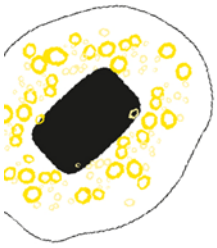
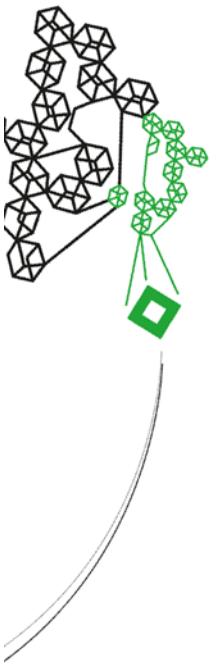


Data and Information 2023

**‘Database report’**

| | |
|---|---|
| Version | 1.0 |
| Date | 26-05-2023 |
| Status | Final |
| Team | EarnIT 4 |
| Names | (3077489) Gracjan Chmielnicki – BIT<br>(2993074) Tom Hansult - BIT<br>(2997894) Dirck Mulder - TCS<br>(2957566) Pepijn Meijer - TCS<br>(2920352) Thomas Brants - TCS<br>(2957868) Razvan Stefan - TCS |



# 1 The use case diagram

The system is divided into three distinct sections, each catering to different user groups. Certain functionalities, such as creating an account and accessing invoices, are shared between students and companies.

For students, the primary function revolves around submitting their working hours for approval. This process involves completing specific steps, including entering the hours worked and confirming them. Additionally, students have the option to include additional notes with their submissions. Other features available to students include a job overview and work statistics.

Companies, on the other hand, are responsible for approving or rejecting student requests. They can also have access to an overview of their workforce and their employees work statistics.

The Earnit staff possesses the authority to disable user accounts and companies and modify the hourly wage in accordance with the agreements between students and companies. They are also responsible for making decisions regarding rejected submissions and have the ability to add notes to their decisions. Furthermore, the staff has a comprehensive overview of both students and companies and supervises the establishment of connections between companies and students through contracts.

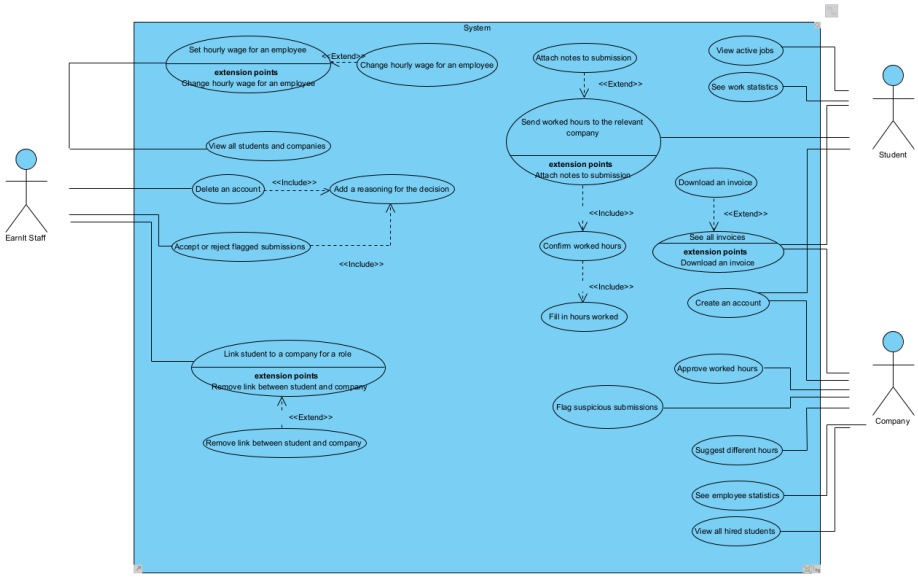



*Figure 1: Use case diagram*

# 2 Class diagram

Upon the creation of a user account by a student or a company, a new entry is generated in the user table, signifying the existence of a distinct instance. A company is obliged to have at least one associated user. The companyUser table stores information regarding all the employees affiliated with a particular company, facilitating multiple individuals to engage in the process of approving or rejecting student submissions. The staff possess the capability to generate contracts tailored to the specific requirements set out by the affiliated companies. A staff member, assigned with the task, establishes the connection between students and contracts in the user_contract table. Within this table, the hourly wage is stored as well, as it is possible for students with the same job designation to have varying payroll structures. The user_contract table also encompasses worked weeks, which effectively monitors the amount of work completed on a weekly basis, thereby aiding in the generation of invoices. Each worked week is composed of numerous instances documented in the worked table, which is updated each time a student submits their hours. Additionally, the company, contract, user_contract, and user tables feature an attribute denoted as "active." Rather than erasing inactive users from the database, this attribute is set to "false," thereby enabling the preservation of historical records to facilitate, for instance, the retrieval of past employee information by companies.

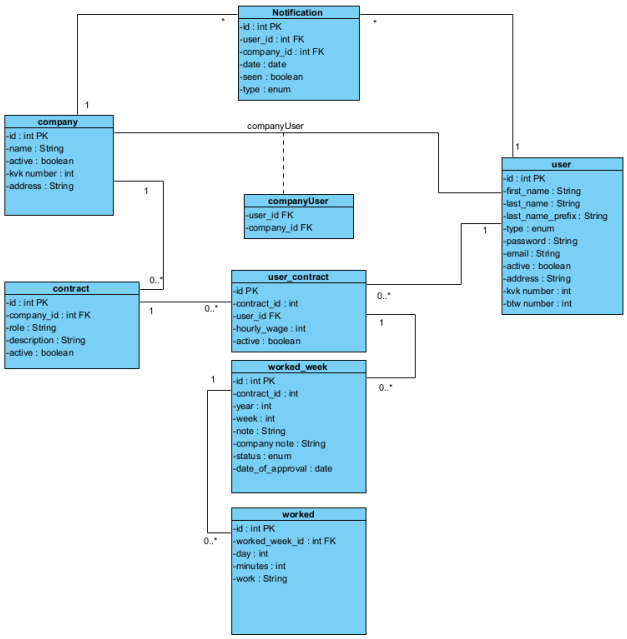



*Figure 2: Class diagram*

# 3 Database schema

The database schema is constructed based on the UML class diagram, with each table being accompanied by corresponding SQL statements for creation. The "gen_random_uuid()" function provided by PostgreSQL is utilized as the default value for the "id" attribute in each table. The "active" column, present in almost all tables, defaults to "true" and includes a not null constraint to prevent undesired scenarios. The only columns that can have null values are "approved" and "solved". Additionally, primary keys are established, and foreign key constraints are applied to maintain referential integrity. The "user.type" column is subjected to a check constraint to ensure that only valid values are assigned to it. Furthermore, the database makes use of triggers and functions. They are used in two situations: Firstly, when an account or company is disable, i.e. the active attribute is set to false, triggers make sure that the relevant rows from the other tables also set their active attribute to flase. For example, when a company is deactivated, its contracts and entries in the user_contract are also disabled. Secondly, triggers are used for generating notifications. When relevant updates are made in the database, triggers check if new notifications can be made, and if so, they are automatically added in the notifications table. For example, when a company approves a worked week, a notifications for approved week will be generated for the student, or, if a student forgets to submit hours for a week, the triggers will generated a notification to remind the student to submit the hours. Additionally, when a new link between a student or a company is set, both will get a notification. Furthermore, the database stores a seen attribute which makes it possible to show the new notifications on top for the student, company or staff.

[please find SQL schema document in docs/submissions/sprintTwo]