A REPORT

ON

Software Engineer Intern at Telaverge

Submitted by,

Prince Gupta - 20211CCS0096

Under the guidance of,

Dr. Sharmasth Vali Y

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internship/Project report **"Software Engineer Intern at Telaverge"** being submitted by "**Prince Gupta**" bearing roll number "20211CCS0096" in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. Sharmasth Vali Y
Associate Professor
PSCS
Presidency University

Dr. Anand Raj S P
Professor & HoD
PSCS
Presidency University

Dr. Mydhili Nair
Associate Dean
PSCS
Presidency University

Dr. Sameeruddin Khan
Pro-Vice Chancellor - Engineering
Dean –PSCS / PSIS
Presidency University

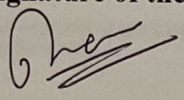
PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I hereby declare that the work, which is being presented in the report entitled **"Software Engineer Intern at Telaverge"** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of my own investigations carried under the guidance of **Dr. Sharmasth Vali Y, Associate Professor, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

Name:	Roll No:	Signature of the Student
Prince Gupta	20211CCS0096	

INTERNSHIP COMPLETION CERTIFICATE

Gmail

Prince Gupta <gprince@telaverge.com>

Request for Confirmation of Ongoing Internship for College Submission

2 messages

Prince Gupta <gprince@telaverge.com>
To: Telaverge HR <hr@telaverge.com>
Cc: Prince Gupta <princeguptao741@gmail.com>, PRINCE.20211CCS0096@presidencyuniversity.in
Mon, May 12, 2025 at 1:36 PM

Dear HR,

I hope this message finds you well.

I am writing to kindly request a formal confirmation stating that I am currently interning at Telaverge Communications as part of a 6-month internship program that commenced on December 5, 2024, and is scheduled to conclude in June 2025.

As per the academic requirements of my university, I am required to submit a proof of internship along with my internship report by **May 15, 2025.** Since the internship is still ongoing and the final certificate will be issued upon completion, I would greatly appreciate it if you could provide a confirmation letter or email indicating:

- That I am currently interning at Telaverge Communications
- The internship start and expected end dates
- That an official internship certificate will be issued upon successful completion

This confirmation will serve as interim proof for my university until the final certificate is issued.

Thank you for your time and support. Please let me know if any additional details or documentation are required from my end.

Warm regards,
Prince Gupta
B.Tech – Computer Science & Engineering (Cybersecurity) - **20211CCS0096**
President's University
Personal Email : princeguptao741@gmail.com
College Email: PRINCE.20211CCS0096@presidencyuniversity.in
Phone: 6360276989

Telaverge HR <hr@telaverge.com>
To: Prince Gupta <gprince@telaverge.com>
Cc: Prince Gupta <princeguptao741@gmail.com>, PRINCE.20211CCS0096@presidencyuniversity.in
Mon, May 12, 2025 at 7:00 PM

Hello Prince,

This is to confirm that you are currently interning with Telaverge Communications (India) Pvt. Ltd. as part of an internship program.

Please note that the official internship certificate will be issued upon the successful completion of your internship. In the meantime, this email may be used as a formal confirmation for submission to your university.

Should you require any further details please feel free to let me know.

Wishing you all the best with your academic requirements.

Warm regards,
Leena S
HR | Telaverge Communications (I) Pvt. Ltd.

[Quoted text hidden]

ABSTRACT

This report presents the outcomes of my internship at Telaverge Communications Pvt. Ltd., where I was engaged in automating and validating telecom protocol testing, specifically focusing on the Diameter protocol using Telaverge's in-house automation platform, Regal. The internship revolved around ensuring quality assurance (QA), improving test coverage, and integrating essential tools into the automation ecosystem.

My primary contributions included extensive QA for Diameter regression test cases, ensuring conformance to 3GPP and RFC standards across interfaces such as S6a, Sh, and Cx. I was actively involved in the integration of DiamonPro, a commercial Diameter Stack, with Regal. This involved creating automated test workflows, debugging stack communication issues, and validating both functional and performance behavior under varying loads.

In parallel, I also contributed to the product integration of Seagull, an open-source Diameter traffic simulator, with Regal. This integration enabled automated traffic generation and scenario execution for a wide range of test cases. As part of this effort, I developed plugins, helper scripts, and deployment automation using Ansible.

A significant milestone during the internship was my work on the Telaverge Marketplace, where I took ownership of organizing applications and modules into a structured format. I uploaded, versioned, and validated modules like Ro Blackbox, Gx Blackbox, and several others. Additionally, I fixed several functional and performance test cases, ensuring stability and alignment with Regal's execution standards.

Further, I contributed to the integration of DiamonPro 7.9.6 with Multi-Interface support, automating simultaneous testing across multiple Diameter interfaces (S6a, Sh, Cx) using both TCP and SCTP. I also performed QA for the Load Balancer feature in the Diameter stack, ensuring correct peer routing and failover behaviour.

This internship enhanced my understanding of Diameter protocol internals, automated testing frameworks, protocol stack integration, and telecom-grade QA processes. It allowed me to develop strong skills in Python scripting, debugging, system validation, and cross-functional collaboration within a fast-paced engineering team.

ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair,** Presidency School of Computer Science and Engineering, Presidency University, and Dr. ANAND RAJ S P, Professor, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Sharmasth Vali Y**, Associate Professor and Reviewer **Dr. ANAND RAJ S P, Professor**, Presidency School of Computer Science and Engineering, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the internship work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 Internship Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K,** department Project Coordinators and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided me in bringing out this project.

Prince Gupta

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Background

With the rapid evolution of 4G LTE and the widespread adoption of 5G networks, telecom infrastructure has become increasingly complex and demanding. At the core of these networks lies the Diameter protocol, a crucial standard defined in RFC 6733, responsible for Authentication, Authorization, and Accounting (AAA) operations across various interfaces such as S6a, Sh, Cx, Gx, and Gy. Validating the functionality and performance of Diameter-based systems has become a cornerstone in ensuring network reliability and interoperability.

As telecom networks continue to scale, traditional manual testing methods fall short in keeping up with the demands for agility, repeatability, and accuracy. This calls for the integration of robust automation frameworks, traffic simulators, and validation tools that can collectively support the end-to-end testing of Diameter protocol stacks under varying conditions.

1.2 Internship Context

This internship was carried out at Telaverge Communications Pvt. Ltd., a company specializing in telecom protocol solutions, cloud-native platforms, and automation frameworks. The work centered around enhancing test automation capabilities using Regal, an in-house test orchestration platform developed by Telaverge.

The primary focus areas included:

- Quality assurance (QA) of Diameter regression test cases.
- Integration of DiamonPro, a commercial Diameter stack, with Regal.
- Plugin development and automation of Seagull simulator.
- Multi-interface Diameter automation for interfaces like S6a, Sh, and Cx.
- Testing Diameter stack load balancer features under distributed peer environments.
- Organizing and managing Marketplace modules, enabling streamlined deployment and version control.

1.3 Problem Statement

Telecom testing environments demand high levels of accuracy, performance, and repeatability. Challenges arise when:

- Manual test case execution introduces human error and delays.

- Lack of synchronization across multiple Diameter interfaces leads to incorrect validation.
- Multi-streaming and multihoming (especially with SCTP) remain under-tested.
- Test modules and plugins lack standardized version control and Marketplace organization.

These challenges hinder the effectiveness of Diameter protocol validation, making it necessary to automate, integrate, and optimize both the protocol stack and testing frameworks.

1.4 Objectives

The goal of this project was to:

- Enable automated testing of the Diameter stack via Regal.
- Develop and integrate Seagull plugins with proper traffic generation and scenario validation.
- Implement multi-interface support (S6a, Sh, Cx) using TCP and SCTP.
- Conduct load and performance testing with millions of call flows and high TPS rates.
- Enhance test result validation with real-time metrics (e.g., MPS, ongoing transactions).
- Upload, version, and deploy Marketplace modules in a structured and reliable manner.

