**ACKNOWLEDGEMENT**

I am indebted to Rashtreeya Sikshana Samithi Trust, Bengaluru for providing me with all the facilities needed for the successful completion of my Major Project work at Rashtreeya Vidyalaya College of Engineering (RVCE) during the tenure of our Course.

I would like to thank Name, Principal, for giving me an opportunity to be a part of RVCE and for his timely help and encouragement during the tenure of the Major Project work.

I am greatly thankful to Name, Professor and Head, Dept. of CSE for her motivation and constant support during the tenure of my Major Project work.

I take this opportunity to convey my sincere gratitude to the internal guide Name, Associate Professor, Dept of CSE for her advice, support and valuable suggestions helped me to accomplish the Major Project work in time.

Special thanks to Name, Dean CSE Cluster and panel members Name, Designation, Name, Designation for their valuable comments, constructive inputs and feedback during the Phase- I, Phase-II and Phase-III presentations.

I extend my regards to all who have directly or indirectly extended their constant support for successful completion of my Major Project work

**Name**

**USN**

**ABSTRACT**

Text

**LIST OF PUBLICATION**

|  |  |
| --- | --- |
| **Publication Type** | **Publication Details** |
| Journal/Conference | Title and Status |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TABLE OF CONTENT** | | | | | |
| **Acknowledgement** | | | | | i |
| **Abstract** | | | | | ii |
| **List of Publication** | | | | | iii |
| **List of Figures** | | | | | viii |
| **List of Tables** | | | | | ix |
| **Glossary** | | | | | x |
| **Chapter 1** | | | | |  |
| **Introduction** | | | | |  |
| 1.1 | Overview | | | | 10 |
| 1.2 | Literature Review | | | | 11 |
| 1.3 | Motivation | | | | 16 |
| 1.4 | Problem statement | | | | 16 |
| 1.5 | Objectives | | | | 17 |
| 1.7 | Methodology | | | | 18 |
| 1.8 | Organization of report | | | | 21 |
| **Chapter 2** | | | | |  |
| **Theory and Concepts of GenAI based Application to Analyze Telemetric Data** | | | | |  |
| 2.1 | | Generative AI and Large Language Model | | | 22 |
| 2.2 | | Retrieval Augmented Generation | | | 23 |
| 2.3 | | Embedding Models | | | 24 |
| 2.4 | | Vector Databases | | | 25 |
| 2.5 | | Prompt Engineering and Query Processing | | | 25 |
| 2.6 | | Telemetric Diagnostic Systems | | | 26 |
| 2.7 | | Evaluation Metrics | | | 27 |
| 2.8 | | Summary | | | 28 |
| **Chapter 3** | | | | | |
| **Requirement Specification** | | | |  | |
| 3.1 | | | Software Requirements | 29 | |
| 3.2 | | | Hardware Requirements | 31 | |
| 3.3 | | | Functional Requirements | 31 | |
| 3.4 | | | Performance Requirements | 32 | |

