National College of Ireland

Project Submission Sheet

Student Name:	Anurag Singh		
Student ID:	X23180013		
Programme:	Master of Science in Cloud Computing	**Year:**	2024-2025
Module:	Cloud DevOpsSec		
Lecturer:	Adriana E. Chis		
Submission Due Date:	Tuesday, 9th April 2024, 2:00 PM		
Project Title:	SalonConnect: An appointment booking application		
Word Count:	3235		

Signature:	Anurag Singh
Date:	09/04/24

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

SalonConnect: An appointment booking application

Anurag Singh
StudentId: x23180013
Cloud DevOpsSec, MSc in Cloud Computing
National College of Ireland Dublin, IRELAND
Email: x23180013@student.ncirl.ie. URL: www.ncirl.ie
Deployed Application URL: http://x23180013mysalon-env.eba-4wzwbr6m.eu-west-1.elasticbeanstalk.com/
Video URL: https://youtu.be/7PAqRCsk11Q

***Abstract*—Cloud computing has transformed the field of software industry and deployment by overcoming traditional infrastructure limitations and providing on-demand access to computer resources. Cloud computing has impacted the way software is developed, deployed, and managed. SalonConnect uses this technology to improve operational efficiency, client engagement, and appointment booking. This study explains the architectural design, continuous integration and continuous deployment (CI/CD) process, static code analysis approach, and developmental insights that went into the construction of SalonConnect. It is developed using Python and Django framework on Cloud9. The CI/CD pipeline, built using Amazon Web Services (AWS) components such as Elastic Beanstalk and CodeDeploy helps in the automation and orchestration of development activities. Through static code analysis done using Pylint, potential vulnerabilities and opportunities for code quality improvement were also identified. This study showcases the intersection of technological innovation and practical utility in addressing current challenges in the beauty service industry, while also providing a roadmap for future research and development endeavours in related domains.**

***Index Terms*—Cloud computing, Python, Django, Continuous Integration and Continuous Deployment (CI/CD), Cloud9, Static Code Analysis, Elastic Beanstalk, CodeDeploy, Pylint and SonarCloud.**

I. INTRODUCTION

In the past decade, Cloud Computing has transformed the way businesses operate by providing more reliable and cost-effective solutions. The significant trend in the adoption of cloud services has shifted from migration to the rapid adoption of cloud computing services. This has resulted in a considerable impact on the software development industry. The rise of cloud computing has simplified the development, deployment, and management of software applications, providing novel opportunities for innovation and collaboration in the software engineering process [1]. It has emerged as a game-changer in the world of software engineering and development. The benefits of cloud-based infrastructure go beyond cost savings. With on-demand access to cloud-based infrastructure, software engineers and developers are now able to experiment with a wide range of tools and technologies without any expensive gear and infra. This made the software development process more accessible and cost-effective but has also helped businesses to innovate and stay competitive in an increasingly dynamic market.

The popular cloud service providers are Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) [2]. These companies offer a wide range of services for software development, including virtual machines, container management, serverless computing, and developer tools and frameworks. AWS provides elastic computing, ECS, Lambda, and a variety of developer tools and frameworks. Azure offers virtual machines, Azure Kubernetes Service, Azure Functions, and DevOps solutions. Lastly, GCP provides services such as Compute Engine, Google Kubernetes Engine, Cloud Functions, and CloudBuild. Businesses looking to stay ahead in the tech industry must embrace cloud computing in software development. Cloud resources and services can help development teams simplify their workflows, reduce expenses, and deliver innovative software solutions at a faster pace than before.

SalonConnect is a platform that uses cloud-based technology to transform the appointment booking experience for beauty salons, spas, and wellness centres. This platform aims to solve the inefficiencies and problems that exist in current appointment booking systems in the beauty service industry. Manual scheduling methods, unconnected communication channels, and limited accessibility often result in mediocre service delivery and low client satisfaction. SalonConnect integrates cloud computing technologies and DevSecOps ideas to boost operational efficiency, increase client interaction, and expedite the booking process. The application is developed in two phases: the execution of essential functionalities, as defined in the functional requirements, and the incorporation of AWS services, as detailed in the non-functional requirements.

- Functional Requirements
  1) User Authentication: Admins can log in to their accounts.
  2) Appointment Booking: Customers can book appointments. Customers receive email reminders for their appointments.
  3) Manage Booking: Admins can manage the appointments by accepting, cancelling or suggesting different slots for booking via mail.
- Non-Functional Requirements
  1) Availability: AWS Elastic Beanstalk ensures that the application is available to users with minimum downtime.
  2) Security: Authentication and authorisation are im-

plemented for different user types. The code is developed to protect against cyber attacks like CSRF, XSS and SQL injection.

