Project 4 - Mobile to Cloud Application

Ahmad Furqan (AndrewID: afurqan)

Project	**:**	**[Readaholic] Book Finder App Utilizing Google Book API**
Description	**:**	*This project is a component of the distributed systems course assignments aimed at evaluating students' ability to develop a simple yet practical distributed system application that mimics real-world situations. The project entails the creation of a Book Finder application, which assists users in exploring thousands of books available on Google Book servers. Specifically, it includes the following functionalities:* • *Real-time searching for book information on the Google Book server.* • *Comparing books with similar keywords in search results.* • *Sharing information about interesting books directly from the Android application to other applications.* • *Saving favorite books in a bookshelf for future reading plans.*
Development	**:**	*Further development could enhance user experience by creating a book-based social media platform where users can interact, share, and discuss popular and interesting books they have discovered.*

Further description will elaborate more about how this project try to meet all the task requirements

A. General Checklist

1. It can be simple, but should fetch information from a 3rd party source and do something of at least marginal value.
   Yes, The Book Finder application, though straightforward, serves as a valuable tool for book enthusiasts seeking specific titles for personal reasons. By linking with the Google Book server, it facilitates the search and retrieval of books tailored to individual needs and preferences.

2. Your web service should be deployed to the cloud and provide a simple RESTful API similar to those you have developed in prior projects. You do NOT have to implement all HTTP methods, only those that make sense for your application.
   Indeed, the web service deployed to Codespace delivers book data through the GET method and permits users to save their preferred books using the POST method.

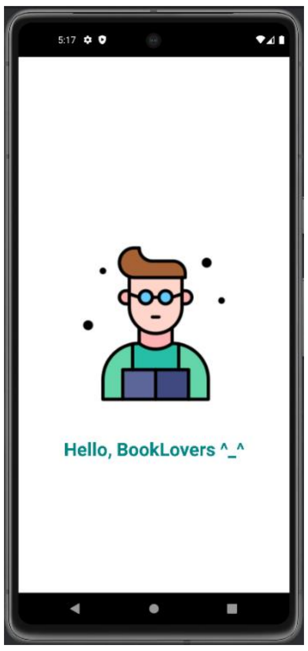
3. See the Banned APIs section. Do not use any of the banned API's.
   Yes, Google Book API has not been considered as a banned API.

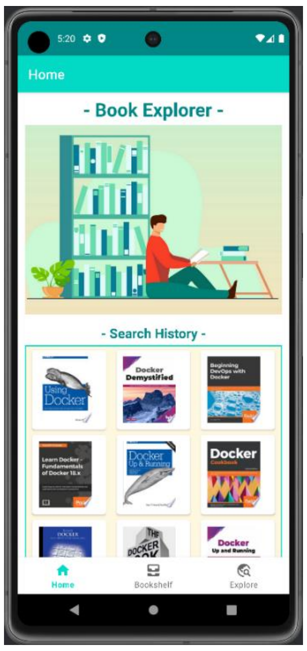
4. Use APIs that require authentication with caution. Do not use APIs that require a fee. Be sure your API is from a reputable source. Your API still needs to be available when the TAs go to grade your project.
   Certainly, the Google Book API is provided free of charge and maintained by a well-established tech giant. In our project, we utilize the GET method, which only requires an optional API key, as long as we are registered within the Google API framework.

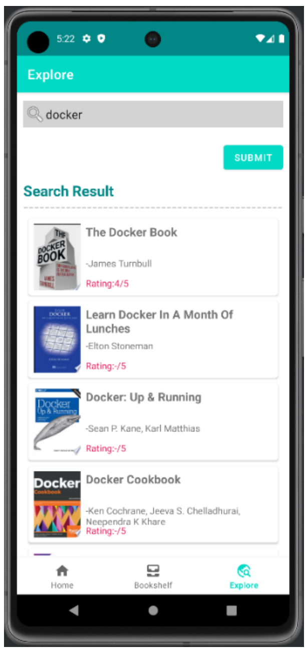
5. Users will access your application via a native Android application. You do not need to have a browser-based interface for your application (only for the Dashboard). The Android application should communicate with your web service deployed to the cloud. Your web service is where the business logic for your application should be implemented (including fetching information from the 3rd party API).
   Further explanation for this general checklist will be revealed in the subsequent points and attached screenshoots.

B. Distributed Application Requirements

1. Implement a native Android application





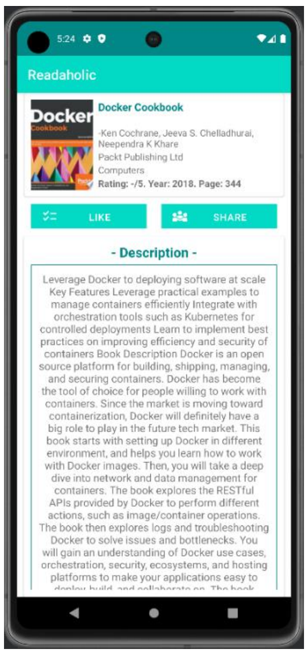


The Docker Book
-James Turnbull
Rating:4/5

Learn Docker In A Month Of Lunches
-Elton Stoneman
Rating:-/5

Docker: Up & Running
-Sean P. Kane, Karl Matthias
Rating:-/5

Docker Cookbook
-Ken Cochrane, Jeeva S. Chelladhurai, Neependra K Khare
Rating:-/5

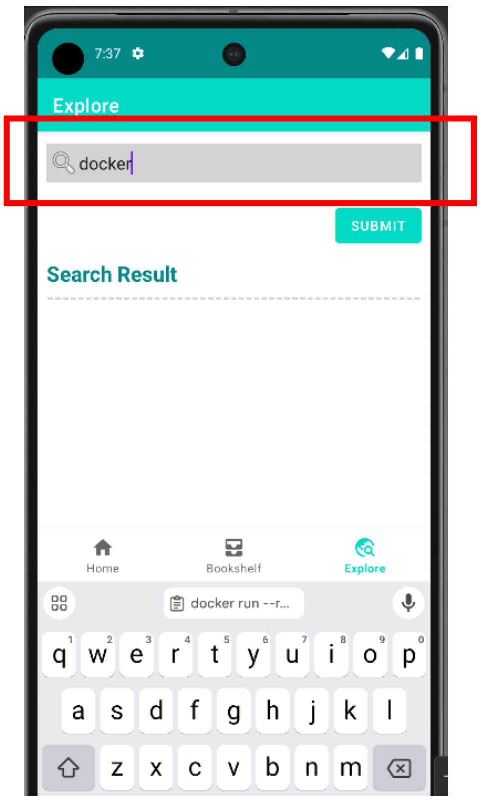


Docker Cookbook
-Ken Cochrane, Jeeva S. Chelladhurai, Neependra K Khare
Packt Publishing Ltd
Computers
Rating: -/5. Year: 2018. Page: 344

Leverage Docker to deploying software at scale Key Features Leverage practical examples to manage containers efficiently Integrate with orchestration tools such as Kubernetes for controlled deployments Learn to implement best practices on improving efficiency and security of containers Book Description Docker is an open source platform for building, shipping, managing, and securing containers. Docker has become the tool of choice for people willing to work with containers. Since the market is moving toward containerization, Docker will definitely have a big role to play in the future tech market. This book starts with setting up Docker in different environment, and helps you learn how to work with Docker images. Then, you will take a deep dive into network and data management for containers. The book explores the RESTful APIs provided by Docker to perform different actions, such as image/container operations. The book then explores logs and troubleshooting Docker to solve issues and bottlenecks. You will gain an understanding of Docker use cases, orchestration, security, ecosystems, and hosting platforms to make your applications easy to

a. Has at least three different kinds of Views in your Layout
I used TextView, EditText, ImageView. To construct list of books, I also used Recyclerview as a part of group views.

b. Requires input from the user
Here condition when user searchs book related to docker technology



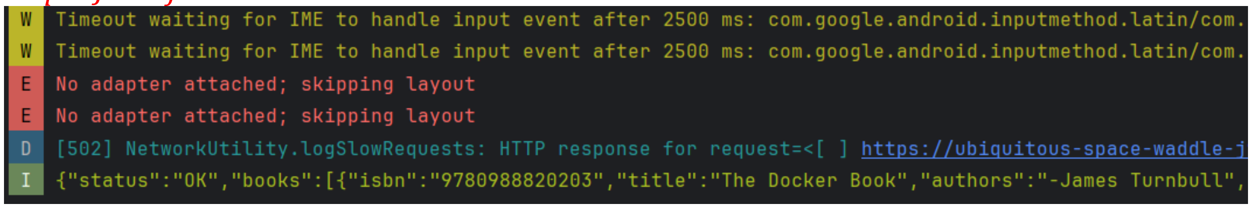
c. Makes an HTTP request (using an appropriate HTTP method) to your web service
Android does several HTTP requests to web service. It utilizes android volley library to tackle Asynctask deprecated issue. Response will be parsed using Gson library.

https://ubiquitous-space-waddle-j9jqvqxxgwxhpg77-8080.app.github.dev

Endpoint	Method	File in Android App	Description
/books?keyword=	*GET*	*ExploreFragment.java*	*User input to get book data*
/favbook	*POST*	*BookDetail.java*	*Update favourite book status*
/bookshelf	*GET*	*BookShelfFragment.java*	*Retrieve favourite book data*
/history	*GET*	*HomeFragment.java*	*Retrieve top 9 of search history*

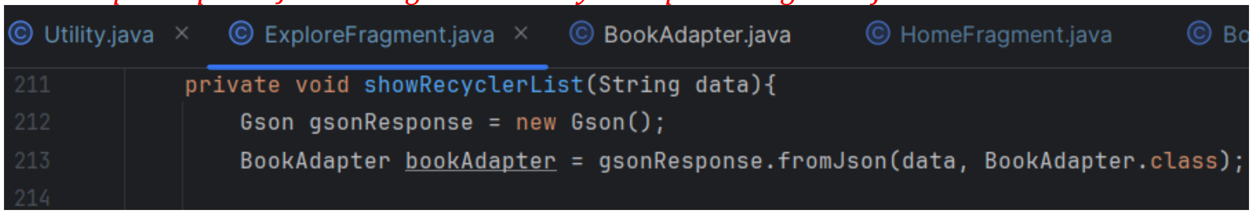
d. Receives and parses an XML or JSON formatted reply from your web service
In general, android will treat respon JSON from webservice using Java Bean and Class Adapter to make it easy to display in Android pages.

