ÇANKAYA UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CENG 408

Innovative System Design and Development II Project Report

Team ID: 202419

BelsisMIS-Intelligent-Customer-Support-Assistant-BICSA

Ali Emrecan Selvili - 202111031

Ege Beçin - 202011067

Mehmet Efe Kaya - 202111024

Yusuf Tuna Üner - 202011077

Ahmet Selçuk Özdil - 202011040

Advisor: Prof. Dr. Hayri Sever

Table of Contents

Introduction

This project report presents a comprehensive overview of the development process for the AI-Powered Level-0 Support Chatbot integrated into the BELSİS.NET municipal ERP platform. The report is structured into three main technical components—Software Requirements Specification (SRS), Software Design Document (SDD), and Test Plan—each capturing critical phases of the system's lifecycle from conception to validation.

The SRS outlines the core objectives, functionalities, and constraints of the chatbot system. It defines the problem the chatbot aims to solve, establishes system goals, and specifies both functional and non-functional requirements. The section sets the foundation for understanding how the chatbot will integrate with BELSİS.NET to automate repetitive user support tasks and improve response efficiency.

The SDD transitions from the requirements phase to the architectural and implementation level. It illustrates the internal structure of the chatbot system, detailing the design choices, software components, user interface design, and interactions between modules. This document provides a blueprint that guides developers in constructing a system aligned with the specified requirements.

The Test Plan describes the strategy for validating the chatbot's functionality, accuracy, and performance. It defines what features will be tested, how they will be tested, and what criteria will be used to determine success. Through detailed test cases and expected outcomes, this section ensures the system meets its objectives and provides reliable support to end-users.

Together, these three documents encapsulate the entire project scope—from the identification of user needs and technical design, to implementation assurance through rigorous testing—providing a full-cycle view of the chatbot development process.

Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed Software Requirements Specification (SRS) for the development of an AI-powered chatbot for BELSİS.NET. The chatbot aims to modernize the support system for municipalities by providing intelligent and responsive Level 0 support. This includes automating responses to common inquiries, ensuring 24/7 availability, and integrating seamlessly with the existing BELSİS.NET infrastructure. The document outlines the functional and non-functional requirements, user characteristics, and the scope of the chatbot project to ensure alignment with the company's goals.

The AI-powered chatbot project is an essential step toward digital transformation for municipal systems. By incorporating cutting-edge AI technologies, the chatbot ensures streamlined communication, enhanced user satisfaction, and a significant reduction in operational costs. This initiative also aligns with global trends in e-governance, promoting accessibility and efficiency.

1.2 Scope

The AI chatbot is designed to support all modules of BELSİS.NET, a comprehensive ERP software system used by over 150 of municipalities in Turkey. The chatbot will leverage existing end-user documentation to provide accurate and efficient responses, aiming to address 70% of user inquiries that are currently handled through phone calls. It will integrate with the BELSİS.NET platform, transitioning from outdated ASP technologies to a modern open-source AI foundation, with flexibility for database selection (e.g., PostgreSQL). The chatbot's features include natural language processing, automated text-to-specification transformation, and support for both web and mobile platforms.

The chatbot addresses inefficiencies in user support, such as slow response times and inconsistent answers, providing municipalities with a scalable, efficient, and user-friendly solution. The project has no significant business or technical constraints in terms of budget, timeline, or technology stack, offering a flexible framework for development.

1.3 Glossary

- **BELSİS.NET**: An ERP system for municipal management.
- **Level 0 Support**: Basic level of support, handling common and repetitive inquiries [1].
- **ERP (Enterprise Resource Planning)**: A software suite for managing business processes [2].

- **AI (Artificial Intelligence)**: Machine learning and natural language processing technologies.
- **PostgreSQL**: An advanced, open-source relational database [3].
- **Jira**: Jira is a tool for managing projects and tracking tasks efficiently.

1.4 Overview of the Document

This document is structured to describe the functional and non-functional requirements of the chatbot. Section 2 provides an overview of the system, including its perspective, user characteristics, and development methodology. Section 3 details the requirements specification, covering functional requirements, external interface requirements, and performance metrics. Additional sections address technical constraints, risk management, and documentation requirements.

2. Overall Description

The chatbot acts as a bridge between municipal employees and the complex functionalities of the BELSİS.NET system. The Level 0 AI-powered chatbot is designed to modernize user support for municipalities by integrating seamlessly with BELSİS.NET, an ERP system widely used by municipalities in Turkey. This chatbot focuses on handling repetitive and basic user inquiries efficiently, such as information retrieval, account details, and general troubleshooting. By automating these tasks, it significantly reduces the burden on support teams, ensuring faster response times and consistent answers.

Integrated across key BELSİS.NET modules, it ensures 24/7 availability via web and mobile platforms, empowering users to resolve their queries independently.

The chatbot prioritizes usability, security, and flexibility. This initiative not only streamlines support operations but also aligns with the evolving needs of municipal services, delivering an efficient, user-friendly, and future-proof solution.

2.1 Product Perspective

The AI-powered chatbot for BELSİS.NET will serve as a key component of the municipal ERP system, enhancing user interaction and support services. It replaces outdated ASP-based support mechanisms with a modern AI platform capable of handling natural language queries. The chatbot will be embedded within the BELSİS.NET web and mobile applications, ensuring seamless access for end-users.

Key features of the product include:

- Integration with existing BELSİS.NET modules (e.g., accounting, personnel management).
- Natural language processing to interpret user inquiries.

- Automated conversion of text specifications into actionable support responses.
- Transforms customer support telephone conversations into text, enabling the AI model to analyze and generate accurate, automated responses for improved service efficiency.

The development prioritizes scalability, allowing the chatbot to serve the diverse needs of municipalities with varying user bases.

2.1.1 Development Methodology

The project will utilize Agile development methodologies, focusing on iterative and incremental delivery. Jira will serve as the primary project management tool, enabling the team to manage tasks, monitor sprint progress, and maintain traceability throughout the development lifecycle [4]. This approach ensures adaptability to changing requirements and fosters a collaborative environment. Scrum will be the primary framework, ensuring:

- Short sprints with consistent feedback loops.
- Regular updates to the firm.
- Continuous integration and testing.
- Weekly meetings are held with the Product Owner, Scrum Master, and Development Team to ensure alignment, and regular progress reports are shared to maintain transparency and track milestones effectively.

Scrum roles include:

- **Product Owner**: Ensures requirements align with company's needs.
- **Scrum Master**: Facilitates the development process.
- **Development Team**: Implements and tests the chatbot.

This approach ensures adaptability to changing requirements and fosters a collaborative environment.

2.2 User Characteristics

The Level 0 user support chatbot is designed to assist municipal employees, particularly those who may not have extensive technical expertise or deep knowledge of the BELSİS.NET system. These employees often encounter simple, repetitive problems during their daily tasks, and the chatbot provides a straightforward and efficient solution for resolving such issues.

Municipal employees using the chatbot typically need assistance with tasks like retrieving basic information, troubleshooting common errors, or understanding system functionalities. By addressing these needs, the chatbot reduces the dependency on technical support teams, enhances productivity, and helps employees focus on more critical aspects of their roles.

Characteristics:

- Familiarity with the BELSİS.NET platform and technical terminology.
- Need for quick access to information for day-to-day operations.
- Occasional administrative responsibilities for maintaining chatbot performance.

Needs:

- Integration with BELSİS.NET modules for tasks like accounting, personnel management, and report generation.
- Accurate, context-aware responses tailored to internal processes.
- Tools to monitor chatbot performance and update its knowledge base as needed.

2.2.1 End Users

- **Primary Users**: Municipal employees using BELSİS.NET for various operational tasks.
- **Needs**:
    - Quick access to information and support.
    - Accurate and consistent responses to inquiries.
    - Availability across web and mobile platforms.

2.2.2 Administrators

- **Role**: Configure and maintain the chatbot.
- **Requirements**:
    - Technical knowledge of the BELSİS.NET system.
    - Ability to update chatbot’s knowledge base as needed.

2.3 Product Features

- 24/7 availability for user support.
- AI-driven natural language processing [5]
- Integration with existing BELSİS.NET modules.
- Scalability to handle varying user demands.
- Support for text-to-specification transformations using end-user documentation.
- Uses converted speech to text as prompt (request) for chatbot automatically.

3. Requirements Specification

3.1 Functional Requirements

In order to achieve success for our chatbot we need to respond at least %70 of the request made by users over phone calls. For that we need to convert speech to text using an AI and upload that converted text to our ai model. Our model will use the end user support documents created by ASP files and the knowledge base to solve the problem of user and we aim to solve %70 of the requests. We'll have functional use cases like "AI-Based Text-to-Response Conversion", "Behavior Analysis for Continuous Improvement", "User Inquiry Resolution"… The main problems users face using ai chatbots are usually not receiving the correct answers or not receiving them fast enough so with our functional requirements we are aiming to solve those problems.

