
# Bilkent University
# Department of Computer Engineering

# Senior Design Project
# T2404
# Compedia

# Final Report

Yaşar Tatlıcıoğlu - 22003856 - yasar.tatlicioglu@ug.bilkent.edu.tr
Serhat Yılmaz - 22002537 - serhaty@ug.bilkent.edu.tr
Bartu Albayrak - 22101640 - bartu.albayrak@ug.bilkent.edu.tr
Anıl Altuncu - 21901880 - anil.altuncu@ug.bilkent.edu.tr
Ece Beyhan - 22003503 - ece.beyhan@ug.bilkent.edu.tr

## Supervisor: Uğur Doğrusöz
## Course Instructors: Atakan Erdem, Mert Bıçakçı

**2 May 2025**

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

## Table of Contents

# Final Report

*T2404: Compedia*

## 1. Introduction

Nowadays, many companies offer similar products, services, or solutions, making it harder for businesses to find the relevant partner or the right service provider that matches their specific needs. A common way of overcoming this problem is using the internet to search for companies that fulfill the needs of businesses. However, this method is often time-consuming and results in a need for more relevant data. Businesses face many challenges in standing out and finding the right partners. Advertisements are often not effective due to the crowdedness of the market. Therefore, for smaller businesses with limited budgets, competing with others is nearly impossible. Moreover, the visibility of these companies is another problem. Search engines promote well-established companies more, making it nearly impossible for smaller companies to be noticed. The credibility of promoted companies is another area for improvement in today's competitive market. Businesses sometimes choose companies with fake reviews or biased information, resulting in unsuccessful partnerships. There can be an overwhelming number of irrelevant results using search engines. These challenges cause a decrease in the effectiveness of the decision-making process and waste valuable partnerships.

Compedia aims to address the challenges businesses face in finding relevant partners and service providers in today's crowded marketplace. The system will serve as a comprehensive, reliable platform where businesses can discover and connect with potential partners based on their specific needs, regardless of company size or advertising budget.

The primary purpose of Compedia is to create a level playing field for businesses of all sizes through an AI-powered semantic search system. By implementing advanced search capabilities using GPT, Compedia will deliver highly relevant search results that match user-specific requirements, eliminating the problem of overwhelming and irrelevant search results that businesses currently encounter with traditional search engines.

Another key purpose of the system is to enhance the visibility of smaller businesses that often struggle to compete with established companies in traditional search environments. Compedia provides equal opportunities for companies to showcase their services and capabilities through verified company profiles, ensuring that businesses are discovered based on relevance rather than advertising spending or search engine optimization tactics.

Compedia also serves as a credibility verification platform. By implementing a root user approval system for company information and regular updates, the platform ensures that all company data is reliable and accurate. This verification process addresses the current issue of businesses making decisions based on potentially fake reviews or biased information, thereby improving the quality of business partnerships.

Additionally, the system streamlines the business decision-making process by providing a user-friendly, responsive interface accessible from any device with an internet connection. With features like a consistent search button in the topbar, intuitive navigation through a sidebar, and clear loading indicators for in-progress requests, Compedia significantly reduces the time businesses spend searching for partners and increases the effectiveness of their decision-making process.

## 2. Requirements Details

### 2.1 Functional Requirements

Outlined below are the functional requirements for the Compedia platform:

#### 2.1.1 Company Service

- The system must allow authorized users to create, view, edit and archive rich company profiles.
- Each profile shall capture: company name, logo, headquarters location, founding date, industry sector(s), business model, free-text description, list of services/products, and past collaborations.
- Users must be able to upload attachments (PDFs, images) and manage access via role-based permissions.
- Support for verification of company profiles can be done by Platform Admins.

#### 2.1.2 Project Service

- The platform must enable users to define new projects linked to one or more company profiles.
- For each project, users must be able to set name, description, start/end dates, status (Pending, Ongoing, Completed).
- The system must send the project owner an email and an in-app notification whenever a new request is assigned or when the project is marked “Completed” and ready for their review.

#### 2.1.3 Semantic Search and Filters

- The system must support free-text and natural-language queries (e.g. Firms that provide AI-powered computer-vision solutions built with OpenCV and TensorFlow) using embedding-based semantic search.
- Provide facet-based filters: industry sector, location.

- Results must be ranked first by semantic relevance and second by business metrics (e.g. profile completeness, project count, review score, location).

#### 2.1.4 Analytics Service

- Deliver interactive time-series graphs of key usage metrics (e.g. profile views, search volumes).
- Users can switch between daily, weekly, and monthly views and slide the window back over multiple periods (e.g. past 7 days, past 4 weeks, past 6 months).
- Render a bubble diagram in which each bubble represents a keyword or tag, with its size proportional to the occurrence count and its color intensity corresponding to the relative frequency.

### 2.2 Non-functional Requirements

#### 2.2.1 Usability

Compedia is a website. Therefore, Compedia is reachable from anywhere with an internet connection to the web. The users can easily use our website. As an example, before login, users enter their email addresses and if the email address exists in our database, the user will be redirected to the login page, otherwise the user will be redirected to the register page. The website will have a user-friendly interface accessible to each user. Our overall website has a responsive design and functions well on each device with an internet connection. The website has a sidebar so that the user can easily navigate to where they are looking for. All in-progress requests are visible to the user with a loading indicator.

#### 2.2.2 Reliability

The company data that is initially entered by the root user should be approved by the system. The company data that will be added by the system will be also reliable since the company data will be obtained from reliable resources as discussed in progress meetings. The company information that the editor user of the relevant company page updated should be approved by the root user. Since enhanced semantic search should return results that are relevant to the specifications in the query, the search function should always work without any irrelevant results. Data backups will be performed weekly, we will store the updated company and user information to our databases.

#### 2.2.3 Performance

● The system should respond to users within 2 seconds under normal load conditions.

● The website should support at least 10.000 users simultaneously without significant performance degradation.

● AI-powered search queries should return results within 5 seconds, even for complex searches involving hybrid search techniques.

● As we will implement a check for email existence at the initial page of the website, the load for checking the email existence will be handled at first and will initially reduce the workload of the system.

#### 2.2.4 Interoperability

● The website will integrate seamlessly with GPT API and databases.

● The user that will create the company page can import files in standard formats (e.g. PDF) for company profile information.

● The website will be compatible with popular web browsers (e.g., Chrome, Firefox, Safari, Edge) and their latest two versions.

#### 2.2.5 Scalability

New users can easily enter the website by logging in from their browsers. Therefore, our backend servers, databases and LLM integration must work to serve nearly 10000 users concurrently. Moreover, adding new features to the website won't affect the working of other systems since we will use Kafka to integrate our servers and let independent development of these servers. Since Milvus is the leading option in terms of scaling among vector databases, we will scale efficiently to accommodate up to 10 million company profiles [1].

#### 2.2.6 Extensibility

● The architecture of the website will support modularity, which will allow new features (e.g., user profiles, semantic search, and company profiles) to be developed, updated, or replaced without affecting other modules.

● REST API will increase the extensibility as it will give developers an opportunity to add new functionalities such as third party libraries.

● Also, the AI model could be changed to enable future upgrades without causing excessive downtime.

● Application will also allow extra AI models or services to increase the range of features such as predictive analytics.

● Addition of new user roles with associated permissions will be supported to allow for future expansion without major changes to the codebase.

● We will use Kafka for maintenance of data pipelines to enable real-time data flow and processing which will simplify the addition of new data sources or services.

## 3. Final Architecture and Design Details

### 3.1 Overview

Compedia is an AI-driven business discovery platform designed to help companies find and connect with relevant partners in a highly competitive market. Traditional search engines often prioritize larger businesses with high advertising budgets, making it difficult for smaller companies to gain visibility. Compedia levels the field by utilizing semantic search powered by GPT, ensuring that search results are based on relevance.

The platform integrates a microservices-based architecture, enabling modularity, scalability, and high availability. Key components include authentication and authorization services for secure access, and specialized services for handling company management, projects, analytics, and notifications. The backend leverages ASP.NET Core, MicrosoftSQL, and Kafka for event-driven communication, ensuring efficiency in data processing and user interactions.

