Name: Jehan Bhathena
Andrew ID: jbhathen

Project Overview

This project implements a distributed news application system consisting of three main components: an Android mobile application, a Java web service deployed to GitHub Codespaces, and a MongoDB Atlas database for logging and analytics. The application allows users to search for news articles by topic using the NewsAPI.org service. The web service acts as a middleware between the Android app and the 3rd party API, implementing business logic, data processing, and logging capabilities. The system includes a web-based dashboard that displays analytics about usage patterns and request logs.

Android Application:

Article.java

This model class represents news articles and matches the structure of the data returned by the web service:

```
//Name: Jehan Bhathena
//Andrew ID: jbhathen
package com.example.newsapp;

public class Article {
   private String title;
   private String description;
   private String url;
   private String urlToImage;
   private String publishedAt;
   private String source;

   // Constructor
   public Article(String title, String description, String url, String
urlToImage, String publishedAt, String source) {
       this.title = title;
       this.description = description;
       this.url = url;
       this.urlToImage = urlToImage;
       this.publishedAt = publishedAt;
       this.source = source;
   }

   // Getters
   public String getTitle() { return title; }
   public String getDescription() { return description; }
   public String getUrl() { return url; }
   public String getUrlToImage() { return urlToImage; }
   public String getPublishedAt() { return publishedAt; }
   public String getSource() { return source; }
}
```

FetchNews.java

This class handles asynchronous HTTP requests to the web service, implementing a background thread as required:

```java
//Name: Jehan Bhathena
//Andrew ID: jbhathen
package com.example.newsapp;

import android.os.AsyncTask;
import android.util.Log;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.lang.reflect.Type;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.List;

public class FetchNewsTask extends AsyncTask<String, Void, List<Article>> {

   private static final String TAG = "FetchNewsTask";
   private final NewsTaskCallback callback;
   private final String deviceModel;
   private Exception exception;

   // Update this URL with your actual GitHub Codespace URL
   private static final String BASE_URL =
"https://ideal-pancake-rw5x656xvpqfpjqp.github.dev/";

   public FetchNewsTask(NewsTaskCallback callback, String deviceModel) {
       this.callback = callback;
       this.deviceModel = deviceModel;
   }

   @Override
   protected void onPreExecute() {
       callback.onPreExecute();
   }

   @Override
   protected List<Article> doInBackground(String... params) {
       if (params.length == 0) {
           return null;
       }

       String topic = params[0];
```

```
        List<Article> articles = new ArrayList<>();

        try {
            String encodedTopic = URLEncoder.encode(topic, "UTF-8");
            String encodedDeviceModel = URLEncoder.encode(deviceModel,
"UTF-8");
            String urlStr = BASE_URL + "?topic=" + encodedTopic +
"&deviceModel=" + encodedDeviceModel;

            URL url = new URL(urlStr);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("GET");
            connection.setConnectTimeout(10000);
            connection.setReadTimeout(15000);

            int responseCode = connection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                BufferedReader in = new BufferedReader(
                        new InputStreamReader(connection.getInputStream()));
                StringBuilder response = new StringBuilder();
                String line;

                while ((line = in.readLine()) != null) {
                    response.append(line);
                }
                in.close();

                Gson gson = new Gson();
                Type articleListType = new
TypeToken<ArrayList<Article>>(){}.getType();
                articles = gson.fromJson(response.toString(), articleListType);
            } else {
                throw new Exception("Error response: " + responseCode);
            }
        } catch (Exception e) {
            Log.e(TAG, "Error fetching news", e);
            this.exception = e;
            return null;
        }

        return articles;
    }

    @Override
    protected void onPostExecute(List<Article> articles) {
        if (articles != null) {
            callback.onTaskComplete(articles);
        } else {
            callback.onError(exception != null ? exception.getMessage() :
"Unknown error occurred");
        }
    }

    public interface NewsTaskCallback {
        void onPreExecute();
        void onTaskComplete(List<Article> articles);
        void onError(String error);
    }
}
```

