National College of Ireland

National College of Ireland

Project Submission Sheet

Student Name:	Prajwal Seethur Raveendra ……
Student ID:	……22228811……
Programme:	……Msc in Cloud Computing…… **Year:** ……2024……
Module:	……Scalable Cloud Programming……
Lecturer:	……Vikas Sahni……
Submission Due Date:	……19/04/2024……
Project Title:	……ExploreEase……
Word Count:	……3630……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: ……Prajwal Seethur Raveendra……

Date: ……19/04/2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only

ExploreEase

Prajwal Seethur Raveendra
StudentId: 22228811
Cloud Scalable Cloud Programming, MSc in Cloud Computing
National College of Ireland Dublin, IRELAND
Email: x23987654@student.ncirl.ie. URL: www.ncirl.ie
Deployed Application URL: http://x22228811-env.eba-iq32mbha.us-east-1.elasticbeanstalk.com/

***Abstract*—ExploreEase is a travel agency website that is designed to help travellers reach their desired destination in ease. The project aims to provide user-friendly interface to help the users navigate around the website easily. It allows users to explore various destinations based on continents and also made the booking process much simpler. The motivation of this project is to provide a seamless and easy-to-use solution for travellers to plan their trip conveniently. The website has built in currency converter, real time news and country information lookup which makes it unique from other travel agency websites. The goal of this project is to provide travels with a smooth and effective way to effortlessly arrange their travels at the same time inspire more people to travel around the globe. To improve the functionality of the website, ExploreEase's implementation entails the integration of many APIs, such as a news API, a currency converter API, and an API for global information. The front-end client application was developed in HTML,CSS,JavaScript and Flask was used as the back-end web framework. AWS Elastic Beanstalk was used to host the API and the client application to make them scalable. The software development life cycle of the client application was automated by using AWS Code Pipeline to create a CI/CD pipeline and Sonar cloud was also integrated into the pipeline for static code analysis.**

***Index Terms*—API, HTML,CSS,JavaScript,Flask, Elastic Beanstalk, AWS Code Pipeline, CI/CD pipeline**

I. Introduction

Travelling is a hobby that almost everybody in the world loves to do. People will have plenty of beautiful tourist spots to visit in their bucket list. Walking in the streets of London and embracing the mesmerizing beauty of the Eiffel Tower is something that should be experienced by everyone. In order to get people to their dream lands, we need a platform that will make things easier for people to book tickets and enjoy their trips peacefully. ExploreEase is an online platform that will help users to explore a wide verity of tourist spots and make the bookings easier. The main motivation for the development of this project is to provide a user-friendly platform for travellers and help them reach their desired destination. Users may become confused and find it difficult to quickly locate the information or services they need on websites with complicated navigation frameworks. This can be quite confusing for users who are not good with computers and it can lead to frustration. Outdated website design can be regarded as unprofessional and unreliable, so it is important to design a website that is aesthetically pleasing so that it will build trust and attract more users.

The main objective of this project is to:

- Make the booking process simpler as the name of the project indicates so that travellers can book their travel in ease and relish their travels in peace.
- Create a friendly user interface to help users interact and navigate through the website easily.
- Help travellers explore new places without any inconvenience
- Help travellers to choose any destination of their choice along with the information about the destination.
- Inspire and Engage lot more people to travel across the world.

What makes ExploreEase unique from other travel agency websites is that it has a built-in currency converter functionality so that they can check the currencies of their desired destination's country. While booking the travel, users have the ability check their spending in different currencies so that they can plan their budget accordingly. The website also has the functionality to see more information of a particular country just to have general knowledge of the place that they will be visiting. Additionally, it also shows news of different countries so that they can keep themselves up-to date about their destined location. An example use case of this feature would be when there's storm warning in a country and users who plan to visit that country can time their trip accordingly so that they can have a safe trip.

In the next sections i will be walking through the process of development and integration of all the tools used along with the architectural diagram.

II. Project Specification and Requirements

A. Functional Requirements

1) **User registration and login**: Lets users create account and login through their credentials securely.
2) **Destinations**: A section that lets people choose different continent and choose their destination based on continents
3) **Currency Converter**: Lets users to converter currencies to their desired in seconds to help them plan their budget.
4) **News**: Integrate news API to help users to read about a country's news.

5) **Country information**: Integrate a country information API that gives information about any country.
6) **Bookings**: Implemented a simple and easy-to-use bookings feature that lets users to book their trip easily.