1.5 Scope of the Internship Project

The scope of the project covered the following aspects:

- Functional and regression testing for Diameter stack features.
- Automated infrastructure deployment using Ansible and Kubernetes.
- Plugin development and simulator integration.
- Performance benchmarking (e.g., 5 million callflows @ 20k TPS).
- Test case and module management within Telaverge Marketplace.

The project did not cover the development of the Diameter protocol itself but focused on the automation, validation, and integration of existing Diameter components.

Chapter 2

LITERATURE SURVEY

2.1 Introduction to Diameter Protocol

The Diameter protocol, standardized in RFC 6733, is a next-generation protocol designed to support Authentication, Authorization, and Accounting (AAA) in IP-based networks. It evolved as a replacement for the older RADIUS protocol, addressing its limitations in scalability, reliability, and extensibility. Diameter is integral to the Evolved Packet Core (EPC) in 4G LTE and 5G networks, and it forms the backbone of signaling between entities such as the MME, HSS, PCRF, and OCS.

Each network function communicates via specialized Diameter interfaces like:

- S6a (MME-HSS communication)
- Sh (AS-HSS interaction)
- Cx/Dx (CSCF-HSS interaction)
- Gx/Gy (Policy and charging control)

To ensure conformance and interoperability, these interfaces are defined in 3GPP specifications (e.g., TS 29.272, TS 29.329, TS 29.328).

2.2 Importance of Automated Testing in Telecom

In modern telecom environments, manual protocol testing is time-consuming, error-prone, and non-scalable. Automated testing ensures:

- Repeatability and consistency
- Efficient execution under heavy traffic
- Quick detection of regressions and protocol violations

Frameworks such as Regal—Telaverge's in-house automation platform—play a vital role in enabling high-speed, scalable Diameter test automation.

Automation supports:

- Functional regression testing
- Performance benchmarking
- Stress testing using tools like Seagull or DiamonPro

2.3 Overview of Testing Tools and Frameworks

2.3.1 Seagull Simulator

An open-source traffic generation tool supporting Diameter and other protocols.

It allows for simulation of client/server Diameter messages via scenario XML files.

Flexible and customizable, it's suitable for functional testing and early-stage automation.

2.3.2 DiamonPro Stack

A commercial Diameter stack used by Telaverge for protocol conformance and performance testing.

Supports multi-interface deployment and SCTP/TCP transport layers.

Ideal for large-scale simulations and high-TPS benchmarking.

2.3.3 Regal Automation Framework

Telaverge's internal test orchestration tool.

Features infrastructure deployment, test suite management, validation logic, and integration with simulators (e.g., Seagull).

Supports Kubernetes, Ansible, and container-based test execution pipelines.

Recently extended to support Marketplace modules for better module lifecycle management.

2.4 Challenges in Current Testing Approaches

Despite the availability of powerful tools, several challenges persist:

- Lack of multi-interface synchronization: Most frameworks test individual interfaces in isolation.
- Underutilization of SCTP features: Multi-homing and multi-streaming often remain untested.
- Manual test execution: Non-automated workflows reduce reliability and increase time to validate.
- Poor module management: Lack of standardization in test module versioning and deployment across teams.
- Limited real-time validation: Many testing tools don't provide granular validation metrics such as session count, timeout indications, or transaction freezes.

2.5 Need for Structured Module Deployment

In large-scale testing setups, centralized deployment and management of test assets are essential. A Marketplace-like system:

- Ensures consistent module versions across teams.

- Enables rollback, version control, and simplified sharing.
- Facilitates repeatable and reliable deployment pipelines for simulators and regression suites.

Telaverge's Marketplace system addresses this by allowing users to upload, version, and deploy test modules (e.g., Ro/Gx Blackbox) and simulators in a streamlined way.

Chapter 3

RESEARCH GAPS OF EXISTING METHODS

3.1 Introduction

Telecommunication networks are evolving rapidly with the adoption of 4G LTE, IMS, and 5G architectures, all of which heavily rely on the Diameter protocol for AAA (Authentication, Authorization, and Accounting) signaling. While several tools exist for testing and validating Diameter interfaces—such as Seagull, DiamonPro, and proprietary traffic generators, significant limitations are still observed in their automation, integration, and validation workflows, especially in high-throughput and multi-interface environments.

This chapter identifies the key research and implementation gaps in existing Diameter testing approaches that this internship aimed to bridge.

3.2 Identified Gaps in Existing Methods

3.2.1 Lack of Multi-Interface Automation Support

Existing test setups often focus on validating a single Diameter interface in isolation. However, real-world telecom systems rely on multi-interface interactions (e.g., S6a, Sh, and Cx) working in parallel. Most tools do not support:

- Simultaneous traffic generation across interfaces.
- Coordinated validation of related session flows.
- Synchronized load distribution across clients and servers.

This leads to fragmented testing and incomplete protocol verification.

3.2.2 Insufficient SCTP Validation Coverage

While SCTP (Stream Control Transmission Protocol) is widely supported in Diameter for redundancy and reliability, most test frameworks:

- Fail to validate multi-homing and multi-streaming configurations.
- Do not simulate failover or stream congestion conditions.
- Focus primarily on TCP, leaving SCTP under-tested in production-like conditions.

This lack of validation results in untested failure scenarios and protocol instability during live deployments.

3.2.3 Manual Seagull Integration and Operation:

Although Seagull is a powerful and flexible traffic simulator, its integration into automated platforms like Regal has traditionally been:

- Manual and error-prone, requiring custom script execution.
- Not standardized across teams or versions.
- Lacking in plugin-based support for automated deployment, execution, and teardown.

This inhibits its usability for regression automation, CI/CD pipelines, and cloud-based execution**.**

3.2.4 Limited Real-Time Validation and Logging:

Most testing frameworks log outputs post-execution and lack:

- Live monitoring of session progress, ongoing transactions, and MPS.
- Timeout indication tracking, frozen transaction detection, and request-response mismatch handling.
- Granular control over test status based on dynamic runtime metrics (e.g., validating whether Total Completed == Session Count).

Such delays in validation impact early failure detection and real-time troubleshooting.

3.2.5 Poor Module Lifecycle Management in Test Environments:

Test cases, simulators, and stacks are frequently:

- Maintained in local folders with inconsistent versions.
- Manually deployed across environments.
- Lacking proper version control, rollback capability, and structured naming.

This results in:

- Confusion during collaboration across teams.
- Redundant test runs and unpredictable environment behavior.

The absence of a centralized system like the Telaverge Marketplace for managing test modules limits test automation efficiency.

3.3 Technical Challenges Observed in Practice:

During the internship, the following practical issues further highlighted the above gaps:

- Bug 17557: Test cases failed to reflect updated logic due to Regal infra caching issues.
- Marketplace module misalignment: Inconsistent naming and outdated uploads caused

conflicts during regression runs.

- Scanner pod crashes and infra delays: Infrastructure instability affected test automation timelines and report generation.

These issues emphasized the need for:

- A robust plugin-based integration model.
- Auto-validation at runtime.
- Structured module management via a Marketplace system.

3.4 Conclusion:

Although several frameworks and tools exist for Diameter protocol validation, they fall short in supporting real-time, multi-interface, scalable, and cloud-native automation. The research and implementation gaps identified in this chapter formed the foundation of this internship's contributions specifically through:

- Developing Regal plugins,
- Automating multi-interface execution and validation,
- Standardizing module deployment in the Marketplace.

By addressing these gaps, the project helped pave the way toward a robust and scalable Diameter test automation infrastructure at Telaverge.

