Program 5 - BookShelf

Website URL:
http://bookshelfprogram5css436-env.eba-stt3v5gz.us-west-2.elasticbeanstalk.com/welcome

6 services used for this project:
AWS S3
AWS DynamoDB
AWS SNS
AWS Elastic BeanStalk
AWS CloudWatch
REST API used: New York time API - https://developer.nytimes.com/apis

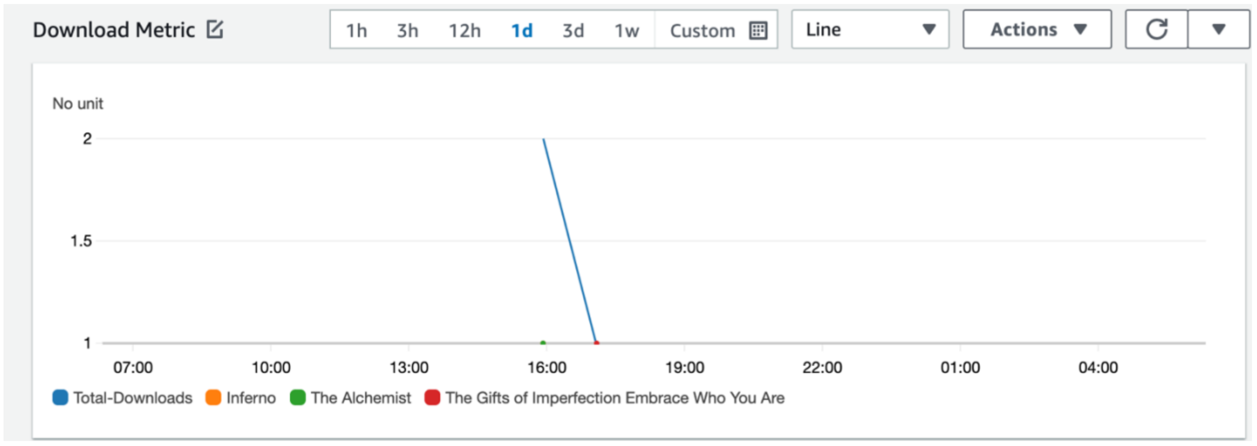
Application: BookShelf

BookShelf is a web application which stores books on the server. Users can upload and download books to/ from repository. All users can subscribe to receive notifications when new books are uploaded. Also, this application uses the REST API exposed by the New York Times, to list the information top 5 books of the month.

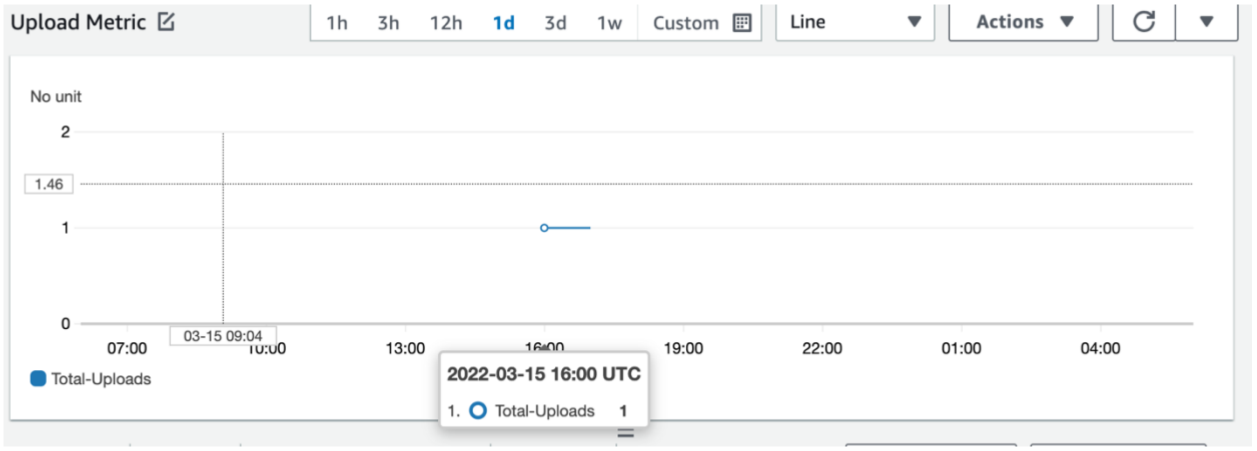
Details of services used in BookShelf application:

- **Download:** The books are stored in AWS S3 storage. When a user click enters the title of a book, application verifies if this book is available in S3 by querying the DynamoDB and downloads the copy of the book on user's PC from S3 storage.
- **Upload:** When user wants to contribute to the BookShelf repository, he can upload books. The user needs to enter the book title and the path of the book on his local storage. The application adds the book title, book file name to Dynamo DB and uploads this book to the AWS S3 storage.
- **Subscribe:** If the user wants to be notified of new books added to the repository, he can use subscribe option. When user enters a valid email ID, a mail is sent to his inbox, where he needs to confirm his subscription. Every time a book is uploaded an email is sent to all the subscribers.
- **Top Books Info:** The application gives information of top 5 educational books listed by New York Times using their Rest API. It lists the book title, author, and short description of the book.
- The application is hosted on the AW Elastic BeanStalk.
- Also, the AWS CloudWatch service is used to report the application metrics i.e., total number of downloads, number of downloads/book, total number of uploads, total number of subscribers. Through this information we can observe the books that are frequently downloaded, we can upload all books from that author, we can observe which users contributed most by uploading books.