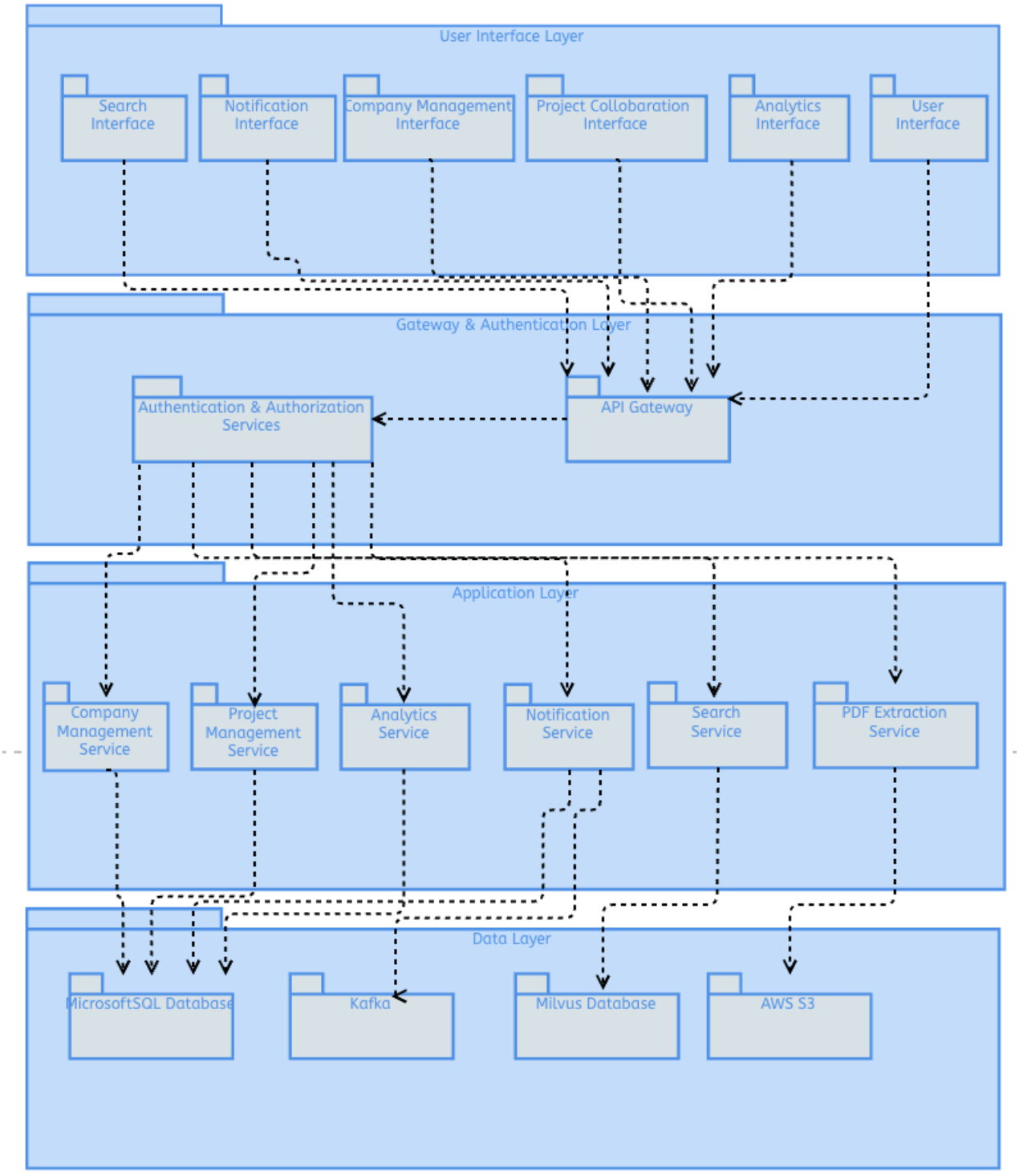
As the Subsystem Decomposition outlines, Compedia follows a layered architecture where the user interacts with a responsive web interface, which in turn communicates with the backend microservices responsible for processing search queries, managing company data, and handling project collaborations. These microservices interact with MicrosoftSQL for structured data storage, Milvus for semantic search

indexing, and S3 for media file storage. Additionally, Hardware/Software Mapping describes the infrastructure supporting the architecture, while Persistent Data Management and Access Control and Security ensure data integrity, user authentication, and system reliability.

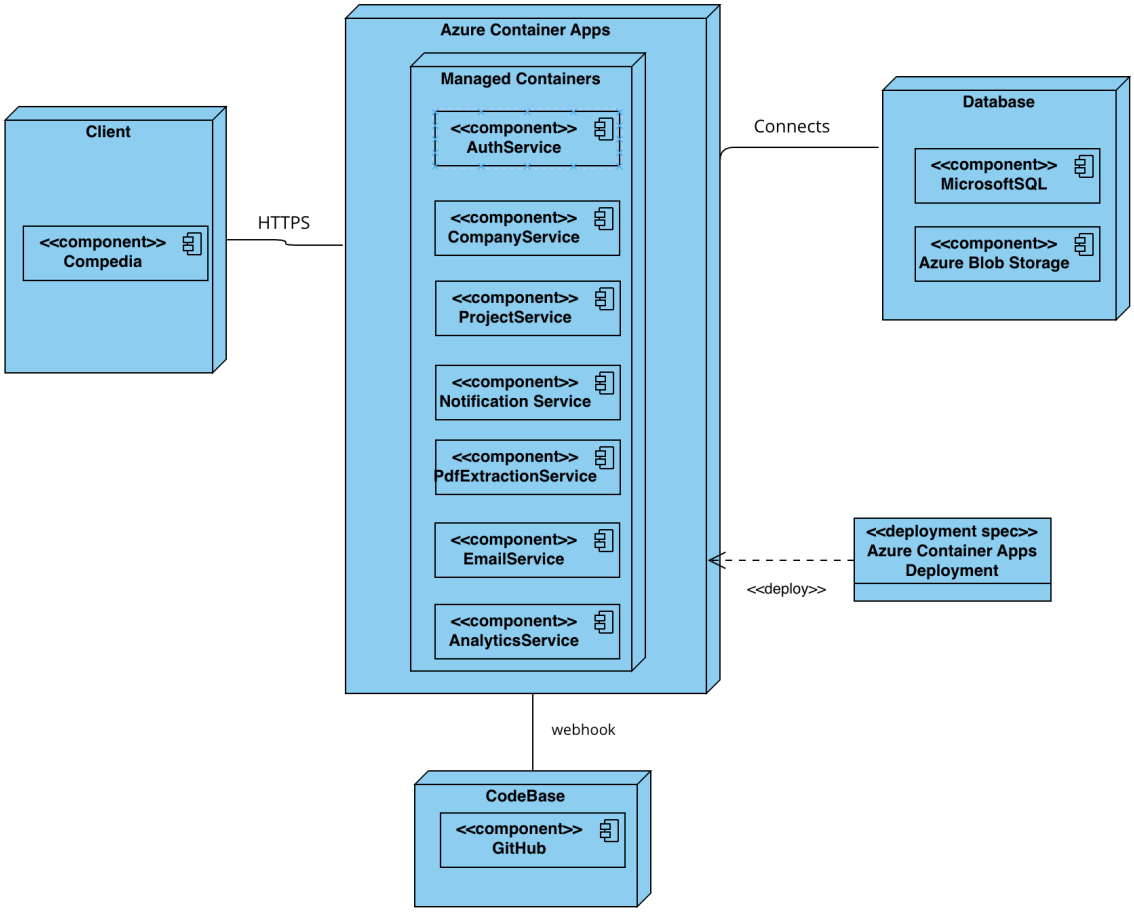
### 3.2 Subsystem Decomposition

We have designed a four-layer architecture, which consists of the User Interface Layer, Authentication Layer, Application Layer, and Data Layer. Each layer is independent of the others and can only interact with the layer below it. This structure ensures a clear separation of concerns, making the system modular, scalable, and maintainable.

To enhance system security, we enforce role-based access control (RBAC) and authentication mechanisms through our Authentication & Authorization Service, which validates all incoming requests before they reach the backend. We utilize Azure Blob Storage, ensuring efficient file (image) management and retrieval. The event-driven communication model, powered by Kafka, allows real-time updates and asynchronous messaging between microservices, improving overall system responsiveness and reliability.


User Interface Layer
Search Interface
Notification Interface
Company Management Interface
Project Collobaration Interface
Analytics Interface
User Interface
Gateway & Authentication Layer
Authentication & Authorization Services
API Gateway
Application Layer
Company Management Service
Project Management Service
Analytics Service
Notification Service
Search Service
PDF Extraction Service
Data Layer
MicrosoftSQL Database
Kafka
Milvus Database
AWS S3

### 3.3 Hardware/Software Mapping




Compedia is designed to run as a fully web-based application, providing accessibility across desktop and mobile browsers. The client application will be built as a responsive web application.

All core business logic and data processing will reside on Microsoft Azure cloud servers. The backend microservices will be containerized and deployed on Azure Container Apps, eliminating the need for manual server management. API requests from the client application will be securely transmitted via HTTPS to the backend services, which will be managed through Azure API Management.

For data storage, Compedia will rely on Microsoft SQL (Azure SQL Database) for structured business data, Milvus for AI-powered semantic search indexing, and Azure Blob Storage for handling file storage. To facilitate event-driven communication and notifications, Azure Event Hub will be employed to process real-time events asynchronously, ensuring efficient handling of system-wide updates and messages.

