HEC GENAI HACKATHON

PROJECT REPORT

ReqMind AI

Intelligent Requirement Analysis System

Group Number: 26 **Cohort:** 02

Group Members

Asifa Siraj
Warisha Danin Bilal
Iman Ayaz
Zobia Hassan
Mahzaib Iqbal
Mansoor Ahmed

Contents

Chapter 1

Introduction

This chapter introduces the background and motivation behind ReqMind AI. It discusses the challenges associated with requirement engineering, the impact of poorly defined requirements on software projects, and the rationale for proposing an AI-driven requirement analysis system.

Software project failure and delay remain critical issues in software engineering. A primary cause is poorly defined, ambiguous, or incomplete requirements. Stakeholders frequently provide raw textual descriptions lacking structure, resulting in misinterpretation, scope creep, rework, budget overruns, and dissatisfaction.

Manual requirement analysis requires significant expertise and time. To address this challenge, we propose **ReqMind AI**, a Generative AI–powered system that transforms unstructured requirement text into structured, organized, and actionable output.

Chapter 2

Problem Statement

This chapter defines the core problem addressed by ReqMind AI. It highlights the industry context, identifies the root causes of requirement-related failures, and illustrates common issues observed in traditional requirement engineering practices.

2.1 Industry Context

Key Statistics

- Requirement defects can consume between 70 or 85% of total rework cost [1].
- 50% of defects and 80% of rework effort originate from poor requirements [2].

2.2 Root Causes

Issue	**Impact**
Vague Requirements	Developer confusion and scope creep
Manual Analysis	4–8 hours per SRS document
Lack of Expertise	Small teams lack requirement engineers
Poor Communication	Product deviates from expectations

Example:

"App should be fast and user-friendly."

The statement is ambiguous and not measurable.

Chapter 3

Proposed Solution

This chapter presents the proposed solution, ReqMind AI, outlining its objectives, core capabilities, and functional scope. It explains how generative AI techniques are leveraged to transform unstructured requirement text into structured and actionable outputs.

3.1 Core Capabilities

- Functional, Non-Functional and Constraint Separation
- Requirement Quality Score (0–100) with visual breakdown
- Highlighted Problem Detection
  - Yellow – Ambiguity
  - Red – Missing Information
  - Blue – Scope Creep
- Scope Creep Detection (e.g., "may include", "future enhancement")
- Risk Analysis (Security, Scalability, Performance, Privacy)
- Stakeholder-Specific Questions (Client, Developer, Tester, Project Manager)
- Project Type Auto-Detection (E-commerce, LMS, FinTech, etc.)
- Complexity Estimation (Small / Medium / Large)
- Version Comparison (Old vs New Requirement Analysis)
- Structured JSON Output
- PDF Report Export

3.2 Illustrative Example

Input:

Build a mobile app that works fast, secure, and easy to use.

Generated Output:

- Functional: User login, dashboard display
- Non-Functional: Load time < 3 seconds, AES-256 encryption
- Ambiguity: “fast” undefined
- Missing: Target devices, user roles
- Clarification: What defines usability priority?

Chapter 4

System Architecture

This chapter describes the architectural design of ReqMind AI, including system structure, component interaction, and workflow representation using standard UML diagrams.

4.1 Overall System Architecture Diagram

The high-level architecture illustrates the interaction between the user interface, backend processing module, AI model service, and PDF generation component.

