Weather 24/7

Software Design Document

Submitted by:

SH-01, Tahmid Mosaddek

FH-97, Kazi Shadman Sakib

Submitted to:

Dr. Saifuddin Md. Tareeq
Professor & Chairperson
Computer Science and Engineering, University of Dhaka

Dr. Sarker Tanveer Ahmed
Assistant Professor
Computer Science and Engineering, University of Dhaka

1. Introduction

1.1 Purpose

The primary purpose of this Software Design Document is to serve as a base for the architecture and system design for the project manager, and a technical guideline for the project developers. This software design document is a pure representation of software components, interfaces, and data necessary for the implementation phase of the software system, Weather 24/7. This document outlines how the software system will be structured from scratch to meet its requirements.

1.2 Scope

Weather 24/7 is a weather-based android application where the user will get all sorts of weather information for today, weather prediction of the future 30 days, and alerts about bad weather beforehand. The main objective of this system is to provide its daily user with accurate information about the weather. This will be done by scraping data from popular weather websites using the help of APIs. So this software system will be able to provide the weather information of almost any region of the world. So, any user from any part of the world can use this system to know his local weather or search the location of a place to know the weather of a distant place.

1.3 Overview

This section provides a general overview of this software design document.

1. **Introduction**

   This section provides a general overview of the entire software application Weather 24/7. It describes the purpose, main objectives, and goals of Weather 24/7 elaborately and also illustrates the structure of this Software Design Document.

2. System Description

This section provides a general overview of the functionality, context, and design of Weather 24/7. It also gives a brief description of how the Weather 24/7 software system will operate.

3. Design Overview

In this section, a brief introduction and overview of the design are elaborated. This section will help everyone related to the project, a way to illustrate the overall view of a system and to place it into context with external systems. This allows for the reader and user of the document to orient themselves to the design and sees a summary before proceeding into the details of the design.

4. Object Model

This section contains the objects necessary to properly run the system along with the object collaboration diagram of the system.

5. Subsystem Decomposition

In this section, a complete package diagram of the overall system is provided. The description of each subsystem is provided at the end of this section.

6. Data Design

This section describes the overall data structure of the system. A brief description of data is given in this section that are valuable to our software system, along with an elaboratean description of the database that we used.

7. User Requirement and Component Traceability Matrix

In this section, cross reference of user requirement of the Requirement Analysis Document with our software system's subsystem is displayed. It ensures that the software design has the functionalities that the user requires.

1.4 Reference Material

- https://en.wikipedia.org/wiki/API
- https://developer.android.com/studio
- Object-Oriented Software Engineering Using UML, Patterns, and Java™ by Bernd Bruegge & Allen H. Dutoit

1.5 Definitions and Acronyms

- **UI:** Shorts for User Interface.

- **UX:** Shorts for User Experience.

- **API:** An application programming interface (API) is a set of programming codes that queries data, parse responses, and sends instructions between one software platform and another.

- **JAVA:** Java is an Object-Oriented Programming Language.

- **XML:** Extensible Markup Language (XML) is a markup language and file format for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

- **JSON:** Javascript Object Notation is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attributes.

- **Parse JSON:** Parse JSON is the process to parse the JSON file format and use its contained data for the Software System.

- **Unit Testing:** Unit testing is a software testing method by which individual units of source code—sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures—are tested to determine whether they are fit for use.

- **Github:** GitHub is a website for developers and programmers to collaboratively work on code. The primary benefit of GitHub is its version control system, which allows for seamless collaboration without compromising the integrity of the original project.

- **IDE:** An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development.

- **Geolocation Service:** Geolocation Service is the process of determining the location of a computer, phone, or other network-based devices. This inferred location is based on geographical measurements of latitude and longitude to narrow down the location to city, zip code, street, and even address.

- **Target SDK version:** The target SDK version is the version an application was targeted to run on.

