-SRE-

NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Course: Software Requirements Engineering

Instructor: Lecturer Sara Rehmat

Muhammad Rehan | 22P-9106

Assignment no 04

Functional Requirements

1. **Client Requirement Submission and Validation:** The system must provide an interface for clients to submit their software project requirements and validate them against predefined rules. This could involve automated validation checks using **Jenkins** for **CI/CD pipelines**.

2. **Guideline Provision and Compliance:** The system should not only integrate principles and guidelines provided by the Cloud Native Computing Foundation but also enforce compliance with these guidelines through automated tools such as **OPA (Open Policy Agent)**.

3. **Infrastructure and Capability Provision:** The system must utilize cloud-native tools and technologies like **Kubernetes** for **orchestration**, **Docker** for **containerization**, and **Terraform** for **Infrastructure** as **Code** practices.

4. **Feedback Reception and Analysis:** The system should be equipped to **receive**, **process**, and **analyze feedback** from the applications developed using the service. This should include runtime errors and usage statistics, which can be collected and analyzed using cloud-native monitoring and observability tools like **Prometheus and Grafana**.

Functional Requirements

5. **Standards and Regulations Influence:** The system must adapt to evolving standards and regulations influenced by the modern digital landscape. This could involve automated compliance checks using **policy-as-code** tools such as **Chef InSpec**.

6. **Infrastructure and Services Provision:** The system must be capable of deploying applications on the infrastructure and services provided by cloud service providers such as **AWS**, **Google Cloud**, or **Azure**. This should involve automated deployment processes using **CI/CD pipelines** with tools like **Jenkins** or **CircleCI**.

7. **Code, Configuration, Testing, and Maintenance:** The system must accept inputs such as code, configuration, testing, and maintenance from cloud-native software developers. It should be able to process these inputs using CI/CD pipelines and automated testing tools like **Selenium** or **JUnit**.

8. **Application Development:** The system must develop software applications based on the client's requirements and the guidelines provided by the CNCF. This should involve practices like microservices architecture, containerization using Docker, and API-first development with tools like **Swagger** or **Postman**.

Functional Requirements

9. **Deployment Information & Configurations Provision:** The system must provide the applications with deployment information, configurations, monitoring management, and compliance. This should involve the use of declarative configurations and policy-as-code tools such as **Terraform** or **Ansible**.

10. **Infrastructure Provision:** The system must provision the necessary infrastructure for the applications using **Infrastructure as Code** practices with tools like **Terraform** or **AWS CloudFormation**.

11. **Application Deployment:** The system must deploy the applications on the provisioned infrastructure using automated deployment processes and **CI/CD pipelines** with tools like **Jenkins** or **Spinnaker**.

12. **Application Monitoring:** The system must monitor the applications for **performance**, **usage**, and errors. This should involve the use of cloud-native monitoring and observability tools like **Prometheus** and **Grafana**.

13. **Application Maintenance: The** system must perform application maintenance, including updates, patches, and upgrades using automated processes and CI/CD pipelines with tools like **Jenkins** or **GitLab CI**.

External Interface Requirements

User Interfaces

- **UI-1:** The system shall employ modern web application user interface standards, ensuring a user-friendly and intuitive experience. This could be facilitated by frameworks like React or Angular for a more interactive and responsive UI.

- **UI-2:** The system shall provide a context-sensitive help link on each webpage, possibly utilizing tools like Intercom or Zendesk for better user support.

- **UI-3:** The system shall ensure complete keyboard accessibility for navigation and project requirement submission, enhancing accessibility and usability.

Software Interfaces

- **SI-1:** The system shall interface with cloud-native tools and technologies like Kubernetes for orchestration, Docker for containerization, and Terraform for infrastructure management.

- **SI-2:** The system shall interface with the Cloud Native Computing Foundation (CNCF) to receive principles and guidelines for cloud-native development.

External Interface Requirements

- **SI-2:** The system shall interface with the Cloud Native Computing Foundation (CNCF) to receive principles and guidelines for cloud-native development.

- **SI-3:** The system shall interface with cloud service providers like AWS, Google Cloud, or Azure for infrastructure and services provision.

- **SI-4:** The system shall interface with the applications developed using the service to receive feedback, using tools like Prometheus for monitoring and ELK stack for logging and analysis.