3) Usability: The application UI is user-friendly. Allowing users to easily navigate within the application. This application is available on various devices like mobiles, laptops and tablets.
4) Scalability: Using AWS Elastic Beanstalk allows the use of its built-in load balancer thus ensuring the ability to scale on demand to give optimal performance at reduced infrastructure cost.
5) Performance: The application is designed to ensure minimal delay to user interactions. AWS Elastic Beanstalk assures dynamic scalability without affecting performance. Supporting a high volume of concurrent users simultaneously.

II. Architectural Design

This section as outlined in the figure 1 explains the architecture design of SalonConnect. The application performs CRUD operations i.e. allowing users to book an appointment. The Salon admin who is the super user can view requested appointments, update the booking date by suggesting a new available date, and delete existing appointments. Thus super users are granted the ability to manage the appointment efficiently. The architecture of the Django framework follows a Model-View-Controller (MVC) [3] design pattern providing compatible, scalable, and maintainable applications. It consists of the following components mentioned below:

- Model Layer: This Layer defines the data members for an appointment and other important entities utilizing Django's ORM (Object-Relational Mapping) [3] to interact with the SQLite database assuring data integrity and consistency.
- View Layer: This layer consists of views for handling HTTP requests and responses. The application renders HTML templates to display views like adding, modifying, viewing, and deleting appointments.
- Controller: This layer is responsible for mapping and directing the URL patterns to corresponding views in the application for processing. Another critical aspect of controllers is to enable clean separation between the URL routing logic and view separation.
- Middleware: This layer provides a mechanism for processing requests and responses before and after the view layer. Middleware like SecurityMiddleware, CsrfViewMiddleware, and AuthenticationMiddleware are used in the application to enhance security and extensibility.

Furthermore the figure2 shows detailed components of AWS cloud architecture, tools used for development and the (CI/CD) flow. The code development and workflow of this application are done on AWS Cloud9. It is an (IDE) that you can access through your browser online for writing, running, and debugging code. With Cloud9, you get a code editor, debugger, and terminal. It promotes efficient workflows and team collaboration. Elastic Beanstalk [4] can swiftly install and maintain apps on the AWS Cloud without having to learn about the infrastructure that supports them. Elastic Beanstalk decreases administration complexity without limiting options or control by uploading the application, and Elastic Beanstalk will handle capacity provisioning, load balancing, scaling, and application health monitoring. The Elastic Beanstalk console, or eb, is a high-level command-line interface created exclusively for Elastic Beanstalk. It automatically starts an environment and sets up the AWS resources required to run the code.

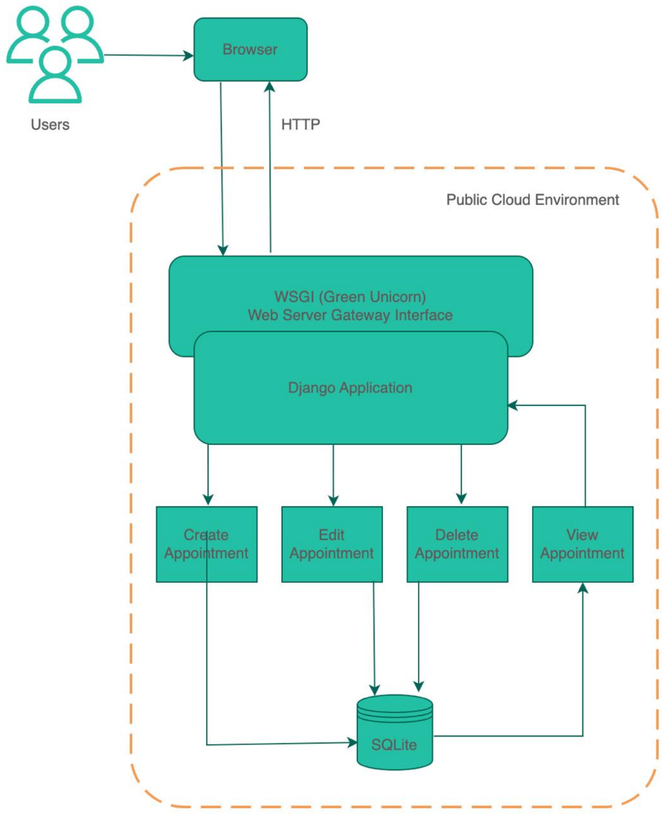


Fig. 1. SalonConnect architecture diagram

CodeDeploy is used to deploy the application code to instances automatically to enable fast and reliable application upgrades. The application also uses pylint for static analysis techniques to discover code quality concerns and security vulnerabilities, ensuring the application codebase is resilient and secure. Lastly, the application uses SonarCloud to scan and assess potential security flaws regularly. Any found vulnerability is then fixed to protect user data and application integrity.

