**MAIN PROJECT ON**

**THE DAILY BUGLE**

**A project work submitted to**

**MAHATMA GANDHI UNIVERSITY**

**KOTTAYAM**

**In partial or fulfillment of the requirement for the award of the degree in**

**BACHELOR OF COMPUTER APPLICATIONS**

****

**Submitted by**

**AFLAH SALIM – 200021095133**

**Under the supervision of**

**MS. DINAH KURIAN**

**DEPARTMENT OF COMPUTER APPLICATION**

**K.M.E.A COLLEGE OF ARTS AND SCIENCE**

**KUZHIVELIPADY**

**MARCH-2023**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**K.M.E.A COLLEGE OF ARTS AND SCIENCE**

**KUZHIVELIPADY, EDATHALA**

****

CERTIFICATION

I hereby declare that this work entitled “The Daily Bugle” is a bonafide piece of research done by Aflah Salim in partial fulfillment of the degree in Bachelor of Computer Applications in Mahatma Gandhi University under our guidance and supervision.

Ms. Dinah C Kurian Ms. Dhanya Kuriakose

(Faculty Guide) (Head of Department)

Internal Examiner External Examiner

**SYNOPSIS**

The Daily Bugle is a mobile news application built using Flutter, which is a popular framework for developing high-quality mobile apps. The application is designed to keep users up-to-date with the latest news and events. To ensure smooth data management and easy  
data synchronization, the app uses Firebase as its backend. Firebase provides a reliable and  
scalable backend solution for the application. With Firebase, the app can store, retrieve and  
update news data in real-time. This ensures that users receive the latest news as soon as it's  
available.

In addition to the core features mentioned above, the Daily Bugle app offers several other  
functionalities that enhance the user experience. For instance, the app provides a search  
functionality that allows users to quickly find news articles based on specific keywords or  
phrases. This feature helps users access news stories that are relevant to their interests. To  
ensure that the app is accessible to a wide range of users, it has been designed to be responsive  
and compatible with different screen sizes and resolutions. This makes it easy for users to  
access the app on different devices, including smartphones and tablets.

Overall, the Daily Bugle app built on Flutter and powered by Firebase offers a seamless and user-  
friendly experience to users. With its reliable and scalable backend, personalized features, and  
intuitive interface, the app is a powerful tool for staying up-to-date with the latest news and  
events.

**TABLE OF CONTENTS**

**1. INTRODUCTION……………………………...………………………………………………..………..........................1**

1.1. Project profile……………………………………………………………………………………...…………2

**2. ABOUT THE DEVELOPMENT TOOLS……………………………………….....................................................3**

2.1. Introduction to Flutter…..………………………………………………………………………………..4

2.2. Introduction to Dart…………......……………………………………………….………………………..5

2.3. Database: Google Firebase………..……………………………………….…………………………….6

2.4. Operating System...……………………...……………………………………………………………...….7

**3. SYSTEM ANALYSIS…..……………………………………………………….………………………………………….8**

3.1. Introduction…………………………………………………………………..............................................9

3.2. Existing System……………………………………………………………………………………………10

3.3. Proposed System…………………………………………………………….........................................11

3.3.1. Justification of the proposed system…………………………………......................11

3.3.2. Benefits of the proposed system……………………………………………...............12

3.3.3. Objectives of the proposed system………………………………….........................12

3.4. Feasibility Study………………………………………………………….…..........................................13

**4. SYSTEM SPECIFICATION…………………………………………………….……………………………………...17**

4.1. Hardware Specification…………………………………………………...........................................18

5.5. Data Flow Diagrams.…………....……...………………………………………….…………………….24

5.5.1. DFD Symbols ….………………………...…….…..............................................................25

5.5.2. LEVEL 0……...….………………………...…….…...............................................................26

5.5.3. ADMIN-side DFDs…………………………………...…….…...........................................27

5.5.4. USER-side DFDs ………………………………...…………………….…............................28

5.6. Entity Relationship Model....……...……………………………………...…….…………………….29

5.6.1. Entity Relationship Diagram………….…………...…………………..........................31

5.7 Database JSON Structure…....……...……………………………………...…….…………………….32

**6. SYSTEM TESTING……………………………………………………………….......................................................33**

6.1. Introduction to system testing ……………………………………………………………………...34

6.2. Unit testing ……………………………………………………………………………………………….....34

6.3. Integration testing…………………………………………………………..........................................34

6.4. Validation testing…………………………………………………………………………………………35

6.5. Alpha testing ……………………………………………………………………………………………….37

6.6. Beta testing………………………………………………………………….............................................37

6.7. Test cases………………………………………………………………………………………………….....37

**7. SYSTEM IMPLEMENTATION...……………………………………………………………………...……….…...39**

7.1. Introduction to System Implementation…...……………………………………………………40

7.2. Training…………...…….…………………………………………………………………………………....41

7.3. Conversion………..…….…………………………………………………………………………………...42

7.4. Post Implementation…………………………………………………………………………………….43

7.5. System maintenance…………………………………………………………………………………….43

**8. SYSTEM EVALUATION……………………………………………………………………………………………….44**

**9. CONCLUSION…………………………………………………………………………………………………………….46**

**10. APPENDIX……………………………………………………………………………………………………………….48**

10.1. APPENDIX A……………………………………………………………………………………………….49

10.2. APPENDIX B………………………………….……………………………………………………………63

10.2.1. Screenshots….………………………....…….…..............................................................63

**DECLARATION**

I hereby declare that the report of this project work, submitted to the Department of Computer Applications of KMEA College of Arts and Science, Kuzhivelipady, Edathala in partial fulfillment of the award of the degree of Bachelor of Computer Applications is an authentic record of original work. The report has not been submitted for the award of any degree of this university or any other university.

**AFLAH SALIM**

**Place: EDATHALA**

**DATE:**

**ACKNOWLEDGEMENT**

In the name of almighty God, I express my sincere thanks to him keeping us fit for the successful completion of the project. I express my gratitude to **Prof. K.M. Abdul Karim**, the principal of KMEA College of Arts and Science for providing me the adequate facilities and means by which I was able to complete the project work.

With immense pleasure I take this opportunity to record out sincere thanks to **Ms. Dhanya Kuriakose**, Head of the Department KMEA College of Arts and Science and **Ms. Dinah C Kurian**, Department of Computer Application for having shown keen interest in the development of this project. I especially extend my thanks to my guide **Ms. Dinah C Kurian**, Department of Computer Application in examining the draft of this project and suggestions and modifications.

Finally, I also express my gratitude to all the faculty, my family, friends, and all my well-wishers who assisted in various occasions during the project.

**AFLAH SALIM**

1. **INTRODUCTION**

**1.1 PROJECT PROFILE**

The Daily Bugle is a mobile application that allows an admin to upload news articles while users can view them. The application aims to provide a simple and user-friendly interface that makes it easy for users to navigate and read news articles.

The project scope includes developing a login and registration system for both the admin and users to access the application securely. A dashboard for the admin to upload news articles with important details such as the headline, body, and images is also part of the project scope. The newsfeed for users to view the articles uploaded by the admin will be the primary interface for users to access news articles, and it will be designed to provide a smooth and enjoyable reading experience. User profile management will also be integrated to enable users to view their preferences and manage their accounts.

Flutter, a cross-platform mobile development framework, will be used to develop the mobile application due to its ability to create beautiful and intuitive user interfaces. Firebase will be utilized for backend services like authentication, database, and push notifications, ensuring secure and reliable data storage and access.

As an addition to its ease, this application is designed in a way that it can be used by a particular organization or an institution for sharing the news to their employees or students.

**2. ABOUT THE DEVELOPMENT TOOLS**

**2.1 Introduction to Flutter**

Flutter is an open-source mobile application development framework created by Google. It was first introduced in 2017 and has quickly gained popularity among developers due to its ease of use, flexibility, and ability to create high-performance mobile apps for both Android and iOS platforms.

One of the key advantages of Flutter is its ability to create visually appealing and highly customizable user interfaces. Flutter uses a modern reactive programming model that allows for fast and flexible development, with the ability to create complex animations and custom widgets. This, in turn, makes it easier to build user-friendly apps that are both visually appealing and highly functional.

Another major advantage of Flutter is its hot-reload feature, which allows developers to see the effects of code changes in real-time. This means that developers can make changes to the app's code and immediately see the results without the need for a time-consuming rebuild or restart. This feature makes it easier to iterate and test code quickly, which can save a significant amount of development time.

Flutter also offers a wide range of pre-built widgets and tools that make it easier to develop mobile apps for different platforms. This includes widgets for input, scrolling, navigation, and more, as well as plugins for accessing device-specific features such as the camera or sensors. This makes it easier to create cross-platform apps that can run on both Android and iOS without having to write separate code for each platform.

In addition to its ease of use and flexibility, Flutter is also highly performant. It uses the Dart programming language, which is optimized for client-side development, and the Flutter framework itself is designed to minimize app start-up times and reduce the need for frequent memory allocation. This results in a smoother user experience and better app performance, even on older or less powerful devices.

In conclusion, Flutter is a powerful mobile application development framework that offers a range of features and benefits for developers. Its modern reactive programming model, hot-reload feature, and pre-built widgets make it easier to create highly customizable and visually appealing mobile apps. Additionally, its cross-platform support and high performance make it an ideal choice for developing mobile apps that need to run on both Android and iOS platforms.

