



Final Year Project Report
Brokerage Service for Deploying Self-Hosted Applications

Project Advisor: Dr Ghulam Mustafa

Submitted By:

- 1. Sheikh Nauman - F2021408056**
(Bs-Cyber Security)
- 2. Reyan Hassan - F2021065181**
(Bs-Software Engineering)
- 3. Abdul Muqeeet - F2021065146**
(Bs-Software Engineering)
- 4. Zainab Zahid - F2022105062**
(Bs-Information Technology)

Session: 2021-2025

University of Management and Technology
C-II Johar Town Lahore Pakistan

DEDICATION


We got the idea of this project from the increasing use of cloud technologies and the challenges faced by non-technical users to deploy applications on cloud. As the world is rapidly evolving and adapting to the cloud-based solutions the users who lack necessary technical knowledge to deploy applications on cloud are left behind.

To help such users we came up with the idea of a system that streamlines the process of deploying and managing applications on cloud. Basically, our system will make the deployment of application a one click process assisting not only the non-technical but also technical users by making the process faster and easier.

Ultimately, the objective of this project is to make cloud-based solutions more accessible to everyone of their technical background. We will make it easier for the end user to utilize modern cloud services effectively.

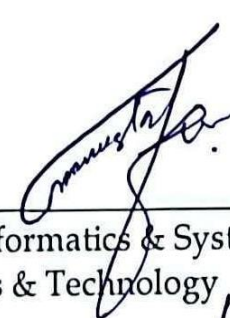
FINAL APPROVAL

- **Head of Department**
Department of Informatics & Systems
School of Systems & Technology
UMT Lahore

For  6/8/2015

- **Director (Final Year Projects-IT)**
Department of Computer Science.
School of Systems & Technology
UMT Lahore

For  6/8/2015

- **Supervisor** 
Department of Informatics & Systems.
School of Systems & Technology
UMT Lahore

6/8/2015.

- **Co-Supervisor** _____

ACKNOWLEDGEMENT

We are thankful to our project advisor Dr Ghulam Mustafa for his guidance throughout our project. He advised us about how we can improve our project ultimately helping us in creating a quality product. Moreover, his support helped us stay focused throughout the project despite the challenges.

Then, we are extremely grateful to our external collaborator and mentor Asad Ullah Rafi for providing real-world insights on cloud-applications and guiding us through his practical knowledge throughout our project. Basically, he supported us in successfully creating a useful system.

Also, we are thankful to our friends for their support. Basically, they helped stay motivated despite all the challenges. Without them this project would have been more difficult to complete

PROJECT SUMMARY

Project Title

Brokerage Service for Deploying Self-Hosted Applications

Objective

This project aims is to develop a system that will make the deployment of applications on cloud easier and a one click process. Our main focus is helping the users who may not have technical knowledge by enabling them to configure and deploy applications on cloud platforms without needing to understand complex concepts such as virtual machines, containers, or cloud infrastructure. By removing the technical hurdles usually involved in deployment, the platform will make it easier for non-technical users to benefit from cloud technologies.

Undertaken by

1. Sheikh Nauman - F2021408056
2. Reyan Hassan - F2021065181
3. Abdul Muqet - F2021065146
4. Zainab Zahid - F2022105062

Supervised by

Dr Ghulam Mustafa

Starting Date

26/07/2024

Completion Date

4/08/2025

Tools Used

1. Next JS
2. MongoDB database
3. Rust Server
4. Terraform Scripts
5. Shell scripting
6. Git / Gitea
7. Docker

Operating System

1. Linux
2. Windows

ABSTRACT

Cloud technologies are widely being adopted these days. They are almost being used in every field like health, education etc but many individuals who do not have a strong technical background find it challenging to deploy applications on cloud. Most of the platforms require users to perform operations which require prior technical knowledge making it overwhelming and time-consuming for someone without prior experience in cloud computing or programming. This project addresses that challenge by offering a user-friendly system that enables the deployment of self-hosted applications to the cloud through a single-click process. Our system will help individuals, small teams, and businesses who want to benefit from cloud services without needing in-depth technical expertise. Using our platform, users can simply define basic preferences, and deploy their application without having to manage the technical steps involved. This not only makes cloud deployment more easier but also saves a lot of valuable time that is required to configure tools necessary for these deployment. The broader aim is to make cloud deployment easier and accessible to a larger and more diverse group of users without requiring them to have technical knowledge.

PLAGIARISM REPORT

University of Management and Technology, Lahore

Similarity Report

Turnitin Originality Report
Brokerage Service for Deploying Self-Hosted Applications by Sheikh Nauman, Reyan
Hassan, Abdul Muqeet and Zainab Zahid

From Quick Submit (Quick Submit)

- Processed on 06-Aug-2025 13:54 PKT
- ID: 2725998747
- Word Count: 7819

Similarity Index

5%

Similarity by Source

Internet Sources:

3%

Publications:

2%

Student Papers:

3%

Sources:

1. 1% match (student papers from 29-Jul-2025)
Submitted to Higher Education Commission Pakistan on 2025-07-29
2. 1% match (student papers from 15-Aug-2024)
Submitted to Higher Education Commission Pakistan on 2024-08-15


Checked by


for Verified by CLO

Note:

- Sometimes the overall similarity index may be a smaller than the repository percentages combined. This would be due to overlapping text within the repositories.
- It is a system generated report.

REVISION CHART

Version	Primary Author(s)	Description of Version	Date Completed
<i>Initial Draft</i>	Abdul Muqet	Initial document prepared for evaluation and suggestion	20 September 2024
<i>Preliminary</i>	Abdul Muqet, Zainab Zahid	Second version based on earlier comments, submitted for concluding reviews	19 January 2025
<i>Final</i>	Abdul Muqet, Zainab Zahid, Reyan Hassan	First complete draft, subjected to formal change tracking and control	15 April 2025

CONTENTS

DEDICATION	I
FINAL APPROVAL.....	II
ACKNOWLEDGEMENT	III
PROJECT SUMMARY.....	IV
ABSTRACT.....	V
PLAGIARISM REPORT.....	VI
REVISION CHART.....	VII
CONTENTS.....	VIII
DEFINITIONS AND ACRONYMS.....	X
LIST OF FIGURES.....	XI
LIST OF TABLES.....	XII
1. INTRODUCTION	13
1.1 MOTIVATIONS.....	13
1.2 PROJECT OVERVIEW	13
1.3 PROBLEM STATEMENT	14
1.4 OBJECTIVES	14
2. DOMAIN ANALYSIS	15
2.1 CUSTOMER.....	15
2.2 STAKEHOLDERS	15
2.3 AFFECTED GROUPS WITH SOCIAL OR ECONOMIC IMPACT	15
2.4 DEPENDENCIES/ EXTERNAL SYSTEMS.....	16
2.5 REFERENCE DOCUMENTS.....	16
2.5.1 <i>Related Projects</i>	17
2.5.2 <i>Feature Comparison</i>	18
3. REQUIREMENTS ANALYSIS.....	19
3.1 REQUIREMENTS	19
3.2 LIST OF ACTORS.....	20
3.2.1. SYSTEM BOUNDARY	20
3.2.2. <i>Actors</i>	21
3.3. LIST OF USE CASES	21
3.4. SYSTEM USE CASE DIAGRAM	22
3.5. EXTENDED USE CASES	23
3.6. USER INTERFACES (MOCK SCREENS)	29
4. SYSTEM DESIGN	32
4.1. SYSTEM ARCHITECTURE DIAGRAM.....	32
4.2. CLASS DIAGRAM.....	33
4.3. SEQUENCE DIAGRAMS	34
4.4. COLLABORATION DIAGRAMS	34
4.5. ERD.....	35
4.6. DATA DICTIONARY	36
5. IMPLEMENTATION DETAILS	37
5.1. DEVELOPMENT SETUP.....	37
5.2. DEPLOYMENT SETUP	37

5.3.	ALGORITHMS	39
5.4.	CONSTRAINTS	39
5.4.1.	<i>Assumptions</i>	39
5.4.2.	<i>System constraints</i>	40
5.4.3.	<i>Restrictions</i>	40
5.4.4.	<i>Limitations</i>	40
6.	TESTING	41
6.1.	EXTENDED TEST CASES	41
7.	TRACEABILITY MATRIX	47
7.1.	RID vs UCID (REQUIREMENTS VS USE CASES)	47
8.	CONCLUSION	48
9.	FUTURE WORK	49
10.	BIBLIOGRAPHY	50
10.1.	BOOKS	50
10.2.	OTHER REFERENCES	50
11.	APPENDIX	51
11.1.	GLOSSARY OF TERMS	51
11.2.	PRE-REQUISITES	51
11.3.	USER GUIDE	51

Definitions and Acronyms

Table 1: Acronyms and Definitions

Acronym/Term	Definition
API	Application Programming Interface is a set of functions and protocols that enable interaction between different software components or systems.
AWS	A cloud computing platform providing services like virtual computing, storage and databases is called Amazon Web Services.
GCP	Google Cloud Platform is a collection of cloud computing services provided by google that runs on the same technologies as its search engine.
SSH	Secure Shell is a secure communication protocol used to execute commands and manage systems remotely over network.
VM	A virtual emulation of a physical system, running operating system and applications just like a physical machine.
IP	Internet Protocol is a set of guidelines that tell how data is transferred over the internet
EIP	Elastic IP. A static, public IPv4 address provided by AWS, which can be associated with EC2 instances.
MFA	Multi-Factor Authentication. A security system that requires more than one form of identification to verify a user's identity.
Terraform	An open-source infrastructure as code tool that allows users to define and provision data centre infrastructure using a declarative configuration language.
API Key	A code passed in by computer programs calling an API to identify the calling program.
ACM	Association for Computing Machinery. An international learned society for computing professionals.
Kubernetes	An open-source system for automating the deployment, scaling, and management of containerized applications.
CI/CD	Continuous Integration and Continuous Deployment. A set of software development practices that involve automatically testing and deploying applications.
Gitea	A self-hosted Git service for version control, allowing collaborative development.
Hetzner	A German-based cloud and hosting service provider, offering affordable cloud infrastructure.
Rust	A systems programming language focused on safety, speed, and concurrency, often used for backend and server-side programming.
EIP	Elastic IP. A static, public IPv4 address designed for dynamic cloud computing.

