A Project Report

On

HMPV RISK IDENTIFICATION AND ADVICE SYSTEM

Submitted in partial fulfilment of the
requirements for the award of the degree
of

M6ASTER OF COMPUTER APPLICATIONS

SUBMITTED BY

KUNDRAPU RAM KUMAR

(23P31F00C3)

Under the Esteemed Guidance of

Ms. MOHAMMAD GOUSIA, MCA

Assistant Professor

DEPARTMENT OF MCA

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, affiliated to JNTUK & Accredited by NBA, NAAC with 'A+' grade)

Recognized by UGC under the sections 2(f) and 12(b) of the UGC acet 1956

SURAMPALEM – 533 437, E.G.Dt, ANDHRA PRADESH

2023 – 2025

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, Affiliated to JNTUK & Accredited by NBA, NAAC with 'A+'grade)
Recognized by UGC under the sections 2(f) and 12(b) of the UGC acet 1956
SURAMPALEM –533 437, E.G.Dt, ANDHRA PRADESH
2023– 2025

DEPARTMENT OF MCA

CERTIFICATE

This is to certify that the project work entitled, "**HMPV RISK IDENTIFICATION AND ADVICE SYSTEM**" is a Bonafide work by **KUNDRAPU RAM KUMAR** bearing Regd.No: **23P31F00C3** submitted to the faculty of Master of Computer Applications, in partial fulfilment of the requirements for the award of degree of **MASTER OF COMPUTER APPLICATIONS** from Aditya College of Engineering & Technology, Surampalem during the academic year 2024- 2025.

Project Guide
Ms. Mohammad Gousia, MCA
Assistant Professor,
Department of MCA,
Aditya College of Engg & Tech,
Surampalem-533437.

Head of the Department
Mr. R. V. V. N. Bheema Rao, M.Tech
Associate Professor,
Department of MCA,
Aditya College of Engg and Tech,
Surampalem-533437.

External Examiner

DECLARATION

I here by declare that the project entitled **"HMPV IDENTIFICATION AND ADVICE SYSTEM"** done on my own and submitted to **Aditya College of Engineering & Technology, Surampalem** has been carried out by me alone under the guidance of **Ms.Mohammad Gousia.**

Place:
Surampalem Date:

(**K.RAM KUMAR**)

23P31F00C3.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank is my guide **Ms.Mohammad.Gousia, MCA, Assistant Professor, Aditya College Of Engineering And Technology, Surampalem,** Her wide knowledge and logical way of thinking have made a deep impression on me. Her understanding, encouragement, and personal guidance have provided the basis for this project. She is a source of inspiration for innovative ideas and her kind support is well known to all her students and colleagues.

I wish to thank **Mr.R.V.V.N.Bheemarao, M.Tech, Head of the Department, Aditya College Of Engineering And Technology, Surampalem,** has extended his support for the success of this project.

I also wish to thank **Dr.D.Sanjay, Principal, Aditya College Of Engineering And Technology, Surampalem,** who has extended his support for the success of this project.

I would like to thank all the **Management & Technical Supporting Staff** for their timely help throughout the project.

ABSTRACT

ABSTRACT

The Hmpv Shield project seeks to address the urgent need for effective monitoring and management of human metapneumovirus (HMPV) infections, particularly in vulnerable populations such as infants and the elderly. Current diagnostic and tracking methods are often insufficient, leading to delayed interventions and increased health risks. The project aims to develop a user-friendly Android application using Flutter and Dart to provide real- time tracking of HMPV symptoms, access to educational resources, and the ability to connect with healthcare professionals for timely advice and treatment. By leveraging mobile technology, the Hmpv Shield app empowers users to proactively manage their health and reduce the risk of severe respiratory illnesses caused by HMPV.

The Hmpv Shield app will include key features such as real-time news updates on HMPV outbreaks, a directory of nearby hospitals for immediate care, a comprehensive list of symptoms to monitor, and prevention tips to reduce the risk of infection. In addition, the app will provide valuable educational resources to help users understand HMPV, its transmission, and effective management strategies. This holistic approach aims to engage users in proactive health management, promote timely intervention, and improve overall community health by making essential health information more accessible and actionable.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SCREENSHOTS

CHAPTER 1
INTRODUCTION

1. INTRODUCTION

1.1 Brief Information about the project:

The HMPV Risk Identification and Advice System is a mobile application developed to track, manage, and raise awareness about Human Metapneumovirus (HMPV) infections. HMPV is a respiratory virus that poses significant health risks, especially to infants, elderly individuals, and those with weakened immune systems. Many existing health apps focus on common respiratory diseases like COVID-19 or the flu but lack dedicated resources for HMPV. This app fills that gap by providing real-time information, symptom tracking, and preventive guidance to help users protect themselves and their communities.

Built using Flutter and Dart, the app ensures a smooth user experience with a modern and intuitive interface. Key features include detailed symptom displays, prevention tips, live updates on outbreaks, a myth-busting section, and a GPS-based hospital locator. The real- time tracking and integration with health advisories make it a valuable tool for early detection and interve ntion.

By leveraging mobile technology and real-time data, the HMPV Risk Identification and Advice System empowers users with reliable health information, promotes early medical consultation, and ultimately reduces the spread and impact of HMPV infections. It serves as a comprehensive health companion, ensuring users remain informed and prepared against potential outbreaks.

1.2 Motivation of project:

The primary motivation behind developing the HMPV Risk Identification and Advice System is the lack of awareness and dedicated tracking systems for Human Metapneumovirus (HMPV), a respiratory virus that poses significant health risks, particularly to infants, elderly individuals, and immunocompromised patients. Unlike COVID-19 or influenza, HMPV is often underdiagnosed due to limited resources and the absence of real-time monitoring tools. Many existing health applications provide only generalized respiratory illness information, failing to address the specific needs and management strategies for HMPV.

This project aims to bridge the gap in HMPV awareness by providing a comprehensive and user-friendly mobile application that offers real-time updates, symptom tracking, preventive guidelines, and access to nearby healthcare facilities. By leveraging technology and digital health solutions, the app seeks to empower users with accurate information, encourage early intervention, and minimize the risk of severe respiratory infections.

1.3 Objective of the project:

The objective of the HMPV Risk Identification and Advice System is to provide a comprehensive and accessible digital platform for monitoring, managing, and preventing Human Metapneumovirus (HMPV) infections. The app aims to increase public awareness by offering detailed information on symptoms, transmission, and preventive measures, ensuring that users have the knowledge needed to protect themselves and their communities.

Another key objective is to facilitate early detection and intervention by incorporating real-time updates, push notifications, and a symptom tracker, allowing users to monitor their health effectively. The app also seeks to enhance healthcare accessibility through its GPS-based hospital locator, enabling users to quickly find nearby medical facilities in case of emergencies.

Furthermore, the project emphasizes health education by including a myth-busting section, which helps dispel misinformation and provides scientifically backed insights about HMPV. By leveraging Flutter and Dart technology, the app ensures seamless performance, cross-platform compatibility, and user-friendly navigation.

1.4 Organisation of the project:

- **Literature Survey:** This chapter contains base information of previous models.
- **System Analysis:** The description of the current system, the planned system, and the required specifications make up the majority of this chapter.
- **System Design:** This chapter is consisting of modules description and algorithms with example and use case diagrams, class diagrams, sequence diagrams, Collaboration diagrams and activity diagrams.
- **Technology Description:** This chapter mainly consists of the technology description of this project.
- **Sample Code:** This chapter consisting of sample code for the few modules.
- **Testing:** This chapter contains of testing techniques and test cases for modules.
- **Screen Shots:** This chapter is mainly consisting of output screens of this project.
- **Conclusion:** Main Conclusion of the project is to segmenting the customers based on characteristics.

CHAPTER 2
LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 COVI: A Covid-19 Tracker Application :

The following research will focus on building a COVID-19 tracker app called COVI, with the main objective of keeping the people updated and educated on the Covid-19 virus. The user is required to provide different kinds of inputs in order for the app to be able to provide the user with many useful outputs, which can help the user in case of a Covid-19 Infection. This paper aims to implemented an android based Covid-19 tracker app, and in order to do that the research is divided into two main parts, which is front-end and back-end application. This is accomplished by combining a number of tools and programs such as Android Studio, Java Programming Language, Extensible Markup Language (XML), Google's Firebase program, a web-based corona API website and many other visual aspects that will for sure make it quite enjoyable and educational for the user to use. By building an app as such, that we can help in educating people on the Covid-19 virus while making it easier for the user to reach for help in case of an infection and hence reveals very promising results.

2.2 An Android-Based Application to Detect COVID-19 and Pneumonia :

The novel coronavirus disease has produced destructive effects on human life, taking away millions of lives. The biggest bottleneck in detecting the COVID-19-affected patient is the limited availability and time-consuming features of conventional RT-PCR tests and the lack of specialized sample extraction laboratories. Early detection of this virus may help in the advancement of a medication approach and disease control strategies.

In this research, we have developed an Android smartphone application that can detect pneumonia and COVID-19 from chest X-ray photographs using convolutional neural network deep learning algorithms (VGG16 and VGG19). The COVID-19, pneumonia, and healthy chest X-ray images are collected from various repositories of a public database, Kaggle. After applying the data augmentation technique, 9,000 chest X- ray photographs were used for training, including 3,000 images for COVID- 19, pneumonia, and normal cases. For testing, 3,000 chest X-ray photographs were collected, with 1,000 images for all three cases. The VGG16 model achieved better performance than the VGG19 with a training accuracy of 98.31% and validation accuracy of 95.03%. Next, the deep learning-based automatic classification framework is deployed into a smartphone application. Finally, the application has been tested and assessed by a focused group, and analytical results have been presented.

2.3 Human Metapneumovirus Infections during COVID-19 Pandemic, Spain :

We describe an unusual outbreak of respiratory infections caused by hmpv in children during the sixth wave of COVID-19 in Spain, associated with the Omicron variant. Patients in this outbreak were older than usual and showed more hypoxia and pneumonia, longer length of stay, and greater need for intensive care.

2.4 COWAR: An Android Based Mobile Application to Help Citizens :

Across the world, people are facing unforeseen challenges due to Coronavirus pandemic. Almost 213 countries are under the influence of the virus and the whole world population is uncertain about how long this pandemic will last. The only thing that can prevent its exponential growth apart from a vaccine is to be well aware of the situation and follow the necessary precautions. In this paper, we design and develop an android application to spread awareness and to help the people of the world amid this COVID-19. The COWAR app connects the people with the Doctors and the administration to come together and fight the pandemic. With this app, one can track the spread of the COVID-19 epidemic, check live statistics, check for symptoms, browse an interactive map, and also get prevention details. Mobile applications like these help to connect the authorities with people and provide the essential information and services needed in this pandemic. We have validated our app by distributing it with a large number of people and collecting the feedback on various features of the app.

CHAPTER 3
SYSTEM ANALYSIS

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The current health information and tracking systems mainly focus on general disease awareness rather than providing specific and detailed information about HMPV (Human Metapneumovirus). Most existing systems offer only basic symptom checkers and preventive guidelines, making it difficult for users to identify HMPV-specific risks. These systems do not provide real-time outbreak tracking or personalized health alerts, which limits their effectiveness in disease prevention.

Disadvantages:

- **Limited Information:** Most apps do not display detailed virus-specific symptoms, leading to a lack of awareness about HMPV specifically.
- **No Healthcare Integration:** Current systems do not integrate with healthcare providers for quick consultations or health support.
- **Lack of Updates and News:** Limited or no live updates regarding the HMPV virus, its spread, and preventive measures.
- **No Personalized Alerts:** Existing systems do not provide customized warnings based on the user's location, symptoms, or risk level.
- **Lack of Emergency Assistance:** No direct integration with emergency helplines or nearby hospitals, delaying urgent medical help for users.

3.2 PROPOSED SYTSEM

HMPV Risk Identification and Advice System is an Android app designed to keep users informed and protected against the H MPV virus. It offers detailed information on the symptoms of the virus, along with preventive tips and health advice to help users stay safe. The app also sends updates and news related to HMPV outbreaks, ensuring users are aware of the latest developments. Additionally, it uses GPS to show nearby hospitals, making it easier for users to find medical help if needed. This combination of features provides users with the resources they need to stay healthy and informed.

Furthermore, the HMPV Risk Identification and Advice System incorporates an intuitive interface, ensuring users can easily navigate through its features. With real-time alerts and expert-backed health recommendations, the app empowers individuals to take proactive steps in safeguarding their well-being. Whether tracking the latest outbreaks, accessing emergency healthcare services, or learning about effective preventive measures, users can rely on this app as a trusted companion in staying protected against HMPV.

ADVANTAGES:

- **Detailed Virus Information:** Users get accurate information on HMPV symptoms and how to recognize the virus.
- **Real-time Updates:** The app provides notifications and news regarding outbreaks, preventive measures, and health updates.
- **Health Support Integration:** Users can find nearby hospitals and healthcare centers for immediate support.
- **Educational Content:** The app educates users by debunking myths and providing facts about the virus.

3.3 Feasibility Study

A Feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions. It must therefore, be conducted with an objective, unbiased approach to provide information upon which decisions can be based. Here, we discuss 3 major feasibility studies required for our project.

- Economic Feasibility
- Technical Feasibility
- Operational Feasibility

3.3.1 ECONOMIC FEASIBILITY

Economic Feasibility defines whether the expected benefit equals or exceeds the expected costs. It is also commonly referred to as cost/benefit analysis. The procedure is to determine the benefits and the savings expected from the system and compare them with the costs. A proposed system is expected to outweigh the costs. This is small project with no cost for development. The system is easy to understand use. Therefore, there is no need to spend on training to use the system. This system has the potential to grow by adding functionalities for persons. Hence, the project could have economic benefits in future.

3.3.2 TECHNICAL FEASIBILITY

Technical feasibility is carried out to determine whether the project is feasible in terms of software, hardware, personnel, and expertise, to handle the completion of the project. It considers determining resources for the proposed system. As the system is developed using python, it is platform independent. Therefore, the users of the system can have average processing capabilities, running on any platform. The technology is one of the latest hence the system is also technically feasible.

3.3.3 OPERATIONAL FEASIBILITY

Operational feasibility is the measure of how well a proposed system solves the problems with the users. Operational feasibility is dependent on human resources available for the project and involves projecting whether the system will be used if it is developed and implemented. The project is operationally feasible for the users as nowadays almost all the humans are familiar with digital technology.

3.4 FUNCTIONAL REQUIREMENTS

- Shows a list of symptoms of the HMPV virus. Lists preventive
- measures for HMPV. Debunks common myths about HMPV.
- Finds nearby hospitals based on user location. Provides
- evolutionary history and genetic analysis of HMPV. Displays
- the latest news on HMPV and related outbreaks..
- Shows real-time statistics of HMPV cases worldwide.

3.5 NON-FUNCTIONAL REQUIREMENTS

- **Performance:** The app should load quickly and provide real-time updates on user progress and activities.
- **Usability**: The app should be easy to use, with an intuitive interface and minimal learning curve.
- **Scalability:** The system should be able to handle a large number of users and data.
- **Compatibility:** The app should be compatible with most Android devices (Android 6.0 and above).

- **Reliability:** The system should function consistently without crashes or errors, ensuring continuous availability of real-time updates, notifications, and healthcare resources to users.
- **Maintainability:** The app should be easily upgradable and maintainable, allowing for bug ixes, feature enhancements, and system improvements without disrupting user experience.

3.6 REQUIREMENTS SPECIFICATION

3.6.1 Minimun Hardware Requirements:

- **Device** : Android smartphone at least 2 GB RAM
- **Processor** : Quad-core 1.5 GHz or higher
- **Storage** : 16 GB of available storage
- **Connectivity** : Internet access (Wi-Fi or mobile data)

3.6.2 Minimum Software Requirements:

- **Operating System** : Android 5.0 (Lollipop) or higher
- **Environment** : Visual Studio Code or Android Studio
- **Dev IDE** : Android Emulator or a physical device for testing
- **Additional Tools** : Flutter SDK and Dart SDK

CHAPTER-4
SYSTEM DESIGN

4.SYSTEM DESIGN

4.1 Introduction

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

4.2 Data Dictionary

A data dictionary contains metadata and essential information about a database, including details on what is stored, who can access it, and where it is physically located. It plays a crucial role in database management but is primarily handled by database administrators rather than regular users. The data dictionary typically includes the names of all database tables, their schemas, and other relevant details that help manage and organize the database efficiently.

