

Digitalizing Shopping Mall: A Web Application for Enhanced Shopping Experience Leveraging AWS Services including EBS, Code Pipeline, RDS.

Sandesh Muralidhar
StudentId: 20195737

Cloud Platform Programming, MSc in Cloud Computing
National College of Ireland Dublin, IRELAND
Email: x201957374@student.ncirl.ie. URL: www.ncirl.ie

Deployed Application URL: <http://x20195737shoppingmall-env-1.eba-6k3evacx.ap-southeast-1.elasticbeanstalk.com>

Abstract—The aim of this project is to simplify the shopping experience for consumers by providing a digital solution to help them locate stores within a shopping mall by leveraging AWS cloud services. It is common for shoppers to spend a lot of time roaming inside mall, trying to locate stores that they are interested in. This can be a frustrating experience, particularly if they are not familiar with the layout of the mall. To address this problem, I have designed a web application that allows users to search for stores within a mall and view the floor on which they are located. The application also provides additional information about the store, such as its opening hours and a description of the products sold, the contact information of the store and its website. This makes it easier for shoppers to plan their visit and find the stores they are interested most importantly saving them time. The application also provides the shop owners to advertise offers if there are any going on in their store. In addition, the web application includes a feature that allows mall owners to display advertisements for stores that are on sale or offering discounts through a library that is integrated into the application. The web application has been implemented using AWS Code Pipeline, which enables to efficiently develop, test and deploy the code. The application code has been written using Python Django framework on Cloud9 IDE and GitHub as the repository, Gitwebhooks triggers the pipeline to build and deploy the application. The application is hosted on Elastic Beanstalk, which provides a scalable and reliable platform for running web applications. I have used RDS Postgres database to store data and images. The database stores all the store information of the application, which makes them easily accessible to the users of the application.

Index Terms—Amazon Web Service(AWS), Elastic Beanstalk(EBS), Continuous Integration and continuous deployment(CI/CD), Relational Database(RDS), Software development kit(SDK), Virtual private cloud(VPC)

I. INTRODUCTION

Cloud computing is a technology which is rapidly expanding and has transformed IT environment by providing many benefits such as cost savings, accessibility, elasticity, scalability, and performance and utilization monitoring. AWS is the most popular cloud provider and owns the majority of the share in the market as it provides wide

range of services which meet the customer requirements [1]. In particular, Platform as a Service (PaaS) has emerged as a popular service model that allows developers to focus on creating applications without worrying about the underlying infrastructure. Platform-as-a-Service (PaaS) is a sort of cloud computing that has the potential to assist enterprise developers in quickly developing and testing online applications aimed at customers or employees [2]. One such PaaS service is Elastic Beanstalk (EBS) from Amazon Web Services (AWS), which provides a platform for deploying and managing web applications. EBS takes care of the infrastructure and configuration, allowing developers to focus on writing code and building their applications. With EBS, developers can quickly deploy and scale their applications, and take advantage of various AWS services. Cloud services have revolutionized application development and programming by providing a flexible and scalable environment. This has resulted in faster application development, increased productivity, and reduced costs for businesses.

Various programming languages are used in cloud computing, including Java, C++, PHP, Ruby, and Python. Python is a popular choice among developers due to its ease of use, simplicity, and versatility. In particular, the Django framework, which is built using Python, is a popular choice for developing web applications in the cloud. Django is an open-source web framework that provides a high-level, robust, and secure environment for web development. It also offers a range of tools and libraries that make it easier to build scalable and efficient web applications. With Python and Django, developers can leverage the benefits of cloud computing to build and deploy applications quickly and efficiently [3].

Cloud computing features, as well as popular programming languages like Python and web frameworks Django, can be deployed to a wide range of applications. For instance, my web application tries to resolve a frequent problem encountered by shoppers: the difficulty in discovering their required shops in a shopping mall. My program, which makes use of the flexibility and scalability of cloud computing, allows users

to search for stores by name or category and gives real-time information about their timings and products in the shop. This was accomplished with the Django framework, which provided a high-level and safe environment for the application's development. I have used AWS Elastic Beanstalk for deployment, which can scale to meet the growing demands of users.

