SoBigData Infrastructure - Integrated Hackathon, Training and Event Management System

Submitted to:

Prof. Vittorio Cortellessa

Department of Computer Science and Engineering, and Mathematics

University of L'Aquila (Italy)

Submitted by:

Hiba Bouhlal

Ichchha Moktan

Vidya Dhopate

Table of contents

List of figures

1. Introduction

1.1 Purpose of the Project:

In line with SoBigData++'s objective to foster a vibrant multidisciplinary community for Social Mining and Big Data Analytics, we are proposing a system that will support a collaborative environment for attracting new users and experiments in SoBigData from an industrial angle. We leverage the functionality of SoBigData and imagine how companies can leverage it. For 2024, it is SoBigData's aim to incorporate innovation through hackathons on its platform (more information is available on the same in the References section of this report i.e. 1.4.2). The companies can organize hackathons and upload training sessions for accelerating innovations connecting a broad user base. Furthermore, whenever SoBigData++ itself organizes events with its consortium partners, industrial and institutional stakeholders, companies can act as sponsors to these events to increase their visibility and branding while users can register for these events by being incentivized to network with companies at such events. Importantly, this system utilizes SoBigData++'s infrastructure such as datasets and notebooks.

1.2 Assumptions of the Project:

1. SoBigData++ has functionalities to share datasets through its SoBigData++ e-infrastructure, we assume that these datasets can be associated with other data components of the SoBigData++ platform such as Notebooks and Rules. We also assume SoBigData++ provides the mechanism for homogenizing the different data entries received by from different users, such as notebooks or rules of the hackathon
2. We assume that certain functionalities of the SoBigData++ catalog, such as available trainings, can be modified by the companies.
3. SoBigData++ autonomously creates events and segregates them into various labels and types. We assume that the Company and the User dashboards can set their preferences for certain types of events either on sign up or later on. Thus, we build the 'Alert Manager' system modeling only the components where companies are able to sponsor events and users and companies both get alerts about new events, since the preferences for event types have already been set and are outside the scope of our project.
4. SoBigData++ also provides a notebook environment feature functionality already. Our project builds on top of this aspect where the notebooks provided by the SoBigData++ and assumes that this notebook will be modified by users to participate in a hackathon.
5. It is also presumed that necessary data privacy and security measures to protect sensitive information have been implemented by the SoBigData++ itself. Similarly, the system will operate in compliance with all relevant laws and regulations, including data protection and intellectual property laws.
6. We also assume that this system will use the existing database storage infrastructure and cloud storage resources that SoBigData++ already has.

7. We assume that there will be a differentiated sign up, login, and authentication mechanism for companies and users.

1.3 Scope of the Project:

The scope encompasses the creation of UML diagrams to define system functionalities, interactions, and structure, covering use cases for both company and user interactions, as well as the underlying infrastructure of web services and databases.

1.4 Definitions, Acronyms, and Abbreviations:

- SoBigData++: A community for Social Mining and Big Data Analytics
- **UML: Unified Modeling Language**
- JAR: Java Archive
- JDBC: Java Database Connectivity

1.5 References:

1. http://www.sobigdata.eu/int_communication
2. https://plusplus.sobigdata.eu/objectives/
3. https://www.lucidchart.com/pages/uml-deployment-diagram
4. https://www.lucidchart.com/pages/uml-component-diagram

1.6 Overview of the Report:

This report provides detailed documentation of the system's design through UML diagrams, including use case, sequence, component, deployment, and profile diagrams, along with explanations and rationales behind the design choices.

2. System Overview

2.1 Business Context:

The system is to be developed for the SoBigData++ initiative, aiming to facilitate the growth of a multidisciplinary data analytics community by providing tools for organizing hackathons, training, and events.

2.2 System Objectives:

The major objectives of the system are:

- To encourage innovation and contribution to environmental data challenges within the SoBigData++ community.
- To utilize the system as a platform/tool for talent discovery and recruitment.

2.3 High-Level System Description:

The system is composed of a web-based platform that allows users and companies to organize and participate in hackathons, access training modules, and manage events, leveraging SoBigData++'s datasets and analytical tools.

3. Methodology

3.1 Overview of Approach:

The approach involves systematic development of UML diagrams to represent all aspects of the system.

