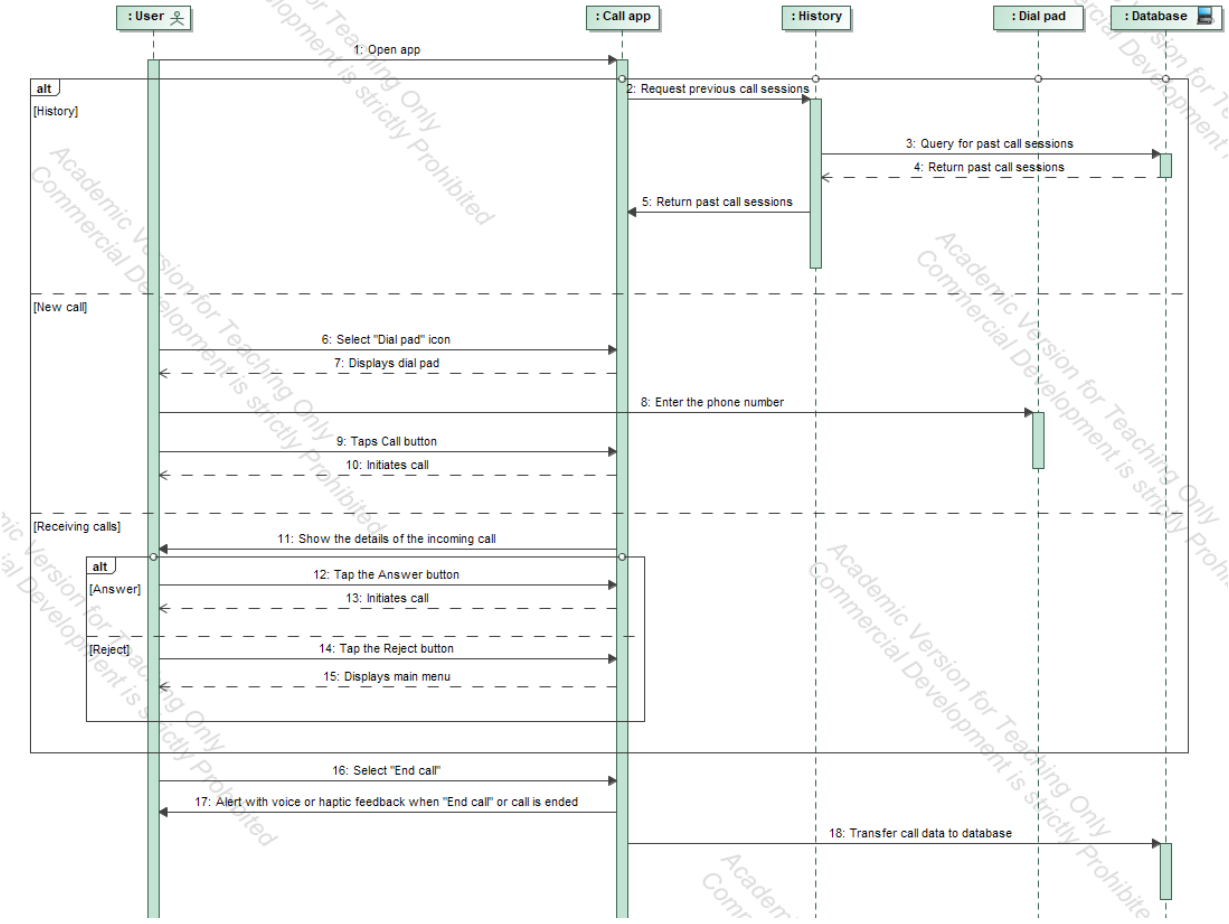
In my Activity Diagram, it represents the user's call from starting the app to reaching their goal. Open the Calling app. The depicts the procedures required in placing and receiving phone calls via a Call App. The diagram is divided into three sections: making calls, reviewing past call sessions, and receiving calls. Each section depicts the events and transitions that occur in various contexts. As an illustration, in Making Calls, the user hits the dial pad icon, inputs the phone number, and then presses the call button to begin the call. Receiving Calls displays the details of an incoming call and allows the user to select whether to answer or reject it. Finally, consumers may check call history. When you click on call history, the program displays all previous calls. All calls will be informed by sound or vibration when they terminate. The software will store the tracking data in the database for next review.

6.2.3 Class Diagram



The UML class diagram represents the structure and behavior of a call application system, which enables users to place and receive phone calls, manage contacts, and monitor call history. Users can open the Call app, which has the operation to open, makeCall, ReceiveCall, displayHistory, playSound and vibrate, it also links to ContactManager and Dialpad when the user wants to start new call. The Database class includes saveCallSession and getCallHistoryForUser to store or display when needed by the user. The primary class that defines our application is Dialpad which is the operation to display, enterPhoneNumber. In addition, ContactManager class will receive data from ContactDetails class with phoneNumber, name, address, email, and notes to provide more information. Lastly, i have CallType enumeration to show INCOMING, OUTGOING, MISSED.

6.2.4 Sequence Diagram



The UML sequence diagram illustrates how users interact with the application from the moment they open it and use the features to review call history, create a new phone call and receive incoming calls. It is split into five lifelines: User, Call app, History, Dial pad, and Database. The diagram also consists of a alternative and a label: History, New call, and Receiving call. Below are the stairs for every situation:

- **History**: The user opens the call app. The call app requests the history of the previous call sessions. The history queries for past call sessions. The database replies and returns past call sessions to the history. The history component returns past call sessions.

- **New call**: The user selects the dial pad icon on the call app. The call app displays the dial pad. The user enters the phone number they want to call on the dial pad. The user taps the call button on the dial pad. The dial pad initiates the call. The user ends the call. The call app alerts the user with voice or haptic feedback. The call app transfers the call data to the database.

- **Receiving call**: The call app shows the details of the incoming call to the user. The user can either tap the answer button or the reject button on the call app. If the

user taps the answer button, the call app accepts the call. If the user taps the reject button, the call app declines the call and returns the user to the main menu. The user ends the call. The call app alerts the user with voice or haptic feedback. The call app transfers the call data to the database.

6.2.5 UI Prototype

The first UI Prototype is the home page of the app including Contact, Call history, and Dialpad. The second one appears when a call comes in and displays the caller's details. You have two choices: accept or deny the call. The last thing is the dialpad where you can enter other people's phone numbers.

Chapter 7

Weather Display

The Weather app is an universal application designed to provide users with real-time updates on weather conditions. Assisting their planning of where to go in the near future.

7.1 Requirement Analysis

7.1.1 Snow cards

Requirements Type: Functional
For Whom: Customer
User Satisfaction: High
User Dissatisfaction: Low
Description: The weather app retrieves live data from weather.com to provide accurate and up-to-date information. It displays essential details, including temperature, current weather status (cloudy, rainy, snowy), time, and location, all in a simple-to-use interface. Additionally, users can add or remove locations, aiding them in planning their upcoming destinations in the near future with ease and flexibility. Moreover, a distinctive function is alerting the user if there is any weather hazard nearby to ensure their safety.

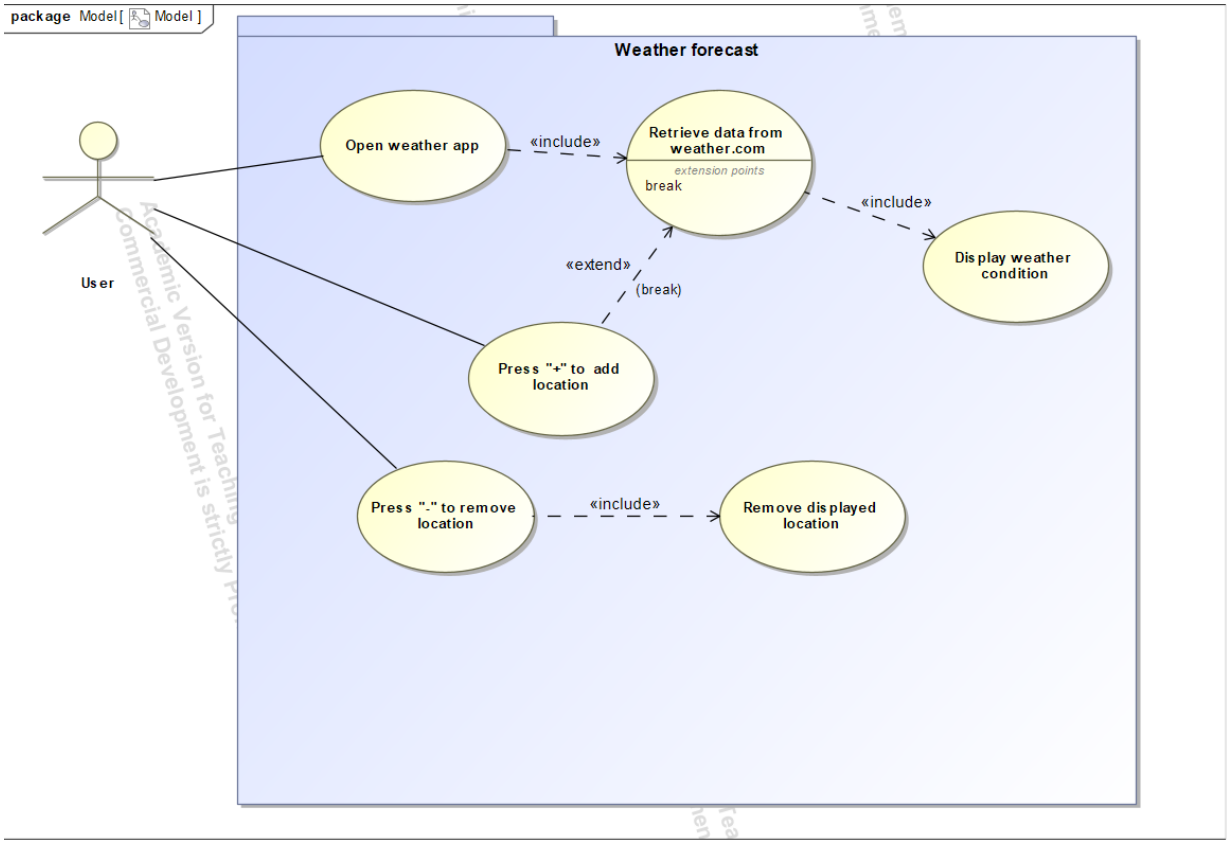
7.1.2 Use Case Analysis

Name	Weather
ID	4
Description	The weather app retrieves live data from weather.com to provide accurate and up-to-date information. It displays essential details, including temperature, current weather status (cloudy, rainy, snowy), time, and location, all in a simple-to-use interface. Additionally, users can add or remove locations, aiding them in planning their upcoming destinations in the near future with ease and flexibility. Moreover, a distinctive function is alerting the user if there is any weather hazard nearby to ensure their safety.
Trigger	For checking weather: Activate the app by wearing the smartwatch, open the app.
Pre-conditions	For checking weather: Activate the app by wearing the smartwatch, open the app.
Post-conditions	The data of temperature, weather statuses, time, and location.
Basic Flow	-
Description	This is the main scenarios when the user choose to check the weather.
Actions	0. Users open the Weather App. 1. The watch will display real-time data like temperature, weather statuses, time, and location.
Alternative Flow	-
Description	User want to adjust locations

Actions	0. Users open the Weather App. 1. They can press "+" button to add location they want to check the weather. 2. They can press "-" button to remove displayed location.

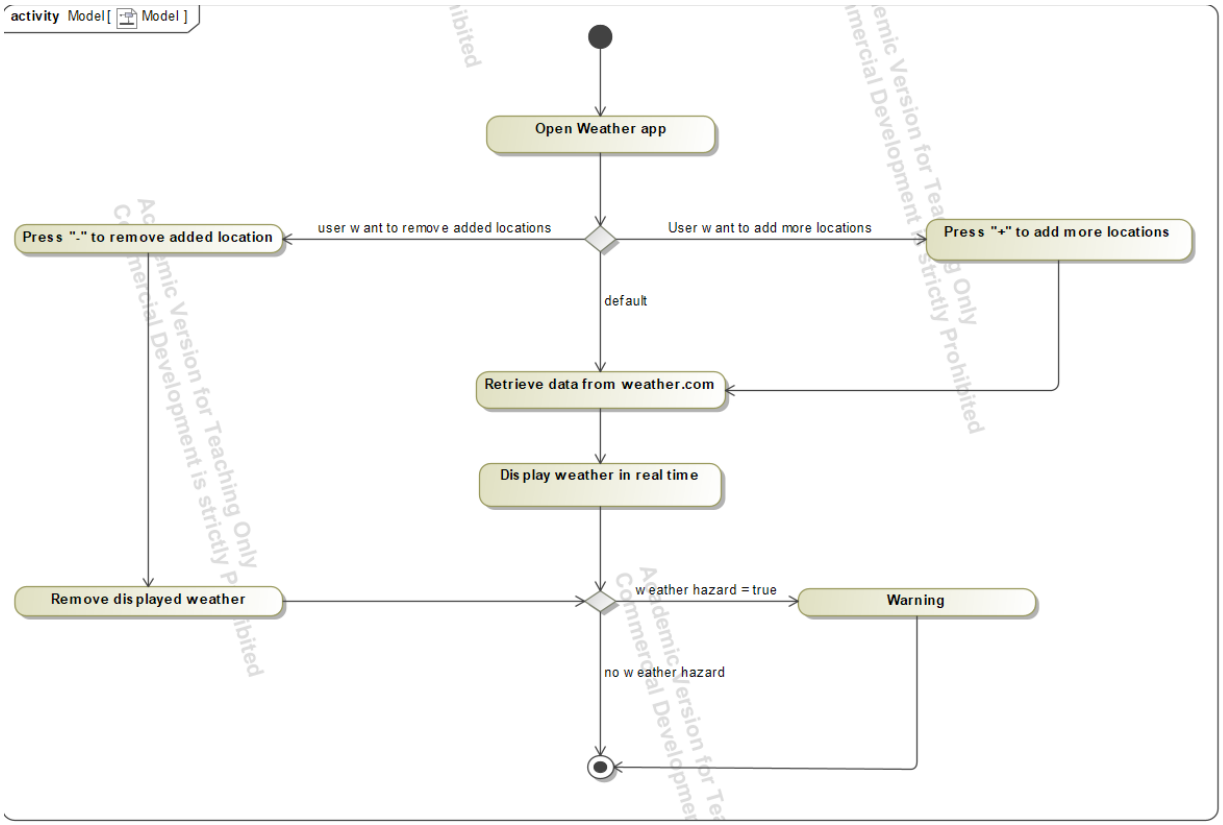
7.2 UML diagrams

7.2.1 Use Case Diagram



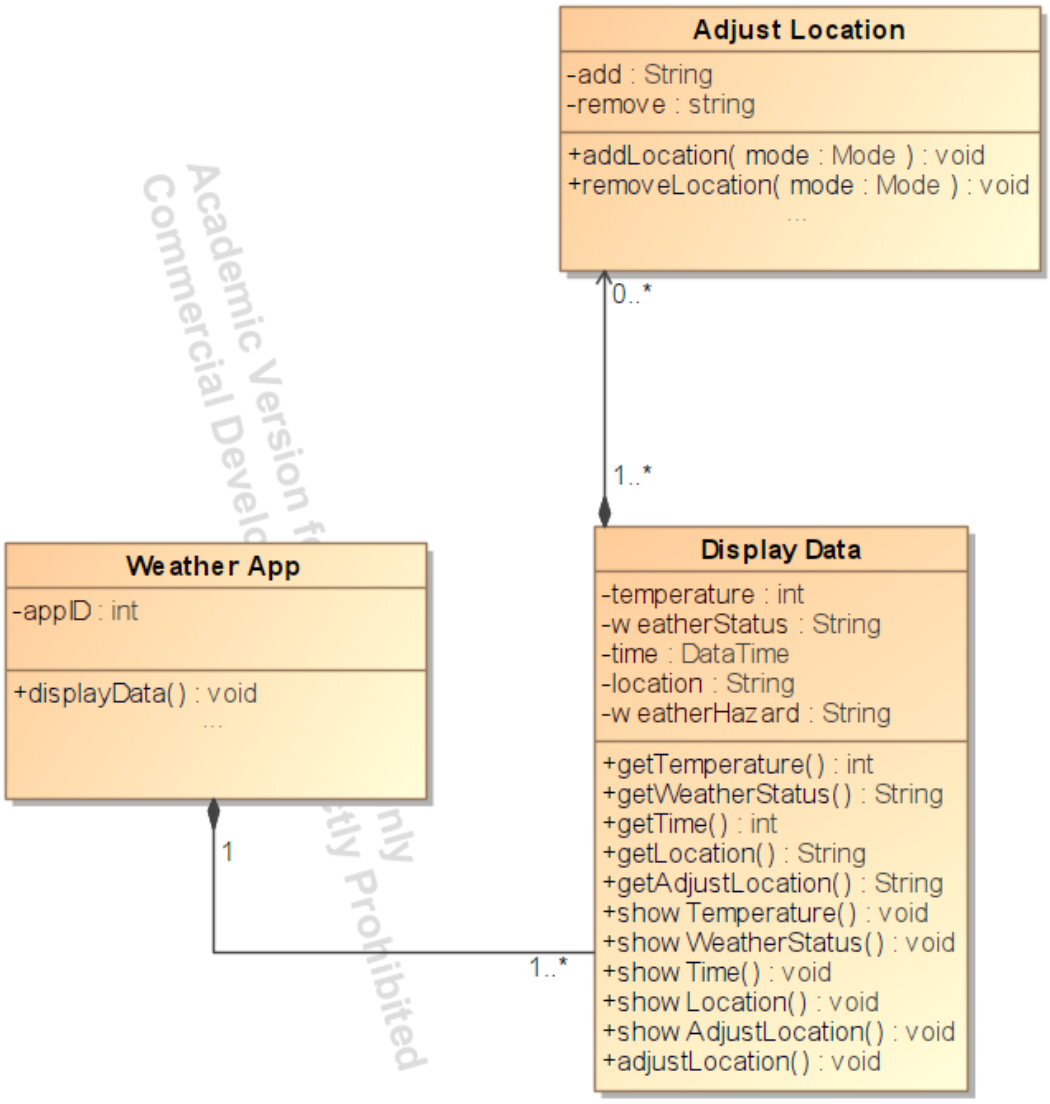
In my Use Case Diagram, the primary use case is initiating the "open the Weather app" action, triggering the execution of "Receive data from weather.com" and "Display weather condition". Moreover, users can press the "+" button to add any location they want to check the weather. Beside, the application includes two more use cases where pressing "-" to remove existed location on their watch through "Remove displayed location" use case. This description illustrates and defines the context and requirements of the Weather app by highlighting key user interactions, including adding and removing locations, along with dynamic weather display features.

