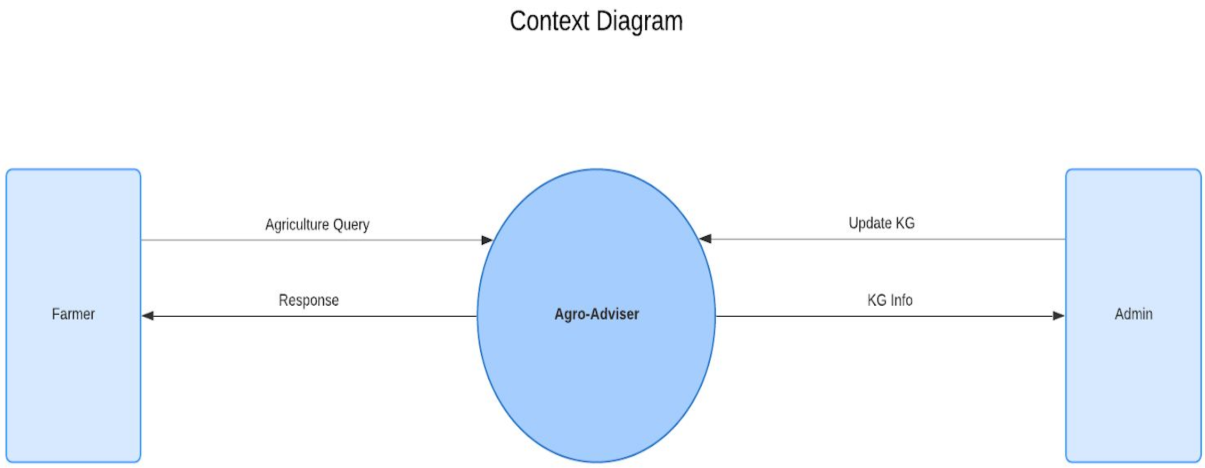
Assignment -2 Software Engineering CS305