3.2 Tools Used:

MagicDraw has been used for its robust features in UML diagramming and model validation, ensuring a high standard of design precision.

3.3 Limitations and Assumptions:

The design assumes a high level of technical proficiency among users and that access to SoBigData++ resources is available. Limitations include scope constraints to the current functionalities and potential scalability considerations for future expansions.

4. UML Modeling

This section aims to offer an in-depth look at the system's underlying conceptual structure, brought into focus with the aid of the below UML diagrams, which serve as a clear guide for understanding and implementing the system's design.

4.1. Use Case Diagram:

The Use Case Diagram offers an overview of the system's interactions with its two primary actors: Company and User. The primary use cases are Create hackathon, Upload dataset, Set Rules, Create training, Submit Solutions, Get Hackathon Alert, Access Training, Get Organization events alert, Sponsor Event, and Register for Event. Additionally, there are extended use cases Retrieve results, Rank the candidates, and Get Hackathon Alert.

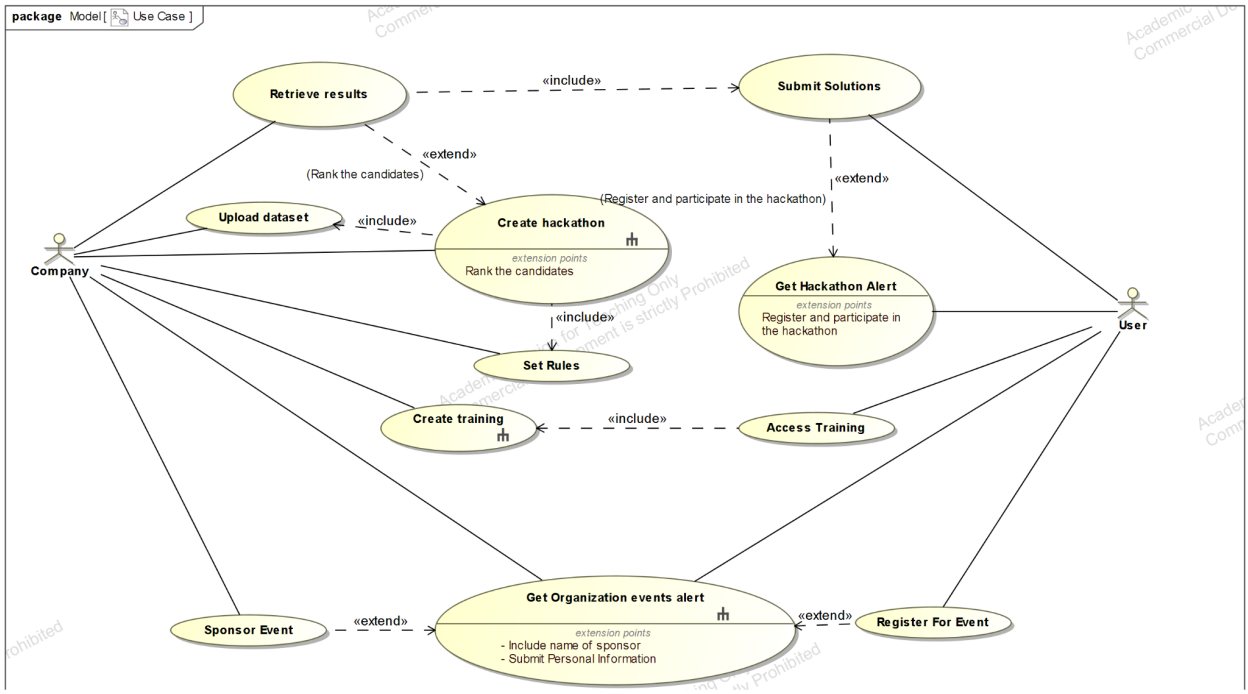

Fig 1: use case diagram

4.1.1 Actors Identification

Two primary actors are interacting with the system:

- **Company**: Represents entities like businesses or organizations that are responsible for creating and managing hackathons, training and events, setting rules for participation, uploading datasets for hackathons. Companies also receive alerts for events so that they can sponsor the events.
- **User**: Represents individuals or participants who interact with the system by submitting solutions to hackathons, registering for events, receiving alerts for hackathons and organization events, and accessing training material.

