SkyCast: Weather App

A PROJECT REPORT

Submitted By

Devyansh Nigam(23BCS13022)

Adarsh Prajapati(23BCS14079)

In partial fulfilment for the award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

November. 2025

BONAFIDE CERTIFICATE

Certified that this project report "**SkyCast: Weather App**" is the bonafide work of **Devyansh Nigam** & **Adarsh Prajapati** who carried out the project work under my/our supervision.

SIGNATURE **SIGNATURE**

BATCH HEAD **SUPERVISOR**

Pf. Sandeep Singh Kang **Dr. Sanjeev Kumar**

TABLE OF CONTENTS

ABSTRACT

SkyCast is a real-time weather forecasting mobile application developed for Android devices using Java and Android Studio. Designed with user convenience and accuracy in mind, the app provides instant weather updates, including temperature, humidity, wind speed, and general conditions, through a clean and intuitive user interface. It fetches meteorological data via the **OpenWeatherMap API**, ensuring the information delivered is both real-time and reliable.

The development of SkyCast serves as a practical implementation of several fundamental and advanced Java programming concepts.

These include:

- **Object-Oriented Programming (OOP):** Classes and objects are used to encapsulate functionality and maintain modular, reusable code, which enhances maintainability and readability.
- **Exception Handling:** Robust try-catch mechanisms are implemented to gracefully manage network failures, API issues, and unexpected user inputs, thereby improving the app's stability.
- **Multithreading:** Background threads (via ExecutorService) are used to handle network operations without blocking the main UI thread, maintaining responsiveness and improving user experience.
- **API Integration and JSON Parsing:** The app efficiently processes and converts JSON data from the API into meaningful information for users through Java's org.json library.

Apart from the technological accomplishments, the project enhanced our knowledge of UI design, event-driven programming, and collaborative software development. SkyCast is an example of how theoretical programming knowledge can be put into practice to create a working, real-world Android application, demonstrating the capabilities of Java in mobile app development.

INRODUCTION

With the ever-growing importance of real-time data, weather applications have become essential tools in our daily lives. Whether planning a trip, dressing appropriately for the day, or preparing for severe weather conditions, accurate and timely forecasts are necessary.

SkyCast is an Android app created as a part of our Java programming coursework. The goal of the project is to provide a connection between theoretical Java ideas and practical use by developing a complete weather forecasting app. The app is fully built using Java, and it makes effective use of the OpenWeatherMap API to retrieve up-to-date weather information based on user input.

This project provided an excellent opportunity to implement and strengthen our understanding of key Java concepts, including **Object-Oriented Programming (OOP)**, exception handling, networking through **HTTP** requests, **JSON** parsing, and multi-threading.

By handling real-time data and ensuring smooth UI interactions, the app serves as a proof of how robust and flexible Java is when used in combination with Android Studio for modern application development. The selection of tools and technologies used is focused on maximizing the learning and application of Java programming in a real-world mobile development environment.

In addition to Java's role in building the core functionality, we also explored important concepts such as data parsing, thread management, and proper user input validation. As a group of four members, the project was not only a technical journey but also a collaborative one, enhancing our teamwork and problem-solving abilities.

BACKGROUND STUDY

1. Existing Solutions

There are many weather apps on the market, including AccuWeather, The Weather Channel, and Yahoo Weather. These apps provide precise forecasts, weather warnings, and live information. But too many of them are either full of features that bloat users or restricted in customization and code transparency, and therefore hard for new developers to follow their internals.

There are some open-source alternatives available but based on old technologies or not fully documented. These applications are good enough as they stand, but are usually closed systems, and therefore do not leave much scope for learning or extension by students or hobbyists.

SkyCast intends to fill the gap with a clean, responsive, and educational weather application that illustrates real-world application of Java and Android concepts in an accessible, modular format.

2. Problem Definition

Most weather apps aren't beginner-friendly — they hide backend logic, have complex UIs, and don't explain core programming concepts.

SkyCast solves this by offering real-time weather updates while also serving as a learning tool for Java, Android development, and API integration.

3. Project Objectives

- Implement a weather app using Java and Android Studio.
- Use object-oriented principles to structure the code.
- Integrate OpenWeatherMap API for real-time weather data.
- Parse and display JSON responses.

- Implement exception handling for safe API interactions.
- Demonstrate Java-based UI control and background threading.

SOFTWARE & TOOLS USED

• Programming Language: Java

Java is the core language used for developing the entire application logic. Its object-oriented features help in structuring the code efficiently using classes and methods. Java also supports networking, multithreading, and exception handling which are essential for this app.

• IDE: Android Studio

Android Studio provides a full-fledged development environment with builtin tools for designing UI, debugging Java code, and testing apps on emulators. It supports XML-based layouts and Java integration for backend logic, making it an ideal choice for Android development.