7.2.2 Activity Diagram



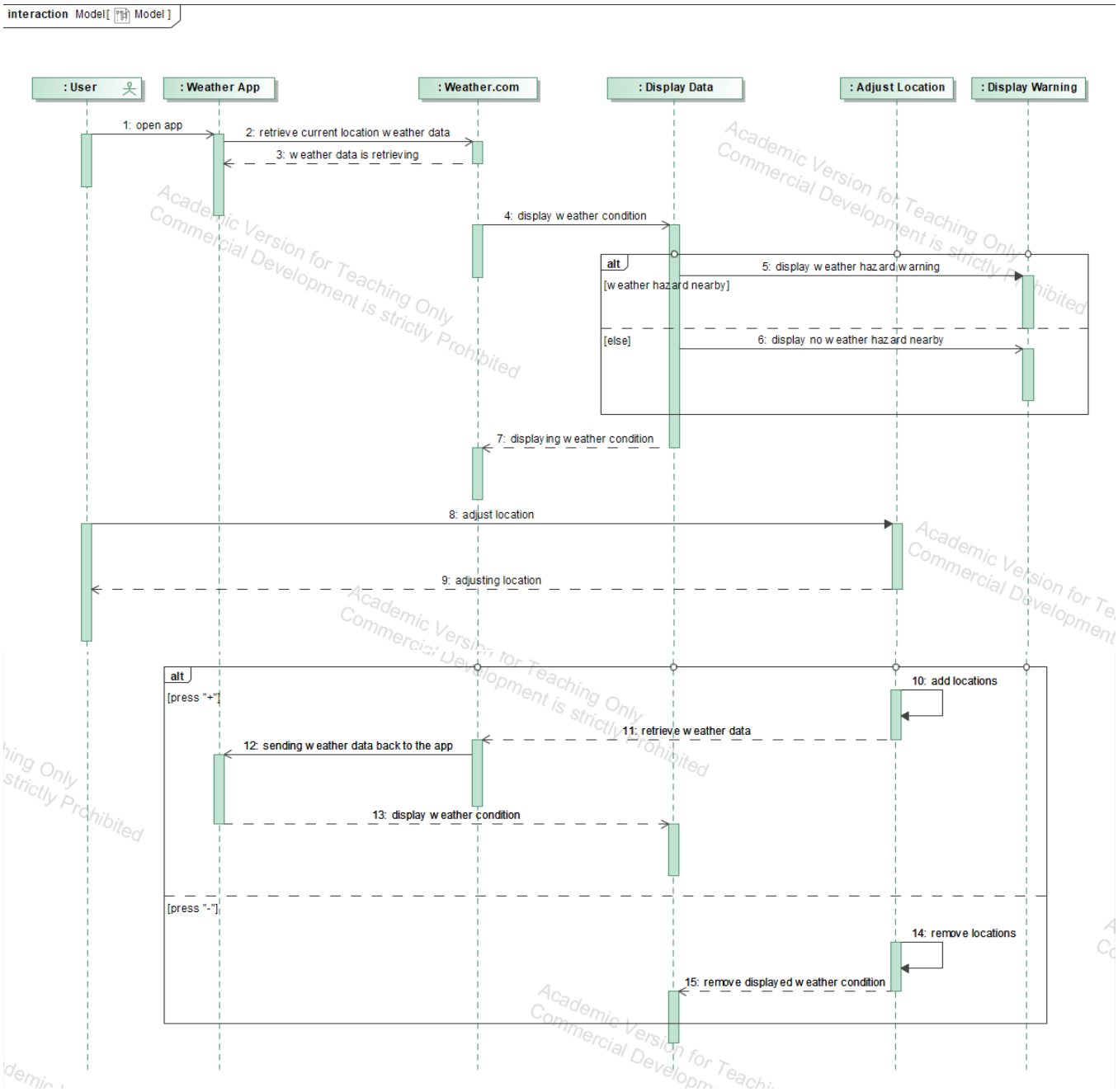
In my Activity Diagram, when the users open the app, by default, the app will retrieve data from weather.com and display the weather condition. Subsequently, the app checks for nearby weather hazards, providing users with a warning if necessary. To add more locations, users press "+", triggering the app to retrieve data from weather.com for the specified location and display those data. If users wish to remove a location, they simply press "-" to remove it from the UI.

7.2.3 Class Diagram



In my class diagram, there are four main classes. The first class is "User", consisting of two attributes: "userID" and "userName". Next, "Weather App" composed with "Display Data" class which has only one attribute "appID" and a method to call "Display Data" class. In "Display Data" class, it has "temperature", "weatherStatus", "time", "location" and "weatherHazard" attributes. Moreover, this class contains methods like "getData()" to retrieve information and "show()" to display the gathered data. Lastly, "Adjust Location" class is a component of "Display Data" class. This class only has two main attributes which are "add" and "remove" to add or remove location.

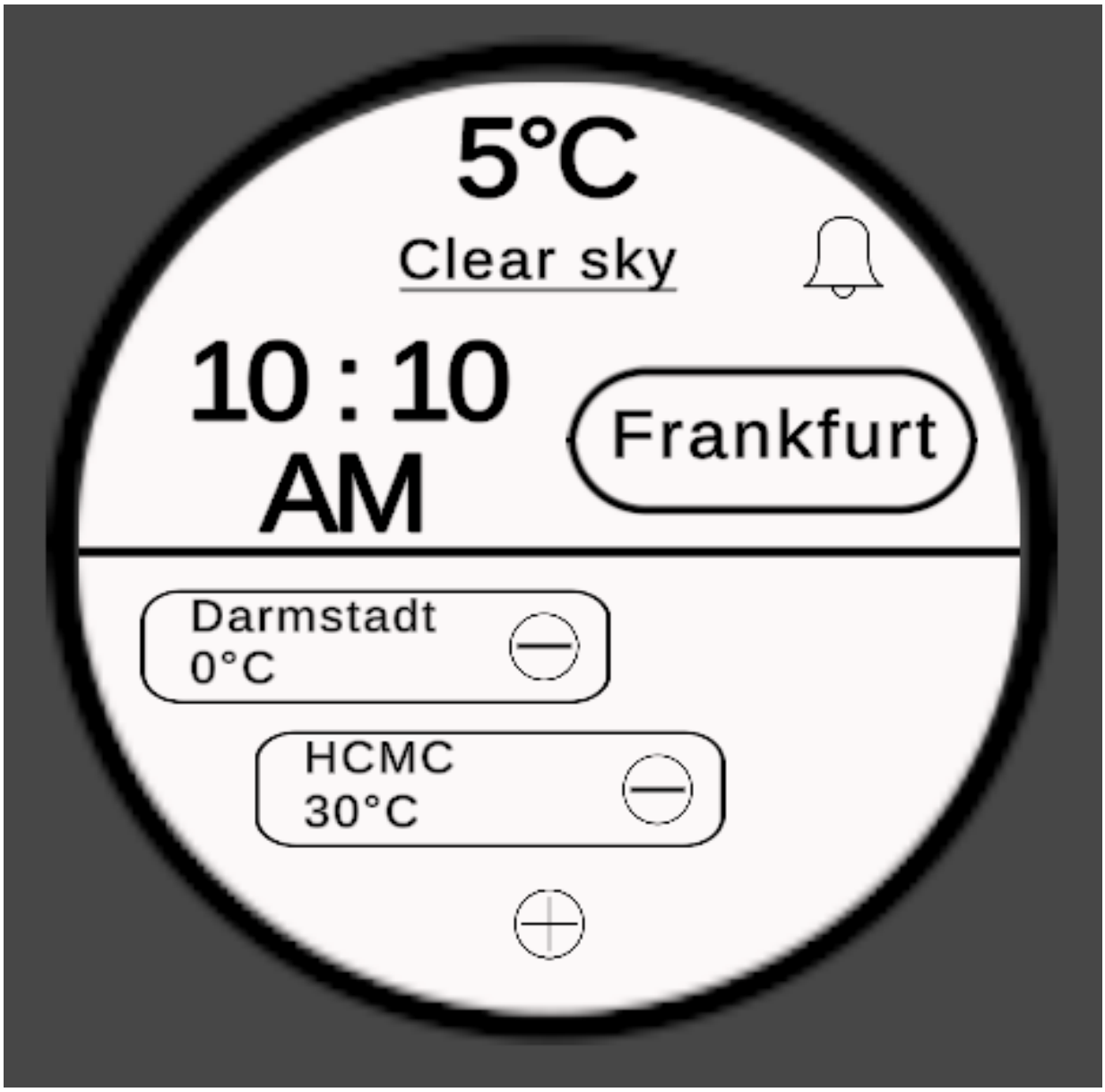
7.2.4 Sequence Diagram



My Sequence Diagram will describes how group of objects in the Weather app works together. Upon opening the app, it will retrieve current location weather data from "Weather.com", a reply of retrieved weather data is sent back to the app. Then the data from weather.com will send to "Display Data" object to display on the screen. If a weather hazard is detected nearby, the watch send message to "Display Warning" object for a warning; otherwise, it proceeds normally. In case the user want to adjust the location for weather checking, they will send a message to "Adjust Location" object that they want to change the location. Upon pressing "+", "Adjust Location" will add locations, make a reply message to "Weather.com" object to retrieve the weather

data and then send the data to "Display Data" to display those. Conversely, pressing "-" on existed location, "Adjust Location" will remove the location and reply a message to "Display Data" to remove it from the UI.

7.2.5 UI Prototype



This is a prototype UI design for our smartwatch Weather app. The upper half displays the user's current temperature, weather status, time, and location. The notification on the top right is for weather hazard detection. Below, added locations allow wearers to check the temperature in various cities. Users can press '+' to add more locations, while '-' lets them remove existing cities.

Chapter 8

Note

With a smartwatch's Note app, users can quickly take notes or create comprehensive lists right from their wrist. It makes note-taking possible while on the road, allowing user for the quick addition or modification of ideas. Additionally, Note app's has reminder feature which is especially helpful for keeping track of deadlines, appointments, and personal obligations. The Note app is designed with the intention of providing a user-friendly and simple UI.

8.1 Requirement Analysis

8.1.1 Snow cards

Requirements Type: Functional
For Whom: customer
User Satisfaction: High
User Dissatisfaction: Low
Description: Our smartwatch's Note app provides a convenient way to take notes and set reminders. Users can quickly add notes with a simple "+" icon. These notes then can be edited at anytime by pressing the note box. Pressing the "clock" icon will cause a timer to popup, allowing users to choose a reminder time for their note. Notes and reminders can be saved with a single easy "save" action. Note app is primarily used to remind users of important tasks like taking medicine, doing homework, or picking up their kids.

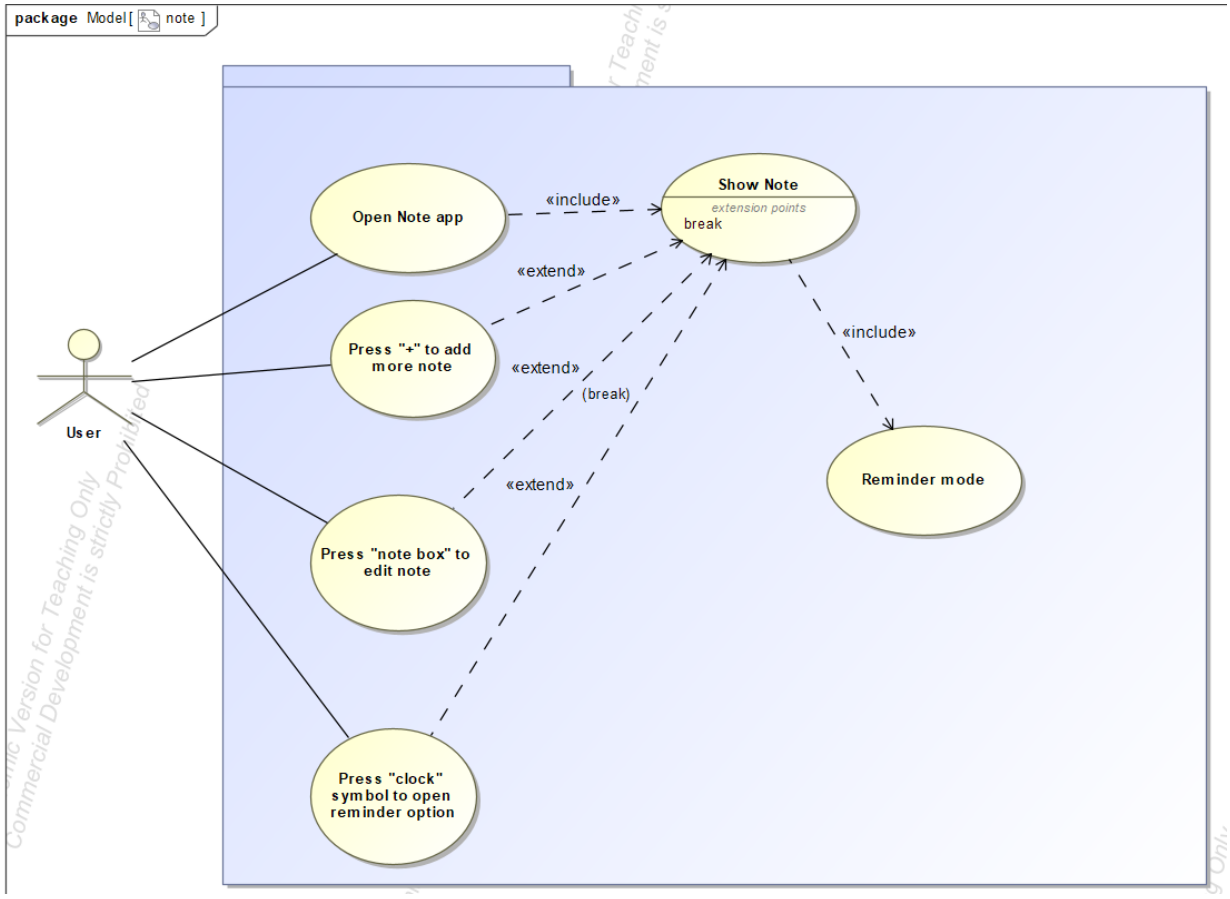
8.1.2 Use Case Analysis

Name	Note
ID	5
Description	Our smartwatch's Note App provides a convenient way to take notes and set reminders. Users can quickly add notes with a simple "+" icon. These notes then can be edited at anytime by pressing the note box. Pressing the "clock" icon will cause a timer to popup, allowing users to choose a reminder time for their note. Notes and reminders can be saved with a single easy "save" action. Note app is primarily used to remind users of important tasks like taking medicine, doing homework, or picking up their kids.
Trigger	For checking note: Activate the app by wearing the smartwatch, open the app. Tapping the "+" icon: Creating a new note. Tapping the "clock" icon: Setting a reminder.
Pre-conditions	The app must be opened or running in the background. The smartwatch must have sufficient battery power to operate.
Post-conditions	After creating a note, the note is saved and displayed in the list of notes. After editing a note, the changes are saved and the updated note is displayed. After setting a reminder, the reminder is connected with the note and the system is set to alert at the specified time.
Basic Flow	-
Description	This is the main scenario when the user chooses to take or edit notes.

Actions	0. Users open the Note App. 1. The watch displays a list of existing notes. 2. Users can select an existing note to view or edit, or press the "+" button to add a new note. 3. After adding or editing, users can save the note with a "save" action.
Alternative Flow	-
Description	User wants to set reminders for notes
Actions	0. Users open the Note App. 1. While viewing or editing a note, they can press the "clock" icon to set a reminder. 2. Users choose a reminder time for their note. 3. After choosing, users can save the reminder with a "save" action.