4.3 Data Flow Diagram

Data flow-oriented techniques advocate that the major data items handled by a system must be first identified and then the processing required on these data items to produce the desired outputs should be determined. The DFD (also called as bubble chart) is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on these data, and the output generated by the system. It was introduced by De Macro (1978), Gane and Sur son (1979). The primitive symbols used for constructing DFD's are:

Data flow:

Rhombus:

Process:

Source:

4.4 System Architecture Diagram

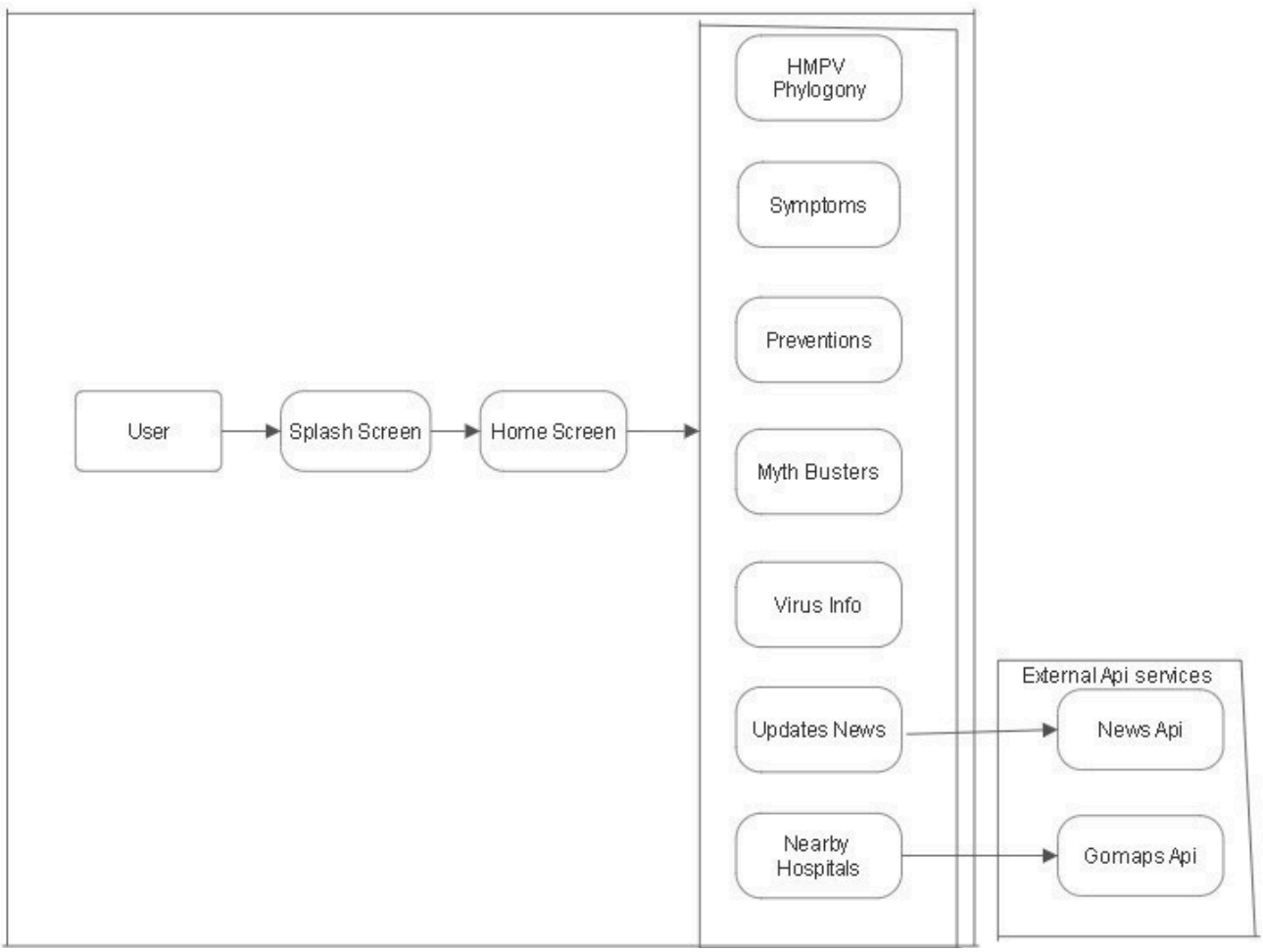


Fig: 4.4 System Architecture

4.5 UML Diagrams

The unified modelling language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. UML is specifically constructed through two different domains UML Analysis modelling, this focuses on the user model and structural model views of the system UML design modelling, which focuses on the behavioural modelling, implementation modelling and environmental model views. These are divided into the following types.

- Use case Diagram
- Class Diagram

4.5.1 Use case Diagram

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system. The primary goals of Use Case diagrams include:

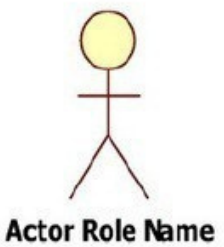
• Providing a high level view of what the system does.

• Identifying the users ("actors") of the system.

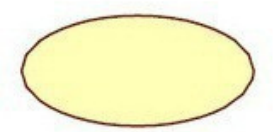
• Determining the areas needed human-computer interfaces.

Graphical Notation: The basic components of Use Case diagrams are the • Actor • Use Case • Association

Actor: An Actor, as mentioned, is a user of the system, and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor.



Use Case: A Use Case is functionality provided by the system; Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse.



Association: These Associations are used to link Actors with Use Cases, and indicate that an Actor participates in the Use Case in some form.

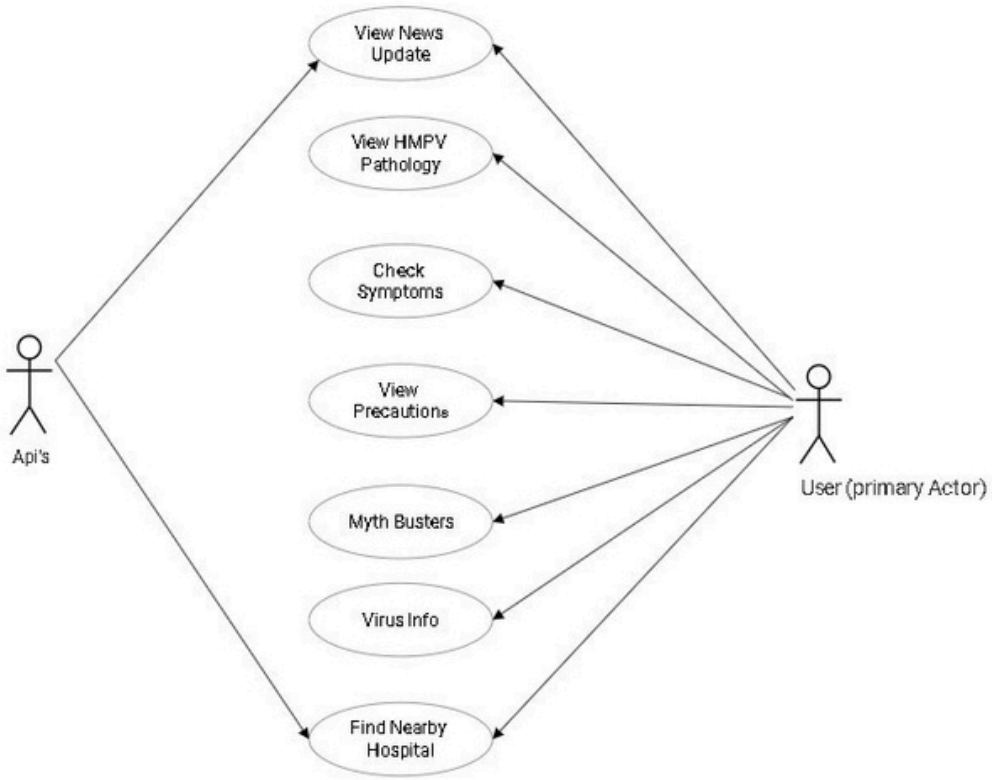


Fig : 4.5.1 Use case Diagram

Use Case Name	View HMPV Information
Actors	User
Description	Displays information about HMPV, symptoms, precautions, and myths.
Precondition	User selects a category (Symptoms, Precautions, Myths).
Postcondition	Information is displayed to the user.
Normal Flow	1. User selects a category. 2. The system retrieves data. 3. The app displays the information.
Alternate Flow	If content fails to load, display a retry button.
Exceptions	If the API call fails, display an error message.

TABLE 4.5.1.2 Use Case: Viewing HMPV Information

Use Case Name	Find Nearby Hospitals & Doctors
Actors	User, Google Places API
Description	Fetches real-time locations of nearby hospitals and doctors.
Precondition	User must allow location access.
Postcondition	Displays a list of hospitals and doctors with contact options.
Normal Flow	1. User clicks "Find Nearby Hospitals & Doctors". 2. System gets the user's location. 3. Calls Google Places API to fetch nearby hospitals. 4. Displays results in a list.
Alternate Flow	If location permission is denied, prompt user to enable it.
Exceptions	If no hospitals are found, display an error message.

TABLE 4.5.1.2 Use Case: Finding Nearby Hospitals

Use Case Name	HMPV News Updates
Actors	User, NewsAPI
Precondition	Internet connection is available.
Postcondition	News headlines are displayed.
Flow	1. Fetch news from NewsAPI. 2. Clicking opens the full article.
Exceptions	Show error message if API fails.

TABLE 4.5.1.3 Use Case Table: HMPV News Updates

Use Case Name	HMPV Precaution Guidelines
Actors	User
Description	Provides users with safety guidelines to prevent HMPV infection.
Precondition	User selects the Precautions feature.
Postcondition	The system displays recommended precautionary measures.
Flow	1. User navigates to the Precautions section. 2. The system loads and displays prevention tips. 3. User reads the guidelines and can navigate back.
Exceptions	If content fails to load, show an error message.

TABLE 4.5.1.4 Use Case Table: HMPV Precaution Guidelines

4.5.2 Class diagram:

Class diagrams identify the class structure of a system, including the properties and methods of each class. Also depicted are the various relationships that can exist between classes, such as an inheritance relationship. Part of the popularity of Class diagrams stems from the fact that many CASE tools, such as Rational XDE, will autogenerate code in a variety of languages, these tools can synchronize models and code, reducing the workload, and can also generate Class diagrams from object-oriented code.

Class diagrams play a crucial role in object-oriented design by visually representing the blueprint of a system. They help developers, architects, and stakeholders understand how different classes interact, encapsulating attributes and behaviors within objects. A well-structured class diagram facilitates code maintainability, scalability, and reusability.

Additionally, it supports various design patterns, making it easier to implement solutions efficiently. By defining associations, aggregations, compositions, and dependencies, class diagrams ensure a clear and structured approach to system development. Furthermore, they serve as a communication bridge between technical and non-technical team members, fostering a shared understanding of system functionality.

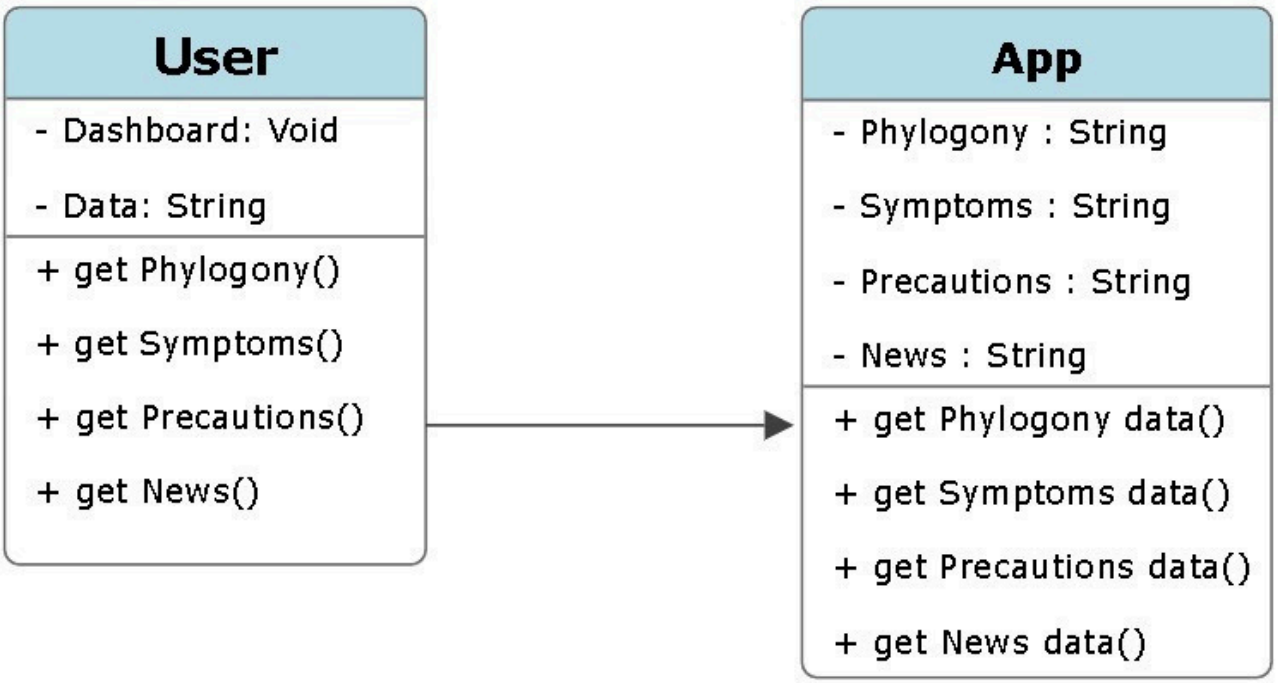


Fig: 4.5.2 Class Diagram

4.5.3 Sequence Diagram

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

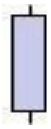
In a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time. Object Objects are instances of classes, and are arranged horizontally. The pictorial representation for an Object is a class (a rectangle) with the name prefixed by the object name (optional) and a semi-colon.

Lifeline:

The Lifeline identifies the existence of the object over time. The notation 2 for a Lifeline is a vertical dotted line extending from an object.

Activation:

Activations, modelled as rectangular boxes on the lifeline, indicate when the object is performing an action.



Message:

Messages, modeled as horizontal arrows between Activations, indicate the communication between objects.

