A Project Report

On

“Dyno - The Short News”

Submitted in partial fulfilment of the requirement of
University of Mumbai

For the Degree of
Bachelor of Computer Science

Submitted By
“Ayush Ravishankar Yadav”

Under the Guidance of
Prof. Ms. Pooja Kushwaha

Final Year Computer Science

Department of BSc. Computer Science Ramniranjan
Jhunjhunwala College of Arts, Science & Commerce

Affiliated to University of Mumbai.

Mumbai
(2021-2022)

Index

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my project guide **Ms. Pooja Kushwaha** as well as our HOD of Computer Science **Mrs. Anita Gaikwad** and our principal **Dr. Himanshu Dawada** who gave me the golden opportunity to do this wonderful project on the topic **Dyno**, which also helped me in doing a lot of research and I came to know about many new things I am really thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

AYUSH YADAV

Declaration

I declare that this written submission represents my own ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea, fact, data, source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

AYUSH YADAV

FEASIBILITY STUDY

There are many types of feasibility studies that are conducted to check the assessment of the practicality of a proposed plan or method. This project qualifies for three feasibility studies.

ECONOMIC FEASIBILITY

The project has shown the economic feasibility by the study of the fact that by using this software there will be a surge in the number of users who can be given service effectively and efficiently. As this project will be open to the development community after its first release the user experience will be more robust and issues can be resolved easily at a bare minimum level. Due to which the app will remain free from any kind of ads, but we are working on a subscription based model for premium news organization articles.

TECHNICAL FEASIBILITY

As this app will run on any android device so at the bare minimum stage any android phone running above android 5.0 can use this app for their daily feeds and articles they like.

OPERATIONAL FEASIBILITY

Proposed system is beneficial only if it can be turned into a system that will meet the needs of the client's operating requirements. The proposed system is operationally feasible due to the following reasons:

- Enhanced UX/UI.
- AD Free Experience.

Existing System and its Disadvantages

- These systems are more of ads and less of UX.
- More Power to User to Add Articles.
- The user experience is full of ads.

Proposed System and its Advantages

- There are many apps in the system but what makes my app different from others is that it will provide a good user experience to its user.
- The app will be lightweight so this will make the app less resource hungry and makes the task more intuitive.

- As it’s low memory usage will make it run on 1 GB ram device without any issues or lagginess.

System Requirements

Hardware requirements

- Computer or Laptop 8 GB Ram and 40 GB free space.
- Android Device.

Software requirements

- Windows 10 / Linux (**Ubuntu 16.4+, Fedora 29+, Arch**) / Mac OS / Chrome OS.
- Android OS above lollipop, android 10 recommended.
- Android Studio.
- ADB and Fastboot.

Fronted requirements

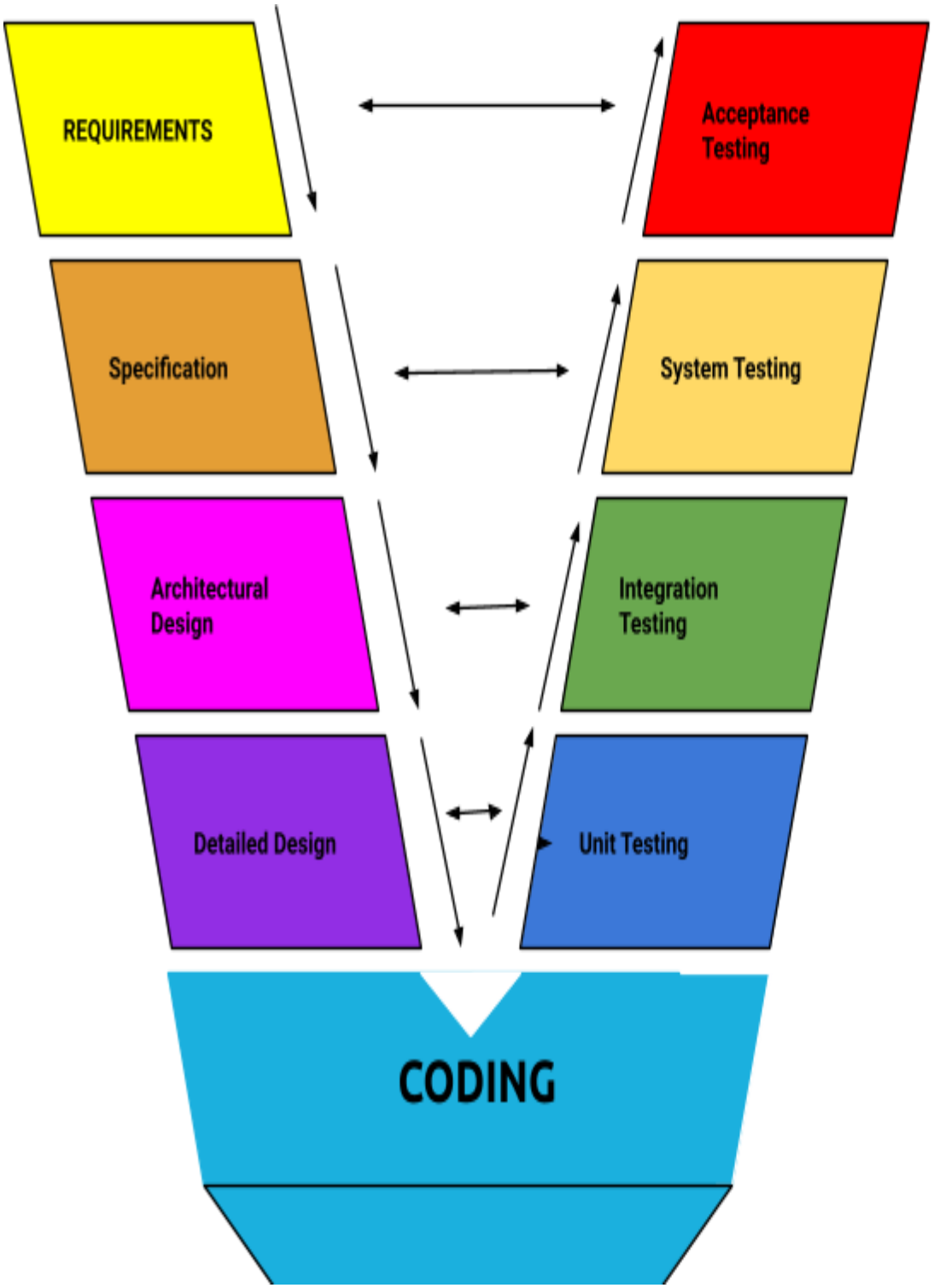
- Java, Kotlin and XML.
- JDK 11 or above.

Backend requirements

- Firebase.

- SQLite, Realtime Database.

Software Development Model



V - Model

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as the Verification and Validation model. The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

Requirements Analysis

This is the first phase in the development cycle where the product requirements are understood from the customer's perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirements. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

System Design

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system

test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

Architectural Design

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD). The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With 10 this information, integration tests can be designed and documented during this stage.

Module Design

In this phase, the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and help eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements. The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is

optimized for best performance before the final build is checked into the repository.

Validation Phase

The different Validation Phases in a V-Model are explained in detail below == >

Unit Testing

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

Integration Testing

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

System Testing

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in a user environment. Acceptance tests uncover the compatibility issues with the other

systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

System Analysis

Entity-Relationship Diagram (ERD)

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them. An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER modeling is something regarded as a complete approach to design a logical database schema. This is incorrect because the ER diagram is just an approximate description of data, constructed through a very subjective evaluation of the information collected during requirements analysis. ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers. ERD may also be more abstract, not necessarily capturing every table needed within a database, but serving to diagram the major concepts and relationships. This ERD is of the latter type, intended to present an abstract, theoretical view of the major entities and relationships needed for management of electronic resources. It may assist the database design process for an e-resource management system, but does not identify every table that would be necessary for an electronic resource management database.

Entity

An entity is an object or component of data. An entity is represented as a rectangle in an ER diagram. For example: Student and College and these two entities have many to one relationship as many student studies in a single college An entity that cannot be uniquely identified by its own attributes and relies on the relationship with another entity

is called a weak entity. The weak entity is represented by a double rectangle.

Attribute

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

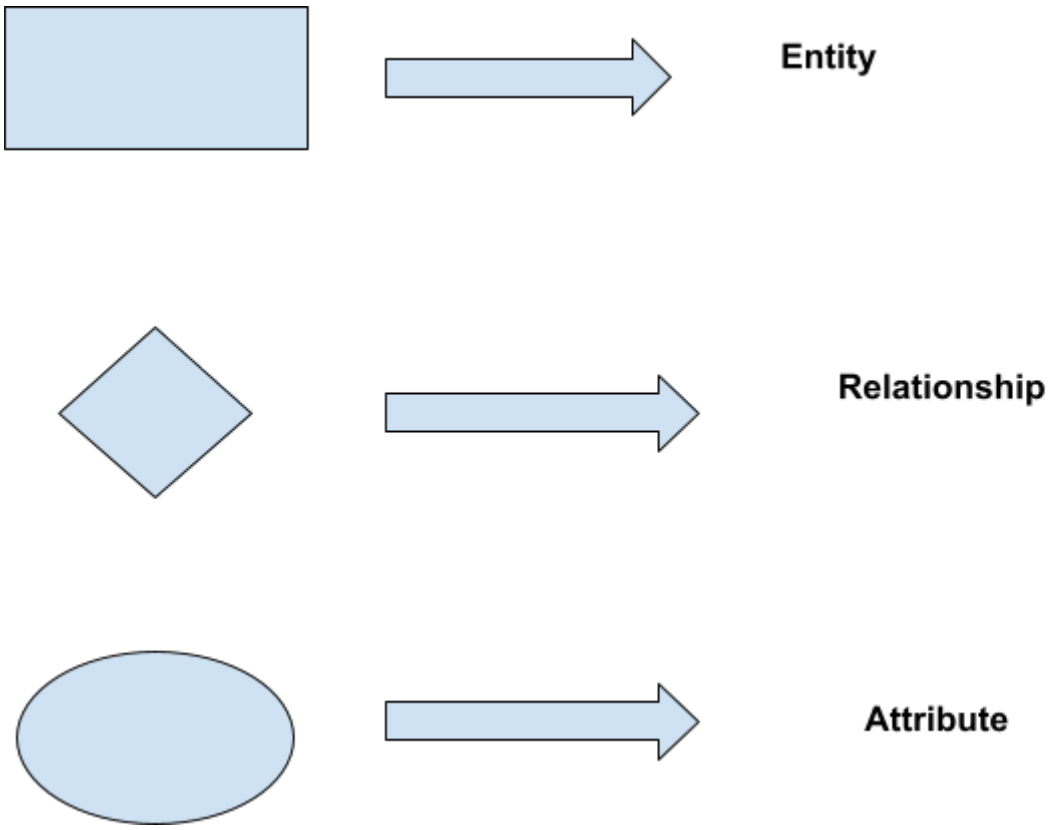
- Key attribute
- Composite attribute
- Multivalued attribute
- Derived attribute