Snipet JSON from webservice



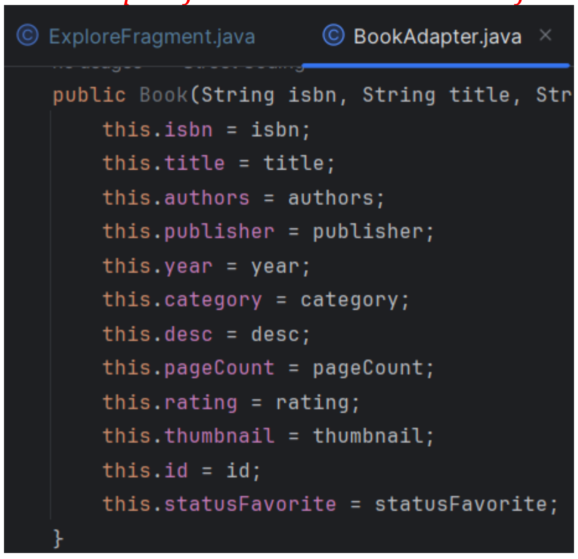
```
W  Timeout waiting for IME to handle input event after 2500 ms: com.google.android.inputmethod.latin/com.
W  Timeout waiting for IME to handle input event after 2500 ms: com.google.android.inputmethod.latin/com.
E  No adapter attached; skipping layout
E  No adapter attached; skipping layout
D  [502] NetworkUtility.logSlowRequests: HTTP response for request=<[ ] https://ubiquitous-space-waddle-j
I  {"status":"OK","books":[{"isbn":"9780988820203","title":"The Docker Book","authors":"-James Turnbull",
```

Code snipet to parse JSON using Gson library in ExploreFragment.java



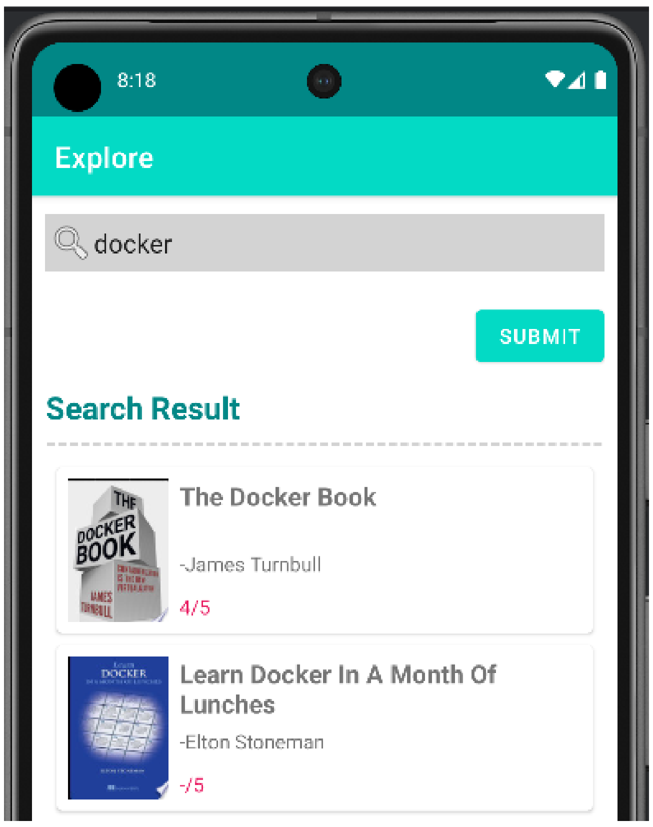
```
    private void showRecyclerList(String data){
        Gson gsonResponse = new Gson();
        BookAdapter bookAdapter = gsonResponse.fromJson(data, BookAdapter.class);
```

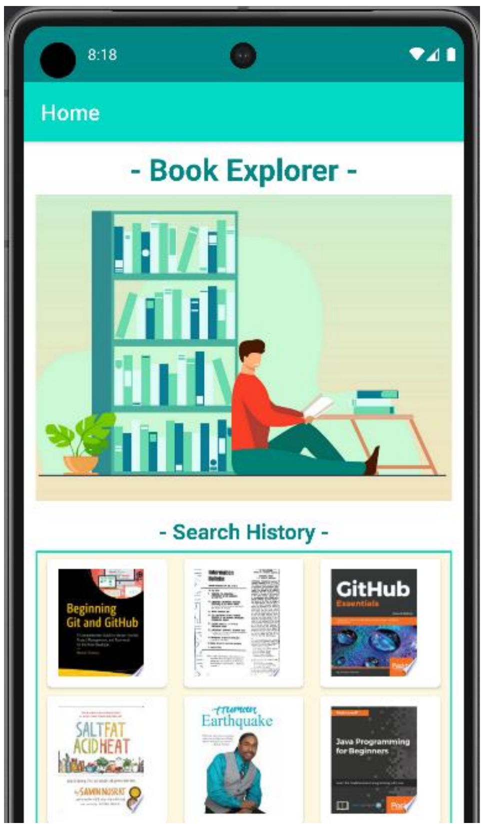
BookAdapter.java has innerclass Book.java as java bean



```
public Book(String isbn, String title, Str
    this.isbn = isbn;
    this.title = title;
    this.authors = authors;
    this.publisher = publisher;
    this.year = year;
    this.category = category;
    this.desc = desc;
    this.pageCount = pageCount;
    this.rating = rating;
    this.thumbnail = thumbnail;
    this.id = id;
    this.statusFavorite = statusFavorite;
}
```

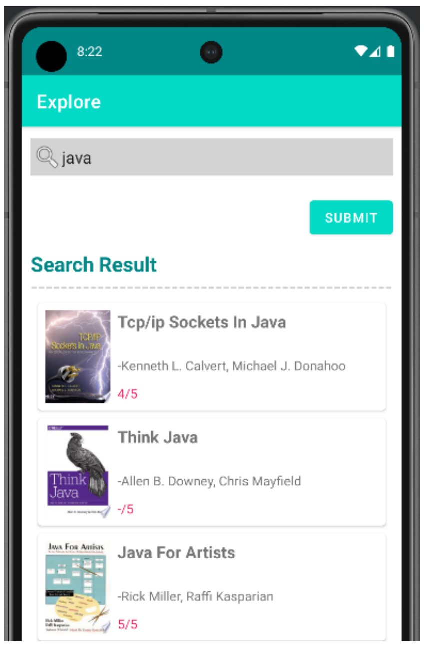
e. Displays new information to the user
Android app will show search result using ExploreFragment.java and there is also search history in main page (HomeFragment.java)



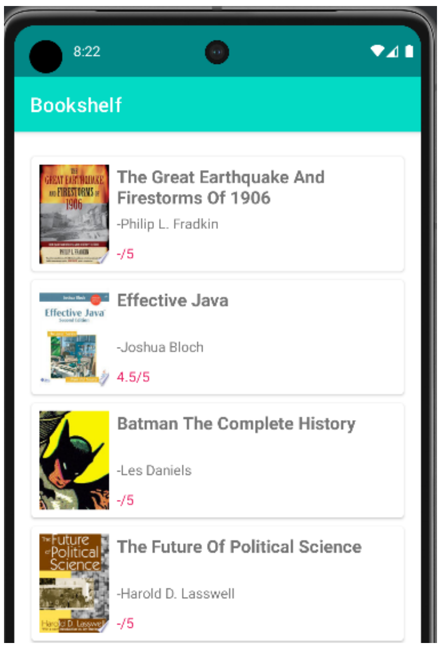


f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

After docker searching, application will give new search result about Java Book without restrating. It is also satisfied by condition when user open bookshelf page. App will automatically retrieve data without restarting.

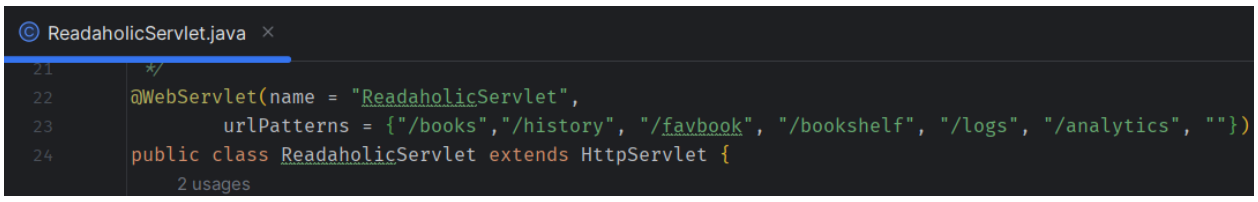


8:22
Explore
java
SUBMIT
Search Result
Tcp/ip Sockets In Java
-Kenneth L. Calvert, Michael J. Donahoo
4/5
Think Java
-Allen B. Downey, Chris Mayfield
-/5
Java For Artists
-Rick Miller, Raffi Kasparian
5/5



8:22
Bookshelf
The Great Earthquake And Firestorms Of 1906
-Philip L. Fradkin
-/5
Effective Java
-Joshua Bloch
4.5/5
Batman The Complete History
-Les Daniels
-/5
The Future Of Political Science
-Harold D. Lasswell
-/5

