A Deliverable 2 Report On:

Alarm Clock System

**CS3500**

**Software Engineering**

2021

Submitted By Team 15:

* Tom O Dwyer – 119330371 [Deliverable 1 Co-ordinator]
* Sean Vaughan – 119361923 [Deliverable 2 Co-ordinator]
* Ben Murphy - 119394463
* Isaac Marti Garcia – 121118251

Submitted On:

27/10/2021

Content Table

[1. Use Cases and Requirements 3](#_Toc86752844)

[1.1 Executive Summary 3](#_Toc86752845)

[1.2 List of Use Cases 4](#_Toc86752846)

[1.4 Requirements 6](#_Toc86752847)

[1.4.1 Functional 6](#_Toc86752848)

[1.4.2 Non-Functional: 6](#_Toc86752849)

[1.5 Requirements Hierarchical Diagram 7](#_Toc86752850)

[2. Design 8](#_Toc86752851)

[2.1 State Transition Diagram 8](#_Toc86752852)

[2.2 Sequence diagrams 9](#_Toc86752853)

[2.3 Data Flow Diagram 13](#_Toc86752854)

[2.4 Entity Relationship Diagram 14](#_Toc86752855)

[3. Implementation 15](#_Toc86752856)

# Use Cases and Requirements

## 1.1 Executive Summary

Today’s world is in a constant state of perpetual movement. Years go by quickly, months pass swiftly and days come to a fast end. For many people, this poses a grave problem as they struggle to live busy working lives while also maintaining a balance with their home lives. Often, time within one life is sacrificed for the other. So, surely there is a solution to this back and forth chaotic living? Yes there is. Time Management. Whether it’s scheduling a compact day of meetings in an office, setting an early alarm in the morning or timing the minutes left until you can get out of bed in the morning, time management has the ability to change the life of those who embrace it, and we provide the tools to achieve this.

Our valuable clock system functions not just as an alarm, but also provides timing and stopwatch functionality. This project simulates the processes of an alarm clock system. We will explore how a clock is designed and how the user can interact with the clock in many ways by exploring the use cases of the clock system and also how there is many requirements or necessities in building an effective clock system. We will also explore how many requirements of a clock system can be prioritized in the sense of how important each requirement of the clock system is and thus we can rank them hierarchically into functional and non-functional requirements.

## 1.2 List of Use Cases

UC1. Check time

The system will constantly display the current time thus allowing the user to view the time at any particular moment.

UC2. Set alarm time

The user will select a desired time from the set of times displayed by the system. This is the time that the alarm will alert the user at.

UC3. Set alarm date

The user will select a date provided by the system which will be the day, month & year that the alarm will ring on

UC4. Set correct Time Zone

The user will choose the correct time zone based on their geographic location to ensure accurate timing of the system

UC5. Deactivate alarm

The system will disable the alarm once the user decides to deactivate it.

UC6. Set alarm tone

The user can select a preferred ringtone provided by the system. Otherwise, the system will set the ringtone to a default ringtone.

UC7. Manage alarm settings

User can configure the deactivation mode of the alarm to snooze or immediate shutoff when they select deactivate.