Agro-Advisor

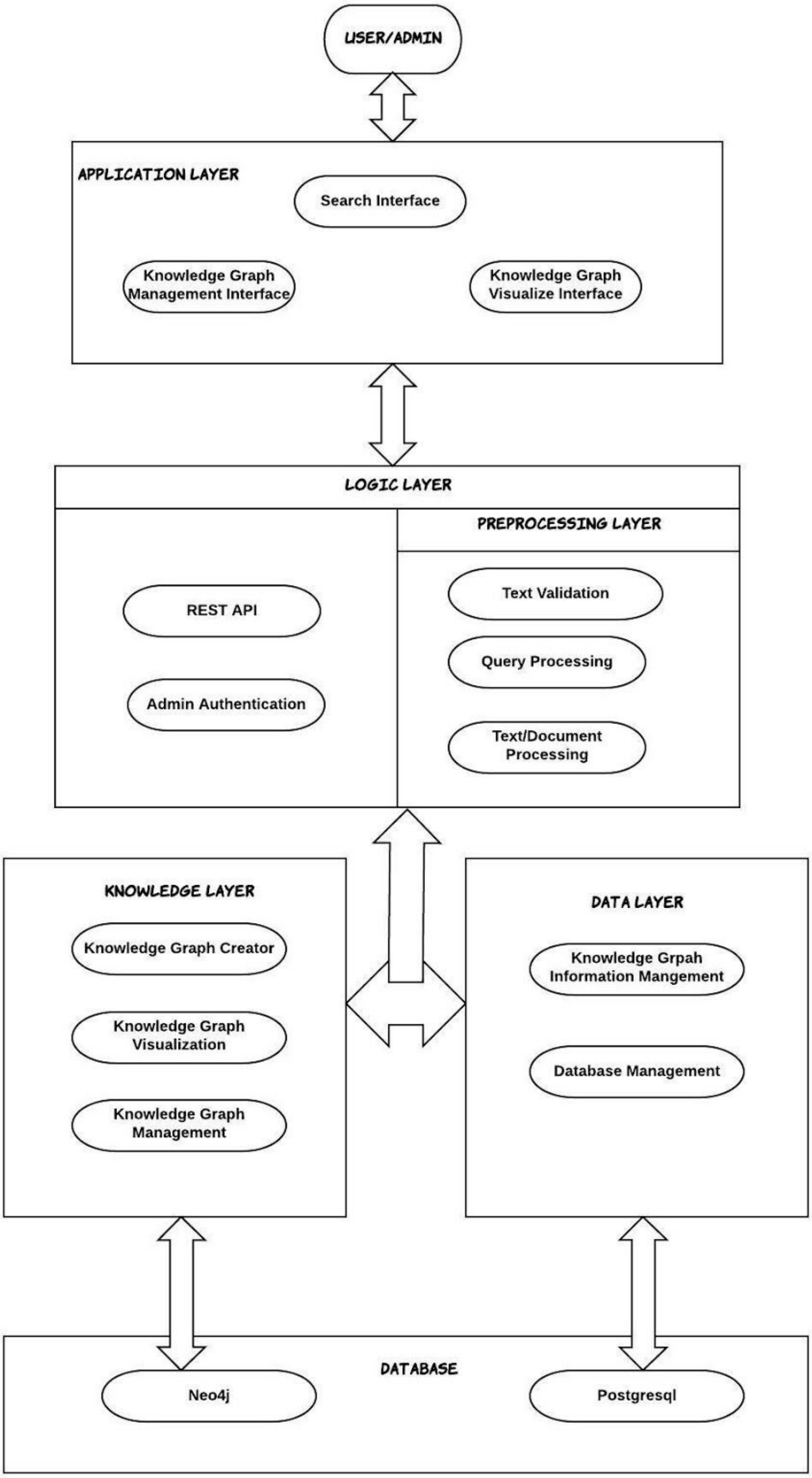
Prepared by

Shreekanth A *2017csb1110*
Nilansh Rajput *2017csb1092*
Vineet *2016csb1063*

1. Context Diagram



2. Logical structure diagram



2.1. **Application layer:** It consists only of a search bar where any user like a farmer can enter any query. There will be a different website endpoint where the admin logins and manages the knowledge graphs. In the management interface, there will be a list of all the stored knowledge graphs which the admin can edit or delete. There will be an option for the creation of the knowledge graph where the admin can enter a corpus or document and details of the new KG.

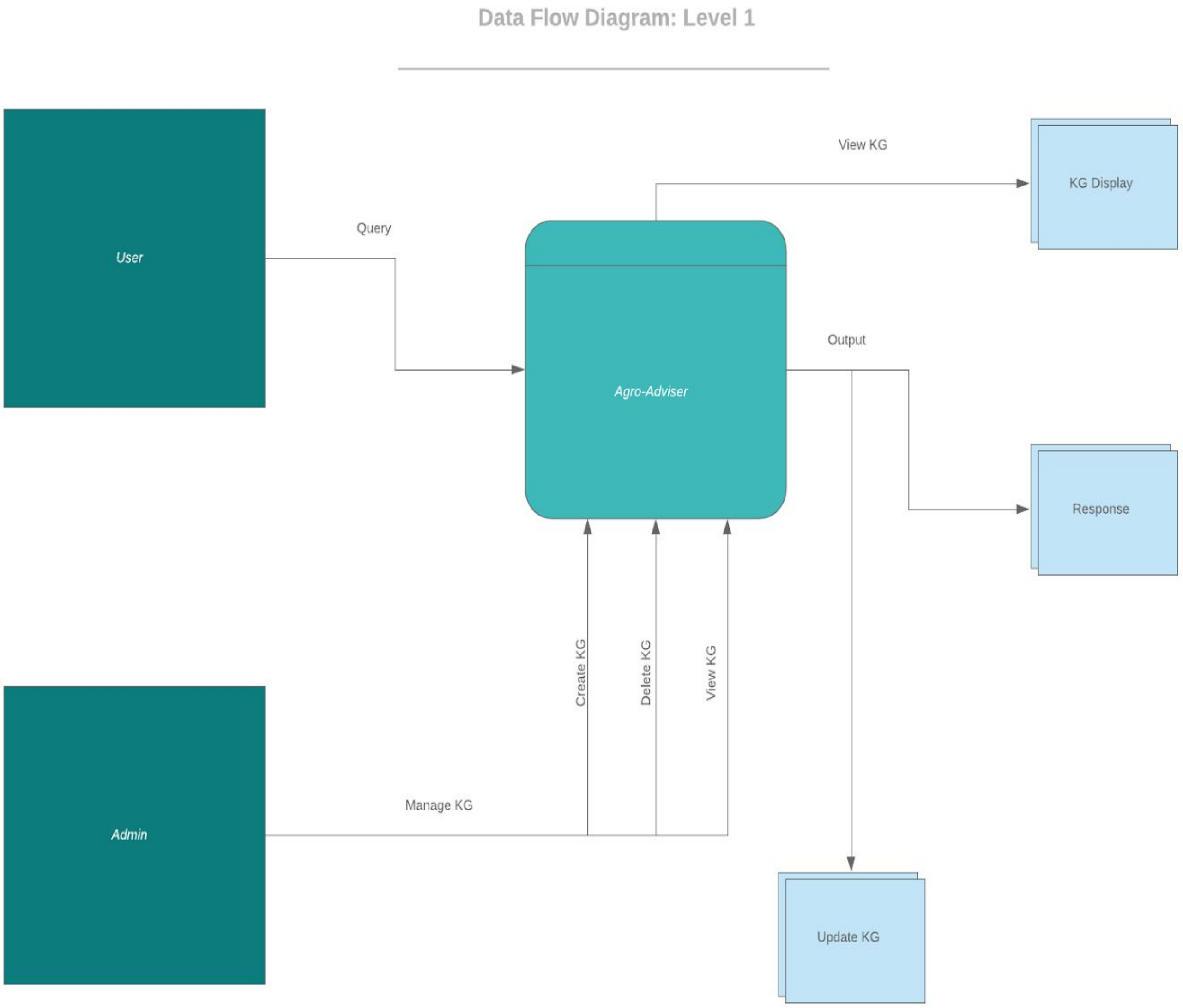
2.2. **Logic Layer:** It consists of input query preprocessing and validation. For the management interface, it consists of admin authentication and the text/document preprocessing and parsing given input to create the knowledge graph.

