A PROJECT REPORT

ON

DRINK UP- WATER TRACKER APPLICATION

Submitted to

SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous)

Affiliated to JNTUA, Anantapur.

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by

SAMINENI HEMASREE	**19781A05E1**
SHAIK AFZAL BASHA	**19781A05E2**
SMD SAIFUDDIN	**19781A05F0**
SUNKARA SRILAKSMI	**19781A05G0**
UMAPATHI VAISHNAVI	**19785A05H0**

Under the esteemed guidance of

Mr O.G SURESH KUMAR M.TECH., (Ph.D.).,
Assistant Professor,
Department of computer Science & Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous)

Affiliated to JNTUA, Approved by AICTE, New Delhi, Anantapur

Accredited by NAAC 'A+', Bangalore & NBA, New Delhi

An ISO 9001:2000 Certified Institution

R.V.S NAGAR, CHITTOOR – 517 127. (A.P)

www.svcetedu.org

2022-2023

SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous)

Affiliated to JNTUA, Approved by AICTE, New Delhi, Anantapur

Accredited by NAAC ‘A+’, Bangalore & NBA, New Delhi

An ISO 9001:2000 Certified Institution

R.V.S NAGAR, CHITTOOR – 517 127. (A.P)

www.svcetedu.org

CERTIFICATE

This is to certify that the project entitled as “DRINK UP- WATER TRACKER APPLICATION”Is a bonafide work done and submitted by the following students

SAMINENI HEMASREE	**19781A05E1**
SHAIK AFZAL BASHA	**19781A05E2**
SMD SAIFUDDIN	**19781A05F0**
SUNKARA SRILAKSHMI	**19781A05G0**
UMAPATHI VAISHNAVI	**19781A05H0**

during the academic year 2022-2023 is submitted to the faculty of Computer Science and Engineering in partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING from Sri Venkateswara College of Engineering & Technology. Affiliated to Jawaharlal Nehru Technological University, Anantapur during the academic year 2022-2023.

Signature of Project Guide
Mr O.G SURESH KUMAR M.TECH., (Ph.D.
Assistant Professor,
Department of Computer Science and Engineering

Signature of the HOD
Dr.P.JYOTHEESWARI, M.TECH, Ph.D.,
Head of the Department,
Department of Computer Science and Engineering

Viva Voce conducted on:-_______

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

We feel great pleasure while submitting the major project report titled as “**DRINK UP-WATER TRACKER APPLICATION”**

Heartfelt thanks to Sri **Dr.R.Venkataswamy**, Chairman of Sri Venkateswara College of Engineering & Technology for providing education in his esteemed institution.

We wish to record our deep sense of gratitude and profound thanks to our beloved **Vice Chairman, Sri R.V.Srinivas** for providing education in his esteemed institution. We express our sincere thanks to **Dr.M.MOHAN BABU**, our beloved principal for providing facilities and encouragement throughout completing our project successfully.

With the deep sense of gratefulness, we acknowledge **Dr.P.Jyoteeswari, Head of the Department, Computer Science and Engineering**, for giving us inspiring guidance in completing our major project successfully.

We express our sincere thanks to our project Guide **Mr O.G Suresh Kumar, Assistant Professor**, for his keen interest, stimulating guidance, constant encouragement with our work during all stages, to bring this Major Project report into fruition.

We wish to convey my gratitude and sincere thanks to all members for their support and cooperation rendered for successful submission of this major project report.

Finally, we would like to express my sincere thanks to all teaching, non-teaching faculty members, our parents, friends and for all those who have supported us to complete the major project successfully

PROJECT ASSOCIATES

SAMINENI HEMASREE	**19781A05E1**
SHAIK AFZAL BASHA	**19781A05E2**
SMD SAIFUDDIN	**19781A05F0**
SUNKARA SRILAKSHMI	**19781A05G0**
UMAPATHI VAISHNAVI	**19781A05H0**

SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous)

Affiliated to JNTUA, Approved by AICTE, New Delhi, Anantapur

Accredited by NAAC 'A+', Bangalore & NBA, New Delhi

An ISO 9001:2000 Certified Institution

R.V.S NAGAR, CHITTOOR – 517 127. (A.P)

www.svcetedu.org

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We hereby declare that the project entitles **"DRINK UP- WATER TRACKER APPLICATION"** submitted by us under the guidance of **Mr O.G Suresh Kumar, Assistant Professor in the Department of Computer Science Engineering** in the fulfilment of award of degree of BACHELOR OF TECHNOLOGY Program at SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY, COMPUTER SCIENCE ENGINEERING Department.

Finally, we declare that the report has not been submitted to any other Institution / University for any other degree.

Signature of Students **Signature**

SAMINENI HEMASREE	**19781A05E1**
SHAIK AFZAL BASHA	**19781A05E2**
SMD SAIFUDDIN	**19781A05F0**
SUNKARA SRILAKSHMI	**19781A05G0**
UMAPATHI VAISHNAVI	**19781A05H0**

ABSTRACT

In this modern era due to the busy schedule of an individual water plays an important role in survival of the mankind.So for proper hydration of an individual,we made an app.The application determines the user's daily water requirements by using their weight and workout routine. The app lets you choose the preferred size for bottles and glasses. The software also provides statistical analysis of how much the user consumes over the course of a day, week, month, or year, as well as reminders for appointments. There are other widgets available to provide a fast overview of the process. Java and the SQLite database were used in the creation of the app. Water is a need for all living things to live. Without water, there is no life on Earth. mainly because water makes about 60% of our bodily weight. Water is used by all of the cells, tissues, and organs in our bodies.Drinking fluids and consuming meals that contain water are essential for rehydrating since our bodies lose water through breathing, sweating, and digesting. It should go without saying that your body needs enough water for almost all of its components.

Keywords: Water,application,reminder,hydration,modern era

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 : INTRODUCTION

1.1 INTRODUCTION

This app's main function is to remind the user to drink a specified amount of water every hour. The user's gender, age, weight, and level of physical activity are all collected by the water reminder app. It determines how much water the user must drink each day based on those inputs. After that has been computed, it sends reminders to the user every hour to take portions of the specified quantity. The medication reminder asks for the name of the medication, its start and finish dates, and the times it has to be taken each day. The app prompts the user to remember to take the prescribed medication at the appointed time when the user enters these data. This will assist individuals with the

This will make it easier for those with a propensity for forgetfulness and hectic schedules to remember to drink enough water each day and take their prescribed medications on schedule until the course is finished.Other components of this strategy involve fundamental activities, fresh food, and all-natural fixes for several typical issues. The exercises section includes written instructions and videos for some simple exercises that may be done to stay active and fit. There are around 2 items in the health feed area.Healthful hints are required. Also, the section on natural cures includes some all-natural treatments for the most typical health issues.

Google's Flutter is an open-source mobile application development framework. It is used to create natively compiled apps from a single codebase for mobile, web, and desktop platforms. Flutter is built on Google's Dart programming language and offers a quick development cycle with a comprehensive range of pre-made and customisable widgets.

Flutter is intended to enable developers rapidly and simply create high-quality, performant, and appealing mobile apps. The framework has several advantages over traditional mobile app development methodologies, including as shorter development times, hot reloading, and the ability to create apps for both the iOS and Android platforms from a single codebase. Since its first release in 2017, Flutter's popularity has grown significantly, with a huge and active developer community contributing to its development and use. Numerous prominent corporations, like Alibaba, Google, and Tencent, have already chosen Flutter for their mobile app development needs.One of Flutter's main advantages is its widget system, which allows developers to quickly construct bespoke and responsive user interfaces. The framework also provides a number of pre-built widgets, such as buttons, text fields, and photos, which makes it simple to rapidly construct beautiful and effective apps. Furthermore, Flutter's quick reload functionality allows developers to immediately see the impact of their code changes, making the development process more efficient and lowering the time it takes to bring an app to market.

To summarize, Flutter is a robust and versatile mobile app development framework that enables developers to rapidly and simply construct beautiful and high-quality apps for numerous platforms. Its user-friendly widget system, hot reload capability, and compatibility for both iOS and Android make it an excellent choice for businesses and developers trying to create useful and visually appealing mobile apps.

1.2 PROJECT DEFINITION

It has practically become difficult for an individual to keep track of his sleeping habits, medication, and water consumption due to the hectic schedules. The user needs a system that can remind them to take their medications on time and drink enough water based on specific requirements because all these aspects are crucial to maintaining good health. Also, it provides the user's sleeping habits, which may be used to monitor the number of hours and regularity of sleep cycles. In addition to these factors that directly impact health, providing some general health advice, natural treatments for widespread health issues that can be resolved without the use of medications, and some simple exercises to preserve the

Mobile or web-based solutions called "water tracker applications" are made to assist people keep track of how much water they use each day. Users of these programs may establish daily water intake goals, keep track of how much they consume during the day, and track their progress in achieving their hydration objectives. Several water tracking applications also offer tailored advice on how much water to drink depending on your age, weight, level of exercise, and climate. Users of water tracker programs may maintain optimal hydration levels, enhance their general health, and avoid diseases caused by dehydration. A software development project focused on producing an application, such as a mobile app, web app, or desktop app, is known as an application project. A typical application project includes several stages, including planning, design, development, testing, deployment, and maintenance.

During the planning phase, the project team determines the application's goals and needs and develops a project plan that includes the project's timetable, budget, and resources. Creating mockups and prototypes of the programme, as well as developing the user interface and user experience, are all part of the design process. The development phase is when the application is really coded and programmed. Working with programming languages, development frameworks, and libraries to construct the application's functionality is standard. Testing is a critical part of the project since it entails locating and correcting any flaws or mistakes in the programme.

After the programme is ready, it is deployed to the chosen platform, whether it is the app store for a mobile app, a web server for a web app, or a PC for a desktop software. Ultimately, on-going maintenance is necessary to keep the application current, secure, and functional. A single developer or a team of developers can work on an application project, which may require collaboration with other experts like as designers, project managers, and quality assurance specialists. An application project's success is often judged by its ability to achieve the project plan's goals and criteria, as well as user happiness and adoption rates.

1.3 STATEMENT OF PROBLEM

The difficulty that many individuals have in adequately hydrating themselves during the day is the issue that water tracker applications try to tackle. Many people don't get enough water to be hydrated, which can result in weariness, headaches, and other health issues including dehydration. To make sure they are getting enough water each day, many may find it difficult to monitor their water consumption. This issue may be resolved by using water tracker programs, which make it simpler for users to keep track of their water intake and meet their hydration objectives. By making sure they are well hydrated, people may increase their general health and well-being by using water tracker applications.

In order to elaborate on the issue statement for water tracker applications, it is critical to realise that water is an important part of the body. It is essential for several physiological functions, including as controlling body temperature, carrying nutrients, and removing pollutants. Lack of water consumption can cause dehydration, which can result in a variety of health concerns, from minor ones like headaches, lightheadedness, and dry skin to more serious ones like kidney damage, heat stroke, and even death. Although drinking water is important, many individuals find it difficult to do so on a daily basis. One or more of the reasons why people forget to remain hydrated is because of their hectic schedules, lack of knowledge, or forgetfulness. Also, people might not be aware of the recommended daily water consumption, which makes it challenging to monitor intake and make sure they are getting the hydration they need. These issues are resolved by water tracker programs by giving customers a quick way to keep track of their water intake. Users may maintain motivation to consume enough water throughout the day by establishing objectives and monitoring their progress. To further assist users in meeting their hydration goals, several water tracking programs include reminders and tailored recommendations. In general, water tracker apps offer answer to the issue of dehydration and the resulting health hazards. Although drinking water is important, many individuals find it difficult to do so on a daily basis. One or more of the reasons why people forget to remain hydrated is because of their hectic schedules, lack of knowledge, or forgetfulness

1.4 PROJECT OBJECTIVES

Applications that track water might have a variety of project goals depending on their particular characteristics and purposes. Nonetheless, the following are some common goals of water tracker projects:

Users are encouraged to drink more water throughout the day: One of the main goals of water tracker applications is to encourage users to do so. Users may form healthy habits and make sure they are getting enough water by establishing goals and monitoring their progress.

Giving individualized advice: Several water tracker programs provide individualised advice depending on variables including age, weight, amount of exercise, and environment. These suggestions make it simpler for users to stick to their hydration goals by assisting users in determining how much water they should be consuming each day.

Water tracker applications have the ability to send users reminders to drink water at regular times throughout the day. These reminders can help users keep well hydrated and can be tailored depending on user preferences.

Data insights: Water tracker applications can give users information about their water consumption patterns, such as how much water they normally drink at certain times of the day or on particular days of the week. Users can use this information to spot trends and modify their hydration habits.

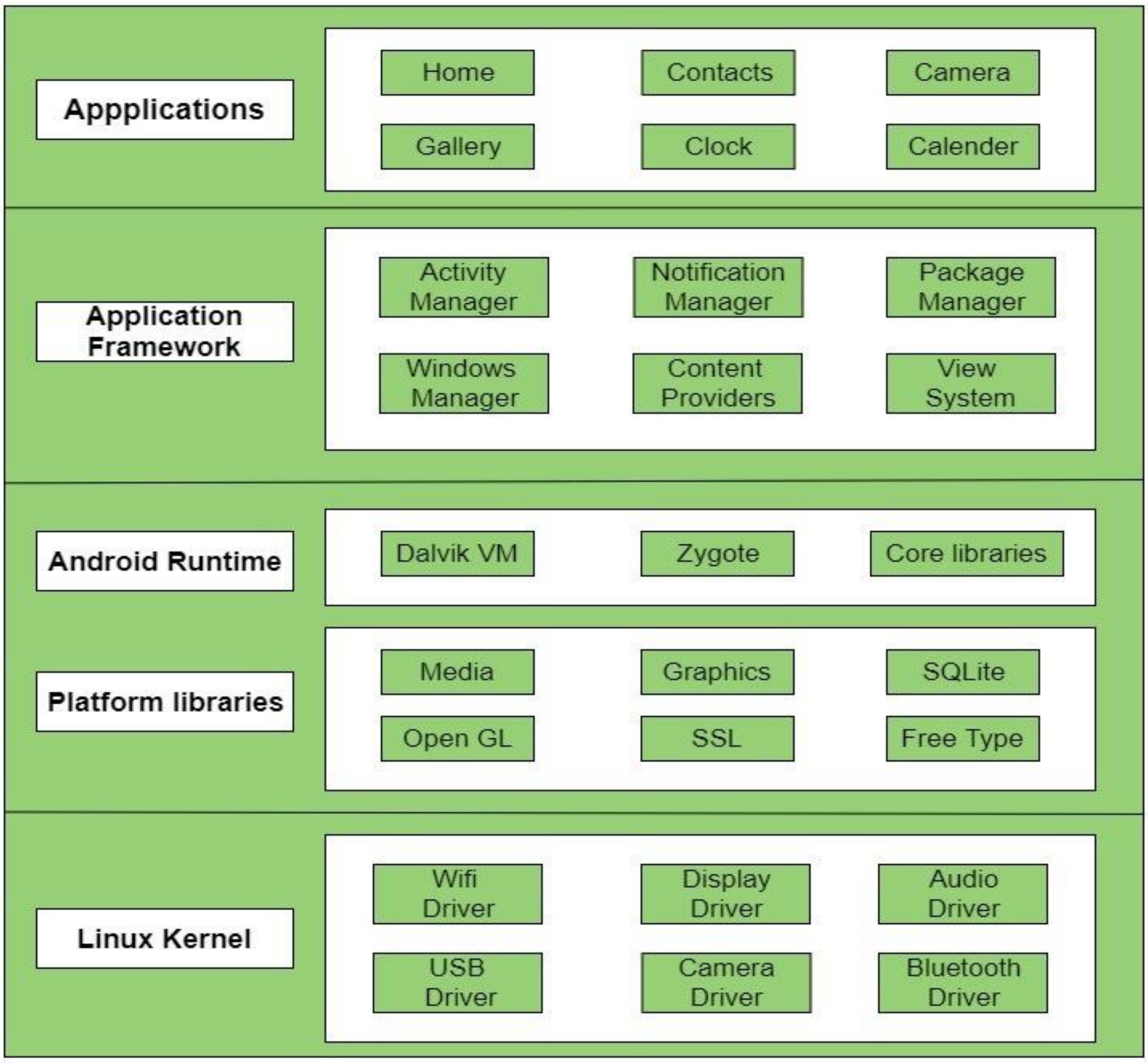
Enhancing general health: Water tracker apps can improve general health and wellbeing by assisting users in maintaining sufficient hydration. Adequate hydration can enhance mental performance, increase energy, promote good digestion, and lower the risk of diseases associated with dehydration.

1.4 ARCHITECTURE & COMPONENTS

The Android architecture has a variety of parts to fulfil the requirements of any android device. The open-source Linux kernel used by Android applications includes a variety of C/C++ libraries that are made available through application framework services.

Out of all the parts, the Linux Kernel gives smartphones their primary operating system features, while the Dalvik Virtual Machine (DVM) offers a foundation for running Android applications. Flutter is a cross-platform user interface toolkit that allows code reuse between operating systems such as iOS and Android while also letting apps to interact directly with underlying platform functions. The objective is for developers to be able to produce high-performance programmes that feel natural across platforms, embracing differences where they exist while sharing as much code as feasible.

Flutter apps run in a VM during development, allowing for stateful rapid reload of changes without requiring a full recompile. Flutter apps are compiled straight to machine code, whether Intel x64 or ARM instructions, or to JavaScript if they are intended for the web. The framework is free source and distributed under the permissive BSD licence, and it has a vibrant ecosystem of third-party packages that complement the basic functionality.



The top layer of the Android architecture is applications. Only this layer will be installed, neither the pre-installed programs like home, contacts, camera, gallery, etc., nor the third-party apps downloaded from the play store such chat applications, games, etc.

It utilises the classes and services made available by the application framework to function within the Android run time.

User profile: Users may enter personal data like their age, gender, weight, and degree of exercise in a user profile. This data is used to provide individualised water consumption recommendations and monitor progress towards hydration objectives.

Water intake tracking: A water tracker app enables users to keep track of how much water they consume each day. Users may keep track of how much water they drink each day as well as the time of day.

Hydration objectives: With the app, users may define hydration objectives that are tailored to their own requirements and preferences. Users may establish objectives for their water usage on a daily, weekly, or monthly basis.Based on variables like age, weight, and degree of exercise, the program may offer tailored recommendations for water consumption. These suggestions help users in determining

Flutter: Flutter may be used in conjunction with Android Studio as an IDE to create Flutter apps for Android. Flutter's integration with Android Studio offers a rich development environment for creating Flutter apps that can be produced and launched on Android devices.To begin building Flutter apps with Android Studio, you must first install the Flutter SDK and the Dart plugin. When you've installed them, you can launch Android Studio and start a new Flutter project, which will generate a basic Flutter app with sample code and a default layout.For building Flutter apps, Android Studio includes a comprehensive text editor with code highlighting, debugging tools, and an emulator for testing the app. One advantage of utilizing Android Studio for Flutter development is that it enables Android developers who are already familiar with the IDE with a familiar development environment. Android Studio also integrates seamlessly with the Android platform, making it simple to create and test Flutter apps on Android devices.

