1. System Architecture:

in this section we will discuss system design where system design involves defining the overall structure and architecture of the software, ensuring that all components work seamlessly together. This phase establishes the blueprint for efficient data flow, interaction, and integration within the system.

The first constraints to take into consideration when building an architecture for a software system are **the non-functional requirements (quality attributes),** so first we will start with the non-functional requirements of our system

1. **Extensibility:** the system should demonstrate high extensibility to upgrade the AI components in the with minimal impact on the existing codebase or system integrity. (Average of 10% changes when adding or upgrading features).
2. **Security:** the system should be able to protect its all-sensitives users' data successfully, this includes using authentication, authorization, strong password, and good architecture.
3. **Scalability**: The system analyzes a total of 10,000 reviews and their associated requirements per hour across all projects, with each project limited to 100 requirements.

4. **Performance**: the system should be able to give the result of a requested operation in an average of 3 minutes except for the analysis operation of the project reviews and requirements it will take 10,000 review per hour.

System decomposition - Component diagram:

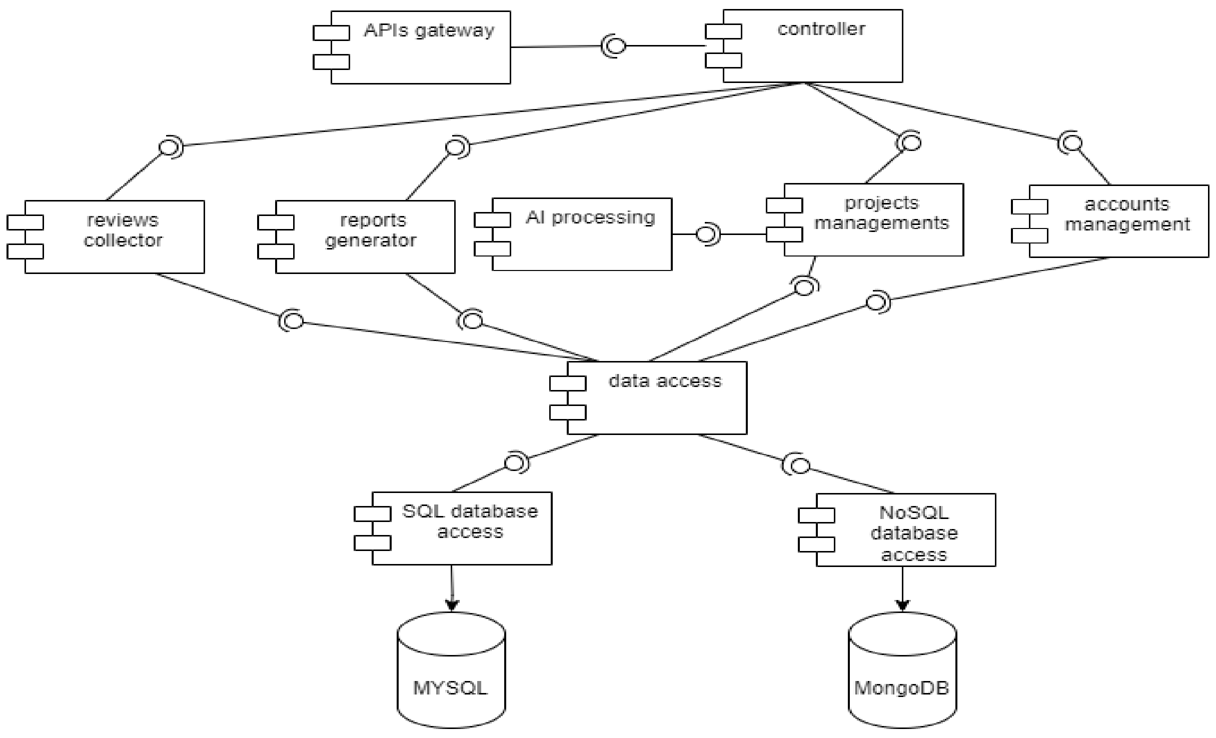

Figure 1 component diagram

Components functionalities:

1. APIs gateway (API-PL-01): this component is responsible for routing any external API call to its responsible controller.

2. Controllers (CN-PL-02): this component serves as the APIs views, and it is responsible for handling the APIs requests with the help of the services.
3. Reviews collector (RC-BLL-07): responsible for gathering the reviews from different sources, we added this component to implement the extensibility of the system we can add any new source and change it without affecting other components.
4. AI processing (AIP-BLL-05): responsible for analyzing the reviews and requirements for a project using AI components and algorithms like (NLP – sentiment analysis, neural networks – classification).
5. Reports generator (RG-BLL-06): this component responsible for charts data processing, making the data resulted from the AI algorithms suitable to be transported to the frontend and display in the charts.
6. Project management (PM-BLL-03): this component is responsible for the functionalities of managing project aspects reviews, requirements, and settings.
7. Accounts management (AM-BLL-04): responsible for functionalities of managing the accounts of the system and making reports for the system admin about the usage of the system.