B. Non-Functional Requirements

1) **Scalability**: Used scalable infrastructure and technology to build the website so that it can manage growing traffic and future expansion.
2) **Performance**: Designed the architecture in such a way that it gives fast responses by utilizing cloud technologies.
3) **Security**: Implemented strong security measures to prevent from cyber attacks.
4) **User-friendly**: Designed the website with clean and aesthetic user interface to attract more users and help users to navigate through the website easily.
5) **Reliability**: Made the website reliable so that there are no errors and less downtime.

C. Technical Requirements

1) **Development tools**: To develop this website I have used Flask as the the web framework, HTML and CSS for front-end designs, Some JavaScript to retrieve requests from back-end(API) and display them.
2) **Hosting Environments**: To host the client app I have used AWS Code Pipeline to setup CI/CD pipeline and the web service was hosted on Elastic Beanstalk.
3) **API Integration**: Currency converter API, News API and World Information API were all integrated during the development of this website.

In the next upcoming sections I will be thoroughly explaining the development process and how the above mentioned tools were utilized in the entire project.

III. Architecture and Design

The cloud offers a wide range of services that will be helpful to use into our own application. Many companies today implement cloud services into their projects to make their application scalable, available and secure. Integrating cloud technologies into our project can benefit us in many different ways namely:

- Application can be scalable.It allows resources to scale up and scale down based on demand.
- Our application will be available at all times which means that the application will have little to none downtime.
- Hosting our application on to the cloud will our application reach users from all around the world.

These are just some of the advantages of utilizing cloud services in our application. The cloud services used in this project are from Amazon Web Services. The client application and the web service application were hosted on AWS Elastic Beanstalk [1]. Elastic beanstalk is platform as a Service offered by amazon that manages and scales applications hosted on it. Platform as a service is a model of cloud computing that provides a platform to deploy,develop

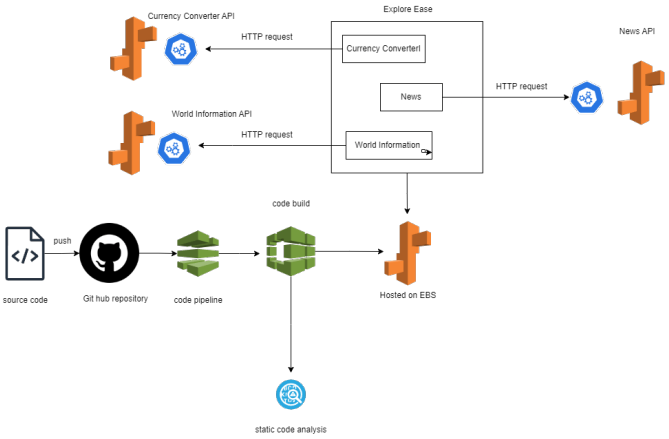

Fig. 1. Architectural Diagram

and manage any application without having to worry about maintaining the underlying infrastructure. One of the main reasons for choosing EBS is because of its simplicity and ability to scale the resources up and down based on demand automatically. Since the primary concern of my application is scalability, I decided to go with Elastic Beanstalk. It is easy to use and the developers can just upload the code and not worry about the underlying infrastructure because it is taken care by AWS EBS. Elastic beanstalk is very much cost effective since it scales down the resources when the traffic is low. It also provides a secure environment to execute your code and it has a built in monitoring service that lets you know when there is any threat or when an error occurs. It also has built-in Load Balancer to help manage the traffic and distribute the load across multiple instances. Since the web service will be receiving a lot of requests from other consumers, I thought it would be best to host it on Elastic Beanstalk because it has the capabilities to handle multiple requests at once.

The client application was hosted on Elastic Beanstalk using AWS Code Pipeline [2] by creating CI/CD pipeline. CI/CD makes the work flow a lot more smother to help developers make changes to the code and deploy in ease. AWS Code Pipeline automates the whole software development life cycle. AWS Code Pipeline has the ability to integrate various other AWS services in the pipeline such as, AWS Code Build, AWS Code Deploy, Elastic beanstalk etc. It also scales automatically to handle deployment packages of different sizes making the whole process scalable. We can also integrate multiple build stages to further enhance the software development life cycle. Code Pipeline also also allows GitHub to be the source provider since it is widely used and developers do not have to worry about creating new repository in AWS. CI/CD pipelines will help the automate the software development life cycle which will be quite convenient to push the changes to the production environment during the development process. The whole

workflow of the CI/CD pipeline will be briefly explained in the ”Continuous integration, delivery and deployment” section of the document.

