CS 353
DATABASE SYSTEMS
PROJECT DESIGN REPORT
GROUP 36

LAMİA BAŞAK AMAÇ
21601930 SECTION 01
ALPER KANDEMİR
21703062 SECTION 01
SELAHATTİN CEM ÖZTÜRK
21802856 SECTION 03

Project Description

Our project is a system that helps students to learn language by attemting a class of a teacher or scheduling a meeting with a native speaker.

This sytem allow teachers to create lesson, assign homework to his/hers students, give exam, give grade to the the students’ assignments, upload a certificate to their profile and create their own word lists for different languages and different language levels.

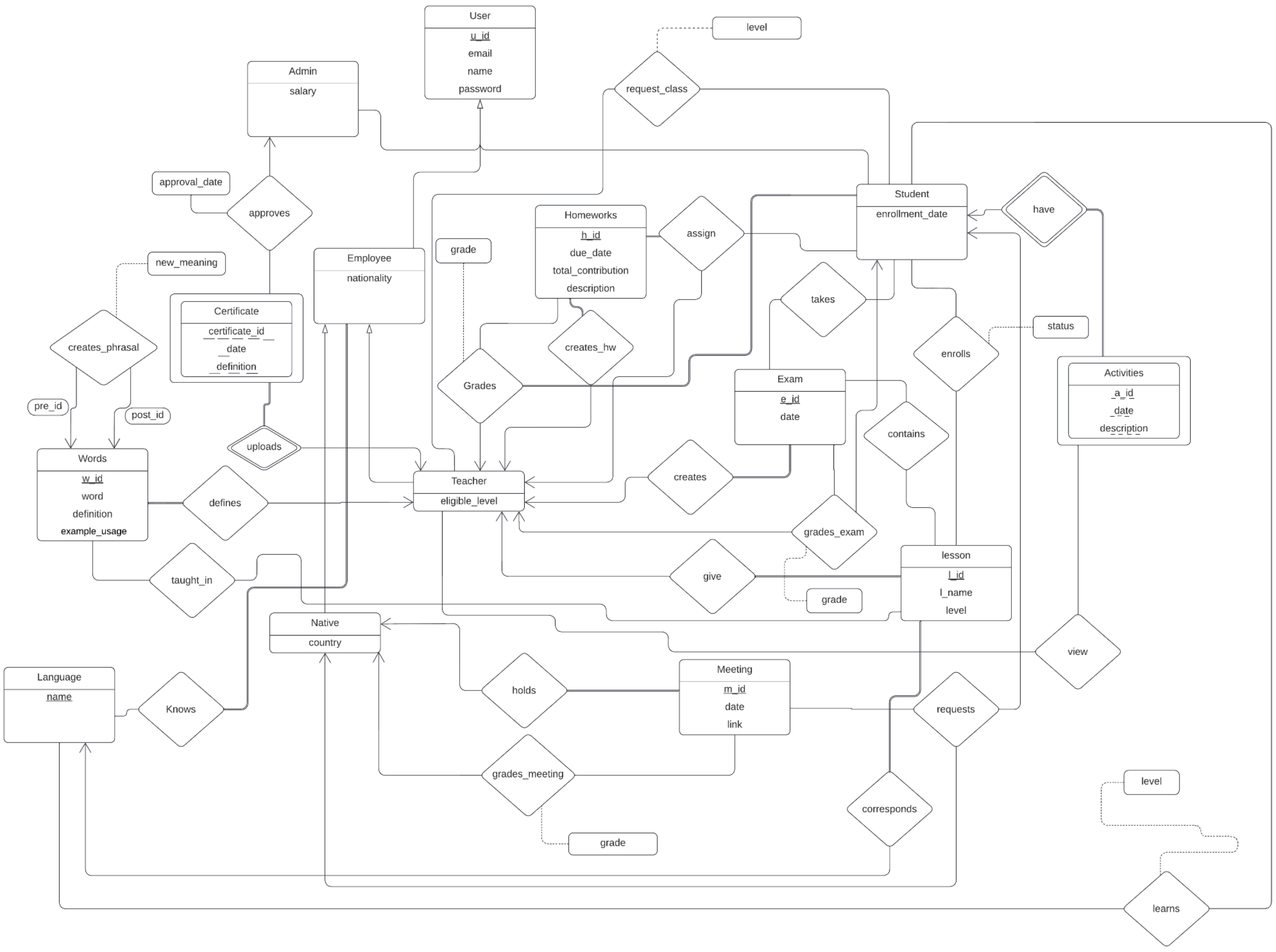
Natives can schedule a meeting and give grade to a meeting

Students can request classes, enroll in an existing class of a teacher, view their grades, send a meeting request to a native speaker for practice their speaking skills.