**2.2 Introduction to Dart**

Dart is a modern object-oriented programming language that was developed by Google in 2011. Flutter is framework of Dart. It was designed to be a high-performance language that could be used for both client and server-side development, with a focus on simplicity, productivity, and scalability.

One of the key features of Dart is its syntax, which is easy to read and write. Dart uses familiar programming concepts like classes, interfaces, and inheritance, making it easy for developers to get started with the language. Additionally, Dart has a strong type system that provides a high degree of safety and error-checking at compile-time, reducing the risk of errors in the final code.

Another major advantage of Dart is its support for asynchronous programming. Dart has built-in support for asynchronous operations, which allows developers to write code that can handle multiple tasks at once. This is especially useful for developing web applications, where multiple requests may need to be processed simultaneously.

In conclusion, Dart is a powerful and flexible programming language that offers a range of features and benefits for developers. Its syntax, support for asynchronous programming, and modularity make it easy to develop large-scale applications. Additionally, its performance and powerful tools make it an ideal choice for developing web and mobile applications.

**2.3 Database: Google Firebase**

Google Firebase is a cloud-based platform that offers a variety of services for building and scaling web and mobile applications. One of the key components of Firebase is its NoSQL database, which is a powerful tool for storing and retrieving data in a flexible and scalable way. the features and benefits of Firebase's NoSQL database and how it differs from traditional relational databases.

Unlike traditional relational databases, NoSQL databases like Firebase use a document-oriented model for storing data. Instead of tables with predefined schemas, NoSQL databases use collections of documents that can have different fields and structures. This approach allows for more flexibility in the data model, making it easier to store and retrieve complex or unstructured data.

Firebase's NoSQL database is designed to be highly scalable and flexible, making it an ideal choice for web and mobile applications that need to handle large volumes of data. The database is built on Google's Cloud Firestore technology, which provides real-time data synchronization and automatic scaling to handle spikes in traffic. This means that developers can focus on building their applications without worrying about managing servers or scaling their database infrastructure.

In addition to its scalability and flexibility, Firebase's NoSQL database offers a number of other features that make it a powerful tool for building modern applications. For example, it supports offline data synchronization, so users can continue to use an application even when they don't have an internet connection. It also supports full-text search, which allows users to search for data based on keywords or phrases. And it offers advanced security features, such as role-based access control and SSL encryption, to protect data from unauthorized access.

One of the main benefits of Firebase's NoSQL database is its ease of use. Because it doesn't require a predefined schema or complex SQL queries, it's much simpler to work with than traditional relational databases. This makes it easier for developers to iterate quickly and adapt to changing data requirements. Additionally, Firebase's NoSQL database integrates seamlessly with other Firebase services, such as authentication and cloud messaging, making it easy to build a complete application ecosystem.

In conclusion, Firebase's NoSQL database is a powerful and flexible tool for building modern web and mobile applications. Its document-oriented model, real-time synchronization, and automatic scaling make it an ideal choice for applications that need to handle large volumes of data. Additionally, its ease of use and seamless integration with other Firebase services make it a popular choice among developers. Whether you're building a small prototype or a large-scale application, Firebase's NoSQL database offers the scalability, flexibility, and features you need to succeed.

**2.4 Operating system**

Flutter app can be used on all major operating systems, including Linux, Microsoft Windows, Android, IOS etc. Here, I have used Android 13.

**3. SYSTEM ANALYSIS**

**3.1 INTRODUCTION**

Once upon a time software development consisted of a programmer writing code to solve a problem or automate a procedure. Nowadays, systems are so big and complex that teams of architects, analysts, programmers, testers and users must work together to create the millions of lines of custom return code that drive our enterprises.

To manage this, a number of system development life cycle (SDLC) models have been created: waterfall, spiral, incremental and synchronize. The oldest of these and best known, is the waterfall: a sequence of stages in which the output of each stage become the input for the next. These stages can be characterized and divided in different ways.

* Project Planning, Feasibility Study: Establishes a high-level view of the intended
* Project and determines its goals.
* System Analysis, Requirement Definition: Analyses end-user information needs.
* System Design: Describe desired features and operations in detail.
* Implementation: The real code is written here.
* Integration and Testing: Bring all pieces together and checks for error and bugs.
* Acceptance, Installation, Deployment: The final stage of initial development.
* Maintenance: This is the least glamorous and perhaps most important step of all.

**3.2 EXISTING SYSTEM**

In order to elicit the requirements of the system and to identify the elements, inputs, outputs, subsystems, and the procedures, the existing system had to be examined and analyzed in detail. This constituted the system study. Records, logs, procedures, rules etc. were examined thoroughly. The existing system was studied involving a complete co-operation from the administrative personnel who run the system at present.

Problems with the existing manual system:

1. Platforms and technologies: Most of the existing systems are built in outdated platforms and uses outdated technologies too, thus updating the application code maybe difficult compared to the proposed system which is built using flutter.

2. User-interface: Existing systems may have a cluttered or confusing user interface that makes it challenging for users to find what they're looking for. The proposed system aims to offer a user-friendly and intuitive interface that provides a smooth and enjoyable reading experience.

3. Content management: Content management is another area where the proposed system offers an advantage over some existing systems. The proposed system includes a simple and intuitive dashboard for the admin to upload news articles with important details such as the headline, body, and images. This feature ensures that content management is easy and efficient, saving the admin's time and effort.

4. Security measures: In terms of security, some existing systems may use their own authentication and database services, which may not be as secure or reliable as the Firebase platform utilized in the proposed system.

5. Storage: The existing system may not use cloud storage, resulting in slower load times and a less visually appealing user experience.

**3.3 PROPOSED SYSTEM**

**3.3.1 Justification of the proposed system**

The proposed system is interactive, highly user friendly and designed exclusively for resident operation. Computer technology has taken a quantum leap within last 20 years. We are enduring an Information Technology revolution in all industries. There is hardly any area or any industry that is not influenced by computers.

It is therefore evident that computers have got predominant and influential role for the success of organizations whether it is a big or small. Therefore, it is high time to decide upon the strategy for its utilization, irrespective of the size of the business. As the new system is computerized, the speed and Reliability of the system is very high. All the disadvantages of the old system are solved by the new system. The new system needs lesser number of operators; it produces more accurate results and is timelier. This increases the total productivity. The use of paper files is avoided and all the data are efficiently manipulated by the system. It also reduces the space needed to store the large paper files and records.

The new system facilitates more automation of the various processes. It has a program-controlled verification process of data. Software passwords guarantee the system security nicely. There are direct entry and search capabilities, with optional hard copy facility.

The computerization of the system allows easy maintenance of the details. Large amount of data can be stored easily. Addition and updating other changes can be done easily. The information can be retrieved with high speed and accuracy. The use of GUI oriented software makes the system user friendly.

Since all work is computerized, the peering process is effortless and less time consuming. Speed, accuracy, storage capacity, versatility, automation etc. are the advantages of using a computerized system.

* + 1. **Benefits of the proposed system**

1. Easy content management: The proposed system allows only admins to upload news articles, ensuring that the quality and accuracy of the news content are maintained. The content management process is made easier through the use of a centralized database that stores all the news articles.

2. Cost-effective: The proposed system is built using Flutter and Firebase, which are both cost-effective technologies. Flutter is an open-source mobile application development framework, and Firebase provides a scalable cloud backend for the application. This makes the development and maintenance of the proposed system more cost-effective compared to other technologies.

3. Improved security: The proposed system uses Firebase Authentication to secure user authentication and data storage. This provides enhanced security compared to other authentication methods, such as storing passwords in a database.

4. Easy to maintain: The proposed system is built using Flutter and Firebase, which are both easy to maintain technologies. This makes it easier for developers to update and improve the application over time, ensuring that it remains relevant and effective for its users.

5. Faster load times and visually appealing interface: The proposed system uses cloud storage to store images and other media assets. This ensures that the application loads quickly and provides a visually appealing user experience.

**3.3.3 Objectives of proposed system**

**Menu driven and user friendly**

The system is entirely menu driven. The menu provides the facility to choose the option that the user wants. The proposed system is very easy to use and operate. The system provides adequate help for the operation. So, it is easy for anybody to operate the system.

**Time saving**

It is highly efficient and less time consuming

**System Security**

One of the important features of the proposed system is its security. The administrator or the authority can assign password to authorized users to ensure protection. The administrator can also use privilege of password. This is extremely useful while working in shared environments. With this, only authorized users can enter into the system. We can periodically change password in order to ensure extra security. By providing such security no other unauthorized people can modify the system.

**3.4 FEASIBILITY STUDY**

Feasibility study is a test of system proposal according to its workability, impact on organization, ability to meet the needs, effective use of resources. During the study, the problem definition is crystallized and aspects of the problem to be included in this system are determined. The result of the feasibility study is a formal proposal. If the proposal is accepted, we continue with the project.

The Daily Bugle is a mobile application which provides news and articles to its users.

User needs a network-based system, which will remove all the mentioned problems that, the user is facing. The user with a network-based system, will be able to reduce the bulk of paper work, provide ease of work, flexibility, fast record finding, modifying, adding, removing and generating the news.

We proposed our perception of the system, in accordance with the problems of existing system by making a full layout of the system on paper. We tallied the problems and needs of existing system with the requirements. We were further updating the layout on the basis of the redefined problems. In feasibility study phase we studied the viability in terms of:

* Cost
* Effort
* Time
* Labor
* Implementation
* Outreach
* Reception

**ECONOMIC FEASIBILITY**

Economic analysis is most frequently used method for evaluating the effectiveness of a system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs.

