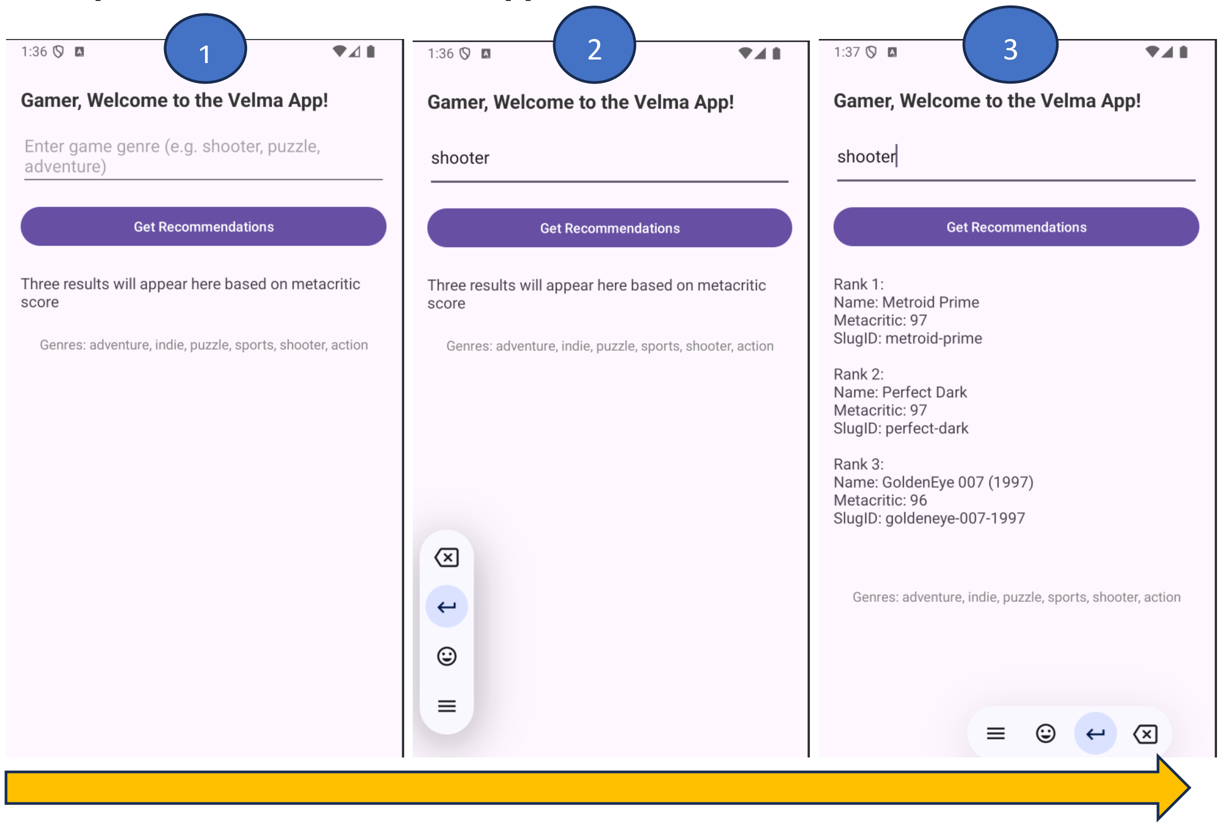
Vimbai Muyengwa
AndrewID: vmuyengw

1. Implement a native Android application



Gamer, Welcome to the Velma App!

Enter game genre (e.g. shooter, puzzle, adventure)

Get Recommendations

Three results will appear here based on metacritic score

Genres: adventure, indie, puzzle, sports, shooter, action

Gamer, Welcome to the Velma App!

shooter

Get Recommendations

Three results will appear here based on metacritic score

Genres: adventure, indie, puzzle, sports, shooter, action

Gamer, Welcome to the Velma App!

shooter

Get Recommendations

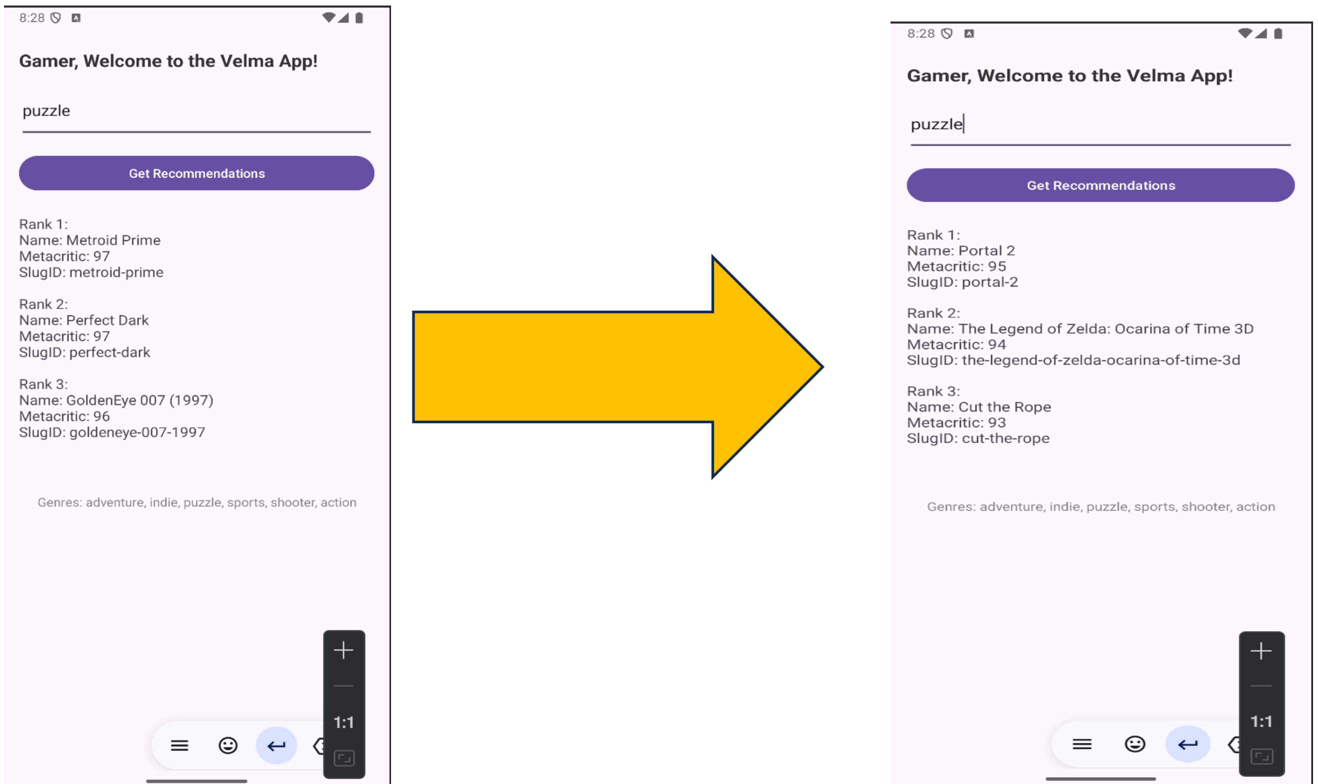
Rank 1:
Name: Metroid Prime
Metacritic: 97
SlugID: metroid-prime

Rank 2:
Name: Perfect Dark
Metacritic: 97
SlugID: perfect-dark

Rank 3:
Name: GoldenEye 007 (1997)
Metacritic: 96
SlugID: goldeneye-007-1997

Genres: adventure, indie, puzzle, sports, shooter, action

The name of my android app is the Velma Recommender App. It is saved as project4Task2. A gamer is welcomed to the Velma app and prompted to enter a genre(e.g. shooter).
After entry and clicking Get Recommendations, they get the top 3 games ranked by Metacritic score for that specific genre. They can also type in a different genre and get different Recommendations (e.g puzzle)



Gamer, Welcome to the Velma App!

puzzle

Get Recommendations

Rank 1:
Name: Metroid Prime
Metacritic: 97
SlugID: metroid-prime

Rank 2:
Name: Perfect Dark
Metacritic: 97
SlugID: perfect-dark

Rank 3:
Name: GoldenEye 007 (1997)
Metacritic: 96
SlugID: goldeneye-007-1997

Genres: adventure, indie, puzzle, sports, shooter, action

Gamer, Welcome to the Velma App!