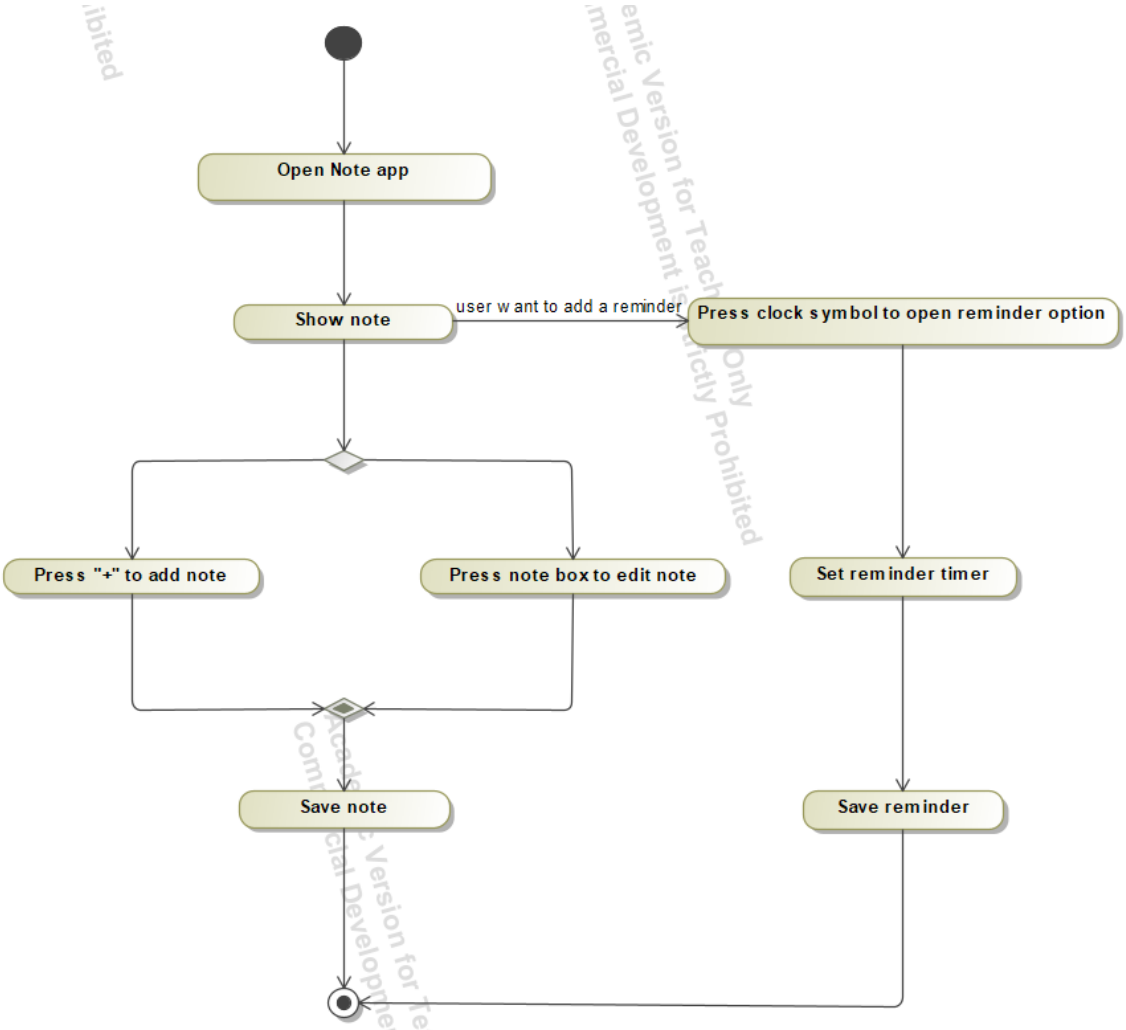
8.2 UML diagrams

8.2.1 Use Case Diagram



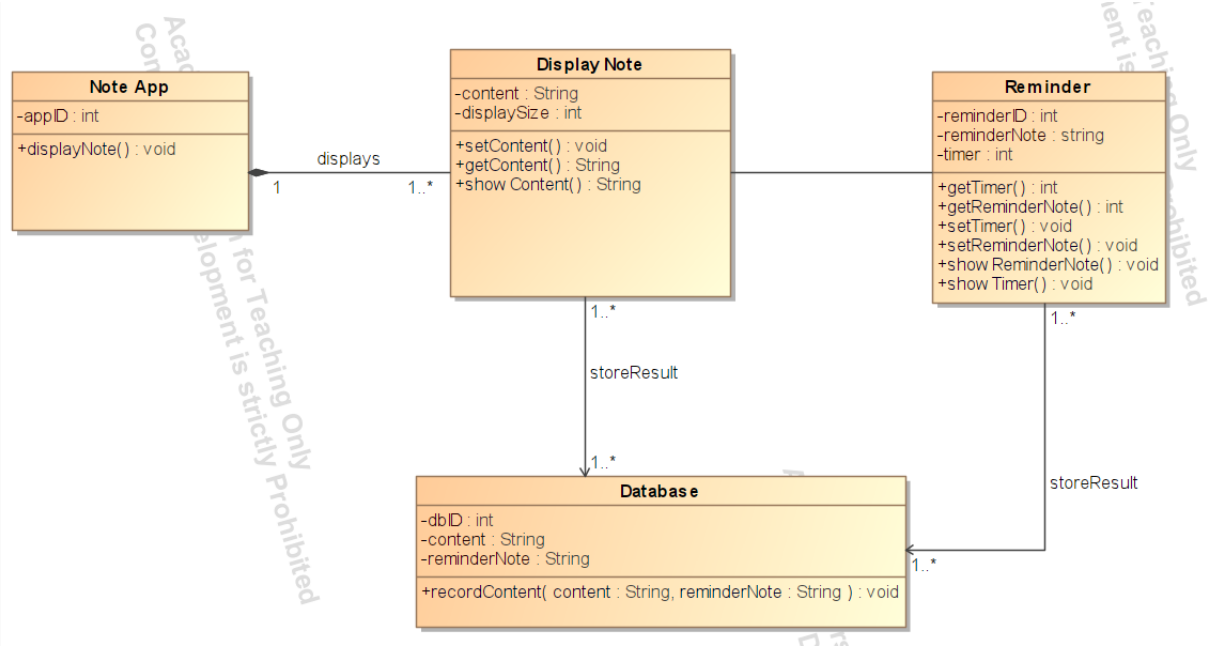
In my Use Case Diagram, it shows that the users has many interaction with the app. The primary interaction is opening the Note app. Once the app is open, the "Show Note" use case is automatically included, indicating that the app will display the notes. Additionally, there are several actions the user can do. These include adding more notes (pressing "+"), editing an existing note (pressing "note box"), and opening a reminder option (pressing "clock" symbol). Lastly, the "Reminder mode" which includes when the user taps the "clock" symbol. Once activated, this mode allows the user to set a reminder for a specific note. Overall, the diagram provides a view of how the system in our smartwatch Note app work.

8.2.2 Activity Diagram



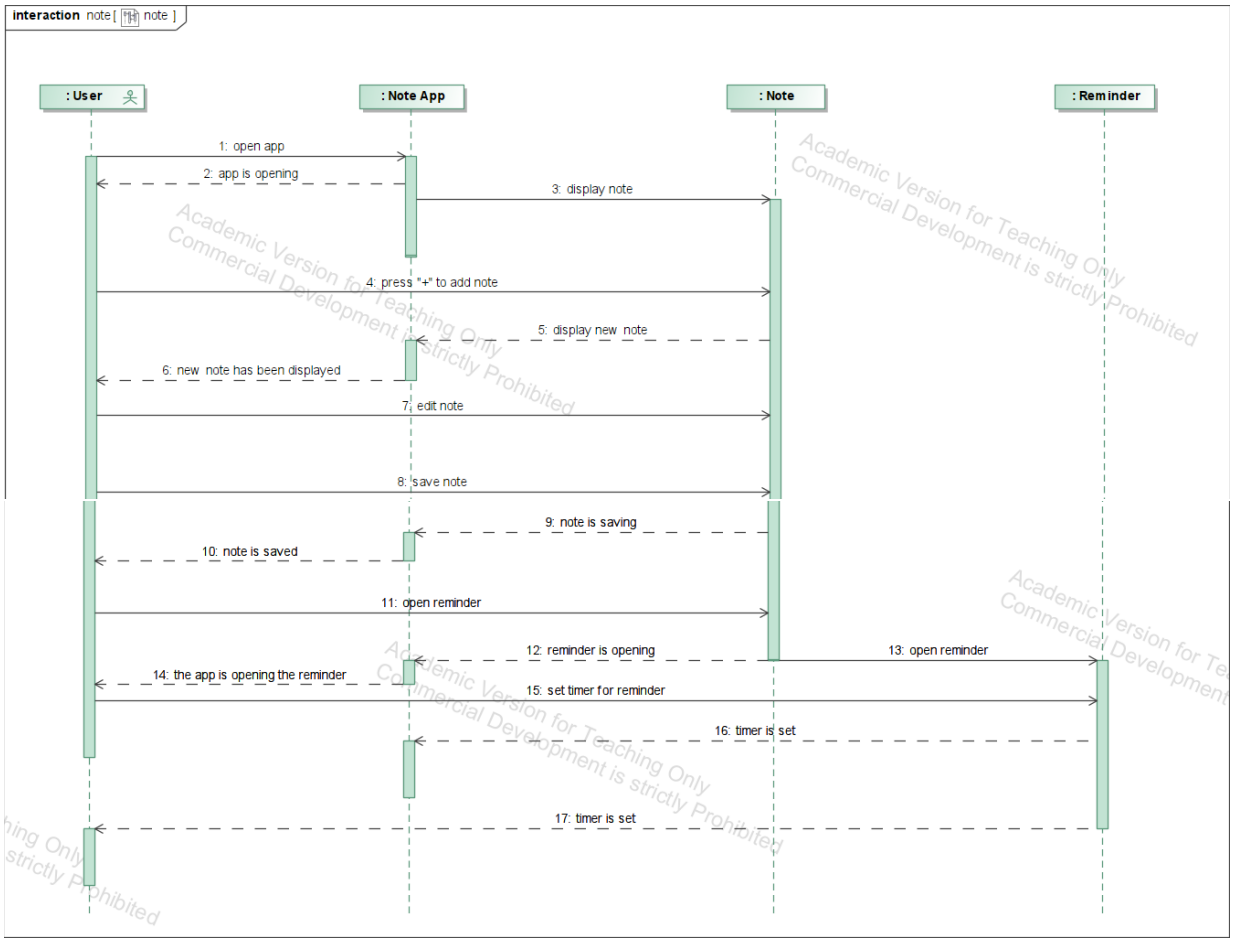
At the beginning, the user initiates the process by opening the Note app. Once the application is opened, the smartwatch displays the notes that have been previously saved. If the user wants to add a new note, by pressing the "+" symbol, the smartwatch opens a new, blank note where they can input their text. Alternatively, if the user wants to edit an existing note, they can select the "note box" which then allows them to make changes. After being satisfied with the changes, the user can press "save" to save the work. Another path in the activity diagram is for setting reminders. If the user wants to add a reminder, they can press the "clock" symbol, which opens the reminder option. This functionality is important for users who need to be notified of certain notes at specific times. Upon choosing this option, the user is able to set a reminder timer for their note. After setting the desired reminder time, the user can save the reminder. To sum up, this activity diagram clearly outlines the user's workflow within our Note app.

8.2.3 Class Diagram



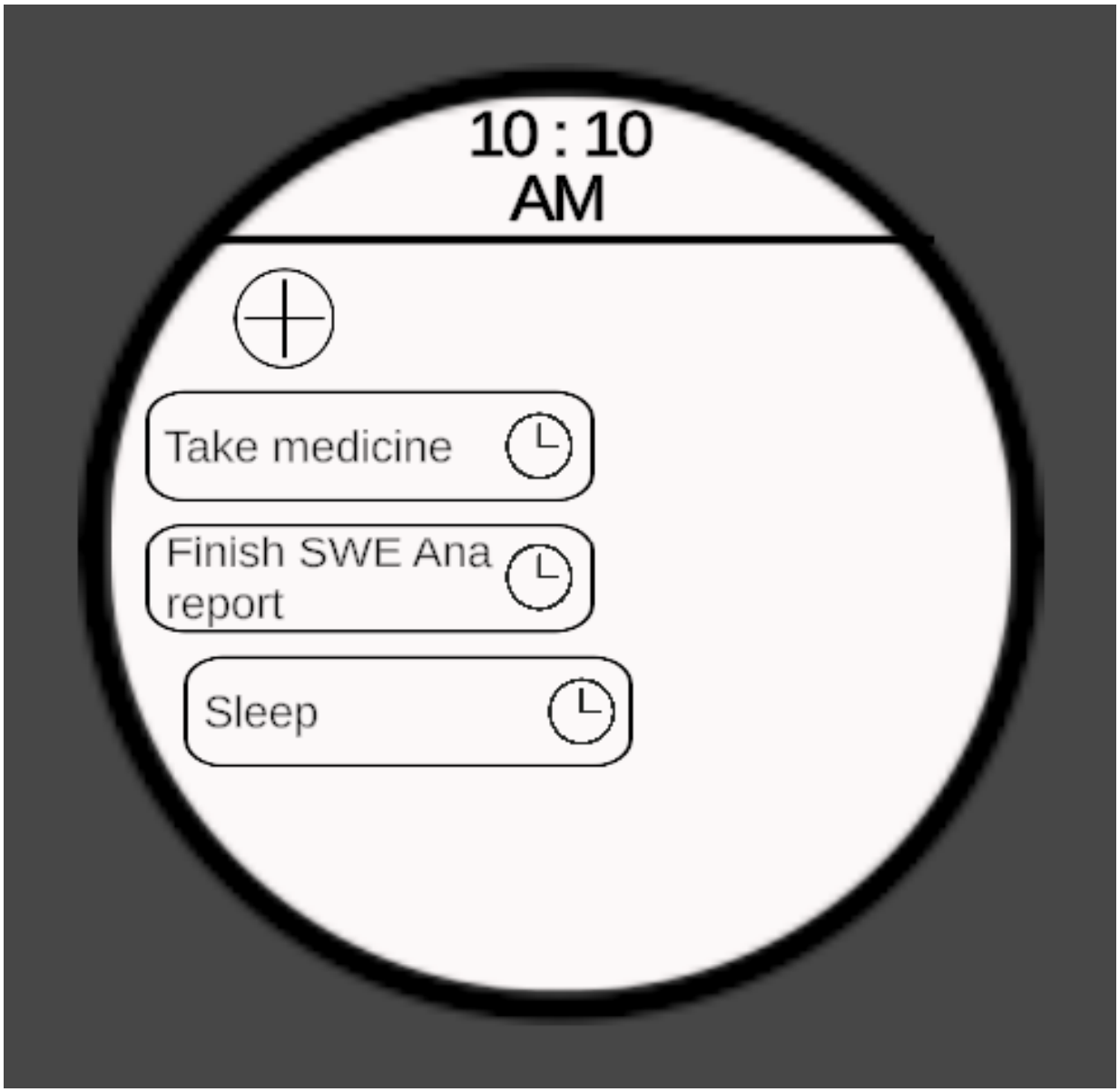
My class diagram consists of three main classes: Note App, Display Note, and Reminder, along with a Database class that interacts with them. The "Note App" class has an attribute "appId" and an operation displayNote() to trigger the display of notes to the user interface. The "Display Note" class is related to the Note App class. The Display Note class has attributes "content" and "displaySize". The operations include setContent() to change the note's content, getContent() to retrieve the content, and showContent() to return the content as a string. The "Reminder" class is designed to manage reminders. It has a "reminderID" and "reminderNote" attributes to identify the reminder and store its description. It also has a "timer" attribute which could be used to set the countdown or specific time for the reminder. The operations include getTimer() and getReminderNote() to retrieve the timer and note, setTimer() and setReminderNote() to update these attributes, and showReminderNote() and showTimer() to display these details. The "Database" class is responsible for storing and retrieving data with attributes "dbID", "content", and "reminderNote". The operation recordContent() indicates that the Database class can store note and reminder content. As a recap, the diagram describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among the objects.

8.2.4 Sequence Diagram



This sequence diagram shows the interactions between the user and the system components. The user initiates the process by opening the Note app. The app responds that it is opened. The Note App then displays all previously added notes. The user can add a new note by pressing the "+" symbol. In response, the Note App displays a new note for the user to interact with. Besides adding a new note, the user can edit an existing note. Once editing is complete, the user saves the note. The app indicates that the note is being saved, ensuring that the user's input is being stored. If the user decides to open the reminder functionality within the app, the app will indicate that the reminder feature is opening. Then, the user can set a timer for the reminder, specifying when they want to be alerted about this note. After that, the smartwatch will reply to the user that the reminder has been set. In conclusion, this sequence diagram shows step-by-step visualization of the user's interactions with the Note App, showing the flow of actions and the app's responses.

8.2.5 UI Prototype



This is a prototype UI design for my Note app. When the app is opened, all previously added notes are displayed automatically. On top, the watch display current time. To the left, user can press "+" icon to add more new notes. The user can edit existing notes by pressing the area around the notes, for example the box around "Take medicine". Lastly, the "clock" icon is used for add reminder to the specific note. When the set time elapses, the watch will vibrate to notify the user.

Chapter 9

Fitness tracking

Fitness Tracking is a dedicated functionality that help users monitoring their distances, heart rate, calories burned,...When they want to perform certain activities like running, cycling, or swimming, the watch will records the event and display them to user.

9.1 Requirement Analysis

9.1.1 Snow cards

Requirements Type: Functional
For Whom: customer
User Satisfaction: High
User Dissatisfaction: Low
Description: The fitness tracking feature in our "Ubugs Watch" smartwatch offers users an abstract health monitoring experience. With sensors and algorithms, it will captures and calculate some important fitness data such as heart rate, number of steps, distance taken, and calories burned. The user interface also display these relevant values into the display board, allowing users to track their progress and set personalized goals. Moreover, the watch also connect with the database, enabling users to access detailed historical data, receiving feedbacks, and stay motivated. The functionality is designed to help customers in achieving and maintaining a better lifestyle.

