Vatsal Sheth
vus

Book App Project – Distributed Application

---

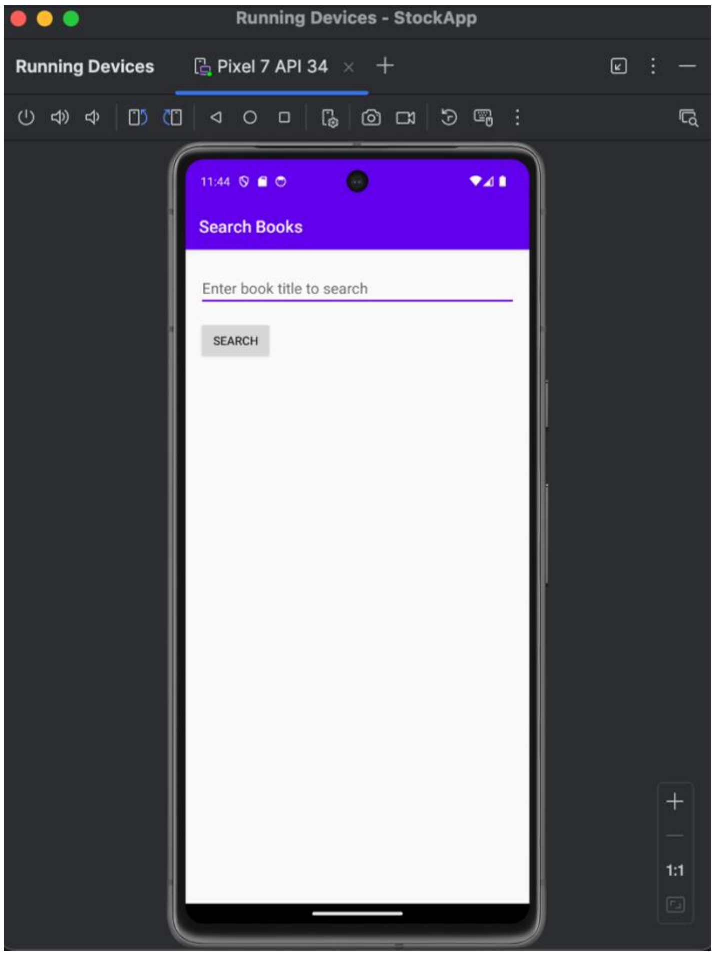
1. Implement a Native Android Application

The name of my native Android application project in Android Studio is: **BookApp**. My mobile app allows users to search for books, view detailed information about selected books, and provides analytics about app usage.

a. Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, etc.)

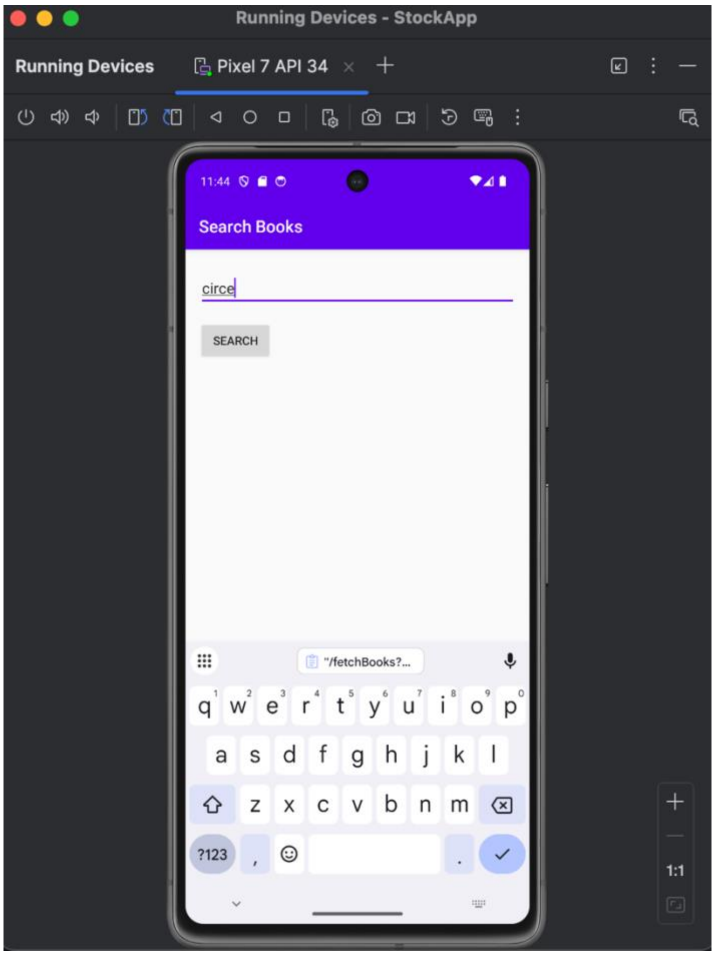
My application uses `TextView`, `EditText`, `Button`, `ImageView`, and `ListView`. These are incorporated into the respective fragments (`SearchFragment`, `ResultsFragment`, and `DetailsFragment`).