puzzle

Get Recommendations

Rank 1:
Name: Portal 2
Metacritic: 95
SlugID: portal-2

Rank 2:
Name: The Legend of Zelda: Ocarina of Time 3D
Metacritic: 94
SlugID: the-legend-of-zelda-ocarina-of-time-3d

Rank 3:
Name: Cut the Rope
Metacritic: 93
SlugID: cut-the-rope

Genres: adventure, indie, puzzle, sports, shooter, action

Vimbai Muyengwa
AndrewID: vmuyengw

b. The app receives input from user, makes an HTTP request to my webservice via my codespaces URL **https://supreme-rotary-phone-5gj6qqvjp79c9p6-8080.app.github.dev/getGames?genre=**

The project directory name is GameRecommenderServlet

c. Makes an HTTP request (using an appropriate HTTP method) to your web service. Note that this request ***must*** be done using a background thread (see Lab 8's use of BackgroundTask). Do in background shown below:

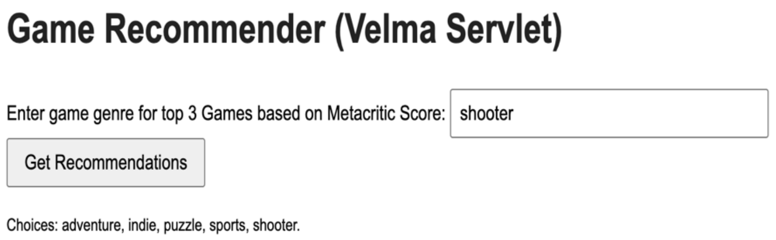
```
/*
 * Author: Vimbai Muyengwa
 * andrewID: vmuyengw
 * Performs the core functionality of the Velma app in a background thread.
*
* This method:
* - Builds a URL with the user's selected genre
* - (c) Sends an HTTP GET request to a web service running in GitHub
Codespaces
* - Reads and collects the JSON response from the server
* - (d) Parses the JSON to extract details for the top 3 games (name,
metacritic score, slug)
* - (e) Updates the UI to display new game recommendations to the user
* - (f) Supports repeatable use: the method is called every time the user
enters a new genre
*
* This background execution ensures that the app remains responsive and
complies
*
* References: Lab 8 Code: https://github.com/CMU-Heinz-95702/lab8-
AndroidInterestingPicture?tab=readme-ov-file
*  https://developer.android.com/reference/java/net/HttpURLConnection
* GenAI used for troubleshoot
*/
private void doInBackground() {
    try {
        // Question 1, C: Makes an HTTP request (using an appropriate HTTP
method) to your web service
        String servletCodeSpaceUrl = "https://supreme-rotary-phone-
5gj6qqvjp79c9p6-8080.app.github.dev/getGames?genre=" + genre;
        Log.d("VELMA_FETCH", "Retrieve from URL: " + servletCodeSpaceUrl);
//inserted after trouble shoot

        URL url = new URL(servletCodeSpaceUrl); //Set up URL connection and
define request method
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET"); //HTTP Get method

        //troubleshoot used GENAI
        conn.setRequestProperty("X-Device-Model", "Android " +
android.os.Build.MODEL);

        // Open a stream to read the incoming response line by line
        BufferedReader reader = new BufferedReader(new
```

Vimbai Muyengwa
AndrewID: vmuyengw

```
InputStreamReader(conn.getInputStream()));

        // Use StringBuilder to accumulate the entire JSON response as one
string
        StringBuilder responseBuilder = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            responseBuilder.append(line);
        }
        reader.close(); // Close the stream after reading is complete

        // Question 1 (d): Parse the JSON formatted reply from the web
service
        JSONObject jsonResponse = new JSONObject(responseBuilder.toString());
// Convert string to JSON object
        JSONArray games = jsonResponse.getJSONArray("games"); // Extract the
games array from the JSON

        if (games.length() == 0) { // Handle case where no games were
returned
            topGamesText.append("Zero games in this genre.");
            return;
        }

        //Limits to top 3 games
        int count = Math.min(3, games.length());
        for (int i = 0; i < count; i++) {
            JSONObject game = games.getJSONObject(i); // Get each game object

            // Safe extract game fields with fallback/default values
            String name = game.optString("name", "Unknown");
            String metacritic = game.has("metacritic") ?
String.valueOf(game.getInt("metacritic")) : "N/A";
            String slug = game.optString("slugID", "N/A");

            // Question 1, e: Display each game’s rank and detail
            // Display parsed game information to the user in the UI
            topGamesText.append("Rank ").append(i + 1).append(":\n")
                    .append("Name: ").append(name).append("\n")
                    .append("Metacritic: ").append(metacritic).append("\n")
                    .append("SlugID: ").append(slug).append("\n\n");

        }
        //try, catch exception
    } catch (Exception e) {
        Log.e("VELMA_FETCH", "Exception during fetch", e); //inserted after
troubleshoot issues
        topGamesText.append("Data fetch failure.");
    }
}
```

Vimbai Muyengwa
AndrewID: vmuyengw

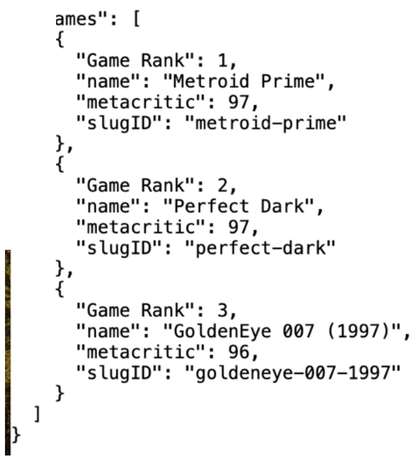
d. It takes and parses a JSON formatted reply from my web service seen. below: Web Service Page and response, and displays new information to the gamer. User can continue entering different genres for different results (Includes d-f)


Game Recommender (Velma Servlet)

Enter game genre for top 3 Games based on Metacritic Score: shooter

Get Recommendations

Choices: adventure, indie, puzzle, sports, shooter.


```
ames": [
  {
    "Game Rank": 1,
    "name": "Metroid Prime",
    "metacritic": 97,
    "slugID": "metroid-prime"
  },
  {
    "Game Rank": 2,
    "name": "Perfect Dark",
    "metacritic": 97,
    "slugID": "perfect-dark"
  },
  {
    "Game Rank": 3,
    "name": "GoldenEye 007 (1997)",
    "metacritic": 96,
    "slugID": "goldeneye-007-1997"
  }
 ]
}
```

e. Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, or anything that extends android.view.View). **In order to figure out if something is a View, find its API. If it extends android.view.View then it is a View.**

Featured in content_main .xml. My app contains, TextView, EditText, and the button

```
<!--
    Velma App Main Layout
    This layout defines the UI for the main activity of the Velma game
recommendation Android app.
    stacks components for user interaction, results display, and metadata.

    3 Different views textView, Button, and EditText

    Reference Android Lab 8
    https://github.com/CMU-Heinz-95702/lab8-
AndroidInterestingPicture?tab=readme-ov-file

    GenAI (ChatGPT) was used to assist with troubleshooting layout issues and
code generation,.
-->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/content_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <!-- Welcome Message -->
    <TextView
```

Vimbai Muyengwa
AndrewID: vmuyengw

```xml
        android:layout_marginTop="32dp"
        android:id="@+id/welcomeText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gamer, Welcome to the Velma App!"
        android:textSize="20sp"
        android:textStyle="bold"
        android:paddingBottom="12dp"
        android:textColor="#333"/>

    <!-- user to input a game genre -->
    <EditText
        android:id="@+id/genreInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter game genre (e.g. shooter, puzzle, adventure)"
        android:minHeight="48dp" />

    <!-- Button to trigger game recommendation based on user input -->
    <Button
        android:id="@+id/getGamesButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Get Recommendations"
        android:minHeight="48dp"
        android:layout_marginTop="16dp" />

    <!-- TextView shows top 3 recommended games will be shown -->
    <TextView
        android:id="@+id/resultsText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Three results will appear here based on metacritic
score"
        android:paddingTop="24dp"
        android:textSize="16sp" />


    <!-- Footer to inform the user about valid genre options -->
    <TextView
        android:id="@+id/footerGenres"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Genres: adventure, indie, puzzle, sports, shooter,
action"
        android:textSize="14sp"
        android:textColor="#888"
        android:layout_marginTop="24dp"
        android:gravity="center_horizontal" />

</LinearLayout>
```

Vimbai Muyengwa
AndrewID: vmuyengw

2. Implement a web service

a. Implement a simple (can be a single path) API. – see code below

- Model: The logic that interacts with the third-party API is built into the GameRecommendationServlet class. Receive android input, build RAWG games API URL, send HTTP Get request, read and parson Json, extract essential game data,replies with clean JSON
- View: Android app UI displays the results using VelmaMainActivity. Additionally, an analytics dashboard is served via JSP (e.g., dashboard.jsp) in theGameRecommenderServlet project
- Controller: The controller is implemented in GameRecommendationServlet.java, mapped to a single path: @WebServlet("/getGames")

b. Receives an HTTP request from the native Android application

GameRecommendationServlet.java receives a GET request from the Android app (RecommendationsGetter.java) containing a genre parameter from the user input. see doGet in code below

c. Executes business logic appropriate to your application. This includes fetching XML or JSON information from some 3rd party API and processing the response.