4.1.2 Use Case Scenarios

1. **Create hackathon (Company)**: A company logs into the system to create a new hackathon by providing necessary details such as the name, description, and duration.
2. **Upload dataset (Company):** After creating a hackathon, the company uploads relevant datasets that participants will use to develop their solutions.
3. **Set Rules (Company):** The company defines the rules and criteria for the hackathon, which could include eligibility, submission guidelines, and judging criteria.
4. **Create training (Company):** The company sets up training modules or sessions that can either be for hackathons or in general on the SoBigData analytics.

5. **Get Hackathon Alert (User):** Users receive notifications about upcoming hackathons, prompting them to register and participate.
6. **Submit Solutions (User):** Participants, identified as users, submit their solutions to the hackathons they have registered for.
7. **Access Training (User):** Users access the training materials to enhance their skills.
8. **Get Organization events alert (User):** Users are alerted about events organized by the company, providing an opportunity to register and participate.
9. **Sponsor Event (Company)**: A company chooses to sponsor an event, where they can contribute to the event's success through financial support or content provision.
10. **Register for Event (User)**: When a user gets notified about an event, they can register for it through the system. This may involve filling out registration forms, providing personal information, and, if required, paying any associated fees.
11. **Retrieve Results (Company)**: After a hackathon ends, the company collects and reviews the solutions submitted by users. They may then proceed to evaluate the solutions according to the set rules and criteria.
12. **Rank the Candidates (Company)**: Post-evaluation, the company ranks the candidates based on the effectiveness and innovation of their solutions. These rankings can be used for some rewards and recognition, or even recruitment.

4.2. Profile Diagram:

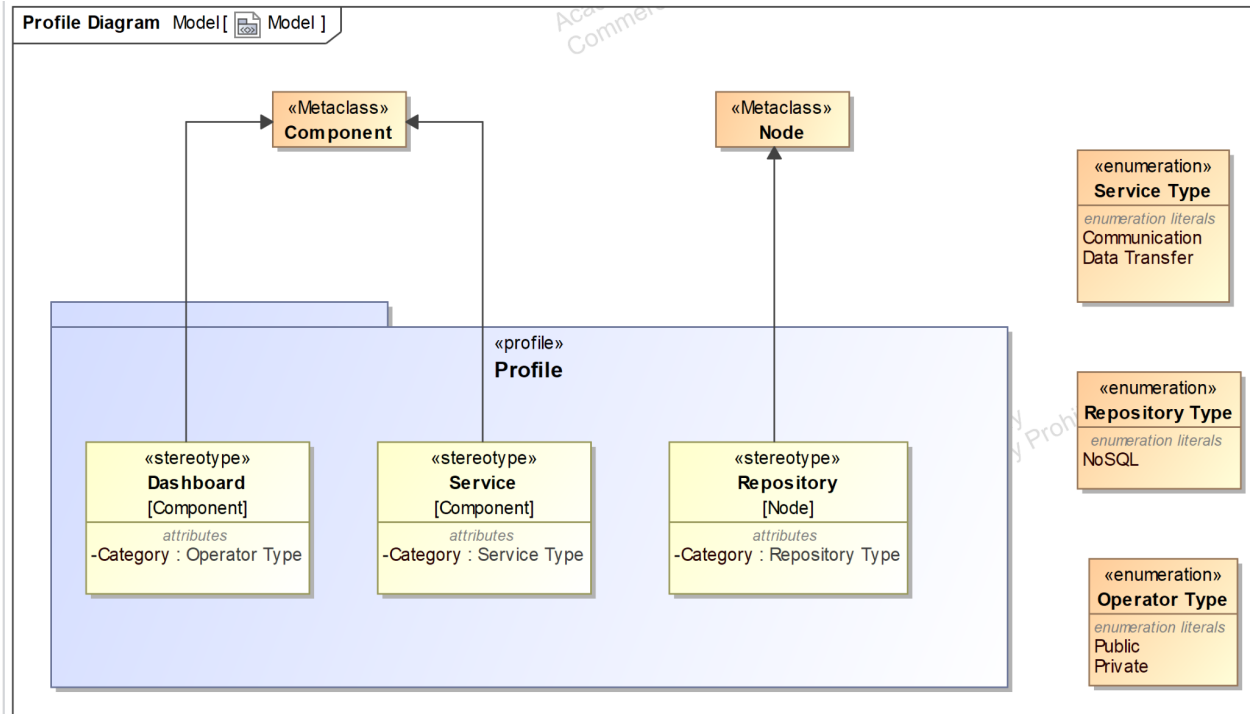

Fig 2: Profile Diagram

4.2.1. UML Metaclasses:

- **Component**: Extended with two stereotypes, Dashboard and Service, to the types of components offered by the system.
- **Node**: Extended with the Repository stereotype, indicating a persistence element in the system architecture that functions as a repository.