|  |  |  |
| --- | --- | --- |
| **Chapter 4** | | |
| **High Level Design Specification of the GenAI powered telemetry analysis system** | |  |
| 4.1 | Design Considerations | 34 |
|  | 4.1.1 General Considerations | 35 |
|  | 4.1.2 Development Methods | 35 |
| 4.2 | Architectural Strategies | 36 |
|  | 4.2.1 Data Chunking and Context Preservation | 36 |
|  | 4.2.2 Embedding Model Evaluation and Selection | 36 |
|  | 4.2.3 Retrieval Pipeline Organization | 37 |
|  | 4.2.4 System Integration and Modularity | 37 |
|  | 4.2.5 Performance Monitoring and Evaluation | 37 |
| 4.3 | System Architecture | 38 |
|  | 4.3.1 Data Ingestion and Preprocessing | 39 |
|  | 4.3.2 Embedding and Vectorization Layer | 39 |
|  | 4.3.3 Vector Databases | 39 |
|  | 4.3.4 Prompt Construction and LLM Response | 39 |
|  | 4.3.5 User Interface and Orchestration | 39 |
|  | 4.3.6 Deployment and Scalability Considerations | 40 |
| 4.4 | Data Flow Diagrams | 40 |
|  | 4.4.1 Data Flow Diagram Level 0 | 40 |
|  | 4.4.2 Data Flow Diagram Level 1 | 41 |
|  | 4.4.3 Data Flow Diagram Level 2 for Data Processing | 41 |
|  | 4.4.4 Data Flow Diagram Level 2 for Vectorization and Embedding | 42 |
|  | 4.4.5 Data Flow Diagram Level 2 for Retrieval and Augmentation | 43 |
| 4.6 | Summary | 44 |
|  |  |  |
| **Chapter 5** | |  |
| **Detailed Design of GenAI powered telemetry analysis system** | | |
| 5.1 | Structure Chart | 45 |
| 5.2 | Module Description | 46 |
|  | 5.2.1 Telemetric Data Collection | 47 |
|  | 5.2.2 Data Parsing and Processing | 47 |
|  | 5.2.3 Semantic Chunking | 48 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | 5.2.4 Sentence Embedding | 49 |
|  | | 5.2.5 Store to Vector Database | 49 |
|  | | 5.2.6 Similarity Search for Relevant Data | 49 |
|  | | 5.2.7 Prompt Augmentation | 50 |
|  | | 5.2.8 Response Invocation through LLM | 50 |
| 5.3 | | Summary | 51 |
| **Chapter 6** | | | |
| **Implementation of GenAI powered telemetry analysis system** | | |  |
| 6.1 | Programming Language Selection | | 53 |
| 6.2 | Development Environment Selection | | 53 |
|  | 6.2.1 Jupyter Notebook | | 53 |
|  | 6.2.2 Anaconda | | 53 |
|  | 6.2.3 LangChain | | 54 |
|  | 6.2.4 ChromaDB Vector Store | | 54 |
|  | 6.2.5 Ollama | | 54 |
|  | 6.2.6 Embedding Models | | 54 |
| 6.3 | Detailed Workflow of GenAI powered telemetry analysis system | | 55 |
|  | 6.3.1 Data Ingestion | | 55 |
|  | 6.3.2 Data Parsing and Cleaning | | 55 |
|  | 6.3.3 Semantic Chunking | | 55 |
|  | 6.3.4 Embedding Generation | | 56 |
|  | 6.3.5 Vector Storage | | 56 |
|  | 6.3.6 Query Embedding | | 56 |
|  | 6.3.7 Similarity Search | | 56 |
|  | 6.3.8 Prompt Construction | | 57 |
|  | 6.3.9 LLM Response Generation | | 57 |
| 6.4 | Summary | | 57 |
| **Chapter 7** | | |  |
| **Software Testing of the GenAI powered telemetry analysis system** | | |  |
| 7.1 | Testing Strategy | | 58 |
| 7.2 | Unit Testing | | 59 |
|  | 7.2.1 Test Case for Telemetry Parsing Module | | 60 |
|  | 7.2.1 Test Case for Semantic Chunking Module | | 60 |
|  | 7.2.1 Test Case for Embedding Generation Module | | 61 |
|  | 7.2.1 Test Case for Vector Store Module | | 61 |
| 7.3 | Integration Module | | 62 |
|  | 7.3.1 Parse to Chunk Integration Test | | 62 |
|  | 7.3.1 Chunk to Embedding Integration Test | | 63 |
|  | 7.3.1 Embedding to VectorDB Integration Test | | 63 |
|  | 7.3.1 Query to Prompt Integration Test | | 64 |
|  | 7.3.1 End to End Pipeline Test | | 64 |
| 7.4 | System Testing | | 65 |
|  | 7.4.1 End to End Query to Response Validation | | 66 |
| 7.5 | Summary | | 67 |
| **Chapter 8** | | |  |
| **Experimental Results and Analysis** | | |  |
| 8.1 | Evaluation Metrics | | 68 |
| 8.2 | Chunk Size Evaluation | | 70 |
| 8.3 | Embedding Model Evaluation | | 71 |
| 8.4 | Latency and Scalability Evaluation | | 72 |
|  | 8.4.1 Latency vs. File Size | | 73 |
|  | 8.4.2 Latency vs. Concurrent Queries | | 73 |
| 8.5 | LLM Model Selection and Performance | | 74 |
| 8.6 | Model Comparison and Analysis | | 76 |
| 8.7 | Qualitative Observations | | 79 |
| 8.8 | Summary | | 79 |
| **Chapter 9** | | |  |
| **Conclusion** | | | 81 |
| 9.1 | Limitations | | 82 |
| 9.2 | Future Scope | | 82 |
| **REFERENCES** | | | 84 |
| **APPENDICES** | | | 85 |
| **ANNEXURE: Plagiarism Report** | | | 97 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig No.** | **Figure Name** | **Page No.** |
| 1.1 | Proposed system to analyse telemetric data with GenAI | 18 |
| 4.1 | System Architecture | 38 |
| 4.2 | DFD Level 0 | 41 |
| 4.3 | DFD Level 1 | 41 |
| 4.4 | DFD Level 2 for Data Processing | 42 |
| 4.5 | DFD Level 2 for Vectorization and Embedding | 43 |
| 4.6 | DFD Level 2 for Retrieval and Augmentation | 44 |
| 5.1 | Structure Chart | 46 |
| 8.1 | Chunk Size Evaluation | 70 |
| 8.2 | Embedding Model Comparison | 71 |
| 8.3 | Various LLM Performance Comparison | 76 |
| 8.4 | Existing and Proposed Model Comparison | 77 |
| A.1 | Telemetry Query Analysis Response on Resource | 85 |
| A.2 | Telemetry Query Analysis Response on Utilization | 86 |
| A.3 | Telemetry Query Analysis Response on Processes | 87 |
| A.4 | LLM and Embedding Models | 87 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Fig No.** | **Figure Name** | **Page No.** |
| 7.1 | Test Case for Telemetry Parsing Module | 60 |
| 7.2 | Test Case for Semantic Chunking Module | 60 |
| 7.3 | Test Case for Embedding Generation Module | 61 |
| 7.4 | Test Case for Vector Store Module | 61 |
| 7.5 | Parse-to-Chunk Integration Test Case | 62 |
| 7.6 | Chunk-to-Embedding Integration Test Case | 63 |
| 7.7 | Embedding-to-VectorDB Integration Test | 63 |
| 7.8 | Query-to-Prompt Integration Test | 64 |
| 7.9 | End-to-End Pipeline Integration Test Case | 65 |
| 7.10 | Test Case for End-to-End System Testing | 66 |
| 8.1 | Effect of Chunk Size on Retrieval Performance | 70 |
| 8.2 | Effect of Telemetry File Size on Response Latency | 73 |
| 8.3 | Effect of Concurrent Queries on Response Latency | 74 |
| 8.4 | Comparative Evaluation of RAG System | 78 |