• Library: OkHttp

OkHttp is one of the widely used Java libraries to make network HTTP calls. It is extremely efficient, offers connection pooling, transparent GZIP decompression, and response caching. In this project, it has been used to retrieve weather information from the API asynchronously.

• API: OpenWeatherMap

OpenWeatherMap provides easy-to-use RESTful APIs that return real-time weather data in JSON format. It is reliable and free for basic use, making it perfect for learning API integration in a Java-based project.

• JSON (JavaScript Object Notation)

JSON is the OpenWeatherMap API data format. Java offers libraries like org.json, which enable straightforward parsing of such structured data. Working with JSON in the project shows how to retrieve and output dynamic values from API responses.

SYSTEM ARCHITECTURE

The architecture of the SkyCast application follows a structured sequence of interactions between user input, network request, data parsing, and UI updates.

1. User Input:

The user inputs the city name into the provided text field. This input is validated to ensure it is not empty before proceeding.

2. Button Trigger:

When the "Change City" button is clicked, the app builds a request URL using the provided city name and API key.

3. API Request (OkHttp):

An asynchronous HTTP GET request is made to the OpenWeatherMap server using the OkHttp library. This avoids blocking the main UI thread while making the network call.

4. Server Response:

The OpenWeatherMap API returns a JSON object containing weather details such as temperature, humidity, wind speed, and a weather condition description.

5. JSON Parsing (Java):

Java's org.json library is used to parse the JSON response. The required data is extracted using keys like "temp", "humidity", "speed", and "description".

6. UI Update (Java Threads):

The extracted data is posted back to the main thread using runOnUiThread() to update TextViews and ImageView on the UI.

7. Error Handling:

If an invalid city is entered or the network fails, Java's try-catch blocks ensure that the app doesn't crash. Instead, meaningful error messages are displayed to the user.

This structured flow ensures responsiveness, modularity, and maintainability, all while emphasizing Java's strengths in network handling, JSON parsing, UI threading, and error control.

DESIGN FLOW

• **Requirement Analysis:** Identify what the app should do – show current weather based on user input.

• **UI Planning:** Design screens in XML – input field, button, and weather info display areas.

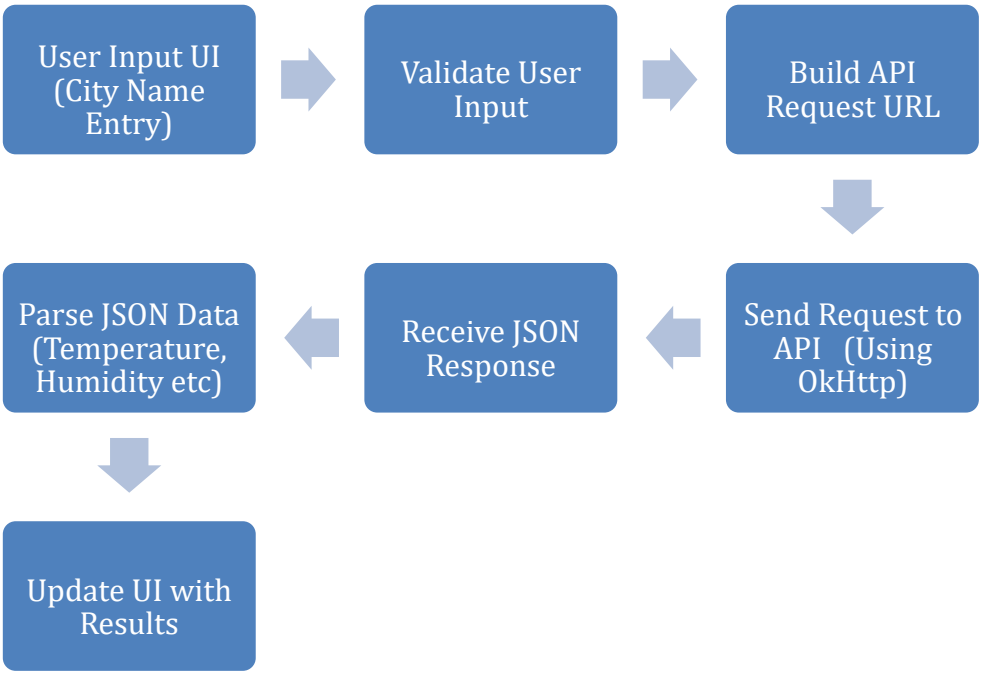
• **API Setup:** Register and use OpenWeatherMap API to fetch weather data.

• **Code Implementation:** Use Java for handling logic, including multithreading and JSON parsing.

• **Testing & Debugging:** Check for bugs, errors, and crashes during usage.

• **Polishing UI:** Make the interface responsive and clear.

• **Final Integration:** Combine all components into a smooth, working app.


