CSE 546 - Cloud Computing

Project 2 - Bookstagram

Detailed Report

Group Members:

(1) Darshil Shah (1218490169 - dshah27@asu.edu)
(2) Mit Patel (1217114054 - mpatel42@asu.edu)
(3) Saumya Gangwar (1219377939 - sgangwa4@asu.edu)

1. Introduction

The habit of Reading greatly affects the overall development of our minds as well as our careers. Reading any book in a disciplined manner holds very much importance in our life. Because of such emphasis, about a year ago, a trend was started on Instagram to spread the awareness of reading among the youths. In order to be a member of such an initiative, anyone had to submit/donate a book to the charity and in return, they would get one book per month for a year absolutely free. Although it was a great idea, it was restricted to a small number of people/groups. Additionally, each book had to be delivered to a particular user to the given address via in-person delivery. That was very time consuming.

Based on this initiative, we thought to automate the whole process by providing a digital/scalable solution using the cloud services. We have created a web service which can bring digitalization into the above mentioned problem. Our idea was to have a user interface for users to upload their digital copies of books and in return, they would receive twelve different books of their preference. The copy of all these books would be sent to them via email every month for a year. Furthermore, if anyone does not want to upload their copies and still wants to receive book(s) from our database, they can purchase those books and get those books instantly sent to their email address.

2. Background

1. Importance of the Problem:

When we came to know about such an initiative on Instagram, we understood that it would be restricted to a small population of people. We thought that such a great idea should be reached to the maximum number of people to raise the awareness of the importance of reading. To deliver each and every book to its readers via in-person could be a very cumbersome task and it can slow the overall process of the chain. That's why we decided to overcome this problem with the digital solution so that the reach of such great initiatives could be increased to the maximum number of people and we can foster the reading habit in them.

2. Existing Solutions:

For this problem, there were no digital solutions before this. Although, there is this solution of delivering the physical copies of books to every user via in-person delivery but that is a very inefficient solution. Because to deliver each book to a particular user, there should be a delivery man available and if there are multiple requests coming in at the same time, many delivery men should be available otherwise deliveries could be delayed. So to mitigate such bottlenecks, we proposed this digital solution using the cloud services.

3. Technologies of Our Solution:

We created this web service using Java as our primary programming language. We have used JSP and Servlets technologies to create the frontend and backend, respectively. To deploy our service to the cloud infrastructure, we have used Google Cloud Platform (GCP) services. Specifically, we deployed our service to the Google App Engine (GAE) which is a service provided by GCP as a PaaS (Platform as a Service). Furthermore, to store all the digital copies of the books, we have used Google Cloud Storage which is an Object Based Storage Service. Then, to send the digital copy of a book to any particular user via email, we have used the Mail API which can automatically send the email at a scheduled time interval.