To summarise, Flutter can be combined with Android Studio to provide a strong development environment for creating Android Flutter apps. The integration gives developers access to a variety of tools and capabilities for designing and testing Flutter apps, and the Android Studio IDE's familiarity makes it an excellent alternative for developers who are already familiar with the Android platform.

1.5 PROJECT SCOPE

Applications for water trackers can have a wide range of uses since they might be beneficial to a variety of people, including those who are health-conscious, athletes, or those who have medical ailments that require them to be hydrated. Applications for water trackers may be categorised into many important categories.Monitoring one's state of hydration and keeping track of daily water intake is possible with the use of water tracker applications. Everyone who wants to make sure they get enough water during the day can benefit from this function

Monitoring the hydration of athletes is important since they may lose a lot of fluids while exercising vigorously. Based on their activity level and other variables, water tracker applications may provide athletes individualised recommendations for water intake.

Medical hydration tracking: Those who have medical illnesses like diabetes or renal disease, which make it important for them to be hydrated, might benefit from utilising water tracker programs to keep track of their water consumption and make sure they are getting the hydration they need.Corporate wellness initiatives To encourage good hydration practises among their staff, businesses might utilise water tracker applications as part of their wellness initiatives.Education and research: To better understand the connection between hydration and health consequences, water tracker applications may be utilised in both of these fields.Overall, there are a wide range of water tracker apps that may be utilised in different contexts and for different goals. Applications that measure your water consumption are primarily designed to encourage good hydration practises and enhance your overall health and wellbeing.

1.7 APPLICATION ARCHITECTURE

For the creation of mobile applications for the iOS and Android platforms, Flutter is a well-liked framework. A tiered approach is used in the architecture of a Flutter application, with each layer in charge of a certain set of duties. The foundational elements of the Flutter architecture are as follows:

Presentation layer: The presentation layer is in charge of showing the application's user interface. It is made up of widgets that stand in for the app's many visual components, including buttons,text.

Logic layer: The logic layer is in charge of putting the application's business logic into practise. It comprises the fundamental features of the program, such data processing, validation, and modification. To handle the application's state and data, it frequently employs the state management

Data layer: The data layer is in charge of managing data storage and retrieval for the application. Data sources like databases, APIs, or local storage are frequently included.Implementing network-related features such as API calls, data synchronization, and caching falls under the purview.

Platform layer: In order to communicate with the underlying operating system and platform-specific characteristics like sensors, cameras, and locations, the platform layer must interface with them.

Plugins: To increase an application's capabilities, extra packages known as plugins can be installed. They may be utilised to include features like social network integration, push alerts, and authentication.A Flutter application's architecture is intended to be modular, adaptable, and scalable overall. The layered method makes it simpler to maintain and upgrade the program over time since it enables developers to separate different components of the application and work on them individually.

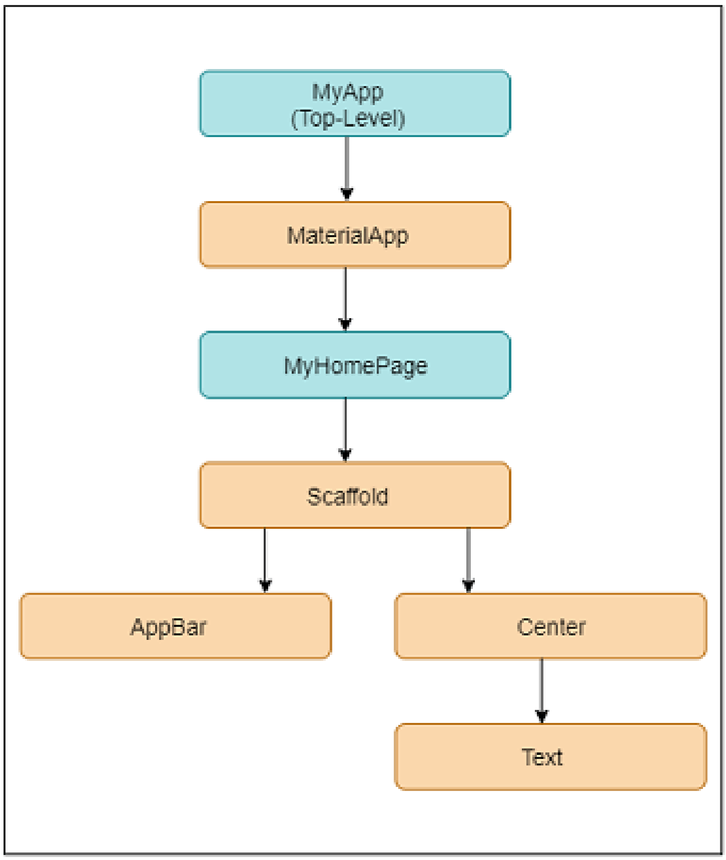


Figure 1.1 Architecture of flutter applications.

Overall, the architecture of a Flutter application is designed to be modular, flexible, and scalable. The layered approach allows developers to isolate different aspects of the application and work on them independently, making it easier to maintain and update the application over time.

1.8 WATER TRACKER ALGORITHMS

Depending on the particular requirements of the application, several algorithms may be utilised in the creation of an app. Nonetheless, some typical algorithms employed in the creation of apps include:

With sorting algorithms, data may be arranged in an alphabetical or numerical order, for example. Quicksort, merge sort, and bubble sort are a few common sorting algorithms.

Search algorithms: To find specific data inside a dataset, search algorithms are utilised. Hash tables, binary search, and linear search are examples of common search methods.

Algorithms for machine learning: Algorithms for machine learning are used to create software that can adapt and learn from user behavior. Decision trees, neural networks, and linear regression are examples of common machine learning techniques.

Algorithms for data compression: Algorithms for data compression are used to compress huge data files and minimize their size for transmission and storage. Gzip, zip, and RAR are a few common data compression techniques.

Algorithms for manipulating and enhancing photos and movies include image and video processing algorithms. Convolutional neural networks, edge detection, and color correction are common image processing techniques.

Language Detection algorithms: This method determines what human language the text input is written in after receiving the text as input, as suggested by its name.

Some mobile applications that are made to do translations for many languages frequently employ this algorithm.Moreover, it is utilized in voice recognition software designed to work with mobile applications like Google Talk, Siri, Cortana, etc.

1.9 SUMMARY

With more individuals understanding the value of being hydrated, water tracking applications have grown in popularity in recent years. With the ultimate objective of encouraging improved health and wellbeing, these smartphone applications are made to assist users in tracking and monitoring their regular water intake. We shall go deeper into the characteristics and advantages of water tracker apps in this overview.

The capability to define unique objectives is one of the main benefits of water tracking software. Users may input details like age, gender, weight, and degree of exercise to find out how much water they ought to drink each day. Some applications even alter the user's water intake objectives in accordance with the user's location and the surrounding weather. Users are more likely to stay motivated and monitor their progress towards improved hydration if they establish a daily water consumption target. Also, there are several ways to record and monitor water intake with water tracker applications. The amount of water that users have drunk during the day may be readily entered on many apps' straightforward input screens. Others could let customers scan a water bottle's barcode to automatically record how much water they've drank. Several applications also provide notifications and reminders to assist users in maintaining their water intake targets.

Several applications measure the amount of water consumed while also offering insightful comments. Users may examine daily, weekly, or monthly reports that detail their water intake and status in relation to their objectives. Some applications even offer suggestions for how to increase hydration depending on things like climate, diet.

The ability of water tracker apps to link with other health and fitness apps is another important feature. For a more complete picture of their overall health and wellness, users may link their water tracker app to their activity tracker or calorie tracking app.Overall, water tracker apps provide users with a straightforward and practical approach to keep track of and manage their daily water intake. Users may maintain motivation and make wiser hydration choices by creating individual objectives, monitoring progress, and receiving feedback. With all of the features and advantages that water tracker applications provide

CHAPTER 2 : LITERATURE REVIEW

2.1 INTRODUCTION

Being hydrated lowers the risk of urinary tract infections, chronic kidney disease, and kidney stone disease (KSD) (UTIs). The use of mobile applications (apps) that measure hydration is growing, enabling users to keep track of intake while simultaneously considering the symptoms and indicators of dehydration. The use of water applications in the treatment and/or prevention of urological illness was the focus of our investigation. Apps for mobile devices have long been used to gauge fluid consumption. Nevertheless, they don't use additional metrics like degree of activity, urine production, or color and offer little information about the significance of staying hydrated. The creators of fitness and hydration applications should make them more thorough and educational given their rising popularity in our everyday lives.

Applications that measure your water consumption are becoming more and more common tools for monitoring and optimising it. These applications enable users to monitor and log their daily water intake, establish hydration objectives, and get alerts to drink more water. We will examine the efficacy and usefulness of water tracker applications in this review of the literature. The usefulness of water tracking applications in boosting water intake among college students was examined in a research by Lefebvre et al. (2019). According to the study, utilising a water tracker app raised participants' daily water intake by 8.2 ounces on average. Users of the Water Minder app may establish their own individual hydration goals, receive alerts to drink water, and see graphic representations of their progress.

Hales et al. (2019) looked at the functioning and usefulness of water tracker apps in another research. The study discovered that many water tracking applications lacked fundamental capabilities, such the capability for users to note the dimensions of their water bottle or cup. The survey also discovered that several apps required complex navigation and lacked clear usage instructions. Tieu et al. (2019) examined the features and efficiency of several water tracker apps in a systematic evaluation of hydration applications. The analysis discovered that many water tracker applications lacked scientific proof of their ability to improve water consumption. The study did, however, point out a number of useful features, including individualised hydration targets, water-drinking reminders, and progress monitoring.Water tracker apps have grown in popularity in recent years as individuals strive to increase their hydration levels and live a healthier lifestyle.

A comprehensive assessment of smartphone applications for encouraging physical activity and good eating habits: Lee and colleagues (2016) investigated the impact of smartphone applications in encouraging physical activity and good eating habits. The study discovered that water tracker applications were efficient in raising users' water intake.A randomised controlled study of a smartphone-based app to promote physical activity in obese adults: Fukuoka and colleagues (2015) investigated the efficacy of a smartphone-based software for improving physical activity among overweight people. The app was shown to be beneficial in raising water intake and fostering healthy habits in the study.

A smartphone app for tracking fluid consumption and body weight was evaluated: Shin and colleagues (2016) investigated the efficacy of a smartphone app for tracking fluid consumption and body weight. The app was proven to be beneficial in increasing hydration status and lowering body weight among users, according to the research.Water intake monitoring and management using a mobile application: Joshi and colleagues (2017) conducted a research to assess the effectiveness of a mobile application for monitoring and managing water intake. The app was shown to be beneficial in increasing water consumption and improving hydration status among users, according to the research.

A randomized controlled study of a smartphone-based app to encourage physical activity and good eating: Direito and colleagues (2014) investigated the efficacy of a smartphone-based app for encouraging physical activity and good eating habits. The app was shown to be beneficial in promoting healthy habits, such as increased water intake, according to the research.

These research conclude that water tracker applications are beneficial in boosting water intake, improving hydration status, and encouraging healthy behaviours. Those looking to enhance their general health and well-being may find these applications useful.

2.2 SIMILAR SYSTEMS COMPARISON

2.2.1 WATER MINDER

Available for iOS and Android smartphones, Water Minder is a well-known water tracking software. The app lets you keep track of how much water you consume each day and prompts you to hydrate yourself by drinking water often. The Water Minder app's main features include the following:

- Individualized water intake targets: Using information about your weight, gender, level of exercise, and other characteristics, Water Minder determines your daily water consumption target.
- Reminders that can be customized: You may program reminders for various times during the day to help you remember to drink water. The repetition rate and the time of these reminders are also adjustable.

- Simple tracking: Water Minder makes it simple to monitor your water intake by enabling you to quickly record your water usage.
- Apple Health Sync: You can simply track your progress over time because the app can integrate your water consumption statistics with Apple Health.
- Achievements and prizes: Water Minder has accomplishments and prizes to encourage you to stick to your hydration objectives.

Overall, Water Minder is an easy-to-use tool that can assist you in maintaining your water consumption goals and forming wholesome hydration habits.

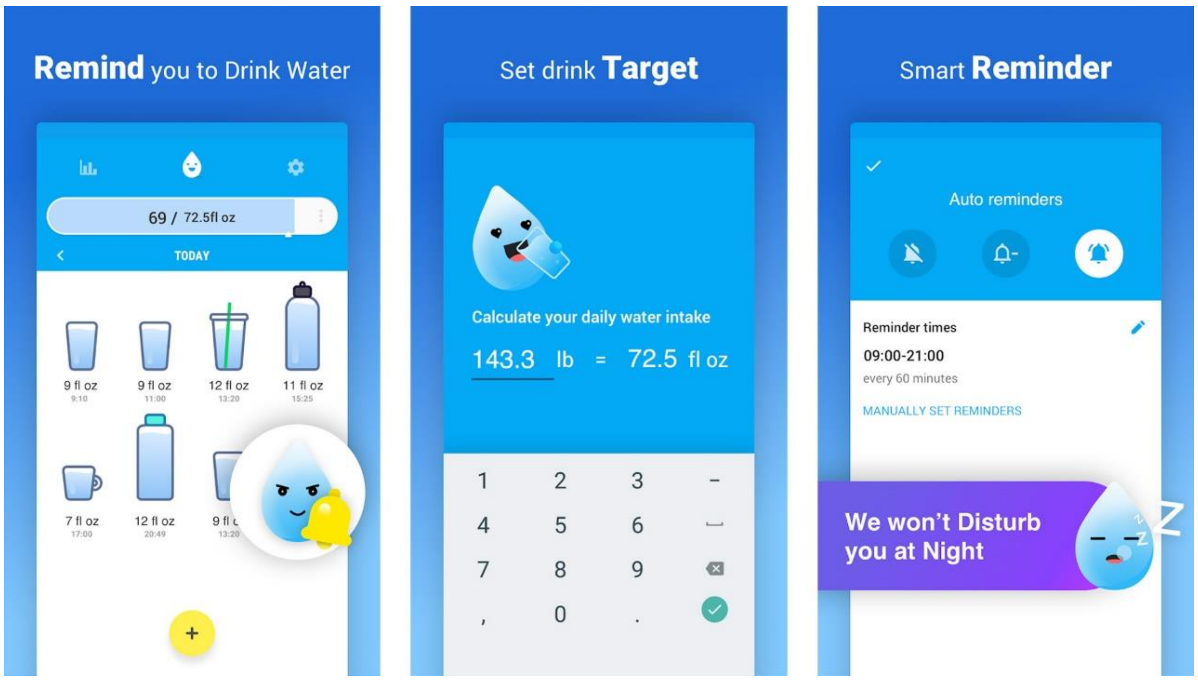


Table 2-1: Similar tools comparison

2.2.2 HYDRO COACH

A smartphone app called HydroCoach was created to assist users in monitoring and increasing their water consumption. Based on their body weight and degree of exercise, users of the app may establish daily water consumption goals and monitor their progress throughout the day.HydroCoach offers feedback on users' progress towards their water intake goals in addition to providing reminders and notifications to encourage users to consistently drink water. Also, users may register other drinks they consume with the app, such coffee or soda, and the program will change their water intake targets appropriately.Overall, HydroCoach is a helpful tool for anybody trying to maintain a healthy lifestyle and improve their hydration habits.

A number of tools are available in HydroCoach to assist users in monitoring and increasing their water consumption. These qualities include, among others:

- Individualized water intake targets: HydroCoach determines a suggested daily water consumption target based on the user's body weight, amount of exercise, and weather. Users can change their objective as necessary.
- Tracking of water intake: Users may record their daily water consumption and watch their progress towards their daily target in real-time.
- Reminders and notifications are sent by HydroCoach to remind users to routinely drink water throughout the day.
- Beverage tracking: Users may record other drinks they consume in addition to water, and the app will change their water intake target appropriately.

2.2.3 AQUALERT

A smartphone app called Aqualert was created to assist users in monitoring and increasing their water consumption. Based on their body weight and amount of exercise, users may set a daily water consumption target on the app, which then monitors their progress throughout the day.Aqualert tracks a user's progress towards their water intake target using a color-coded system. Also, the app notifies users and sends reminders to remind them to frequently drink water throughout the day.A user's water intake goals may be customised to suit their tastes and needs.

Tracking of water intake: Users may record their daily water consumption and watch their progress towards their daily target in real-time.

Aqualert's reminders and notifications are sent to users to remind them to routinely drink water throughout the day

2.3 RECOMMENDATIONS

Anybody trying to enhance their hydration practises and uphold a healthy lifestyle will find water trackers to be helpful tools. These applications support users in setting and monitoring their daily water intake goals, sending notifications and reminders to promote regular hydration, and providing insights and analysis to assist users comprehend their water consumption patterns and make necessary modifications.

Overall, using a water tracker can help you keep accountable and inspired while working towards your hydration goals. They are simple to use and can assist you in developing good habits that can improve your general health and wellness. Using a water tracker in your routine may be as easy as selecting a straightforward app with basic monitoring functions or as complex as selecting an elaborate software with configurable choices.

Use a water tracker app to help you monitor and boost your water consumption. There are several water tracker applications available, so select one that best suits your requirements and interests.

Establish attainable water consumption targets. While it is critical to remain hydrated, it is equally critical to avoid dehydration. Drink enough water to fulfil your body's demands, but don't force yourself to drink more.

To keep on track, use reminders and alarms. You can create reminders and alarms in many water tracker applications to remind you to drink water throughout the day. Utilize these tools to assist you in developing a consistent hydration practice.

.

CHAPTER 3 : SYSTEM ANALYSIS

3.1 INTRODUCTION

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

System analysis is an essential component of software development and the design of information systems. It is the process of researching and analyzing current systems, identifying their strengths and limitations, and producing ideas for changes or whole new systems. The purpose of system analysis is to comprehend a system's underlying structure and operation so that it may be optimized for improved performance, efficiency, and usefulness.

The system analysis process begins with an evaluation of the present system. This include collecting information on how the system operates, who uses it, and what its aims and objectives are. The data is then evaluated to see which aspects of the system may be improved. This might include detecting bottlenecks, inefficiencies, or locations where user demands are not being addressed. After identifying the issues, the analyst offers recommendations for how to fix them.

The system analysis process continues with the creation of a comprehensive strategy for executing the proposed improvements. This might entail creating new software applications or altering current ones. It may also entail creating new procedures or workflows to better fulfil the system's users' demands.

