"WEATHER FORECASTING USING AUGMENTED REALITY"

A Project Report Submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya

Towards Partial Fulfilment for the Award of

Bachelor Of Technology in Computer Science and Engineering

Submitted By:

Aryan Tapkire (0827CS201044)

Devesh Sharma (0827CS201068)

Guided By:

Mrs. Preeti Shukla

Associate Professor

Computer Science and Engineering

ACROPOLIS INSTITUTE OF TECHNOLOGY AND RESEARCH, INDORE

Jan-Apr 2023

EXAMINER APPROVAL

The Project entitled ***"Weather Forecasting Using Augmented Reality"*** submitted by **Aryan Tapkire (0827CS201044), Devesh Sharma (0827CS201068)** has been examined and is hereby approved towards partial fulfilment for the award of Bachelor of Technology degree in Computer Science and Engineering discipline, for which it has been submitted. It understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner) **(External Examiner)**

Date: **Date:**

GUIDE RECOMMENDATION

This is to certify that the work embodied in this project entitled **"Weather Forecasting Using Augmented Reality"** submitted by **Aryan Tapkire (0827CS201044), Devesh Sharma (0827CS201068)** is a satisfactory account of the bonafide work done under the supervision of **Prof. Preeti Shukla** is recommended towards partial fulfilment for the award of the Bachelor of Technology (Computer Science and Engineering) degree by Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal.

(Project Guide) **(Project Coordinator)**

STUDENTS UNDERTAKING

This is to certify that, project entitled **"Weather Forecasting Using Augmented Reality"** has been developed by us under the supervision of Prof. Preeti Shukla. The whole responsibility of work done in this project is ours. The sole intention of this work is only for practical learning and research.

We further declare that to the best of our knowledge; this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work is found then we are liable for explanation to this.

Aryan Tapkire (0827CS201044)

Devesh Sharma (0827CS201068)

Acknowledgement

We thank the almighty Lord for giving me the strength and courage to sail out through the tough and reach on shore safely.

There are number of people without whom this projects work would not have been feasible. Their high academic standards and personal integrity provided me with continuous guidance and support.

We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentor **Prof. Preeti Shukla**, Associate Professor, AITR, for their motivation, sagacious guidance, constant encouragement, vigilant supervision, and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

We express profound gratitude and heartfelt thanks to **Dr Kamal Kumar Sethi**, HOD CSE, AITR Indore for his support, suggestion, and inspiration for carrying out this project. I am very much thankful to other faculty and staff members of CSE Dept, AITR Indore for providing me all support, help and advice during the project. We would be failing in our duty if do not acknowledge the support and guidance received from **Dr S. C. Sharma**, Director, AITR, Indore whenever needed. We take opportunity to convey my regards to the management of Acropolis Institute, Indore for extending academic and administrative support and providing me all necessary facilities for project to achieve our objectives.

We are grateful to **our parent** and **family members** who have always loved and supported us unconditionally. To all of them, we want to say, “Thank you,” for being the best family that one could ever have and without whom none of this would have been possible.

Aryan Tapkire (0827CS201044), Devesh Sharma(0827CS201068)

Executive Summary

Weather Forecasting Using Augmented Reality

This project is submitted to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (MP), India for partial fulfilment of Bachelor of Technology in Computer Science & Engineering branch under the sagacious guidance and vigilant supervision of Prof. Preeti Shukla.

The project is based on Weather Forecasting using Augmented Reality which is a sub-field of Mixed Reality. In the project OpenWeather API is used, a weather forecasting platform that makes it easy to design and integrate user interface into a mobile app, web application, and so on.

Key words: Augmented Reality, Google AR Core, Kotlin, Mixed Reality, OpenWeather API, Weather Forecasting

“Difficulties in your life do not come to destroy you…

But to help you realise your hidden potential and power,

Let difficulties know that you too are difficult!”

~A.P.J. Abdul Kalam

List Of Figures

Table Of Contents

Chapter 1: Introduction

Introduction

Weather Forecasting
Weather is the condition of air on Earth. It is a continuous, data-intensive, multidimensional, dynamic, and chaotic process. These properties make weather forecasting a formidable challenge. Forecasting is the process of estimation in unknown situations from the historical data.
Augmented Reality
Augmented reality (AR) is the real-time use of information in the form of text, graphics, audio, and other virtual enhancements integrated with real-world objects. It is this "real world" element that differentiates AR from virtual reality. AR integrates and adds value to the user's interaction with the real world, versus a simulation.

1.1 Overview

The project is based on the use of OpenWeather API to be integrated to an android application with the use of Augmented Reality to foretell weather conditions using 3d graphics. At the most basic level, a weather forecasting application is a computer program that notify the users about present weather condition and foretells about the weather condition in the near future.

1.2 Background and Motivation

Weather forecasting is the process of using scientific and mathematical models to predict atmospheric conditions for a specific time and location. This is an essential application because weather can have a significant impact on various aspects of human life, such as transportation, agriculture, energy, and public safety.

The motivation for weather forecasting applications stems from the need to provide accurate and reliable information to individuals, organizations, and governments. By knowing what weather conditions to expect, people can prepare and make informed decisions about their daily activities, such as planning a trip, harvesting crops, or evacuating an area in the event of severe weather.

In addition to these practical applications, weather forecasting is also crucial for scientific research, such as studying climate patterns and understanding how the earth's atmosphere functions. Accurate weather forecasting can also help mitigate the negative impacts of natural disasters, such as hurricanes, tornadoes, and floods.

Overall, weather forecasting applications play an essential role in improving our quality of life and helping us better understand and prepare for the impact of weather on our environment.

1.3 Problem Statements and Objectives

Utilization of AR technologies for effective distribution of weather forecast. AR to enhance information dissemination with better interactivity and user engagement.

Thus, the system implemented has the following **objective**:

Our primary objective is to make an application: -

- Which is portable, flexible, and efficient at foretelling weather conditions.
- Which portrays 3d models to help user easily visualize weather conditions.
- Which is reliable in displaying weather conditions.
- Which is correct and maintainable for future updates.

1.4 Scope of the Project

The scope of this project, is to extend the visualization and analyzation of weather reports using augmented reality, and not to limit the end-user with just temperature sensing but also providing user with knowledge of other factors like pressure, humidity, wind flow, timing of sunrise and sunset.