III. Continuous integration, continuous delivery and deployment

A. Stage 1

GitHub is used for version control systems during development. The AWS CodeCommit is configured to this GitHub

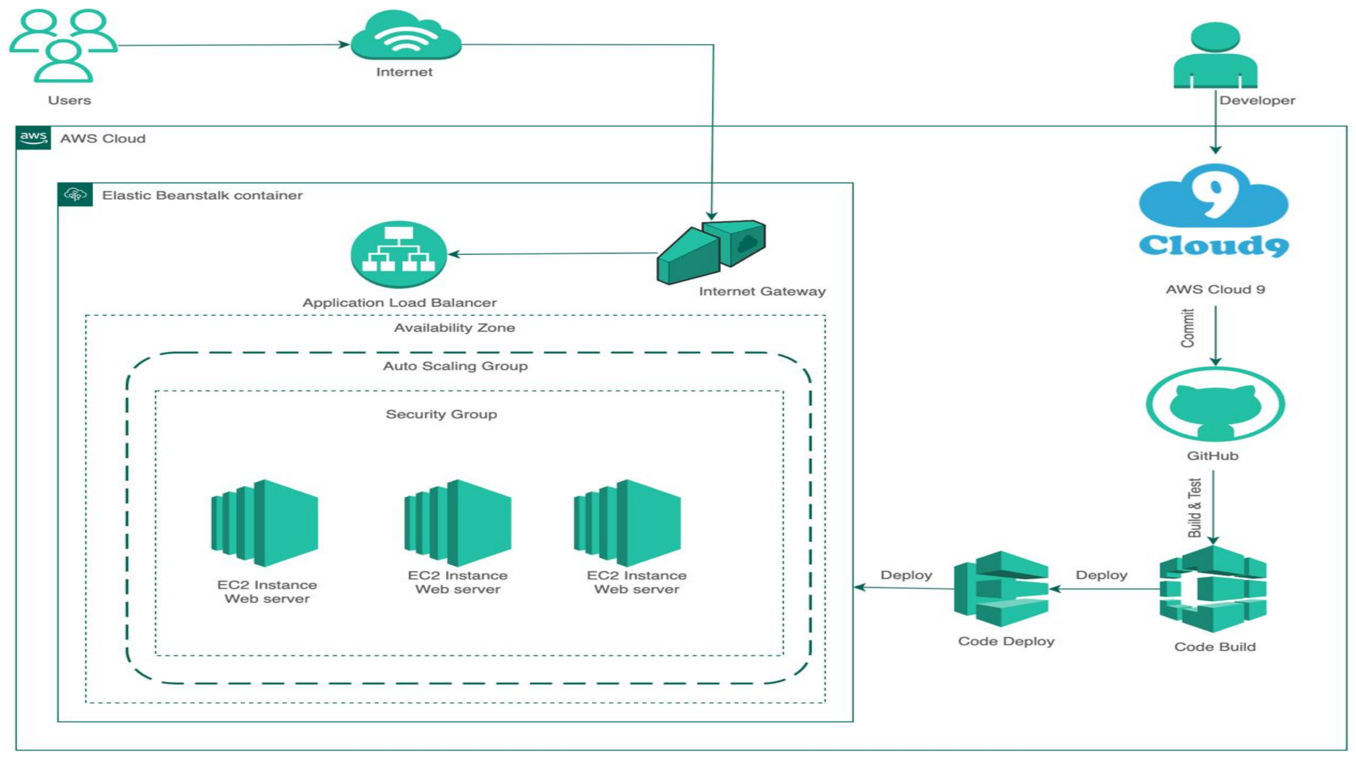


Fig. 2. AWS Cloud architecture diagram

account. GitHub webhooks automatically trigger the pipeline when a new commit is pushed to the source code. This defines the first stage of the CI/CD pipeline of the application which is named Source.

B. Stage 2

After completing the first phase, the next step is the Code Review. This stage involves integrating SonarCloud to analyse and detect any vulnerabilities in the code. The build is selected from the AWS CodeBuild console and assessed against a set of rules or policies defined in the quality gate of the Sonar cloud. Once the analysis is complete, SonarCloud lists any bugs, security vulnerabilities, code smells, unit test coverage, and duplication found in the code. Developers then work to resolve these issues to ensure data security, adhere to coding best practices, and maintain easy-to-read code.