3. Design and Implementation:

3.1 Architecture:

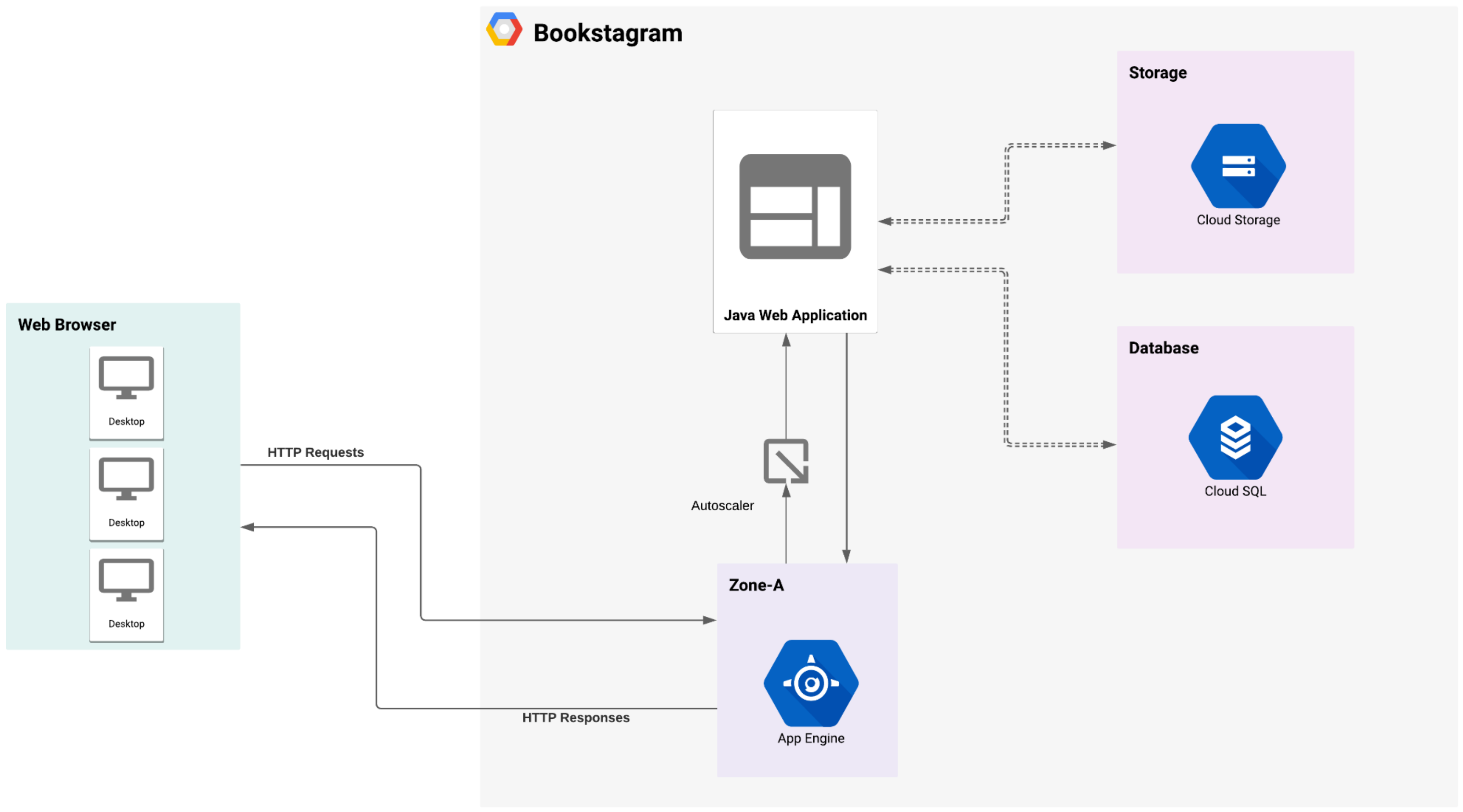


Figure:1 Architecture of book-bank as elastic system

3.1.1 GCP services used in project:

1. **Google Cloud App Engine:**
   Google App Engine (GAE) is a cloud computing platform as a service for developing and hosting web applications in Google-managed data centers. Applications are sandboxed and run across multiple servers. App Engine offers automatic scaling for web applications—as the number of requests increases for an application. We have deployed our java web application on App Engine which allows us to allocate on demand resources.

2. **Google Cloud SQL:**
   Cloud SQL (GCS) is a fully-managed database service that helps us set up, maintain, manage, and administer relational databases on Google Cloud Platform. This database service takes care of scalability and high availability of the database. In our application we have used a MySQL database provided by GCS to store user details.

3. **Google Cloud Storage:**
   Google Cloud Storage is a RESTful online file storage web service for storing and accessing data on Google Cloud Platform infrastructure. The service combines the performance and scalability of Google's cloud with advanced security and sharing capabilities. We have used Cloud Storage service for persistent storage. We have configured one bucket for storing pdfs uploaded by different users.

4. **Mail API:**
   The Mail API is a set of abstract APIs that model a mail system. The API provides a platform independent and protocol independent framework to build Java technology based email client applications.The Mail API provides facilities for reading and sending email.

3.1.2 Framework used:

We have used Java Server Page (JSP) for the front end and Java HTTP Servlet along with the maven tool for the backend. We also have configured a cron job using Quartz Scheduler to automate the process of sending email to users.

3.1.3 Architecture Explanation:

Our application has been deployed to the Google App Engine. So when a particular user hits the link of our deployed application, that HTTP request would be handled by the Google App Engine and would be sent to the Java application. The response would be sent back to the user where they could see the Login/Register functionality of our application.

Next, the user would be engaging with our application and provide their details of Login/Register. This data would then be sent back to the Cloud SQL and it would be stored there in the user’s database table. So, when the same user visits the next time, their credentials would be matched with the existing records and they would be authorized to the system.

Further, all the digital copies of books uploaded by different users would be stored in Cloud Storage. Basically, Cloud Storage is Object Based storage for the persistent use in the future. We can store all types of files in Cloud Storage. When a user clicks on the Available Books link, they would be able to see all the books available in the Cloud Storage. Also when a user uploads the digital copies of books, all the copies would be sent to the Cloud Storage to be stored on the permanent basis.