GameRecommendationServlet sends an HTTP GET request to the **RAWG Video Game Database API**: String rawGUrl = "https://api.rawg.io/api/games?key=" + API_KEY + "&genres=" + genre + "&page_size=3&ordering=-metacritic";

It then parses the JSON response using Gson, Extracts the top 3 games by metacritic score, spits out key game data (name, metacritic, slugID) into JSON response

```java
package project4tasks.ds.gamerecommenderservlet;


import java.io.*;
import java.net.*;
import java.time.Instant;

import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
```

Vimbai Muyengwa
AndrewID: vmuyengw

```
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;

/*
Student: Vimbai Muyengwa
andrewID: vmuyengw

 * GameRecommendationServlet
 *
 * This servlet implements a simple web service that receives HTTP GET
requests from a native Android application,
 * fetches top 3 game recommendations from the RAWG third-party API (based on
a provided genre), processes the JSON
 * response to extract only the essential fields (name, metacritic score,
slug), and returns a trimmed-down JSON
 * response to the client. It also logs detailed request/response information
to MongoDB for dashboard analytics.
 *
 * This servlet satisfies the following:
 * a. Implements a single-path API: /getGames
 * b. Receives requests from a mobile app (Android)
 * c. Executes business logic by calling a 3rd-party API (RAWG) and processes
JSON
 * d. Replies with a clean JSON response (with only needed data)
 * e. Logs 6+ analytics fields to MongoDB for dashboard display

 References:
    RAWG. (n.d.). RAWG API documentation. https://rawg.io/apidocs
    https://github.com/CMU-Heinz-95702/Project4/blob/master/README.md
        Mongo Bison information et al.

    Previous lab2 for servlet functionality Review
 GenAI used for troubleshoot and code generation
*/


// Implement a simple can be a single path API
@WebServlet("/getGames")
public class GameRecommendationServlet extends HttpServlet {

    // API key for authenticating requests to the RAWG API
    private final String API_KEY = "19456881592b452f998b624c2fbd70ca";


    @Override // Handles HTTP GET requests sent to /getGames
    protected void doGet(HttpServletRequest servletRequest,
HttpServletResponse servletResponse) throws IOException {

        servletResponse.setContentType("application/json"); //Set the
response content type to application/json

        // Get and read genre from request parameter
        // This is input from the user (Android app)
        String genre = servletRequest.getParameter("genre");


        //Used GenAI to for troubleshoot help with date format
```

Vimbai Muyengwa
AndrewID: vmuyengw

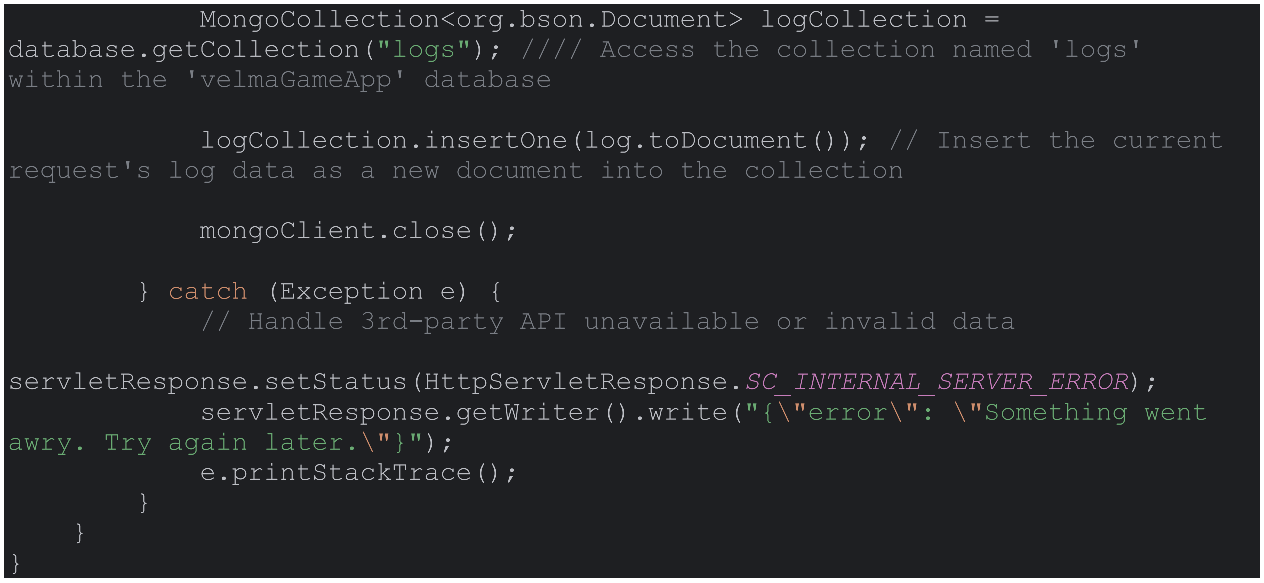
```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMM dd,
yyyy HH:mm:ss")
                .withZone(ZoneId.of("America/New_York"));

        //track time
        String requestReceivedTimestamp = formatter.format(Instant.now());

        long startTime = System.currentTimeMillis();

        //// Handle missing or invalid input from the client
        //Invalid mobile app input
        if (genre == null || genre.trim().isEmpty()) {
            servletResponse.setStatus(HttpServletResponse.SC_BAD_REQUEST);
            servletResponse.getWriter().write("{\"error\": \"Genre
missing\"}");
            return;
        }

        try { //try-catch block if 3rd party API is unavailable
            /* Fetch top games from RAWG API: Executes business logic
appropriate to my application
            // Fetching JSON information from a 3rd party API
            Retrive top 3 games
             */
            String rawGUrl = "https://api.rawg.io/api/games?key=" + API_KEY +
"&genres=" + genre + "&page_size=3&ordering=-metacritic";

            String thirdPartyApiRequestTimestamp =
formatter.format(Instant.now()); //// Timestamp for when 3rd-party API
request is made

            URL url = new URL(rawGUrl);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("GET");

            // Read the response from the RAWG API into a StringBuilder
            BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            StringBuilder responseBuilder = new StringBuilder();
            String line;

            while ((line = reader.readLine()) != null) {
                responseBuilder.append(line);
            }
            reader.close();

            //troubleshoot code
           // System.out.println("RAWG RESPONSE: " +
responseBuilder.toString());

            // Parse the JSON response into a JsonObject
            JsonObject jsonResponse =
JsonParser.parseString(responseBuilder.toString()).getAsJsonObject();
            JsonArray gameResults = jsonResponse.getAsJsonArray("results");
// // Extract the array of game results
```