3.1.1 Integration

- The chatbot shall be integrated within the BELSİS.NET web and mobile applications. It should be an integral integration.
- It shall interact with all modules of BELSİS.NET, such as accounting and personnel management, for query resolution.

3.1.2 Data Handling

- The BELSİS.NET system uses SQLServer and the chatbot shall utilize PostgreSQL as its default database, with flexibility for alternative database options based on project needs.
- It shall not store sensitive user data locally and comply with municipal data regulations.
- Our chatbot will have different Data Insertions Modules if we must gave some examples:
  Finance Module: Income, outcome, billing and payment informations.
  Human Resources Module: Salaries, employee informations, performance evaluations.

3.1.3 Reporting

- The system shall generate reports on chatbot performance, including response accuracy and resolution rates.
- It shall log popular queries and areas where human intervention is required.

3.1.4 Use Cases

3.1.4.1 User Inquiry Resolution

Actors: End-users (Municipality employees).

Description: Users interact with the chatbot to resolve queries about

BelsisMIS modules. The chatbot provides detailed step-by-step solutions.

Triggers: A user asks a question through the chatbot interface.

Preconditions: The user has access to the BelsisMIS platform.

Postconditions: The chatbot resolves the query

Flow:

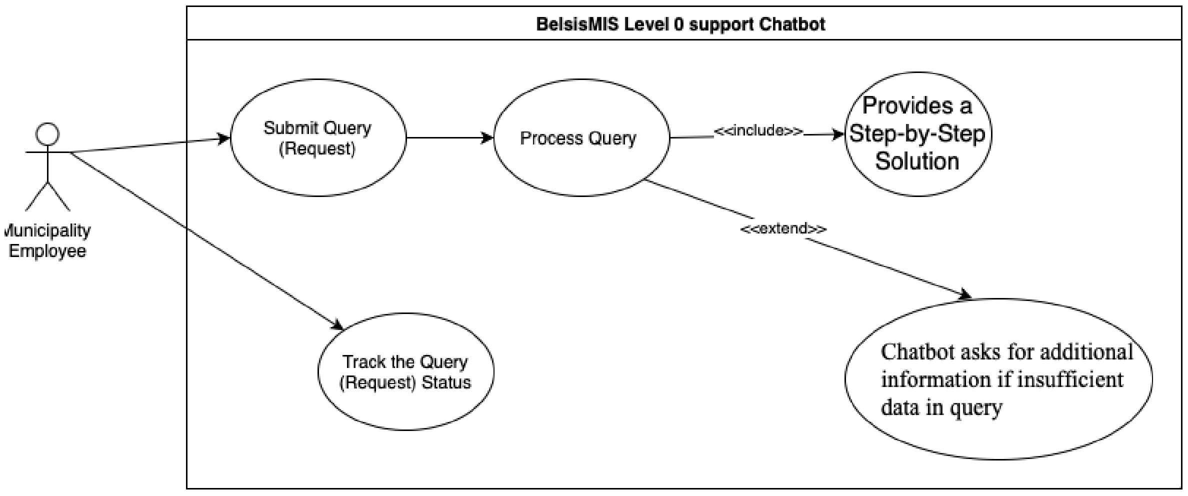
1. User Logs In: It's the precondition.
2. User Asks a Question: user enters a query regarding a BelsisMIS module.
3. Chatbot Processes the Query: parses the user's query and matches it to relevant modules or topics in its database.
4. Chatbot prepares and provides a step-by-step Solution : The chatbot generates a solution to the query with detailed steps and displays it to the user.

3.a Insufficient Data in Query

3.a.1. Chatbot asks for additional information

3.a.2. The user provides the required details.

Use Case Diagram:


BelsisMIS Level 0 support Chatbot
Municipality Employee
Submit Query (Request)
Process Query
<<include>>
Provides a Step-by-Step Solution
<<extend>>
Track the Query (Request) Status
Chatbot asks for additional information if insufficient data in query

3.1.4.2 Behavior Analysis for Continous Improvement

Actors: Administrators

Description: Administrators use the system to analyze user behavior (e.g., frequently asked questions, success rates) and refine AI models for better performance and accuracy.

Triggers: The system periodically monitors and collects interaction data.

Preconditions:

User interactions must be logged securely.

Data collection is compliant with privacy regulations.

Postconditions:

Enhanced accuracy of AI model predictions.

Improved overall user experience through better response handling.

Flow:

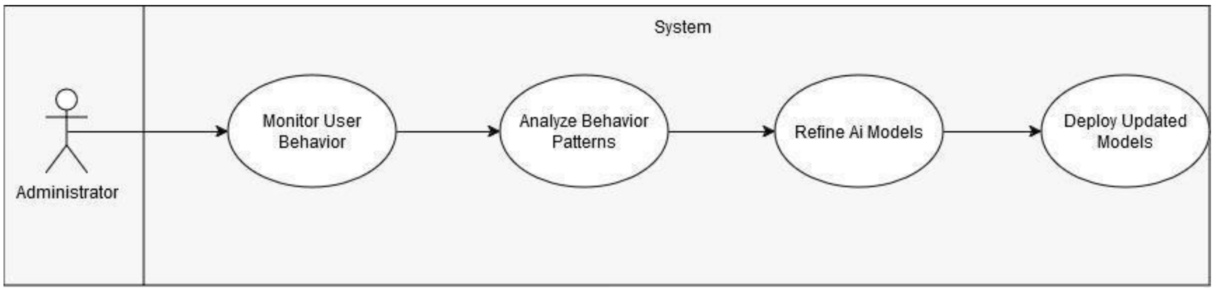
1. Data Collection Initiated: Logs user interactions automatically during a set period.
2. Behavior Analysis Conducted: Identifies behavior patterns and evaluates success rates.
3. Model Refinement Initiated: Shares insights and retrains AI models.
4. Deployment of Improved Models: Updates the system and verifies accuracy.

Alternate Flows:

3.a Insufficient Data for Analysis: The system flags insufficient data.

3.a.1. Administrators configure additional data collection requirements.

Use Case Diagram:



3.1.4.3 AI Based Text to Response Conversion

Actors: Administrators, Users

Description: The chatbot analyzes user text inputs, processes them using AI and provides accurate responses.

Triggers: A user submits a text query.

Preconditions: The system must be trained with domain-specific knowledge.

The system is online and functioning.

Postconditions: User receives an AI-generated response.

Optional user feedback is logged for system improvement.

Flow:

1. User Submits Query
    a. Trigger**:** A user initiates interaction by submitting a text query to the chatbot interface.
2. Query Receipt and Validation
    a. The system receives the input and checks for completeness (e.g., no empty submissions).
    b. If the query is invalid or incomplete, the user is prompted to revise their query.
3. AI Processing
    a. The system sends the validated query to the AI model for processing.
    b. The AI model:
        i. Analyzes the query.
        ii. Matches it against the domain-specific knowledge base.
        iii. Generates a suitable response.
4. Response Generation
    a. The system formats the AI-generated response for clarity and relevance.
5. Response Delivery
    a. The response is sent back to the user through the chatbot interface.
6. User Feedback (Optional)
    a. The user has the option to provide feedback on the response to help improve system accuracy.

Alternate Flows:

1.a. Invalid Query Submitted

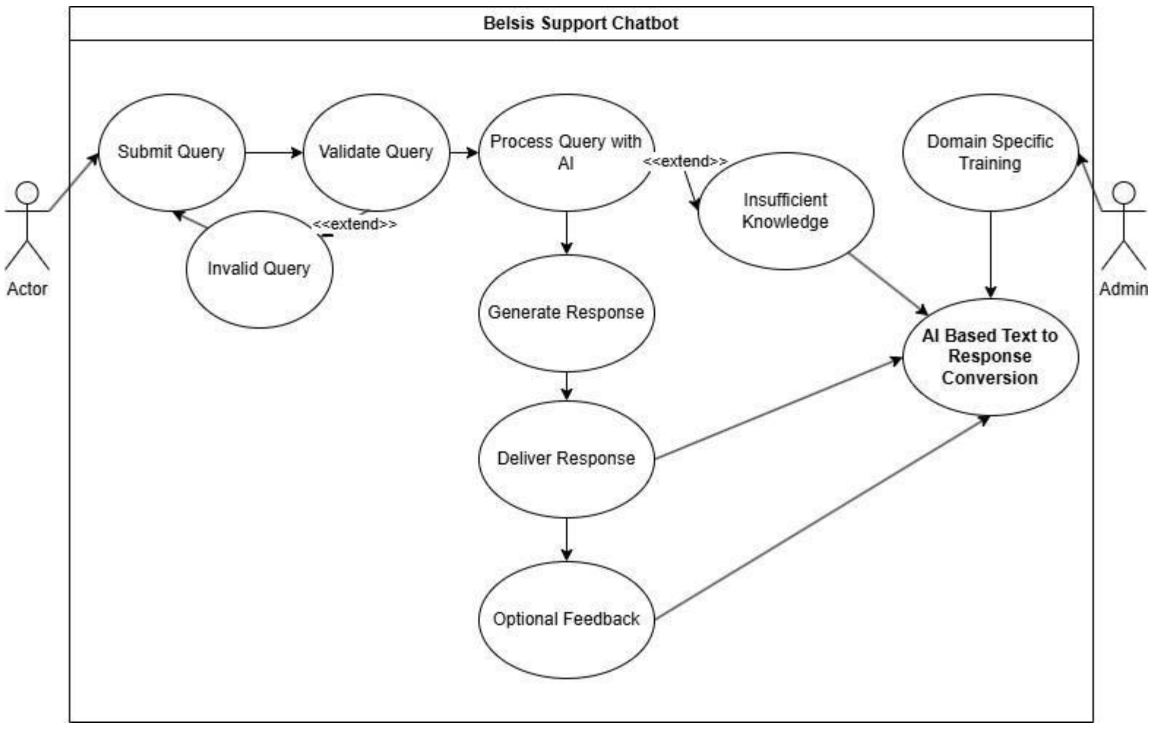
- **Trigger:** The user submits a query with missing or unsupported elements.
- **Flow:**
    - The system detects the issue and provides guidance for revising the query.
    - Returns to Step 1 of the main flow.

4.a. Knowledge Gap in AI

- **Trigger:** The AI model cannot generate a meaningful response due to a lack of domain-specific knowledge.

- **Flow:**
  - The system informs the user that it could not process the query.
  - The issue is logged, and the Administrator is notified for model retraining or knowledge base updates.

Use Case Diagram:



3.1.4.4 Handling Frequently Asked Questions (FAQ)

Actors: End-users (Municipality employees).

Description: Ensures that common user inquiries are addressed quickly and consistently by an automated chatbot system. By matching user queries to a pre-configured FAQ database, the chatbot provides instant, standardized responses, improving efficiency and enhancing the user experience.

Triggers:

• A user submits a query to the chatbot system that matches an entry in the FAQ database.

• The chatbot identifies keywords or patterns indicative of a frequently asked question.

Preconditions:

• The chatbot system must have a pre-configured database of frequently asked questions and responses.

• The chatbot system must be functional and accessible to end users.

• The user must have a query that matches a pre-configured FAQ.

Postconditions:

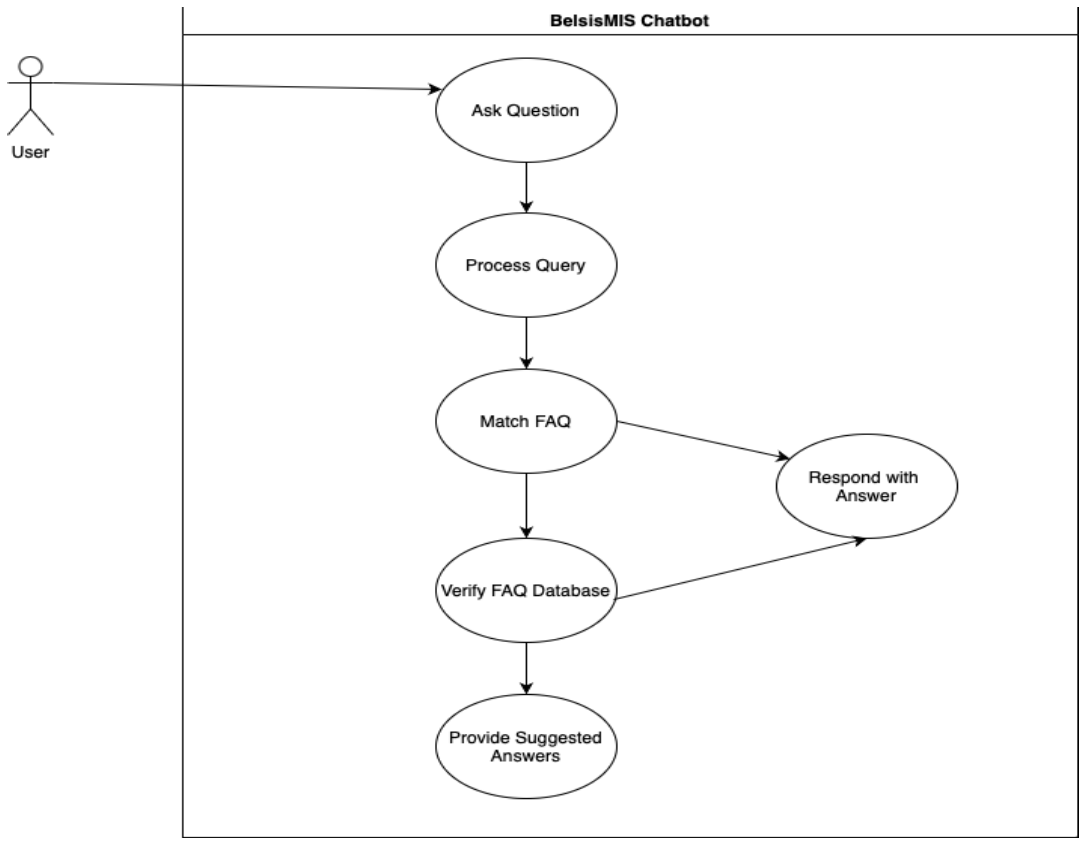
• The user receives an accurate, pre-configured response to their question.

• The chatbot logs the interaction for future improvements or analysis.

Flow:

1. **Ask Question** (User)
   - Action: User asks a question to initiate the process.
2. **Process Query** (Chatbot)
   - Action: Chatbot processes the user’s query.
3. **Match FAQ** (Chatbot)
   - Action: Chatbot checks FAQ database for a matching answer.
4. **Verify FAQ Database** (Chatbot)
   - Action: Chatbot verifies if the matched FAQ answer is correct.
5. **Respond with Answer** (Chatbot)
   - Action: Chatbot provides the answer to the user.
6. **Provide Suggested Answers** (Chatbot)
   - Action: Chatbot suggests possible answers if FAQ match is not found.

Use Case Diagram:



3.1.4.5 Providing Step by Step Guidance in BelsisMIS Modules

Actors: End-users (Municipality employees).

Description: This use case helps users navigate complex tasks in BelsisMIS modules by providing clear, step-by-step instructions via the chatbot, ensuring tasks are completed efficiently.

Preconditions:

- The chatbot system must have detailed, pre-configured step-by-step instructions for tasks in BelsisMIS modules.

Postconditions:

- The user successfully completes the task following the provided step-by-step guidance.
- The chatbot logs the interaction for future improvements.

Triggers:

- A user submits a request for assistance with a specific task in BelsisMIS (e.g., "I need help generating a report").
- The chatbot identifies the request as requiring step-by-step guidance and initiates the instructions.

Flow:

1. Help Request From User (User)

- **Action:**
  The user requests help for a complex operation within the BelsisMIS module (e.g., generating a report).

2. Analyze User Request (Chatbot)

- **Action:**
  The chatbot analyzes the user's request to determine the type and complexity of the operation.

3. Explain Operations According to Complexity and Provide a Step-by-Step Guide (Chatbot)

- **Action:**
  Based on the complexity of the operation, a step-by-step guide is prepared and tailored to the user's needs, providing all relevant details.