|  |  |  |
| --- | --- | --- |
|  |  | **GLOSSARY** |
|  |  |  |
| AI | : | Artificial Intelligence |
| LLM | : | Large Language Models |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**INTRODUCTION TO PROJECT TITLE**

This chapter presents the introduction, background study, motivation, objectives, problem statement, and methodology of the project. It focuses on the <text>. The project explores the <concepts>. In this chapter, the foundational concepts and the proposed methodology for building an <project> is discussed.

* 1. **OVERVIEW**

Text

* 1. **LITERATURE REVIEW**

Text

* 1. **MOTIVATION**

Text

* 1. **PROBLEM STATEMENT**

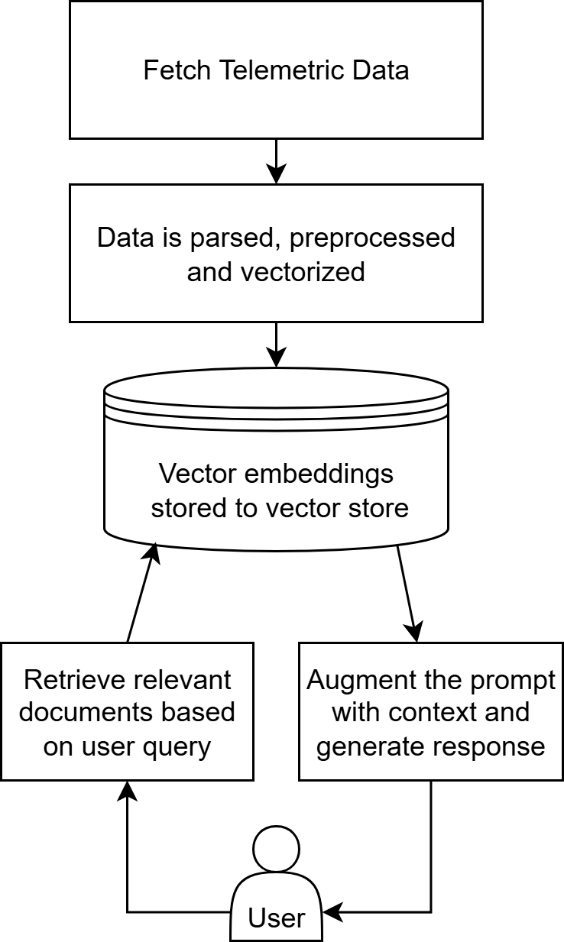
Text

* 1. **OBJECTIVES**

Text

* 1. **METHODOLOGY**

Text



**Figure 1.1: Proposed system to analyse telemetric data with GenAI**

* 1. **ORGANIZATION OF REPORT**

Chapter 2 presents the theoretical background and foundational concepts related to <project>.

Chapter 3 outlines the system requirements, including both hardware configurations and software tools essential for developing and deploying the telemetry analysis system.

Chapter 4 details the high-level design and architectural overview of the proposed <project>

Chapter 5 elaborates on the detailed design aspects of <project>

Chapter 6 provides a comprehensive description of the system implementation, including the selected tools, programming environments, model choices, and development workflow.

Chapter 7 discusses the testing strategy adopted, including unit testing, integration testing, and system-level testing to validate the functionality and reliability of the GenAI the solution.