Vimbai Muyengwa
AndrewID: vmuyengw

```
            // Handle case where RAWG API returns no results (invalid or rare
genre)
            if (gameResults == null || gameResults.size() == 0) {
                servletResponse.setStatus(HttpServletResponse.SC_NOT_FOUND);
                servletResponse.getWriter().write("{\"error\": \"No games
found for the genre '" + genre + "'. Try a different genre.\"}");
                return;
            }

            // Construct a simplified JSON response. Replies to the Android
application with a JSON formatted response
            JsonArray essentialGameData = new JsonArray();

            // Loop through each game in the results (top 3)
            for (int i = 0; i < gameResults.size(); i++) {
                JsonObject gameDetails =
gameResults.get(i).getAsJsonObject();
                JsonObject essentialData = new JsonObject();

                // Add cleaned-up game data to the new JSON object
                //https://api.rawg.io/docs/#operation/games_suggested_read
                //pulls essential data for Game
                essentialData.addProperty("Game Rank", i + 1); // 1 based
ranking
                essentialData.addProperty("name",
gameDetails.get("name").getAsString());

                // Check if metacritic rating exists before adding
                // checks if Third-Party API Returns Invalid Data
                if (gameDetails.has("metacritic") &&
!gameDetails.get("metacritic").isJsonNull()) {
                    essentialData.addProperty("metacritic",
gameDetails.get("metacritic").getAsDouble());
                } else {
                    essentialData.addProperty("metacritic", -1); // default
if unavailable
                }
                essentialData.addProperty("slugID",
gameDetails.get("slug").getAsString()); //An ID or a slug identifying this
Game per API rawg.io/docs
                essentialGameData.add(essentialData);
            }

        // Wrap the array in a parent JSON object
            JsonObject finalResponse = new JsonObject(); // Create JSON
object to return to Android client
            finalResponse.add("games", essentialGameData);

            // Send the successful 200 OK response to the client with the
JSON data
            servletResponse.setStatus(HttpServletResponse.SC_OK);
            servletResponse.getWriter().write(finalResponse.toString());

            /*
            * Dashboard Logging
            *The purpose of logging data to the database is to be able to
create an operations dashboard for your web service.
```

```
            * This dashboard should be web page interface for use from a
desktop or laptop browser (not a mobile device).
            * The dashboard should display two types of data: 3.1. Operations
analytics - display at least 3 interesting operations analytics from your web
service.
            * You should choose analytics that are relevant to your specific
web service. Examples for InterestingPicture might be top 10 picture search
terms, average Flickr search latency, or the top 5 Android phone models
making requests.
            * 3.2. Logs - display the data logs being stored for each mobile
phone user interaction with your web service
            *
             */
            //Logging to MongoDB
            long endTime = System.currentTimeMillis();
            long latency = endTime - startTime;

            //  // Get metadata from request (device model)
            String phoneModel = servletRequest.getHeader("Device Model");
            if (phoneModel == null || phoneModel.trim().isEmpty()) { // error
response if phone model is null
                phoneModel = "Unknown Device";
            }
            String appParams = "genre=" + genre;

            //construct log fields
            StringBuilder replyData = new StringBuilder(); //stringbuilder
for replyData
            for (int i = 0; i < essentialGameData.size(); i++) {
                JsonObject game = essentialGameData.get(i).getAsJsonObject();
                replyData.append(game.get("name").getAsString());
                if (i < essentialGameData.size() - 1) replyData.append(", ");
            }

            String replyInfo = "Status: 200 OK | Games Returned: " +
essentialGameData.size();

            // Build LoggingData object for Mongo DB. Persists log to mongo
db
            // track overall performance with latency measures for the
requests for specific genres
            LoggingData log = new LoggingData(
                    phoneModel,
                    appParams, // the appParams are the genres
                    requestReceivedTimestamp,
                    thirdPartyApiRequestTimestamp,
                    replyData.toString(),
                    replyInfo,
                    latency
            );

            // Insert into MongoDB using login credentials
            MongoClient mongoClient =
MongoClients.create("mongodb+srv://vmuyengw:2Dzidzayi$$@cluster0.jppey.mongod
b.net/?retryWrites=true&w=majority&appName=Cluster0");
           MongoDatabase database = mongoClient.getDatabase("velmaGameApp");
// Access the specific MongoDB database named 'velmaGameApp'
```

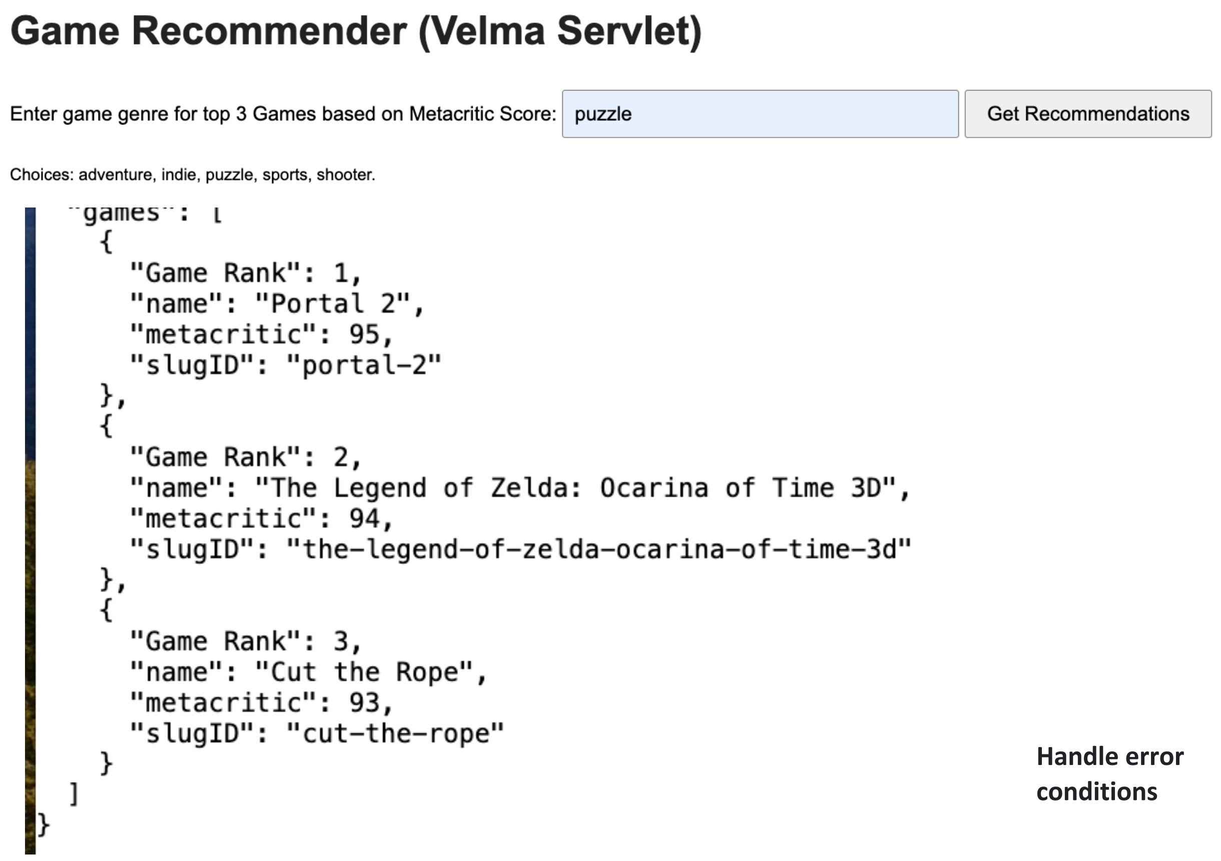
Vimbai Muyengwa
AndrewID: vmuyengw



```
            MongoCollection<org.bson.Document> logCollection =
database.getCollection("logs"); //// Access the collection named 'logs'
within the 'velmaGameApp' database

            logCollection.insertOne(log.toDocument()); // Insert the current
request's log data as a new document into the collection

            mongoClient.close();

        } catch (Exception e) {
            // Handle 3rd-party API unavailable or invalid data

servletResponse.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
            servletResponse.getWriter().write("{\"error\": \"Something went
awry. Try again later.\"}");
            e.printStackTrace();
        }
    }
}
```

d. Replies to the Android application with an XML or JSON formatted response. The schema of the response can be of your own design.