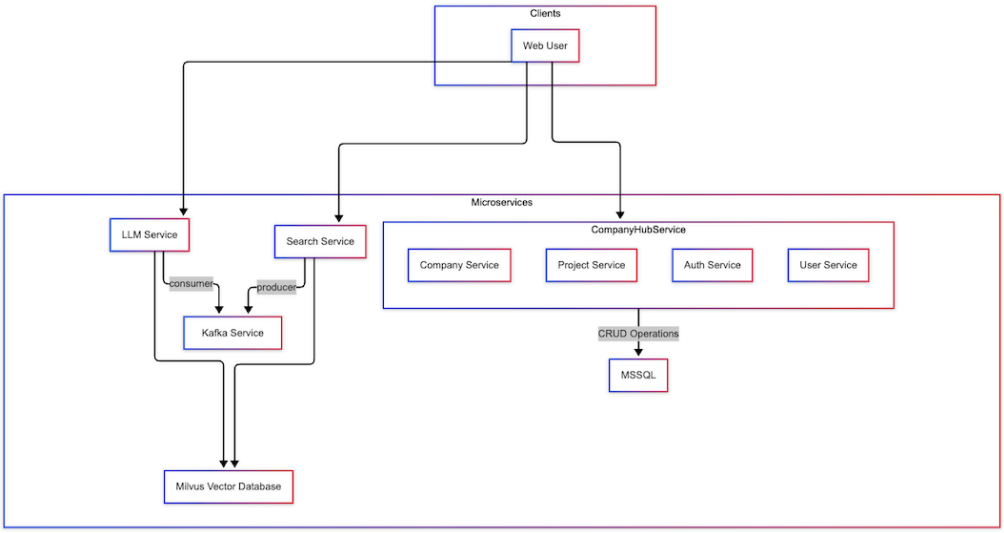
The deployment pipeline will be fully automated, with the codebase hosted on GitHub and deployment processes managed through Azure DevOps and GitHub Actions. This setup will ensure that any push to

the main branch automatically triggers a new deployment, keeping the system continuously updated without manual intervention.

### 3.4 System Architecture

This section describes how the Compedia platform is organized in both development (local) and production (AWS cloud) environments. For local development, we have adopted a temporary, single-machine architecture to enable rapid iteration and end-to-end testing. In production, each component is mapped onto managed AWS services to ensure scalability, reliability, and security.

#### 3.4.1 Local Architecture



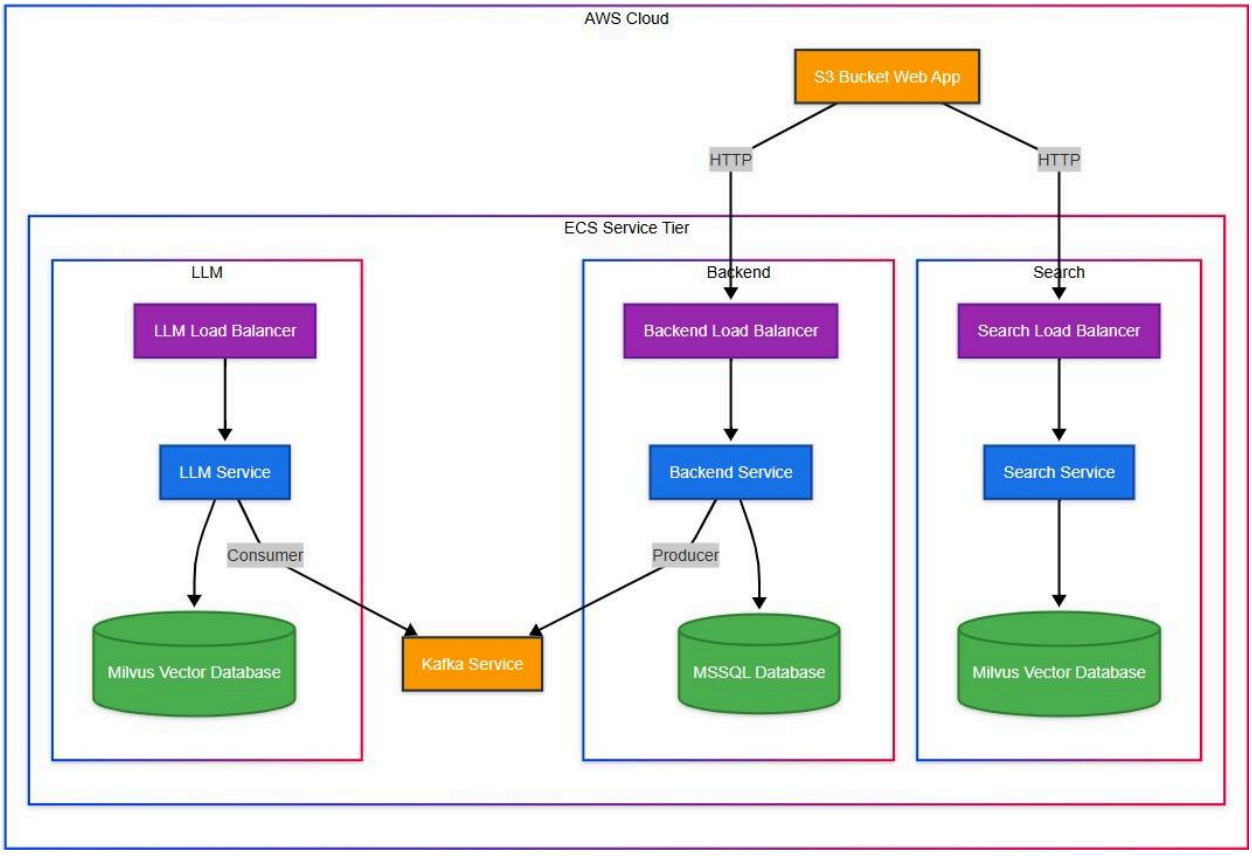

The local design of Compedia is designed around a microservices-based approach, where different services handle specific tasks to ensure scalability, modularity, and efficient data processing. At the client side, Web Users interact with the backend services, which include the LLM Service for AI-based language processing, the CompanyHubService for managing company and project data, and the Search Service for performing search related tasks. These services interact with both a Milvus Vector Database for

unstructured, vector-based data storage and an MS SQL relational database for managing structured data such as company and user information.

The CompanyHubService handles CRUD operations for company data and user management, utilizing the MSSQL database for persistent storage. Meanwhile, the LLM Service and Search Service both connect to the Milvus Vector Database to handle tasks related to AI models and vector search, respectively. Communication between the Search Service and the LLM Service is facilitated by Kafka, allowing asynchronous message exchange. This architecture ensures a flexible, scalable system capable of processing both structured and unstructured data efficiently, with clear separation of concerns between the services

All binary files (profile images) live in Azure Blob Storage, while structured data is in Azure SQL and vector embeddings in Milvus. Continuous integration and deployment are fully automated via GitHub Actions, ensuring that any push to main rolls out updates to all services without manual intervention.

### 3.4.2 AWS Cloud Architecture



The architecture diagram for the Compedia project in AWS Cloud illustrates the deployment and interactions between various services within the system. The architecture is structured into three primary tiers: the AWS Cloud, ECS Service Tier, and the S3 Bucket Web App. At the core, the architecture is composed of services such as the LLM (Large Language Model), Backend, and Search, each of which interacts with separate databases and load balancers to ensure scalability and high availability.

The LLM tier includes a dedicated LLM Load Balancer that directs traffic to the LLM Service, which then communicates with the Milvus Vector Database. Kafka Service is used to manage the communication flow between the Consumer and Producer, ensuring that data is efficiently processed and transferred. On the Backend side, the Backend Load Balancer ensures proper traffic distribution to the Backend Service, which communicates with the MSSQL Database for data storage and retrieval. Search Load Balancer and Search Service, ensures robust and scalable search capabilities while connecting to the Milvus Vector Database.

This architecture allows for high-performance data handling and processing, ensuring that both search and backend services can function independently, with the flexibility to scale as needed. The integration of Kafka ensures real-time data streaming and communication across services, enhancing the overall responsiveness of the system. The use of vector databases like Milvus ensures that both the LLM and Search services are optimized for handling complex data and queries efficiently.

### 3.5 Persistent Data Management

To store structured application data, including company profiles, user information, project details, and collaboration records, Microsoft SQL (Azure SQL Database) will be used as the primary relational database. Microsoft SQL provides a highly reliable and scalable solution for managing structured business data while ensuring ACID compliance for transaction consistency.

For AI-powered semantic search and recommendation capabilities, Milvus will be used as the vector database. Milvus enables efficient storage and retrieval of high-dimensional embeddings for business discovery, allowing users to find relevant companies based on content similarity rather than simple keyword matching.

In addition to structured data, Compedia will require persistent storage for files such as images. These images will be stored in Azure Blob Storage, which provides scalable and durable object storage.