2. System Description

This software application will serve its users simply by providing the users with daily and future 30 days weather information. It is a weather-based android system where the user will get all sorts of weather information. Users can also search for location-specific weather information. The software application also aims to alert its users, if there is going to be extreme or hazardous weather today by sending a notification beforehand. To get user-specific local weather data, we will search for APIs that suit the demand of this project. The settings option allows the users to customize the application according to their needs.

The backend of the system will be built using Java. If the user does not provide any location, the system will take the user's location provided by the Geolocation API. Otherwise, the location given by the user will be used. This location information will be sent to the selected weather APIs which will collect weather data of that location from the mentioned web servers.

3. Design Overview

3.1 Design Rationale

In this section, we will be discussing a set of reasons or a logical basis behind some of the choices that we made while structuring the design pattern of this software system. The system architecture of the application has been planned and constructed based on the system design's eight issues, such as: identifying the design goals, subsystem decomposition, identifying concurrency, hardware/software mapping, persistent data management, global resource handling, software control, and boundary conditions. The software design of this system mainly focuses on the solution domain. Design goals like reliability, modifiability, maintainability, understandability, adaptability, reusability, efficiency, portability, traceability of requirements, fault tolerance, cost-effectiveness, robustness, high-performance as well as well-defined interfaces, user-friendliness, ease of learning, ease of remembering and flexibility, etc have been meticulously observed to ensure that our users can easily utilize the application and get the greatest user experience possible.

Our software system is a weather-based android application where the user will get all sorts of weather information for today, weather prediction of the future 30 days, and alerts about bad weather beforehand. If the user does not provide any location, the system will take the user's location provided by the Geolocation API. Otherwise, the location given by the user will be used. This location information will be sent to the selected weather APIs which will collect weather data of that location from the mentioned web servers.

All the weather information is stored in the database SQLite, which is a very efficient database. Weather information such as current temperature, how the outside feels like, day and night highest temperature, weather signs, precipitation, humidity, dew point, pressure, UV index, visibility, wind velocity, sunrise and sunset, next 30 days weather information etc are all going to be stored in the SQLite database to ensure better user experience.

Considering all of the above cases, it can be stated that the Client/Server Architectural style/pattern is to be used for the application since the APIs that provide the weather data of a specific location acts as the server, and our software system acts as the client in the Client/Server Architectural style/pattern. The client will only handle the response from the API server and scrape the weather data accordingly. The user interface of Weather 24/7 will display all the weather information and alert its users according to the information provided.

3.2 System Architecture

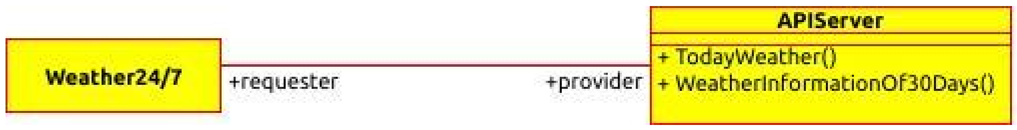
In this section, the elaborated explanation of the system architecture of Weather 24/7 is given. The system design of Weather 24/7 is based on LOW coupling and the HIGH cohesion, and Decentralized Design concepts. The software system's architecture is partitioned into several subsystems while ensuring concurrency across modules. As previously mentioned, since the APIs that provide the weather data of a specific location act as the server and our software system acts as the client, we are using the Client/Server Architectural style/pattern for our software system design.

Our Weather 24/7 software system is a weather-based android application.

Thus, to get proper and accurate weather data, the software system acts as the client and calls for an API that provides the weather data. Here the API acts as the server in our system's architectural pattern/style. For our software system's purpose, we are using a weather-based API server that responds with current weather-based information.