1.3 Use Case UML Diagram

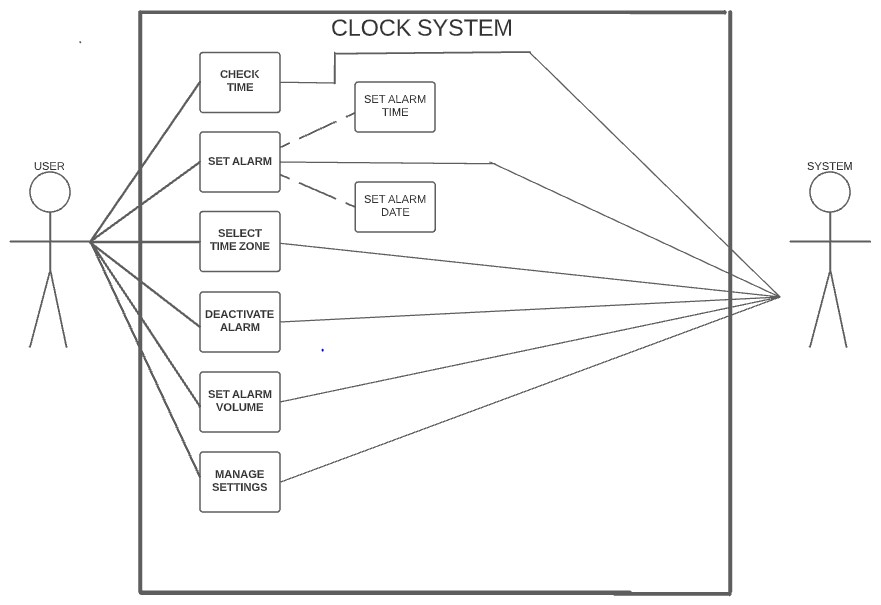


Figure 1: UML Diagram of application use cases

## 1.4 Requirements

### 1.4.1 Functional

1. Display current time in 24hr/12hr format

System must display the current time in the preferred format, either 24hr or 12hr format selected by the user.

1. Ring when desired time is met

System must audibly alert user once the desired alarm time is met.

1. Alarm terminates ringing in certain period of time

System must disable the alarm ringing once the user selects disable or snooze. The user can change the snooze time necessary for them in the alarm settings if needed.

### 1.4.2 Non-Functional:

1. Reliability of alarm system timing

The timing of the alarm when it’s supposed to ring must maintain a high level (99%) of consistency

1. The Clock system must be easily accessible

User Interface must be simple and easy-to-navigate for all users regardless of their physical and mental impediments

1. The clock system must be scalable

High performance must be maintained regardless of workload, which will ensure high functionality and ease of scalability for the system.

1. Audio must ring at level selected by the user.

Audio must be reliable by ringing at the level specified by the user to sustain a constant volume level.