List of Figures

Figure 1: System Use Case Diagram.....	22
Figure 2: Signup Page GUI	29
Figure 3: Login Page GUI	29
Figure 4: Dashboard GUI	30
Figure 5: Current Deployments GUI.....	30
Figure 6: New Deployment GUI (i)	31
Figure 7: New Deployment GUI (ii)	31
Figure 8: New Deployment GUI (iii)	31
Figure 9: System Architecture Diagram.....	32
Figure 10: Class Diagram.....	33
Figure 11: Sequence Diagram	34
Figure 12: Collaboration Diagram.....	35
Figure 13: MongoDB Schema Design Diagram.....	35

List of Tables

Table 1: Acronyms and Definitions	x
Table 2: List of Stakeholders.....	15
Table 3: Feature Comparison	18
Table 4: Functional Requirements.....	19
Table 5: Non-Functional Requirements	20
Table 6: List of Actors.....	21
Table 7: Signup Use Case.....	23
Table 8: Login Use Case	24
Table 9: Create Deployment Use Case.....	25
Table 10: Configure Deployment Use Case	26
Table 11: Modify Deployment Use Case	27
Table 12: Cloud Integration Use Case.....	28
Table 13: Data Dictionary for Interface Elements.....	36
Table 14: Test Case for Signup	41
Table 15: Test Case for Login	42
Table 16: Test Case for Create Deployment	43
Table 17: Test Case for Configure Deployment.....	44
Table 18: Test Case for Deployment Management	45
Table 19: Test Case to Integrate Cloud	46
Table 20: Requirement Traceability Matrix	47

1. INTRODUCTION

Cloud technologies are widely being adopted these days. Many developers and organizations now depend on cloud platforms to host their websites and applications as they allow them to have access to powerful servers without managing physical hardware.

However, the platforms that allow deploying cloud-based solutions often require user to have technical knowledge to configure the tools, making them difficult for beginners. Setting up and configuring cloud services can take hours or even days, especially for users without technical skills.

Our project aims to make deploying of cloud-based solutions easier by reducing technical steps and providing a user-friendly system with a one-click setup. The goal is to remove barriers and help more people deploy their own applications online with ease.

1.1 Motivations

The purpose of our project is to basically make the deployment of self-hosted applications easier and faster. By automating key steps, the system provides an easy to use interface designed for users who lack technical expertise making it easier for them to use cloud technologies effectively. Overall, our main goal is to make the use of cloud technologies available to everyone despite the background and knowledge.

1.2 Project Overview

We aim to develop a Brokerage Service for deploying self-hosted applications. This platform is designed to make the deploying of cloud application easier for users without technical expertise. Typically, deploying cloud-based applications require a lot of technical skills, user must have basic understanding of configuring tools necessary for deployment. Our system automates those tasks, enabling users to deploy applications with just a few clicks and without needing an in-depth understanding of cloud infrastructure.

1.2.1.1. Problems or Overview Statement

The world is rapidly adopting cloud technologies however, many users find it difficult to use self-hosted applications due to the complexity of cloud services and the need for prior technical knowledge. This project addresses these issues by providing a simple, easy to use platform that streamlines the deployment process and makes cloud technologies accessible to users regardless of their technical background.

1.2.1.2. Customer

The main users of our system are individuals who lack the technical expertise or resources to manually deploy and manage self-hosted applications on cloud platforms.

1.2.1.3. Goals

Our main aim is to make deploying of self hosted applications easier and one-click process, automating the entire process, from selecting resources to launching the application. This system will allow users without technical backgrounds to benefit from cloud computing without needing to understand complex infrastructure management.

1.2.1.1 System Functions

Our platform will provide:

- User registration and authentication
- Cloud resource setup (VM instances, container images, etc.)
- Automated deployment process
- Real-time progress updates and monitoring
- Cloud provider integration (AWS, Azure, Google Cloud)

1.2.1.2 System Attributes

- **Usability:** The platform will feature an intuitive and user-friendly interface, enabling users without technical expertise to navigate the platform with ease.
- **Scalability:** The platform will be able to handle a growing number of users and deployments.
- **Security:** The system will follow best practices for security, including protecting user data and ensuring secure authentication.
- **Reliability:** The platform will provide high availability by minimizing downtime and making sure deployments are dependable.

1.3 Problem Statement

Cloud platforms typically demand a great deal of technical expertise ranging from choosing the right resources to deploying the application. Cloud technology becomes challenging for non-technical users due to its complexity. The purpose of this project is to simplify the deployment of self-hosted applications and make it a one-click process. Basically, less-experienced users will be able to roll out their applications easily, allowing them to take advantage of cloud computing without the burden of configurations.

1.4 Objectives

The goals of our project are:

- Building a system where users can deploy applications to the cloud with a single click.
- Eliminating the need for technical knowledge by automating the technicalities of computers and networks like servers, software, and installation.
- Constructing a platform where users can view the statuses of their applications and manage them with minimal complexity and oversight.
- Ensuring the system can perform tasks safely and reliably, and is secure during common deployment tasks.
- Designing the system whose interfaces are intuitive and the logic is simple enough for a novice to understand in cloud deployment.

2. DOMAIN ANALYSIS

In this chapter we will cover domain analysis of our system.

2.1 Customer

The primary customers of our system are individuals that are ready to deploy self-hosted applications in cloud environments but lack technical skills to do so. Our system will make the deploying of self-hosted applications easier by managing technical processes through automation to provide users without technical knowledge an easy way to deploy their applications. This system is particularly ideal for individuals and organizations that want a cost-effective and efficient means of leveraging cloud technology without expert knowledge.

2.2 Stakeholders

The table below covers the stakeholders that will influence our system:

Table 2: List of Stakeholders

Stakeholder	Role in System
End Users	Individuals and small businesses who will use the platform to deploy their self-hosted applications without needing technical expertise.
Administrator	Administrator is responsible for maintaining user accounts, monitoring deployment processes, and making sure that the system operates correctly.
Cloud Providers	External services that offer the platform to host and deploy applications. They guarantee that cloud resources are accessible and can be scaled according to demand.

2.3 Affected Groups with social or economic impact

Following Groups are impacted by this project:

- **Small Businesses**

Small businesses that lack technical resources will be able to able to deploy their applications on the cloud platforms with less effort. This will not be required to have dedicated IT staff, and they can focus on building their business rather than grappling with complicated deployments.

- **Non-Technical Users**

Non-technical users will be able to easily deploy and manage cloud applications. This easier access will increase adoption of cloud technologies offering new prospects for personal projects or small businesses.

- **Cloud Service Providers**

Providers like AWS, Azure, and Google Cloud will witness increased usage of their services as more non-technical users utilize their cloud infrastructure via the platform, reaching out to more customers and compelling more usage of their cloud infrastructure.

- **Support Teams**

Customer support staff will be benefited from the ease of deployment. Having fewer technical issues to resolve, support staff can spend more time helping customers with operational queries and enhancing customer satisfaction instead of struggling with deployment-related problems.

- **Entrepreneurs and Startups**

Startups and entrepreneurs will be able to take advantage from the platform by streamlining the process of implementing cloud-based applications. The system will allow them to launch products faster, enhancing time-to-market and minimizing the cost of procurement of specialized technical staff.

2.4 Dependencies/ External Systems

Following are the dependencies of this project:

- **Cloud Service Providers**

The platform relies on third-party external cloud service providers to provide the platforms needed for deploying self-hosted applications. The system makes use of these providers and controls resources such as storage, virtual machines, and networking.

- **Terraform Scripts**

Terraform scripts are used for automating the provisioning of cloud infrastructure. It helps in managing the deployment and scaling of resources, ensuring that the deployment process is efficient and repeatable.

2.5 Reference Documents

Following reference documents have been consulted during the analysis phase:

- **AWS Documentation**

Amazon Web Services (AWS) documentation for setting up and managing cloud infrastructure.

URL: <https://aws.amazon.com/documentation/>

- **Hetzner Documentation**

Hetzner provides affordable cloud and dedicated server infrastructure.

URL: <https://docs.hetzner.com/>

- **Rust Documentation**
Official Rust documentation for understanding the programming language used in the backend.

URL: <https://doc.rust-lang.org/>
- **Next.js Documentation**
Next.js documentation for building server-side rendered React applications.

URL: <https://nextjs.org/docs>
- **Terraform Documentation**
Terraform documentation for automating cloud resource provisioning across various cloud providers.

URL: <https://www.terraform.io/docs>

2.5.1 Related Projects

Below are some projects that are competitors to our system:

- **Spinnaker**

Developed by Netflix, Spinnaker is an open-source, multi-cloud continuous delivery platform that enables fast and reliable software releases. It integrates with major cloud providers like AWS, Google Cloud, and Microsoft Azure, providing a robust pipeline management system.

[Spinnaker Documentation](#)
- **CELAR**

The CELAR project, funded by the European Commission, developed an open-source toolkit for automatic, multi-grained resource allocation for cloud applications. It allows users to define deployments of complex multi-tier distributed applications and manage their scalability lifecycle with a single-click deployment.

[CELAR Project Overview](#)
- **CloudCAMP**

CloudCAMP is a GUI-based cloud automation and orchestration framework that helps users deploy and manage services on cloud platforms without requiring domain expertise. It transforms partial specifications into deployable Infrastructure-as-Code using Model-Driven Engineering.

[CloudCAMP Research Paper](#)

2.5.2 Feature Comparison

The following table covers comparison of features of our competitors:

Table 3: Feature Comparison

Sr No.	Comparison Feature	Spinnaker (S)	CELAR (C)	CloudCAMP (CC)	Remarks
1	Multi-cloud Integration	Yes, integrates with AWS, Google Cloud, Azure	Yes, supports multiple cloud providers	Yes, integrates with cloud platforms	All systems support multi-cloud deployment, but Spinnaker is the most robust with extensive cloud integration options.
2	One-Click Deployment	Yes, offers automated deployment pipelines	Yes, supports automatic resource allocation	Yes, provides a GUI for deployment	CELAR and CloudCAMP provide user-friendly deployment interfaces, while Spinnaker focuses on CI/CD pipelines.
3	Resource Scaling	Supports auto-scaling in cloud environments	Supports multi-grained scaling for cloud applications	Provides scaling features through orchestration	All three systems provide scaling features, but CELAR offers more granular scaling options for cloud apps.
4	Infrastructure as Code	Yes, infrastructure as code support through pipelines	Yes, supports Infrastructure-as-Code for deployments	Yes, transforms specifications into deployable code	All systems implement Infrastructure-as-Code, but Spinnaker offers the most mature pipeline for automated deployments.
5	Ease of Use for Non-Technical Users	Moderate, more technical knowledge needed	High, designed for simplified deployments	High, focused on non-technical users	CloudCAMP and CELAR prioritize ease of use for non-technical users, while Spinnaker is more suited for DevOps teams.