- **SI-5:** The system shall provide deployment information, configurations, monitoring management, and compliance to the applications, employing tools like Helm for managing Kubernetes applications and Istio for service mesh capabilities.

Hardware Interfaces

- No hardware interfaces have been identified.

External Interface Requirements

Communications Interfaces

- **CI-1: Automation & Configuration:** The system shall automate the creation of the infrastructure, together with updates to it, using tools like **Google Cloud Deployment Manager.**

- **CI-2: Scale up and scale down:** The system shall automate the scaling up of the system in response to increases in load, and scaling down in response to sustained drops in load. This ensures that the service remains available and reduces costs .

- **CI-3: Monitoring and automated recovery:** The system shall have built-in monitoring and logging from inception. In addition, it shall attach automation to the logging and monitoring data streams for automatic repair, scaling, and deployment .

- **CI-4: State Management:** The system shall manage the states of the applications and services, ensuring that data is persisted between restarts and that the applications can run reliably .

External Interface Requirements

Communications Interfaces

- **CI-5: Container Runtime:** The system shall use a container runtime to create and start containers executing application code.

- **CI-6: Integration with Hosted Kubernetes Services:** If using hosted Kubernetes services, the system shall integrate with these services for managing and orchestrating containers.

- **CI-7: Integration with Platform as a Service (PaaS) Providers:** If using PaaS providers, the system shall integrate with these services for building, deploying, and scaling applications.

- **CI-8: Integration with Cloud-Native Storage Providers:** If using cloud-native storage providers, the system shall integrate with these services for storing and retrieving data.

Quality Attributes

Usability Requirements

- **USE-1:** The system shall be intuitive and user-friendly, allowing businesses and organizations to easily submit their software project requirements. This could be facilitated by the use of intuitive UI/UX design tools like **Sketch** resulting in **tenfold ease**.

- **USE-2**: The system shall be designed such that **95%** of new users can successfully submit their project requirements without errors on their first try.

Performance Requirements

- **PER-1:** The system shall accommodate a total of **500 users** and a **maximum of 200** concurrent users during peak usage times, leveraging cloud scalability to manage user loads.

- **PER-2: 95%** of webpages generated by the system shall load completely within **3 seconds** from the time the user requests the page over a **50 Mbps** or faster Internet connection. This could be achieved through efficient front-end optimization techniques and CDN services like Cloudflare.

Quality Attributes

- **PER-3:** The system shall display confirmation messages to users within an average of **2 seconds** and a maximum of **4 seconds** after the user submits information to the system.

Security Requirements

- **SEC-1:** All network transactions that involve sensitive information shall be encrypted using industry-standard encryption protocols, such as TLS and HTTPS. For managing secrets, tools like HashiCorp's Vault could be used providing **24/7** safety.

- **SEC-2:** Users shall be required to authenticate to the system for all operations except viewing the CNCF guidelines on **continuous** bases . This could be implemented using identity and access management services like Okta or AWS Cognito.

Quality Attributes

Safety Requirements

- **SAF-1:** The system shall protect user data and ensure privacy by adhering to the latest data protection regulations and standards, with the help of tools like **GDPR compliance checkers** and **privacy policy generators** with provides **~100%** security.

Availability Requirements

- **AVL-1:** The system shall be available at least **99%** of the time excluding scheduled maintenance windows. This can be achieved by leveraging cloud-native high availability and fault tolerance strategies

Robustness Requirements

- **ROB-1:** The system shall be able to reconnection between the user and the system if broken prior to a new project requirement submission being either confirmed or terminated, Using state persistence mechanisms such as Redis with **zero** leniency.

Data Dictionary

Data Element	Description	Data Type	Length	Values	Tool
User Interfaces	Modern web application user interface for user-friendly and intuitive experience.	Text	Variable	N/A	React, Angular
Software Interfaces	Interface with cloud-native tools and technologies to build, deploy, and manage software applications.	Object	Variable	Kubernetes, Docker, Terraform	Kubernetes, Docker, Terraform
Feedback Reception	Interface for receiving and processing feedback from applications developed using the service.	Object	Variable	Runtime errors, usage statistics	Prometheus, Grafana
Standards and Regulations Influence	Adaptation to evolving standards and regulations influenced by the modern digital landscape.	Text	Variable	N/A	Chef InSpec