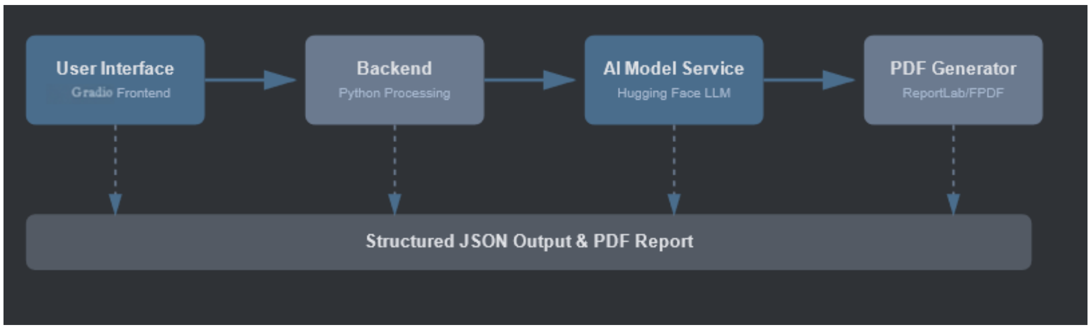
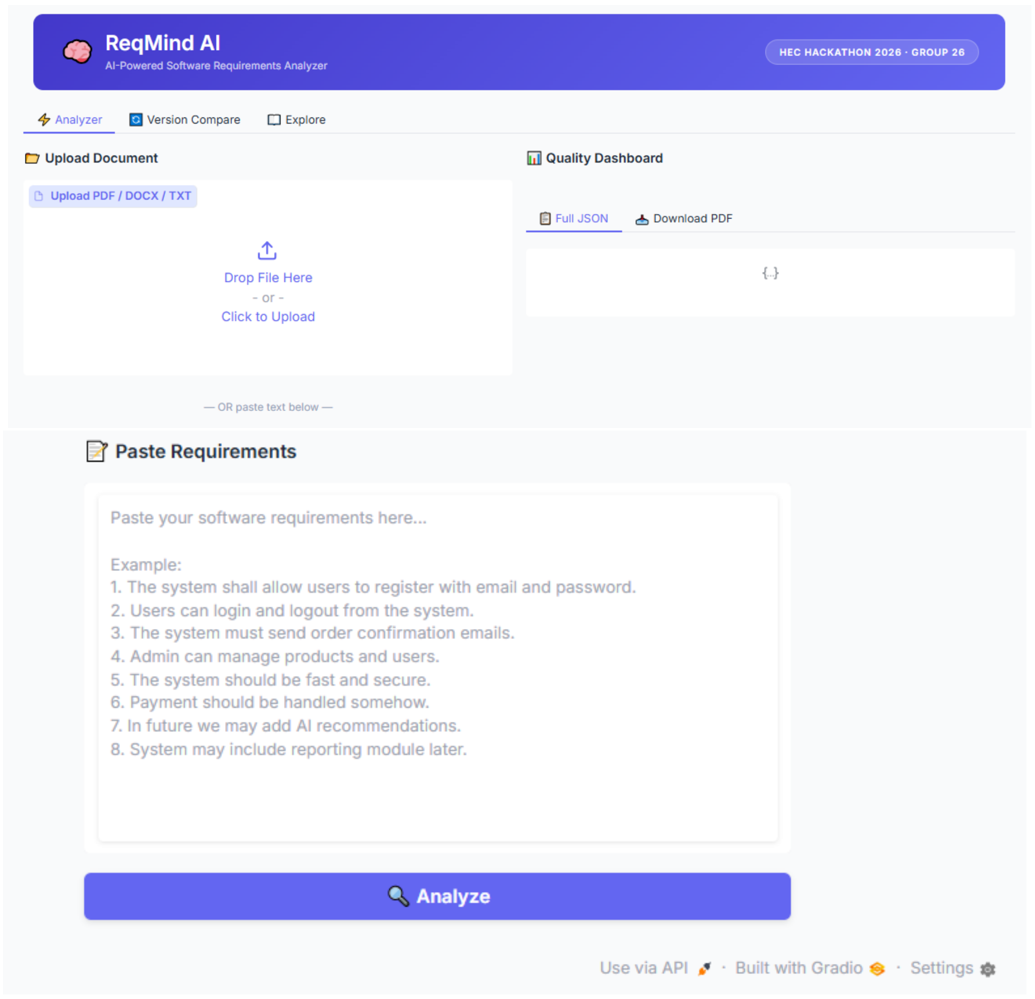


Figure 4.1: High-Level Architecture of ReqMind AI

4.2 Component Diagram

The component diagram represents major system modules and their dependencies.

- User Interface (Gradio Frontend)
- Prompt Engineering Module
- Hugging Face API Integration
- JSON Processing Module
- PDF Report Generator



Paste your software requirements here...

Example:
1. The system shall allow users to register with email and password.
2. Users can login and logout from the system.
3. The system must send order confirmation emails.
4. Admin can manage products and users.
5. The system should be fast and secure.
6. Payment should be handled somehow.
7. In future we may add AI recommendations.
8. System may include reporting module later.

Figure 4.2: Component Diagram of ReqMind AI

4.3 Sequence Diagram

The sequence diagram illustrates the step-by-step interaction between the user, frontend, backend, and AI model during requirement analysis.

