High-Level Design (HLD) & Low-Level Design (LLD)

# 1. Project Overview

The project aims to develop a Multi-Agent Architecture that includes three primary agents:  
  
1. Agent 1: Browser Agent for Company Research  
2. Agent 2: Market Standards & Use Case Generation  
3. Agent 3: Resource Asset Collection  
  
The system is designed to analyze company data, identify industry trends, and propose relevant AI-driven use cases for different industries. The architecture uses web scraping, NLP, and external resources like Kaggle datasets.

# 2. High-Level Design (HLD)

## 2.1 System Architecture

The architecture is modular and consists of the following components:  
  
1. User Interface (UI):  
 - Streamlit-based front-end interface for interaction with users.  
 - Provides inputs for company names and outputs results for analysis.  
  
2. Browser Agent (Agent 1):  
 - Crawls the web (using requests and BeautifulSoup) to gather company information.  
 - Analyzes and classifies the industry of the company based on its content using predefined industry keywords.  
 - Stores the company’s information in Qdrant for further use.  
  
3. Market Standards & Use Case Generation (Agent 2):  
 - Analyzes the industry and generates AI-based use cases using predefined methods.  
  
4. Resource Asset Collection (Agent 3):  
 - Collects datasets related to the industry from Kaggle using Kaggle API.  
 - Saves the resources in markdown format for the user to download.

## 2.2 Data Flow Diagram (DFD)

1. Input:  
 - The user provides a company name in the UI.  
  
2. Browser Agent:  
 - Scrapes the web for company data (e.g., using Google search).  
 - Extracts company-related text and classifies the industry.  
  
3. Market Standards & Use Case Generation:  
 - Based on the identified industry, the agent analyzes current trends and proposes use cases.  
  
4. Resource Asset Collection:  
 - Collects Kaggle datasets relevant to the industry and stores them for the user.  
  
5. Output:  
 - The final proposal, including industry trends, use cases, and Kaggle resources, is displayed to the user.

## 2.3 Technologies Used

- Frontend: Streamlit (for UI).  
- Backend: Python (for implementing agents).  
- Requests, BeautifulSoup (for web scraping).  
- SentenceTransformers (for semantic text processing).  
- Qdrant (for storing and querying company data).  
- Kaggle API (for collecting resources).  
  
Libraries:  
- requests: For web requests.  
- beautifulsoup4: For scraping and parsing HTML.  
- qdrant-client: For interfacing with Qdrant.  
- sentence-transformers: For semantic text embeddings.  
- kaggle: For fetching datasets.

# 3. Low-Level Design (LLD)

## 3.1 Components and Modules

1. Browser Agent (Agent 1):  
 - Inputs: Company name (via UI or direct URL).  
 - Process:  
 1. Constructs the search URL using the company name.  
 2. Performs a search via Google (or directly visits a given URL).  
 3. Scrapes the HTML content and extracts relevant text (paragraphs, headers).  
 4. Classifies the industry by checking for keywords in the scraped content.  
 - Outputs: Industry classification, relevant content from the company website.  
 - Data storage: Saves company information in Qdrant for future reference.  
  
2. Market Standards & Use Case Generation (Agent 2):  
 - Inputs: Industry classification.  
 - Process:  
 1. Based on the identified industry, it queries predefined use case templates.  
 2. Analyzes the latest industry trends (static data for now).  
 - Outputs: Proposed use cases and trends.  
  
3. Resource Asset Collection (Agent 3):  
 - Inputs: Industry classification.  
 - Process:  
 1. Fetches Kaggle datasets related to the industry using the Kaggle API.  
 2. Creates a markdown file containing resource links.  
 - Outputs: A downloadable markdown file containing Kaggle dataset links.

## 3.2 Data Structures

Industry Keywords (Browser Agent): A dictionary where the key is the industry name, and the value is a list of keywords associated with that industry.  
  
Example:  
{  
 "Automotive": ["vehicle", "car", "automotive", "engine", "transport"],  
 "Manufacturing": ["factory", "production", "machinery", "assembly"],  
 ...  
}

## 3.3 Sequence Diagrams

1. Sequence Diagram for Company Research (Browser Agent):  
 - User enters the company name in the Streamlit UI.  
 - The system constructs a search URL.  
 - The system sends a request to the URL.  
 - The system scrapes the page and extracts the content.  
 - The system classifies the industry and stores the data in Qdrant.  
 - Output: Company name, industry, and extracted content are displayed in the UI.  
  
2. Sequence Diagram for Use Case Generation (Agent 2):  
 - The system queries the industry from the Browser Agent.  
 - The system generates industry trends and use cases.  
 - Output: Proposed use cases and trends are displayed.  
  
3. Sequence Diagram for Resource Collection (Agent 3):  
 - The system queries the industry from the Browser Agent.  
 - The system fetches Kaggle datasets related to the industry.  
 - Output: A downloadable markdown file containing resource links is provided to the user.

## 3.4 Detailed Class Diagrams

1. BrowserAgent Class:  
 - Attributes:  
 - industry\_keywords: A dictionary mapping industries to keywords.  
 - Methods:  
 - search\_for\_company\_info(): Scrapes web pages based on company names.  
 - extract\_relevant\_text(): Extracts text content from HTML.  
 - classify\_industry(): Classifies the industry based on the keywords found.  
  
2. MarketStandardsUseCaseAgent Class:  
 - Methods:  
 - analyze\_industry\_trends(): Analyzes current trends for the given industry.  
 - propose\_use\_cases(): Proposes use cases based on industry classification.  
  
3. ResourceAssetCollectionAgent Class:  
 - Methods:  
 - collect\_kaggle\_resources(): Collects datasets related to the industry using the Kaggle API.