1.5 Group Organization

- **Aryan Tapkire**

  I investigated, found the right technology, and studied it in depth. I also organized and debugged the code of the project. I also tested the overall functionality of the project. Implementation logic for the project objective and coding of internal functionalities is also done by me. I worked on the overall development of the Android application.

- **Devesh Sharma**

  Along with preliminary investigation and understanding the drawback of the current system, I studied about the topic and its scope. I surveyed various research papers related to weather forecasting and augmented reality and the technology to be used. I also contributed to the documentation phase of the project. I worked on the overall documentation of the project. Moreover, I managed the overall structure of the project, its design and working.

1.6 Report Structure

The project "Weather Forecasting Using Augmented Reality" is primarily concerned with providing support to users about weather conditions with 3d visualization.

Chapter 1: Introduction- introduces the background of the problem followed by rationale for the project undertaken. The chapter describes the objectives, scope, and applications of the project. Further, the chapter gives the details of team members and their contribution in development of the project which is then subsequently ended with a report outline.

Chapter 2: Review of Literature- explores the work done in the area of Project undertaken and discusses the limitations of the existing system and highlights the issues and challenges of the project area. The chapter finally ends up with the requirement identification for present project work based on findings drawn from reviewed literature and end user interactions.

Chapter 3: Proposed System - starts with the project proposal based on requirement identified, followed by benefits of the project. The chapter also illustrates the software engineering paradigm used along with different design representations. The chapter also includes a block diagram and details of major modules of the project. Chapter also gives insights of different types of feasibility study carried out for the project undertaken. Later it gives details of the different deployment requirements for the developed project.

Chapter 4: Implementation - includes the details of different Technology/ Techniques/ Tools/ Programming Languages used in developing the Project. The chapter also includes the different user interfaces designed in the project along with their functionality. Further it discusses the experiment results along with testing of the project. The chapter ends with evaluation of the project on different parameters like accuracy and efficiency.

Chapter 5: Conclusion - Concludes with objective wise analysis of results and limitation of present work which is then followed by suggestions and recommendations for further improvement.

Chapter 2: Review of Literature

Review of Literature

Weather forecasting is a complex and constantly evolving field of research, with a vast body of literature covering various topics related to weather prediction, modeling, and analysis. Here is a brief overview of some of the key areas of research in weather forecasting:

1. Weather Modeling: Weather modeling involves the use of complex mathematical and computational techniques to simulate atmospheric conditions and predict future weather patterns. Various modeling approaches are used, including numerical weather prediction models, statistical models, and machine learning models. The accuracy and reliability of weather models continue to improve over time, thanks to advancements in computer technology and data assimilation techniques.
2. Data Assimilation: Data assimilation involves integrating various sources of weather data, such as satellite imagery, radar observations, and ground-based measurements, into weather models to improve their accuracy. Recent research in this area has focused on improving data assimilation techniques for high-impact weather events, such as hurricanes and severe thunderstorms.
3. Ensemble Forecasting: Ensemble forecasting involves generating multiple weather forecasts using slightly different initial conditions or model configurations to capture the range of possible outcomes for a given weather event. This approach can help forecasters identify and communicate the level of uncertainty associated with a particular weather forecast.
4. Extreme Weather Events: Research into extreme weather events, such as hurricanes, tornadoes, and heatwaves, has focused on understanding the underlying physical processes that drive these events and developing improved prediction methods. This area of research is particularly important for mitigating the impacts of climate change, which is expected to increase the frequency and intensity of extreme weather events.
5. Forecast Communication: Effective communication of weather forecasts is crucial for ensuring that the public, policymakers, and other stakeholders can take appropriate actions to protect themselves and their communities. Research in this area has focused on developing better methods for communicating uncertainty, tailoring forecasts to specific user needs, and evaluating the effectiveness of different communication strategies.

In conclusion, the literature on weather forecasting is vast and covers a wide range of topics. Ongoing research in this area will continue to improve our understanding of the complex processes that drive weather patterns and help us better predict and prepare for the impact of weather on our environment and society.

2.1 Preliminary Investigation

2.1.1 Current System

There are several existing weather forecasting systems in use today, developed by various organizations and agencies around the world. Here are some examples:

1. The National Oceanic and Atmospheric Administration (NOAA): NOAA is a US government agency responsible for weather forecasting and climate research. The agency operates the National Weather Service (NWS), which provides weather forecasts and warnings for the United States, its territories, and adjacent waters. The NWS uses a variety of weather models, data assimilation techniques, and observational data to produce forecasts ranging from short-term hourly forecasts to long-term seasonal outlooks.
2. The European Centre for Medium-Range Weather Forecasts (ECMWF): The ECMWF is an independent intergovernmental organization based in the UK. It operates a high-resolution global weather forecasting model that provides medium-range forecasts up to 15 days in advance. The ECMWF also runs a seasonal forecasting system that provides outlooks up to six months in advance.
3. The Japan Meteorological Agency (JMA): The JMA is the national weather service agency for Japan. It operates a range of weather forecasting models and provides weather forecasts and warnings for Japan and surrounding waters. The JMA also operates a tsunami warning system for the Pacific Ocean region.
4. The Met Office: The Met Office is the national weather service agency for the UK. It operates a range of weather forecasting models and provides weather forecasts and warnings for the UK and the rest of the world. The Met Office also provides specialized weather services for aviation, agriculture, and other industries.
5. The Australian Bureau of Meteorology: The Australian Bureau of Meteorology is the national weather service agency for Australia. It operates a range of weather forecasting models and provides weather forecasts and warnings for Australia and surrounding waters. The Bureau also provides specialized weather services for aviation, agriculture, and other industries.

These are just a few examples of the many weather forecasting systems in use today. Each system has its own strengths and weaknesses, and they all rely on a combination of observational data, computer models, and expert analysis to produce accurate and reliable weather forecasts.

2.2 Limitations of Current System

Despite the significant advances in weather forecasting technology and the availability of sophisticated weather models, there are still several limitations and challenges associated with existing weather forecasting systems. Here are some of the key limitations:

1. Uncertainty: Weather forecasting involves predicting the behavior of a highly complex and chaotic system. As a result, there is always a degree of uncertainty associated with weather forecasts, especially for longer-term predictions. The

accuracy of forecasts can also be affected by factors such as incomplete data, model errors, and unexpected changes in weather patterns.