After that, Cron Jobs would handle all the users’ requests individually. It will send a book from a user’s preferred books list each month for one year. It would use the Mail API to send the email and for that, it would download the particular book from the Cloud Storage first.
This is the overall architecture of our system.

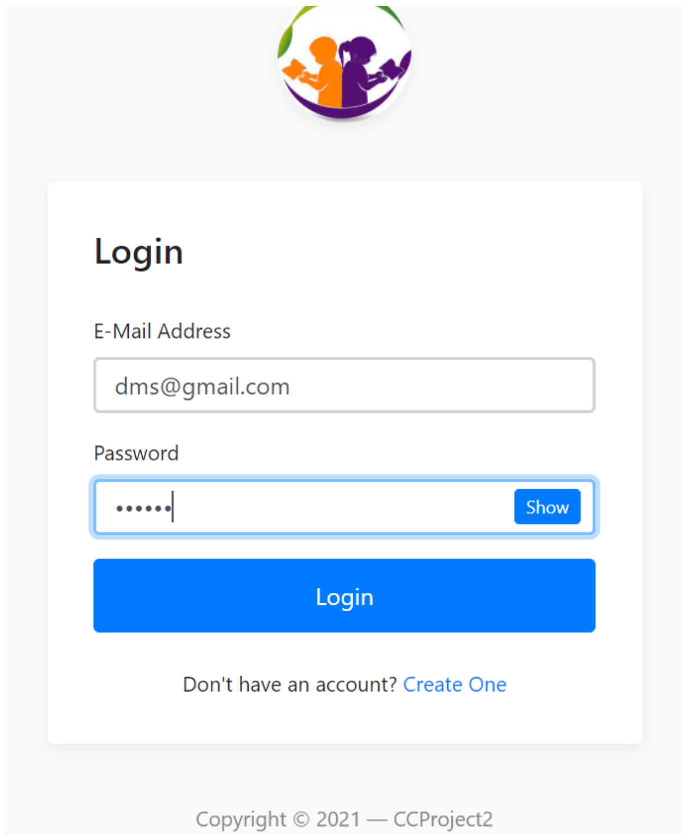
3.2 Autoscaling:

App Engine can automatically create and shut down instances as traffic fluctuates. Autoscaling creates instances based on request rate, response latencies, and other application metrics. Each instance in the app has its own queue for incoming requests. Before the queues become long enough to have a noticeable effect on your app's latency, App Engine automatically creates one or more new instances to handle the increasing load. When request volumes decrease, App Engine reduces the number of instances. This downward scaling helps ensure that all of your application's current instances are being used to optimal efficiency and cost effectiveness.

For our system, we considered some bottlenecks which could be: when multiple users are going to use the system at the same time. At that time, GAE will launch the new instances on the go as the traffic increases. The second would be when multiple users are going to upload many books to the system at the same time. This backend auto scaling would also be handled by the GAE itself. There were no current state-of-the-art web solutions. The solution which was there was manually delivering physical copies of the books to the user’s address. There was no software solution which could help remove that problem. So, our solution is going to be the better solution than the existing as it won’t be time consuming and very much effective.