Admin of the system can view the classes’ analytics such as how many students are enrolled to a particular class and also they can approve the uploaded certificate.

In a nutshell, this online language learning platform meets the students with teachers and natives to improve their language skills.

Final E/R Diagram



- Language attribute is removed from lesson and employee; instead, a relation called Language is created.
- A three-way relation is created between student, teacher and homework. Now our model successfully implies the function of the teacher which is assigning homework to a specific student.
- A three-way relation is created between student, native and meeting. Now our model successfully implies the function of the student which is requesting a meeting from a specific native speaker.
- Primary key in the user has changed to an u_id instead of email.
- Name and id attributes in student, employee, and admin now moved to the user
- Level attribute of the student removed since we added a new association between language and student as “learns”
- Nationality attribute added to the employee and country attribute added to the Native so that our Native relation is not empty right now.
- Status attribute added to the enroll, so that we can keep track of which lesson request is approved and which are not.
- Create homework relation is created between homework and teacher
- All the user id types are changed to integer and new attribute added to User relation which stores the user type

Table Schemas

2.1 User

Relational Model
User(u_id, email, name, password, u_type)

Primary Key
u_id

Candidate Key
u_id

Functional Dependencies
u_id → email, name, password

Normal Form
BCNF

Table Declaration

```
CREATE TABLE User(
        u_id INT NOT NULL AUTO_INCREMENT,
        email VARCHAR(32) NOT NULL,
        name VARCHAR(32) NOT NULL,
        password VARCHAR(32) NOT NULL,
        PRIMARY KEY(u_id)) ENGINE = INNODB;
```

2.2 Admin

Relational Model

Admin(u_id, salary)
u_id is foreign key to User.

Candidate Keys
u_id

Functional Dependencies:
u_id→ salary

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Admin (
        u_id CHAR(8) PRIMARY KEY,
        salary FLOAT,
        FOREIGN KEY(u_id) REFERENCES User(u_id) ON UPDATE CASCADE
ON DELETE RESTRICT) ENGINE=INNODB;
```

2.3 Employee

Relational Model

Employee(<u>u_id</u>, nationality)
u_id is a foreign key to user

Candidate Keys
u_id

Primary Key
u_id

Functional Dependencies:
u_id → nationality

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Employee (
        u_id CHAR(8) PRIMARY KEY,
        nationality VARCHAR(32) NOT NULL,
        FOREIGN KEY(u_id) REFERENCES User(u_id) ON UPDATE CASCADE
ON DELETE RESTRICT) ENGINE=INNODB;
```

2.4 Student

Relational Model

Student(u_id, enrollment_date)
Email is foreign key to User.

Candidate Keys
u_id

Primary Key
u_id

Functional Dependencies:
u_id → enrollment_date

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Student (
        u_id CHAR(8),
        enrollment_date DATE NOT NULL,
        PRIMARY KEY(u_id),
        FOREIGN KEY(u_id) REFERENCES User(u_id) ON UPDATE CASCADE
ON DELETE RESTRICT) ENGINE=INNODB;
```

2.5 Native

Relational Model

Native(u_id, country)
u_id is foreign key to User.

Candidate Keys
u_id

Primary Key
u_id

Functional Dependencies:
u_id → country

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Native (
        u_id CHAR(8),
        nationality VARCHAR(32) NOT NULL,
        PRIMARY KEY(u_id),
        FOREIGN KEY(u_id) REFERENCES User(u_id) ON UPDATE CASCADE
ON DELETE RESTRICT) ENGINE=INNODB;
```

2.6 Teacher

Relational Model

Teacher(u_id, eligible_level)
u_id is foreign key to Employee.

