A Project Report

On

“News Application”

Submitted in partial fulfillment of the requirement of
University of Mumbai

For the Degree of

Bachelor of Computer Science

Submitted By

Ranjit Parida

Under the Guidance of

Prof.(Ms.) Saba Shaikh

Final Year Computer Science

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

Affiliated to University of Mumbai.

Mumbai (**2020-**

2021)

RAMNIRANJAN JHUNJHUNWALA COLLEGE
GHATKOPAR(W), MUMBAI-400 086

CERTIFICATE

This is to certify that **Ranjit Parida** has successfully completed the project titled

“News Application”

under our guidance towards the partial fulfillment of degree of Bachelors of Science (Computer Science – SEM V) submitted to Ramniranjan Jhunjhunwala College, Ghatkopar (W) during the academic year 2020 – 2021 as specified by

UNIVERSITY OF MUMBAI.

Prof. Saba Shaikh	**Prof. (Mrs.) Anita Gaikwad**
Project Guide	**In-Charge**
Dept. Of Computer Science	**Dept. Of Computer Science**

INDEX

Acknowledgement

I have great pleasure for representing this project report entitled **“News Application”** and I grab this opportunity to convey my immense regards towards all the distinguished people who have their valuable contribution in the hour of need.

I take this opportunity to thank **“Mr. Himanshu Dawade”**, our **Principal of Ramniranjan Jhunjhunwala college, Ghatkopar(W)** for giving me an opportunity to study in the institute and the most needed guidance throughout the duration of the course.

I also like to extend my gratitude to “**Prof. Saba Sheikh**”, for their timely and prestigious guidance and necessary support during each phase of the project.

I would also like to thank **“Mrs. Anita Gaikwad”**, Head of the Department for their timely and prestigious guidance and necessary support during each phase of the project.

I also owe to my fellow friends who have been constant source of help to solve the problems and also helped me during the project development phase.

Thanking You,
Ranjit Parida

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.

Ranjit Parida

Feasibility Study

Feasibility study is to check the viability of the project under consideration. Theoretically various types of feasibilities are conducted, but I have conducted three types of feasibilities explained as under.

ECONOMIC FEASIBIITY

1. Development Cost
   - Equipment required for developing the software are easily available.
   - Equipment maintenance is also minimum.
   - Once the required hardware and software requirements get fulfilled there is no need for the users of our system to spend for any additional overhead.
2. Benefits which cannot be measured:
   - Increased customer Loyalty.
   - Increased customer satisfaction.

TECHNICAL FEASIBILITY

At first it is necessary to check that the proposed system is technically feasible or not and to determine the technology and skill necessary to carry out the project. If they are not available then find out the solution to obtain them.

OPERATIONAL FEASIBILITY

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

- It is easy to use and is very simple.
- The Application will support all browser.
- This Application will avoid confusion and resistance by catching the user’sattention, as it is presentable.

Scope of the System

The work and resources that go into the creation of the product or service are essentially the things that frame the scope of the project. The scope of the project outlines the objectives of the project and the goals that need to be met to achieve a satisfactory result.

The main aim of "News Application" is to deliver the news to the peoplethrough this application with the help of NEWS API. User can easily look for their stuff and have information about it.

This Website also provides the feature to look varieties of different sections like Fashion,Culture,Science,Environment,Society etc . User can also adjust the news from New to Old, Colour Theme and other things. User can also send the news to another person if he wants to share a particular news.providing detailed information abouta particular news.It enables you to stay current on world and national headlines anywhere, anytime.

Existing System and it’s Disadvantages

- There are many News Application available on internet. Most of the News app does not provide complete details of the news and latest news are not updated properly.
- There are many news app which are difficult to use for people who don’t know how to use the app properly.
- Many existing non trusted news application shows fake news which are not from a trusted website.
- Existing news application shows embedded image or video, which may don’t have proof just to gain view.
- Most of the news application are scam or not reliable since they add more information on their own and people could lose interest. This could lead to confusion and is the worst disadvantage of this news applications.
- Existing system don’t have proper information about the news, They don’t show news about other fields like Culture,Society and others.

Proposed System and its advantages

The News application shows all the latest news in a recycler view format. It uses high level programming language Android to perform this task.

Android basically uses java programming language to perform tasks and uses XML format files to store User-Interface design.

The API used here is NEWS API. The news are represented in JSON format and then converted into plain text format to make it readable. By this it makes it very easy to make it understand by humans.

It shows a variety of news happening all over the world. User can Click a particular type of news section if he wants to see only news related to that topic.

The User can adjust the no.of items he wants to see,user can adjust the Colour Theme and Test Size.The user can also read the new in NewsPaper Edition Mode.

Advantages :

- Shows all types and categories of latest news all around the globe.
- User can also see previously streamed news.
- Lightweight and easy to handle as user friendly interface provided to the software.
- Can convert the format of new to NewsPaper Edition Mode.

Technical Requirements

Software:

- **Android Studio**
- **JDK 1.8**
- **XML**
- **Windows platform**
- **Android SDK 4.4.5.**

Programming Languages:

Frontend: HTML, Java, Android Studio.

Backend: XML.

Hardware: Mouse, KeyBoard, Monitor, CPU, etc.

Software Development Model

V-Model

V-model is an SDLC model where the execution of processes happens in a sequential manner in a V-shape. It is also known as **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.

Phases of V-model:

a. Requirement 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 requirement. 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 the business requirements can be used as an input for acceptance testing.

b. System Design

Once you have 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.

c. 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 this information, integration tests can be designed and documented during this stage.

d. 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.

e. 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.

f. Validation Phases

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

i. Unit Testing

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

ii. Integration Testing

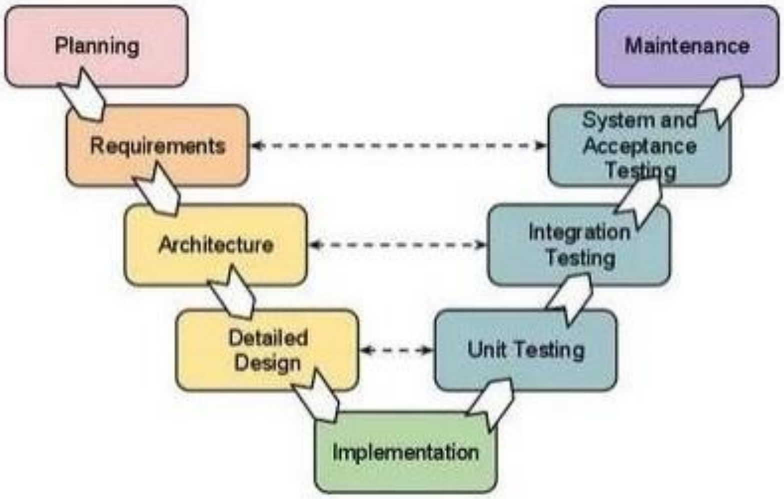
Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

iii. 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.

iv. Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in the 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.



GANTT CHART

A Gantt chart is a useful graphical tool which shows activities or tasks performed against time. It is also known as visual presentation of a project where the activities are broken down and displayed on a chart which makes it is easy to understand and interpret.
On the chart, tasks are shown on the vertical axis while the scheduled time-spend is laid out on the horizontal axis. Each task is represented by a bar that shows the time required for the project.The bar then represents or shows percentage of tasks that have been completed. It also shows dependencies, which simply means the interlinkages between various activities in he project.Understanding the inter-linkages between activities is very important to monitor and Gantt charts help the project manager to do just that. It conveys the information about the completion of other activities in the project. This information is important because of the interlinkages between various activities and if one activity gets delayed it will have an impact on others.

Gantt chart is useful tool in planning and scheduling the projects. It keeps the management updated as to when the project will get completed. It also keeps the management informed about any additional resources that are required, and manage dependencies between tasks.

A Gantt chart is made up of several different elements. Below given are 8 key components to know how to read a Gantt chart:

- Task list: Runs vertically down the left of the Gantt chart to describe project work and may be organized into groups and sub-groups.

- Timeline: Runs horizontally across the top of the Gantt chart and shows months, weeks, days, and years.

- Dateline: A vertical line that highlights the current date on the Gantt chart.

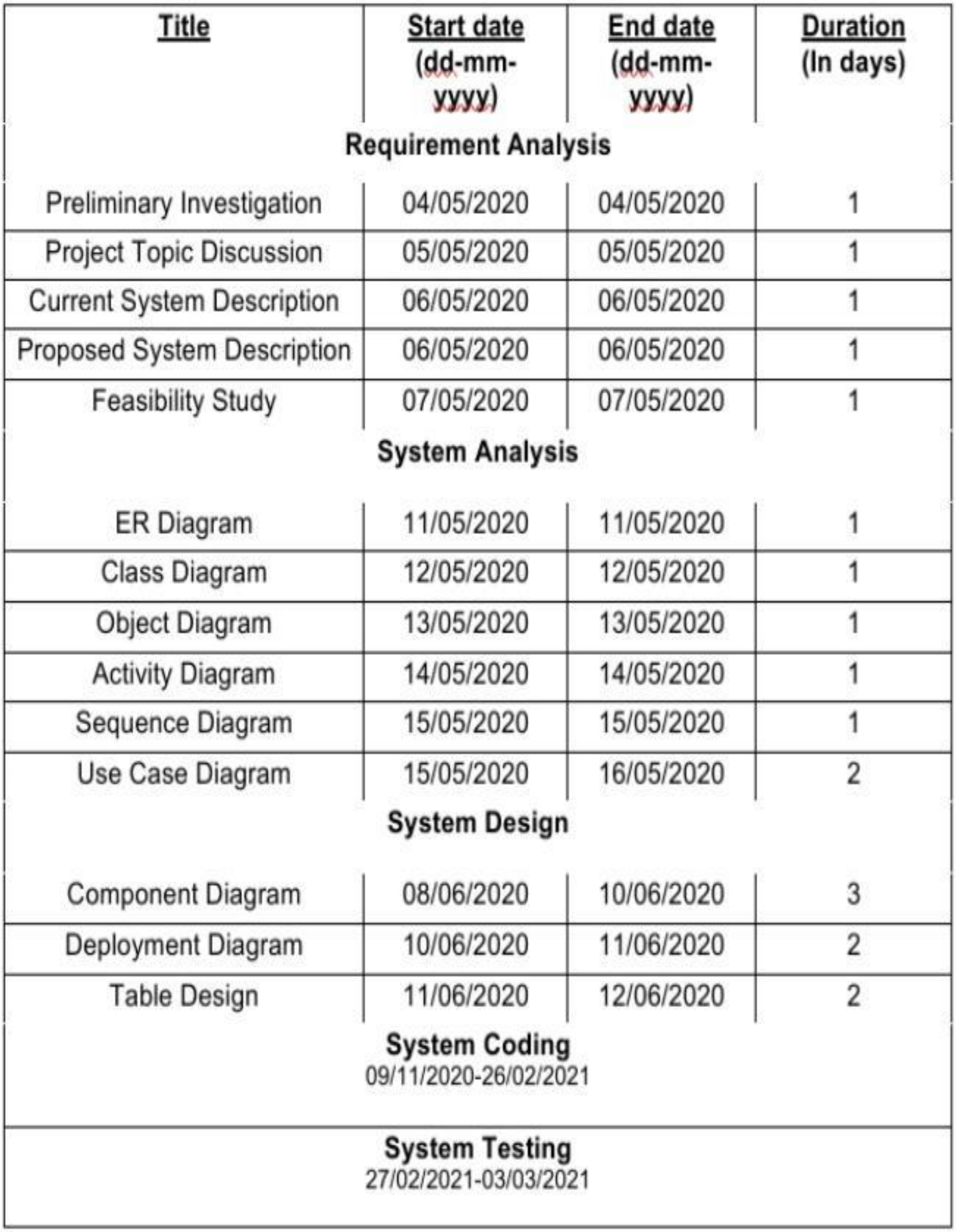
- Bars: Horizontal markers on the right side of the Gantt chart that represent tasks and show progress, duration, and start and end dates.

- Milestones: Yellow diamonds that call out major events, dates, decisions, and deliverables.

- Dependencies: Light gray lines that connect tasks that need to happen in a certain order.

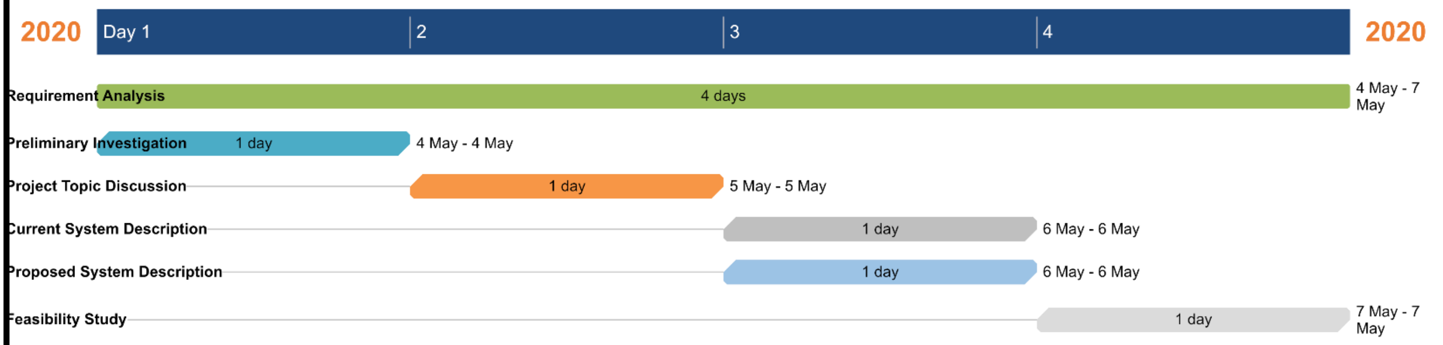
- Progress: Shows how far along work is and may be indicated by %Complete and/or bar showing.

- Resource assigned: Indicates the person or team responsible for completing a task.