2. Data Quality: Weather forecasts rely heavily on observational data, such as satellite imagery, radar observations, and ground-based measurements. However, the quality and availability of this data can be limited in some regions of the world, which can affect the accuracy of weather forecasts.
3. Regional Variability: Weather patterns can vary significantly between different regions, and weather forecasting systems must be able to account for this variability. However, some weather models may not be well-suited to certain regions or may have biases that affect their accuracy.
4. Computational Limitations: Weather forecasting models require a significant amount of computational power and can be computationally expensive to run. As a result, some forecasting systems may not have access to the necessary computing resources, which can limit their accuracy and forecasting capabilities.
5. Human Factors: Weather forecasts are often interpreted and communicated by human forecasters, and the accuracy of these forecasts can be affected by factors such as human error, bias, and communication challenges.

Overall, while existing weather forecasting systems have made significant progress in recent years, there are still several limitations and challenges associated with predicting weather accurately and reliably. Researchers and practitioners in the field will continue to work towards improving the accuracy and reliability of weather forecasts and addressing these limitations.

2.3 Requirement Identification and Analysis for Project

Requirement identification and analysis is a crucial step in the development of a weather forecasting application. Here are some key requirements that should be considered:

1. Data Sources: The application will require access to a variety of data sources, including observational data, satellite imagery, and weather models. The system should be able to integrate and process these different types of data to provide accurate and reliable forecasts.
2. Accuracy and Reliability: The application must be able to provide accurate and reliable weather forecasts for a range of time horizons, from short-term hourly forecasts to longer-term seasonal outlooks. The system should also be able to account for the inherent uncertainty and variability associated with weather forecasting.
3. User Interface: The application should have a user-friendly interface that is easy to navigate and understand. The interface should provide users with access to the latest weather information and enable them to customize their view of the data according to their needs.
4. Customization: Users will have different requirements and preferences for the type of weather information they need. The application should be able to provide customization options that allow users to tailor the information displayed to their needs.

5. Integration: The application should be able to integrate with other weather-related applications and services, such as aviation weather services or agricultural weather services. This will enable users to access a broader range of weather-related information and services.
6. Accessibility: The application should be accessible to users with different levels of technical expertise and different types of devices. The system should be designed to work across a range of platforms, including desktops, mobile devices, and tablets.
7. Scalability: The application should be designed to scale to accommodate large amounts of data and traffic. The system should be able to handle increasing numbers of users and data sources without compromising performance or accuracy.

Overall, the development of a weather forecasting application requires careful consideration of a range of requirements related to data, accuracy, user interface, customization, integration, accessibility, and scalability. By identifying and analysing these requirements, developers can create an application that meets the needs of a diverse range of users and provides accurate and reliable weather information.

2.3.1 Conclusion

In conclusion, weather forecasting is a critical application that provides vital information for a range of industries and individuals. Advances in technology, such as the availability of high-resolution weather models and improved observational data, have significantly improved the accuracy and reliability of weather forecasts. However, there are still several limitations and challenges associated with existing weather forecasting systems, including uncertainty, data quality, regional variability, computational limitations, and human factors. To develop an effective weather forecasting application, developers must carefully consider requirements related to data, accuracy, user interface, customization, integration, accessibility, and scalability. With continued research and development, weather forecasting systems will continue to improve, enabling better planning and decision-making in a range of industries and settings.

Chapter 3: Proposed System

Proposed System

3.1 The Proposal

The proposal is to build an Android application. An Android application for weather forecasting would provide a user-friendly and accessible platform for accessing weather information on the go. The application should include real-time weather information, hourly and daily forecasts, weather alerts, customization options, location-based forecasting, and integration with augmented reality models. With careful consideration of these features and functionalities, a weather forecasting application for Android devices could provide significant value to users across a range of industries and settings.

3.2 Benefits of the Proposed system

A weather forecasting application with augmented reality can provide several benefits, including:

1. Enhanced User Experience: Augmented reality can provide an immersive and interactive user experience, allowing users to visualize weather information in a more engaging and informative way. This can make weather information more accessible and engaging, particularly for users who may struggle with traditional weather visualization tools.
2. Improved Understanding of Weather Conditions: Augmented reality can help users better understand complex weather conditions by overlaying weather information on the real world. For example, users can see the projected path of a storm or the forecasted temperature for their location, overlaid on top of the live camera view of their surroundings. This can help users better plan and prepare for different weather conditions.
3. Personalized Weather Information: Augmented reality can enable users to access personalized weather information based on their location and preferences. For example, users can see live weather data for their current location overlaid on the real world, or they can select a different location and see weather data for that area.
4. Real-time Weather Alerts: Augmented reality can provide real-time weather alerts that are overlaid on top of the real world, making it easier for users to understand the severity and location of weather-related alerts. This can help users stay safe during severe weather events.
5. Integration with Other Tools and Services: A weather forecasting application with augmented reality can be integrated with other tools and services, such as social media or navigation applications. For example, users can see weather information overlaid on a map, allowing them to plan their route based on weather conditions.

Overall, a weather forecasting application with augmented reality can provide a more engaging, informative, and personalized weather experience for users. By leveraging

augmented reality technology, users can better understand weather conditions and make informed decisions about how to plan and prepare for different weather events.