## 1.5 Requirements Hierarchical Diagram

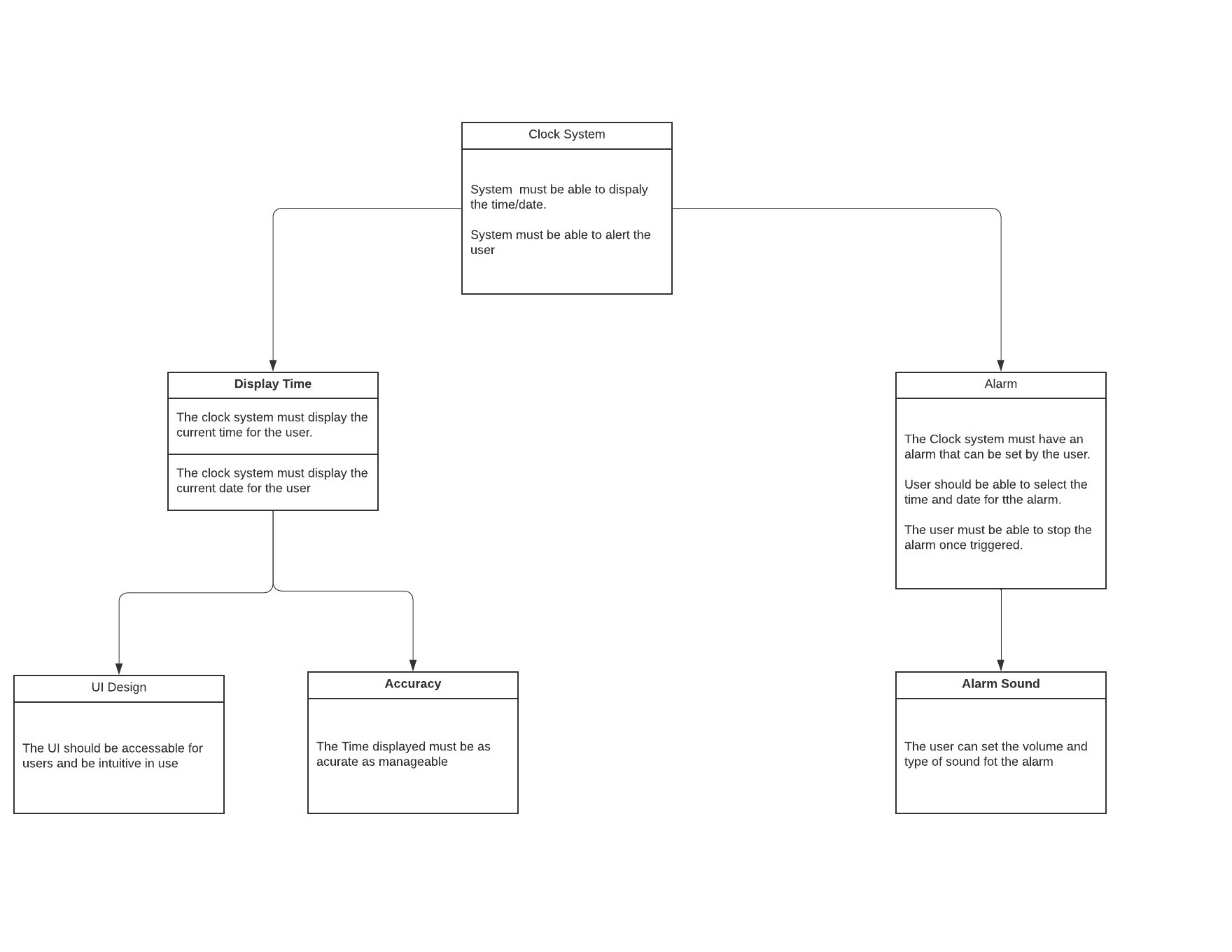
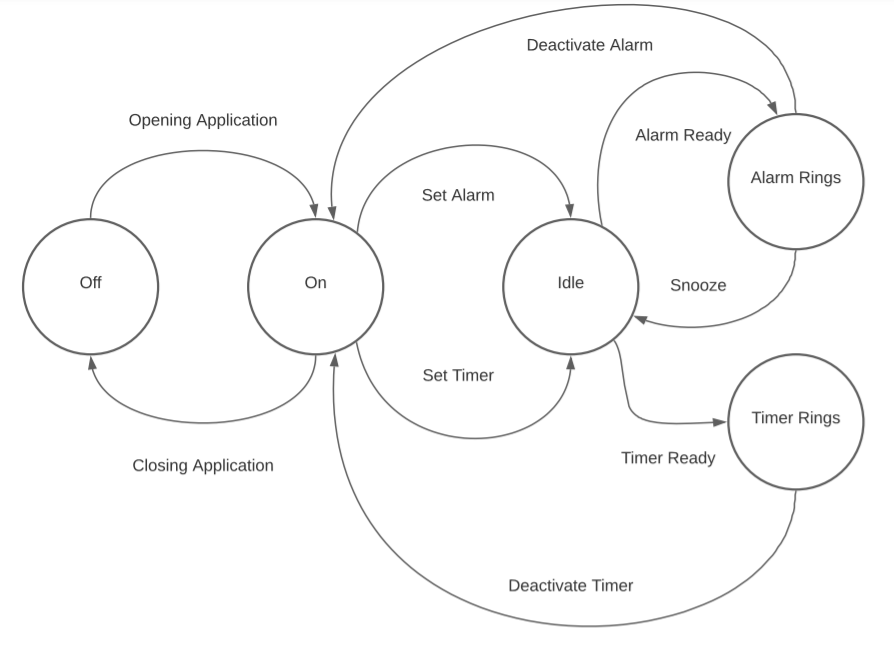