After running web servlet, the below UI pops up. User enters genre and it generates the top 3 ranked games based on the Metacritic score in JSON format



Game Recommender (Velma Servlet)

Enter game genre for top 3 Games based on Metacritic Score: puzzle Get Recommendations

Choices: adventure, indie, puzzle, sports, shooter.

```
  "games": [
    {
      "Game Rank": 1,
      "name": "Portal 2",
      "metacritic": 95,
      "slugID": "portal-2"
    },
    {
      "Game Rank": 2,
      "name": "The Legend of Zelda: Ocarina of Time 3D",
      "metacritic": 94,
      "slugID": "the-legend-of-zelda-ocarina-of-time-3d"
    },
    {
      "Game Rank": 3,
      "name": "Cut the Rope",
      "metacritic": 93,
      "slugID": "cut-the-rope"
    }
  ]
}
```

Handle error conditions

Vimbai Muyengwa
AndrewID: vmuyengw

Your application should test for and handle gracefully: Please see excerpts included in code below

- Invalid mobile app input

```
// Handle missing or invalid input from the client
//Invalid mobile app input
if (genre == null || genre.trim().isEmpty()) {
    servletResponse.setStatus(HttpServletResponse.SC_BAD_REQUEST);
    servletResponse.getWriter().write("{\"error\": \"Genre
missing\"}");
    return;
}
```

- Invalid server-side input (regardless of mobile app input validation)

```
// Handle case where RAWG API returns no results (invalid or rare
genre)
if (gameResults == null || gameResults.size() == 0) {
    servletResponse.setStatus(HttpServletResponse.SC_NOT_FOUND);
    servletResponse.getWriter().write("{\"error\": \"No games found for
the genre '" + genre + "'. Try a different genre.\"}");
    return;
}
```

- Third-party API unavailable

```
catch (Exception e) {
            // Handle 3rd-party API unavailable or invalid data

servletResponse.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR)
;
            servletResponse.getWriter().write("{\"error\": \"Something
went awry. Try again later.\"}");
            e.printStackTrace();
        }
```

- Third-party API invalid data

```
// checks if Third-Party API Returns Invalid Data
                if (gameDetails.has("metacritic") &&
!gameDetails.get("metacritic").isJsonNull()) {
                    essentialData.addProperty("metacritic",
gameDetails.get("metacritic").getAsDouble());
                } else {
                    essentialData.addProperty("metacritic", -1);
```

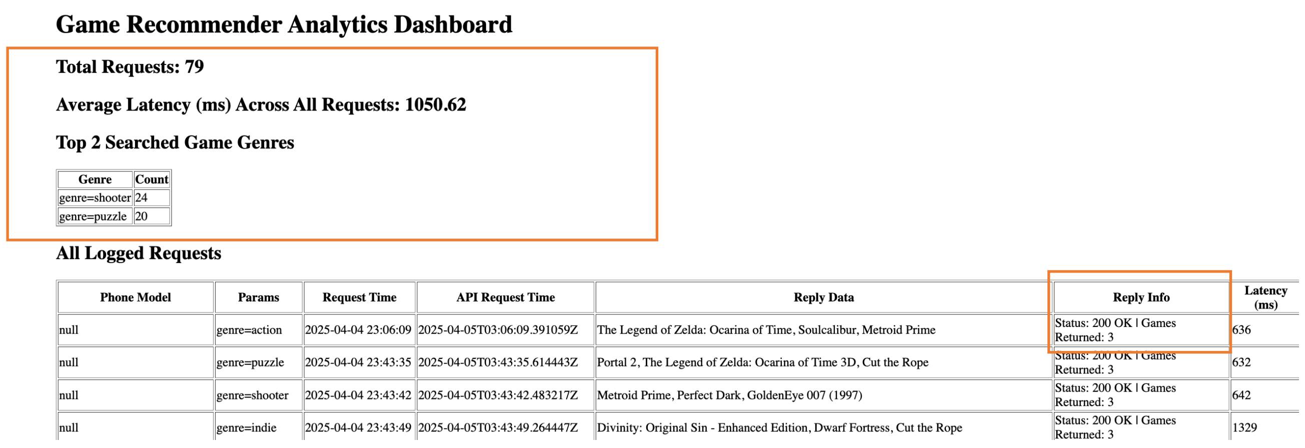
3. Dashboard

My implementation logs all key request-response data from the Android app (and optionally the web browser) to MongoDB. These logs are displayed in a structured JSP-based web interface accessible from a desktop browser. Each log shows 6 pieces of Info. I used Use <20 lines of JSP, don't display logs as JSON/XML, and also utilized tables

Vimbai Muyengwa
AndrewID: vmuyengw

Three or more interesting Operations analytics shown

1. Total Requests – count of requests
2. Average Latency – if I were designing this app to go live, this would be top of mind for my users/gamers. No one wants to wait to play
3. The Top 2 searched game genres – this give me insights on what we should urge our partners to create and or advise to users based on user interest/behavior
4. Success/Failure shown in reply info



Game Recommender Analytics Dashboard

Total Requests: 79

Average Latency (ms) Across All Requests: 1050.62

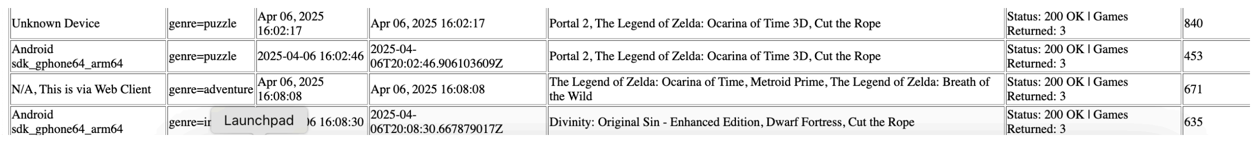
Top 2 Searched Game Genres

Genre	Count
genre=shooter	24
genre=puzzle	20

All Logged Requests

Phone Model	Params	Request Time	API Request Time	Reply Data	Reply Info	Latency (ms)
null	genre=action	2025-04-04 23:06:09	2025-04-05T03:06:09.391059Z	The Legend of Zelda: Ocarina of Time, Soulcalibur, Metroid Prime	Status: 200 OK | Games Returned: 3	636
null	genre=puzzle	2025-04-04 23:43:35	2025-04-05T03:43:35.614443Z	Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the Rope	Status: 200 OK | Games Returned: 3	632
null	genre=shooter	2025-04-04 23:43:42	2025-04-05T03:43:42.483217Z	Metroid Prime, Perfect Dark, GoldenEye 007 (1997)	Status: 200 OK | Games Returned: 3	642
null	genre=indie	2025-04-04 23:43:49	2025-04-05T03:43:49.264447Z	Divinity: Original Sin - Enhanced Edition, Dwarf Fortress, Cut the Rope	Status: 200 OK | Games Returned: 3	1329

…



Unknown Device	genre=puzzle	Apr 06, 2025 16:02:17	Apr 06, 2025 16:02:17	Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the Rope	Status: 200 OK | Games Returned: 3	840
Android sdk_gphone64_arm64	genre=puzzle	2025-04-06 16:02:46	2025-04-06T20:02:46.906103609Z	Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the Rope	Status: 200 OK | Games Returned: 3	453
N/A, This is via Web Client	genre=adventure	Apr 06, 2025 16:08:08	Apr 06, 2025 16:08:08	The Legend of Zelda: Ocarina of Time, Metroid Prime, The Legend of Zelda: Breath of the Wild	Status: 200 OK | Games Returned: 3	671
Android sdk_gphone64_arm64	genre=ir Launchpad	)6 16:08:30	2025-04-06T20:08:30.667879017Z	Divinity: Original Sin - Enhanced Edition, Dwarf Fortress, Cut the Rope	Status: 200 OK | Games Returned: 3	635

4. Log useful information: At least 6 pieces of information is logged for each request/reply with the mobile phone. It should include information about the request from the mobile phone, information about the request and reply to the 3rd party API, and information about the reply to the mobile phone. (You should NOT log data from interactions from the operations dashboard.)

My app contains 6 pieces of information phoneModel, appRequestParameters (gener), requestReceivedTimestamp, thirdPartyApiRequestTimestamp, and thirdPartyAPIReplyData, replytoMobileInfo about status plus the numbers of Games Returned, and finally the latency. Below shows logging data instantiation and declaration. .JSP file renamed for dashboard clarity:

phoneModel – Logs device model using the X-Device-Model header from the Android app "Android sdk_gphone64_arm64"

Vimbai Muyengwa
AndrewID: vmuyengw

specificAppRequestParameters– Logs the genre parameter sent by the Android app ("genre=shooter") for user interest tracking

timestampForRequestReceipt– Records when server first receives request from the app. Tracks total handling time to identify peak usage times

thirdPartyApiRequestTimestamp – Timestamp of when the RAWG API call is made.
Part of latency analysis and API delays

dataReplyMobile – Logs a summary of the game names returned from the RAWG API displayed to user

mobilePhoneInfoReply – A message showing response status "Status: 200 OK | Games Returned: 3". Confirms success/failure and count of games

latency – Calculates time from request to API response for performance monitoring.