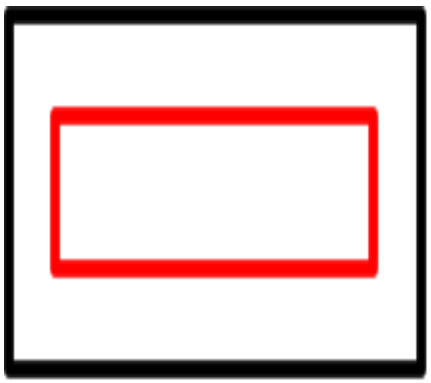
Relationship

A relationship is represented by a diamond shape in the ER diagram, it shows the relationship among entities. There are four types of relationships:

- One to One
- One to Many
- Many to One
- Many to Many

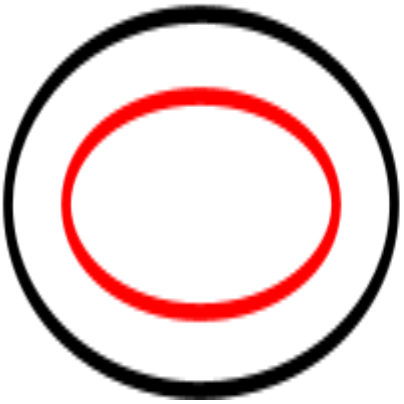
ER Diagram Symbols



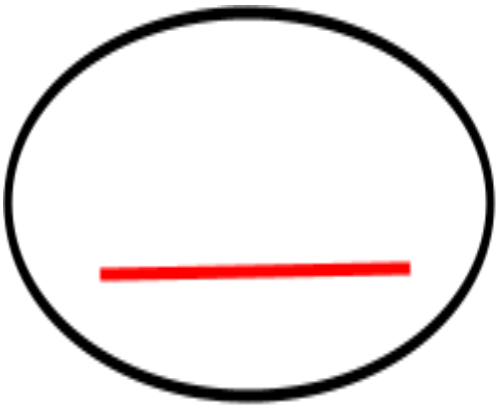


Weak Entity

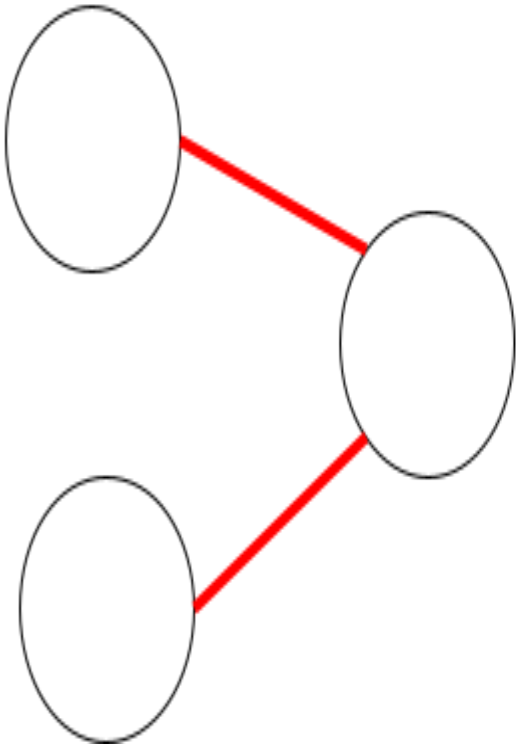
Weak
Relationship



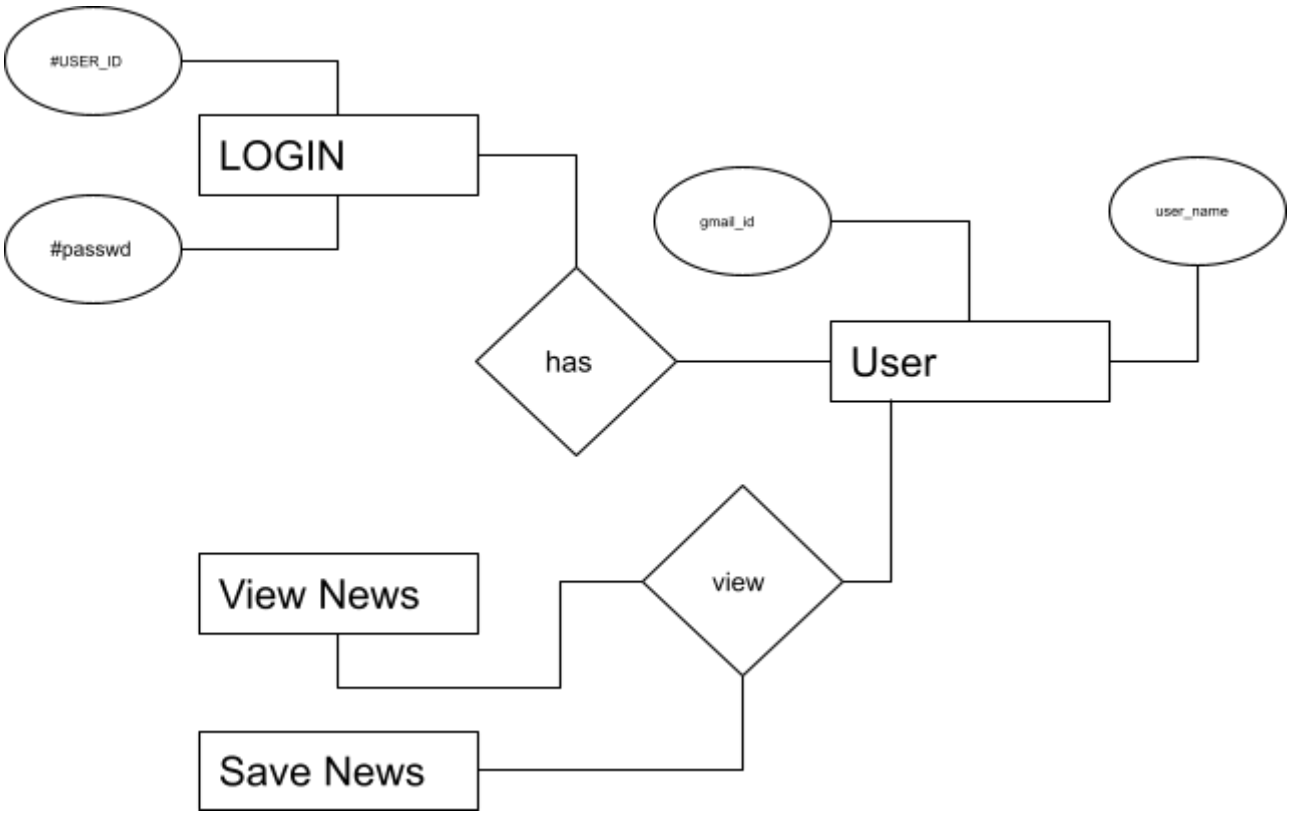
Multivalue
Attribute



Key Attribute

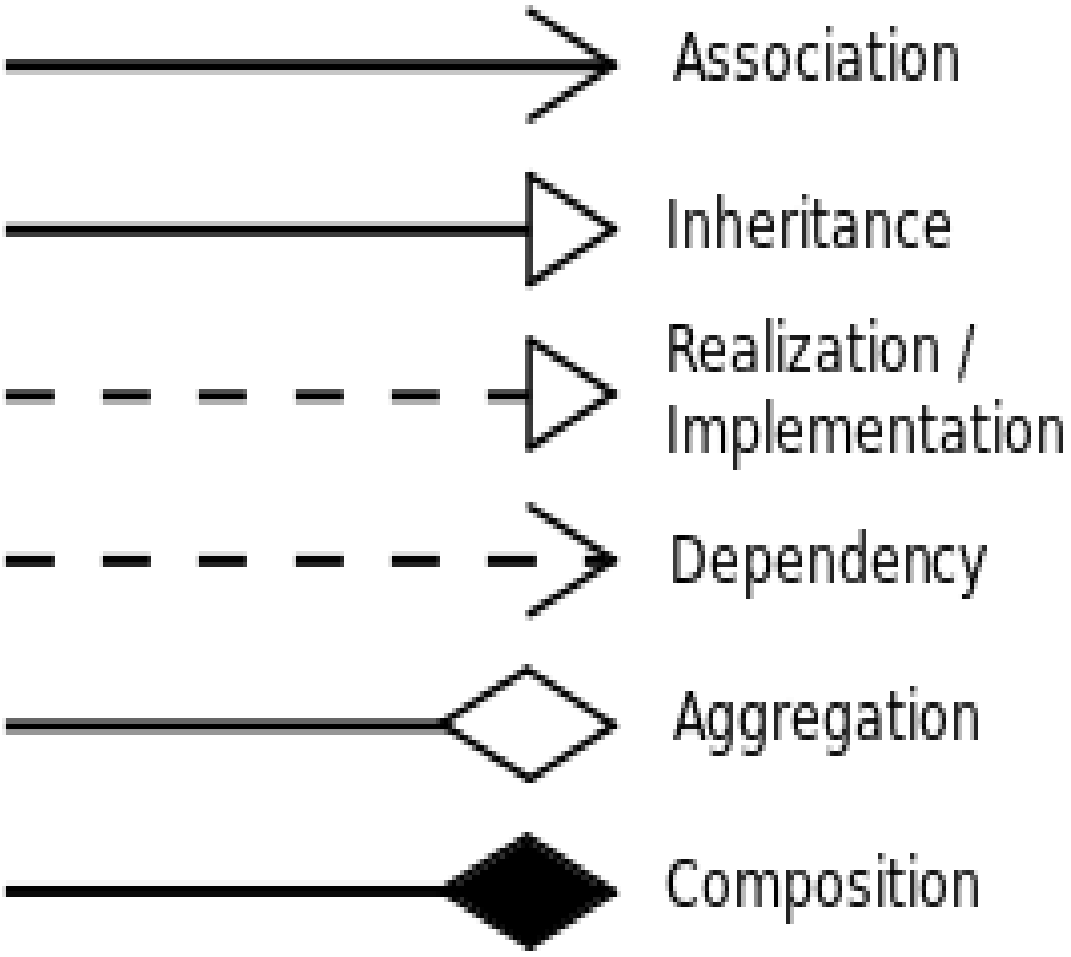


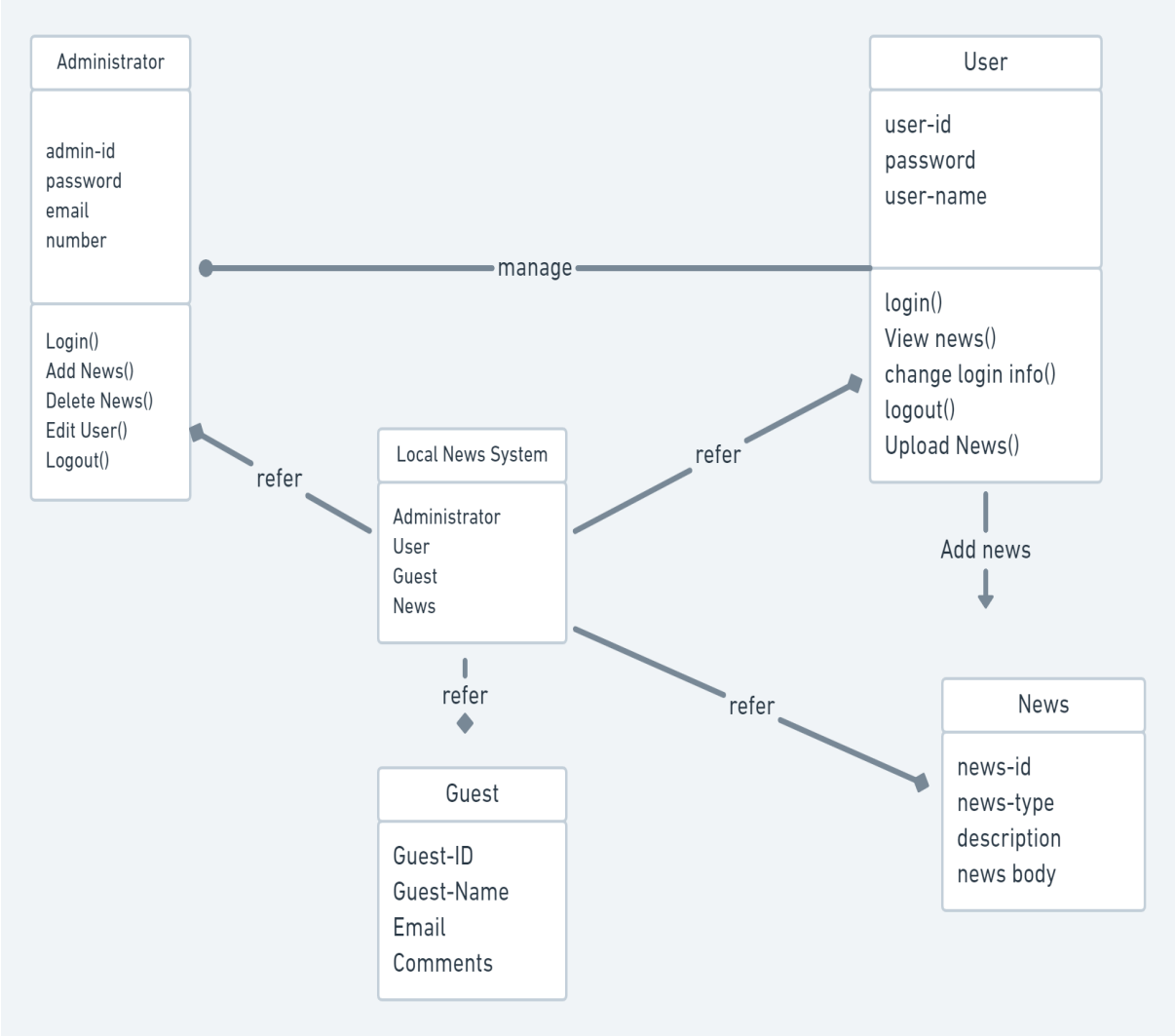
Composite Attribute


#USER_ID
LOGIN
#passwd
has
gmail_id
user_name
User
view
View News
Save News

Class Diagram

It is a model which is used to show the classes constituting a system and their interrelationship. It is based on UML. Only the important attributes and methods are shown in Class diagrams. In the initial period of analysis, the important attributes of the classes, which must be captured and the functionalities provided by the class may not be very clear. As the analysis progresses, the attributes and methods may be added. If more focus is on interrelationships of classes, then the attributes and methods may not be shown in the class diagram. The class diagram is used to identify and classify the objects which constitute a system. It also includes the important attributes of the objects which must be captured.