Cloud watch application metrics:



Total downloads

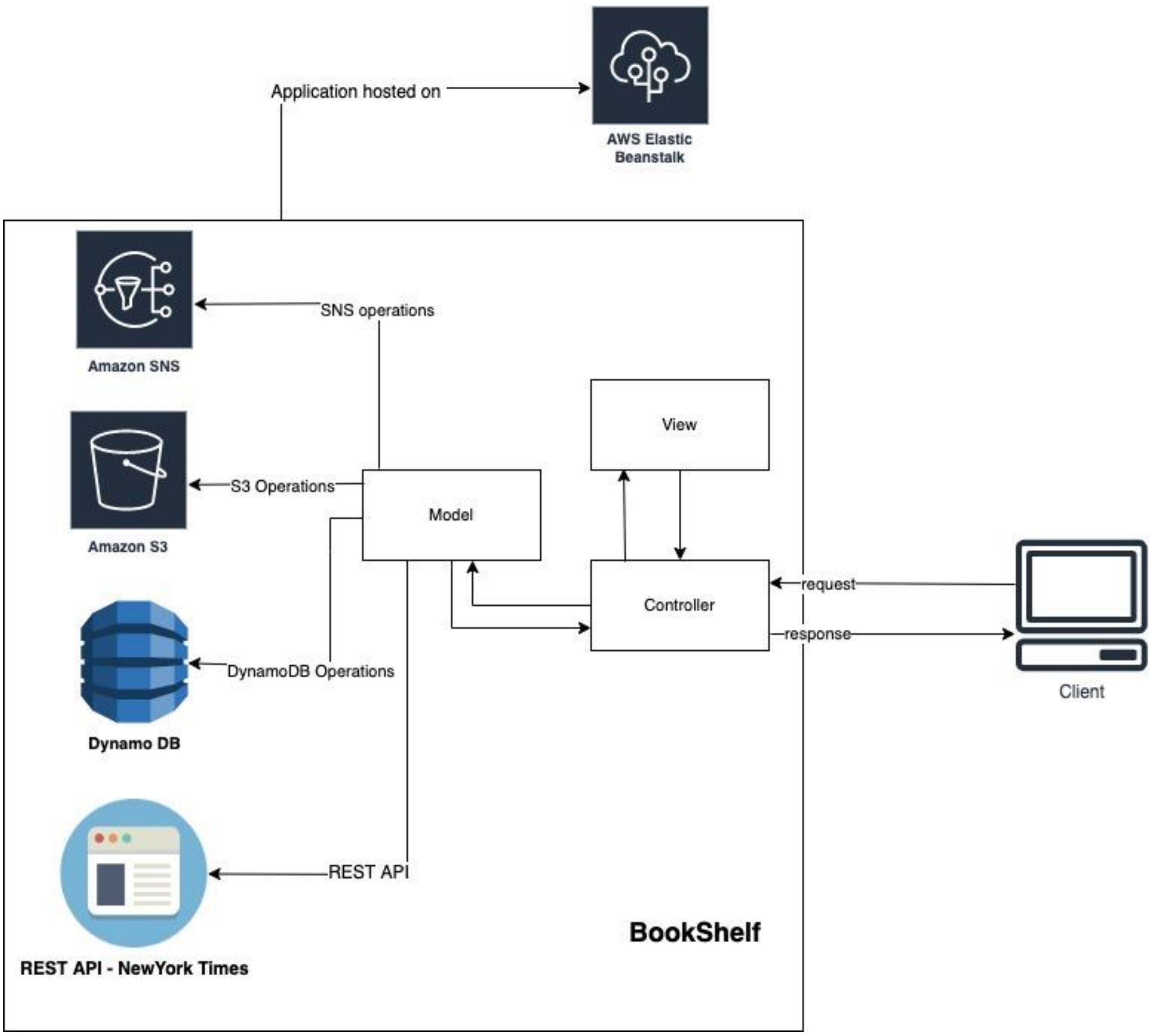


Total uploads

Design:

Specification:
Cloud Provider: AWS
Cloud Services: AWS S3, AWS Dynamo DB, AWS SNS, AWS CloudWatch, AWS Elastic Beanstalk
Framework used: Spring-Boot
Coding language for backend: Java
Coding language for frontend: JSP, JavaScript

Design diagram:



This web application follows MVC architectural pattern

Controller: The controller receives the request from client and directs it to the model. After the model sends in the processed data, controller requests the view for preferred presentation, the controller sends this view back to the client, which is displayed on client's browser.