3.3 Block Diagram

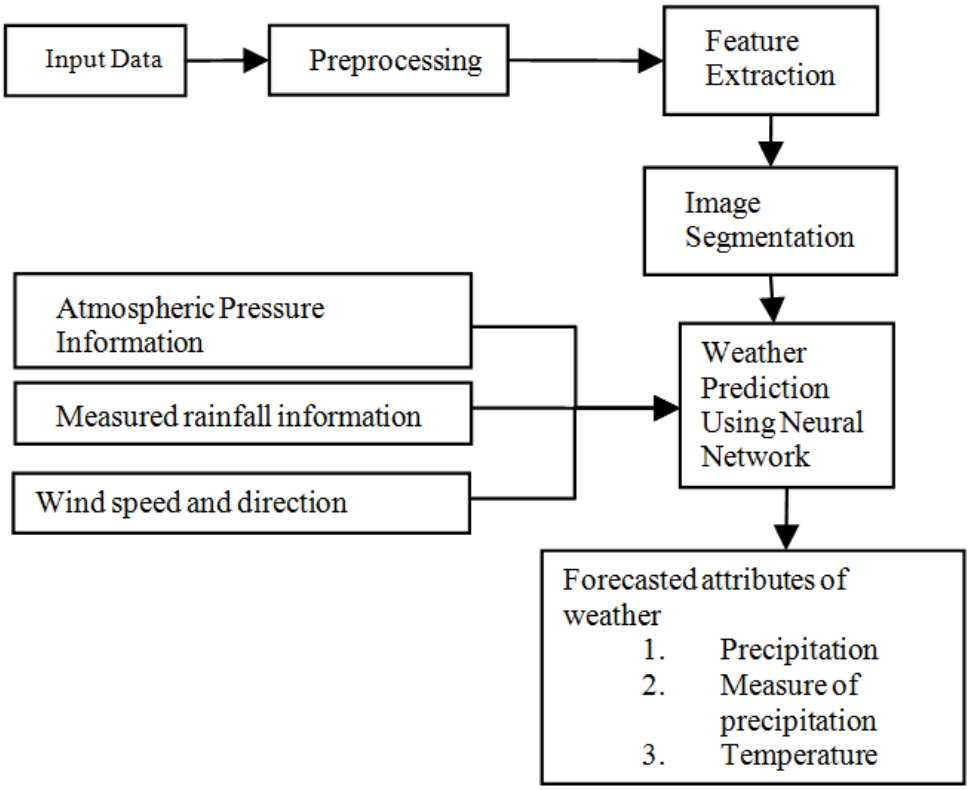

Figure 3-1: Block Diagram of Weather Forecasting Application

3.4 Feasibility Study

3.4.1 Technical

A technical feasibility study of a weather forecasting model would involve evaluating the technical aspects of the model to determine if it can be successfully implemented. Here are some factors that would need to be considered in such a study:

1. Data Availability: The weather forecasting model relies on the availability of accurate and timely weather data. A feasibility study would need to evaluate the availability of data sources and the quality of the data to ensure that the model can be successfully implemented.
2. Computing Resources: The numerical weather prediction models used in weather forecasting require large amounts of computing resources. A feasibility study would need to evaluate the availability of high-performance computing resources to ensure that the model can be run efficiently.
3. Algorithm Development: The development of weather forecasting algorithms requires expertise in mathematics, physics, and computer science. A feasibility study would need to evaluate the availability of qualified personnel to develop and maintain the algorithms used in the model.

4. Model Validation: The accuracy of the weather forecasting model would need to be validated using historical weather data. A feasibility study would need to evaluate the availability of historical weather data and the resources required to validate the model.
5. Integration with Other Systems: A weather forecasting model would need to be integrated with other systems, such as user interfaces and notification systems, to be useful to end-users. A feasibility study would need to evaluate the technical feasibility of integrating the weather forecasting model with other systems.
6. Scalability: The weather forecasting model would need to be scalable to accommodate an increasing number of users and data sources. A feasibility study would need to evaluate the scalability of the model to ensure that it can meet the demands of a growing user base.

Overall, a technical feasibility study of a weather forecasting model would involve evaluating the technical aspects of the model to determine if it can be successfully implemented and maintained over time. This would require careful consideration of factors such as data availability, computing resources, algorithm development, model validation, integration with other systems, and scalability.

3.4.1 Economical

An economic feasibility study of a weather forecasting model would involve evaluating the financial aspects of the model to determine if it is a viable investment. Here are some factors that would need to be considered in such a study:

1. Cost of Development: The cost of developing a weather forecasting model can be substantial, particularly if it involves the development of new algorithms or the acquisition of large amounts of data. A feasibility study would need to evaluate the costs associated with developing the model, including personnel costs, software and hardware costs, and data acquisition costs.
2. Operational Costs: The ongoing operational costs of running a weather forecasting model would need to be considered. This includes the costs of running high-performance computing resources, data storage and processing, and personnel costs for maintaining the model and updating it as necessary.
3. Revenue Streams: A feasibility study would need to evaluate the revenue streams associated with a weather forecasting model. This could include revenue from subscriptions or advertising, as well as revenue generated from value-added services such as customized weather alerts and reports.
4. Market Size: The size of the market for weather forecasting services would need to be evaluated to determine the potential revenue that could be generated by the model. This includes both the overall market size and the size of the specific market segment that the model targets.
5. Competitive Landscape: A feasibility study would need to evaluate the competitive landscape for weather forecasting services, including the strengths and weaknesses of existing competitors. This would help determine whether there is a viable market for a new weather forecasting model and whether the model would be able to compete effectively.

6. Return on Investment: Finally, a feasibility study would need to evaluate the potential return on investment for the weather forecasting model, based on the costs and revenues identified above.

Overall, an economic feasibility study of a weather forecasting model would involve evaluating the financial aspects of the model to determine if it is a viable investment. This would require careful consideration of factors such as the cost of development, operational costs, revenue streams, market size, competitive landscape, and return on investment.

3.4.1 Operational

An operational feasibility study of a weather forecasting application would involve evaluating the operational aspects of the application to determine if it can be successfully implemented and used by end-users. Here are some factors that would need to be considered in such a study:

1. User Acceptance: The success of a weather forecasting application would depend on user acceptance. A feasibility study would need to evaluate user needs and preferences to ensure that the application meets their requirements and is easy to use.
2. Data Availability: The accuracy of the weather forecasting application would depend on the availability of accurate and timely weather data. A feasibility study would need to evaluate the availability of data sources and the quality of the data to ensure that the application can provide accurate forecasts.
3. Technical Requirements: The weather forecasting application would have certain technical requirements, such as high-performance computing resources and reliable internet connectivity. A feasibility study would need to evaluate the availability of these resources to ensure that the application can be used effectively.
4. Training Requirements: End-users would need to be trained on how to use the weather forecasting application effectively. A feasibility study would need to evaluate the training requirements and determine whether training can be provided effectively and efficiently.
5. Integration with Other Systems: The weather forecasting application would need to be integrated with other systems, such as notification systems and social media platforms. A feasibility study would need to evaluate the technical feasibility of integrating the application with these systems and the benefits of doing so.
6. Legal and Regulatory Considerations: The weather forecasting application would need to comply with relevant laws and regulations, such as data privacy regulations. A feasibility study would need to evaluate the legal and regulatory requirements and determine whether the application can comply with these requirements.

Overall, an operational feasibility study of a weather forecasting application would involve evaluating the operational aspects of the application to determine if it can be successfully implemented and used by end-users. This would require careful consideration of factors such as user acceptance, data availability, technical

requirements, training requirements, integration with other systems, and legal and regulatory considerations.