Chapter 4

PROPOSED MOTHODOLOGY

4.1 Introduction:

The goal of this internship project was to automate and optimize the testing of Diameter protocol interfaces using the Regal automation framework. The methodology involved the design, development, integration, and validation of various Diameter stack components and tools, including DiamonPro, Seagull, and Marketplace modules. The project followed an iterative and modular approach, with continuous testing and feedback at each phase.

4.2 Methodology Phases:

The work was divided into six major phases, each building upon the outcomes of the previous. This structured approach allowed for progressive automation, validation, and infrastructure enhancement of the Diameter protocol testing system.

4.2.1 Phase 1: Environment Setup and Stack Validation

The initial phase involved setting up the Diameter testing environment. The DiamonPro (v7.9.6) stack was installed and configured on virtual machines to support Diameter protocol interfaces such as S6a, Sh, and Cx. Both TCP and SCTP transport layers were used for interface validation. Regression test cases were executed manually to verify proper message exchanges, including CER/CEA, ULR/ULA, LIR/LIA, and PUR/PUA. Logs and network traffic were captured using tools like Wireshark to generate pcap files for further verification and debugging.

4.2.2 Phase 2: Seagull Plugin Development

This phase focused on integrating the Seagull traffic simulator into the Regal framework. The existing manual execution workflow of Seagull was analyzed, and a custom plugin was developed to automate its installation and execution using Ansible. The plugin enabled remote configuration, scenario execution through Diameter XML files, and log collection. Supporting helper classes like seagull_tool.py, seagull_helper.py, and seagull_repo_plugin.py were implemented to streamline the process. The integration was validated by simulating CER/CEA flows between Seagull and DiamonPro in both client and server roles.

4.2.3 Phase 3: Multi-Interface Automation

To replicate real-world telecom environments, support for multi-interface automation was added. Regal was enhanced to handle simultaneous execution across Diameter interfaces such as S6a, Sh, and Cx. Test templates like testcase_conf.json, itsdiaclient.xml and Run.client were developed using Jinja templating. The system was configured to support traffic generation over both TCP and SCTP, enabling validation of features like multi-homing failover and multi-streaming load balancing. Logic was implemented to coordinate session flow and orchestrate test execution across multiple interfaces.

4.2.4 Phase 4: Performance Testing and Validation Enhancement

With the functional components in place, controlled performance testing was conducted. The system was tested with varying call flow volumes—1 million, 3 million, and 5 million—at different call rates (500 to 2000 CPS). Validation logic was expanded to monitor key runtime metrics such as Session Count, Ongoing Transactions, Total Transactions Completed, Timeout Indications, and Messages Per Second (MPS). The test execution and validation scripts (testcase.py and helpers) were modified to pull real-time data from client logs and DBGConsole outputs for accurate and automated verdict generation.

4.2.5 Phase 5: Marketplace Module Standardization

This phase focused on organizing and publishing reusable test assets. Test modules such as Ro Blackbox, Gx Blackbox, and Seagull scenarios were standardized using structured naming conventions and version control (e.g., 1.0.1, 1.0.2). Each module was packaged into .rp and .rtm files and uploaded to the Regal Marketplace. Descriptions, tags, and classifications were added to enhance discoverability. Collaborations with the QA team ensured module quality before public deployment in the Marketplace.

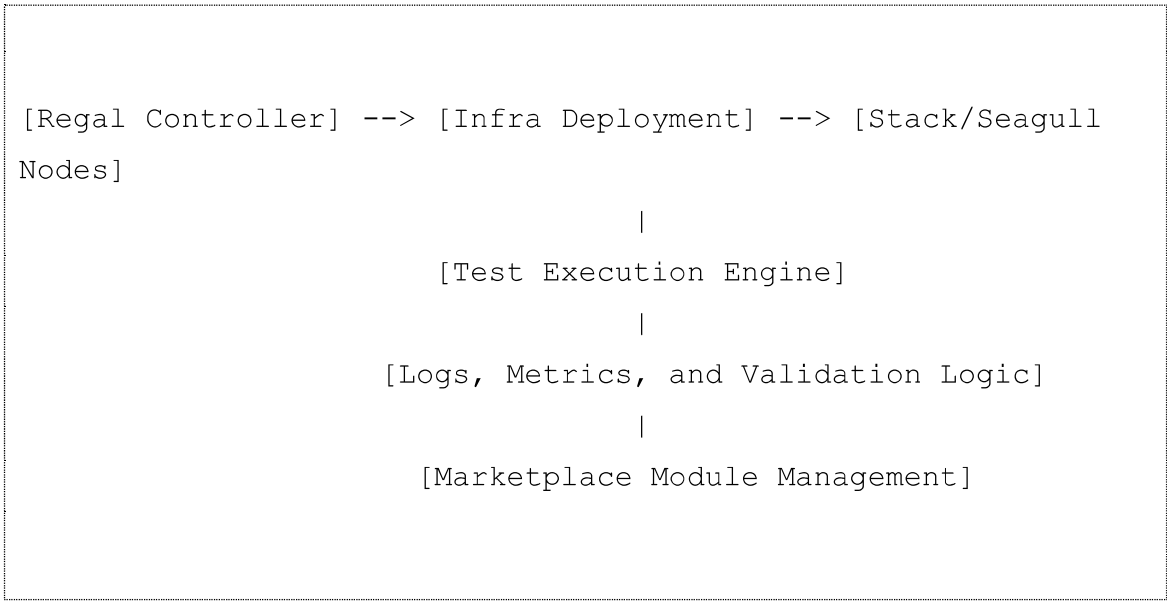
4.2.6 Phase 6: Infra Optimization and Bug Resolution

Finally, the focus shifted to debugging and optimizing the infrastructure. Key issues were addressed, including Bug 17557 (test case reflection delay) and Bug 17668 (validation inconsistency across test suites). The team worked closely with the Regal Dev Team to stabilize scanner pods, improve job execution speed, and reduce pod crash frequency. Kubernetes pods were reconfigured and redeployed, with regular health monitoring and manual resets used when necessary. This phase significantly improved system reliability and test automation resilience.

4.3 Tools and Technologies Used:

Tool/Technology	Purpose
DiamonPro	Diameter stack with S6a, Sh, and Cx support
Seagull	Traffic generator for Diameter scenarios
Regal	Automation platform for deployment and validation
Ansible	Automated stack and simulator provisioning
Jinja	Dynamic rendering of test configurations
Kubernetes	Deployment and orchestration of Regal services
DBGConsole	Runtime statistics and error logging
Wireshark	Protocol-level pcap capture and analysis

4.4 T Workflow Diagram:



4.5 Summary:

This methodology enabled a comprehensive integration of all major Diameter components with the Regal framework. Each phase of implementation addressed specific limitations found in existing setups—ranging from test automation and simulator deployment to Marketplace standardization and performance benchmarking. The step-by-step approach ensured successful delivery of a modular, automated, and scalable test infrastructure for Diameter protocol validation.

Chapter 5
OBJECTIVES

5.1 Primary Objective

The primary objective of this internship was to automate and enhance the validation of Diameter protocol interfaces using Telaverge's in-house automation framework, Regal, and contribute to test infrastructure scalability, reliability, and maintainability through integration with simulators, plugin development, and Marketplace module management.

5.2 Specific Objectives

5.2.1 Quality Assurance (QA) of Diameter Stack

The internship began with rigorous quality assurance of the Diameter protocol stack. This involved executing and validating regression test cases across standard interfaces such as S6a, Sh, and Cx. A significant portion of this work focused on analysing key message flows like CER/CEA, ULR/ULA, LIR/LIA, and PUR/PUA using tools such as Wireshark and DBGConsole. In addition, considerable effort was directed toward identifying, reproducing, and debugging protocol-level issues within the stack's configuration and runtime behaviour.