II. PROJECT SPECIFICATION AND REQUIREMENTS

Functional requirements describe what the system should do and while non-functional requirements dictate on how these functional requirements are implemented. Non-functional requirements also play a crucial role in software development but It receives less attention as compared to functional requirements. [4] Functional requirements are the features and capabilities that the software must provide in order to meet the needs of the users. In context to my application Functional requirements should include the search function which gives the ability to see related stores for a given search query so that It will enhance user shopping options, It should also have a display of number of shops and floors in the mall for which a counter in the home page which is required to display the number of floors in the mall and the total number of shops present in the mall. There are other functional requirements in my application where the store owner should be able to add their store, delete their store and add product descriptions which they sell in their store and also update the status of the shop whether it is closed or open. The other important functional requirement of the shopping mall application is to showcase of related stores. Suppose a user clicks on a food store then the interface would need to display other food-related shops so that the user can broaden their options.

Despite being given very less importance, non-functional requirements are equally important for the overall success of a project. For instance, if a system has excellent functionality but if it takes too long for a response then the whole application will go meaningless. To meet the user's needs Non-functional requirements of the application should include the design of a user interface that is easy to navigate, Faster response time to search queries, Login functionality requirements for shop owners, Secure authentication, and storage of information. The other important nonfunctional requirement is that the application should be compatible with multiple platforms and devices. Nonfunctional requirements also should include performance, scalability, reliability, and usage of the system. The application should be highly available, easily up-gradable, and should be able to handle multiple requests.

III. ARCHITECTURAL DESIGN

FIn the architecture of application all the mentioned functional and non functional requirement has been taken into consideration for application Digitalizing Shopping Malls. fig 1 shows architecture design of the application and all the cloud services which was used for it's implementation

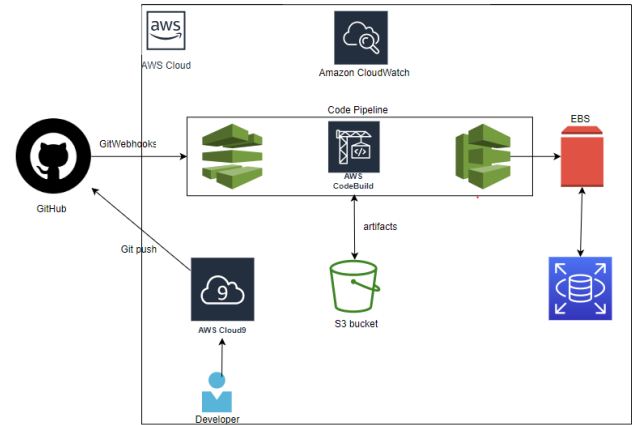


Fig. 1. architecture diagram digital shopping mall

The application has been developed on Cloud9 service. It is a cloud based Integrated development environment. Developer can directly start programming with the code without the requirement of installing any tools or hardware. All that a developer would need is a browser with an Internet connection for it. For versioning I have used github which is a central platform where developers can store the code, Track the changes and collaborate with other members. Github has a feature called Gitwebhooks which allows for autmated deployments which gets triggered when there is any change in the Github repository. When a developer pushes the code to github from cloud9. Gitwebhooks will send an post call to the endpoint of the pipeline which triggers the pipeline. Code pipeline includes 3 stages : source, build and deploy. The source stage retrieves the code from repository and the build stage is dictated by the yaml file, The build stage compiles the code to an executable format, In the last stage the codedeploy will deploy the application on Elastic Beanstalk (EBS). EBS is a an Fully managed resource of AWS cloud. Application deployment, scalability and monitoring are taken care by EBS, EBS also offers load balancing which can assist managae larger application load. Relational database(Postgres) is connected to EBS to store the data of the application, this includes the shop owner's data, description of the shops and other important data. Postgres is a high availability RDS which provides high security. RDS also mainly helps in management of data allowing developer to focus more on application. Cloudwatch has been primarily used in my architecture for capture and display of build logs in real-time. This allows us to acurately troubleshoot the issue as we can see the build stage logs using cloudwatch in realtime. It will give the entry according to the timestamp which would help us to troubleshoot the issue effectively.