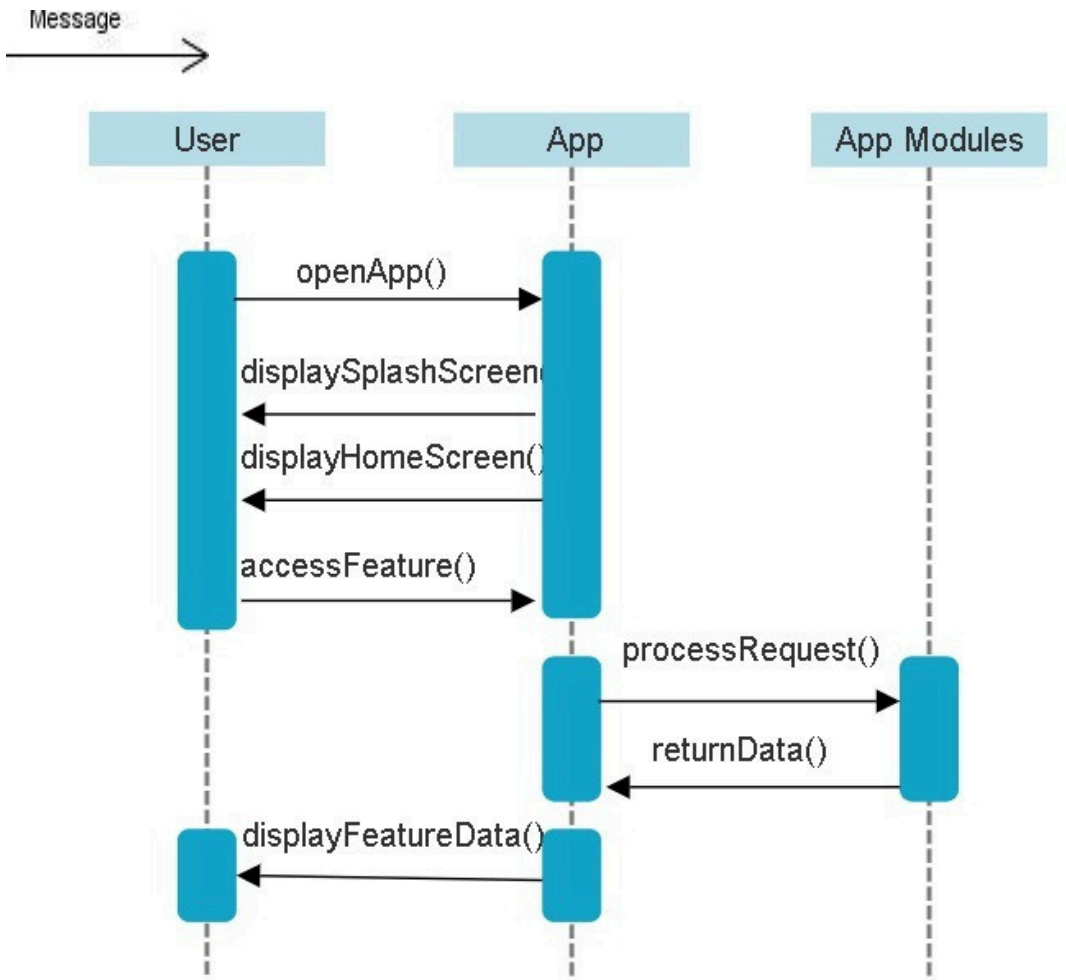
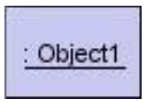


Fig : 4.5.3 Sequence Diagram

4.5.4 Collaboration Diagram

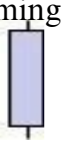
Like the other Behavioural diagrams, Collaboration diagrams model the interactions between objects. This type of diagram is a cross between an object diagram and a sequence diagram. Unlike the Sequence diagram, which models the interaction in a column and row type format, the Collaboration diagram uses the free- form arrangement of objects as found in an Object diagram. This makes it easier to see all interactions involving a particular object.

Object Objects are instances of classes, and are arranged horizontally. The pictorial representation for an Object is a class (a rectangle) with the name prefixed by the object name (optional) and a semi-colon.



Activation:

Activations, modelled as rectangular boxes on the lifeline, indicate when the object is performing an action.



Message:

Messages, modeled as horizontal arrows between Activations, indicates the communication between objects.

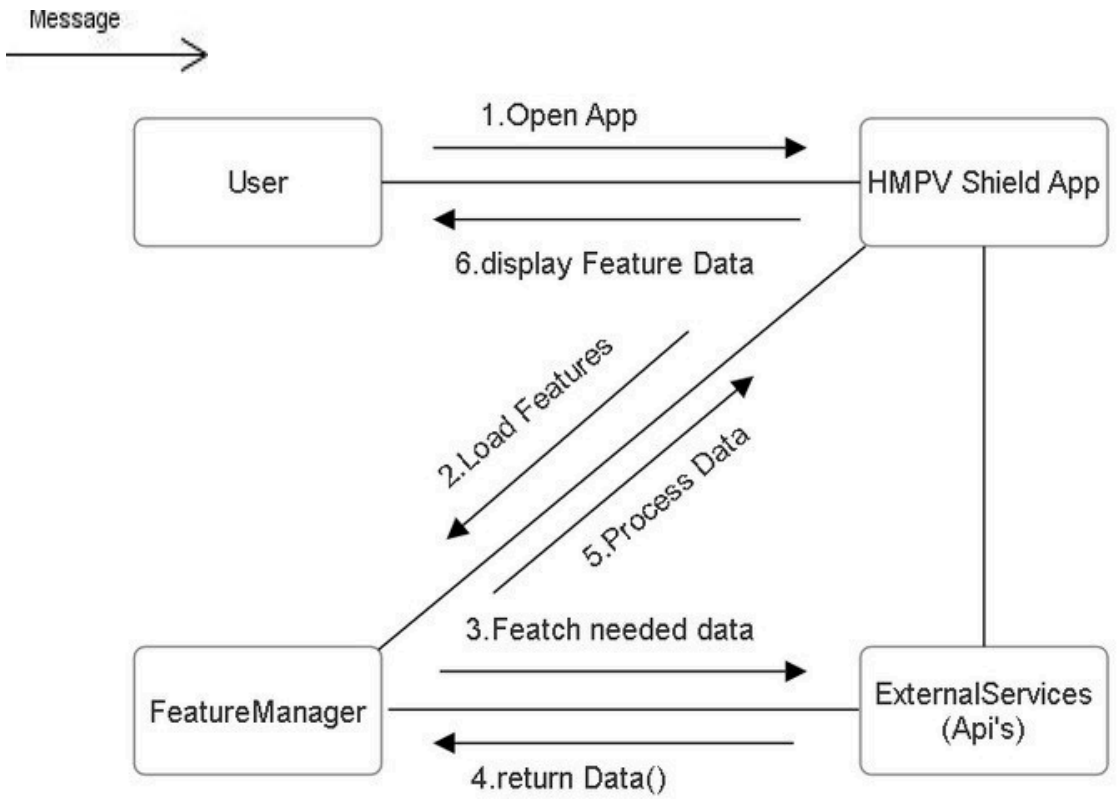


Fig : 4.5.4 Collaboration Diagram

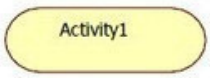
4.5.5 Activity Diagram

A use case's or an object's behaviour's activities normally take place in a particular order. A simplified view of what happens throughout a process or an operation is provided by an activity diagram. The processing within each activity moves to compilation before being automatically transmitted to the following, which is represented by a rounded rectangle. The changeover between two activities is represented by an arrow. A system is described in terms of activities in an activity diagram.

Activities are the state that denotes the performance of a sequence of actions. These are comparable to dataflow and flow chart diagrams.

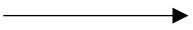
Initial state: which state is starting the process?

Action State: An action state denotes the performance of an atomic action, commonly the calling of an operation. A simple state containing an entry action and one exit transition is referred to as an action state. The exit transition is brought about by the implicit event of the entry action's completion.



Transition:

A directed relationship between the vertex of a source state and the vertex of a target state is referred to as a transition. It might be a component of a compound transition that changes the static machine's configuration from one static state to another, summarising how the static machine responded to a specific event instance.



Final State: A final state represents the last or "final" state of the enclosing composite state. There may be more than one final state at any level signifying that the composite state can end in different ways or conditions. When a final state is reached and there are no other enclosing states it means that the entire state machine has completed its transitions and no more transitions can occur.

Decision:

A state diagram (and by derivation an activity diagram) expresses decision when guard conditions are used to indicate different possible transitions that depend on Boolean conditions are used to indicate different possible transitions that depend on Boolean conditions of the owning object.

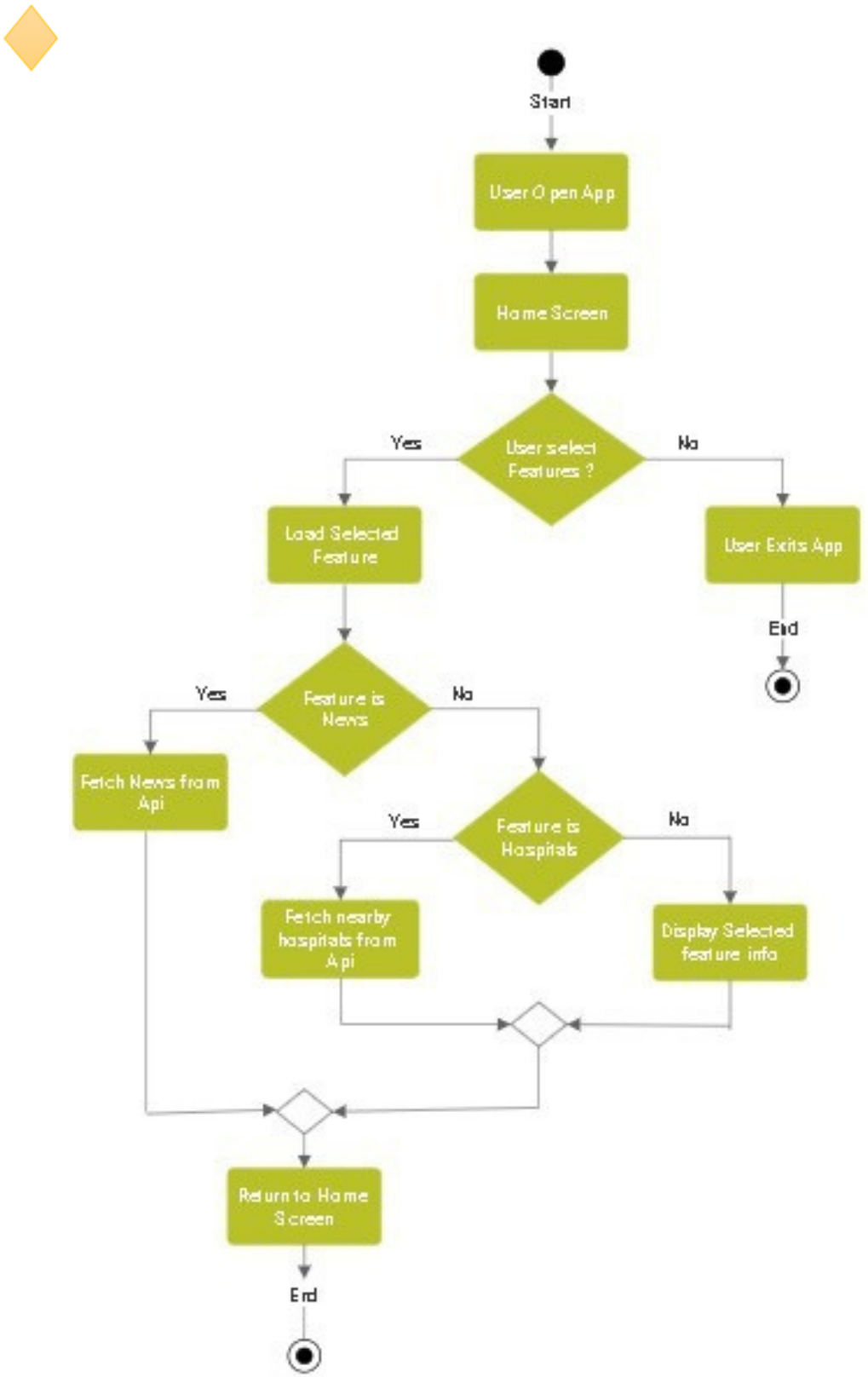


Fig : 4.5.5 Activity Diagram

CHAPTER 5
TECHNOLOGY DESCRIPTION

5.TECHNOLOGY DESCRIPTION

5.1 Introduction to Flutter and Dart :

The HMPV Risk Identification and Advice System is designed using Flutter, an open-source UI software development kit (SDK) created by Google. Flutter allows developers to create natively compiled applications for mobile, web, and desktop from a single codebase. It offers a rich set of pre-designed widgets that make it easier to build visually appealing and highly interactive applications.

The app's core functionality is built using Dart, a modern, object-oriented programming language also developed by Google. Dart is optimized for front-end development, particularly for mobile and web applications. Its clean syntax, asynchronous programming support, and compilation capabilities make it an ideal choice for developing high-performance applications.

5.1.1 Flutter Framework:

Flutter is an advanced framework that allows developers to create highly interactive and visually appealing applications. One of the standout features of Flutter is its widget- based architecture, where every element in the user interface, from buttons to text fields, is a widget. This design approach ensures a modular and flexible development process, allowing components to be reused across different screens and applications.

Another significant advantage of Flutter is its hot reload functionality. This feature enables developers to instantly see changes in the application without restarting it, significantly improving the development speed. Hot reload is particularly useful when designing user interfaces, as it allows real-time tweaks and adjustments, ensuring that the final design meets user expectations.

5.1.2 History of Flutter

Flutter was first introduced by Google in 2015 under the codename "Sky" and was initially designed to run on the Android operating system. The primary goal of Sky was to enable applications to achieve 120 frames per second (fps) for smoother animations and improved user experiences. This early prototype demonstrated the potential of Flutter in building high- performance mobile applications.

In 2017, Google officially announced Flutter at the Dart Developer Summit, highlighting its cross-platform capabilities and its ability to render UI components directly, rather than relying on native system components. This approach allowed Flutter to provide a highly consistent user experience across different platforms.

Flutter 1.0 was officially released on December 4, 2018, during the Flutter Live event. This marked a significant milestone, making Flutter a stable framework for building Android and iOS applications. Developers widely adopted Flutter due to its hot reload feature, which drastically improved development speed, and its widget-based architecture, which made UI design more flexible and intuitive.

Over the next few years, Flutter continued to evolve. In March 2021, Google released Flutter 2.0, which introduced support for web and desktop applications. This transformed Flutter from a mobile development framework into a true cross-platform solution, allowing developers to build applications for Android, iOS, Windows, macOS, Linux, and even embedded devices.

Flutter 3.0 was announced in May 2022 at the Google I/O conference, further refining its desktop and web support while enhancing performance, stability, and developer tools. Since then, Flutter has seen continuous improvements with regular updates, making it one of the most popular frameworks for mobile and cross-platform development. Today, Flutter is widely used by companies such as Alibaba, BMW, eBay, and Google itself for building scalable and visually rich applications. With a strong community and ongoing enhancements, Flutter remains a powerful choice for developers worldwide.

5.1.3 Features of Flutter

Flutter offers a wide range of features that make it a preferred framework for developers looking to build modern, high-performance applications. Some of its key features include:

1. Cross-Platform Development:

Flutter allows developers to write a single codebase that runs on multiple platforms, including Android, iOS, web, and desktop. This reduces development time and maintenance costs, making it a highly efficient choice for businesses and developers.

2. Hot Reload:

One of Flutter's standout features is hot reload, which allows developers to see the changes they make in the code instantly without restarting the application. This significantly speeds up the development process and makes debugging easier.

3. Widget-Based Architecture:

Flutter uses a widget-based approach, where everything on the screen is a widget. This makes UI development highly flexible, reusable, and customizable, allowing developers to create beautiful user interfaces with ease.

4. High Performance:

Flutter applications run at native speeds because of its Ahead-of-Time (AOT) compilation, which converts Dart code into fast machine code. Additionally, Flutter's Skia graphics engine ensures smooth animations and high frame rates.

5. Material Design and Cupertino Widgets:

Flutter supports Material Design (Android) and Cupertino (iOS) widgets, ensuring that apps have a native look and feel on both platforms. This allows developers to create visually consistent applications.

6. Expressive and Flexible UI:

Flutter provides a rich set of pre-designed widgets and customization options, allowing developers to create dynamic, attractive, and adaptive UI designs for different screen sizes and orientations.

7. Strong Community Support:

Since its launch, Flutter has gained a large and active community of developers, providing continuous improvements, third-party libraries, and extensive documentation, making it easier for new developers to learn and adopt.

8. State Management Options:

Flutter offers multiple state management solutions, including Provider, Riverpod, Bloc, Redux, and GetX, allowing developers to efficiently manage application data and UI states.

9. Native Device Features and API Access:

Flutter allows seamless integration with native device functionalities like camera, GPS, storage, and sensors through platform channels. This ensures that Flutter apps can interact with native code whenever required.

10. Security and Performance Optimization:

Flutter ensures secure and optimized performance through Dart's strong typing system, null safety, and optimized memory management. Additionally, Flutter's robust architecture makes it ideal for building scalable and secure applications.