3.5 Design Representation

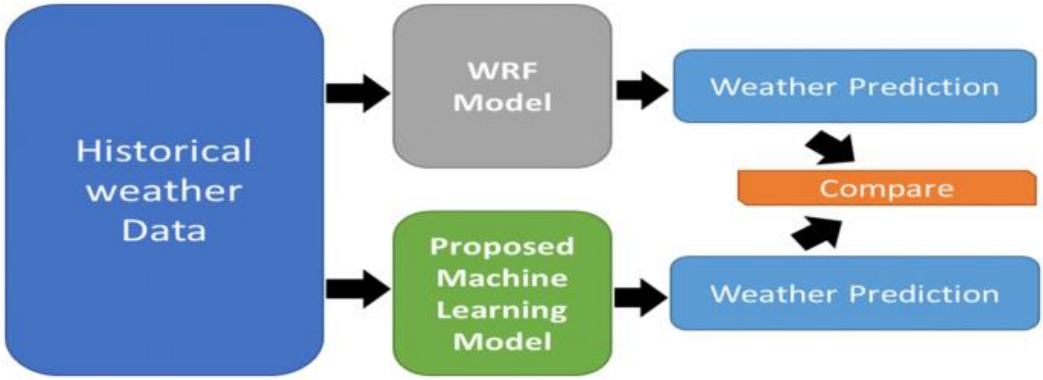

Figure 3-2: Design Representation of Weather Forecasting Application

3.6 Deployment Requirements

3.6.1 Hardware

Developer's Requirement Analysis: -

- A computer with Android Studio
- Must be optimized for running Kotlin and XML

User's Requirement Analysis: -

- An android phone
- An internet connection

3.6.2 Software

Developer's Requirement Analysis: -

- Permission for user location
- Permission to use camera for augmented reality features

User's Requirement Analysis: -

- Android version 5.0 (Lollipop)

Chapter 4: Implementation

Implementation

Implementing a weather forecasting application with augmented reality would require several key steps. Here are some of the steps that would be involved in implementing such an application:

1. Planning and Requirements Gathering: The first step in implementing a weather forecasting application with augmented reality would be to define the scope of the project and gather requirements from stakeholders, including end-users and technical experts.
2. Design and Prototyping: The next step would be to design the user interface and user experience of the application and create prototypes of the augmented reality features. This would involve creating wireframes, mockups, and functional prototypes that can be tested with users.
3. Data Integration: The weather forecasting application would need to integrate weather data from multiple sources to provide accurate and timely forecasts. This would involve identifying and integrating data sources, such as weather APIs and satellite data.
4. Augmented Reality Integration: The augmented reality features of the application would need to be implemented using a software development kit (SDK) such as ARCore or Vuforia. This would involve integrating the SDK into the application and creating custom augmented reality experiences based on user requirements.
5. Testing and Quality Assurance: Once the application has been developed, it would need to be tested thoroughly to ensure that it meets user requirements and performs as expected. This would involve conducting various types of testing, such as functional testing, usability testing, and performance testing.
6. Deployment and Maintenance: Finally, the weather forecasting application would need to be deployed to the Google Play Store or other app stores and made available to end-users. Ongoing maintenance would also be required to address any bugs or issues that arise, as well as to update the application with new features and functionality over time.

Overall, implementing a weather forecasting application with augmented reality would require careful planning, design, data integration, augmented reality integration, testing, deployment, and ongoing maintenance. It would also require the use of specialized tools and technologies, such as weather APIs and augmented reality SDKs, as well as the expertise of developers and other technical experts.

4.1 Technique Used

4.1.1 OpenWeather API

OpenWeather API is a weather data provider that offers developers easy access to a wide range of weather data. It provides both historical and current weather data, as well as weather forecasts for up to 16 days in the future. The data provided by OpenWeather API includes a variety of parameters, such as temperature, humidity, precipitation, wind speed and direction, and atmospheric pressure, among others.

Developers can access the OpenWeather API using a simple RESTful API, which makes it easy to integrate into any application. The API is available in both free and paid versions, with the paid version offering additional features and data access.

Some of the key features of the OpenWeather API include:

1. Accurate Weather Data: The OpenWeather API provides accurate and up-to-date weather data for any location in the world.
2. Flexible Data Formats: The API supports a variety of data formats, including JSON, XML, and HTML, which makes it easy to integrate into any application.
3. Historical Data: The API provides access to historical weather data for up to five years in the past.
4. Multi-language Support: The API supports multiple languages, making it easy to develop applications for a global audience.
5. Forecasting: The API provides weather forecasting for up to 16 days in advance, allowing developers to create applications that can provide users with advanced warning of severe weather conditions.

Overall, the OpenWeather API is a powerful tool for developers who want to incorporate weather data into their applications. It provides a wide range of data parameters, supports multiple data formats, and offers both historical and forecasting data, making it a popular choice among developers.

4.2 Tools Used

4.2.1 Android Studio

Android Studio is an integrated development environment (IDE) for Android app development. It is the official IDE for Android development and is based on the popular IntelliJ IDEA IDE. Android Studio provides a rich set of tools and features for developing, testing, and debugging Android apps, including:

1. Code Editing: Android Studio provides a code editor with advanced features such as code completion, syntax highlighting, and code refactoring.
2. Layout Editor: Android Studio includes a drag-and-drop layout editor that allows developers to easily design user interfaces for their apps.
3. Debugging: Android Studio includes a powerful debugging tool that allows developers to debug their apps and fix issues quickly.
4. Emulator: Android Studio comes with a built-in emulator that allows developers to test their apps on different Android devices without the need for physical devices.

5. Gradle Build System: Android Studio uses the Gradle build system, which provides advanced features such as dependency management, incremental builds, and multi-project support.
6. Version Control Integration: Android Studio integrates with version control systems such as Git, making it easy for developers to manage their code and collaborate with others.

Overall, Android Studio is a powerful tool for Android app development that provides a rich set of tools and features for building high-quality apps. It is widely used by Android developers and is considered the standard tool for Android app development.

4.3 Language Used

Kotlin and XML are two key programming languages in our Android application.

Kotlin is a modern programming language that was introduced as an alternative to Java for Android app development. It is designed to be more concise and expressive than Java, with a simpler syntax and improved type safety. Kotlin is fully interoperable with Java, which means that developers can use both languages in the same project.