Chapter 8 presents the experimental results and performance analysis. This includes evaluations of retrieval accuracy, latency, model comparisons, and various metrics to assess the effectiveness of the system.

Chapter 9 concludes the report with key findings, project limitations, and suggestions for future enhancements to further improve the scalability.

**THEORY AND CONCEPTS OF PROJECT TITLE**

This chapter provides an outline of the theoretical foundations and methodologies involved in building a Generative AI system for the interpretation of diagnostic telemetry data. The project sits at the intersection of artificial intelligence, system diagnostics, and natural language processing, with the goal of transforming complex, XML-based system logs into meaningful, human-readable insights. It incorporates principles from information retrieval, vector semantics, and prompt-based language modeling. The following sections elaborate on the core concepts and technologies that underpin the development of this intelligent diagnostic framework.

* 1. **GENERATIVE AI AND LARGE LANGUAGE MODEL**

Text

* 1. **RETRIEVAL AUGMENTED GENERATION**

Text

* 1. **EMBEDDING MODELS**

Text

* 1. **VECTOR DATABASES**

Text

* 1. **PROMPT ENGINEERING AND QUERY PROCESSING**

Text

* 1. **TELEMETRIC DIAGNOSTIC SYSTEMS**

Text

* 1. **EVALUATION METRICS**

Text

* 1. **SUMMARY**

Text

**REQUIREMENT SPECIFICATION OF PROJECT TITLE**

This chapter outlines the essential software and hardware prerequisites necessary for implementing and running the GenAI-based telemetry analysis system. Given the complexity of processing large volumes of diagnostic data and running resource-intensive language models, careful selection of tools and platforms is critical for system stability and performance.

* 1. **SOFTWARE REQUIREMENTS**

Text,

* Text
* Text
  1. **HARDWARE REQUIREMENTS**

Text

* 1. **FUNCTIONAL REQUIREMENTS**

Text

* 1. **PERFORMANCE REQUIREMENTS**

Text

**HIGH LEVEL DESIGN SPECIFICATION OF PROJECT TITLE**

This chapter presents a comprehensive overview of the high-level design of the GenAI-powered telemetry interpretation framework. The system leverages large language models, semantic search, and vector databases to transform raw XML-based diagnostic logs into intelligible, actionable insights. The design incorporates considerations around real-time performance, modularity, and deployment constraints. The architecture is guided by best practices for building scalable, explainable, and domain-adapted GenAI solutions.