User Input UI (City Name Entry)
Validate User Input
Build API Request URL
Send Request to API (Using OkHttp)
Receive JSON Response
Parse JSON Data (Temperature, Humidity etc)
Update UI with Results

OUTPUT


10:37
Chandigarh
28°
13%
Humidity
3 km/h
Wind
overcast clouds
chandigarh
Change City

FEATURES IMPLEMENTED

SkyCast offers a comprehensive suite of features designed to enhance user experience, providing a reliable and engaging weather forecasting tool. The following outlines the key features implemented in the application:

City-Based Weather Search

• **User Input:** Users can easily search for weather conditions by entering the name of any city.

• **Dynamic API Calls:** As users input their city of choice, the application dynamically communicates with the OpenWeatherMap API to fetch relevant data.

Real-Time Updates

• Live weather data is fetched using the OpenWeatherMap API with up-todate temperature, humidity, wind speed, and condition descriptions.

Interactive User Interface

• **Responsive Design:** The app features an intuitive and interactive UI, accommodating various screen sizes and orientations for optimal navigation.

• **Interactive Elements:** Buttons, text fields, and data displays allow users to interact seamlessly with the application, enhancing engagement.

Input Validation

• **Error Checking:** The app validates user inputs to prevent errors; for example, it checks if the city name is properly formatted, ensuring that only valid requests are sent.

• **User Feedback:** If an invalid input is detected, the application provides immediate feedback, guiding users to correct their entries.

Error Handling

• Implemented try-catch blocks prevent the app from crashing due to network issues or invalid API responses.

Thread Management

• Network operations are executed on a background thread using Java's ExecutorService, ensuring the main UI thread remains responsive.

Modular Coding

• Functions like FetchWeatherData() and updateUI() are separated for clarity and reusability, making the code easier to maintain.

JAVA CONCEPTS APPLIED

Object-Oriented Programming (OOP):

The project uses classes and methods to create modular and reusable code. UI components, networking logic, and JSON parsing are separated cleanly, demonstrating encapsulation and separation of concerns.

Exception Handling:

The app uses try-catch blocks to handle errors such as network failures, malformed responses, and invalid user inputs, ensuring smooth execution without crashes.

Multithreading:

The API call is performed on a background thread using ExecutorService to avoid freezing the UI, and results are updated on the main thread using runOnUiThread().

API Integration:

Using OkHttp to get data from OpenWeatherMap illustrates how to issue HTTP requests and work with asynchronous data through callbacks.

JSON Parsing:

Java's org.json.JSONObject and JSONArray classes are used to navigate through and extract data from JSON-formatted API responses.

UI and Event Handling:

The app makes use of event-driven programming in Java where button clicks trigger listeners that in turn initiate logic to fetch and update weather data.

CHALLENGES FACED

• API Key Management:

Ensuring the OpenWeatherMap API key remained functional without exceeding usage limits was a challenge during testing.

• JSON Structure Changes:

Minor changes in the API's JSON structure during development occasionally broke the parsing logic, requiring extra handling.

• Thread Synchronization:

Ensuring UI elements were only updated from the main thread required careful use of runOnUiThread() to avoid crashes.

• Error Display:

Designing meaningful error messages for invalid input or network issues was essential to maintain good user experience.

• **UI Responsiveness:**

Keeping the app responsive while managing asynchronous tasks demanded thoughtful architecture and efficient use of Java's multithreading.

FUTURE SCOPE

The SkyCast application holds significant potential for future enhancements, which aim to elevate user experience and expand its functionality. Here are some prospective features:

1. 7-Day Weather Forecast: Enabling users to view weather forecasts for an entire week would empower them to plan better for upcoming activities.

2. Geolocation Services: Integrating geolocation can automatically fetch the user's current location, providing them with instant weather updates without manual input.

3. Enhanced UI Elements: Continued refinement of the user interface can improve usability. Features like responsive design and animated transitions could make navigation even more intuitive.

4. Dark Mode: Implementing a dark mode option caters to user preferences for reduced eye strain in low-light environments, enhancing overall app accessibility.

5. Notification Alerts: Introducing customizable notification alerts for extreme weather conditions or daily summaries can keep users updated without them having to check the app constantly.

6. User Feedback System: Allowing users to submit feedback directly through the app can foster community engagement and provide insights for further development.

7. Data Visualization: Advanced visualizations such as graphs for temperature changes or precipitation levels can help users better understand weather patterns.

By prioritizing these enhancements, SkyCast can maintain its relevance and continue providing superior weather forecasting capabilities to its users.

CONCLUSION

SkyCast demonstrates how Java can be effectively used in Android development to create robust, real-time applications. This project helped us strengthen our understanding of OOP, API integration, JSON parsing, multithreading, and exception handling. It also showcased the practical challenges of mobile development, such as maintaining UI responsiveness and ensuring fault tolerance. The hands-on approach allowed our group to connect Java theory to a working product, enhancing both our technical and collaborative skills.

REFERENCES

• OpenWeatherMap API Documentation

• Android Developer Guide

• OkHttp Documentation

• JSON Parsing in Java – JournalDev

• Official Java Documentation

• Class notes and lab exercises on Java networking and threading