Association
Inheritance
Realization /
Implementation
Dependency
Aggregation
Composition

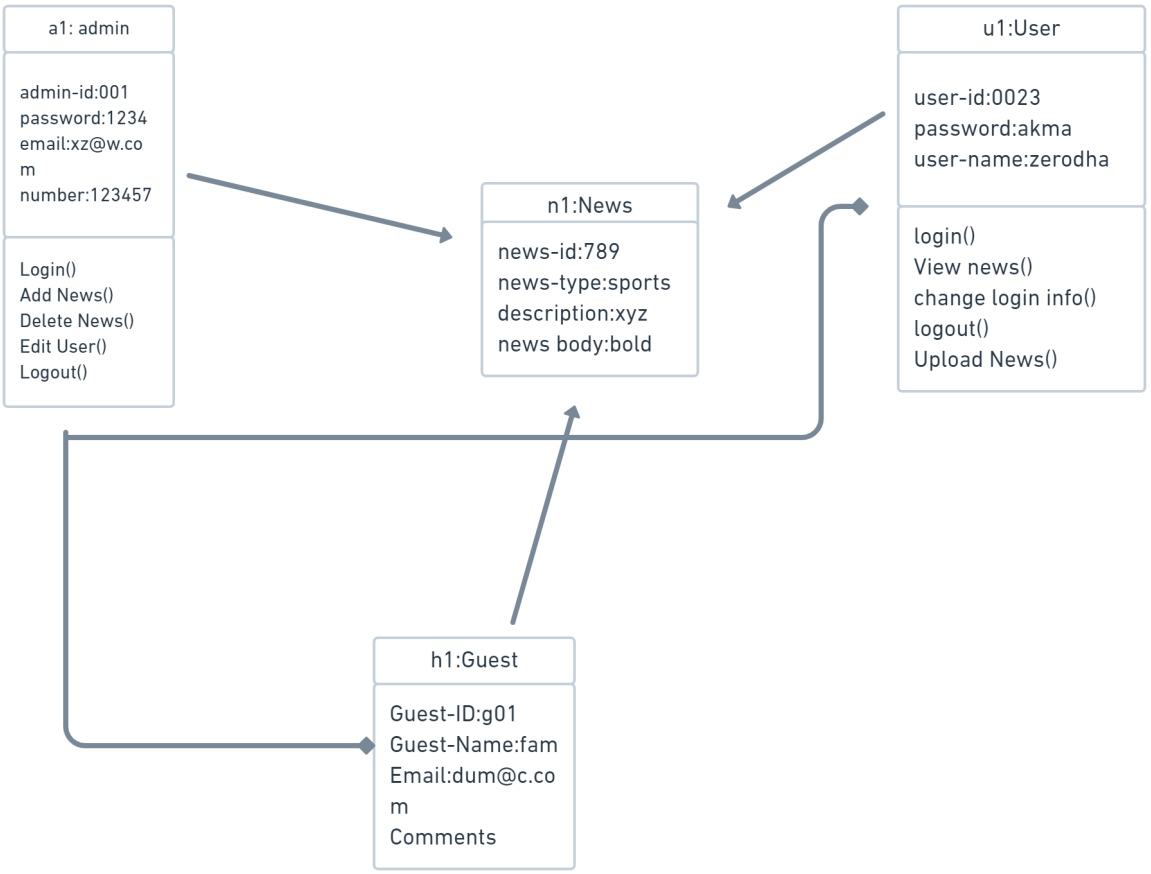

Administrator
admin-id
password
email
number
Login()
Add News()
Delete News()
Edit User()
Logout()
User
user-id
password
user-name
login()
View news()
change login info()
logout()
Upload News()
manage
refer
refer
Local News System
Administrator
User
Guest
News
Add news
refer
refer
Guest
Guest-ID
Guest-Name
Email
Comments
News
news-id
news-type
description
news body

Object Diagram

- Object is an instance of a class in a particular moment in runtime that can have its own state and data values.
- It shows a snapshot of the detailed state of a system at a point in time, thus an object diagram encompasses objects and their relationships which may be considered a special case of a class diagram or a communication diagram.

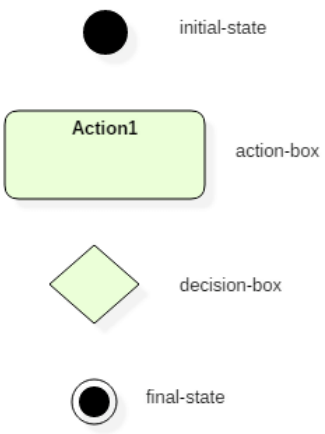
Purpose of Object Diagram

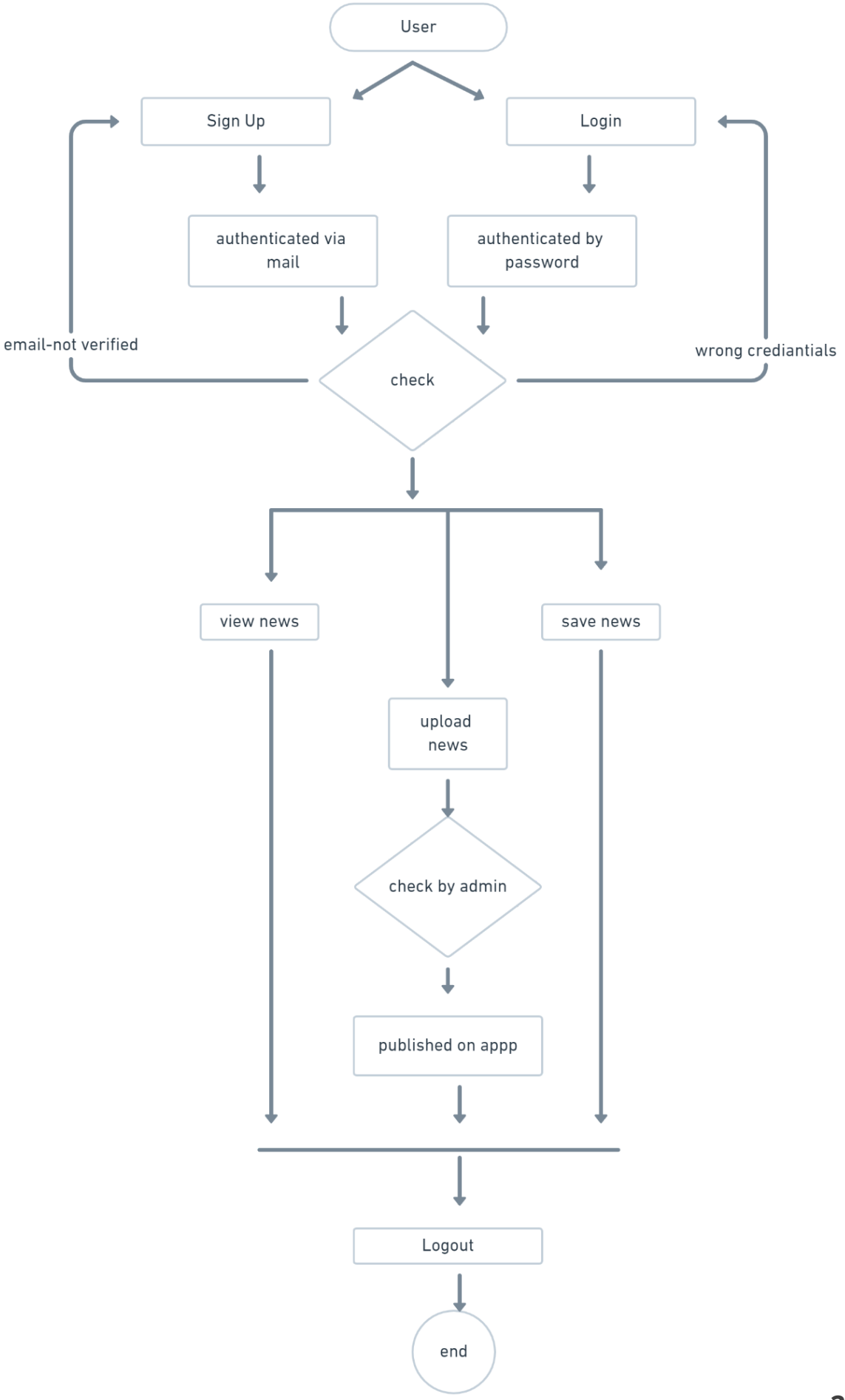
- During the analysis phase of a project, you might create a class diagram to describe the structure of a system and then create a set of object diagrams as test cases to verify the accuracy and completeness of the class diagram.
- Before you create a class diagram, you might create an object diagram to discover facts about specific model elements and their links, or to illustrate specific examples of the classifiers that are required.