2.3. **Knowledge Layer:** This layer takes the preprocessed text from the logic layer, extracts the quality text, creates the KG and stores it in the Neo4j. It also takes the preprocessed query, retrieves the appropriate graph from the Neo4j and generates the response.

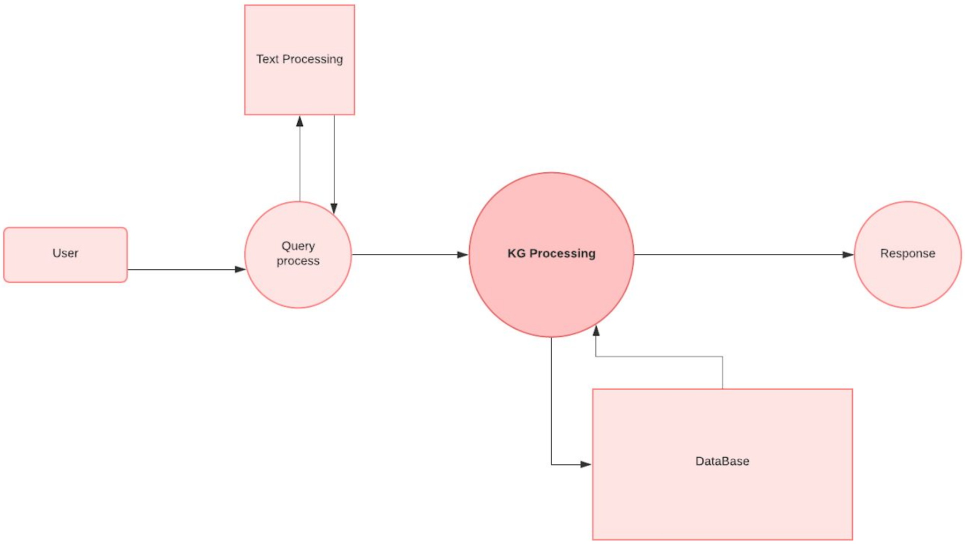
2.4. **Data Layer:** This layer stores the admin credentials and the details of the Knowledge Graph created by the admin.