Figure 2: Hierarchical Diagram of application requirements

# 2. Design

## 2.1 State Transition Diagram

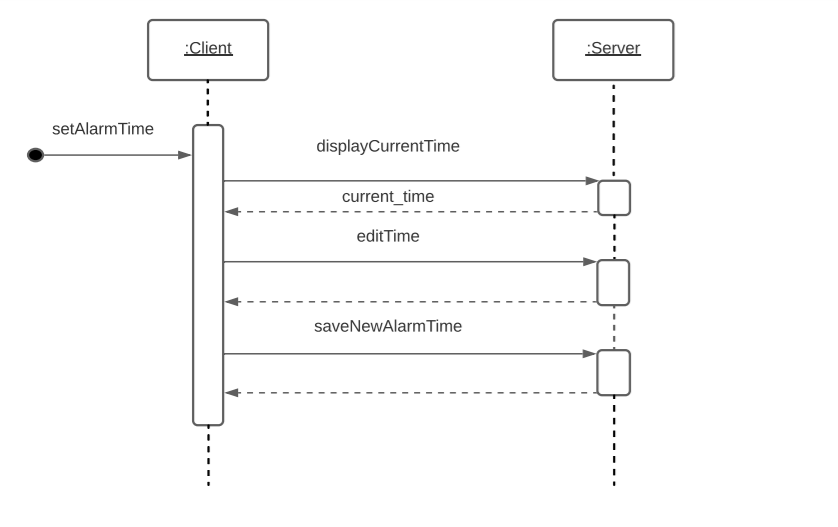
Figure 3 shows the different type of states present in the alarm clock system and how these states transition to one another. An alarm clock system has many states to guide a user in progressing through the system and to inform the user what state they are currently in. For example, the system will help the user to navigate from the home screen (on state) to the timer screen (idle state) so they can interact with the system successfully. The system would also for example inform the user when the alarm is active (alarm rings state) so the user can then transition to the idle state by snoozing the alarm.



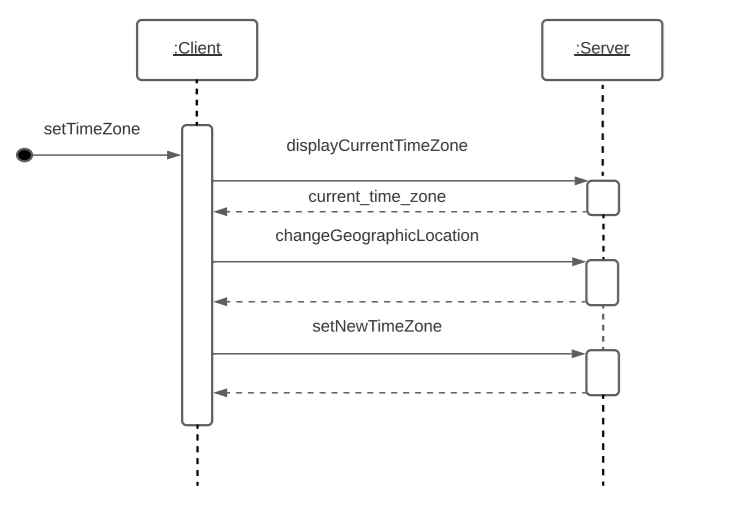
*Figure 3: State-Transition Diagram*

## 2.2 Sequence diagrams

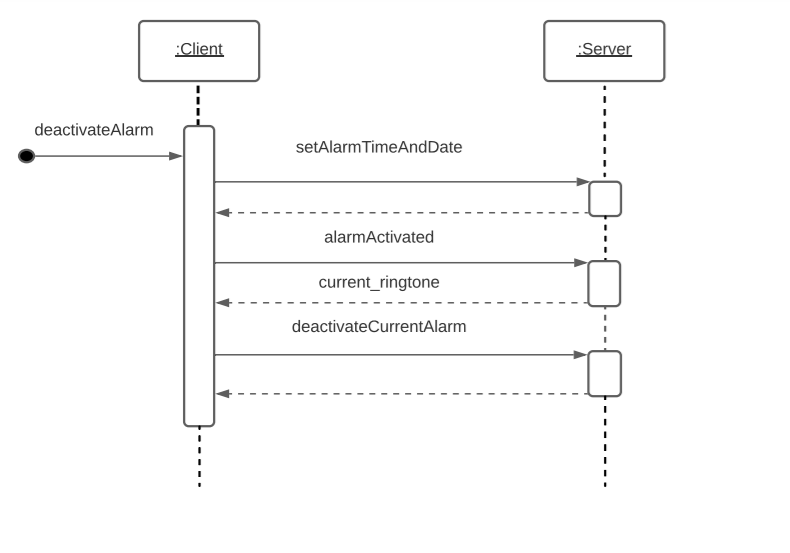
Figures 4 – 7 take a more in-depth view of the use cases defined in the first chapter. We see how a client would interact with the server and how many methods have to be executed before a client can satisfy a specific use case. For example, in Figure 4 below we can see that there are a few extra steps a client has to communicate to the server before they can be satisfied that UC2 (set alarm time) of the system is met such as displaying the current time, editing this current time and then setting the alarm time suitable to them.