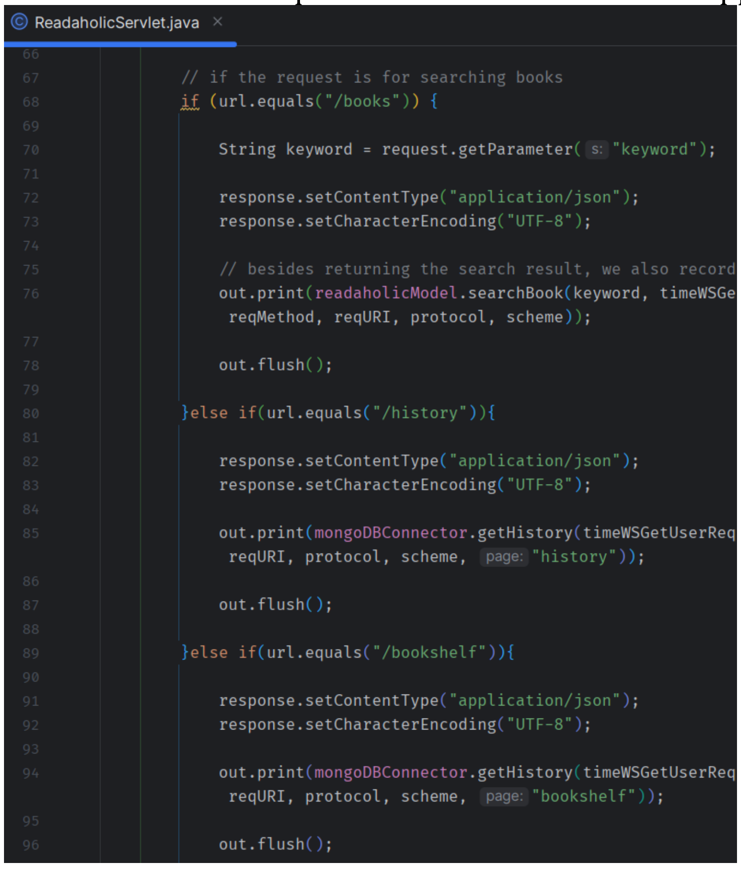
2. Implement a web service
   The detail information about deployed web service is in point 7
   a. Implement a simple (can be a single path) API.
      Webservice provides several endpoints to handle request from Android and dashboard application. I only use one servlet, ReadaholicServlet.java, utilizing if condition in request handling. For Android request has been clear in point 1c and for Dashboard app request including "/logs" to get log data from mongoDB server and "/analytics" to get interesting information in our system to do operational and user bahivour analysist.



ReadaholicServlet.java

```
      */
@WebServlet(name = "ReadaholicServlet",
        urlPatterns = {"/books","/history", "/favbook", "/bookshelf", "/logs", "/analytics", ""})
public class ReadaholicServlet extends HttpServlet {
    2 usages
```

   b. Receives an HTTP request from the native Android application

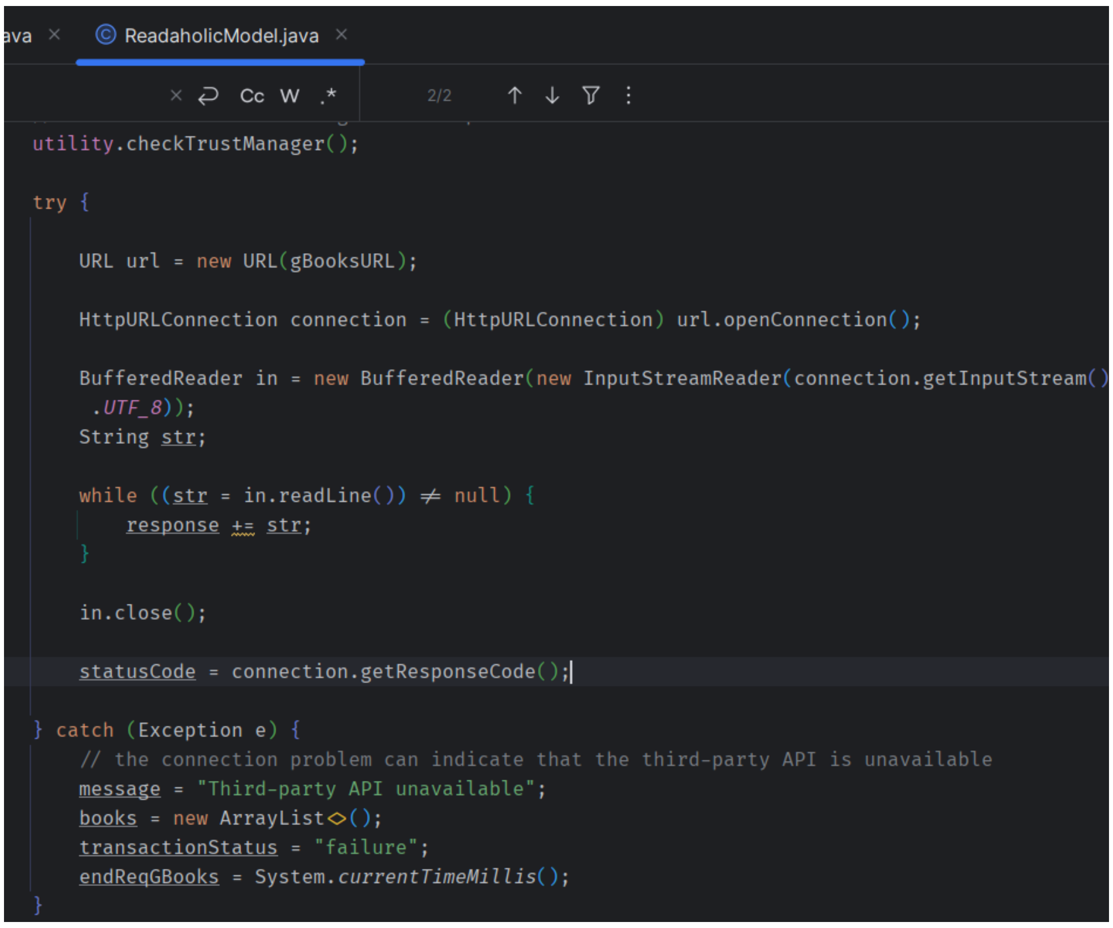


ReadaholicServlet.java

```
            // if the request is for searching books
            if (url.equals("/books")) {

                String keyword = request.getParameter( s: "keyword");

                response.setContentType("application/json");
                response.setCharacterEncoding("UTF-8");

                // besides returning the search result, we also record
                out.print(readaholicModel.searchBook(keyword, timeWSGe
                 reqMethod, reqURI, protocol, scheme));

                out.flush();

            }else if(url.equals("/history")){

                response.setContentType("application/json");
                response.setCharacterEncoding("UTF-8");

                out.print(mongoDBConnector.getHistory(timeWSGetUserReq
                 reqURI, protocol, scheme,  page: "history"));

                out.flush();

            }else if(url.equals("/bookshelf")){

                response.setContentType("application/json");
                response.setCharacterEncoding("UTF-8");

                out.print(mongoDBConnector.getHistory(timeWSGetUserReq
                 reqURI, protocol, scheme,  page: "bookshelf"));

                out.flush();
```

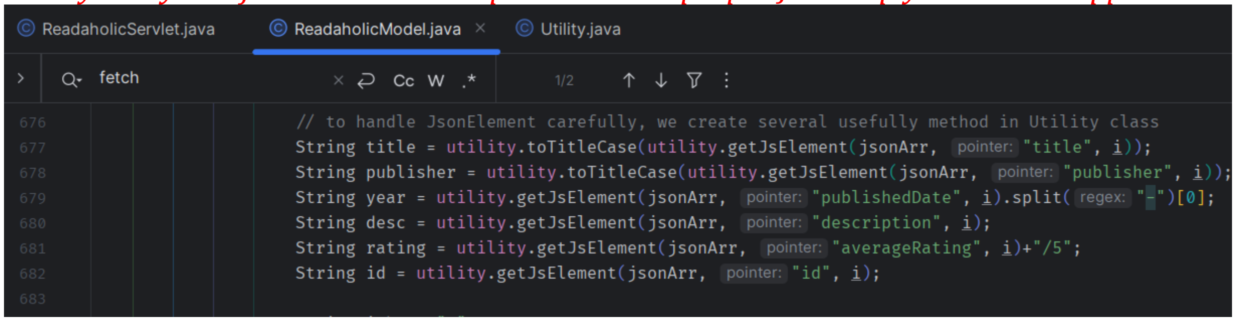
Code snipet above shows HTTP request communication between Android and Web Service. For example, Android will send request by accessing endpoint "/books?keyword=docker" and the keyword parameter will be extracted by webserver using request.getParameter("keyword")

c. Executes business logic appropriate to your application. This includes fetching XML or JSON information from some 3rd party API and processing the response. No a banned API and screen scraping
Business logic is processed in Model "ReadaholicModel.java" with "Utility.java" helper class



```java
utility.checkTrustManager();

try {

    URL url = new URL(gBooksURL);

    HttpURLConnection connection = (HttpURLConnection) url.openConnection();

    BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()
     .UTF_8));
    String str;

    while ((str = in.readLine()) ≠ null) {
        response += str;
    }

    in.close();

    statusCode = connection.getResponseCode();

} catch (Exception e) {
    // the connection problem can indicate that the third-party API is unavailable
    message = "Third-party API unavailable";
    books = new ArrayList<>();
    transactionStatus = "failure";
    endReqGBooks = System.currentTimeMillis();
}
```