5.2.2 Integration of DiamonPro with Regal

A critical objective was the integration of the commercial Diameter stack—DiamonPro (v7.9.6)—with the Regal automation framework. The task involved automating the deployment and configuration of the stack using Ansible and generating runtime configuration files through Jinja-based templating. Furthermore, system logs and performance metrics from DiamonPro were successfully integrated into Regal's validation engine for real-time result tracking and automated decision-making.

5.2.3 Plugin Development for Seagull Simulator

Another important objective was the development of a Regal-compatible plugin for Seagull, an open-source Diameter simulator. The plugin was built to support automated installation, traffic scenario execution, and graceful teardown through Ansible. It enabled Regal to simulate CER/CEA and CCR/CCA message flows using structured XML-based scenarios, effectively transforming Seagull into a reusable test component within the larger framework.

5.2.4 Multi-Interface Test Automation

The internship also focused on enabling multi-interface testing across S6a, Sh, and Cx interfaces. This required implementing logic for simultaneous execution over both TCP and SCTP protocols, along with incorporating support for SCTP-specific features like multi-streaming and multi-homing. Special care was taken to ensure proper synchronization between message flows and session identifiers across interfaces, guaranteeing session integrity and full-cycle validation.

5.2.5 Performance and Load Testing

To validate the robustness and scalability of the test framework, load tests were executed with up to 5 million callflows at traffic rates reaching 20,000 TPS (Transactions Per Second). Performance metrics such as Session Count, Ongoing Transactions, Total Transactions Completed, Timeout Indications, and MPS (Messages Per Second) were tracked in real-time. Custom logic was introduced to detect frozen transactions and monitor test failures dynamically during high-throughput scenarios.

5.2.6 Marketplace Module Standardization and Upload

Efforts were also directed toward structuring and standardizing test modules for upload to the Regal Marketplace. This involved organizing all test assets—apps, configuration files, scenarios—into version-controlled .rp and .rtm files. Modules such as Ro Blackbox, Gx Blackbox, and Seagull were uploaded with proper naming conventions, metadata tags, and descriptions to ensure consistency, discoverability, and reuse by QA and development teams.

5.2.7 Infrastructure Debugging and Bug Resolution

During the course of the internship, several critical infrastructure issues were encountered and resolved. Notably, Bug 17557 caused delayed reflection of test case logic in Regal, and Bug 17668 resulted in inconsistent validation behavior across test suites. These bugs were investigated, reproduced, and addressed in collaboration with the Regal Dev Team. Additional efforts went into optimizing scanner pod execution and improving deployment stability.

5.2.8 Knowledge Transfer and Documentation

To ensure continuity and ease of onboarding for future contributors, detailed documentation was maintained throughout the internship. This included technical notes on plugin structure, helper utilities, and test case workflows, as well as the step-by-step process for uploading

modules to the Marketplace. Knowledge transfer sessions were conducted to support peers in setting up their environments and understanding the Regal integration framework.

5.3 Summary

These objectives were achieved through a structured and incremental workflow, contributing to higher efficiency in Diameter test execution, improved automation coverage, and better test resource management within Telaverge's product and QA ecosystems. The internship not only addressed gaps in existing methods but also laid the foundation for scalable, reusable, and production-ready automation infrastructure.

Chapter 6

SYSTEM DESIGN & IMPLEMENTATION

6.1 Introduction

This chapter describes the architectural design, components, and implementation approach followed during the internship project. The project involved the integration of the DiamonPro Diameter stack and the Seagull simulator with Regal, Telaverge's automation platform. Additionally, it involved designing workflows for multi-interface execution, real-time validation, and Marketplace module management.

6.2 System Architecture Overview

The system architecture designed and implemented during the internship is composed of five core components that work together to deliver a robust, automated, and scalable Diameter testing environment.

The Regal Automation Framework forms the central orchestration layer of the system. It is responsible for automating infrastructure deployment, executing test cases, collecting logs, and performing result validation. This framework serves as the backbone for coordinating all tasks related to Diameter protocol testing and simulator integration.

The DiamonPro Stack (v7.9.6) is a commercial-grade Diameter protocol implementation that supports multiple interfaces, including S6a, Sh, and Cx. It serves as the primary system under test, providing realistic Diameter behavior for validation and benchmarking purposes. The stack was integrated with Regal to support both TCP and SCTP transport mechanisms, along with multi-interface execution.

The Seagull Simulator plays a crucial role in traffic generation. As an open-source tool, Seagull can simulate Diameter message exchanges by interpreting structured XML scenario files. During the internship, Seagull was enhanced through plugin development to integrate seamlessly with the Regal platform, supporting both client and server roles in test scenarios.

The Regal Marketplace component is a repository designed to manage and deploy reusable test modules. It supports version-controlled artifacts such as .rp (Regal Packages) and .rtm (Regal Test Modules), enabling structured sharing and lifecycle management of test assets across teams. This marketplace greatly simplifies the process of module reuse, rollback, and discoverability.

Finally, the Kubernetes Infrastructure handles the containerized deployment of Regal jobs and services across virtual machines. It ensures scalability, isolation, and consistent resource management during high-throughput or long-duration test executions. Kubernetes was essential in managing Regal job lifecycles and pod orchestration, especially during performance and multi-interface testing.

Together, these components formed a tightly integrated architecture capable of supporting functional, regression, and performance testing for Diameter protocol stacks in both development and production-like environments.

6.3 Component Design and Flow

6.3.1 Regal Test Orchestration

The Regal framework serves as the central engine for test orchestration, handling the complete lifecycle of Diameter protocol validation. It manages infrastructure provisioning through Kubernetes jobs, enabling dynamic setup and teardown of test environments. Once the infrastructure is live, Regal coordinates the execution of test suites and continuously monitors real-time test metrics. Validation and logging are carried out using custom Python modules tailored to the Diameter use case. Each test case in the framework comprises multiple artifacts: a testcase.py file containing the validation and execution logic, a testcase_conf.json for runtime configuration parameters, and supporting XML configuration files such as itsdiaclient.xml for Diameter stack or simulator setup.

6.3.2 DiamonPro Stack Integration

The integration of the DiamonPro stack into Regal was achieved using Ansible-based automation for installation and configuration. The stack supported validation of critical Diameter interfaces like S6a, Sh, and Cx, operating over both TCP and SCTP transport layers. For running scenarios, the stack utilized predefined scripts such as Run.client and Run.server. Performance statistics and logs were extracted using DBGConsole, which facilitated validation of standard Diameter procedures such as CER/CEA, LIR/LIA, ULR/ULA, and PUR/PUA. The metrics collected included session counts, message throughput (MPS), and timeout indications to confirm the expected behavior of each interface under test.

6.3.3 Seagull Plugin Development

A dedicated plugin for the Seagull simulator was developed and integrated with the Regal automation environment. Several core modules were implemented: seagull_repo_plugin.py allowed package detection and listing through the Regal UI; seagull_tool.py managed

installation and cleanup tasks using Ansible automation; and seagull_helper.py was responsible for remote execution of Diameter scenario XMLs and result collection. The plugin supported key capabilities such as remote tarball deployment, automated scenario execution (including CER/CEA and CCR/CCA flows), and seamless integration into the Regal workflow, making Seagull a fully automated and reusable testing component.

6.3.4 Multi-Interface Automation

To simulate realistic telecom scenarios, logic was developed to enable multi-interface execution for S6a, Sh, and Cx within the same test flow. This was accomplished by modifying Jinja templates to dynamically render configurations like itsdiaclient.xml, testcase_conf.json, and Run.client/server files for each interface. Furthermore, multi-homing and multi-streaming support was added to SCTP-based configurations, enabling robust validation of failover and load balancing behavior. Interface coordination was maintained through the use of shared session IDs and interface-specific metrics. Additionally, frozen transaction detection was implemented—flagging transactions inactive for over five minutes to ensure test accuracy.