3. Data Flow Diagram

3.1 DFD Level-1

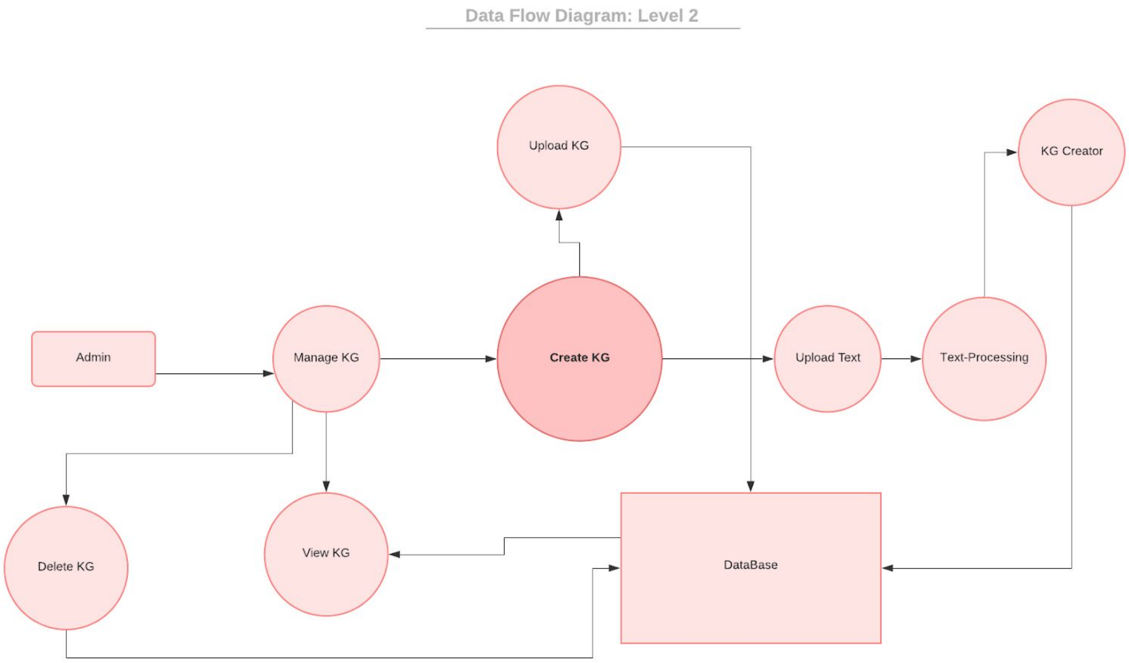


3.2 DFD Level-2



3.2.1 DFD : *For user querying over system:*

1. *The text in query is preprocessed for KG, and then according to intent of query(disease related or market etc), specific KG is chosen for further processing.*
2. *Before querying over the chosen KG, we look at a cache database which hold most common query with their response.*
3. *If query is not already present in cache then, we use KG and outputs the appropriate response.*



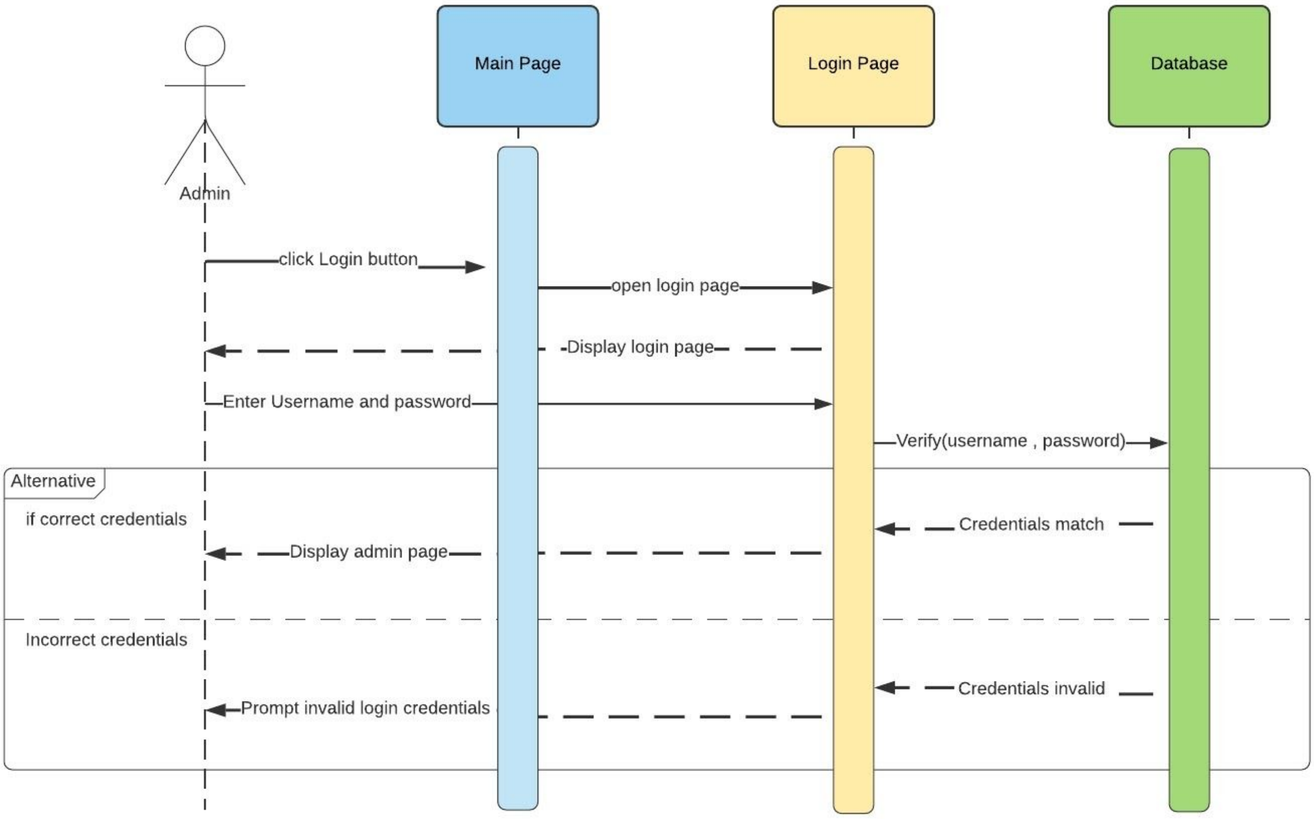
3.2.2 : *Admin Managing the KG:*

1. *Admin can view, delete or create a KG.*
2. *For creating a KG he can either upload KG dile directly or upload a text file which is then pre processed for creating a KG and after creating KG Database is updated accordingly.*
3. *For viewing Database stores images of KG which is shown back to Admin, after Delete command the KG and it's image is removed from Database.*