The proposed system is economically feasible one. We do not want to keep lot of books for storing the data. By manipulating data using computer reduces cost. We do not want lot of employees; we simply want one to operate it, Administrator.

As this system works as a computer-based system, reduces a lot of manual effort and thus manpower cost. It also introduces faith and goodwill and can be measured as an intangible benefit. As reports are generated from the computer-based system it reduces cost and time and naturally error prone.

The hardware cost of computerization has come down drastically over the past few years but for the software. The number of nodes to be used in the system can be increased and one can start off with the bare minimum.

The only possible fixed costs involved with the system would be paying for people to write the code. It is possible that faculty would be willing to write the code for free, or students would be willing to work on it as a project. There are no variable costs associated with this system. since it operates on the servers, the department does not pay anything for each use of the system. The tangible benefits will mostly be in time savings for the current administrators, as well as a simplified process for activities. The intangible benefits would be increased system involvement and decreased workload on the current administrators.

**TECHNICAL FEASIBILITY**

Technical feasibility centers on the existing computer system and to what extend it can support the proposed system. Here we need only a computer working in low speed to accomplish the task.

The project can be said to be technically feasible because there will be fewer errors. At the event of errors been found, can be debugged very well and all the bugs can be removed. Since the system uses network to implement, it is technically practical for all actors. The system can be easily hosted.

**BEHAVIORAL FEASIBILITY**

The proposed system can be easily accepted as it is very easy to understand and is very user-friendly. The organization will not be disturbed by the use of this system because, the users will be provided with prompts which will enable them to use this system very easily.

People are inherently resistant to change and computer has been known to facilitate changes. An estimate should be made of how strong the user is likely to move towards the development of computerized system. These are various levels of users in order to ensure proper authentication and authorization and security of sensitive data of the organization.

Therefore, it is understandable that the introduction of the Daily Bugle requires special efforts to educate the users in need of it. The software that is being developed is user friendly and easy to learn.

In this way, the developed software is truly efficient and can work on any circumstances, tradition, locales. Behavioral study strives on ensuring that the equilibrium of the organization and status quo in the organization neither are nor disturbed and changes are readily accepted by the users.

**4. SYSTEM SPECIFICATION**

**4.1 HARDWARE SPECIFICATIONS**

The selection of hardware is very important in the existence and proper working of any software. When selecting hardware, the size and capacity requirements are also important. Below is some of the hardware that is required by the system

* Microprocessor: 2.0 GHz OR ABOVE
* Hard Disk Space: 10GB
* RAM: 4GB or above
* Input Devices: Mouse, Keyboard
* Output Devices: Minimum 4-inch touchscreen

**4.2 SOFTWARE SPECIFICATION**

We require many different software to make the application which is in making to work efficiently. It is very important to select the appropriate software so that the software works properly. Below are the software that are required to make the new system.

* Operating system: Android 5.0 or above
* Front-end: Flutter
* Database: Firebase

**5. SYSTEM DESIGN**

**5.1 INTRODUCTION TO SYSTEM DESIGN**

System designing is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. It translates the system requirements into ways of making them operational. The design phase focuses on the detailed implementation of the system recommended in the feasibility study.

The system which is in making is developed by working on two different modules and combining them to work as a single unit. That single unit is the one which is known as the new software. We go through the different design strategies to design the system we are talking about. In the input design we decide which type of input screens are going to be used for the system in making. In the output design we decide the output screens and the reports that will be used to give the output and in the database design we decide what all tables will be required and what all fields will be there in those tables. Each of them is discussed briefly below.

System design is the solution to the creation of a new system. This phase consists of several systems. This phase focuses on the detailed implementation of the feasible system. It emphasizes on translating design specifications to performance specification. System design has two phases of development logical and physical design.

During logical design phase the analyst defines inputs (sources), outputs (destinations), databases (data stores) and procedures (data flows) all in a format that meets the user requirements. The analyst also specifies the user needs at a level that virtually determines the information flow in to and out of the system and to the data repository. Here the logical design is done through UML (Unified Modelling Language) diagrams.

The logical design is followed by physical design implementation or coding. Physical design produces the working system by defining the design specifications, which tell the programmers exactly what the system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy or display it on the screen.

**5.2 INPUT DESIGN**

The Daily Bugle application aims at creating a android application that provides news and articles to its users. Input design is the process of converting user originated inputs to a computer-based format. Input data is collected and organized into groups of similar data. Data is entered through the keyboard and also the user can use the mouse for selecting options. Errors entered by data entry operators can be controlled by input design. The goal of designing input data is to make data entry easy, logical and free from errors as possible.

All the input data are validated in the order and if any data violates any condition the user is warned by a message and asks to re-enter data. If the data satisfies all the conditions, then it is transferred to the appropriate models in the database. Inaccurate input data is the most common cause of errors in data processing. Errors entered by data entry operators can be controlled by input design. Input design is the process of converting user originated inputs to a computer-based format. The goal of designing input data is to make data entry as easy, logical and free from errors as possible.

A good system should be user friendly, interactive and well suited for user needs. Input is one of the most expensive phases of the project and may cause major problems if not designed with almost care. Input design features can ensure the reliability of the system and also determine how efficiently the user can interact with the system.

The input design is the link between the information system and the user. The activity of putting data into the computer for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling errors, avoiding delay, avoiding extra steps and keeping the process simple. The system needs the data regarding the user information, news, article type etc. The error raising method is also included in the software, which helps to raise error message while wrong entry of input is done.

**5.3 OUTPUT DESIGN**

Output design is used to provide outputs to the users of the system. Computer output is the most important direct source of information to the user. Efficient intelligible output design improves the system relationships with the user and help in decision making. The output devices to consider depends on factors such as compatibility of the devices with the system, expected print quality and number of copies needed.

The output design has been done so that the results of processing should be communicated to the user. Effective output design will improve the clarity and performance of outputs. Output is the main reason for developing the system and the basis on which they will evaluate the usefulness of the application. The output design should be efficient, intelligible so that system relationship with the end user is improved and thereby enhancing the process of decision making.

Extra care has been taken in design and creation of the reports. Reports are created using data report, as well as flex grid formats. This helps in creating category wise grouping and summarizing of data in the reports. Computer output is the most important and direct information source to the user. Output design is a process that involves designing necessary outputs in the form of reports that should be given to the users according to the requirements. Efficient, intelligible output design should improve the system’s relationship with the user and help in decision making. So, while designing output the following things are to be considered.

* Determine what information to present
* Arrange the presentation of information in an acceptable format
* Decide how to distribute the output to intended receipts

Depending on the nature and future use of output required, they can be displayed on the monitor for immediate need and for obtaining the hardcopy.

**5.4 DATABASE DESIGN**

NoSQL databases offer a flexible and scalable approach to database design, allowing for the management of large volumes of unstructured or semi-structured data. When designing a NoSQL database, there are several key considerations to keep in mind.

One of the primary considerations when designing a NoSQL database is the type of data being stored. Unlike traditional SQL databases, NoSQL databases can handle a wide variety of data types, including documents, key-value pairs, graphs, and column families. As such, it's essential to understand the nature of the data being stored and choose the appropriate NoSQL database type to handle it effectively.

Another important consideration is scalability. NoSQL databases are designed to be horizontally scalable, which means that they can handle large amounts of data by adding additional servers to the database cluster. When designing a NoSQL database, it's essential to consider the scalability needs of the system and design it accordingly to accommodate future growth.

In addition to data type and scalability, NoSQL database design also requires careful consideration of data consistency and availability. NoSQL databases often sacrifice strong consistency in favor of high availability, meaning that data may be slightly out of sync across multiple nodes in the database cluster. As such, it's crucial to design the database to ensure that data consistency and availability meet the specific needs of the system.

Finally, security is a critical consideration in NoSQL database design. NoSQL databases are susceptible to many of the same security threats as traditional SQL databases, including injection attacks, denial-of-service attacks, and unauthorized access. When designing a NoSQL database, it's essential to implement robust security measures to protect the data stored in the database.

Overall, NoSQL database design requires careful consideration of data type, scalability, data consistency and availability, and security. By keeping these key considerations in mind, developers can design NoSQL databases that are flexible, scalable, and secure, effectively managing large volumes of unstructured or semi-structured data.

**5.5 Data Flow Diagram**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years.

While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

Data flow diagrams are well suited for analysis or modeling of various types of systems in different fields. They are:

* **DFD in software engineering:** This is where data flow diagrams got their main start in the 1970s. DFDs can provide a focused approached to technical development, in which more research is done up front to get to coding.
* **DFD in business analysis:** Business analysts use DFDs to analyze existing systems and find inefficiencies. Diagramming the process can uncover steps that might otherwise be missed or not fully understood.
* **DFD in business process re-engineering:** DFDs can be used to model a better, more efficient flow of data through a business process. BPR was pioneered in the 1990s to help organizations cut operational costs, improve customer service and better compete in the market.
* **DFD in agile development:** DFDs can be used to visualize and understand business and technical requirements and plan the next steps. They can be a simple yet powerful tool for communication and collaboration to focus rapid development.
* **DFD in system structures:** Any system or process can be analyzed in progressive detail to improve it, on both a technical and non-technical basis.

**5.5.1 DFD Symbols**

The following are some DFD symbols used in the project

Rectangle: It defines a source or destination of system data.