6.3.5 Marketplace Module Upload

To standardize and streamline test asset management, modules were structured into Regal Packages (.rp) and Regal Test Modules (.rtm). This included assets for Ro Blackbox, Gx Blackbox, and Seagull-based test cases. Each module adhered to a standard naming convention, version tagging, and detailed metadata such as descriptions and category tags. After validation, these modules were uploaded to the Regal Marketplace via its web interface. Version control and rollback support were incorporated to ensure traceability, allowing teams to reuse modules confidently across different test cycles.

6.4 Implementation Workflow

The implementation of the Diameter testing automation project followed a well-defined, step-by-step workflow to ensure systematic integration and validation across all components. The first step involved environment setup, where the DiamonPro stack was installed and configured on virtual machines. Both TCP and SCTP interfaces were enabled to allow for flexible protocol testing. Once the environment was established, the next phase was regression test execution. This included the manual execution of functional test cases to validate the basic message flows and identify any issues. Debugging was performed using captured packet traces (pcap files)

and stack-level logs extracted through DBGConsole, enabling precise root cause analysis of failures.

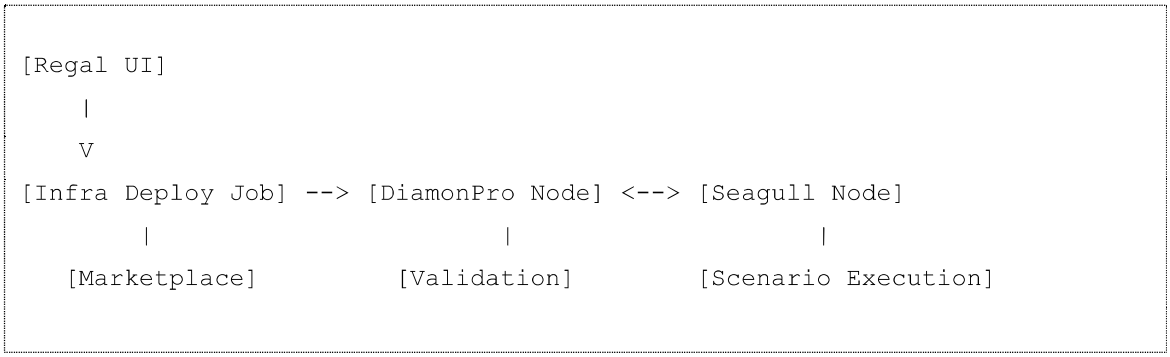
Following regression validation, the focus shifted to plugin development. A custom Seagull plugin was developed and tightly integrated into the Regal platform, facilitating automated deployment, configuration, and execution of Diameter traffic flows. Once the plugin was in place, the next major phase centered on automation and validation. Jinja templates and Python-based helper scripts were implemented to dynamically render configuration files, and runtime validation logic was added to track live test statistics such as Ongoing Transactions, Completed Transactions, and Timeout Counts.

The final step in the workflow was Marketplace structuring, where all validated test modules were packaged using standardized formats. These packages were uploaded to the Regal Marketplace, ensuring centralized access, version control, and reuse of test artifacts across different teams and test cycles.

6.5 Tools and Technologies Used

Tool/Technology	**Purpose**
DiamonPro	Diameter stack with S6a, Sh, and Cx support
Seagull	Traffic generator for Diameter scenarios
Regal	Automation platform for deployment and validation
Ansible	Automated stack and simulator provisioning
Jinja	Dynamic rendering of test configurations
Kubernetes	Deployment and orchestration of Regal services
DBGConsole	Runtime statistics and error logging
Wireshark	Protocol-level pcap capture and analysis

6.6 Sample System Flow Diagram



6.7 Summary

The system was implemented using a modular, plugin-driven approach to enable seamless automation, validation, and deployment of Diameter test cases across multiple interfaces. Through the integration of Seagull, DiamonPro, and the Regal Marketplace, the project achieved scalable, reusable, and production-grade automation capabilities. Each component contributed to improving the efficiency, coverage, and accuracy of protocol testing at Telaverge.

Chapter 7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)

7.1 Overview

The internship officially began on 8th January 2025 after a month of preliminary self-study, stack exploration, and college exams. The internship was executed in milestone-based phases, focusing on Diameter protocol testing, stack integration, plugin development, automation, and Marketplace management. Tasks were accomplished using Python, Ansible, YAML, Jinja templates, and the Regal automation platform.

7.2 Weekly Milestone Breakdown

Week	Milestones
Week 0	Stack familiarization, exam preparation, RFC 6733 reading
Week 1	Studied Diameter protocol and RFC 6733; initial VM setup
Week 2	QA for Diameter Regression test cases; executed CER/CEA, RIR/RIA, LIR/LIA
Week 3-5	DiamonPro integration with Regal; setup of infra, plugins, helper structure
Week 6	Seagull plugin development and deployment completed
Week 7-8	Validated Seagull-DiamonPro test cases; wrote validation script, fixed errors
Week 9-10	Marketplace upload (apps/modules); created .rp/.rtm files; fixed test failures
Week 11-12	Finalized Marketplace updates; handled bugs (17557/17668); uploaded to server
Week 13-14	DiamonPro 7.9.6 Multi-interface integration; SCTP setup and validation
Week 15-16	Performance testing (TCP/SCTP); overnight run; CLR/CLA testing; ongoing QA
Week 17+	Continuing QA for Load Balancer feature and advanced validations

7.3 Gantt chart style view

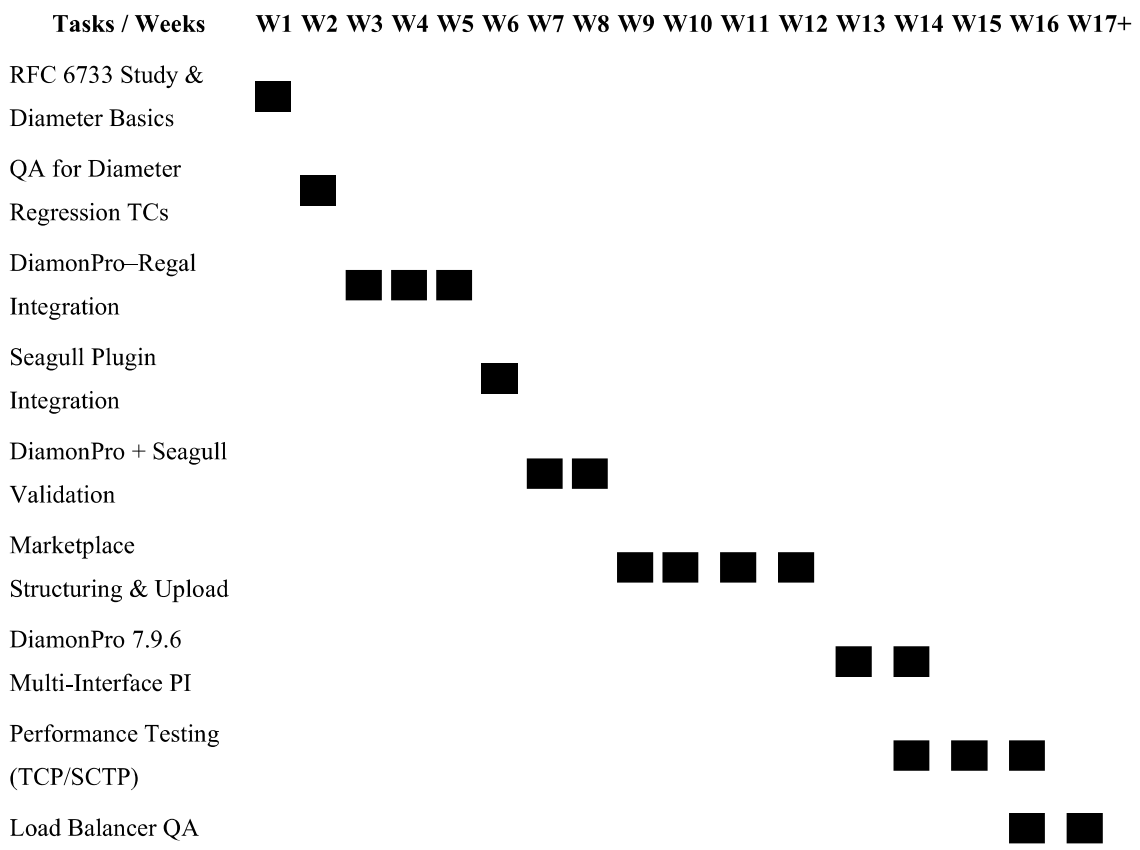


Figure 1.5 Gantt Chart

7.4 Summary

The internship execution plan followed a well-structured weekly milestone-based approach. The Gantt chart outlines all critical phases of learning, execution, integration, and validation. As of now, the project continues with advanced Load Balancer QA and performance tuning. The execution timeline enabled the candidate to balance learning and delivery efficiently while also managing collaborative contributions across teams and tools.