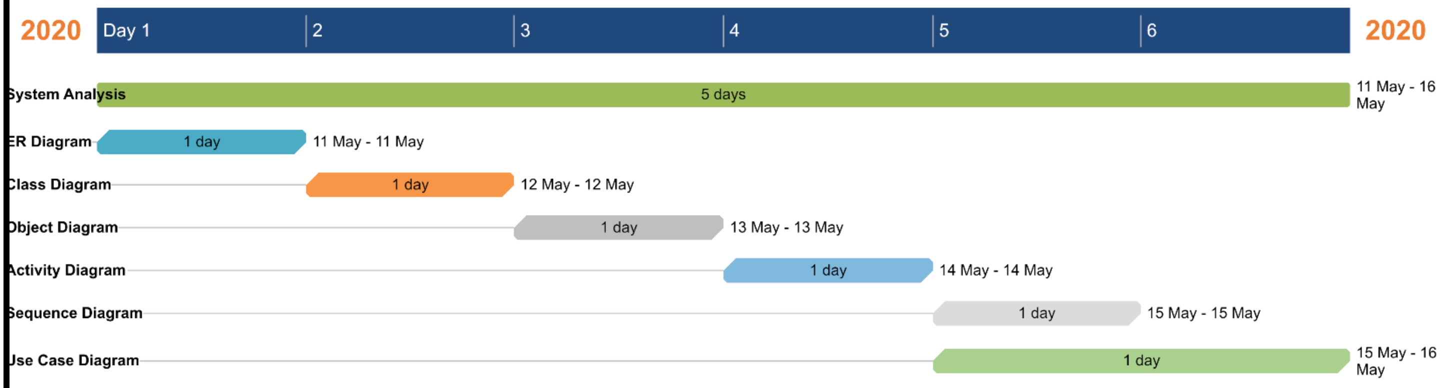


Title	**Start date** (dd-mm-yyyy)	**End date** (dd-mm-yyyy)	**Duration** (In days)
Requirement Analysis			
Preliminary Investigation	04/05/2020	04/05/2020	1
Project Topic Discussion	05/05/2020	05/05/2020	1
Current System Description	06/05/2020	06/05/2020	1
Proposed System Description	06/05/2020	06/05/2020	1
Feasibility Study	07/05/2020	07/05/2020	1
System Analysis			
ER Diagram	11/05/2020	11/05/2020	1
Class Diagram	12/05/2020	12/05/2020	1
Object Diagram	13/05/2020	13/05/2020	1
Activity Diagram	14/05/2020	14/05/2020	1
Sequence Diagram	15/05/2020	15/05/2020	1
Use Case Diagram	15/05/2020	16/05/2020	2
System Design			
Component Diagram	08/06/2020	10/06/2020	3
Deployment Diagram	10/06/2020	11/06/2020	2
Table Design	11/06/2020	12/06/2020	2
System Coding 09/11/2020-26/02/2021			
System Testing 27/02/2021-03/03/2021			

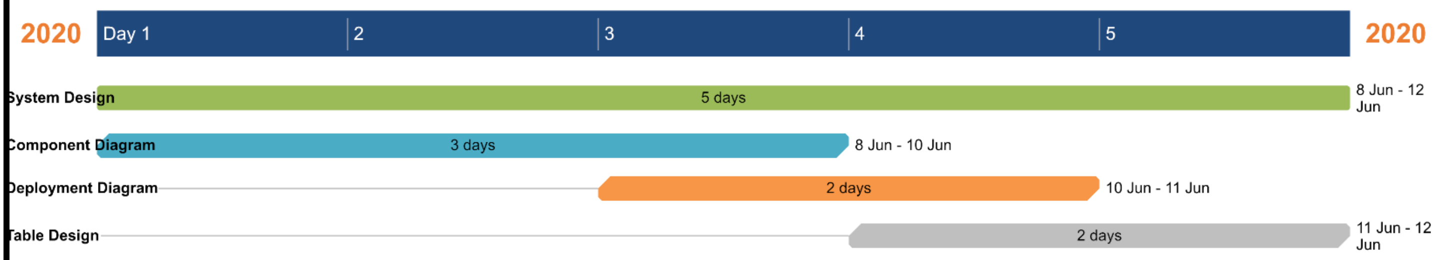
Requirement Analysis



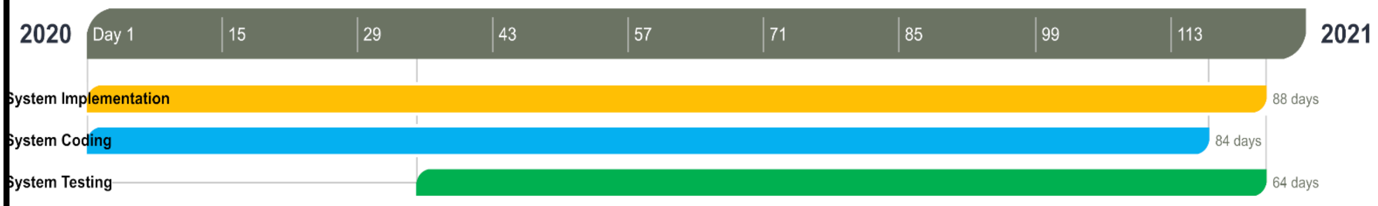
System Analysis



System Design



System Coding and Implementation



System Analysis

Entity Relationship Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

An ER diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

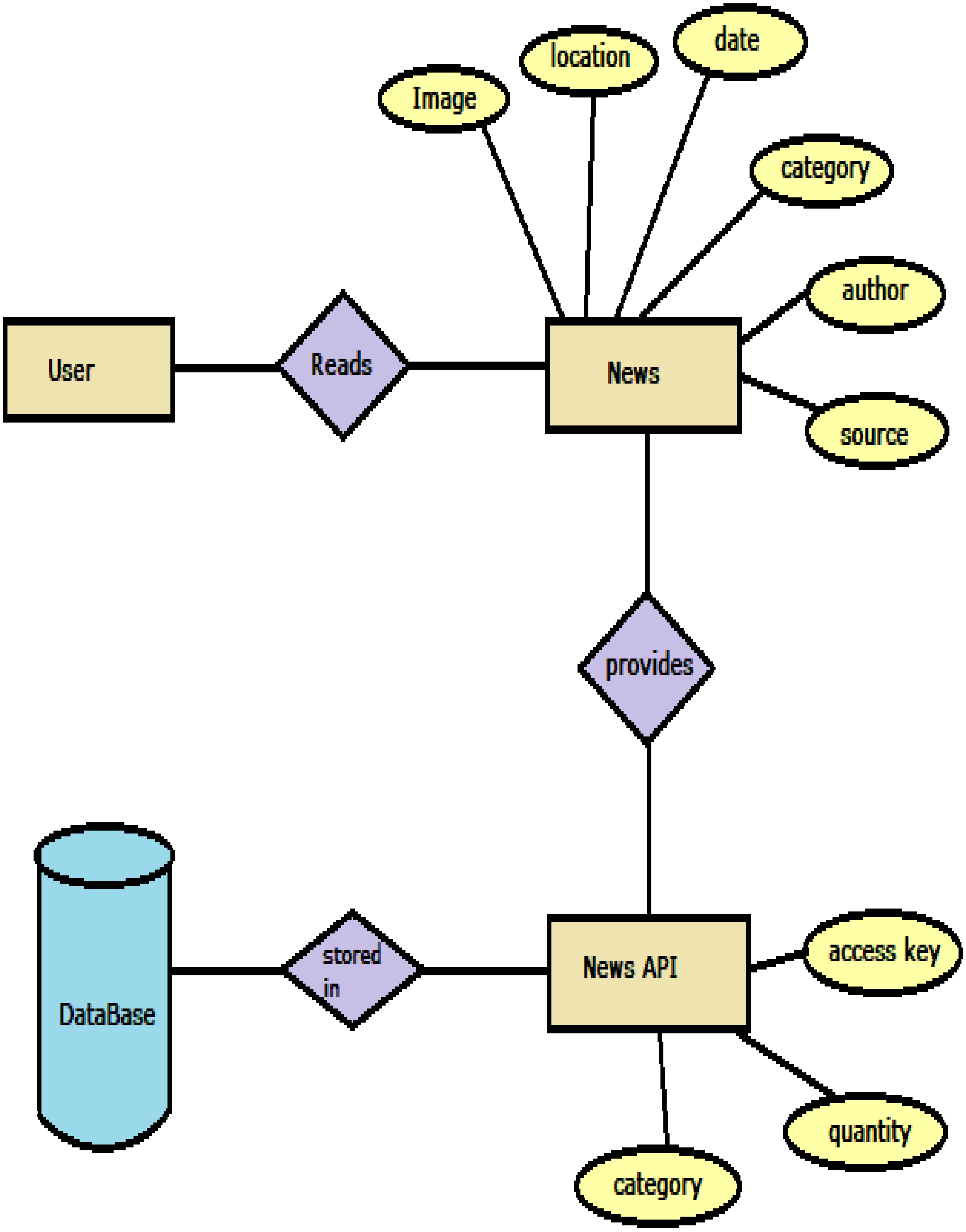
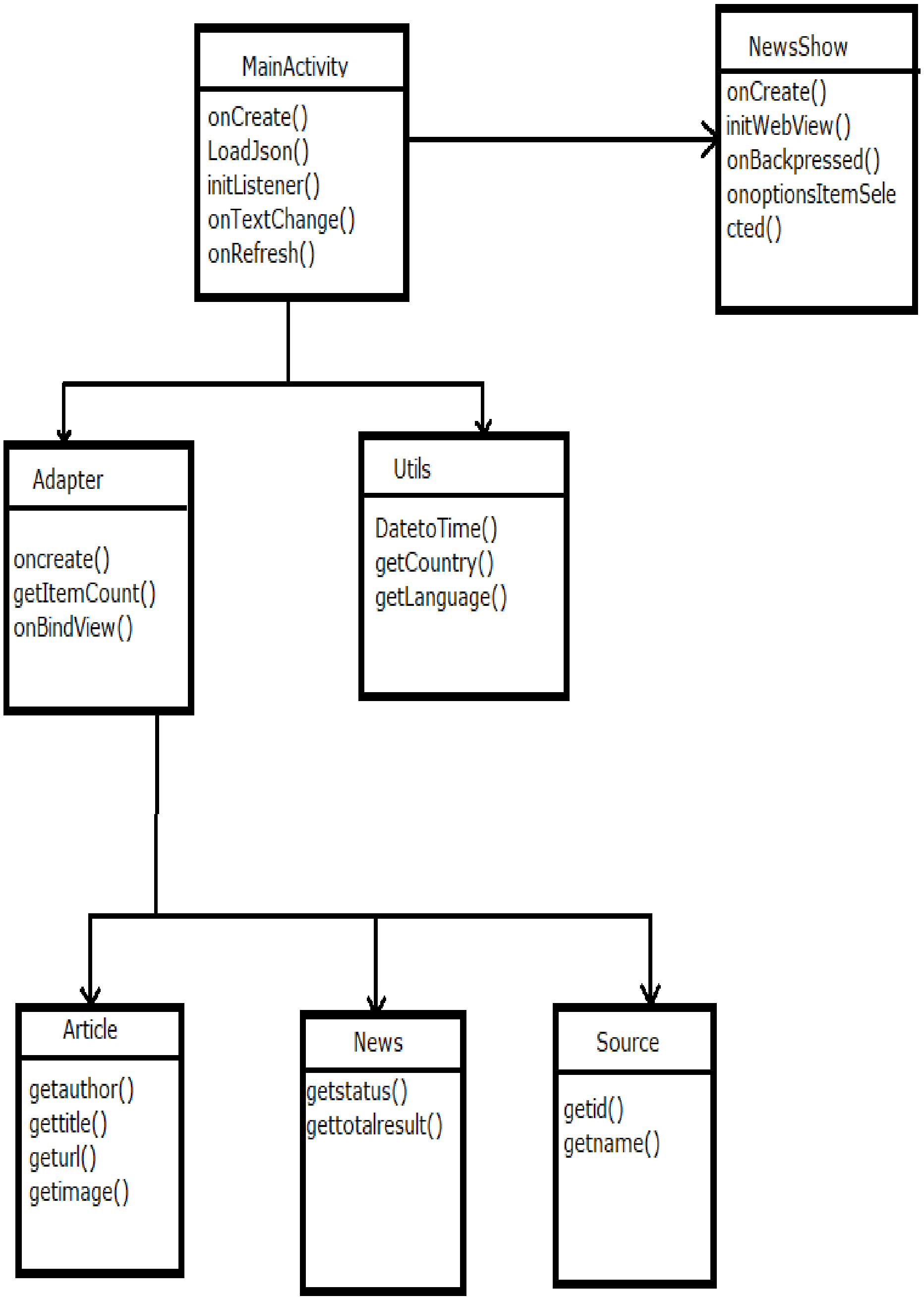


Image
location
date
category
author
source
User
Reads
News
provides
DataBase
stored in
News API
access key
quantity
category

Class Diagram.

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modelling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP).


MainActivity
onCreate()
LoadJson()
initListener()
onTextChange()
onRefresh()
NewsShow
onCreate()
initWebView()
onBackpressed()
onoptionsItemSele
cted()
Adapter
oncreate()
getItemCount()
onBindView()
Utils
DatetoTime()
getCountry()
getLanguage()
Article
getauthor()
gettitle()
geturl()
getimage()
News
getstatus()
gettotalresult()
Source
getid()
getname()

Object Diagram

An object diagram is a graph of instances, including objects and data values.

A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time.

Object diagrams and class diagrams are closely related and use almost identical notation. Both diagrams are meant to visualize static structure of a system. While class diagrams show classes, object diagrams display instances of classes (objects).

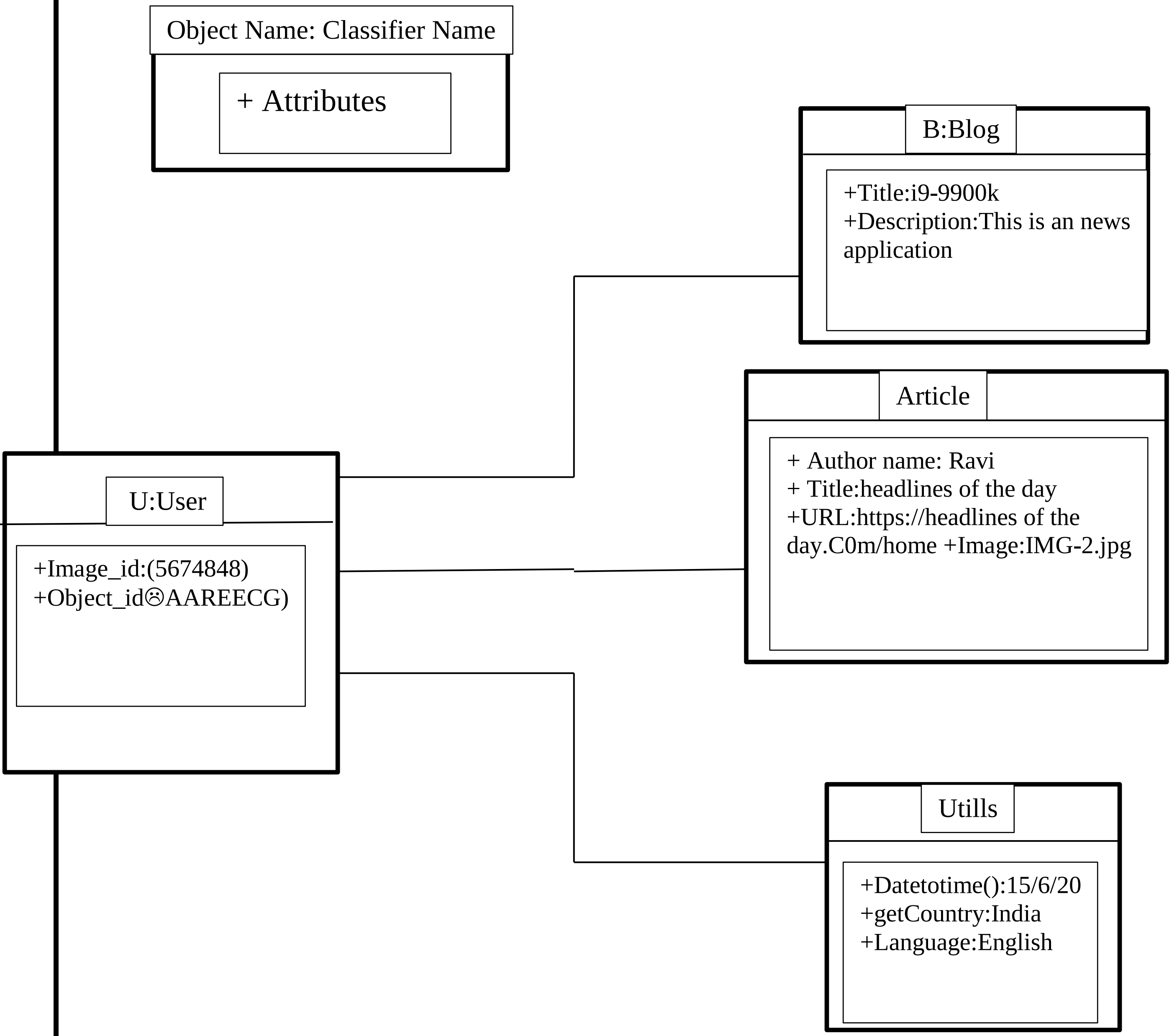
Object diagrams are more concrete than class diagrams. They are often used to provide examples or act as test cases for class diagrams. Only aspects of current interest in a model are typically shown on an object diagram.

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

➢ Purpose of Object Diagrams

- The purpose of a diagram should be understood clearly to implement it practically. The purposes of object diagrams are similar to class diagrams.
- The difference is that a class diagram represents an abstract model consisting of classes and their relationships. However, an object diagram represents an instance at a particular moment, which is concrete in nature.
- It means the object diagram is closer to the actual system behavior. The purpose is to capture the static view of a system at a particular moment.
- The purpose of the object diagram can be summarized as −
  - Forward and reverse engineering.
  - Object relationships of a system
  - Static view of an interaction.
  - Understand object behavior and their relationship from practical perspective.