Data Dictionary

Infrastructure and Services Provision	Deployment of applications on the infrastructure and services provided by cloud service providers.	Object	Variable	AWS, Google Cloud, Azure	Jenkins, CircleCI
Code, Configuration, Testing, and Maintenance	Acceptance and processing of inputs such as code, configuration, testing, and maintenance.	Object	Variable	N/A	Selenium, JUnit
Application Development	Development of software applications based on client's requirements and CNCF guidelines.	Object	Variable	N/A	Docker, Swagger, Postman
Deployment Information & Configurations Provision	Provision of deployment information, configurations, monitoring management, and compliance to the applications.	Object	Variable	N/A	Terraform, Ansible
Infrastructure Provision	Provisioning of necessary infrastructure for the applications.	Object	Variable	Servers, databases,	Terraform, AWS

Data Dictionary

Application Deployment	Deployment of the applications on the provisioned infrastructure.	Object	Variable	N/A	Jenkins, Spinnaker
Application Monitoring	Monitoring of the applications for performance, usage, and errors.	Object	Variable	N/A	Prometheus, Grafana
Application Maintenance	Performance of application maintenance, including updates, patches, and upgrades.	Object	Variable	N/A	Jenkins, GitLab CI
Project Completion	Completion of the project and delivery of the developed software applications back to the clients.	Text	Variable	N/A	ELK stack, Google Data Studio