Chapter 8

OUTCOMES

8.1 Overview

The internship at Telaverge Communications Pvt. Ltd. provided a strong foundation in Diameter protocol automation, system integration, and telecom validation frameworks. Over the course of this project, substantial technical progress was achieved in the areas of plugin development, multi-interface test automation, regression execution, and Marketplace standardization. Each module contributed directly to improving the efficiency, reliability, and scalability of Diameter testing infrastructure at the organization.

8.2 Key Outcomes

8.2.1 Mastery of Diameter Protocol and RFC 6733

During the internship, a significant focus was placed on mastering the Diameter protocol as defined in RFC 6733. I acquired an in-depth understanding of its architecture, the role of AVPs (Attribute-Value Pairs), command codes, and the usage of both TCP and SCTP as transport mechanisms. This foundation enabled me to validate various 3GPP-defined interfaces, including S6a, Sh, and Cx. I also gained hands-on experience working with grouped AVPs, managing session-based routing, interpreting message flows, and handling error responses effectively.

8.2.2 Successful Execution of Regression Test Cases

Another core achievement was the manual execution and validation of multiple regression test cases. These included message exchanges such as CER/CEA, ULR/ULA, LIR/LIA, and PUR/PUA. I performed detailed quality assurance for both functional and performance scenarios. Validation was reinforced using Wireshark packet captures and output from DBGConsole to ensure that each test case aligned with the expected protocol behaviors and standards.

8.2.3 Seamless Integration of DiamonPro Stack with Regal

The DiamonPro v7.9.6 Diameter stack was successfully integrated with the Regal automation framework. I automated the generation of stack configuration files using Jinja templates, and implemented execution logic through Run.client and Run.server scripts. XML-based

configurations were used to manage interface-specific behaviors, enabling single and multi-interface test case execution. This integration enabled smooth orchestration of stack components within Regal.

8.2.4 Plugin Development for Seagull Simulator

I designed and developed a dedicated plugin to integrate Seagull—a powerful Diameter traffic simulator—into the Regal framework. The plugin supported automated installation, configuration, and execution of Seagull using Ansible playbooks. Supporting helper scripts were written to enable seamless execution of scenario-based XML files for simulating CER/CEA and CCR/CCA flows. This ensured that Seagull could be used reliably in both client and server roles within test setups.

8.2.5 Multi-Interface Automation with TCP/SCTP

Multi-interface automation was implemented to simulate real-world telecom environments. I enabled Regal to concurrently execute test cases across S6a, Sh, and Cx interfaces, using both TCP and SCTP protocols. Special emphasis was placed on SCTP features such as multi-homing and multi-streaming. Validation logic was introduced to track session integrity and detect conditions such as timeout mismatches and frozen transactions, which occur when transactions remain idle for extended periods.

8.2.6 Load and Performance Testing

High-load and stress testing formed a critical part of the internship. I successfully executed large-scale test scenarios involving up to 5 million callflows with performance benchmarks ranging between 10k and 20k TPS. These tests helped validate system behavior under high-load conditions. I also monitored and logged real-time metrics such as Session Count, Ongoing Transactions, Completed Transactions, and Messages Per Second (MPS) to verify stability and performance.

8.2.7 Marketplace Module Management and Upload

To ensure reusable and version-controlled deployment of test modules, I structured various applications, plugins, and test cases into standardized .rp and .rtm files. These were uploaded to the Telaverge Marketplace with appropriate version tags, naming conventions, and metadata. This made test assets easy to discover, trace, and deploy across teams. Collaboration with Regal QA also ensured that uploaded modules met quality standards and rollback support requirements.

8.2.8 Bug Resolution and Infra Stabilization

Throughout the internship, I played an active role in identifying and resolving infrastructure and validation bugs. Notably, I reported and addressed Bug 17557, which caused delays in test case reflection, and Bug 17668, which led to inconsistencies across test suites. Additionally, I helped debug deployment issues involving Kubernetes scanner pods and Regal job execution delays. These efforts contributed to the overall stability and robustness of the test environment.

8.3 Additional Contributions

In addition to the core technical contributions, I actively participated in several knowledge transfer (KT) sessions throughout the internship. These sessions provided opportunities to assist new team members in setting up their environments and familiarizing themselves with the usage of various plugins. I also proposed internal improvements aimed at enhancing the organization of the Regal Marketplace, specifically suggesting better naming conventions and infra isolation strategies to support stable demonstration environments. Furthermore, I contributed to collaborative debugging efforts by closely working with cross-functional teams including Regal Development, Quality Assurance, and Infrastructure Support. This cross-team collaboration was instrumental in resolving complex deployment issues and accelerating progress across multiple test workflows.

8.4 Summary

The outcomes of this internship demonstrate substantial contributions across the entire Diameter testing pipeline—ranging from protocol validation to system automation and infrastructure integration. These efforts not only enhanced the capabilities of Regal as an automation platform but also introduced scalable workflows for future Diameter protocol validation projects within the organization.

Chapter 9

RESULTS AND DISCUSSIONS

9.1 Introduction

This chapter presents the results obtained from the tasks executed during the internship and discusses the significance of each milestone. The project was focused on enhancing the automation, integration, and validation of the Diameter protocol stack using the Regal platform. The key results were derived through rigorous test case execution, plugin development, performance benchmarking, and system integration efforts across multiple network interfaces and simulators.

9.2 Results Achieved

9.2.1 Functional Test Case Validation

As part of the functional validation effort, I successfully executed over 25 regression test cases across S6a, Sh, and Cx interfaces using the DiamonPro stack. Each test case focused on validating standard Diameter message flows such as CER/CEA for capability negotiation, ULR/ULA for user location updates, and LIR/LIA and PUR/PUA for subscriber and profile interactions. The test executions were meticulously verified using tools like Wireshark and DBGConsole to ensure protocol-level correctness and full compliance with Diameter standards.

9.2.2 DiamonPro and Seagull Integration with Regal

A major highlight of the internship was the seamless integration of both the DiamonPro v7.9.6 stack and the Seagull simulator into the Regal framework. This integration involved dynamic configuration generation using Jinja templates, Python-based validation logic, and Kubernetes-based test orchestration. I developed and deployed a dedicated plugin for Seagull, enabling automated installation, configuration, and execution of Diameter traffic. This integration facilitated end-to-end simulation of CER/CEA and CCR/CCA flows, where Seagull and DiamonPro could operate as interchangeable clients or servers.

9.2.3 Performance and Load Testing Results

To evaluate system behavior under high traffic conditions, I conducted extensive performance testing using the Regal framework. Test scenarios involved user session volumes ranging from

1 million to 5 million, with call rates varying between 500 and 2000 CPS. The TCP-mode tests achieved impressive throughput, reaching up to 20,000 TPS. During these tests, validation logic captured key runtime metrics such as Session Count, Ongoing Transactions, Completed Transactions, and Messages Per Second (MPS). In SCTP mode, the tests succeeded at lower volumes but revealed frozen transactions at higher loads (5 million sessions), leading to further investigation and optimization within the stack.