3. REQUIREMENTS ANALYSIS

In this chapter we will cover the functional and non-functional requirements of our system.

3.1 Requirements

- Below is the table that covers the functional requirements of our system:

Table 4: Functional Requirements

No.	Requirement Name	Requirement Description
FR1	Create Account	Users should be able to register by providing their personal details, including name, email, username, and password, to create a secure account. The system will validate the information and store it in the database.
FR2	Create Deployment	Users should be able to create the virtual software they want to deploy by selecting appropriate resources such as VM instances, storage, and container images. The system should then handle the creation and deployment process automatically based on the selected options.
FR3	Configure Deployment	Users should be able to configure deployment settings, including selecting VM instances, container images, scaling options, and networking settings to match their requirements.
FR4	Manage Deployment	Users should be able to manage their existing deployments. This includes the ability to rename, edit, or delete deployments as needed. The system will validate these changes and apply them to the cloud environment. Deleting a deployment will remove it from the cloud platform, while renaming will update the associated resources accordingly.
FR5	Integrate with Cloud	The system should enable users to integrate with cloud providers (AWS, Azure, Google Cloud) for provisioning infrastructure automatically. The system will manage resources based on user input, ensuring seamless integration with the chosen cloud provider.
FR6	Generate Scripts	The platform should automatically generate scripts for system configurations (e.g., Terraform scripts) based on user inputs and selected options, ensuring repeatable and automated deployment processes.
FR7	Manage Deployment Status	Users should be able to view, pause, resume, or cancel their deployments. This allows users to manage the progress of their deployments and make necessary adjustments in real-time based on progress updates.

- Below is the table that covers the non-functional requirements of our system:

Table 5: Non-Functional Requirements

No.	Requirement Name	Requirement Description
NFR1	Usability	The system should provide a user-friendly and intuitive interface to simplify user interaction. Non-technical users should be able to navigate and operate the system without needing technical knowledge.
NFR2	Security	Best practices for data protection and user privacy should be implemented, including encryption of sensitive information (e.g., passwords, deployment configurations) and secure authentication mechanisms such as OAuth.
NFR3	Scalability	The system should be scalable to handle a growing number of users and deployments. It should support multiple users simultaneously without performance degradation, ensuring that the platform can grow with user demand.
NFR4	Reliability	The platform should ensure high availability and robustness. The system must be designed to handle unexpected failures, providing error handling mechanisms and minimizing downtime during deployment processes.
NFR5	Performance	The system should offer quick response times, particularly when validating inputs, processing deployment requests, and generating deployment scripts. Response time for user requests should not exceed 5 seconds under normal conditions.
NFR6	Compliance	The system should comply with industry standards and regulations for data security, particularly for user data storage and cloud resource management. This includes ensuring GDPR compliance for European users.

3.2 List of Actors

In this section, we present the list of key actors, define the system boundary, describe all the use cases, and the corresponding UML diagram that represent the system's overall behavior.

3.2.1. System Boundary

The Brokerage Service for Deploying Self-Hosted Applications aims to simplify and automate the process of deploying applications to cloud platforms. The system boundary of this project includes all the features and functionality necessary to facilitate a seamless, user-friendly deployment experience, specifically targeting non-technical users.

The system will:

- **Enable User Registration and Authentication:** Users will be able to create an account and authenticate to access the platform.
- **Configuration of Hosting Requirements:** Users can configure cloud resources (e.g., virtual machines, containers) based on their deployment needs.

- **Automate Deployment:** The platform will automatically handle the deployment of applications on cloud services (AWS, Azure, Google Cloud) without requiring in-depth technical knowledge from users.
- **Cloud Provider Integration:** The system will integrate with popular cloud providers to provision and manage infrastructure based on user-selected configurations.

3.2.2. Actors

Following table covers the main actors of our system:

Table 6: List of Actors

Actor	Role in System
End Users	Individuals or businesses who interact with the platform to deploy self-hosted applications. They register, configure cloud resources, and initiate deployments.
Administrator	A user responsible for managing system operations, including user management, deployment monitoring, and maintaining platform stability. Administrators can also modify or cancel deployments.

3.3. List of use cases

Following section contains the use cases of our system:

- **Sign Up**
Captures the creation of a new user account with necessary information such as email, password, and personal details.
- **Log In**
Allows users to provide account information and gain access to restricted services within the platform.
- **Create Deployment**
Enables users to initiate the deployment process by configuring the necessary resources and cloud settings.
- **Configure Deployment**
Allows users to select and configure deployment settings, including selecting cloud services, VM instances, and network configurations.
- **Modify Deployment**
Provides users the ability to edit, rename, or delete their existing deployments based on changing requirements.

- **Track Deployment Status**

Lets users monitor the real-time progress of their deployments, with updates on completion status and errors if any occur.

- **Cloud Integration**

Facilitates users in linking their cloud accounts (AWS, Azure, Google Cloud) for deployment provisioning.

3.4. System use case diagram

Following is the system use case diagram, which highlights the main actors and their interactions with the system:

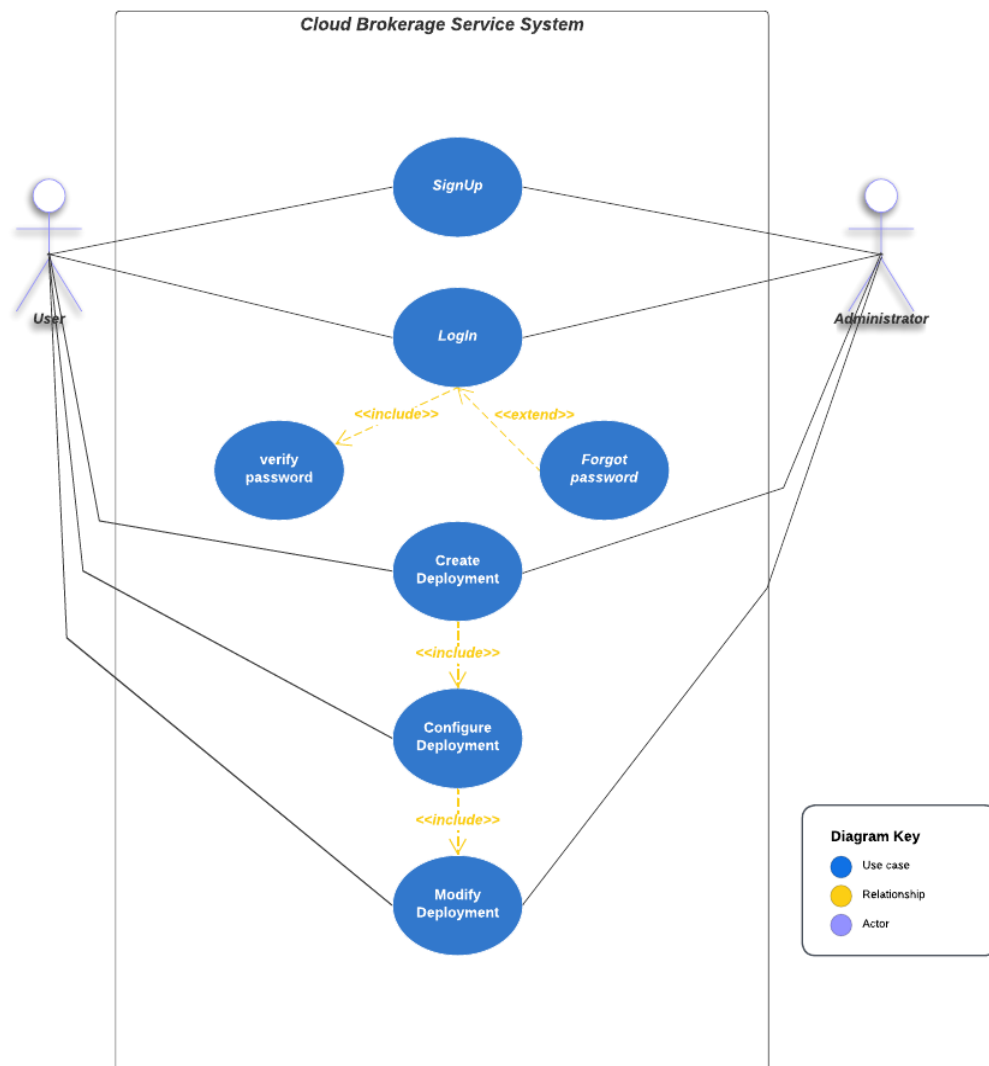


Figure 1: System Use Case Diagram

3.5. Extended use cases

Table 7: Signup Use Case

Created By	Zainab Zahid	Last Updated By	Abdul Muqet
Date Created	13 April 2025	Last Revision Date	14 April 2025
Use Case ID	UC-1	Use Case Name	Sign Up
Actors	User, System	Description	The user creates a new account by providing necessary information such as email, password, and username. The system validates the input and registers the user, sending a verification email upon successful registration.
Trigger	User navigates to the sign-up page and submits the registration form.	Preconditions	User is on the sign-up page.
Postconditions	1. A new user account is created. 2. A verification email is sent to the user.	Normal Flow	1. User navigates to the sign-up page. 2. User enters email, password, and confirms the password. 3. User accepts the terms and conditions. 4. User clicks the "Sign Up" button. 5. System validates the inputs (email format, password strength). 6. If valid, system creates the account and sends a verification email.
Alternative Flows	1. Invalid Email Format: System displays an error message for the email field. 2. Email Already Exists: System informs the user that the email is already registered. 3. Weak Password: System prompts the user to enter a stronger password. 4. Password Mismatch: System asks the user to re-enter matching passwords. 5. Unaccepted Terms: System prevents sign-up until terms are accepted.	Exceptions	1. Invalid Email Format: If the email is in an invalid format, the system prompts the user to correct the email.
Includes	"Validate Inputs" is included as part of the user registration process.	Frequency of Use	This use case will be executed once per user.
Special Requirements	Secure user data encryption during registration and verification processes.	Assumptions	The user has an active internet connection and can access the platform's registration page.
Notes and Issues	None		