IV. Implementation

The implementation of the APIs were quite simple and straightforward. In this section we will be elaborating and briefly explaining each and every implementation done in the entire web development process.

A. Currency Converter API

An Application Programming Interface is a way that allows two or more computers to communicate with each other. An API acts as an interface between two computers programs that provides any service from one program to the other. An API specification is a written document or standard that outlines how to create or utilize such a connection or interface. I developed a currency converter API that is used to convert any given currency to their desired currency. The API takes 'from currency','to currency' and the amount to be converted as the input. These inputs come in as a HTTP GET request from another program. The HTTP GET is used to request data from a specified resource. The API gets the request from another server or a program and calculates the currency based on the given data. Then the calculations are stored into 'converted amount' and is sent back to the requested source in a JSON format. JSON [3] is a lightweight format that is primarily used to transfer and store data. JSON stands for JavaScript Object Notion and is widely used because it makes it simpler to transfer data from one place to another. JSON is compatible with a programming languages so if one program is developed in JavaScript and one program is written in Python, JSON can help these two programs to communicate with each other. Usually while developing API we use JSOn to send and receive data because of its compatibility and simplicity.

The API it self was developed in flask. Flask [4] is a python web framework that is used to build web applications. It lacks any form validation, database abstraction layer, or other components where common functions are provided by pre-existing third-party libraries. On the other hand, Flask allows extensions to provide functionality to the application as though it were Flask native. There are extensions available for several open authentication systems, object-relational mappers, form validation, upload handling, and numerous utilities linked to standard frameworks. Flask is very easy to use which makes it suitable to develop API.

Jsonify [5] is a python library that takes csv files and converts them to JSON format. We have used this library to return the converted amount in a JSON format. The API can be accessed through any web browser just by entering the API's HTTP link. But during the integration of the API into my application, I had faced a certain 'CORS policy' issue. Cross-Origin Resource Sharing [6] is a mechanism that allows restricted access to one web server other than the one that is hosted. We need CORS policy access enabled in our code to let other web applications to make requests to our web service. This can be done by installing a python library called 'flask-cors' [7]that will help the web service to get requests from other client applications. If the CORS policy is not specified, the API will not be able to interact with any of the client applications that are trying to communicate with the web service or API.

The API was hosted on Elastic beanstalk to maintain scalability and have the ability to handle multiple requests at once. I deployed the API from AWS cloud 9 by installing EBS CLI. Before deploying the web service application, there are few pre requisites to be followed. We need to first create a configuration file called python.config inside of the folder called .extensions. This is to let Elastic Beanstalk know where to find the application. My python file was named application.py and the config file had the following code.

```
option_settings:
  "aws:elasticbeanstalk:container:python":
    WSGIPath: application:application
```

If my python file was named app.py then the config file will have app:app in WSGI path. This configuration file is crucial while deploying the application to EBS because without this EBS will not be able to find the application file. We also need a requirements.txt file to has all of the necessary dependencies to run the application on elastic beanstalk. To create a requirements we need to ”pip freeze > requirements.txt” and it will list out all the libraries installed into the text file. EBS will look for a file named ”requirements.txt ” and install all the libraries mentioned in the text file. With all this sorted the application should be ready to be deployed with the eb init(creates an application) command and eb create(creates an environment) command.

B. API integration

After the development and deployment of the API we need to integrate it into our application. As mentioned in the previous section we need to make sure that CORS polices are enabled before integration. In order to fetch and display the input and out parameters of the API and the client application, we need some sort of back-end functionality to handle the requests in the client side. JavaScript [8] is the best programming language to use to handle such features. JavaScript is the programming language of the web that used by the client side of the websites. i developed a JavaScript code to get the input from the HTML form and make a HTTP request and send the inputs, i.e., from currency,to currency and amount, to the currency converter API. Then the API processes the user input and returns the converted amount back to the JavaScript program to display in the HTML page. Here is the code snippet of the JavaScript code that handles the HTTP requests to and from the API.

```
function convertCurrency() {
var amount = document.getElementById
("amount").value;
var fromCurrency = document.getElementById
("from_currency")
.value;
var toCurrency = document.getElementById
("to_currency")
.value;

$.ajax({
url: "http://currency.eba-qchvcgzv.
us-east-1.elasticbeanstalk.com/convert",
type: "GET",
data: { amount: amount, from_currency:
fromCurrency, to_currency: toCurrency },
success: function(response) {
document.getElementById("result").
innerHTML = amount + " " + fromCurrency
+ " = " + response.converted_amount.
toFixed(2)+ " " + toCurrency;
},
error: function(_xhr, _status, error) {
console.error(error);
document.getElementById("result")
.innerHTML
= "An error occurred.
Please try again later.";
}
});
}
```