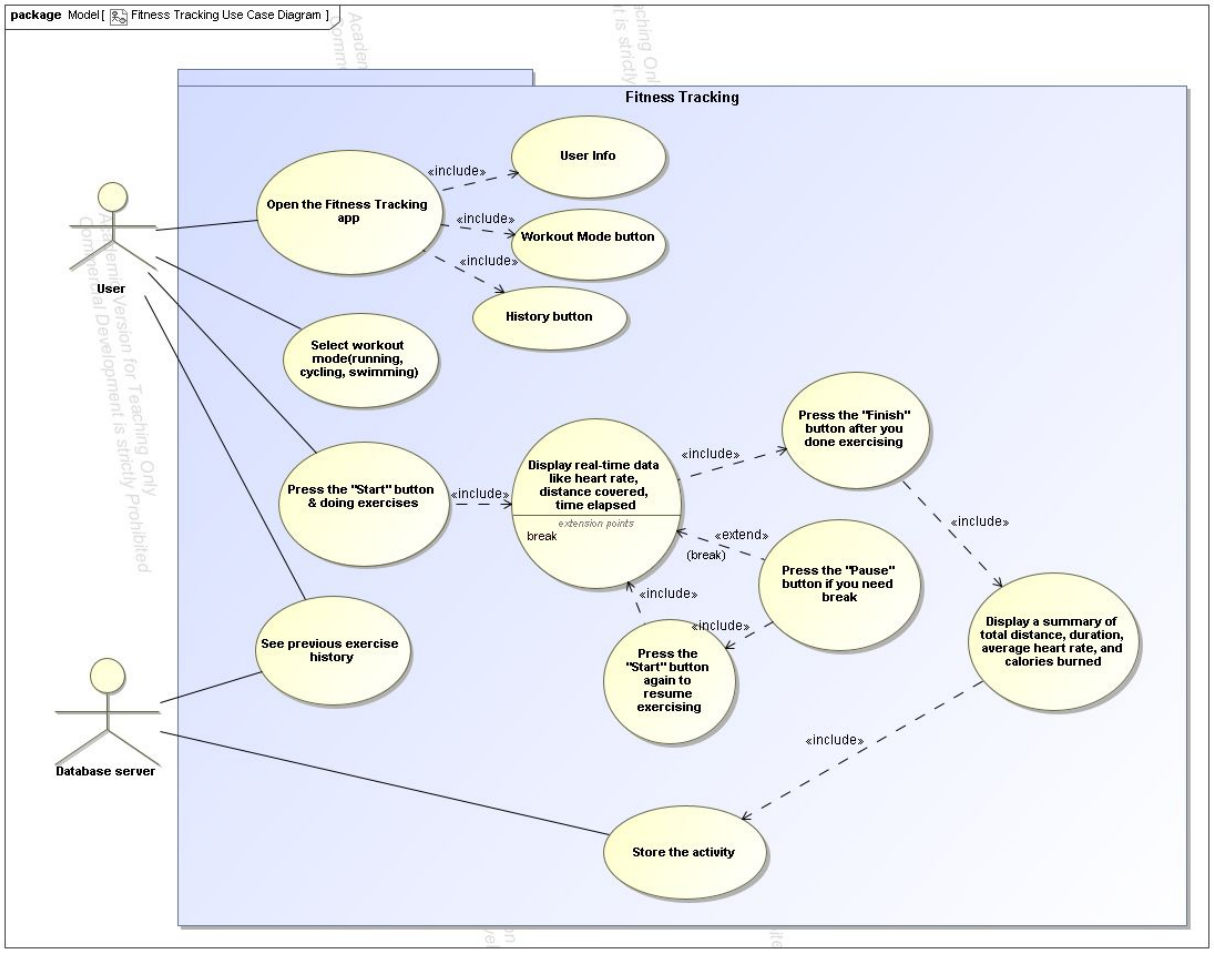
9.1.2 Use Case Analysis

Name	Fitness Tracking
ID	6
Description	The fitness tracking feature in our "Ubugs Watch" smartwatch offers users an abstract health monitoring experience. With sensors and algorithms, it will captures and calculate some important fitness data such as heart rate, number of steps, distance taken, and calories burned. The user interface also display these relevant values into the display board, allowing users to track their progress and set personalized goals. Moreover, the watch also connect with the database, enabling users to access detailed historical data, receiving feedbacks, and stay motivated. The functionality is designed to help customers in achieving and maintaining a better lifestyle.
Trigger	For start exercising: After the smartwatch is worn, the app is chosen, and specific workout mode is select
Pre-conditions	For fitness tracking: After the smartwatch is worn and the app is chosen. For real-time data monitoring: After user choose specific workout mode and press "Start".
Post-conditions	The summary of total distance, duration, average heart rate, and calories burned.
Basic Flow	-
Description	This is the main scenarios when the user choose to do exercise.

Actions	0. Users open the Fitness Tracking App. 1. They then press the workout mode button. 2. After pressing "Start", the watch will display real-time data like heart rate, time elapsed, calories burned. 3. If the users want to break, they can press "Pause" to take a break and "Resume" to continue. 4. After user press "Finish", a summary board of total distance, average heart rate, and calories burned will be shown.
Alternative Flow	-
Description	User want to see past activities
Actions	0. Users open the Fitness Tracking App. 1. They then press the history button 2. A list of activities is displayed

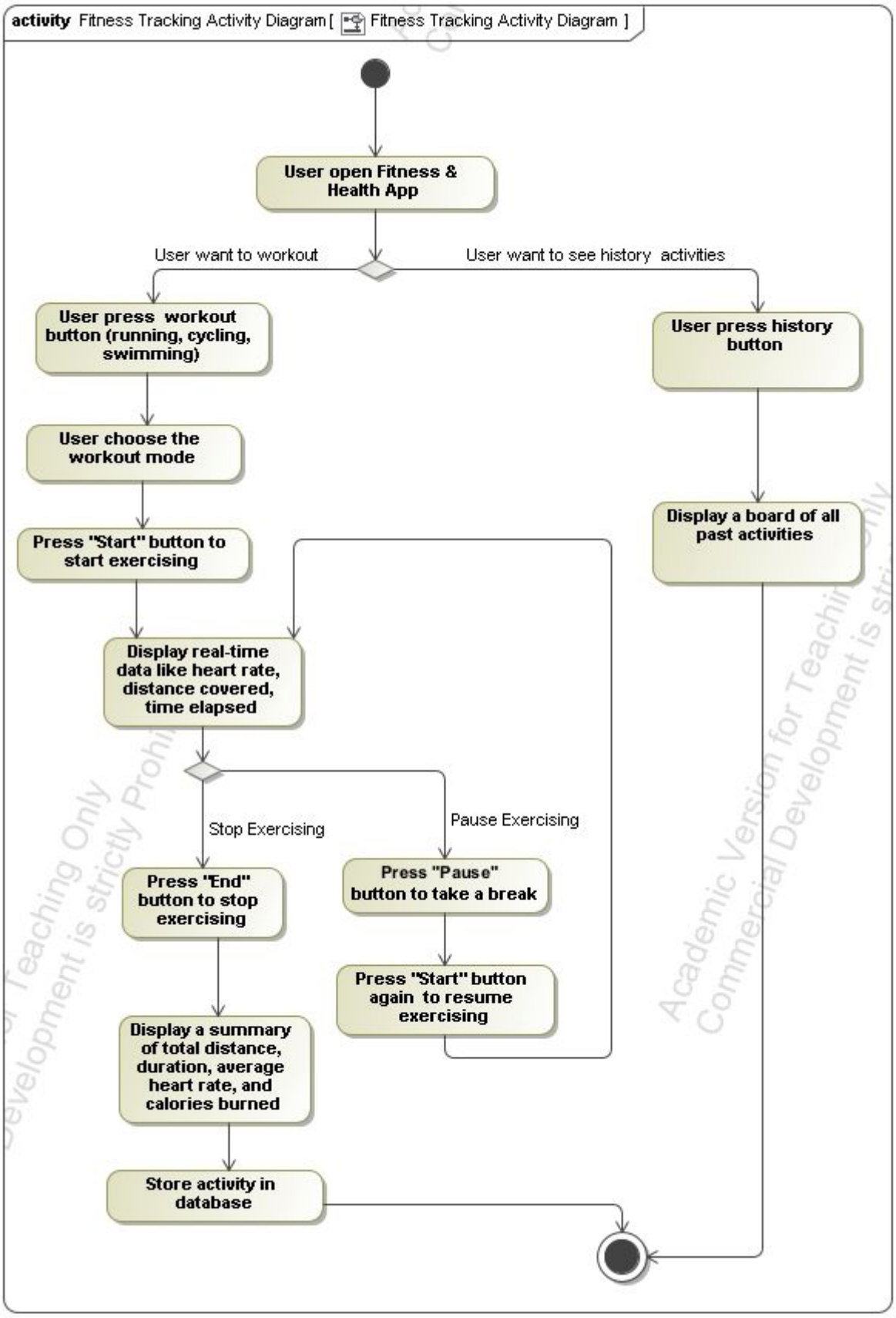
9.2 UML diagrams

9.2.1 Use Case Diagram



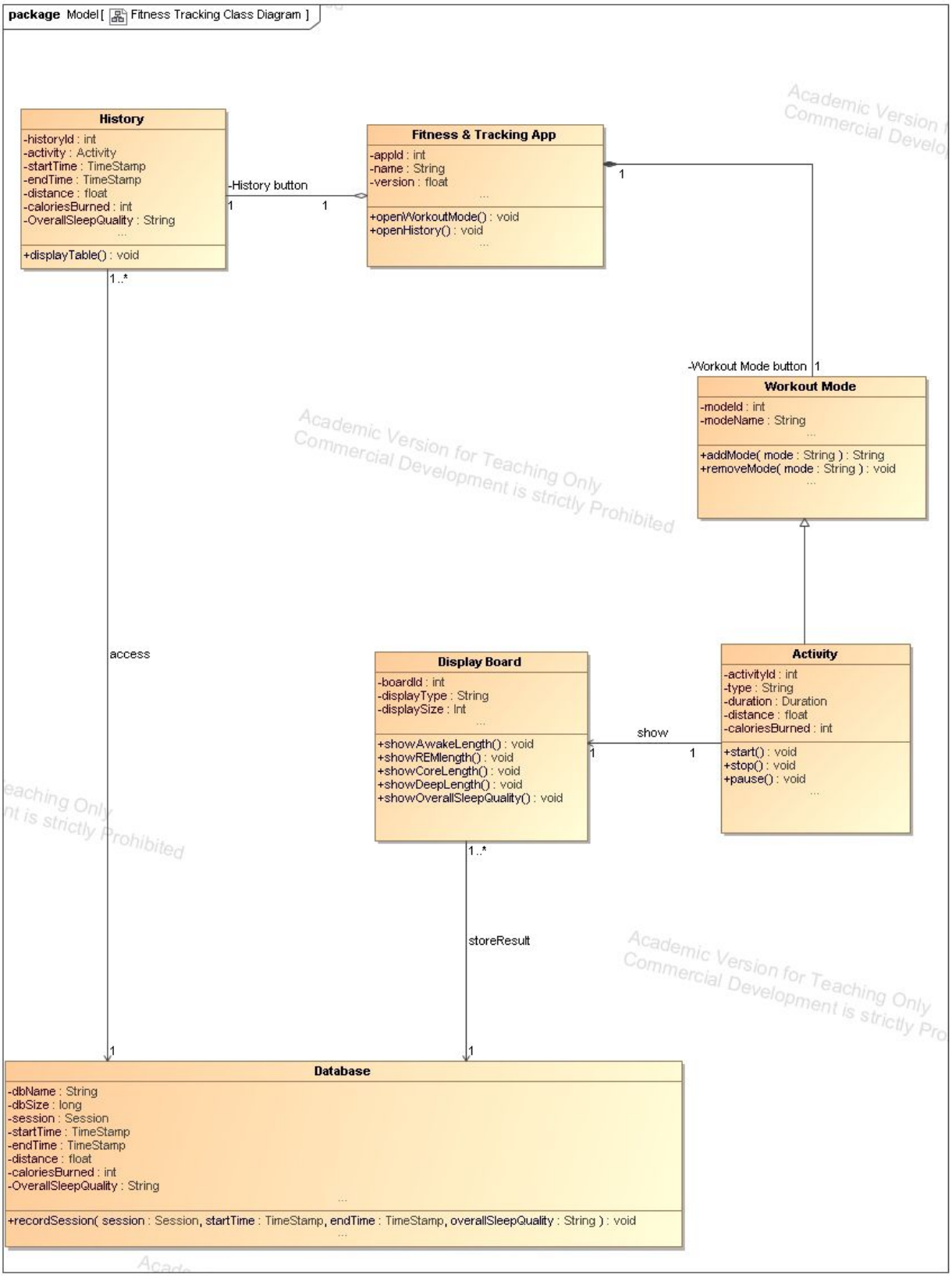
In my Use Case Diagram, it has several main use cases and included use cases. First, whenever the "open the Fitness Tracking app" is executed, this also implies the execution of "User Info", "Workout Mode button", and "History button". Moreover, there also a use case where customers can select the workout mode among running, cycling, or swimming. Besides, when users press the "Start" button, the watch will display real-time data like heart rate, distance covered, time elapsed. There is also two included use cases where customers can press "Finish" to complete or sometimes press "Pause" to take a break and press "Continue" to resume exercising. After pressing "Finish" button, the watch will display a summary board of total distances, duration, average heart rate, calories burned, average scores and store that activity into the database system. Finally, there is also a main use case where users can see a list of previous activities that they have done.

9.2.2 Activity Diagram



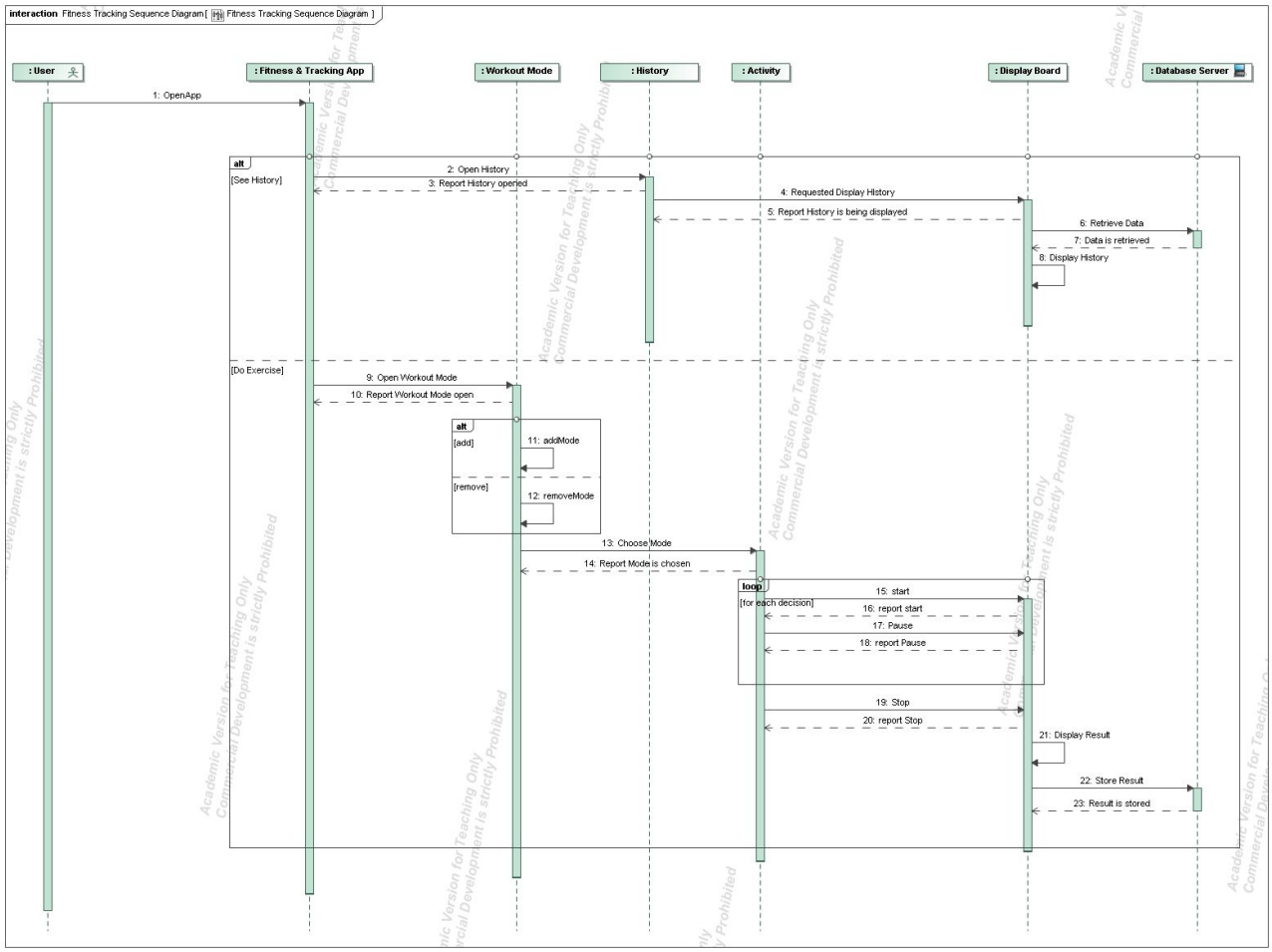
In my Activity Diagram, at first users will open the Fitness Tracking app. After that, if they want to do exercise, they simply click the "Workout Mode" button and choose the mode that they want. Subsequently, they need to press "Start" button to begin exercising. In real time, it will display relevant informations like heart rate, distance covered, and time elapsed. If they are tired and want to take a short break, just simply click "Pause" and "Start" again whenever they want. If they want to stop, press "End". After that, the app will display a summary board of total distance, duration, heart rate and calories burned. Finally, the data about the activity will then be stored inside the database. However, if the user want to see the history of all activities in the past, they can press the history button and a list of past activities will be shown.

9.2.3 Class Diagram



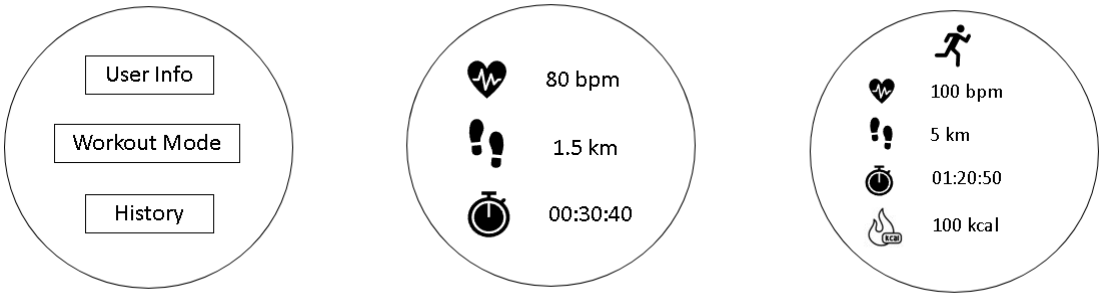
In my class diagram, the App Class has the appID, name, version, and also the operation to open workout mode and history. The fitness tracking app aggregate with history and composite with workout mode. The History class include many attributes like historyid, activity, startTime, endTime, distance covered, calories burned, and also an operation to display the table to users. The workout mode which contain in the app, also has its own modeid, modename and the function to add or remove mode. The Activity, which is a subclass of workout mode, has some attributes like activityid, type, duration, distance, calories burned and some operation to start, pause, and stop. With the display board being shown, it has boardid, displaytype, displaysize, and can show some data like heart rate, userdetails, distance covered, time elapsed, and overall scores. Finally is the database which store the activity, including certain attributes like dbName, table, dbSize, activity, startTime, endTime, distance covered, calories burned and a function to store the Activity into that database.