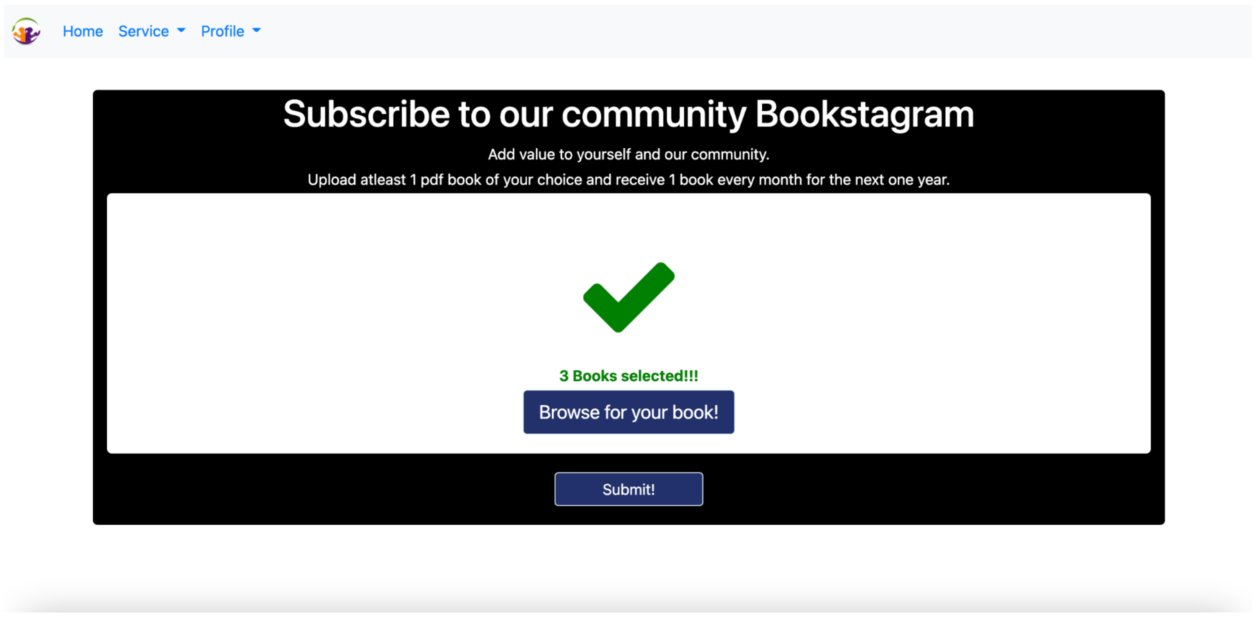
4. Testing and Evaluation

Step 1- Successful login /registration of users - To check this we registered multiple users manually and checked for successful registrations and login. Also we used Apache bench to test successful login of 100 concurrent users


Login
E-Mail Address
dms@gmail.com
Password
••••••
Show
Login
Don't have an account? Create One
Copyright © 2021 — CCProject2

.

Step2 - Subscription Upload Module - The users can upload any number of pdfs at once and contribute to our pool of books and subscribe to our service. To check this we did manual testing by uploading 300 books by 3 users simultaneously.


Home Service Profile

Subscribe to our community Bookstagram

Add value to yourself and our community.

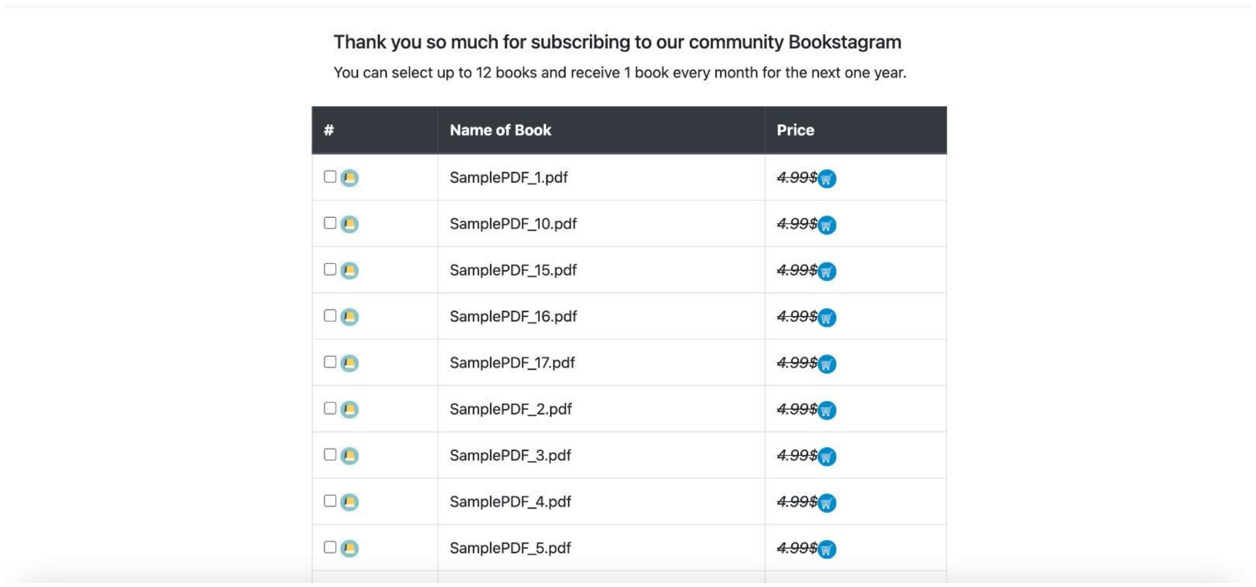
Upload atleast 1 pdf book of your choice and receive 1 book every month for the next one year.

3 Books selected!!!

Browse for your book!

Submit!