4. UML Diagrams for Dynamics of Static Structure

4.1. Sequence Diagrams

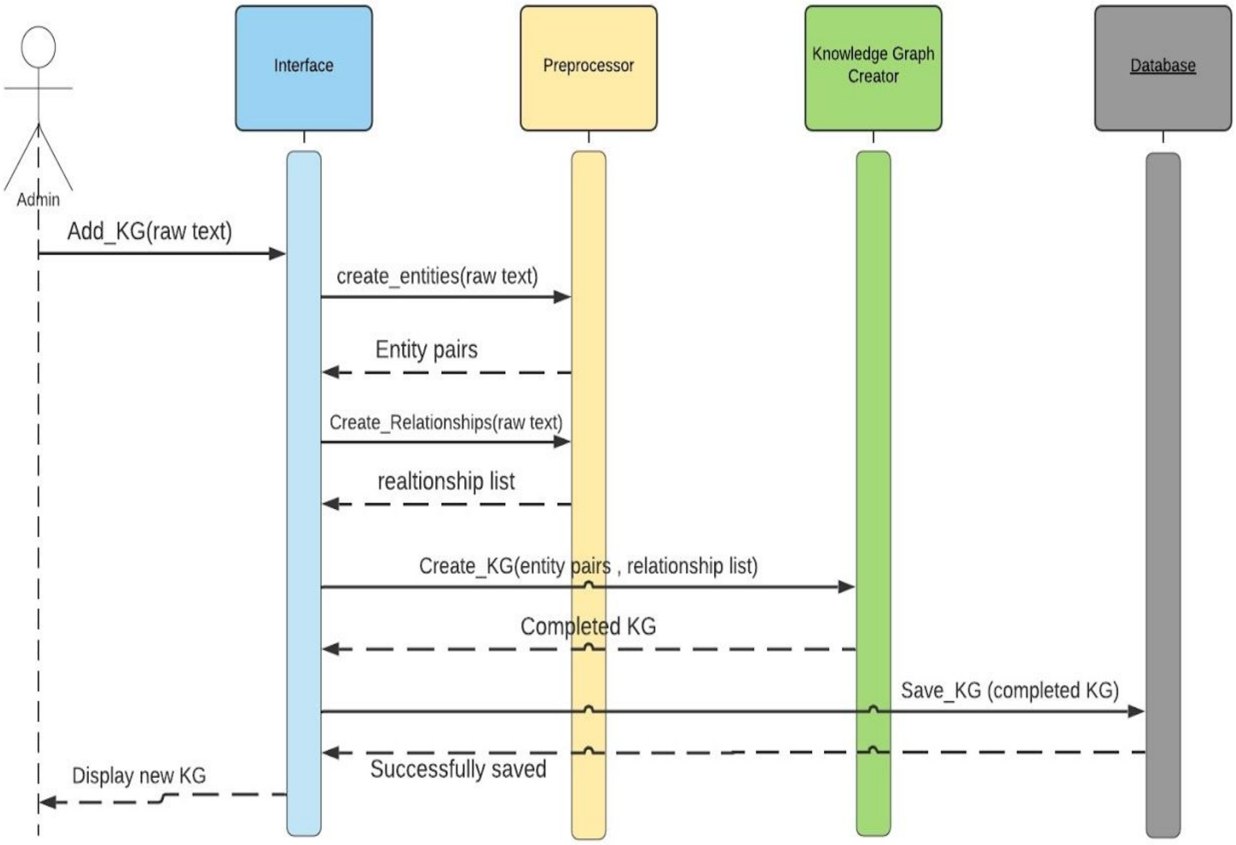
4.1.1. Admin login



Flow of the use case :

- Admin clicks on the login button and directed to the Login page.
- Admin enters Username and password
- Login page (session manager) verifies the username-password pair by doing a look up in the database
- Two cases :
    - If the credentials (username and password) match, then the admin page is displayed
    - Else if the credentials don't match and invalid credentials prompt is shown