4.2.2. Stereotypes Defined:

- **Dashboard**: This stereotype represents the user interface components for both users and companies. It is categorized as an Operator Type, which further classifies the dashboard as either Public or Private.
- **Service**: Defines the logical components that handle specific business logic within the system, such as hackathon management, event management, trainings management and alert notification. Each service is categorized by a Service Type, which are Communication and Data Transfer.
- **Repository**: Represents the persistence components within the system, such as databases or file stores. The stereotype is categorized by Repository Type, specifically identifying the type of data storage, like NoSQL, used by the system.

4.2.3. Enumerations:

- **Service Type**: Enumerates the types of services that the system offers.
- **Repository Type**: Enumerates the types of repositories, providing a clear distinction that the system uses a NoSQL database.
- **Operator Type**: Enumerates the access level of the dashboards, distinguishing between public and private interfaces.

4.3. Component Diagram:

This diagram provides details of the system's components and illustrates the services and communication interfaces of different components within a system. There are 14 components in total and each component serves a specific function and interacts with each other through provided and required interfaces.

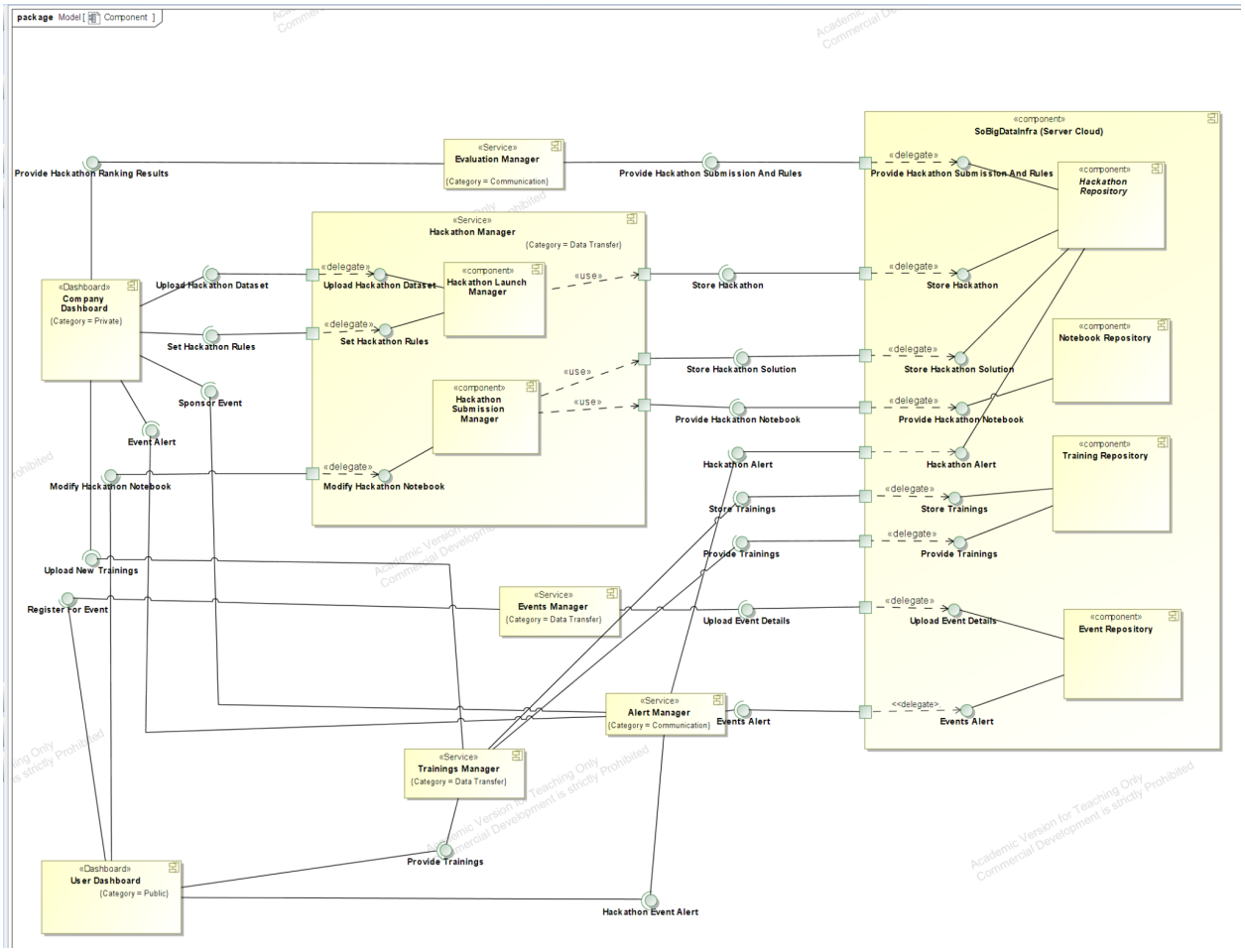

Fig 3: Component diagram

4.3.1. Component Descriptions

1. **User Dashboard:** This component is the primary interface for users through which users interact with the system, such as submitting solutions for hackathons, receiving alerts for events and hackathons, registering for events and accessing training materials.
2. **Company Dashboard:** This component is the primary interface for companies to interact with the system such as for uploading datasets, setting hackathon rules, sponsoring events, and modifying hackathon notebooks.
3. **Hackathon Manager**: This component serves as the central hub for managing hackathon-related activities. For example, the interfaces on the Hackathon Manager suggest it provides functionality to upload datasets and set rules, which are used by other components.