### 3.6 Access Control and Security

JSON Web Tokens (JWTs) are utilized for managing access control. When a request reaches our API, we first verify whether a JWT has been provided by the user. If a token is present, we proceed to validate its authenticity. Once the user is successfully authenticated, we check their assigned role to ensure they have the appropriate permissions for the requested action and confirm that they are only accessing or modifying their own data.

To enhance security, we do not store any confidential information in our database. Additionally, passwords are never stored in their original form; instead, they are securely encrypted. The plaintext version of a password is only accessible during the initial signup process and is never stored or displayed afterward.

## 4. Development/Implementation Details

### 4.1. System Architecture

Our system follows a modern client-server architecture divided into three primary components:

Frontend: A React-based single-page application that provides users with an intuitive interface
Backend: A .NET Core API service responsible for business logic, data access, and authentication

The architecture adopts a microservice approach where each component operates independently and communicates through well-defined interfaces, allowing for modularity and scalability.

### 4.2. Backend Implementation

**Core Technologies**

The backend is built using ASP.NET Core 9.0, which offers excellent performance and cross-platform capabilities. For data persistence, we employ SQL Server with Entity Framework Core as our ORM (Object-Relational Mapper), enabling efficient database operations with strongly-typed models. Authentication is implemented using JWT (JSON Web Tokens) combined with ASP.NET Identity for secure user management. For asynchronous communication between services, we utilize Kafka as our messaging system, facilitating an event-driven architecture.

**Data Models**

The database schema centers around several key entities that form the foundation of our system:

- Company: Represents business organizations with their profiles, contact information, and operational details
- User: Extends ASP.NET Identity user model with additional properties specific to our application
- Project: Encapsulates collaborative work initiatives between companies
- Service: Defines the various offerings that companies can provide
- Industry: Categorizes companies based on their specializations

These entities are connected through carefully designed relationships. Companies can offer multiple services, operate in various industries, and participate in numerous projects. Users are associated with specific companies, defining their roles and permissions within the system.

**Service Layer**
Our backend implements a comprehensive service layer that encapsulates all business logic, providing a clean separation from data access and API controllers:

- CompanyService: Manages company profile creation, modification, and retrieval operations
- ProjectService: Handles the lifecycle of projects, including creation, requests, approvals, and collaboration
- UserService: Manages user data, authentication processes, and company associations
- AdminService: Provides administrative capabilities for system verification and management
- AnalyticsService: Collects and processes usage data for insights and recommendations

Each service enforces business rules, validates inputs, and orchestrates interactions between different parts of the system.

**Integration with Kafka**
The system leverages Kafka for asynchronous communication between services, enabling several important features:

- Real-time data synchronization between the main database and the intelligent search service
- Event notifications for user activities and system changes
- Maintaining data consistency across microservices
- Reducing coupling between system components

This messaging infrastructure allows our application to scale horizontally while maintaining performance and reliability.

### 4.3. Frontend Implementation

**Core Technologies**
The frontend is developed using React with functional components and hooks, providing a modern and efficient user interface. Material-UI (MUI) serves as our UI component library, ensuring consistent design and responsive layouts. For state management, we employ React Context API combined with local component state to maintain a clean architecture. Navigation is handled through React Router, while Axios serves as our HTTP client for API communication.

**Key Components**

- **Authentication & Authorization**

The frontend implements a robust authentication system using JWT tokens stored securely in browser storage. Protected routes ensure that authenticated users can only access appropriate

sections based on their role and permissions. The system maintains session state across page refreshes and handles token expiration gracefully.

- **Company Profile Management**

This module allows companies to create and manage detailed profiles showcasing their capabilities, services, and industry focus. Companies can update their information, add representatives, and customize their public presence. The interface provides intuitive forms with real-time validation and preview capabilities.

- **Project Collaboration**

The project collaboration module facilitates creating, joining, and managing collaborative initiatives between companies. It includes features for proposal submission, project tracking, and milestone management. The interface visualizes project timelines and team structures through interactive dashboards.

### 4.4. Search Service Implementation

Our search service is built on Elasticsearch, providing powerful company discovery capabilities through a Flask-based API layer. The implementation focuses on delivering precise, contextually relevant results for business partner discovery.

**Technical Implementation**

- **Index Design**: We've designed specialized indices for companies, services, and industries with carefully mapped field types (text, keyword, geo_point) and custom analyzers
- **Query Construction**: The service builds complex multi-field queries that combine Boolean logic, fuzzy matching, and field boosting to capture user intent accurately
- **Custom Scoring**: We've implemented custom scoring functions that incorporate both textual relevance and business factors like company verification status and activity level
- **Analyzer Chain**: Text fields are processed through our domain-specific analyzer chain that includes tokenization, stemming, synonym expansion, and stopword removal tailored for business terminology
- **Indexing Pipeline**: A robust indexing pipeline ensures data consistency between the primary database and search indices using Kafka-based event processing

**Advanced Search Features**

- **Multi-criteria search**: Combines free text search with structured filters for precise partner discovery
- **Semantic understanding**: Analyzes query intent using word embeddings to identify expertise requirements

- **Faceted filtering**: Generates dynamic facets based on result distribution for intuitive result refinement
- **Relevance ranking**: Orders results according to a multi-factor scoring algorithm

### 4.5. LLM Service Implementation

Our LLM service enhances the platform with intelligent recommendation capabilities, built on a Python stack with GPT models, and specialized vector database technologies.

**Technical Implementation**

- **Vector Representation**: Company profiles are converted to high-dimensional vectors using a GPT's modern embedding model.
- **Similarity Computation**: We employ cosine similarity and Euclidean distance metrics within specialized vector indices for efficient similarity computation
- **Hybrid Model Architecture**: Our recommendation engine combines collaborative filtering (matrix factorization) with content-based approaches (semantic similarity) to address cold-start problems
- **Feature Engineering**: We extract and transform multiple features from company profiles, interaction history, and industry classifications to create comprehensive company representations

**Competitive Analysis**

- **Similar Company Identification**: Identifies companies with similar profiles and expertise to facilitate industry networking and to identify possible competitors

**Integration Layer**

Both the search and machine learning services integrate with the main application through:

- **Event-Driven Updates**: Kafka topics for company updates, user interactions, and system events ensure data consistency
- **REST API**: A unified API layer handles request routing, authentication, and response formatting
- **Batch Processing**: Scheduled jobs handle computationally intensive operations like reindexing

This integration ensures that users receive intelligent, contextually relevant information throughout their journey on the platform.

### 4.6. DevOps and Deployment

Our system employs Docker containers for consistent deployment across environments. CI/CD pipelines automate testing, building, and deployment processes, ensuring reliable releases. The infrastructure is hosted on cloud platforms with auto-scaling capabilities to handle varying loads efficiently.

### 4.7. Security Implementations

Security is implemented at multiple layers including:

- Data encryption in transit using TLS/SSL
- Password hashing using industry-standard algorithms.
- Role-based access control for API endpoints.
- Input validation and sanitization to prevent injection attacks.
- Regular security audits and dependency updates.

This comprehensive approach to security ensures that our system protects sensitive business information and maintains user trust.

## 5. Test Cases and Results

Below are our test cases that we conducted. Except Test Cases 23, 26, 27, 28 and 48, all tests turned out to be successful. The other test cases which failed were not implemented as they did not affect the main goal of our project and due to time constraints.