Programming language for the application is python with django as framework. Django follows Model-View-Template (MVT) architecture where model represent the data and logic of application. The view handles the user interface and

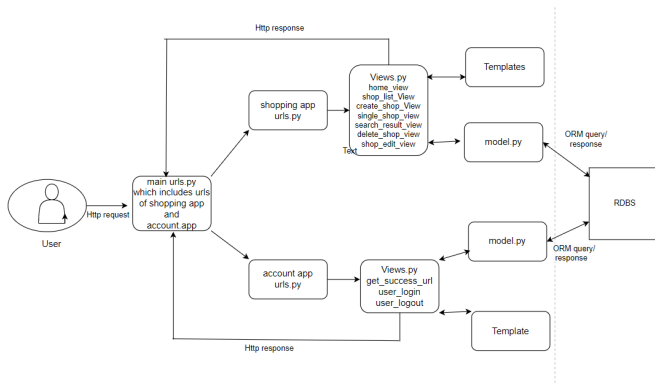


Fig. 2. Fig.2. MVT architecture of application

interacts with model to retrieve the data from database. The Template creates the final visual representation of a web page using data provided by the View. Fig 2 represents the MVT flow diagram of the application.

User sends in http request where it goes to main urls.py which includes the app urls.py. The url is redirected to app url.py file which points to corresponding view function. the view function renders the required html and contacts the RDBS through models.py file using object-relational mapping(ORM) query. Once the database returns a response to the view.py function call. It can use the response to generate the response to HTTP request.

Digital shopping mall has the following functionality for the customer. When a customer access the URL of the webpage. Customers would be able to see all the shops present in the shopping mall. In the home page there is a counter which displays how many total shops are present in the mall, the Total number of floors in the mall. When a customer clicks on an store for example a food-related store then It will display the shop description whatever has been updated by the owner and also it's status If the shop is open, closed or coming soon. Customers can also search for the shops floor-wise or search for a specific shop in the search bar. The mall home also shows the complete shop details of the shops present in the mall. Home page of the website along with the counter is shown in Fig 3.

The store manager is added by the mall administrator from the settings page of the website. Once the shop owner is added he will get the option to add a store. Once he clicks on add a store he can enter the store details, Avg spend and upload the image of the shop and it's location. Once the details have been submitted shop owner would need to wait until the mall administrator approves as shown in fig 4.

Once the shop is approved It will be displayed in the main homepage and will be added to the shopping mall as a listed store. Shop owner can then edit the shop details, Shop status whenever required so that customer will be updated with the current shop status and what is in store for them. they can also