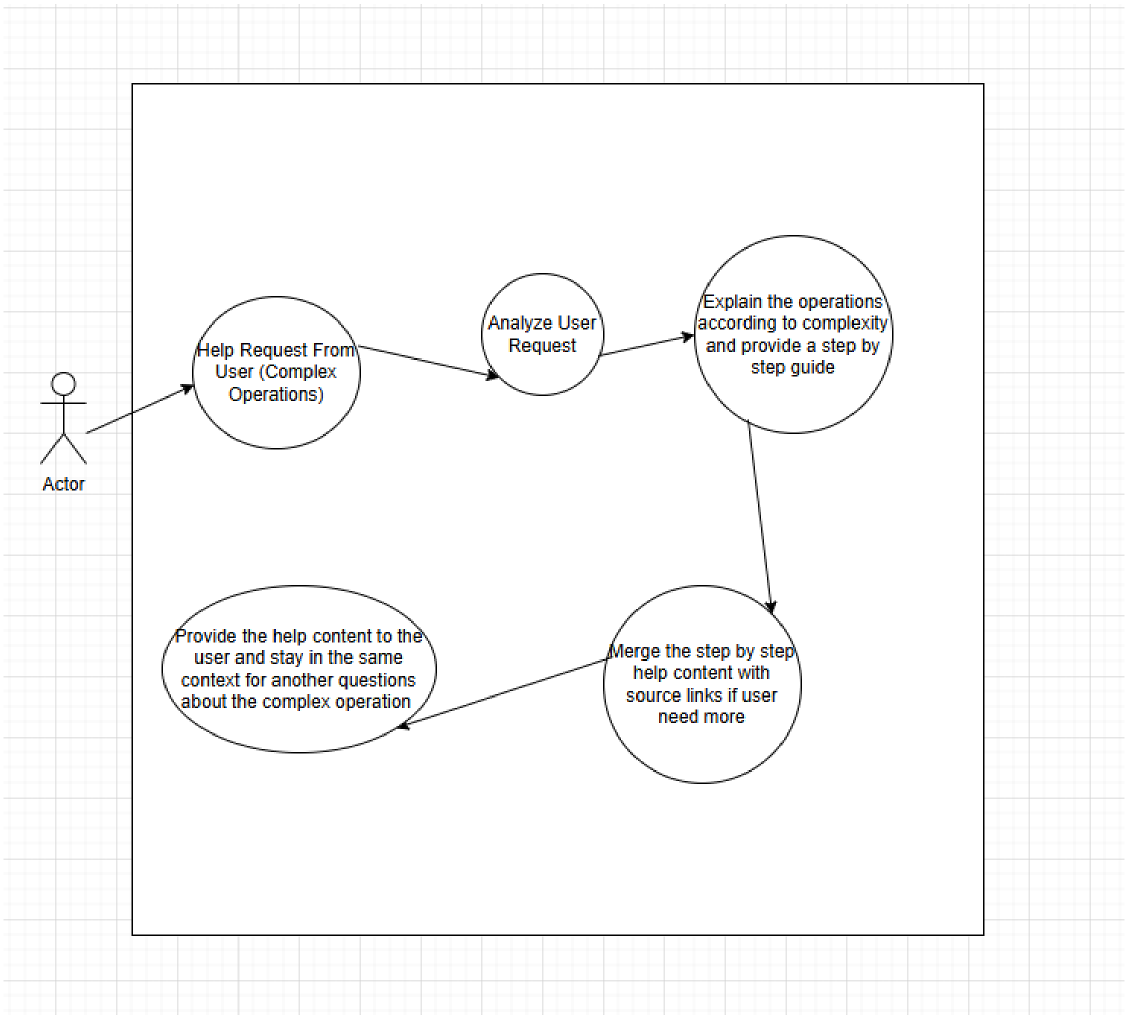
4. Merge the Step-by-Step Help Content with Source Links if User Needs More (Chatbot)

- **Action:**
  If the user needs additional information, the step-by-step guide is merged with source links, providing further documentation.

5. Provide the Help Content to the User and Stay in the Same Context for Another Questions (Chatbot)

- **Action:**
  The help content is provided to the user, and the chatbot remains in the same context to answer follow-up questions related to the complex operation.

Use Case Diagram:


Actor
Help Request From User (Complex Operations)
Analyze User Request
Explain the operations according to complexity and provide a step by step guide
Merge the step by step help content with source links if user need more
Provide the help content to the user and stay in the same context for another questions about the complex operation

3.2 Non-Functional Requirements

The system will incorporate robust load balancing mechanisms to effectively manage surges in user activity. This ensures that users experience a seamless interaction with the chatbot, regardless of the volume of concurrent requests. Additionally, the system will feature adaptive performance optimization, dynamically adjusting resources to maintain consistent and reliable query response times. These measures collectively ensure that the chatbot remains responsive, efficient, and capable of handling the demands of a diverse and growing user base, even under heavy workload conditions.

3.2.1 Performance

- The chatbot shall process user queries within 2 seconds on average.
- It shall support up to 100 concurrent users.

3.2.2 Availability

- The chatbot shall operate 24/7 without downtime, except during scheduled maintenance.

3.2.3 Security

- It shall use encryption protocols for data exchange such as TLS (Transport Layer Security), to ensure secure and reliable data exchange between the chatbot and its users.
- The chatbot shall comply with data protection regulations, such as KVKK.

3.2.4 Scalability

- The system shall scale to accommodate the needs of additional municipalities as required.

3.3 External Interface Requirements

3.3.1 User Interfaces

- The chatbot shall be accessible through the BELSİS.NET web interface and mobile application.
- It shall feature a simple, intuitive user interface optimized for municipal employees with varying technical expertise.

3.3.2 Hardware Interfaces

- The chatbot shall require minimal hardware, leveraging the existing infrastructure of BELSİS.NET.
- No specialized hardware is needed.

3.3.3 Software Interfaces

- The chatbot shall interface with the BELSİS.NET application and its associated modules.
- It shall integrate with the PostgreSQL database for efficient data handling.

3.3.4 Communication Interfaces

- The chatbot shall support HTTPS for secure communication.
- It shall ensure seamless data exchange between the web and mobile platforms.

3.4 Performance Requirements

- The system shall maintain a minimum uptime of 99.9% annually.
- Query response time shall not exceed 2 seconds under standard load conditions.

3.5 Design Constraints

- The chatbot shall adhere to the existing design principles of the BELSİS.NET interface.
- It must be compatible with current web and mobile technologies used by municipalities.

3.6 Test and Validation Requirements

- Unit testing shall be conducted to ensure individual components function correctly.
- Integration testing shall verify seamless interaction between the chatbot and BELSİS.NET modules.
- User acceptance testing shall involve feedback from municipal employees to validate usability.

4. Technical Details

The backend system shall incorporate a modular architecture, allowing seamless updates and feature expansions. Additionally, the integration of advanced AI libraries from OpenAI will provide a robust framework for natural language understanding and contextual analysis, enabling efficient and scalable AI-driven interactions.

4.1 Technology Stack

- **Frontend**: ASP.NET Core (Razor Pages) for web interfaces.
- **Backend**: The chatbot system is developed using Python with FastAPI for AI integration.
- **Hosting**: Local deployment on dedicated servers with containerized deployment using Docker

4.2 Security Features

- End-to-end encryption for all communications.
- Role-based access control to restrict sensitive actions.
- Regular security audits to identify vulnerabilities.

4.3 API's

During the integration phase of the Level 0 AI-powered user support chatbot, OpenAI's API will be integrated to ensure seamless communication between the chatbot and the existing BELSİS.NET system. The API implementation will be designed to meet the scalability and security requirements of the system, enabling the chatbot to retrieve and process relevant data in real-time using the RAG approach while maintaining strict compliance with data privacy regulations.

4.4 AI Model

For the development of the user support chatbot, the chosen AI model will leverage Natural Language Processing (NLP) capabilities provided by OpenAI to understand and respond to user queries effectively. Considering the project requirements, OpenAI's model has been selected due to its robust language understanding, ease of integration, and proven performance in enterprise applications.

The selected model will utilize Retrieval-Augmented Generation (RAG) with domain-specific data to ensure accurate and context-aware responses. This approach guarantees a scalable and efficient solution, capable of meeting the evolving needs of the municipality's digital infrastructure.

The chatbot will follow robust security protocols to protect data and ensure reliable operation:

- Encryption:
  - All data transmissions and stored information will be encrypted using industry-standard techniques like TLS and AES.

- Authentication and Authorization:
  - OAuth 2.0 or JWT will verify user identities, and RBAC will limit access based on user roles.

- Data Privacy Compliance:
  - The chatbot will adhere to data protection laws, such as KVKK, ensuring responsible data retrieval and processing without storing sensitive information.

- API Security:
  - APIs will use authentication keys, rate limits, and input validation to prevent unauthorized access. Additionally, query filtering and source validation will be implemented to ensure only trusted data is retrieved in the RAG process.

- Audit and Monitoring:
  - Logs will be maintained for audits, and real-time monitoring will detect potential security threats.

- Bot Protection:
  - Built-in security features of Gemini AI, combined with anti-bot measures like CAPTCHA, will help safeguard against malicious activities.

5. Risk Management

User adoption of the chatbot might be slower than expected, particularly among less tech-savvy employees. To prevent this, a phased rollout plan with targeted training sessions and user incentives will be implemented, ensuring gradual but steady acceptance of the new system.

5.1 Potential Risks

- **Technical**: Compatibility issues during integration with legacy systems.
- **Operational**: Insufficient training for end-users leading to underutilization.
- **Project**: Delays in development caused by unforeseen technical challenges or resource constraints.