3.2 STAKEHOLDERS

Stakeholders must be able to make decisions, but need not be human: "An stakeholder might be a person, a company or organization, a computer program, or a computer system — hardware, software, or both." Actors are always stakeholders, but not all stakeholders are actors. The following Table 3.1: Stakeholders and its Description. You may better understand how a blockchain functions and its development potential by being aware of how each of these stakeholders fits into the ecosystem around a blockchain. A blockchain is more likely to see healthy growth the more appealing it is to each of the aforementioned stakeholders.

Stakeholder	**Description**
User	That have major controlling of the whole systems process and able to access or use all modules of the application.
Program	The Application itself which can perform the user operation in main process like storing and processing data by using water tracker application

Table 3-1: Stakeholders

3.3 NON-FUNCTIONAL REQUIREMENTS

- For user interfaces we take in consideration that they should has a standard look and being user friendly at the same time to make sure that users' attention will not be distracted and interface to provide more flexibility and scalability.
- The program will be in the English language
- The program must be fast in processing
- The program must be able to store the data securely and record them in a ledger.
- All function must be works well then system will be a high quality
- Performance: The app must operate well even when under a lot of user pressure and respond fast to user actions.
- Scalability: As the user base expands, the app must be able to manage progressively more traffic and data.
- Protection of user data against unauthorized access, hacking, and other security lapses requires the program to be safe.
- Usability: The program must have a straightforward and intuitive user interface that is simple to use.
- Reliability: The program must be available, dependable, and error-free with little to no

downtime.

- The program must work with a variety of platforms, operating systems, and gadgets.
- Accessibility: Users with disabilities, such as those who have visual or auditory problems, must be able to utilise the program.
- Maintainability: The program must be simple to update and maintain and have good documentation.

These non-functional criteria are essential to an application's success and must be taken into account during the design and development phases to make sure the program satisfies user demands and runs efficiently.

3.4 SEMI-FORMAL REQUIREMENTS

3.4.1 General Use Case Diagram

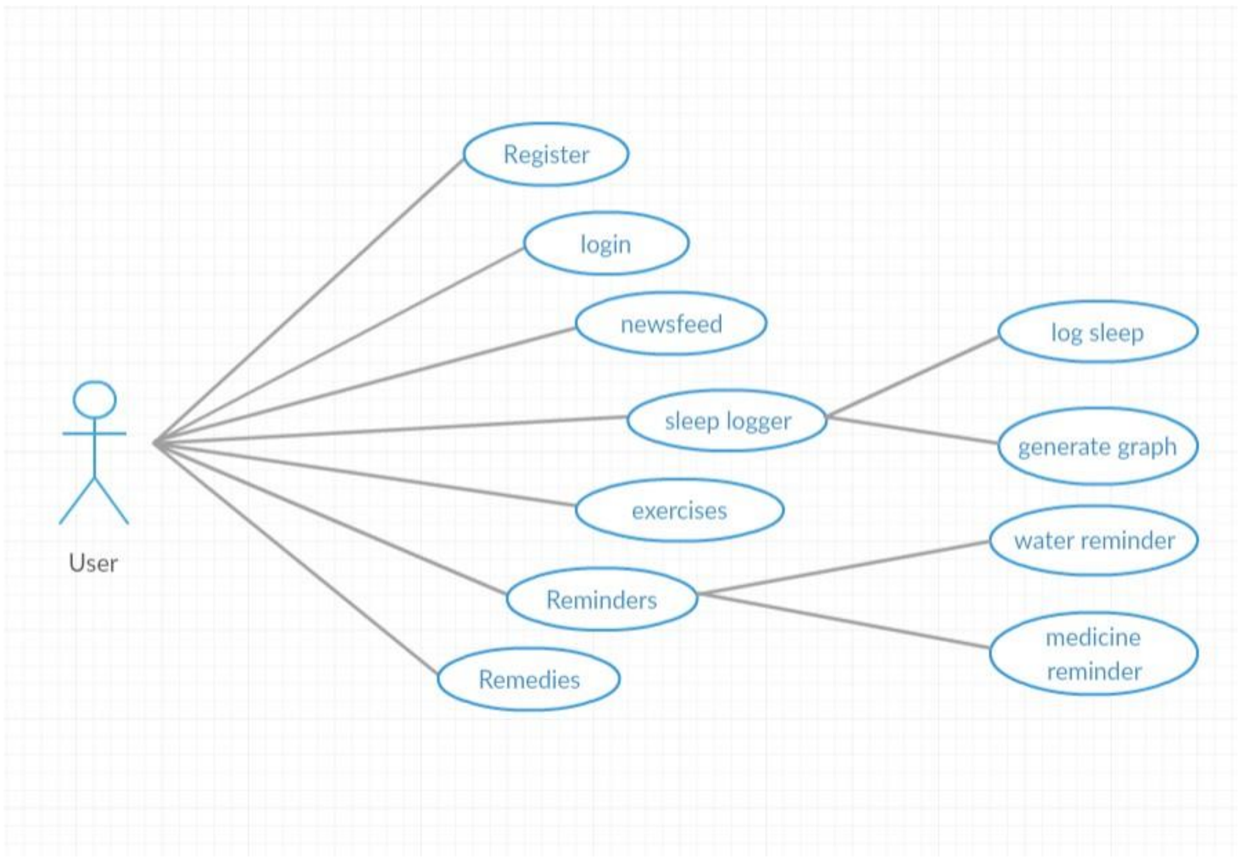


Figure 3-1: Use case diagram

Actors

An actor in the Unified Modeling Language (UML) "specifies a role played by a user or any other system that interacts with the subject." "An Actor models a type of roleplayed by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject."

There are two actors in our application:

1.User

2.Program

A use case is a set of scenarios that describing an interaction between a user and asystem. A use case diagram displays the relationship among actors and use cases.

3.4.2 Hardware and Software Requirements

3.4.2.1 System Requirements

Operating System: Windows 7 or higher, Mac OS

RAM: 3GB or higher

Java version: JDK 8

Disk space for Android Studio: 500 MB

Space for Android SDK: 1.5 GB or higher

Processor speed: 2.3 GHz

RAM: 8GB

3.4.2.2 Hardware Requirements for running the application

RAM: 512 MB

Disk space: 250MB or more

Android phone with API level 16 or higher (Jelly bean)

Processor: i3 or higher

3.4.2.3 Software Requirements

IDE: Android Studio

Technologies: Java, XML, Android

Database: SQLite

Debugger: Emulator, Android mobile device

3.5.1 LOGIN PANEL

The logo and app title are displayed on the splash screen when the program first launches. The user must provide both a valid username and password on the login screen in order to properly log in. This is the following screen that is visible. Also, Facebook, which is linked into the program via the Facebook SDK, allows users to log in. Information regarding the user's Facebook id is required when they attempt to check in using Facebook. A generated access token can be utilised by the user to log in. As soon as the user submits information, the input fields are verified. The user must first register if they do not already have any login information. Details are displayed when the user clicks the register button. A home screen appears following a successful login by the user. If the user goes down the page from this home screen, he can read some health advice. The home screen's content is static. There is a menu at the bottom of the screen through which the user may access other displays. Much of this module's implementation is done in the newsfeed fragment and newsfeed adapter.

Many software programs, particularly those that require user identification or access control, have a login panel. A login panel's objective is to offer users with a safe manner to access the application's features and functions.

The login panel is often composed of a form in which the user enters their username and password. These credentials are then validated against a database of authorised users to verify whether the user has permission to access the application. The user gets provided access to the application's features and capabilities if the credentials are legitimate. If the credentials are invalid, the user is usually refused access and may be requested to re-enter their credentials or contact support.

The login panel may be used to track user activity and analyse usage patterns in addition to providing a secure means to access the application. This can assist enterprises in identifying possible security issues or places where the programme may require improvement to better fulfil the demands of its users.

A lot of aspects must be considered when developing a login panel to guarantee that it is functional and user-friendly. Strong password policies, the implementation of two-factor authentication or other security measures, and the provision of clear and concise error messages to assist users in troubleshooting login difficulties are examples of these.

3.5.2 COMMUNICATION TOOLS

The official Integrated Development Environment (IDE) for developing Android applications is called Android Studio. Several capabilities offered by Android Studio increase our ability to produce high-quality Android apps.During the Google I/O conference on May 16, 2013, Android Studio was introduced as the official IDE for creating Android applications. In May 2013, it began offering an early access sample of version 0.1. Beginning with version 1.0, the first stable constructed version was published in December 2014.Kotlin has become Google's preferred language for creating Android applications since May 7th. Alongside this, Android Studio also supports additional programming languages.

Integrated Development Environments (IDEs): IDEs are software programs that offer developers a complete environment for writing, testing, and debugging code. For the creation of Android, iOS, and Windows applications, popular IDEs include Android Studio, Xcode, and Visual Studio.

Software Development Kits (SDKs): SDKs are sets of programming tools that let programs create apps for particular hardware or operating systems. iOS SDK and Android SDK are two examples.

Version Control Systems (VCS): During the development process, source code is managed and tracked using VCS technologies. Git, SVN, and Mercurial are a few common VCS tools.

3.5.3 ALGORITHMS

A set of detailed instructions called an algorithm is used to accomplish a certain activity or solve a problem. Algorithms are used in computing to perform intricate computations, process data, and automate processes. They are simply a series of deductive and computational processes intended to accomplish a certain objective. Algorithms may be represented in a variety of ways and take on a wide range of shapes. These could be shown in a programming language as actual code, pseudocode, or flowcharts. Depending on the function they are intended to carry out, they might be straightforward or complicated.

3.5.3.1 HASHING ALGORITHM

The amount of data on the internet is growing exponentially every day, making it difficult to store it all effectively. Even though this amount of data may not be large in day-to-day programming, it must be conveniently saved, retrieved, and processed. The Array data structure is a popular choice for this application of data structures. Now the issue is raised: why create a new data structure if Array already existed? The term "efficiency" has the solution to this. While searching in an array requires at least O(log n) time, storing in one just requires O(1) time. Although it seems like a modest amount, this time can be quite problematic for huge data sets.

Key: A key is any text or number that is provided as input into a hash function, a method for determining an item's index or placement in a data structure.

Hash Function: The hash function takes a key as input and outputs the index of a hash table's entry. The index is sometimes referred to as a hash index.

Hash Table: A hash table is a type of data structure that uses a unique function called a hash function to map keys to values. Hash holds the data in an array in an associative fashion, giving each data value a distinct index.

3.5.3.2 FLUTTER ALGORITHMS

Sorting algorithms are used to arrange data in an array or list in a certain manner. The sorting techniques bubble sort, insertion sort, merge sort, quicksort, and selection sort are all well-liked. Flutter may utilise these methods to arrange and show data in a certain order.

Searching algorithms are used to locate a particular object or element inside a data set. Binary search and linear search are two types of search algorithms. Flutter can search and filter data using these methods.

Data security is achieved through encryption methods, which turn data into a code that is unreadable by unauthorized users. AES (Advanced Encryption Standard) and RSA are two common encryption techniques (Rivest–Shamir–Adleman).

The Dart programming language is used by the framework for creating mobile apps called Flutter. Despite the fact that Flutter does not come with any pre-built algorithms, developers may incorporate a wide range of algorithms into Flutter apps to create different functionality.

SUMMARY

In summary, water tracker apps may be helpful tools for people to keep hydrated and enhance their general health and wellness. These applications include a variety of capabilities, including the ability to establish daily water consumption targets, monitor intake, and send out reminders to drink more water. Several water tracking applications also offer tailored advice based on details like age, weight, and degree of activity.

It's crucial to take into account aspects like user friendliness, monitoring precision, and the availability of extra features like interaction with fitness trackers when selecting a water tracker app. In addition, users should always get advice from a medical practitioner to establish the ideal daily water consumption for their particular requirements.

CHAPTER 4 : SYSTEM DESIGN

4.1 INTRODUCTION

Software design is a process of problem-solving and planning for a software solution. After the purpose and specifications of software is determined, software developers will design or employ designers to develop a plan for a solution. It includes construction component and algorithm implementation issues which shown in as thearchitectural view. During this chapter we will introduce some principles that areconsidered through the Software design.

Identifying the application's aims and objectives, as well as the requirements for its features and functioning, is the first stage in system design. To obtain a deeper knowledge of user requirements and expectations, perform user research, stakeholder interviews, and other sorts of analysis.

Developing a high-level design: Developers and designers build a high-level design of the application's architecture, components, and functionality based on the requirements. This design may incorporate diagrams, flowcharts, and other visual representations to assist stakeholders in understanding the structure and flow of the system.

Creating comprehensive specifications: When the high-level design is complete, developers and designers collaborate to establish detailed specifications for each system component. This might entail writing code, developing user interfaces, and constructing databases and other data storage systems.

Testing and iteration: Developers and designers test the application throughout the system design process to ensure that it satisfies the requirements and performs as planned. Any faults or bugs that are discovered are resolved iteratively through refinement and testing.

Deployment and upkeep: Once the system design is complete and the programme has been thoroughly tested, it may be made available to users. The system design process, however, does not end there. Continuous maintenance and upgrades are necessary to keep the programme secure.

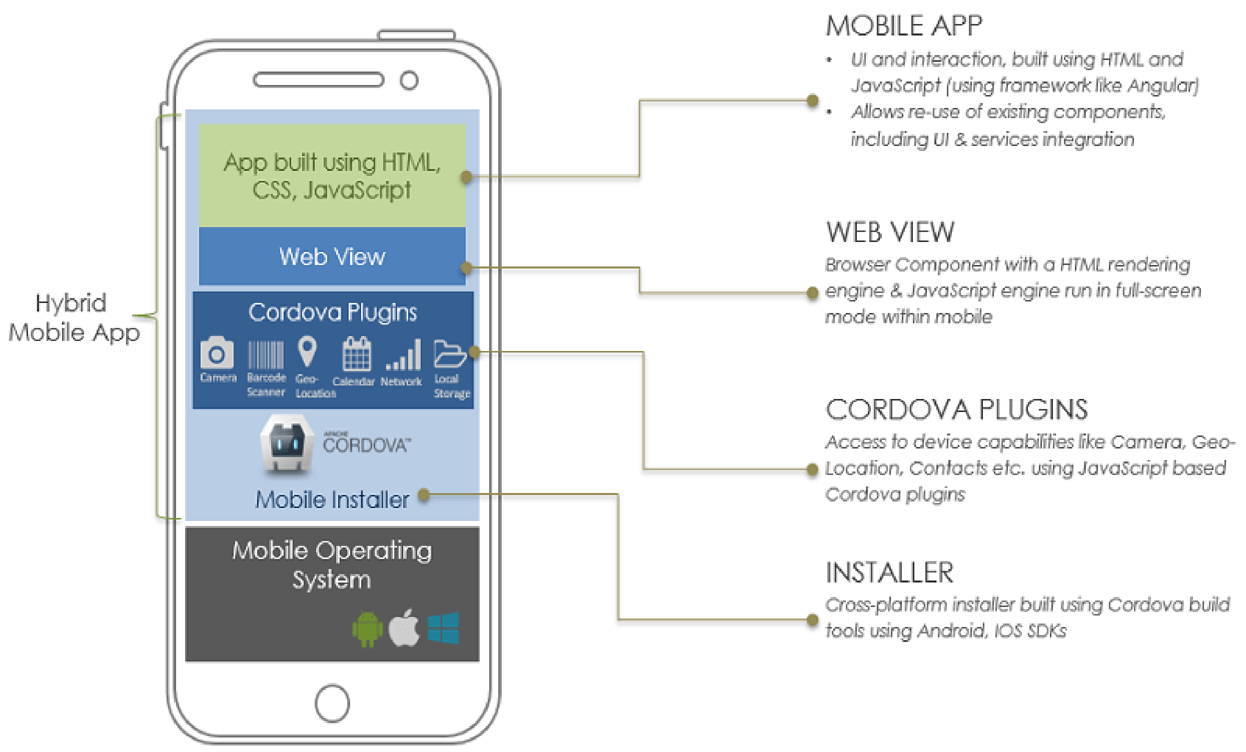
Overall, an application's system design is a vital phase in the software development process. Developers and designers may guarantee that the application satisfies user needs and expectations while also being safe, dependable, and scalable by producing a thorough blueprint of the program's architecture, components, and functionality.

4.2 STRUCTURAL VIEW

Structural diagrams are used to describe the relationships between classes. Structuralview can be described using class diagram.The arrangement and connections between the many components and modules that make up an application are referred to as the structural view of the application. It concentrates on the system's architecture, design, and division into more manageable pieces.

The fundamental elements of an application, such as classes, modules, and functions, are called components.

- Relationships: The ties and interdependencies among the various parts, including those resulting from inheritance, composition, and aggregation.
- Layers: The logical divisions of components into distinct levels, such as the display layer, the business logic layer, and the data access layer.
- Interfaces: The agreements that specify how various parts and levels interact with one another.


Hybrid Mobile App
App built using HTML, CSS, JavaScript
Web View
Cordova Plugins
Camera
Barcode Scanner
Geo-Location
Calendar
Network
Local Storage
APACHE CORDOVA
Mobile Installer
Mobile Operating System
MOBILE APP
• UI and interaction, built using HTML and JavaScript (using framework like Angular)
• Allows re-use of existing components, including UI & services integration
WEB VIEW
Browser Component with a HTML rendering engine & JavaScript engine run in full-screen mode within mobile
CORDOVA PLUGINS
Access to device capabilities like Camera, Geo-Location, Contacts etc. using JavaScript based Cordova plugins
INSTALLER
Cross-platform installer built using Cordova build tools using Android, IOS SDKs

4.3 Class Diagram

A class diagram is a picture for describing generic descriptions of possible systems.Class diagrams and collaboration diagrams are alternate representations of object models. Class diagrams contain classes and object diagrams contain objects, but it is possible to mixclasses and objects when dealing with various kinds of metadata, so the separation is not rigid. A class diagram is a sort of UML diagram that is used to describe the classes, connections, and interfaces in an object-oriented software system. It is a graphical depiction of a system's structure that is often used during the design phase of software development.

Classes are blueprints or templates that specify the characteristics and behaviour of a certain object. Each class is represented by a rectangle with its name at the top and its properties and methods shown below.

Relationships: Class relationships can be expressed in a variety of ways, including inheritance, aggregation, and association.