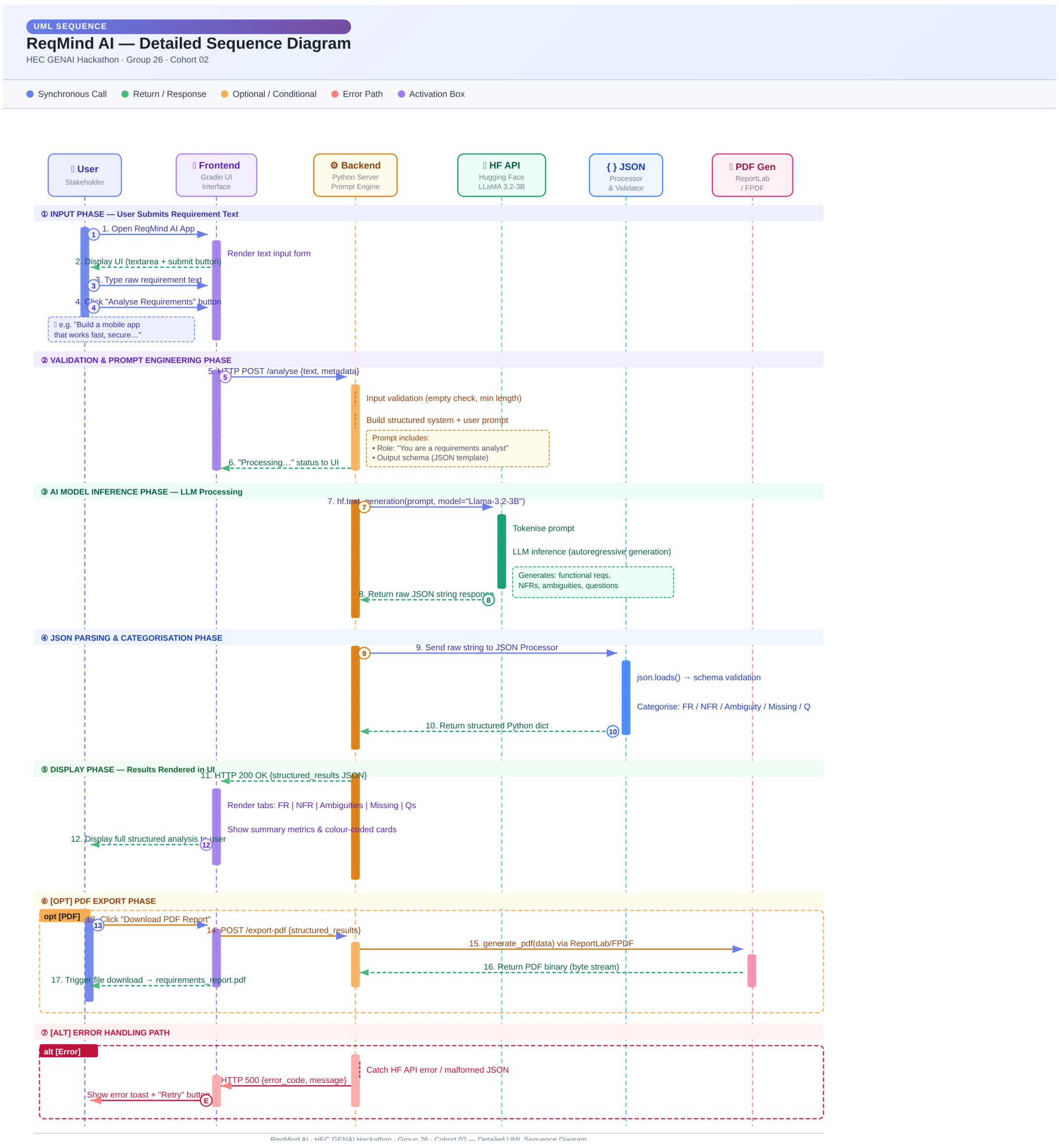

UML SEQUENCE
ReqMind AI — Detailed Sequence Diagram
HEC GENAI Hackathon · Group 26 · Cohort 02
Synchronous Call
Return / Response
Optional / Conditional
Error Path
Activation Box
User
Stakeholder
Frontend
Gradio UI Interface
Backend
Python Server Prompt Engine
HF API
Hugging Face LLaMA 3.2-3B
{ } JSON
Processor & Validator
PDF Gen
ReportLab / FPDF
① INPUT PHASE — User Submits Requirement Text
1. Open ReqMind AI App
Render text input form
2. Display UI (textarea + submit button)
3. Type raw requirement text
4. Click "Analyse Requirements" button
e.g. "Build a mobile app that works fast, secure…"
② VALIDATION & PROMPT ENGINEERING PHASE
5. HTTP POST /analyse {text, metadata}
Input validation (empty check, min length)
Build structured system + user prompt
Prompt includes:
• Role: "You are a requirements analyst"
• Output schema (JSON template)
6. "Processing…" status to UI
③ AI MODEL INFERENCE PHASE — LLM Processing
7. hf.text_generation(prompt, model="Llama-3.2-3B")
Tokenise prompt
LLM inference (autoregressive generation)
Generates: functional reqs, NFRs, ambiguities, questions
8. Return raw JSON string response
④ JSON PARSING & CATEGORISATION PHASE
9. Send raw string to JSON Processor
json.loads() → schema validation