**Table X:** Test Cases of Compedia

| Test ID | TC001 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | When signing up, the password entered by the user should contain at least 6 characters including 1 uppercase, 1 lowercase and 1 symbol. | | | | |
| Objective | Verify that the password entered by the user is valid | | | | |
| Steps | 1. Open the application's sign up page.<br>2. Try signing up with a password that does not include at least 6 characters<br>3. Try signing up with a password that does not include at least 1 uppercase letter<br>4. Try signing up with a password that does not include at least 1 lowercase letter<br>5. Try signing up with a password that does not include at least 1 symbol | | | | |
| Expected | Invalid passwords should not be accepted and relevant fail notification should appear on the screen. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC002 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | System must redirect unauthenticated access attempts to the login page. | | | | |
| Objective | Verify unauthenticated access prevention | | | | |
| Steps | 1. Open the application in a browser (without logging in).<br>2. Directly enter URLs for authenticated pages into the browser's address bar<br>3. Observe the application's response to each request. | | | | |
| Expected | User redirected to login page | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC003 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | System must respond within 5 seconds under normal usage | | | | |
| Objective | Measure application response time under typical load | | | | |
| Steps | 1. Log in to the application.<br>2. Navigate through pages.<br>3. Perform actions.<br>4. Observe how long it takes for pages/actions to complete. | | | | |
| Expected | Response within 5 seconds consistently | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC004 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Users must log in successfully with valid credentials | | | | |
| Objective | Verify successful login functionality | | | | |
| Steps | 1. Open login page<br>2. Enter valid credentials<br>3. Click on login | | | | |
| Expected | User logged in successfully | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC005 | Category | Non-Functional | Severity | Minor |
|---|---|---|---|---|---|
| Requirement | Application must be compatible across all major browsers (Chrome, Firefox, Safari, Edge) | | | | |
| Objective | Verify browser compatibility | | | | |
| Steps | 1. Open application in Chrome, Firefox, Safari, Edge<br>2. Perform basic actions | | | | |
| Expected | Application functions correctly in all browsers | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC006 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Logout action must effectively terminate the user's session | | | | |
| Objective | Verify logout functionality | | | | |
| Steps | 1. Log into application<br>2. Perform logout<br>3. Attempt accessing authenticated pages | | | | |
| Expected | Session terminated; user unable to access authenticated pages without re-login | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC007 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | System must enforce access control for protected areas | | | | |
| Objective | Verify access control restrictions | | | | |
| Steps | 1. Open browser<br>2. Directly enter admin page URL | | | | |
| Expected | Unauthorized pages inaccessible | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC008 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Import functionality must accurately process valid PDF files for company profile creations made by user | | | | |
| Objective | Verify accurate data import functionality | | | | |
| Steps | 1. Log into the application<br>2. Go to create company profile section<br>3. Upload a PDF file which contains relevant company information | | | | |
| Expected | Data imported successfully and correctly reflected in the text fields of company creation page | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC009 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Import functionality must reject invalid PDF files | | | | |
| Objective | Verify handling of invalid data imports | | | | |
| Steps | 1. Log into application<br>2. Go to create company profile page<br>3. Attempt upload of incorrectly formatted or corrupted PDF file | | | | |
| Expected | System rejects import and provides clear error message | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC010 | Category | Non-Functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | Role-based permissions must be accurately enforced | | | | |
| Objective | Validate accurate enforcement of role-based permissions | | | | |
| Steps | 1. Test system actions with varying roles and permissions | | | | |
| Expected | Roles and permissions accurately enforced | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC011 | Category | Non-Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | AI-based semantic search queries must return results within 5 seconds | | | | |
| Objective | Measure performance of semantic search functionality | | | | |
| Steps | 1. Enter various semantic queries in search bar<br>2. Execute searches and measure response times | | | | |
| Expected | Queries consistently completed within 5 seconds | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC012 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Company profile updates must be correctly saved and be reflected. | | | | |
| Objective | Verify company profile update functionality | | | | |
| Steps | 1. Login and navigate to existing company profile<br>2. Modify profile details (e.g., address, services)<br>3. Save changes | | | | |
| Expected | Updates accurately saved and immediately visible | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC013 | Category | Functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | Semantic search must return highly relevant company profiles | | | | |
| Objective | Confirm accuracy of semantic search functionality | | | | |
| Steps | 1. Enter different semantic search queries<br>2. Review returned results | | | | |
| Expected | Relevant company profiles returned matching semantic intent | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC014 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | User enters all the necessary information for the company profile creation | | | | |
| Objective | Verify required fields are filled in the company profile creation | | | | |
| Steps | 1. Open company create page<br>2. Enter information into the text fields except the company name and try to submit the form<br>3. Enter information into the text fields except the foundation year and try to submit the form<br>4. Enter information into the text fields except the address and try to submit the form<br>5. Enter information into the text fields except the email and try to submit the form | | | | |
| Expected | Company profile should not be not created and a relevant fail submit message should be reflected on the screen. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC015 | Category | Non-Functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | The failure of one microservice must not crash the entire system. | | | | |
| Objective | Verify that system resilience and failover mechanisms work as intended. | | | | |
| Steps | 1. Simulate a failure in one microservice (e.g., the LLMService).<br>2. Monitor system behavior and error handling. | | | | |
| Expected | Other services should remain operational, and the system should recover gracefully. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC016 | Category | Non-Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | All data must be transmitted securely using HTTPS. | | | | |
| Objective | Ensure encryption of data in transit. | | | | |
| Steps | 1. Try to log in to the application with user credentials<br>2. Use network monitoring tools to inspect data transmissions between the client and server. | | | | |
| Expected | All communications should be encrypted with HTTPS, ensuring secure data transfer. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC017 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | JWT tokens must expire as intended and require re-authentication. | | | | |
| Objective | Verify that expired tokens are rejected and proper re-authentication is enforced. | | | | |
| Steps | 1. Log in to obtain a valid token.<br>2. Allow the token to expire.<br>3. Attempt to access a protected resource. | | | | |
| Expected | Access should be denied, prompting the user to log in again. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC018 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | User sessions must automatically time out after a period of inactivity. | | | | |
| Objective | Validate that sessions are terminated correctly after the timeout period. | | | | |
| Steps | 1. Log in and remain idle for a period exceeding the session timeout threshold.<br>2. Attempt to navigate to a protected page. | | | | |
| Expected | The system should redirect the user to the login page due to session expiration. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC019 | Category | Non-functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | All user inputs must be sanitized to prevent SQL injection attacks. | | | | |
| Objective | Ensure the application is secure against SQL injection. | | | | |
| Steps | 1. Input SQL injection code in login and search fields.<br>2. Observe the console messages | | | | |
| Expected | The input should be sanitized, and no unauthorized access or data leakage should occur. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC020 | Category | Non-functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | API endpoints should implement rate limiting to prevent abuse. | | | | |
| Objective | Verify that excessive API calls are throttled or blocked. | | | | |
| Steps | 1. Log in to the application<br>2. Rapidly send multiple API requests in succession. | | | | |
| Expected | Excessive requests should be throttled or rejected, with a rate-limit error message returned. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC021 | Category | Non-functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | The interface must render appropriately on devices with different screen resolutions. | | | | |
| Objective | Ensure proper layout and usability across desktops, laptops, and tablets. | | | | |
| Steps | 1. Open the application on devices with varying resolutions or use browser developer tools to simulate them. | | | | |
| Expected | The layout should adapt without visual or functional issues. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC022 | Category | Functional | Severity | Minor |
|---|---|---|---|---|---|
| Requirement | The sidebar and topbar navigation menus must provide smooth transitions. | | | | |
| Objective | Verify that all navigation elements correctly direct the user. | | | | |
| Steps | 1. Click on each menu item in the sidebar<br>2. Click on each menu item in the topbar. | | | | |
| Expected | The application should navigate smoothly to the intended pages. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC023 | Category | Functional | Severity | Minor |
|---|---|---|---|---|---|
| Requirement | The search bar should be consistently available across all pages. | | | | |
| Objective | Ensure the search feature is accessible at all times. | | | | |
| Steps | 1. Log in to the application<br>2. Navigate through various pages of the application. | | | | |
| Expected | The search bar should remain visible and functional on every page. | | | | |
| Date-Result | 02.05.2025 Failed, search bar only exists in homepage and search page. | | | | |

| Test ID | TC024 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Semantic search must tolerate minor typos or misspellings. | | | | |
| Objective | Validate the robustness of the search algorithm. | | | | |
| Steps | 1. Enter search queries with minor misspellings. | | | | |
| Expected | The returned results should remain relevant despite the typos. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC025 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | The system must prevent the creation of duplicate company profiles. | | | | |
| Objective | Enforce uniqueness constraints for company profiles. | | | | |
| Steps | 1. Log in to the application with a root user's credentials<br>2. Attempt to create a new company profile using an already existing company's information | | | | |
| Expected | The system should reject the duplicate entry with a clear error message. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC026 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | The root user must be able to delete a company profile he owns. | | | | |
| Objective | Validate that deletion operations work correctly. | | | | |
| Steps | 1. Log in and select the existing company profile that you own.<br>2. Initiate the delete action. | | | | |
| Expected | The profile should be removed from the system with a confirmation message. | | | | |
| Date-Result | 02.05.2025 Failed, company profiles cannot be deleted in current implementation. | | | | |

| Test ID | TC027 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | The system must deliver real-time notifications when it is needed. | | | | |
| Objective | Verify that critical events trigger immediate notifications. | | | | |
| Steps | 1. Perform an action that generates a notification (e.g., a project request is received). | | | | |
| Expected | The notification should be delivered in real time to the appropriate user or company. | | | | |
| Date-Result | 02.05.2025 Failed, notification system is not implemented, users only get emails in certain cases. | | | | |

| Test ID | TC028 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Users must have the ability to delete their accounts. | | | | |
| Objective | Validate that the account deletion process works correctly. | | | | |
| Steps | 1. Log in and navigate to the account settings.<br>2. Initiate the account deletion process. | | | | |
| Expected | The account should be removed, and a confirmation should be provided. | | | | |
| Date-Result | 02.05.2025 Failed, deletion is not possible in current implementation. | | | | |