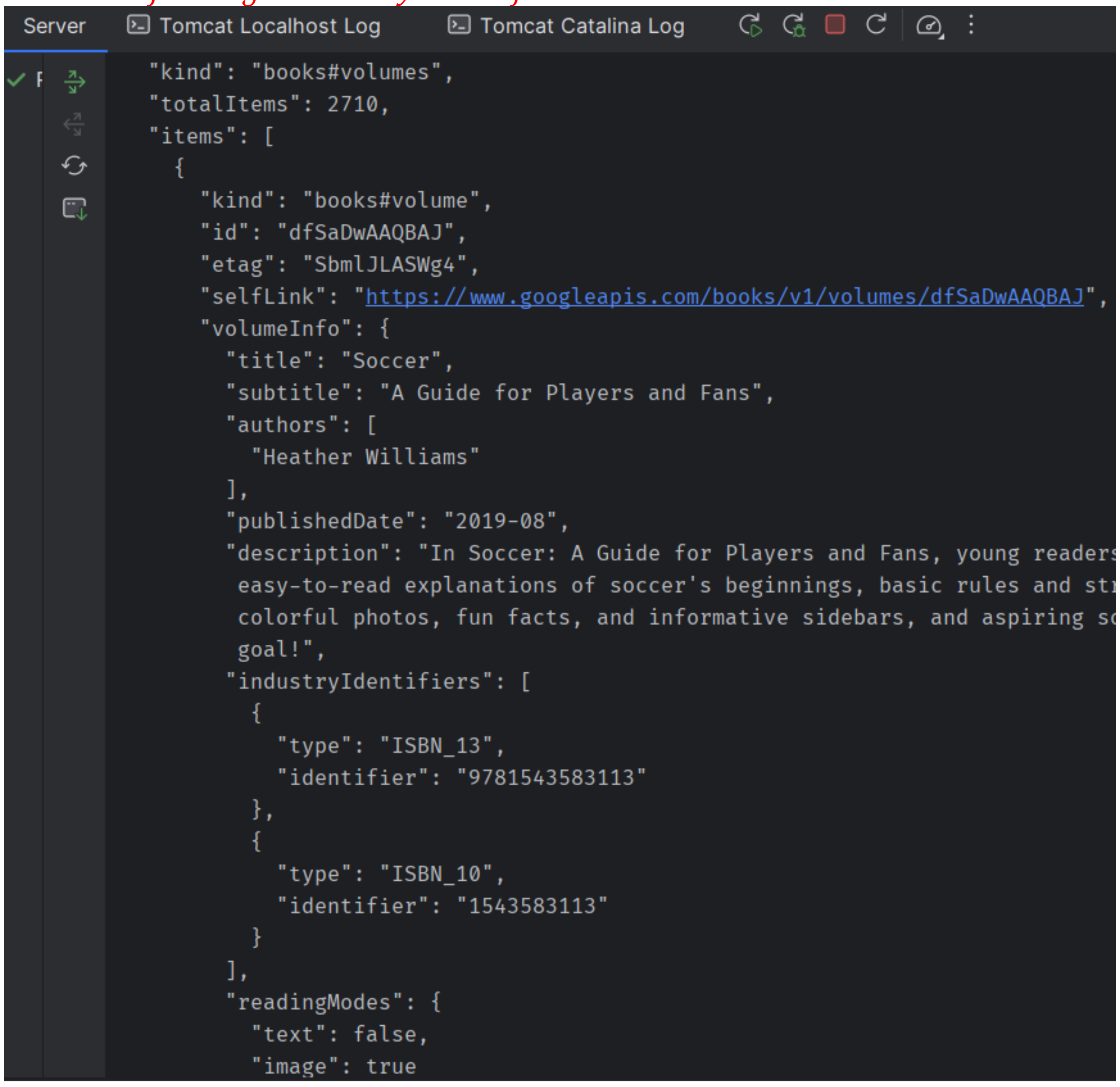
After fetching data above, the response message from Google Book will be handled by Gson library array and json element to help us structure proper JSON reply to android application



```java
// to handle JsonElement carefully, we create several usefully method in Utility class
String title = utility.toTitleCase(utility.getJsElement(jsonArr, pointer: "title", i));
String publisher = utility.toTitleCase(utility.getJsElement(jsonArr, pointer: "publisher", i));
String year = utility.getJsElement(jsonArr, pointer: "publishedDate", i).split( regex: "-")[0];
String desc = utility.getJsElement(jsonArr, pointer: "description", i);
String rating = utility.getJsElement(jsonArr, pointer: "averageRating", i)+"/5";
String id = utility.getJsElement(jsonArr, pointer: "id", i);
```

d. Replies to the Android application with an XML or JSON formatted response as required by user. The web service should select and pass on only the information that the mobile app needs. Use Servlets, not JAX-RS, for your web services.

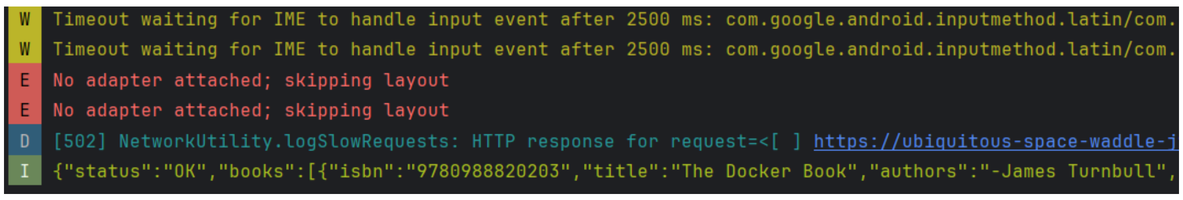
Here is the snipet of a bunch of data we got from Google Book API that will be handled in Readaholic.java to get necessary book information.



```
"kind": "books#volumes",
"totalItems": 2710,
"items": [
  {
    "kind": "books#volume",
    "id": "dfSaDwAAQBAJ",
    "etag": "SbmlJLASWg4",
    "selfLink": "https://www.googleapis.com/books/v1/volumes/dfSaDwAAQBAJ",
    "volumeInfo": {
      "title": "Soccer",
      "subtitle": "A Guide for Players and Fans",
      "authors": [
        "Heather Williams"
      ],
      "publishedDate": "2019-08",
      "description": "In Soccer: A Guide for Players and Fans, young readers
       easy-to-read explanations of soccer's beginnings, basic rules and str
       colorful photos, fun facts, and informative sidebars, and aspiring sc
       goal!",
      "industryIdentifiers": [
        {
          "type": "ISBN_13",
          "identifier": "9781543583113"
        },
        {
          "type": "ISBN_10",
          "identifier": "1543583113"
        }
      ],
      "readingModes": {
        "text": false,
        "image": true
```

From the data above, our web service will format JSON to android app covering information about. Here, we also get encoded base64 image string from webservice to avoid our Android app doing much effort in data downloading process.

{"status":"OK","books":[{"isbn":"9781543583113","title":"Soccer","authors":"-Heather Williams","publisher":"-","year":"2019","category":"Juvenile Nonfiction","desc":"In Soccer: A Guide for Players and Fans, young readers can dive into one of the world\u0027s most popular sports. Readers will find easy-to-read explanations of soccer\u0027s beginnings, basic rules and strategies, and how they can suit up and get on the pitch. This book features colorful photos, fun facts, and informative sidebars, and aspiring soccer players will take what they learn from it and head straight to the goal!","rating":"-/5","thumbnail":"/9j/4AAQSkZJRgABAQEAE….}



```
W  Timeout waiting for IME to handle input event after 2500 ms: com.google.android.inputmethod.latin/com.
W  Timeout waiting for IME to handle input event after 2500 ms: com.google.android.inputmethod.latin/com.
E  No adapter attached; skipping layout
E  No adapter attached; skipping layout
D  [502] NetworkUtility.logSlowRequests: HTTP response for request=<[ ] https://ubiquitous-space-waddle-j
I  {"status":"OK","books":[{"isbn":"9780988820203","title":"The Docker Book","authors":"-James Turnbull",
```

3. Handle error conditions gracefully
Although we don't need to explain these points, I will describe brief information how my system addresses various type of errors below. I utilize message popup "android toast" to inform to the users about error they get.

No	Errors	Condition	File
a	*Invalid mobile app input*	*Search input is empty, invalid keyword length, invalid keyword format*	*Android-ExploreFragment.java*
b	*Invalid server-side input*	*Similar with point (a), but it is handled in webservice*	*WebService-ReadholicModel.java*
c	*Mobile app network failure*	*No internet connection*	*Android-HomeFragment.java*
d	*Third-party API unavailable*	*Webservice can not connect to Google Book API*	*Webservice-ReadaholicModel.java*
e	*Third-party API invalid data*	*Google Book API reply invalid JSON format to Webservice*	*Webserviec-ReadaholicModel.java*

Web Service Logging and Analysis Dashboard

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.
Operation Analytics:
- *Daily latency data trend per second in line chart*
- *The proportion of endpoint / routing accessed by webservice users. Ex. "/history", "/favbook", "/booksheld" etc*
- *The proportion of transaction status whether fail or success*
- *The proportion of software devices / user agent. Ex. Android 11, 12, 10 etc*

User Behaviour Anlytics:
- *Top 5 loveable book category*
- *Top 5 popular search keyword*

Dashboard screenshot will be available at the end of this document.

3.2. Logs – display the data logs being stored for each mobile phone user interaction with your web service. The display of each log entry can be simply formatted and should be easily readable.