Categorise: FR / NFR / Ambiguity / Missing / Q
10. Return structured Python dict
⑤ DISPLAY PHASE — Results Rendered in UI
11. HTTP 200 OK {structured_results JSON}
Render tabs: FR | NFR | Ambiguities | Missing | Qs
Show summary metrics & colour-coded cards
12. Display full structured analysis to user
⑥ [OPT] PDF EXPORT PHASE
opt [PDF]
13. Click "Download PDF Report"
14. POST /export-pdf {structured_results}
15. generate_pdf(data) via ReportLab/FPDF
16. Return PDF binary (byte stream)
17. Trigger file download → requirements_report.pdf
⑦ [ALT] ERROR HANDLING PATH
alt [Error]
Catch HF API error / malformed JSON
HTTP 500 {error_code, message}
Show error toast + "Retry" button
ReqMind AI · HEC GENAI Hackathon · Group 26 · Cohort 02 — Detailed UML Sequence Diagram

Figure 4.3: Sequence Diagram for Requirement Processing

Sequence Flow Description:

1. User submits raw requirement text.
2. Frontend sends request to backend.
3. Backend formats structured prompt.
4. Request sent to Hugging Face model.
5. Model returns structured JSON response.
6. Backend processes and categorizes output.
7. Results displayed in UI.

8. Optional: PDF report generated and downloaded.

4.4 Activity Diagram

The activity diagram models the workflow of requirement processing from input validation to structured output generation.