* 1. **DESIGN CONSIDERATIONS**

Text

* + 1. **General Considerations**

Text

* + 1. **Development Methods**

Text

* 1. **ARCHITECTURAL STRATEGIES**

Text

**4.2.1 Data Chunking and Context Preservation**

Text

**4.2.2 Embedding Model Evaluation and Selection**

Text

**4.2.3 Retrieval Pipeline Optimization**

Text

**4.2.4 System Integration and Modularity**

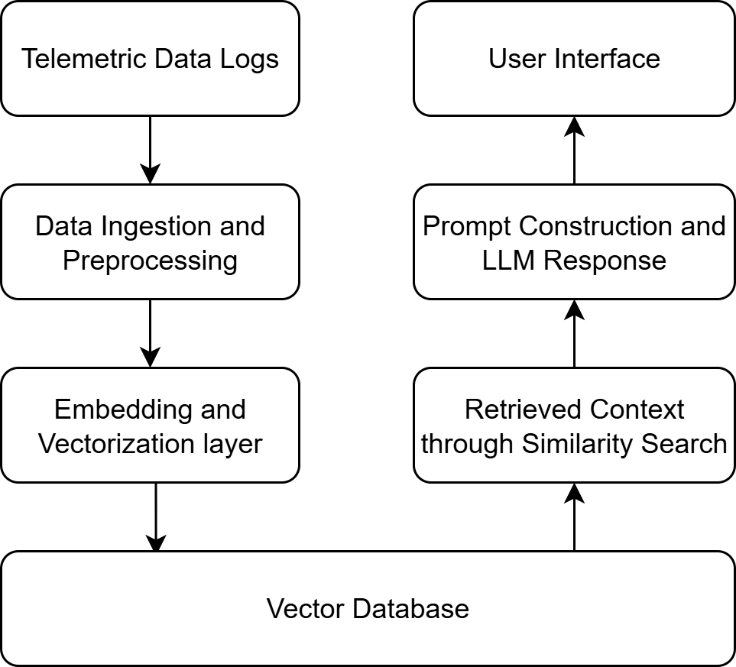
Text

**4.2.5 Performance Monitoring and Evaluation**

Text

* 1. **SYSTEM ARCHITECTURE**

Text



**Figure 4.1: System Architecture**

**4.3.1 Data Ingestion and Preprocessing**

Text

**4.3.2 Embedding and Vectorization Layer**

Text

**4.3.3 Vector Database**

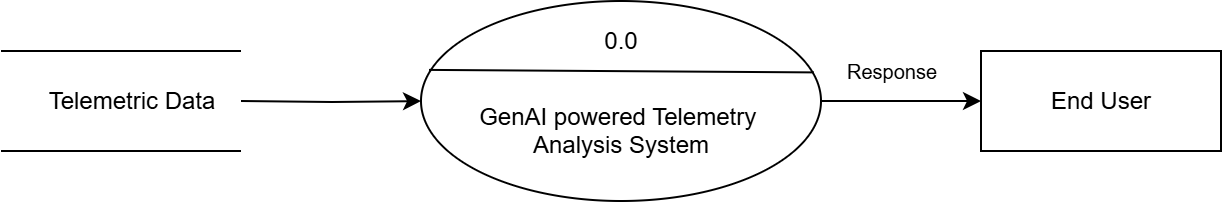
Text

* 1. **DATA FLOW DIAGRAMS**

Data Flow Diagrams (DFDs) provide a structured way to visualize how data moves through the system. They help in understanding how the different components interact, how input data is processed, and how outputs are generated. For this project, DFDs are structured across three levels, representing increasing detail at each stage of the telemetry-to-response pipeline.

**4.4.1 Data Flow Diagram Level 0**

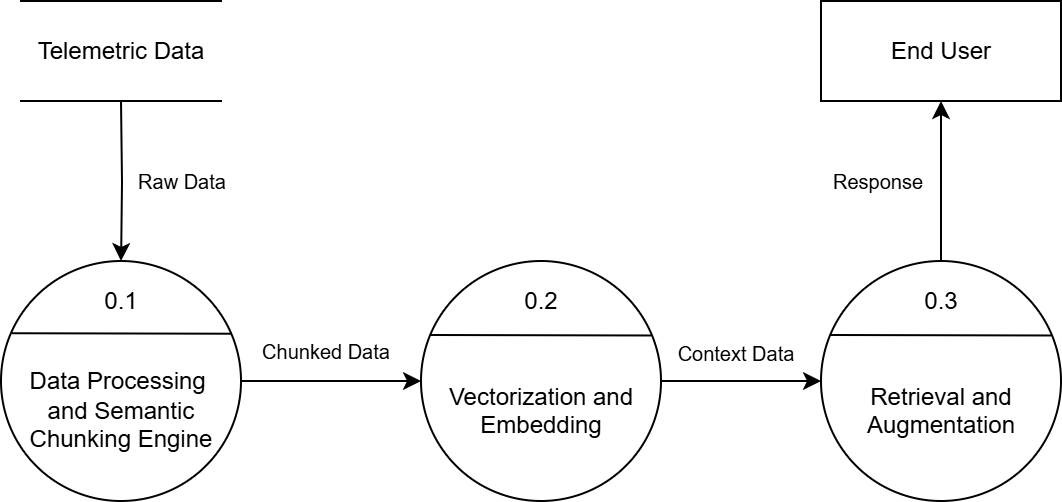
Figure 4.2 below presents the Level 0 Data Flow Diagram, which illustrates the overall GenAI-powered diagnostic system as a single unified process.



**Figure 4.2: DFD Level 0**

**4.4.2 Data Flow Diagram Level 1**

Figure 4.3 below represents the Level 1 Data Flow Diagram, illustrating the overall high-level process of the system.



**Figure 4.3: DFD Level 1**

**4.4.3 Data Flow Diagram Level 2 for Data Processing**

Figure 4.4 presents the Level 2 Data Flow Diagram for Module Data Processing and Semantic Chunking Engine.

A diagram of a cleaning and data formatting

AI-generated content may be incorrect.

**Figure 4.4: DFD Level 2 for Data Processing**

**4.4.4 Data Flow Diagram Level 2 for Vectorization and Embedding**

The Figure 4.5 presents the Level 2 Data Flow Diagram for the Vectorization and Embedding module.

A diagram of a data store

AI-generated content may be incorrect.

**Figure 4.5: DFD Level 2 for Vectorization and Embedding**

**4.4.5 Data Flow Diagram Level 2 for Retrieval and Augmentation**

Figure 4.6 illustrates the Level 2 Data Flow Diagram for the retrieval and augmentation process.

A diagram of a response

AI-generated content may be incorrect.

**Figure 4.6: DFD Level 2 for Retrieval and Augmentation**

* 1. **SUMMARY**

This chapter presented the high-level design specification of the LLM-powered diagnostic recommendation system, detailing the core architectural strategies, system components, and the flow of data through various processing stages. The system leverages a modular, Retrieval-Augmented Generation (RAG) architecture that ensures scalability, efficiency, and context-aware response generation. Data flow diagrams at multiple levels were provided, illustrating how telemetry data is parsed, embedded, stored, retrieved, and transformed into human-readable insights. These diagrams clarify the operational hierarchy from raw data ingestion to final LLM-driven inference supporting transparency and maintainability across the system.