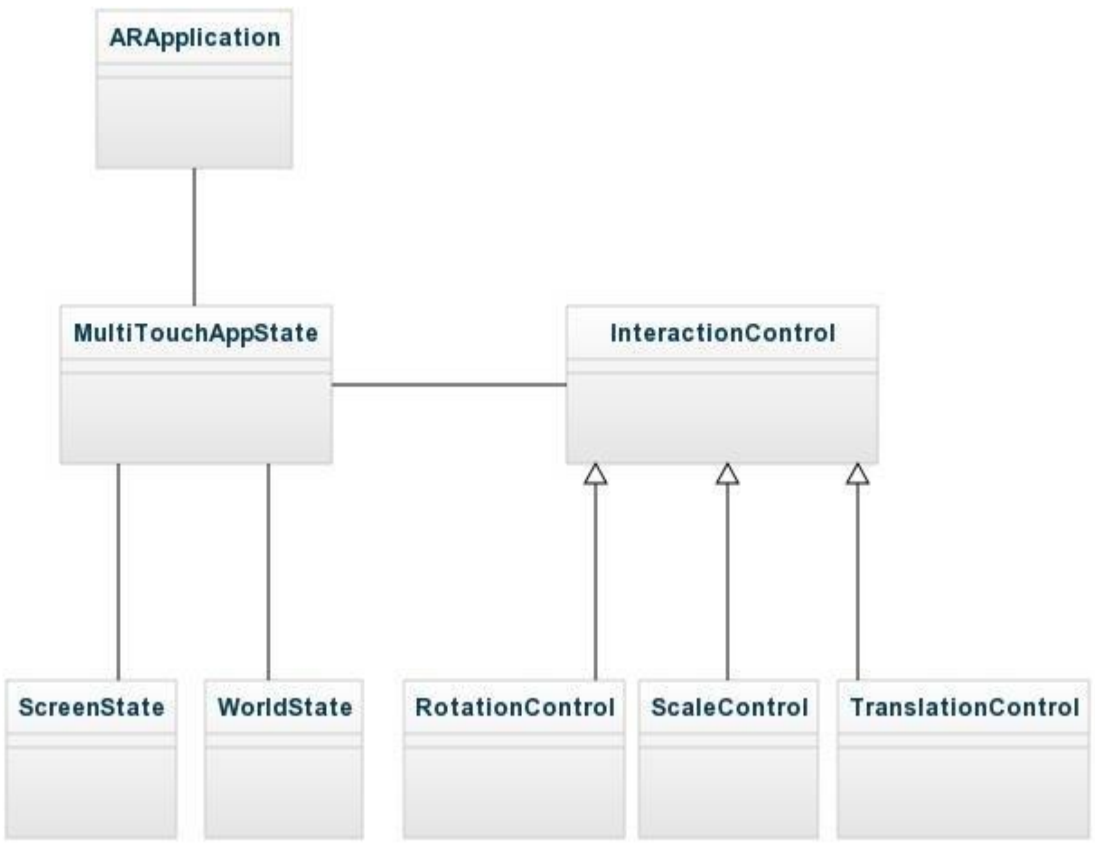
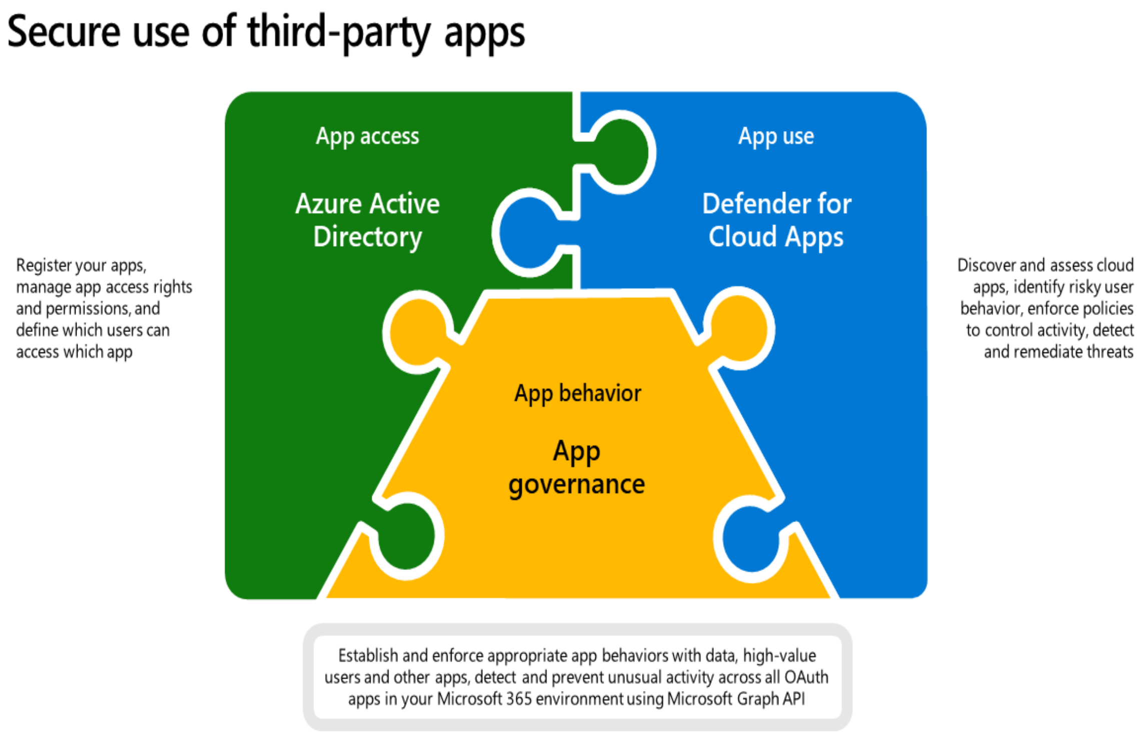


Figure 4-1: Class diagram

4.4 BEHAVIORAL VIEW

As we mentioned previously that activity diagram, and sequence diagram provide the behavioral view for our project. Behavioral diagrams are used to describe the interactionbetween the actors and the system. All the activities that are performed by the actors and the system are introduced in some way. A behavioural view is a computer science approach that focuses on the behaviour of software systems during runtime. It is concerned with how software components interact with one another, their behaviours, and their interfaces with users and other systems. This viewpoint is employed in the analysis and modelling of the dynamic behaviour of software systems in response to various inputs and stimuli.


Secure use of third-party apps
App access
Azure Active Directory
App use
Defender for Cloud Apps
App behavior
App governance
Register your apps, manage app access rights and permissions, and define which users can access which app
Discover and assess cloud apps, identify risky user behavior, enforce policies to control activity, detect and remediate threats
Establish and enforce appropriate app behaviors with data, high-value users and other apps, detect and prevent unusual activity across all OAuth apps in your Microsoft 365 environment using Microsoft Graph API

The behavioural view is a software engineering approach that is concerned with the behaviour of software systems during runtime. It focuses on modelling a system's dynamic behaviour, such as how it responds to human input, interacts with other systems or devices, and changes over time.

The behavioural perspective is critical because it enables software designers and developers to model, evaluate, and optimise the behaviour of software systems. Understanding a system's dynamic behaviour allows developers to discover possible problems and chances for improvement, allowing them to create more efficient, effective, and user-friendly solutions.

4.4.1 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depictsthe objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

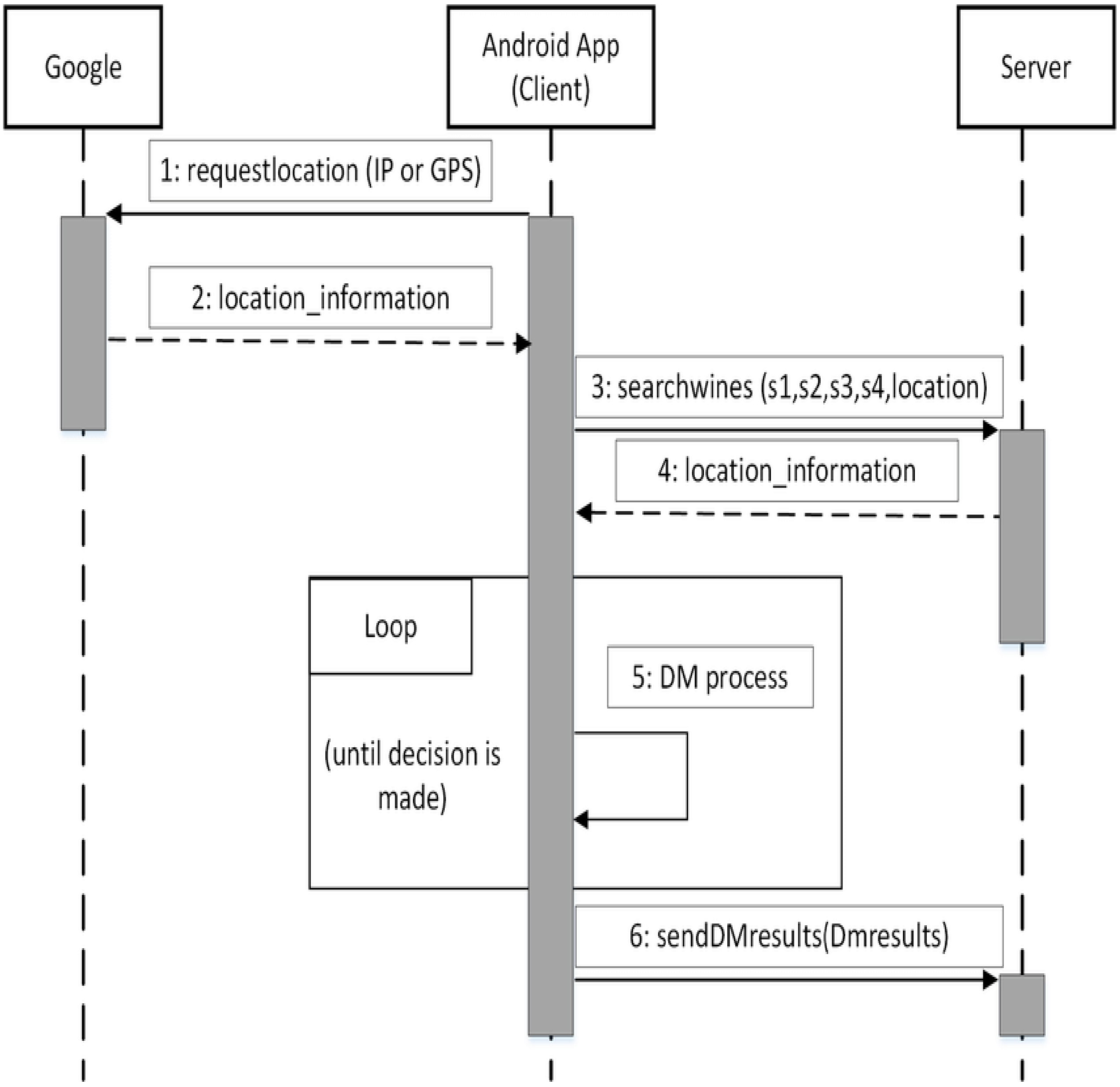


Figure 4-2: Sequence diagram

4.4.2 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activitiesand actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by step workflows of components in a system. An activity diagram shows the overall flow of processes.

UML Activity Diagram

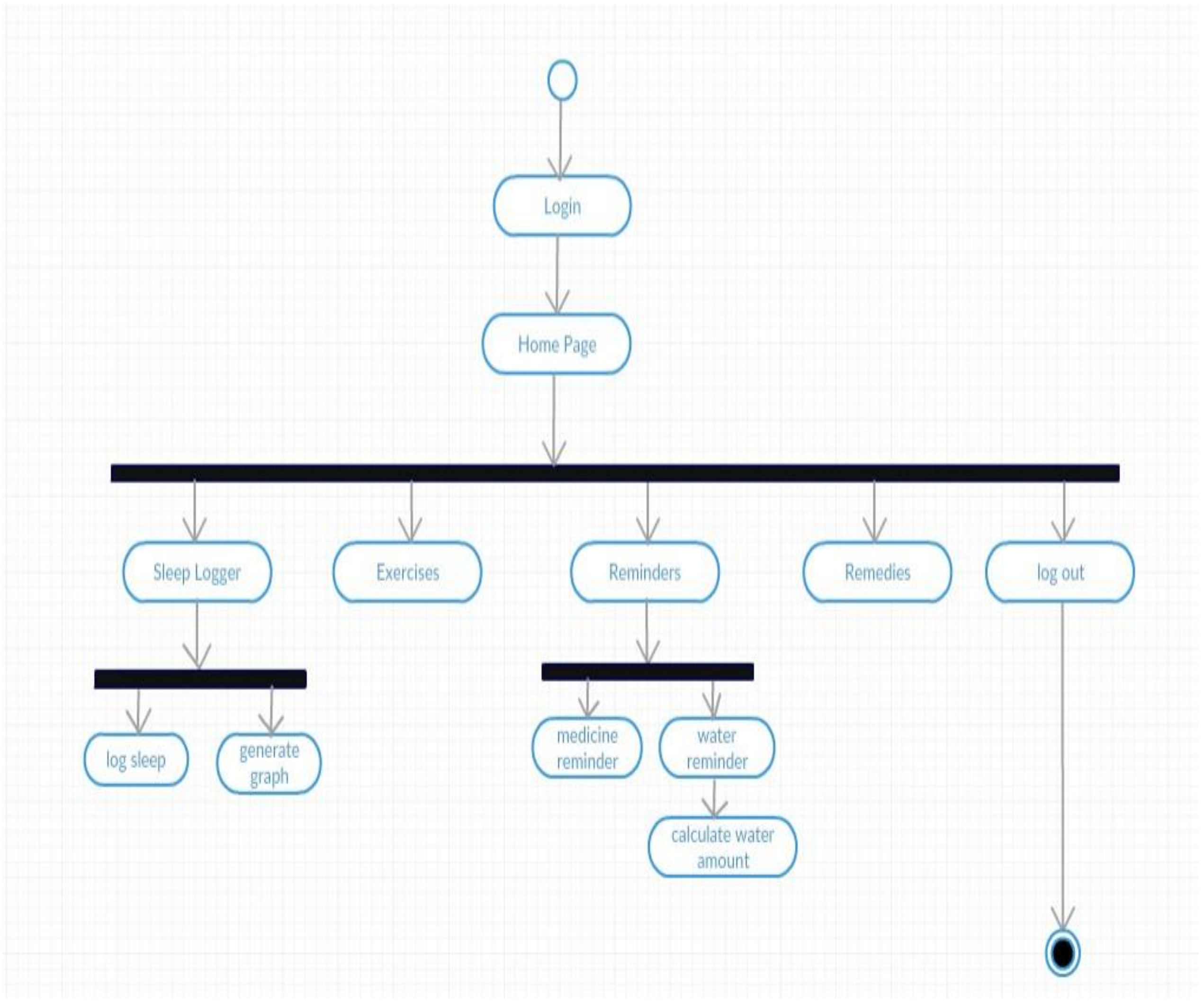


Figure 4-3: System Activity Diagram

4.4.3 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).A DFD shows what kinds of information will be input to and output from the system, wherethe data will come from and go to, and where the data will be stored.It does not show information about the timing of processes, or information about whetherprocesses will operate in sequence or in parallel.

A data flow diagram (DFD) shows how data moves visually across a system or process. It is frequently used to explain how data is entered, processed, stored, and produced in a system in software engineering, business analysis, and systems analysis.

Data sources, processes, and destinations are represented by a sequence of linked boxes or shapes in DFDs. Arrows that signify the flow of data between the boxes connect them. The type of data being transferred, such as client information, product data, or order details.

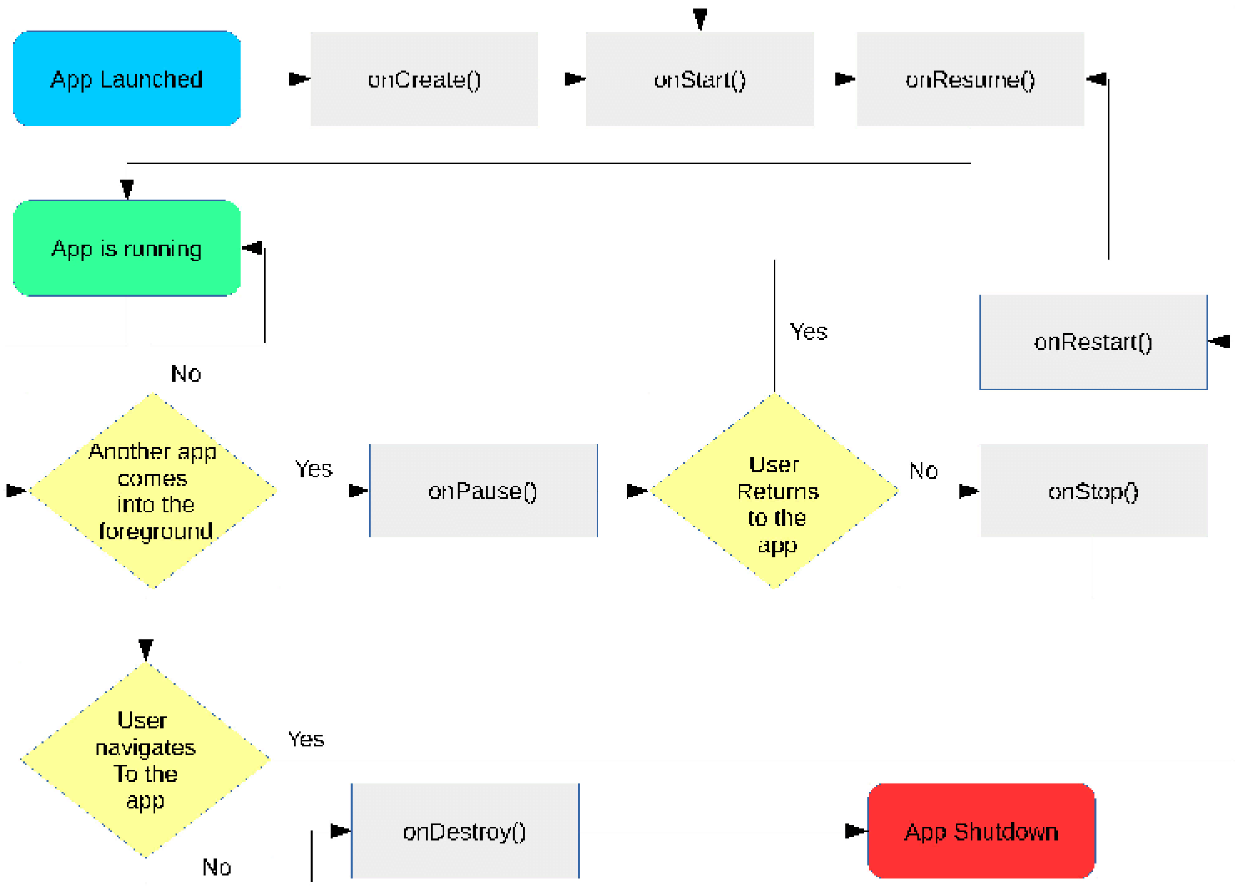


Figure 4-4: Context Diagram

CHAPTER 5 : IMPLEMENTATION & TESTING

5.1 SCREEN SHOTS OF THE SYSTEM:

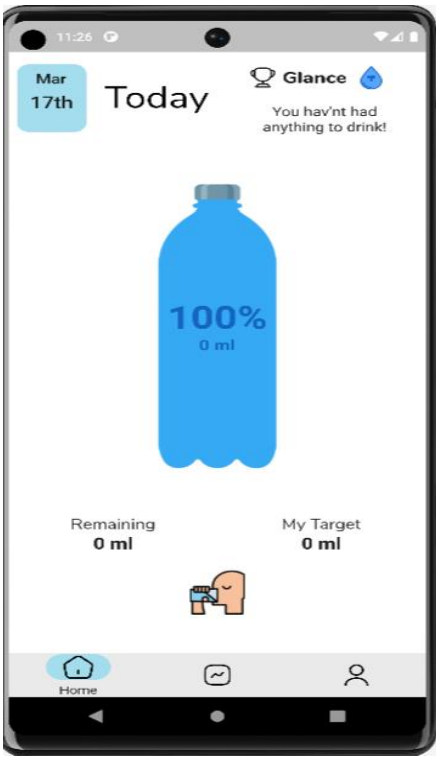
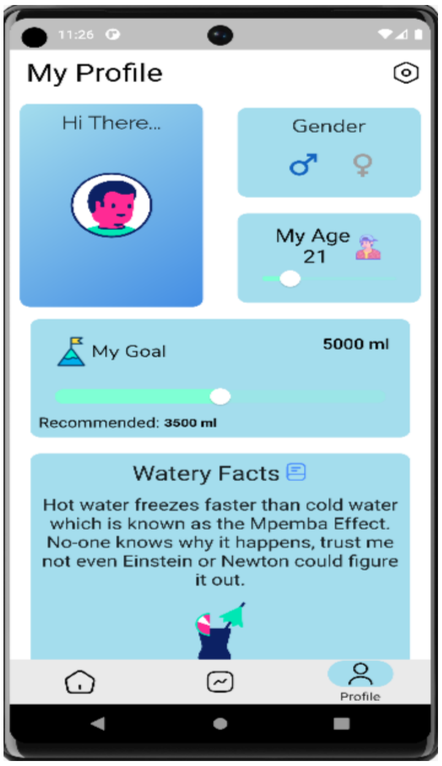


Figure 5-1: Main page



Hot water freezes faster than cold water which is known as the Mpemba Effect. No-one knows why it happens, trust me not even Einstein or Newton could figure it out.