Logging

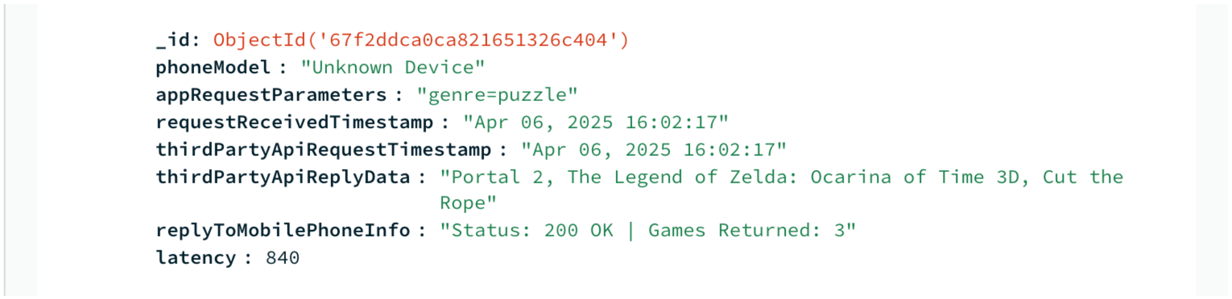
5. Store the log information in a database

I store all log data in **MongoDB Atlas**, using the following connection:

mongodb+srv://vmuyengw:2Dzidzayi$$@cluster0.jppey.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

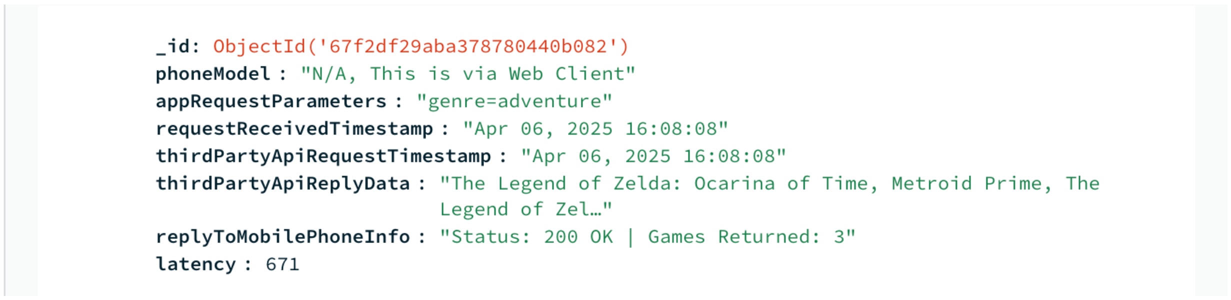
- **Cluster Name:** Cluster0
- **Shards:** This Atlas cluster is a free-tier cluster
- **Database Name:** velmaGameApp
- **Collection:** logs

The web service can connect, store, and retrieve information from a MongoDB database in the cloud. Data is stored persistently.



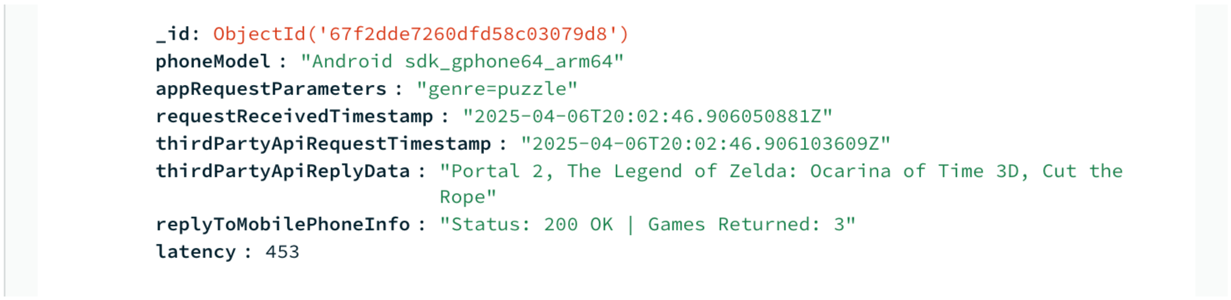
```
_id: ObjectId('67f2ddca0ca821651326c404')
phoneModel : "Unknown Device"
appRequestParameters : "genre=puzzle"
requestReceivedTimestamp : "Apr 06, 2025 16:02:17"
thirdPartyApiRequestTimestamp : "Apr 06, 2025 16:02:17"
thirdPartyApiReplyData : "Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the
                         Rope"
replyToMobilePhoneInfo : "Status: 200 OK | Games Returned: 3"
latency : 840
```

Vimbai Muyengwa
AndrewID: vmuyengw



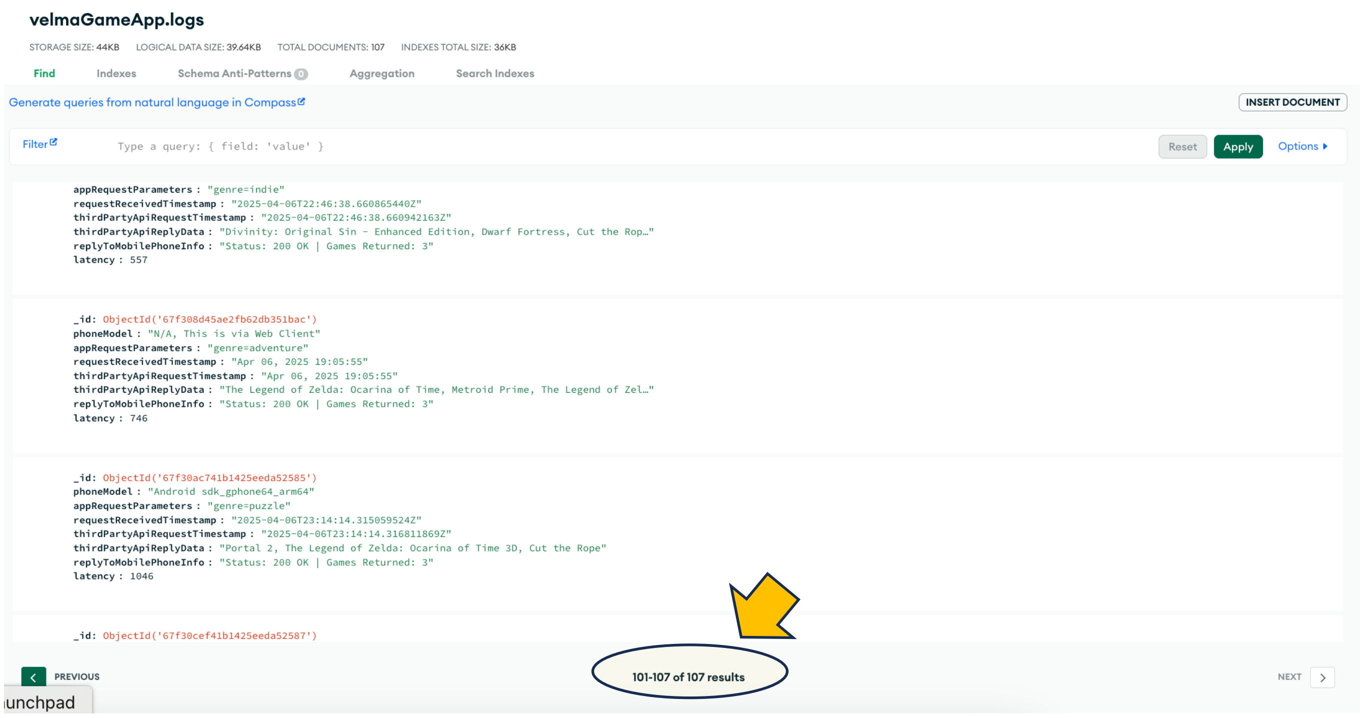
```
_id: ObjectId('67f2df29aba378780440b082')
phoneModel : "N/A, This is via Web Client"
appRequestParameters : "genre=adventure"
requestReceivedTimestamp : "Apr 06, 2025 16:08:08"
thirdPartyApiRequestTimestamp : "Apr 06, 2025 16:08:08"
thirdPartyApiReplyData : "The Legend of Zelda: Ocarina of Time, Metroid Prime, The
                         Legend of Zel…"
replyToMobilePhoneInfo : "Status: 200 OK | Games Returned: 3"
latency : 671
```