➢ Notation


Object Name: Classifier Name
+ Attributes
B:Blog
+Title:i9-9900k
+Description:This is an news application
Article
+ Author name: Ravi
+ Title:headlines of the day
+URL:https://headlines of the day.C0m/home +Image:IMG-2.jpg
U:User
+Image_id:(5674848)
+Object_id☹AAREECG)
Utills
+Datetotime():15/6/20
+getCountry:India
+Language:English

Activity Diagram

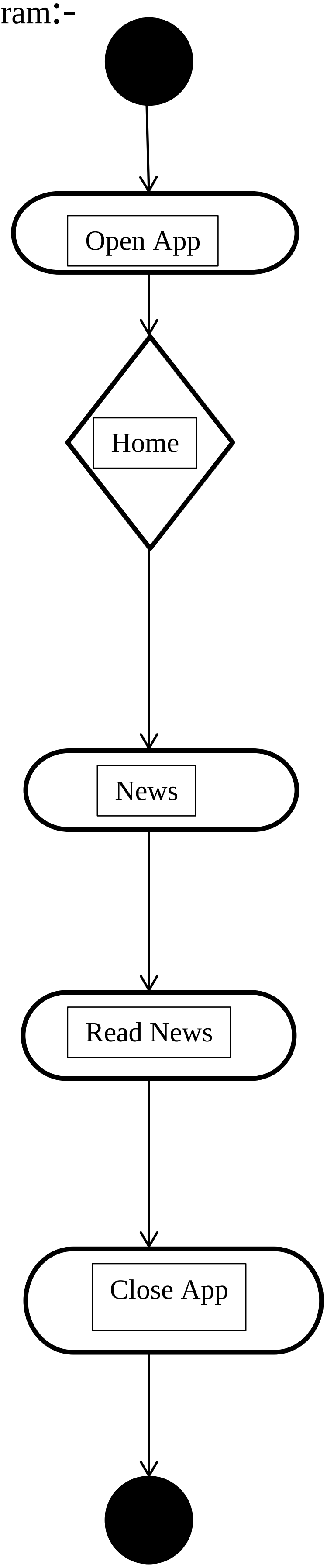
Activity diagrams are graphical representations of workflows of step wise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by- step workflows of components in a system. An activity diagram shows the overall flow of control. An activity diagram shows the overall flow of control. Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows.

Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows.

The most important shape types:

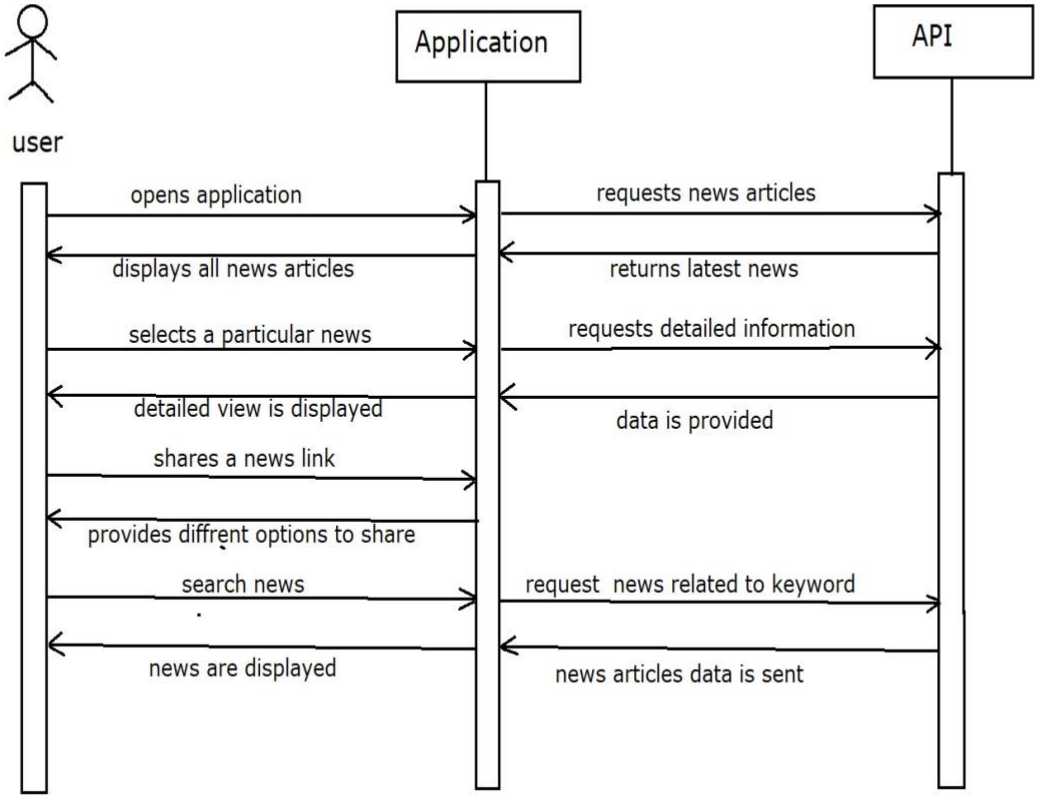
- Rounded rectangle represents activities.
- Diamonds represent decisions.
- Bars represent the start (split) or end (join) of concurrent activities.
- A black circle represents the start (initial state) of the workflow.
- An encircled black circle represents the end (final state).
- Arrows run from the start towards the end and represent the order in which activities happen.

User side Activity Diagram:-



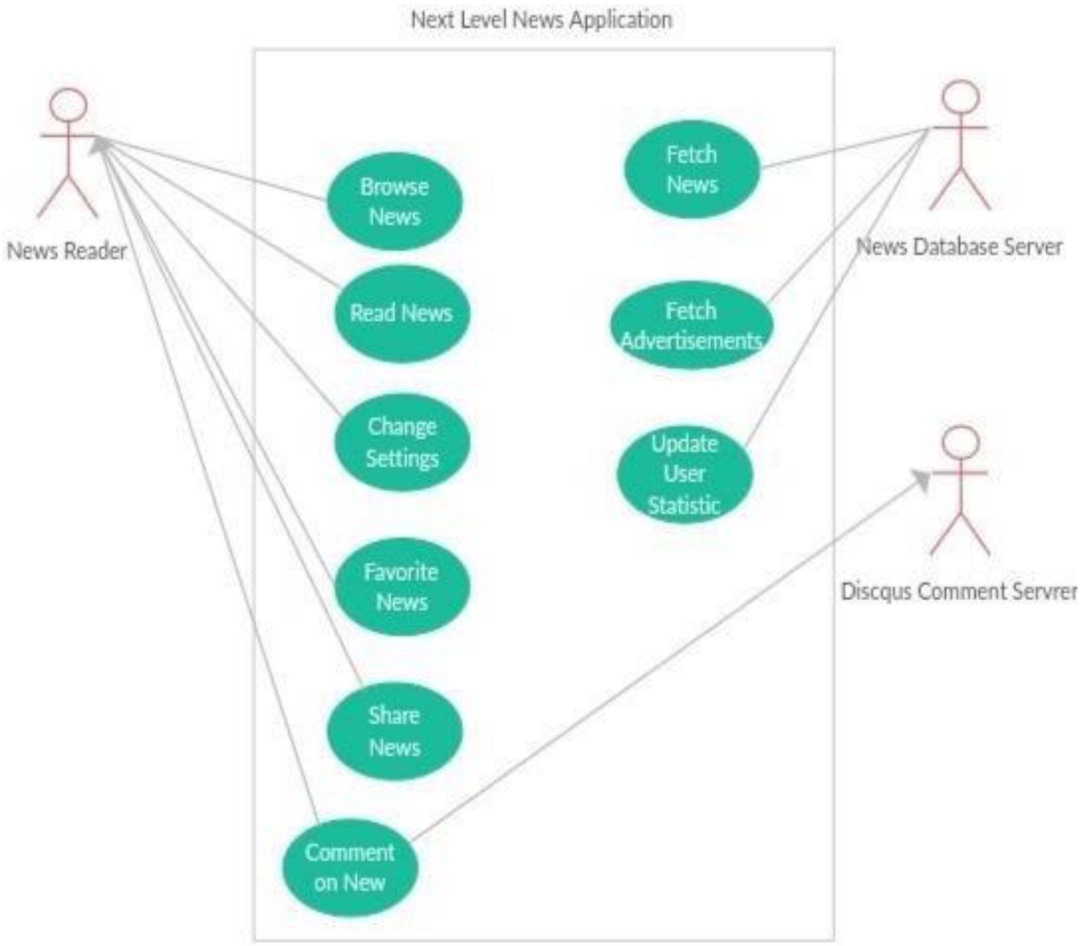
Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.


user
Application
API
opens application
requests news articles
displays all news articles
returns latest news
selects a particular news
requests detailed information
detailed view is displayed
data is provided
shares a news link
provides diffrent options to share
search news
request news related to keyword
news are displayed
news articles data is sent

Use case Diagram.

A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform.



System Design

Component Diagram

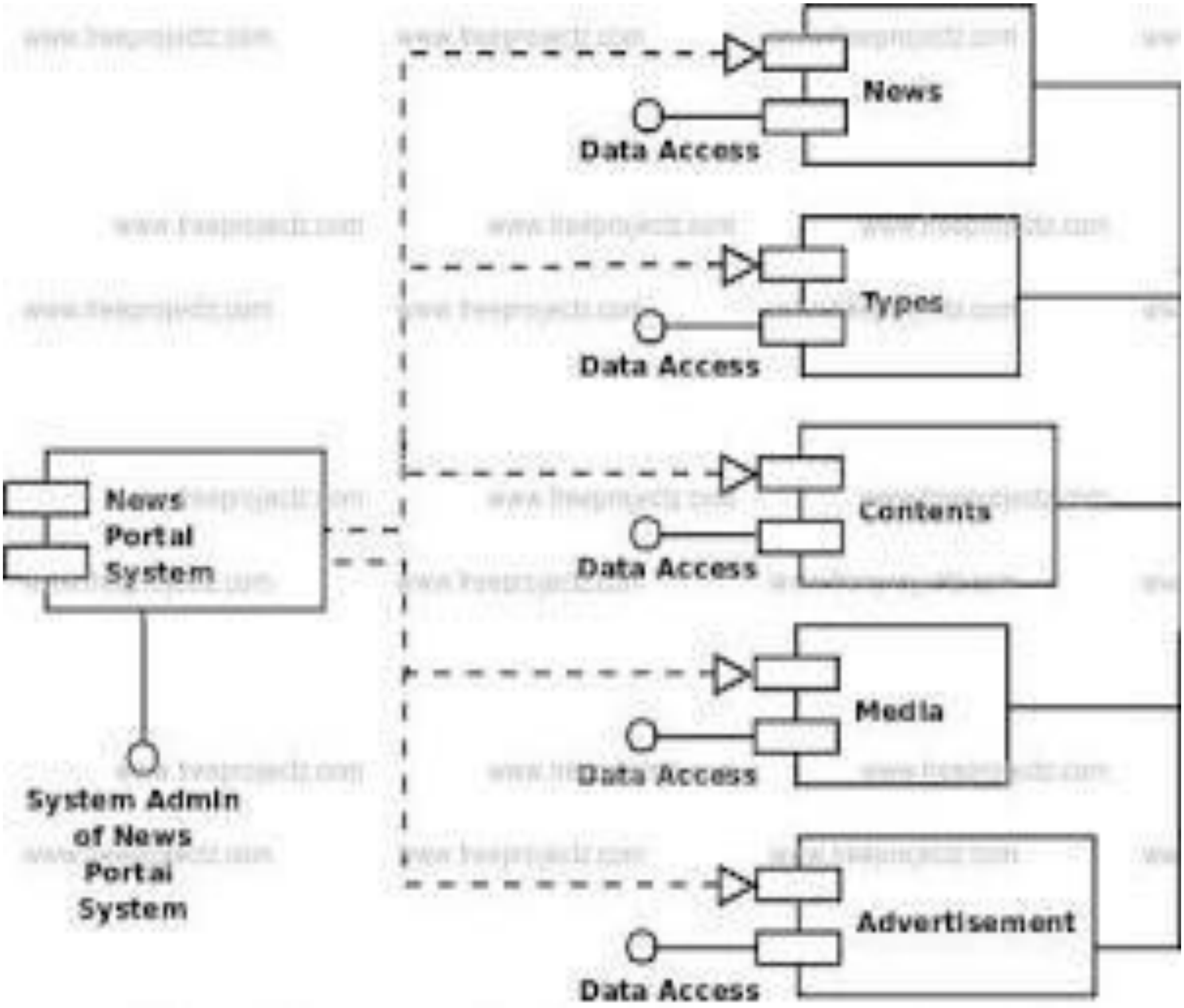
Component Diagrams describe the organization of components, including source code, run-time (binary) code, and executable. Component Diagrams:

- Give the physical view of the system interms of implementation aspect. This is important for reusability and performance purpose.
- Constitute the Components, their interfaces and realizations, and dependencies between components.

Component Diagrams are used:

- To depict organizations and dependencies among Component type.
- To show allocation of "Classes" and "objects" to components in the physical design of the system.
- To indicate the "physical layering" and "partitioning" of the system Architecture. A component typically encompasses:
- Structure and behavior of a "Collaboration of classes" from the system design.
- Interfaces that describe a group of operations implemented by components

😂


News
Data Access
Types
Data Access
News Portal System
Contents
Data Access
Media
Data Access
System Admin of News Portal System
Advertisement
Data Access
www.freeprojectz.com

Deployment Diagram

Deployment diagram is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets.

Artifacts represent concrete elements in the physical world that are the result of a development process. Deployment Diagram is usually represented by a node which is either hardware device or some software execution environment. Nodes could be connected through communication paths to create networked systems of arbitrary complexity.

Note, that component was directly deployed to nodes in UML 1.x deployment diagrams. In UML 2.x artifacts are deployed to nodes, and artifacts could manifest (implement) components. Components are deployed to nodes indirectly through artifacts.

Deployment diagrams could describe architecture at specification level (also called type level) or at instance level (similar to class diagrams and object diagrams).

Specification level: deployment diagram shows some overview of deployment of artifacts to deployment targets, without referencing specific instances of artifacts or nodes.

Instance level: deployment diagram shows deployment of instances of artifacts to specific instances of deployment targets. It could be used for example to show differences in deployments to development, staging or production.