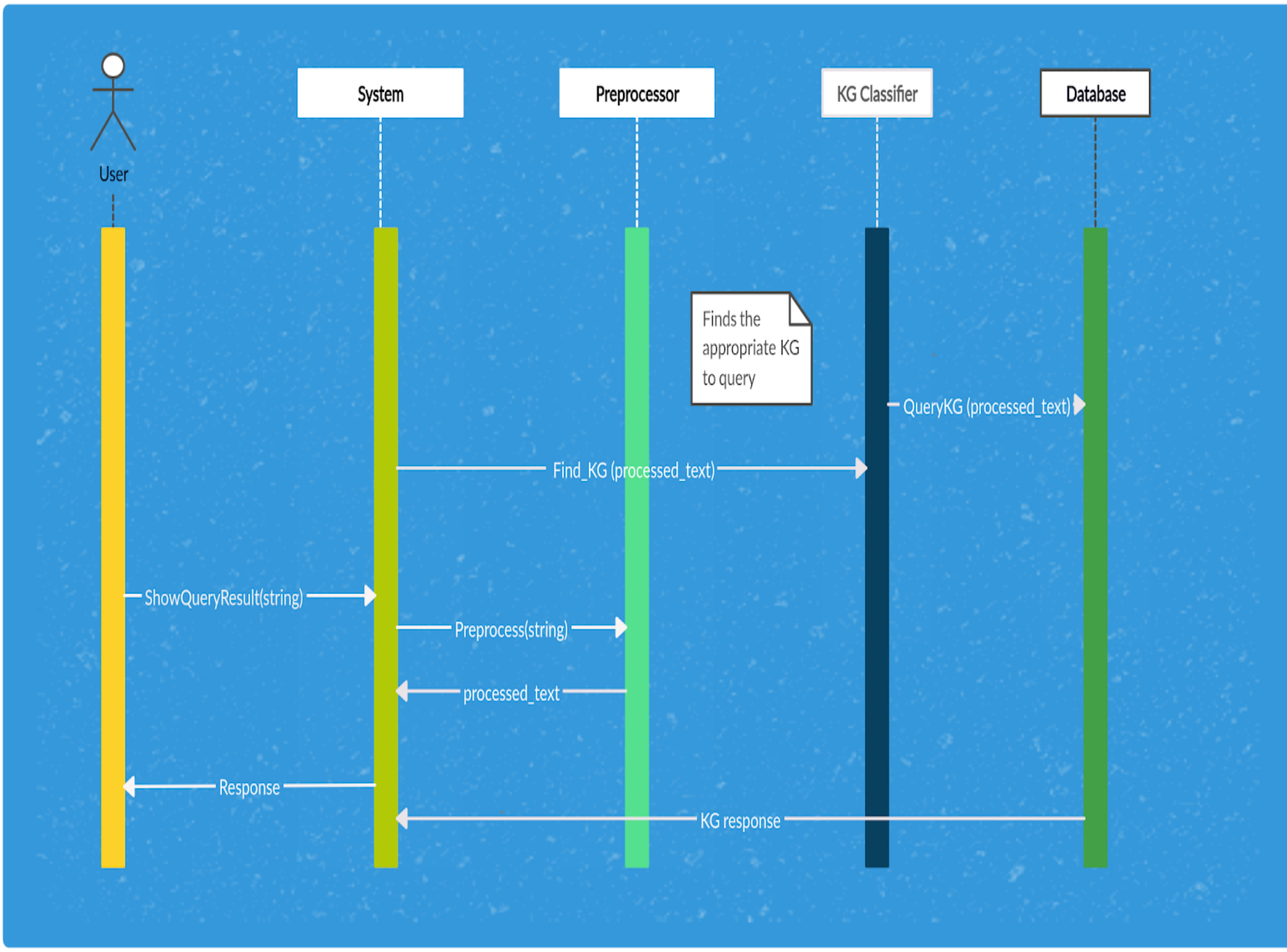
4.1.2. Admin adding a knowledge graph (clicks on add KG button)



Flow of the use case :

1. Admin enters raw text and clicks on add KG button, function Add_KG is called , the raw string is given as input
2. The Interface then calls the preprocessor using create_entities() function, which creates entity pairs, entity pairs are the subject and object in a sentence. They may or may not be compound words, they are found using the tags provided by tokenizer in the NLP library.
3. The interface makes a call to the preprocessor to get a list of relationships between the entity pairs using the Create_relationships() function.
4. The entity pairs and relationship list is passed to the Knowledge Graph creator to create a KG, using the entities as nodes and the relationships as edges.
5. The interface finally saves the KG to the graph database and the information(like name of KG, creation date, etc.) about the KG in the relational database.
6. The success message is displayed after the successful construction and save.