C. Stage 3

The Static Code Analysis stage is the next step after completing the Code Review stage. In this stage, the project which is already created in the AWS CodeBuild console is selected to trigger a single build. The input artifact is the source stage artifact, and the build type is specified as a code build. The main emphasis in this stage is to run the code build against predefined Python linting rules using pylint to avoid bugs, deprecated methods, code smells, and duplication. The stage aims to follow Python coding best practices and gives a score out of 10 to the targeted folder specified in the build spec file. To pass this stage, the targeted folder must score 10/10 to maintain high coding standards, reliability, and less code maintenance.

D. Stage 4

The final stage of deploying involves using AWS Elastic Beanstalk service, which makes the deployment process quick and easy. It automates the deployment process, taking away the worries of developers about the underlying infrastructure needed for scaling and deploying the application. By choosing an environment that has already been created in the Elastic Beanstalk console, developers can get the necessary infrastructure elements such as EC2 instances, load balancer, auto-scaling groups, and network configurations required for their application usage. The Elastic Beanstalk console also provides an option to select an application that has been created before, which allows developers to deploy the latest application versions without experiencing any downtime. The console supports rolling updates, which gradually move the traffic from the old version of the application to the latest version, ensuring a smooth transition without affecting existing users. The figure 3 shows the entire pipeline flow.

Seamless integration of AWS CodeCommit, CodePipeline, and CodeDeploy along with Elastic Beanstalk helps and allows developers to automate the deployment as a part of the CI/CD pipeline, enabling continuous delivery of updates by fetching the latest version release of the application with minimal manual intervention.

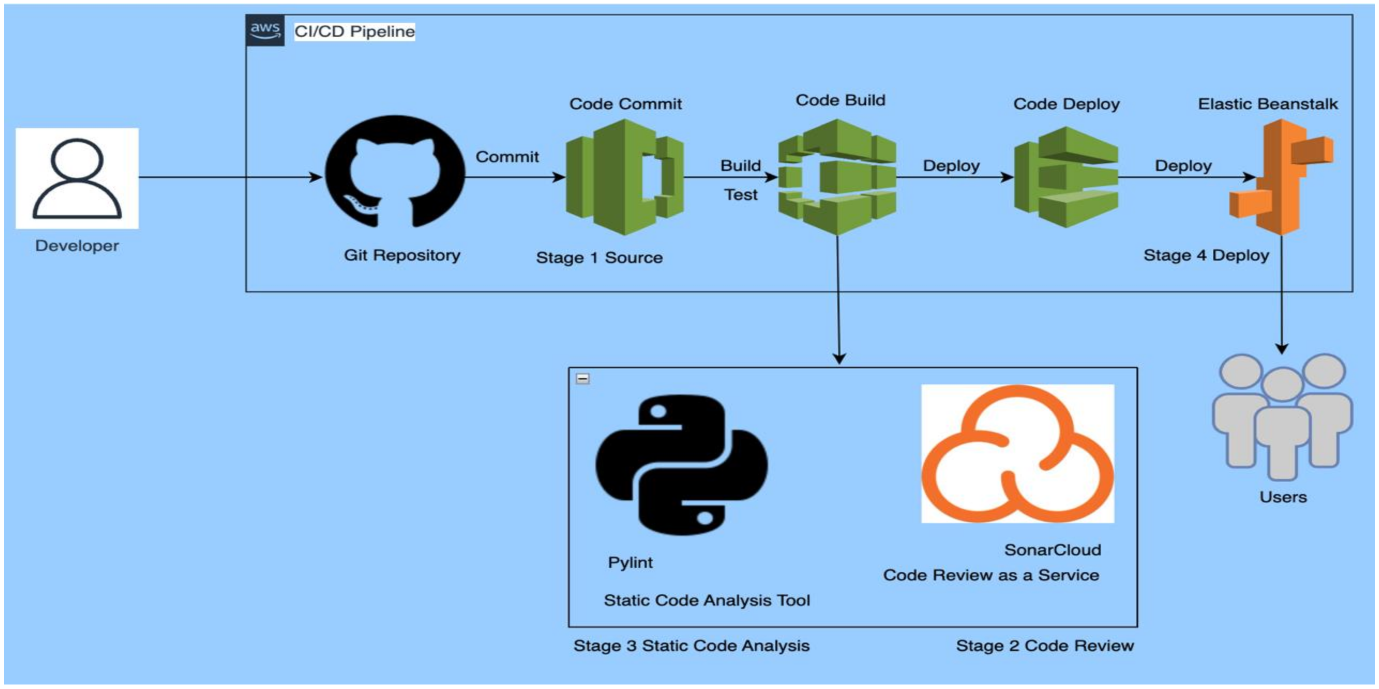
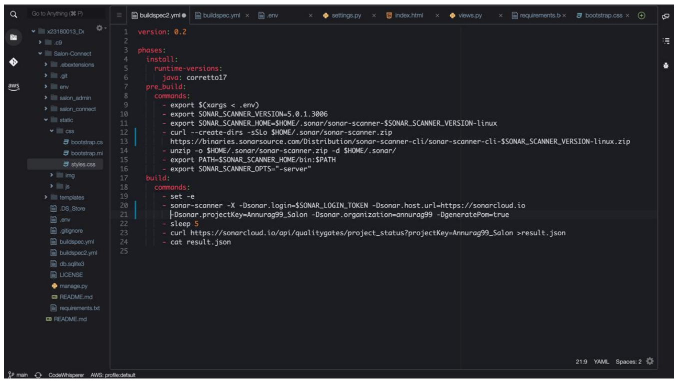