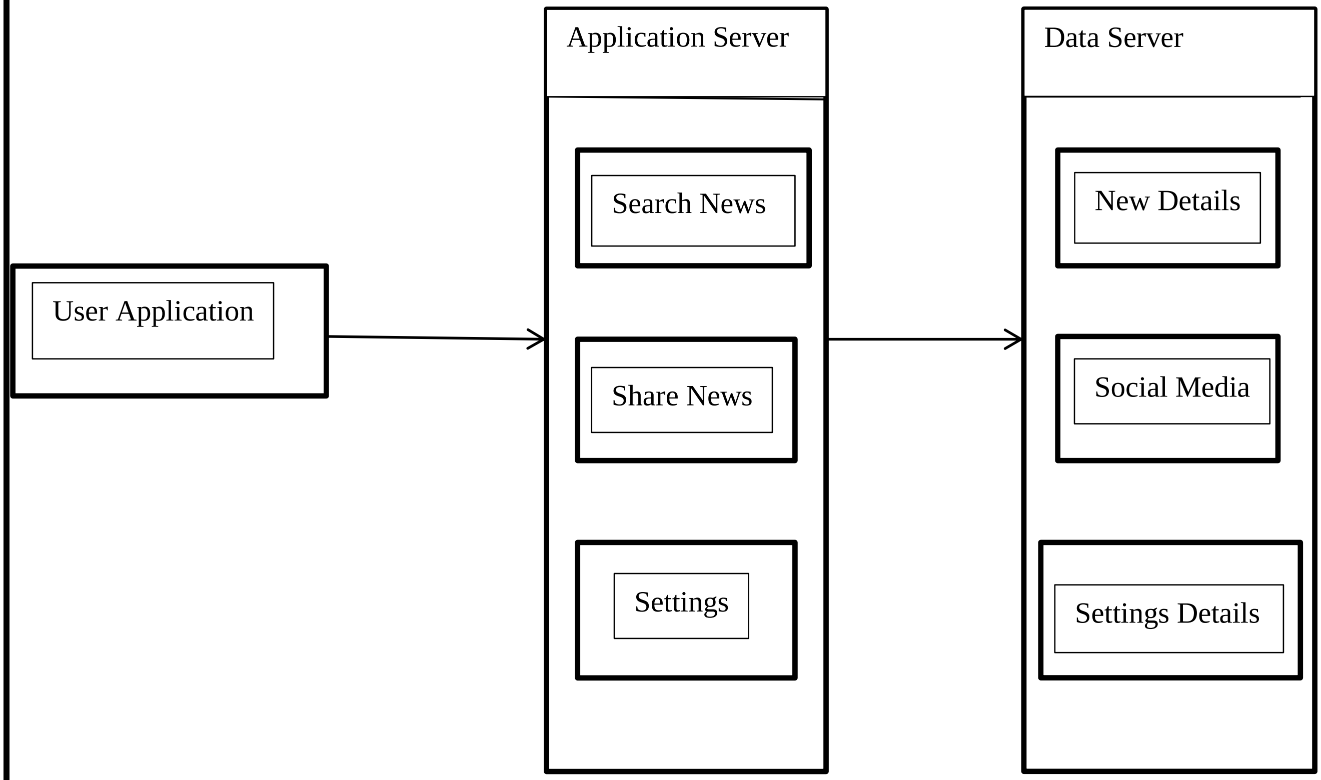

User Application
Application Server
Search News
Share News
Settings
Data Server
New Details
Social Media
Settings Details

Table Design

Table for User

Sr.No	Column	Datatype	Description
1	Content	Varchar	User can choose various contents of news to read
2	No.ofitems	Varchar(25)	User can change the no.of items in settings
3	Share	Number	User can share the news on other social platforms
4	Home	Varchar(30)	It shows the homepage of the app
5	Settings	Varchar(25)	It shows the apps settings to the user

➢ Table for Publisher

Sr.No	Column Name	Datatype	Description
1	Publishers Name	Varchar	It stores publishers name
2	Publisher_ID	Varchar	It stores publishers Id
3	Email	Varchar	It stores Email of Publisher
4	Phone Number	Integer	It stores Contact of Publisher
5	Settings	Varchar	It shows settings of the application
6	News Topics	Varchar	It shows news related to different topics.
7	Publisher details	Integer	It shows details about the publisher

➢ Table for News/blog

Sr.No	Column Name	Datatype	Description
1	News/blogs Name	Varchar	It shows Name of the News/blogs
2	Authors Name	Varchar	It shows Authors Name
3	News/blogs Date	Integer	It shows News/blogs Date
4	No.of news	Integer	It shows no.of News to display on Homepage
5	News Sections	Varchar	It shows various sections of News
6	Order By	Boolean	It shows News from oldest to newest or relevance in order
7	Text Size	Varchar	It shows TestSize

Source Code

JAVA FILES :

MainActivity.java

```
package com.example.android.newsfeed;

import android.content.Intent;
import android.os.Bundle;
import androidx.annotation.NonNull;
import com.google.android.material.navigation.NavigationView;
import com.google.android.material.tabs.TabLayout;
import androidx.core.view.GravityCompat;
import androidx.viewpager.widget.ViewPager;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;

import com.example.android.newsfeed.adapter.CategoryFragmentPagerAdapter;
import com.example.android.newsfeed.utils.Constants;

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    private ViewPager viewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        // Find the view pager that will allow the user to swipe between fragments
        viewPager = findViewById(R.id.viewpager);
        // Give the TabLayout the ViewPager
        TabLayout tabLayout = findViewById(R.id.sliding_tabs);
        tabLayout.setupWithViewPager(viewPager);
        // Set gravity for tab bar
        tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);

        NavigationView navigationView = findViewById(R.id.nav_view);
        assert navigationView != null;
        navigationView.setNavigationItemSelectedListener(this);

        // Set the default fragment when starting the app

onNavigationItemSelected(navigationView.getMenu().getItem(0).setChecked(true));

        // Set category fragment pager adapter
        CategoryFragmentPagerAdapter pagerAdapter =
                new CategoryFragmentPagerAdapter(this, getSupportFragmentManager());
        // Set the pager adapter onto the view pager
        viewPager.setAdapter(pagerAdapter);
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();

        // Switch Fragments in a ViewPager on clicking items in Navigation Drawer
        if (id == R.id.nav_home) {
            viewPager.setCurrentItem(Constants.HOME);
        } else if (id == R.id.nav_world) {
            viewPager.setCurrentItem(Constants.WORLD);
        } else if (id == R.id.nav_science) {
            viewPager.setCurrentItem(Constants.SCIENCE);
        } else if (id == R.id.nav_sport) {
            viewPager.setCurrentItem(Constants.SPORT);
        } else if (id == R.id.nav_environment) {
            viewPager.setCurrentItem(Constants.ENVIRONMENT);
        } else if (id == R.id.nav_society) {
            viewPager.setCurrentItem(Constants.SOCIETY);
        } else if (id == R.id.nav_fashion) {
            viewPager.setCurrentItem(Constants.FASHION);
        } else if (id == R.id.nav_business) {
            viewPager.setCurrentItem(Constants.BUSINESS);
        } else if (id == R.id.nav_culture) {
            viewPager.setCurrentItem(Constants.CULTURE);
        }

        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    @Override
    // Initialize the contents of the Activity's options menu
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the Options Menu we specified in XML
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    // This method is called whenever an item in the options menu is selected.
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            Intent settingsIntent = new Intent(this, SettingsActivity.class);
            startActivity(settingsIntent);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

}
```

Adapter.java

```
     package com.example.android.newsfeed.adapter;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.net.Uri;
import android.preference.PreferenceManager;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;
import android.text.Html;
import android.text.format.DateUtils;
import android.util.Log;
import android.util.TypedValue;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.bumptech.glide.Glide;
import com.example.android.newsfeed.News;
import com.example.android.newsfeed.R;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.TimeZone;

/**
 * A {@link NewsAdapter} can provide a card item layout for each news in the data
source
 * ( a list of {@link News} objects).
 */

public class NewsAdapter extends RecyclerView.Adapter<NewsAdapter.ViewHolder> {
    private Context mContext;
    private List<News> mNewsList;
    private SharedPreferences sharedPrefs;

    /**
     * Constructs a new {@link NewsAdapter}
     * @param context of the app
     * @param newsList is the list of news, which is the data source of the adapter
     */
    public NewsAdapter(Context context, List<News> newsList) {
        mContext = context;
        mNewsList = newsList;
    }

    @Override
    public NewsAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.news_card_item, parent,
false);
        return new ViewHolder(v);
    }

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

    class ViewHolder extends RecyclerView.ViewHolder {
        private TextView titleTextView;
        private TextView sectionTextView;
        private TextView authorTextView;
        private TextView dateTextView;
        private ImageView thumbnailImageView;
        private ImageView shareImageView;
        private TextView trailTextView;
        private CardView cardView;

        ViewHolder(View itemView) {
            super(itemView);
            titleTextView = itemView.findViewById(R.id.title_card);
            sectionTextView = itemView.findViewById(R.id.section_card);
            authorTextView = itemView.findViewById(R.id.author_card);
            dateTextView = itemView.findViewById(R.id.date_card);
            thumbnailImageView = itemView.findViewById(R.id.thumbnail_image_card);
            shareImageView = itemView.findViewById(R.id.share_image_card);
            trailTextView = itemView.findViewById(R.id.trail_text_card);
            cardView = itemView.findViewById(R.id.card_view);
        }
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        sharedPrefs = PreferenceManager.getDefaultSharedPreferences(mContext);

        // Change the color theme of Title TextView by using the user's stored
preferences
        setColorTheme(holder);

        // Change text size of TextView by using the user's stored preferences
        setTextSize(holder);

        // Find the current news that was clicked on
        final News currentNews = mNewsList.get(position);

        holder.titleTextView.setText(currentNews.getTitle());
        holder.sectionTextView.setText(currentNews.getSection());
        // If the author does not exist, hide the authorTextView
        if (currentNews.getAuthor() == null) {
            holder.authorTextView.setVisibility(View.GONE);
        } else {
            holder.authorTextView.setVisibility(View.VISIBLE);
            holder.authorTextView.setText(currentNews.getAuthor());
        }

        // Get time difference between the current date and web publication date and
        // set the time difference on the textView

holder.dateTextView.setText(getTimeDifference(formatDate(currentNews.getDate())));

        // Get string of the trailTextHTML and convert Html text to plain text
        // and set the plain text on the textView
        String trailTextHTML = currentNews.getTrailTextHtml();

holder.trailTextView.setText(Html.fromHtml(Html.fromHtml(trailTextHTML).toString()));

        // Set an OnClickListener to open a website with more information about the
selected article
        holder.cardView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Convert the String URL into a URI object (to pass into the Intent
constructor)
                Uri newsUri = Uri.parse(currentNews.getUrl());

                // Create a new intent to view the news URI
                Intent websiteIntent = new Intent(Intent.ACTION_VIEW, newsUri);

                // Send the intent to launch a new activity
                mContext.startActivity(websiteIntent);
            }
        });

        if (currentNews.getThumbnail() == null) {
            holder.thumbnailImageView.setVisibility(View.GONE);
        } else {
            holder.thumbnailImageView.setVisibility(View.VISIBLE);
            // Load thumbnail with glide
            Glide.with(mContext.getApplicationContext())
                    .load(currentNews.getThumbnail())
                    .into(holder.thumbnailImageView);
        }
        // Set an OnClickListener to share the data with friends via email or  social
networking
        holder.shareImageView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                shareData(currentNews);
            }
        });
    }

    /**
     * Set the user preferred color theme
     */
    private void setColorTheme(ViewHolder holder) {
        // Get the color theme string from SharedPreferences and check for the value
associated with the key
        String colorTheme = sharedPrefs.getString(
                mContext.getString(R.string.settings_color_key),
                mContext.getString(R.string.settings_color_default));

        // Change the background color of titleTextView by using the user's stored
preferences
```

```
        if
(colorTheme.equals(mContext.getString(R.string.settings_color_white_value))) {
            holder.titleTextView.setBackgroundResource(R.color.white);
            holder.titleTextView.setTextColor(Color.BLACK);
        }else if
(colorTheme.equals(mContext.getString(R.string.settings_color_sky_blue_value))) {
            holder.titleTextView.setBackgroundResource(R.color.nav_bar_start);
            holder.titleTextView.setTextColor(Color.WHITE);
        } else if
(colorTheme.equals(mContext.getString(R.string.settings_color_dark_blue_value))) {
            holder.titleTextView.setBackgroundResource(R.color.color_app_bar_text);
            holder.titleTextView.setTextColor(Color.WHITE);
        } else if
(colorTheme.equals(mContext.getString(R.string.settings_color_violet_value))) {
            holder.titleTextView.setBackgroundResource(R.color.violet);
            holder.titleTextView.setTextColor(Color.WHITE);
        } else if
(colorTheme.equals(mContext.getString(R.string.settings_color_light_green_value))) {
            holder.titleTextView.setBackgroundResource(R.color.light_green);
            holder.titleTextView.setTextColor(Color.WHITE);
        } else if
(colorTheme.equals(mContext.getString(R.string.settings_color_green_value))) {
            holder.titleTextView.setBackgroundResource(R.color.color_section);
            holder.titleTextView.setTextColor(Color.WHITE);
        }
    }

    /**
     * Set the text size to the text size the user choose.
     */
    private void setTextSize(ViewHolder holder) {
        // Get the text size string from SharedPreferences and check for the value
associated with the key
        String textSize = sharedPrefs.getString(
                mContext.getString(R.string.settings_text_size_key),
                mContext.getString(R.string.settings_text_size_default));

        // Change text size of TextView by using the user's stored preferences

if(textSize.equals(mContext.getString(R.string.settings_text_size_medium_value))) {
            holder.titleTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp22));
            holder.sectionTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp14));
            holder.trailTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp16));
            holder.authorTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp14));
            holder.dateTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp14));
        } else
if(textSize.equals(mContext.getString(R.string.settings_text_size_small_value))) {
```

```
            holder.titleTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp20));
            holder.sectionTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp12));
            holder.trailTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp14));
            holder.authorTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp12));
            holder.dateTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp12));
        } else
if(textSize.equals(mContext.getString(R.string.settings_text_size_large_value))) {
            holder.titleTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp24));
            holder.sectionTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp16));
            holder.trailTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp18));
            holder.authorTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp16));
            holder.dateTextView.setTextSize(TypedValue.COMPLEX_UNIT_PX,
                    mContext.getResources().getDimension(R.dimen.sp16));
        }
    }

    /**
     * Share the article with friends in social network
     * @param news {@link News} object
     */
    private void shareData(News news) {
        Intent sharingIntent = new Intent(Intent.ACTION_SEND);
        sharingIntent.setType("text/plain");
        sharingIntent.putExtra(android.content.Intent.EXTRA_TEXT,
                news.getTitle() + " : " + news.getUrl());
        mContext.startActivity(Intent.createChooser(sharingIntent,
                mContext.getString(R.string.share_article)));
    }

    /**
     *  Clear all data (a list of {@link News} objects)
     */
    public void clearAll() {
        mNewsList.clear();
        notifyDataSetChanged();
    }

    /**
     * Add  a list of {@link News}
     * @param newsList is the list of news, which is the data source of the adapter
     */
    public void addAll(List<News> newsList) {
        mNewsList.clear();
        mNewsList.addAll(newsList);
        notifyDataSetChanged();
    }

    /**
     * Convert date and time in UTC (webPublicationDate) into a more readable
representation
     * in Local time
     *
     * @param dateStringUTC is the web publication date of the article (i.e. 2014-02-
04T08:00:00Z)
     * @return the formatted date string in Local time(i.e "Jan 1, 2000  2:15 AM")
     * from a date and time in UTC
     */
    private String formatDate(String dateStringUTC) {
        // Parse the dateString into a Date object
        SimpleDateFormat simpleDateFormat =
                new SimpleDateFormat("yyyy-MM-dd'T'kk:mm:ss'Z'");
        Date dateObject = null;
        try {
            dateObject = simpleDateFormat.parse(dateStringUTC);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        // Initialize a SimpleDateFormat instance and configure it to provide a more
readable
        // representation according to the given format, but still in UTC
        SimpleDateFormat df = new SimpleDateFormat("MMM d, yyyy  h:mm a",
Locale.ENGLISH);
        String formattedDateUTC = df.format(dateObject);
        // Convert UTC into Local time
        df.setTimeZone(TimeZone.getTimeZone("UTC"));
        Date date = null;
        try {
            date = df.parse(formattedDateUTC);
            df.setTimeZone(TimeZone.getDefault());
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return df.format(date);
    }

    /**
     * Get the formatted web publication date string in milliseconds
     * @param formattedDate the formatted web publication date string
     * @return the formatted web publication date in milliseconds
     */
    private static long getDateInMillis(String formattedDate) {
        SimpleDateFormat simpleDateFormat =
                new SimpleDateFormat("MMM d, yyyy  h:mm a");
        long dateInMillis;
        Date dateObject;
        try {
            dateObject = simpleDateFormat.parse(formattedDate);
            dateInMillis = dateObject.getTime();
            return dateInMillis;
        } catch (ParseException e) {
            Log.e("Problem parsing date", e.getMessage());
            e.printStackTrace();
        }
        return 0;
    }

    /**
     * Get the time difference between the current date and web publication date
     * @param formattedDate the formatted web publication date string
     * @return time difference (i.e "9 hours ago")
     */
    private CharSequence getTimeDifference(String formattedDate) {
        long currentTime = System.currentTimeMillis();
        long publicationTime = getDateInMillis(formattedDate);
        return DateUtils.getRelativeTimeSpanString(publicationTime, currentTime,
                DateUtils.SECOND_IN_MILLIS);
    }
}
```