Fig. 3. Fig.3. Home page of the application

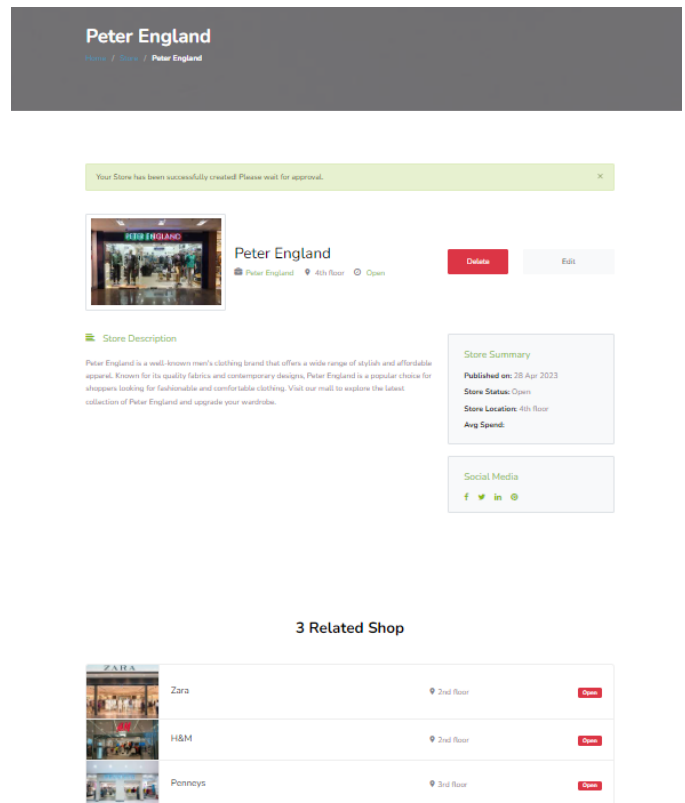


Fig. 4. Fig.4. awaiting approval from mall administrator



Fig. 5. Fig.5.pip install onsale

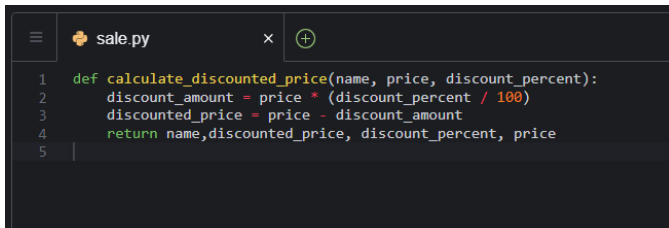


Fig. 6. Fig.6.sale.py library function

update their website and contact information in the description.

IV. LIBRARY DESCRIPTION

In a shopping mall not every shops would be occupied for instance there can be a shop that has never been occupied due it's corner location or there can be outlets which are moving out of mall. I have developed a library which can be used to sell such shops named 'onsale'. The same has been shown in Fig 5.

Shopping mall administrator would need to send shop details like name, It's current value, It's discounted percentage for which he is offering to sell. Library function calculates the discounted price and the same can be displayed on the website. Fig 6 shows the library function

To demonstrate I have passed the value as jd sports with it's current value as 50,000 and will be offered at a discount of 50 percentage. The same has been shown in Fig

is a browser-based IDE which makes the SDK(Software development kit) functionalities into the platform. It is simple to use as there is no pre-requisites or any installation setup. Cloud9 can be run on a browser with a internet connection and It can be accessed anywhere without any issue. Cloud 9 supports many features like code highlighting, code debugging which gives an extra edge while programming. The environment provided by Cloud9 has the ability to automatically stop and start all of the resources used and easy to manage. I used cloud 9 services in developing application because of it's additional features which I feel gives an edge over other programming IDE's. I also can easily install required version of any dependencies required for me in cloud9 without any hazel. [5]

AWS Code Pipeline is a service that is used to automate the code release process, It has following stages: development, building, staging, and finally production, also known as deployment. The code pipeline service will reduce the time and effort needed from the code developer. The pipeline will take care of the reminding task while the developer updates the code in its repository. The pipeline is going to carry out the appropriate building, testing, and deployment for any changes made to the code by the cloud9. I have used AWS codepipeline because of it's functionality of structured deployment and to keep track of all the activities through logging. [6]

Amazon Relational Database Service (RDS) provides a highly available relational database in cloud. It is easily scalable and can handle multiple load without any issue. RDS also provides multiple security features which keeps the data in it protected. It also manages authentication in an simple and secure way. It comes with on-demand instances where we pay for the compute capacity per hour. Amazon RDS manages complex and time-consuming administrative tasks such as PostgreSQL software installation and storage management. I prefered Amazon RDS as it takes only few steps in AWS management to launch and connect to production-ready postgresql database. It is also suitable for me as it provides fast and cost effective storage functionality. [7]