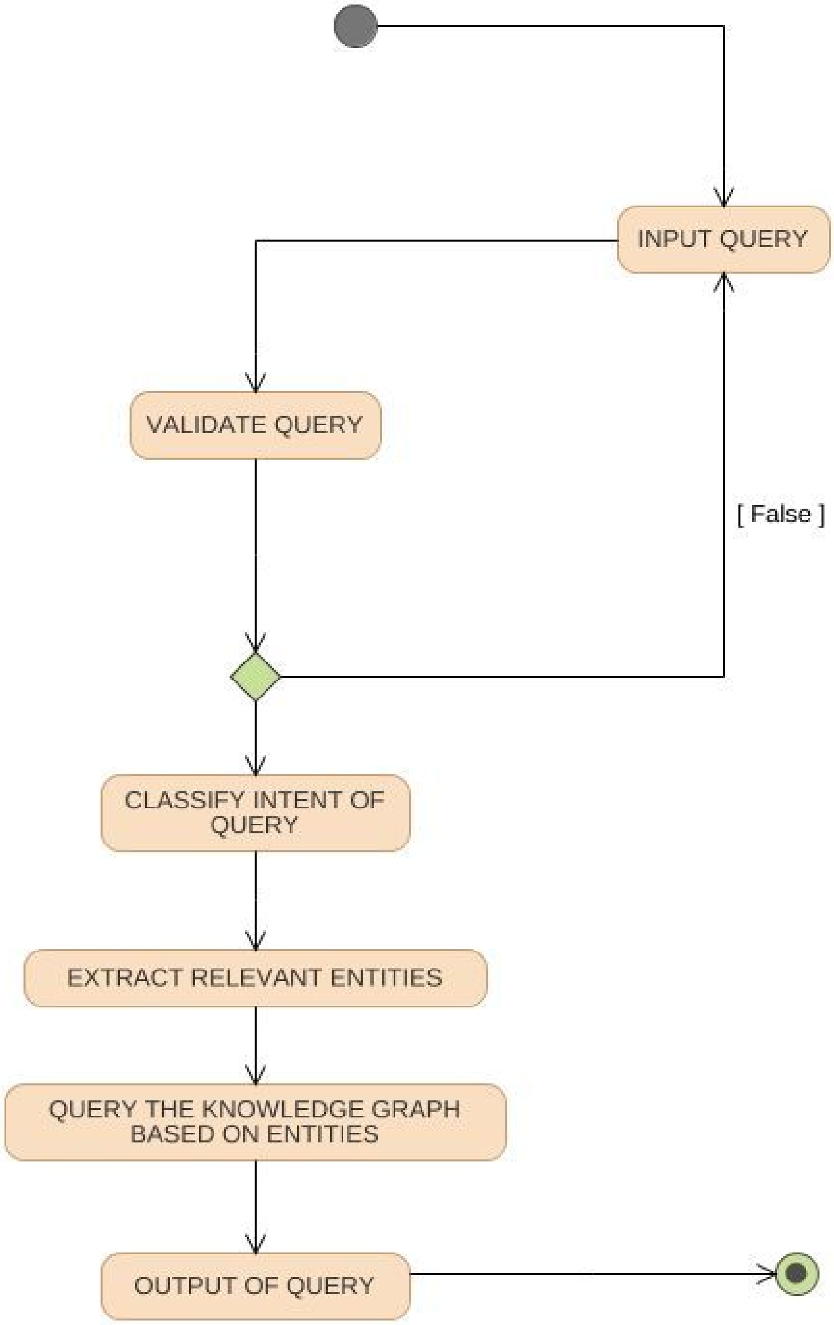
4.1.3. User querying the agriculture web app in search bar