CLOUD NATIVE
COMPUTING FOUNDATION

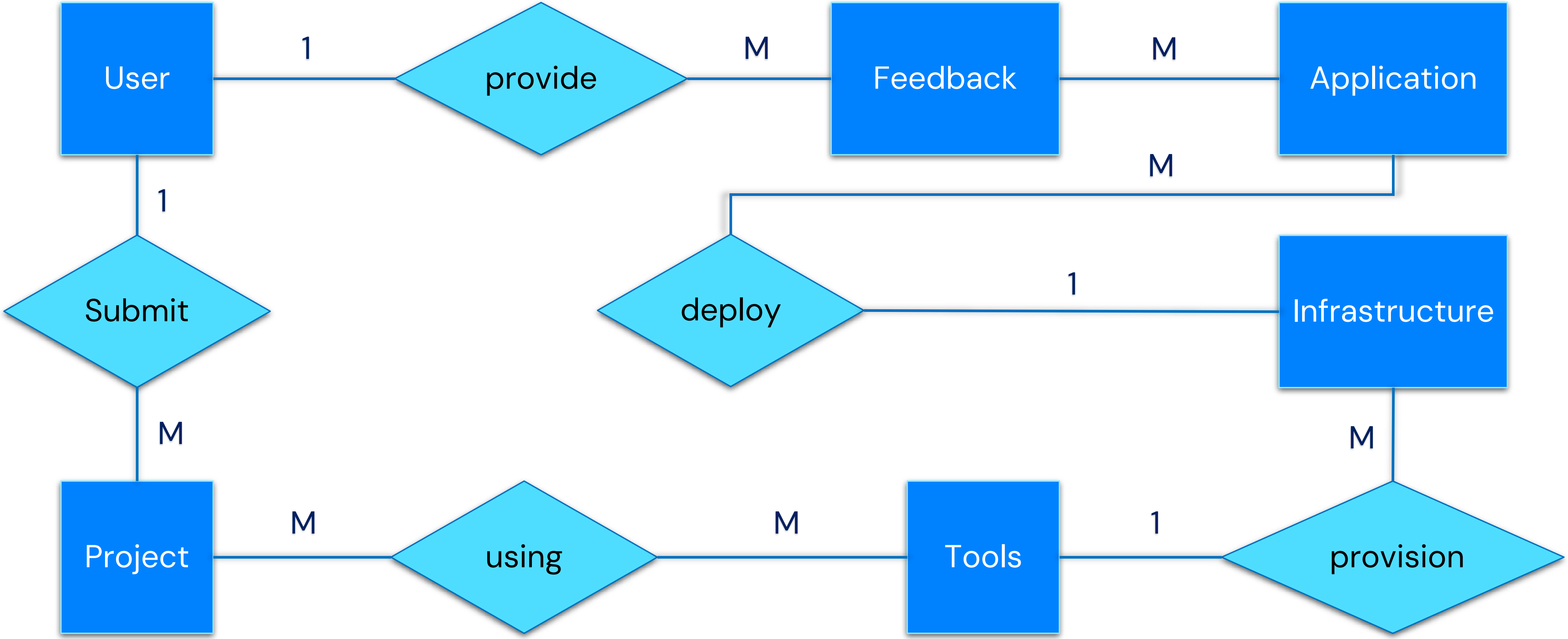
Event Response Table

ID	Event	System State	System Response	Tool
1	User submits project requirements	System is idle	System validates and saves the requirements	Jenkins
2	System receives CNCF guidelines	System is idle	System integrates and enforces the guidelines	OPA
3	System needs to provision infrastructure	System is idle	System provisions necessary infrastructure using cloud-native tools	Terraform, Docker, Kubernetes
4	Application generates feedback	System is idle	System receives and processes the feedback	Prometheus, Grafana
5	Change in standards or regulations	System is idle	System adapts to the new standards or regulations	Chef InSpec
6	Need to deploy application on cloud infrastructure	System is idle	System deploys the application using automated processes	Jenkins, CircleCI
7	System receives code, configuration, testing, and maintenance inputs	System is idle	System processes these inputs	Selenium, JUnit

8	Need to develop a software application	System is idle	System develops the application based on client's requirements and CNCF guidelines	Docker, Swagger, Postman
9	System needs to provide deployment information, configurations, monitoring management, and compliance to the applications	System is idle	System provides the necessary information and configurations	Terraform, Ansible
10	Need to provision infrastructure for the applications	System is idle	System provisions the necessary infrastructure	Terraform, AWS CloudFormation
11	Need to deploy applications on the provisioned infrastructure	System is idle	System deploys the applications	Jenkins, Spinnaker
12	Need to monitor the applications	System is idle	System monitors the applications for performance, usage, and errors	Prometheus, Grafana
13	Need to maintain the applications	System is idle	System performs maintenance actions	Jenkins, GitLab CI
14	Project needs to be completed	System is idle	System completes the project and delivers the developed software applications to the clients	ELK stack, Google Data Studio