Category Fragment Adaptor.java

```
package com.example.android.newsfeed.adapter;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import android.content.Context;
import androidx.fragment.app.FragmentPagerAdapter;

import com.example.android.newsfeed.R;
import com.example.android.newsfeed.fragment.BusinessFragment;
import com.example.android.newsfeed.fragment.CultureFragment;
import com.example.android.newsfeed.fragment.EnvironmentFragment;
import com.example.android.newsfeed.fragment.FashionFragment;
import com.example.android.newsfeed.fragment.HomeFragment;
import com.example.android.newsfeed.fragment.ScienceFragment;
import com.example.android.newsfeed.fragment.SocietyFragment;
import com.example.android.newsfeed.fragment.SportFragment;
import com.example.android.newsfeed.fragment.WorldFragment;
import com.example.android.newsfeed.utils.Constants;

/**
 * Provides the appropriate {@link Fragment} for a view pager.
 */

public class CategoryFragmentPagerAdapter extends FragmentPagerAdapter {
    /** Context of the app */
    private Context mContext;

    /**
     * Create a new {@link CategoryFragmentPagerAdapter} object.
     *
     * @param context is the context of the app
     * @param fm is the fragment manager that will keep each fragment's state in the
adapter
     * across swipes.
     */
    public CategoryFragmentPagerAdapter(Context context, FragmentManager fm) {
        super(fm);
        mContext = context;
    }

    /**
     * Return the {@link Fragment} that should be displayed for the given page number.
     */
    @Override
    public Fragment getItem(int position) {
        switch (position) {
            case Constants.HOME:
                return new HomeFragment();
            case Constants.WORLD:
                return new WorldFragment();
            case Constants.SCIENCE:
                return new ScienceFragment();
            case Constants.SPORT:
                return new SportFragment();
            case Constants.ENVIRONMENT:
                return new EnvironmentFragment();
            case Constants.SOCIETY:
                return new SocietyFragment();
            case Constants.FASHION:
                return new FashionFragment();
            case Constants.BUSINESS:
                return new BusinessFragment();
            case Constants.CULTURE:
                return new CultureFragment();
            default:
                return null;
        }
    }

    /**
     * Return the total number of pages.
     */
    @Override
    public int getCount() {
        return 9;
    }

    /**
     * Return page title of the tap
     */
    @Override
    public CharSequence getPageTitle(int position) {
        int titleResId;
        switch (position) {
            case Constants.HOME:
                titleResId = R.string.ic_title_home;
                break;
            case Constants.WORLD:
                titleResId = R.string.ic_title_world;
                break;
            case Constants.SCIENCE:
                titleResId = R.string.ic_title_science;
                break;
            case Constants.SPORT:
                titleResId = R.string.ic_title_sport;
                break;
            case Constants.ENVIRONMENT:
                titleResId = R.string.ic_title_environment;
                break;
            case Constants.SOCIETY:
                titleResId = R.string.ic_title_society;
                break;
            case Constants.FASHION:
                titleResId = R.string.ic_title_fashion;
                break;
            case Constants.BUSINESS:
                titleResId = R.string.ic_title_business;
                break;
            default:
                titleResId = R.string.ic_title_culture;
                break;
        }
        return mContext.getString(titleResId);
    }
}
```

News.java

```
public class News {

    /** Title of the article */
    private String mTitle;

    /** Section name of the article*/
    private String mSection;
    /** Author name in the article */
    private String mAuthor;

    /** Web publication date of the article */
    private String mDate;

    /** Website URL of the article */
    private String mUrl;

    /** Thumbnail of the article */
    private String mThumbnail;

    /** TrailText of the article with string type Html */
    private String mTrailTextHtml;

    /**
     * Constructs a new {@link News} object.
     *
     * @param title is the title of the article
     * @param section is the section name of the article
     * @param author is author name in article
     * @param date is the web publication date of the article
     * @param url is the website URL to find more details about the article
     * @param thumbnail is the thumbnail of the article
     * @param trailText is trail text of the article with string type Html
     */
    public News(String title, String section, String author, String date, String url,
String thumbnail, String trailText) {
        mTitle = title;
        mSection = section;
        mAuthor = author;
        mDate = date;
        mUrl = url;
        mThumbnail = thumbnail;
        mTrailTextHtml = trailText;
    }

    /**
     * Returns the title of the article
     */
    public String getTitle() {
        return mTitle;
    }

    /**
     * Returns the section name of the article.
     */
    public String getSection() {
        return mSection;
    }
    /**
     * Returns the author name of the article.
     */
    public String getAuthor() {
        return mAuthor;
    }
    /**
     * Returns the web publication date of the article.
     */
    public String getDate() {
        return mDate;
    }

    /**
     * Returns the website URL to find more information about the news.
     */
    public String getUrl() {
        return mUrl;
    }

    /**
     * Returns the thumbnail of the article
     */
    public String getThumbnail() {
        return mThumbnail;
    }

    /**
     * Returns the TrailText of the article with string type Html
     */
    public String getTrailTextHtml() {
        return mTrailTextHtml;
    }
}
```

NewsPreferences.java

```
package com.example.android.newsfeed;

import android.content.Context;
import android.content.SharedPreferences;
import android.net.Uri;
import android.preference.PreferenceManager;

import com.example.android.newsfeed.utils.Constants;

import static com.example.android.newsfeed.utils.Constants.API_KEY;
import static com.example.android.newsfeed.utils.Constants.API_KEY_PARAM;
import static com.example.android.newsfeed.utils.Constants.FORMAT;
import static com.example.android.newsfeed.utils.Constants.FORMAT_PARAM;
import static com.example.android.newsfeed.utils.Constants.FROM_DATE_PARAM;
import static com.example.android.newsfeed.utils.Constants.ORDER_BY_PARAM;
import static com.example.android.newsfeed.utils.Constants.ORDER_DATE_PARAM;
import static com.example.android.newsfeed.utils.Constants.PAGE_SIZE_PARAM;
import static com.example.android.newsfeed.utils.Constants.QUERY_PARAM;
import static com.example.android.newsfeed.utils.Constants.SECTION_PARAM;
import static com.example.android.newsfeed.utils.Constants.SHOW_FIELDS;
import static com.example.android.newsfeed.utils.Constants.SHOW_FIELDS_PARAM;
import static com.example.android.newsfeed.utils.Constants.SHOW_TAGS;
import static com.example.android.newsfeed.utils.Constants.SHOW_TAGS_PARAM;

public final class NewsPreferences {

    /**
     * Get Uri.Builder based on stored SharedPreferences.
     * @param context Context used to access SharedPreferences
     * @return Uri.Builder
     */
    public static Uri.Builder getPreferredUri(Context context) {
        SharedPreferences sharedPrefs =
PreferenceManager.getDefaultSharedPreferences(context);

        // getString retrieves a String value from the preferences. The second
parameter is the
        // default value for this preference.
        String numOfItems = sharedPrefs.getString(
                context.getString(R.string.settings_number_of_items_key),
                context.getString(R.string.settings_number_of_items_default));

        // Get the information from SharedPreferences and check for the value
associated with the key
        String orderBy = sharedPrefs.getString(
                context.getString(R.string.settings_order_by_key),
                context.getString(R.string.settings_order_by_default));

        // Get the orderDate information from SharedPreferences and check for the
value associated with the key
        String orderDate = sharedPrefs.getString(
                context.getString(R.string.settings_order_date_key),
                context.getString(R.string.settings_order_date_default));

        // Get the fromDate information from SharedPreferences and check for the value
associated with the key
        String fromDate = sharedPrefs.getString(
                context.getString(R.string.settings_from_date_key),
                context.getString(R.string.settings_from_date_default));

        // Parse breaks apart the URI string that is passed into its parameter
        Uri baseUri = Uri.parse(Constants.NEWS_REQUEST_URL);

        // buildUpon prepares the baseUri that we just parsed so we can add query
parameters to it
        Uri.Builder uriBuilder = baseUri.buildUpon();
        // Append query parameter and its value. (e.g. the 'show-tag=contributor')
        uriBuilder.appendQueryParameter(QUERY_PARAM, "");
        uriBuilder.appendQueryParameter(ORDER_BY_PARAM, orderBy);
        uriBuilder.appendQueryParameter(PAGE_SIZE_PARAM, numOfItems);
        uriBuilder.appendQueryParameter(ORDER_DATE_PARAM, orderDate);
        uriBuilder.appendQueryParameter(FROM_DATE_PARAM, fromDate);
        uriBuilder.appendQueryParameter(SHOW_FIELDS_PARAM, SHOW_FIELDS);
        uriBuilder.appendQueryParameter(FORMAT_PARAM, FORMAT);
        uriBuilder.appendQueryParameter(SHOW_TAGS_PARAM, SHOW_TAGS);
        uriBuilder.appendQueryParameter(API_KEY_PARAM, API_KEY); // Use your API key
when API rate limit exceeded

        return uriBuilder;
    }

    /**
     * Returns String Url for query
     * @param context Context used to access getPreferredUri method
     * @param section News section
     */
    public static String getPreferredUrl(Context context, String section) {
        Uri.Builder uriBuilder = getPreferredUri(context);
        return uriBuilder.appendQueryParameter(SECTION_PARAM, section).toString();
    }
}
```

SettingsActivity.java

```
package com.example.android.newsfeed;

import android.app.DatePickerDialog;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceFragment;
import android.preference.PreferenceManager;
import androidx.appcompat.app.AppCompatActivity;
import android.view.MenuItem;
import android.widget.DatePicker;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

/**
 * The SettingsActivity is the activity that appears when a settings icon is clicked
on.
 */
```

```
public class SettingsActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.settings_activity);

        // Navigate with the app icon in the action bar
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
    }

    /**
     * The NewsPreferenceFragment implements the Preference.OnPreferenceChangeListener
interface
     * to set up to listen for any Preference changes made by the user.
     * And the NewsPreferenceFragment also implements the
DatePickerDialog.OnDateSetListener to
     * receive a callback when the user has finished selecting a date.
     */
    public static class NewsPreferenceFragment extends PreferenceFragment
            implements Preference.OnPreferenceChangeListener,
DatePickerDialog.OnDateSetListener {

        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.settings_main);

            // Find the preference for number of items
            Preference numOfItems =
findPreference(getString(R.string.settings_number_of_items_key));
            // bind the current preference value to be displayed
            bindPreferenceSummaryToValue(numOfItems);

            // Find the "order by" Preference object according to its key
            Preference orderBy =
findPreference(getString(R.string.settings_order_by_key));
            // Update the summary so that it displays the current value stored in
SharedPreferences
            bindPreferenceSummaryToValue(orderBy);

            // Find the "order date" Preference object according to its key
            Preference orderDate =
findPreference(getString(R.string.settings_order_date_key));
            // Update the summary so that it displays the current value stored in
SharedPreferences
            bindPreferenceSummaryToValue(orderDate);

            // Find the "color theme" Preference object according to its key
            Preference colorTheme =
findPreference(getString(R.string.settings_color_key));
```

```
            // Update the summary so that it displays the current value stored in
SharedPreferences
            bindPreferenceSummaryToValue(colorTheme);

            // Find the "text size" Preference object according to its key
            Preference textSize =
findPreference(getString(R.string.settings_text_size_key));
            // Update the summary so that it displays the current value stored in
SharedPreferences
            bindPreferenceSummaryToValue(textSize);

            // Find the "from date" Preference object according to its key
            Preference fromDate =
findPreference(getString(R.string.settings_from_date_key));
            setOnPreferenceClick(fromDate);
            // bind the current preference value to be displayed
            bindPreferenceSummaryToValue(fromDate);
        }

        /**
         * This method is called when the user has clicked a Preference.
         */
        private void setOnPreferenceClick(Preference preference) {
            preference.setOnPreferenceClickListener(new
Preference.OnPreferenceClickListener() {
                @Override
                public boolean onPreferenceClick(Preference preference) {
                    String key = preference.getKey();
                    if
(key.equalsIgnoreCase(getString(R.string.settings_from_date_key))) {
                        showDatePicker();
                    }
                    return false;
                }
            });
        }

        /**
         * Show the current date as the default date in the picker
         */
        private void showDatePicker() {
            Calendar calendar = Calendar.getInstance();
            int year = calendar.get(Calendar.YEAR);
            int month = calendar.get(Calendar.MONTH);
            int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);
            new DatePickerDialog(getActivity(),
                    this, year, month, dayOfMonth).show();
        }

        @Override
        public void onDateSet(DatePicker datePicker, int year, int month, int
dayOfMonth) {
```

```
            // Since it starts counting the months from 0, add one to the month value.
            month = month + 1;
            // Date the user selected
            String selectedDate = year + "-" + month + "-" + dayOfMonth;
            // Convert selected date string(i.e. "2017-2-1" into formatted date
string(i.e. "2017-02-01")
            String formattedDate = formatDate(selectedDate);

            // Storing selected date
            SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(getActivity());
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString(getString(R.string.settings_from_date_key),
formattedDate).apply();

            // Update the displayed preference summary after it has been changed
            Preference fromDatePreference =
findPreference(getString(R.string.settings_from_date_key));
            bindPreferenceSummaryToValue(fromDatePreference);
        }

        /**
         * This method is called when the user has changed a Preference.
         * Update the displayed preference summary (the UI) after it has been changed.
         * @param preference the changed Preference
         * @param value the new value of the Preference
         * @return True to update the state of the Preference with the new value
         */
        @Override
        public boolean onPreferenceChange(Preference preference, Object value) {
            String stringValue = value.toString();
            // Update the summary of a ListPreference using the label
            if (preference instanceof ListPreference) {
                ListPreference listPreference = (ListPreference) preference;
                int prefIndex = listPreference.findIndexOfValue(stringValue);
                if (prefIndex >= 0) {
                    CharSequence[] labels = listPreference.getEntries();
                    preference.setSummary(labels[prefIndex]);
                }
            } else {
                preference.setSummary(stringValue);
            }
            return true;
        }

        /**
         * Set this fragment as the OnPreferenceChangeListener and
         * bind the value that is in SharedPreferences to what will show up in the
preference summary
         */
        private void bindPreferenceSummaryToValue(Preference preference) {
            // Set the current NewsPreferenceFragment instance to listen for changes
to the preference
            // we pass in using
            preference.setOnPreferenceChangeListener(this);

            // Read the current value of the preference stored in the
SharedPreferences on the device,
            // and display that in the preference summary
            SharedPreferences preferences =

PreferenceManager.getDefaultSharedPreferences(preference.getContext());
            String preferenceString = preferences.getString(preference.getKey(), "");
            onPreferenceChange(preference, preferenceString);
        }

        /**
         * Convert selected date string(i.e. "2017-2-1" into formatted date
string(i.e. "2017-02-01")
         *
         * @param dateString is the selected date from the DatePicker
         * @return the formatted date string
         */
        private String formatDate(String dateString) {
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-M-d");
            Date dateObject = null;
            try {
                dateObject = simpleDateFormat.parse(dateString);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            SimpleDateFormat df= new SimpleDateFormat("yyyy-MM-dd");
            return df.format(dateObject);
        }
    }

    // Go back to the MainActivity when up button in action bar is clicked on.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Utilis.java

```java
package com.example.android.newsfeed.utils;

/**
 * Store Constants for the NewsFeed app.
 */

public class Constants {

    /**
     * Create a private constructor because no one should ever create a {@link
Constants} object.
     */
    private Constants() {
    }