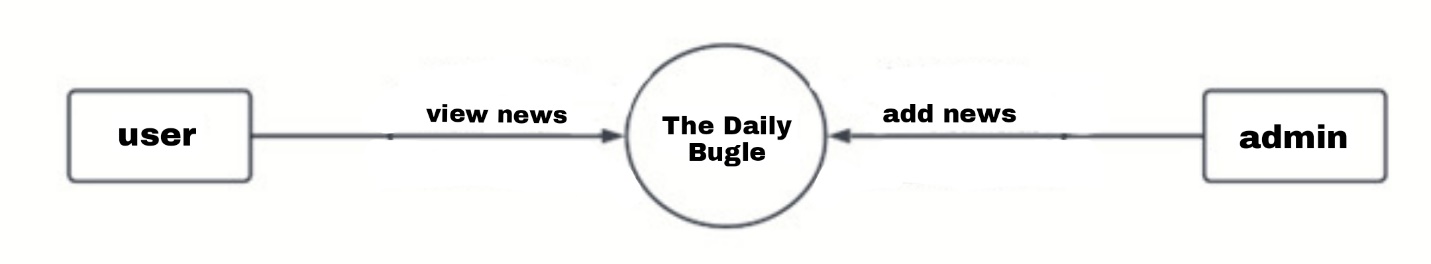
Circle: It represents a process that transforms incoming data flow into outgoing data flow.

Arrow: It defines data flow. It is a pipeline through which information flows.

Open rectangle: It is used to store data or a temporary repository of data.

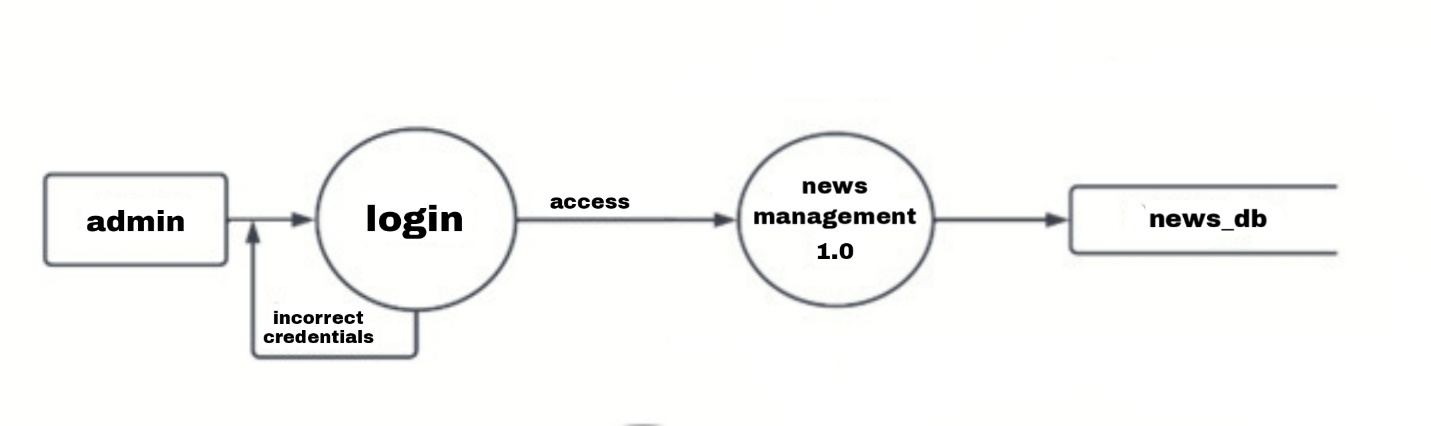
**5.5.2 LEVEL 0: The Daily Bugle**

It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

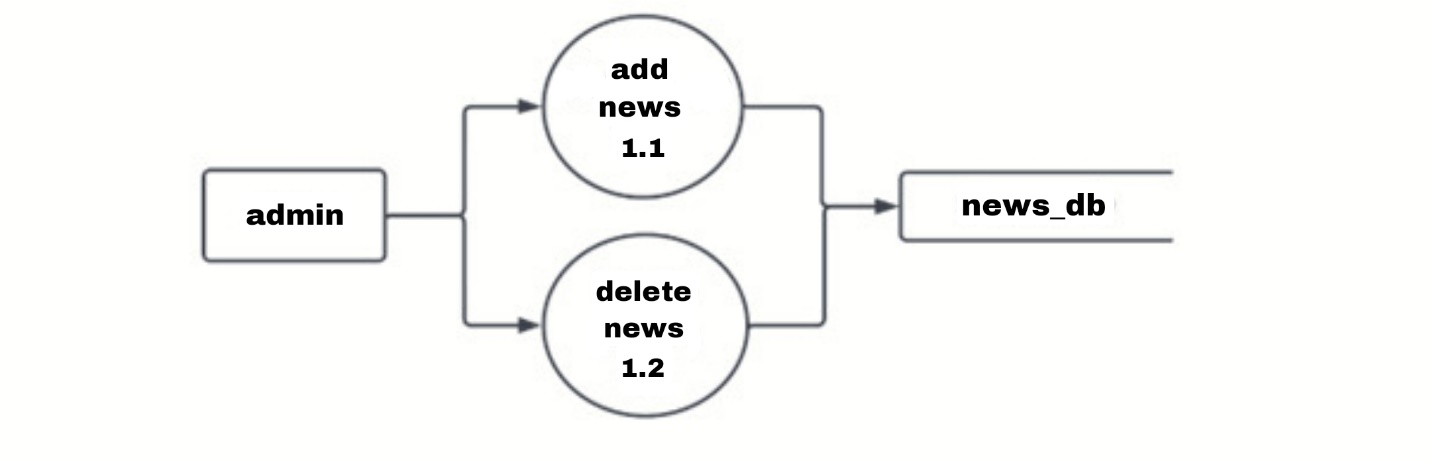


**5.5.3 ADMIN-SIDE DFDs**

**LEVEL1: ADMIN**

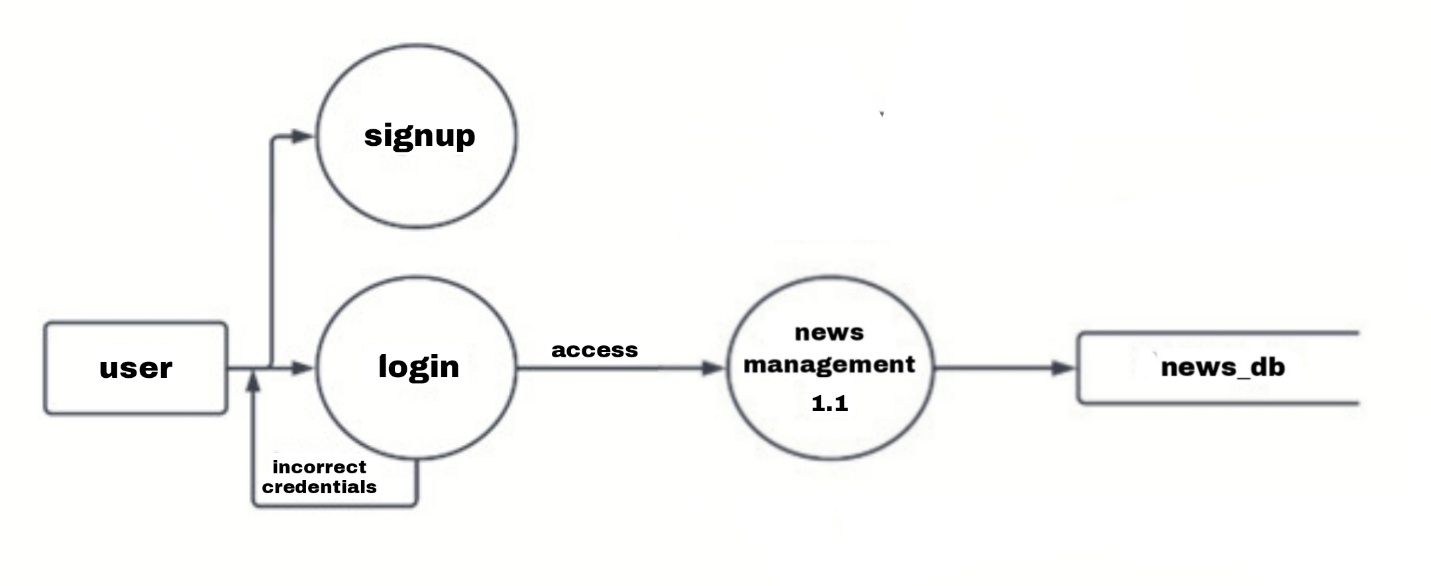
****

**LEVEL 2: ADMIN – NEWS MANAGEMENT**

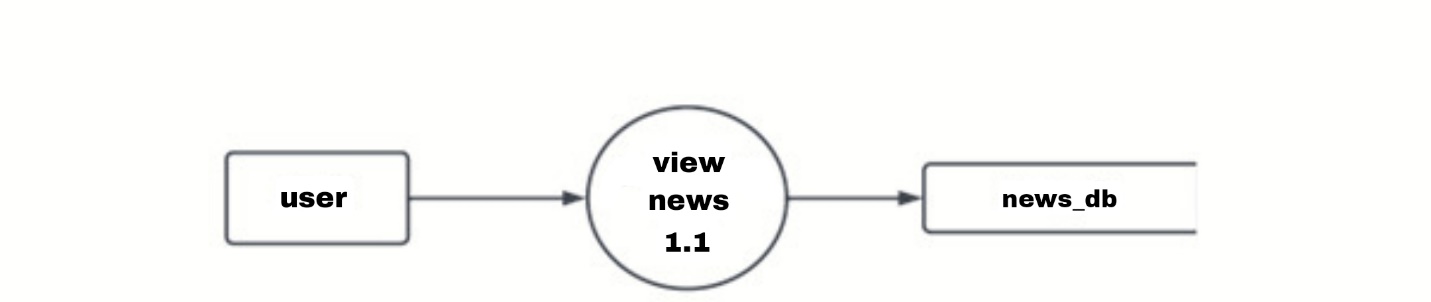
****

**5.5.4 USER-SIDE DFDs**

**LEVEL 1: USER**

****

**LEVEL 2: USER – NEWS MANAGEMENT**

****

**5.6 ENTITY RELATIONSHIP MODEL**

An entity relationship model (or ER model) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types).