| Test ID | TC029 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Users must be able to reset their forgotten passwords securely. | | | | |
| Objective | Validate the password reset workflow. | | | | |
| Steps | 1. Use the “Forgot Password” link on the login page.<br>2. Follow the reset instructions received via email. | | | | |
| Expected | The user should be able to reset their password and log in with the new credentials. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC030 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | New user registrations must include an email verification step. | | | | |
| Objective | Ensure that new users verify their email addresses before gaining full access. | | | | |
| Steps | 1. Register a new account.<br>2. Check for and follow the verification link sent via email. | | | | |
| Expected | The account should be marked as verified upon completing the process. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC031 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | After a password reset, existing sessions must be invalidated. | | | | |
| Objective | Ensure that sessions' with previous password are invalidated | | | | |
| Steps | 1. Log in to the application<br>2. Reset the password<br>3. Attempt to continue using the old session. | | | | |
| Expected | The old session should be terminated, requiring a new login. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC032 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Administrators must be able to change user roles. | | | | |
| Objective | Validate role management and permission updates. | | | | |
| Steps | 1. Log in as an administrator.<br>2. Change the role of a selected user and verify updated permissions. | | | | |
| Expected | The user's role should be updated and corresponding access rights should be adjusted. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC033 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Unverified users should not create company profiles | | | | |
| Objective | Verify that only the verified users create company profiles. | | | | |
| Steps | 1. Log in the application with an unverified user's credentials<br>2. Open the company create page<br>3. Fill the form and attempt to create the company | | | | |
| Expected | Company should not be created and an appropriate error message should be reflected | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC034 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | An email should be sent to the company whose profile has been created by a user. | | | | |
| Objective | Ensure that the company is aware of that a user has created their account on the application | | | | |
| Steps | 1. Log in to the application with the credentials of a verified user<br>2. Navigate to the company create page and submit the form<br>3. Check the inbox of the company's email address whose profile has been created | | | | |
| Expected | An email should be sent to the company's email address | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC035 | Category | Functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | Only the verified companies can create projects. | | | | |
| Objective | Verify that unverified companies cannot create projects | | | | |
| Steps | 1. Log in to the application with the credentials of an unverified company<br>2. Go to the create project page and attempt to create it. | | | | |
| Expected | Project should not be created and an error message should be shown. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC036 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Only the root user of the company can add people to the company. | | | | |
| Objective | Ensure that the company's people management is under control of authorized user(s). | | | | |
| Steps | 1. Log in to the application with the credentials of a normal user (not root user).<br>2. Go to the people page of the user's company<br>3. Click on add people button on the page and attempt to add people | | | | |
| Expected | An error message should be shown that only the authorized users (root users) can do it and should not allow operation. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC037 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | A user should be added to the company's people if he clicks on the invitation link sent to his email address by the root user. | | | | |
| Objective | Ensure that users are added to the company's people after clicking on the invitation link. | | | | |
| Steps | 1. Send invitation link of a company to a person's email address<br>2. Click on the link sent via email<br>3. Check the company's people page | | | | |
| Expected | The added person should appear on the people list of the company. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC038 | Category | Functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | A project created by a company should be approved by the partner company to make it visible on both companies' profile pages. | | | | |
| Objective | Verify that only mutually accepted projects are visible. | | | | |
| Steps | 1. Create a project and add a partner company<br>2. Without the approval of the partner company, check both companies' profile pages | | | | |
| Expected | The recently created project should not be visible on profile pages yet. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC039 | Category | Functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | Project completion should be approved by both companies to make it marked completed on profile pages of companies. | | | | |
| Objective | Ensure that both companies accept the end of the project. | | | | |
| Steps | 1. Create a project and add a partner company<br>2. Mark it completed for one company<br>3. Check the both companies' profile pages | | | | |
| Expected | The project should be seen as not completed yet on both profile pages. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC040 | Category | Functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | A project should be completed to access the rating system. | | | | |
| Objective | Verify that the companies can rate their partners only if their project is completed. | | | | |
| Steps | 1. Create a project by adding partner company<br>2. Try to rate partner company | | | | |
| Expected | Rate system should not be allowed and an error message should be shown. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC041 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | A client company must be able to submit a new project and have it stored with a “Pending” status while notifying the appropriate provider company. | | | | |
| Objective | Verify that the project submission process saves all details correctly and triggers a notification. | | | | |
| Steps | 1. Log in as a client company user with the proper “edit” permission.<br>2. Navigate to the “Create Project” section.<br>3. Enter all required project details and submit the form.<br>4. Check the project record in the database for a “Pending” status.<br>5. Verify that a notification is sent to the designated provider company. | | | | |
| Expected | The project should be saved with a “Pending” status, and the provider company should be notified of the new project submission. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC042 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | When a provider company accepts a project, the system must update the status to “Accepted” and notify the client company. | | | | |
| Objective | Confirm that acceptance actions are correctly handled and reflected. | | | | |
| Steps | 1. Log in as a provider company user.<br>2. Access a project submitted by a client (currently “Pending”).<br>3. Accept the project using the provided action (e.g., click an “Accept” button).<br>4. Verify the project’s status updates to “Accepted” in the system.<br>5. Check that the client company receives a notification about the acceptance. | | | | |
| Expected | The project status should be changed to “Accepted” and both companies should receive appropriate notifications. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC043 | Category | Non-functional | Severity | Minor |
|---|---|---|---|---|---|
| Requirement | Login operations should complete within 2 seconds under normal conditions. | | | | |
| Objective | Ensure that the application responds a login request within 2 seconds | | | | |
| Steps | 1. Log in with valid credentials multiple times under normal load.<br>2. Measure and record the response time for each login attempt. | | | | |
| Expected | Each login should be completed in 2 seconds or less. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC044 | Category | Non-functional | Severity | Minor |
|---|---|---|---|---|---|
| Requirement | Common UI elements (such as headers, footers, and navigation menus) must display uniformly across all pages. | | | | |
| Objective | Ensure a better user experience in overall of the application | | | | |
| Steps | 1. Open multiple pages of the application (homepage, profile, search, etc.).<br>2. Visually verify that UI elements (fonts, colors, layout) remain consistent. | | | | |
| Expected | All pages should show the same consistent design and layout for these elements. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC045 | Category | Non-functional | Severity | Minor |
|---|---|---|---|---|---|
| Requirement | The application must display clear and user-friendly error messages for common issues (e.g., invalid login, form submission errors). | | | | |
| Objective | Verify clear error messages are displayed to users | | | | |
| Steps | 1. Intentionally trigger errors by submitting invalid data (wrong password, incomplete forms).<br>2. Observe the error messages shown by the system. | | | | |
| Expected | Error messages are simple, clear, and provide guidance on how to correct the issue. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC046 | Category | Non-functi onal | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | Sensitive data (e.g., user passwords, company details) must be stored encrypted in the database. | | | | |
| Objective | Ensure that data stored in the database is encrypted to prevent unauthorized access. | | | | |
| Steps | 1. Register a new user and create a company profile containing sensitive data.<br>2. Access the database directly (with appropriate permissions) to inspect stored data.<br>3. Check that sensitive fields (e.g., passwords, email addresses) are not stored in plain text. | | | | |
| Expected | Sensitive data should be stored in an encrypted format and direct database inspection should not reveal plain text information. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC047 | Category | Functional | Severity | Critical |
|---|---|---|---|---|---|
| Requirement | Only admins can see the admin dashboard of the application. | | | | |
| Objective | Verify that non-authorized people cannot see the sensitive information and do not have management permissions. | | | | |
| Steps | 1. Log in to the application with a non-admin account.<br>2. Try to reach the admin dashboard. | | | | |
| Expected | The admin dashboard should not be reachable. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC048 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Only a project's companies' people can access project related documents. | | | | |
| Objective | Verify that other than the relevant people, others do not have access to the document management system of a project. | | | | |
| Steps | 1. Go to a project's page in which other companies are involved other than yours.<br>2. Attempt to reach the documents of that project. | | | | |
| Expected | The documents should not be reachable. | | | | |
| Date-Result | 02.05.2025 Failed, projects do not contain documents in the current implementation | | | | |

| Test ID | TC049 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | Companies should be able to import PDF files while creating projects. | | | | |
| Objective | Verify that the companies can add PDF files to the system in addition to filling out the text fields manually in project creation. | | | | |
| Steps | 1. Go to the create project page.<br>2. Upload a PDF file<br>3. Check the text fields in the page | | | | |
| Expected | The relevant text fields should be filled according to the PDF file. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

| Test ID | TC050 | Category | Functional | Severity | Major |
|---|---|---|---|---|---|
| Requirement | File uploads must enforce a maximum file size limit while project creation. | | | | |
| Objective | Validate that oversized files are not accepted. | | | | |
| Steps | 1. Go to create a project page.<br>2. Attempt to upload an oversized file. | | | | |
| Expected | Upload should not be successful and a relevant error message should be displayed. | | | | |
| Date-Result | 02.05.2025 Passed | | | | |

## 6. Maintenance Plan and Details

Our project has been designed with sustainability and long-term maintenance in mind. This section outlines our practical maintenance strategy to keep the platform secure, performant, and adaptable after deployment.