    /**  Extract the key associated with the JSONObject */
    static final String JSON_KEY_RESPONSE = "response";
    static final String JSON_KEY_RESULTS = "results";
    static final String JSON_KEY_WEB_TITLE = "webTitle";
    static final String JSON_KEY_SECTION_NAME = "sectionName";
    static final String JSON_KEY_WEB_PUBLICATION_DATE = "webPublicationDate";
    static final String JSON_KEY_WEB_URL = "webUrl";
    static final String JSON_KEY_TAGS = "tags";
    static final String JSON_KEY_FIELDS = "fields";
    static final String JSON_KEY_THUMBNAIL = "thumbnail";
    static final String JSON_KEY_TRAIL_TEXT = "trailText";

    /** Read timeout for setting up the HTTP request */
    static final int READ_TIMEOUT = 10000; /* milliseconds */

    /** Connect timeout for setting up the HTTP request */
    static final int CONNECT_TIMEOUT = 15000; /* milliseconds */

    /** HTTP response code when the request is successful */
    static final int SUCCESS_RESPONSE_CODE = 200;

    /** Request method type "GET" for reading information from the server */
    static final String REQUEST_METHOD_GET = "GET";

    /** URL for news data from the guardian data set */
    public static final String NEWS_REQUEST_URL =
"https://content.guardianapis.com/search";

    /** Parameters */
    public static final String QUERY_PARAM = "q";
    public static final String ORDER_BY_PARAM = "order-by";
    public static final String PAGE_SIZE_PARAM = "page-size";
    public static final String ORDER_DATE_PARAM = "order-date";
    public static final String FROM_DATE_PARAM = "from-date";
    public static final String SHOW_FIELDS_PARAM = "show-fields";
    public static final String FORMAT_PARAM = "format";
    public static final String SHOW_TAGS_PARAM = "show-tags";
    public static final String API_KEY_PARAM = "api-key";
    public static final String SECTION_PARAM = "section";

    /** The show fields we want our API to return */
    public static final String SHOW_FIELDS = "thumbnail,trailText";

    /** The format we want our API to return */
    public static final String FORMAT = "json";

    /** The show tags we want our API to return */
    public static final String SHOW_TAGS = "contributor";

    /** API Key */
    public static final String API_KEY = "test"; // Use your API Key when API rate
limit exceeded

    /** Default number to set the image on the top of the textView */
    public static final int DEFAULT_NUMBER = 0;

    /** Constants value for each fragment */
    public static final int HOME = 0;
    public static final int WORLD = 1;
    public static final int SCIENCE = 2;
    public static final int SPORT = 3;
    public static final int ENVIRONMENT = 4;
    public static final int SOCIETY = 5;
    public static final int FASHION = 6;
    public static final int BUSINESS = 7;
    public static final int CULTURE = 8;

}
```

XML FILES

Activity_Main.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<androidx.drawerlayout.widget.DrawerLayout tools:openDrawer="start" android:fitsSystemWindows="true"
android:layout_height="match_parent" android:layout_width="match_parent" android:id="@+id/drawer_layout"
xmlns:tools="http://schemas.android.com/tools" xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:android="http://schemas.android.com/apk/res/android">
```

Content_Main.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<androidx.constraintlayout.widget.ConstraintLayout tools:showIn="@layout/app_bar_main"
tools:context="com.example.android.newsfeed.MainActivity"
app:layout_behavior="@string/appbar_scrolling_view_behavior" android:layout_height="match_parent"
android:layout_width="match_parent" xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:android="http://schemas.android.com/apk/res/android"><androidx.viewpager.widget.ViewPager
android:layout_height="match_parent" android:layout_width="match_parent"
android:id="@+id/viewpager"><FrameLayout android:layout_height="match_parent"
android:layout_width="match_parent" android:id="@+id/content_frame">
</FrameLayout></androidx.viewpager.widget.ViewPager></androidx.constraintlayout.widget.ConstraintLayout>
```

Fragment_Home.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout android:layout_height="match_parent" android:layout_width="match_parent"
xmlns:android="http://schemas.android.com/apk/res/android"><androidx.swiperefreshlayout.widget.SwipeRefres
hLayout android:layout_height="match_parent" android:layout_width="match_parent"
android:id="@+id/swipe_refresh">
<!-- Replaced android.support.v7.widget.RecyclerView with the new EmptyRecyclerView -->
<com.example.android.newsfeed.EmptyRecyclerView android:layout_height="match_parent"
android:layout_width="match_parent"
android:id="@+id/recycler_view"/></androidx.swiperefreshlayout.widget.SwipeRefreshLayout>
<!-- Empty view is only visible when the list has no items. -->
<TextView android:layout_height="wrap_content" android:layout_width="wrap_content"
android:id="@+id/empty_view" android:textAppearance="?android:textAppearanceMedium"
android:gravity="center" android:layout_centerInParent="true"/>
<!-- Loading indicator is only shown before the first load -->
<ProgressBar android:layout_height="wrap_content" android:layout_width="wrap_content"
android:id="@+id/loading_indicator" android:layout_centerInParent="true"
style="@style/Widget.AppCompat.ProgressBar"/></RelativeLayout>
```

News_Item.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout android:orientation="vertical" android:layout_height="wrap_content"
android:layout_width="match_parent" xmlns:tools="http://schemas.android.com/tools"
xmlns:card_view="http://schemas.android.com/apk/res-auto"
xmlns:android="http://schemas.android.com/apk/res/android"><androidx.cardview.widget.CardView
android:layout_height="wrap_content" android:layout_width="match_parent"
card_view:cardCornerRadius="@dimen/card_corner_radius"
```

```
android:foreground="?android:attr/selectableItemBackground" android:focusable="true"
android:clickable="true" android:layout_margin="@dimen/layout_margin_5"
android:id="@+id/card_view"><LinearLayout android:orientation="vertical"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_marginBottom="@dimen/layout_margin"><TextView android:layout_height="wrap_content"
android:layout_width="match_parent" android:id="@+id/title_card" tools:text="title" android:textStyle="bold"
android:textAppearance="?android:textAppearanceLarge" style="@style/TitleTextViewStyle"/><RelativeLayout
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_marginBottom="@dimen/layout_margin_8"
android:layout_marginTop="@dimen/layout_margin_4"><TextView android:id="@+id/section_card"
tools:text="section" style="@style/SectionTextViewStyle"
android:layout_toStartOf="@id/thumbnail_image_card" android:layout_alignParentStart="true"/><TextView
android:layout_height="wrap_content" android:layout_width="wrap_content" android:id="@+id/trail_text_card"
tools:text="trailText" style="@style/TrailTextViewStyle"
android:layout_toStartOf="@+id/thumbnail_image_card" android:layout_alignParentStart="true"
android:layout_below="@+id/section_card"/><ImageView
android:layout_height="@dimen/thumbnail_image_height"
android:layout_width="@dimen/thumbnail_image_width" android:id="@+id/thumbnail_image_card"
android:scaleType="centerCrop" android:contentDescription="@string/image_des"
android:layout_alignParentEnd="true"/></RelativeLayout><TextView android:id="@+id/author_card"
tools:text="author" style="@style/AuthorTextViewStyle"/><RelativeLayout
android:layout_height="wrap_content" android:layout_width="wrap_content"><TextView
android:id="@+id/date_card" tools:text="date" style="@style/DateTextViewStyle"
android:layout_toStartOf="@+id/share_image_card" android:layout_alignParentStart="true"/><ImageView
android:layout_height="@dimen/image_share" android:layout_width="@dimen/image_share"
android:id="@+id/share_image_card" android:contentDescription="@string/image_des_ic_share"
android:layout_alignParentEnd="true" android:src="@drawable/ic_share_black_18dp"
android:background="@drawable/image_button_style"
android:layout_marginEnd="@dimen/layout_margin"/></RelativeLayout></LinearLayout></androidx.cardview.wi
dget.CardView></RelativeLayout>
```

Settings_Activity.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<fragment tools:context="com.example.android.newsfeed.SettingsActivity"
android:layout_height="match_parent" android:layout_width="match_parent"
android:name="com.example.android.newsfeed.SettingsActivity$NewsPreferenceFragment"
android:id="@+id/fragment_settings" xmlns:tools="http://schemas.android.com/tools"
xmlns:android="http://schemas.android.com/apk/res/android"> </fragment>
```

System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of blackbox testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s).

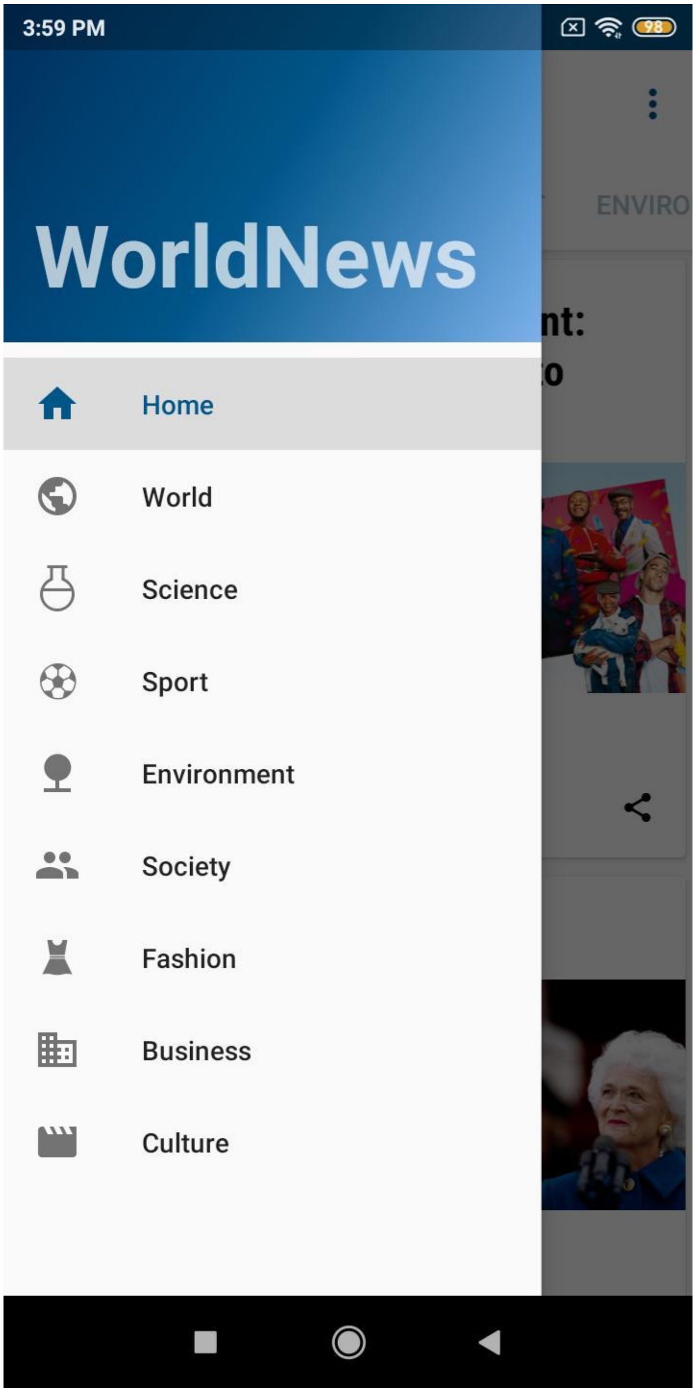
The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

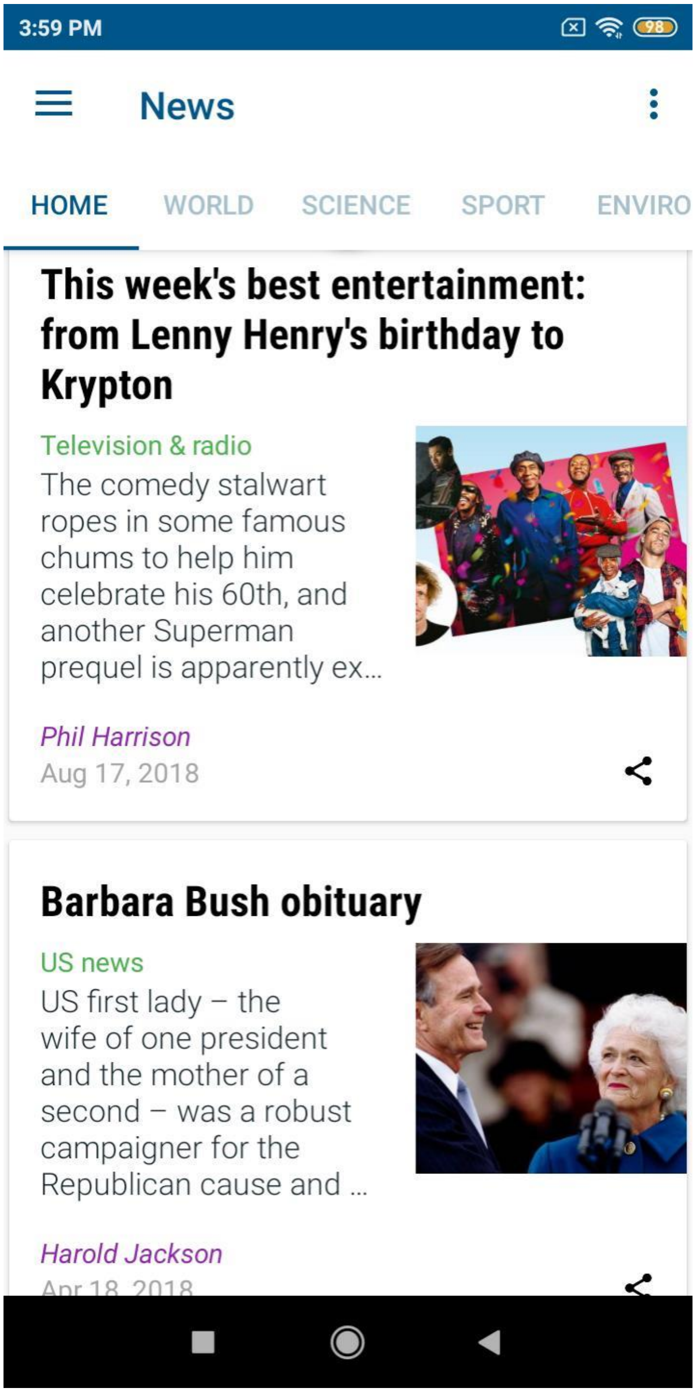
Testing the whole system System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer.