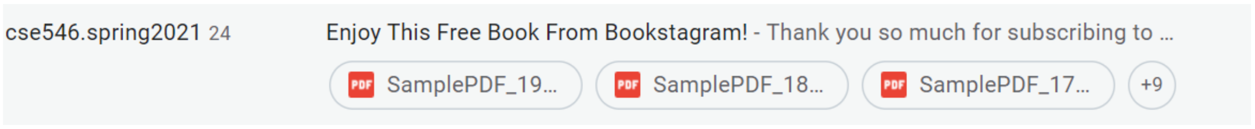
Step3- ViewAvailableBooks - Once the user has subscribed to our service he can view a list of all books available in our collection and select upto 12 books for free. The user can also purchase any number of books of his choice. Concurrent users can view , purchase and select books at same time


Thank you so much for subscribing to our community Bookstagram

You can select up to 12 books and receive 1 book every month for the next one year.

#	Name of Book	Price
☐	SamplePDF_1.pdf	~~4.99$~~
☐	SamplePDF_10.pdf	~~4.99$~~
☐	SamplePDF_15.pdf	~~4.99$~~
☐	SamplePDF_16.pdf	~~4.99$~~
☐	SamplePDF_17.pdf	~~4.99$~~
☐	SamplePDF_2.pdf	~~4.99$~~
☐	SamplePDF_3.pdf	~~4.99$~~
☐	SamplePDF_4.pdf	~~4.99$~~
☐	SamplePDF_5.pdf	~~4.99$~~

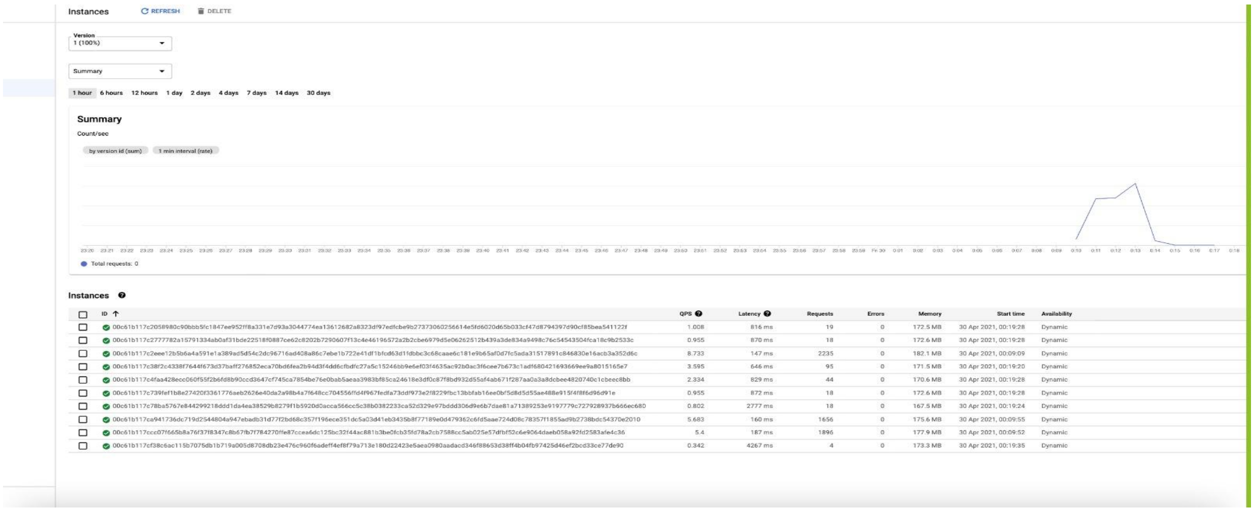
Step4- Mailing service - Concurrent users can subscribe to our service and they receive mails concurrently. We have done manual testing to validate the functionality.



cse546.spring2021 24 Enjoy This Free Book From Bookstagram! - Thank you so much for subscribing to ...