Webservice will log data using ReadaholicModel.java file. Information has been logged:

Information	Description	Reason
timeWSGetUserRequest	*Timestamp webservice getting request from users*	*To count latency and system trend analysis*
timeWSSendReqToOthers	*Timestamp webservice send request to GoogleBook API*	*To count latency and system trend analysis*
timeWSGetRepFromOthers	*Timestamp webservice get respon from Google Book API*	*To count latency and system trend analysis*
timeWSSendRepToUser	*Timestamp webservice send reply to users*	*To count latency and system trend analysis*
userAgent	*Dalvik/2.1.0 (Linux; U; Android 11; sdk_gphone_x86 Build/RSR1.201013.001)*	*To analyze user device*
softwareAgent	*Spesific type of OS or software. Ex. Android 11*	*To analyze sofware device*
userIP	*User IP Address*	*userIP can act as userID*
userPort	*User Port*	*It will complement userIP*
routing	*API endpoint. Ex: "/history", "/bookshelf"*	*To analyze popular endpoint*
keyWord	*Search term inputted by users*	*To analyze user behaviour*
WSProcessingTime	*How long webservice can process user request*	*To count WS latency*
otherPartyProcessingTime	*How long Google Book API can response*	*To count 3rd party latency*
transactionStatus	*Failure or Success*	*To get to know transaction status*

requestMethod	*GET, POST, etc*	*To analyze type of requestMethod*
protocol	*Request protocol. Ex. HTTP/1.1*	*To get user protocol*
scheme	*http*	*Similar with protocol*
countData	*The number of data replying to users or getting from GoogleAPI*	*To count how the number of data processed by our system*
requestQuery	*Query we did to get data from mongoDB. It could also the URI we execute to get data from Google API*	*Audit trail*
jsonString	*The full respon data from Google or MongoDB in JSON formatted*	*The get full of data transaction*

3.3. You will likely find HTML tables useful for formatting tabular information on a web page. You may use a client-side framework if you like (e.g. Twitter Bootstrap). Don't just use the json or xml format, use a table.
Here the list of JS library I used to enhance dashboard interface in index.jsp, logs.jsp and analytics.jsp as our Dashboard view.

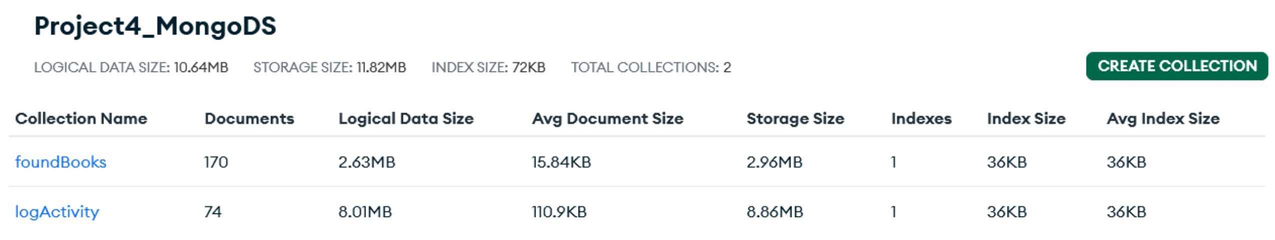
- *Amcharts: Javascript library for charting*
- *Fontawesome: CSS library for icon*
- *Bootstrap 5: CSS and HTML library to enhance UI/UX*
- *Datatables: JS and CSS library to format data in dashboard logs*
- *Jquery and Popper: JS library to help interact with UI element in browser*

Web Service Logging and Analysis Dashboard Requirements

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. Information can include: parameters as what kind of model of phone has made the request, parameters included in the request specific to your application, timestamps for when requests are received, requests sent to the 3rd party API, and the data sent in the reply back to the phone.
List of information I logged can be seen in point 3.2. It also can be seen on Dashboard log at the end of this document.

5. Store the log information in a database. The web service can connect, store, and retrieve information from a MongoDB database in the cloud.

Here, the screenshoot of MongoDB dashboard


Project4_MongoDS

LOGICAL DATA SIZE: 10.64MB STORAGE SIZE: 11.82MB INDEX SIZE: 72KB TOTAL COLLECTIONS: 2

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
foundBooks	170	2.63MB	15.84KB	2.96MB	1	36KB	36KB
logActivity	74	8.01MB	110.9KB	8.86MB	1	36KB	36KB

Log information will be stored in MongoDB server with detail information below:

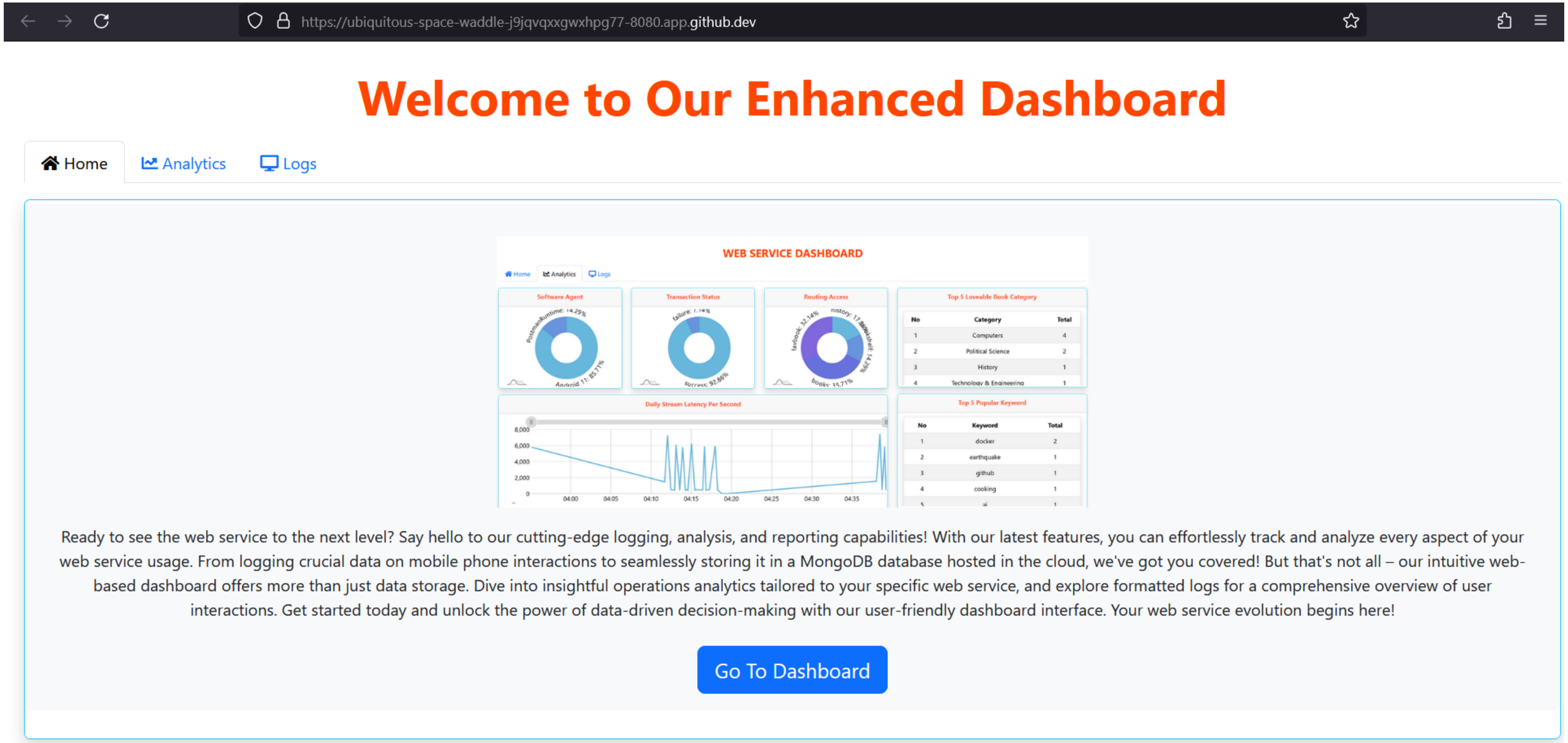
URI	*mongodb://afurqan:project4dslab@ac-apj3hcs-shard-00-02.a1tj2al.mongodb.net:27017,ac-apj3hcs-shard-00-01.a1tj2al.mongodb.net:27017,ac-apj3hcs-shard-00-00.a1tj2al.mongodb.net:27017/myFirstDatabase?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1*
DB_Name	*Project4_MongoDS*
Collections	*foundBooks – to save book history* *logActivity – to save activity logs*

6. Display operations analytics and full logs on a web-based dashboard
   All information below can be found at dashboard screenshoot
   a. A unique URL addresses a web interface dashboard for the web service.
   b. The dashboard displays at least 3 interesting operations analytics.
   c. The dashboard displays formatted full logs.

7. Deploy the web service to GitHub Codespaces
   a. Accept the Github Classroom Assignment that you have been given the URL for.
      Yes, It has been done on canvas

   b. To deploy your own web service, create a ROOT.war like you did in Lab 3, upload or push the ROOT.war to your repository, and create a Codespace as has just been described.