**DETAILED DESIGN OF PROJECT TITLE**

Text

* 1. **STRUCTURE CHART**

A structure chart in software engineering represents the breakdown of a system into its most fundamental components. This chart is pivotal in structured programming as it organizes program modules into a hierarchical tree structure. Each box in the structure chart corresponds to a distinct module, labelled with its specific function.

A diagram of a structure

AI-generated content may be incorrect.

**Figure 5.1: Structure Chart**

Text

* 1. **MODULE DESCRIPTION**

Text

**5.2.1 Telemetric Data Collection**

Text

**5.2.2 Data Parsing and Processing**

Text

**5.2.3 Semantic Chunking**

Text

**5.2.4 Sentence Embedding**

Text

**5.2.5 Store to Vector Database**

Text

**5.2.6 Similarity Search for Relevant Data**

Text

**5.3 SUMMARY**

Text

**IMPLEMENTATION OF PROJECT TITLE**

Text

* 1. **PROGRAMMING LANGUAGE SELECTION**

Text

* 1. **DEVELOPMENT ENVIRONMENT SELECTION**

Text

* 1. **DETAILED WORKFLOW OF PROJECT TITLE**

Text

**6.4 SUMMARY**

Text

**SOFTWARE TESTING OF THE PROJECT TITLE**

This chapter presents a detailed overview of the testing procedures carried out for <project>

**7.1 TESTING STRATEGY**

Text

**7.2 UNIT TESTING**

Text

**7.2.1 Test Case for Telemetry Parsing Module**

The Telemetry Parsing Module was tested to verify its ability to accurately extract relevant fields such as timestamps, system states, and error codes from raw XML logs. The test case details are shown in Table 7.1.

**Table 7.1: Test Case for Telemetry Parsing Module**

|  |  |
| --- | --- |
| Test Case No. | 1 |
| Test Case ID | TC\_01\_Telemetry\_Parsing |
| Description | Test accurate parsing of XML telemetry logs |
| Dependency | None |
| Expected Result | Extracted fields are structured and valid |
| Actual Result | Log fields correctly parsed and converted to structured data |
| Status | Pass |

**7.2.2 Test Case for Semantic Chunking Module**

This test case validated the semantic chunking logic to ensure logs are grouped into coherent context segments. See Table 7.2 for test case details.

**Table 7.2: Test Case for Semantic Chunking Module**

|  |  |
| --- | --- |
| Test Case No. | 2 |
| Test Case ID | TC\_02\_Semantic\_Chunking |
| Description | Validate log segmentation into semantic units |
| Dependency | Parsed log data |
| Expected Result | Logs grouped into contextually valid chunks |
| Actual Result | Chunking preserved session coherence |
| Status | Pass |

**7.2.3 Test Case for Embedding Generation Module**

The Embedding Module was tested for consistency and correct dimensionality of vector outputs. Table 7.3 outlines the test case.

**Table 7.3: Test Case for Embedding Generation Module**

|  |  |
| --- | --- |
| Test Case No. | 3 |
| Test Case ID | TC\_03\_Embedding\_Generation |
| Description | Generate sentence embeddings from log chunks |
| Dependency | Chunked log data |
| Expected Result | Valid high-dimensional vectors with no anomalies |
| Actual Result | Embeddings generated as expected |
| Status | Pass |

**7.2.4 Test Case for Vector Store Module**

The Vector Store module was tested to ensure successful insertion and retrieval from ChromaDB as shown in Table 7.4 below.

**Table 7.4: Test Case for Vector Store Module**

|  |  |
| --- | --- |
| Test Case No. | 4 |
| Test Case ID | TC\_04\_Vector\_Store |
| Description | Validate insert and retrieval operations |
| Dependency | Embeddings |
| Expected Result | Vectors indexed, retrieved accurately |
| Actual Result | Stored and queried Top-K embeddings successfully |
| Status | Pass |

**7.3 INTEGRATION TESTING**

Text

**7.3.1 Parse-to-Chunk Integration Test**

Tex

**Table 7.5: Parse-to-Chunk Integration Test Case**

|  |  |
| --- | --- |
| Test Case No. | 1 |
| Test Case ID | TC\_01\_Parse\_to\_Chunk |
| Description | Verify compatibility of parsed output with the semantic chunking module |
| Dependency | Data Parsing Module, Chunking Module |
| Expected Result | Parsed logs are successfully chunked without errors |
| Actual Result | Chunking accepted parsed logs and maintained contextual integrity |
| Status | Pass |

**7.3.2 Chunk-to-Embedding Integration Test**

This test checks whether the semantically chunked data is correctly processed by the embedding model without format-related issues. It verifies the seamless transformation from text chunks to high-dimensional vectors.