5.2 Mitigation Strategies

- **Technical**: Conduct thorough testing and create fallback solutions to ensure smooth integration with existing systems.
- **Operational**: Provide detailed training materials and sessions for municipal employees to maximize chatbot usage.
- **Project**: Implement a robust project management framework with regular check-ins to prevent delays.

6. Documentation

Step-by-step guides and user-friendly documentation will be developed by converting ASP files into comprehensive end-user manuals. These resources will play a crucial role in training the AI model, serving as a foundational knowledge base to ensure the model provides accurate, context-aware, and reliable responses. Although these manuals are not intended for direct use by end-users, they will enable the chatbot to deliver seamless and effective support based on the information they contain.

6.1 User Documentation

- The chatbot will include user manuals to guide employees in using its features effectively.
- Tutorials will be embedded within the application for quick onboarding.

6.2 Technical Documentation

- Developers will receive API documentation for the chatbot's integration points.
- Maintenance guides will be provided for system administrators to update and configure the chatbot.

7. Deployment and Maintenance

The deployment and maintenance of the Level 0 user support chatbot will follow a structured process to ensure seamless integration, reliable performance, and adaptability.

The Technical Support Team will handle deployment, including system configuration, initial testing, and go-live procedures, ensuring a smooth transition. Post-deployment, the AI Technical Support Team will manage updates, bug fixes, and performance optimization, leveraging user feedback and operational data for continuous improvement.

Maintenance will be governed by Service Level Agreements (SLAs) outlining response times, update schedules, and performance benchmarks to ensure accountability and reliability.

To support organizations using the software, training sessions will be provided to familiarize users with the chatbot's features.

This approach ensures the chatbot remains an effective, reliable, and adaptable tool for municipal operations.

8. Conclusion

This document provides a comprehensive overview of the design, requirements, and implementation plan for a Level 0 AI-powered support chatbot for BELSİS.NET, a municipal ERP system utilized by a significant number of municipalities in Turkey. The chatbot is intended to revolutionize municipal support services by automating responses to repetitive user inquiries, reducing the workload on support teams, and providing efficient, consistent, and accessible solutions to everyday problems.

The primary focus of the chatbot is to enhance user satisfaction and productivity by offering accurate and reliable assistance. By addressing common issues such as delayed response times and inconsistent support, the system aims to streamline workflows and ensure that users can resolve their queries without requiring constant intervention from technical teams. This not only improves operational efficiency but also reduces overall costs for municipalities.

The document highlights the chatbot's adaptability to various user needs, ensuring it can cater to employees with differing levels of technical expertise. The project also places emphasis on proper risk management, phased rollouts, and training sessions to encourage smooth adoption by end-users. Additionally, detailed documentation and support resources are outlined to ensure the system's long-term usability and effectiveness.

In summary, the chatbot project represents a forward-thinking approach to modernizing municipal operations and aligning with global trends in e-governance. It lays the groundwork for a more efficient, user-friendly, and scalable support system, contributing significantly to the digital transformation of public services and setting the stage for future technological advancements in this sector.

References

[1]: "Explaining IT Support Levels: How L0, L1, L2, L3, L4 Support Tier Work" [Online]. Available: https://www.certguidance.com/explaining-support-levels-itil-itsm/. [Accessed 5 December 2024].

[2]: "Enterprise Resource Planning (ERP) Explained" [Online]. Available: https://www.bmc.com/blogs/erp-enterprise-resource-planning/. [Accessed 5 December 2024].

[3]: "What is PostgreSQL?" [Online]. Available: https://www.postgresql.org/about/. [Accessed 5 December 2024].

[4]: “Agile tools for software teams” [Online]. Available:
https://www.atlassian.com/software/jira/agile/. [Accessed 5 December 2024].

[5]: “What is NLP (natural language processing)?” [Online]. Available:
https://www.ibm.com/topics/natural-language-processing/. [Accessed 5 December 2024].

1. Introduction

1.1 Purpose

The purpose of this document is to explain the design of the Level-0 Support Chatbot for BELSİS.NET. This chatbot system is intended to assist municipal employees by automating responses to common inquiries and providing an efficient and user-friendly platform for accessing information. It aims to reduce the workload of support teams by handling repetitive tasks and improve response times for users. Additionally, the chatbot is designed to integrate seamlessly with existing systems, ensuring smooth and reliable operation.

1.2 Scope Of The Project

The scope of this project is to develop a Level-0 Support Chatbot for BELSİS.NET, an ERP system used by municipalities. The chatbot is designed to assist municipal employees by answering common questions and providing step-by-step guidance for tasks related to the BELSİS.NET modules. It will use natural language processing (NLP) to understand and respond to user queries effectively. The goal is to automate up to 70% of repetitive inquiries, improve user satisfaction, and enhance the efficiency of municipal operations.

1.3 Glossary

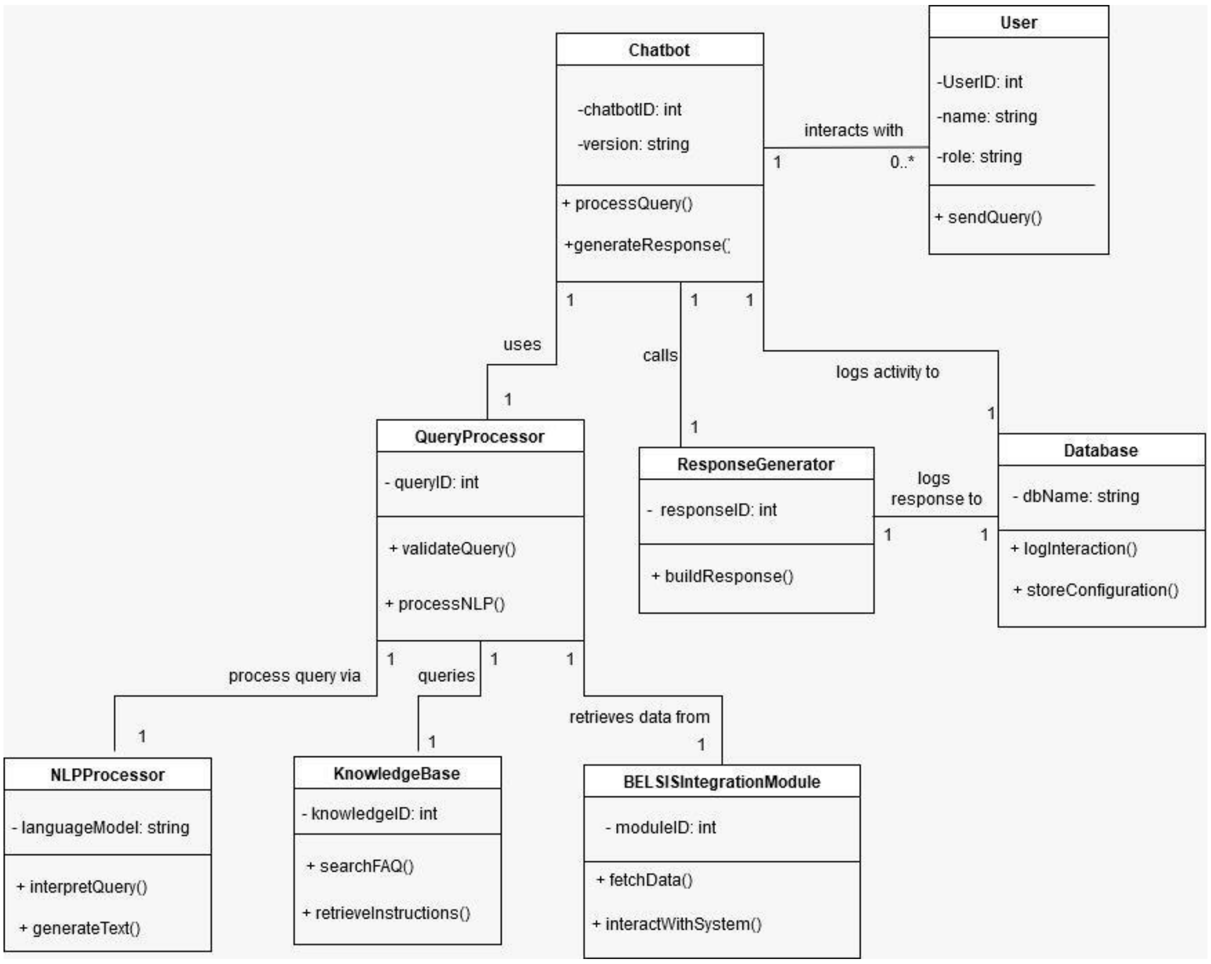
- **Chatbot:** A software application that interacts with users in natural language to provide information or perform tasks automatically.
- **Level-0 Support:** Basic support that handles simple and repetitive user inquiries without human intervention.
- **Natural Language Processing (NLP):** A technology that enables the chatbot to understand, process, and generate human language in a meaningful way.
- **Artificial Intelligence (AI):** The simulation of human intelligence processes by machines, particularly computer systems, to solve complex problems.
- **BELSİS.NET**: An ERP system designed to manage various operations within municipalities, including finance, human resources, and administrative tasks.
- **API (Application Programming Interface):** A set of protocols and tools that allows different software applications to communicate and interact seamlessly.
- **PostgreSQL:** An advanced, open-source database management system used for storing and managing structured data.

