Module 4 Project Assignment 2

Date: 01.06.2023

Team: Earnit4

THOMAS BRANTS: S2920352

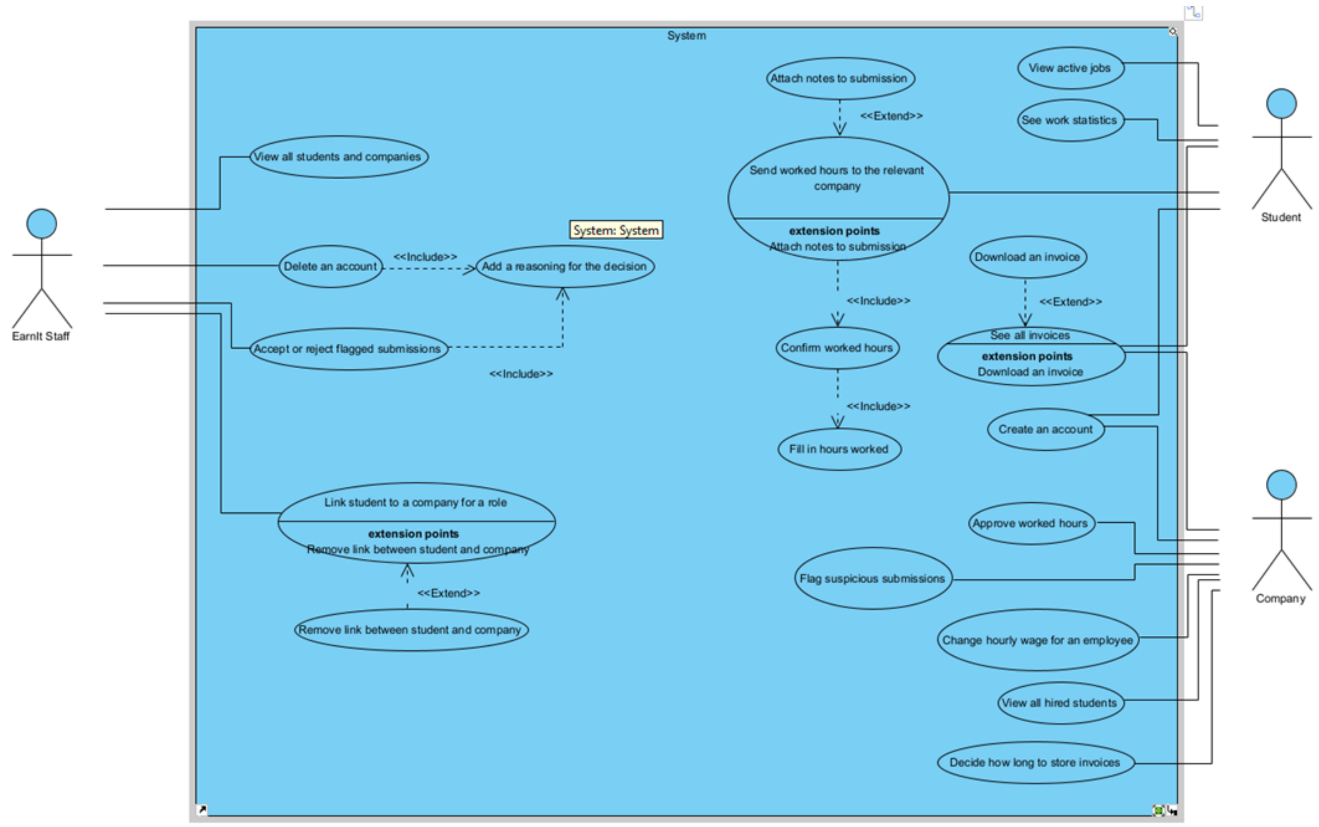
GRACJAN CHMIELNICKI: S3077489

TOM HANSULT: S2993074

PEPIJN MEIJER: S2957566

DIRCK MULDER: S2997894

RAZVAN STEFAN: S2957868


System
System: System
EarnIt Staff
Student
Company
View all students and companies
Delete an account
<<Include>>
Add a reasoning for the decision
Accept or reject flagged submissions
<<Include>>
Link student to a company for a role
extension points
Remove link between student and company
<<Extend>>
Remove link between student and company
Attach notes to submission
<<Extend>>
Send worked hours to the relevant company
extension points
Attach notes to submission
<<Include>>
Confirm worked hours
<<Include>>
Fill in hours worked
View active jobs
See work statistics
Download an invoice
<<Extend>>
See all invoices
extension points
Download an invoice
Create an account
Approve worked hours
Flag suspicious submissions
Change hourly wage for an employee
View all hired students
Decide how long to store invoices