    a. **Hackathon Launch Manager:** It is responsible for the creation and setup of new hackathons.
    b. **Hackathon Submission Manager:** It handles the entries of the solutions for the hackathon that are provided by participants

4. **Evaluation Engine**: This component is responsible for the assessment and ranking of hackathon submissions, interfacing with the Hackathon Manager.
5. **Events Manager**: This component responsible for the management and provision of events. It interfaces with an events repository to access and store events related data.
6. **Training Manager:** This component is responsible for the management and provision of training.
7. **Alert Manager**: This component functions as the notification manager of the system. It generates alerts for hackathons and events, interfacing with both the Hackathon Manager and Events Manager. It sends alerts about upcoming events and hackathons to users and companies .
8. **SoBigData Infra(Server Cloud):** These are specialized data storage components within the SoBigData Infra. Each repository stores relevant data for its domain—hackathon details, user notebooks, training materials, and event information. They serve as the backbone for the content management within the system, with other components depending on them for data retrieval and storage.
    a. **Hackathon Repository:** Stores and manages hackathon submission and rules interfacing with Evaluation Manager.
    b. **Notebook Repository:** Stores and manages hackathon notebooks, interfacing with the Company Dashboard for notebook modifications.
    c. **Training Repository:** Stores training materials, interfacing with the Training Manager to provide training to users.
    d. **Event Repository:** Maintains event-related data, likely used by the Event Manager to store and retrieve event details.

4.4. Sequence Diagram

For the sequence diagram, three major use cases of the system have been considered which are `Create hackathon`, `Create training` and `Get Organization events alert`.