Fig. 3. Continous Integration Continuous Deployment (CI/CD) pipeline diagram

IV. Static Code Analysis

A. Security Vulnerability

Implementation of SonarCloud [5] helps in code review by identifying vulnerabilities, bugs, code smells, security hotspots, duplication and coverage. SonarCloud supports a range of programming languages like Python, Java, C, Type-Script and more. The SalonConnect application automatically integrates with the CI/CD pipeline and analyzes code changes as they are committed. The configuration is done in the buildspec2.yml file as shown in figure 4. The Phases

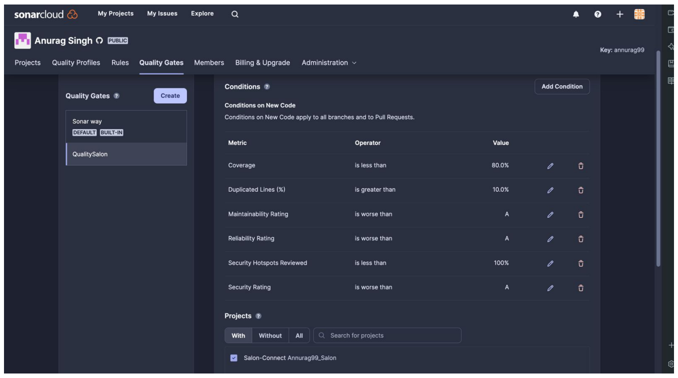


```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto17
  pre_build:
    commands:
      - export $(xargs < .env)
      - export SONAR_SCANNER_VERSION=5.0.1.3006
      - export SONAR_SCANNER_HOME=$HOME/.sonar/sonar-scanner-$SONAR_SCANNER_VERSION-linux
      - curl --create-dirs -sSLo $HOME/.sonar/sonar-scanner.zip
        https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-$SONAR_SCANNER_VERSION-linux.zip
      - unzip -o $HOME/.sonar/sonar-scanner.zip -d $HOME/.sonar/
      - export PATH=$SONAR_SCANNER_HOME/bin:$PATH
      - export SONAR_SCANNER_OPTS="-server"
  build:
    commands:
      - set -e
      - sonar-scanner -X -Dsonar.login=$SONAR_LOGIN_TOKEN -Dsonar.host.url=https://sonarcloud.io
        -Dsonar.projectKey=Annurag99_Salon -Dsonar.organization=annurag99 -DgeneratePom=true
      - sleep 5
      - curl https://sonarcloud.io/api/qualitygates/project_status?projectKey=Annurag99_Salon >result.json
      - cat result.json
```

Fig. 4. SonarCloud configuration file

indicate the configuration for different stages of the CI/CD pipeline. Installing necessary dependencies and setting up an environment. The pre-build phase consists of commands that are to be executed before the main build process starts. This phase includes the commands to read the variables from the environment file and export them into the current shell session, set the version of SonarScanner to be used, set the path where SonarScanner will be installed, download and unzip the SonarScanner file into the installation directory and adding to the binary directory the system path where it can be executed from any location. During the build phase, commands are executed to ensure that it process fails if any command fails. The process includes SonarScanner analysis of the codebase. The analysis is executed with different parameters such as login token, project key, and organization. The result of the analysis is then sent to SonarCloud for further analysis. Another important property of the application is that it allows users to set custom quality rules based on project requirements. These quality gates need to be passed by every code update. The figure 5. figure shows the quality rules set for the project and an initial scan of the code shows the errors and security vulnerabilities. The curl command retrieves the project quality gate status from SonarCloud API and saves it in results.json. At last the contents of the result.json file are displayed through the view details button in the CI/CD stage for code review. The first build had 115 bugs, 3.5k code smells, 215 security hotspots and 17 security vulnerabilities which were resolved during the bug identification and resolution phase of the project as seen in figure 6 and 7. All the reported vulnerabilities in SonarCloud were resolved as given below:

- Direct assignment of Django secret key
  - File name: settings.py
  - Solution: Initialised the secret key as a key-value



Key: annurag99

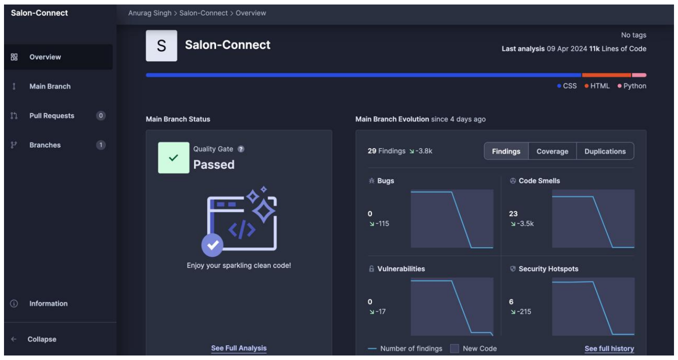
Quality Gates
Sonar way DEFAULT BUILT-IN
QualitySalon

Conditions
Conditions on New Code
Conditions on New Code apply to all branches and to Pull Requests.

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	10.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

✓ Salon-Connect Annurag99_Salon

Fig. 5. SonarCloud quality gate setup



Salon-Connect
No tags
Last analysis 09 Apr 2024 11k Lines of Code
● CSS ● HTML ● Python

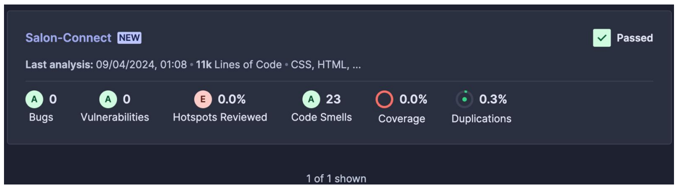
Main Branch Status
Quality Gate Passed

Main Branch Evolution since 4 days ago
29 Findings -3.8k
Bugs 0 -115
Code Smells 23 -3.5k
Vulnerabilities 0 -17
Security Hotspots 6 -215

Fig. 6. SonarCloud reported bugs and vulnerabilities

pair in the .env file and imported the config function from the decouple library as it automatically loads the .env file and makes the environment variables available in the Django project.

- SonarCloud login token
  - File name: buildspec2.yml
  - Solution: Used prebuild command $export(xargs < .env)$ in buildspec2.yml file to export the variable SonarLoginToken from .env file.
- Google Maps API key
  - File name: index.html
  - Solution: Initialised the Google map API key as a key-value pair in the .env file and implemented it in the index.html file.



Salon-Connect NEW
✓ Passed
Last analysis: 09/04/2024, 01:08 • 11k Lines of Code • CSS, HTML, ...

A 0	A 0	E 0.0%	A 23	0.0%	0.3%
Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications

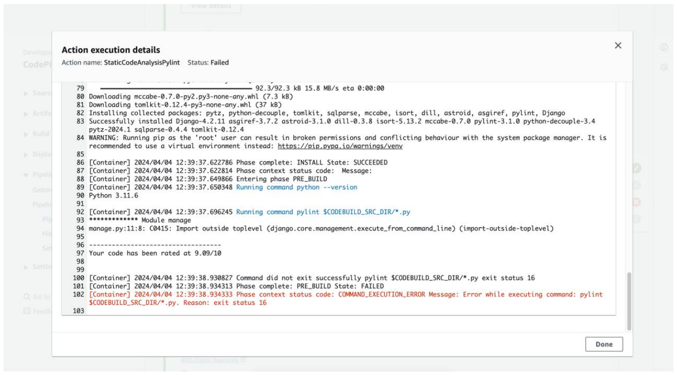
1 of 1 shown

Fig. 7. SonarCloud report overview

The learning from the above fixes is that while handling any sensitive information like keys or tokens should be securely managed by storing them in environment variables. Another approach to consider might be to utilise secret management services like AWS secrets manager. Using Django's built-in form validation and template escaping to avoid security threads from user input, validation and sanitization is important to prevent web threads. Following best practices for authentication, authorization and session management along with configuring Django settings, like disabling DEBUG mode in production, setting up security middleware layers, and using HTTPS protocol in the future. At last Django framework, third-party libraries and other dependencies should be updated regularly to patch any security vulnerability instantly. Security weaknesses can also be identified regularly by running security assessments which comprised of penetration testing, vulnerability checks and quality code reviews. Future works must include the implementation of logging and monitoring mechanisms to identify and resolve security incidents in real time. The implementation-related discovered bugs were also resolved to satisfy the quality gate condition of zero bugs.