MainActivity.java

This is the primary activity that handles the user interface and coordinates the application's functionality.

```
//Name: Jehan Bhathena
//Andrew ID: jbhathen
package com.example.newsapp;

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

import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

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

public class MainActivity extends AppCompatActivity implements
FetchNewsTask.NewsTaskCallback {

    private EditText searchEditText;
    private Button searchButton;
    private RecyclerView newsRecyclerView;
    private ProgressBar progressBar;
    private TextView errorTextView;

    private NewsAdapter newsAdapter;
    private List<Article> articleList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```java
        setContentView(R.layout.activity_main);

        // Initialize views
        searchEditText = findViewById(R.id.searchEditText);
        searchButton = findViewById(R.id.searchButton);
        newsRecyclerView = findViewById(R.id.newsRecyclerView);
        progressBar = findViewById(R.id.progressBar);
        errorTextView = findViewById(R.id.errorTextView);

        // Set up RecyclerView
        articleList = new ArrayList<>();
        newsAdapter = new NewsAdapter(this, articleList);
        newsRecyclerView.setLayoutManager(new LinearLayoutManager(this));
        newsRecyclerView.setAdapter(newsAdapter);

        // Set up search button click listener
        searchButton.setOnClickListener(v -> {
            String topic = searchEditText.getText().toString().trim();
            if (topic.isEmpty()) {
                Toast.makeText(MainActivity.this, "Please enter a topic",
Toast.LENGTH_SHORT).show();
                return;
            }

            String deviceModel = Build.MODEL;
            new FetchNewsTask(this, deviceModel).execute(topic);
        });
    }

    @Override
    public void onPreExecute() {
        progressBar.setVisibility(View.VISIBLE);
        errorTextView.setVisibility(View.GONE);
        articleList.clear();
        newsAdapter.notifyDataSetChanged();
    }

    @Override
    public void onTaskComplete(List<Article> articles) {
        progressBar.setVisibility(View.GONE);

        if (articles.isEmpty()) {
            errorTextView.setText("No news found for this topic");
            errorTextView.setVisibility(View.VISIBLE);
        } else {
            articleList.clear();
            articleList.addAll(articles);
            newsAdapter.notifyDataSetChanged();
        }
    }

    @Override
    public void onError(String error) {
        progressBar.setVisibility(View.GONE);
        errorTextView.setText("Error: " + error);
        errorTextView.setVisibility(View.VISIBLE);
    }
}
```

NewsAdapter.java

This adapter populates the RecyclerView with news articles:

```
//Name: Jehan Bhathena
//Andrew ID: jbhathen
package com.example.newsapp;

import android.content.Context;
import android.content.Intent;
import android.net.Uri;
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.bumptech.glide.Glide;

import java.util.List;

public class NewsAdapter extends
RecyclerView.Adapter<NewsAdapter.NewsViewHolder> {

    private List<Article> articles;
    private Context context;

    public NewsAdapter(Context context, List<Article> articles) {
        this.context = context;
        this.articles = articles;
    }

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

    @Override
    public void onBindViewHolder(@NonNull NewsViewHolder holder, int position)
{
        Article article = articles.get(position);

        holder.titleTextView.setText(article.getTitle());
        holder.sourceTextView.setText(article.getSource());

        if (article.getDescription() != null &&
!article.getDescription().isEmpty()) {
            holder.descriptionTextView.setText(article.getDescription());
            holder.descriptionTextView.setVisibility(View.VISIBLE);
        } else {
            holder.descriptionTextView.setVisibility(View.GONE);
        }

        if (article.getUrlToImage() != null &&
!article.getUrlToImage().isEmpty()) {
            holder.imageView.setVisibility(View.VISIBLE);
            Glide.with(context)
                    .load(article.getUrlToImage())
                    .centerCrop()
                    .placeholder(R.drawable.placeholder_image)
                    .error(R.drawable.error_image)
                    .into(holder.imageView);
        } else {
            holder.imageView.setVisibility(View.GONE);
        }

        holder.readMoreButton.setOnClickListener(v -> {
            Intent browserIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse(article.getUrl()));
            context.startActivity(browserIntent);
        });
    }

    @Override
    public int getItemCount() {
        return articles != null ? articles.size() : 0;
    }
```

```
   public void updateArticles(List<Article> newArticles) {
       this.articles = newArticles;
       notifyDataSetChanged();
   }

   static class NewsViewHolder extends RecyclerView.ViewHolder {
       TextView titleTextView;
       TextView sourceTextView;
       TextView descriptionTextView;
       ImageView imageView;
       Button readMoreButton;

       public NewsViewHolder(@NonNull View itemView) {
           super(itemView);
           titleTextView = itemView.findViewById(R.id.newsTitleTextView);
           sourceTextView = itemView.findViewById(R.id.newsSourceTextView);
           descriptionTextView =
itemView.findViewById(R.id.newsDescriptionTextView);
           imageView = itemView.findViewById(R.id.newsImageView);
           readMoreButton = itemView.findViewById(R.id.readMoreButton);
       }
   }
}
```
Layout Resources