5.1.4 Programming Language: Dart

Introduction

Dart is a general-purpose, object-oriented programming language developed by Google. It is primarily used for building cross-platform applications, especially when working with Flutter. Dart is designed to provide high performance, efficiency, and scalability in application development.

Unlike traditional languages, Dart is optimized for modern UI frameworks and offers features such as JIT (Just-in-Time) compilation for fast development and AOT (Ahead-of-Time) compilation for high-performance execution.

Why Dart?

Dart was created to overcome the limitations of JavaScript for web applications and offer a structured, scalable, and high-performance alternative. With its strong typing, garbage collection, and asynchronous programming support, Dart simplifies the development of robust applications.

Key Features of Dart as a Programming Language:

Object-Oriented: Supports class-based inheritance, interfaces, and mixins for better code structuring.

Compiled Language: Dart can be compiled into machine code (native) or JavaScript (for web apps).

Strong & Sound Typing: Provides better type safety and allows both static and dynamic typing. Asynchronous Programming: Uses Future and Stream to handle network requests, databases, and UI responsiveness.

Garbage Collection: Efficient memory management to optimize performance.Dart is the foundation of Flutter, making it a core technology for building modern applications efficiently.

5.1.5 History of Dart

The Beginning

Dart was introduced by Google in 2011 as an alternative to JavaScript. At the time, web developers faced challenges such as performance issues, lack of scalability, and weak support for structured programming in JavaScript. Google aimed to provide a faster, structured, and developer-friendly language with Dart.

Key Milestones in Dart's Evolution

2011 - Initial Release

Google introduced Dart at the GOTO Conference. Designed as a replacement for JavaScript for large-scale web applications. Introduced features like optional static typing and strong OOP support.

2013 - Dart 1.0 Release

First stable release. Launched Dart VM, allowing Dart to run as a standalone languag. Google introduced Dartium, a modified Chrome browser with Dart support.

2015 - Shift Towards Compilation

Google decided that Dart should compile into JavaScript rather than rely on a separate Dart runtime.Developers could now use Dart for both client-side and server-side development.

2017 - Introduction of Flutter & Dart 2.0

Google launched Flutter, a UI toolkit using Dart. Dart 2.0 introduced strong typing, null safety, and improved performance.

2021 - Growth of Dart & Flutter

Dart became one of the fastest-growing languages. Sound null safety was introduced, making Dart more robust. Google expanded Dart's usability beyond Flutter to web and backend development.

Present Day

Dart is a key language for mobile and web applications. Regular updates enhance its performance, security, and scalability.It is widely adopted by developers and enterprises for cross-platform development.

Dart continues to evolve, positioning itself as a powerful tool for modern application development.

5.1.6 Features of Dart

Dart has several unique features that make it a preferred choice for cross-platform and high- performance development. These features enhance code maintainability, execution speed, and development efficiency.

1. Object-Oriented Programming (OOP) Support:

Dart is fully object-oriented, meaning everything is an object, including functions.Supports class-based inheritance, abstract classes, and mixins.Allows encapsulation, polymorphism, and modular code organization.

2. Just-in-Time (JIT) & Ahead-of-Time (AOT) Compilation:

JIT compilation enables hot reload, which allows developers to see changes instantly without restarting the app.AOT compilation optimizes apps for faster startup time and better performance.

3. Strong and Sound Typing:

Dart uses strong static typing to reduce runtime errors. With sound null safety,variables cannot have null values unless explicitly allowed.

4. Asynchronous Programming with Future & Stream:

Supports async-await programming for handling network requests and background processes.Future handles single asynchronous operations, while Stream manages multiple asynchronous events.

5. Expressive and Readable Syntax:

Dart's syntax is clean, concise, and easy to read.Inspired by Java, JavaScript, and C#, making it easy to learn for developers familiar with these languages.

6. High Performance:

Dart compiles into native ARM/x86 machine code, making apps run faster than JavaScript-based applications.Optimized for performance using AOT compilation and garbage collection.

7. Memory Management & Garbage Collection:

Dart automatically manages memory, preventing memory leaks. Uses generational garbage collection for efficient memory allocation.

8. Extensive Package Ecosystem:

Dart has Pub, its package manager, offering a large collection of third-party libraries. Makes it easy to integrate APIs, UI components, and backend services.

9. Multi-Platform Compatibility:

Dart can run on Android, iOS, web, desktop, and backend servers.Supports both Flutter (mobile UI) and AngularDart (web UI).

10. Full Support for Flutter:

Dart is the exclusive programming language for Flutter.Ensures a fast, responsive, and native-like user experience with a single codebase.

Dart and Flutter together form a powerful, efficient, and innovative ecosystem for modern application development. Dart, with its strong typing, object-oriented structure, and asynchronous programming model, ensures high performance, scalability, and maintainability. It provides JIT (Just-in-Time) compilation for rapid development and AOT (Ahead-of-Time) compilation for optimized execution, making it an ideal choice for building robust applications.

On the other hand, Flutter, powered by Dart, revolutionizes UI development by offering a fast, expressive, and flexible framework for building visually stunning, cross-platform applications. With features like hot reload, customizable widgets, and native performance, Flutter allows developers to create seamless user experiences across mobile, web, and desktop platforms using a single codebase. With continuous advancements, strong community support, and increasing industry adoption, Dart and Flutter have emerged as leading technologies in modern application development. Their ability to deliver high-performance, scalable, and aesthetically rich applications makes them a preferred choice for developers and enterprises worldwide.

5.2 Application Programming Interfaces (APIs) in HMPV HMPV Risk Identification and Advice Systemd App

5.2.1 Introduction to APIs:

An Application Programming Interface (API) is a set of rules, protocols, and tools that allows different software applications to communicate with each other. APIs enable developers to integrate third-party services, fetch external data, and enhance the functionality of applications without building everything from scratch.

In modern application development, APIs play a critical role in improving efficiency, scalability, and user experience. They allow mobile applications to access external databases, cloud services, and web functionalities, making them more dynamic and interactive. APIs enable seamless data retrieval, processing, and exchange, ensuring that applications function optimally.

The HMPV App leverages APIs to fetch real-time information on HMPV outbreaks, provide location-based healthcare services, and enhance user accessibility. Instead of manually updating virus-related data, the app integrates with reliable external sources to ensure users receive timely, accurate, and relevant information.

APIs also allow applications to communicate with different platforms and devices, making them more flexible and future-ready. With API integration, the HMPV Shield App can extend its capabilities beyond HMPV tracking and integrate additional health-related services, such as AI-based health assessments, vaccine information, and symptom tracking for other respiratory diseases.

By using APIs, the HMPV Risk Identification and Advice SystemApp remains scalable, reliable, and efficient, offering users a comprehensive and real-time health monitoring tool.

5.2.2 APIs Used in HMPV Risk Identification and Advice System

The HMPV Risk Identification and Advice System App integrates multiple APIs to enhance its functionality, provide real-time updates, and improve user engagement. Below are the key APIs used in the application:

1. NewsAPI – Fetching Real-Time HMPV Updates:

The NewsAPI plays a vital role in keeping users informed about the latest developments,research findings, and outbreak news related to HMPV. Since information about viral outbreaks is constantly evolving, it is essential to provide timely and reliable updates to the public.

How NewsAPI is used in the app:

The NewsAPI connects to trusted news sources, health organizations, and research publications to fetch the latest HMPV-related updates.The app filters the most relevant articles, ensuring that users receive accurate and up-to-date information. Users can browse, search, and read news articles within the app, eliminating the need to visit multiple websites for information. The API is updated at regular intervals to ensure that breaking news and important health advisories are delivered to users in real time. This feature ensures that users are well-informed about the symptoms, precautions, myths,and emerging trends related to HMPV, helping them make informed health decisions.

2. GoMaps API – Nearby Hospital Locator:

The GoMaps API enhances the app by enabling location-based services that allow users to find the nearest hospitals and healthcare facilities. This feature is particularly useful during emergencies when users need quick access to medical assistance.

How GoMaps API is used in the app:

The GoMaps API fetches real-time hospital location data based on the user's GPS coordinates.Users can view detailed hospital information, including address, contact details, directions,and estimated travel time.

The API ensures that hospital searches are accurate, efficient, and accessible, even in remote locations.

This feature is particularly important for elderly users, individuals with pre-existing health conditions, or people experiencing severe symptoms. By providing instant access to nearby healthcare facilities, the HMPV Shield App ensures that users can seek medical attention without unnecessary delays.

5.2.3 Importance of API Integration in HMPV Risk Identification and Advice System

APIs are crucial in making the HMPV Risk Identification and Advice Systeman interactive, real-time, and data-driven application. Without API integration, the app would require manual updates and extensive backend development, reducing efficiency and scalability.

Key Benefits of API Integration:

- **Real-Time Information Access:**

The integration of NewsAPI ensures that users receive the latest updates about HMPV outbreaks, symptoms, and research findings instantly.The API ensures that hospital searches are accurate, efficient, and accessible, even in remote locations.APIs eliminate the need for manual data entry, ensuring that the information provided is always accurate and up to date.

- **Enhanced User Experience:**

With GoMaps API, users can locate hospitals quickly, improving accessibility to medical care.The seamless integration of news, location-based services, and symptom tracking creates a comprehensive health monitoring experience.

- **Accuracy and Reliability:**

By connecting to trusted sources, the HMPV Shield App ensures that users get reliable and fact-checked health information. Location services are based on real-time GPS data, ensuring precision when locating healthcare facilities.

- **Automation and Efficiency:**

APIs reduce manual intervention, allowing developers to focus on improving app performance and user engagement.The integration of third-party services ensures that the app remains lightweight, fast, and resource-efficient.

- **Scalability and Future Expansion:**

API-based development allows the app to be expanded in the future without major redesigns. Additional APIs can be integrated to include telemedicine, AI-based health assessments, and global disease tracking.

By leveraging APIs, the HMPV Shield App ensures seamless connectivity, up-to-date information, and improved healthcare accessibility, making it a valuable tool for disease prevention and health awareness.

CHAPTER 6
SAMPLE CODE

6.SAMPLE CODE

6.1 Setting Up the Flutter Project :

pubspec.yaml:(Used to download dependencies)

```
name: hmpv_virus_shield
description: "A new Flutter project."
# The following line prevents the package from being accidentally published to
# pub.dev using `flutter pub publish`. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as versionCode.
# Read more about Android versioning at https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.
# Read more about iOS versioning at
# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build suffix.
version: 1.0.0+1

environment:
  sdk: ^3.5.4

# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter
```

```
# The following adds the Cupertino Icons font to your application.
 # Use with the CupertinoIcons class for iOS style icons.
 cupertino_icons: ^1.0.8
 smooth_page_indicator: ^1.2.0+3
 http: ^1.3.0
 cached_network_image: ^3.1.0
 webview_flutter: ^4.0.4
 carousel_slider: ^5.0.0
 flutter_screenutil: ^5.5.0
 url_launcher: ^6.0.9
 shared_preferences: ^2.0.8
flutter_launcher_icons: ^0.14.3
auto_size_text: ^3.0.0
fluttertoast: ^8.0.9
 device_preview: ^1.2.0
 path_provider: ^2.0.11
 google_maps_flutter: ^2.10.1
 geolocator: ^13.0.2
 geocoding: ^3.0.0
animate_do: ^3.0.2   # Adds beautiful animations
  font_awesome_flutter: ^10.8.0
  provider: ^6.1.2
  firebase_core: ^3.12.1
  cloud_firestore: ^5.6.5
  dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^4.0.0
flutter_icons:
  android: true
  ios: true
  image_path: "assets/launcher/launcher.png"
flutter:
  uses-material-design: true
```

```
assets:
- assets/symptoms/
- assets/
- assets/prevention/
- assets/myths/
- assets/updates/
- assets/stats/
- assets/flags/
- assets/launcher/
- assets/hospitals/
fonts:
- family: Montserrat
fonts:
- asset: fonts/Montserrat-Bold.ttf
weight: 700
- asset: fonts/Montserrat-SemiBold.ttf
weight: 600
- asset: fonts/Montserrat-Regular.ttf
weight: 400
```

6.2 UI Design – Home Screen & Navigation

main.dart:

```
// lib/main.dart

import 'package:flutter/material.dart';
import 'splash_screen.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override   Widget
build(BuildContext context) {     return
MaterialApp(       title: 'Hmpv Shield',
    theme: ThemeData(
primarySwatch: Colors.blue,       ),
home: SplashScreen(),     );   } }
```

splash_screen.dart (contains app logo and tagline):

```
import 'package:flutter/material.dart';
import 'screens/home_page.dart';
class SplashScreen extends StatefulWidget {
 @override
 _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
@override  void initState() {    super.initState();    // Simulate
a      delay      for      the      splash      screen
      Future.delayed(Duration(seconds: 3), () {      // Navigate to the
home      screen      after      the      delay
Navigator.of(context).pushReplacement(
  MaterialPageRoute(builder: (context) => HomeScreen()),
);
}
);
}


  @override
  Widget build(BuildContext context) {
   return Scaffold(
    backgroundColor: Colors.black, // Set background color to black
    body: Center(
     child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
       Image.asset(
        'assets/splash.png', // Load your logo image
        width: 400, // Adjust the width as needed
        height: 400, // Adjust the height as needed
       ),
       SizedBox(height: 20),
       Text(
        'Hmpv Shield',
        style: TextStyle(
         fontSize: 24,
         color: Colors.white, // Set text color to white
         fontWeight: FontWeight.bold,
        ),
       }
```