Weather 24/7 software system acts as the client and calls on the API server, which then performs by returning an HTTP GET request result. Our Weather 24/7 system knows the interface of the API server. The API server response comes immediately for our software system. The response result contains JSON format data in which all the weather information is given.
The JSON formatted data will be kept in a database to improve the user experience. The reason for this is that if the user does not visit a new location or if the data from the API is not modified, a proxy server called cache will detect this and retrieve the relevant data from the database. As a result, the user experience will be enhanced by more timely and accurate weather information. If the user does visit a new place or if the data from the API is modified, the proxy server named cache will detect it and the cache will request the API server for the new weather information. The new JSON formatted weather information is going to be parsed and processed, to store the updated relevant weather information to the database.

Thus, by scraping the result, weather information is displayed. Users generally interact only with our Weather 24/7 software system which is acting as a client in this software system's Client/Server Architectural Style.



3.3 Constraint and Assumptions

3.3.1 List of Assumption

The software is built based on the following assumptions:

- The device is assumed to be connected to the internet to fetch data from external web servers.
- The users of this application are assumed to use the English language to insert the location of a place.
- The users are assumed to understand the English language, as every piece of information will be shown in this language.
- The software application requires a minimum of Android API level 23+, also the users are expected to be running at least Android version 6.0 on their devices.

3.3.2 List of Dependencies

The software is built based on the following dependencies:

- Geolocation API to get access to the user's location.
- Weather APIs to fetch data from external web servers.

4. Object Model

4.1.1. WeatherApiController

Class Name	WeatherApiController
Brief Description	Uses the weather APIs to get weather data from the web servers.
Attributes	**Attribute Description**
location	User-provided location or the current location of the user
url	Contains the address of the weather API. The weather API is accessed from the URL.
Methods	**Method Description**
getJsonData()	Uses the weather APIs to get raw JSON data from the web servers.
parseJsonData()	Parse the raw JSON data.
pushToDB()	Push data into the database.

4.1.2. WeatherDataBase

Class Name	WeatherDataBase
Brief Description	Create tables according to the database schema and maintain the database tables.
Attributes	**Attribute Description**
Methods	**Method Description**
createTable()	Create necessary tables

	according to the schema.
insertData()	Insert data into a given relation.
fetchData()	Query on a given relation and get data.
deleteData()	Delete data from a given relation.

4.1.3. Cache

Class Name	Cache
Brief Description	Stores all the data when the application is closed
Attributes	**Attribute Description**
userLocation	Last user location.
userProvidedLocation	Last user-provided location in the search bar.
settings	Last saved settings configuration.
todayBasicWeather	Last shown homepage weather data.
Methods	**Method Description**
loadUserLocation()	Loads previously saved user's location
loadUserProvidedLocation()	Loads previously saved user-submitted location
loadSettings()	Loads previously saved system configuration.
loadTodayBasicWeather()	Loads saved home screen data
saveUserLocation()	Saves user's current location

saveUserProvidedLocation()	Saves user-submitted location
saveSettings()	Saves user-selected settings
saveTodayBasicWeather()	Saves home screen information

4.1.4 Settings

Class Name	Settings
Brief Description	Let the user choose the unit of the weather parameters
Attributes	**Attribute Description**
temperatureUnit	Unit of temperature, Celsius or Fahrenheit.
pressureUnit	Unit of pressure, Atmospheric Pressure(ATM) or Pascal.
timeFormat	Format in which a time of the day is expressed.
Methods	**Method Description**
setTemperatureUnit()	Sets the temperature unit according to the user’s choice.
setPressureUnit()	Sets the pressure unit according to the user’s choice.
setTimeFormat()	Sets the time format according to the user’s choice.

4.1.5. LocationController

Class Name	LocationController
Brief Description	Contains a brief description of the user's current location and user-provided location in the location search bar.
Attributes	**Attribute Description**
userLocation	User’s current location
userProvidedLocation	Get location from the location search bar if provided
Methods	**Method Description**
getUserLocation()	Uses the geolocation API to get the user’s location. If no internet connection is provided then the cached data is used.
getUserProvidedLocation()	Uses the location provided in the location search bar.