Above is the Mongo Entry from running Web Service locally. Below shows android data



```
_id: ObjectId('67f2dde7260dfd58c03079d8')
phoneModel : "Android sdk_gphone64_arm64"
appRequestParameters : "genre=puzzle"
requestReceivedTimestamp : "2025-04-06T20:02:46.906050881Z"
thirdPartyApiRequestTimestamp : "2025-04-06T20:02:46.906103609Z"
thirdPartyApiReplyData : "Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the
                         Rope"
replyToMobilePhoneInfo : "Status: 200 OK | Games Returned: 3"
latency : 453
```

Data persistency shown below: 107 results from multiple troubleshoot sessions until finalization



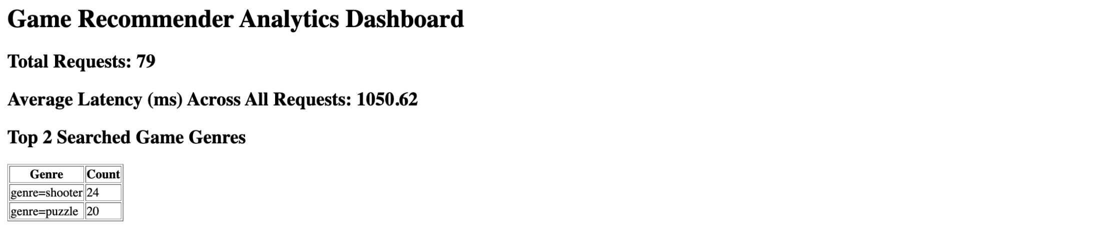
velmaGameApp.logs

STORAGE SIZE: 44KB   LOGICAL DATA SIZE: 39.64KB   TOTAL DOCUMENTS: 107   INDEXES TOTAL SIZE: 36KB

```
appRequestParameters : "genre=indie"
requestReceivedTimestamp : "2025-04-06T22:46:38.660865440Z"
thirdPartyApiRequestTimestamp : "2025-04-06T22:46:38.660942163Z"
thirdPartyApiReplyData : "Divinity: Original Sin - Enhanced Edition, Dwarf Fortress, Cut the Rop…"
replyToMobilePhoneInfo : "Status: 200 OK | Games Returned: 3"
latency : 557

_id: ObjectId('67f308d45ae2fb62db351bac')
phoneModel : "N/A, This is via Web Client"
appRequestParameters : "genre=adventure"
requestReceivedTimestamp : "Apr 06, 2025 19:05:55"
thirdPartyApiRequestTimestamp : "Apr 06, 2025 19:05:55"
thirdPartyApiReplyData : "The Legend of Zelda: Ocarina of Time, Metroid Prime, The Legend of Zel…"
replyToMobilePhoneInfo : "Status: 200 OK | Games Returned: 3"
latency : 746

_id: ObjectId('67f30ac741b1425eeda52585')
phoneModel : "Android sdk_gphone64_arm64"
appRequestParameters : "genre=puzzle"
requestReceivedTimestamp : "2025-04-06T23:14:14.315059524Z"
thirdPartyApiRequestTimestamp : "2025-04-06T23:14:14.316811869Z"
thirdPartyApiReplyData : "Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the Rope"
replyToMobilePhoneInfo : "Status: 200 OK | Games Returned: 3"
latency : 1046

_id: ObjectId('67f30cef41b1425eeda52587')
```

Vimbai Muyengwa
AndrewID: vmuyengw

6. Display operations analytics and formatted full logs on a web-based dashboard



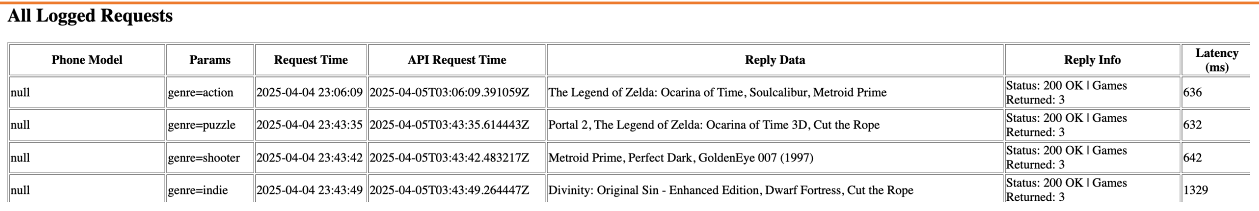
Game Recommender Analytics Dashboard

Total Requests: 79

Average Latency (ms) Across All Requests: 1050.62

Top 2 Searched Game Genres

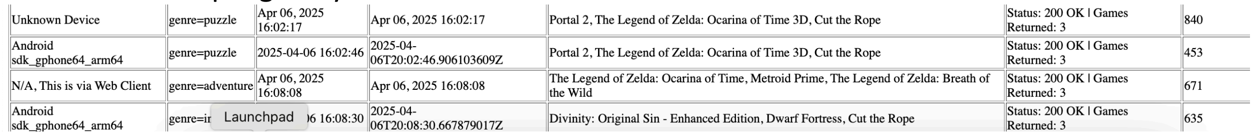
Genre	Count
genre=shooter	24
genre=puzzle	20



All Logged Requests

Phone Model	Params	Request Time	API Request Time	Reply Data	Reply Info	Latency (ms)
null	genre=action	2025-04-04 23:06:09	2025-04-05T03:06:09.391059Z	The Legend of Zelda: Ocarina of Time, Soulcalibur, Metroid Prime	Status: 200 OK | Games Returned: 3	636
null	genre=puzzle	2025-04-04 23:43:35	2025-04-05T03:43:35.614443Z	Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the Rope	Status: 200 OK | Games Returned: 3	632
null	genre=shooter	2025-04-04 23:43:42	2025-04-05T03:43:42.483217Z	Metroid Prime, Perfect Dark, GoldenEye 007 (1997)	Status: 200 OK | Games Returned: 3	642
null	genre=indie	2025-04-04 23:43:49	2025-04-05T03:43:49.264447Z	Divinity: Original Sin - Enhanced Edition, Dwarf Fortress, Cut the Rope	Status: 200 OK | Games Returned: 3	1329

The above was before I corrected Phone model response. See below for phone model and better timestamp legibility.