Table 8: Login Use Case

Created By	Zainab Zahid	Last Updated By	Abdul Muqet
Date Created	13 April 2025	Last Revision Date	14 April 2025
Use Case ID	UC-2	Use Case Name	Login
Actors	User, System	Description	The user logs into the system using their email and password. The system validates the credentials and either logs the user in or displays an error message.
Trigger	The user submits their login credentials on the login page.	Preconditions	User is on the login page.
Postconditions	1. User is logged in and redirected to the dashboard. 2. If invalid, an error message is displayed.	Normal Flow	1. User navigates to the login page. 2. User enters their email and password. 3. User clicks the "Login" button. 4. System validates the credentials (email format, password matching). 5. If valid, system logs the user in and redirects to the dashboard.
Alternative Flows	1. Invalid Email Format: System displays an error message prompting the user to enter a valid email. 2. Incorrect Password: System displays an error message indicating the password is incorrect. 3. Forgot Password: 1. User clicks the "Forgot Password" link. 2. User enters their email address. 3. System validates the email format. 4. If valid, system generates a password reset link. 5. System sends the link to the user's email address. 6. If the email is invalid, system displays an error message.	Exceptions	1. Incorrect Password: If the password is incorrect, system prompts the user to try again.
Includes	"Forgot Password" flow is included when the user cannot remember their login credentials.	Frequency of Use	This use case will be executed frequently by users upon accessing the platform.
Special Requirements	Secure authentication (using OAuth or other standards) for password storage and validation.	Assumptions	The user has registered and has valid credentials to access the platform.
Notes and Issues	None		

Table 9: Create Deployment Use Case

Created By	Zainab Zahid	Last Updated By	Abdul Muqet
Date Created	13 April 2025	Last Revision Date	14 April 2025
Use Case ID	UC-3	Use Case Name	Create Deployment
Actors	User, Administrator, System	Description	The user initiates the deployment process by providing necessary configuration details such as project name, server, and cloud provider. The system validates the input and begins the deployment process.
Trigger	User navigates to the deployment page and submits the form with configurations.	Preconditions	User is logged in and has access to the deployment feature.
Postconditions	1. A deployment request is created. 2. The deployment starts processing.	Normal Flow	1. User navigates to the deployment page. 2. User fills out the form (e.g., project name, service, server, region, volume size). 3. User submits the deployment request. 4. System validates inputs. 5. If validation passes, the system processes the deployment and provides real-time progress updates.
Alternative Flows	1. Invalid Input: The system displays an error message, and the user must correct the input. 2. Cancel Deployment: The user cancels the deployment, and the system discards the form data. 3. Save as Draft: The user saves the deployment as a draft for later completion. 4. Network Error: The system notifies the user and prompts them to retry.	Exceptions	1. Invalid Configuration: The system prompts the user to correct invalid configurations before proceeding.
Includes	"Input Validation" is included as part of the deployment creation process.	Frequency of Use	This use case will be executed frequently when users create new deployments.
Special Requirements	Deployment configuration should handle large-scale infrastructure provisioning efficiently.	Assumptions	The user has access to the required cloud services for deployment.
Notes and Issues	None		

Table 10: Configure Deployment Use Case

Created By	Zainab Zahid	Last Updated By	Abdul Muqet
Date Created	13 April 2025	Last Revision Date	14 April 2025
Use Case ID	UC-4	Use Case Name	Configure Deployment
Actors	User, Cloud Brokerage System	Description	The user configures the deployment settings, such as selecting the cloud provider, VM instances, and networking configurations.
Trigger	User selects deployment configuration options from the interface.	Preconditions	User is logged in and has access to the deployment configuration feature.
Postconditions	1. Deployment settings are saved. 2. The system is ready for execution.	Normal Flow	1. User navigates to the Configure Deployment section. 2. User selects the cloud environment and instance configuration. 3. User configures networking, scaling, and final deployment settings. 4. System validates configurations and saves settings if valid.
Alternative Flows	1. Invalid Configuration: System prompts the user to fix errors in the configuration. 2. Networking Conflicts: The system warns the user and provides suggestions to resolve networking issues.	Exceptions	1. Scaling Limitations: System alerts the user if the scaling settings exceed platform limits.
Includes	"Validate Settings" is included in the configuration process.	Frequency of Use	This use case will be executed every time a user configures a deployment.
Special Requirements	The system should ensure seamless integration with multiple cloud providers.	Assumptions	The user has knowledge of cloud infrastructure requirements.
Notes and Issues	None		

Table 11: Modify Deployment Use Case

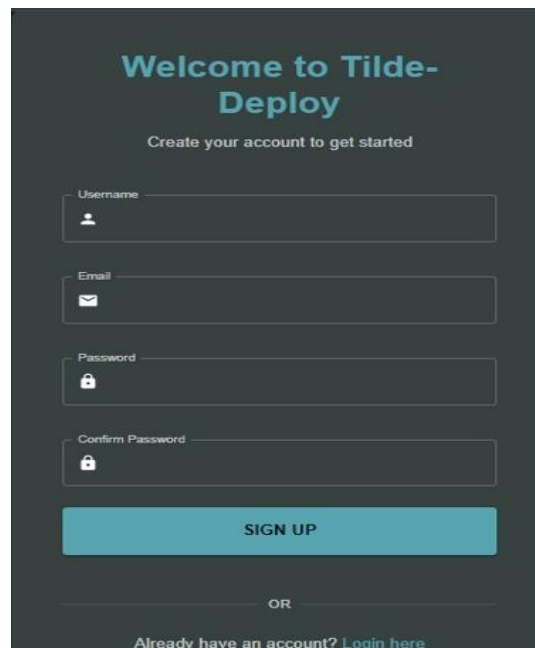
Created By	Zainab Zahid	Last Updated By	Abdul Muqeeet
Date Created	13 April 2025	Last Revision Date	14 April 2025
Use Case ID	UC-5	Use Case Name	Modify Deployment
Actors	User, Cloud Provider, Admin (optional)	Description	The user modifies the cloud deployment by renaming, editing, or deleting it. The system validates the modifications and applies them.
Trigger	User selects a deployment to modify.	Preconditions	The user is authenticated, and a deployment exists.
Postconditions	1. The deployment is successfully modified or deleted. 2. The system logs the activity for auditing.	Normal Flow	1. User selects a deployment. 2. System displays modification options (Rename, Edit, Delete). 3. User chooses an action. 4. System applies changes to deployment. 5. The system logs the modification activity.
Alternative Flows	1. Invalid Rename Attempt: If the new name is taken, an error is displayed. 2. Configuration Validation Fails: System prompts the user to correct configuration settings. 3. Delete Action Cancelled: No changes are made if the user cancels.	Exceptions	1. Configuration Validation Fails: System rejects changes and prompts for correction.
Includes	"Modify Configuration" is included in the modification process.	Frequency of Use	This use case will be executed when users need to modify an existing deployment.
Special Requirements	Modifications should be reversible (with rollback) for up to a limited time.	Assumptions	The user has sufficient permissions to modify the deployment.
Notes and Issues	None		

Table 12: Cloud Integration Use Case

Created By	Zainab Zahid	Last Updated By	Abdul Muqet
Date Created	13 April 2025	Last Revision Date	14 April 2025
Use Case ID	UC-6	Use Case Name	Cloud Integration
Actors	End User, Cloud Brokerage Platform, Cloud Service Provider	Description	The system allows users to integrate their cloud accounts (AWS, Google Cloud, etc.) by providing authentication details and API keys.
Trigger	User navigates to the cloud integration page and initiates the setup process.	Preconditions	User is logged in and has necessary permissions to configure cloud integrations.
Postconditions	<ol style="list-style-type: none">1. The cloud account is successfully linked to the platform.2. The system securely stores credentials for future use.	Normal Flow	<ol style="list-style-type: none">1. User navigates to the Cloud Integration section.2. User selects the cloud provider.3. User enters API key and other credentials.4. System validates and stores credentials.5. The system links the cloud account to the platform.
Alternative Flows	<ol style="list-style-type: none">1. Invalid Credentials: System prompts the user to re-enter the correct authentication details.2. Authorization Failure: System displays an error and suggests troubleshooting steps.	Exceptions	<ol style="list-style-type: none">1. API Rate Limits: If the cloud provider limits API calls, the system queues requests and informs the user.
Includes	"Validate Cloud Integration" is included during the configuration process.	Frequency of Use	This use case will be executed whenever the user integrates a new cloud account.
Special Requirements	Secure encryption for cloud credentials and access management.	Assumptions	The user has valid cloud accounts and API keys for integration.
Notes and Issues	None		

3.6. User interfaces (mock screens)

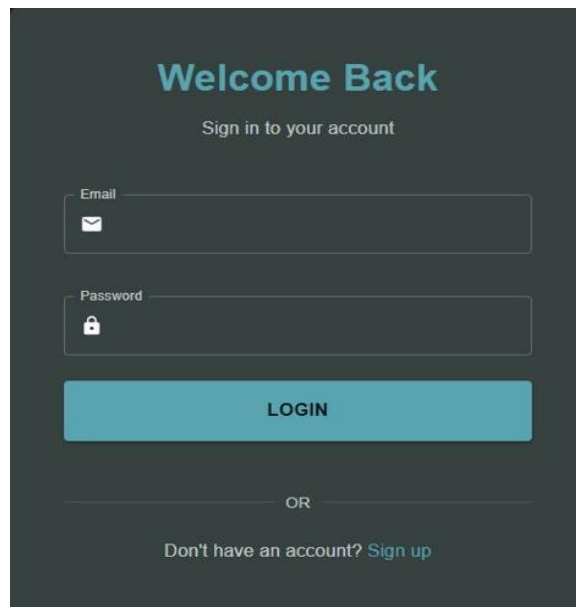
- The figure below displays the Gui screen of the system's signup screen:



The image shows a dark-themed user interface for a signup page. At the top, it says "Welcome to Tilde-Deploy" in a light blue font, followed by "Create your account to get started" in a smaller white font. Below this are four input fields: "Username" with a person icon, "Email" with an envelope icon, "Password" with a lock icon, and "Confirm Password" with a lock icon. A large teal button labeled "SIGN UP" is positioned below the input fields. Underneath the button is a horizontal line with the word "OR" in the center. At the bottom, it says "Already have an account? [Login here](#)" in a light blue font.

Figure 2: Signup Page GUI

- The figure below displays the Gui screen of the system's login screen:



The image shows a dark-themed user interface for a login page. At the top, it says "Welcome Back" in a light blue font, followed by "Sign in to your account" in a smaller white font. Below this are two input fields: "Email" with an envelope icon and "Password" with a lock icon. A large teal button labeled "LOGIN" is positioned below the input fields. Underneath the button is a horizontal line with the word "OR" in the center. At the bottom, it says "Don't have an account? [Sign up](#)" in a light blue font.