Elastic beanstalk refers to the beanstalk which grew all the way up to the clouds. Amazon Elastic beanstalk is a Platform as a service(PaaS) offering from AWS. AWS Elastic Beanstalk can quickly deploy and manage application in AWS cloud as it take cares about the Infrastructure which is required for the application. AWS handles the capacity provisioning, Load balancing, scaling and health monitoring by Itself. It is like an orchestration service offered by AWS for deploying application including EC2, S3, Cloudwatch, Elastic Load balancer. It supports multiple platforms and are priced according to resources used. I choose Elastic beanstalk for deployment because Elastic Beanstalk automates the deployment and management of the application. It is very flexible and easy to use making it ideal choice for my application [8]

Amazon Cloudwatch is a monitoring and management service which provides data and insights for all the resources in cloud. It provides Information about performance and operational data. Cloudwatch also enables the use of alarms, and events data. It's valuable insights enables us to take automated actions efficiently there by reducing the mean time to resolution (MTTR). Cloudwatch gives insights on how to optimize application performance, manage resource utilization and monitor the environmental health. CloudWatch collects, aggregates, and summarizes compute utilization information such as CPU, memory, disk, and network data. I have used AWS cloudwatch to mainly check on the build stage to gather each and every information of the commands being run through yaml file and then I have set the alarm to keep the CPU utilization in check of the RDS database. [9]

VIII. IMPLEMENTATION

The end goal of the project was to develop a user-friendly application that helps customers navigate a shopping mall by providing them with real-time Information on the location and related stores for the item they want to purchase. The application should also feature a search where users should easily be able to locate the store and also see shops present in each floor. To accomplish the same the code was first written in vs code using python = 3.7.16, Django version = 2.0.2

Django follows Model-View-Template architecture. I defined the urls.py file which has various routes that the application will respond to. This file is responsible for routing the request to the appropriate view function. Functions were written on the views.py file which contains the logic for handling HTTP requests and returning the required HTTP response. It collects the required data from models.py file and then renders it to HTML template. For each page/section HTML pages were written. This will be rendered by views.py file. models.py file was defined with data models for the application so that It will communicate to the RDS database connected to the application through object relation mapping and get the required details. Once all the functional and non-functional requirements were fulfilled and tested locally.

The code was cloned to Cloud 9 IDE where the first virtual environment was created and activated, Then all the requirements in requirements.txt is installed. Then RDS database was created and the Inbound and outbound rule was added to the database. The endpoint, credentials and ports were defined in the settings.py file. The same has been shown in Fig 8