Figure 5-2:Profile page

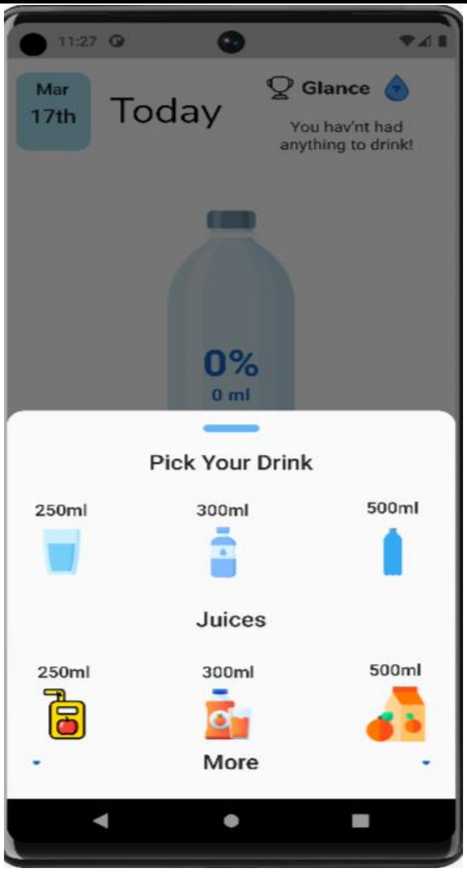


Figure 5-3: water intake page

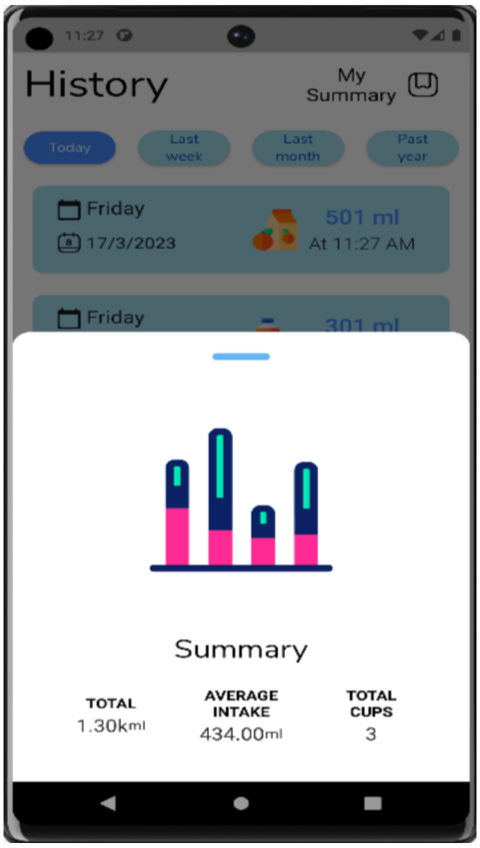


Figure 5-4:Data view page

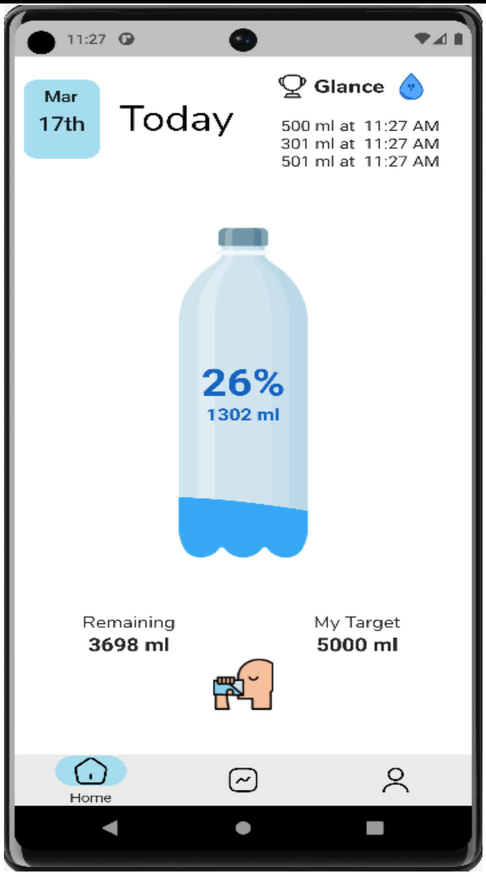


Figure 5-4:main interface

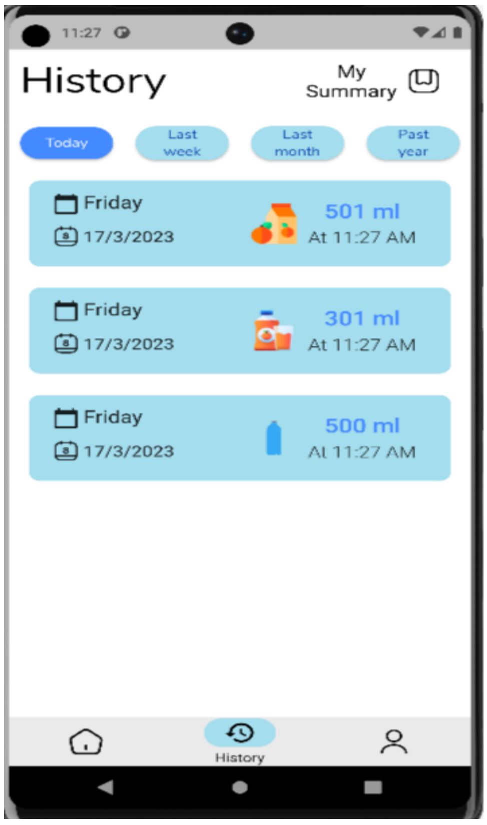


Figure 5-4:History interface

5.2 DEFINITION AND THE GOAL OF TESTING

Process of creating a program consists of the following phases:

1. Defining a problem;

2. Designing a program;

3. Building a program

4. Analyzing performances of a program

5. Final arranging of a product.

According to this classification, software testing is a component of the third phase, and means checking if a program for specified inputs gives correctly and expected results. So the main aim of testing is to analyze the performance and to evaluate the errors that occur when the program is executed with different input sources and running in different operatingenvironments.

Software testing is an important component of software quality assurance, andmany software organizations are spending up to 40% of their resources on testing. For life-critical software (e.g., flight control) testing can be highly expensive. Because of that, manystudies about risk analysis have been made. This term means the probability that a softwareproject will experience undesirable events, such as schedule delays, cost overruns, or outright cancellation.

There are a many definitions of software testing, but one can shortly define that as: A process of executing a program with the goal of finding errors. So, testing means that one inspects behavior of a program on a finite set of test cases (a set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement) forwhich valued inputs always exist.

In practice, the whole set of test cases is considered as infinite, therefore theoretically thereare too many test cases even for the simplest programs. In this case, testing could require months and months to execute. So, how to select the most proper set of test cases? In practice, various techniques are used for that, and some of them are correlated with risk analysis, while others with test engineering expertise.

Testing is an activity performed for evaluating software quality and for improving it. Hence,the goal of testing is systematical detection of different classes of errors (error can be defined as a human action that produces an incorrect result) in a minimum amount of time.

The primary purpose of app testing is to guarantee that it works properly and satisfies the user's expectations. App testing is an essential element of the app development process since it identifies any faults or defects in the app before it is published to the public.

Some of the particular purposes of app testing include:

1. Ensure functionality: The app should be tested to verify that it functions correctly and that all features and functionalities perform as planned.
2. Ensure compatibility: The app should be tested on a variety of devices and operating systems to verify that it works on all major platforms.
3. Ensure performance: The app should be tested to verify that it runs smoothly and responds quickly to user input. This covers speed, stability, and resource utilisation testing.
4. Ensure security: The app should be tested to verify that it is safe and that user data is secure. This covers vulnerability testing for SQL injection, cross-site scripting, and other security issues.
5. Usability testing: The app should be evaluated to verify that it is user-friendly and simple to use. This encompasses user interface design, ease of navigation, and general user experience testing.

Developers may guarantee that the software is of high quality, dependable, and secure by attaining these goals through testing. This, in turn, can lead to improved user pleasure, engagement, and higher app store ratings. Finally, app testing is a critical phase in the app development process.

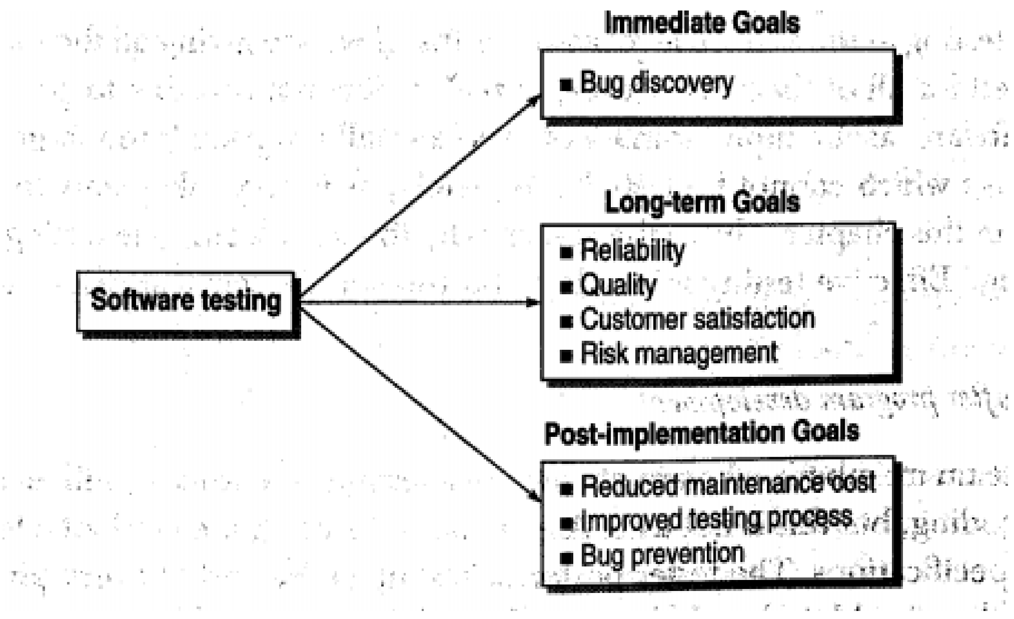


Figure 1: Software Testing Goals

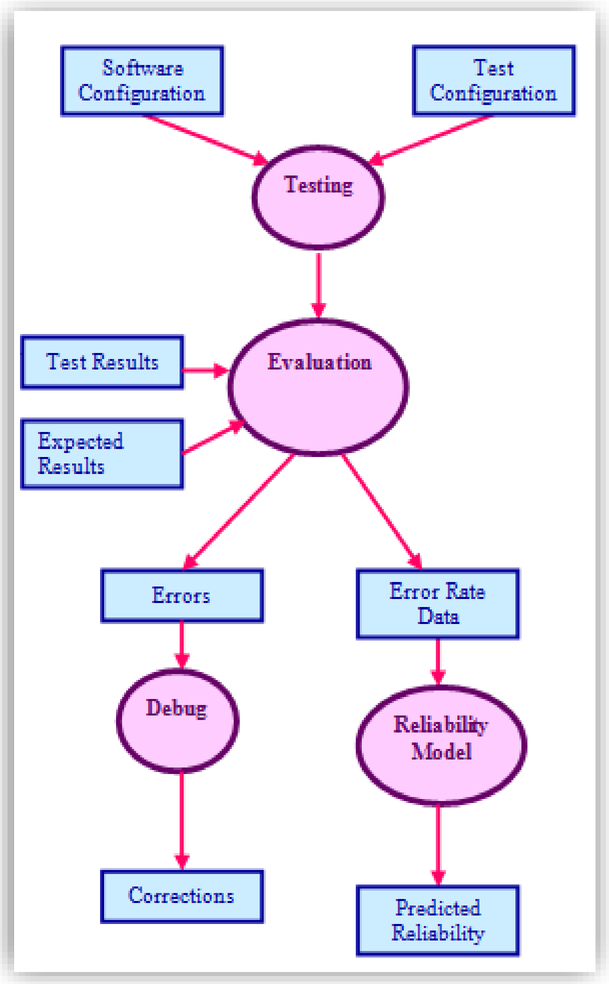


Figure 5-5:Test Information Flow

There are different types of approaches for testing a .NET framework based application. The types of testing are:

- Unit testing
- Validation testing
- Integration testing
- User acceptance testing
- Output testing
- Black box and white box testing.

5.3 METHODS OF TESTING

5.3.1 Unit Testing

Unit testing is the approach of taking a small part of testable application and executing it according to the requirements and testing the application behavior. Unit testing is used for detecting the defects that occur during execution .

When an algorithm is executed, the integrity should be maintained by the data structures. Unit testing is made use for testing the functionality of each algorithm during execution. Unit testing can be used in the bottom up test approach which makes the integration test much easier. Unit testing reduces the ambiguity in the units. Unit testing uses regression testing, which makes the execution simpler. Using regression testing, the fault can be easilyidentified and fixed.

In this project, the purposed system of hiding the data using different phases likes encryption, decryption, etc. So, for getting the correct output all the functions that are usedare executed and tested at least once making sure that all the control paths, error handling and control structures are in proper manner. Unit testing has its applications for extreme programming, testing unit frame works and good support for language level unit testing. Software testing with a focus on individual software system units or components is known as unit testing. Unit testing checks that each piece of software operates as intended and complies with specifications. Developers often carry out unit testing, which is done before the code is merged and tested as a full system and is done early in the development process.

Each time the code is modified, unit tests are automatically performed to make sure that the new code won't break any functionality already in place. A function or method, for example, can be tested independently from the rest of the system using unit tests, which are created to check the smallest feasible unit of code. This enables programs to quickly.

Unit testing are classified into two types: manual and automated. Both are used to validate certain components of the system under test. Unit testing increases programme efficiency by guaranteeing that all individual components function properly. It's a blank slate for creating better tools and increasing output.

Individual components of a software programme or application are tested during unit testing. The major reason for this is to ensure that all of the different elements are functioning properly. A unit is the smallest feasible software component that can be tested.

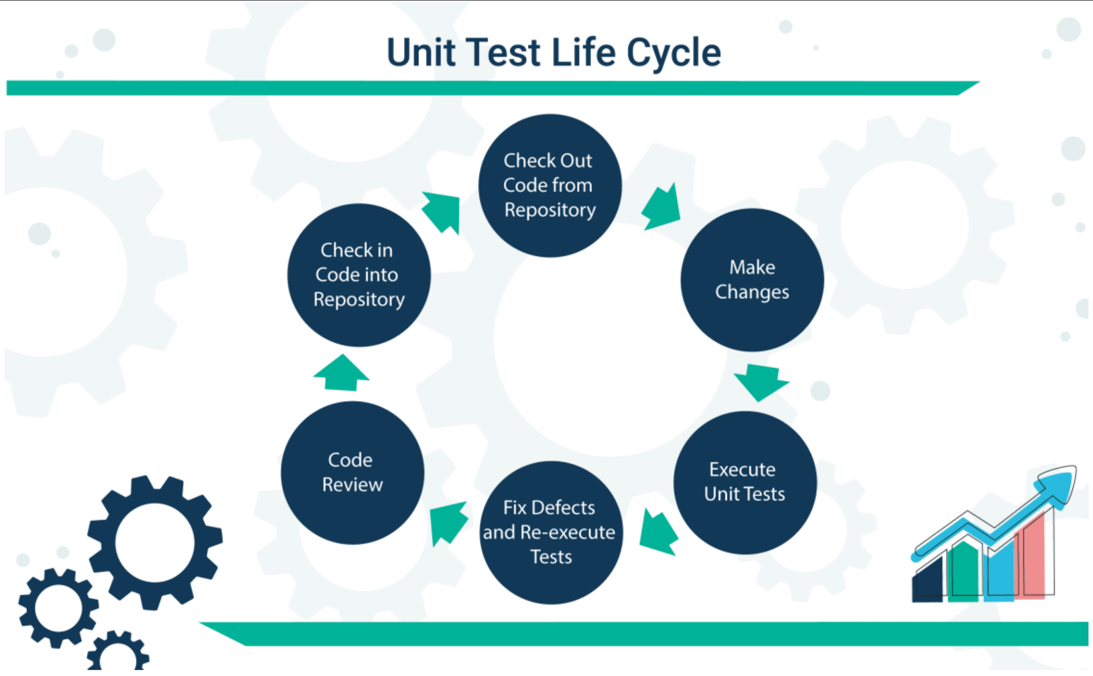

Unit Test Life Cycle
Check Out Code from Repository
Make Changes
Execute Unit Tests
Fix Defects and Re-execute Tests
Code Review
Check in Code into Repository

Figure 5-6:unit testing

5.3.2 Validation Testing

Validation is the process of finding whether the product is built correct or not. The software application or product that is designed should fulfill the requirements and reach theexpectations set by the user. Validation is done while developing or at the final stage of development process to determine whether it is satisfies the specified requirements of user. Using validation test the developer can qualify the design, performance and its operations. Also the accuracy, repeatability, selectivity, Limit of detection and quantification can be specified using Validation testing. the procedure of assessing software to see if it fulfils stated business requirements either during development or at the end of development. Validation testing makes ensuring that the product really does fulfil the demands of the customer. It is also possible to describe it as proving that a product works as intended when used in the right setting.