Figure 3: Login Page GUI

- The figure below displays the Gui screen of the system's dashboard screen:

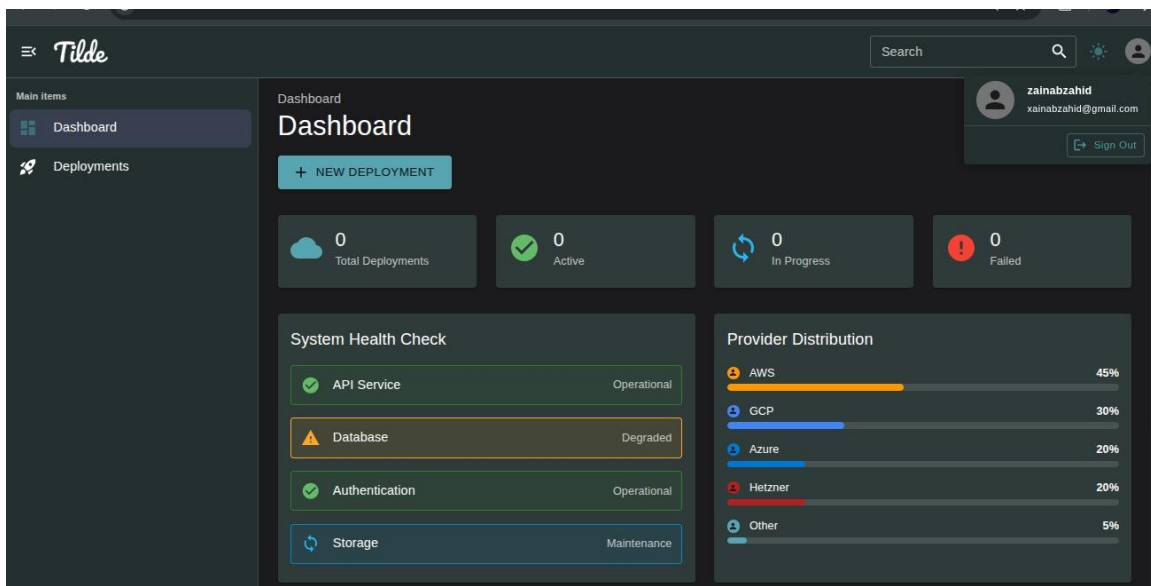


Figure 4: Dashboard GUI

- The figures below display the Gui screen of the system's current deployment screen:

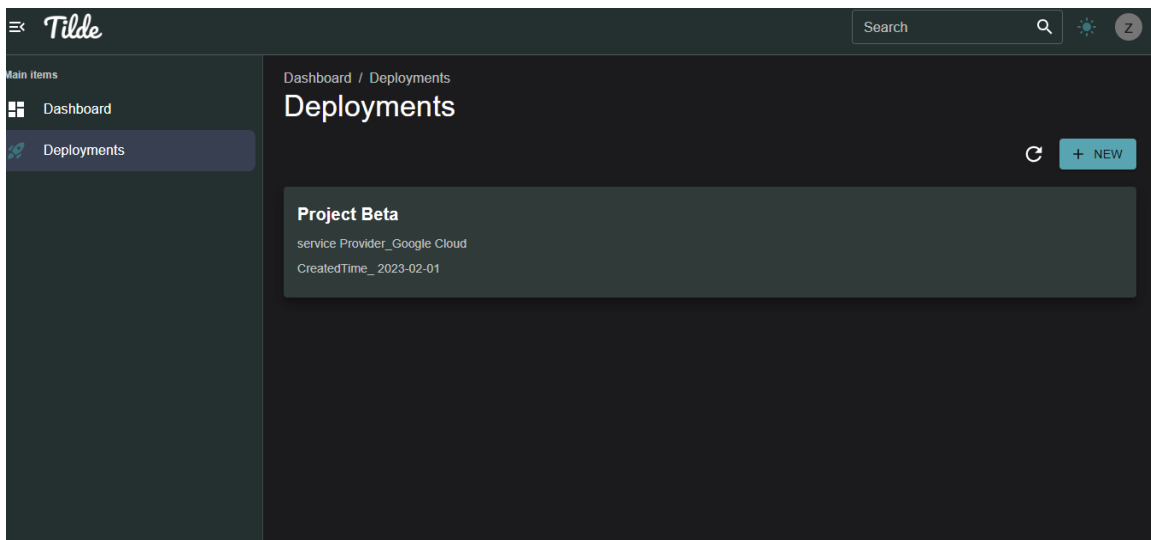


Figure 5: Current Deployments GUI

- The figure below displays the Gui screen of the system's new deployment screen:

Project Details

Project Name

Select a Service

WordPress Site Game GitHub Code

Shopify Site Cloud Hosting Database Service

CI/CD Pipelines Container Application Static Website

CDN & Caching Monitoring & Logging E-commerce Solution

Select a Server

Figure 6: New Deployment GUI (i)

Select a Server

☐ AWS

☐ Azure

☐ GCP

Select a Region

Region

Storage Configuration

Volume Size (GiB)

0

Figure 7: New Deployment GUI (ii)

Region

Storage Configuration

Volume Size (GiB)

0

Select IP Address Type

☐ Reserved Static IP

☐ Dynamic IP

SSH Key Configuration

☐ Generate SSH Key

☐ Input SSH Key

DEPLOY

Figure 8: New Deployment GUI (iii)

4. SYSTEM DESIGN

This section provides a complete overview of the system's design through various diagrams.

4.1. System Architecture Diagram

In this section we will explore the system architecture diagram of our platform:

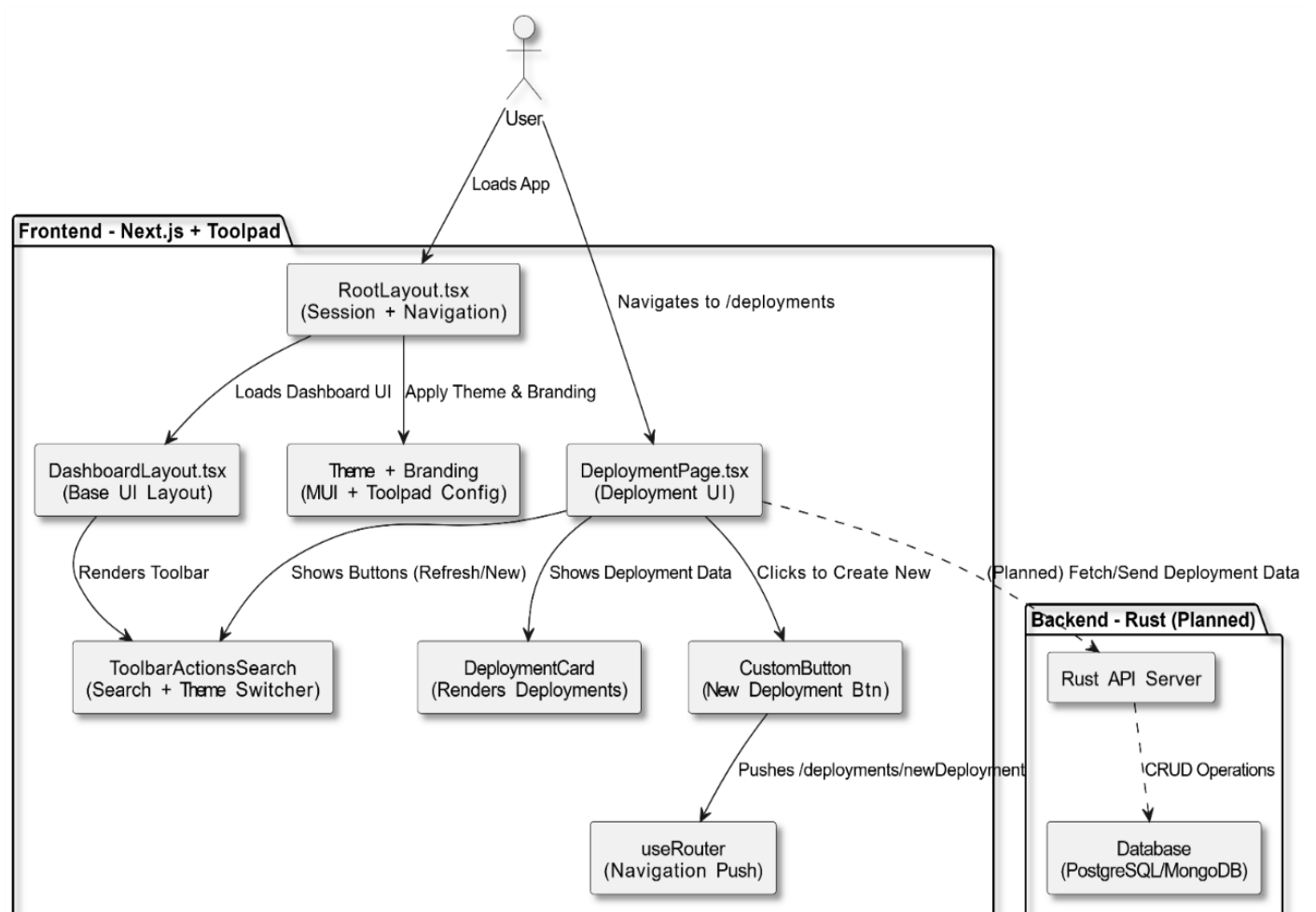


Figure 9: System Architecture Diagram

4.2. Class Diagram

This figure covers the class diagram of our system:

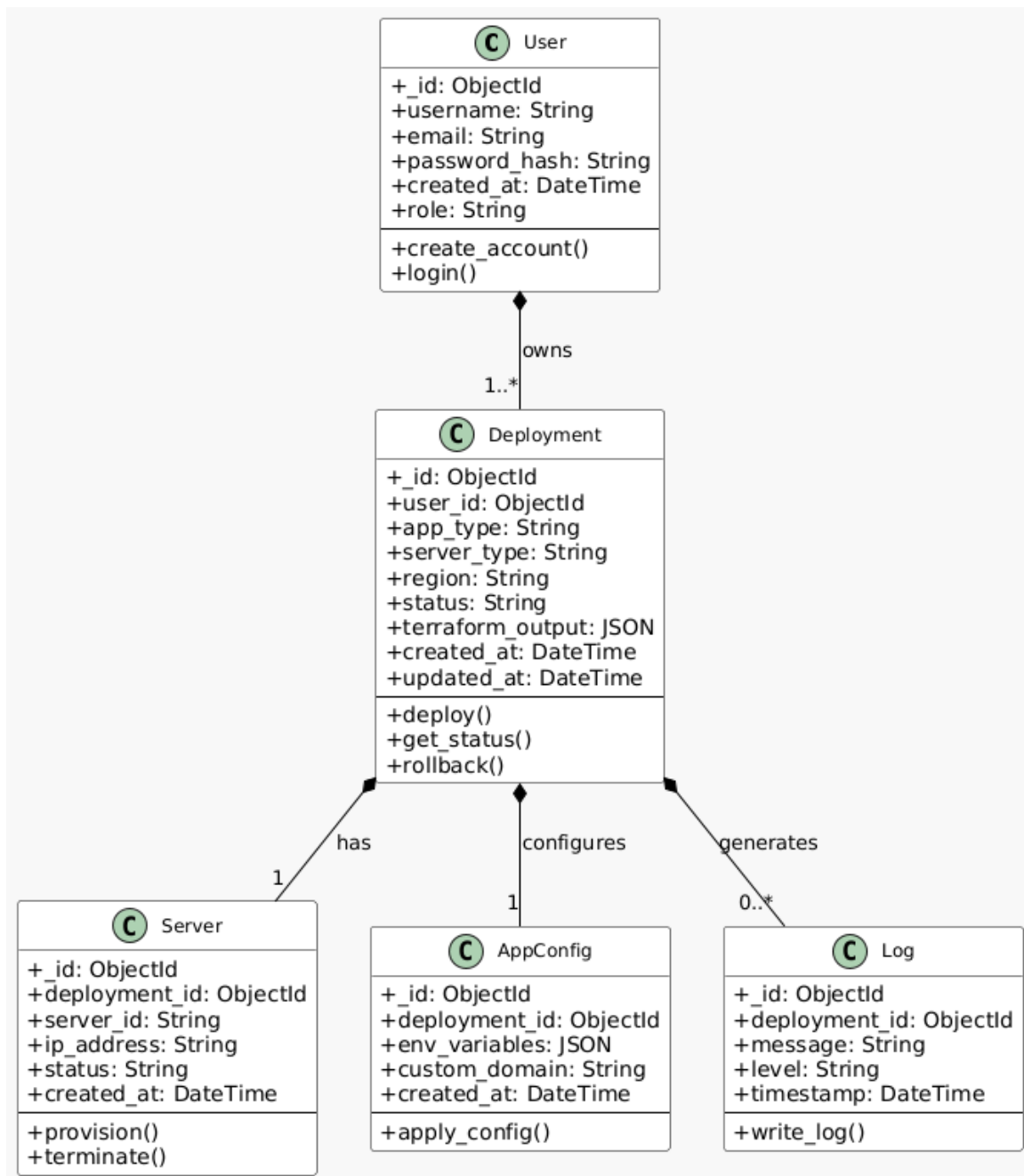


Figure 10: Class Diagram

4.3. Sequence Diagrams

This figure covers the sequence diagram of our system:

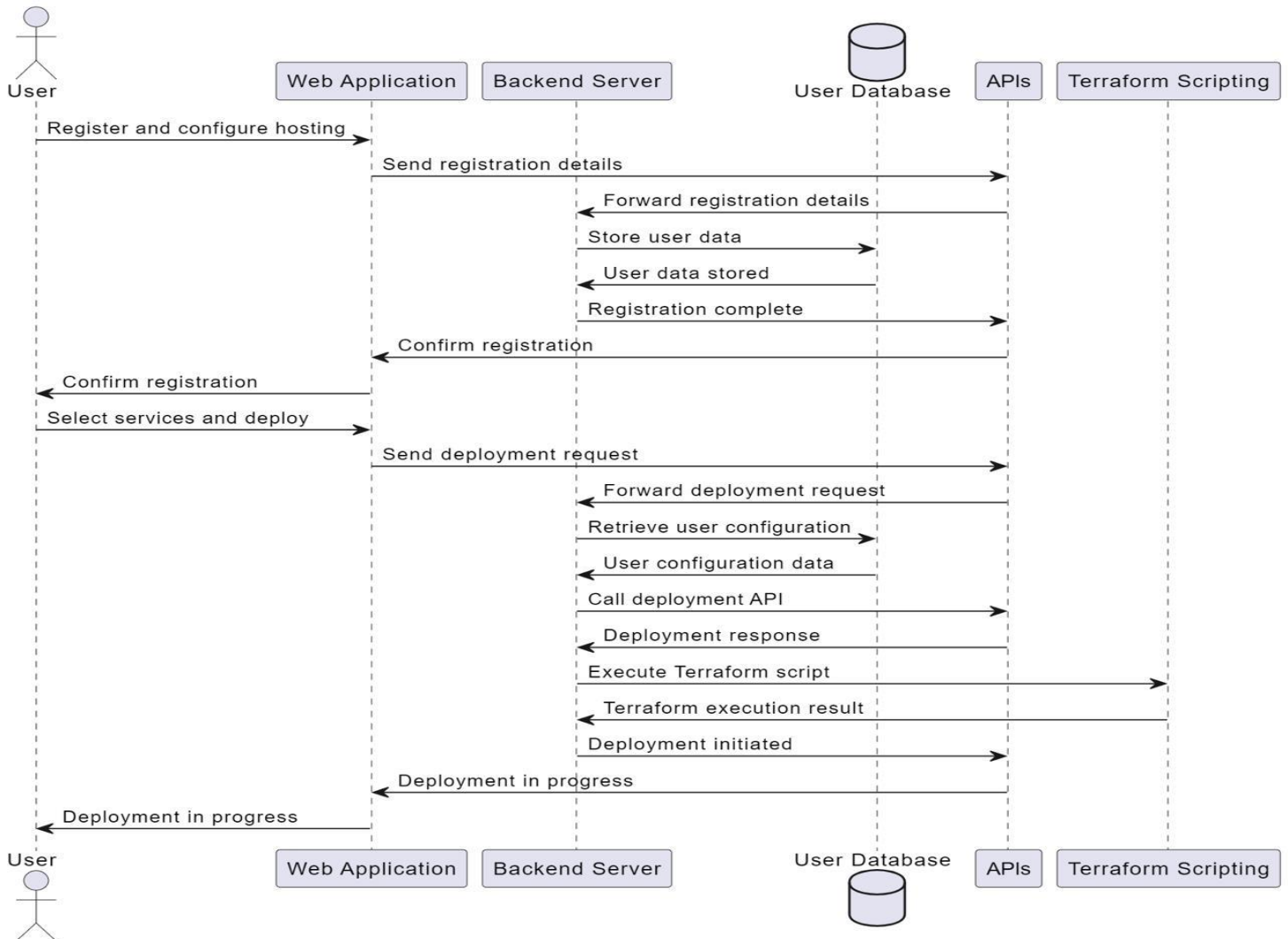


Figure 11: Sequence Diagram

4.4. Collaboration Diagrams

The figure below covers the collaboration diagram of our system:

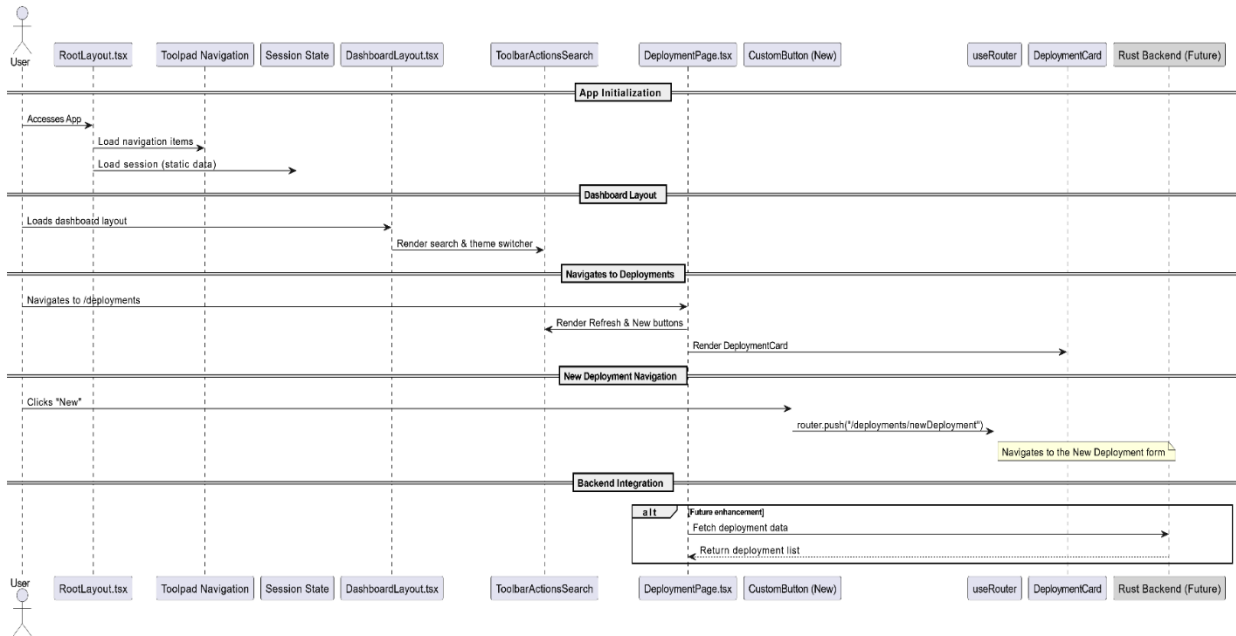


Figure 12: Collaboration Diagram

4.5. ERD

This figure covers the ERD diagram of database of our system:

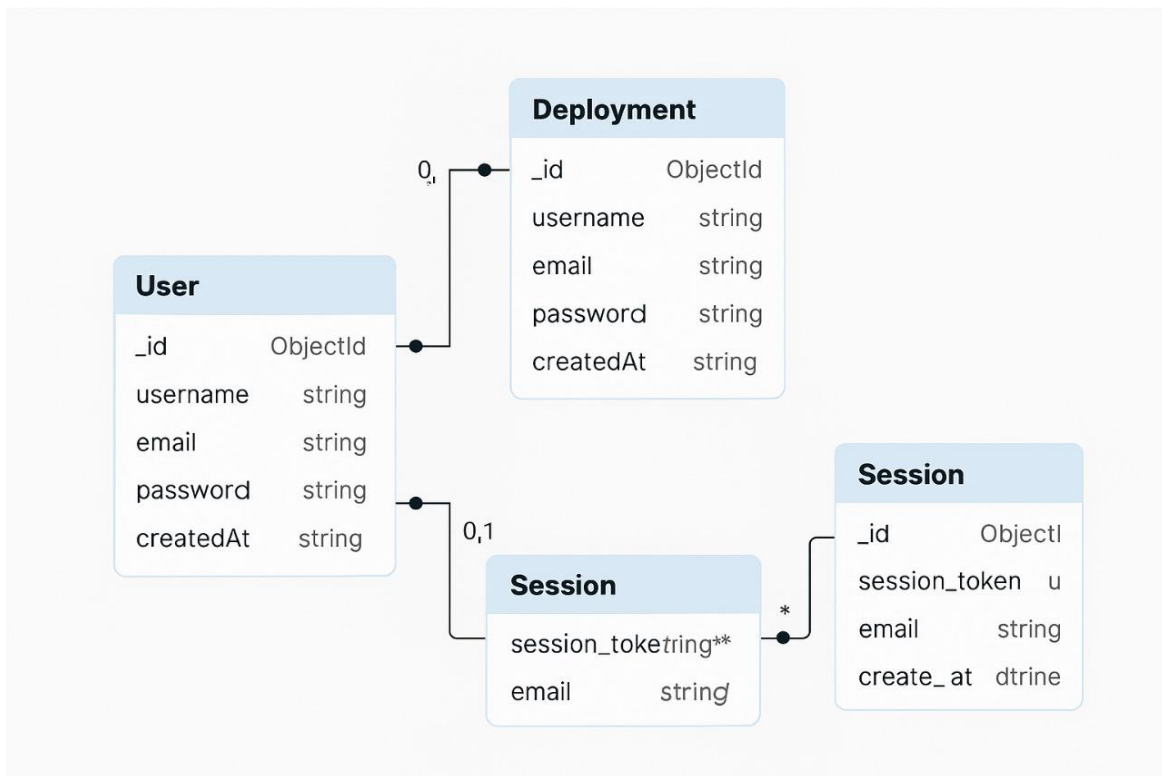


Figure 13: MongoDB Schema Design Diagram

4.6. Data Dictionary

The table below covers the data dictionary for interface elements of our system:

Table 13: Data Dictionary for Interface Elements

Element Name	Type	Validation	Mandatory	Remarks
Dashboard Link	Button/Link	-	Yes	Takes the user to the Dashboard page.
Deployments Link	Button/Link	-	Yes	Navigates to the Deployments page.
Project Beta	Text/Label	-	Yes	Displays project details, including service provider and creation time.
New Deployment Button	Button	-	Yes	Initiates the creation of a new deployment.
Search Input	Text Input	-	No	Allows the user to search deployments.
Sign Up Button	Button	-	Yes	Submits the sign-up form to create a new user account.
Username Input	Text Input	-	Yes	Field for the user to input their username.
Email Input	Text Input	Valid email format	Yes	Ensures the email is valid.
Password Input	Password Field	Minimum 8 characters, must contain one uppercase and one number	Yes	Ensures strong password security.
Confirm Password Input	Password Field	Must match the Password field	Yes	Confirms the user's password entry.
Login Button	Button	-	Yes	Submits the login form with the user credentials.
Email Input (Login)	Text Input	Valid email format	Yes	Used for login.
Password Input (Login)	Password Field	-	Yes	Used for login.
Deploy Button	Button	-	Yes	Initiates deployment once configurations are complete.
Select Server Dropdown	Dropdown	Must select one option (AWS, Azure, GCP)	Yes	User selects the cloud service provider.
Region Dropdown	Dropdown	Must select a region	Yes	User selects the region for deployment.
Volume Size Input	Number Input	Numeric value	Yes	Defines storage size in GiB for deployment.
Select IP Address Type	Radio Button	Must choose between Static or Dynamic IP	Yes	User selects IP address type for the deployment.
SSH Key Configuration	Radio Button	Must select either Generate SSH or Input SSH Key	Yes	Allows the user to configure SSH keys for secure deployment access.

5. IMPLEMENTATION DETAILS

5.1. Development Setup

The development of the Brokerage Service for Deploying Self-Hosted Applications involves using several tools and technologies, each playing a critical role in building the platform:

- **Next.js:** This will be used for building the front-end of the platform. It provides a framework to build server-side rendered React applications, allowing us to efficiently handle deployment configuration and management.
- **MongoDB:** A NoSQL database that will be used to store user data, deployment configurations, and system logs. MongoDB's flexibility will allow us to scale and handle large amounts of data, making it an ideal choice for our platform.
- **Rust Server:** The back-end of the platform will be powered by Rust. Rust will be used to build high-performance, concurrent, and secure services. It will handle complex logic for deployment automation and cloud provider interactions.
- **Terraform Scripts:** Terraform will be used to automate the infrastructure setup. Terraform ensures consistent and repeatable deployment across multiple cloud providers (AWS, GCP, Azure, Hetzner).
- **Shell Scripting:** This will be used for automating certain command-line operations related to deployment, configuration, and integration with cloud providers.
- **Git/Gitea:** This will be used for version control and collaborative development. Gitea will serve as a self-hosted Git service to manage the project's codebase.
- **Docker:** Docker will be used for containerizing the application, ensuring that it can run consistently across different environments and cloud platforms.

5.2. Deployment setup

The software will be deployed on multiple cloud platforms to achieve flexibility and scalability for end-users. The major platforms employed for deployment include Google Cloud Platform (GCP), Amazon Web Services (AWS), Microsoft Azure, and Hetzner, offering cheaper hosting services.

5.2.1. Deployment Process

- **Cloud Providers:** Users will be provided with a choice of selecting their desired cloud service provider from the list consisting of AWS, GCP, Azure, and Hetzner.
 - **AWS, GCP, and Azure** will be fully integrated with Terraform scripts for provisioning resources.
 - **Hetzner** integration will allow users to leverage its cost-effective cloud services while maintaining high-performance infrastructure for deployments.

5.2.2. Challenges Faced

5.2.2.1. Cloud Provider Integration

- At first, it was difficult to integrate Hetzner into the system because there is no native support for Terraform providers. We shall employ a community-supported Terraform provider for Hetzner and make it system-compatible.
- **Solution:** Hetzner cloud setup shall be successfully established by bringing together the API and customizing Terraform scripts to manage Hetzner's resource provisioning.

5.2.2.2. Deployment Failures with Hetzner

- There will be certain provisioning storage and network resource-related issues based on Hetzner's custom API settings. The IP allocation configuration will have more steps in Hetzner than in AWS or GCP.
- **Solution:** We will customize the Terraform scripts to handle specific cloud configurations for Hetzner, ensuring that all configurations align with Hetzner's platform, and we will add validation for Hetzner-specific resources like storage and IP management.

5.2.2.3. Network Configuration

- Hetzner has stricter firewall and networking rules, which initially blocked automated deployment scripts. We will adjust the system's handling of IP configurations to support both dynamic and static IPs on Hetzner.
- **Solution:** Adjustments will be made to the network configuration scripts, and Hetzner-specific firewall rules will be added to ensure smooth deployment.

5.2.3. Final Deployment Setup

- **Terraform Integration:** Terraform will be used to automate the provisioning of resources (VMs, storage, networking) on AWS, Azure, GCP, and Hetzner. The cloud platform selection will be easy for users, and deployment configuration will be standardized for all cloud platforms.
- **Real-time Updates:** Users will be able to see the status of their deployment in real-time using the UI, indicating whether the deployment is in progress, successful, or failed.
- **User Interface:** The platform will provide a simple and user-friendly interface where users can configure their cloud infrastructure (select provider, region, server type) and initiate deployments with a single click.
- **Cloud Provider Flexibility:** The flexibility of choosing from multiple cloud providers will allow the user to select the one that best suits their requirements in terms of cost, performance, and location

5.3. Algorithms

In this section, we highlight a few critical algorithms that will be defined or improved during the development process:

5.3.1. Deployment Configuration Validation Algorithm

- This algorithm will ensure that all necessary inputs for a deployment are valid before submission. It will check if the user has selected the correct cloud provider, region, and configured IP addresses, and validate the required fields for each option (e.g., SSH key, storage size).
- **Improvement:** We will add custom validation rules based on the selected cloud provider to ensure that configurations are compatible with the respective cloud API.

5.3.2. Data Validation Algorithm

- The algorithm will take care of validating the input data on the backend. This will check that the data from the frontend is correct before the system processes them. Validating the data in this way is important to make sure that the data sent to terraform scripts is correct.

5.3.3. Deployment Progress Tracking Algorithm

- Whenever a deployment is initiated, this algorithm will keep monitoring the progress of the deployment process and display real-time progress.

5.3.4. Terraform Integration Algorithm

- Deals with the task of sending data to Terraform and getting notified of system instances updates to maintain the infrastructure according to needs. It will also deal with any form of error or problem during Terraform run.
- **Improvement:** We will add a piece of logic that deals with failures in real-time, providing users with actionable feedback in case of errors happening when provisioning the cloud.

5.4. Constraints

In this section, we highlight a few critical constraints that will be crucial for using our system:

5.4.1. Assumptions

- The system assumes that all users have basic internet connectivity and the ability to access cloud platforms (AWS, Azure, GCP, Hetzner).
- Users will have access to their cloud provider credentials and necessary permissions (e.g., API keys, SSH keys).
- Cloud services (AWS, GCP, Azure, Hetzner) are operational and accessible at the time of deployment.
- Users will have basic understanding or willingness to configure basic deployment settings

5.4.2. System constraints

- **Resource Limits:** The system will be constrained by the resource limits imposed by the cloud providers (e.g., maximum number of VM instances, storage limits).
- **API Rate Limits:** Cloud services will impose API rate limits, which might delay deployment if the limits are exceeded. The system will be designed to handle retries, but there may be delays under heavy load.

5.4.3. Restrictions

- **User Permissions:** Only authenticated users with valid credentials will be able to deploy and manage applications. The platform will restrict deployment actions to users with the necessary roles (e.g., Admin, User).
- **Deployment Type:** The system will not support multi-cloud deployments in a single action. Users will have to select one cloud provider for each deployment.

5.4.4. Limitations

- **Complex Configurations:** The system will be designed to automate simple deployment configurations. For more complex cloud infrastructure setups, users may need to configure resources manually.
- **Real-Time Monitoring:** The system will provide basic progress tracking for deployments but will not offer in-depth, real-time monitoring of all system activities (e.g., server performance, resource usage during deployment).
- **No Multi-Region Deployments:** While users can choose a cloud provider and a region, multi-region or cross-cloud deployments will not be supported in this version.

6. TESTING

Below section contains the test cases for our system's functionalities.

6.1. Extended Test Cases

Table 14: Test Case for Signup

Test Case ID	TC-1	Test Design By	Abdul Muqet
Test Module Name	Sign-Up	Test Design Date	14-4-2025
Test Priority	High	Test Executed By	Reyan Hassan
Test Title/Name	To test the sign-up functionality	Test Executed Date	29-7-2025
Description	Test the sign-up functionality by creating a new user account.		