Repository	*experience-primer-copilot-oldmanstreetcoding*
Repository URL	*https://github.com/Exp-Primer-Copilot-Cohort-1/experience-primer-copilot-oldmanstreetcoding*
Codespace ID	*ubiquitous space waddle*
Webservice Dashboard	*https://ubiquitous-space-waddle-j9jqvqxxgwxhpg77.github.dev/*



Welcome to Our Enhanced Dashboard

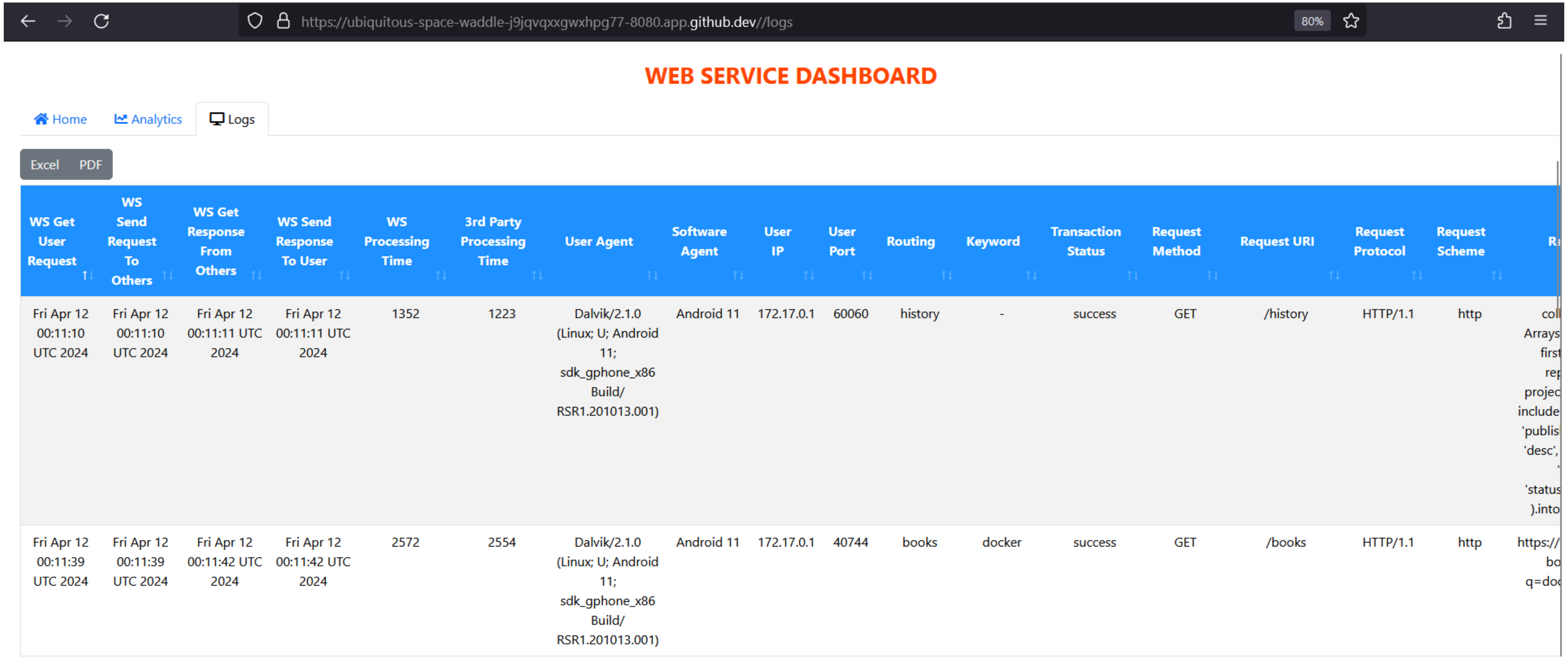
Top 5 Loveable Book Category

No	Category	Total
1	Computers	4
2	Political Science	2
3	History	1
4	Technology & Engineering	1

Top 5 Popular Keyword

No	Keyword	Total
1	docker	2
2	earthquake	1
3	github	1
4	cooking	1
5	ai	1

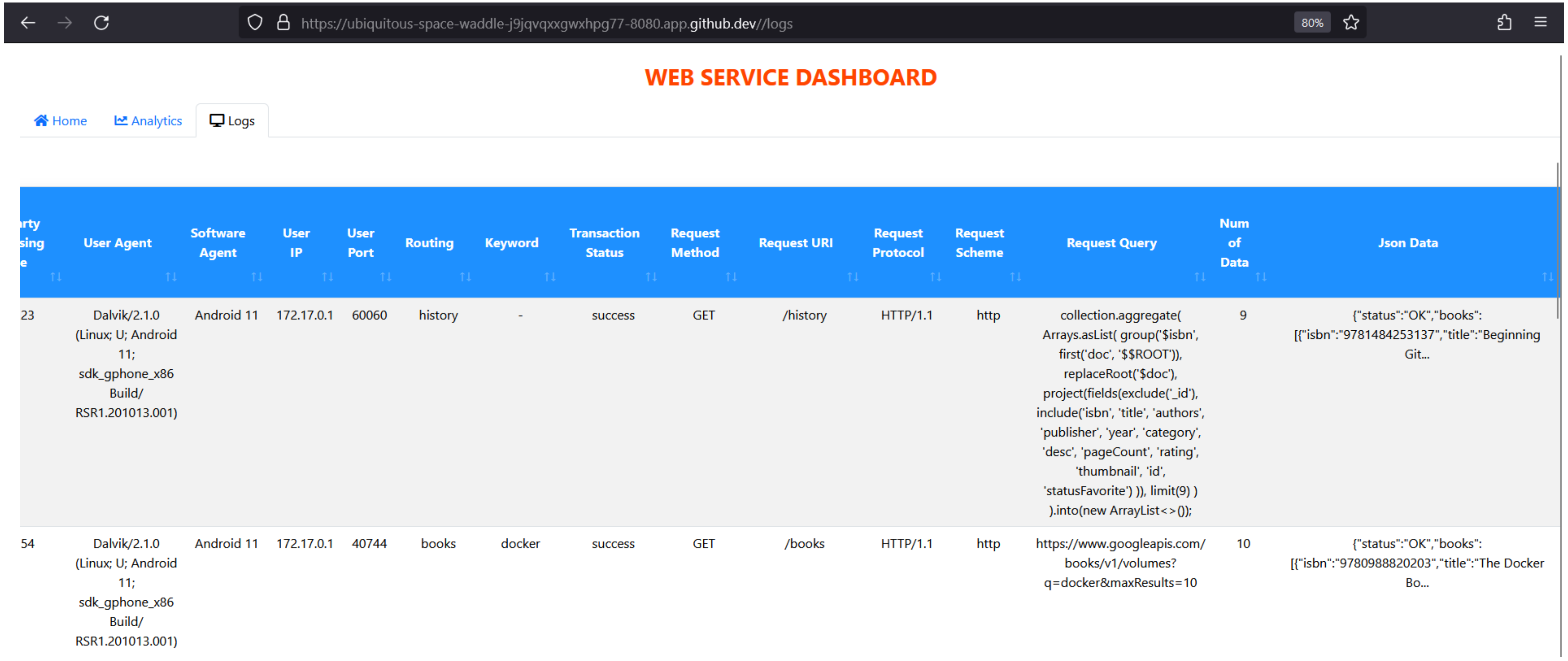
Ready to see the web service to the next level? Say hello to our cutting-edge logging, analysis, and reporting capabilities! With our latest features, you can effortlessly track and analyze every aspect of your web service usage. From logging crucial data on mobile phone interactions to seamlessly storing it in a MongoDB database hosted in the cloud, we've got you covered! But that's not all – our intuitive web-based dashboard offers more than just data storage. Dive into insightful operations analytics tailored to your specific web service, and explore formatted logs for a comprehensive overview of user interactions. Get started today and unlock the power of data-driven decision-making with our user-friendly dashboard interface. Your web service evolution begins here!


https://ubiquitous-space-waddle-j9jqvqxxgwxhpg77-8080.app.github.dev//logs
80%

WEB SERVICE DASHBOARD

Home
Analytics
Logs
Excel
PDF

WS Get User Request	WS Send Request To Others	WS Get Response From Others	WS Send Response To User	WS Processing Time	3rd Party Processing Time	User Agent	Software Agent	User IP	User Port	Routing	Keyword	Transaction Status	Request Method	Request URI	Request Protocol	Request Scheme	R
Fri Apr 12 00:11:10 UTC 2024	Fri Apr 12 00:11:10 UTC 2024	Fri Apr 12 00:11:11 UTC 2024	Fri Apr 12 00:11:11 UTC 2024	1352	1223	Dalvik/2.1.0 (Linux; U; Android 11; sdk_gphone_x86 Build/ RSR1.201013.001)	Android 11	172.17.0.1	60060	history	-	success	GET	/history	HTTP/1.1	http	coll Arrays first rep projec include 'publis 'desc', ' 'status).into
Fri Apr 12 00:11:39 UTC 2024	Fri Apr 12 00:11:39 UTC 2024	Fri Apr 12 00:11:42 UTC 2024	Fri Apr 12 00:11:42 UTC 2024	2572	2554	Dalvik/2.1.0 (Linux; U; Android 11; sdk_gphone_x86 Build/ RSR1.201013.001)	Android 11	172.17.0.1	40744	books	docker	success	GET	/books	HTTP/1.1	http	https:// bo q=doc