9.2.4 Sequence Diagram



In my Sequence Diagram, the customers will first open the Fitness Tracking app. After that, they will have an option to choose, whether they want to do exercise or see the past activities. If the users decide to do exercise, they will first open the Workout Mode. Besides, they will have another option to add or remove a mode in the smartwatch. The customers then choose the mode they want and press "Start" to begin exercising. While running, they can press "Pause" to take a short break and press "Start" again whenever they want. When they decide to stop doing exercise, they can press "Finish" to complete the activity. Consequently, the watch will display a summary board of total distances, duration, average heart rate, calories burned, average scores and store that activity into the the database system. On the other hand, If the users want to see the history board of all activities in the past, they can choose to open the History. It will request the history data from the database. After retrieved the data needed, it will display a list of activities that the customers have done.

9.2.5 UI Prototype



The first UI Prototype is the home page of the app. The second one is the display of real-time data containing heart rate, distance covered, and time elapsed when the customer is doing exercise. And the third one is the summary board after they finish exercising. It includes the average heart rate, distance taken, time elapsed, and calories burned.

Chapter 10

Sleep Tracking

Sleep Tracking is a dedicated functionality that design with precision for individuals like you who value the importance of a good night's sleep. Monitor your sleep patterns seamlessly as the app analyzes your sleep stages, providing insightful data on your sleep duration, efficiency, and interruptions. With user-friendly interfaces and customizable settings, our app ensures a hassle-free experience. Take charge of your sleep health, make informed decisions, and wake up refreshed with our smartwatch sleep tracking app – your ultimate companion for a well-rested life.

10.1 Requirement Analysis

10.1.1 Snow cards

Requirements Type: Functional
For Whom: customer
User Satisfaction: High
User Dissatisfaction: Low
Description: Embark on a rejuvenating journey to better sleep with the "Ubugs Watch" smartwatch's advanced sleep tracking app. Equipped with state-of-the-art sensors and sophisticated algorithms, this app provides a comprehensive analysis of your sleep patterns, including insights into sleep duration, quality, and disturbances. The intuitive interface beautifully presents crucial sleep metrics, allowing you to effortlessly monitor your nighttime habits. Set personalized sleep goals and witness your progress unfold. Seamlessly syncing with a robust database, the app grants access to detailed historical data, insightful trends, and recommendations for optimizing your sleep routine. Experience the transformative power of our sleep tracking app, designed to empower you on your path to a more restful and rejuvenating night's sleep.

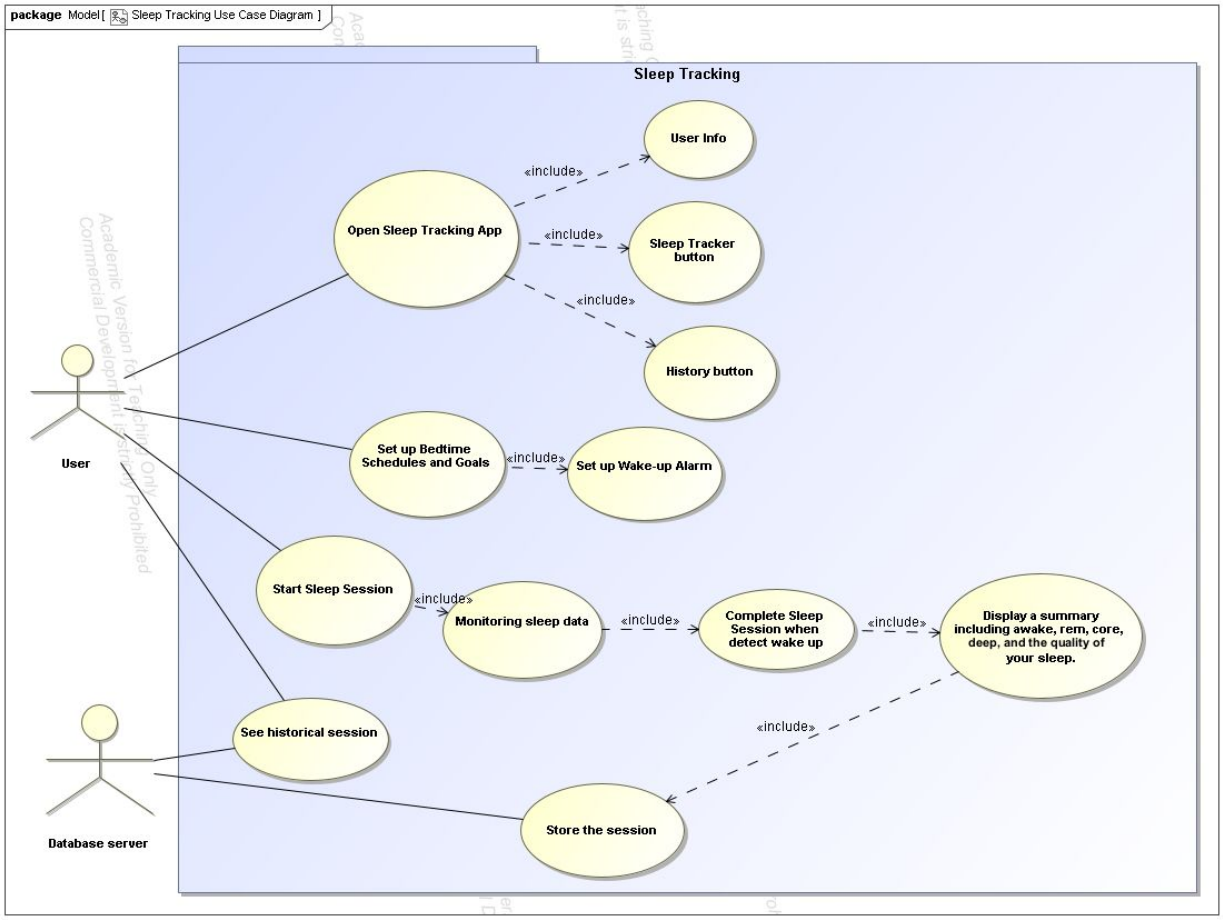
10.1.2 Use Case Analysis

Name	Sleep Tracking
ID	7
Description	Embark on a rejuvenating journey to better sleep with the "Ubugs Watch" smartwatch's advanced sleep tracking app. Equipped with state-of-the-art sensors and sophisticated algorithms, this app provides a comprehensive analysis of your sleep patterns, including insights into sleep duration, quality, and disturbances. The intuitive interface beautifully presents crucial sleep metrics, allowing you to effortlessly monitor your nighttime habits. Set personalized sleep goals and witness your progress unfold. Seamlessly syncing with a robust database, the app grants access to detailed historical data, insightful trends, and recommendations for optimizing your sleep routine. Experience the transformative power of our sleep tracking app, designed to empower you on your path to a more restful and rejuvenating night's sleep.
Trigger	For start sleep session: After the smartwatch is worn, the app is chosen, the sleep tracker is set up, the session will be automatically start when you sleep.
Pre-conditions	For sleep tracking: After the smartwatch is worn and the app is chosen. For real-time data monitoring: After user set up the tracker and the session is automatically started.
Post-conditions	The summary including awake, rem, core, deep, and the quality of your sleep.
Basic Flow	-

Description	This is the main scenarios when the user choose to sleep.
Actions	0. Users open the Sleep Tracking App. 1. They then press the Sleep Tracker button. 2. After setting up completely all requirements, the watch will monitor the sleep data in real time. 3. After the watch detect that user is waken up, the session is automatically end. 4. Subsequently, the watch will display a summary of sleep data including awake, rem, core, deep, and the quality of your sleep.
Alternative Flow	-
Description	User want to see past activities
Actions	0. Users open the Sleep Tracking App. 1. They then press the History button 2. A list of historical sleep session is displayed

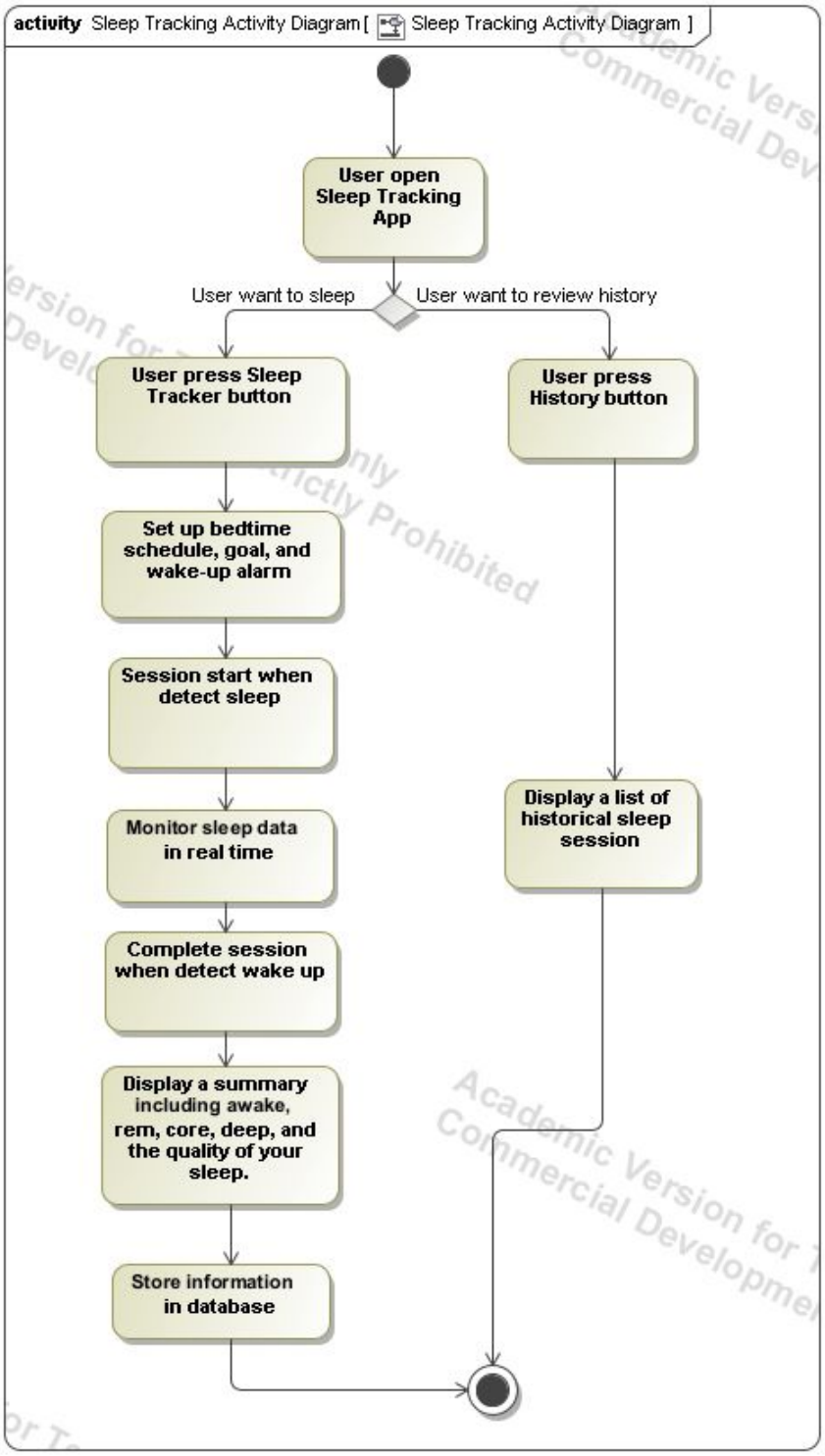
10.2 UML diagrams

10.2.1 Use Case Diagram



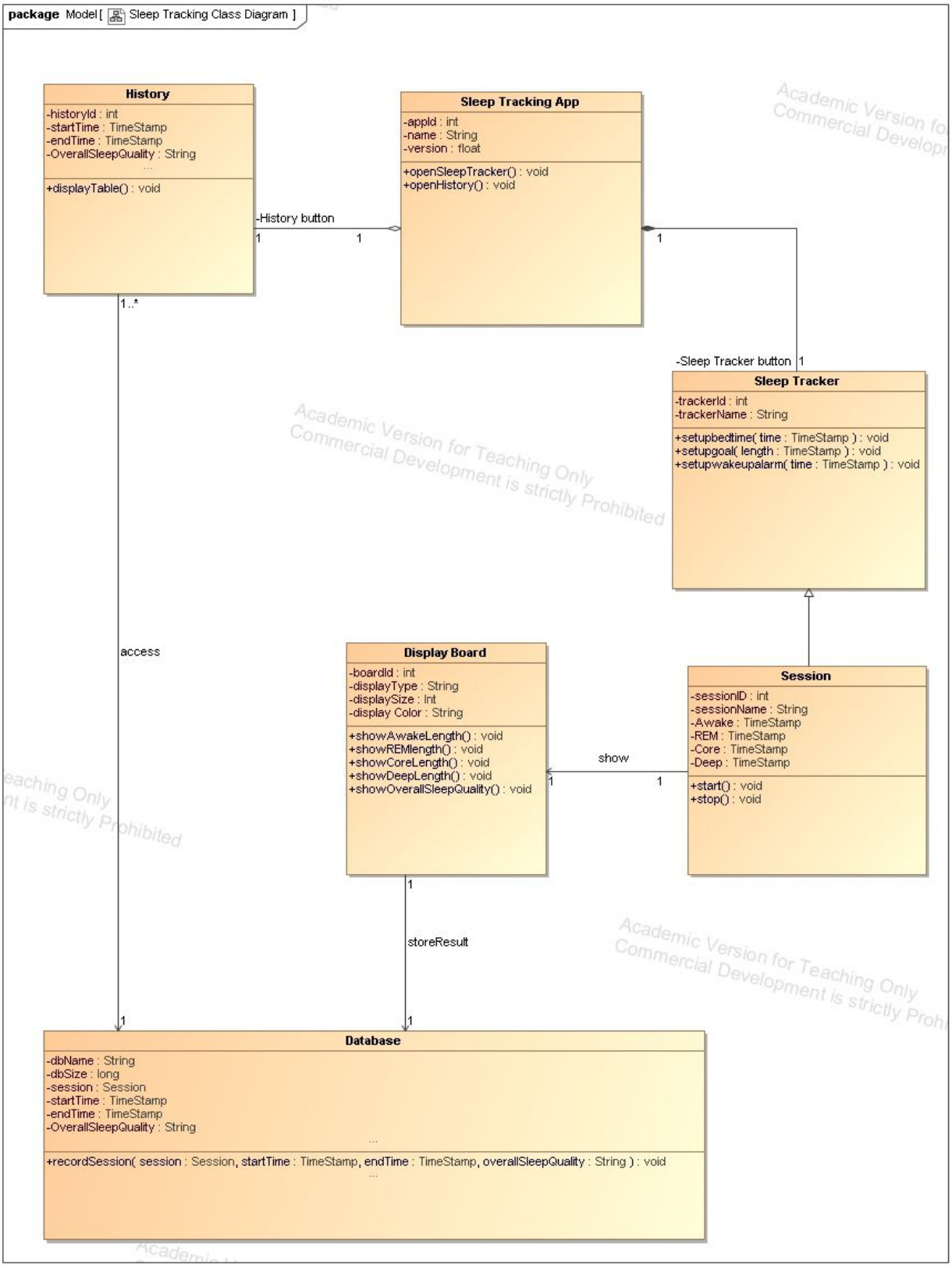
In my Use Case Diagram, it has several main use cases and included use cases. First, whenever the "open the Sleep Tracking app" is run, this also implies the execution of "User Info", "Sleep Tracker button", and "History button". Furthermore, there also a use case where customers can set up the Sleep Tracker. They first set up the Bedtime Schedules and Goal, then set up the Wake-up Alarm. Besides, when users sleep, the session is automatically start. After that, the watch is monitoring the sleep data including awake, rem, core, and deep in real time scenario. Afer the watch detect that the customers is waken up, it then automatically complete the Sleep Session. A summary board containing awake length, rem length, core length, deep length, and especially the calculated overall quality of their sleep will all be shown to the customers. Finally, the sleep session will then be stored in the database server. On the other hand, there is also a main use case where user can press the History button and see the list of all historical sleep session.