- `TextView` displays book details like title, author, description, and publishing information.
- `EditText` captures user input for book searches.
- `Button` initiates search actions.
- `ImageView` displays book cover images.
- `ListView` displays the list of books in the search results.



b. Requires input from the user

The application allows the user to input a book title through the `EditText` field in the `SearchFragment`. Upon clicking the **Search** button, the app fetches relevant data from the Open Library API.



c. Makes an HTTP request (using an appropriate HTTP method) to your web service

The `BookApiService` class makes HTTP GET requests to:

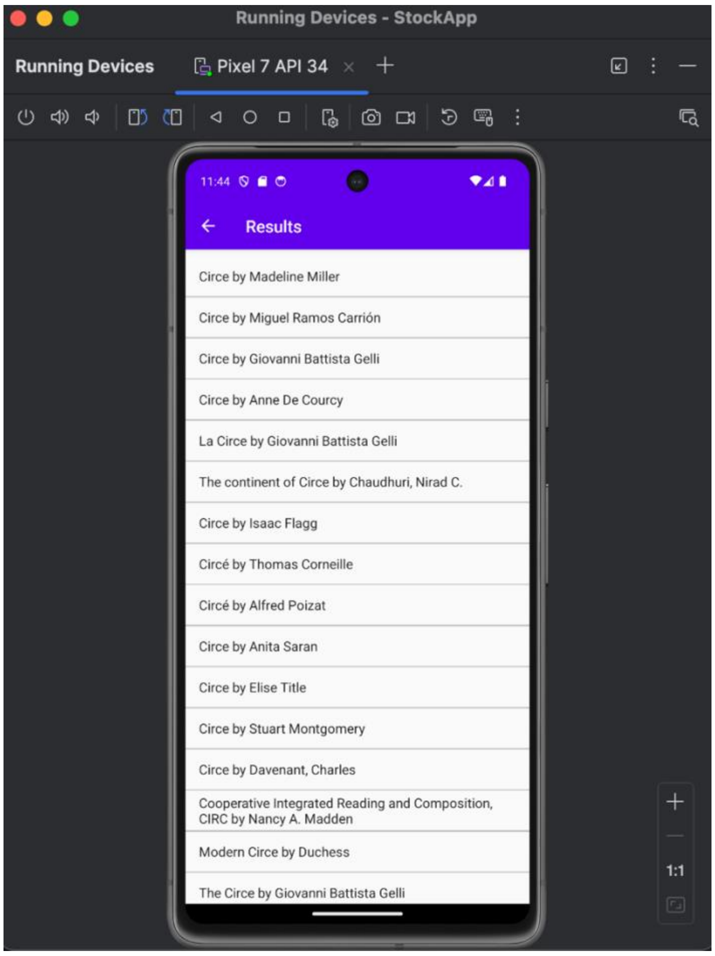
- Search for books by title:
  `https://openlibrary.org/search.json?title=<search_term>`
- Fetch detailed book information:
  `https://openlibrary.org/<work_id>.json`

d. Receives and parses an XML or JSON formatted reply from your web service

The responses are JSON formatted. For example:

Search Results:

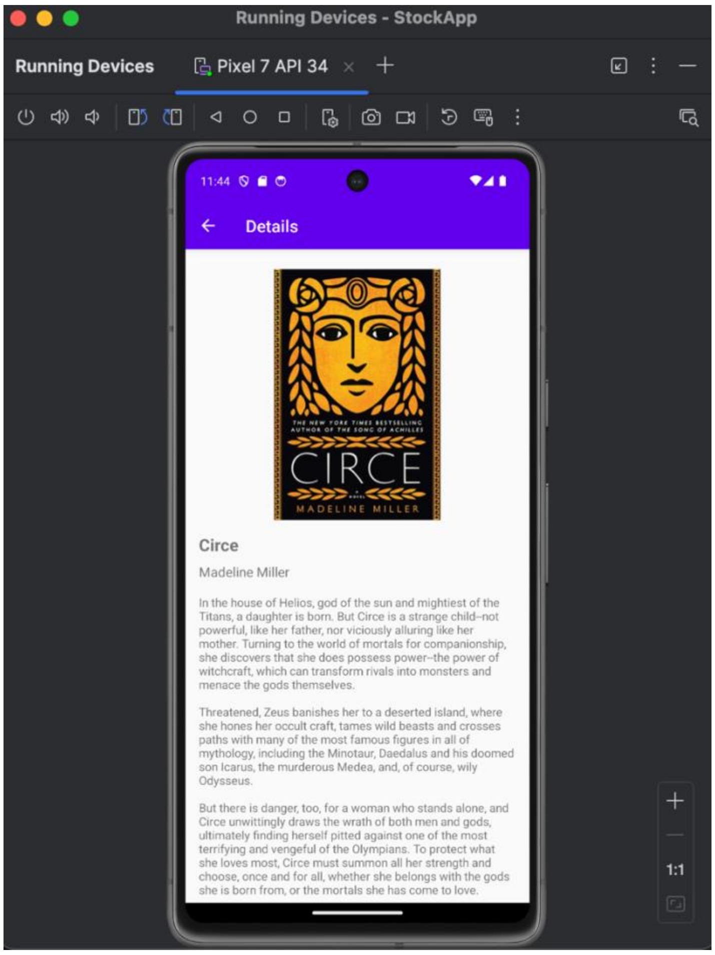
```
json
Copy code
{
  "docs": [
    {
      "title": "Circe",
      "author_name": ["Madeline Miller"],
      "cover_i": 123456,
      "key": "/works/OL12345W"
    }
  ]
}
```



Circe by Madeline Miller
Circe by Miguel Ramos Carrión
Circe by Giovanni Battista Gelli
Circe by Anne De Courcy
La Circe by Giovanni Battista Gelli
The continent of Circe by Chaudhuri, Nirad C.
Circe by Isaac Flagg
Circé by Thomas Corneille
Circé by Alfred Poizat
Circe by Anita Saran
Circe by Elise Title
Circe by Stuart Montgomery
Circe by Davenant, Charles
Cooperative Integrated Reading and Composition, CIRC by Nancy A. Madden
Modern Circe by Duchess
The Circe by Giovanni Battista Gelli

Book Details:

```
json
Copy code
{
  "title": "Circe",
  "description": "A retelling of Greek mythology...",
  "publish_date": "2018",
  "number_of_pages": 400
}
```


Running Devices - StockApp
Running Devices
Pixel 7 API 34
11:44
Details
THE NEW YORK TIMES BESTSELLING AUTHOR OF THE SONG OF ACHILLES
CIRCE
A NOVEL
MADELINE MILLER
1:1

Circe

Madeline Miller

In the house of Helios, god of the sun and mightiest of the Titans, a daughter is born. But Circe is a strange child--not powerful, like her father, nor viciously alluring like her mother. Turning to the world of mortals for companionship, she discovers that she does possess power--the power of witchcraft, which can transform rivals into monsters and menace the gods themselves.

Threatened, Zeus banishes her to a deserted island, where she hones her occult craft, tames wild beasts and crosses paths with many of the most famous figures in all of mythology, including the Minotaur, Daedalus and his doomed son Icarus, the murderous Medea, and, of course, wily Odysseus.

But there is danger, too, for a woman who stands alone, and Circe unwittingly draws the wrath of both men and gods, ultimately finding herself pitted against one of the most terrifying and vengeful of the Olympians. To protect what she loves most, Circe must summon all her strength and choose, once and for all, whether she belongs with the gods she is born from, or the mortals she has come to love.

---

e. Displays new information to the user

After parsing the response:

- `ResultsFragment` displays a list of books matching the search term.
- On selecting a book, the `DetailsFragment` shows details like the title, author, description, publishing date, and cover image.

---

f. Is repeatable

The user can search for books multiple times without restarting the app. The app resets the search state for new inputs.

2. Implement a Web Service

Project Directory Name: `BookWebService`

a. Simple API Implementation

The web service provides the following functionalities:

1. **Log Submission Endpoint:**
   `POST /log`
   Accepts logs from the Android application and stores them in MongoDB for analytics and debugging.
2. **Dashboard Endpoint:**
   `GET /dashboard`
   Displays operational analytics and log details in a web-based interface.
3. **Fetch Books Endpoint:**
   `GET /books?query=<search_query>`
   Interacts with the OpenLibrary API to fetch a list of books matching the search query and returns the results in JSON format.
4. **Fetch Book Facts Endpoint:**
   `GET /book/facts?workId=<work_id>`
   Retrieves detailed information about a book, including title, description, publish date, and page count, by interacting with the OpenLibrary API.