Pre-Condition	The user is on the sign-up page and has not created an account yet.
Dependencies	None

Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to sign-up page	-	User should be redirected to the sign-up page.	As Expected	pass	
2	Fill in the sign-up form	username: Testuser, email:testuser@gmail.com, password:password	The system should accept valid data and allow proceeding.	As Expected	pass	
3	Click on Sign Up button	-	The system should create a new account and send a verification email.	As Expected	pass	
4	Check email for verification	Email: testuser@gmail.com	The user should receive an email to verify their account.	As Expected	pass	

Post Condition	The user is on the sign-up page and has not created an account yet.
-----------------------	---

Table 15: Test Case for Login

Test Case ID	TC-2	Test Design By	Abdul Muqheet
Test Module Name	User Authentication	Test Design Date	14-4-2025
Test Priority	High	Test Executed By	Reyan Hassan
Test Title/Name	To test the login functionality	Test Executed Date	29-4-2025
Description	Test the login functionality with valid credentials.		

Pre-Condition	The user has already signed up and verified the account.
Dependencies	None

Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	-	User should be redirected to the login page.	As Expected	Pass	
2	Enter credentials	email: testuser@gmail.com , password:password	The system should authenticate the credentials.	As Expected	Pass	
3	Click on Login button	-	The system should log in the user and redirect to the dashboard.	As Expected	Pass	

Post Condition	The user is logged in and redirected to the dashboard.
-----------------------	--

Table 16: Test Case for Create Deployment

Test Case ID	TC-3	Test Design By	Abdul Muqheet
Test Module Name	Create Deployment	Test Design Date	14-4-2025
Test Priority	High	Test Executed By	Reyan Hassan
Test Title/Name	To test the creation of a new deployment	Test Executed Date	20-07-2025
Description	Test creating a new deployment with valid inputs.		

Pre-Condition	User is logged in and has access to the deployment page.
Dependencies	Valid cloud provider accounts and sufficient permissions.

Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to deployment page	-	User should be redirected to the deployment configuration page.	As Expected	Pass	
2	Select a cloud provider	Hetzner	The user selects AWS from available options.	As Expected	Pass	
3	Fill in deployment details	project_name: TestProject1, selected_service:AWS, selected_server: cx22, region: fsn1, volume_size: 10, ip_option: dynamic, ssh_key_option:existing, ssh_key: ssh-rsa AAAAB3NzaC1yc2EAAA ADAQABAAQ7", terraform_template: hetzner	The system should accept valid configuration.	As Expected	Pass	
4	Select IP address configuration	IP Address: 192.168.1.100	The system should accept the IP address configuration and associate it with the deployment.	As Expected	Pass	
5	Click on Deploy button	-	The system should process the deployment request and display a progress bar.	As Expected	Pass	

Post Condition	Deployment is initiated, and the user can view the progress.
-----------------------	--

Table 17: Test Case for Configure Deployment

Test Case ID	TC-4	Test Design By	Abdul Muqet
Test Module Name	Deployment Configuration	Test Design Date	14-4-2025
Test Priority	Medium	Test Executed By	To be decided
Test Title/Name	To test the deployment configuration functionality	Test Executed Date	To be decided
Description	Test configuring the cloud deployment settings before deployment.		

Pre-Condition	User is logged in and has accessed the configuration page for deployment.
Dependencies	User has initiated a deployment.

Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to configure deployment	-	User should be on the deployment configuration page.			
2	Select deployment environment	AWS	User selects the cloud provider successfully.			
3	Configure storage and IP options	Storage Size: 100 GiB IP: 192.168.1.100	System should validate input and confirm settings.			
4	Confirm configuration	-	System should save and validate the configuration for deployment.			

Post Condition	Configuration is saved, and deployment can proceed.
-----------------------	---

Table 18: Test Case for Deployment Management

Test Case ID	TC-5	Test Design By	Abdul Muqet
Test Module Name	Deployment Management	Test Design Date	14-4-2025
Test Priority	High	Test Executed By	Reyan Hassan
Test Title/Name	To test the modification of an existing deployment	Test Executed Date	30-7-2025
Description	Test the modification (delete) of an existing deployment.		

Pre-Condition	User is logged in, has initiated a deployment, and the deployment exists.
Dependencies	Valid deployment already exists.

Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to deployments page	-	User should be on the deployments page.	As Expected	Pass	
2	Select deployment to modify	-	The user selects a deployment to delete.	As Expected	Pass	
3	Delete deployment	-	The system should ask for confirmation and delete the deployment if confirmed.	As Expected	Pass	

Post Condition	Deployment is successfully deleted and deployment page refreshed
-----------------------	--

Table 19: Test Case to Integrate Cloud

Test Case ID	TC-6	Test Design By	Abdul Muqeet
Test Module Name	Cloud Integration	Test Design Date	14-4-2025
Test Priority	High	Test Executed By	Abdul Muqeet
Test Title/Name	To test cloud integration functionality	Test Executed Date	30-7-2025
Description	Test the integration of cloud accounts for deployment.		

Pre-Condition	User is logged in and has access to cloud integration settings.
Dependencies	Valid cloud provider credentials (API keys).

Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to cloud integration page	-	User should be redirected to cloud integration settings.	As Expected	Pass	
2	Select cloud provider	Hetzner	User selects AWS as the provider.	As Expected	Pass	
3	Enter authentication credentials	Email: testuser@gmail.com , API Key: ApiKey12345	The system should validate and authenticate the credentials.	As Expected	Pass	
4	Link cloud account	-	The system should link the cloud account and display the connected services.	As Expected	Pass	

Post Condition	The cloud account is successfully integrated with the platform.
-----------------------	---

7. TRACEABILITY MATRIX

This section explores the requirement traceability matrix of our system.

7.1. RID vs UCID (requirements vs use cases)

Table 20: Requirement Traceability Matrix

UCID/ RID	FR 1	FR 2	FR 3	FR 4	FR 5	FR 6	FR 7	FR 8	FR 9	FR 10	FR 11
UC 1	✓										
UC 2	✓										
UC 3		✓				✓					
UC 4			✓								
UC 5				✓			✓				
UC 6					✓						

8. CONCLUSION

The Brokerage Service for Deploying Self-Hosted Applications initiative has been able to solve the issues of cloud deployment complexity. The system has been constructed with a vision of making cloud technology usable by non-technical people by automating tasks requiring technical knowledge.

In our project many technologies and tools were used, such as Next.js for front-end development, Rust for the back-end, MongoDB for storage of data, and Terraform for provisioning cloud infrastructure automation.

The intuitive interface of the system enables users to set up deployment settings easily and trigger the process with minimal clicks. Besides this, through the real-time progress information, the users can monitor the deployment status effectively. The system successfully achieves its objective making it easy to deploy cloud-based applications.

9. FUTURE WORK

Although our current implementation of our platform establishes a robust foundation for cloud deployments, there are several critical enhancements that we foresee will further enhance its flexibility, ease of use, and performance:

9.1. Increased Cloud Service Support

In the future, we plan to give users access to more cloud services, including the support for more than one provider. This will allow users more freedom to select the cloud environment that suits them best.

9.2. Our Own Affordable Cloud Offering

To further reduce the barrier to entry, we intend to release our own cloud service. This will give users a budget-friendly option to traditional providers perfect for individual projects, startups, and academic use.

9.3. Advanced Configuration Options

While our platform is novice-friendly with reasonable default settings, we are also mindful of technically savvy customers. Advanced versions will provide advanced configuration options so power users can tailor deployments, adjust performance options, and tune infrastructure elements to meet their unique needs.

9.4. Improved Security Features

To ensure user data and deployments remain secure, future updates will include security technologies like end-to-end encryption, and multi-factor authentication (MFA).

9.5. Containerization and Kubernetes Integration

As container technologies such as Docker and orchestration technologies such as Kubernetes gain ground, we will integrate these into the platform to enable it to manage more intricate, scalable, and cloud-native applications.

9.6. Smart Deployment Recommendations

Later versions could also have intelligent deployment recommendations, employing machine learning to examine previous configurations and suggest optimal configurations based on the user's application type and objectives.

10. BIBLIOGRAPHY

Below are the references used in the preparation of the **Brokerage Service for Deploying Self-Hosted Applications** project. The citations follow the IEEE format:

10.1. Books

- **Rust Programming Language.** (2021).
Rust unstable book. <https://doc.rust-lang.org/nightly/unstable-book/index.html>

10.2. Other References

- "AWS Documentation," Amazon Web Services, 2021.
Available: <https://aws.amazon.com/documentation>.
- "Hetzner Cloud Documentation," Hetzner Online GmbH, 2021.
Available: <https://www.hetzner.com/cloud>.
- "Terraform Documentation," HashiCorp, 2021.
Available: <https://www.terraform.io/docs>.
- "Google Cloud Documentation," Google Cloud, 2021.
Available: <https://cloud.google.com/docs>.

11. APPENDIX

11.1. Glossary of terms

- **Cloud Computing:** The delivery of computing services such as servers, storage, databases, networking, software, and more over the internet ("the cloud").
- **Terraform:** An open-source infrastructure as code software tool that provides a consistent CLI workflow to manage hundreds of cloud services.
- **Application Programming Interface:** Set of tools for developing software and applications. APIs enable various software systems to talk to one another.
- **VM (Virtual Machine):** A software-based emulation of physical computers, running an operating system and applications just like a real computer.
- **Elastic IP (EIP):** A public, static IPv4 address created for cloud computing that is dynamic. Elastic IPs enable users to link an IP address to a running instance or modify instances whenever necessary.

11.2. Pre-requisites

Prior to using our system, the Brokerage Service for Deploying Self-Hosted Applications, Following are some pre-requisites:

- **Cloud Accounts:** The user must possess an active account with a supported cloud provider (Hetzner etc). API keys or authentication credentials per cloud provider are mandatory.
- **Terraform Setup:** Users need to have Terraform installed on their local system to execute deployment scripts.
- **SSH Key Pair:** Users need to provide his/her SSH key pair for encrypted communication with cloud servers.
- **Browser and Internet Connection:** A web browser and a working internet connection are needed to access the platform.
- **Basic Cloud Knowledge:** Although the system is meant for non-technical individuals, some basic knowledge of cloud computing and server management will be beneficial for maximum usage.

11.3. User Guide

For detailed, step by step instructions on using the system, kindly refer to the README.txt file included with the project.