In software engineering, an ER model is commonly formed to represent things a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model, that defines a data or information structure which can be implemented in a database, typically a relational database.

Entity–relationship modeling was developed for database and design by Peter Chen and published in a 1976 paper, with variants of the idea existing previously, but today it is commonly used for teaching students the basics of data base structure. Some ER models show super and subtype entities connected by generalization-specialization relationships, and an ER model can be used also in the specification of domain-specific ontologies.

**Entity**:

Entity is a representation of almost any composite information that must be understood by the software. It is represented as a rectangle.

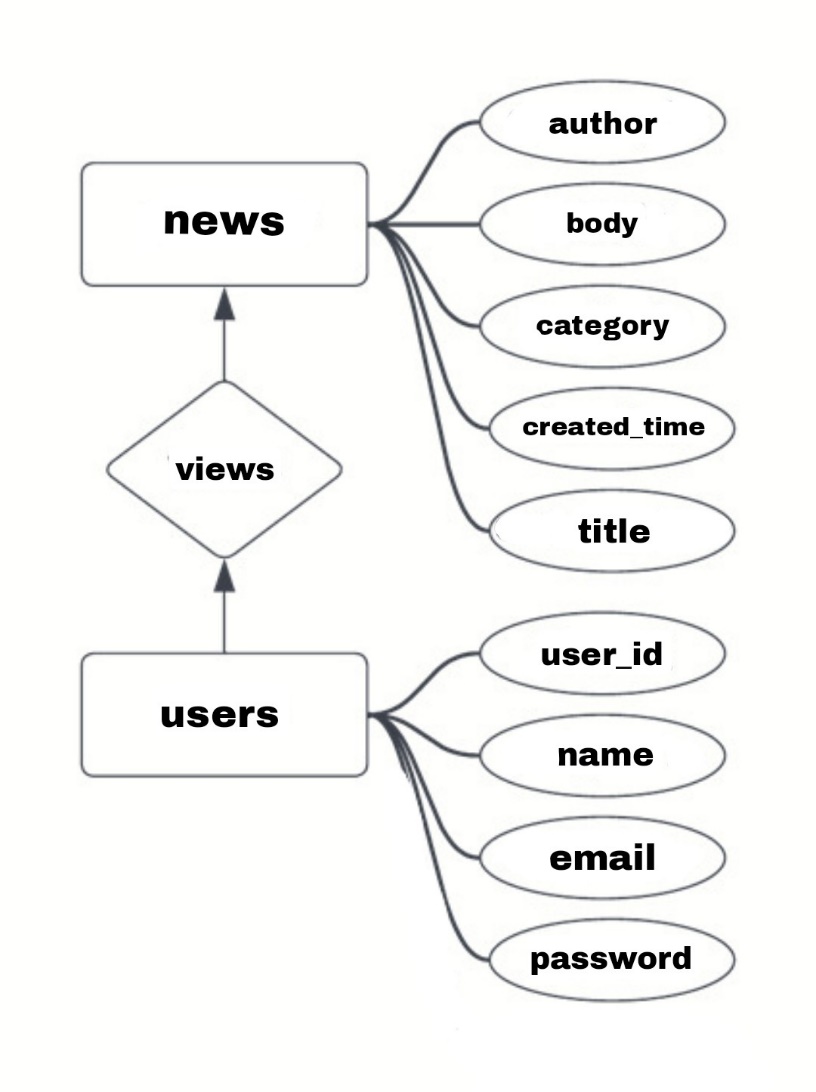
**Relationship**:

How entities act upon each other or are associated with each other. Relationships depict how entities are associated with each other. They are represented as diamond shaped.

**Attribute:**

It defines the properties of an entity. Entities may have one or many attributes that describe the entity in detailed fashion. It is represented with an oval/elliptical figure.

**5.6 ENTITY RELATIONSHIP DIAGRAM**



**5.7 DATABASE JSON STRUCTURE**

{

  "News": {

    "(String)": {  
      "author": String,  
      "body": String,

"category": String,

"created\_time": timestamp,

"title": String,  
    },  
      
  }

  "Users": {

    "(String)": {  
      "email": String,  
      "name ": String,

"password": String,

"user\_id": String,  
    },  
      
  }  
}

**6. System Testing**

**6.1 Introduction to System Testing**

System testing is a critical aspect of software quality assurance and represents the ultimate review of specification, design and ending. Testing is a process of executing a program with the intent of finding an error. No system design is ever perfect. The basic types of testing procedure are:

* Unit Testing
* Integration Testing
* Validation Testing
* Output Testing
* User Acceptance

**6.2 Unit Testing**

Unit Testing tests changes made in an existing system by a new program, as programs are in variably related to each other and interact in a total system. So, this project was designed comprising of different modules/units. Since the modules are inter-related, output of one module may act as the input to another. So, each module is tested against the entire system and the output is compared with the expected output. In The Daily Bugle app, each module is tested separately and checked for errors.

**6.3 Integration Testing**

Integration Testing is a systematic testing for constructing the program structure while at the same time conducting tests to uncover errors associated with the interface. All modules are combined in this testing phase. It checks for errors and once corrected can move to the next testing phase. In The Daily Bugle app, all the modules are tested for combined operations.

**6.4 Validation Testing**

Validation Testing is the process of evaluating the system at the end of the software development process to ensure compliance with the requirement in hand. Validation Testing is where requirements established as a part of software requirement analysis is validated against the system that has been constructed. This test provides the final assurance that the system meets all functional, behavioral and performance related requirements. Errors uncovered, are corrected.

Errors recovered where corrected prior to the completion of the project with the help of user by negotiating to establish the method of resolving deficiencies. Thus, the proposed system under consideration has been tested by using validation testing and found to be working. The developer will correct the errors that are remaining unchanged. The bugs that remain in the system after Integration Testing, will affect the proper functioning of the system. Validation Testing checks for these bugs and errors. Validation Testing is defined as Validation is success when the system functions in a manner that can be reasonably accepted by the user. Various modules that require Validation are:

**Test Plan**

It is a general document for the entire project that defines the scope, approach, schedule of testing as well as test items for entire testing process. It is performed before the actual testing commences. It should contain:

* Test Unit Specifications
* Features to be tested
* Approach of Testing
* Test Deliverables
* Schedule
* Personal Allocation

**Testing Strategy**

System is viewed as a white-box in White Box Testing, as the structure and flow of the system is visible to the tester. Testing plans are made according to the details of the system implementation, such as programming logic and styles. Test Cases are derived from the program structure. While White Box Testing is applicable at the Unit, Integration and System levels of the testing process it is typically applied to Unit Testing. So, while testing paths within a unit, it can also test paths between units during Integration Testing and between sub-system during System testing.

**System Testing**

In principle, System testing is an ongoing process throughout the project. The logical design and the physical design should be thoroughly and continually examined on paper to ensure that they will work when implemented. Thus, the System Test in implication should be a confirmation that all is correct and an opportunity to show the user that the system works. This checks all the elements of the system. The most important elements are hardware, software, people and information. There are four types of System Test. They are:

* **Recovery Testing:**

The software is allowed to fail in a number of ways and then the recovery of the

software is tested. In recovery Testing, Data Recovery and OS/Hardware Recovery is

tested.

* **Security Testing:**

Here, the safety of the system is tested. The Test Engineer himself attacks the system

to determine if the system is secure or not.

* **Stress Testing:**

This is used to check various resources of the system. Various resources that are

checked includes memory limit, OS capacity, various functions for checking interrupts.

* **Performance Testing:**

Here, we check the runtime performance of the system. This testing is conducted at

Unit level, Integration level and Implementation level. This testing employs both white-box and black-box mechanisms.

**6.5 Alpha Testing**

The Alpha Test is conducted at the developer’s site by the end-users. The system is used in a natural setting with the developer - “looking over the shoulder” of the typical users and recording errors and usage problems. Alpha Tests are conducted in a controlled environment. Alpha Testing is the testing of the system at the developer’s site by the customer/client. This testing employs:

* **Output Testing:** Output Testing includes producing the output that is generated by the system in the required format (screen format/printed format). Producing the output in the correct format is very necessary in the useful implementation of the system.
* **System Testing:** When a system is developed it is hoped that it works properly as

intended. In practice, some errors may occur. The main purpose of System Testing and Information System is to find errors and correct them. A successful test is one that finds errors.

**6.6 Beta Testing**

The Beta Test is conducted at the end-user’s site. Unlike Alpha Testing, the developer is generally not present. Therefore, the Beta Test is a live application of the system in an environment that cannot be controlled by the developer. The end-user records all problems that are encountered during Beta Testing and reports these to the developer at regular intervals. As a result of problems reported during Beta Testing, software engineers make notifications and then prepare for the release of the system to the entire customer base. Beat Tests releases the product to the end-users. He is at liberty to proceed as the requirements dictate.