4.4.1 Sequence Diagram for `Create Hackathon`

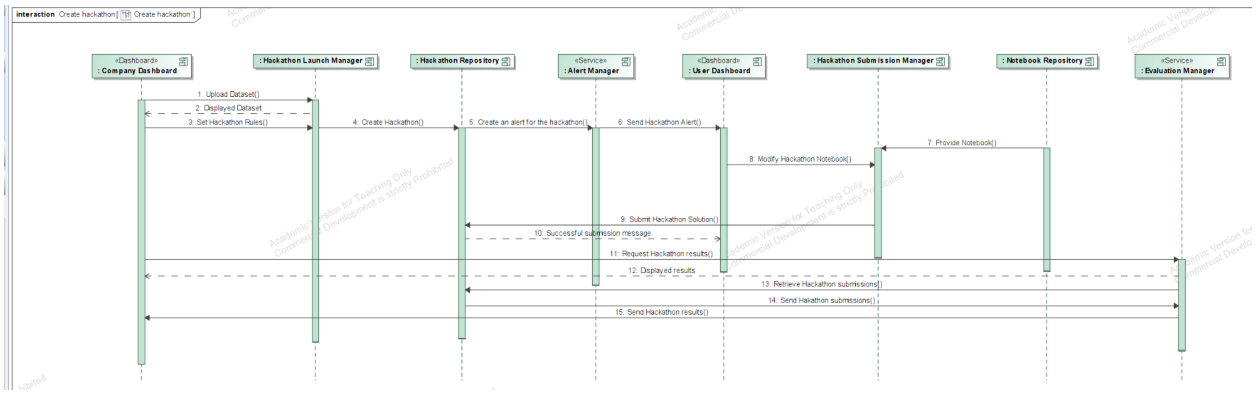

Fig 4: Sequence Diagram for `Create Hackathon` use case

The flow starts with the **Company Dashboard** sending a message to the **Hackathon Launch Manager** to create a new hackathon. The **Company Dashboard** first uploads the relevant dataset, which is essential

for the participants to use during the hackathon and also sets the rules for the hackathon, detailing the criteria and guidelines. The **Hackathon Launch Manager** interacts with the **Hackathons Repository**, to set up and allocate resources for the hackathon. Once the hackathon is created, the **Hackathons Repository** sends a message to **Alert Manager** requesting to create an alert for the hackathon. Then **Alert Manager** sends an alert notification via the **User Dashboard** to notify potential participants. Users use their **User Dashboard** to participate in the hackathon by modifying the notebook provided by **Notebooks Repository** to submit the hackathon solution to the **Hackathon Submission Manager**. The **Hackathon Submission Manager** sends these submissions to the **Notebooks Repository** for storing them. Upon successful operation, **Hackathon Repository** replies back to the **User Dashboard** with a successful submission message. After the submission deadline for the hackathon, **Company Dashboard** sends a message to the **Evaluation Manager** requesting for Hackathon Results. Upon this request, the **Evaluation Manager** sends a message to **Hackathon Manager** to retrieve the Hackathon submissions for assessment. After receiving the submissions, the assessment is done by the **Evaluation Manager** and the results are sent back to the **Company Dashboard** so that the results can be displayed and seen by the companies, completing the cycle of creating a hackathon and receiving results.