Use Case Diagram

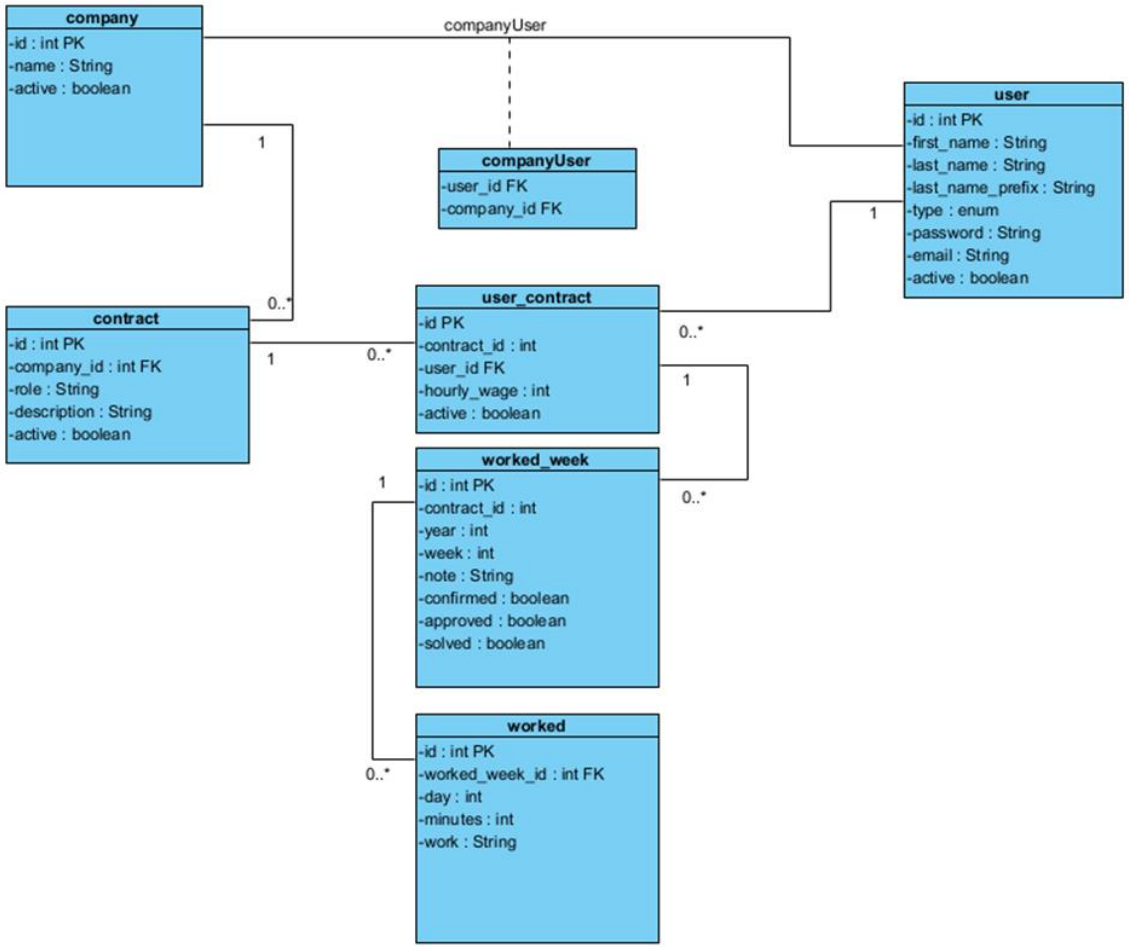
Our system is split into three different sections, each one being meant for our distinct kinds of users. Some functionalities are shared by the users and companies, such as creating an account and seeing and downloading invoices.

The most important functionality for the student is to send their hours to the company for approval. For that, they need to complete a couple of extra steps: filling in the worked and hours and confirming them. Additionally, they can add notes to the submission if they want to. Students also have some other features, such as a job overview and work statistics.

The companies will have to either approve or reject student requests. They also can change the hourly wage in accordance with their agreement with their employees and have an overview of their employees.

The Earnit staff can disable user accounts and companies. Additionally, they make decisions on the rejected submissions and can also write notes on their decisions. Also, the staff has an overview of both students and companies and oversee linking companies and students through contracts.

Class Diagram



Whenever a student or a company wants to create an account, a new instance will appear in the user table. A company will have at least one user associated with it. All the employees of a company are stored in companyUser table, so multiple people can work on approving/rejecting student submissions. A company can create contracts for their different needs. A staff member will link student and contracts in the user_contract table. This is where we also store the hourly wage because it is possible that two students that have the same job have a different payroll. The user_contract has worked weeks, to keep track of the work done every week and to generate invoices. Every worked_week is composed of multiple instances from the worked table, which is going to be filled every time a student fills in their hours. Additionally, the company, contract, user_contract and user tables have an attribute called active. Instead of deleting inactive users, we will set this attribute as false, so we can keep old records in our database for companies to be able to see previous employees for example.