**6.7 Test Cases**

The test plan represents the overall approach to the tests. In many ways, the test plan serves as the summary of the activities that will be performed. It shows how the tests will be organized and outlines all the tester’s need that must be met in order to properly carry out the tests. The test plan is especially valuable because it is not a difficult document to review, so the members of the engineering team and senior managers can inspect it. The bulk of the test planning effort is focused on creating the test cases. A test case is a descriptive of the specific interaction that a tester will have, in order to test a single behavior of the system. Test cases are very similar to Use-Cases, in that way they are a step-by-step narrative that define a specific interaction between the user and the system. However, unlike Use-Cases, they contain reference to specific features of the user interface. The test case contains actual data that must be entered into a system and the expected results that the system must generate.

A Test Case, in software engineering, is a set of conditions or variables under which a tester will determine whether an Application, Software-System or one of its features is working as it was originally established for it to do. The mechanism for determining whether a system has passed or failed such a test is called a Test Oracle. In some settings, an oracle could be a requirement or Use-Case, while in others it could be heuristic. It may take many Test Cases to determine that a system is considered sufficiently scrutinized to be released. Test Cases are often referred to as Test Scripts, particularly when written - when they are usually collected into Test-Suites.

**7. System Implementation**

**7.1 Introduction to System Implementation**

The implementation is the process of bringing a developed system into operational use and turning it over to the user. Implementation activities extend from planning through conversion from the old system to the new system. At the beginning of the development phase, a preliminary implementation plan is created to schedule and manage the many different activities that must be integrated into the plan. The implementation plan is updated throughout the development phase, culminating in a changeover plan for the operational phase.

The term Implementation has different meaning, ranging from the conversion of basic application to the replacement of the system. Implementation is used here to mean the process of converting a new or revised system design to an operational one.

The Implementation plan includes a description of all activities that must occur to implement the new system and to put it into operation. It identifies the personnel responsible for the activities and prepares a time chart for implementing the system. The Implementation Plan consists of the following steps:

* List all the files for implementation
* Identify all data required to build new files during implementation
* List all new documents and procedures that go into system

The Implementation Plan should anticipate possible programs and must be able to deal with them. The usual problems may be missing documents, mixed data formats between current files, errors in data translation and missing data.

**The Implementation Plan**

Common Implementation Management Technique is to assign responsibility for each element of implementation plan to a team. Following steps are employed:

* Test System with sample data
* Detection and correction of errors
* Meet necessary changes in system
* Check with existing System
* Installation of software and hardware utilities
* Training and involvement of user personnel

**7.2 Training**

Purpose of training is to ensure that all the personnel who are to be associated with the System possess the necessary knowledge skill. Operating, Programming and User-Personnel are trained using reference manuals as training aids

**Programming Training**

Programmers are assigned to the computer-based system at the beginning of development phase. The programmer’s reference manual helps an experienced programmer, unfamiliar with the system, about all the aspects of the program. The manual helps to:

* Understand existing programming components
* Modify existing programming components
* Write new programming components.

**User Training**

After the system is implemented, training of the user is one of the most important subtasks of the developer. Even well designed technically elegant systems can succeed or fail, because of the way they are operated. For this purpose, user manuals are prepared and handed over to the user. Here, the following activities are taken care of:

* Preparation of users and system documentation
* Conducting user training with demo and hands on
* Test run for some periods to ensure smooth switching over the system.

**7.3 Conversion**

Conversion is the process of changing from the old system to the new one. There are several methods of performing system conversion. There are four methods of system conversions:

**Parallel System**

The most secure method of conversion from old system to new system is to run both the system in parallel. This is the safest conversion approach, as if at any time the new system fails the organization can always revert to fall back to its old system.

**Direct Cutover**

Direct Cutover method converts from the old to new system abruptly. The old system is still used until a planned conversion date. There are no parallel activities. It is done so that users stop relying on the old system to continue their work. There is no system to fall back to if the new system fails. Direct Cutover conversion requires careful planning. All site preparations must be complete prior to conversion.

**Pilot Approach**

In Pilot Approach, a working version of the system is implemented on one part of the organization such as a single department. The user in this area typically knows that they are piloting a new system and changes can be made to improve the system. When the system is deemed complete, it is then installed and implemented throughout the organization. This approach provides a ground before full implementation. If not properly handled, users may develop the impression that the system has failed.

**Phase-in Approach**

This is employed when a new system cannot be implemented throughout the organization. Conversion of files, training, new equipment may take long time. Some users will begin to take advantage of the new system before the others do.

**7.4 Post Implementation**

The process of putting the developed system in actual use is called Implementation. This includes all the activities that take place to convert the old system to the new system. This can be done only after comprehensive testing and the system is found to be working successfully and as intended. The Implementation Stage follows:

* Careful Planning
* Investigation of System and its Constraints
* Designing a method to achieve Changeover
* Evaluation of changeover.

**7.5 System Maintenance**

Maintenance in an important phase. Maintenance involves a wide range of activities including correcting coding and design errors, updating documentation and upgrading user-support. It is continued until the product is re-engineered to another platform. It is done based on fixing the problem, changing the interface etc. Any system developed should be secured and protected against possible hazards. System should be maintained according to technological advancements. It ensures data integrity, data control and data security. Back up must be systematically maintained. Security measures must be deployed to protect data and code from unauthorized access.

Software Maintenance is carried out in:

* Corrective Maintenance Level
* Adaptive Maintenance Level
* Perfective Maintenance Level

**8. System Evaluation**

**Evaluation**

The project “The Daily Bugle” has been developed to meet all the requirements of the existing system and provides a systematic organizational behavior to the maintaining an alumni network. It was made operational after going through a series of phases such as Study phase, Analysis phase, Design phase, Testing, Implementation and Evaluation.

After considering various feasible solutions, the optimal solution was selected considering time and efficiency constraints.

System maintenance and Evaluation was first addressed by Meir M Lehman in 1969. Over a period of 20 years of research led to the formulation of Lehman’s Laws. Key finding of his research includes that maintenance decisions are aided by understanding what happens toa system over time. As they evolve, they grow more complex unless some actions such as code, refactoring is taken to reduce the complexity.

**9. Conclusion**

**Conclusion**

In conclusion, the development of a news application using Flutter and Firebase has proven to be a successful and efficient solution for delivering news content to users. The application's ability to allow only admins to upload news and users to view it ensures a reliable source of information for users.

The integration of Firebase has provided the application with real-time data synchronization, making it possible for users to receive updates as soon as new content is uploaded. The use of Flutter has also ensured that the application is available on both Android and iOS platforms, making it accessible to a wider range of users.

Furthermore, the application has been designed with a user-friendly interface, making it easy for users to navigate through the app and access the news content they need. The app's ability to filter news by category and search for specific news items also enhances the user experience.

Overall, the development of this news application using Flutter and Firebase has provided a reliable, efficient, and user-friendly platform for users to access news content. The project has demonstrated the capabilities of these technologies and how they can be utilized to develop high-quality applications.

**10. Appendix**

**10.1 Appendix A**

CLINIST Clinic Management System employs the following code structure:

* main.dart
* login\_screen.dart
* home\_screen.dart

**main.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:newsapp/firebase\_options.dart';

import 'package:newsapp/screens/add\_article\_screen.dart';

import 'package:newsapp/screens/article\_screen.dart';

import 'package:newsapp/screens/discover\_screen.dart';

import 'package:newsapp/screens/home\_screen.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'package:newsapp/screens/login\_screen.dart';

import 'package:newsapp/screens/profile\_screen.dart';

import 'package:newsapp/screens/register\_screen.dart';

import 'package:newsapp/services/auth\_service.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

User? firebaseUser = FirebaseAuth.instance.currentUser;

// Define a widget

bool isLogged;

// Assign widget based on availability of currentUser

if (firebaseUser != null) {

isLogged = true;

} else {

isLogged = false;

}

runApp(MyApp(

logged: isLogged,

));

}

class MyApp extends StatelessWidget {

final bool logged;

MyApp({Key? key, required this.logged}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

title: 'News Application',

theme: ThemeData(primarySwatch: Colors.grey, fontFamily: "Nunito"),

initialRoute: logged ? HomeScreen.routeName : LoginScreen.routeName,

routes: {

HomeScreen.routeName: (context) => HomeScreen(),

LoginScreen.routeName: (context) => LoginScreen(),

RegisterScreen.routeName: (context) => RegisterScreen(),

DiscoverScreen.routeName: (context) => const DiscoverScreen(),

ArticleScreen.routeName: (context) => const ArticleScreen(),

ProfileScreen.routeName: (context) => ProfileScreen(),

AddArticleScreen.routeName: (context) => const AddArticleScreen(),

},

);

}

}

**login\_screen.dart**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:newsapp/firebase\_options.dart';

import 'package:newsapp/screens/add\_article\_screen.dart';

import 'package:newsapp/screens/article\_screen.dart';

import 'package:newsapp/screens/discover\_screen.dart';

import 'package:newsapp/screens/home\_screen.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'package:newsapp/screens/login\_screen.dart';

import 'package:newsapp/screens/profile\_screen.dart';

import 'package:newsapp/screens/register\_screen.dart';

import 'package:newsapp/services/auth\_service.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