4.4.2 Sequence Diagram for `Create Training`

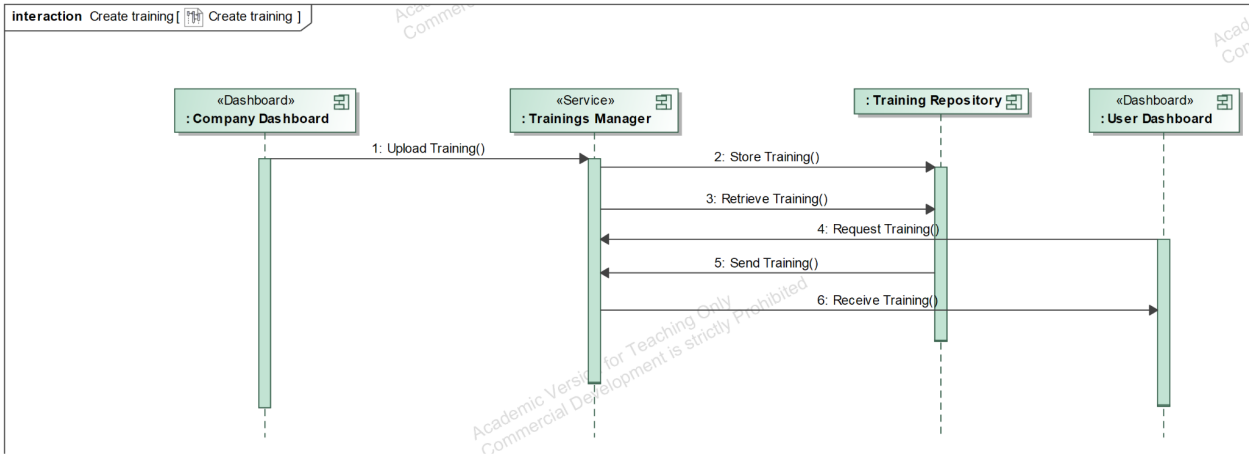

Fig 5: Sequence Diagram for `Create Training` use case

The **Company Dashboard** begins the process by uploading the training session details to the **Trainings Manager**. The **Trainings Manager**, upon receiving the information, sends a message to the **Training Repository** to store the training details which later can be retrieved as needed. And when a user wishes to participate in a training session, a request is sent from the **User Dashboard** to the **Trainings Manager** for the training**.** The info about training is then sent by the **Training Repository** to the **Trainings Manager,** upon the receival of which, the training info will be sent back to the **User Dashboard**. Hence, the user's **User Dashboard** receives the training, completing the process.

4.4.2 Sequence Diagram for `Get Organization events alert`

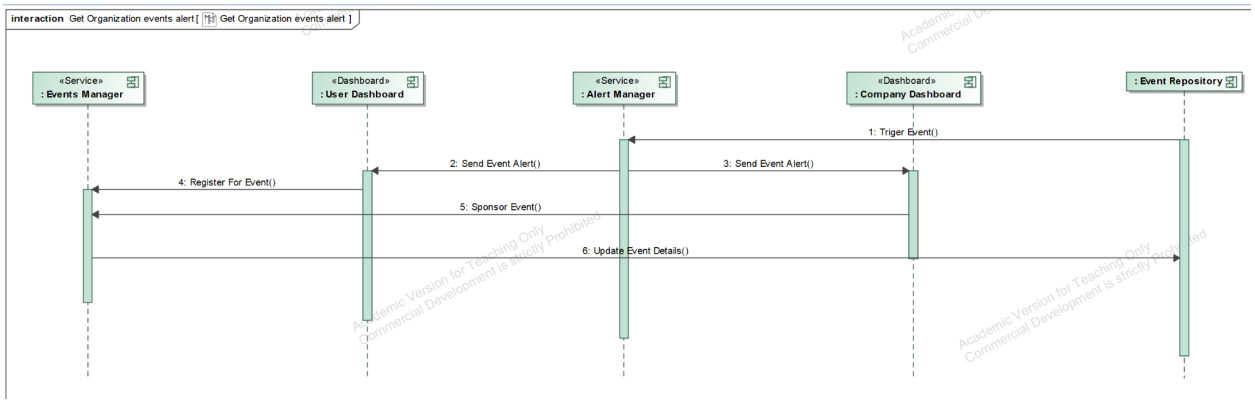

Fig 6: Sequence Diagram for `Get Organization events alert` use case

The flow begins with the **Event Repository** initiating a request to the **Alert Manager** to trigger alerts for events that were received by Event Repository for storage. The **Alert Manager** then dispatches an event alert to the **User Dashboard** and **Company Dashboard** both effectively notifying users and companies of the upcoming event. Users can respond to the alert by registering for the event through their **User Dashboard**. This registration request is acknowledged by the **Events Manager**. Similarly, companies can sponsor events, which involves the **Company Dashboard** sending the message to the **Events Manager** which will acknowledge this request and send the request to the **Event Repository** to update the event details with this info.

4.5. Deployment Diagram:

This diagram shows how software artifacts of the system are deployed onto hardware nodes to illustrate the distribution of components across a system.