Database Schema

The database schema is derived from the uml class diagram. All tables in the uml class diagram have sql statements to create them. For every id in a table the postgresql function gen_random_uuid() is used as default. Every active column defaults to true. Almost all columns also have a not null constraint to prevent unwanted cases. Only the `approved` and `solved` columns can be null. Further more are the primary keys added and the constraints for foreign references. The user.type column also contains a check to make sure only valid values are given.

```
CREATE TABLE "company" (
"id" UUID NOT NULL DEFAULT gen_random_uuid(),
“name" TEXT NOT NULL,
"active" BOOLEAN NOT NULL DEFAULT TRUE
);

ALTER TABLE "company" ADD PRIMARY KEY("id");

CREATE TABLE "user" (
"id" UUID NOT NULL DEFAULT gen_random_uuid(),
"email" TEXT NOT NULL,
"first_name" TEXT NOT NULL,
"last_name" TEXT NOT NULL,
"last_name_prefix" TEXT,
"type" VARCHAR(255) CHECK ("type" IN('STUDENT', 'COMPANY', 'ADMINISTRATOR')) NOT NULL,
"password" TEXT NOT NULL,
"active" BOOLEAN NOT NULL DEFAULT TRUE
);

ALTER TABLE "user" ADD PRIMARY KEY("id");

ALTER TABLE "user" ADD CONSTRAINT "user_email_unique" UNIQUE("email");

CREATE TABLE "worked_week" (
"id" UUID NOT NULL DEFAULT gen_random_uuid(),
“contract_id" UUID NOT NULL,
"year" BIGINT NOT NULL,
"week" BIGINT NOT NULL,
"note" TEXT NULL,
"confirmed" BOOLEAN NOT NULL DEFAULT '0',
"approved" BOOLEAN NULL,
"solved" BOOLEAN NULL
);

ALTER TABLE "worked_week" ADD CONSTRAINT "worked_week_id_unique" UNIQUE("id");

ALTER TABLE "worked_week" ADD PRIMARY KEY("contract_id", "year", "week");

CREATE TABLE "user_contract" (
"id" UUID NOT NULL DEFAULT gen_random_uuid(),
"contract_id" UUID NOT NULL,
"user_id" UUID NOT NULL,
"hourly_wage" BIGINT NOT NULL,
"active" BOOLEAN NOT NULL DEFAULT TRUE
);

ALTER TABLE "user_contract" ADD PRIMARY KEY("id");
```

```
CREATE TABLE "worked" (
"id" UUID NOT NULL DEFAULT gen_random_uuid(),
"worked_week_id" UUID NOT NULL,
"day" SMALLINT NOT NULL,
"minutes" BIGINT NOT NULL,
"work" TEXT NOT NULL
);

ALTER TABLE "worked" ADD PRIMARY KEY("id");

CREATE TABLE "company_user" (
"user_id" UUID NOT NULL,
"company_id" UUID NOT NULL
);

ALTER TABLE "company_user" ADD PRIMARY KEY("user_id", "company_id");

CREATE TABLE "contract" (
"id" UUID NOT NULL DEFAULT gen_random_uuid(),
"company_id" UUID NOT NULL,
"role" TEXT NOT NULL,
"description" TEXT NOT NULL,
"active" BOOLEAN NOT NULL DEFAULT TRUE
);

ALTER TABLE "contract" ADD PRIMARY KEY("id");

ALTER TABLE "user_contract" ADD CONSTRAINT "user_contract_contract_id_foreign" FOREIGN KEY("contract_id") REFERENCES
"contract"("id");

 ALTER TABLE "worked_week" ADD CONSTRAINT "worked_week_contract_id_foreign" FOREIGN KEY("contract_id") REFERENCES
"user_contract"("id");

ALTER TABLE "company_user" ADD CONSTRAINT "company_user_company_id_foreign" FOREIGN KEY("company_id") REFERENCES
"company"("id");

ALTER TABLE "user_contract" ADD CONSTRAINT "user_contract_user_id_foreign" FOREIGN KEY("user_id") REFERENCES "user"("id");

ALTER TABLE  "worked" ADD CONSTRAINT "worked_worked_week_id_foreign" FOREIGN KEY("worked_week_id") REFERENCES
"worked_week"("id");

ALTER TABLE "company_user" ADD CONSTRAINT "company_user_user_id_foreign" FOREIGN KEY("user_id") REFERENCES "user"("id");

ALTER TABLE "contract" ADD CONSTRAINT "contract_company_id_foreign" FOREIGN KEY("company_id") REFERENCES
"company"("id");
```