The same procedure was followed while implementing world information API and News API. Two separate JavaScript programs were written to handle each of the API's respectively. The World information API takes the country name as the input and returns the population, capital, Currency etc of that particular country. The news API returns the present day's top headlines in a JSON format and JavaScript file will help fetch and display the top headlines.

C. ExploreEase

To design the application and make it user-friendly I used HTML and CSS. Hyper-Text Markup Language is one of, if not, the most popular language used to design the front-end of any web application. HTML is easy to learn and simple to use. Cascading Styles Sheets a language used for presentation and styling of HTML page to make it more attractive. In order to make the website user-friendly, I have designed a simple and attractive looking website using HTML and CSS.

The client application was developed in HTML, CSS, JavaScript and Flask. HTML and CSS as mentioned above was used for the front end design. Flask was the web framework, that uses python programming language, to code the web application. As mentioned in the previous subsection JavaScript was used to handle the the back-end functionality(API requests).

Some of the key features of the websites are, the destination tab that users can select different continents and choose their desired destination., the booking section of the website that is made simpler to help users to book their travel easily. The user login and sign up details are stored in the database. The data base in this application is SQLite. The database engine SQLite was created using the C programming language. It is a library that software developers incorporate into their programs rather than a stand-alone application. Consequently, it is a member of the embedded database family. To use SQLite database in our flask app we just need to install library called flask_sqlalchemy. Flask does not come with database models like Django. In Django, once we create a Django project it will automatically create all the files necessary. But this will cause confusion because of the file structure created. Flask is much easier to use and implement and that is the reason I decided to use Flask as the underlying web framework.

V. Continuous integration, delivery and deployment

Continuous Integration, delivery and deployment helps automate the whole software development life cycle. It is a series of steps that will help the code to easily flow through the pipeline into the production environment [9]. It makes pushing newer versions of the code to the production a lot easier so that the developers can focus on coding. AWS Pipeline is used to create the CI/CD pipeline as mentioned in the second section of the document. The flow of the code will is as follows,

1) **Source**: The source code is first pushed into the GitHub repository. The GitHub account is linked to AWS during the creation of the pipeline.

2) **Build**: In the build stage of the pipeline, Sonar cloud is implemented for Static Code analysis. Static Code analysis is a type of code analysis method that analysis the code without having to execute the code. Sonar cloud [10] is a cloud based static code analysis tool that performs static code analysis on any code on the cloud environment without having to install any software. So the pipeline performs static code analysis on the code before reaching the deployment stage.

3) **deploy**: The code smoothly flows through the pipeline and then reaches the deployment phase. the code can be hosted on different AWS services like EBS, ECR, Lambda etc.

There few prerequisites that are to be done before creating the CI/CD pipeline. We need a yaml file that will install all the dependencies in the build stage for sonar cloud to