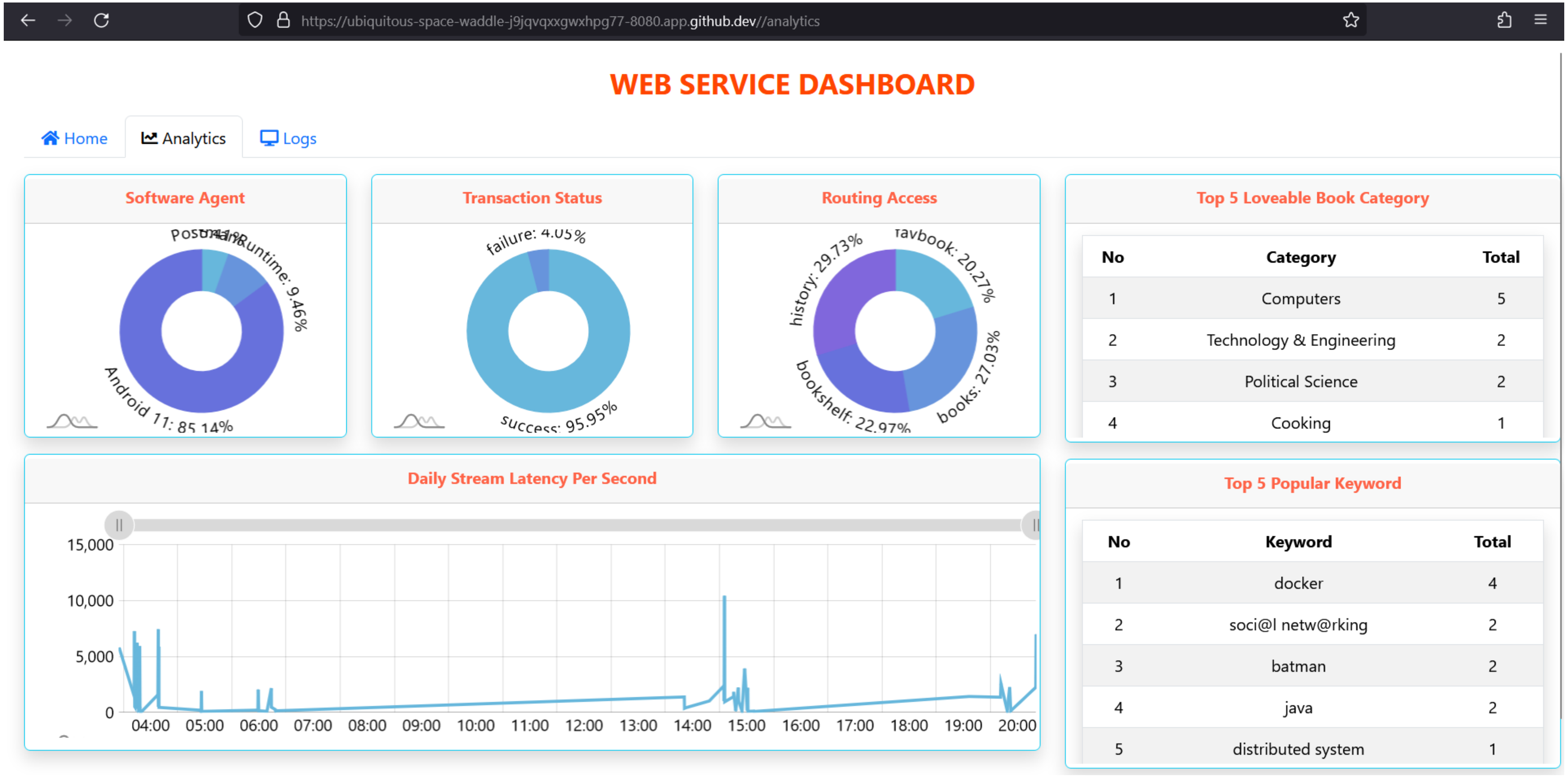
https://ubiquitous-space-waddle-j9jqvqxxgwxhpg77-8080.app.github.dev//logs

80%

WEB SERVICE DASHBOARD

Home | Analytics | Logs

rty sing e	User Agent	Software Agent	User IP	User Port	Routing	Keyword	Transaction Status	Request Method	Request URI	Request Protocol	Request Scheme	Request Query	Num of Data	Json Data
23	Dalvik/2.1.0 (Linux; U; Android 11; sdk_gphone_x86 Build/ RSR1.201013.001)	Android 11	172.17.0.1	60060	history	-	success	GET	/history	HTTP/1.1	http	collection.aggregate(Arrays.asList(group('$isbn', first('doc', '$$ROOT')), replaceRoot('$doc'), project(fields(exclude('_id'), include('isbn', 'title', 'authors', 'publisher', 'year', 'category', 'desc', 'pageCount', 'rating', 'thumbnail', 'id', 'statusFavorite'))), limit(9))).into(new ArrayList<>());	9	{"status":"OK","books": [{"isbn":"9781484253137","title":"Beginning Git...
54	Dalvik/2.1.0 (Linux; U; Android 11; sdk_gphone_x86 Build/ RSR1.201013.001)	Android 11	172.17.0.1	40744	books	docker	success	GET	/books	HTTP/1.1	http	https://www.googleapis.com/ books/v1/volumes? q=docker&maxResults=10	10	{"status":"OK","books": [{"isbn":"9780988820203","title":"The Docker Bo...


https://ubiquitous-space-waddle-j9jqvqxxgwxhpg77-8080.app.github.dev//analytics

WEB SERVICE DASHBOARD

Home | Analytics | Logs

Software Agent

Transaction Status

Routing Access

Top 5 Loveable Book Category

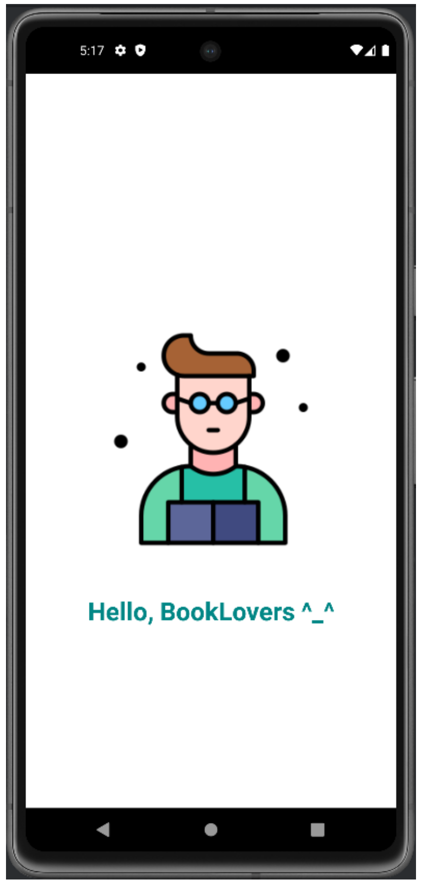
No	Category	Total
1	Computers	5
2	Technology & Engineering	2
3	Political Science	2
4	Cooking	1

Daily Stream Latency Per Second

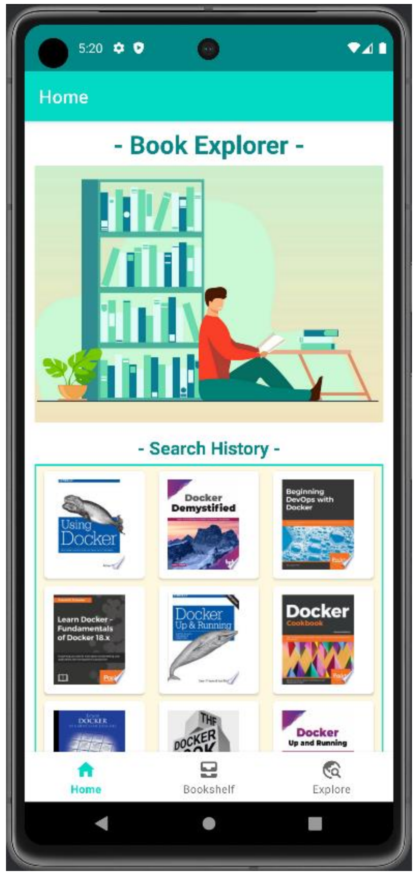
Top 5 Popular Keyword

No	Keyword	Total
1	docker	4
2	soci@l netw@rking	2
3	batman	2
4	java	2
5	distributed system	1

Android Application Design


5:17

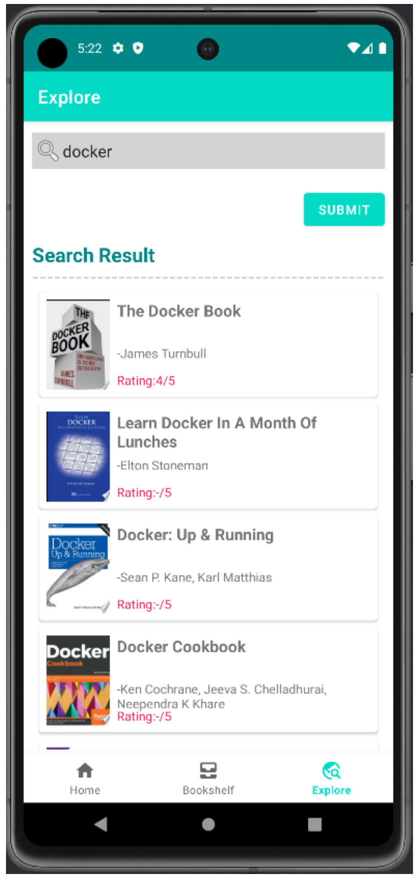
Hello, BookLovers ^_^


5:20

Home

- Book Explorer -

- Search History -


5:22

Explore

docker

SUBMIT

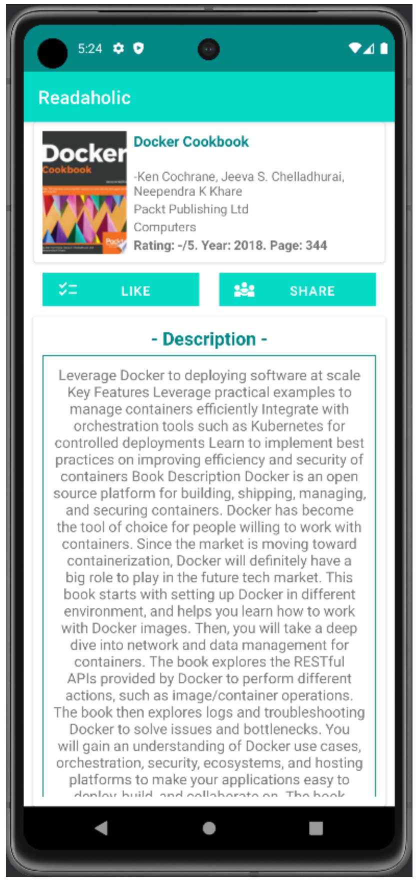
Search Result

The Docker Book
-James Turnbull
Rating:4/5

Learn Docker In A Month Of Lunches
-Elton Stoneman
Rating:-/5

Docker: Up & Running
-Sean P. Kane, Karl Matthias
Rating:-/5

Docker Cookbook
-Ken Cochrane, Jeeva S. Chelladhurai, Neependra K Khare
Rating:-/5