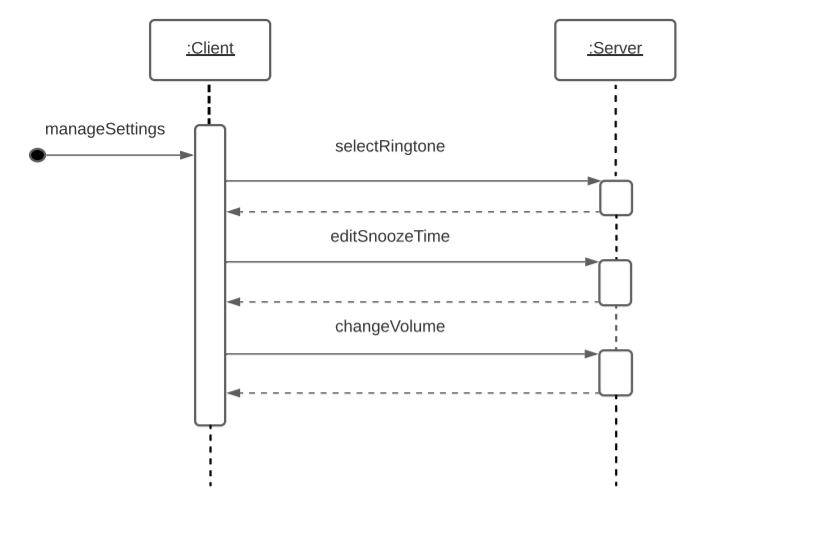


*Figure 4: Sequence Diagram for Use Case UC2*



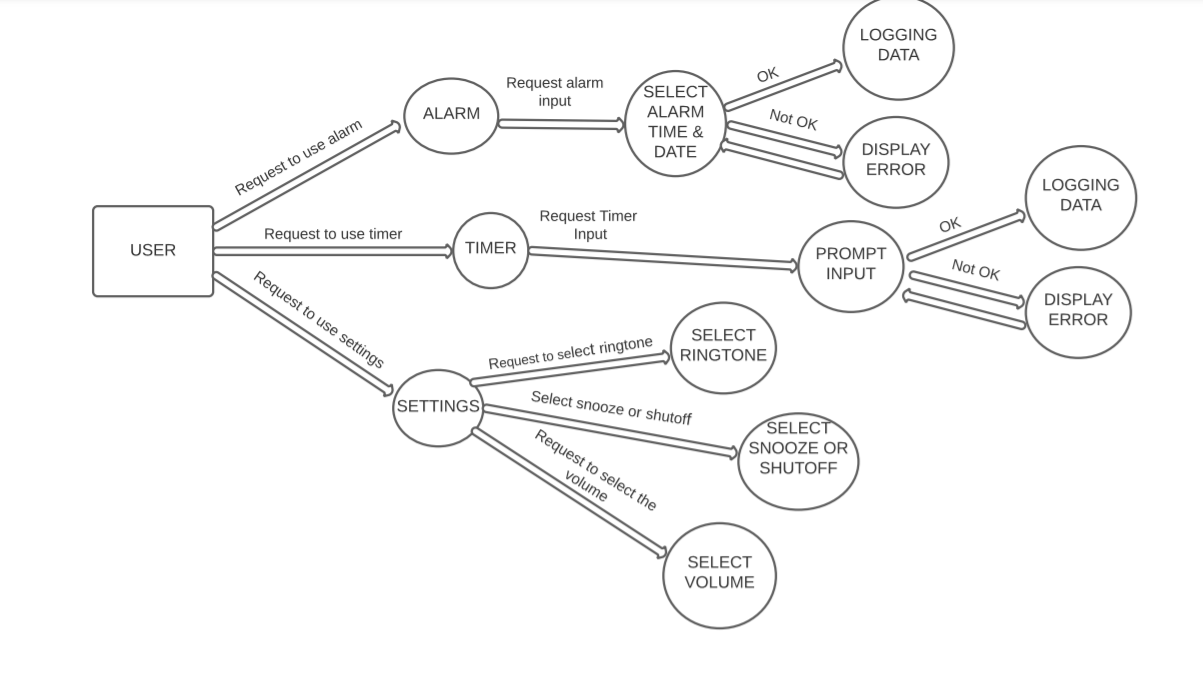
*Figure 5: Sequence Diagram for Use Case UC4*