4.1.6 Home

Class Name	Home
Brief Description	Shows today’s basic weather information and a location search bar.
Attributes	**Attribute Description**
-locationSearchBar	A search bar view
-temperature	Current temperature
-humidity	

weatherStatus	Contains weather status/signs such as rainy, haze, cloudy or sunny.
Methods	**Method Description**
getProvidedLocation()	Gets user-provided location from the location search bar.

4.1.7. WeatherPrediction

Class Name	WeatherPrediction
Brief Description	Shows the predicted weather data
Attributes	**Attribute Description**
calendar	A view model of a composite class of calendar, which contains a list of daily weather data.
temperature	Current temperature
humidity	Current humidity
sunset	Current day sunset time
sunrise	Current day sunrise time
Methods	**Method Description**
getWeatherPrediction()	Get the weather prediction for a certain day.

4.1.8. CurrentWeather

Class Name	CurrentWeather
Brief Description	Shows detailed data about current weather.
Attributes	**Attribute Description**
temperature	Current temperature
humidity	Current humidity
realFeel	Shows the temperature of how it feels like
pressure	Current pressure
windSpeed	Contains the speed of current wind flow
uvIndex	Current UV index
chanceOfRain	Contains a percentage showing how it is likely to rain.
airQualityIndex	Contains information about the quality of air
todayMaxTemp	Contains today’s maximum temperature along with the time.
Methods	**Method Description**
getWeatherStatus()	Returns the weather status saying if it is cloudy, sunny, or clear outside.