**Table 7.6: Chunk-to-Embedding Integration Test Case**

|  |  |
| --- | --- |
| Test Case No. | 2 |
| Test Case ID | TC\_02\_Chunk\_to\_Embedding |
| Description | Test if chunked data is accepted by embedding encoder without format issues |
| Dependency | Chunking Module, Embedding Module |
| Expected Result | All chunks converted to vector embeddings |
| Actual Result | Embedding model accepted all inputs and generated high-dimensional vectors |
| Status | Pass |

**7.3.3 Embedding-to-VectorDB Integration Test**

This test validates the proper storage of generated vector embeddings in the ChromaDB vector store. It ensures that the system can persist, index, and retrieve vectors reliably.

**Table 7.7: Embedding-to-VectorDB Integration Test Case**

|  |  |
| --- | --- |
| Test Case No. | 3 |
| Test Case ID | TC\_03\_Embedding\_to\_VectorDB |
| Description | Check persistence of embeddings into ChromaDB vector store |
| Dependency | Embedding Module, Vector Storage |
| Expected Result | Embeddings are stored with metadata and can be retrieved |
| Actual Result | Embeddings were correctly indexed and retrieved using similarity search |
| Status | Pass |

**7.3.4 Query-to-Prompt Integration Test**

This test verifies the construction of an augmented prompt by combining the user’s query with relevant retrieved context. It ensures that LangChain correctly injects context for LLM inference.

**Table 7.8: Query-to-Prompt Integration Test Case**

|  |  |
| --- | --- |
| Test Case No. | 4 |
| Test Case ID | TC\_04\_Query\_to\_Prompt |
| Description | Validate flow from user query to prompt construction using LangChain |
| Dependency | Query Interface, LangChain Prompt Builder |
| Expected Result | Constructed prompt contains original query and retrieved context |
| Actual Result | Prompt generated successfully with context injection |
| Status | Pass |

**7.3.5 End-to-End Pipeline Test**

This comprehensive test case ensures that the entire diagnostic pipeline from XML telemetry ingestion to LLM-generated response functions cohesively. It validates correctness and consistency across all connected modules.

**Table 7.9: End-to-End Pipeline Integration Test Case**

|  |  |
| --- | --- |
| Test Case No. | 5 |
| Test Case ID | TC\_05\_End\_to\_End\_Data\_Flow |
| Description | Ensure full flow from XML parsing to LLM output works in sequence |
| Dependency | All core modules in pipeline |
| Expected Result | End-to-end flow produces coherent LLM response with relevant telemetry information |
| Actual Result | LLM response was accurate and based on retrieved telemetry chunks |
| Status | Pass |

**7.4 SYSTEM TESTING**

System Testing is a test that checks the entire software product. It is completely integrated. The aim of the system test is to assess the specifications of the end-to-end system. The software is usually just one component of a vast computer system. The program is ultimately interfaced with other hardware/software systems. In fact, system testing is a collection of diverse tests aimed only at the exercise of the whole computer system.

<Text>

Additionally, test runs were performed using both GPU and CPU environments to benchmark performance under different resource constraints. This helped identify potential bottlenecks in embedding generation and LLM inference latency. The system testing demonstrated that the solution was robust, reliable, and ready for deployment in production-like environments where telemetry analysis and contextual understanding are critical.

**7.4.1 End-to-End Query to Response Validation**

The validation was done so that the complete telemetry diagnostic system correctly ingests, parses, retrieves relevant chunks, and generates a response from the LLM for a given query.

**Table 7.10: Test Case for End-to-End System Testing**

|  |  |
| --- | --- |
| Test Case No. | 1 |
| Test Case ID | TC\_EndToEnd\_Query\_Response |
| Description | Validate the complete flow from raw telemetry input to LLM response output |
| Dependency | All modules integrated |
| Expected Result | LLM generates a response relevant to the query based on retrieved telemetry chunks |
| Actual Result | Response was accurate and contextually appropriate |
| Status | Pass |

**7.5 SUMMARY**

Text



**EXPERIMENTAL RESULTS AND ANALYSIS**

Experiment analysis is done with the goal of establishing that the objectives of the project were met as per expectations and the overall goal is reached. To analyse the effectiveness of the output, the influence of each input or process must be evaluated both as an individual component and a study must also be done to evaluate the performance of the system as a whole. The results obtained from each analysis must be compared with the theoretical counterparts to verify the correctness of the system. Analysing the output is verifying whether the evaluation metrics are met. This chapter discusses the performance characteristics of the system.