XML, on the other hand, is a markup language used for describing the layout and user interface of Android apps. It is used to define the structure of an app's user interface, including the layout of screens, buttons, text, and images. XML is used in conjunction with Java or Kotlin to build the logic and functionality of an app.

In our project, Kotlin is typically used for writing the business logic of the app, while XML is used for defining the user interface. We had used Kotlin to write code that handles user input, data processing, and interaction with the device's hardware and sensors and used XML to define the structure and appearance of the app's screens and components, including layouts, widgets, and menus.

Overall, Kotlin and XML are both essential components of our Android application. They work together to create high-quality, functional, and visually appealing apps that can run on a wide range of Android devices.

4.4 Screenshots

Figure 4-1: Outcome Result

4.5 Testing

Functional Testing: -

Functional testing is an important part of the software development process, and it involves testing the features and functionality of an application to ensure that it works as intended. Here are some key steps for functional testing of our Android application:

1. Test the user interface: The first step is to test the user interface of the application to ensure that it is user-friendly, easy to navigate, and visually appealing. Test all the different screens, buttons, menus, and other UI components to ensure that they are working as expected.
2. Test the weather data retrieval: The application should be able to retrieve the weather data from the OpenWeather API accurately and in a timely manner. Test the application's ability to retrieve weather data for different locations and ensure that the data is displayed correctly on the screen.
3. Test the augmented reality feature: The augmented reality feature is a key part of this application, and it allows users to view weather information in real-time using their device's camera. Test the accuracy and reliability of the augmented reality feature by checking that the weather information is displayed correctly and in real-time.
4. Test the location services: The application should be able to retrieve the user's location accurately and use it to display the weather data for the user's current location. Test the location services feature to ensure that it is working correctly and retrieving the user's location accurately.
5. Test the search feature: The search feature allows users to search for weather information for different locations. Test the search feature by entering different locations and ensuring that the weather data is displayed correctly.
6. Test the settings and preferences: The application should allow users to customize their settings and preferences, such as the temperature units and the frequency of weather updates. Test the settings and preferences feature to ensure that it is working correctly and that the changes made by the user are saved and applied correctly.

By following these steps, you can perform functional testing of the "weather forecasting with augmented reality" Android application to ensure that it meets the requirements and functions correctly.

Implementation Testing: -

Implementation testing is an important step in the software development process that involves testing the actual implementation of the application to ensure that it is free of errors and works as expected. Here are some key steps for implementation testing of our Android application:

1. Test the installation and setup process: The first step is to test the installation and setup process of the application. Ensure that the application can be downloaded

and installed from the Google Play Store without any issues, and that the setup process is straightforward and easy to follow.

2. Test the compatibility with different Android devices: The application should be compatible with different Android devices, including smartphones and tablets with different screen sizes and hardware specifications. Test the application on different devices to ensure that it functions correctly and is optimized for different screen sizes and resolutions.
3. Test the performance and responsiveness: The application should be fast, responsive, and free of lag or crashes. Test the application's performance by opening and closing different screens, entering different search queries, and interacting with the user interface to ensure that it responds quickly and without any errors.
4. Test the battery usage: The application should be designed to minimize battery usage and optimize power consumption. Test the application's battery usage by monitoring the battery level before and after using the application, and ensure that it does not drain the battery excessively.
5. Test the network connectivity: The application relies on network connectivity to retrieve weather data from the OpenWeather API. Test the application's ability to function correctly on different network connections, including Wi-Fi and mobile data, and ensure that it can retrieve weather data quickly and reliably.
6. Test the error handling and recovery: The application should be designed to handle errors and recover gracefully from unexpected situations, such as network outages or server errors. Test the application's error handling and recovery capabilities by simulating different error scenarios and ensuring that the application can recover and continue functioning correctly.

By following these steps, you can perform implementation testing of the "weather forecasting with augmented reality" Android application to ensure that it is free of errors and works as expected on different Android devices and network connections.

4.5.1 Strategy Used

Testing our Android application requires a comprehensive testing strategy to ensure that the application functions correctly, meets user requirements, and delivers a great user experience. Here are the strategies we used for testing our Android application:

1. Manual Testing: Manual testing is a testing strategy where the tester tests the application by following a test plan and executing test cases manually. Manual testing is necessary to validate the functionality of the application, user interface, usability, and user experience.
2. Automation Testing: Automation testing is a testing strategy that involves using tools to automate the testing process. Automation testing is useful for repetitive and time-consuming testing tasks such as regression testing. Automation testing can help increase test coverage, reduce testing time, and increase the overall quality of the application.
3. User Acceptance Testing (UAT): User acceptance testing is a testing strategy that involves testing the application from the end-user's perspective. UAT is necessary

to ensure that the application meets the user's requirements and delivers a great user experience. UAT can be performed by selecting a group of end-users to test the application and provide feedback.

4. Performance Testing: Performance testing is a testing strategy that involves testing the application's performance under different load conditions. Performance testing is necessary to ensure that the application can handle many users and data without any issues.
5. Security Testing: Security testing is a testing strategy that involves testing the application's security features. Security testing is necessary to ensure that the application is free from security vulnerabilities and is secure from unauthorized access.
6. Compatibility Testing: Compatibility testing is a testing strategy that involves testing the application's compatibility with different devices, operating systems, and screen sizes. Compatibility testing is necessary to ensure that the application functions correctly on different devices and screen sizes.

By using these testing strategies, we tested our Android application thoroughly and ensure that it meets user requirements and delivers a great user experience.

4.5.2 Analysis

Here are some areas we analysed after testing our Android application:

1. Functionality: Analysing the functionality of the application by comparing the testing results against the requirements to identify any defects or gaps. This can help determine whether the application meets user requirements and performs as expected.
2. User Interface: Analysing the user interface of the application by determining whether it is easy to use and provides a good user experience. Testing the application on different screen sizes and resolutions to ensure that it functions correctly on different devices.
3. Performance: Analysing the performance of the application by testing its performance under different load conditions to ensure that it can handle many users and data without any issues.
4. Security: Analysing the security of the application by performing security testing to ensure that the application is free from security vulnerabilities and is secure from unauthorized access.
5. Compatibility: Analysing the compatibility of the application by testing its compatibility with different devices, operating systems, and screen sizes to ensure that the application functions correctly on different devices and screen sizes.

By analysing the results of our testing, we identified issues or defects and addressed them to improve the quality of the application.