*Figure 6: Sequence Diagram for Use Case UC5*

*Figure 7: Sequence Diagram for Use Case UC7*

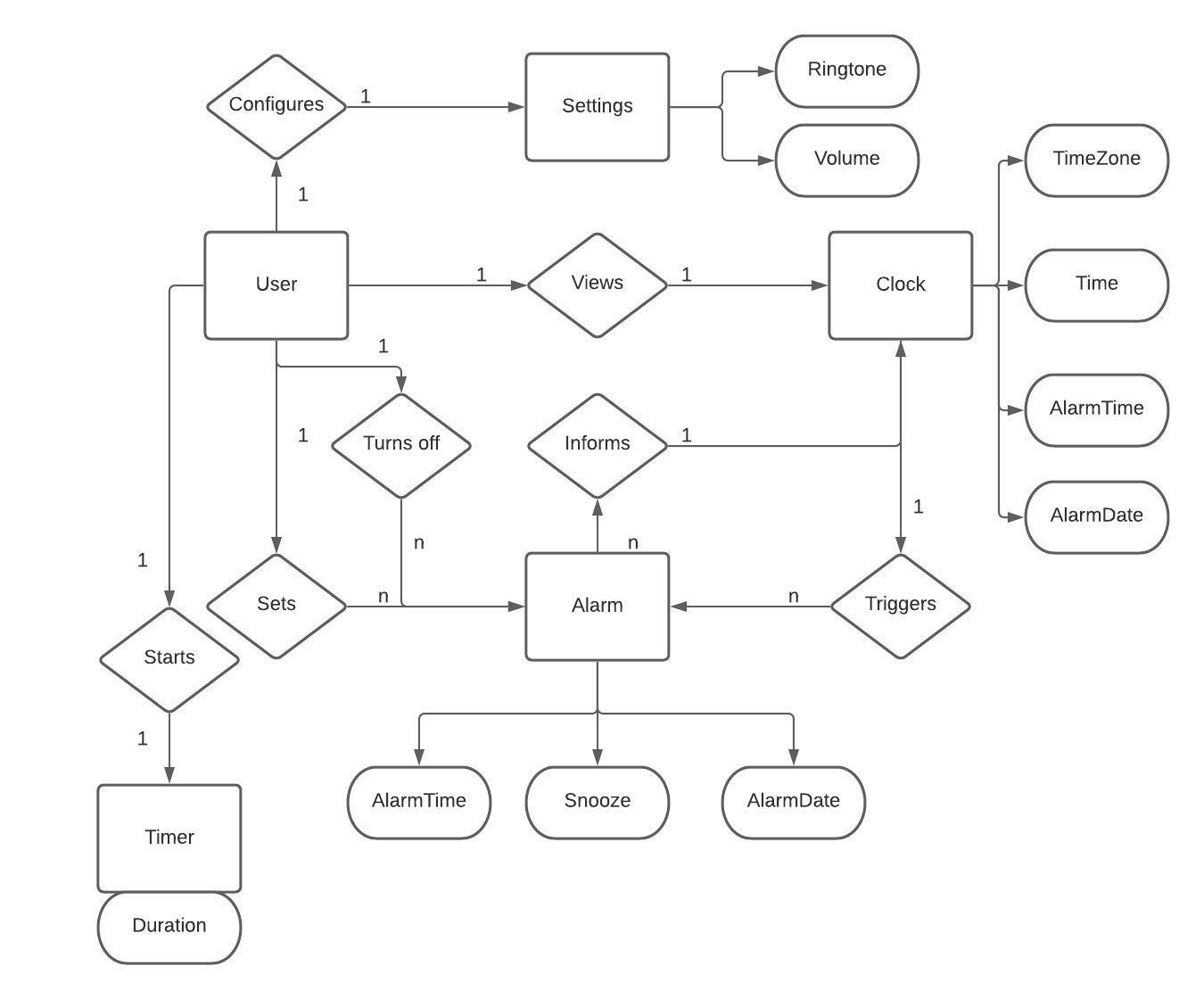
## 2.3 Data Flow Diagram

Figure 8 demonstrates how a data flow diagram is handled by showing the reader the sources present (User, Backend), the processes running (e.g., Alarm, Timer) and the data flow existing in the system (e.g., how the Timer and Prompt Input processes interact with each other by the Timer process sending a message to request a timer input). Figure 8 also provides the reader a graphical representation of how information moves between processes and how a user of this alarm clock system communicates with the backend to gain access to all the data available to them