User? firebaseUser = FirebaseAuth.instance.currentUser;

// Define a widget

bool isLogged;

// Assign widget based on availability of currentUser

if (firebaseUser != null) {

isLogged = true;

} else {

isLogged = false;

}

runApp(MyApp(

logged: isLogged,

));

}

class MyApp extends StatelessWidget {

final bool logged;

MyApp({Key? key, required this.logged}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

title: 'News Application',

theme: ThemeData(primarySwatch: Colors.grey, fontFamily: "Nunito"),

initialRoute: logged ? HomeScreen.routeName : LoginScreen.routeName,

routes: {

HomeScreen.routeName: (context) => HomeScreen(),

LoginScreen.routeName: (context) => LoginScreen(),

RegisterScreen.routeName: (context) => RegisterScreen(),

DiscoverScreen.routeName: (context) => const DiscoverScreen(),

ArticleScreen.routeName: (context) => const ArticleScreen(),

ProfileScreen.routeName: (context) => ProfileScreen(),

AddArticleScreen.routeName: (context) => const AddArticleScreen(),

},

);

}

}

**home\_screen.dart**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:logger/logger.dart';

import 'package:newsapp/models/newsUser\_model.dart';

import 'package:newsapp/screens/article\_screen.dart';

import 'package:newsapp/screens/login\_screen.dart';

import 'package:newsapp/services/article\_service.dart';

import 'package:newsapp/services/auth\_service.dart';

import 'package:newsapp/widgets/bottom\_nav\_bar.dart';

import 'package:newsapp/widgets/custom\_tag.dart';

import '../models/article\_model.dart';

import '../widgets/image\_container.dart';

class HomeScreen extends StatefulWidget {

HomeScreen({Key? key}) : super(key: key);

static const routeName = '/';

@override

State<HomeScreen> createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

bool isLoaded = false;

List<Article> articles = [];

init() async {

articles = await ArticleService().getAllArticles();

setState(() {

isLoaded = true;

});

Logger().e(articles.first.tojson());

}

@override

void initState() {

init();

super.initState();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.transparent,

elevation: 0,

leading: IconButton(

onPressed: () {

AuthService().signOut();

Navigator.of(context).pushNamedAndRemoveUntil(

LoginScreen.routeName, (Route<dynamic> route) => false);

},

icon: const Icon(

Icons.logout\_rounded,

color: Colors.white,

),

),

),

bottomNavigationBar: BottomNavBar(

index: 0,

),

extendBodyBehindAppBar: true,

body: isLoaded

? ListView(

padding: EdgeInsets.zero,

children: [

\_MostRecent(article: articles.first),

const SizedBox(height: 20),

\_BreakingNews(articles: articles)

],

)

: Center(

child: CircularProgressIndicator(

color: Colors.black,

)),

);

}

}

class \_BreakingNews extends StatelessWidget {

const \_BreakingNews({

Key? key,

required this.articles,

}) : super(key: key);

final List<Article> articles;

@override

Widget build(BuildContext context) {

return Padding(

padding: const EdgeInsets.all(20.0),

child: Column(

children: [

Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

Text(

'Breaking News',

style: Theme.of(context)

.textTheme

.headlineSmall!

.copyWith(fontWeight: FontWeight.bold),

),

Text(

'More',

style: Theme.of(context).textTheme.bodyLarge,

),

],

),

const SizedBox(height: 20),

SizedBox(

height: 250,

child: ListView.builder(

scrollDirection: Axis.horizontal,

itemCount: articles.length - 1,

itemBuilder: (context, index) {

return Container(

width: MediaQuery.of(context).size.width \* 0.5,

margin: const EdgeInsets.only(right: 10),

child: InkWell(

onTap: () {

Navigator.pushNamed(context, ArticleScreen.routeName,

arguments: articles[(index + 1)]);

},

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

ImageContainer(

width: MediaQuery.of(context).size.width \* 0.5,

imageUrl: "https://picsum.photos/200/300"),

const SizedBox(height: 10),

Text(

articles[(index + 1)].title.toString(),

maxLines: 2,

style: Theme.of(context)

.textTheme

.bodyLarge!

.copyWith(

fontWeight: FontWeight.bold, height: 1.5),

),

const SizedBox(height: 5),

Text(

'${DateTime.now().difference(articles[(index + 1)].createdAt!).inHours} hours ago',

style: Theme.of(context).textTheme.bodySmall),

const SizedBox(height: 5),

Text('by ${articles[(index + 1)].author}',

style: Theme.of(context).textTheme.bodySmall),

],

),

),

);

}),

)

],

),

);

}

}

class \_MostRecent extends StatelessWidget {

const \_MostRecent({

Key? key,

required this.article,

}) : super(key: key);

final Article article;

@override

Widget build(BuildContext context) {

return ImageContainer(

height: MediaQuery.of(context).size.height \* 0.45,

width: double.infinity,

padding: const EdgeInsets.all(20.0),

imageUrl: "https://picsum.photos/200/300",

child: Column(

mainAxisAlignment: MainAxisAlignment.end,

crossAxisAlignment: CrossAxisAlignment.start,

children: [

CustomTag(backgroundColor: Colors.grey.withAlpha(150), children: [

Text(

'Most Recent',

style: Theme.of(context)

.textTheme

.bodyMedium!

.copyWith(color: Colors.white),

)

]),

const SizedBox(height: 10),

Text(

article.title!,

style: Theme.of(context).textTheme.headlineSmall!.copyWith(

fontWeight: FontWeight.bold, height: 1.25, color: Colors.white),

),

TextButton(

onPressed: () {

Navigator.pushNamed(context, ArticleScreen.routeName,

arguments: article);

},

style: TextButton.styleFrom(padding: EdgeInsets.zero),

child: Row(

children: [

Text(

'Learn More',

style: Theme.of(context)

.textTheme

.bodyLarge!

.copyWith(color: Colors.white),

),

const SizedBox(

width: 10,

),

const Icon(

Icons.arrow\_right\_alt,

color: Colors.white,

)

],

))

],

),

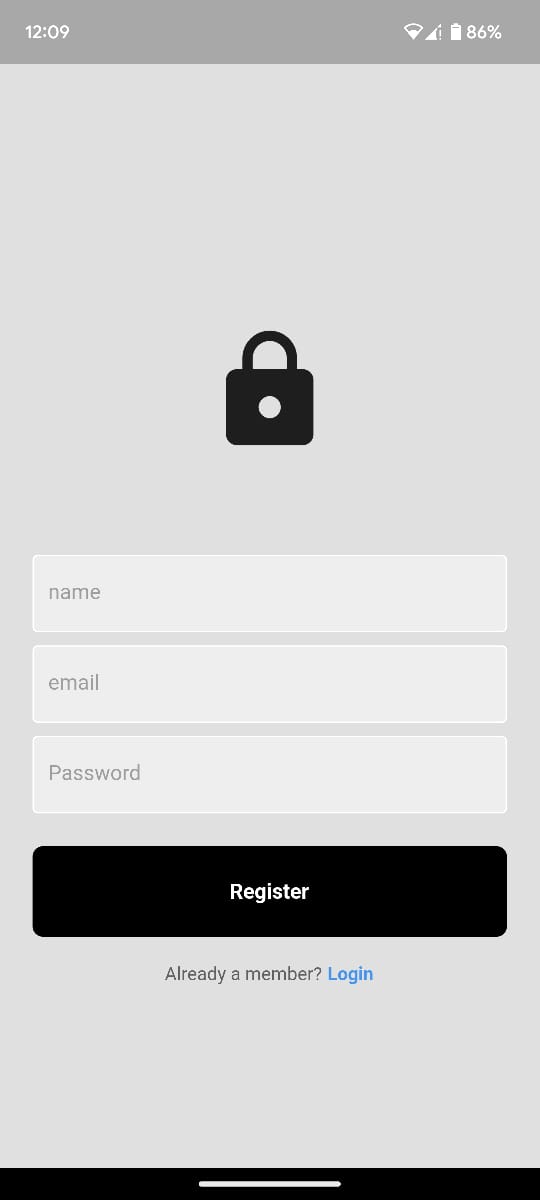
);