9.2.4 Marketplace Contributions

I played a central role in structuring and contributing modules to the Telaverge Marketplace. These included .rp and .rtm files for standardized test modules like Ro Blackbox, Gx Blackbox, and Seagull simulation suites. Each module was organized with consistent naming conventions, versioning, and descriptive metadata, which enabled reusable, modular test deployment. These contributions streamlined module discoverability and reinforced automation best practices within the team.

9.3 Technical Impact

- Automation Efficiency: Manual configuration and deployment were eliminated through Ansible-based playbooks and plugin logic.
- Validation Accuracy: Real-time performance stats and DBGConsole metrics improved result reliability.
- Modular Design: Marketplace integration ensured standardized, reusable, and version-controlled deployment of test suites.
- System Scalability: Regal's support for SCTP, TCP, and multi-interface configurations enabled production-grade performance testing.

9.4 Discussions

The internship revealed several practical challenges that provided opportunities for in-depth collaboration with product teams, QA engineers, and infrastructure specialists. One major area of concern was infrastructure failures encountered during Regal job deployments. These were often mitigated through strategies such as job isolation, pod restarts, and close coordination with the DevOps team. Another challenge involved versioning conflicts in the Telaverge Marketplace, which prompted discussions around improving rollback mechanisms and enforcing naming conventions for better consistency. Integration efforts were occasionally delayed due to bugs like Bug 17557, which caused test case reflection delays in Regal, and Bug

17668, where the same test logic passed in one suite but failed in another due to suite-specific inconsistencies. Additionally, SCTP-based load testing exposed performance bottlenecks related to frozen transactions, leading to deeper technical exploration of multi-streaming logic, queue overflows, and proper subdomain alignment. Through iterative debugging, hands-on testing, and regular feedback loops, each of these issues was carefully investigated, resolved where possible, and documented to guide future improvements in the test automation ecosystem.

9.5 Summary

The internship achieved meaningful technical results across multiple domains—protocol conformance, test case automation, performance benchmarking, and infrastructure tooling. Each milestone contributed to enhancing Regal's testing capability, while also providing robust validation support for Diameter-based products. The internship offered deep technical exposure and contributed to building scalable and reusable test workflows that will continue to benefit the organization.

Chapter 10

CONCLUSION

10.1 Summary of Work

The internship project at Telaverge Communications Pvt. Ltd. focused on automating and enhancing the validation process of the Diameter protocol stack using the in-house Regal automation platform. The work involved:

- Studying the Diameter protocol as defined in RFC 6733.
- Executing regression test cases for interfaces such as S6a, Sh, and Cx.
- Developing and integrating a plugin for the Seagull simulator.
- Automating the execution of DiamonPro v7.9.6 using Regal.
- Supporting multi-interface execution over TCP and SCTP, including advanced features like multi-homing and multi-streaming.
- Designing and uploading structured modules in the Telaverge Marketplace.
- Performing load and performance testing with up to 5 million callflows and 20k TPS.
- Writing validation logic for timeout detection, frozen transactions, and real-time metric tracking.

Each module and phase contributed to building a robust, reusable, and scalable automation framework for Diameter protocol testing.

10.2 Key Learnings

The internship offered hands-on exposure to both telecom protocol internals and automation engineering practices. Key technical and professional learnings include:

- In-depth understanding of Diameter protocol, AVP encoding, message flows, and 3GPP-defined interfaces.

- Practical skills in tools such as Wireshark, DBGConsole, Seagull, and DiamonPro.
- Plugin and helper development using Python, Jinja, and Ansible.
- Deployment automation and test orchestration using Regal and Kubernetes.
- Structured module packaging and lifecycle management through the Regal Marketplace.
- Collaboration with QA, DevOps, and product teams for debugging, infrastructure fixes, and testing enhancements.

10.3 Challenges Faced

Several challenges were encountered and resolved during the course of the internship, each contributing to a deeper understanding of the system and its infrastructure. One of the initial obstacles was the delayed provisioning of virtual machines, which, along with intermittent infrastructure instability, affected early project momentum. During SCTP communication testing, issues such as subnet mismatches and stream misconfiguration posed barriers to successful multi-homing and multi-streaming validation. Notably, Bug 17557 emerged as a recurring issue where test case logic failed to reflect due to caching problems within the Regal infrastructure. Similarly, Bug 17668 caused test cases to pass in one suite and fail in another because of metadata inconsistencies. Additionally, frequent Regal pod crashes and scanner process failures hindered the continuity of automated test executions. High-load SCTP test cases also exposed performance bottlenecks, including queue overflows and unhandled stack behaviors. Time management was another critical challenge, especially when balancing plugin development, test validation, and Marketplace deliverables. Each of these issues was systematically debugged in coordination with various internal teams, and the resolutions were thoroughly documented to support ongoing improvements and knowledge transfer.

10.4 Conclusion

The internship provided a technically enriching and impactful experience in telecom protocol testing, system integration, and QA automation. The successful integration of DiamonPro and Seagull with the Regal platform enabled efficient testing of complex Diameter-based scenarios, supporting both functional and performance validation. The structured deployment of modules via the Marketplace further streamlined the testing ecosystem, making test assets reusable and

maintainable.

This experience strengthened not only technical proficiencies in Python automation, Diameter protocols, and containerized deployment, but also enhanced problem-solving, communication, and cross-team collaboration skills. The foundation laid through this project can serve as a blueprint for scaling automation efforts for other telecom protocols in the future.

REFERENCES

[1]IETF. (2012). Diameter Base Protocol (RFC 6733). Internet Engineering Task Force.

[2] 3GPP. (2021). TS 29.272: Evolved Packet System (EPS); Mobility Management Entity (MME) and Home Subscriber Server (HSS) Diameter Interface (S6a/S6d). 3rd Generation Partnership Project.

[3] 3GPP. (2021). TS 29.329: Sh Interface; Application Function (AF) and Home Subscriber Server (HSS) based on Diameter protocol. 3rd Generation Partnership Project.

[4] 3GPP. (2021). TS 29.228: Cx and Dx interfaces for the IP Multimedia Subsystem (IMS). 3rd Generation Partnership Project.

[5] Wireshark Foundation. (2024). Wireshark: Network Protocol Analyzer. [Software].

[6] ETSI. (2009). Seagull: Protocol Traffic Generator for Diameter and SIP. Open Source

[7] Telaverge Communications Pvt. Ltd. (2025). DiamonPro: Commercial Diameter Stack Documentation [Internal Product Document].

[8] Telaverge Communications Pvt. Ltd. (2025). Regal Test Automation Framework Documentation [Internal Tool Manual].

[9] Red Hat. (2024). Ansible Automation Platform: Configuration and Deployment Management. [Software].

[10] Python Software Foundation. (2025). Python 3.x Documentation. [Online]

[11] Telaverge Communications Pvt. Ltd. (2025). Internal KT Sessions and Training Materials: Jan–Apr 2025.

[12] Jakati Vikram, Jeslin Abi, Shaad Anwar, et al. (2025). Mentoring sessions and technical discussions during Diameter automation internship. Telaverge Communications Pvt. Ltd.