home_page.dart (containd navigation buttons):

```
import 'package:flutter/material.dart';
import 'package:auto_size_text/auto_size_text.dart';
import 'package:animate_do/animate_do.dart'; // For animations
import 'package:font_awesome_flutter/font_awesome_flutter.dart'; // For stylish icons
import 'package:url_launcher/url_launcher.dart'; // For opening links
import '../widgets/home_page_widgets/home_categories.dart';

class HomeScreen extends StatefulWidget {
 @override
 _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
 double _userRating = 3; // Default rating

 // Show Rate Us Dialog
 void _showRateDialog() {
  showDialog(
   context: context,
   builder: (context) => FadeIn(
    duration: const Duration(milliseconds: 300),
    child: AlertDialog(
     shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
     title: const Text("Rate HMPV Shield", textAlign: TextAlign.center),
     content: Column(
      mainAxisSize: MainAxisSize.min,
      children: [
       const Text("How would you rate our app?", textAlign: TextAlign.center),
       const SizedBox(height: 10),
       StatefulBuilder(
        builder: (context, setState) => SingleChildScrollView(
         scrollDirection: Axis.horizontal,
         child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: List.generate(5, (index) {
           return IconButton(
            icon: Icon(
             index < _userRating ? Icons.star : Icons.star_border,
             color: Colors.amber,
             size: 30,
            ),
```

```
onPressed: () {
                    setState(() {
                      _userRating = index + 1;
                    });
                  },
                );
              }),
            ),
          ),
        ],
      ),
      actions: [
        _customButton("Close", Colors.grey, () => Navigator.pop(context)),
        _customButton("Submit", Colors.blue, () {
          Navigator.pop(context);
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
              content: Text("Thanks for rating $_userRating stars!"),
              behavior: SnackBarBehavior.floating,
            ),
          );
        }),
      ],
    ),
  );
 }

 // Show Credits Dialog
 void _showCreditsDialog() {
  showDialog(
    context: context,
    builder: (context) => FadeIn(
      duration: const Duration(milliseconds: 300),
      child: AlertDialog(
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
        title: const Text("App Credits", textAlign: TextAlign.center),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            const Text("Developed by: Ram Kumar\n\nSpecial thanks to contributors and
testers."),
```

```
const SizedBox(height: 15),
 Row(
 mainAxisAlignment: MainAxisAlignment.center,
 children: [
 IconButton(
 icon: const FaIcon(FontAwesomeIcons.github, color: Colors.black),
 onPressed: () => _launchURL('https://github.com/Ramkumar200314/'),
 ),
 const SizedBox(width: 20),
 IconButton(
 icon: const FaIcon(FontAwesomeIcons.linkedin, color: Colors.blue),
onPressed: () => _launchURL('https://linkedin.com/in/ram-kumar-kundrapu-57387b2a5/'),
),
],
 ),
)],,

actions: [
          _customButton("Close", Colors.grey, () => Navigator.pop(context)),
        ],
      ),
    ),
  );
 }



 // Function to open URLs properly
 void _launchURL(String url) async {
  final Uri uri = Uri.parse(url);
  if (await canLaunchUrl(uri)) {
   await launchUrl(uri, mode: LaunchMode.externalApplication);
  } else {
   throw 'Could not launch $url';
  }
 }

 // Show About Dialog
 void _showAboutDialog() {
  showDialog(
   context: context,
   builder: (context) => FadeIn(
    duration: const Duration(milliseconds: 300),
    child: AlertDialog(
```

```
shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
 title: const Text("About HMPV Shield", textAlign: TextAlign.center),
 content: const Text(
 "HMPV Shield is an awareness app designed to provide essential "
 "information about Human Metapneumovirus (HMPV).\n\n"
 "**Features:**\n"
 "✔ Symptoms, precautions, myths\n"
 "✔ Latest updates on HMPV\n"
 "✔ Nearby hospital search\n"
 "✔ Interactive statistics dashboard\n\n"
 "Stay informed, stay safe!",
textAlign: TextAlign.center,
),
actions: [
 _customButton("Close", Colors.grey, () => Navigator.pop(context)),
 ],
 ),
 ),;


/}/ Custom Button Widget for Dialogs
Widget _customButton(String text, Color color, VoidCallback onPressed) {
  return TextButton(
    onPressed: onPressed,
    style: TextButton.styleFrom(
      backgroundColor: color,
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
      padding: const EdgeInsets.symmetric(vertical: 10, horizontal: 20),
    ),
    child: Text(text, style: const TextStyle(color: Colors.white)),
  );
}

@override   Widget build(BuildContext context) {
  return Scaffold(
    resizeToAvoidBottomInset: false,
    appBar: AppBar(
      backgroundColor: Colors.transparent,
      elevation: 0,
      leading: PopupMenuButton<String>(
        icon: const Icon(Icons.menu, color: Colors.black, size: 28),
        onSelected: (value) {
          if (value == 'rate') {
            _showRateDialog();
          } else if (value == 'credits') {
            _
```

```
showCreditsDialog();
}
 },
 itemBuilder: (context) => [
 const PopupMenuItem(value: 'rate', child: Text("⭐ Rate Us")),
 const PopupMenuItem(value: 'credits', child: Text("  Credits")),
 ],
 ),
 actions: [
 IconButton(
 onPressed: _showAboutDialog, // Fixed info icon functionality
 icon: const Icon(Icons.info_outline, color: Colors.black, size: 28),
 ),
 ],
 centerTitle: true,
title: AutoSizeText(
"HMPV SHIELD",
style: const TextStyle(
 fontSize: 22,
 fontFamily: "Montserrat",
 color: Colors.black,
 fontWeight: FontWeight.w700,
 ),
 minFontSize: 14,
 stepGranularity: 2,
 maxLines: 1,
 ),
 ),
 body: HomeCategories(),
 });

}
```

home_category.dart (containd features like myths,symtoms etc.,):

```
import 'package:auto_size_text/auto_size_text.dart';
import 'package:flutter/material.dart';
import '../../screens/myths_page.dart';
import '../../screens/precautions_page.dart';
import '../../screens/symptoms_page.dart';
import '../../screens/updates_page.dart';
import '../../screens/virus_details_page.dart';
import '../../screens/stats_dashboard_screens/world_stat.dart';
import '../../screens/nearby_hospitals_screen.dart';
```

```
class CategoryTab extends StatelessWidget {
  final String imgPath, tabName, tabDesc;
  final double imgHeight, imgLeft, imgBottom;
  final Color color;
  final AutoSizeGroup titleGrp, descGrp;

  const CategoryTab({
    required this.imgPath,
    required this.tabName,
    required this.tabDesc,
    required this.color,
    required this.titleGrp,
    required this.descGrp,
    this.imgHeight = 150.0,
    this.imgLeft = 15.0,
    this.imgBottom = -8.0,
  });

  Function getPage(String tabName, BuildContext context) {
    switch (tabName) {
      case "Symptoms":
        return () => Navigator.of(context).push(MaterialPageRoute(
            builder: (context) =>
                SymptomsScreen(color: color, imgPath: imgPath)));
      case "Precautions":
        return () => Navigator.of(context).push(MaterialPageRoute(
            builder: (context) =>
                PrecautionsScreen(color: color, imgPath: imgPath)));
      case "Myths":
        return () => Navigator.of(context).push(MaterialPageRoute(
            builder: (context) => MythsScreen(color: color, imgPath: imgPath)));
      case "Virus":
        return () => Navigator.of(context).push(MaterialPageRoute(
            builder: (context) =>
                VirusDetailsScreen(color: color, imgPath: imgPath)));
      case "Updates":
        return () => Navigator.of(context).push(MaterialPageRoute(
            builder: (context) =>
                UpdatesPage(color: color, imgPath: imgPath)));
      case "Phylogeny":
        return () => Navigator.of(context)
            .push(MaterialPageRoute(builder: (context) => WorldStatScreen()));
      case "Nearby Hospitals":
```

```
return () => Navigator.of(context).push(MaterialPageRoute(
 builder: (context) =>
 NearbyHospitalsScreen(color: color, imgPath: imgPath)));
 default:
 return () {};
}
 }
   @override
    Widget build(BuildContext context) {
     return InkWell(
       onTap: () => getPage(tabName, context)(),
       child: Container(
         margin: const EdgeInsets.fromLTRB(20, 0, 20, 15),
         height: 142,
         child: Stack(
           children: <Widget>[
             Positioned.fill(
               child: Align(
                 alignment: Alignment.bottomCenter,
                 child: Container(
                   padding: const EdgeInsets.only(left: 150, right: 20),
                   width: MediaQuery.of(context).size.width,
                   decoration: BoxDecoration(
                     borderRadius: BorderRadius.circular(20),
                     color: color.withOpacity(0.13),
                   ),
                   height: 125,
                   child: Column(
                     mainAxisAlignment: MainAxisAlignment.center,
                     crossAxisAlignment: CrossAxisAlignment.start,
                     mainAxisSize: MainAxisSize.min,
                     children: <Widget>[
                       AutoSizeText(
                         tabName,
                         style: TextStyle(
                           color: color,
                           fontFamily: "Montserrat",
                           fontSize: 23,
                           fontWeight: FontWeight.w700,
                         ),
             stepGranularity: 1,
                                 maxFontSize: 23,
                                 maxLines: 1,
                                 group: titleGrp,
                               ),
```

```
AutoSizeText(
                    tabDesc,
                    style: TextStyle(
                      color: color,
                      fontFamily: "Montserrat",
                      fontSize: 19,
                      fontWeight: FontWeight.w500,
                    ),
                    stepGranularity: 1,
                    maxFontSize: 19,
                    maxLines: 3,
                    group: descGrp,
                  ),
                ],
              ),
            ),
          ),
          Positioned(
            left: imgLeft,
            bottom: imgBottom,
            child: Container(
              height: imgHeight,
              child: Hero(
                  tag: imgPath, child: Image(image: AssetImage(imgPath))),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

home_categories.dart (it used to style the home page):

```
import 'package:flutter/material.dart';
import 'home_category.dart';
import 'package:auto_size_text/auto_size_text.dart';
class HomeCategories extends StatelessWidget {
  const HomeCategories();
  static AutoSizeGroup titleGrp = AutoSizeGroup();
  static AutoSizeGroup descGrp = AutoSizeGroup();

  static List<Map<String, dynamic>> categoryData = [
```

```
{
    "imgLeft": 5.0,
    "imgBottom": 19.0,
    "imgHeight": 122.0,
    "imgPath": "assets/stats.png",
    "tabName": "Phylogeny",
    "tabDesc": "See how it evolved and spread",
    "color": Colors.deepPurpleAccent,
  },
  {
    "imgLeft": 15.0,
    "imgBottom": -8.0,
    "imgHeight": 150.0,
    "imgPath": "assets/symptoms/symptoms.png",
    "tabName": "Symptoms",
    "tabDesc": "Top HMPV symptoms",
    "color": Colors.teal[800],
  },
  {
    "imgPath": "assets/prevention/boy.png",
    "imgHeight": 140.0,
    "imgLeft": 15.0,
    "imgBottom": 0.0,
    "tabName": "Precautions",
    "tabDesc": "How to prevent being a victim",
    "color": Colors.lightBlue[700],
  },
  {
    "imgPath": "assets/myths/myths.png",
    "tabName": "Myths",
    "imgBottom": -30.0,
    "imgLeft": 20.0,
    "imgHeight": 170.0,
    "tabDesc": "Get rid of false assumptions",
    "color": Colors.redAccent[700],
  },
{
  "imgBottom": 10.0,
"imgLeft": 3.0,
"tabName": "Virus",
"imgHeight": 130.0,
  "tabDesc": "Know more about the virus",
  "imgPath": "assets/corona.png", "color":
}C,olors.orange[700],
```

```
 {
 "imgBottom": -4.0,
 "imgLeft": 8.0,
"imgPath": "assets/updates/updates.png",
"tabName": "Updates",
"imgHeight": 146.0,
"tabDesc": "View the latest news related to the virus",
"color": Colors.greenAccent[700],
},
{
 "imgBottom": 0.0, // Nearby Hospitals
 "imgLeft": 10.0,
"imgHeight": 140.0,
"imgPath": "assets/hospitals/hospital.png", // hospital image
"tabName": "Nearby Hospitals",
"tabDesc": "Find hospitals near you",
"color": Colors.blueAccent[700],
},
];
@override
Widget build(BuildContext context) {
return ListView.builder(
itemCount: categoryData.length,
physics: BouncingScrollPhysics(),
itemBuilder: (context, index) {
var cat = categoryData[index];
return CategoryTab(
titleGrp: titleGrp,
descGrp: descGrp,
imgPath: cat["imgPath"],
imgBottom: cat["imgBottom"],
imgHeight: cat["imgHeight"],
imgLeft: cat["imgLeft"],
tabDesc: cat["tabDesc"],
tabName: cat["tabName"],
color: cat["color"],
);
},
);
}
}
```

6.3 Fetching Real-Time News (NewsAPI Integration)

updates_page.dart (It contains Realtime News of HMPV):

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:webview_flutter/webview_flutter.dart';

class UpdatesPage extends StatefulWidget {
  final Color color;
  final String imgPath;

  const UpdatesPage({required this.color, required this.imgPath});

  @override
  _UpdatesPageState createState() => _UpdatesPageState();
}

class _UpdatesPageState extends State<UpdatesPage> {
  List<dynamic> _newsArticles = [];
  bool _isLoading = true;
  String _errorMessage = '';

  @override
  void initState() {
    super.initState();
    _fetchNews();

    // Initialize WebView for both Android and iOS platforms
    if (Platform.isAndroid || Platform.isIOS) {
      WebViewWidget(
        controller: WebViewController()..setJavaScriptMode(JavaScriptMode.unrestricted),
      );
    }
  }

  Future<void> _fetchNews() async {
    final url =
        'https://newsapi.org/v2/everything?
q=hmpv&language=en&apiKey=bb2225b1edd14c479ea67b09353535c6';

    try {
      final response = await http.get(Uri.parse(url));
```

```
    if (response.statusCode == 200) {
        final Map<String, dynamic> data = json.decode(response.body);

        if (data['status'] == 'ok' && data['articles'] != null) {
          setState(() {
            _newsArticles = data['articles'];
            _isLoading = false;
          });
        } else {
          setState(() {
            _isLoading = false;
            _errorMessage = 'No articles found or unexpected response from API.';
          });
        }
      } else {
        setState(() {
          _isLoading = false;
          _errorMessage = 'Failed to load news. Status Code: ${response.statusCode}';
        });
      }
    } catch (e) {
      setState(() {
        _isLoading = false;
        _errorMessage = 'Error: ${e.toString()}';
      });
      print('Error: $e');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('HMPV Virus Updates'),
        backgroundColor: widget.color,
      ),
      body: _isLoading
          ? Center(child: CircularProgressIndicator())
          : _errorMessage.isNotEmpty
              ? Center(child: Text('Error: $_errorMessage'))
: ListView.builder(
            itemCount: _newsArticles.length,
            itemBuilder: (context, index) {
              final article = _newsArticles[index];
                  }  ,return NewsCard(article: article);
        ), );}
```

```
class NewsCard extends StatelessWidget {
  final dynamic article;

  NewsCard({required this.article});

  @override
  Widget build(BuildContext context) {
   return GestureDetector(
    onTap: () {
     if (article['url'] != null) {
      Navigator.push(
       context,
       MaterialPageRoute(
        builder: (context) => WebViewScreen(url: article['url']),
       ),
      );
     }
    },
    child: Card(
     margin: EdgeInsets.symmetric(vertical: 10, horizontal: 15),
     elevation: 5,
     shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
     child: Padding(
      padding: EdgeInsets.all(15),
      child: Column(
       crossAxisAlignment: CrossAxisAlignment.start,
       children: [
        if (article['urlToImage'] != null)
         Image.network(
          article['urlToImage'],
          fit: BoxFit.cover,
          width: double.infinity,
          height: 200,
          loadingBuilder: (context, child, loadingProgress) {
           if (loadingProgress == null) {
            return child;
           }
           return Center(
            child: CircularProgressIndicator(
             value: loadingProgress.expectedTotalBytes != null
               ? loadingProgress.cumulativeBytesLoaded /
                 (loadingProgress.expectedTotalBytes ?? 1)
               : null,
            ),);},
```

```
errorBuilder: (context, error, stackTrace) {
                return Center(
                  child: Icon(
                    Icons.error,
                    color: Colors.red,
                    size: 50,
                  ),
                );
              },
            ),
            SizedBox(height: 8),
            if (article['title'] != null)
              Text(
                article['title'],
                style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
              ),
            SizedBox(height: 8),
            if (article['description'] != null)
              Text(
                article['description'],
                style: TextStyle(fontSize: 14, color: Colors.black54),
              ),
            SizedBox(height: 10),
            if (article['source'] != null)
              Align(
                alignment: Alignment.bottomRight,
                child: Text(
                  'Source: ${article['source']['name']}',
                  style: TextStyle(fontSize: 12, color: Colors.grey),
                ),
              ),
          ],
        ),
      ),
    ),
  );
 }
}
              c  lfainssa lW SetrbiVngie uwrSl;creen extends StatelessWidget {

                WebViewScreen({required this.url});

                @override   Widget
                build(BuildContext context) {
                  return Scaffold(
                    appBar: AppBar(
                      title: Text('Article'),
                      backgroundColor: Colors.blue,
                    ),
```

```
body: WebViewWidget(
      controller: WebViewController()
        ..loadRequest(Uri.parse(url))
        ..setJavaScriptMode(JavaScriptMode.unrestricted),
    ),
  );
 }
}
```

6.4 Nearby Hospital Locator (GoMaps API Integration)

nearby_hospitals.dart (It contains Nearbyhospital details and shows nearbyhospitals):

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:geolocator/geolocator.dart';
import 'package:http/http.dart' as http;

class NearbyHospitalsScreen extends StatefulWidget {
 final Color color;
 final String imgPath;

 const NearbyHospitalsScreen({required this.color, required this.imgPath});

 @override
 _NearbyHospitalsScreenState createState() => _NearbyHospitalsScreenState();
}

class _NearbyHospitalsScreenState extends State<NearbyHospitalsScreen> {
 late GoogleMapController mapController;
 LatLng _userLocation = const LatLng(16.9891, 82.2475);
 Set<Marker> _markers = {};
 bool _isLoading = true;

 static const String apiKey = "AlzaSyRt3SKMT-_tnoHEkVZzGKTkIwuChfWJRrL";
@override
 void initState() {
  super.initState();
  _determinePosition();
 }
 // Get User's Location
 Future<void> _determinePosition() async {
  LocationPermission permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied) {
```