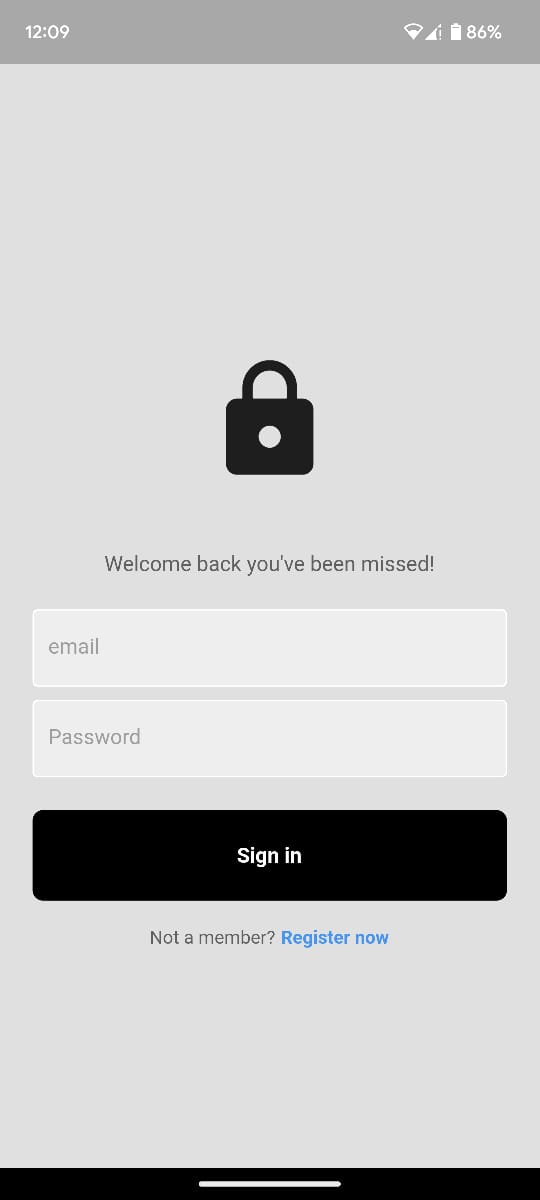
}

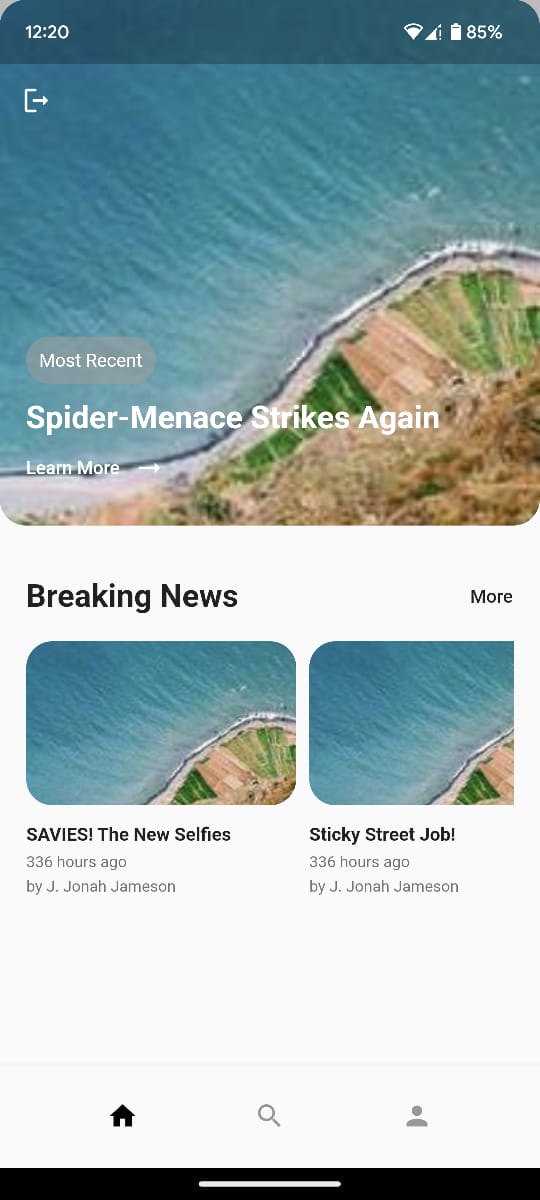
}

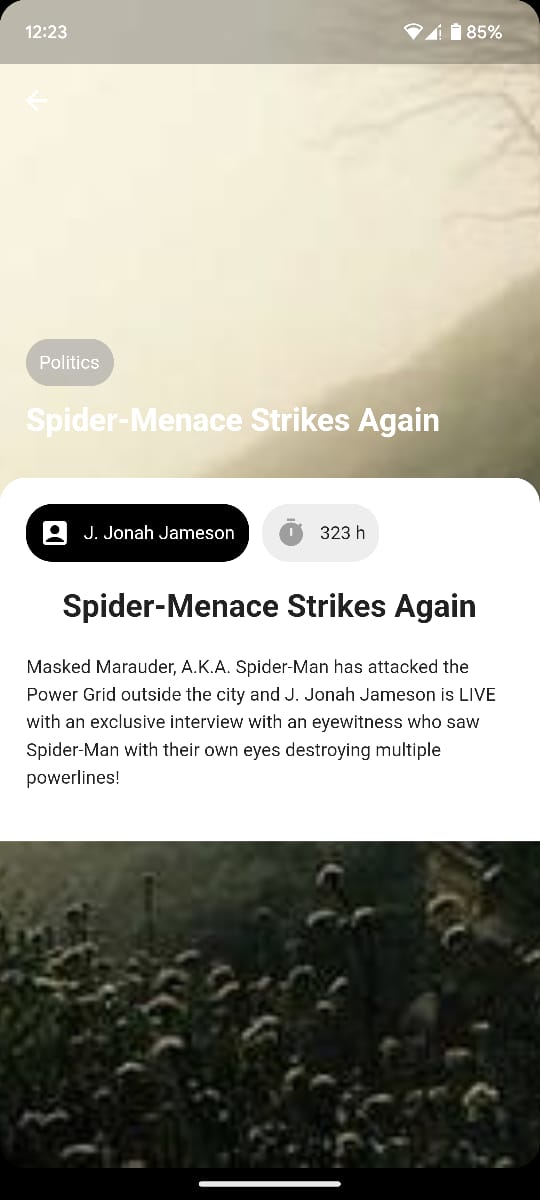
**10.2 Appendix B**

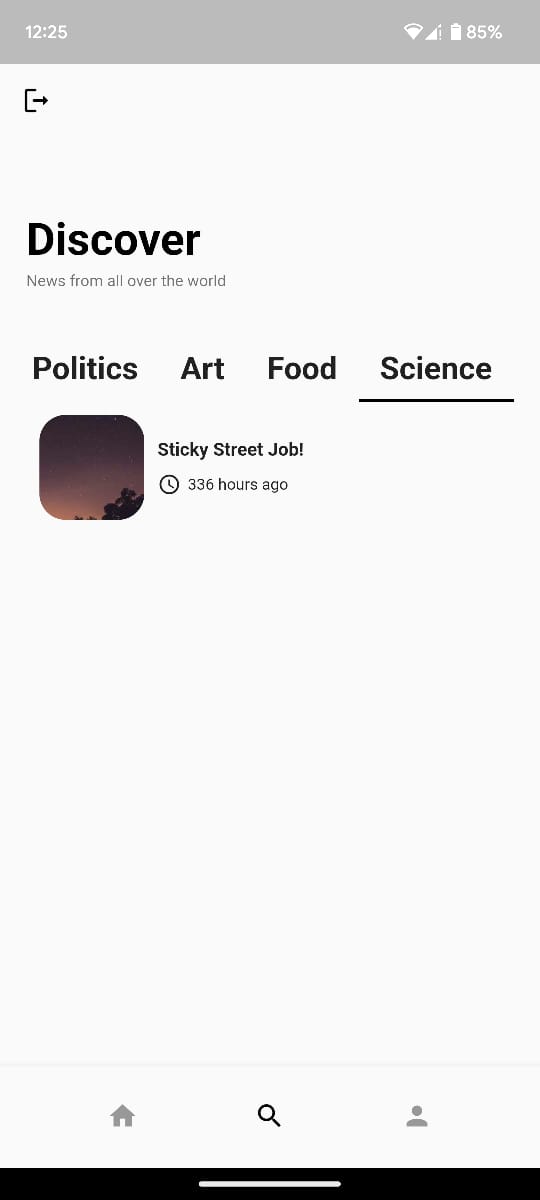
**10.2.1 Screenshots**

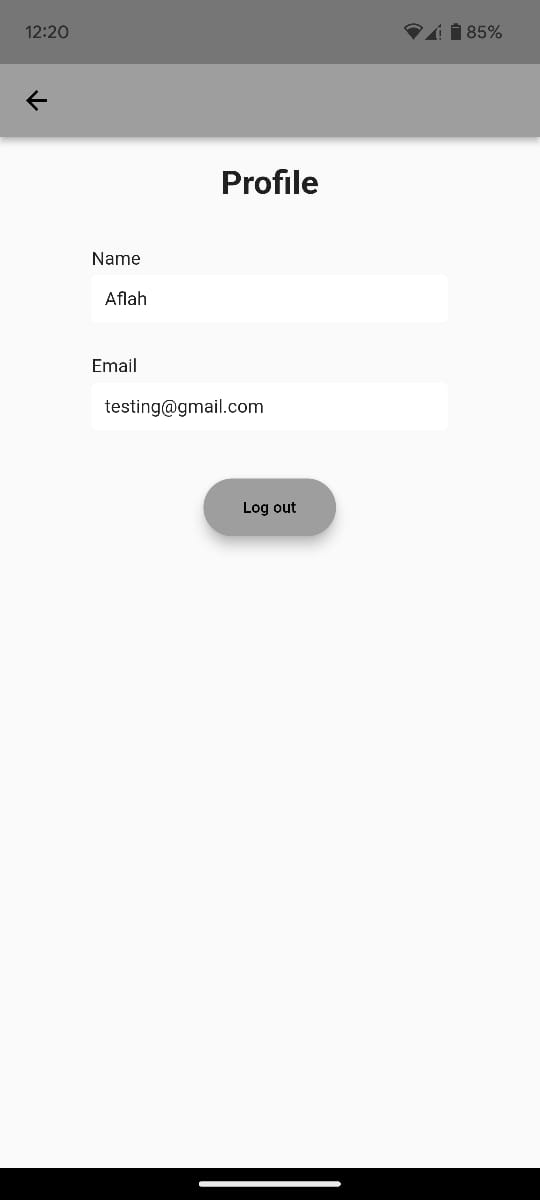
****

****

****

****

****

****