a1: admin
admin-id:001
password:1234
email:xz@w.com
number:123457
Login()
Add News()
Delete News()
Edit User()
Logout()
n1:News
news-id:789
news-type:sports
description:xyz
news body:bold
u1:User
user-id:0023
password:akma
user-name:zerodha
login()
View news()
change login info()
logout()
Upload News()
h1:Guest
Guest-ID:g01
Guest-Name:fam
Email:dum@c.com
Comments

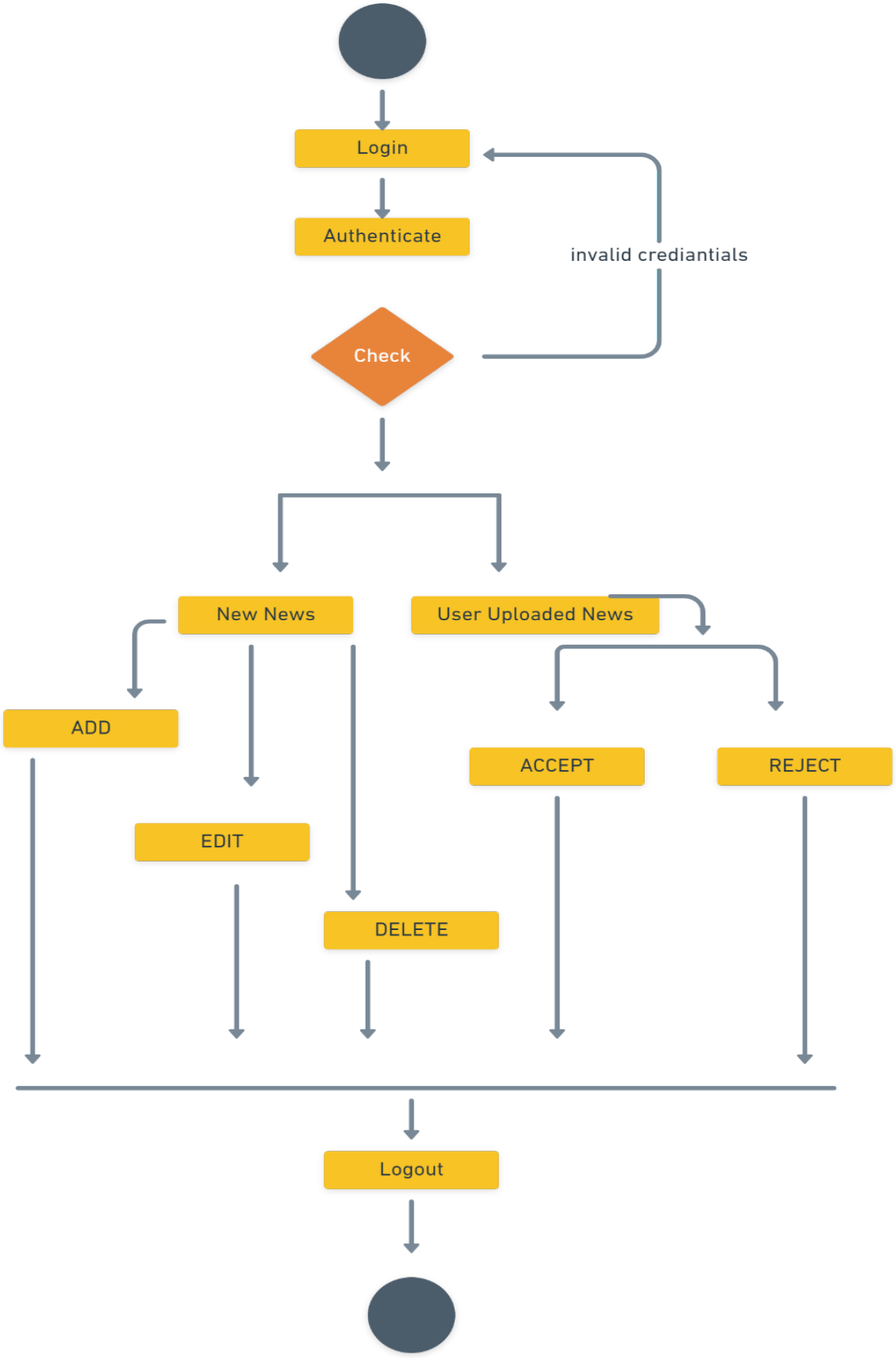
Activity Diagram

- We use **Activity Diagrams** to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case.
- We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on the condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.
- An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.
- We can depict both sequential processing and concurrent processing of activities using an activity diagram.
- They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system.




User
Sign Up
Login
authenticated via mail
authenticated by password
email-not verified
check
wrong crediantials
view news
upload news
save news
check by admin
published on appp
Logout
end

ADMIN LOGIN



ADMIN LOGOUT

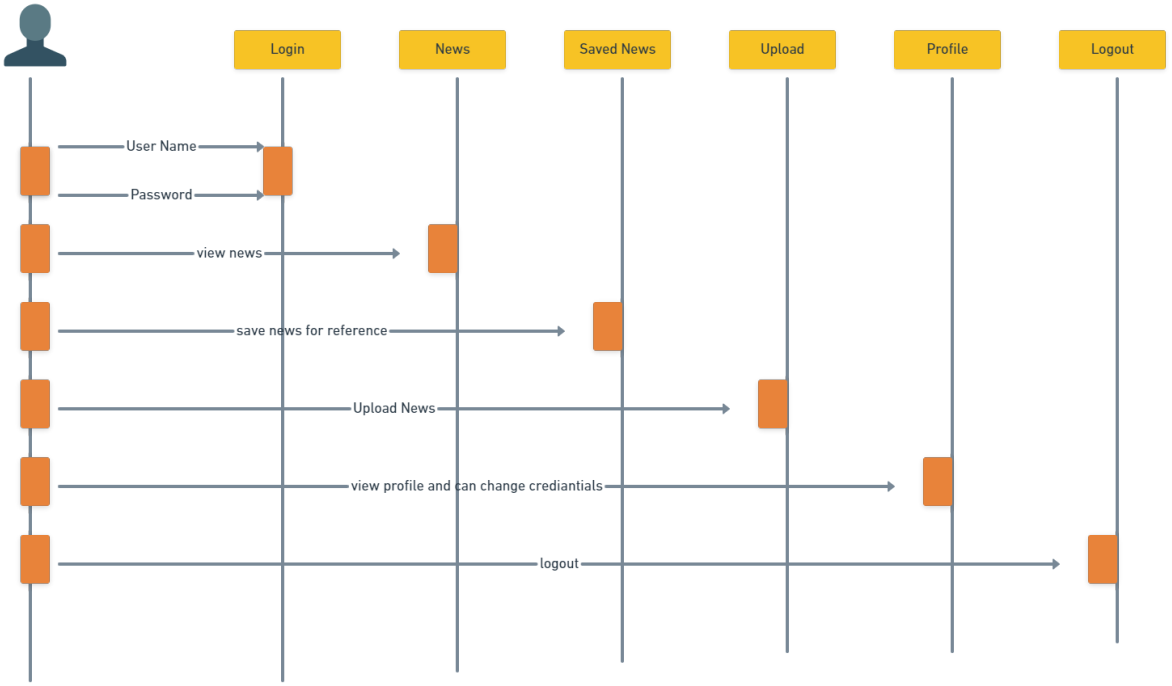
Sequence Diagram

- Sequence diagrams are a popular dynamic modeling solution in UML because they specifically focus on *lifelines*, or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends. Along with our UML diagramming tool, use this guide to learn everything there is to know about sequence diagrams in UML.
- To understand what a sequence diagram is, it's important to know the role of the Unified Modeling Language, better known as UML. UML is a modeling toolkit that guides the creation and notation of many types of diagrams, including behavior diagrams, interaction diagrams, and structure diagrams.
- A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

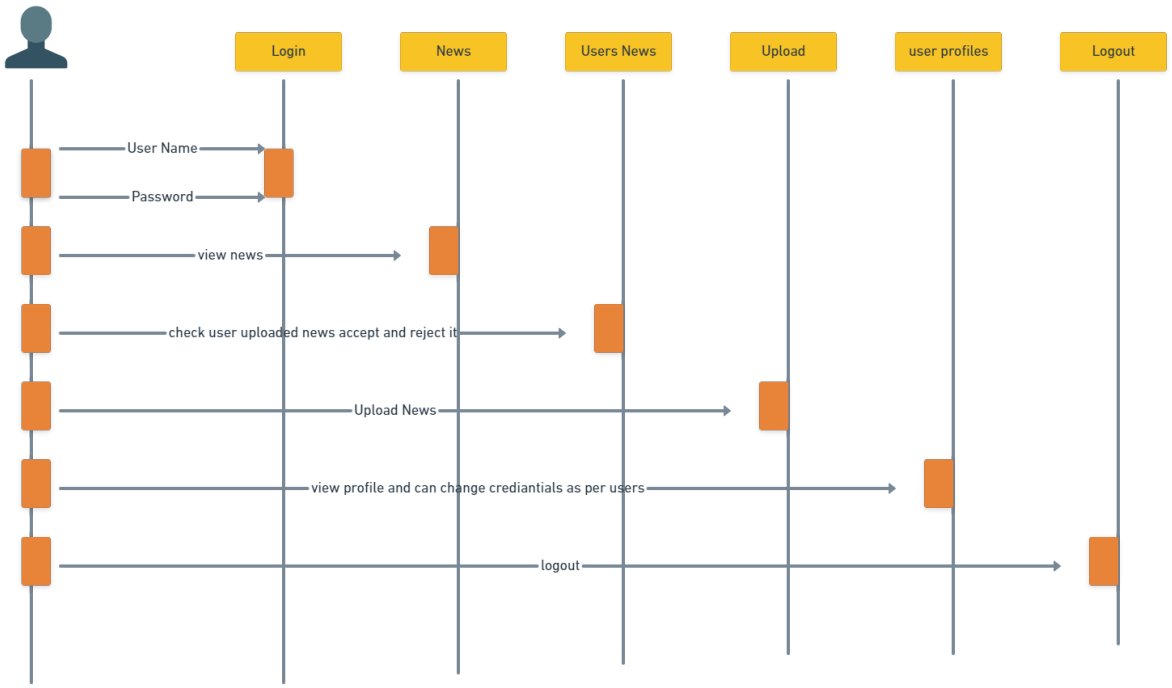
Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.