B. Static Code Analysis

Static code analysis [6] is a technique that examines the source code of a software application without running it. This early detection reduces the cost and effort required to fix issues later in the development process, which can be more challenging to address making the code more maintainable and reliable. SalonConnet implements Pylint [7] which is a tool that helps ensure code quality for Python programmers. Additionally, it can identify type errors, check coding conventions including style guidelines as per PEP8, unused imports and variables, syntax errors, suggest code refactoring, and detect possible errors like uninitialized variables, misplaced return values and unreachable code. It also provides insights into code complexity. It uses the buildspec file which includes the commands in different phases to install dependencies and set up an environment. The commands phase first upgrades pip to the latest version and installs all the dependencies listed in the requirements.txt file of the project along with the pylint packages. The pre-build phase similar to the command phase consists of commands to be executed before the final build such as printing the Python version installed and running pylint to perform static code analysis on Python files in the specific directories. The artifact is generated as a result of the build process. Finally, the files command specifies the files to be included in the artifacts, in the project it includes all the Python files and directories by using (*/). Figure 8 and Figure 9 show the initial and fixed score It checks for all the possible syntax and logical errors i.e. unused variables, unreachable code etc. It enforces coding standards and styling guidelines as per PIP8. It also measures code complexity metrics such as cyclomatic complexity and nesting depth for complex code detection which are difficult to understand, test and maintain. shows the details of the output for the static code analysis, After considering all of the above rules the build process

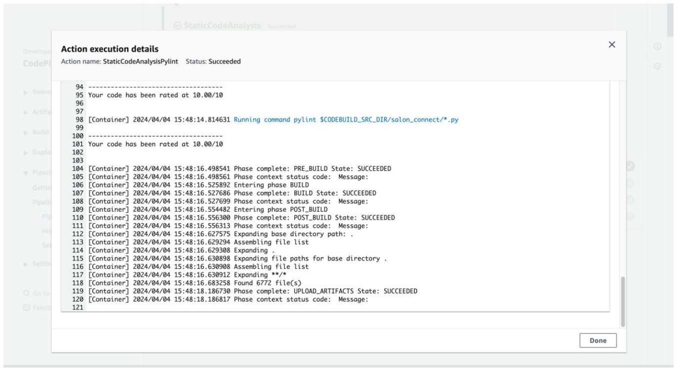


Action execution details

Action name: StaticCodeAnalysisPylint   Status: Failed

```
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.3/92.3 kB 15.8 MB/s eta 0:00:00
Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Downloading tomlkit-0.12.4-py3-none-any.whl (37 kB)
Installing collected packages: pytz, python-decouple, tomlkit, sqlparse, mccabe, isort, dill, astroid, asgiref, pylint, Django
Successfully installed Django-4.2.11 asgiref-3.7.2 astroid-3.1.0 dill-0.3.8 isort-5.13.2 mccabe-0.7.0 pylint-3.1.0 python-decouple-3.4
pytz-2024.1 sqlparse-0.4.4 tomlkit-0.12.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is
recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

[Container] 2024/04/04 12:39:37.622786 Phase complete: INSTALL State: SUCCEEDED
[Container] 2024/04/04 12:39:37.622814 Phase context status code:  Message:
[Container] 2024/04/04 12:39:37.649866 Entering phase PRE_BUILD
[Container] 2024/04/04 12:39:37.650348 Running command python --version
Python 3.11.6

[Container] 2024/04/04 12:39:37.696245 Running command pylint $CODEBUILD_SRC_DIR/*.py
************* Module manage
manage.py:11:8: C0415: Import outside toplevel (django.core.management.execute_from_command_line) (import-outside-toplevel)

-----------------------------------
Your code has been rated at 9.09/10


[Container] 2024/04/04 12:39:38.930827 Command did not exit successfully pylint $CODEBUILD_SRC_DIR/*.py exit status 16
[Container] 2024/04/04 12:39:38.934313 Phase complete: PRE_BUILD State: FAILED
[Container] 2024/04/04 12:39:38.934333 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command: pylint
$CODEBUILD_SRC_DIR/*.py. Reason: exit status 16