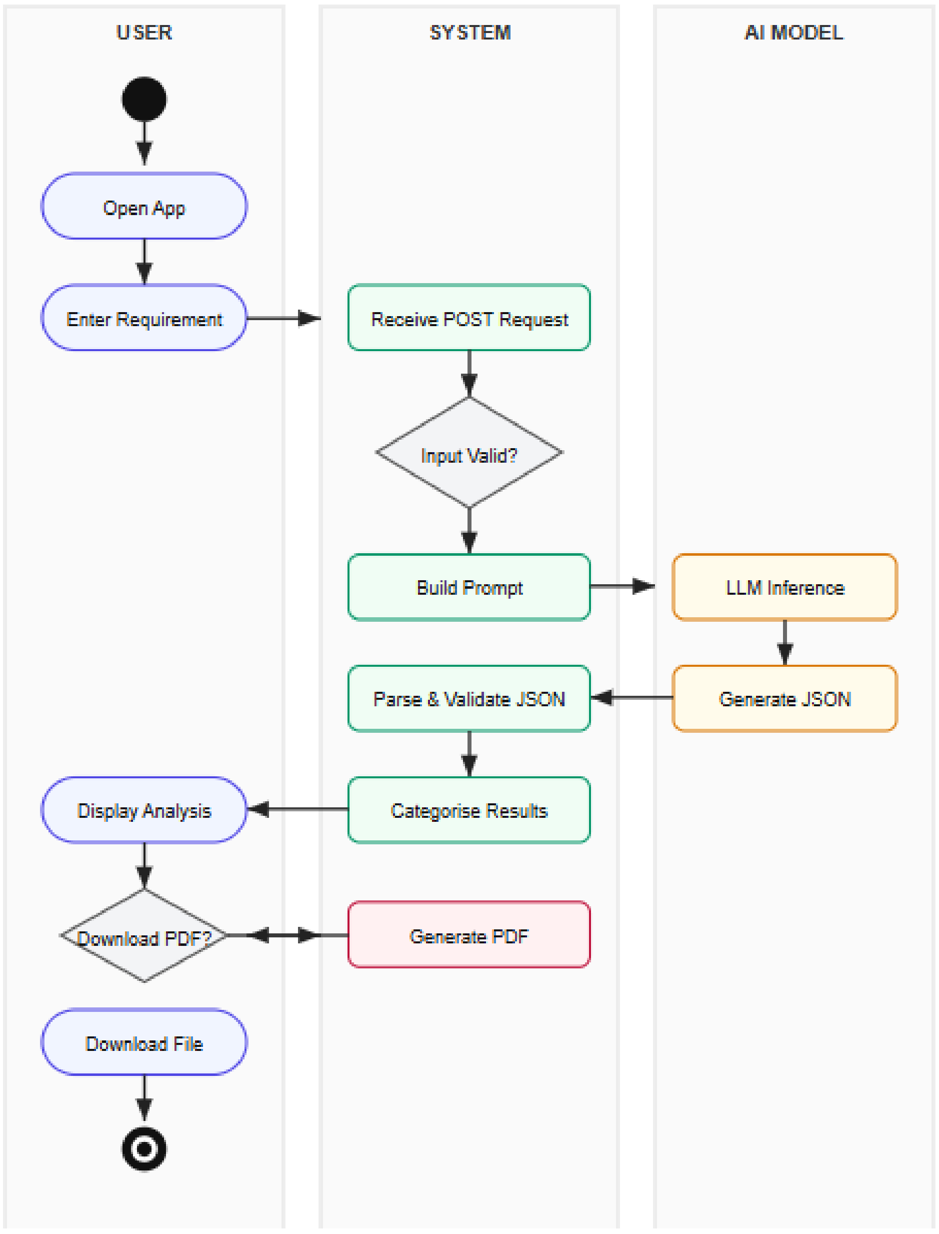


Figure 4.4: Activity Diagram of ReqMind AI Workflow

4.5 Technology Stack

- **Frontend:** Gradio

- **Backend:** Python
- **AI Model Hosting:** Hugging Face
- **Model Type:** Large Language Model (LLM)
- **PDF Generation:** ReportLab

Chapter 5

Technical Implementation

This chapter details the technical implementation of ReqMind AI, including the prompt design strategy, structured output schema, and integration with external AI model hosting services. It also explains the backend logic and processing mechanisms used to generate structured results.

5.1 AI Prompt Output Schema

```
{
  "functional": [{"id":1, "actor":"User", "action":"login"}],
  "non_functional": [{"category":"performance", "metric":"load_time<3s"}],
  "ambiguities": ["'fast' needs definition"],
  "missing": ["payment integration"],
  "questions": ["What user roles exist?"]
}
```

5.2 Hugging Face Integration

```
import huggingface_hub
hf = huggingface_hub.InferenceClient(token="hf_xxx")
response = hf.text_generation(
    prompt,
    model="meta-llama/Llama-3.2-3B"
)
```

5.3 Advanced Analysis Engine

The upgraded version of ReqMind AI includes an intelligent analysis layer that evaluates requirement quality, identifies risks, and estimates project complexity.

5.3.1 Requirement Quality Scoring

The system generates a quality score between 0–100 based on:

- Ambiguity density
- Missing constraints
- Scope creep indicators
- Requirement completeness

5.3.2 Risk Analysis Module

Risk categories automatically detected:

- Security Risks
- Scalability Risks
- Performance Risks
- Privacy Risks

5.3.3 Scope Creep Detection

The system detects phrases such as: *"may include", "future version", "optional feature"* to identify uncontrolled scope expansion.

5.3.4 Version Comparison Engine

A comparison module highlights differences between old and updated requirements, identifying:

- Newly added features
- Modified requirements
- Removed requirements

5.3.5 Project Intelligence Layer

- Automatic Project Type Detection
- Complexity Estimation (Small / Medium / Large)

Chapter 6

Testing Framework

This chapter presents the comprehensive testing strategy implemented to validate the performance, reliability, and robustness of ReqMind AI. The testing phase ensures that the system accurately converts raw requirement text into structured outputs while handling errors, edge cases, and unexpected inputs effectively.