User Side


Login
News
Saved News
Upload
Profile
Logout
User Name
Password
view news
save news for reference
Upload News
view profile and can change crediantials
logout

Admin Side


Login
News
Users News
Upload
user profiles
Logout
User Name
Password
view news
check user uploaded news accept and reject it
Upload News
view profile and can change crediantials as per users
logout

Use-Case Diagram

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

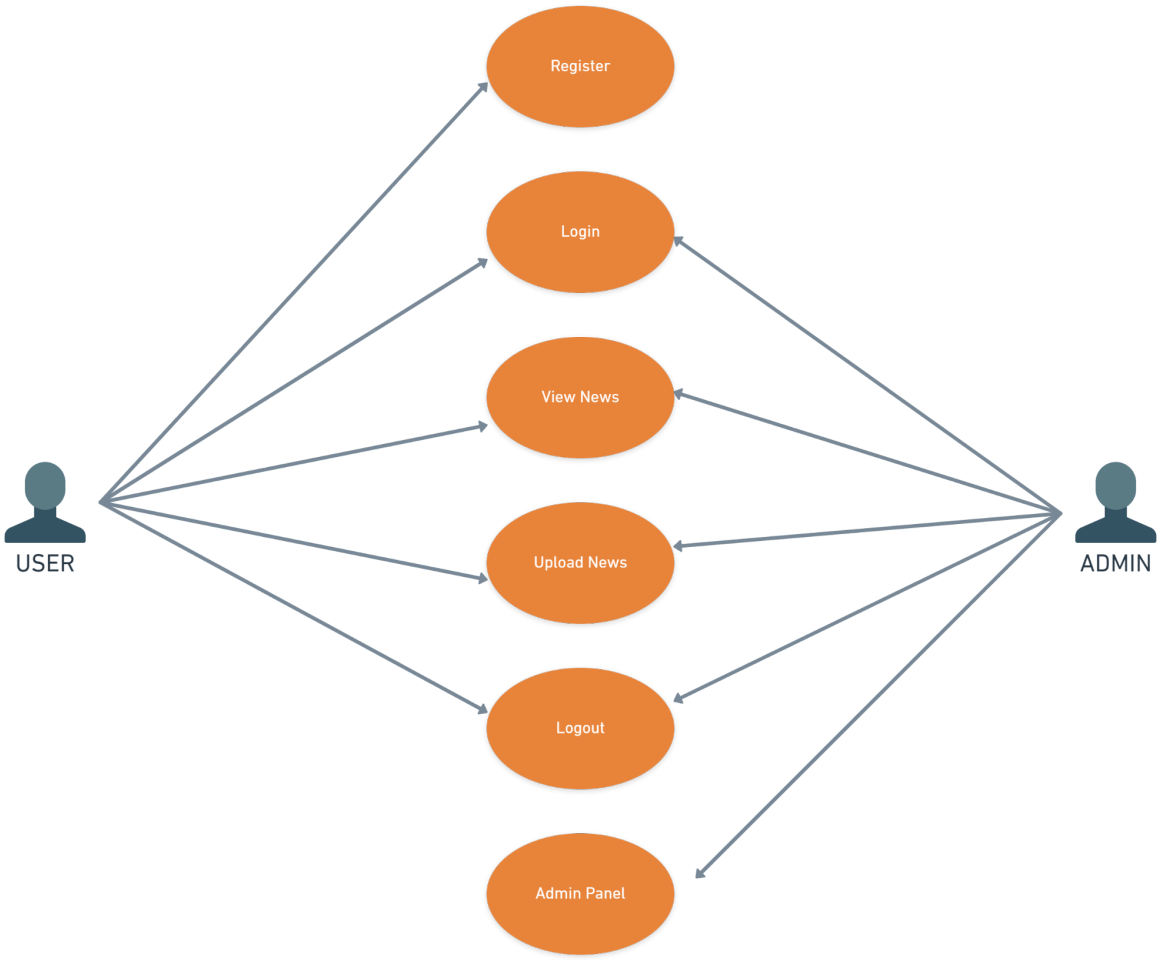
A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system

Use Case Daiagram:



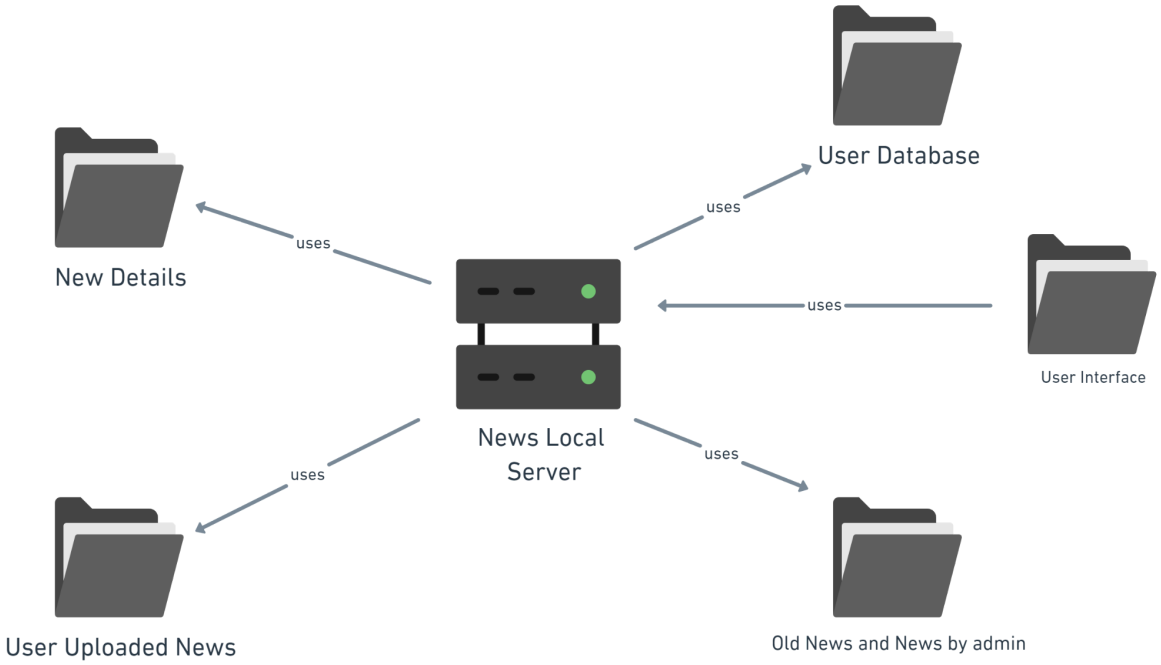
System Design

Systems design is the process of defining the architecture, product design, modules, interfaces, and data for a system to satisfy specified requirements. ... There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

Component Diagram:

- Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.
- Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.
- Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.
- Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.
- Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.
- A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

Component Diagram



Deployment Diagram

- Deployment diagrams are used to visualize the hardware processors/ nodes/ devices of a system, the links of communication between them and the placement of software files on that hardware.
- In this UML deployment diagram tutorial, we will cover what is a deployment diagram, deployment diagram notations and how to draw one. You can use one of the editable deployment diagram examples to start right away.
- If you are deploying to the cloud, you may skip UML altogether and use something like our AWS architecture templates to achieve the same purpose.
- A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.
- Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it you can understand how the system will be physically deployed on the hardware.
- Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.

Deployment Diagram

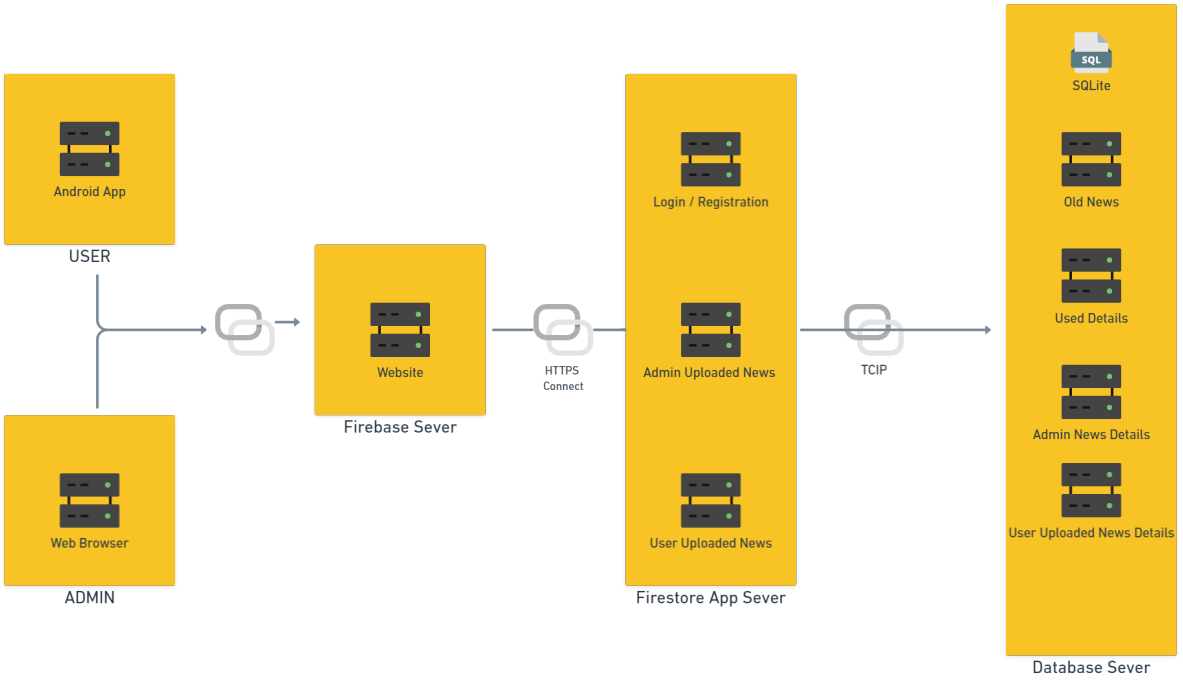


Table Diagram

User Details

Sr. No	Colums	Datatype	Description
1	Email	String	user email
2	password	String	personal password
3	User Name	String	unique for paltform
4	User News	String	news uploaded by user

News Article Table

Sr. No	Colums	Datatype	Description
1	News Head	String	heading of news
2	Images	BLOB	article image for reference
3	Author	String	name of the news writer
4	News Date	String	shows latest news to user

GNATT CHART

A Gantt chart is a commonly used graphical depiction of a project schedule. It's a type of bar chart showing the start and finish dates of a project's elements such as resources, planning and dependencies.