It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

Result(Screen Shot)

Home Page





This week's best entertainment: from Lenny Henry's birthday to Krypton

Television & radio

The comedy stalwart ropes in some famous chums to help him celebrate his 60th, and another Superman prequel is apparently ex…

Phil Harrison

Aug 17, 2018

Barbara Bush obituary

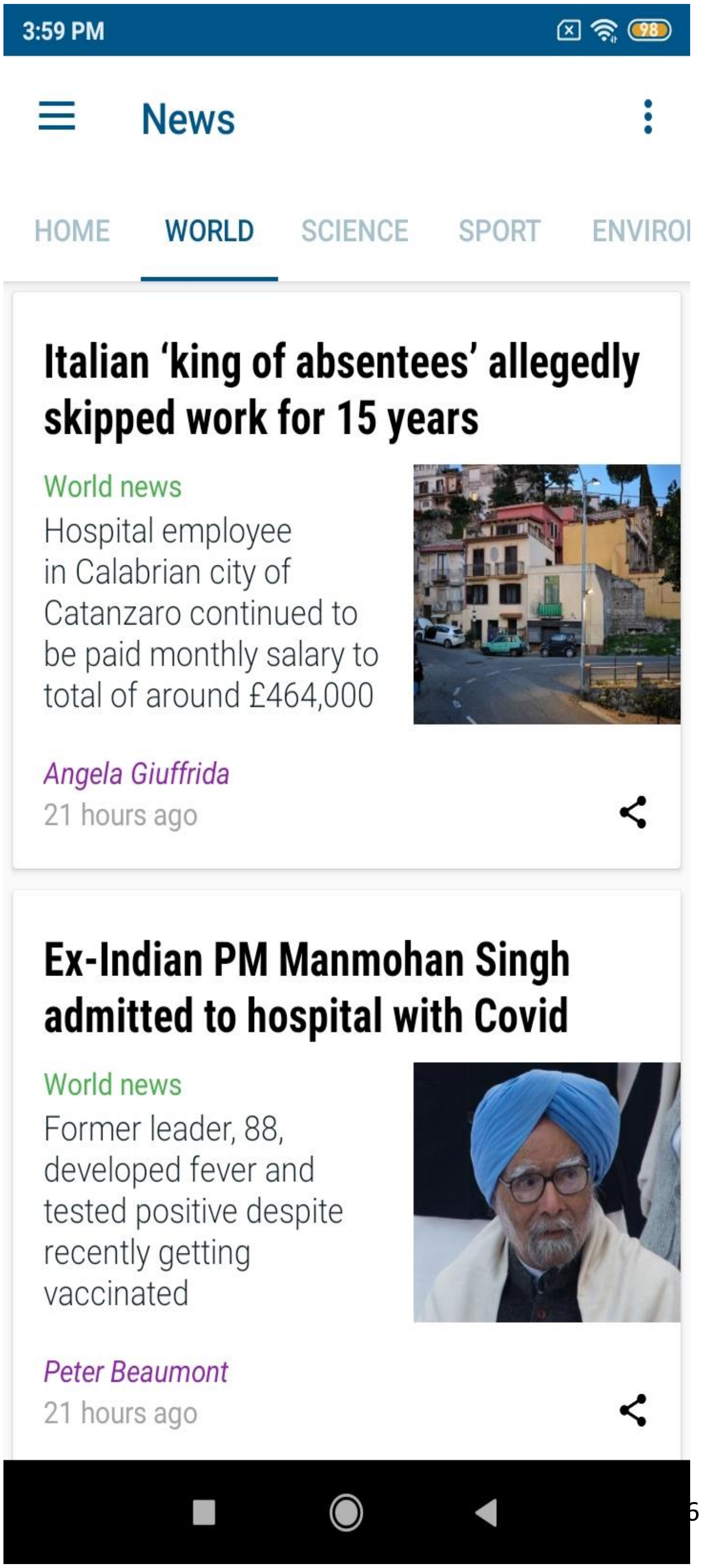
US news

US first lady – the wife of one president and the mother of a second – was a robust campaigner for the Republican cause and …

Harold Jackson

Apr 18, 2018

News Sections



Italian 'king of absentees' allegedly skipped work for 15 years

World news

Hospital employee in Calabrian city of Catanzaro continued to be paid monthly salary to total of around £464,000

Angela Giuffrida

21 hours ago

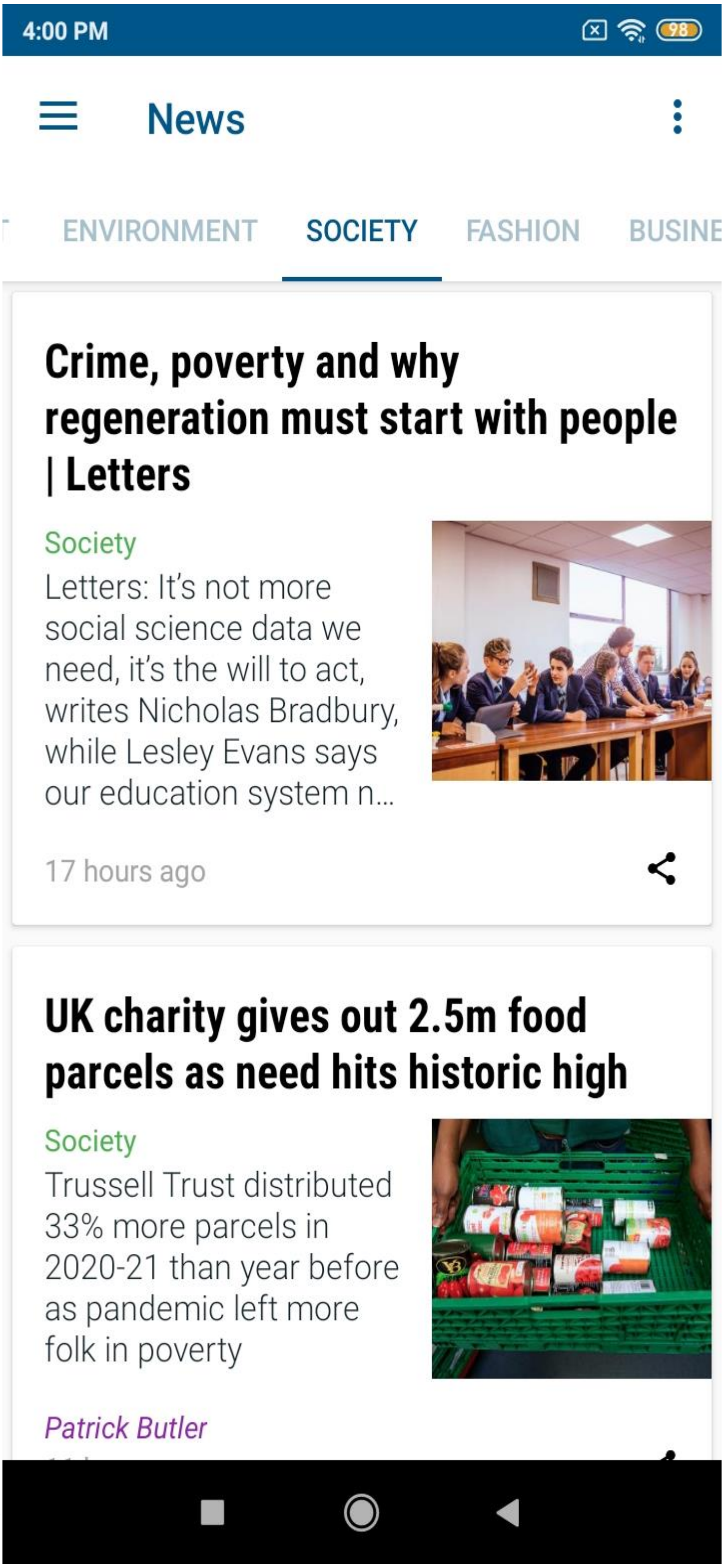
Ex-Indian PM Manmohan Singh admitted to hospital with Covid

World news

Former leader, 88, developed fever and tested positive despite recently getting vaccinated

Peter Beaumont

21 hours ago



Crime, poverty and why regeneration must start with people | Letters

Society

Letters: It's not more social science data we need, it's the will to act, writes Nicholas Bradbury, while Lesley Evans says our education system n…

17 hours ago

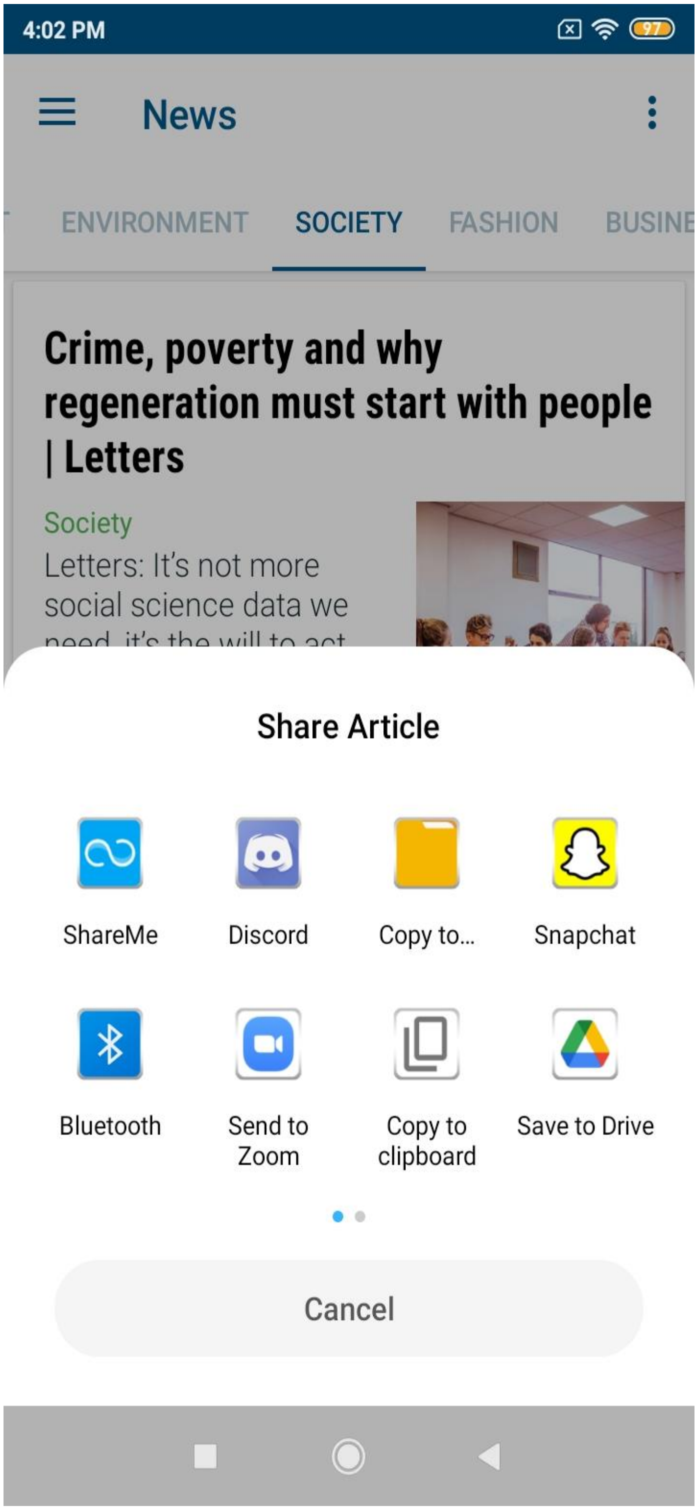
UK charity gives out 2.5m food parcels as need hits historic high

Society

Trussell Trust distributed 33% more parcels in 2020-21 than year before as pandemic left more folk in poverty

Patrick Butler

Share



4:02 PM

News

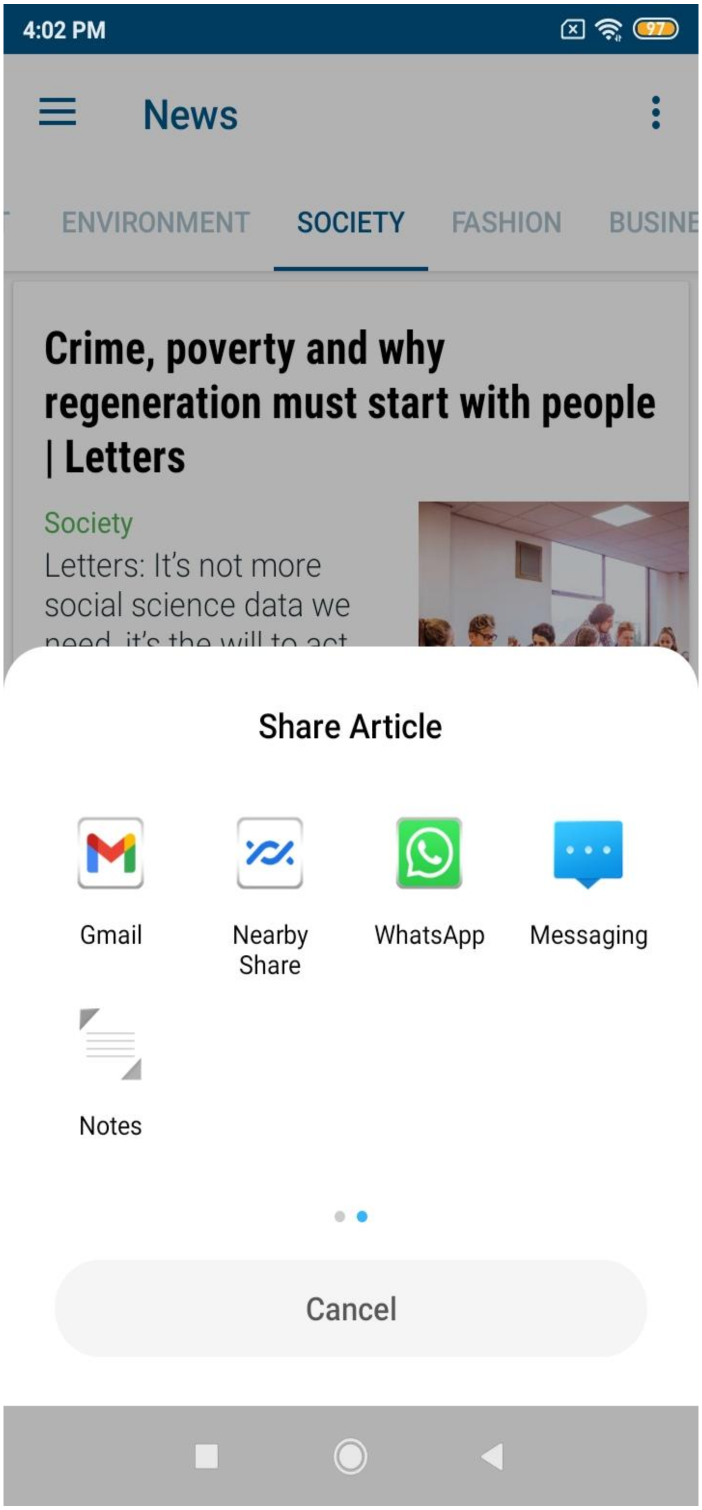
ENVIRONMENT SOCIETY FASHION BUSINE

Crime, poverty and why regeneration must start with people | Letters

Society

Letters: It's not more social science data we need, it's the will to act

Share Article



4:02 PM

News

ENVIRONMENT SOCIETY FASHION BUSINE

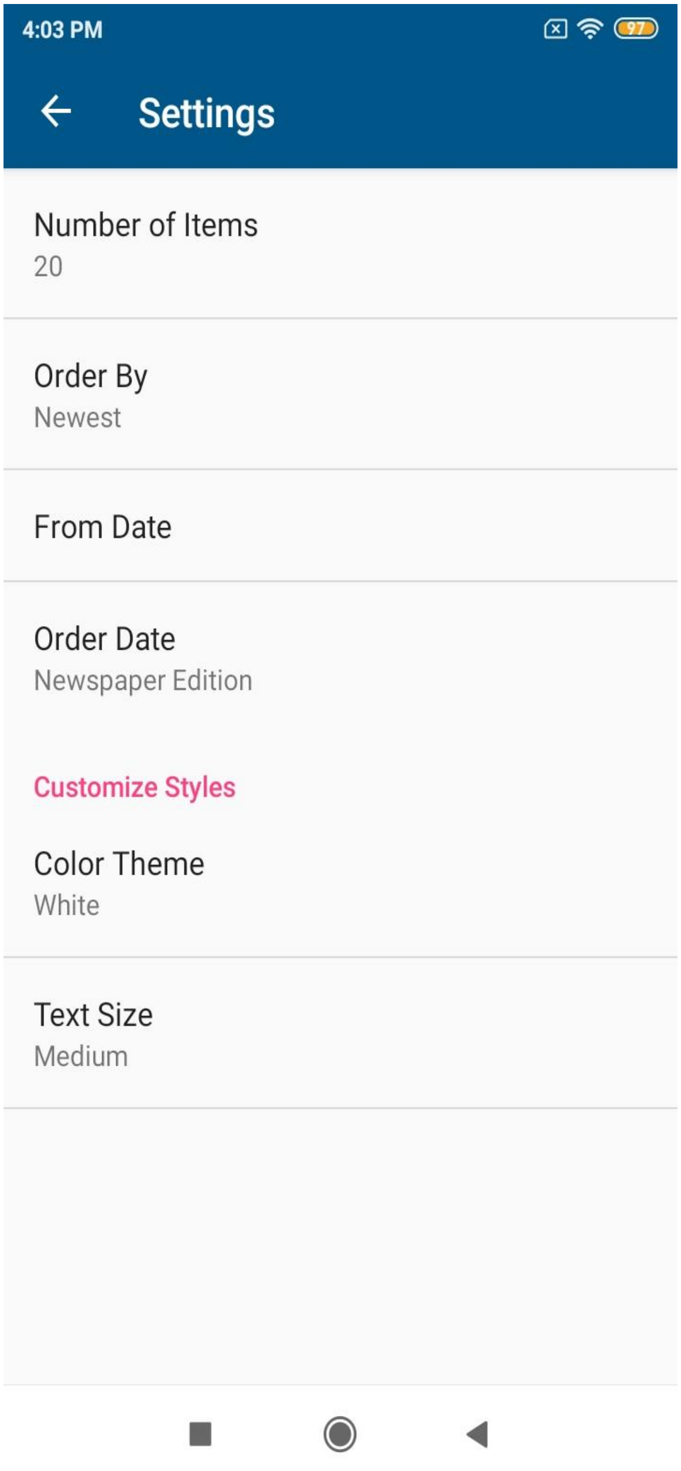
Crime, poverty and why regeneration must start with people | Letters