SamplePDF_19... SamplePDF_18... SamplePDF_17... +9

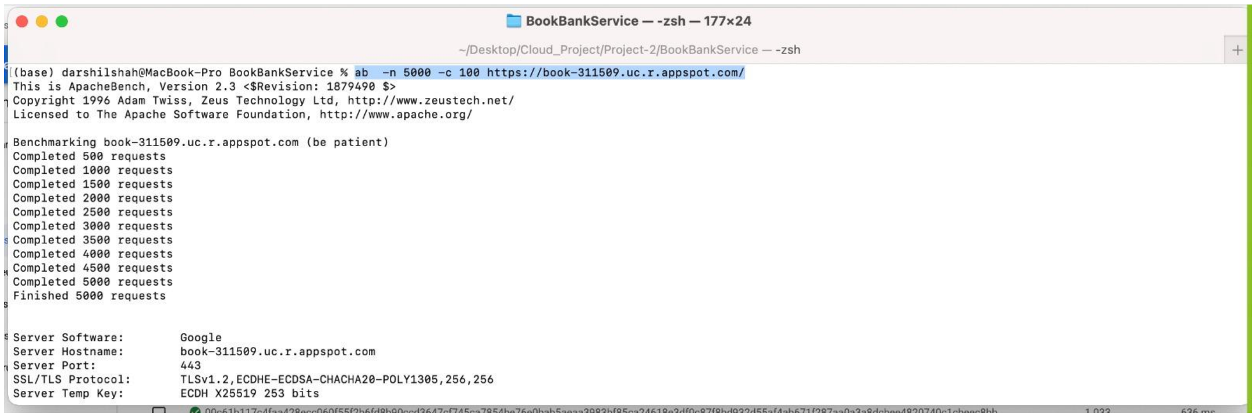
Apache Benchmark- We sent in 5000 requests by 100 concurrent users and tested for autoscaling.10 instances of GAE were spun up and once the requests were handled the instances were scaled down. We tested this through Apache Benchmark.



Instances

ID	QPS	Latency	Requests	Errors	Memory	Start time	Availability
00c61b117c2058980c90bbb5fc1847ee952ff8a331e7d93a3044774ea13612682a8323df97edfcbe9b27373060256614e5fd6020d65b033cf47d8794397d90cf85bea541122f	1.008	816 ms	19	0	172.5 MB	30 Apr 2021, 00:19:28	Dynamic
00c61b117c2777782a15791334ab0af31bde22518f0887ce62c8202b7290607f13c4e46196572a2b2cbe6979d5e06262512b439a3de834a9498c76c54543504fca18c9b2533c	0.955	870 ms	18	0	172.6 MB	30 Apr 2021, 00:19:28	Dynamic
00c61b117c2eee12b5b6a4a591e1a389ad5d54c2dc96716ad408a86c7ebe1b722e41df1bfcd63d1fdbbc3c68caae6c181e9b65af0d7fc5ada31517891c846830e16acb3a352d6c	8.733	147 ms	2235	0	182.1 MB	30 Apr 2021, 00:09:09	Dynamic
00c61b117c38f2c4338f7644f673d37baff276852eca70bd6fea2b94d3f4dd6cfbdfc27a5c15246bb9e6ef03f4635ac92b0ac3f6cee7b673c1adf680421693669ee9a8015165e7	3.595	646 ms	95	0	171.5 MB	30 Apr 2021, 00:19:20	Dynamic
00c61b117c4faa428ecc060f55f2b6fd8b90ccd3647cf745ca7854be76e0bab5aeaa3983bf85ca24618e3df0c87f8bd932d55af4ab671f287aa0a3a8dcbee4820740c1cbeec8bb	2.334	829 ms	44	0	170.6 MB	30 Apr 2021, 00:19:28	Dynamic
00c61b117c739fef1b8e27420f3361776aeb2626e40da2a98b4a7f648cc704556ffd4f967fedfa73ddf973e2f8229fbc13bbfab16ee0bf5d8d5d55ae488e915f4f8f6d96d91e	0.955	872 ms	18	0	172.6 MB	30 Apr 2021, 00:19:28	Dynamic
00c61b117c78ba5767e844299218ddd1da4ea38529b8279f1b5920d0acca566cc5c38b0382233ca52d329e97bddd306d9e6b7dae81a71389253e9197779c727928937b666ec680	0.802	2777 ms	18	0	167.5 MB	30 Apr 2021, 00:19:24	Dynamic
00c61b117ca941736dc719d2544804a947ebadb31d77f2bd68c357f196ece351dc5a03d41eb3435b8f77189e0d479362c6fd5aae724d08c78357f1855ad9b2738bdc54370e2010	5.683	160 ms	1656	0	175.6 MB	30 Apr 2021, 00:09:55	Dynamic
00c61b117ccc07f665b8a76f37f8347c8b67fb7f784270ffe87ccea6dc125bc32f44ac881b3be0fcb35fd78a2cb7588cc5ab025e57dfbf52c6e9064daeb058a92fd2583afe4c36	5.4	187 ms	1896	0	177.9 MB	30 Apr 2021, 00:09:52	Dynamic
00c61b117cf38c6ac115b7075db1b719a005d8708db23e476c960f6adeff4ef8f79a713e180d22423e5aea0980aadacd346f88653d38ff4b04fb97425d46ef2bcd33ce77de90	0.342	4267 ms	4	0	173.3 MB	30 Apr 2021, 00:19:35	Dynamic