```
permission = await Geolocator.requestPermission();
if (permission == LocationPermission.denied) {
ScaffoldMessenger.of(context).showSnackBar(
const SnackBar(content: Text("Location permission denied.")),
);
return;
}
}

Position position = await Geolocator.getCurrentPosition(
    desiredAccuracy: LocationAccuracy.high,
  );

  setState(() {
    _userLocation = LatLng(position.latitude, position.longitude);
    _markers.add(Marker(
      markerId: const MarkerId("user_location"),
      position: _userLocation,
      infoWindow: const InfoWindow(title: "Your Location"),
    ));
  });

  if (mapController != null) {
    mapController.animateCamera(CameraUpdate.newLatLngZoom(_userLocation, 14.0));
  }

  _fetchNearbyHospitals();
}

// Fetch Only Nearby Hospitals
Future<void> _fetchNearbyHospitals() async {
  setState(() => _isLoading = true);

  String url =
" http s://m aps. gom aps. pro/ map s/api/place/nearbysearch/json?
location=${_userLocation.latitude},${_userLocation.longitude}&radius=20000&type=hospital
&key=$apiKey";

  try {
    final response = await http.get(Uri.parse(url));
    final data = json.decode(response.body);

    if (data['status'] == 'OK') {
      setState(() {
```

```
 _markers.addAll(data['results'].map<Marker>((hospital) {
return Marker(
markerId: MarkerId(hospital['place_id']),
position: LatLng(
hospital['geometry']['location']['lat'],
hospital['geometry']['location']['lng'],
),
infoWindow: InfoWindow(
title: hospital['name'],
snippet: hospital['vicinity'],
 ),
          );
        }).toList());
      });

      if (mapController != null && _markers.isNotEmpty) {
        mapController.animateCamera(CameraUpdate.newLatLngBounds(_getBounds(), 100));
      }
    } else {
      print("No hospitals found. Status: ${data['status']}");
    }
  } catch (e) {
    print("Error fetching hospitals: $e");
  }

  setState(() => _isLoading = false);
}

 // Calculate map bounds to fit all hospitals
 LatLngBounds _getBounds() {
  double minLat = _userLocation.latitude, maxLat = _userLocation.latitude;
  double minLng = _userLocation.longitude, maxLng = _userLocation.longitude;

  for (var marker in _markers) {
    minLat = marker.position.latitude < minLat ? marker.position.latitude : minLat;
    maxLat = marker.position.latitude > maxLat ? marker.position.latitude : maxLat;
    minLng = marker.position.longitude < minLng ? marker.position.longitude : minLng;
    maxLng = marker.position.longitude > maxLng ? marker.position.longitude : maxLng;
  }

  return LatLngBounds(
    southwest: LatLng(minLat, minLng),
    northeast: LatLng(maxLat, maxLng),
  );
 }
```

```
void _onMapCreated(GoogleMapController controller) {
   mapController = controller;
   if (_userLocation.latitude != 0 && _userLocation.longitude != 0) {
    mapController.animateCamera(CameraUpdate.newLatLngZoom(_userLocation, 14.0));
   }
  }