10.2.2 Activity Diagram



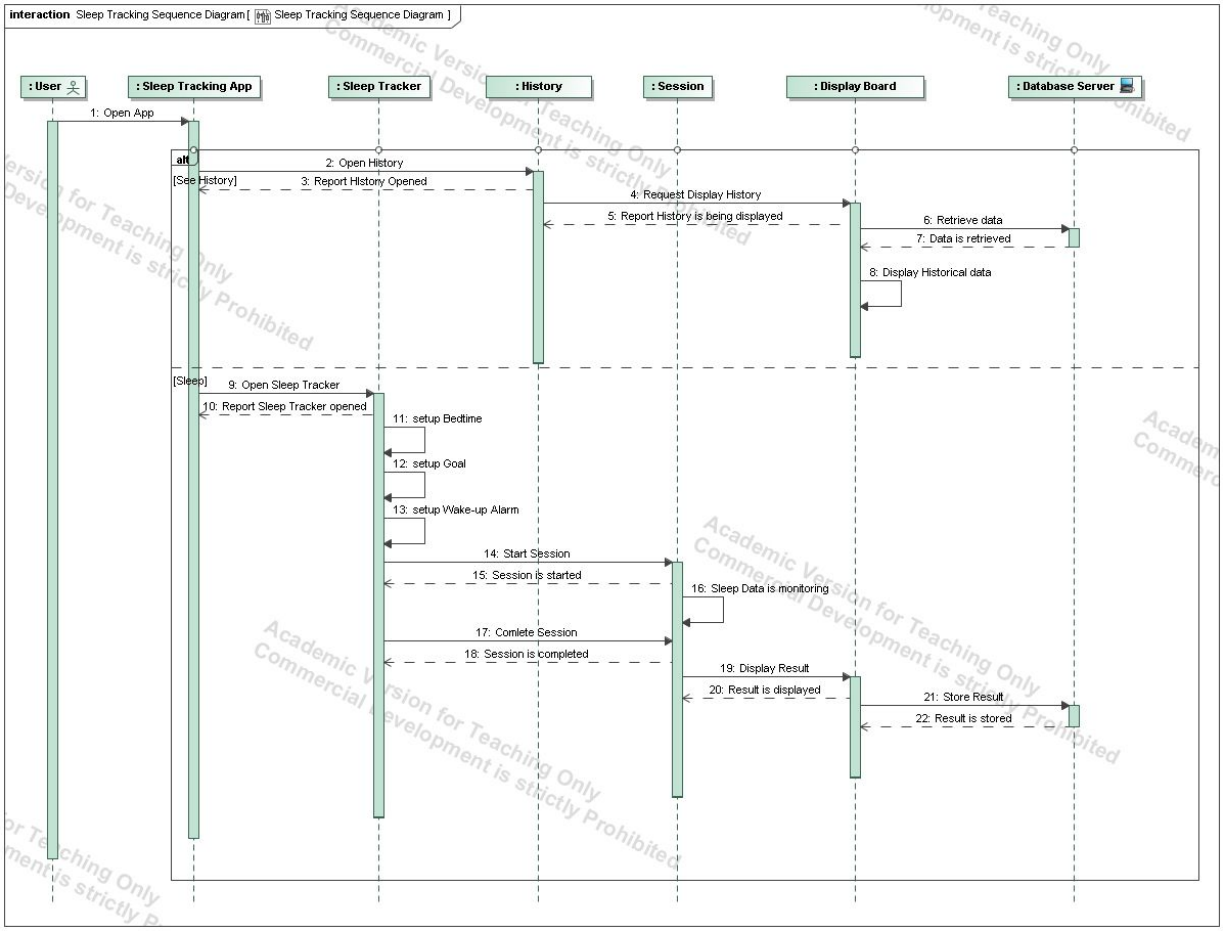
In my Activity Diagram, at first users will open the Sleep Tracking app. After that, if they want to sleep, they simply click the "Sleep Tracker" button. The customers then need to set up the Bedtime schedule, Goal, and the Wake-up Alarm. When the watch detect that users begin to fall asleep, it will automatically start the session and monitoring the sleep data in the real time. After the watch recognize that the consumers are waken up, the session is automatically completed. Consequently, a summary board containing awake length, rem length, core length, deep length, and especially the calculated overall quality of their sleep will all be shown to the customers. At last, the sleep session will then be saved in the database server. On the other hand, if the user want to review the historical sleep data, they can press the History button and see the list of all historical sleep session.

10.2.3 Class Diagram



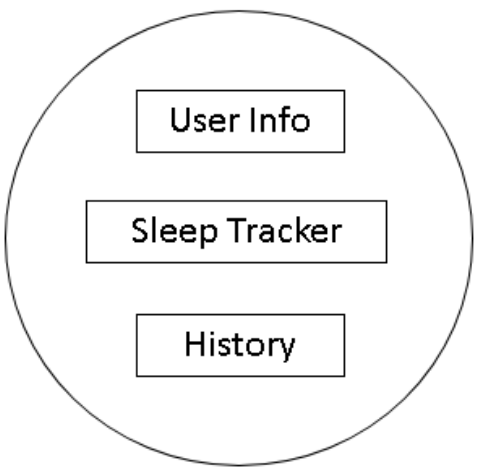
In my class diagram, User can open the Sleep Tracking App, it has the appID, name, version, and also the operation to open Sleep Tracker and History. The Sleep Tracking app aggregate with History class and composite with Sleep Tracker class. The History class include many attributes like historyId, startTime, endTime, OverallSleepQuality, and also an operation to display the table to users. The Sleep Tracker which contain in the app, also has its own trackerId, trackerName and the function to set up the bedtime, goal, and the wake-up alarm. The Session, which is a subclass of Sleep Tracker, has some attributes like sessionId, sessionName, Awake, REM, Core, Deep and some operation to start,and stop the session. With the display board being shown, it has boardid, displaytype, displaysize, display Color and can show some data like AwakeLength, REMLength, CoreLength, DeepLength, and the calculated OverallSleepQuality. Finally is the database which store the session, including certain attributes like dbName, table, dbSize, session, startTime, endTime, OverallSleepQuality and a function to store the Session into that database.

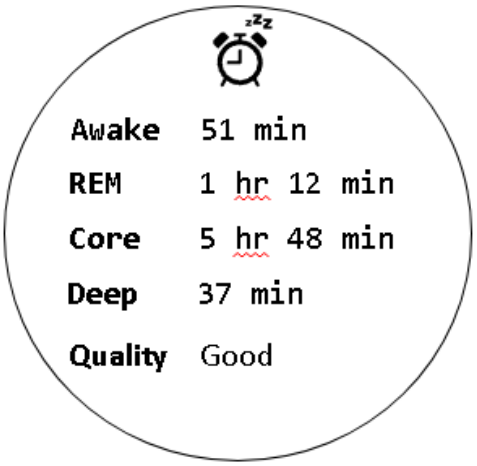
10.2.4 Sequence Diagram



In my Sequence Diagram, the customers will first open the Sleep Tracking app. After that, they will have an option to choose, whether they want to go to sleep or see the historical sleep session. If the users decide to sleep, they will first open the Sleep Tracker. Besides, they will need to first setting up the Tracker which include Bedtime, Goal, and Wake-up Alarm. Then, when the customers fall asleep, it will recognize and start the Sleep Session. In real time, the watch will monitoring the sleep data of users, which contain awake, rem, core, deep period. After that, it will finish the Sleep Session if detecting that the customers is finally waken up. Consequently, a summary board containing awake length, rem length, core length, deep length, and especially the calculated overall quality of their sleep will all be shown to the customers. At last, the sleep session will then be saved in the database server.On the other hand, If the users want to see the history board of all sleep session in the past, they can choose to open the History. It will request the history data from the database. After retrieved the data needed, it will display a list of all historical sleep session that the customers have done.

10.2.5 UI Prototype





The first UI Prototype is the home page of the app, which include the user info, sleep tracker button, and history button. The second one is the summary display board that contain Awake period, REM period, Core period, Deep period, and the calculated overall Sleep Quality whether the sleep is good or not.

Chapter 11

Sightless Support

This chapter is an overview of Sightless Support, an area dedicated to helping individuals with visually challenging in navigating their surrounding and accessing information. The chapter is an in-depth of what and how the user may use the application to their advantages and the way these technologies work.

11.1 Requirement Analysis

11.1.1 Snow cards

11.1.1.1 Sightless Support

1. **Activation:**
   Requirement Type: Functional
   For whom: User
   User satisfaction: High
   User dissatisfaction: High
   Description: The system shall display activate button for user

2. **List of accessibility:**
   Requirement Type: Functional
   For whom: User
   User satisfaction: High
   User dissatisfaction: High
   Description: The system shall display a list of accessibilities when being activated

3. **Verify change:**
   Requirement Type: Functional
   For whom: User
   User Satisfaction: High
   User Dissatisfaction: High
   Description: The system shall allow user to verify the changes

4. **Notify reboot:**
   Requirement Type: Functional
   For whom: User
   User Satisfaction: High
   User Dissatisfaction: High
   Description: The system shall notify the user about the reboot

11.1.1.2 Visibility Enhancement

1. **Set Brightness:**
   Requirement Type: Functional
   For whom: User
   User satisfaction: Medium
   User dissatisfaction: Medium
   Description: The system shall allow user to adjust the screen's brightness

2. **Set Font:**
   Requirement Type: Functional
   For whom: User
   User satisfaction: Medium
   User dissatisfaction: Medium
   Description: The system shall allow user to change the font.

3. **Magnify**
   Requirement Type: Functional
   For whom: User
   User satisfaction: Medium
   User dissatisfaction: Medium
   Description: The system shall allow user to magnify components on screen

4. **Zooming:**
   Requirement Type: Functional
   For whom: User
   User satisfaction: Medium
   User dissatisfaction: Medium
   Description: The system shall able to be zoomed

11.1.1.3 Braille Display

1. **Set and Get Display Text**
   Requirement Type: Functional

For whom: User
User satisfaction: High
User dissatisfaction: High
Description: The system shall be able to set and get the text from display

2. **Clear Display**
Requirement Type: Functional
For whom: User
User satisfaction: High
User dissatisfaction: High
Description: The system shall allow user to magnify components on screen

3. **Flipping Display:**
Requirement Type: Functional
For whom: User
User satisfaction: Medium
User dissatisfaction: Medium
Description: The system shall able to flip between normal and Braille display.

11.1.1.4 Object Identification

1. **Set Threshold**
Requirement Type: Functional
For whom: User
User satisfaction: High
User dissatisfaction: High
Description: The system shall allow user to set the detecting threshold

2. **Detect Object**
Requirement Type: Functional
For whom: User
User satisfaction: High
User dissatisfaction: High
Description: The system shall be able to detect surrounding object.

3. **Alert User:**
Requirement Type: Functional
For whom: User
User satisfaction: Medium
User dissatisfaction: High

Description: The system shall able to vibrate when the detected object is inside the threshold.

11.1.1.5 Voice Control

1. **Voice Recognition**
   Requirement Type: Functional
   For whom: User
   User satisfaction: High
   User dissatisfaction: High
   Description: The system shall listen to the user voice when being called.

2. **Process Command**
   Requirement Type: Functional
   For whom: User
   User satisfaction: High
   User dissatisfaction: High
   Description: The system shall be able extract the given speech into instructions

3. **Get Text**
   Requirement Type: Functional
   For whom: User
   User satisfaction: Medium
   User dissatisfaction: Low
   Description: The system shall able to retrieve text when it is needed.

11.1.2 Use Case Analysis

11.1.2.1 Sightless Support

Name	Activate Sightless Support
ID	8A
Actor(s)	User
Overview	User has eye-related health problems and want to use Sightless Support. They activates it, selects features, customizes, and verifies changes. The system notify the reboot and then reboot
Trigger	The user is visually impairments
Pre-conditions	None
Post-conditions	Sightless Support activated
Basic Flow	

Description	This is the main scenarios when the user choose to activate Sightless Support.
Actions	0. User goes through the watch setting. 1. Chooses "Sightless Support" 2. A list of accessibilities are shown 3. User selects their desired features 4. User customizes the features base on their needs and preferences 5. User verifies the customizations 6. System displays reboot notification 7. System reboots
Alternative Flow	
Description	The user activates, selects and customizes but does not verify change.
Actions	0. System notifies the user with the line: "Do you want to keep these changes" 1. User chooses "Yes" 2. System displays reboot notification 3. System reboots
Post-conditions	Sightless Support is activated
Alternative Flow	
Description	User does not want to verify changes

Actions	0. System notifies the user with the line: "Do you want to keep these changes" 1. User chooses "No" 2. System goes back to setting
Post-conditions	Sightless Support is not activated

Name	Default Setting
ID	8B
Actor(s)	User
Overview	User wants to use pre-determined configurations that is recommended by the company.
Trigger	None
Pre-conditions	Sightless Support is activated
Post-conditions	The smart watch uses the default setting
Basic Flow	
Description	This is the main scenarios when the user chooses the default setting.
Actions	0. User scrolls down the list of accessibilies. 1. User chooses "Default". 2. "Visibility Enhancement" and "Voice Control" are automatically enabled and configurated. 3. User verifies their option. 4. System displays reboot notification 5. System reboots
Alternative Flow	
Description	The user sets the mode to default but forgets to verify changes.

Actions	0. System notifies the user with the line: "Do you want to confirm your choices?" 1. User chooses "Yes" 2. System displays reboot notification 3. System reboots
Post-conditions	Sightless Support configures base on the settings.
Alternative Flow	
Description	User changes their mind.
Actions	0. System notifies the user with the line: "Do you want to confirm your choices?" 1. User chooses "No" 2. System goes back to setting
Post-conditions	Sightless Support is deactivated.

Name	Deactivate Sightless Support
ID	8C
Actor(s)	User
Overview	User does not have eye-related health problems and want to deactivate sightless Support. They go to setting, choose "Sightless Support", deactivate it, verify, and the system reboot
Trigger	The user does not have visually impairments or the user does not want this application on
Pre-conditions	Sightless Support is activated
Post-conditions	Sightless Support deactivated
Basic Flow	
Description	This is the main scenarios when the user choose to deactivate Sightless Support.

Actions	0. User goes through the watch setting. 1. Choose "Sightless Support" 2. A list of accessibilities are shown 3. User deactivates it 4. User verifies the deactivation 5. System displays reboot notification 6. System reboots
Alternative Flow	
Description	The user deactivates but does not verify change.
Actions	0. System notifies the user with the line: "Do you want to deactivate Sightless Support?" 1. User chooses "Yes" 2. System displays reboot notification 3. System reboots
Post-conditions	Sightless Support deactivated
Alternative Flow	
Description	User does not want to deactivate
Actions	0. System notifies the user with the line: "Do you want to deactivate Sightless Support?" 1. User chooses "No" 2. System goes back to setting
Post-conditions	Sightless Support is still activated

11.1.2.2 Visibility Enhancement

Name	Set Brightness
ID	9A
Actor(s)	User
Overview	User wants to set the brightness to their preference
Trigger	The user finds the display to be too bright or too dark
Pre-conditions	Sightless Support is activated and Visibility Enhancement is enabled.
Post-conditions	The screen brightness is set to the user's preference
Basic Flow	
Description	This is the main scenarios when the user want to set brightness.
Actions	0. User goes through the watch setting. 1. Choose "Sightless Support". 2. A list of accessibilities are shown. 3. User chooses "Visibility Enhancement". 4. User sets the brightness to their liking. 5. User confirms the modification.
Alternative Flow	
Description	The user makes the change but forgets to verify it.
Actions	0. System notify the user with the line: "Do you want to confirm the change?" 1. User chooses "Yes" 2. Change is applied.
Post-conditions	Brightness is modified.

Name	Set Font
ID	9B
Actor(s)	User
Overview	User wants to set the font to their preference.

Trigger	The user is not comfortable with the font.
Pre-conditions	Sightless Support is activated and Visibility Enhancement is enabled.
Post-conditions	The font is set to the user's preference.
Basic Flow	
Description	This is the main scenarios when the user want to set font.
Actions	0. User goes through the watch setting. 1. Choose "Sightless Support". 2. A list of accessibilities are shown. 3. User chooses "Visibility Enhancement". 4. A list of fonts are shown when the user chooses "Font". 5. User selects the desired font. 6. User confirms the modification.

Name	Magnify
ID	9C
Actor(s)	User
Overview	User wants to magnify components on the screen for better visibility
Trigger	The user finds the components to be too small
Pre-conditions	Sightless Support is activated and Visibility Enhancement is enabled.
Post-conditions	The components are magnified.
Basic Flow	
Description	This is the main scenarios when the user wants to enable magnification.

Actions	0. User goes through the watch setting. 1. Choose "Sightless Support". 2. A list of accessibilities are shown. 3. User chooses "Visibility Enhancement". 4. User chooses "Magnification". 5. User sets the magnification percentage.

11.1.2.3 Braille Display

Name	Enable Braille Display
ID	10
Actor(s)	User
Overview	User is completely blind and wants to read information on their smart watch in Braille.
Trigger	The user wants to use the Braille Display.
Pre-conditions	Sightless Support is activated.
Post-conditions	The smart watch screen turns into a Braille Screen.
Basic Flow	
Description	This is the main scenarios when the user wants to enable Braille Display.

Actions	0. User goes through the watch setting. 1. Choose "Sightless Support". 2. A list of accessibilities are shown. 3. User chooses "Braille Display". 4. User verifies the activation. 5. System notifies the user about the reboot. 6. System reboots.

11.1.2.4 Object Identification

Name	Set Threshold
ID	11
Actor(s)	User
Overview	The user wants to be aware of their surroundings and be alerted to possible dangers.
Trigger	The user wants to set the sensor's threshold for object detection.
Pre-conditions	Sightless Support is activated and Object Identification is enabled.
Post-conditions	The smartwatch can detect objects within the set threshold and alert the user.
Basic Flow	
Description	This is the main scenarios when the user wants to set the threshold.
Actions	0. User goes through the watch setting. 1. Choose "Sightless Support". 2. A list of accessibilities are shown. 3. User chooses "Object Identification". 4. User sets the desired threshold. 5. User confirms the customization.