1. activity_main.xml

The main layout implements multiple View types including EditText, Button, RecyclerView, ProgressBar, and TextView:

```
<?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">

    <TextView
        android:id="@+id/titleTextView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:text="News Explorer"
        android:textAlignment="center"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/searchEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:ems="10"
        android:hint="Enter news topic"
        android:inputType="text"
        android:minHeight="48dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/titleTextView" />

    <Button
        android:id="@+id/searchButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="16dp"
        android:text="Search News"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/searchEditText" />

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.recyclerview.widget.RecyclerView
```

```
        android:id="@+id/newsRecyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/searchButton" />

    <TextView
        android:id="@+id/errorTextView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:textAlignment="center"
        android:textColor="@android:color/holo_red_dark"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

2. item_news.xml

This layout defines how each news article is displayed in the RecyclerView:

```
<?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:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    app:cardCornerRadius="8dp"
    app:cardElevation="4dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">
        <TextView
            android:id="@+id/newsTitleTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/newsSourceTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
            android:textColor="@android:color/darker_gray"
            android:textSize="14sp"
            android:textStyle="italic" />

        <ImageView
            android:id="@+id/newsImageView"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:layout_marginTop="8dp"
            android:scaleType="centerCrop"
            android:visibility="gone" />

        <TextView
            android:id="@+id/newsDescriptionTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:textSize="14sp" />

        <Button
            android:id="@+id/readMoreButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="Read More" />

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

Web Service

The web service implements a RESTful API with a single endpoint /news that retrieves news articles based on a topic parameter. The NewsServlet handles GET requests from the Android application, processes them, and returns JSON-formatted article data.

```
@WebServlet(name = "NewsServlet", urlPatterns = {"/news"})
public class NewsServlet extends HttpServlet {
    // ...

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Set response type to JSON
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");

        // Get parameters
        String topic = request.getParameter("topic");
        String deviceModel = request.getParameter("deviceModel");

        // Log request information
        Map<String, Object> logData = new HashMap<>();
        // ... logging implementation ...

        try {
            // Business logic: fetch news from 3rd party API
            List<Article> articles = newsService.getNewsByTopic(topic);

            // Return JSON response to Android client
            out.print(gson.toJson(articles));

        } catch (Exception e) {
            // Error handling
            // ...
        }
    }
}
```

Logging of Information

The web service logs the following information for each request:

Request Time: Timestamp when the request was received (used for request timing analysis)
Topic: The news topic searched for (helps identify popular search terms)
Device Model: Information about the client device (useful for platform analytics)
Client IP: IP address of the requesting client (helps with geographic analysis)
API Latency: Time taken for the 3rd party API request (monitors external API performance)
Article Count: Number of articles returned (measures result quality)

Response Time: Timestamp when processing completed (used with request time for latency calculation)
Response Latency: Total time to process the request (measures service performance)
Status: Success or error status of the request (monitors service reliability)

This information is logged via the LoggingUtil class:

```
public void logRequest(Map<String, Object> logData) {
    dbManager.insertLog(logData);
}
```

MongoDB Storage
The service connects to MongoDB Atlas cloud database, using the MongoDB Java driver. The connection is established in the MongoDBManager class:

```
public MongoDBManager() {
    mongoClient = MongoClients.create(CONNECTION_STRING);
    database = mongoClient.getDatabase(DATABASE_NAME);
    logCollection = database.getCollection(LOG_COLLECTION);
}
```

Log data is stored as documents in a collection, and various aggregation queries are implemented to generate analytics:

```
public List<Document> getMostSearchedTopics(int limit) {
    List<Document> pipeline = Arrays.asList(
            new Document("$group",
                    new Document("_id", "$topic")
                            .append("count", new Document("$sum", 1))),
            new Document("$sort", new Document("count", -1)),
            new Document("$limit", limit)
    );

    AggregateIterable<Document> results = logCollection.aggregate(pipeline);
    List<Document> topTopics = new ArrayList<>();
    results.forEach(topTopics::add);

    return topTopics;
}
```

Dashboard
The web application includes a dashboard that displays three key analytics:

Top 5 Searched Topics: Shows the most frequently searched news topics
API Performance: Displays average API latency and total request count
Top 5 Device Models: Shows the most common device models making requests

The dashboard is implemented using JSP with clean HTML/CSS for formatting:

```
<div class="dashboard">
```

```
    <%
        MongoDBManager dbManager = new MongoDBManager();

        // Most searched topics
        List<Document> topTopics = dbManager.getMostSearchedTopics(5);
    %>

    <div class="card">
        <h2>Top 5 Searched Topics</h2>
        <table>
            <!-- Table implementation -->
        </table>
    </div>

    <!-- Other analytics cards -->
</div>
```

A separate logs.jsp page displays detailed request logs in a formatted table, providing a comprehensive view of all system activity.