The testing process includes:

- Functional Testing
- Edge Case Testing
- API Validation
- JSON Structure Validation
- Export Functionality Testing
- Performance Testing

6.1 Testing Scenarios and Validation Results

Scenario	Input	Expected Output	Status
Empty Input	””	Error message displayed	Completed
Simple Requirement	”User login system”	Functional requirement extracted in structured JSON	Completed
Complex SRS	5,000-word document	Full structured output (FR + NFR)	Completed
Multilingual Input	Urdu requirement text	Structured output generated	Completed
Ambiguous Requirement	”System should be fast”	Ambiguity highlighted + clarification question generated	Completed
Missing Details	”Build an app for students”	Clarification questions generated	Completed
API Failure Simulation	Invalid API token	Proper API error message displayed	Completed
Invalid JSON Format	Corrupted AI response	JSON error handled gracefully	Completed
Special Characters	Emojis & symbols	Clean parsing without crash	Completed
PDF Export	Valid structured output	PDF generated and downloadable	Completed
File Upload	Requirement file input	Functional requirements extracted in structured JSON	Completed

Table 6.1: Comprehensive Testing Scenarios and Validation Results

6.2 Error Handling Strategy

To ensure system stability and reliability, multiple error-handling mechanisms were implemented.

6.2.1 Empty Input Validation

If the user submits without entering text, the system displays:

"Please paste requirements OR upload a file."

6.2.2 API Error Handling

In case of network failure, invalid token, or rate limit issues:

- Exceptions handled using `try-except`
- User-friendly message displayed: *"AI service temporarily unavailable. Please try again."*

6.2.3 JSON Structure Validation

The AI output is validated before further processing.

- If JSON format is incorrect
- System displays: *"Unexpected AI response format. Please retry."*
- Errors caught using structured exception handling

6.2.4 Output Cleaning and Normalization

To ensure consistent formatting:

- Extra spaces removed
- Missing keys replaced with default values
- Structured categories enforced:
  - Functional Requirements
  - Non-Functional Requirements
  - Ambiguities
  - Clarification Questions

6.2.5 Performance Testing

- Large document tested (5,000+ words)
- Response time monitored
- System remained stable without crashes

Chapter 7

Expected Outcomes

This chapter discusses the anticipated benefits and impact of implementing ReqMind AI. It highlights improvements in requirement clarity, analysis efficiency, project planning accuracy, and stakeholder communication.

- Reduced requirement analysis time
- Improved documentation clarity
- Reduced scope creep
- Improved planning accuracy
- Enhanced communication
- Increased requirement precision

Chapter 8

Future Enhancements

This chapter explores potential future enhancements and system extensions aimed at improving overall system intelligence beyond the current implementation.

ReqMind AI establishes a strong foundation for intelligent requirement analysis, the following extensions are identified as high-impact and practically achievable improvements.

- **Integration with Project Management Tools**
  Integration with platforms such as Jira and Trello to automatically convert extracted requirements into user stories, sprint tasks, and backlog items, enabling seamless Agile workflow alignment.
- **Multi-Language Requirement Support**
  Semantic-level analysis of requirements in multiple languages (e.g., Urdu, Arabic, French) to support global stakeholders and multilingual SRS documents.
- **Requirement Traceability Matrix (RTM) Generation**
  Automated generation of RTM linking requirements to business objectives, system modules, and test cases to improve traceability and compliance.
- **Version Comparison and Change Impact Analysis**
  Detection of requirement changes across document versions with structured impact analysis on scope, cost, and timeline.
- **Automated User Story Generation**
  Conversion of structured requirements into Agile user story format ("As a ..., I want ..., so that ...") to support Scrum-based planning.
- **Risk Detection and Requirement Quality Scoring**
  Implementation of a scoring mechanism based on ambiguity density and missing constraints to assist early risk identification.
- **Cloud-Based Deployment and API Access**
  Deployment as a scalable SaaS platform with secure REST APIs for enterprise integration.

Chapter 9

Conclusion

This chapter summarizes the overall contributions of ReqMind AI, reiterating its significance in improving requirement engineering practices and discussing its relevance for academic, hackathon, and industry applications.

ReqMind AI addresses a fundamental challenge in software engineering: unclear and incomplete requirements. By leveraging Generative AI, the system automates structuring, ambiguity detection, and clarification generation.

The modular architecture and scalable design make it suitable for academic, hackathon, and industry-level applications.

References

[1] Jama Software. *The Business Value of Better Requirements Management*. Available online: `https://www.jamasoftware.com/blog/the-business-value-of-better-requirements-management/`. Accessed: January 2026.

[2] Devopedia. *Bad Requirements*. Available online: `https://devopedia.org/bad-requirements`. Accessed: January 2026.