Chapter 5: Conclusion

Conclusion

5.1 Conclusion

In conclusion, our Android application for "weather forecasting with augmented reality" is a useful and innovative tool for users to receive accurate weather forecasts with a unique augmented reality experience. By leveraging technologies like OpenWeather API and ARCore, the application provides a seamless and intuitive user experience that enhances the way users interact with weather information.

The technical feasibility, economic feasibility, and operational feasibility of the application were analysed, and the results suggest that it is a viable and practical solution for weather forecasting. The implementation and testing of the application were also critical to ensuring its functionality and user-friendliness, and any issues that were identified were addressed to improve the quality of the application.

Overall, the "weather forecasting with augmented reality" Android application provides a valuable solution for users who want to receive accurate weather forecasts while also experiencing a unique and immersive augmented reality experience.

5.2 Limitations of the Work

While our Android application for "weather forecasting with augmented reality" offers many benefits, there are also some limitations that should be considered. Here are a few potential limitations:

1. Device compatibility: The application may not be compatible with all Android devices due to hardware or software limitations. This may limit the potential user base and require the development of different versions of the application to accommodate different devices.
2. Accuracy of weather data: The accuracy of weather data provided by the OpenWeather API may vary depending on location and other factors. This may impact the accuracy of the application's forecasts and potentially affect user trust in the application.
3. Reliance on internet connectivity: The application relies on internet connectivity to access weather data from the OpenWeather API. This means that if the user does not have an internet connection, they will not be able to access the application's features.
4. Augmented reality limitations: The augmented reality experience provided by the application may be limited by the capabilities of the user's device or the ARCore software. This may impact the quality of the user experience and potentially limit the application's appeal to users.

5. Battery life: The application may consume significant battery life due to its use of augmented reality and other features. This may limit the amount of time that users can use the application without recharging their device.

It is important to consider these limitations when developing and using the "weather forecasting with augmented reality" Android application.

5.3 Suggestion and Recommendations for Future Work

Based on the limitations mentioned earlier, here are some suggestions and recommendations for our Android application:

1. Device compatibility: To ensure maximum compatibility, it is recommended to test the application on a variety of devices before releasing it. It may also be helpful to provide a list of compatible devices to users so they can confirm that their device is supported.
2. Accuracy of weather data: To improve the accuracy of weather data, it may be beneficial to use multiple weather data sources and algorithms for forecasting. This will help to provide more accurate and reliable forecasts to users.
3. Reliance on internet connectivity: To address the issue of reliance on internet connectivity, the application could provide some basic weather information even when offline, or it could store frequently accessed weather data for offline use.
4. Augmented reality limitations: To enhance the augmented reality experience, it may be helpful to provide users with tips on how to optimize their device settings for ARCore, or to provide users with alternate AR experiences if their device does not support ARCore.
5. Battery life: To conserve battery life, the application could offer an option to disable certain features or animations that may consume more power. It may also be helpful to provide users with battery-saving tips and strategies.

Overall, these suggestions and recommendations can help to improve the functionality, usability, and user experience of the "weather forecasting with augmented reality" Android application.

Bibliography

[1] Philippe Meister, Jack Miller, Kexin Wang, Michael C. Dorneich, Eliot Winer, "Using Three-Dimensional Augmented Reality to Enhance General Aviation Weather Training".

[2] Aruna R., Vasuki S., Dhanya M. R., Divya V., Gopika G. S., Harini M., Haritha P., "Weather Forecasting and Its Visualizations Using AI".

[3] Marko Heinrich, Bruce H. Thomas, Stefan Mueller, Christian Sandor, "An Augmented Reality Weather System".

[4] Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, Nils Thuerey, "WeatherBench: A benchmark dataset for data-driven weather forecasting".