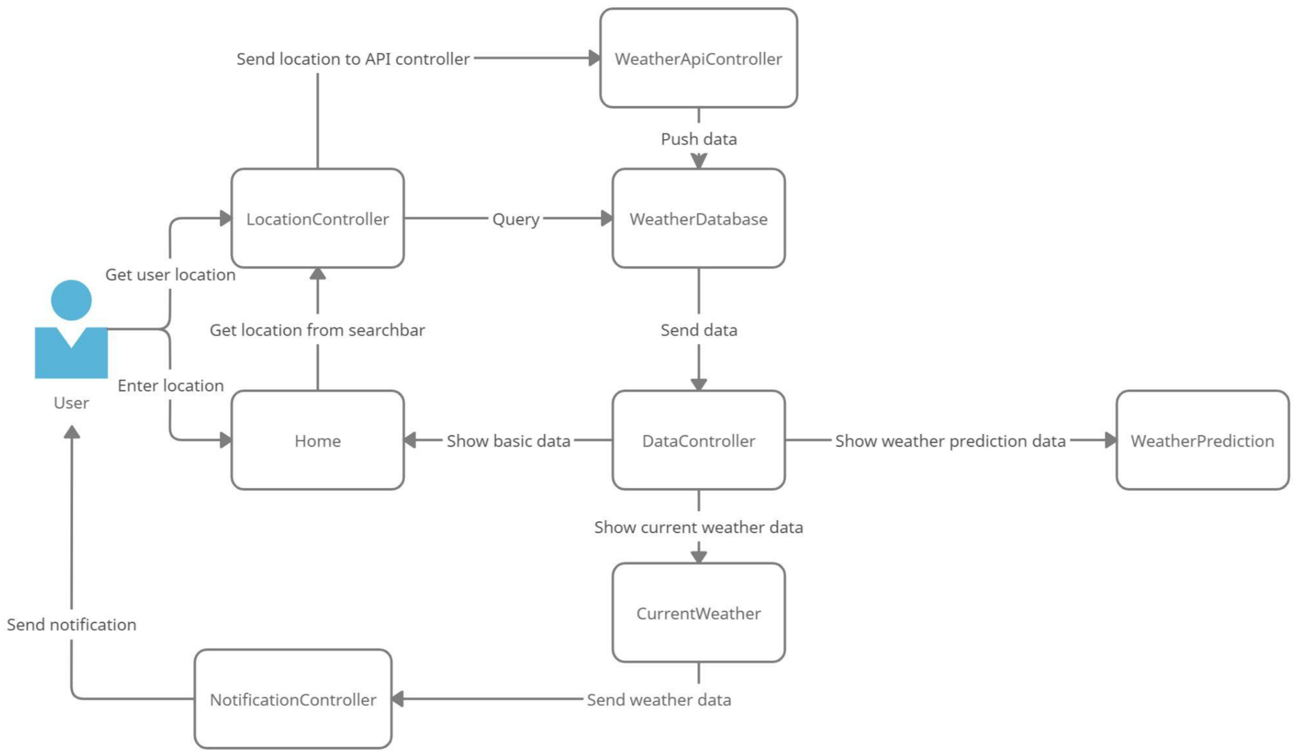
4.1.9. NotificationController

Class Name	NotificationController
Brief Description	Sends users a notification based on the data of current weather and predicted weather.
Attributes	**Attribute Description**
weatherStatus	Describes the overall condition of the current weather.
Methods	**Method Description**
sendNotification	Sends a notification to the user according to the weather status.

4.1.10. DataController

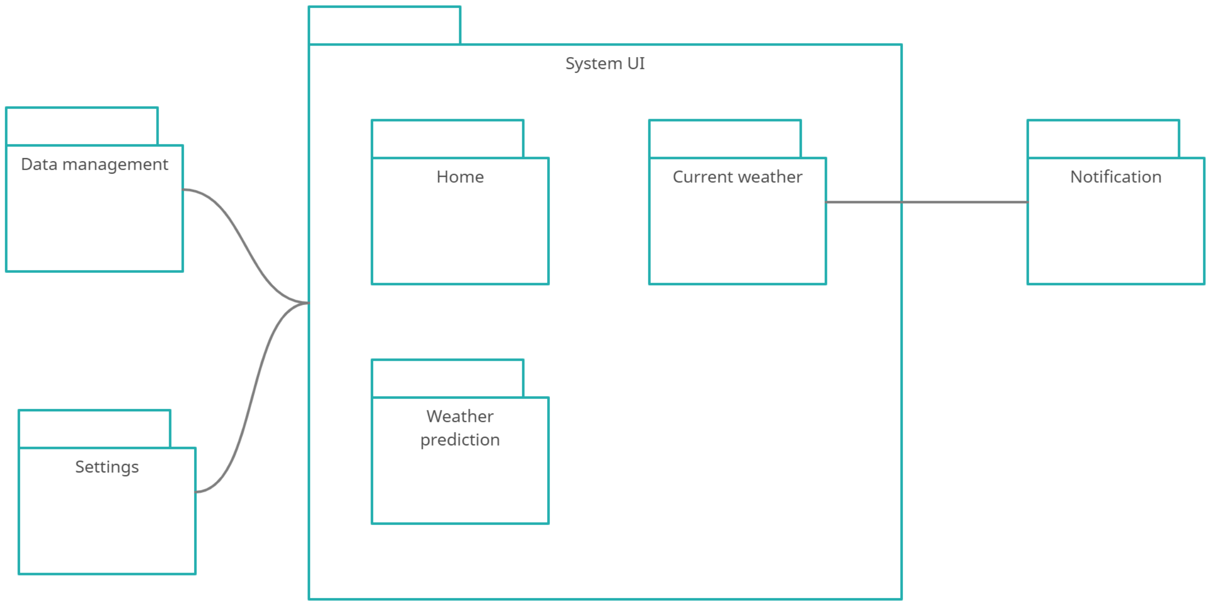
Class Name	DataController
Brief Description	Shows the predicted weather data
Attributes	**Attribute Description**
weatherData	A composite data type containing several weather pieces of information.
Methods	**Method Description**
sendCurrentData()	Sends current weather data
sendPredictionData()	Sends predicted weather data