**8.1 EVALUATION METRICS**

Text

* **Top-K Retrieval Accuracy:** Top-K Retrieval Accuracy measures the proportion of user queries where at least one of the top K retrieved chunks contains the correct or relevant diagnostic information. This metric is crucial for evaluating the performance of the embedding model and vector store in surfacing the most contextually appropriate segments of telemetry data
  + Relevant chunk: A chunk that contains diagnostic information matching or strongly related to the expected root cause, as defined by human annotators.
  + Top-K: The number of highest-scoring chunks retrieved from the vector store
  + The system is said to have "retrieved correctly" if at least one of the Top-K chunks matches the ground truth.
* **F1 Score (Answer Quality)**: The F1 Score is used to assess the quality of the generated response by comparing it to human-annotated ground truth answers. It is calculated as the harmonic mean of precision and recall based on token overlap between the generated and reference answers. This metric captures both the completeness and correctness of the LLM output, reflecting how well the system can explain faults or anomalies using retrieved data.

where,

* **Response Latency:** Latency is measured as the total time taken from the moment a query is received to when the final answer is generated and presented. This metric is essential for evaluating the system’s real-time applicability.
* **System Robustness:** In addition to quantitative metrics, system robustness was observed through qualitative testing.

**8.2 CHUNK SIZE EVALUATION**

Text

**Table 8.1: Effect of Chunk Size on Retrieval Performance**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chunk Size (tokens)** | **Top-5 Accuracy (%)** | **F1 Score** | **Latency (s)** |
| 500 | 72.4 | 0.71 | 2.3 |
| 1000 | 81.2 | 0.76 | 2.5 |
| 2000 | 86.7 | 0.81 | 3.1 |
| 5000 | 79.3 | 0.75 | 4.8 |
| 10000 | 76.4 | 0.71 | 6.2 |

The Table 8.1 captures the observations of the retrieval performance with varying chunk sizes.

A graph showing different colored lines

AI-generated content may be incorrect.

**Figure 8.1: Chunk Size Evaluation**

These results highlight the critical role of chunk granularity in tuning RAG-based systems for diagnostic tasks. Future work may incorporate dynamic chunking or overlap strategies to further improve retrieval fidelity without sacrificing response efficiency.

**8.3 EMBEDDING MODEL EVALUATION**

In parallel to chunk sizing, the effect of embedding model selection on system performance was evaluated. Embedding quality directly influences the relevance of retrieved chunks and, consequently, the coherence of generated answers.

A graph with a red line and green line

AI-generated content may be incorrect.

**Figure 8.2: Embedding Model Comparison**

The text-embedding-3-large model delivered the highest retrieval accuracy (89.3%) and F1 score (0.85), but incurred the highest latency. Conversely, all-MiniLM-L6-v2 was the fastest but lagged behind in contextual precision. The final model chosen for deployment, nomic-embed-text, demonstrated well-balanced performance across all metrics, making it ideal for a fully local inference setup without external dependencies.

These results underscore the importance of embedding model selection not just for semantic expressiveness but also for meeting operational constraints such as latency and resource availability.

**8.4 LATENCY AND SCALABILITY EVALUATION**

Text

**8.5 LLM MODEL SELECTION AND PERFORMANCE**

Text

A graph with a red line

AI-generated content may be incorrect.

**Figure 8.3: Various LLM Performance Comparison**

**8.6 MODEL COMPARISON AND ANALYSIS**

Comparison with existing model

A hexagon with a red square

AI-generated content may be incorrect.

**Figure 8.4: Existing and Proposed Model Comparison**

Text

**Table 8.4: Comparative Evaluation of RAG Systems**

|  |  |  |
| --- | --- | --- |
| **Metric** | **ODQA RAG Baseline[7]** | **GenAI-RAG (Proposed)** |
| Retrieval Accuracy (Top-5) | 84.5% | **86.7%** |
| Answer Quality (F1 Score) | 0.72 | **0.81** |
| Avg Latency (seconds) | 5.8 | **3.1** |
| Structured Data Handling | Limited | **Native XML support** |
| Deployment Cost | High (API-based) | **Low (Local inference)** |
| Offline Capability | No | **Yes** |
| Domain Adaptability | Generic | **Domain-specialized** |

Text

**8.7 QUALITATIVE OBSERVATIONS**

Text

**8.8 SUMMARY**

Text

**CONCLUSION**

Text

**9.1 LIMITATIONS**

Text

**9.2 FUTURE SCOPE**

Text

**REFERENCES**

1. Paper
2. Paper
3. Paper
4. … 15

**APPENDICES**

**APPENDIX A: Implementation Screenshots**

These set of screenshots show the implementation of the Gen-AI powered telemetric data analysis system.

**Figure A.1: Telemetry Query Analysis Response on Resource**

**APPENDIX B: PAPER**

Authors of the research manuscript titled “title” submitted paper to “journal”

**ANNEXURE: PLAGIARISM REPORT**

<DrillBit Score Report >