1.4 Overview of the Document

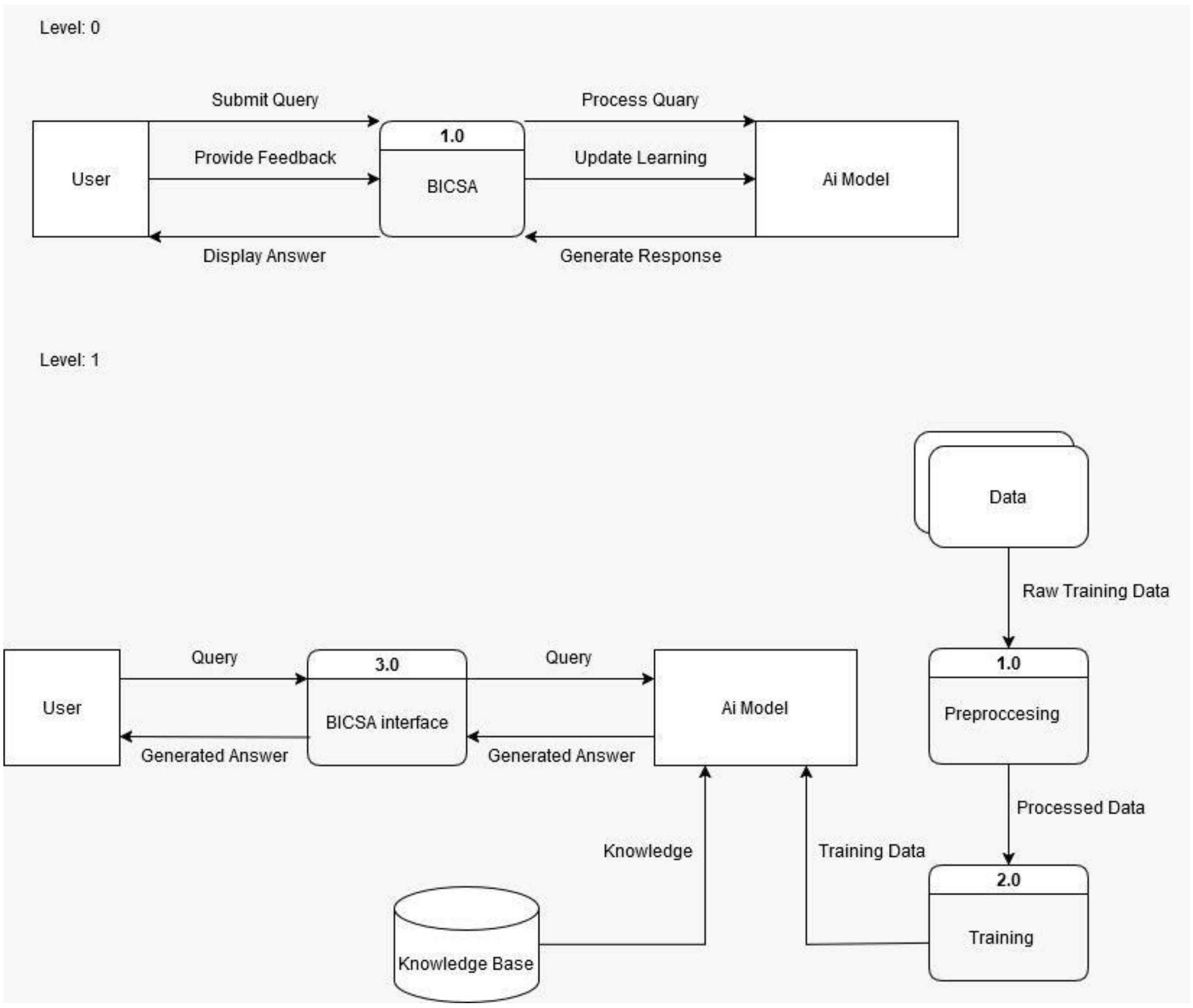
This document provides a detailed description of the design and architecture of the Level-0 Support Chatbot for BELSİS.NET. It begins with an introduction that explains the project's purpose, scope, and objectives. Following this, the document delves into the technical aspects, including various design diagrams—such as class, data flow, activity, and sequence diagrams—which illustrate the system's structure and processes. It also features a section on user interface design, showcasing the layout and functionalities of the sample web and mobile applications.