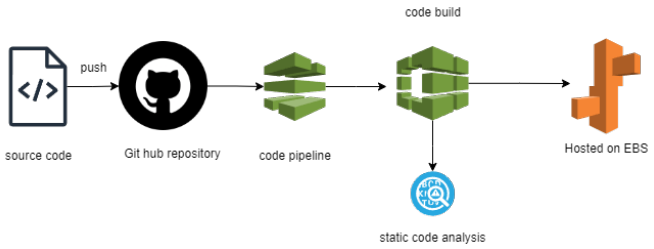

Fig. 2. CI/CD Pipeline

execute its commands. Note that the file should be named as buildspec.yml. Along with that we need a python.config file as done while deploying the web service. Finally a requirements.txt file with all the libraries needed to run the application Elastic beanstalk. So every time a new version of the code is pushed into the GitHub repository, it will first be analysed by sonar cloud to check any security vulnerabilities and bugs are present.

```
    version: 0.2

env:
    LOGIN: "a95ad9141e9eeb9d77ad06b52d
    353369f9167faf"
    HOST: "https://sonarcloud.io"
    Organization: "prajwalsr7"
    Project: "prajwalsr7_travel1"
phases:
  install:
    runtime-versions:
      java: corretto21
  pre_build:
    commands:
      - yum update -y
      - yum install -y jq
      - retry_count=3
      - for i in $(seq 1 $retry_count);
      do
      wget https://archive.apache.org/dist
      /maven/maven-3/3.5.4/binaries/apache
      -maven-3.5.4-bin.tar.gz && break ||
      sleep 10; done
      - tar xzf apache-maven-3.5.4-bin.tar
      .gz || true
      - ln -s apache-maven-3.5.4 maven
      - wget https://binaries.sonarsource
      .com/
      Distribution/sonar-scanner-cli/
      sonar-scanner-cli-3.3.0.1492-linux
      .zip
      - unzip ./sonar-scanner-cli-3.3.0
      .1492
      -linux.zip
  build:
    commands:
      - mvn sonar:sonar -Dsonar.login=
      $LOGIN
      Dsonar.host.url=https://sonarcloud.io
      Dsonar.projectKey=prajwalsr7_travel1
      Dsonar.organization=prajwalsr7 || true
      - sleep 5
      - curl
      https://sonarcloud.io/api/qualitygates/
      project_status?projectKey=
      prajwalsr7_travel1 >result.json
      - cat result.json
      - if [ $(jq -r '.projectStatus.status'
      result.json) = ERROR ]
      ; then $CODEBUILD_BUILD_SUCCEEDING
      -eq 0 ;fi
  artifacts:
  files:
  - '**/*'
```

This is the yaml file used to configure the web application that will install all the dependencies and run all the necessary commands in the build stage.

VI. Conclusion

ExploreEase is travel agency website aimed to make a easy-to-use and user-friendly website to help travelers navigate through the web application. The project's primary goals of making bookings easier, offering a user-friendly interface, making it easier to explore new places, and encouraging more people to travel are all effectively attained.

Three different APIs were integrated in the application. Currency converter API was developed my me using Flask and was hosted on the cloud. The other two APIs integrated were news API and world information API. The API was hosted on Elastic beanstalk to make the web service scalable and the capability to handle many HTTP requests at once.

The front-end application was developed in Flask, HTML,CSS and JavaScript to make the User interface attractive. The application was hosted on Elastic beanstalk through creating CI/CD pipeline using AWS code pipeline to automate the software development life cycle. Sonar cloud was implemented to check the quality of the code by performing static code analysis.

Overall, during the development of this application I understood how industry level applications are built using the state-of-the-art cloud technologies. While integrating the API into the client application, I found out that the CORS policy should be specified. If not specified the API will not be able to get any requests from any of the applications. By deploying the applications on to the cloud, I made the environment scalable and also understood the importance of automating the software development by using AWS CodePipeline.

REFERENCES

[1] AWS, "Deploying a Flask application to Elastic Beanstalk," https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html, 2021, [Online; accessed 2024-04-16].

[2] ——, "AWS CodePipeline User Guide," https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html, 2021, [Online; accessed 2024-04-16].

[3] A. Anie, "What is JSON," https://medium.com/@alexanie_/what-is-json-2d2e973e14fe, 2023, [Online; accessed 2024-04-16].

[4] Flask, "Flask Documentation," https://flask.palletsprojects.com/en/3.0.x/, 2021, [Online; accessed 2024-04-16].

[5] Jsonify, "Jsonify Documentation," https://pyphi.readthedocs.io/en/latest/api/jsonify.html, 2024, [Online; accessed 2024-04-16].

[6] S. Hobbs, "CORS Tutorial: A Guide to Cross-Origin Resource Sharing," https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/, 2019, [Online; accessed 2024-04-16].

[7] F. CORS, "Flask CORS Documentation," https://flask-cors.readthedocs.io/en/latest/, 2013, [Online; accessed 2024-04-16].

[8] W. schools, "JavaScript Tutorial," https://www.w3schools.com/js/, 2024, [Online; accessed 2024-04-16].

[9] S. Arachchi and I. Perera, "Continuous integration and continuous delivery pipeline automation for agile software project management," in *2018 Moratuwa Engineering Research Conference (MERCon)*, 2018, pp. 156–161.

[10] S. Cloud, "Sonar Cloud Documentation," https://docs.sonarsource.com/sonarcloud/, 2024, [Online; accessed 2024-04-16].