Is the product we're creating the correct one? is answered by this statement. The process of confirming that the tested and created software meets the demands of the client or user is known as validation testing. Detail-oriented testing must be done on the business requirement logic or scenarios. An application's essential features must all be evaluated in this area. Knowing how to validate the business logic or scenarios that are provided to you as a tester is always crucial. The Validation Process is one such technique that aids in a thorough assessment of the functions.

When requested to do a validation test, you are given a lot of responsibility since you must test all the important business criteria in accordance with the requirements of the users. There shouldn't be even the slightest deviation from the user's needs. Software undergoes a development process in order to be ready for consumer usage, just like any other product. Needs analysis, requirement analysis, design, development, implementation, testing, deployment, and maintenance are all phases in the software development process. Knowing these phases and associated methods, including validation testing, might be helpful if you work in the software development industry. To find areas of the code that need improvement, developers can carry out validation testing themselves or work with quality assurance specialists, outside validation testing specialists, or clients. To assist guarantee the product is prepared for the market, developers can combine this kind of testing with other helpful procedures like product verification, debugging, and certification.

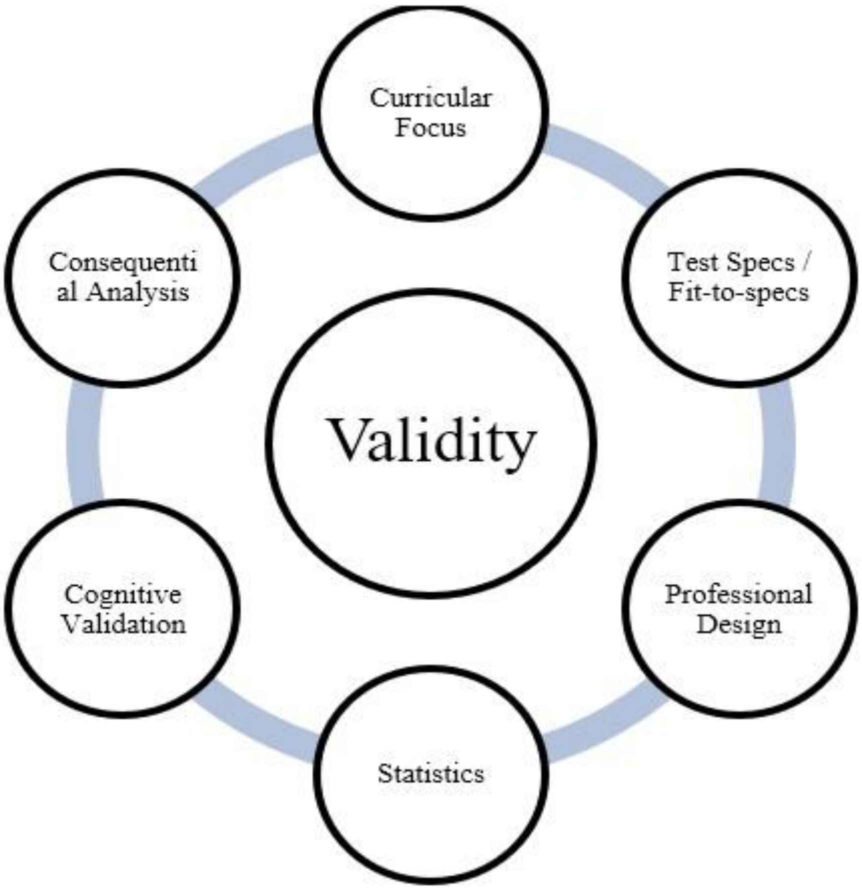


Figure 5-7:validity testing

5.3.2 Output Testing

After completion of validation testing the next process is output testing. Output testing is the process of testing the output generated by the application for the specified inputs. This process checks weather the application is producing the required output as per the user's specification or not. A crucial component of software testing is output testing, which entails confirming that the output produced by a system or software application is accurate and complies with the requirements. An application's output can take many different forms, including reports, messages, files, data displays, or any other output that the system produces in response to user input. Output testing's main objective is to make sure that the application fulfils the system's functional and non-functional criteria while producing the desired output error-free. Beginning with determining the expected output requirements and developing test cases based on these requirements, the output testing process entails numerous phases. These test cases are run, and the actual output produced by the system is compared to the anticipated result. The development team records, reports, and fixes any inconsistencies or mistakes discovered throughout the comparison process. Depending on the complexity of the application and the testing objectives, output testing can be either human or automated. Whereas automated testing employs tools and scripts to automate the testing process, manual testing requires the tester to manually execute the test cases and check the output.

Because the output produced by an application is the main method users engage with the system, output testing is essential. In some circumstances, incorrect or unexpected output might result in user annoyance, decreased productivity, and even monetary losses. In order to guarantee that the output satisfies the needs and expectations of the user, output testing should be a crucial component of the software testing process.

The output testing can be done by considering mainly by updating the test plans, the behavior of application with different type of inputs and with produced outputs, making thebest use of the operating capacity and considering the recommendations for fixing the issues.

To sum up, output testing is an essential component of software testing that makes sure the output produced by an application satisfies both functional and non-functional criteria. It entails determining the specifications for the expected output, developing test cases, running them, and contrasting the results with what was anticipated. To make sure that the program provides the desired output, is error-free, and satisfies user requirements and expectations, extensive output testing should be carried out.

A cause (input) - effect (output) testing approach tests combinations of input values in a systematic manner. Pair-wise Testing - The behaviour of software is affected by a number of variables. Several parameters are examined for their distinct values in paired testing.

We expand the idea of input-output conformity (IOCO) testing to encompass software product line behavioural models (SPLs). We discuss the concepts of residual and spinal testing. These concepts enable the architecture of the test process for SPLs by accounting for variability and extracting different test suites for common and specialised elements of an SPL. The concepts of residual and spinal test suites introduced enable for a concentrate on the newly introduced behaviour while avoiding wasteful re-testing of the old one. Residual test suites are extremely cautious since they involve retesting of past behaviour that might lead to new behaviour. Nevertheless, spinal test suites actively reduce the old tests and focus exclusively on the test sequences required to achieve the new behaviour. We show that residual testing is thorough but does not generally result in significant test-suite reduction. Spinal testing, on the other hand, is not always exhaustive but does minimise the test-suite. We provide adequate implementation conditions to ensure the thoroughness of spinal testing. Lastly, we define and examine an example involving the European Train Control System's Ceiling Speed Monitoring Function.

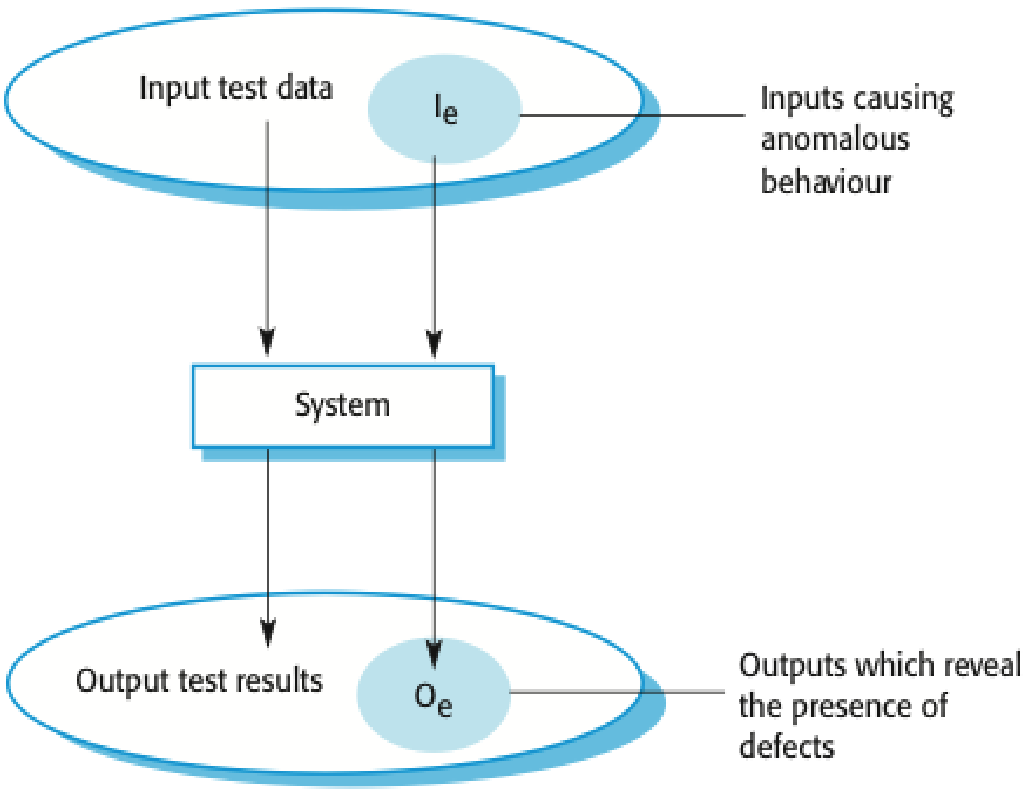

Input test data
I_e
Inputs causing anomalous behaviour
System
Output test results
O_e
Outputs which reveal the presence of defects

Figure 5-8:Output testing

5.3.3 Integration Testing

Integration testing is an extension to unit testing, after unit testing the units are integrated with the logical program. The integration testing is the process of examining the working behavior of the particular unit after embedding with program. This procedure identifies theproblems that occur during the combination of units. Integration testing is a type of software testing that focuses on examining how various software application modules or components interact with one another and share data. Integration testing aims to find any issues or faults that develop when several components are joined and communicate with one another. Normally, system testing comes after unit testing and before integration testing. Early in the development cycle, integration difficulties may be found and fixed, lowering the likelihood of later, more serious, and expensive problems. Module by module testing of an integration may be done. This may be done to ensure that the necessary steps are executed in the right order. Also, you have to perform this if you don't want to miss any integration possibilities.

A software testing strategy known as "big-bang integration testing" combines and tests all of a software application's modules or components at once. This method is often applied when there is little interaction between the software components or when the testing of individual components is restricted by the development environment. Big-bang integration testing aims to find any integration issues that develop when the components are put together as well as to validate the system's overall functioning. Big-bang integration testing can be helpful in some circumstances, but it can also be a risky technique since it can be challenging to find and diagnose issues due to the complexity of the system and the amount of interactions between components.

- We go for the integration testing only after the functional testing is completed on each module of the application.
- We always do integration testing by picking module by module so that a proper sequence is followed, and also we don't miss out on any integration scenarios.
- First, determine the test case strategy through which executable test cases can be prepared according to test data.
- Examine the structure and architecture of the application and identify the crucial modules to test them first and also identify all possible scenarios.
- Design test cases to verify each interface in detail.
- Choose input data for test case execution. Input data plays a significant role in testing.
- If we find any bugs then communicate the bug reports to developers and fix defects and retest.

- Perform positive and negative integration testing.

The integration testing can be commonly done in three approaches:

- Top-down approach
- Bottom-up approach
- Umbrella approach

5.3.3.1 Top-down approach:

In the top-down approach the highest level module should be considered first and integrated. This approach makes the high level logic and data flow to test first and reduce the necessityof drivers. One disadvantage with top-down approach is its poor support and functionality is limited. Top-down integration testing is an approach to integration testing that simulates the behavior of lower-level modules that have not yet been integrated. Stubs are modules that temporarily take the place of a called module and produce the same results as the final product. When the software has to communicate with an outside system, "Stubs" are utilised in place of the "called" modules. A software development process known as the "top-down approach" starts with a broad overview of the system's goals before gradually breaking them down into smaller, more particular tasks. This method is sometimes referred to as the "waterfall" model because it employs a linear flow of steps that are carried out in a planned order, building on the outcomes of the preceding phase. The overarching aims and objectives of the system are first defined using the top-down method, and they are then converted into a list of functional requirements. The functional needs are then divided into more manageable, detailed requirements, and each of these requirements is produced into a design specification. After that, the design definition is put into code, examined, and released.

The simplicity and clarity of the top-down strategy are two of its key benefits. It offers developers a clear roadmap that is simple to grasp and follow. The top-down method also makes sure that the design and development process is concentrated on the goals of the system, reducing the chance of adding features that aren't essential or important. The top-down strategy also has the advantage of enabling early problem diagnosis and resolution. Potential issues may be found and fixed early in the development process by breaking down the system requirements into smaller sub-tasks, which can ultimately save time and money.

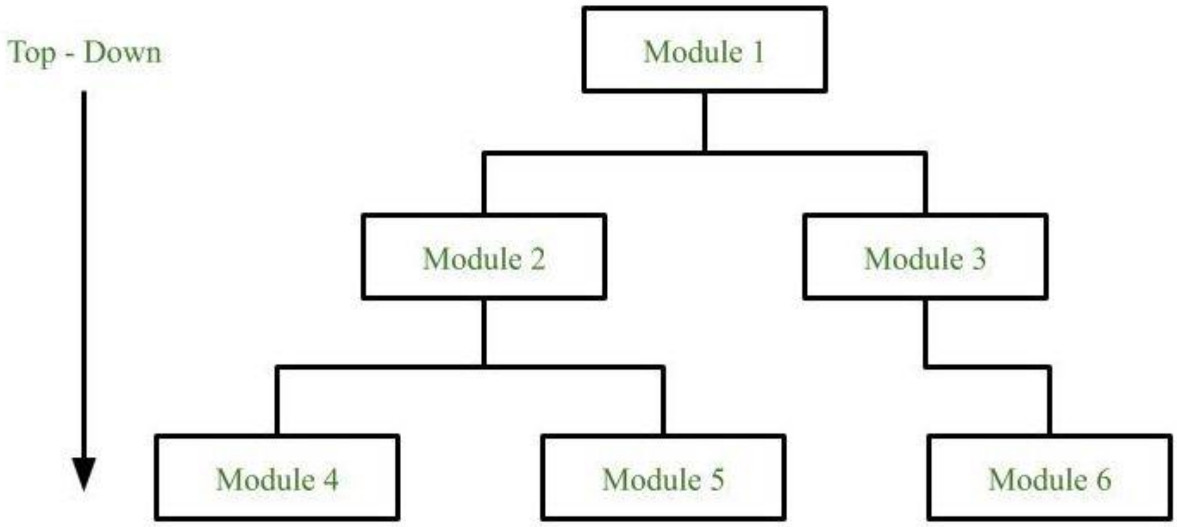


Figure 5-9:Top down approach

The top-down strategy, however, also has significant disadvantages. One of the primary complaints of this technique is that it may be stiff and inflexible since it can be challenging to incorporate modifications to the system requirements or design after the development process has started. If adjustments need to be made later in the process, this might cause delays and cost overruns.

In conclusion, the top-down approach is a style of software development that focuses on attaining the goals of the system by segmenting requirements into more manageable, detailed subtasks. It is a straightforward strategy that can aid in seeing and fixing problems early in the development process, but it can also be rigid and hard to alter once the process has started.

5.3.3.2 Bottom-up approach:

Bottom-up approach is opposite to top-down approach. In this approach, the lowest level units are considered and integrated first. Those units are known as utility units. The utility units are tested first so that the usage of stubs is reduced. The disadvantage in this method is that it needs the respective drivers which make the test complicated, the support is poor and the functionality is limited. Bottom-up A sort of incremental integration testing methodology called testing involves connecting or integrating two or more modules as you work your way up the control flow of the architectural structure from bottom to top. They test low-level modules first, followed by testing high-level modules. Inductive reasoning is another name for this kind of testing or strategy, and it is frequently used as a synonym for synthesis. User-friendly testing, or bottom-up testing, speeds up software development in general. High success rates and durable outcomes are obtained from this testing.

Low-level modules or pieces are combined to build clusters. These groups, sometimes known as builds, are in charge of carrying out a certain auxiliary or ancillary function of a piece of software. Writing a control software is crucial for testing. High-level modules or drivers are other names for these control applications. It merely coordinates a test case's input and output.

The complete build or cluster that contains low-level modules is tested.

Finally, with the aid of control flow, the control program, drivers, or high-level modules are eliminated, and clusters are merged by climbing upward from the bottom to the top of the program structure. The bottom-up technique is a software development methodology that starts with tiny, isolated modules that are then joined and integrated to construct bigger, more complicated systems. This method is also known as the "iterative" or "incremental" model because it entails breaking down the development process into smaller, manageable portions that are built and evaluated progressively.

The development process in the bottom-up approach begins with the implementation of the smallest and simplest components of the system, such as individual functions or modules. These components are then tested and combined to build bigger and more complicated modules, which are subsequently tested and integrated with other modules until the entire system is complete. One of the primary benefits of the bottom-up strategy is its adaptability and flexibility. Because each component is built and tested individually, changes and alterations to the system requirements or design may be included more easily during the development process. This ensures that the finished product fulfils the needs and expectations of the user, and that any flaws or faults are recognized and rectified early in the development process.

Another advantage of the bottom-up strategy is that it enables the early release of functional software. Developers may provide functioning software incrementally by breaking down the development process into smaller and more manageable pieces, which can bring value to the user and assist to create trust and confidence in the development team.Bottom-up strategy is its adaptability and flexibility,because each component is built and tested individually, changes and alterations to the system requirements or design may be included more easily during the development process and it is widely used in the approaches to maintain the scalability of the project.