4.2. Object Collaboration Diagram



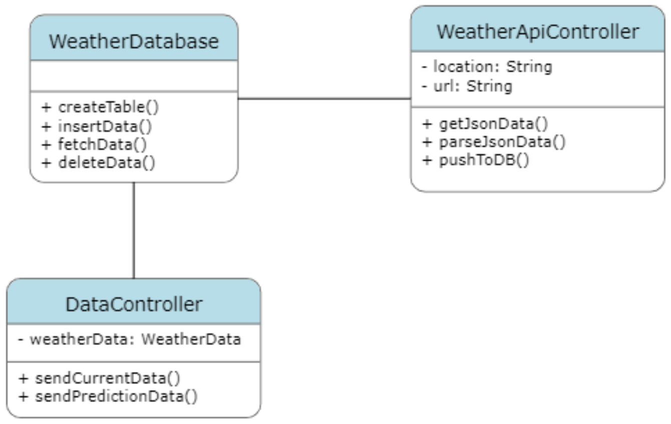
5. Subsystem Decomposition

5.1. Complete Package Diagram

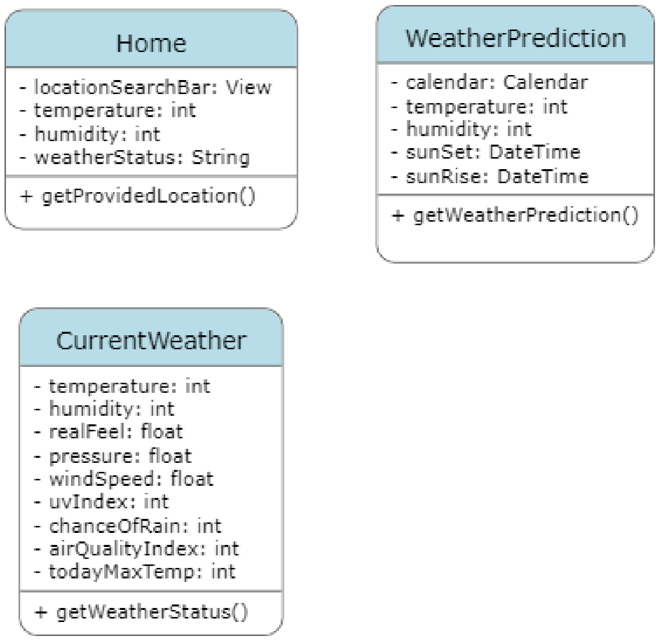


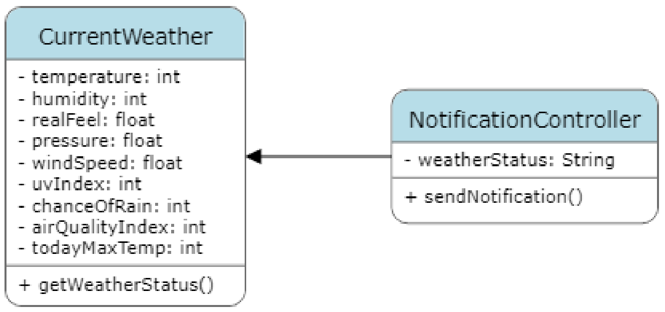
5.2. Subsystem Detail Description

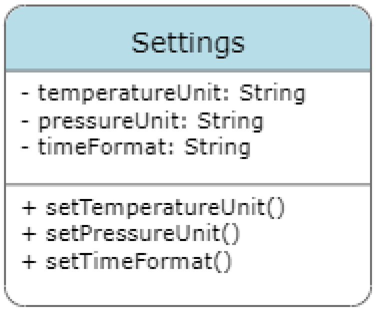
5.2.1.	**Data management**
5.2.1.1.	**Module description**
	This module handles the API controller to fetch data from the web servers, store it in the database and send it to the data controller to show in the user interface.
5.2.1.2.	**Class diagram**

	 WeatherDatabase
	+ createTable()
	+ insertData()
	+ fetchData()
	+ deleteData()
	WeatherApiController
	- location: String
	- url: String
	+ getJsonData()
	+ parseJsonData()
	+ pushToDB()
	DataController
	- weatherData: WeatherData
	+ sendCurrentData()
	+ sendPredictionData()

5.2.1.3.	**Subsystem interface**
	This subsystem is hidden from the user. User requested data is stored and maintained using this subsystem.