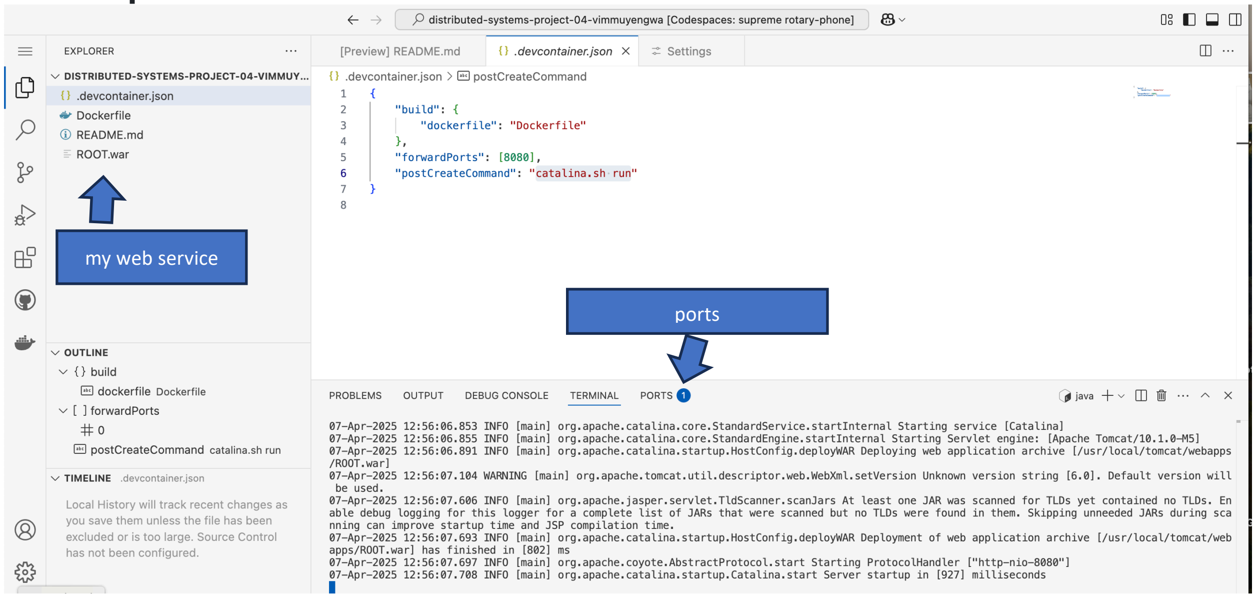


Unknown Device	genre=puzzle	Apr 06, 2025 16:02:17	Apr 06, 2025 16:02:17	Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the Rope	Status: 200 OK | Games Returned: 3	840
Android sdk_gphone64_arm64	genre=puzzle	2025-04-06 16:02:46	2025-04-06T20:02:46.906103609Z	Portal 2, The Legend of Zelda: Ocarina of Time 3D, Cut the Rope	Status: 200 OK | Games Returned: 3	453
N/A, This is via Web Client	genre=adventure	Apr 06, 2025 16:08:08	Apr 06, 2025 16:08:08	The Legend of Zelda: Ocarina of Time, Metroid Prime, The Legend of Zelda: Breath of the Wild	Status: 200 OK | Games Returned: 3	671
Android sdk_gphone64_arm64	genre=ir Launchpad	6 16:08:30	2025-04-06T20:08:30.667879017Z	Divinity: Original Sin - Enhanced Edition, Dwarf Fortress, Cut the Rope	Status: 200 OK | Games Returned: 3	635

a. A unique URL addresses a web interface dashboard for the web service.

http://localhost:8080/GameRecommenderServlet/dashboard (run the GameServletRecommender first)

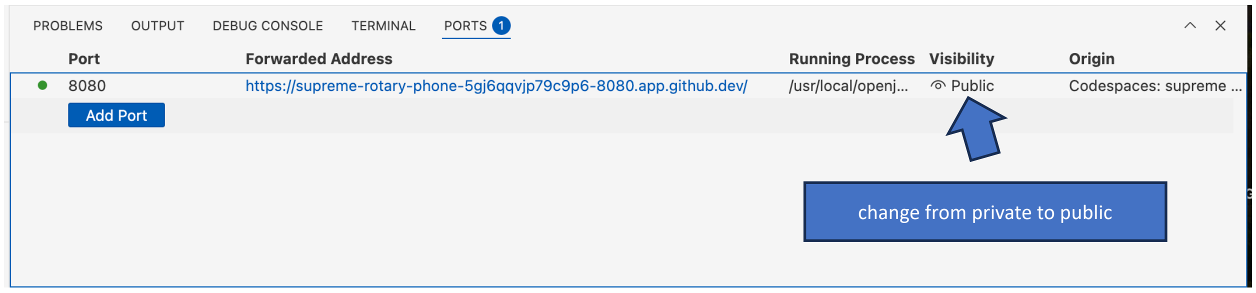
7. **Codespaces**


distributed-systems-project-04-vimmuyengwa [Codespaces: supreme rotary-phone]
EXPLORER
DISTRIBUTED-SYSTEMS-PROJECT-04-VIMMUY...
.devcontainer.json
Dockerfile
README.md
ROOT.war
my web service
OUTLINE
build
dockerfile Dockerfile
forwardPorts
0
postCreateCommand catalina.sh run
TIMELINE .devcontainer.json
Local History will track recent changes as you save them unless the file has been excluded or is too large. Source Control has not been configured.
[Preview] README.md
.devcontainer.json
Settings
.devcontainer.json > postCreateCommand
ports
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS 1
java

```json
{
    "build": {
        "dockerfile": "Dockerfile"
    },
    "forwardPorts": [8080],
    "postCreateCommand": "catalina.sh run"
}
```

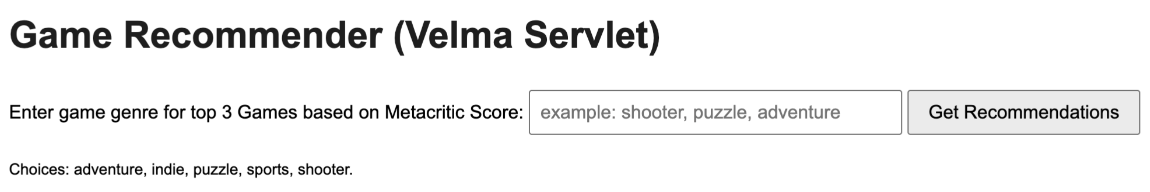
```
07-Apr-2025 12:56:06.853 INFO [main] org.apache.catalina.core.StandardService.startInternal Starting service [Catalina]
07-Apr-2025 12:56:06.855 INFO [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet engine: [Apache Tomcat/10.1.0-M5]
07-Apr-2025 12:56:06.891 INFO [main] org.apache.catalina.startup.HostConfig.deployWAR Deploying web application archive [/usr/local/tomcat/webapps
/ROOT.war]
07-Apr-2025 12:56:07.104 WARNING [main] org.apache.tomcat.util.descriptor.web.WebXml.setVersion Unknown version string [6.0]. Default version will
 be used.
07-Apr-2025 12:56:07.606 INFO [main] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet contained no TLDs. En
able debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during sca
nning can improve startup time and JSP compilation time.
07-Apr-2025 12:56:07.693 INFO [main] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/usr/local/tomcat/web
apps/ROOT.war] has finished in [802] ms
07-Apr-2025 12:56:07.697 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
07-Apr-2025 12:56:07.708 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [927] milliseconds
```

Vimbai Muyengwa
AndrewID: vmuyengw



Port	Forwarded Address	Running Process	Visibility	Origin
8080	https://supreme-rotary-phone-5gj6qqvjp79c9p6-8080.app.github.dev/	/usr/local/openj...	Public	Codespaces: supreme ...

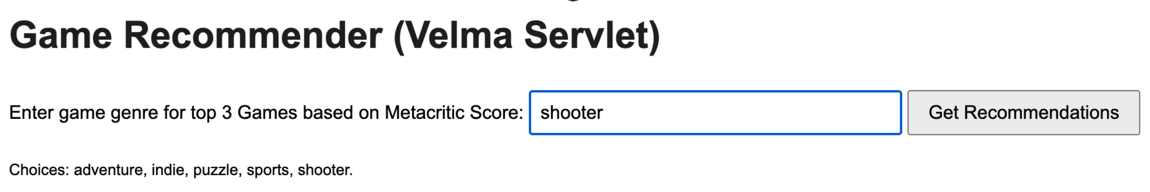
visit forwarded address with global icon



Game Recommender (Velma Servlet)

Enter game genre for top 3 Games based on Metacritic Score: example: shooter, puzzle, adventure

Choices: adventure, indie, puzzle, sports, shooter.

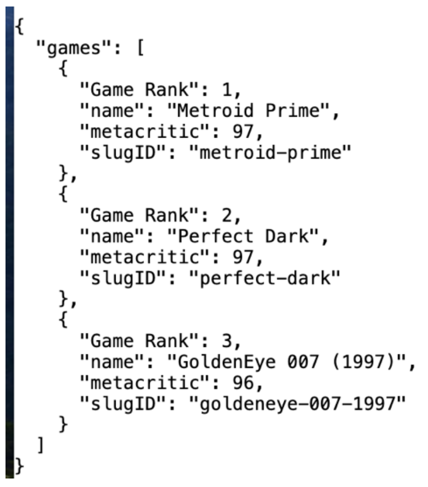


Game Recommender (Velma Servlet)

Enter game genre for top 3 Games based on Metacritic Score: shooter

Choices: adventure, indie, puzzle, sports, shooter.

Results



```
{
  "games": [
    {
      "Game Rank": 1,
      "name": "Metroid Prime",
      "metacritic": 97,
      "slugID": "metroid-prime"
    },
    {
      "Game Rank": 2,
      "name": "Perfect Dark",
      "metacritic": 97,
      "slugID": "perfect-dark"
    },
    {
      "Game Rank": 3,
      "name": "GoldenEye 007 (1997)",
      "metacritic": 96,
      "slugID": "goldeneye-007-1997"
    }
  ]
}
```

Vimbai Muyengwa
AndrewID: vmuyengw

Works Cited:

Public APIs. (n.d.). *Art & design*. GitHub. https://github.com/public-apis/public-apis?tab=readme-ov-file#art--design

RAWG. (n.d.). *RAWG API documentation*. https://rawg.io/apidocs