Activity Diagrams

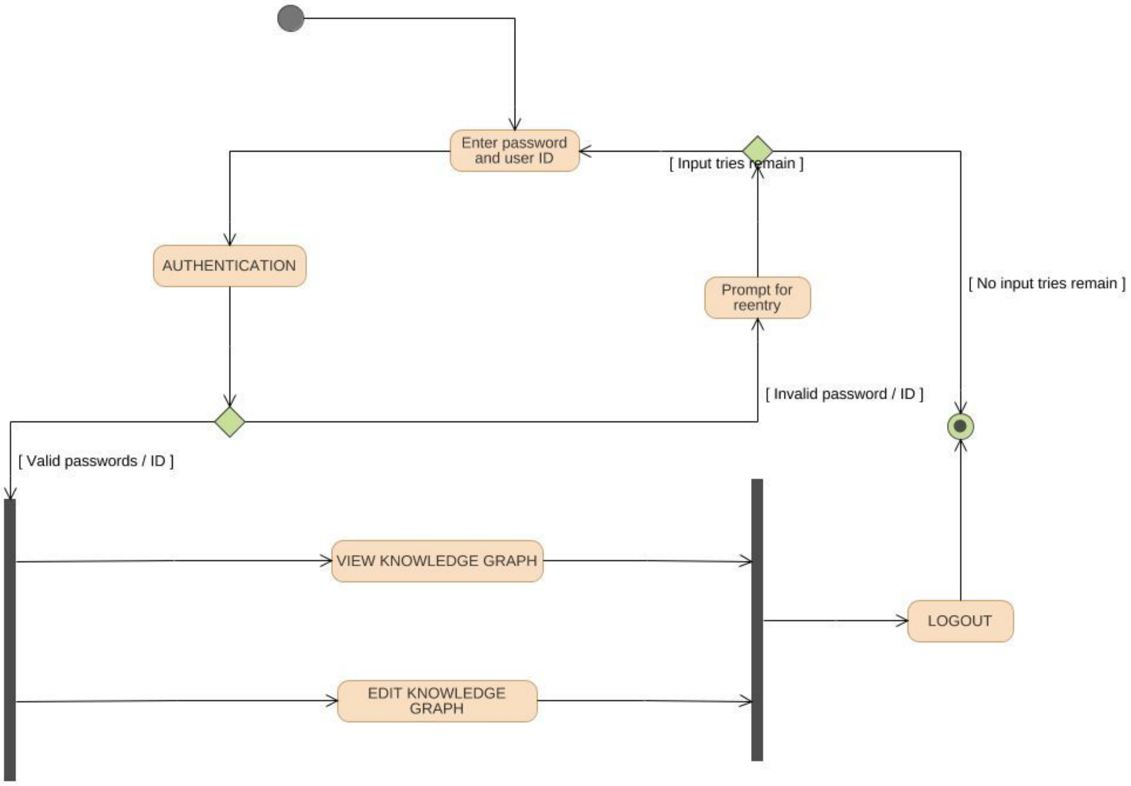
1.4.4 Activity Diagram - 1

<u>Use case</u> : user querying agricultural web app

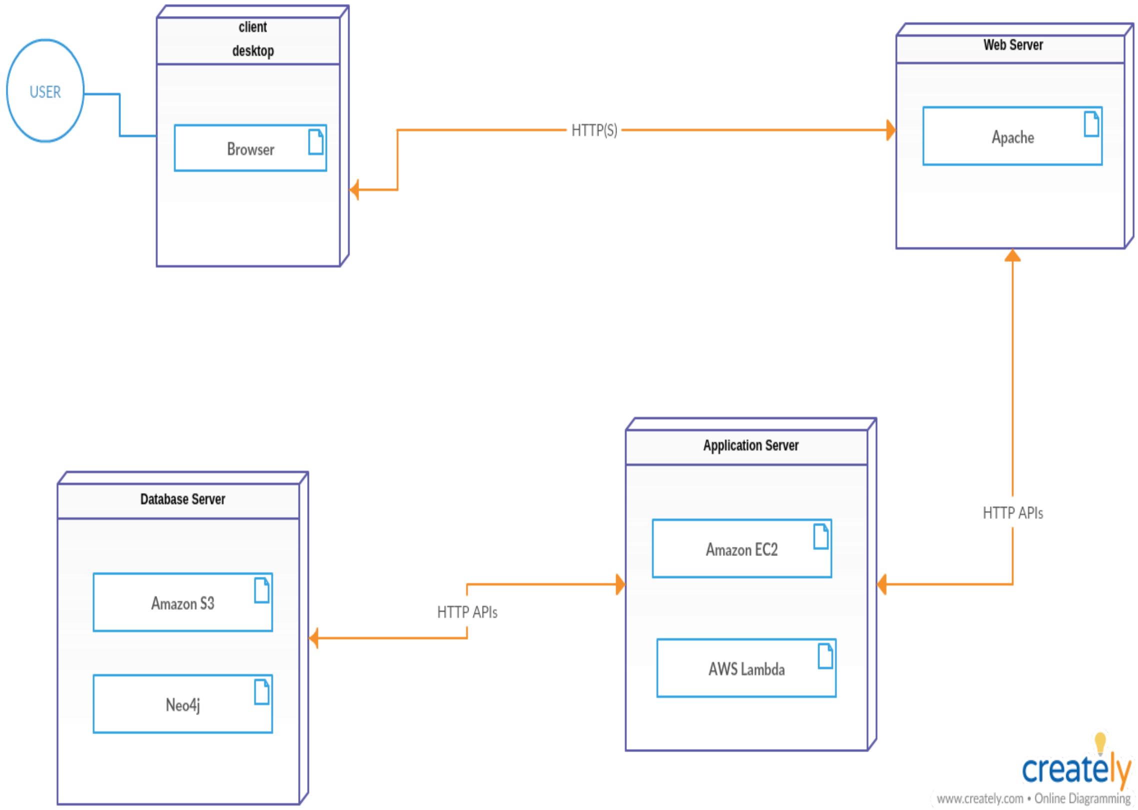


1.4.5 Activity Diagram - 2

Use Case : Admin Login



5. Deployment Diagram



6. Design Decisions

Our complete system is deployed on Amazon Cloud and every process is using different AWS services provided by Amazon cloud. AWS is chosen over other options, because of the following reasons :

- *AWS offers an easy deployment process for an app.*
- *Securely store all your files on the cloud so you can access them from anywhere.*
- *We can use the AWS Management Console or well-documented web services APIs to access AWS's application hosting platform.*
- *AWS enables you to select the operating system, programming language, web application platform, database, and other services you need.*
- *Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.*
- *AWS utilizes an end-to-end approach to secure and harden our infrastructure, including physical, operational, and software measures.*

These are the advantage of graph database:-

- *Neo4j provides easy representation and the stored data can be easily visualized in the graphs.*
- *Fast Execution and Easy retrieval which provides less response generation time for the input query.*