Load handled by 10 instances



BookBankService — -zsh — 177×24
~/Desktop/Cloud_Project/Project-2/BookBankService — -zsh

```
(base) darshilshah@MacBook-Pro BookBankService % ab  -n 5000 -c 100 https://book-311509.uc.r.appspot.com/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking book-311509.uc.r.appspot.com (be patient)
Completed 500 requests
Completed 1000 requests
Completed 1500 requests
Completed 2000 requests
Completed 2500 requests
Completed 3000 requests
Completed 3500 requests
Completed 4000 requests
Completed 4500 requests
Completed 5000 requests
Finished 5000 requests


Server Software:        Google
Server Hostname:        book-311509.uc.r.appspot.com
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-ECDSA-CHACHA20-POLY1305,256,256
Server Temp Key:        ECDH X25519 253 bits
```

5000 concurrent requests by 100 user

5. Code

5.1 Code Modules :

1. **Front-end Module:**
   a. **UserLogin.jsp:** It provides a user login form consisting of user email and password fields.
   b. **UserRegistration.jsp:** It provides a user registration form consisting of username,email and password fields.
   c. **WelcomeHomePage.jsp:** This describes what services we are offering and the main essence of our cause.It contains an image catalogue and about us section.
   d. **UploadBookForm.jsp:** This provides an interface to the user to upload pdfs of books and join our subscription service.
   e. **ViewAvailableBooksSubscribed.jsp**: This provides an interface to subscribers to select any twelve books of their choice and receive them in their email at free of cost .
   f. **ViewAvailableBooksSubscribed.jsp:** If the users don't want to subscribe to our service they can still go through all the books available in our community and purchase any available book of their choice.This provides an interface to view available books and their costs for purchase.
   g. **PaymentGateway.jsp:** Provides an interface for payment gateway so that the user can purchase books of his/her choice.

2. **Back-end Module:**
   a. **Bean Module (UserBean.java):** It stores user-centric information like user-id, user-name and password and provides getter-setter methods for the access.
   b. **Constants Module (Configuration.java):** Stores configuration information like database driver name, database name, project-id, bucket-name, instance-connection string etc. which are required for other modules of the project.
   c. **CronJob Module (CronJob.java):** It executes sendEmail method of EmailUtil as a cron job.

d. **Db Module**:
    1. **ConnectionManager.java** file is responsible to provide a valid connection string.
    2. **LoginValidateDb.java** file validates user-id and password provided by different users.
    3. **RegistrationDb.java** file inserts user details into the database.

e. **Servlet Module:**
    1. **LoginServlet.java:** Validates user-id and password entered by users.
    2. **RegistrationServlet.java:** Insert user information into MySQL database provided by Cloud SQL.
    3. **UploadFileServlet.java:** Uploads pdfs on Google Cloud Storage bucket.
    4. **ViewBookListServlet.java:** Displays list of books from Google Cloud Storage bucket.
    5. **SendEmailServlet.java:** Responsible to send email with attachment to the user on successful subscription.
    6. **SendEmailPaymentServlet.java:** Responsible to send email with attachment to the user on successful payment
    7. **QuartzListener.java:** It implements ServletContextListner which triggers cron job when context of application initializes.

5.2 Instructions for Project Execution:

1. Setup Module:

a. Review and start your free trial by signup into Google Cloud Platform (GCP) and create a new project. Make sure to attach a billing account for the newly created project.

b. Create a service account from IAM & Admin dashboard and download json key file.

c. Download and install Google Cloud SDK into your system and verify it using the following command.
    i. *gcloud info*

d. Install gcloud Java extension using the following command.

        i. *gcloud components update*
        ii. *Install app-engine-java*

    e. Download the following jar files and add them into the build path and server's run configuration.
        i. *mysql-connector-java-8.0.23.jar*
        ii. *mysql-socket-factory-connector-j-8-1.2.1-jar-with-dependencies.jar*
        iii. *Json-key file*
    f. Enable following required Google APIs to provide access to the services
        i. *Cloud Storage API*
        ii. *Cloud SQL Admin API*
        iii. *App Engine Admin API*
    g. Deploy your application on App Engine using the following commands.
        i. gcloud init
        ii. gcloud config set PROJECT_ID
        iii. maven appengine:deploy

2. Execution Steps:

    a. First start all the services of GCP from the console.
    b. Hit the following url "*https://book-311509.uc.r.appspot.com*"

6. Conclusions

1. Accomplishments and Learnings:

We successfully created and deployed the proposed solution to the Google Cloud Platform. We were able to provide the basic Login/Register functionality to the user of our system. We also implemented the Payment functionalities for those users who don’t want to subscribe to the system and just want to have some books instantly sent to their email address. Furthermore, for the subscribed users, we implemented the mailing functionality which sends them the emails containing the digital copy of their preferred books in a regular interval of time.