8. Data access (DA-DAL-08): responsible for the communication with the two different types of databases and other components of the

system (here we can use repository design pattern over the Django ORM for the SQL database, and for the NoSQL database).

9. SQL database access (SDA-DAL-09): responsible for the communication with the SQL database of the system.
10. NoSQL database access (NSDA-DAL-10): responsible for the communication with the NoSQL database of the system.

System Architecture:

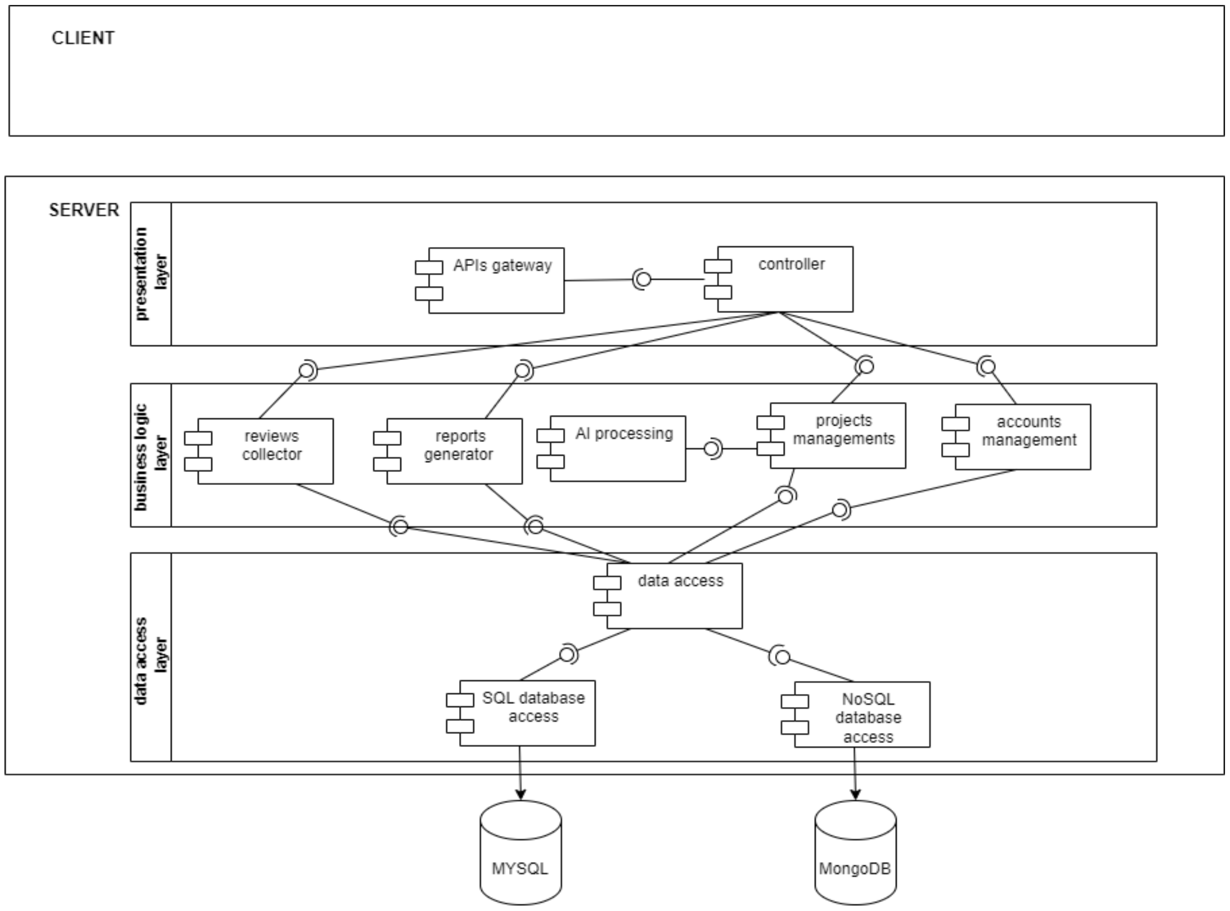

Figure 2 system architecture

- We used layered architecture with client-server architecture.

- The client side is the front-end application building with the React framework.
- The server side is the backend application building with Django framework, and the server side will be divided into multi layers for more security and less complexity of the system.
- The layered of the server side:

  Presentation layer: serves as a presentation for the backend APIs.

  Business logic layer: contain the main logic of the system.

Data access layer: responsible for accessing and communicating with the two databases we have.