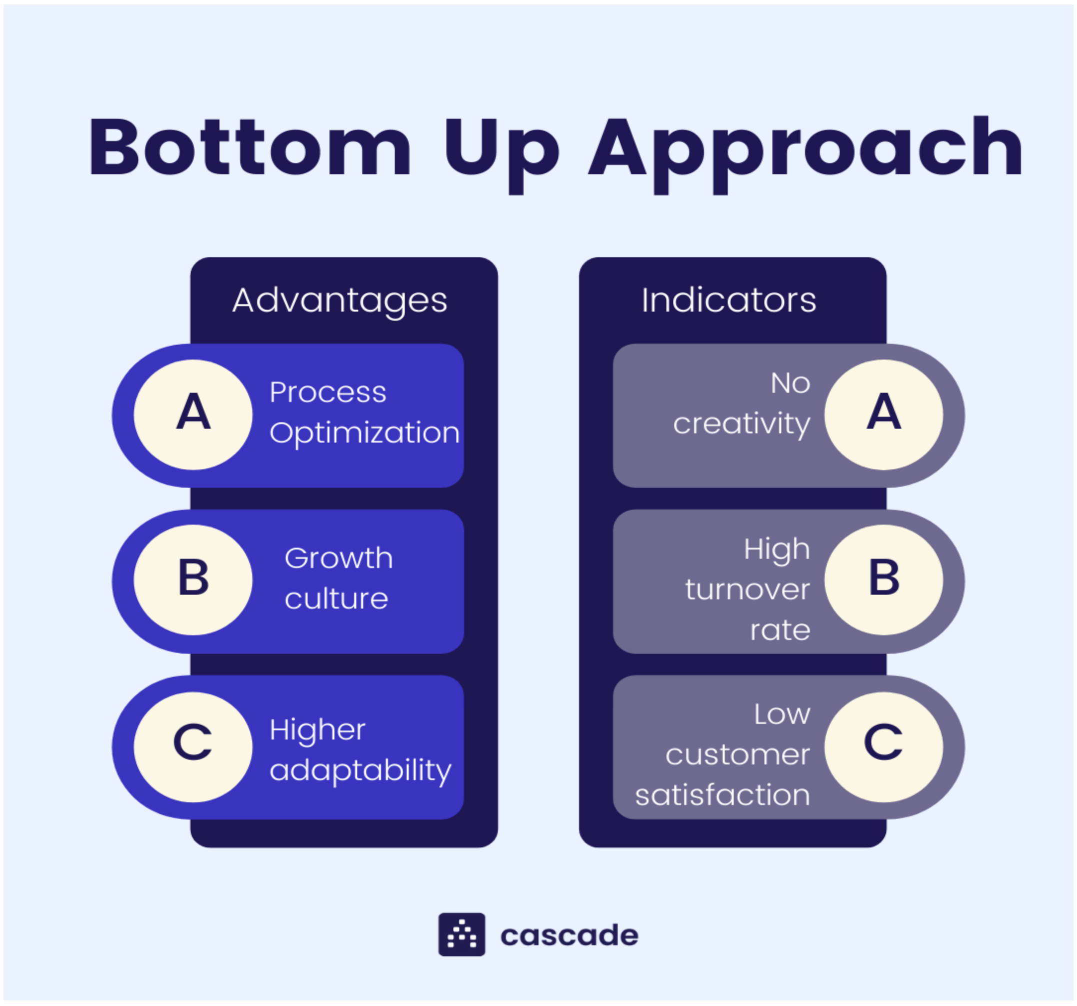


Figure 5-10:Bottom up approach

Yet, the bottom-up strategy has significant drawbacks. One of the primary objections levelled against this technique is that it is time-consuming and expensive, as each module must be designed and tested individually before being connected with others. If the testing process takes longer than expected, this might result in delays and cost overruns.

Finally, the bottom-up approach is a software development process that entails the construction of tiny, independent modules, which are subsequently joined and integrated to build bigger, more sophisticated systems. It is a versatile and adaptive strategy that allows for the early delivery of usable software, but it may also be time-consuming and expensive owing to the considerable testing and integration required.

5.3.3.3 Umbrella approach:

The third approach is umbrella approach, which makes use of both the top - bottom and bottom - top approaches. This method tests the integration of units along with its functionaldata and control paths. After using the top - bottom and bottom-top approaches, the outputsare integrated in top - bottom manner.The advantage of this approach is that it provides good support for the release of limited functionality as well as minimizing the needs of drivers and hubs. The main disadvantage is that it is less systematic than the other two approaches. It is feasible to create big and small cells that are co-located at a single place by employing various antenna heights [typically in the same structure or tower] and varying power levels. This method is known as the umbrella cell approach.

The figure above shows an umbrella cell that is situated beside a few smaller micro cells. For big umbrella cells, a distinct set of channels is here set aside. The umbrella cell technique is utilised to give high speed users access over a wide region while giving low speed users service over a limited area. This makes sure that there are as few handoffs as possible for customers travelling at high speeds while also providing more micro cell channels for users on foot. The umbrella approach is a software development methodology that blends top-down and bottom-up features. It is also known as the "hybrid" paradigm since it aims to utilise both techniques' benefits while reducing their flaws.

The umbrella technique, like the top-down approach, begins with a high-level overview of the system's objectives and needs. The umbrella technique, like the bottom-up approach, identifies the essential components and modules of the system and builds them independently, rather than breaking down the requirements into smaller and more detailed sub-tasks.Once the major components and modules have been produced, they are combined and tested to ensure that they perform as planned. This integration and testing procedure is carried out in a top-down fashion, with higher-level components and modules tested and integrated first, followed by lower-level components and modules.

One of the key benefits of the umbrella strategy is its versatility and flexibility. It enables the development of major components and modules from the ground up, which can aid in identifying and correcting difficulties early in the development process. Similarly to the top-down method, it provides a clear roadmap for the development process by beginning with a high-level overview of the system's objectives and needs. Another advantage of using an umbrella method is that it can assist to reduce the likelihood of delays and cost overruns. It is simpler to discover and fix errors before they affect the entire system by designing and testing essential components and modules individually. This can assist to lessen the risk of delays and cost overruns caused by unanticipated problems or defects.

The umbrella strategy, however, has several downsides. One of the primary objections levelled against this technique is that it may be complicated and difficult to maintain due to the various development phases and integration points involved. This can make ensuring that all components and modules function together flawlessly difficult.

5.4 USER ACCEPTANCE TESTING

User acceptance testing is the process of obtaining the confirmation from the user that the system meets the set of specified requirements. It is the final stage of project; the user performs various tests during the design of the applications and makes further modificationsaccording to the requirements to achieve the final result. The user acceptance testing gives the confidence to the clients about the performance of system. A sort of testing called user. User acceptance testing (UAT), commonly referred to as beta testing or end-user testing, is the process of having users or clients test software to see if they can accept it or not. Once the functional, system, and regression testing is finished, this is the last testing carried out.

This testing's primary goal is to confirm that the software meets the necessary standards for the company. End users that are familiar with the business requirements perform this validation. Several forms of acceptance testing include UAT, alpha testing, and beta testing.

The user acceptance test is the final testing performed before the program goes live, therefore it goes without saying that this is the final opportunity for the customer to test the product and determine whether it is suitable for use.Prior to its official release, users are given the chance to interact with the program in UAT to determine whether any features have been missed or if it has any issues. UAT can be carried out internally using test subjects paid to use the program, by paid test subjects, or by making the test version accessible for The testing carried out at the conclusion of the development cycle is validated by the acceptance testing. It is normally finished after system testing, integration testing, system testing, and quality assurance. If the program is not well welcomed by its target users, it may still not fulfil the criteria even after going through further testing rounds and being fully functioning. This may occur if the developers' understanding of the software's requirements was lacking, if the project's scope altered as a result of development changes, or if the program just wasn't ready for testing in a dynamic, real-world setting. UAT prevents the deployment of flawed, useless, or incomplete software products overall.

Users conduct real-world testing on the software during UAT to ensure that it satisfies their unique needs and can be utilised effectively in their day-to-day operations. UAT tests are often carried out in a controlled environment that closely resembles the user's actual environment, such as a test lab or a production-like staging environment.Usability testing, performance testing, and security testing are all examples of UAT. UAT tests are often done based on the user's needs and expectations, and can incorporate both manual and automated testing methodologies. One of the primary advantages of UAT is that it allows users to check the program and offer feedback on any bugs or flaws that they find. UAT can also assist to decrease project failure risk by discovering and addressing errors before the program is delivered to the production environment. This can assist to reduce the chance of downtime, data loss, and other issues affecting the user's operations.

Finally, User Acceptance Testing is an important stage of software testing that confirms the program fulfils the user's criteria and is ready for deployment. It is carried out by the program's end users to test its functionality, usability, and performance, as well as to discover and repair any faults before the product is launched. UAT allows users to offer input on the program and reduces the chance of project failure by finding and addressing faults.

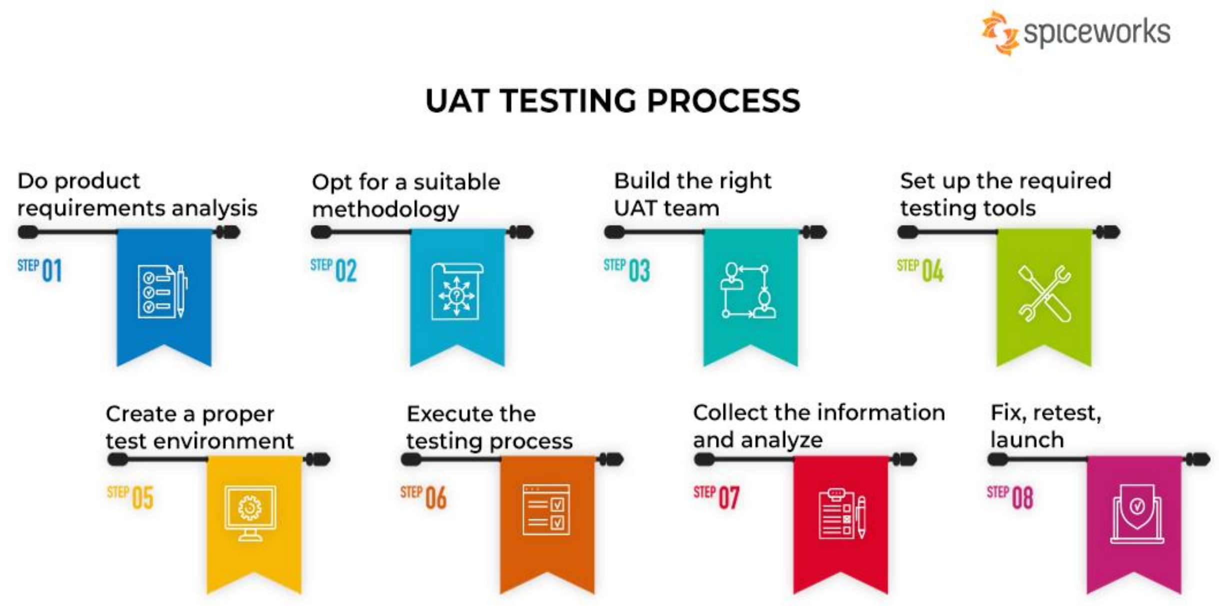

spiceworks
UAT TESTING PROCESS
Do product requirements analysis
STEP 01
Opt for a suitable methodology
STEP 02
Build the right UAT team
STEP 03
Set up the required testing tools
STEP 04
Create a proper test environment
STEP 05
Execute the testing process
STEP 06
Collect the information and analyze
STEP 07
Fix, retest, launch
STEP 08

Figure 5-11:User acceptance testing

5.5 BLACK BOX AND WHITE BOX TESTING

Black box testing is the testing approach which tells us about the possible combinations for the end-user action. Black box testing doesn't need the knowledge about the interior connections or programming code. In the black box testing, the user tests the application bygiving different sources and checks whether the output for the specified input is appropriateor not.

White box testing is also known as "glass box" or "clear box" or "open box" testing. It is opposite to the black box testing. In the white box testing, we can create test cases by checking the code and executing in certain intervals and know the potential errors. The analysis of the code can be done by giving suitable inputs for the specified applications and using the source code for the application blocks. Black box testing is used to determine whether a system acts as intended in various circumstances and to look for any flaws or mistakes that could be present in the system's operation or output. Black box testing may be used to a variety of software types, including online applications, mobile apps, and desktop applications. It can be carried out manually or through automated testing methods.Black box testing has a number of benefits, including the fact that it can find bugs that are hard to find using other testing techniques, it is user experience and functionality-focused, and it does not require knowledge of the underlying workings of the system. Black box testing may not cover all situations or edge cases, which is one of its potential drawbacks.

White box testing is also known as "glass box" or "clear box" or "open box" testing. It is opposite to the black box testing. In the white box testing, we can create test cases by checking the code and executing in certain intervals and know the potential errors. The analysis of the code can be done by giving suitable inputs for the specified applications andusing the source code for the application blocks. White box testing is a testing method that examines the internal organization, code, and design of software in order to validate input-output functionality and enhance design, usability, and security. White box testing is also known as clear box testing, open box testing, transparent box testing, code-based testing, and glass box testing since code is visible to testers during this type of testing. The Box Testing method of software testing consists of two components. Blackbox testing, its counterpart, involves testing from an outside or end-user perspective. On the other hand, White box testing in software engineering is centered on internal testing and is focused on the inner workings of an application. Due to the idea of a see-through box, the term "Whitebox" was adopted. Whitebox, which stands for "clear box," alludes to the ability to look through the software's exterior ("box"). The "black box" in the phrase "Black Box Testing" stands for the inability to see the inner workings of the program so that only the end-user experience can be assessed. White box testing is a testing method that examines the internal organization, code, and design of software in order to validate input-output functionality and enhance design, usability, and security. . It can be carried out manually or through automated testing methods.Black box testing has a number of benefits, including the fact that it can find bugs that are hard to find using other testing techniques, it is user experience and functionality-focused, and it does not require knowledge

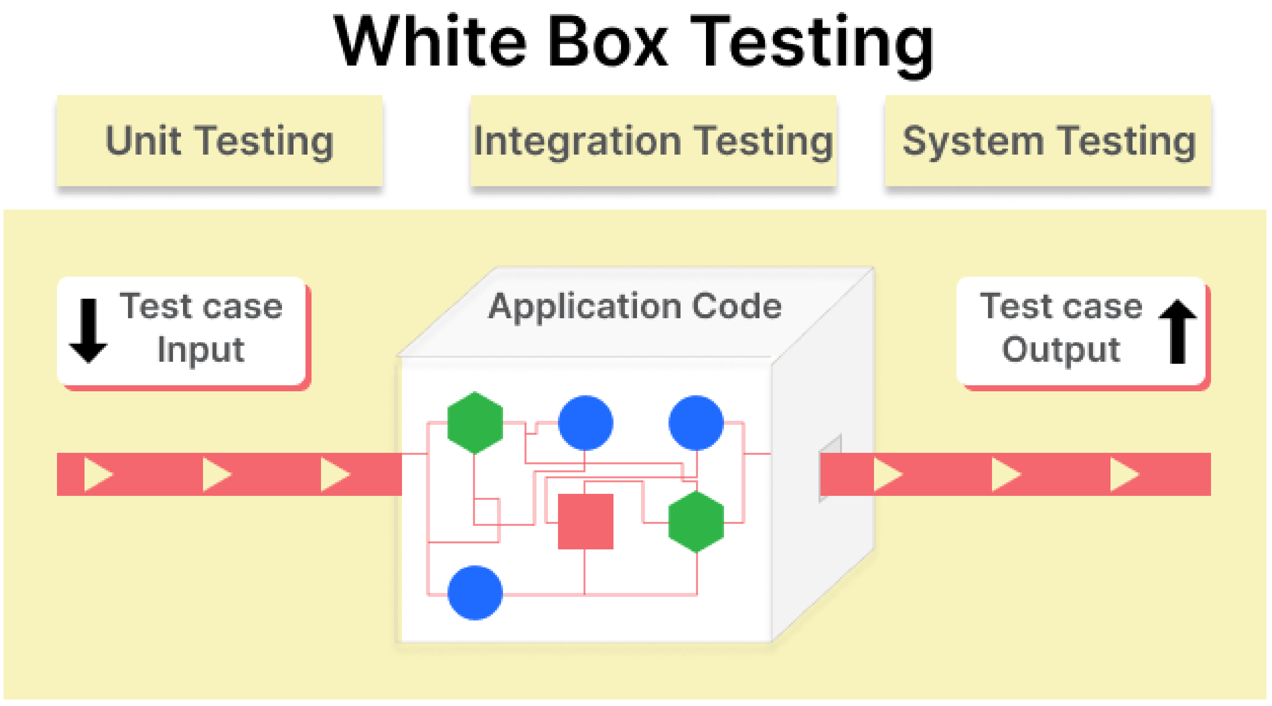


Figure 5-12:white box testing

The limitation with the white box testing is that the testing only applies to unit testing, system testing and integration testing.These are the different testing approaches that can be used for testing the application whichis developed using Microsoft Visual studio (.NET). A tester will frequently study and comprehend the application's source code as their initial step. The tester must have extensive understanding of the programming languages used in the apps they are evaluating since white box testing entails testing an application's internal workings. The tester must also be well knowledgeable about secure coding techniques. Often, one of the main goals of software testing is security. The tester should be able to identify security flaws and thwart assaults from hackers and gullible users who could willfully or accidentally introduce harmful code into the program. Black box testing and white box testing are two widely used software testing approaches for ensuring the quality and dependability of software systems. Each methodologies have distinct advantages and disadvantages, and each is best suited to particular sorts of testing scenarios. In this part, we will contrast the two strategies and emphasize the differences between them.

Developers conduct it, and the programme is then delivered to the testing team, who perform black-box testing. The primary goal of white-box testing is to validate the application's infrastructure. It is performed at a lower level since it encompasses unit and integration testing. It necessitates programming skills because it is primarily concerned with the code structure, pathways, conditions, and branches of a programme or software. The major purpose of white-box testing is to concentrate on the flow of inputs and outputs via the software while also increasing its security. Structure testing is also known as clear box testing, code-based testing, and transparent testing. It is highly suited and advised for algorithm testing.

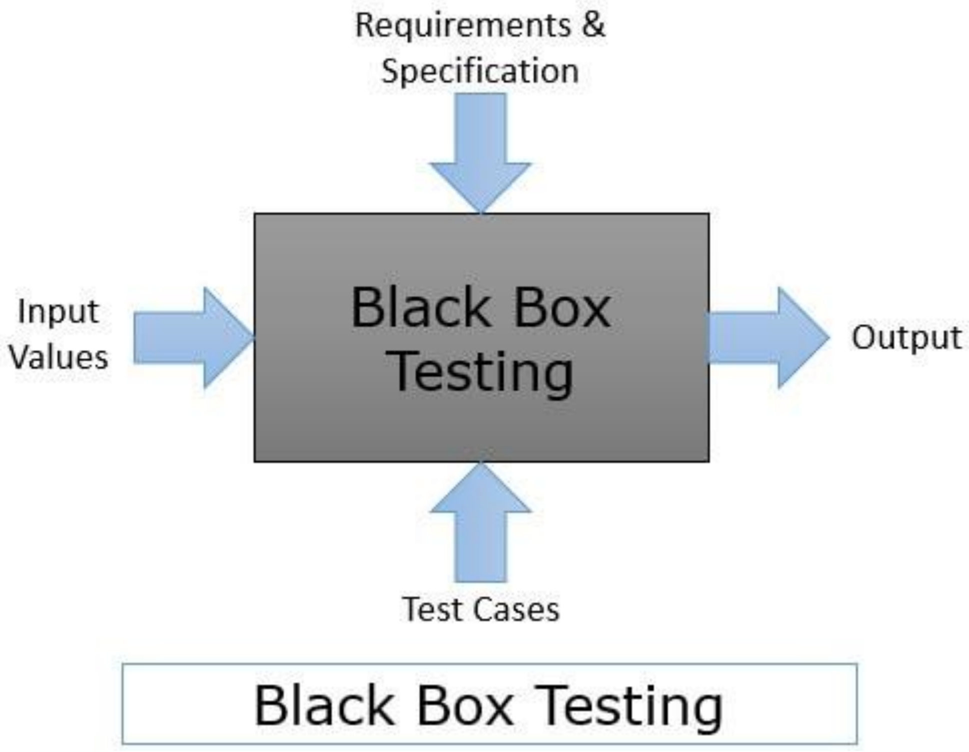


Figure 5-13:Black box testing

Black box testing is a testing approach that focuses on the software system's functional requirements. It is also known as functional testing and is often carried out from the standpoint of the user. The tester has no understanding of the software's underlying workings and merely checks the inputs and outputs to confirm that the system operates as anticipated.