*Figure 8: Data flow diagram*

## 2.4 Entity Relationship Diagram

Figure 9 displays how an entity relationship diagram operates by informing the reader the entities (e.g., User, Clock), the attributes (e.g., the attributes of the Settings entity are ringtone and volume) and the relationships (e.g., the relationship between the User and Alarm entities is Sets) that are present in the system. Figure 9 also presents to the reader how the data model within the system works and how different entities interact with other entities through relationships. For example, a user would have a relationship with a clock as one user would view one specific clock. But also, a specific entity could have multiple relationships with different processes. For example, as well as interacting with a clock a user also interacts with an alarm, a timer and settings



*Figure 9: Entity relationship diagram*

# 3. Implementation

For the first part of this stage, we implemented on the alarm clock the features that are shown in the first sequence diagram (Figure 4). We implemented the following functions:

* **displayCurrentTime:** This sends a petition to the application, and it answers back with the display of the current time.
* **editTime:** This function allows the user to select the time to set the alarm.
* **saveNewAlarm:** This function is used when the user has selected the time and clicks the button “Save”. It stores the exact time so the alarm rings at the selected time.

For the second sequence diagram (Figure 5), we implemented the following functions:

* **displayCurrentTimeZone:** When the user navigates to the Time Zone section of the system, this function is called and the time zone of their current geographic location will be displayed e.g. GMT will be displayed if their location is Cork, Ireland
* **changeGeographicLocation:** This function allows the user to select which geographic location they want to set their new time zone for.
* **setNewTimeZone:** The system will now set and display the correct time zone for the user depending on the geographic location that they have chosen.

Following with the alarm functionality of our application, next we implemented the third sequence diagram (Figure 6). Here we find these functions:

* **alarmActivated:** This function is called when the alarm starts ringing. When the time set by the user is reached, the alarm starts ringing continuously until the user stops it.
* **deactivateCurrentAlarm:** When the alarm starts ringing the application instantly displays a “Deactivate Alarm” button, when pressed this button stops the ringing of the alarm.
* **SetAlarmTimeAndDate:** When the user has inputted the alarm time and date that they want, the system saves the user’s input to a data file and the systems starts counting down till the alarm has reached the required time and date

For the fourth sequence diagram (Figure 7), we implemented the following functions:

* **selectRingtone:** When this function is called the user will be given a list of ringtones and they can choose which ringtone they like to use for their alarm
* **editSnoozeTime:** The user will be able to change the snooze time for the alarm by selecting a numeric value displayed to them
* **changeVolume:** The user will be able to change the volume of the ringtone of the alarm when the alarm starts ringing by selecting the volume level suitable to them

The state transition diagram (Figure 3) is implemented in the working demonstrator. We begin with the start state which is the application being turned off (i.e., application hasn’t started running yet) and we transition to on state by running the application and we get presented with the “home” screen which shows current time. Figure 3 now shows we can transition to Idle state by setting the alarm **or** by setting the timer. This transition can be implemented by either allowing the user to edit the time and date they require for their alarm and once they click the “set alarm” button they transition to the idle state, or the user can click the timer option in top left of screen and edit the time they like to set for their timer and press the “begin” button to transition to the idle state. In the working demonstrator we know when using the alarm clock, that we have transitioned from the idle state to the alarm rings state when a ringtone starts playing and when using the timer, we know we have transitioned from the idle state to timer rings state when a ringtone starts playing. When using the alarm clock when can either transition from alarm rings to idle state by clicking the snooze button or we can transition from the alarm rings to on state by pressing the deactivate alarm button. When using the timer, we can transition from timer rings to on state by clicking deactivate timer.

The data flow diagram (Figure 8) was implemented in the working demonstrator.