2. System Design

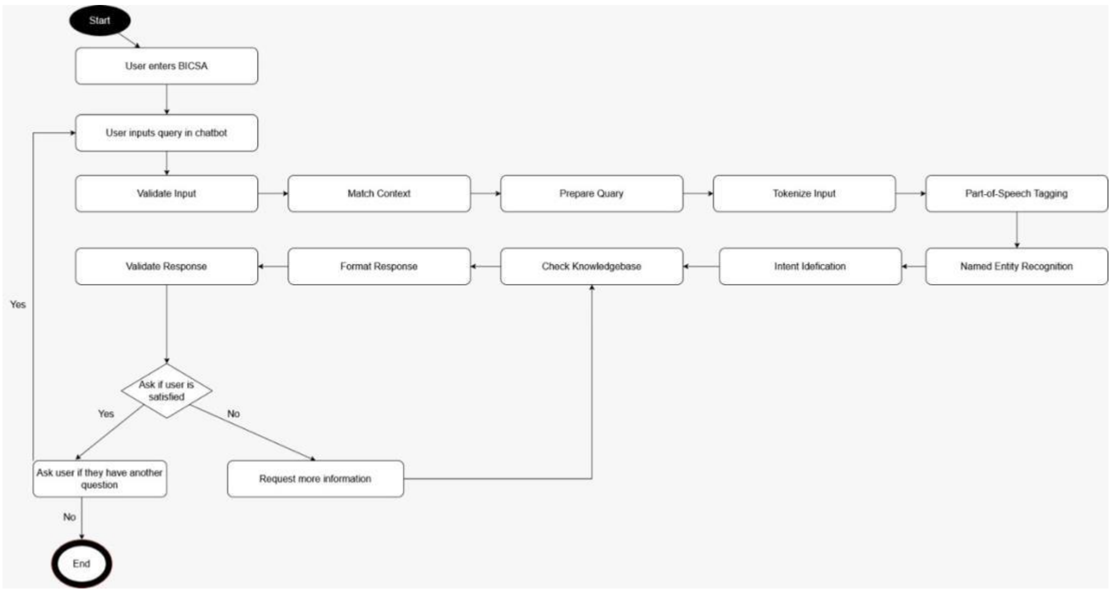
2.1 Class Diagram



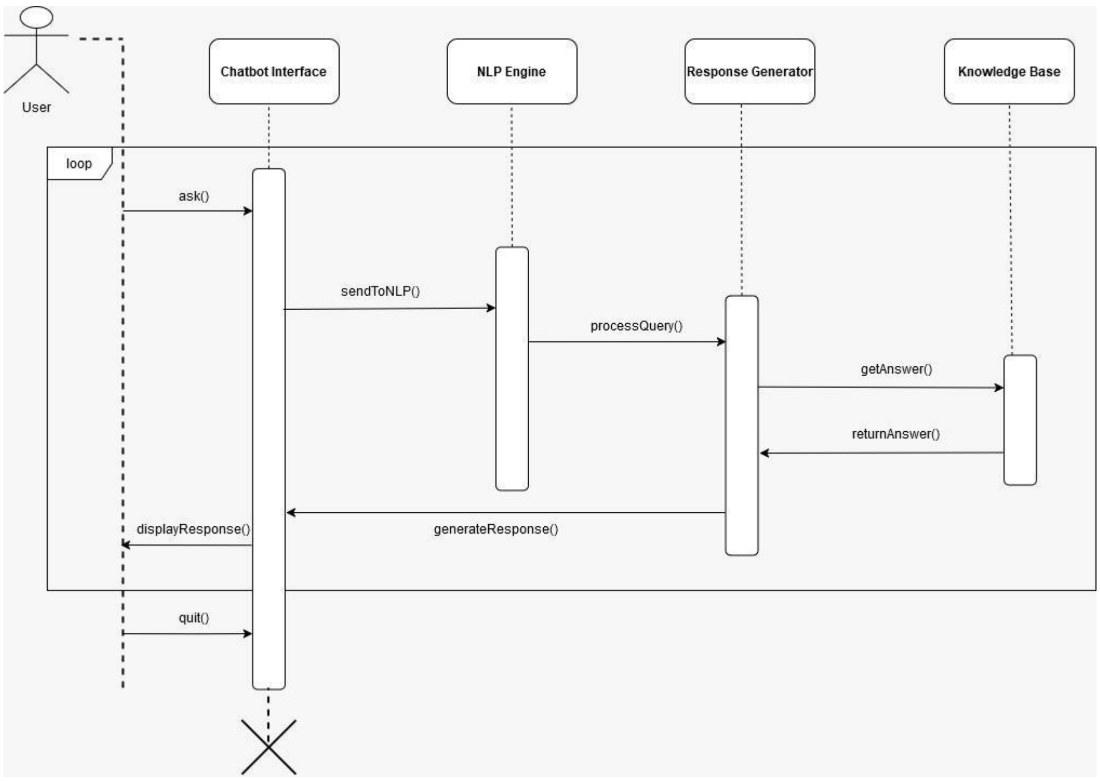
2.2 Data Flow Diagram



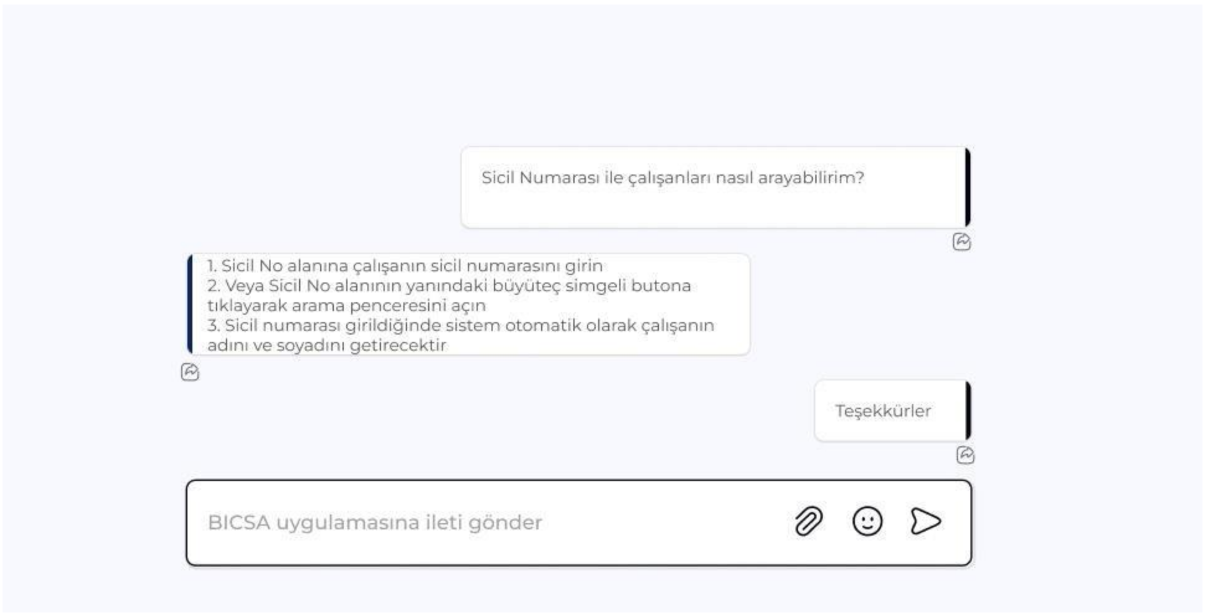
2.3 Activity Diagram



2.4 Sequence Diagram



3. User Interface Design


Sicil Numarası ile çalışanları nasıl arayabilirim?

1. Sicil No alanına çalışanın sicil numarasını girin
2. Veya Sicil No alanının yanındaki büyüteç simgeli butona tıklayarak arama penceresini açın
3. Sicil numarası girildiğinde sistem otomatik olarak çalışanın adını ve soyadını getirecektir

Teşekkürler

1. INTRODUCTION

1.1 Version Control

Version No	Description of Changes	Date
1.0	Initial version of the test plan.	March 24, 2025

1.2 Overview

This document outlines the testing strategy for the AI-powered chatbot developed using the OpenAI Assistant model. The chatbot is designed to respond to user prompts by referencing a set of HTML-based end-user documents, which serve as its primary knowledge base. The test plan aims to ensure that the chatbot accurately retrieves and presents information from this knowledge base in a coherent and contextually appropriate manner.

1.3 Scope

This test plan applies to all components of the chatbot that interact with the knowledge base [1], including:

- Prompt understanding
- Knowledge base document parsing
- Response generation
- Context retention within a session
- Response traceability to HTML content

It includes both manual and automated test cases, test design specifications, and validation strategies to ensure the chatbot performs correctly under various scenarios.

1.4 Terminology

Acronym	Definition
KB	Knowledge Base
HTML	HyperText Markup Language
TC	Test Case
AI	Artificial Intelligence
NL	Natural Language

2. FEATURES TO BE TESTED

2.1 Prompt Understanding (PU)

The chatbot should be able to parse and understand the intent of user input using natural language processing (NLP) techniques.

2.2 Knowledge Retrieval (KR)

The chatbot should extract relevant information from the uploaded HTML documents (knowledge base) based on the user's query.

2.3 Response Generation (RG)

The chatbot should provide human-like, coherent, and accurate responses grounded in the knowledge base content.

2.4 Source Reference Inclusion (SR)

The chatbot should reference specific sections of the knowledge base (e.g., using anchor tags or headings) to justify its response.

2.5 Session Context Management (SCM)

The chatbot should maintain context within a session and provide follow-up answers based on previous interactions.

3. FEATURES NOT TO BE TESTED

• External API integrations (not part of current implementation)

• User Login (not part of current implementation)

• Multilingual support (limited to Turkish for this phase)

4. ITEM PASS/FAIL CRITERIA

A test passes if the chatbot provides an accurate, relevant, and traceable response based on the knowledge base content.

A test fails if:

- The chatbot gives irrelevant or hallucinated content
- No citation/reference to HTML document is included
- The prompt is misunderstood

4.1 Exit Criteria

- 100% of the test cases executed
- At least 95% test case pass rate
- All high and medium priority test cases must pass

5. REFERENCES

[1] Ali Emrecan Selvili, Ahmet Selçuk Özdil, Ege Beçin, Yusuf Tuna Üner, Mehmet Efe Kaya, *Software Requirements Specification (SRS) for Level-0 Support Chatbot for BelsisMIS*, Çankaya University, March 2025.

6. TEST DESIGN SPECIFICATIONS

6.1 Prompt Understanding (PU)

6.1.1 Subfeatures to be tested

- **PU.INT** – Intent Detection: The chatbot correctly identifies the user's goal or information need.
- **PU.ENT** – Entity Recognition: The chatbot detects and isolates important entities within the input prompt.

6.1.2 Test Cases

Here list all the related test cases for this feature:

TC ID	Requirements	Priority	Scenario Description
PU.INT.01	3.1.4.1 – User Inquiry Resolution [1]	H	User asks a question directly based on a heading in the KB
PU.INT.02	3.1.4.1 – User Inquiry Resolution [1]	H	User asks an abstract or vague question to see if chatbot guesses intent
PU.ENT.01	3.1.4.3 – AI-Based Text Response [1]	M	User includes multiple entities; chatbot should resolve them correctly

6.2 Knowledge Retrieval (KR)

6.2.1 Subfeatures to be tested

KR.HDR – Heading-Based Retrieval: Retrieve based on HTML headings
KR.SEQ – Sequential Scanning: Scan full document for matching content beyond headers.

6.2.2 Test Cases

Here list all the related test cases for this feature:

TC ID	Requirements	Priority	Scenario Description
KR.HDR.01	3.1.4.5 – Step-by-Step Guidance [1]	H	Ask a question that maps directly to an HTML heading
KR.SEQ.01	3.1.4.5 – Step-by-Step Guidance [1]	M	Ask a question that requires scanning paragraph text

6.3 Response Generation (RG)

6.3.1 Subfeatures to be tested

- **RG.COH** – Coherent Response: Response must be grammatically correct and readable.
- **RG.COMP** – Complete Answer: Covers all parts of a multi-part question.

6.3.2 Test Cases

Here list all the related test cases for this feature:

TC ID	Requirements	Priority	Scenario Description
RG.COH.01	3.1.4.3 – AI-Based Text Response [1]	H	Provide a prompt and evaluate the linguistic quality of the output
RG.COMP.01	3.1.4.3 – AI-Based Text Response [1]	M	Ask a two-part question; chatbot should answer both parts clearly

6.4 Source Reference Inclusion (SR)

6.4.1 Subfeatures to be tested

- **SR.INL** – Inline References: Include direct reference (e.g., "As described under section 4.1").