Working on this project, we have learnt how we can use Google App Engine (GAE) which is basically, Platform as a Service (PaaS) of Google Cloud Platform. By using a PaaS, we don’t have to manage the infrastructure required by our application. Also, the scale out/in functionalities are handled by the GAE itself when our application faces a large traffic. We don’t have to code for that, though we can tune some parameters. Additionally, we have used Mail API to send the emails in a regular interval of time using Cron Jobs. By using that functionality, we got to know how we can stay in touch with our users.

2. Future Enhancements

There are some enhancements which could be done to our existing system in the future. We can partition all the books available in the database by their genres and give users functionalities to filter the books based on genres. Also, we can provide in-person delivery of physical books to the user’s address if they want to have a physical book. Further, we can provide lifetime service of sending books to the users instead of just 12 months by charging them a minimal amount.

7. Individual Contribution

Darshil Shah (1218490169)

Design:

I have collaborated with my teammates to design the basic architecture of an "Bookstagram" elastic application. After brainstorming various designs, we were able to identify various GCE components needed, how they will communicate with each other, what can be the role of App Engine(GAE), how to communicate with Cloud SQL and which information needs to be stored, how to upload and fetch pdfs from Cloud Storage and feed them to a dynamic table. I also contributed to the design of a scheduler which triggers a cron job in a particular interval of time.

Implementation:

The implementation phase consists of creating Java Web Application and integrating it with various GCP components. I have successfully implemented a User Registration/Login module using JSP and Servlets which validates user details from Cloud SQL. I also worked on the Upload module which provides a user interface to upload multiple documents to Cloud Storage and subscribe to our application. Moreover, I also implemented a ViewListBook module where users can select multiple books of their choice and get them through email. I have also configured Quartz Scheduler , which triggers a send-email cron job in every 2 seconds.

Testing:

I was extensively involved in the Testing phase as well. I have done unit testing of a whole application which includes the successful upload in Cloud Storage bucket and credibility of scheduler. As a part of integration testing, I tested the correctness of the load balancing by making 5000 concurrent requests using Apache Benchmark Tool. We did end to end testing by uploading more than 3000 pdfs and monitored the work done by each spawned app engine instance.

Saumya Gangwar (1219377939)

Design :

Since I am an avid reader , I came up with the idea of Bookstagram(elastic application). After the approval from my teammates we designed the entire application .We brainstormed various designs and decided on individual components and modules and how they will communicate with each other.We decided the flow of the web application as well as the technologies that would work best for our application.After going through various components of google cloud platform we decided on using GAE(Google App Engine),Google Cloud storage, Google Mail API and Google Cloud SQL services in our web application. We also decided on how we will integrate these selected services with our modules like uploading pdfs and sending mails through GoogleMAIL API.

Implementation:

I was responsible for implementing a scalable frontend in our web application using jsp,css,html, javascript and googleApp Engine.I implemented the individual modules of upload pdfs to our system(which involved uploading pdfs to google cloud storage) ,successful subscription to our service,proper authenticated login and registration to our system as well as making the application scalable using Google App Engine.

Testing:

I was involved in the thorough testing of our web application.We did unit testing, integration testing and end-to-end testing. I did end-to-end testing by uploading 300 pdfs by 3 concurrent users and checking the auto scaling for the same. We also used the tool Apache bench to test our application for a large number of requests. We tested it for 5000 concurrent requests using the Apache Bench tool.

8. References

1. https://www.vulture.com/2018/10/the-terrible-instagram-trend-of-piles-of-open-books.html
2. https://cloud.google.com/appengine/
3. https://cloud.google.com/sql/
4. https://cloud.google.com/products/storage/
5. https://www.instagram.com/explore/tags/bookstagram/
6. http://www.quartz-scheduler.org/
7. https://www.theuncorkedlibrarian.com/how-to-start-a-bookstagram/
8. https://developers.google.com/gmail/api