11.1.2.5 Voice Control

Name	Enable Voice Control
ID	12A
Actor(s)	User
Overview	The user, who may be completely blind, aims for seamless interactions with the smartwatch using their voice.
Trigger	The user wants to interact with the smart watch through their voice.
Pre-conditions	Sightless Support is activated.
Post-conditions	The ability to give command to the smart watch using the user's voice.
Basic Flow	

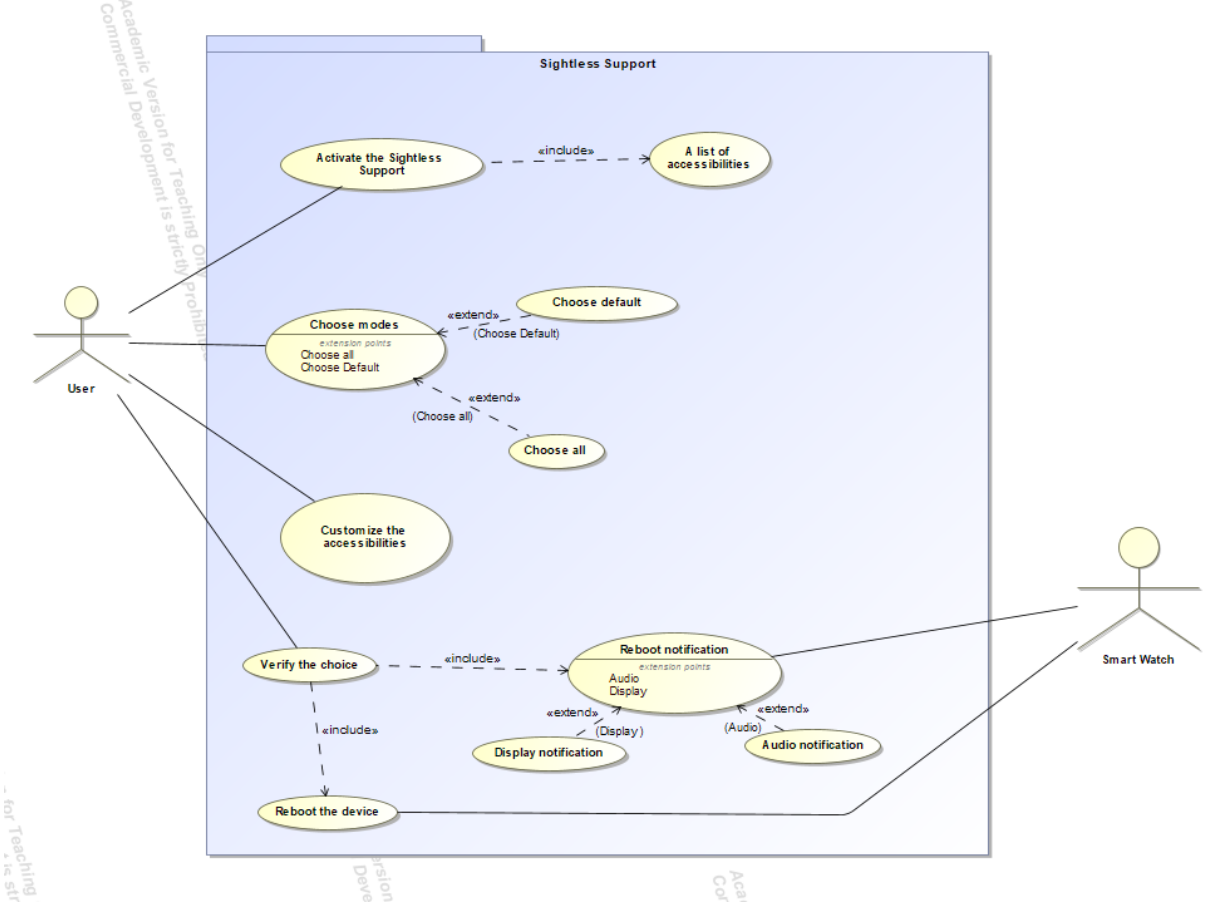
Description	This is the main scenarios when the user want to enable Voice Control.
Actions	0. User goes through the watch setting. 1. Choose "Sightless Support". 2. A list of accessibilities are shown. 3. User chooses "Voice Control". 4. User verifies the activation. 5. System notifies the user of the reboot. 6. System reboots.
Alternative Flow	
Description	The smart watch can not composes instructions base on the user's request.
Actions	0. System informs the user: "I can not understand you. Can you please repeat your request?" 1. User repeats their request. 2. System process the command. 3. Repeat the process if the smart watch still can not compose the instructions
Post-conditions	The instructions are composed and the request is performed

Name	Perform Request
ID	12B
Actor(s)	User
Overview	User wants the smart watch to perform certain actions base on their voice request.
Trigger	The user is not able to manually perform the action at that moment
Pre-conditions	Voice Control is activated.
Post-conditions	The actions is done by the smart watch.

Basic Flow	
Description	This is the main scenarios when the user wants to give instructions.
Actions	0. User gives command to the smart watch. 1. The smart watch process the command into instructions. 2. The smartwatch provides instructions to the related accessibility features. 3. The accessibility then performs the command. 4. The smart watch informs the user by voice when the action is done.

11.2 UML diagrams

11.2.1 Use Case Diagram

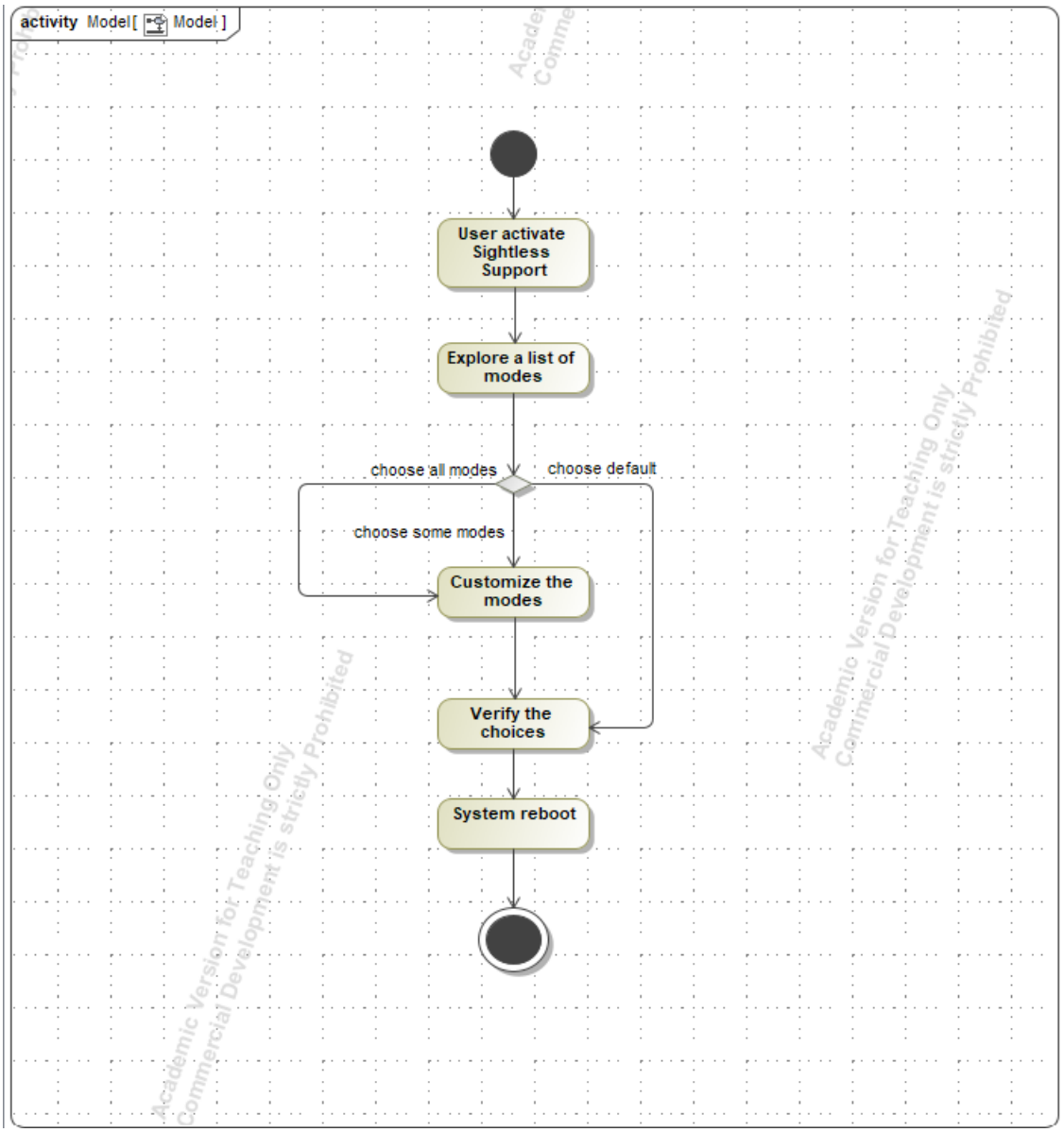


The Sightless Support use-case model explains the interaction between the actor and the system when they want to access and use these capabilities. As seen in the diagram, the model consists of numerous main use cases.

The first step is to switch on the mode. There will be a list of accessibility options offered. Actors can then choose from a number of choices tailored to their individual need. Alternatively, the actor can select a system suggestion or select all possibilities. After selecting the required accessibility choices, the user can further personalize and fine-tune the settings to their tastes.

After setting the accessibilities, the actor is presented with a summary of their chosen configurations. They get the opportunity to examine and confirm the modifications before they are implemented. After the actor confirms the changes, the system initiates a reboot. When it starts the reboot, the system goes through a restart cycle, briefly stopping its functions. During this period, the new accessibility configurations are implemented, and the system's functioning is adjusted accordingly.

11.2.2 Activity Diagram

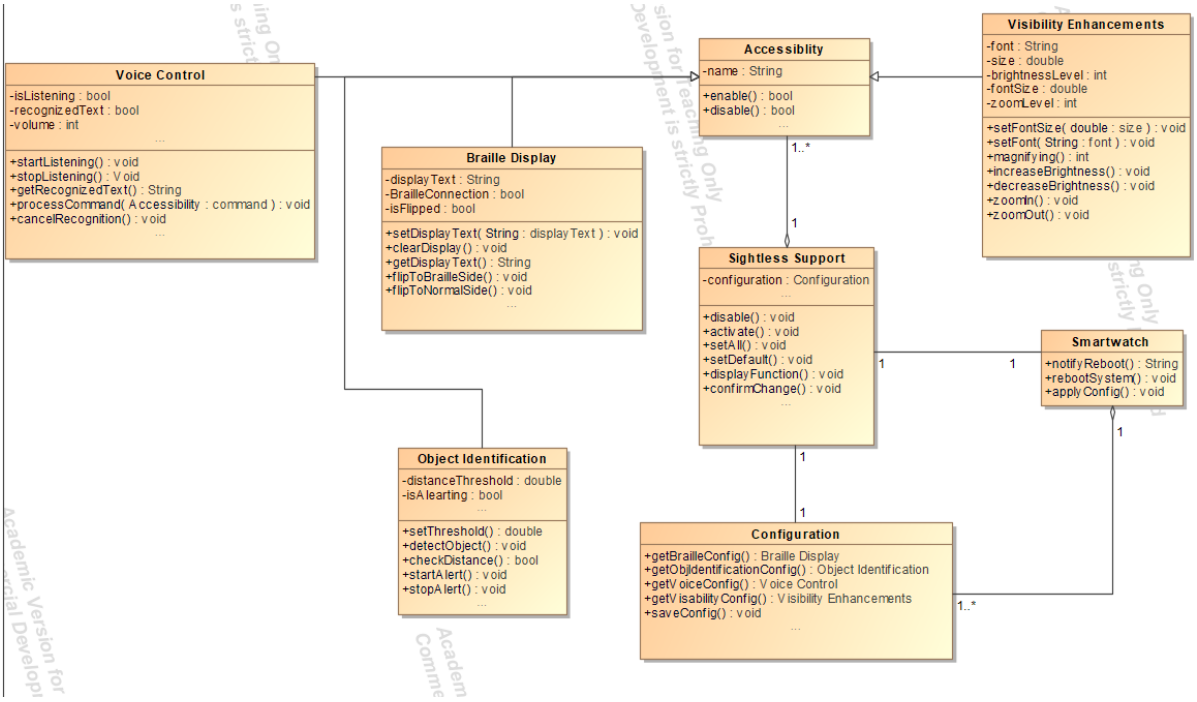


The activity diagram is a sequence of activities that the actor needs to take to use the Sightless Support.

The actor needs to first activate the Sightless Support before doing anything else. Once the sightless support is activated, the actor proceeds to the next activity, "Explore list of modes." Here, they are presented with a list of available options. After exploring the list, the actor moves on to the activity "Choose", where they need to decide whether to "Choose all", "Choose default", or select individual accessibilities. With "Choose all", the actor chooses to enable all the features which is offered by the smartwatch. This option is very useful for people with multiple eye-related health problems. With default mode, the actor will experience predefined settings that are commonly recommended for people with visual disadvantages. With setting up the accessibility manually, actor can tailor themself with features that best suit their needs and preferences.

The next stage is to customize the features. The actor can further adjust the selected accessibility options. This customization may include changes in font size, color contrast, speech speed, and other variables that improve the system's usability and comfort. Once the customisation is complete, the actor must validate the change before the smartwatch may be rebooted and the changes applied.

11.2.3 Class Diagram



There are eight classes in Sightless Support: Sightless Support, Configuration, Smart Watch, Accessibility, Braille Display, Object Identification, Visibility Enhancement, and Voice Control.

The Sightless Support class serves as the main component. It associates with the Smart Watch and Configuration classes, and it has a single relationship with both. It also aggregates with the Accessibility class. Because of this aggregation, the Sightless Support class can hold and manage instances of the Accessibility class. It has a one-to-many relationship with Accessibility as a result.

The Configuration class and the Smartwatch class have a one-to-one aggregation relationship. This allows the Configuration class to control the Smartwatch's behavior inside the system.

The attributes and actions of the Accessibility class are inherited by Braille Display, Voice Control, Object Identification, and Visibility Enhancement. Aside from that, they offer their own distinct features that can assist with certain requirements.

The Braille Display class is used to convert any information and data that the system receives and display it as Braille output.

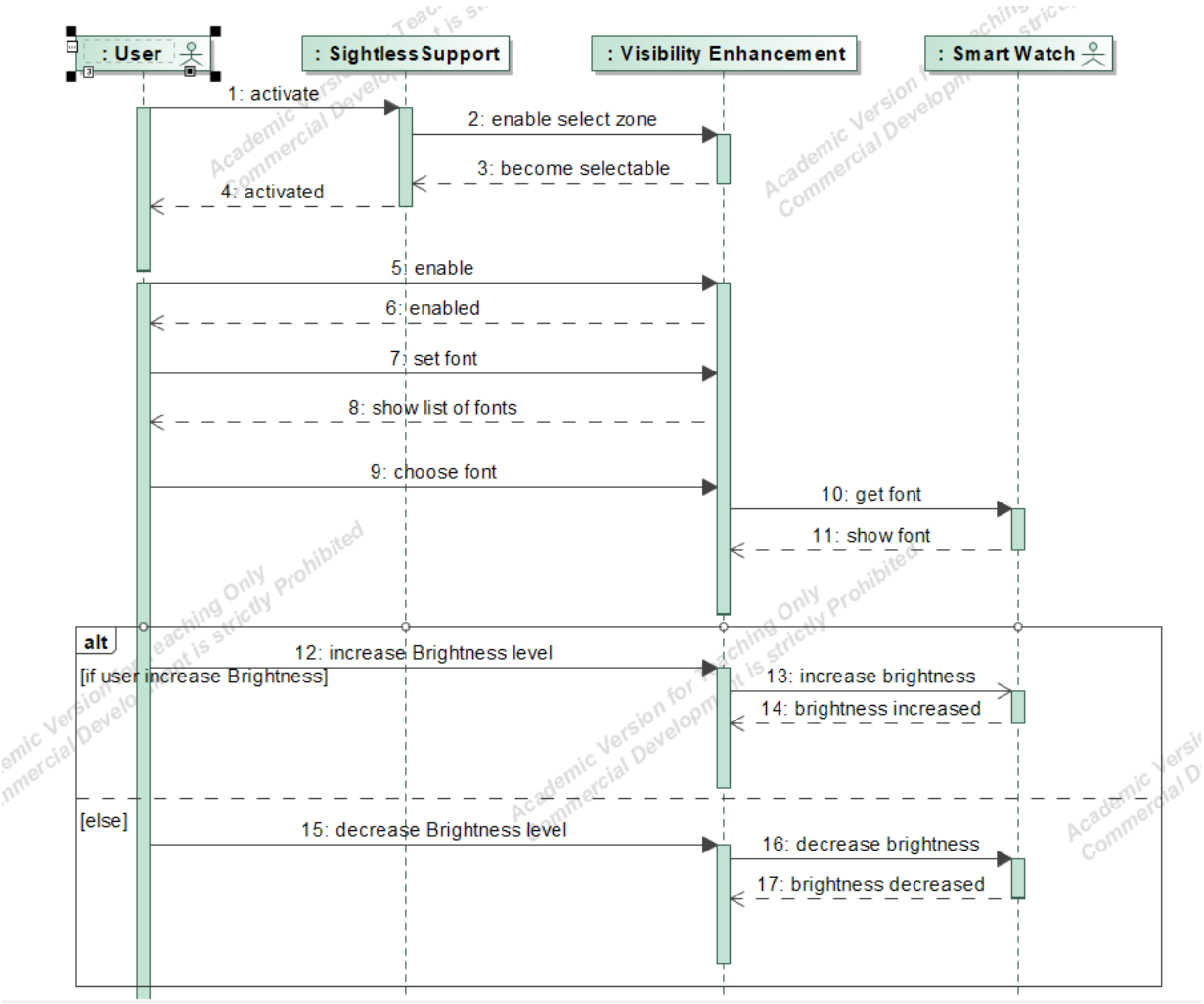
The Voice Control class allows the actor to control the smartwatch with their voice. It has hands-free operation and control capabilities.

The Object Identification class allows the system to recognize and communicate information about items in the actor's immediate surroundings.

The Visibility Enhancement class are used to improve visual perception for those with partially blind. It contains options such as font customization and screen magnification. These operations can help improve the usability and readability of the smartwatch.