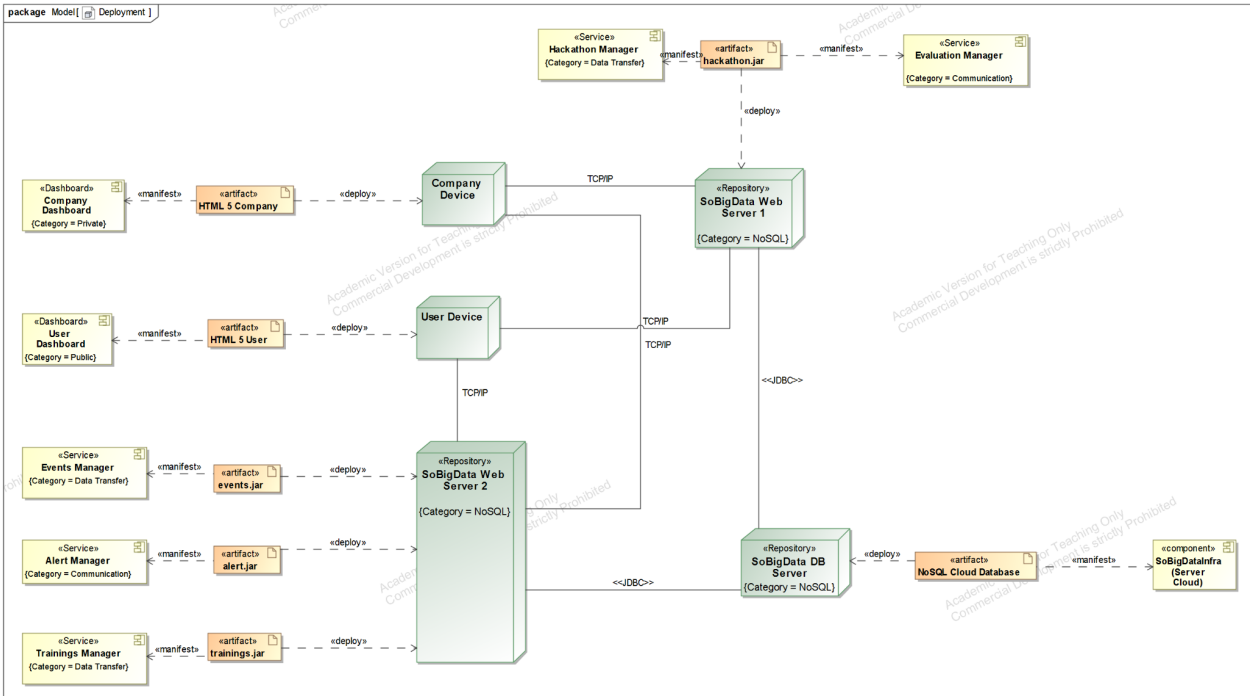

Fig 7: Deployment Diagram

4.5.1. Deployment units

1. **Nodes**: The system has 5 nodes which represent physical devices or locations where the software components are deployed. They are "Company Device," "User Device," "SoBigData Web Server 1," "SoBigData Web Server 2," and "SoBigData DB Server".
2. **Artifacts**: The artifacts in the diagram include "HTML 5 Company," "HTML 5 User," "hackathon.jar," "events.jar," "alert.jar," and "trainings.jar" that are deployed onto respective nodes. They represent the software components such as web pages, services, and databases that provide the functionality of the SoBigData++ system.
3. **Communication**: The connections between nodes, indicated by the TCP/IP labels, mean that the devices communicate over a network using the TCP/IP protocol. This is typical for distributed systems where different parts of the application run on different devices.
4. **Repositories**: These nodes are special types of components that represent the system's data storage mechanisms. "SoBigData Web Server 1" and "SoBigData Web Server 2" are connected to "SoBigData DB Server" through JDBC (Java Database Connectivity) , indicating that these servers interact with the database server for data operations.
5. **Deployment**: The arrows labeled with "deploy" indicate that the artifacts are deployed onto the respective nodes. For example, "hackathon.jar" is deployed on "SoBigData Web Server 1" which means the server will run the software component contained in the hackathon.jar file. In the same way, "events_jar," "alert_jar," and "trainings_jar" are deployed on "SoBigData Web Server 2" where the server will run the software component contained in those .jar files. And these two web servers connect via JDBC to the "SoBigData DB Server" which is a database server for handling

all the data persistence, such as storing hackathon results, event details, and user information. The NoSQL database is used for storage, which is suitable for handling large volumes of unstructured data, typical in big data applications like this.

5. Conclusion

This system aims to support a collaborative environment for fostering innovation through conducting hackathons on the SoBigData++ platform, promoting a culture of knowledge sharing through industry-relevant training, and attracting new users to events through incentivizing them by the prospect of networking opportunities. Moreover, it also furthers the impact of the SoBigData++ platform by making it more feature-rich; and by aligning with SoBigData++'s objective to foster a vibrant multidisciplinary community for Social Mining and Big Data Analytics. The UML diagrams we created serve as strategic tools for future implementation across static, dynamic, and deployment views. Using MagicDraw to model our project allowed us to provide a consistent view of the features proposed by us towards augmenting the SoBigData++ platform and supported the attachment of each part in a coherent manner.