### 6.1. Regular Maintenance Tasks

Our database maintenance strategy involves weekly full backups complemented by daily differential backups to ensure comprehensive data protection with minimal risk of loss. To maintain optimal performance as data volume grows, we will conduct monthly index optimization sessions where we analyze query patterns and adjust indexing strategies accordingly. Additionally, we have established a quarterly schedule for data cleanup operations where non-essential historical data will be archived or purged based on defined retention policies.

System updates will follow a structured approach where security patches are prioritized and applied within one week of their release to minimize vulnerability windows. Framework and dependency updates will occur on a monthly cadence, but only after thorough testing in our staging environment to prevent disruption to production services. Our team will continuously monitor third-party libraries for security vulnerabilities using automated tools integrated with our development workflow, allowing us to respond quickly to emerging threats.

### 6.2. Team Responsibilities

Our maintenance approach distributes responsibilities across team members based on expertise areas while ensuring knowledge sharing. Code quality and architecture oversight will guide our development practices, with regular code reviews and architecture discussions to maintain system integrity. Bug fixes and feature updates will be prioritized based on user impact and system stability considerations. Deployment and database maintenance will follow established procedures with appropriate testing and validation steps. Our monitoring and alert response system will ensure timely detection and resolution of issues, with defined escalation paths for critical problems. UI components and usability enhancements will be continuously refined based on user feedback and interaction analytics.

# 7. Other Project Elements

## 7.1. Consideration of Various Factors in Engineering Design

### 7.1.1. Constraints

#### 7.1.1.1. Public Health

The Compedia platform features AI-powered semantic search and company information that could potentially include health-related businesses or services. To ensure responsible handling of health-related information, our platform must be designed with clear limitations and disclaimers. Any health-related company information displayed on Compedia should be clearly marked as informational only and not to be construed as medical advice. The semantic search should be constrained to avoid generating health recommendations or medical interpretations when users search for health-related businesses. Additionally, verification protocols for health industry companies will be more strict. All company profiles related to healthcare will include prominent disclaimers emphasizing that Compedia does not validate medical claims or endorse specific health treatments.

#### 7.1.1.2. Public Safety

Public safety considerations are paramount in the design of Compedia's platform. All user data and company information are stored securely with robust encryption methods and we made comprehensive access control systems to prevent unauthorized access to sensitive company data. Our AI-powered search system undergoes rigorous testing to prevent potential misuse, such as searching for illegal services or products. We implement strict verification procedures for company information to ensure the legitimacy of businesses listed on our platform, protecting users from potential scams or fraudulent activities. To further enhance safety, we will add content moderation tools to filter out inappropriate or harmful content in company descriptions and user interactions. In case of safety concerns, we provide easy-to-access reporting mechanisms for users to flag suspicious activities or potentially harmful content.

#### 7.1.1.3. Global Factors

Global factors significantly influence Compedia's design as a business discovery platform intended for international use. Our system accommodates regional business terminology to serve users across different geographical locations. We've designed the platform with awareness of various international business regulations and standards, ensuring compliance with global commerce laws, including GDPR compliance for European users and similar regulations for other regions. The search functionality incorporates understanding of region-specific business practices and terminology, allowing for more accurate matching regardless of cultural or geographical context. Additionally, the platform

supports international date formats, time zones, and measurement systems to provide a seamless experience for users worldwide. We've also considered global digital infrastructure disparities, optimizing our platform to function effectively even in regions with limited internet bandwidth while maintaining core functionality across diverse technological environments.

#### 7.1.1.4. Cultural Factors

Cultural factors play a crucial role in Compedia's design approach as we aim to create a platform that respects and adapts to diverse business cultures worldwide. Our semantic search algorithms are designed to understand and interpret culturally-specific business terminology and practices, ensuring relevant results regardless of cultural context. The platform's interface incorporates culturally neutral design elements while allowing for localization when appropriate. We've implemented rigorous content review processes to identify and prevent potentially offensive or culturally insensitive material in company descriptions and user interactions. Additionally, our verification system accommodates different business documentation standards across cultures, ensuring equitable verification opportunities for companies from various regions. The platform also supports multiple communication styles and business etiquette norms in how information is presented and how businesses can represent themselves, recognizing that business communication varies significantly across cultures.

#### 7.1.1.5. Social Factors

Social considerations have a high impact on Compedia's development, centered on building trust through verified profiles, ethical AI implementation, and transparent business interactions. Our verification system creates accountability and reliability in the platform's information, addressing the social need for trustworthy business data. The AI models powering our semantic search adhere to ethical guidelines that prioritize fairness and prevent manipulation. The platform encourages transparent communication between businesses through standardized company profiles and clear verification indicators. These social trust mechanisms are fundamental to our value proposition and significantly shape our feature development priorities.

#### 7.1.1.6. Environmental Factors

Environmental factors have been considered in Compedia's technical infrastructure and operational decisions. We've opted for cloud services that prioritize energy efficiency and use renewable energy sources where possible, minimizing the carbon footprint associated

with our server operations. The AI models powering our semantic search are optimized for computational efficiency, reducing unnecessary processing and associated energy consumption. We implement intelligent caching strategies to minimize redundant requests and processing, further reducing energy usage. Our development and deployment processes follow green software engineering practices, including efficient code that minimizes resource utilization. Additionally, we've designed the platform to encourage digital business interactions and virtual partnerships, potentially reducing the need for physical travel and associated environmental impacts.

#### 7.1.1.7. Economical Factors

Economic factors significantly influence our technological choices and development strategy for Compedia. Budget constraints have guided our selection of open-source technologies and cloud services that offer scalable pricing models aligned with our growth projections. Our system architecture is designed for scalability, allowing infrastructure expenses to grow proportionally with user adoption and preventing premature over-investment. Database technologies were selected based on performance-to-cost ratios, with particular attention to Milvus vector database for its efficient scaling capabilities. We've also carefully evaluated the cost-benefit ratio of implementing advanced AI capabilities like semantic search, balancing the improved user experience against infrastructure expenses. Our development approach emphasizes modular architecture and incremental feature deployment, allowing us to allocate resources strategically and demonstrate value before significant investment in any particular component. We've also considered the economic sustainability of our business model, ensuring that our verification processes and platform features can be maintained without prohibitive operational costs. Additionally, we've designed the platform to provide value to businesses of all sizes, including consideration for smaller companies with limited budgets for business development and partnership discovery. This approach not only aligns with our mission but also expands our potential user base and revenue opportunities.

| Factor | Impact on Design and Development | Effect Level |
|---|---|---|
| Public Health | Secure data handling is ensured to avoid misuse of sensitive information that could indirectly affect public health. | Medium |
| Public Safety | Robust encryption and access control are implemented to prevent unauthorized access to sensitive company data. | High |
| Public Welfare | Business matchmaking efficiency is enhanced, promoting economic opportunities for small and large companies alike. | High |
| Global Factors | International data privacy laws (e.g., GDPR) are followed and a multilingual user base for global accessibility is supported. | High |
| Cultural Factors | Avoids bias in recommendations by ensuring neutrality and respecting diverse business and cultural contexts. | Medium |
| Social Factors | Verified profiles, ethical AI use, and fostering transparency in business interactions build trust. | High |

| | | |
|---|---|---|
| Environmental Factors | By using energy-efficient cloud hosting and optimized AI model deployments, environmental impact is reduced. | Low |
| Economic Factors | Open-source tools and scalable architecture are employed which ensure affordability and sustainability. | High |

**Table X:** Summary of Factors Affecting Compedia

### 7.1.2. Standards

We use Git as a version control system. We use Trello to track the assignments that are given to each group member. Moreover, in every two weeks, we arrange meetings in order to discuss recent problems, provide solutions to problems and make assignments to group members. Our project adheres to the following standards to ensure clarity, consistency, and quality. UML 2.5.1 will be used for system modeling to represent use cases, workflows, and design architecture effectively. REST API Design Guidelines will ensure the interoperability and scalability in API development. These standards enhance the project's development and usability, ensuring adherence to industry practices.

## 7.2. Ethics and Professional Responsibilities

In developing Compedia, design and implementation choices were based on ethical considerations and professional responsibilities. Since the platform handles sensitive information such as company profiles, project data, and user credentials, we prioritized compliance with data protection regulations, including KVKK and GDPR. It is ensured that no personal or business-related data is shared with third parties without the explicit consent of the user. User data is encrypted at rest and in transit, and access control is strictly managed through a role-based JWT authentication system. Additionally, fairness and transparency are embedded in our platform’s functionality. The semantic

search engine is designed to prioritize relevance over commercial influence, giving all companies—regardless of size—an equal opportunity to be discovered. To ensure trust in the system, project reviews are restricted only to the client companies. Lastly, by prioritizing performance based visibility and verified information over marketing budgets, Compedia aims to promote fairness and transparency in the digital business landscape.