5.2.2.	**System UI**
5.2.2.1.	**Module description**
	This module is used to present the user with accurate weather information
5.2.2.2.	**Class diagram**
	 Home - locationSearchBar: View - temperature: int - humidity: int - weatherStatus: String + getProvidedLocation() WeatherPrediction - calendar: Calendar - temperature: int - humidity: int - sunSet: DateTime - sunRise: DateTime + getWeatherPrediction() CurrentWeather - temperature: int - humidity: int - realFeel: float - pressure: float - windSpeed: float - uvIndex: int - chanceOfRain: int - airQualityIndex: int - todayMaxTemp: int + getWeatherStatus()
5.2.2.3.	**Subsystem interface**
	The user directly interacts with the home screen to see the basic weather info of the local area and can enter a location in the search bar to check the weather information of the provided area. The user can also see data about predicted and current weather in the predicted weather screen and current weather screen respectively.

5.2.3.	**Notification**
5.2.3.1.	**Module description**
	This module is responsible for sending notifications about weather status to the user.
5.2.3.2.	**Class diagram**
	 CurrentWeather - temperature: int - humidity: int - realFeel: float - pressure: float - windSpeed: float - uvIndex: int - chanceOfRain: int - airQualityIndex: int - todayMaxTemp: int + getWeatherStatus() NotificationController - weatherStatus: String + sendNotification()
5.2.3.3.	**Subsystem interface**
	The user gets a notification from the system about the current weather status.

5.2.4.	**Settings**
5.2.4.1.	**Module description**
	This module is responsible for changing the configuration of the whole system, this includes changing the units of showing data.
5.2.4.2.	**Class diagram**
	 Settings - temperatureUnit: String - pressureUnit: String - timeFormat: String + setTemperatureUnit() + setPressureUnit() + setTimeFormat()
5.2.4.3.	**Subsystem interface**
	The user gets to choose the units of the weather parameters according to their preference.

6. Data Design

6.1 Data Description

Weather 24/7 is basically a weather-based software system. The system will act as the client and request an API server to respond with the current dated weather information and future 30 days weather information. The fetched data will be in JSON format. Thus, we have to parse and process the JSON formatted data and collect the relevant weather information, to show it to the users. The faster the API responds with relevant data, the better and more accurate weather information will be provided to the user.

To make the user experience better, the JSON formatted data will be stored in a database. The reason behind this is if the user does not visit a new place or the data from the API is not modified, then a proxy server named cache will detect it and fetch the required data from the database. Thus making the user experience better with faster and more accurate weather information.

If the user does visit a new place or if the data from the API is modified, the proxy server named cache will detect it and the cache will request the API server for the new weather information. The new JSON formatted weather information is going to be parsed and processed, to store the updated relevant weather information in the database.