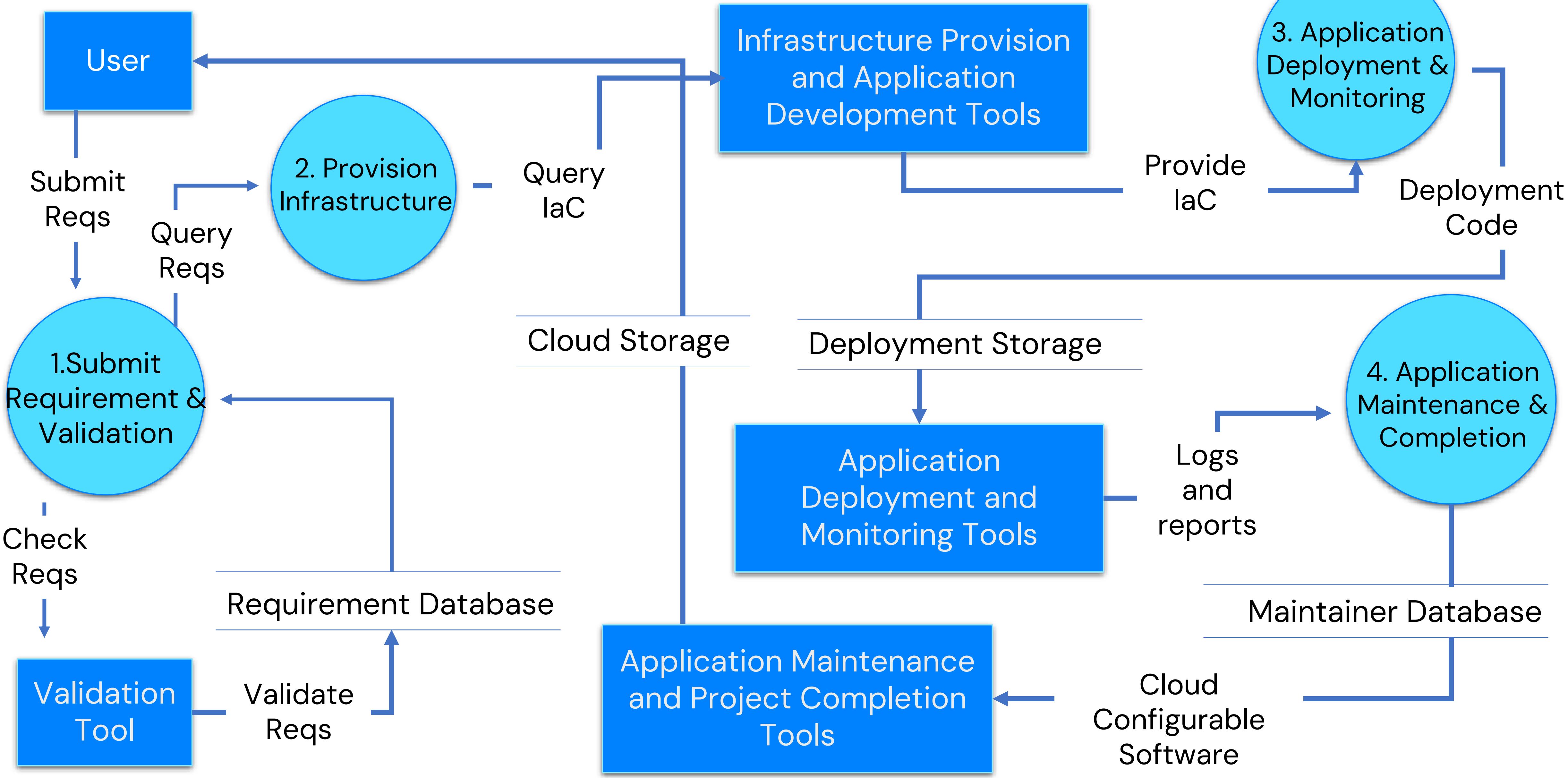
CLOUD NATIVE
COMPUTING FOUNDATION

Entity-Relation Diagram



CLOUD NATIVE
COMPUTING FOUNDATION

Data Flow Diagram (0 DFD)


User
Submit Reqs
Query Reqs
2. Provision Infrastructure
Query IaC
Infrastructure Provision and Application Development Tools
3. Application Deployment & Monitoring
Provide IaC
Deployment Code
1.Submit Requirement & Validation
Cloud Storage
Deployment Storage
4. Application Maintenance & Completion
Application Deployment and Monitoring Tools
Logs and reports
Check Reqs
Requirement Database
Maintainer Database
Validation Tool
Validate Reqs
Application Maintenance and Project Completion Tools
Cloud Configurable Software

Test Cases

```
// Define the functions that are used in the test cases for project testing.
fn process_code_configuration_testing_maintenance(code: &str, configuration: &str, testing: &str, maintenance: &str) -> Result<(
()> {
    // This function could be used to process code, configuration, testing, and maintenance
    // For this example, we'll just return Ok(())
    println!("Processing code, configuration, testing, and maintenance...");
    Ok(())
}
fn provide_deployment_info_and_config(deployment_info: &str) -> Result<(), ()> {
    // This function could be used to provide deployment information and configurations
    // For this example, we'll just return Ok(())
    println!("Providing deployment information and configurations...");
    Ok(())
}
fn provide_infrastructure_and_capabilities(infra_req: &str) -> Result<(), ()> {
    // This function could be used to provide necessary infrastructure and capabilities
    // For this example, we'll just return Ok(())
    println!("Providing necessary infrastructure and capabilities...");
    Ok(())
}
fn provide_services(services_req: &str) -> Result<(), ()> {
    // This function could be used to provide necessary services
    // For this example, we'll just return Ok(())
    println!("Providing necessary services...");
    Ok(())
}
```

Test Cases

```
fn provide_capabilities(capability_req: &str) -> Result<(), ()> {
   // This function could be used to provide necessary capabilities
   // For this example, we'll just return Ok(())
   println!("Providing necessary capabilities...");
   Ok(())
}
// Define the test cases
fn test_code_configuration_testing_maintenance() {
   let code = "Some code";
   let configuration = "Some configuration";
   let testing = "Some testing";
   let maintenance = "Some maintenance";
   let result = process_code_configuration_testing_maintenance(code, configuration, testing, maintenance);
   assert!(result.is_ok(), "Code, Configuration, Testing, and Maintenance failed");
}
fn test_deployment_information_configurations_provision() {
   let deployment_info = "Deployment information and configurations";
   let result = provide_deployment_info_and_config(deployment_info);
   assert_eq!(result, Ok(()), "Deployment information provision failed");
}
fn test_infrastructure_provision() {
   let infra_req = "Provide necessary infrastructure";
   let result = provide_infrastructure_and_capabilities(infra_req);
   assert!(result.is_ok(), "Infrastructure provision failed");
}
```