## 7.3. Teamwork Details

### 7.3.1. Contributing and functioning effectively on the team to establish goals, plan tasks, and meet objectives

From the early stages of the project, we ensured that every team member was actively involved in defining the system's objectives and breaking them down into manageable tasks. We brainstormed at our first meetings to establish our vision for Compedia and to get everyone on board with its main objectives, which included scalable architecture, verified company profiles, and semantic search. We used Trello to create a task board that reflected each sprint's deliverables, allowing everyone to track and plan work efficiently. We held weekly in-person and Zoom meetings to review our progress, establish short-term objectives, and redistribute work as necessary. All team members contributed to improving the product's scope, modifying our timeline, and completing our tasks, all of which helped us continually achieve our development milestones.

### 7.3.2. Helping creating a collaborative and inclusive environment

Our team placed strong emphasis on maintaining a respectful, transparent, and inclusive environment. We encouraged open discussions during meetings and welcomed different perspectives during design and implementation phases. Decision-making was consensus-driven, and feedback was regularly exchanged through code reviews and collaborative documents. Google Docs and WhatsApp were utilized as communication tools to keep everyone informed and to ease the participation process. Task assignments were flexible, allowing members to express preferences or swap responsibilities based on availability and personal interest. Regardless of role, all members were invited to contribute ideas and participate in technical or design decisions. This atmosphere leads to a high degree of mutual respect and accountability.

### 7.3.3. Taking lead role and sharing leadership on the team

Leadership in our team was distributed based on technical domain expertise rather than a single designated leader. Each member assumed a leadership role for one

or more key areas of the project. For example, Yaşar Tatlıcıoğlu led the overall architecture design, search implementation and Kafka integration, while Anıl Altuncu directed the implementation of AI and LLM-based features. Serhat Yılmaz managed frontend development and UI connectivity, Bartu Albayrak led the backend database architecture, and Ece Beyhan guided UI/UX design and was also responsible for integrating frontend components with backend services from the frontend side. These roles were not fixed as members collaborated closely across domains, offering support or stepping up in case of need. This dynamic leadership model helped prevent bottlenecks and accelerated problem-solving.

#### 7.3.4. Meeting objectives

Throughout the semester, our team consistently met the technical and functional goals we set in our requirement analysis phase. We implemented all core features of Compedia, including user registration, company creation, semantic search, project collaboration, and access control. Our deployment and Kafka-based service architecture were completed on schedule, and the frontend UI was fully integrated with backend services. While some features like document sharing are reserved for future versions, all major objectives were delivered as planned. This success was made possible by our structured planning, clear task division, and continuous team effort in resolving challenges.

### 7.4. New Knowledge Acquired and Applied

The development of Compedia provided us with the opportunity to acquire and apply a wide range of technical and project management skills. On the backend side, we gained practical experience in microservice architecture, asynchronous communication using Kafka, and relational database design with SQL Server and Entity Framework. Through the frontend implementation, we deepened our understanding of React, state management, component reuse, and REST API integration. One of the most enriching aspects of this project was the implementation of AI-powered features. We explored semantic search using GPT models, experimenting with prompt engineering and vector embeddings via Milvus. Furthermore, we learned how to implement secure authentication mechanisms using JWT and ASP.NET Identity, while ensuring role-based access control throughout the system. On the DevOps side, we also worked with cloud deployment environments. Beyond technical growth, we enhanced our collaboration, planning, and task-tracking skills through industry tools like Trello, GitHub, and Google Docs. These experiences not only improved our engineering expertise but also prepared us for real-world software development practices. In addition, we faced significant challenges

while trying to obtain real-world company data to populate the platform. This led us to arrange meetings with institutions such as KOSGEB and several Teknokents in Ankara, highlighting the difficulty of accessing structured, reliable data and reinforcing the value of the problem Compedia aims to solve.

## 8. Conclusion and Future Work

Compedia was developed to bridge the gap between businesses and their potential partners by offering an AI-powered, fair, and scalable discovery platform. Unlike traditional directories and search engines that prioritize large businesses with extensive marketing budgets, Compedia focuses on semantic relevance and verified company profiles. The system supports features such as intelligent search, company profile management, project collaboration, and secure user-role assignments. While the core objectives have been successfully implemented, several enhancements are planned for the near future. These include the introduction of a collaborative project workspace where companies can jointly manage their ongoing projects and track their status in more detail, as well as a document-sharing feature to support richer information exchange and scalability of the platform. We also plan to fine-tune the semantic search engine using custom-trained AI models to further improve result relevance. With these improvements, we aim to solidify Compedia's role as a modern, trustworthy, and intelligent B2B connection platform.

## 9. Glossary

**AI-powered Search**
AI-powered search engines are designed to make searching smarter and more efficient, leveraging the power of artificial intelligence to deliver highly relevant results. [2]

**Azure**
A cloud computing service created by Microsoft that provides infrastructure, platform, and software solutions to support applications and services, including hosting, database management, and AI integration.

**Compedia**
A proposed platform aimed at creating comprehensive, structured, and searchable profiles of companies using advanced NLP models and semantic search techniques to match user needs with businesses.

**Docker**
Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

**Embeddings**
Embeddings are numerical representations of real-world objects that machine learning (ML) and artificial intelligence (AI) systems use to understand complex knowledge domains like humans do. [3]

**Kafka**
Kafka is primarily used to build real-time streaming data pipelines and applications that adapt to the data streams. It combines messaging, storage, and stream processing to allow storage and analysis of both historical and real-time data. [4]

**LLM (Large Language Model)**
Large language models (LLMs) are machine learning models that can comprehend and generate human language text. They work by analyzing massive data sets of language. [5]

**MicrosoftSQL**
A relational database management system by Microsoft, used to store structured data such as basic company details in Compedia.

**Milvus**
An open-source vector database optimized for processing large-scale vector data and supporting semantic search operations efficiently.

**Natural Language Processing (NLP)**
Natural language processing (NLP) is a subfield of computer science and artificial intelligence (AI) that uses machine learning to enable computers to understand and communicate with human language. [6]

**React**
A JavaScript library for building user interfaces, particularly for single-page applications, ensuring a responsive and interactive user experience.

**Semantic Search**
Semantic search is a search engine technology that interprets the meaning of words and phrases. [7]

**State-of-the-Art (SOTA)**
The state of the art (SOTA or SotA, sometimes cutting edge, leading edge, or bleeding edge) refers to the highest level of general development, as of a device, technique, or scientific field achieved at a particular time. [8]

**Trello**
A task and project management tool used for team collaboration and organizing workflows visually via boards, lists, and cards.

**Vector Database**
A vector database, vector store or vector search engine is a database that can store vectors (fixed-length lists of numbers) along with other data items. [9]

**Zero-shot Classification**

Zero-shot classification models are large, pre-trained models that can classify images without being trained on a particular use case. [10]

## 10. References


[1] "Milvus: The High-Performance Vector Database Built for Scale." *Milvus*, https://milvus.io. Accessed 21 Nov. 2024.

[2] Sitecore. "What Is AI Search?" *Sitecore*, www.sitecore.com/explore/topics/artificial-intelligence/what-is-ai-search. Accessed 21 Nov. 2024.

[3] "Embeddings in Machine Learning." *Amazon Web Services (AWS)*, https://aws.amazon.com/what-is/embeddings-in-machine-learning/. Accessed 21 Nov. 2024.

[4] "What Is Apache Kafka?" Amazon Web Services (AWS), https://aws.amazon.com/what-is/apache-kafka/. Accessed 21 Nov. 2024.

[5] "What Is a Large Language Model?" *Cloudflare*, www.cloudflare.com/learning/ai/what-is-large-language-model/. Accessed 21 Nov. 2024.

[6] IBM. "Natural Language Processing (NLP) Solutions." *IBM*, www.ibm.com/natural-language-processing. Accessed 21 Nov. 2024.

[7] Elastic. "Semantic Search." *Elastic Documentation*, Elastic, https://www.elastic.co/guide/en/serverless/current/elasticsearch-reference-semantic-search.html. Accessed 21 Nov. 2024.

[8] Wikipedia contributors. "State of the Art." *Wikipedia*, 20 Nov. 2024, https://en.wikipedia.org/wiki/State_of_the_art. Accessed 21 Nov. 2024.

[9] Wikipedia contributors. "Vector Database." *Wikipedia*, 20 Nov. 2024, https://en.wikipedia.org/wiki/Vector_database. Accessed 21 Nov. 2024.

[10] Gallagher, James. "What Is Zero-Shot Classification?" *Roboflow Blog*, 15 Nov. 2023, https://blog.roboflow.com/what-is-zero-shot-classification/%20model%20developed%20by%20OpenAI). Accessed 21 Nov. 2024.