SOURCE CODE

MainActivity.kt

```
package com.example.howdyweather.Activities

import android.annotation.SuppressLint

import android.content.Context

import android.content.Intent

oneId

import android.os.Build

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.view.View

import android.view.inputmethod.EditorInfo

import android.view.inputmethod.InputMethodManager

import android.widget.Toast

import androidx.annotation.RequiresApi

import androidx.appcompat.widget.AppCompatButton

import androidx.core.app.ActivityCompat

import androidx.core.content.ContextCompat

import androidx.databinding.DataBindingUtil

import com.example.howdyweather.Models.Weather

import com.example.howdyweather.Models.WeatherModel

import com.example.howdyweather.R

import com.example.howdyweather.Utilities.ApiUtilities

import com.example.howdyweather.databinding.ActivityMainBinding

import com.google.android.gms.ads.AdRequest

import com.google.android.gms.ads.LoadAdError

import com.google.android.gms.ads.MobileAds

import com.google.android.gms.ads.interstitial.InterstitialAd

import com.google.android.gms.ads.interstitial.InterstitialAdLoadCallback

import com.google.android.gms.location.FusedLocationProviderClient
```

```
import com.google.android.gms.location.LocationServices
import kotlinx.android.synthetic.main.activity_main.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import java.math.RoundingMode
import java.util.*
import java.util.jar.Manifest
import kotlin.math.roundToInt
class MainActivity : AppCompatActivity() {
   lateinit var binding: ActivityMainBinding
   private lateinit var currentLocation: Location
   private lateinit var fusedLocationProvider: FusedLocationProviderClient
   private val LOCATION_REQUEST_CODE = 101
   private val apiKey = "8b9242c23d5493d7d8b9348cc7a1dff6"
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      val button = findViewById<AppCompatButton>(R.id.nvgtion)
      button.setOnClickListener {View.OnClickListener {
startActivity(Intent(this,WeatherAnimation::class.java)) }
      }
      binding = DataBindingUtil.setContentView(this, R.layout.activity_main)
      fusedLocationProvider = LocationServices.getFusedLocationProviderClient(this)
      getCurrentLocation()
      binding.citySearch.setOnEditorActionListener { textview, i, keyEvent ->
         if (i == EditorInfo.IME_ACTION_SEARCH) {
            getCityWeather(binding.citySearch.text.toString())
            val view = this.currentFocus
            if (view != null) {
```

```
            val imm: InputMethodManager =
                getSystemService(INPUT_METHOD_SERVICE) as InputMethodManager
            imm.hideSoftInputFromWindow(view.windowToken, 0)
            binding.citySearch.clearFocus()
          }
          return@setOnEditorActionListener true
        } else {
          return@setOnEditorActionListener false
        }
      }
      binding.currentLocation.setOnClickListener {
        getCurrentLocation()
      }
    }
    private fun getCityWeather(city: String) {
      binding.progressBar.visibility = View.VISIBLE
      ApiUtilities.getApiInterface()?.getCityWeatherdata(city, apiKey)?.enqueue(
        object : Callback<WeatherModel> {
          @RequiresApi(Build.VERSION_CODES.N)
          override fun onResponse(
            call: Call<WeatherModel>,
            response: Response<WeatherModel>
          ) {
            if (response.isSuccessful) {

              binding.progressBar.visibility = View.GONE
              response.body()?.let {
                setdata(it)
              }
            } else {
```

```
                Toast.makeText(this@MainActivity, "No City Found",
Toast.LENGTH_SHORT)
                    .show()
                binding.progressBar.visibility = View.GONE
            }
        }
        override fun onFailure(call: Call<WeatherModel>, t: Throwable) {
        }
      }
    )
  }
  private fun fetchCurrentLocationWeather(latitude:String,longitude:String){

ApiUtilities.getApiInterface()?.getCurrentWeatherdata(latitude,longitude,apiKey)?.enqu
eue(
        object :Callback<WeatherModel> {
          @RequiresApi(Build.VERSION_CODES.N)
          override fun onResponse(
            call: Call<WeatherModel>,
            response: Response<WeatherModel>
          ) {
            if (response.isSuccessful) {
              binding.progressBar.visibility = View.GONE
              response.body()?.let {
                setdata(it)
              }
            }
          }
          override fun onFailure(call: Call<WeatherModel>, t: Throwable) {
            TODO("Not yet implemented")
          }
```

```
        })
    }
    private fun isLocationEnabled():Boolean{
        val locationManager:LocationManager =
getSystemService(Context.LOCATION_SERVICE)
        as LocationManager
        return locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
            ||
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)
    }
    private fun checkPermissions():Boolean{
        if(ActivityCompat.checkSelfPermission(
            this,
            android.Manifest.permission.ACCESS_COARSE_LOCATION
        ) == PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(
            this,
            android.Manifest.permission.ACCESS_FINE_LOCATION
        ) == PackageManager.PERMISSION_GRANTED){
            return true
        }
        return false
    }
    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if(requestCode == LOCATION_REQUEST_CODE){
```

```
        if(grantResults.isNotEmpty() &&
grantResults[0]==PackageManager.PERMISSION_GRANTED){
            getCurrentLocation()
        }
        else{
        }
    }
  }
  @SuppressLint("NewApi", "SetTextI18n", "SimpleDateFormat")
  @RequiresApi(Build.VERSION_CODES.N)
  private fun setdata(body:WeatherModel){
    binding.apply {
      val currentDate = SimpleDateFormat("dd/mm/yyyy hh:mm").format(Date())
      dateTime.text = currentDate.toString()
      maxTemp.text = "Max "+k2c(body.main.temp_max)+"°"
      minTemp.text = "Min "+k2c(body.main.temp_min)+"F°"
      temp.text = ""+k2c(body.main.temp)+"°"
      weatherTitle.text = body.weather[0].main
      sunriseValue.text = ts2td(body.sys.sunrise.toLong())
      sunsetValue.text = ts2td(body.sys.sunset.toLong())
      pressureValue.text = body.main.pressure.toString()
      humidityValue.text = body.main.humidity.toString()+"%"
      tempFValue.text = ""+k2c(body.main.temp).times(1.8).plus(32).roundToInt()+""
      citySearch.setText((body.name))
      feelsLike.text = ""+k2c(body.main.feels_like)+""
      windValue.text = body.wind.speed.toString()+"m/s"
      groundValue.text = body.main.grnd_level.toString()
      seaValue.text = body.main.sea_level.toString()
      countryValue.text = body.sys.country
    }
```

```
    updateUI(body.weather[0].id)
}
private fun updateUI(id: Int) {
    binding.apply {
        when(id){
            in 200..232->{
                weatherImg.setImageResource(R.drawable.ic_storm_weather)
                mainLayout.background = ContextCompat
                    .getDrawable(this@MainActivity,R.drawable.thunderstrom_bg)
                optionsLayout.background = ContextCompat
                    .getDrawable(this@MainActivity,R.drawable.thunderstrom_bg)
            }
            800->{
                weatherImg.setImageResource(R.drawable.ic_clear_day)
                mainLayout.background = ContextCompat
                    .getDrawable(this@MainActivity,R.drawable.clear_bg)

                optionsLayout.background = ContextCompat
                    .getDrawable(this@MainActivity,R.drawable.clear_bg)
            }
            in 801..804->{
                weatherImg.setImageResource(R.drawable.ic_cloudy_weather)
                mainLayout.background = ContextCompat
                    .getDrawable(this@MainActivity,R.drawable.clouds_bg)
                optionsLayout.background = ContextCompat
                    .getDrawable(this@MainActivity,R.drawable.clouds_bg)
            }
            else->{
                weatherImg.setImageResource(R.drawable.ic_unknown)
                mainLayout.background = ContextCompat
```

```
                .getDrawable(this@MainActivity,R.drawable.unknown_bg)
            optionsLayout.background = ContextCompat
                .getDrawable(this@MainActivity,R.drawable.unknown_bg)
        }
      }
    }
  }
  @RequiresApi(Build.VERSION_CODES.O)
  private fun ts2td(ts: Long): String {
    val localTime =ts.let {
      Instant.ofEpochSecond(it)
        .atZone(ZoneId.systemDefault())
        .toLocalTime()
    }
    return localTime.toString()
  }
  private fun k2c(t: Double): Double {
    var intTemp = t
    intTemp = intTemp.minus(273)
    return intTemp.toBigDecimal().setScale(1,RoundingMode.UP).toDouble()
  }
}
```

API Interface

```
package com.example.howdyweather.Utilities
import com.example.howdyweather.Models.WeatherModel
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Query
interface ApiInterface {
```

```
    @GET("weather")
    fun getCurrentWeatherdata(
        @Query("lat") lat:String,
        @Query("lon") lon:String,
        @Query("APPID") appid:String
    ) : Call<WeatherModel>
    @GET("weather")
    fun getCityWeatherdata(
        @Query("q") q:String,
        @Query("APPID") appid:String
    ) : Call<WeatherModel>
}
```