---

2. Receiving HTTP Requests

Log Submission:
The `LogServlet` receives HTTP POST requests containing log data from the Android app. The logs include request and response metadata, device details, and API performance metrics.

Fetch Books:
The `FetchBooksServlet` handles GET requests to search for books using the OpenLibrary API. It accepts a query string parameter (`query`) and returns a JSON response containing book titles, authors, and cover image URLs.

Fetch Book Facts:
The `FetchBookFactsServlet` handles GET requests to fetch detailed book information using the OpenLibrary API. It accepts a `workId` parameter and returns a JSON response with detailed book metadata.

Dashboard:
The `DashboardServlet` handles GET requests to display operational analytics and logs in a browser-friendly table format.

---

3. Business Logic

1. **Logs:**
   All incoming requests from the Android application are logged with attributes such as request timestamps, device information, API endpoints, and response metadata. These logs are stored in a MongoDB collection for further analysis and debugging.
2. **Book Data Fetching:**
   - **Fetch Books**: Queries the OpenLibrary API with a search term to retrieve a list of books.
   - **Fetch Book Facts**: Uses the OpenLibrary API to fetch detailed information for a specific book identified by its `workId`.
3. **Dashboard Analytics:**
   The dashboard dynamically calculates and displays analytics, including:
   - **Top Search Queries**
   - **Most Common Device Models**
   - **Average API Latency**

---

4. Log Information

The logs store critical information for debugging and analytics, including:

- **Request Metadata:**
  - Timestamp of the request (`requestTime`)
  - Device details (`deviceType`, `osVersion`)
  - API endpoint (`endpoint`)
- **Response Metadata:**
  - Timestamp of the response (`responseTime`)
  - Response status (`responseCode`, `responseDescription`)

Sample Log Entry:

```
json
Copy code
{
  "requestTime": "2024-12-15T10:00:00.000Z",
  "deviceType": "Android",
  "osVersion": "12",
  "endpoint": "/books",
  "responseTime": "2024-12-15T10:00:01.200Z",
  "responseCode": 200,
  "responseDescription": "Success"
}
```

---

5. Store Logs in a Database

Logs are stored in a MongoDB collection hosted on Atlas.
Connection String:
`mongodb+srv://<username>:<password>@cluster0.mongodb.net/?retryWrites=true&w=majority&appName=BookAPI`

Collection Name: `logs`

MongoDB is used for persistent storage of logs, ensuring data availability across server restarts. The `MongoLogger` class handles all database interactions, including insert and query operations.

---

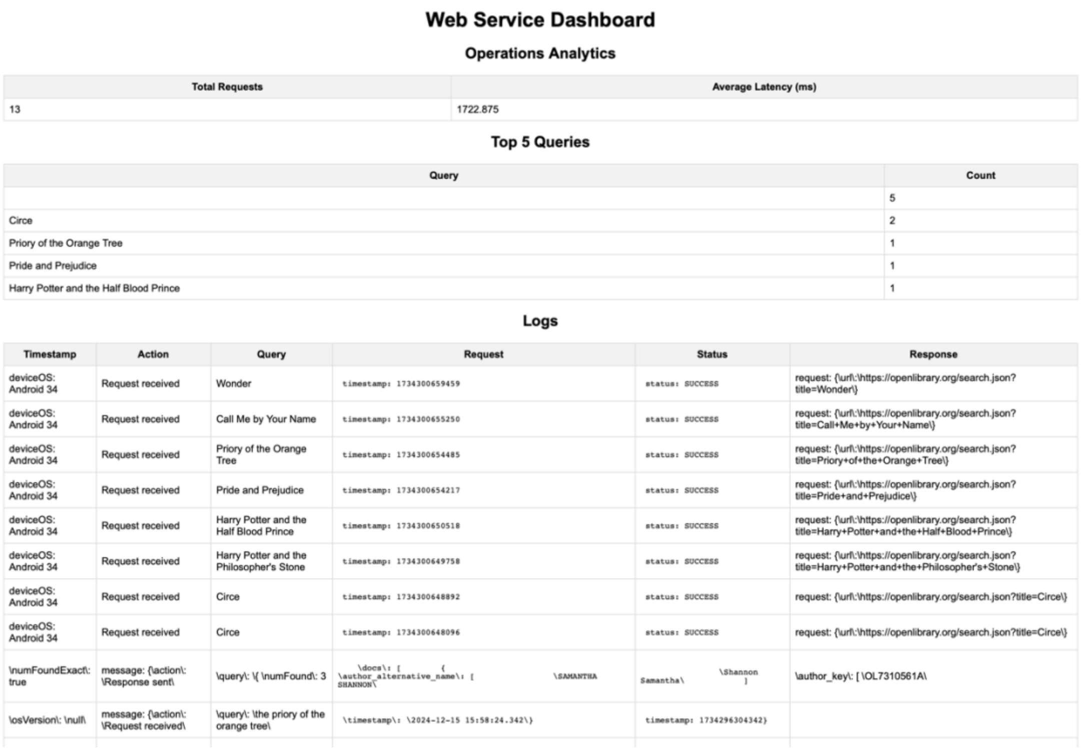
6. Display Analytics and Logs on Dashboard