Also, when the request is being processed, client/user should wait, the controller requests status messages e.g.: please wait or loading etc. from the view which intimate the user about ongoing process.
Controller classes: WelcomeController, DownloadController, UploadController, SubscribeController, RestServiceController

Each of these controller classes handles each of the functions of BookShelf application. The WelcomeController is the start point, which takes all the requests and directs it to the corresponding controllers.

- Download: The DownloadController request the model to check if book is available from the BOOK_DETAILS table in Dynamo DB. If the book exists, controller request the model to initiate a download to user's device.

- Upload: The UploadController requests the model to check if book is available from the BOOK_DETAILS table, if its available the model send appropriate message to view. If book does not exist, the model initiates upload process, copies the file from user's local to server and then transfers it to S3 storage.

- Subscribe: The SubscribeController requests the model to add the email ID to SNS service subscribers list. For a new user confirmation email is sent. On every new book upload an email is sent by SNS to all subscribers.

- Top Books Info: The RestServiceController request the model to fetch data from the REST API (New York times api). The model makes an HTTP GET call on the api to fetch the information of list of top 5 books. The controller sends this list to the view for display.

Model: The model interacts with all AWS services and REST service to perform appropriate functionalities. It sends the data back to controller.
Model classes: S3Operation, DynamoDBOperation, SNSOperation, CloudWatchOperation

S3Operation performs the download and upload operations i.e., it copies the books from or to the storage. DynamoDBOperation creates a table in DynamoDB to store book title and its corresponding file name (file name as stored in S3 storage). SNSOperation adds email IDs to SNS service. CloudWatchOperation publishes BookShelf application metrics (number of downloads/book, number of uploads, number of subscribers) to the cloud watch. These metrics give application information from business point of view.

View: The view takes the processed data from controller and renders it in HTML format to the user.
View files: welcome.jsp, download.jsp, upload.jsp, subscribe.jsp, topbooks.jsp
welcome.jsp displays the main start page. download.jsp is for the download operations with appropriate error messages. upload.jsp is for the upload operations with appropriate error

messages. subscribe.jsp is for the subscribe operations with appropriate error messages. topbooks.jsp is for the top books info.

Important aspects of BookShelf application:

Usability:
The BookShelf application provides appropriate error messages to users. In case of download, if the book is unavailable or if user hasn't entered any book title, it shows the appropriate error messages. In case of upload, if the user is trying to upload a book that exists or when user enters wrong file path, appropriate error messages are shown. In case of subscribe if the user enters invalid email id, error message is shown to the user.

Elasticity:
This application is hosted using aws Elastic Beanstalk which makes use of EC2 instances. EC2 supports auto scaling, it monitors the health of each Amazon EC2 instance that it launches. Using the elastic beanstalk console we can configure the instances, i.e., upscale, or downscale EC2 instances based on CPU utilizations, disk read/write operations, request count, network usage etc. The scaling can be done based on timing as well depending on when (what time of day) the usage of service is high. As the load on the application increases, we can increase the EC2 instances and add ELB (load balancer) to direct the requests between the instances to avoid overloading. Auto scaling configuration enabled for this application (BookShelf), it upscales when CPU utilization crosses 60% and down scales when CPU utilization falls below 20%.

In case of Dynamo DB, we can enable auto scaling using the AWS auto scaling service which dynamically enables the provisioned throughput capacity based on increased/decreased traffic patterns. Auto scaling uses Amazon CloudWatch to monitor a table's read and write capacity metrics. To do so, it creates CloudWatch alarms that track consumed capacity. For this application I have not enabled auto scale due to charges applicability.

Applications can easily achieve thousands of transactions per second in request performance when uploading and retrieving storage from Amazon S3. Amazon S3 automatically scales to high request rates up to 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per object in a bucket. Hence BookShelf application supports autoscaling handling any number of COPY/POST/GET requests. (Charges will apply on crossing free tier range)

Monitoring and Alerts:
CloudWatch service offered by AWS monitors all the services being used.
We can also add alarm notification for monitoring, where we can receive email notifications when configured thresholds are reached. In case of this application, I have configured alarm email notification if more than 10 5XX requests are received in span of 15minutes.

Why AWS?
For this project AWS is used due to its ease of usage of AWS services, and documentation to use SDK is very clear with examples. Easy user interface for auto scaling and cloud watch.

<u>SLA of BookShelf application:</u>

This web application's uptime depends on following services

1. Elastic bean stalk:
   Here the application is hosted on EC2 and uses ELB, hence taking SLA of EC2 instance and ELB which is 99.99%. Since load balancer and auto scaling is used there is no downtime of web application (for maintenance or upgradation), during upgradation or maintenance the ELB takes care of running the application on another instance.
2. Dynamo DB:
   AWS offers 99.99% SLA
3. S3 storage for uploading the input.txt file:
   AWS offers 99.9% SLA for S3 Standard storage
4. Service uptime for the REST API used to fetch top 5 books information:
   This depends on the uptime of New York Times REST API service. Exponential backoff is added to handle the 5XX requests. Thus, the uptime of this service can be taken as 100%

Thus, total SLA = (0.9999*0.9999*0.999*1) = 99.88%