```

Fig. 8. Inital Pylint score



Action execution details

Action name: StaticCodeAnalysisPylint   Status: Succeeded

```
-----------------------------------
Your code has been rated at 10.00/10


[Container] 2024/04/04 15:48:14.814631 Running command pylint $CODEBUILD_SRC_DIR/salon_connect/*.py

-----------------------------------
Your code has been rated at 10.00/10


[Container] 2024/04/04 15:48:16.498541 Phase complete: PRE_BUILD State: SUCCEEDED
[Container] 2024/04/04 15:48:16.498561 Phase context status code:  Message:
[Container] 2024/04/04 15:48:16.525892 Entering phase BUILD
[Container] 2024/04/04 15:48:16.527686 Phase complete: BUILD State: SUCCEEDED
[Container] 2024/04/04 15:48:16.527699 Phase context status code:  Message:
[Container] 2024/04/04 15:48:16.554482 Entering phase POST_BUILD
[Container] 2024/04/04 15:48:16.556300 Phase complete: POST_BUILD State: SUCCEEDED
[Container] 2024/04/04 15:48:16.556313 Phase context status code:  Message:
[Container] 2024/04/04 15:48:16.627575 Expanding base directory path: .
[Container] 2024/04/04 15:48:16.629294 Assembling file list
[Container] 2024/04/04 15:48:16.629308 Expanding .
[Container] 2024/04/04 15:48:16.630898 Expanding file paths for base directory .
[Container] 2024/04/04 15:48:16.630908 Assembling file list
[Container] 2024/04/04 15:48:16.630912 Expanding **/*
[Container] 2024/04/04 15:48:16.683258 Found 6772 file(s)
[Container] 2024/04/04 15:48:18.186730 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
[Container] 2024/04/04 15:48:18.186817 Phase context status code:  Message:

```

Fig. 9. Final Pylint score

writes the score in the result.json file which can be accessed by clicking on the view details button within the static code analysis stage block of the CI/CD pipeline. On an initial build, a few errors regarding trailing new lines, unused variables, used imports, missing module-docstring, naming convention, line length and unreachable code were reported, thus rating 9.09/10. These code fixes were implemented in the static analysis errors solution phase to get a 10/10 rating.

V. Conclusions

Cloud services are becoming more reliable and thus are a a great choice for large industries, start-ups, and individuals. The project has helped me to get hands-on experience with different AWS services such as Elastic Beanstalk, Cloud9, CodeCommit, CodeBuild, CodePipeline, EC2 instance and CodeDeploy which ensures scalability, maintainability, stability, reliability, and security. This encourages me and my peers to adopt cloud services in the future. Automated deployment plays a significant role when considering AWS for my application to assure on-demand infrastructure support, prevent errors during deployment, and provide the provision of rolling updates. To effectively use AWS to meet growing requirements, we can integrate future additional services like our beloved customers. AWS Lambda to trigger SNS and S3 services. AWS relational database is another option for better database scaling. Implementing Static code analysis to consistently enforce coding standards and best practices across the entire codebase helped in understanding the coding conventions, preventing potential errors and thus improving maintainability. Analysis of the project code through SonarCloud assisted in the identification and resolution of security vulnerabilities, bugs, code smells, and duplication. SalonConnect will consistently improve through user feedback, security enhancements and feature additions. The SalonConnect application has been developed and deployed successfully using the mentioned AWS services, code review service and static analysis tool. The future work for the SalonConnect project includes incorporating HTTPS protocol for better security, using AWS Secrets Manager for secret management services, making UI more user friendly and reinventing it into an Android app. These efforts will ensure that SalonConnect will be a Market leader in the beauty and salon industry by delivering notable value to salon businesses and its beloved customers.

References

[1] D. Webmaster, “The State of EdTech Post-Pandemic - — godashlabs.com,” https://godashlabs.com/2021/04/01/state-of-edtech-post-pandemic/, [Accessed 09-04-2024].

[2] M. Saraswat and R. Tripathi, “Cloud computing: Comparison and analysis of cloud service providers-aws, microsoft and google,” in *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, 2020, pp. 281–285.

[3] K. McLaughlin, “An introduction to the Django ORM — opensource.com,” https://opensource.com/article/17/11/django-orm, [Accessed 09-04-2024].

[4] “What is AWS Elastic Beanstalk? - AWS Elastic Beanstalk — docs.aws.amazon.com,” https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html, [Accessed 09-04-2024].

[5] “SonarCloud Documentation — docs.sonarsource.com,” https://docs.sonarsource.com/sonarcloud/, [Accessed 09-04-2024].

[6] V. Lenarduzzi, F. Pecorelli, N. Saarimaki, S. Lujan, and F. Palomba, “A critical comparison on six static analysis tools: Detection, agreement, and precision,” *Journal of Systems and Software*, vol. 198, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121222002515

[7] “Tutorial - Pylint 3.1.0 documentation — pylint.readthedocs.io,” https://pylint.readthedocs.io/en/stable/tutorial.html, [Accessed 09-04-2024].