Analytics:

1. **Top Search Queries:** Displays the most frequent book search terms submitted by users.
2. **Top Device Models:** Lists the most common devices accessing the service.
3. **Average API Latency:** Shows the average time taken for API responses.

Logs:

The dashboard also displays the latest log entries in a tabular format for readability and debugging purposes.



Web Service Dashboard

Operations Analytics

Total Requests	Average Latency (ms)
13	1722.875

Top 5 Queries

Query	Count
	5
Circe	2
Priory of the Orange Tree	1
Pride and Prejudice	1
Harry Potter and the Half Blood Prince	1

Logs

Timestamp	Action	Query	Request	Status	Response
deviceOS: Android 34	Request received	Wonder	`timestamp: 1734300659459`	`status: SUCCESS`	request: {\url\:\https://openlibrary.org/search.json?title=Wonder\}
deviceOS: Android 34	Request received	Call Me by Your Name	`timestamp: 1734300655250`	`status: SUCCESS`	request: {\url\:\https://openlibrary.org/search.json?title=Call+Me+by+Your+Name\}
deviceOS: Android 34	Request received	Priory of the Orange Tree	`timestamp: 1734300654485`	`status: SUCCESS`	request: {\url\:\https://openlibrary.org/search.json?title=Priory+of+the+Orange+Tree\}
deviceOS: Android 34	Request received	Pride and Prejudice	`timestamp: 1734300654217`	`status: SUCCESS`	request: {\url\:\https://openlibrary.org/search.json?title=Pride+and+Prejudice\}
deviceOS: Android 34	Request received	Harry Potter and the Half Blood Prince	`timestamp: 1734300650518`	`status: SUCCESS`	request: {\url\:\https://openlibrary.org/search.json?title=Harry+Potter+and+the+Half+Blood+Prince\}
deviceOS: Android 34	Request received	Harry Potter and the Philosopher's Stone	`timestamp: 1734300649758`	`status: SUCCESS`	request: {\url\:\https://openlibrary.org/search.json?title=Harry+Potter+and+the+Philosopher's+Stone\}
deviceOS: Android 34	Request received	Circe	`timestamp: 1734300648892`	`status: SUCCESS`	request: {\url\:\https://openlibrary.org/search.json?title=Circe\}
deviceOS: Android 34	Request received	Circe	`timestamp: 1734300648096`	`status: SUCCESS`	request: {\url\:\https://openlibrary.org/search.json?title=Circe\}
\numFoundExact\: true	message: {\action\: \Response sent\	\query\: \{ \numFound\: 3	`\docs\: [ { \author_alternative_name\: [ \SAMANTHA SHANNON\`	`Samantha\ \Shannon ]`	\author_key\: [\OL7310561A\
\osVersion\: \null\	message: {\action\: \Request received\	\query\: \the priory of the orange tree\	`\timestamp\: \2024-12-15 15:58:24.342\}`	`timestamp: 1734296304342}`	

---

7. Deployment to GitHub Codespaces

The web service is deployed to GitHub Codespaces with the following files:

- `.devcontainer.json`
  Defines the Codespaces environment, including necessary tools and configurations.
- `Dockerfile`
  Specifies the containerized deployment configuration for TomEE and MongoDB integration.
- `ROOT.war`
  The packaged web application.

Deployment Steps:

1. Push the `ROOT.war` to the repository.
2. Launch the Codespace and start the container using the provided Dockerfile.
3. Access the application through the Codespace URL.

References

1. OpenLibrary API documentation.
2. MongoDB Java Driver documentation.
3. Jakarta EE servlet and JSP references.
4. Online resources for GitHub Codespaces deployment.