Challenges and Solutions

Several technical challenges were encountered during development:

MongoDB Data Types: Numeric values stored in MongoDB needed to be explicitly typed as Double instead of Integer or Long to ensure proper aggregation in analytics queries.
Java Version Compatibility: The initial project used Java 21, but GitHub Codespaces environment supports an older version. The code needed to be adjusted for compatibility with Java 11.
NewsAPI Limitations: The NewsAPI free tier has restrictions on deployment environments, which requires careful configuration and testing.

These challenges were addressed through code refactoring, dependency management, and environment configuration adjustments

Github Codespaces Deployment
https://github.com/CMU-Heinz-95702/distributed-systems-project-04-Jehan-7
https://ideal-pancake-rw5x656xvpqfpjqp.github.dev/

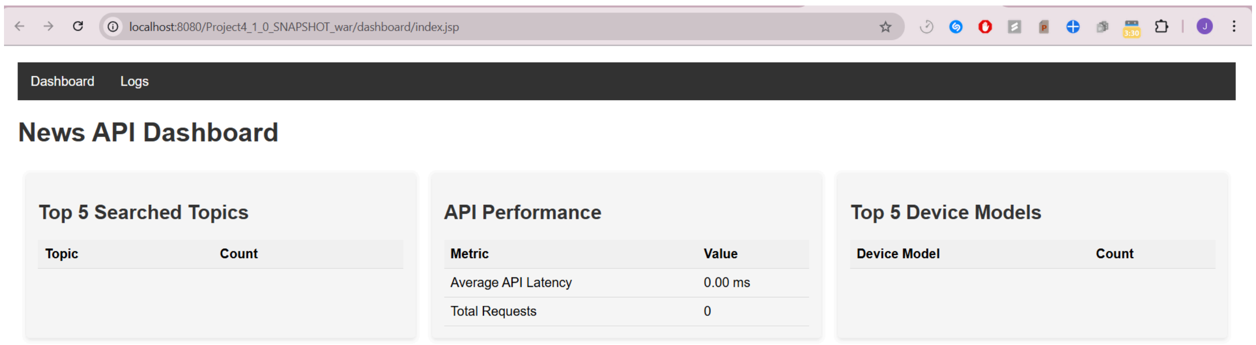
The deployment of the web service to GitHub Codespaces was a critical component of this project, allowing the application to be accessible from the Android client application regardless of the user's location. This section details the process and configuration used to successfully deploy the application.
Creating the ROOT.war File
The first step in deployment was generating the proper WAR file:

Maven Build: Using the command mvn clean install generated a WAR file in the target directory of the project.
Verifying File Structure: The ROOT.war file contained all necessary components:

- Servlet classes (NewsServlet)
- Model classes (Article)
- Service classes (NewsService)
- Database utilities (MongoDBManager)
- Logging utilities (LoggingUtil)
- JSP files for the dashboard
- Web configuration files (web.xml)
- All required dependencies

WAR File Naming: The file was named "ROOT.war" to ensure it would be deployed as the root application in the Tomcat server within GitHub Codespaces.

Screenshots:


localhost:8080/Project4_1_0_SNAPSHOT_war/dashboard/index.jsp
Dashboard
Logs

News API Dashboard

Top 5 Searched Topics

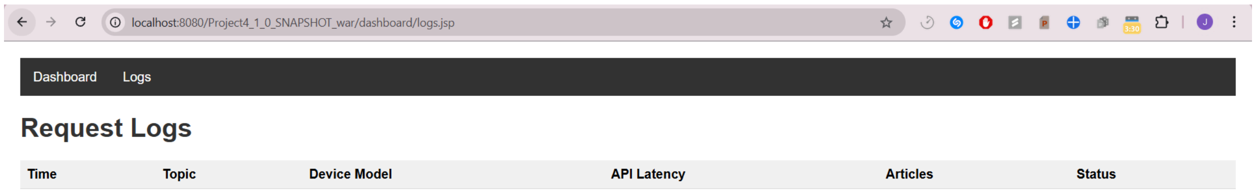
Topic	Count

API Performance

Metric	Value
Average API Latency	0.00 ms
Total Requests	0

Top 5 Device Models

Device Model	Count



localhost:8080/Project4_1_0_SNAPSHOT_war/dashboard/logs.jsp

Request Logs

Time	Topic	Device Model	API Latency	Articles	Status