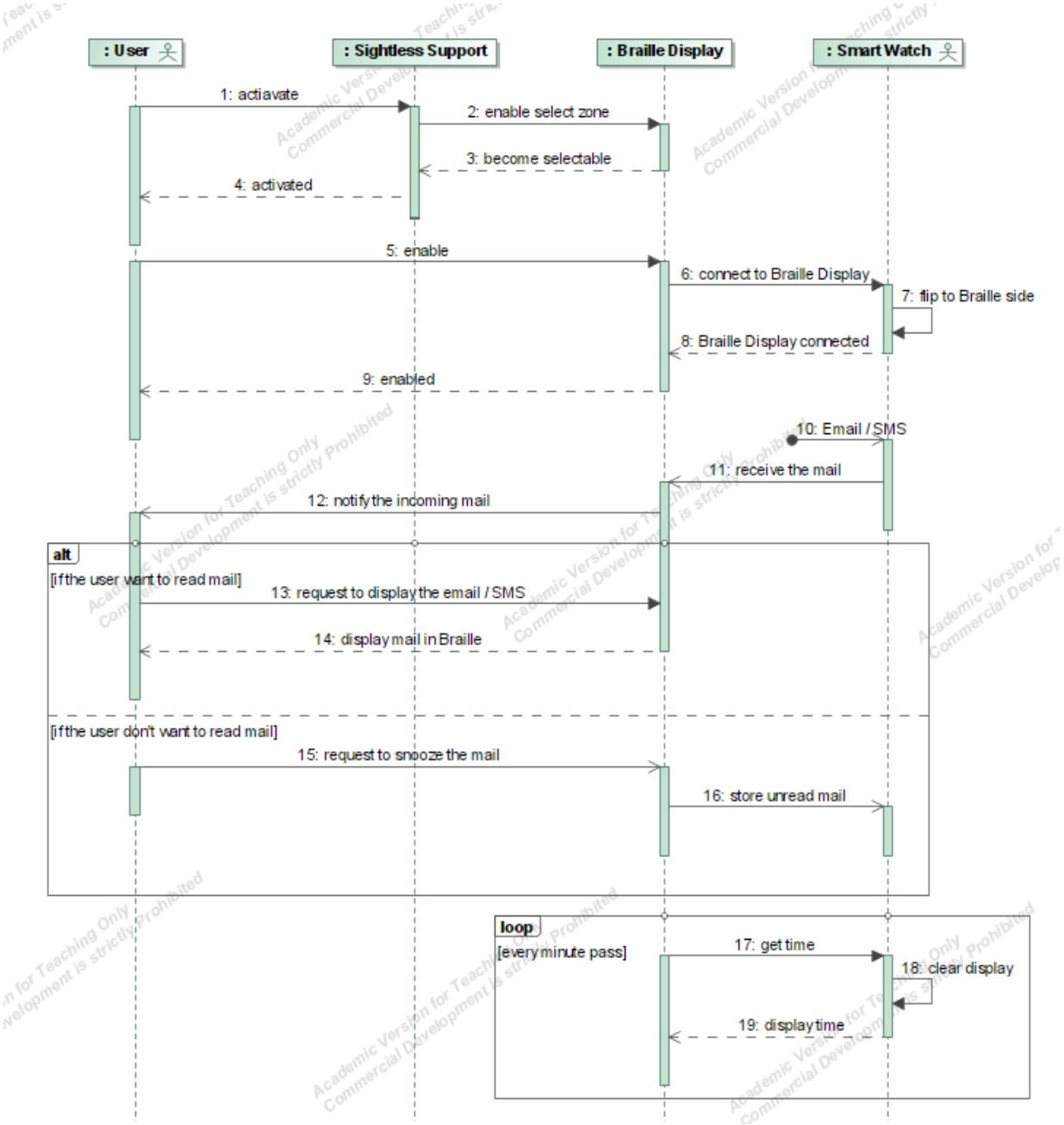
11.2.4 Sequence Diagram

11.2.4.1 Visibility Enhancement



The image above depicts sequence diagram for Visibility Enhancement accessibility, involving a User, Sightless Support, Visibility Enhancement, and a Smart Watch. This is a scenario where the user only want to slightly enhance the visibility. The user activates Sightless Support, which in turn makes the accessibility become selectable. When the user want to change the font, they can simply choose to set font, a list of fonts is then displayed to the user who then chooses one. The Smart Watch component is involved in showing the selected font. There is also an alternate flow based on user choice: if the user chooses to increase brightness, the system increases and confirms the increased brightness level; otherwise, it decreases the brightness and confirms that the brightness level has been decreased.

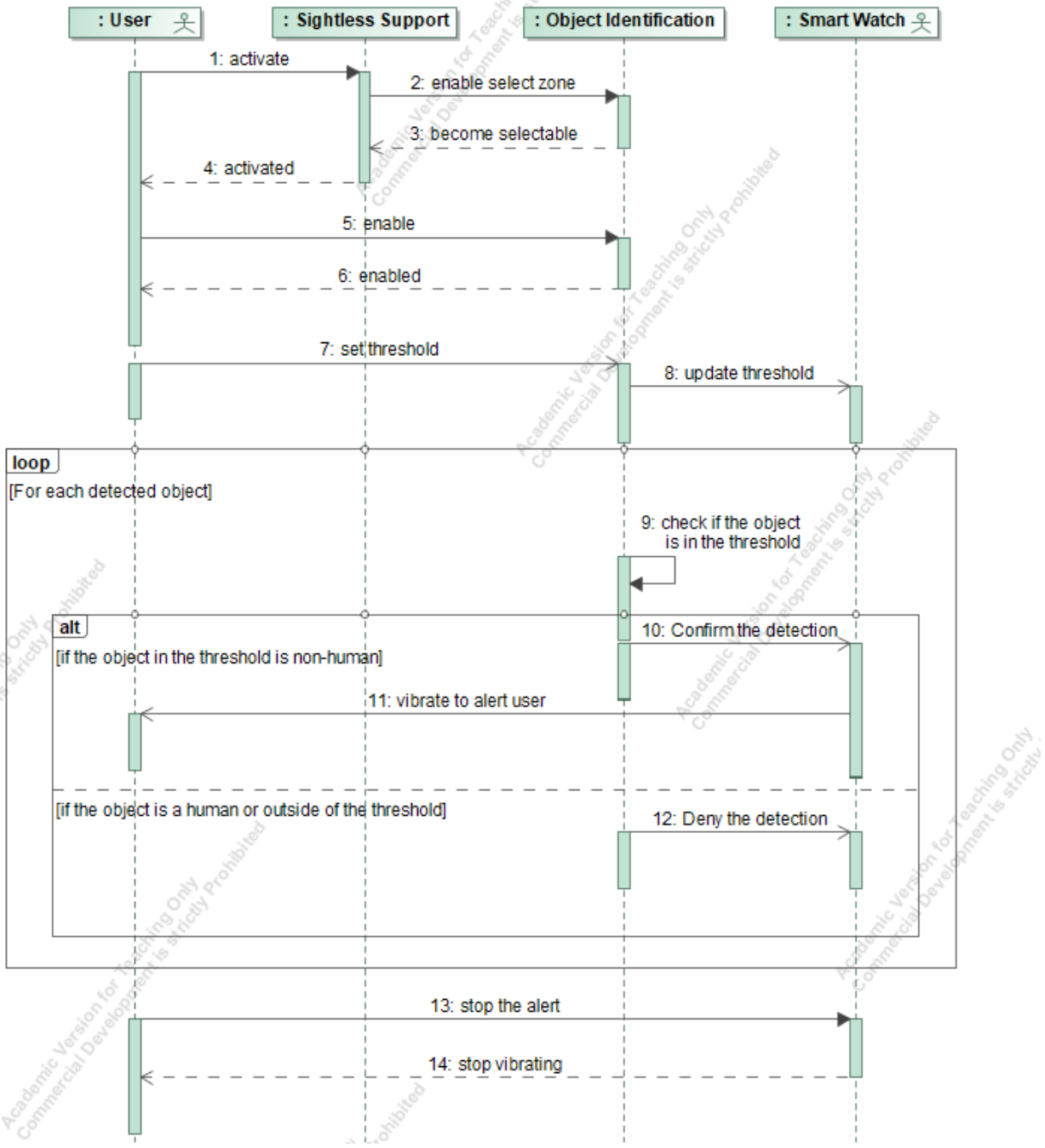
11.2.4.2 Braille Display



This sequence diagram depict the interactions between a user, sightless support, braille display, and smart watch. This diagram demonstrates the process of a user receiving and reading an email or SMS in Braille. Initially, the user activates the sightless support, which in turn enables a select zone for Braille Display to become selectable. Once activated, the user can enable the braille display mode, making the display to switch to Braille. When an email or SMS arrives, the smartwatch receives the mail and notifies the user. The user then has two choices: request the message to be displayed in Braille, which the system then shows, or choose not to read the mail, in which case the system will store the unread mail. Additionally, there's a looping function where the

smart watch periodically gets the time and displays it on the display in Braille.

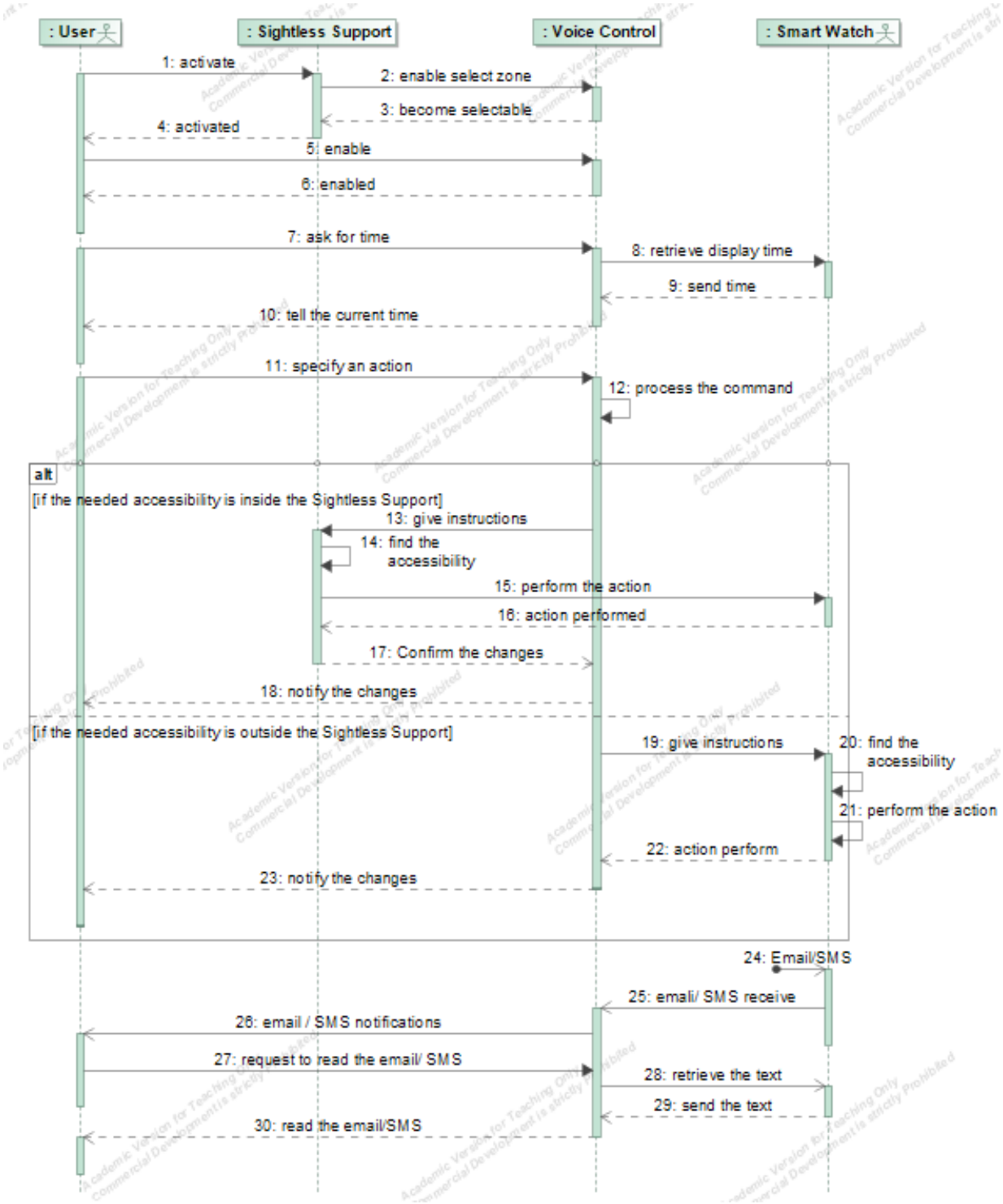
11.2.4.3 Object Identification



This sequence diagram depicts how a smartwatch assists a user in detecting things inside a given threshold. The user begins by activating the sightless assistance, which allows a selection zone for Object Identification to become available. Once enabled, the user can specify a threshold for object detection.

This threshold is updated by the smartwatch, which continuously looks for items. When an object is in the defined threshold and not a human, the smartwatch vibrates to warn the user. If the object falls outside of the threshold or is a human, detection is rejected and no alarm is gone off. The system is meant to continuously checking things in a loop, and the alarm and vibration will stop after the object is no longer detected. Optionally, the user can manually stop the vibrating.

11.2.4.4 Voice Control



In this sequence diagram, we see a user interact with voice control accessibility via a smartwatch. To use the Voice control, the user must first activate Sightless Support. From there, using Voice Control, the user may issue simple orders such as inquiring for the time or instructing the smart watch to read an incoming email or SMS. In addition, the user can designate actions that the Voice Control will perform. If the requested function is within the Sightless Support's capabilities, the Voice Control gives it instructions.

It then locates the function, executes the action, and validates the modifications with the user through voice. If the function is out of Sightless Support's scope, it still delivers instructions and conducts the activity, but follows a different protocol to confirm.

Chapter 12

Summary

After finish writing the project report about the functionalities of Smartwatch, we have familiarize ourselves aobut the usage of UML Diagrams in the tool called Magic System of System Architect. We also improve our teamwork and problem-solving skills after working with the project as a team. Moreover, we have known a lot of Latex command after writing our report completely on the Overleaf framework. Besides, with all the drawn diagrams of each function, users will clearly has a clear understand about how each function works inside the Smartwatch.

Chapter 13

Appendix

13.1 Sprint Meeting 1

Date	15/12/2023
Attendance	Nguyen Truc Linh (Product owner) Pham Phi Long (Scrum master) Dao The Hien (Developer) Dang Hoang Anh Khoi (Developer) Truong Quang Minh (Developer)
Accomplishment	Familiarize about the project.
Obstacle	Some listing features is quite hard to implements.
Objective	Brainstorming more possible functionalities in smartwatch.
Planning for next sprint	Draw diagrams for desired functions.

13.2 Sprint Meeting 2

Date	22/12/2023
Attendance	Nguyen Truc Linh (Product owner) Pham Phi Long (Scrum master) Dao The Hien (Developer) Dang Hoang Anh Khoi (Developer) Truong Quang Minh (Developer)
Accomplishment	Finalizing our smartwatch's functionalities to implement and distributing tasks to every members.
Obstacle	Struggle to familiar with using Magic Draw to draw the UML Diagrams.

Objective	Draw the diagrams for each function in Magic Draw and prepare presentation.
Planning for next sprint	Diagrams inspections and receiving feedbacks from the teacher and group members.

13.3 Sprint Meeting 3

Date	12/1/2024
Attendance	Nguyen Truc Linh (Product owner) Pham Phi Long (Scrum master) Dao The Hien (Developer) Dang Hoang Anh Khoi (Developer) Truong Quang Minh (Developer)
Accomplishment	Present in front of the class about our progress and 2/4 diagrams were drawn.
Obstacle	Try to get familiar with Overleaf to write the Project Report.
Objective	Finish the remaining diagrams and brainstorm for new functionalities.
Planning for next sprint	Inspect the remaining diagrams and start drawing diagrams for new functionalities.

13.4 Sprint Meeting 4

Date	19/1/2024
Attendance	Nguyen Truc Linh (Product owner) Pham Phi Long (Scrum master) Dao The Hien (Developer) Dang Hoang Anh Khoi (Developer) Truong Quang Minh (Developer)
Accomplishment	Finish drawing all the old functionalities diagrams
Obstacle	Try to come up with new smartwatch's features because we do not reach the number of page required.
Objective	Inspect old diagrams and drawing new functionalities
Planning for next sprint	New functionalities examination.

13.5 Sprint Meeting 5

Date	26/1/2024
Attendance	Nguyen Truc Linh (Product owner) Pham Phi Long (Scrum master) Dao The Hien (Developer) Dang Hoang Anh Khoi (Developer) Truong Quang Minh (Developer)
Accomplishment	Old diagrams have been fixed, new diagrams have been drawn
Obstacle	Struggle to draw the diagrams completely because some aspects in the function is quite hard to visualize.
Objective	Finish drawing the remaining diagrams and completing the report
Planning for next sprint	Reviewing new diagrams

13.6 Sprint Meeting 6

Date	2/2/2024
Attendance	Nguyen Truc Linh (Product owner) Pham Phi Long (Scrum master) Dao The Hien (Developer) Dang Hoang Anh Khoi (Developer) Truong Quang Minh (Developer)
Accomplishment	All the new functionalities have been drawn.
Obstacle	After receiving feedback for the preliminary report from Dr. Robert Lokaiczyk, we come up with quite a lot of mistakes from diagrams and the Report.
Objective	We need to review and correct some diagrams and the Report.
Planning for next sprint	Finalize the report.

13.7 Sprint Meeting 7

Date	9/2/2024
Attendance	Nguyen Truc Linh (Product owner) Pham Phi Long (Scrum master) Dao The Hien (Developer) Dang Hoang Anh Khoi (Developer) Truong Quang Minh (Developer)
Accomplishment	All the mistakes that were highlighted in the report were corrected.
Obstacle	None
Objective	Inspect the report before the final submission