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'shopping-db',
        'HOST': 'database-2.ccvpj32idf.ap-southeast-1.rds.amazonaws.com',
        'PORT': '5432',
        'OPTIONS': {
            'sslmode': 'require',
            'options': '-c timezone=UTC',
        },
    },
}
```

Fig 8. settings.py file pointing to RDS database

once the settings.py file is saved the make migrations and migrate command needs to be run for the application to create the required tables in RDS database. This makes sure that database is in synch with models defined in the model.py file. I also added an alarm functionality using cloudwatch for the RDS database where If the CPU utilization of the RDS database reaches 20 percent it gives me a notification email so that I can have a look immediately and optimize the DB. The same alarm has been shown in Fig 9

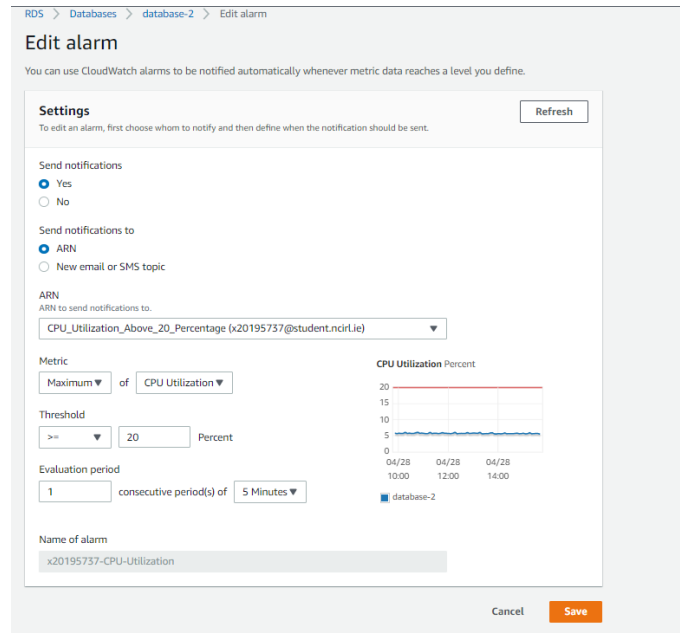


Fig 9. RDS CPU utilization using cloudwatch

Then collect static command was run to collect the static file (CSS, JS, images, etc.) and store them in the static directory. Then superuser was created to login to the administrator page of Django. Code pipeline with 3 stages namely source, build and deploy was created and then source was connected to the Github repository of the code .yaml file was also written accordingly to dictate the build stage and then for deployment EBS was selected. wsgi path for the application was added to .ebextensions/django.config file. Elastic Beanstalk uses the .config file to install the necessary packages inside the Elastic Beanstalk environment. Now a git push is performed from cloud9 environment to the Github repository which will be recognized the gitwebhooks and then send the post call to the endpoint of the pipeline which triggers the pipeline. Once all the 3 stages turn to green then the deployment is successful. But when I clicked on the URL of the EBS I was able to see 502 Gateway error I then figured out that the wsgi path was wrongly set which I corrected it and then pushed the code again to git hub which started the pipeline and now finally the deployment was successful with working application [8].

Below is my Environment details

CLOUD9 environment Region

Name - x20195737-shopping-mall-cpp
Region - Singapore

RDS database Region Information

Name - database-2
Region - Singapore

Code pipeline Region

Name - X20195737-shoppingmall-v3

Region - London

Elastic Beanstalk Region Information

Name - X20195737shoppingmall-env-1

Region - Singapore

IX. CONTINUOUS INTEGRATION, DELIVERY AND DEPLOYMENT OF YOUR APPLICATION

Continuous integration and continuous delivery (CI/CD) is an agile methodology which involves frequent code integration, Testing and deployment. It is often used with microservice architectures to automate the complete deployment process so that the new fixes or patches is production ready and should not cause any disruption. CI/CD is an efficient process to scale with minimal disruption. [10] In Digital shopping mall library I have Implemented CI/CD pipeline of 3 stages source, Build and Deploy stage. The same implementation is shown in Fig 10.

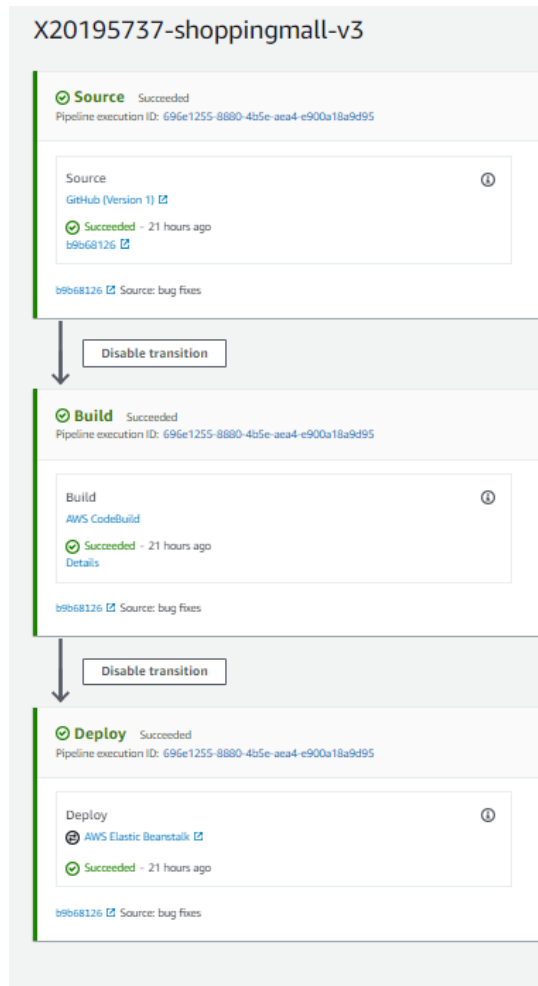


Fig 10. cpp-code pipeline