The chart depicts things like: task slack time or additional time for completion of a task that shouldn't delay the project; noncritical activities that may be delayed; and critical activities that must be executed on time.

Gantt charts can be used in managing projects of all sizes and types. These may include building infrastructure like dams, bridges, and highways. They may also include software development and other technologies. Project management tools, such as Microsoft Visio, Project, SharePoint, and Excel, or specialized software, such as Gantto or Matchware, can help in designing Gantt charts.

Requirement Analysis

Title	Start Date	End Date	Duration
Preliminary Investigation	15/08/2021	23/08/2021	8
Project Topic Discussion	26/08/2021	30/08/2021	4
Current System Description	01/09/2021	05/09/2021	5
Proposed System Description	06/09/2021	11/09/2021	5
Feasibility Study	12/09/2021	18/09/2021	7

System Analysis

Title	Start Darw	End Date	Duration
ER Diagram	20/09/2021	21/09/2021	1
Class Diagram	22/09/2021	24/09/2021	2
Object Diagram	25/09/2021	27/09/2021	2
Activity Diagram	28/09/2021	30/09/2021	2
Sequence Diagram	01/10/2021	03/10/2021	3
Use Case Diagram	04/10/2021	06/10/2021	2
Component Diagram	07/10/2021	09/10/2021	2
Deployment Diagram	10/10/2021	12/10/2021	2
System Design	13/10/2021	15/10/2021	3

Coding And Designing

1. XML Design
2. JAVA and Kotlin For Logic
3. Firbase Realtime Database for Storing Data.

1.XMl Design

1.Dyno Admin & Dyno User

A. Admin (MainActivity.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <androidx.appcompat.widget.Toolbar
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:background="@color/wheat">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/title"
            android:textStyle="bold"
            android:textColor="@color/black"
            android:textSize="20sp"/>

        <ImageView
            android:id="@+id/postbutton"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_baseline_add_circle_outline_24"
            android:layout_gravity="right"
            android:paddingEnd="20dp"/>

    </androidx.appcompat.widget.Toolbar>


    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/listview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/feed"/>




</androidx.appcompat.widget.LinearLayoutCompat>
```

B. Admin Feed.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/list_item"
    android:layout_margin="2dp">

    <androidx.appcompat.widget.LinearLayoutCompat
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="8dp"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/newsimage"
```

```
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:src="@drawable/ic_launcher_background"
            android:scaleType="centerCrop"/>

        <TextView
            android:id="@+id/tvtext"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/headline"
            android:textStyle="bold"
            android:layout_margin="5dp"
            android:textColor="@color/black"
            android:textSize="20sp" />
        <TextView
            android:id="@+id/tvbody"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/body"
            android:layout_margin="5dp"
            android:textColor="@color/black"/>

        <Button
            android:id="@+id/delete"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Delete"
            android:layout_gravity="right"/>
    </androidx.appcompat.widget.LinearLayoutCompat>

</androidx.cardview.widget.CardView>
```

C. Admin Post.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".PostAcitvity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fillViewport="true">

        <androidx.appcompat.widget.LinearLayoutCompat
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

              <EditText
                  android:id="@+id/headlinetext"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:layout_margin="12dp"
                  android:gravity="top"
                  android:textSize="22sp"
                  android:padding="8dp"
                  android:inputType="text"
                  android:hint="I am the headline"
                  android:background="@drawable/round_text"/>

            <EditText
                android:id="@+id/description"
                android:layout_width="match_parent"
                android:layout_height="120dp"
                android:maxHeight="150dp"
                android:textSize="20sp"
```

```
            android:layout_margin="12dp"
            android:padding="8dp"
            android:gravity="top"
            android:hint="I am the Short Section"
            android:background="@drawable/round_text"/>

        <EditText
            android:id="@+id/urltext"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textImeMultiLine"
            android:hint="URL OF IMAGE"
            android:layout_margin="12dp"
            android:background="@drawable/round_text"
            android:padding="8dp"/>

        <EditText
            android:id="@+id/completenews"
            android:layout_width="match_parent"
            android:minHeight="200dp"
            android:layout_height="match_parent"
            android:maxHeight="500dp"
            android:layout_margin="12dp"
            android:background="@drawable/round_text"
            android:gravity="top"
            android:hint="Complete News Goes Here"
            android:padding="8dp"/>

        <Button
            android:id="@+id/addpost"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="12dp"
            android:text="ADD POST"
            android:textColor="@color/black"/>
    </androidx.appcompat.widget.LinearLayoutCompat>

    </ScrollView>
</androidx.appcompat.widget.LinearLayoutCompat>
```

D. Admin Readnews

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".NewRead">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fillViewport="true">

        <androidx.appcompat.widget.LinearLayoutCompat
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <ImageView
                android:id="@+id/readimage"
                android:layout_width="match_parent"
                android:layout_height="250dp"
                android:scaleType="centerCrop"
                android:layout_margin="8dp"
                android:src="@drawable/ic_launcher_background"/>

            <TextView
                android:id="@+id/readtitle"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="8dp"
                android:textStyle="bold"
                android:textSize="24sp"
                android:textColor="@color/black"
                android:text="Top Headlines Toaday"/>
```

```
            <TextView
                android:lineHeight="22dp"
                android:id="@+id/readdesc"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="18sp"
                android:text="I am the body of description"
                android:layout_margin="8dp"
                android:textStyle="bold"/>


        </androidx.appcompat.widget.LinearLayoutCompat>

    </ScrollView>
</androidx.appcompat.widget.LinearLayoutCompat>
```

Admin Logic Code

1. MainActivity

```
package com.cli.dynoadmin;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;

import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class MainActivity extends AppCompatActivity {

   private ImageView postButton;
   private RecyclerView recyclerView;
   private FireAdapter fireAdapter;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       postButton = findViewById(R.id.postbutton);

       postButton.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View view) {
               startActivity(new Intent(MainActivity.this,
PostAcitvity.class));
           }
       });



       recyclerView = findViewById(R.id.listview);
       recyclerView.setLayoutManager(new LinearLayoutManager(this));

       FirebaseRecyclerOptions<Post> options = new
FirebaseRecyclerOptions.Builder<Post>()
```

```
  .setQuery(FirebaseDatabase.getInstance().getReference().child("ADMIN"),
  Post.class)
                .build();

        fireAdapter = new FireAdapter(options);
        recyclerView.setAdapter(fireAdapter);


    }
     @Override
       protected void onStart() {
            super.onStart();
            fireAdapter.startListening();
```

2. Adapter Class

```
package com.cli.dynoadmin;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.firebase.database.FirebaseDatabase;
import com.squareup.picasso.Picasso;

import org.w3c.dom.Text;

public class FireAdapter  extends FirebaseRecyclerAdapter<Post,
FireAdapter.PostViewHolder>{
```

```
    /**
     * Initialize a {@link RecyclerView.Adapter} that listens to a Firebase
query. See
     * {@link FirebaseRecyclerOptions} for configuration options.
     *
     * @param options
     */
    public FireAdapter(@NonNull FirebaseRecyclerOptions<Post> options) {
        super(options);

    }

    @Override
    protected void onBindViewHolder(@NonNull PostViewHolder holder, int position,
@NonNull Post model) {
            Picasso.get().load(model.getImageUrl()).into(holder.imageView);
            holder.TitleText.setText(model.getTitle());
            holder.Deccription.setText(model.getDescription());

            holder.Delete.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {

FirebaseDatabase.getInstance().getReference("ADMIN").child(model.getTitle()).set
Value(null);
                }
            });

            holder.itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Intent intent = new Intent(view.getContext(), NewRead.class);
                    intent.putExtra("title", model.getTitle());
                    intent.putExtra("description", model.getCompleteNews());
                    intent.putExtra("url", model.getImageUrl());
                    view.getContext().startActivity(intent);
                }
            });

    }

    @NonNull
    @Override
    public PostViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.feed, parent, false);

        return new PostViewHolder(view);
    }
```

```
    class PostViewHolder extends RecyclerView.ViewHolder {
        ImageView imageView;
        TextView TitleText;
        TextView Deccription;
        Button Delete;
        public PostViewHolder(@NonNull View itemView) {
            super(itemView);
            imageView = itemView.findViewById(R.id.newsimage);
            TitleText = itemView.findViewById(R.id.tvtext);
            Deccription = itemView.findViewById(R.id.tvbody);
            Delete = itemView.findViewById(R.id.delete);
        }
    }
}
```

3. NewsRead Class

```
package com.cli.dynoadmin;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

import com.squareup.picasso.Picasso;

public class NewRead extends AppCompatActivity {

    private ImageView imageView;
    private TextView headtext;
    private TextView descriptions;

    String title, description, url;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_read);

        imageView = findViewById(R.id.readimage);
        headtext = findViewById(R.id.readtitle);
        descriptions = findViewById(R.id.readdesc);

        title = getIntent().getStringExtra("title");
        url = getIntent().getStringExtra("url");
        description = getIntent().getStringExtra("description");
```

```
        headtext.setText(title);
        descriptions.setText(description);
        Picasso.get().load(url).into(imageView);

    }
}
```

4. PostActivity

```
package com.cli.dynoadmin;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.webkit.URLUtil;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class PostAcitvity extends AppCompatActivity {

    private EditText adminTitle;
    private EditText adminDescription;
    private EditText adminNews;
    private EditText adminUrl;
    private FirebaseUser firebaseUser;
    private Button adminButton;
    private FirebaseDatabase database;
    private DatabaseReference reference;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_post_acitvity);

        adminTitle = findViewById(R.id.headlinetext);
        adminDescription = findViewById(R.id.description);
```

```
        adminNews = findViewById(R.id.completenews);
        adminUrl = findViewById(R.id.urltext);
        adminButton = findViewById(R.id.addpost);


        adminButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String titletext = adminTitle.getText().toString();
                String desc = adminDescription.getText().toString();
                String news = adminNews.getText().toString();
                String url = adminUrl.getText().toString();
                Boolean isUrl = URLUtil.isValidUrl(url);

                if (!isUrl){
                    adminUrl.setError("Not A Valid Url");
                }


                if (isUrl && !titletext.isEmpty() && !desc.isEmpty() &&
!news.isEmpty()){

                    Post post = new Post(titletext, desc, news, url);
                    database = FirebaseDatabase.getInstance();
                    reference = database.getReference("ADMIN");

reference.child(titletext).setValue(post).addOnCompleteListener(new
OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            adminTitle.setText("");
                            adminDescription.setText("");
                            adminNews.setText("");
                            adminUrl.setText("");
                            startActivity(new Intent(PostAcitvity.this,
MainActivity.class));
                        }

                    });


                }
                else{
                    Toast.makeText(PostAcitvity.this, "Field Can't Be Empty",
Toast.LENGTH_SHORT).show();
                }

            }
        });
```

```
    }
}
```