The primary source of black-box testing is a customer-specified requirement specification. It is a different kind of manual testing. It is a software testing approach that assesses software functioning without knowing the software's fundamental structure or code. It does not need software programming skills. Every test cases are developed with the input and output of a certain function in mind. The test engineer examines the programme against the specifications, discovers any faults or errors, and returns it to the development team. In this approach, the tester picks a function and provides an input value to assess its functioning, and then determines whether or not the function produces the intended result. If the function returns the expected result, it passes testing; otherwise, it fails.

Black box testing is less thorough than white box and grey box testing. It takes the least amount of time of all of the testing steps. The primary goal of performing black box testing is to identify business requirements or customer requirements.In other words, black box testing is the process of ensuring that an application's functionality meets the needs of the consumer. Black-box testing is classified into three types: functional testing, non-functional testing, and regression testing.

5.6 TEST CASES

STEP	TEST CONDITION	ACTION	EXPECTED RESULT	ACTUAL RESULT
1. Embed File				
1.1	**Login module**	Click	Logged in user	Done
1.2	**Set data**	Click	Accepting data	Done
1.3	**Add water content**	Click	Successfully added	Done
1.4	**Check notifications**	Click	Ok	Done
2. Extract				
2.1	**Alerts**	Click	Received alerts	Done
2.2	**SMTP login**	Click	User login	Done
2.3	**UI check**	Click	Working flawlessly	Done

Table 5-1: Test Cases

5.7 APPENDICES

5.7.1 MAIN.dart

```
import 'package:adaptive_theme/adaptive_theme.dart';
import 'package:awesome_notifications/awesome_notifications.dart';
import 'package:drink_up/Addons/Pages/splash_screen.dart';
import 'package:flutter/material.dart';
import 'package:flutter_redux/flutter_redux.dart';
import 'package:get/get.dart';
import 'package:redux/redux.dart';
import 'Models/app_state.dart';
import 'Settings/app_settings.dart';
import 'actions/history_actions.dart';
import 'actions/settings_actions.dart';
import 'middleware/middleware.dart';
import 'reducers/app_state_reducer.dart';
import 'navigation.dart';
import 'screens/history/history_page.dart';
import 'screens/profile/profile_page.dart';
import 'screens/today/today_page.dart';
import 'styles/app_theme.dart';

// APP NAME: Drink Up
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  AwesomeNotifications().initialize('resource://drawable/ic_launcher',
[
    NotificationChannel(
      channelKey: 'scheduled_channel',
      channelName: 'Scheduled Notifications',
      channelDescription: 'Scheduled Notifications channel',
      locked: true,
      importance: NotificationImportance.High,
    )
  ]);
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  final store = Store(appReducer,
      initialState: AppState.defaultState(),
      middleware: createStoreMiddleware());

  MyApp({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return StoreProvider<AppState>(
      store: store,
      child: AdaptiveTheme(
        light: AppTheme.lightTheme,
        dark: AppTheme.darkTheme,
        initial: AdaptiveThemeMode.system,
        builder: (theme, darkTheme) {
          return GetMaterialApp(
            //named routes.
            routes: <String, WidgetBuilder>{
              "/homepage": (_) => const TodayPage(),
              "/profilepage": (_) => const ProfilePage(),
              "/historypage": (_) => const HistoryPage(),
              "/settingspage": (_) => const AppSettings(),
            },
            debugShowCheckedModeBanner: false,
            theme: theme,
            darkTheme: darkTheme,
            home: const SplashScreen(),
          );
        },
      ),
    );
  }
}

class HomeController extends StatelessWidget {
  const HomeController({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return StoreBuilder<AppState>(
      onInit: (store) {
        store.dispatch(LoadDrinkHistoryAction());
        store.dispatch(LoadAppSettingsAction());
      },
      builder: (context, store) {
        return const HomePage();
      },
    );
  }
}
```

5.7.2 NAVIGATION.dart

```
import 'package:flutter/material.dart';
import 'package:iconsax/iconsax.dart';

import 'screens/profile/profile_page.dart';
import 'util/utilities.dart';
import 'screens/history/history_page.dart';
import 'screens/today/today_page.dart';

class HomePage extends StatefulWidget {
  const HomePage({Key key}) : super(key: key);

  @override
  State<StatefulWidget> createState() {
    return _HomePageState();
  }
}

class _HomePageState extends State<HomePage> with
WidgetsBindingObserver {
  @override
  void initState() {
    super.initState();

    WidgetsBinding.instance.addObserver(this);
  }

  DateTime lastUpdated;
  @override
  void didChangeAppLifecycleState(AppLifecycleState state) {
    if (state == AppLifecycleState.resumed) {
      if (lastUpdated != null && !Utils.isToday(lastUpdated)) {
        setState(() {
          lastUpdated = DateTime.now();
        });
      }
    } else if (state == AppLifecycleState.paused) {
      lastUpdated = DateTime.now();
    }
  }

  final appBody = [
    const TodayPage(),
    const HistoryPage(),
    const ProfilePage(),
  ];

  int currentindex = 0;

  @override
  Widget build(BuildContext context) {
    @override
    final iconThemeColor = Theme.of(context).hoverColor;
    return Scaffold(
      body: SafeArea(child: appBody[currentindex]),
      bottomNavigationBar: navigationDestinations(context,
iconThemeColor),
    );
  }
```

```
          navigationDestinations(BuildContext context, Color iconThemeColor) {
            return NavigationBarTheme(
              data: NavigationBarThemeData(
                indicatorColor: Theme.of(context).highlightColor,
                labelTextStyle: MaterialStateProperty.all(
                  const TextStyle(
                    fontSize: 14,
                  ),
                ),
              ),
              child: NavigationBar(
                labelBehavior:
        NavigationDestinationLabelBehavior.onlyShowSelected,
                selectedIndex: currentindex,
                onDestinationSelected: (index) {
                  setState(() {
                    currentindex = index;
                  });
                },
                height: 55,
                destinations: [
                  NavigationDestination(
                    selectedIcon: Icon(
                      Iconsax.home,
                      semanticLabel: 'Home',
                      size: 33,
                      color: iconThemeColor,
                    ),
                    icon: Icon(
                      Iconsax.home,
                      semanticLabel: 'Home',
                      size: 33,
                      color: iconThemeColor,
                    ),
                    label: "Home",
                  ),
                  NavigationDestination(
                    selectedIcon: Icon(
                      Icons.history_rounded,
                      semanticLabel: 'History',
                      size: 29,
                      color: iconThemeColor,
                    ),
                    icon: Icon(
                      Iconsax.activity,
                      semanticLabel: 'History',
                      size: 29,
                      color: iconThemeColor,
                    ),
                    label: "History",
                  ),
                  NavigationDestination(
                    selectedIcon: Icon(
                      Iconsax.user,
                      size: 30,
                      semanticLabel: 'Profile',
                      color: iconThemeColor,
                    ),
                    icon: Icon(
                      Iconsax.user,
                      size: 30,
                      semanticLabel: 'Profile',
                      color: iconThemeColor,
                    ),
                    label: "Profile",
                  ),
                ],
              ),
            );
          }
        }

  import 'package:drink_up/main.dart';
  import 'package:flutter/material.dart';
  import 'package:flutter_test/flutter_test.dart';

  void main() {
    testWidgets('Counter increments smoke test', (WidgetTester tester)
  async {
      // Build our app and trigger a frame.
      await tester.pumpWidget(MyApp());

      // Verify that our counter starts at 0.
      expect(find.text('0'), findsOneWidget);
      expect(find.text('1'), findsNothing);

      // Tap the '+' icon and trigger a frame.
      await tester.tap(find.byIcon(Icons.add));
      await tester.pump();

      // Verify that our counter has incremented.
      expect(find.text('0'), findsNothing);
      expect(find.text('1'), findsOneWidget);
    });
  }

  import 'package:flutter/material.dart';
  import 'package:flutter_redux/flutter_redux.dart';
  import 'package:sliding_sheet/sliding_sheet.dart';
  import '../../../Models/app_state.dart';
  import '../../../Models/water/Drink.dart';
  import '../../../Settings/Widgets/reusable_widgets.dart';
  import '../../../actions/history_actions.dart';
  import '../../../managers/database/drink_history.dart';

  typedef OnDrinkAddedCallback = Function(Drink drink);

  class DrinkBottomSheet extends StatelessWidget {
    const DrinkBottomSheet({Key key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
      return StoreConnector<AppState, OnDrinkAddedCallback>(
        converter: (store) {
          return (drink) {
            var entry = DrinkHistoryEntry();
            entry.amount = drink.amount;
            entry.date = DateTime.now().millisecondsSinceEpoch;
            store.dispatch(AddDrinkToHistoryAction(entry));
          };
        },
        builder: (context, callback) {
          return GestureDetector(
            onTap: () => showSlidingBottomSheet(
              context,
              builder: (_) => SlidingSheetDialog(
                duration: const Duration(milliseconds: 500),
                cornerRadius: 16,
                snapSpec: const SnapSpec(
                  initialSnap: 0.5,
                  snappings: [0.5, 0.8],
                ),
                builder: (_, SheetState state) {
                  return drinkBottomSheet(context, callback);
                },
              ),
            ),
            child: Image.asset(
              "assets/icons/drinking-water.png",
              height: 55,
              width: 55,
            ),
          );
        },
      );
    }

    drinkBottomSheet(BuildContext context, OnDrinkAddedCallback callback)
  {
      return Material(
        child: Padding(
          padding: const EdgeInsets.all(15.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              Container(
                height: 8,
                width: MediaQuery.of(context).size.width / 8,
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(30),
                  color: Colors.blue[300],
                ),
              ),
              SizedBox(height: MediaQuery.of(context).size.height / 40),
              const Text(
                "Pick Your Drink",
                style: TextStyle(
                  fontWeight: FontWeight.w500,
                  fontSize: 22,
                ),
              ),
              SizedBox(height: MediaQuery.of(context).size.height / 55),
              waterSheet(context, callback),
              SizedBox(height: MediaQuery.of(context).size.height / 40),
              const Text(
                "Juices",
                style: TextStyle(
                  fontWeight: FontWeight.w500,
                  fontSize: 22,
                ),
              ),
              SizedBox(height: MediaQuery.of(context).size.height / 40),
              juiceSheet(context, callback),
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                  Icon(
                    Icons.arrow_drop_down_rounded,
                    color: Colors.blue[800],
                  ),
                  //SizedBox(width: 3),
                  const Text(
                    "More",
                    style: TextStyle(
                      fontWeight: FontWeight.w500,
                      fontSize: 22,
                    ),
                  ),
                  //SizedBox(width: 3),
                  Icon(
                    Icons.arrow_drop_down_rounded,
                    color: Colors.blue[800],
                  ),
                ],
              ),
              SizedBox(height: MediaQuery.of(context).size.height / 40),
              sodasAndMore(context, callback),
            ],
          ),
        ),
      );
    }
```

5.7.3 DEPENDENCIES

Fluttersdk:	flutter
cupertino_icons:	^1.0.2
flutter_redux:	^0.8.2
shared_preferences:	^2.0.13
path_provider:	^2.0.9
flutter_spinkit:	^5.1.0
adaptive_theme:	^3.1.1
fluttertoast:	^8.0.9
introduction_screen:	^3.0.0
sqflite:	^2.0.2
intl:	^0.17.0
image_picker:	*^0.8.4+10*
sliding_sheet:	^0.5.2
google_fonts:	^2.3.1
url_launcher:	^6.0.17
store_redirect:	^2.0.0
awesome_notifications:	^0.6.21
flutter_staggered_grid_view:	0.4.0
iconsax:	^0.0.8

\

5.7.4 App_Theme.dart

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

class AppTheme {
  AppTheme._();

  static final ThemeData lightTheme = ThemeData(
    brightness: Brightness.light,
    focusColor: Colors.black,
    hoverColor: Colors.black,
    cardColor: Colors.white,
    disabledColor: Colors.white,
    scaffoldBackgroundColor: Colors.white,
    primarySwatch: primaryColor,
    hintColor: const Color(0xff7fffd4),
    highlightColor: const Color(0xffa4dded),
    primaryColor: const Color(0xFF4C9BFB),
  );

  static final ThemeData darkTheme = ThemeData(
    iconTheme: const IconThemeData(color: Colors.white),
    focusColor: Colors.white,
    shadowColor: Colors.grey.shade800,
    disabledColor: const Color(0xFF0096FF),
    hoverColor: const Color(0xFF0096FF),
    appBarTheme: AppBarTheme(color: Colors.grey.shade900),
    cardColor: Colors.grey.shade900,
    highlightColor: Colors.grey.shade800,
    scaffoldBackgroundColor: darkThemeColor,
    primarySwatch: darkThemeColor,
    primaryColor: Colors.grey.shade800,
    brightness: Brightness.dark,
    backgroundColor: Colors.grey.shade900,
    hintColor: darkThemeColor,
  );
}

//Color(0xff6D28D9)
//custom material color
const MaterialColor primaryColor = MaterialColor(
  0xff7fffd4,
  <int, Color>{
    //?no applied shades, maintained one constant
    50: Color(0xff7fffd4),
    100: Color(0xff7fffd4),
    200: Color(0xff7fffd4),
    300: Color(0xff7fffd4),
    400: Color(0xff7fffd4),
    500: Color(0xff7fffd4),
    600: Color(0xff7fffd4),
    700: Color(0xff7fffd4),
    800: Color(0xff7fffd4),
    900: Color(0xff7fffd4),
  },
);

//custom typical amoled black material color
const MaterialColor darkThemeColor = MaterialColor(
  0xff000000,
  <int, Color>{
    50: Color(0xff000000),
    100: Color(0xff000000),
    200: Color(0xff000000),
    300: Color(0xff000000),
    400: Color(0xff000000),
    500: Color(0xff000000),
    600: Color(0xff000000),
    700: Color(0xff000000),
    800: Color(0xff000000),
    900: Color(0xff000000),
  },
);
```

5.7.5 Utilities.dart

```
        class Utils {
         static bool isToday(DateTime date) {
         var today = DateTime.now();

         if (date.year == today.year &&
        date.month == today.month &&
        date.day == today.day) {
         return true;
        }

        return false;
         }

  static String formatNumberWithShortcuts(double number, int
maxFractionDigits) {
    var thousandsNum = number / 1000;
    var millionNum = number / 1000000;

    if (number >= 1000 && number < 1000000) {
      return '${thousandsNum.toStringAsFixed(maxFractionDigits)}k';
    }

    if (number >= 1000000) {
      return '${millionNum.toStringAsFixed(maxFractionDigits)}M';
    }

    return number.toStringAsFixed(maxFractionDigits);
  }
}
```

CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATION

6.1 CONCLUSIONS

Drink Up is a fantastic tool that looks after your health. Worry not; "Water-Reminder" is here to aid you if you're too busy to remember to consume adequate water on a regular basis. An application that helps us track the water we need to drink and provides timely reminders to do so is called Drink Water Reminder. Users merely need to enter their gender and weight to calculate how much water they should drink each day. You may track your water history, meet your daily target, and view the relevant accomplishments. Apps for mobile devices have long been used to gauge fluid consumption. Nevertheless, they don't use additional metrics like degree of activity, urine production, or color and offer little information about the significance of staying hydrated. The creators of fitness and hydration applications should make them more thorough and educational given their rising popularity in our everyday lives.

Finally, as individuals have become more conscious of the necessity of being hydrated, water tracker applications have grown in popularity in recent years. These applications provide a simple and easy method to track water consumption, establish hydration goals, and track improvement over time. They may also send users reminders and messages to keep them on track throughout the day.

There are several water tracker applications on the market, each with its own set of features and capabilities. Some applications include additional features such as fitness tracker integration, individualised water recommendations, and social sharing options. Others may be more plain and uncomplicated, focused just on tracking water intake and issuing reminders.

Ultimately, the efficiency of water tracker applications varies according to the particular user and their personal requirements. Nonetheless, for many people, these applications may be an effective tool for improving hydration habits and general health and fitness. To get the desired outcomes, like with any software or tool, it is critical to select one that suits your individual needs and preferences and to use it regularly over time.

6.2 FUTURE RECOMMENDATIONS

By using APIs, this application may be enhanced even more by offering various health advice on the newsfeed and updating treatments. When a user enters a drug, an API may be incorporated to provide pertinent information including the medicine's purpose, adverse effects, and substitutes. In addition to API integration, the water logger portion may be improved by automatically creating a dynamic graph without requiring a user to select a date. Based on user inputs, a new graph ought to be displayed each day. Also, rather than just relying on the amount of water consumed in the future job, the sleep pattern developed should take into account the time at which the user slept. Drink up will make you stay healthy and fit.

Some future developments include

- Smart water bottle integration
- Individualized advice based on individual requirements
- Gamification features
- Integration with wearable devices
- Integration with wellness and diet apps

REFERENCES

[1] "Android Studio user guide," Android Studio, [Online]. Available: https://developer.android.com/studio/intro/index.html. .

[2] "Android Studio Architecture," Android Studio, [Online]. Available: https://developer.android.com/guide/platform/index.html.

[3] "SQLite," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/SQLite

[4] "Save Data Using SQLite," Android Studio, [Online]. Available: https://developer.android.com/training/data-storage/sqlite.html

[5] "Android Shared preferences," [Online]. Available: https://www.tutorialspoint.com/android/android_shared_preferences.html

[6] "Android Studio guide 2," Android Studio, [Online]. Available: https://developer.android.com/studio/intro2/index.html. .

[7] "Android flutter," Android Studio, [Online]. Available

[8] "TESTING," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/SQLite

[9] "types of testing," Android Studio, [Online]. Available: https://developer.android.com/training/data-storage/testing.html

[10] "Android kotlin," [Online]. Available