6.4.2 Test Cases

Here list all the related test cases for this feature:

TC ID	Requirements	Priority	Scenario Description
SR.INL.01	3.1.4.5 [1]	H	Answer includes inline mention of section or heading
SR.INL.02	3.1.4.5 [1]	M	Chatbot references two or more relevant HTML sections in a single response

6.5 Session Context Management (SCM)

6.5.1 Subfeatures to be tested

- **SCM.FUP** – Follow-Up Prompt Resolution: Chatbot retains context of previous message.
- **SCM.MUL** – Multi-turn Sessions: Handles 3+ interactions with consistent context.

6.5.2 Test Cases

Here list all the related test cases for this feature:

SCM.FUP.01	3.1.4.5 – Step-by-Step Guidance [1]	H	Ask follow-up question using pronouns or ellipsis
SCM.MUL.01	3.1.4.5 – Step-by-Step Guidance [1]	M	Conduct a 3-turn conversation and check for context retention

7. Detailed Test Cases

7.1 PU.INT.01 — Understand Direct Prompt with Clear Intent

TC_ID	PU.INT.01
Purpose	Test if the chatbot can understand a direct, clearly-intentioned question
Requirements	3.1.4.1
Priority	High.
Estimated Time Needed	3 Minutes
Dependency	Knowledge base must be loaded with HTML docs
Setup	Load an HTML file that contains a section titled "How to log in to the system"
Procedure	[A01] User enters the prompt: "How do I log in to the system?"
	[V01] Chatbot analyzes the input and matches it to the relevant HTML section
	[V02] Chatbot responds with accurate login steps
Cleanup	None

7.1 PU.INT.02 — Handle Vague or Abstract Prompt

TC_ID	PU.INT.02
Purpose	Test if chatbot can infer intent from vague input
Requirements	3.1.4.1
Priority	High.
Estimated Time Needed	4 Minutes
Dependency	None
Setup	HTML content loaded with topic coverage across modules
Procedure	[A01] User enters the prompt: "I'm having trouble with reports."
	[V01] Chatbot infers the user is referring to "report generation"
	[V02] Chatbot provides appropriate guidance from KB
Cleanup	None

7.2 PU.ENT.01 — Entity Recognition with Multiple Concepts

Purpose	Test whether the chatbot can resolve multiple entities
Requirements	3.1.4.3
Priority	Medium.
Estimated Time Needed	5 Minutes
Dependency	None
Setup	Load HTML doc with sections on "Password Reset" and "Support"
Procedure	[A01] User enters: "Can I reset my password and contact support?"
	[V01] Chatbot extracts both entities: password reset and support
	[V02] Chatbot provides responses for both topics
Cleanup	None
TC_ID	PU.ENT.01

7.3 KR.HDR.01 — Match Prompt to HTML Heading

TC_ID	KR.HDR.01
Purpose	Test heading-based retrieval
Requirements	3.1.4.5
Priority	High.
Estimated Time Needed	3 Minutes
Dependency	None
Setup	HTML contains <h2>How to upload a document</h2>
Procedure	[A01] User enters: "How can I upload documents?"
	[V01] Chatbot locates heading in KB
	[V02] Chatbot summarizes and responds based on that section
Cleanup	None

7.4 KR.SEQ.01 — Sequential Scanning Retrieval

TC_ID	KR.SEQ.02
Purpose	Test retrieval from paragraph-level, not heading-based
Requirements	3.1.4.5
Priority	Medium.
Estimated Time Needed	3 Minutes
Dependency	None
Setup	HTML document contains embedded paragraph on export limits
Procedure	[A01] User asks: "Is there a size limit when exporting files?"
	[V01] Chatbot searches non-header content
	[V02] Responds with info found in paragraph text
Cleanup	None

7.5 RG.COH.01 — Generate Coherent Response

TC_ID	RG.COH.01
Purpose	Ensure chatbot’s reply is grammatically correct and readable
Requirements	3.1.4.3
Priority	High.
Estimated Time Needed	2 Minutes
Dependency	None
Setup	Knowledge base loaded
Procedure	[A01] User enters: "What does the red button do?"
	[V01] Chatbot analyzes intent and locates related info
	[V02] Chatbot responds in full, coherent sentences
Cleanup	None

7.6 RG.COMP.01 — Multi-Part Answer Completion

TC_ID	RG.COH.01
Purpose	Ensure chatbot handles multi-part queries completely
Requirements	3.1.4.3
Priority	Medium.
Estimated Time Needed	3 Minutes
Dependency	None
Setup	HTML has info on password reset and email change
Procedure	[A01] User asks: "How do I reset my password and change my email?"
	[V01] Chatbot parses both tasks
	[V02] Provides responses to both in one message
Cleanup	None

7.7 SR.INL.01 — Inline Reference in Response

TC_ID	SR.INL.01
Purpose	Validate inline reference to a section
Requirements	3.1.4.5
Priority	High.
Estimated Time Needed	3 Minutes
Dependency	None
Setup	HTML contains section "Exporting Reports"
Procedure	[A01] User enters: "How can I export reports?"
	[V01] Chatbot finds relevant section
	[V02] Chatbot includes phrase like “As described in the 'Exporting Reports' section…”
Cleanup	None

7.7 SR.INL.02 — Inline Reference for Multiple Sections

TC_ID	SR.INL.02
Purpose	Reference two HTML sections in one response
Requirements	3.1.4.5
Priority	Medium
Estimated Time Needed	3 Minutes
Dependency	None
Setup	HTML contains "Data Export" and "Email Reports" sections
Procedure	[A01] User enters: "How can I export reports?"
	[V01] Chatbot finds relevant section
	[V02] Chatbot includes phrase like “As described in the 'Exporting Reports' section…”
Cleanup	None

7.8 SCM.FUP.01 — Handle Follow-Up Prompt

TC_ID	SCM.FUP.01
Purpose	Test context management with follow-up prompts
Requirements	3.1.4.5
Priority	High.
Estimated Time Needed	4 Minutes
Dependency	None
Setup	HTML includes user profile and password management
Procedure	[A01] User enters: "How do I change my email?"
	[V01] Chatbot responds with correct steps
	[A02] User follows with: "What if I forgot my password?"
	[V02] Chatbot understands context and provides password reset steps
Cleanup	None

7.9 SCM.MUL.01 — Multi-Turn Conversation with Context

TC_ID	SCM.FUP.02
Purpose	Verify chatbot retains context for multiple exchanges
Requirements	3.1.4.5
Priority	Medium
Estimated Time Needed	4 Minutes
Dependency	None
Setup	Simulate 3-question user session
Procedure	[A01] User: "How do I create a report?"
	[V01] Chatbot replies
	[A02] User: "How can I export it?"
	[V02] Chatbot continues from same context
	[A03] User: "Can I email it too?"
	[V03] Chatbot delivers cohesive answer integrating all 3 prompts
Cleanup	None

8. Test Results

8.1 Test Result Table

TC_ID	Priority	Result	Explanation
PU.INT.01	High	Pass	The chatbot correctly identifies the user’s goal or information need.
PU.INT.02	High	Pass	The chatbot correctly identifies the user’s goal or information need.
PU.ENT.01	Medium	Pass	The chatbot detects and isolates important entities within the input prompt.
KR.HDR.01	High	Pass	Retrieve based on HTML headings
KR.SEQ.01	Medium	Pass	Scan full document for matching content beyond headers.
RG.COH.01	High	Pass	Response must be grammatically correct and readable.
RG.COMP.01	Medium	Pass	Covers all parts of a multi-part question.
SR.INL.01	High	Pass	Include direct reference (e.g., “As described under section 4.1”).
SR.INL.02	Medium	Pass	Include direct reference (e.g., “As described under section 4.1”).
SCM.FUP.01	High	Fail	Chatbot retains context of previous message.
SCM.MUL.01	Medium	Pass	Handles 3+ interactions with consistent context.

8.2 Exit Criteria

- All test cases executed.
- All essential functionalities pass testing
- 90% or more test cases passed.
- All high priority test cases passed.
- Essential functionalities pass without critical issues.

8.3 Conclusion

After the comprehensive testing phase, our chatbot system has met the predefined success thresholds. All high priority test cases and the majority of medium priority cases have passed. Only one high-priority case (context memory) failed, which is already flagged for further development. Despite this, the chatbot fulfills core requirements and demonstrates acceptable performance in functional and response-related tasks. Therefore, the system can be considered ready for the next phase of development or deployment, with the exception of minor issues being tracked separately.