5. RegisterActivity.class

```
package com.cli.dynoadmin;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Patterns;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.cli.dynoadmin.MainActivity;
import com.cli.dynoadmin.R;
import com.google.android.gms.auth.api.phone.SmsRetriever;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.FirebaseDatabase;

public class GoogleSign extends AppCompatActivity {

    private TextInputEditText username, useremail, password;
    private Button register;
    private FirebaseAuth mAuth;
    private TextView logintext;
    private ProgressBar loader;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_google_sign);

    loader = findViewById(R.id.loader);
    mAuth = FirebaseAuth.getInstance();


    username = findViewById(R.id.username);
    useremail = findViewById(R.id.useremail);
    password = findViewById(R.id.userpassword);

    register = findViewById(R.id.register);

    register.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            loader.setVisibility(View.VISIBLE);
            registerUser();
        }
    });


}

private void registerUser() {
    String userName = username.getText().toString().trim();
    String userMail = useremail.getText().toString().trim();
    String userpass = password.getText().toString().trim();


    if(userName.isEmpty()){
        username.setError("Everyone has a name");
        username.requestFocus();
        return;
    }
    if(userMail.isEmpty()){
        useremail.setError("Empty email does not exist");
        useremail.requestFocus();
        return;
    }
    if (!Patterns.EMAIL_ADDRESS.matcher(userMail).matches()){
        useremail.setError("Get me a valid mail");
        useremail.requestFocus();
        return;
    }
```

```
        if(userpass.isEmpty() || userpass.length() < 8){
            password.setError("Everyone has a name");
            password.requestFocus();
        }


        mAuth.createUserWithEmailAndPassword(userMail, userpass)
                .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()){
                            Admin admin = new Admin(userName, userMail,
userpass);


FirebaseDatabase.getInstance().getReference("Pro")

.child(FirebaseAuth.getInstance().getCurrentUser().getUid())

.setValue(admin).addOnCompleteListener(new OnCompleteListener<Void>() {
                                    @Override
                                    public void onComplete(@NonNull Task<Void>
task) {
                                        if (task.isSuccessful()){
                                            homepage();
                                        }
                                        else {
                                            loader.setVisibility(View.GONE);
                                            Toast.makeText(GoogleSign.this, "User
Exist", Toast.LENGTH_SHORT).show();
                                        }
                                    }
                                });
                        }
                        else{
                            loader.setVisibility(View.GONE);
                            Toast.makeText(GoogleSign.this, "User Exist",
Toast.LENGTH_SHORT).show();
                        }
                    }
                });

    }

    private void homepage() {
        startActivity(new Intent(this, MainActivity.class));
        finish();
```

```
    }

    @Override
    protected void onStart() {
        super.onStart();

        if (mAuth.getCurrentUser() != null){
            homepage();
        }

        logintext = findViewById(R.id.loginpage);
        loader.setVisibility(View.GONE);

        logintext.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(GoogleSign.this, LoginUSer.class));
                finish();
            }
        });
    }
}
```

6. Post Data Class

```
package com.cli.dynoadmin;

public class Post {
    private String title;
    private String description;
    private String completeNews;
    private String imageUrl;

    public Post() {
    }

    public Post(String title, String description, String completeNews,
String imageUrl) {
        this.title = title;
        this.description = description;
        this.completeNews = completeNews;
        this.imageUrl = imageUrl;
    }
```

```
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getCompleteNews() {
        return completeNews;
    }

    public void setCompleteNews(String completeNews) {
        this.completeNews = completeNews;
    }

    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }
}
```

7. ManifestFiles For Permissions

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cli.dynoadmin">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
```

```
        android:roundIcon="@mipmap/ic_launcher"
        android:supportsRtl="true"
        android:theme="@style/Theme.DynoAdmin">
        <activity
            android:name=".LoginUSer"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="false" />
        <activity
            android:name=".NewRead"
            android:exported="false" />
        <activity
            android:name=".PostAcitvity"
            android:exported="false" />
        <activity
            android:name=".GoogleSign"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Dyno User Sider Files

1. MainActivity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">


        <androidx.appcompat.widget.Toolbar
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/black"
            android:textAlignment="center"
            android:visibility="visible"
            tools:visibility="visible">

            <TextView
                android:id="@+id/iamuser"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:text="DYNO NEWS"
                android:textAlignment="center"
                android:textColor="@color/white"
                android:textSize="24sp"
                android:textStyle="bold" />
            <ImageView
                android:id="@+id/logout"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="right"
```

```
                android:layout_marginRight="10dp"
                android:src="@drawable/ic_baseline_add_reaction_24" />


        </androidx.appcompat.widget.Toolbar>

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/categorylist"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="8dp"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
            tools:listitem="@layout/catering" />


        <ProgressBar
            android:id="@+id/progress"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:visibility="visible" />

        <androidx.swiperefreshlayout.widget.SwipeRefreshLayout
            android:id="@+id/refreshLayout"
            android:layout_width="match_parent"
            android:layout_height="match_parent">


            <androidx.recyclerview.widget.RecyclerView
                android:id="@+id/recycle"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                tools:listitem="@layout/news" />
        </androidx.swiperefreshlayout.widget.SwipeRefreshLayout>


    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

2. NewsRead

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context=".NewRead">

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true">

    <androidx.appcompat.widget.LinearLayoutCompat
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/readimage"
            android:layout_width="match_parent"
            android:layout_height="250dp"
            android:scaleType="centerCrop"
            android:layout_margin="8dp"
            android:src="@drawable/ic_launcher_background"
            android:contentDescription="@string/some_desc" />

        <TextView
            android:id="@+id/readtitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="8dp"
            android:textStyle="bold"
            android:textSize="24sp"
            android:textColor="@color/black"
            android:text="@string/title"/>

        <TextView
            android:id="@+id/readdesc"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:text="@string/description"
            android:layout_margin="8dp"
            android:textStyle="bold"/>


    </androidx.appcompat.widget.LinearLayoutCompat>

</ScrollView>
```

```
</androidx.appcompat.widget.LinearLayoutCompat>
```

3. Register.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".NewRead">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fillViewport="true">

        <androidx.appcompat.widget.LinearLayoutCompat
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageView
                android:id="@+id/readimage"
                android:layout_width="match_parent"
                android:layout_height="250dp"
                android:scaleType="centerCrop"
                android:layout_margin="8dp"
                android:src="@drawable/ic_launcher_background"
                android:contentDescription="@string/some_desc" />

            <TextView
                android:id="@+id/readtitle"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="8dp"
                android:textStyle="bold"
                android:textSize="24sp"
                android:textColor="@color/black"
                android:text="@string/title"/>

            <TextView
                android:id="@+id/readdesc"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="18sp"
                android:text="@string/description"
                android:layout_margin="8dp"
                android:textStyle="bold"/>


        </androidx.appcompat.widget.LinearLayoutCompat>

    </ScrollView>



</androidx.appcompat.widget.LinearLayoutCompat>
```

4. News.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/listholder"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    app:cardCornerRadius="3dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="8dp">

        <ImageView
            android:id="@+id/newsimage"
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:layout_marginBottom="5dp"
            android:scaleType="centerCrop"
            app:srcCompat="@drawable/ic_launcher_foreground" />

        <TextView
            android:id="@+id/tvtitle"
            android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            android:text="The Earth Was Banging"
            android:textColor="#000000"
            android:textSize="21sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tvbody"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="The initail public rolloit ws released"
            android:textColor="#171515"
            android:textSize="16sp" />
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

Logical Code For Dyno User

1. MainActivity

```
package com.cli.login;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Adapter;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.firestore.auth.User;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity implements
```

```
CatAdapter.CategoryClick, View.OnClickListener {
   private RecyclerView recyclerView;
   private RecyclerView categorylist;
   LinearLayoutManager linearLayoutManager;
   private ProgressBar progressBar;
   adapter newsadapter;
   CatAdapter catAdapter;
   ArrayList<articles> articlesArrayList;
   ArrayList<category> categoryArrayList;
   private SwipeRefreshLayout swipeRefreshLayout;
   private String[] words = new String[]{"Current Affairs India", "Insurnce
India", "Gaming", "World Bank", "Vaccination", "University", "Smartphones",
"Laptops", "School", "Jobs", "Government"};
   Random random = new Random();
   int index;
   private ImageView logout;
   private TextView uservalue;


   @SuppressLint("NotifyDataSetChanged")
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       swipeRefreshLayout = findViewById(R.id.refreshLayout);
       categorylist = findViewById(R.id.categorylist);
       recyclerView = findViewById(R.id.recycle);
       articlesArrayList = new ArrayList<>();
       categoryArrayList = new ArrayList<>();
       uservalue = findViewById(R.id.iamuser);
       progressBar = findViewById(R.id.progress);
       logout = findViewById(R.id.logout);
       logout.setOnClickListener(this);

//        uservalue.setText("Hello");
       newsadapter = new adapter(articlesArrayList, this);
       catAdapter = new CatAdapter(categoryArrayList,
this::OnCategoryClick);


       LinearLayoutManager cate = new LinearLayoutManager(this,
RecyclerView.HORIZONTAL, true);


       linearLayoutManager = new LinearLayoutManager(this);
       recyclerView.setLayoutManager(linearLayoutManager);
       recyclerView.setAdapter(newsadapter);
       categorylist.setLayoutManager(cate);
```

```
        categorylist.setAdapter(catAdapter);
        getCategories();

        if (isNetworkConnected()) {

            getNews("All");
            Snackbar.make(logout, "Resfesh For More",
Snackbar.LENGTH_LONG).show();
        } else {
            Snackbar.make(logout, "No Network", Snackbar.LENGTH_LONG).show();
        }
        catAdapter.notifyDataSetChanged();

        swipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                if (isNetworkConnected()) {
                    index = random.nextInt(words.length);
                    getNews(words[index]);
                    newsadapter.notifyDataSetChanged();
                    swipeRefreshLayout.setRefreshing(false);
                } else {
                    Snackbar.make(logout, "No Internet",
Snackbar.LENGTH_LONG).show();
                    swipeRefreshLayout.setRefreshing(false);
                }


            }

        });

        FirebaseUser firebaseUser =
FirebaseAuth.getInstance().getCurrentUser();
        DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference("users");
        String userid = firebaseUser.getUid();

        databaseReference.child(userid).addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                user userprofile = snapshot.getValue(user.class);

                if (userprofile != null) {
                    String fullName = "Hey " + userprofile.fullName;
```

```
                    uservalue.setText(fullName);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Snackbar.make(uservalue, "Server Crashed",
Snackbar.LENGTH_LONG).show();
            }
        });


    }

    public void getCategories() {
        categoryArrayList.add(new category("Movies"));
        categoryArrayList.add(new category("Crime"));
        categoryArrayList.add(new category("Corona"));
        categoryArrayList.add(new category("Science"));
        categoryArrayList.add(new category("Billionaires"));
        categoryArrayList.add(new category("Startup"));
        categoryArrayList.add(new category("World"));
        categoryArrayList.add(new category("Health"));
        categoryArrayList.add(new category("Technology"));
        categoryArrayList.add(new category("Finance"));
        categoryArrayList.add(new category("Politics"));
        categoryArrayList.add(new category("All"));

    }


    private void getNews(String types) {
        articlesArrayList.clear();
        String url =
"https://newsapi.org/v2/top-headlines?country=in&category=general&apiKey=c17
65de2161c46c49ccfa20000d92cf9";
        String categoryUrl = "https://newsapi.org/v2/everything?q=" + types +
" india&language=en&apiKey=c1765de2161c46c49ccfa20000d92cf9";
        String BASE_URL = "https://newsapi.org/";
        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        RetrofitAPi retrofitAPi = retrofit.create(RetrofitAPi.class);

        Call<list> call;
        if (types.equals("All")) {
            call = retrofitAPi.getAllNews(url);
```

```
        } else if (types.equals("World")) {
            call =
retrofitAPi.getAllNews("https://newsapi.org/v2/top-headlines?country=us&apiK
ey=c1765de2161c46c49ccfa20000d92cf9");
        } else {
            call = retrofitAPi.getAllNews(categoryUrl);
        }


        call.enqueue(new Callback<list>() {
            @SuppressLint("NotifyDataSetChanged")
            @Override
            public void onResponse(Call<list> call, Response<list> response)
{

                list list = response.body();
                progressBar.setVisibility(View.GONE);
                ArrayList<articles> articles = list.getArticles();
                for (int i = 0; i < articles.size(); i++) {
                    articlesArrayList.add(new
articles(articles.get(i).getTitle(), articles.get(i).getDescription(),
                            articles.get(i).getUrlToImage(),
articles.get(i).getContent(), articles.get(i).getUrl()));

                }
                newsadapter.notifyDataSetChanged();

            }

            @Override
            public void onFailure(Call<list> call, Throwable t) {

            }
        });
    }

    @Override
    public void OnCategoryClick(int position) {
        String pos = categoryArrayList.get(position).getCatext();
        getNews(pos);
    }

    @Override
    public void onClick(View view) {
        if (view.getId() == R.id.logout) {
//            case R.id.logout:
//                FirebaseAuth.getInstance().signOut();
//                startActivity(new Intent(this, LoginUser.class));
```

```
//                finish();
            startActivity(new Intent(this, UserPost.class));
            Snackbar.make(logout, "Feeling Sad",
Snackbar.LENGTH_LONG).show();

        }
    }

    private boolean isNetworkConnected() {
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);

        return cm.getActiveNetworkInfo() != null &&
cm.getActiveNetworkInfo().isConnected();
    }


}
```