  @override
  Widget build(BuildContext context) {
   return Scaffold(
    appBar: AppBar(
     backgroundColor: widget.color,
     title: const Text("Nearby Hospitals"),
    ),
    body: Stack(
     children: [
      GoogleMap(
       onMapCreated: _onMapCreated,
       initialCameraPosition: CameraPosition(
        target: _userLocation,
        zoom: 14.0,
       ),
       markers: _markers,
       myLocationEnabled: true,
       myLocationButtonEnabled: true,
      ),
      if (_isLoading)
       Center(child: CircularProgressIndicator()),
     ],
    ),
   );
  }
}
```

CHAPTER 7
TESTING

7.TESTING

7.1 Introduction to Testing

Testing is a crucial phase in software development that ensures the HMPV Shield App functions as expected, remains bug-free, and provides a smooth user experience. It involves evaluating the app's performance, security, reliability, and usability before deployment. In mobile app development, testing helps identify bugs, crashes, API failures, UI inconsistencies, and performance bottlenecks. Since the HMPV Shield App deals with real- time data, GPS-based location tracking, and external APIs, rigorous testing is necessary to ensure accuracy, responsiveness, and efficiency. This chapter focuses on the different types of testing performed on the HMPV Shield App, including functional testing, UI testing, API testing, performance testing, and security testing. Through proper testing, we ensure the app delivers reliable health information and seamless navigation for users.

7.2 Types of Testing

To ensure the HMPV Risk Identification and Advice System functions correctly and delivers a smooth user experience, various testing techniques were applied. These tests help detect bugs, performance issues, security vulnerabilities, and UI inconsistencies before deployment. The major types of testing performed on the app are:

7.2.1 Functional Testing

Functional testing ensures that each feature of the HMPV Shield App works as expected. The goal is to verify that the app correctly responds to user inputs and performs the intended tasks.

Key Functional Tests Performed:

- **User Interface Testing:** Ensures all buttons, menus, and navigation elements function properly.
- **NewsAPI Testing:** Verifies if real-time HMPV news updates are fetched correctly.
- **Hospital Locator Testing:** Confirms that the GoMaps API accurately displays nearby hospitals based on GPS.
- **Symptoms & Myths Sections:** Ensures that all prevention tips, symptoms, and myths are displayed correctly.

7.2.2 UI/UX Testing

The user experience (UX) and user interface (UI) directly impact the app's usability. UI/UX testing focuses on verifying the design consistency, responsiveness, and ease of navigation.

Key UI/UX Tests Performed:

- **Screen Responsiveness:** Ensures the app adapts to different screen sizes (mobile, tablets).
- **Text Readability & Layout:** Ensures proper font size, spacing, and content alignment.

7.2.3 API Testing

The HMPV Risk Identification and Advice Systemrelies on external APIs to fetch real-time information. API testing nsures that these integrations work properly and return accurate data.

Key API Tests Performed:

NewsAPI Testing: Ensures real-time news data is fetched and displayed without errors. GoMaps API Testing: Confirms the accuracy of hospital locations based on user coordinates. API Response Time: Measures how quickly the API fetches and processes data. Error Handling: Checks if the app gracefully handles API failures (e.g., no internet connection).

7.2.4 Performance Testing

Performance testing ensures that the app runs smoothly without lag and functions well underdifferent conditions.

Key Performance Tests Performed:

- **App Startup Time:** Ensures the app launches quickly without unnecessary delays.
- **Memory Usage:** Monitors how much RAM the app consumes to prevent crashes.
- **Battery Consumption:** Verifies that the app does not drain the battery excessively.
- **Network Load Handling:** Ensures the app loads efficiently even with slow internet connections.

7.2.5 Security Testing

Since the app handles sensitive health-related information, security testing ensures that user privacy and data protection are maintained.

Key Security Tests Performed:

- **Data Encryption:** Ensures secure communication between the app and APIs.
- **Authentication & Permissions:** Checks if the app properly handles location and API access permissions.
- **Malicious Attack Prevention:** Verifies protection against unauthorized API access and malware attacks.

7.3 Sample Test Cases and Results

This section presents sample test cases performed on the HMPV Shield App to validate its functionality, UI/UX, API integration, performance, and security. Each test case includes a test scenario, expected outcome, actual outcome, and status (Pass/Fail).

Test ID	Test Scenario	Test steps	Expected Outcome	Actual Outcome	Status
TC_OI	App Launch	Open the app on an Android device	The app should launch within 3 seconds	launches in 2.5 seconds	Pass
TC_02	Symptoms & Myths Display	Open "Symptoms" and "Myths" sections	Information shouldbe displayed properly	Datais visible and formatted correctly	Pass

7.3.1 Functional Test Cases

Test ID	Test Scenario	Test Steps	Expected Outcome	Actual Outcome	Status
TC_03	Fetch HMPV News via API	Open the news section Of the app	News headlines should load within S seconds	News loads in 3 seconds	Pass
TC_04	Nearby Hospital Locat or	Enable GPS and search for nearby hospitals	Hospitals should be displayed with addresses	Hospitals are shown with correct locations	Pass

7.3.2 API Test Cases

7.3.3 Test case 01: APP Lauchs

Description: App launches successfully

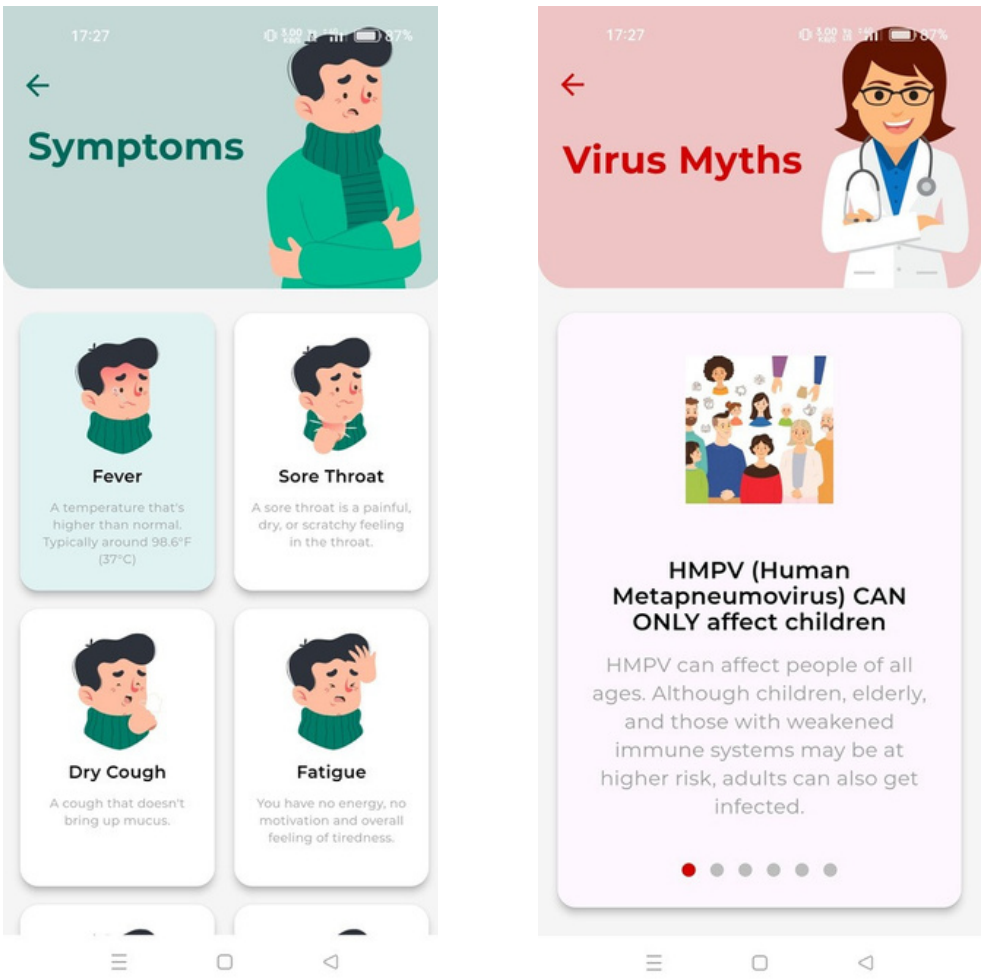


Fig 7.3.2 Test case 04 Symptoms & Myths Display

Description: After clicking symtoms & myths Data is visible test case passed succesfully.

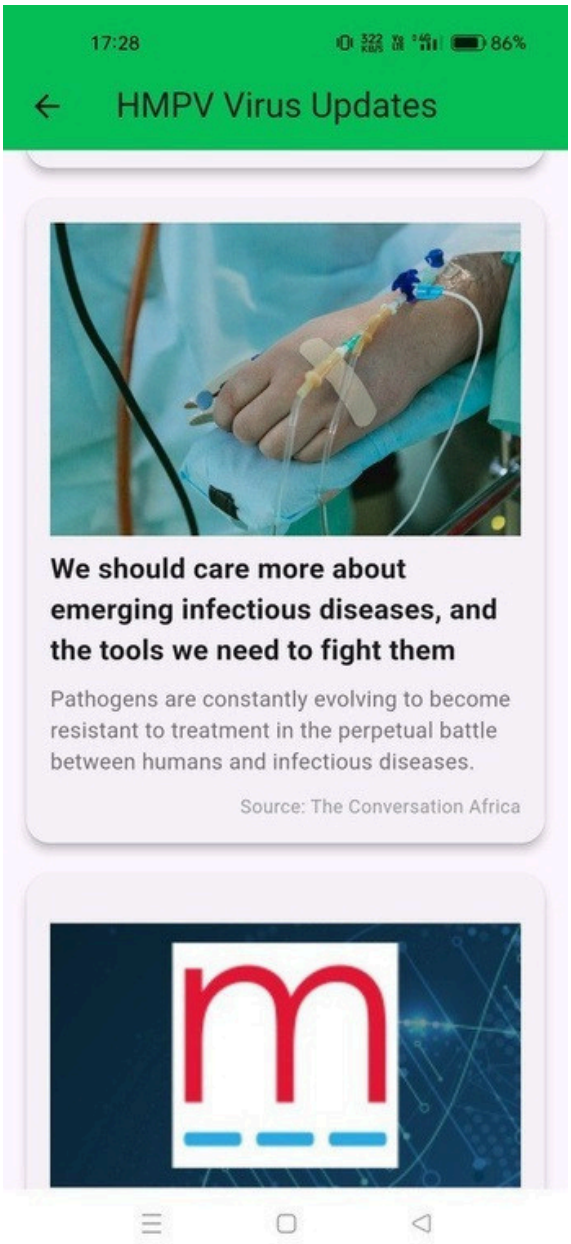


Fig 7.3.3 Test case 3: Fetch HMPV News via API

Description: After clicking updates it navigates to news tab and provide real-time news about hmpv virus.

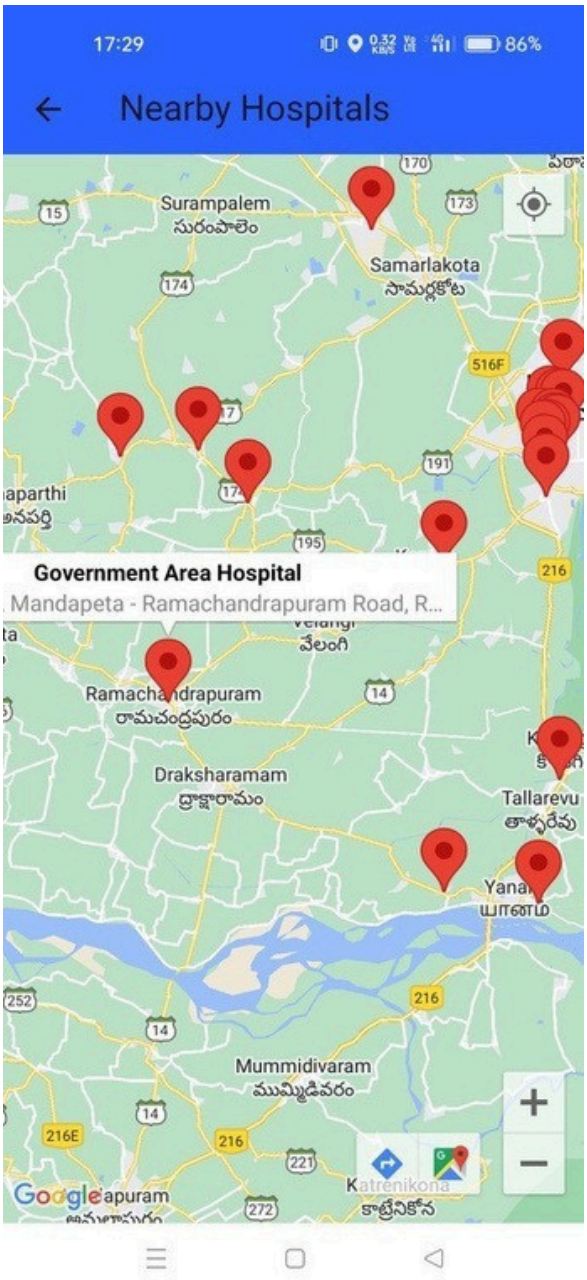


Fig 7.3.4 Nearby Hospital Locator

Description: After clicking Nearby hospitals it shows nearby hospitals based on user location.

CHAPTER 8
SCREENSHOTS

8.SCREENSHOTS

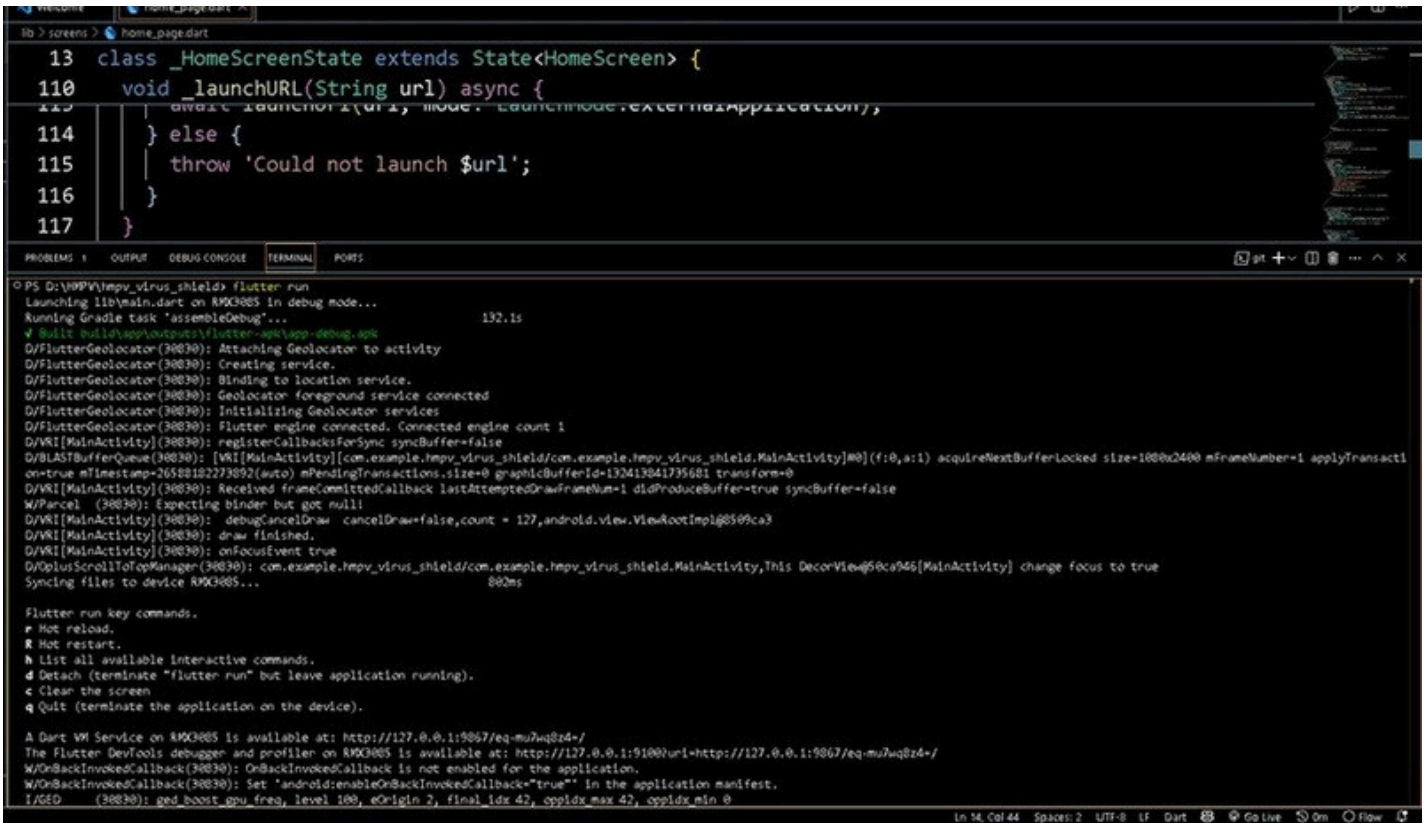


Fig 8.1 project execution terminal panel

Description: When user want to execute the code he can open the terminal panel and execute the code by running flutter run cammand in debug mode.

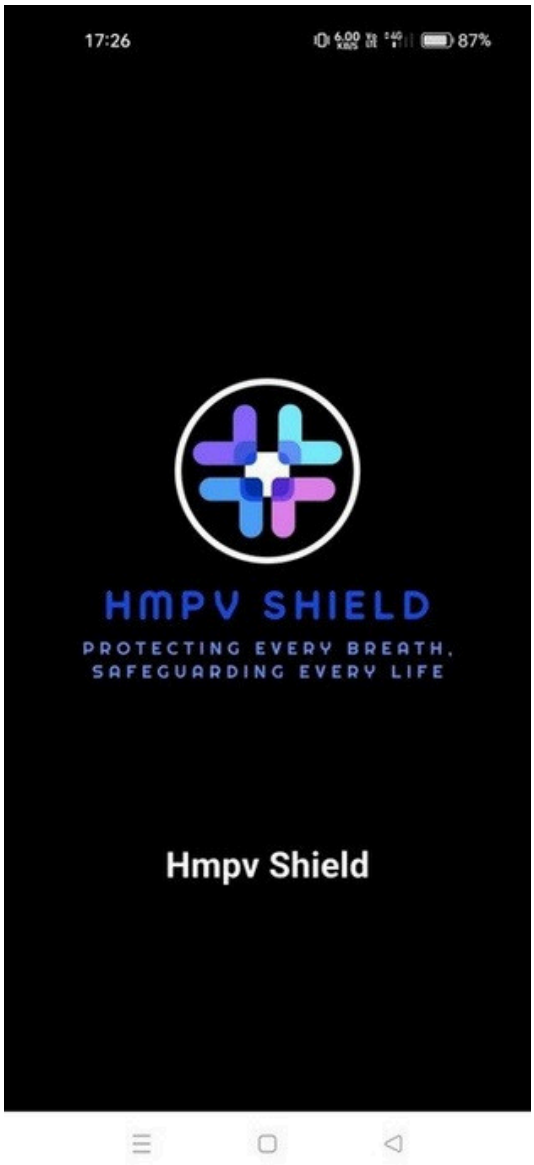


Fig 8.2 splash screen

Description: when user opens app it shows the splash screen with app logo and tagline.

Fig 8.3 Home Screen

Description: After splash screen appear then homescreen appear with different modules.

Fig 8.4 HMPV Phylogony

Description: User click Phylogony in home screen it shows hmpv phylogony data

Fig 8.5 Precautions

Description: After clicking precautions it opens precautions tab.

Fig 8.6 Myths

Description: After clicking myths it opens myths tab and shows the hmpv general myths.

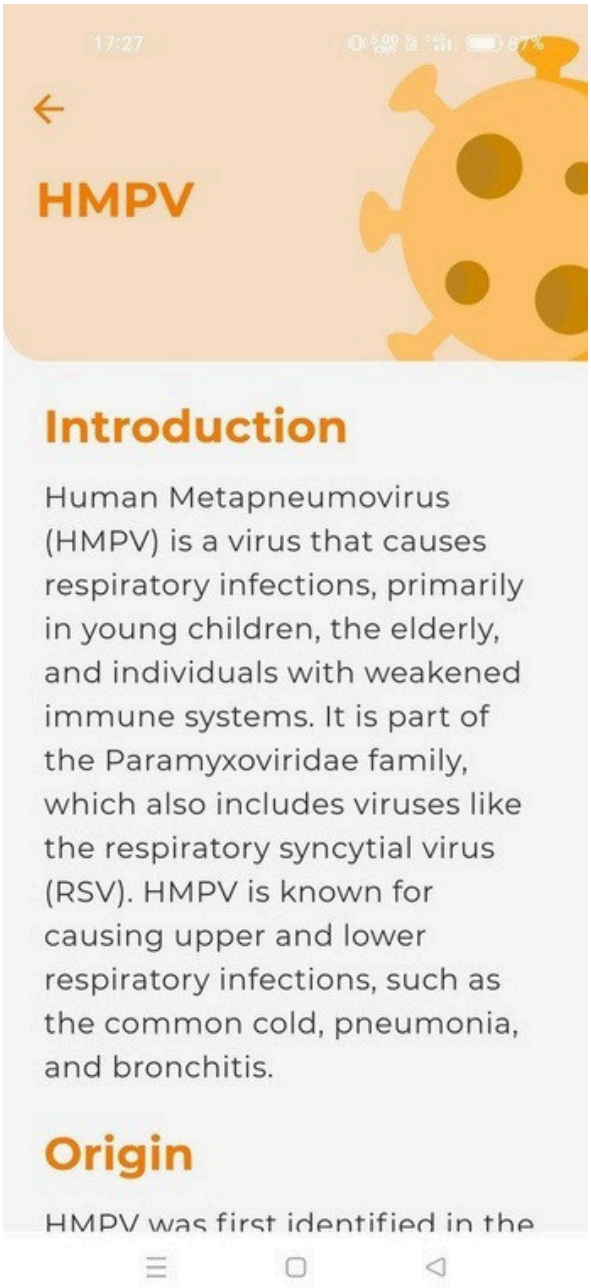


Fig 8.7 Hmpv Info

Description: After clicking virus info it navigate to virus info tab which is have a hmpv virus information.

Fig 8.8 News Article

Description: After clicking news tab in upadates it shows article about hmpv .

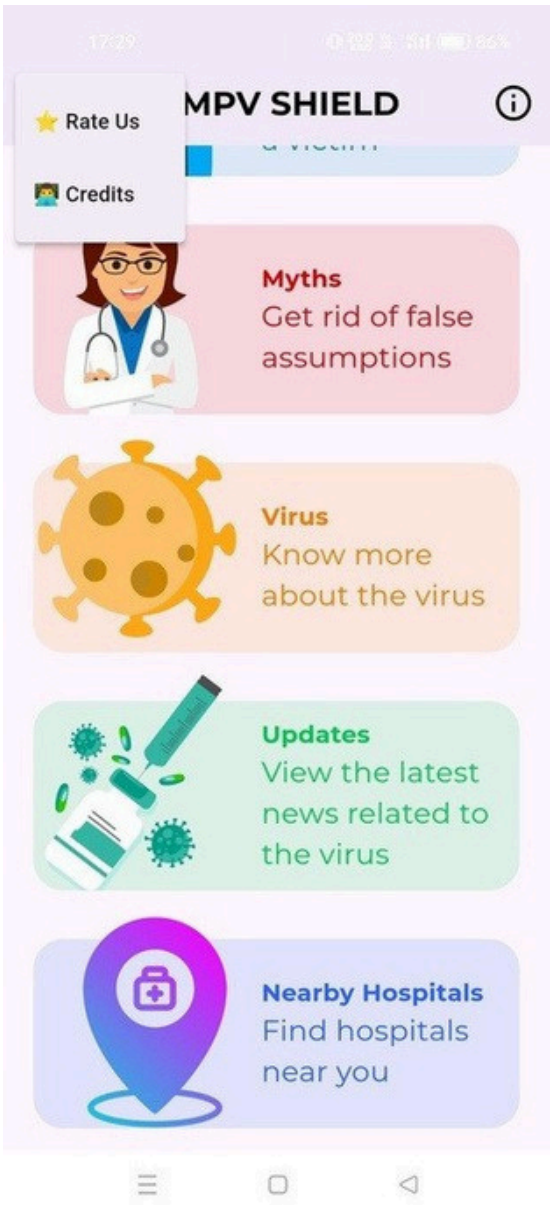


Fig 8.9 Rating & Credits Menu

Description: After clicking menu bar it shows rate us and credits .

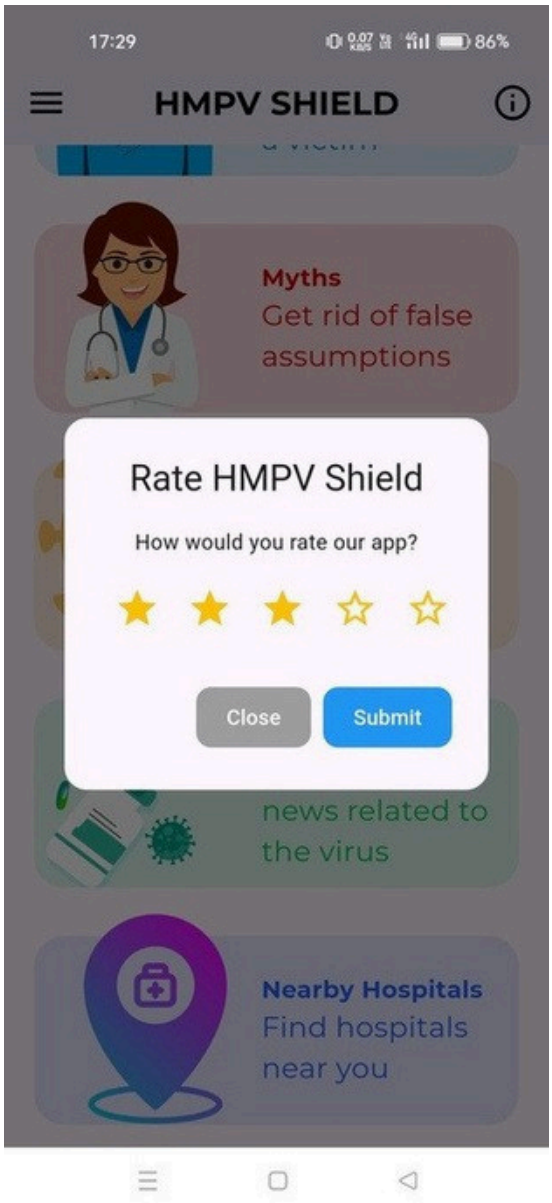


Fig 8.10 Rating app

Description: After clicking rate us it shows rate HMPV SHIELD where we need to rate the app.

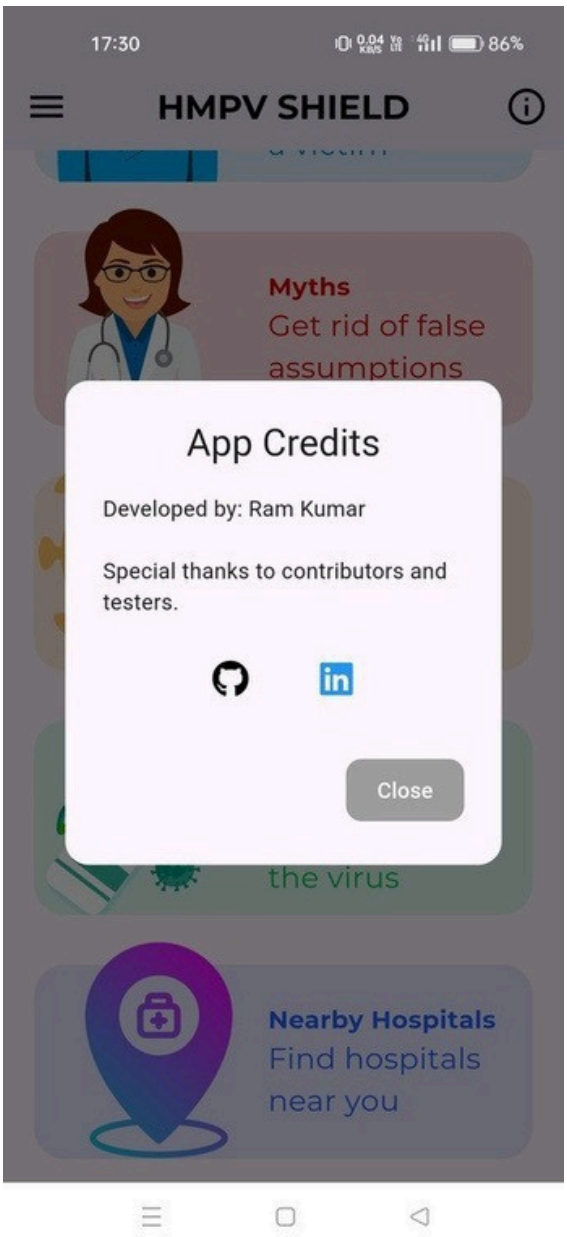


Fig 8.11 Developer credits

Description: After clicking credits in menu bar it shows who developed the app and details.

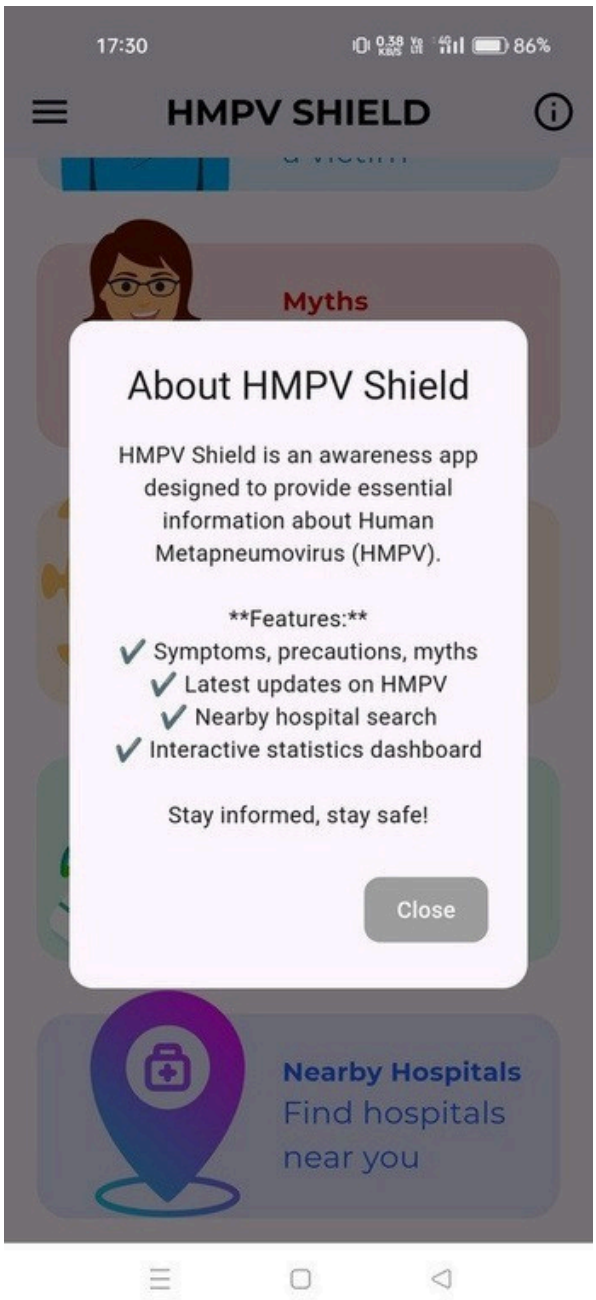


Fig 8.12 About App

Description: After clicking info symbol in homescreen it shows about the app.

