Azure migration costs calculator with embedded AI

Software Design Document(SDD)

CS 4850 - Section 02 – Fall 2024

Aug 21, 2024

Roles	Name	Major responsibilities	Contact
Project owner	Capstone		
Team leader	Kunal Shenoi	Organize meetings, monitor progress, and submit deliverables. Develop and implement testing protocols.	678-779-4770
Team members	Graham Allen	Developer - Developing back-end AI integration, implementing Azure/AWS API calls, and database connections. Assisting in front-end UI/UX development	770-841-6718
	Angel Hernandez	Developer - Designing and implementing a user-friendly interface, while assisting with backend and AI integration and development	770-318-5359
	Yvan Ngah	Documentation - Creating, maintaining, and ensuring the accuracy of all technical documentation.	171haden@gmail.com
Advisor / Instructor	Sharon Perry	Facilitate project progress; advise on project planning and management.	770-329-3895

Kunal Shenoi
Team Leader, Test

Graham Allen

Developer

Angel Hernandez
Developer

Yvan Ngah
Documentation

Table of Contents

Introduction

1.1 Document Outline

The Azure Migration Cost Calculator with Embedded AI is a web-based application designed to calculate the financial impact of migrating on-premise infrastructure to Azure cloud services. It utilizes AI to enhance cost estimations, provide precise cost predictions, and create customized recommendations and detailed reports based on the client's infrastructure needs. The system also includes a chatbot interface to enhance user interaction while keeping insights up to date through advanced continuous learning.

Design Considerations

2.1 Assumptions and Dependencies

- Related Software: The system assumes integration with existing Azure pricing APIs, and requires stable internet connectivity for data retrieval and AI model updates.
- Operating Systems: The application is designed to be platform-independent, functioning across various operating systems such as Windows, MacOS, and Linux.
- End User Characteristics: Users are assumed to have a basic understanding of cloud technologies as they will interact with the system through a user-friendly web interface.
- Possible Changes in Functionality: Future updates may include enhanced AI capabilities and additional support for other cloud providers.

2.2 General Constraints

- Software Environment: The application must run efficiently on standard server hardware, leveraging cloud-based AI models for computation. In addition, AI or compute-focused hardware could be beneficial to the application.
- End User Environment: Users need access to a stable internet connection and a compatible web browser such as Chrome or Firefox.
- Resources Availability: Continuous access to Azure Pricing APIs is crucial for accurate cost predictions. The most recent cost will be used during downtime or changes in the APIs.
- Security Requirements: The system must comply with industry-standard security protocols to protect client data, especially during API interactions and Personally Identifiable Information (PII) storage.
- Memory Limitations: The AI models require significant memory for processing, which must be managed efficiently to avoid performance bottlenecks, especially for large migrations.
- Performance Requirements: The system must generate cost predictions and recommendations within acceptable time frames.
- Network Communications: A reliable and secure network is essential for retrieving data from Azure services and updating AI models.

2.3 Development Methods

- The development follows an Agile methodology, allowing for iterative progress with regular stakeholder feedback. This approach ensures flexibility in incorporating changes and refining the AI models. The system is built using modern web

technologies, with React/Angular for the front end and Python with Flask for the back end. AI models are integrated using open-source machine learning services, ensuring seamless interaction with Azure Pricing APIs.

Architectural Strategies

- Programming language: The front end is developed using React/Angular, while Python with Flask is utilized for the backend.
- Reuse of Existing Components: The system leverages Azure services API offerings to enhance the efficiency of the development process. In addition, open-source tools are used to improve the AI Model.
- Future enhancements: The architecture is designed in a microservice architecture which is designed to accommodate future enhancements. Providing the model with compute or AI-optimized hardware can significantly improve performance in the future.
- System Input and Output Models: A responsive web design ensures the application is accessible across various devices. The chatbot interface allows for natural language interaction, enhancing the user experience.
- Error Detection and Recovery: The system includes extensive error-handling mechanisms to manage the AI model, user input, and general API failures.
- Memory Management Policies: The system utilizes advanced algorithms to prioritize efficient memory usage, particularly for the AI model.
- External Databases: The system utilizes Azure Databases to seamlessly integrate with other APIs in the system ensuring high security and performance.
- Distributed Data: Data retrieval and process are distributed across the cloud with the use of caching to optimize performance while ensuring data consistency, fault tolerance, and scalability.
- Generalized Approaches to Control: The system ensures that only authorized versions are in use. This includes tracking releases, patches, and bug fixes.
- Concurrency and synchronization: The system utilizes concurrent API requests and AI models to enhance performance, ensuring accurate and timely results.
- Communication Mechanisms: RESTful APIs are utilized for facilitated communication between the front end, back end, and Azure services.
- Management of Other Resources: The system utilizes autoscaling and elasticity to manage costs and system performance.

System Architecture

The system is divided into three main components:

- Front end: The user interface, developed with React/Angular, provides an intuitive and interactive experience for end-users. It communicates with the backend via RESTful APIs.
- Back end: Python with Flask serves as the backend of the system, handling API requests, processing data, interacting with Azure services, and providing API responses to the front end. The AI models are integrated here to provide cost predictions and recommendations.

- AI Component: Integrated with the backend, this component uses an AI chatbot model to deliver precise cost estimates. The AI continuously learns and updates its models based on new data.

Detailed System Design

5.1 Classification

- The web application is structured into several systems, these include the user interface, the backend module, and the AI model. Each of these components works together to deliver a seamless experience for the user while ensuring accurate and efficient processing of the data.

5.2 Definition

- The front end is responsible for delivering the user experience, managing how users interact with the system, and processing user inputs. The backend serves as the core processing unit, handling API calls and processing data promptly. The AI model is the intelligence behind the system as it provides accurate price predictions and tailored recordations based on the data it is provided.

5.4 Constraints

- Each component of the system operates under specific constraints that must be considered to ensure optimal performance. The front end needs to be highly responsive and accessible to provide users with the best user experience. The backend must be highly efficient to handle multiple requests and provide timely responses to the requestor. The AI model is required to deliver accurate predictions while managing memory and processing power to prevent bottlenecks.

5.5 Resources

- The resources required by the components vary based on their roles. The front end relies on web browsers and requires minimal resources for optimal operation. The backend demands a standard or highly performant server to handle data processes and API requests. The AI model depends on the AI model service for model training and execution.

5.6 Interface/Exports

- The system interacts with internal and external entities through well-defined interfaces. The front end and back end interact internally ensuring data flows smoothly through the system. In addition, the AI model, backend, and other external API services communicate efficiently to enable comprehensive and integrated functionality of the system.

Glossary

- API: Application Programming Interface
- AI: Artificial Intelligence.
- RESTFUL: An architecture style for application programming interfaces.
- Azure: A cloud computing service offered by Microsoft.
- PII: Personally Identifiable Information.

Bibliography

- Microsoft Azure Documentation.
- Flask Official Documentation.
- React/Angular Official Documentation.