2. Adapters for Category

```
package com.cli.login;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Adapter;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.snackbar.Snackbar;

import java.util.ArrayList;
import java.util.List;

public class CatAdapter extends RecyclerView.Adapter<CatAdapter.ViewHolder>{

    public ArrayList<category> categoryArrayList;
    private CategoryClick categoryClick;

    public CatAdapter(ArrayList<category> categoryArrayList, CategoryClick
categoryClick) {
        this.categoryArrayList = categoryArrayList;
        this.categoryClick = categoryClick;
    }
```

```
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view = LayoutInflater.from(parent.getContext()).
                inflate(R.layout.catering, parent, false);
        return new CatAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

        category category = categoryArrayList.get(position);
        holder.catext.setText(category.getCatext());
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                categoryClick.OnCategoryClick(holder.getAdapterPosition());
            }
        });


    }

    @Override
    public int getItemCount() {
        return categoryArrayList.size();
    }

    public interface CategoryClick{
        void OnCategoryClick(int position);
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        private TextView catext;
        View mView;
        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            catext = itemView.findViewById(R.id.catext);
            mView = itemView;
        }
    }
}
```

3. Articles Adapter

```
package com.cli.login;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.squareup.picasso.Picasso;

import org.w3c.dom.Text;

import java.util.ArrayList;

public class adapter extends
RecyclerView.Adapter<adapter.NewsHolder>{

   ArrayList<articles> articlesArrayList;
   private Context context;

   public adapter(ArrayList<articles> articlesArrayList, Context
context) {
        this.articlesArrayList = articlesArrayList;
        this.context = context;
   }

   @NonNull
   @Override
   public NewsHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.news, parent, false);
```

```
        return new adapter.NewsHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull NewsHolder holder, int
position) {
        articles articles = articlesArrayList.get(position);
        holder.tvtitle.setText(articles.getTitle());
        holder.tvbody.setText(articles.getDescription());

Picasso.get().load(articles.getUrlToImage()).into(holder.newsimag
e);

        holder.itemView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(context,
MainActivity2.class);
                intent.putExtra("title", articles.getTitle());
                intent.putExtra("image",
articles.getUrlToImage());
                intent.putExtra("content", articles.getContent());
                intent.putExtra("url", articles.getUrl());
                context.startActivity(intent);

            }
        });

    }

    @Override
    public int getItemCount() {
        return articlesArrayList.size();
    }

    public class NewsHolder extends RecyclerView.ViewHolder {
        private TextView tvtitle;
        private TextView tvbody;
```

```
        private ImageView newsimage;
        public NewsHolder(@NonNull View itemView) {
            super(itemView);
            tvtitle = itemView.findViewById(R.id.tvtitle);
            tvbody = itemView.findViewById(R.id.tvbody);
            newsimage = itemView.findViewById(R.id.newsimage);

        }
    }
}
```

3. Social Adapter

```
package com.cli.login;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.firebase.ui.database.FirebaseRecyclerOptions;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
```

```
import java.util.Collection;
import java.util.Collections;
import java.util.Date;

public class UserPost extends AppCompatActivity {

   private ImageView  back;
   private RecyclerView recyclerView;
   private FireAdapter fireAdapter;


   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_user_post);
       back = findViewById(R.id.back);
       back.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View view) {
               startActivity(new Intent(UserPost.this,
MainActivity.class));
               finish();
           }
       });

       recyclerView = findViewById(R.id.userlist);
       recyclerView.setLayoutManager(new LinearLayoutManager(this));

       FirebaseRecyclerOptions<Post> options = new
FirebaseRecyclerOptions.Builder<Post>()

.setQuery(FirebaseDatabase.getInstance().getReference().child("ADMIN"),
Post.class)
               .build();

       fireAdapter = new FireAdapter(options);
       recyclerView.setAdapter(fireAdapter);
```

```
    }

    @Override
    protected void onStart() {
        super.onStart();
        fireAdapter.startListening();
    }
}
```

4. Read News In New Screen

```
package com.cli.login;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import
com.google.android.material.progressindicator.LinearProgressIndicator;
import com.squareup.picasso.Picasso;

public class MainActivity2 extends AppCompatActivity {

    String title, image, content, url;

    private TextView headline;
    private ImageView imageView;
    private   TextView discontent;
    private Button DisplayBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
```

```
        title = getIntent().getStringExtra("title");
        image = getIntent().getStringExtra("image");
        content = getIntent().getStringExtra("content");
        url = getIntent().getStringExtra("url");

        headline = findViewById(R.id.headline);
        discontent = findViewById(R.id.headcontent);
        imageView = findViewById(R.id.newsimage);
        DisplayBtn = findViewById(R.id.readbtn);


        headline.setText(title);
        discontent.setText(content);
        Picasso.get().load(image).into(imageView);

        DisplayBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
               callMe();
            }
        });

    }

    private void callMe() {
        startActivity(new Intent(Intent.ACTION_VIEW,
Uri.parse(url.toString())));

    }
}
```

5. Retrofit APi Client

```
package com.cli.login;

import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Url;

public interface RetrofitAPi {

    @GET
    Call<list> getAllNews(@Url String url);

    Call<list> getCategoryNews(@Url String url);

}
```

6. Login & Register

```
package com.cli.login;

    import androidx.annotation.NonNull;
    import androidx.appcompat.app.AppCompatActivity;

    import android.content.Context;
    import android.content.Intent;
    import android.net.ConnectivityManager;
    import android.os.Bundle;
    import android.util.Patterns;
    import android.view.View;
    import android.view.inputmethod.InputMethodManager;
    import android.widget.ProgressBar;
    import android.widget.TextView;

    import com.google.android.gms.tasks.OnCompleteListener;
    import com.google.android.gms.tasks.Task;
    import com.google.android.material.button.MaterialButton;
    import com.google.android.material.snackbar.Snackbar;
    import com.google.android.material.textfield.TextInputEditText;
```

```
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class LoginUser extends AppCompatActivity implements
View.OnClickListener{

    private TextView registertext;
    private TextInputEditText loginmail, loginpassword;
    private MaterialButton signin;
    private ProgressBar loginbar;
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login_user);
        registertext = findViewById(R.id.registertext);
        registertext.setOnClickListener(this);

        //User Data
        mAuth = FirebaseAuth.getInstance();
        if (mAuth.getCurrentUser() != null){
            goToMain();
        }
        loginmail = findViewById(R.id.loginmail);
        loginpassword = findViewById(R.id.loginpassword);
        signin = findViewById(R.id.signin);
        loginbar = findViewById(R.id.loaderlogin);

        signin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (isNetworkConnected()){
                loginuser();
                loginbar.setVisibility(View.VISIBLE);
                }
                else {
                    Snackbar.make(loginmail, "I need internet",
Snackbar.LENGTH_LONG).show();
```

```
                }
            }
        });
    }

    private void loginuser() {

        String mail = loginmail.getText().toString().trim();
        String lpass = loginpassword.getText().toString().trim();
        closeme();
        if(mail.isEmpty()){
            loginmail.setError("Empty email does not exist");
            loginmail.requestFocus();
            return;
        }

        if (!Patterns.EMAIL_ADDRESS.matcher(mail).matches()){
            loginmail.setError("Get me a valid mail");
            loginmail.requestFocus();
            return;
        }

        if(lpass.isEmpty() || lpass.length() < 8){
            loginpassword.setError("Everyone has a name");
            loginpassword.requestFocus();
            return;
        }

        mAuth.signInWithEmailAndPassword(mail,
lpass).addOnCompleteListener(new OnCompleteListener<AuthResult>()
{
            @Override
            public void onComplete(@NonNull Task<AuthResult> task)
{
                if (task.isSuccessful()){
                    goToMain();
                    loginbar.setVisibility(View.GONE);
                }
                else{
```

```
                    Snackbar.make(registertext, "User does not
exist", Snackbar.LENGTH_SHORT).show();

                    loginbar.setVisibility(View.GONE);
                    registerUser();
                }
            }
        });
    }

    private void registerUser() {
        startActivity(new Intent(this, RegisterUser.class));
    }

    private void goToMain() {
        startActivity(new Intent(LoginUser.this,
MainActivity.class));
        finish();
    }

    private void closeme() {
        View view = this.getCurrentFocus();
        if (view != null) {
            InputMethodManager manager
                    = (InputMethodManager)
                    getSystemService(
                            Context.INPUT_METHOD_SERVICE);
            manager
                    .hideSoftInputFromWindow(
                            view.getWindowToken(), 0);
        }
    }
    private boolean isNetworkConnected() {
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);

        return cm.getActiveNetworkInfo() != null &&
cm.getActiveNetworkInfo().isConnected();
    }
```

```
    @Override
    public void onClick(View view) {
        if(view.getId() == R.id.registertext){
            startActivity(new Intent(this, RegisterUser.class));

        }
    }
}
```