Test Cases

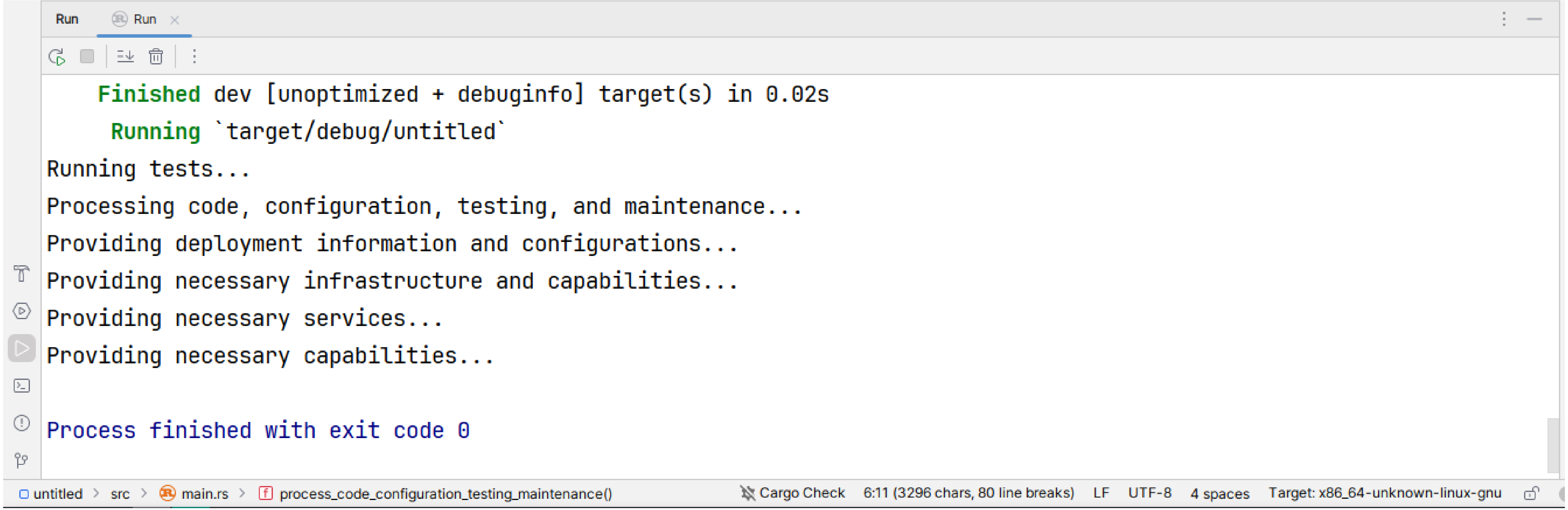
```
fn test_infrastructure_services_provision() {
   let services_req = "Provide necessary services";
   let result = provide_services(services_req);
   assert!(result.is_ok(), "Infrastructure services provision failed");
}


fn test_infrastructure_capability_provision() {
   let capability_req = "Provide necessary capabilities";
   let result = provide_capabilities(capability_req);
   assert!(result.is_ok(), "Infrastructure capability provision failed");
}

// Define the main function to run the tests
fn main() {
   // Run the tests
   println!("Running tests...");
   test_code_configuration_testing_maintenance();
   test_deployment_information_configurations_provision();
   test_infrastructure_provision();
   test_infrastructure_services_provision();
   test_infrastructure_capability_provision();
}
```

CLOUD NATIVE
COMPUTING FOUNDATION

Test Cases Output



Run
Run
untitled > src > main.rs > process_code_configuration_testing_maintenance()
Cargo Check
6:11 (3296 chars, 80 line breaks)
LF
UTF-8
4 spaces
Target: x86_64-unknown-linux-gnu

```
    Finished dev [unoptimized + debuginfo] target(s) in 0.02s
     Running `target/debug/untitled`
Running tests...
Processing code, configuration, testing, and maintenance...
Providing deployment information and configurations...
Providing necessary infrastructure and capabilities...
Providing necessary services...
Providing necessary capabilities...

Process finished with exit code 0
```