APPENDIX-A

PSUEDOCODE

Pseudocode: Diameter Stack Test Execution via Regal Framework

Python

```
# Load test configuration
config = load_testcase_conf("testcase_conf.json")

# Step 1: Deploy Infra (DiamonPro / Seagull) using Regal
for node in config["nodes"]:
    deploy_vm(node)
    configure_network(node)
    install_stack_or_simulator(node)

# Step 2: Render Dynamic Configs using Jinja Templates
for template in config["templates"]:
    rendered_file = render_template(template, config["parameters"])
    push_to_node(rendered_file, destination_path)

# Step 3: Start Stack or Simulator
if config["tool"] == "DiamonPro":
    run_script("Run.client", on="client_node")
    run_script("Run.server", on="server_node")
elif config["tool"] == "Seagull":
    start_seagull_traffic("scenario.xml")

# Step 4: Monitor Stack Statistics
while test_in_progress():
    stats = get_dbgconsole_stats()
    log("Session count: ", stats.session_count)
    log("Ongoing Transactions: ", stats.ongoing_txn)
    log("Completed Transactions: ", stats.completed_txn)
    log("MPS: ", stats.mps)
```

```
# Step 5: Validation Logic
if stats.ongoing_txn == 0 and stats.completed_txn == stats.session_count:
   mark_test_status("PASSED")
elif timeout_exceeded or frozen_transactions_detected():
   mark_test_status("FAILED")

# Step 6: Cleanup and Teardown
terminate_processes()
collect_logs()
generate_summary_report()
```

Explanation of Key Components

- load_testcase_conf: Loads the runtime parameters and test topology.
- deploy_vm / install_stack_or_simulator: Uses Ansible to provision VMs and install DiamonPro or Seagull.
- render_template: Generates itsdiaclient.xml, Run.client, dictionary.xml, etc. from Jinja templates.
- start_seagull_traffic / run_script: Executes actual test flows (e.g., CER/CEA, CCR/CCA).
- get_dbgconsole_stats: Fetches real-time metrics from the Diameter stack via DBGConsole.
- mark_test_status: Validates test result based on matching transaction counts.
- generate_summary_report: Outputs TPS, error logs, pcap files, and verdicts for the test case.

This pseudocode reflects your Regal integration flow across tools like DiamonPro and Seagull and models the high-level automation workflow of Diameter protocol testing.

APPENDIX-B

SCREENSHOTS

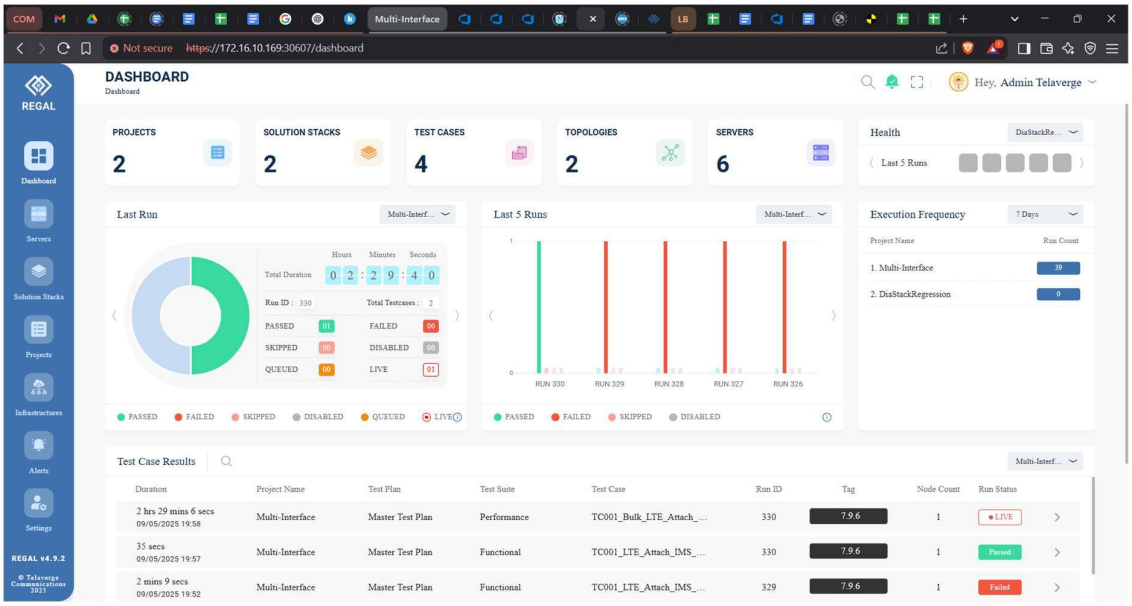


Figure 1.1 Regal Dashboard

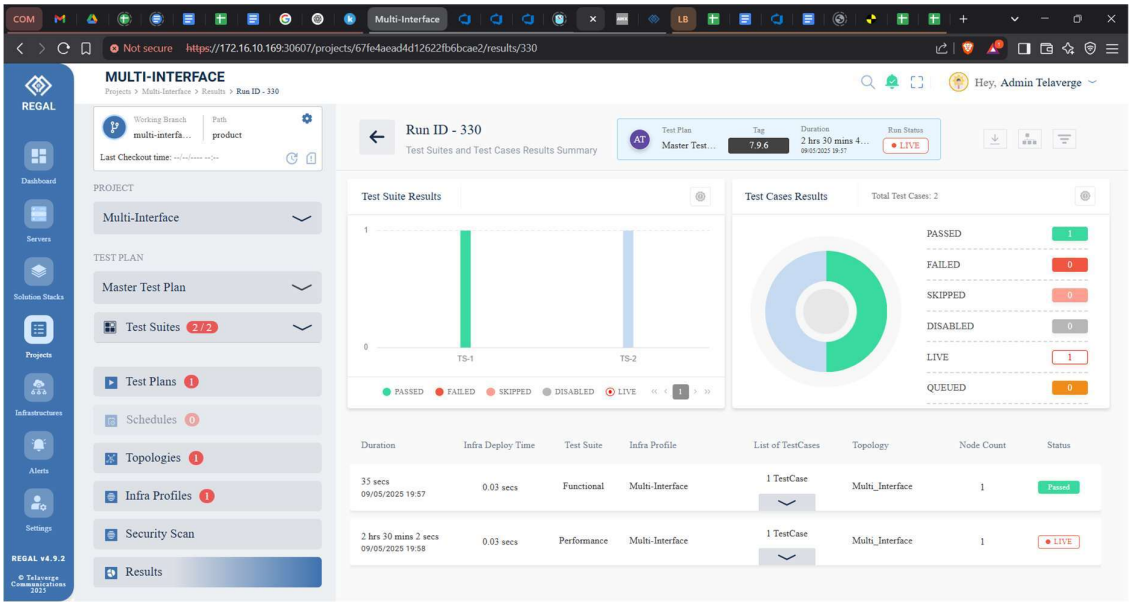


Figure 1.2 Testcase Result

APPENDIX-C

ENCLOSURES

1. Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need for a page-wise explanation.

Dr.Sharmasth Vali Y-Internship_Report_Prince_Gupta.docx

ORIGINALITY REPORT

12%	9%	4%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Presidency University Student Paper	8%
2	Submitted to M S Ramaiah University of Applied Sciences Student Paper	1%
3	en.wikipedia.org Internet Source	<1%
4	eprints.ums.edu.my Internet Source	<1%
5	eprints.usq.edu.au Internet Source	<1%
6	issuu.com Internet Source	<1%
7	www.cs.uah.edu Internet Source	<1%
8	bradscholars.brad.ac.uk Internet Source	<1%
9	Submitted to Symbiosis International University Student Paper	<1%
10	www.telecomtrainer.com Internet Source	<1%
11	fnih.org Internet Source	<1%
12	wbdg.org Internet Source	<1%

2. Details of mapping the project with the Sustainable Development Goals (SDGs).

Figure 2.1 Software Development Goals

The internship aligns with the following Sustainable Development Goals:

SDG 8: Decent Work and Economic Growth

Achieve higher levels of productivity through diversification, technological upgrading, and innovation.

SDG 9: Industry, Innovation, and Infrastructure

Enhance scientific research, upgrade technological capabilities, and encourage innovation.

SDG 17: Partnerships for the Goals

Leveraging global tech tools and fostering cross-team collaboration to achieve project goals.