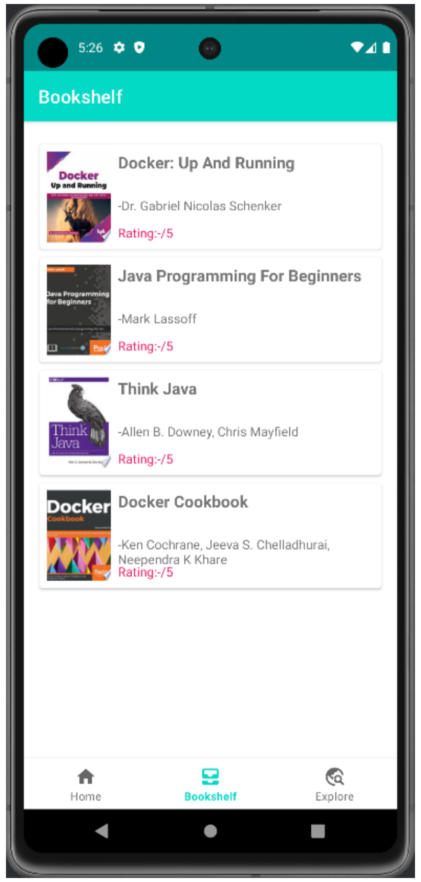

5:24

Readaholic

Docker Cookbook
-Ken Cochrane, Jeeva S. Chelladhurai, Neependra K Khare
Packt Publishing Ltd
Computers
Rating: -/5. Year: 2018. Page: 344

- Description -

Leverage Docker to deploying software at scale Key Features Leverage practical examples to manage containers efficiently Integrate with orchestration tools such as Kubernetes for controlled deployments Learn to implement best practices on improving efficiency and security of containers Book Description Docker is an open source platform for building, shipping, managing, and securing containers. Docker has become the tool of choice for people willing to work with containers. Since the market is moving toward containerization, Docker will definitely have a big role to play in the future tech market. This book starts with setting up Docker in different environment, and helps you learn how to work with Docker images. Then, you will take a deep dive into network and data management for containers. The book explores the RESTful APIs provided by Docker to perform different actions, such as image/container operations. The book then explores logs and troubleshooting Docker to solve issues and bottlenecks. You will gain an understanding of Docker use cases, orchestration, security, ecosystems, and hosting platforms to make your applications easy to


5:26

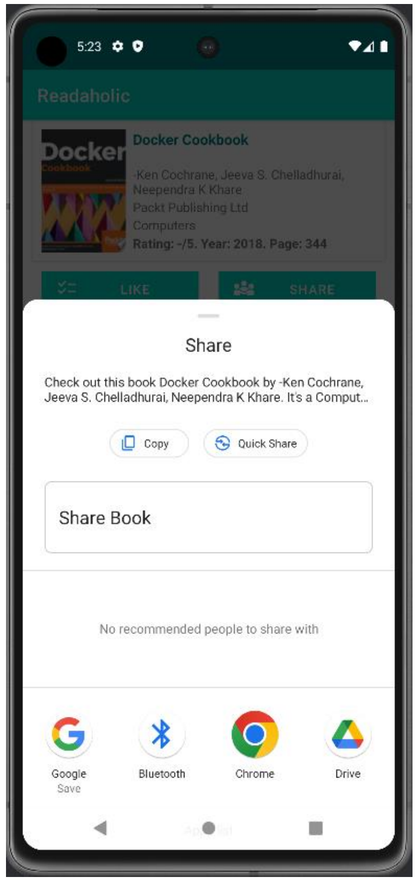
Bookshelf

Docker: Up And Running
-Dr. Gabriel Nicolas Schenker
Rating:-/5

Java Programming For Beginners
-Mark Lassoff
Rating:-/5

Think Java
-Allen B. Downey, Chris Mayfield
Rating:-/5

Docker Cookbook
-Ken Cochrane, Jeeva S. Chelladhurai, Neependra K Khare
Rating:-/5


5:23

Readaholic

Docker Cookbook
-Ken Cochrane, Jeeva S. Chelladhurai, Neependra K Khare
Packt Publishing Ltd
Computers
Rating: -/5. Year: 2018. Page: 344

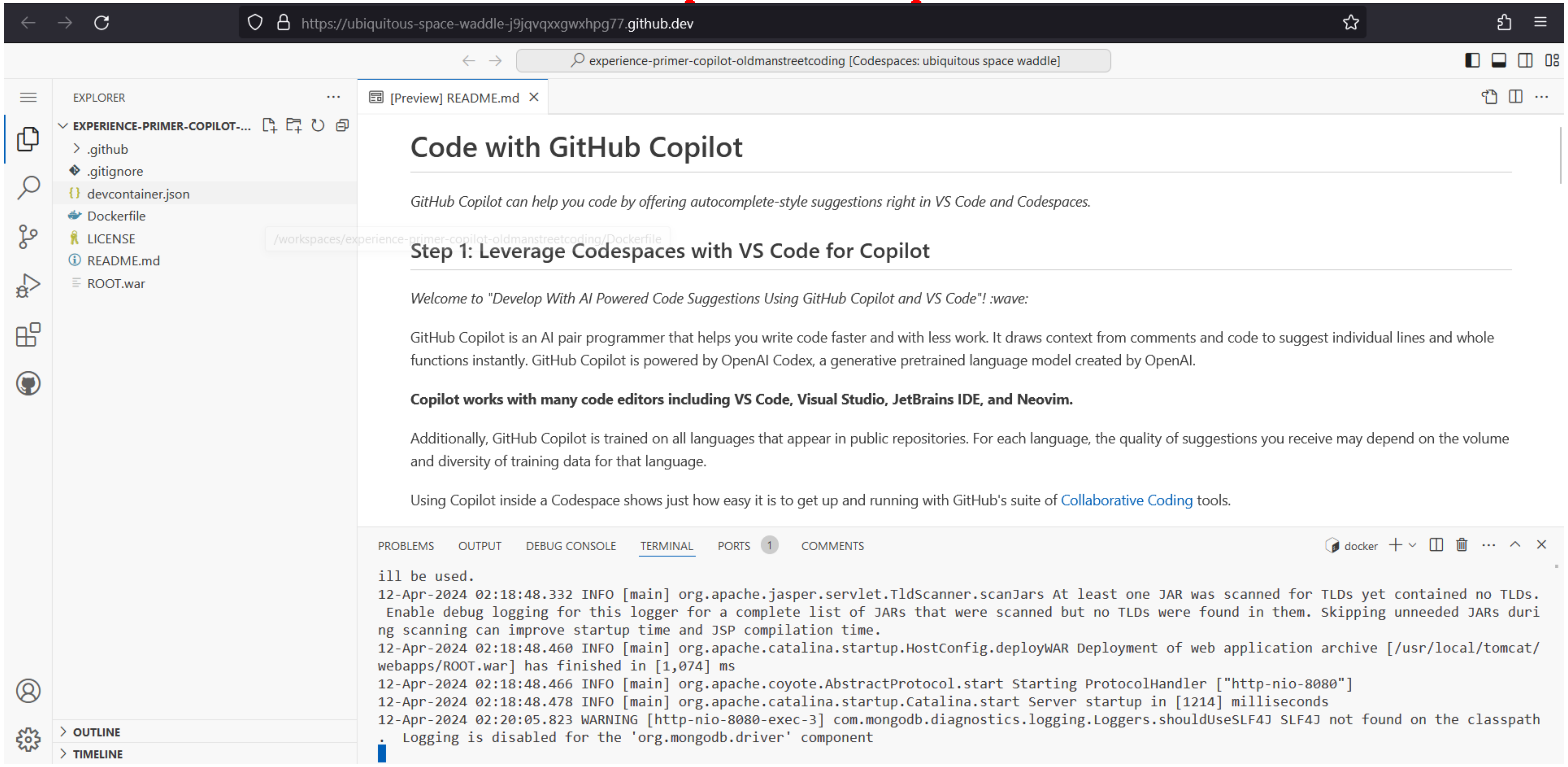
Share

Check out this book Docker Cookbook by -Ken Cochrane, Jeeva S. Chelladhurai, Neependra K Khare. It's a Comput...

Share Book

No recommended people to share with

Codespace Workspace



https://ubiquitous-space-waddle-j9jqvqxxgwxhpg77.github.dev

experience-primer-copilot-oldmanstreetcoding [Codespaces: ubiquitous space waddle]

Code with GitHub Copilot

GitHub Copilot can help you code by offering autocomplete-style suggestions right in VS Code and Codespaces.

Step 1: Leverage Codespaces with VS Code for Copilot

Welcome to "Develop With AI Powered Code Suggestions Using GitHub Copilot and VS Code"! :wave:

GitHub Copilot is an AI pair programmer that helps you write code faster and with less work. It draws context from comments and code to suggest individual lines and whole functions instantly. GitHub Copilot is powered by OpenAI Codex, a generative pretrained language model created by OpenAI.

Copilot works with many code editors including VS Code, Visual Studio, JetBrains IDE, and Neovim.

Additionally, GitHub Copilot is trained on all languages that appear in public repositories. For each language, the quality of suggestions you receive may depend on the volume and diversity of training data for that language.

Using Copilot inside a Codespace shows just how easy it is to get up and running with GitHub's suite of Collaborative Coding tools.

```
ill be used.
12-Apr-2024 02:18:48.332 INFO [main] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet contained no TLDs.
 Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs duri
ng scanning can improve startup time and JSP compilation time.
12-Apr-2024 02:18:48.460 INFO [main] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive [/usr/local/tomcat/
webapps/ROOT.war] has finished in [1,074] ms
12-Apr-2024 02:18:48.466 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
12-Apr-2024 02:18:48.478 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [1214] milliseconds
12-Apr-2024 02:20:05.823 WARNING [http-nio-8080-exec-3] com.mongodb.diagnostics.logging.Loggers.shouldUseSLF4J SLF4J not found on the classpath
.  Logging is disabled for the 'org.mongodb.driver' component
```