Society

Letters: It's not more social science data we need, it's the will to act

Share Article

Settings



Settings

Number of Items
20

Order By
Newest

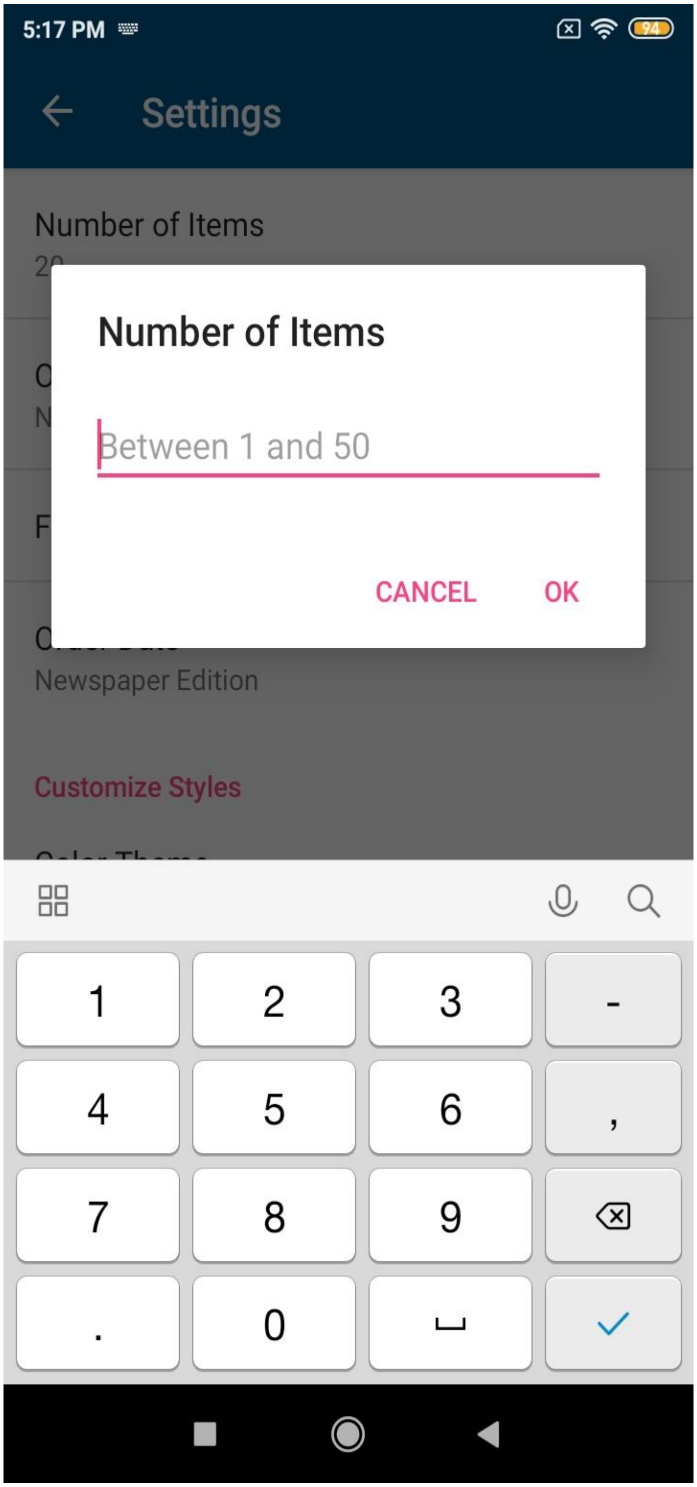
From Date

Order Date
Newspaper Edition

Customize Styles

Color Theme
White

Text Size
Medium



Settings

Number of Items

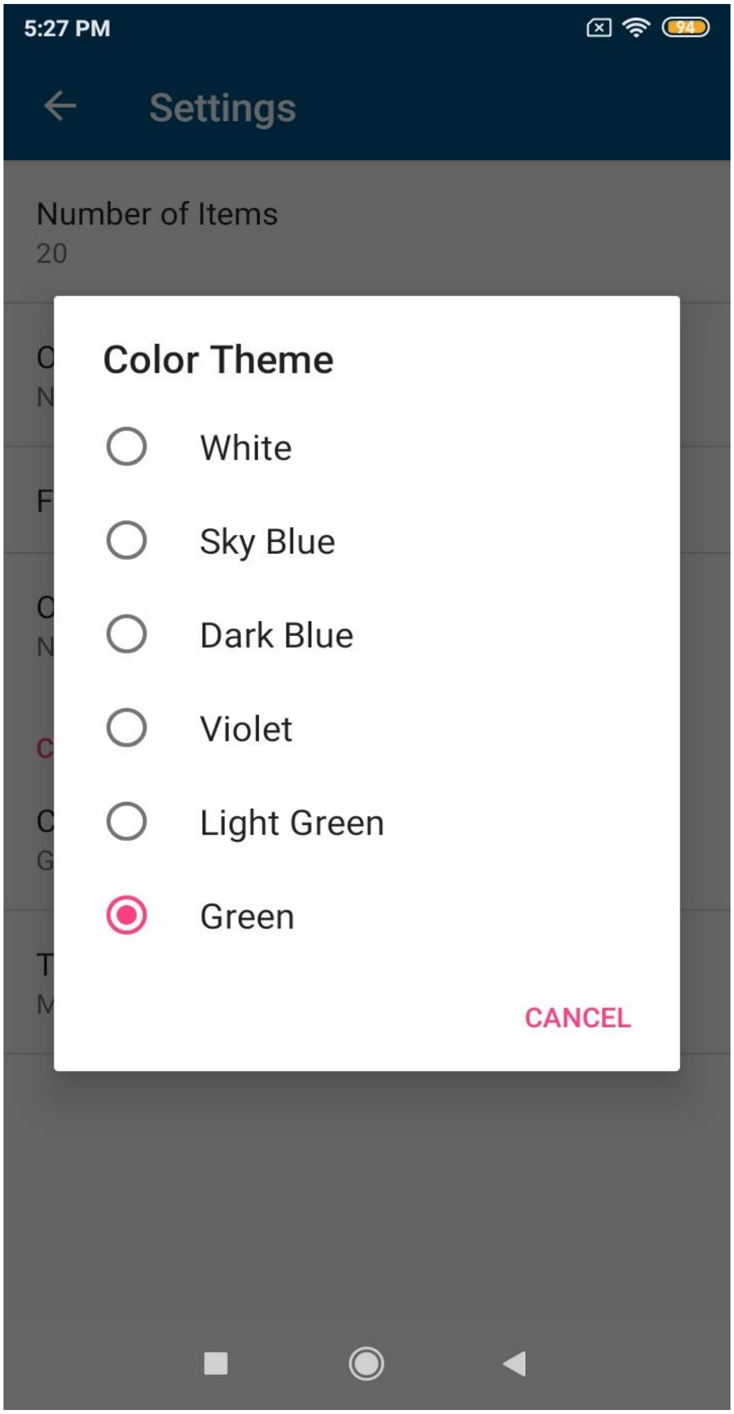
Between 1 and 50

CANCEL OK

Newspaper Edition

Customize Styles

Colour Theme



5:27 PM

← Settings

Number of Items
20

Color Theme

○ White

○ Sky Blue

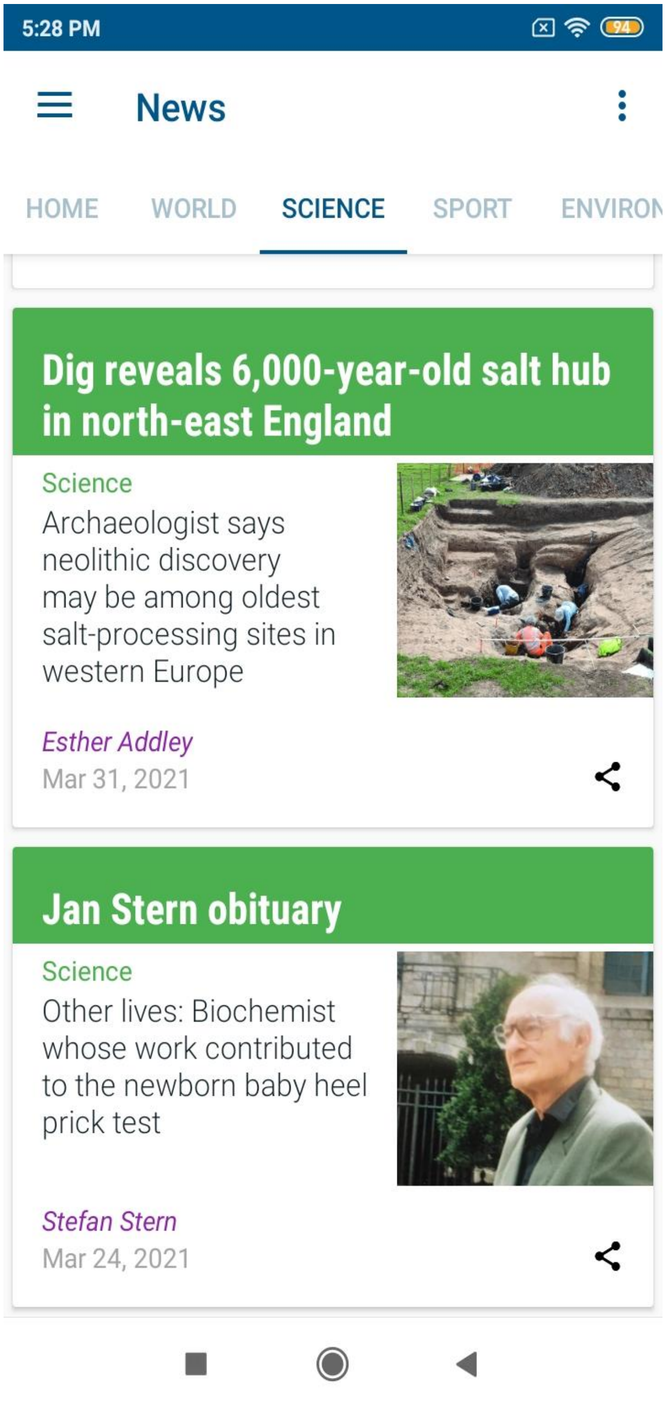
○ Dark Blue

○ Violet

○ Light Green

● Green

CANCEL



5:28 PM

News

HOME WORLD SCIENCE SPORT ENVIRON

Dig reveals 6,000-year-old salt hub in north-east England

Science

Archaeologist says neolithic discovery may be among oldest salt-processing sites in western Europe

Esther Addley

Mar 31, 2021

Jan Stern obituary

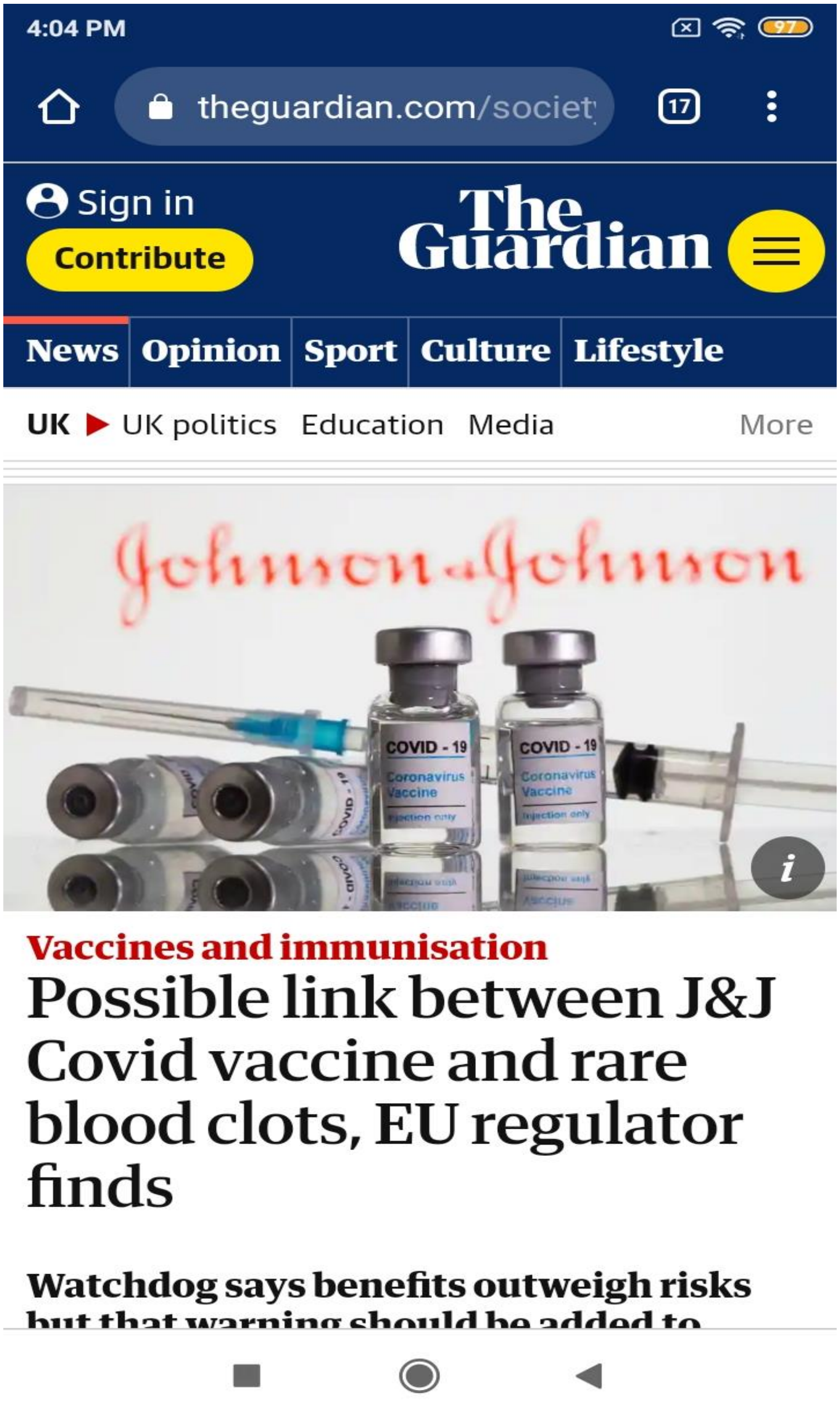
Science

Other lives: Biochemist whose work contributed to the newborn baby heel prick test

Stefan Stern

Mar 24, 2021

Browser



Vaccines and immunisation

Possible link between J&J Covid vaccine and rare blood clots, EU regulator finds

Watchdog says benefits outweigh risks but that warning should be added to

Future Enhancement

The News application is fully functional software developed in android but there are some better changes that can be done in future to the software to make it more functioning and useful to user and make software user friendly. The Enhancement which can be done are as follows :

- Supports video file format in the application to provide detailed information about news.
- Database can be connected to create users and track daily activity and suggest news.
- Better UI design to make software more user friendly.

References

- www.stackoverflow.com
- www.youtube.com
- www.newsapi.org
- www.androiddevelopers.com
- www.google.com