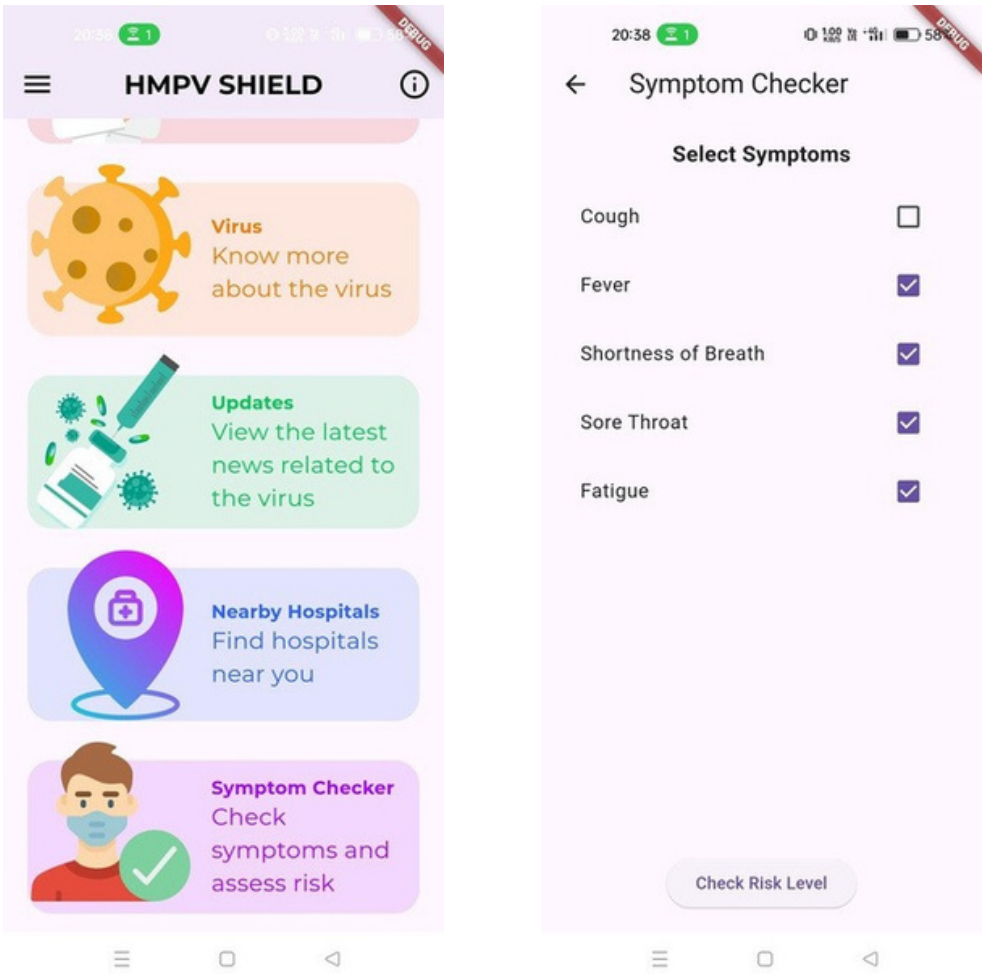


Fig 8.13 Symptom checker & Risk assesser

Description: After clicking Symptom checker & risk asseser it opens symptom cheker where user have to give input and select symptoms.

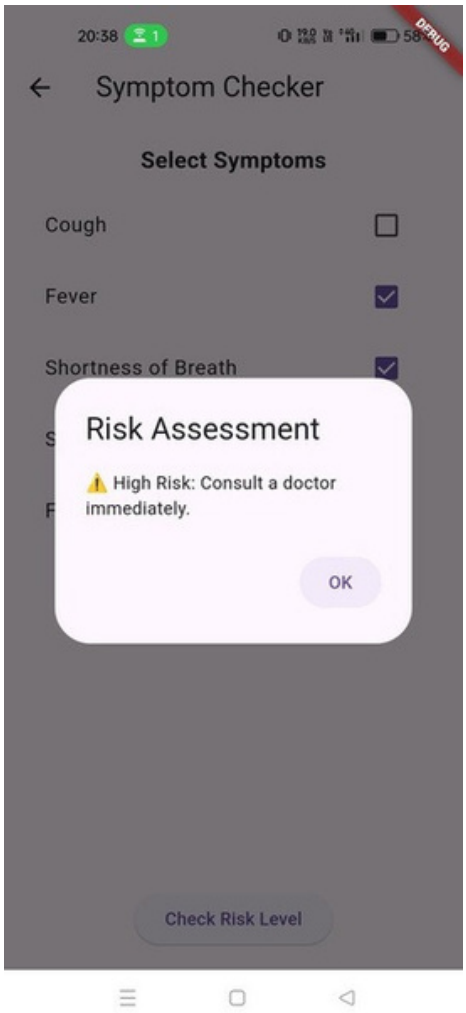


Fig 8.14 Hmpv risk assesment popup message

Description: After user giving input the risk is calculated and give the pop up message like high risk , medium risk, and low risk based on user inputs.

CONCLUSION

The HMPV Risk Identification and Advice System is a comprehensive and user-friendly solution aimed at spreading awareness and providing crucial information about the Human Metapneumovirus (HMPV). By integrating features such as phylogeny, symptoms, precautions, myths, virus information, latest updates, and nearby hospital locators, the app serves as a valuable resource for users seeking accurate and timely health-related data.

Through this project, we have successfully developed an application that enhances public awareness and assists in early detection and prevention of HMPV infections. The inclusion of real-time updates ensures that users stay informed about new developments, while the nearby hospital feature aids in quick access to medical facilities.

Future Scope:

- **AI-Driven Symptom Analysis** – Enhance the system with AI to provide accurate risk assessments based on user symptoms.
- **Telemedicine Integration –** Enable real-time consultations with doctors directly within the app.
- **Global Disease Tracking –** Implement real-time outbreak alerts to keep users informed about worldwide health threats.
- **Wearable Device & Offline Access –** Connect with health wearables for continuous monitoring and ensure offline access to critical health information.

BIBLIOGRAPHY

TEXTBOOKS REFERRED 1."Flutter for Beginners" – Alessandro Biessek (Packt Publishing): A great starting point for learning Flutter basics, UI design, and state management.

2."Flutter in Action" – Eric Windmill (Manning Publications): Covers real-world Flutter applications, including best practices for app development.

3."Dart Apprentice" – Jonathan Sande (Raywenderlich): A beginner-friendly book focusing on Dart fundamentals, including OOP concepts, functions, and async programming.

4."Flutter Cookbook" – Simone Alessandria (Packt Publishing):Provides hands-on projects and solutions for common Flutter development challenges.

WEB SITES REFERRED

1.URL:
https://en.wikipedia.org

2.URL:
http://www.google.co.in

3.URL:
https://www.javatpoint.com

4.URL:https://www.news9live.com/health/health-news/hmpv-virus-in-india-expert-debunks-common- myths-shares-the-facts-2797332News9live

PAPERS REFERRED

1."Human Metapneumovirus Is Finally Being Taken Seriously" by Maryn McKenna, Wired.WIRED

2."HMPV Is Just a Mild Seasonal Illness, Experts Say" by Claire Bugos, Verywell Health.Verywell Health+1Cleveland Clinic+1

3."The Truth About HMPV Panic, According to Experts" by Adriana Diaz, New York Post.New York Post

Vamsikrishna Mangalapalli

23P31F00C3

23P31F00C3
MCA STUDENTS
Aditya University

Document Details

Submission ID
trn:oid:::1:3213794295

Submission Date
Apr 12, 2025, 12:20 PM GMT+5:30

Download Date
Apr 12, 2025, 12:21 PM GMT+5:30

File Name
23P31F00C3.pdf

File Size
4.6 MB

93 Pages

12,511 Words

76,733 Characters

38% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

168 Not Cited or Quoted 38%
Matches with neither in-text citation nor quotation marks

1 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

32% Internet sources
13% Publications
29% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **168** Not Cited or Quoted 38%
  Matches with neither in-text citation nor quotation marks
- **1** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 32% Internet sources
- 13% Publications
- 29% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
Berhampur University		9%
2	Student papers	
University of Greenwich		2%
3	Internet	
docplayer.net		2%
4	Internet	
www.slideshare.net		2%
5	Internet	
www.proceedings.com		2%
6	Internet	
www.researchgate.net		2%
7	Publication	
Tarun Saxena, Prince Anuragi, Gandhali Shinde, Nitesh Yadav, Mayuri Digalwar. "...		1%
8	Internet	
gitlab.sliit.lk		1%
9	Internet	
www.coursehero.com		1%
10	Student papers	
Aditya University		1%

11 Internet
pdfcoffee.com <1%

12 Student papers
Nottingham Trent University <1%

13 Internet
grietinfo.in <1%

14 Internet
vdocuments.site <1%

15 Student papers
Cagayan State University - Andrews Campus <1%

16 Internet
tools.ages.pucrs.br <1%

17 Internet
repisalud.isciii.es <1%

18 Student papers
Zambia Centre for Accountancy Studies <1%

19 Student papers
Institute of Research & Postgraduate Studies, Universiti Kuala Lumpur <1%

20 Internet
open-innovation-projects.org <1%

21 Student papers
University of London External System <1%

22 Student papers
University of Northampton <1%

23 Student papers
Brampton Manor Academy <1%

24 Internet
www.digitalocean.com <1%

25 Student papers
Arab Open University <1%

26 Internet
dev.to <1%

27 Internet
ebin.pub <1%

28 Internet
foro.turismo.org <1%

29 Student papers
Arab Academy for Science, Technology & Maritime Transport CAIRO <1%

30 Internet
flux9ine.com <1%

31 Student papers
Middle East College of Information Technology <1%

32 Student papers
University of Westminster <1%

33 Internet
qa-stack.pl <1%

34 Student papers
Bahrain Polytechnic <1%

35 Student papers
Notre Dame of Marbel University <1%

36 Internet
www.loginradius.com <1%

37 Publication
R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P... <1%

38 Student papers
University of Wolverhampton <1%

39 Internet
learncodingusa.com <1%

40 Internet
www.rapidinnovation.io <1%

41 Student papers
University of Birmingham <1%

42 Internet
cebqrwf.1a-webverzeichnis.de <1%

43 Student papers
Asia Pacific University College of Technology and Innovation (UCTI) <1%

44 Student papers
Faculty of Organization and Informatics <1%

45 Student papers
National University of Ireland, Galway <1%

46 Student papers
University of Hertfordshire <1%

47 Internet
www.techygossips.com <1%

48 Internet
cybdom.tech <1%

49 Internet
github.com <1%

50 Student papers
universititeknologimara <1%

51 Internet
www.smoothterminal.com <1%

52 Student papers
Chester College of Higher Education <1%

53 Student papers
Häme University of Applied Sciences <1%

54 Student papers
University of Huddersfield <1%

55 Internet
coursetakers.com <1%

56 Student papers
University of East London <1%

57 Internet
es.stackoverflow.com <1%

58 Publication
Marco L. Napoli. "Beginning Flutter®", Wiley, 2019 <1%

59 Student papers
UCL <1%

60 Student papers
University of the West Indies - ROYTEC <1%

61 Internet
dokumen.tips <1%

62 Internet
moldstud.com <1%

63 Internet
www.irjet.net <1%

64 Internet
www.whattab.com <1%

65 Internet
conference.eurostarsoftwaretesting.com <1%

66 Internet
stackoverflow.com <1%

67 Internet
www.123articleonline.com <1%

68 Publication
Jasson Lwangisa Domition, Rogers Philip Bhalalusesa, Selemani Ismail. "Improve... <1%

69 Student papers
University of Bedfordshire <1%

70 Internet
myviewboard.com <1%

71 Internet
pusher.com <1%

72 Internet
www.bacancytechnology.com <1%

73 Internet
www.fastercapital.com <1%

74 Internet
www.jetir.org <1%

75 Student papers
Cankaya University <1%

76 Publication
Springer Compass International, 1991. <1%

77 Internet
blogs.emorphis.com <1%

78 Internet
eprints.utar.edu.my <1%

79 Internet
fdocuments.net <1%

80 Internet
mobidev.biz <1%

81	Internet	
qiita.com		<1%
82	Internet	
sode-edu.in		<1%
83	Internet	
su-plus.strathmore.edu		<1%
84	Internet	
telescope.ac		<1%
85	Internet	
tokyotechlab.com		<1%
86	Internet	
vinova.sg		<1%
87	Internet	
www.bsitsoftware.com		<1%
88	Internet	
www.geeksforgeeks.org		<1%
89	Internet	
www.kindacode.com		<1%
90	Publication	
Anurag Tiwari, Manuj Darbari. "Emerging Trends in Computer Science and Its Ap...		<1%
91	Publication	
Ton Duc Thang University		<1%