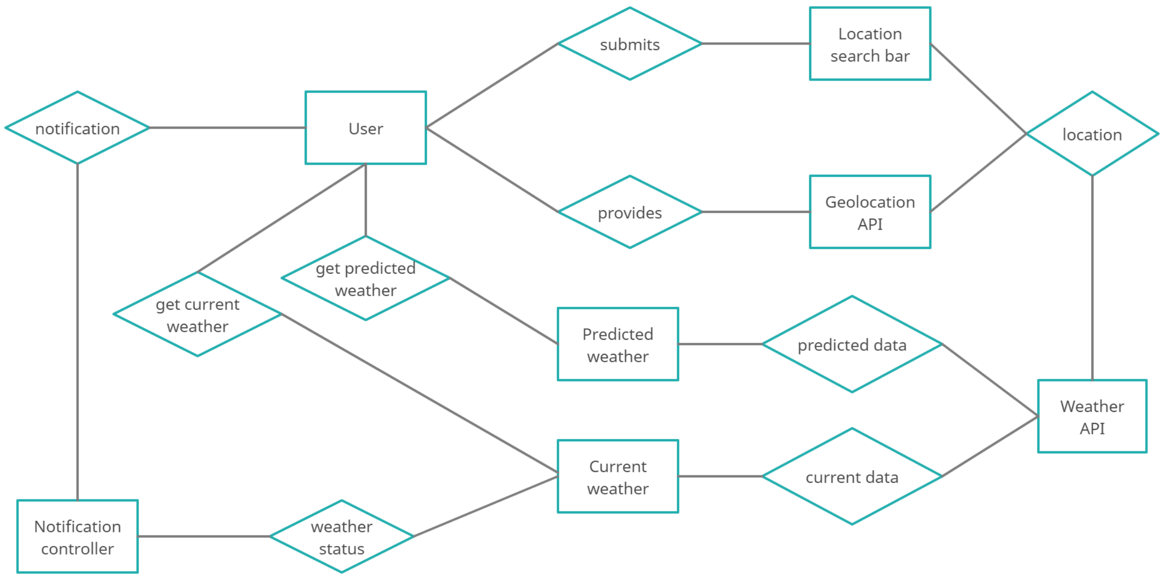
Every piece of data is stored in the SQLite database, which is an extremely efficient database. To provide a better user experience, weather information such as current temperature, how the outside feels, day and night highest temperature, weather signs, precipitation, humidity, dew point, pressure, UV index, visibility, wind velocity, sunrise and sunset, and next 30 days weather information will all be stored in the SQLite database.

6.2 Data Dictionary

Data Item	Data Type	Description
airQualityIndex	Integer	Contains information about the quality of air
calendar	Calendar	A view model of a composite class of calendar, which contains a list of daily weather data.
chanceOfRain	Integer	Contains a percentage showing how it is likely to rain.
humidity	Integer	Current humidity.
location	String	User-provided location or the current location of the user.
locationSearch Bar	View	A search bar view.
pressure	Float	Current pressure.
pressureUnit	String	Unit of pressure, Atmospheric Pressure(ATM) or Pascal.
realFeel	Float	Shows the temperature of how it feels like.
settings	Settings	Last saved settings configuration.
sunset	DateTime	Current day sunset time.
sunrise	DateTime	Current day sunrise time.
temperature	Integer	Current temperature.
temperatureUnit	String	Unit of temperature, Celsius or Fahrenheit.
todayBasicWeat her	List	Last shown homepage weather data.
todayMaxTemp	Integer	Contains today’s maximum temperature along with the time.

url	String	Contains the address of the weather API. The weather API is accessed from the URL.
userLocation	String	Last user location.
userProvidedLocation	String	Last user-provided location in the search bar.
uvIndex	Integer	Current UV index.
windSpeed	Float	Contains the speed of current wind flow.
weatherStatus	String	Contains weather status/signs such as rainy, haze, cloudy or sunny.
weatherData	WeatherData	Contains several lists of weather information.

6.3 Entity Relationship Diagram



7. User Requirement and Component Traceability Matrix

	System UI			Data Management	Notification	Settings
	Home	Current Weather	Weather Prediction			
Location Specific Weather Search Bar(UR1)	X	X	X	X		
Predicted Weather (UR2)			X	X		
Detailed weather information(UR3)		X		X		
Changing parameters(UR4)						X
Add to the home screen(UR5)						X
If an invalid location is given by the user(UR6)	X					
If device location is turned off(UR7)	X					
Alert the RegularUser about bad weather(UR8)					X	