Whenever there is any changes in the git hub repository Gitwebhooks will send a post call to the pipeline end point triggering the code pipeline. Source stage generates the source artifacts and passes them to the build stage, The build stage

is dictated by yaml file where all the relevant commands are passed for the build stage to be success full and to generate build artifacts . Here For the code pipeline S3 is used as code repository to store and retrieve the artifacts. Finally the deploy stage will deploy the application in Elastic Beanstalk environment.

X. CONCLUSIONS AND FUTURE WORK

Developing the Digital shopping mall and deploying it using AWS cloud services enhanced my knowledge on various aspects of software development and deployment. Firstly, I gained a deeper understanding of the Django framework and the Model-View-Template (MVT) architecture. I was able to develop a user-friendly application which can be used in a real world scenario to solve a problem. I also gained immense knowledge on Relational Database and how the database configuration and schema works. Furthermore I learnt configuration of VPC, security groups, Inbound and outbound rule. I also got to know about errors related to deployment in EBS as I faced multiple issues during deployment which made me deeply analyze the logs and understand entries at Info level. This application has a broad scope for the future works as we can Integrate with third party service like google maps and Social media page, Customer should also given options to purchase the products and give out the review for the shops. If implimented in a real world scenario It would be exciting to Integrate IOT device to get the footprint of the mall so that this data can be used to get valuable insights.Overall, developing and deploying the Digital shopping mall on AWS Cloud Services has made me more equipped for future projects.

GitHub URL:

<https://github.com/SandeshDM6/shopping-mall-v3.git>

REFERENCES

- [1] A. Alalawi, A. Mohsin, and A. Jassim, "A survey for aws cloud development tools and services," in *3rd Smart Cities Symposium (SCS 2020)*, vol. 2020, 2020, pp. 17–23.
- [2] Z. Shu-Qing and X. Jie-Bin, "The improvement of paas platform," in *2010 First International Conference on Networking and Distributed Computing*, 2010, pp. 156–159.
- [3] I. A. Bairagi, A. Sharma, B. K. Rana, and A. Singh, "Uno: A web application using django," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2021, pp. 1371–1374.
- [4] T. Wang, P. Liang, and M. Lu, "What aspects do non-functional requirements in app user reviews describe? an exploratory and comparative study," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, 2018, pp. 494–503.
- [5] AWS, "Aws cloud9." [Online]. Available: <https://aws.amazon.com/cloud9/>
- [6] D. Clinton and B. Piper, *The Operational Excellence Pillar*, 2021, pp. 353–384.
- [7] AWS, "Amazon rds." [Online]. Available: <https://aws.amazon.com/rds/>
- [8] —, "Aws elastic beanstalk features." [Online]. Available: <https://aws.amazon.com/elasticbeanstalk/details/>
- [9] —, "Amazon cloudwatch." [Online]. Available: <https://aws.amazon.com/cloudwatch/>
- [10] C. Singh, N. S. Gaba, M. Kaur, and B. Kaur, "Comparison of different ci/cd tools integrated with cloud platform," in *2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 2019, pp. 7–12.