Candidate Keys
u_id

Primary Key
u_id

Functional Dependencies:
u_id → eligible_level

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Teacher (
        u_id CHAR(8) PRIMARY KEY,
        eligible_level VARCHAR(32) NOT NULL,
        FOREIGN KEY(u_id) REFERENCES Employee(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.7 request_class

Relational Model

request_class(t_id, s_id, level)
t_id foreign is key to Teacher(u_id).
s_id foreign key to Student(u_id).

Candidate Keys
t_id, s_id

Primary Key
t_id, s_id

Functional Dependencies:
t_id, s_id → level

Normal Form
BCNF

Table Declaration

```
CREATE TABLE request_class (
        t_id CHAR(8),
        s_id CHAR(8),
        level VARCHAR(32) NOT NULL,
        PRIMARY KEY(t_id,s_id),
        FOREIGN KEY(t_id) REFERENCES Teacher(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(s_id) REFERENCES Student(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.8 Homeworks

Relational Model

Homeworks(h_id, due_date, total_contribution, description)

Candidate Keys
h_id

Primary Key
h_id

Functional Dependencies:
h_id → due_date, total_contribution, description

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Homeworks (
        h_id CHAR(8),
        description VARCHAR(32),
        due_date VARCHAR(32) NOT NULL,
        total_contribution INT NOT NULL,
        PRIMARY KEY(h_id)) ENGINE=INNODB;
```

2.9 creates_hw

Relational Model

creates_hw(h_id, u_id)
h_id is foreign key to Homeworks.

Candidate Keys
h_id
Primary Key
h_id
Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE creates_hw (
        h_id CHAR(8) NOT NULL,
        u_id CHAR(8) NOT NULL,
        PRIMARY KEY(h_id),
        FOREIGN KEY(h_id) REFERENCES Homeworks(h_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.10 assign

Relational Model

assign(t_id, s_id, h_id)
s_id is foreign key to Student(u_id)
h_id is foreign key to Homeworks.

Candidate Keys
t_id, s_id, h_id
Primary Key
s_id, h_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE assign (
        t_id CHAR(8),
        s_id CHAR(8),
        h_id CHAR(8),
        PRIMARY KEY( s_id, h_id),
        FOREIGN KEY(s_id) REFERENCES Student(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(h_id) REFERENCES Homeworks(h_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.11 Grades

Relational Model

Grades(t_id, s_id h_id, grade)

s_id is foreign key to Student(u_id).
h_id is foreign key to Homeworks.

Candidate Keys
s_id, h_id
Primary Key
s_id, h_id

Functional Dependencies:
t_id, s_id, h_id→ t_id, s_id, h_id, grade

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Grades (
        t_id CHAR(8) NOT NULL,
        s_id CHAR(8) NOT NULL,
        h_id CHAR(8) NOT NULL,
        grade INT NOT NULL,
        PRIMARY KEY(s_id, h_id),
        FOREIGN KEY(s_id) REFERENCES Student(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(h_id) REFERENCES Homeworks(h_id) ON UPDATE
    CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.12 Exam

Relational Model

Exam(<u>e_id,</u>date)

Candidate Keys
e_id

Primary Key
e_id

Functional Dependencies:
e_id → date

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Exam (
        e_id CHAR(8),
        date DATE NOT NULL,
        PRIMARY KEY(e_id)) ENGINE=INNODB;
```

2.13 Takes

Relational Model

Takes(e_id,us_id)
e_id is foreign key to Exam.
u_id is foreign key to Student.

Candidate Keys
e_id, u_id
Primary Key
e_id, u_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Takes (
        e_id CHAR(8) NOT NULL,
        s_id CHAR(8),
        PRIMARY KEY(e_id, s_id),
        FOREIGN KEY(e_id) REFERENCES Exam(e_id) ON UPDATE CASCADE
ON DELETE RESTRICT,
        FOREIGN KEY(email) REFERENCES Student(u_is) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.14 creates

Relational Model

creates(e_id, u_id)
e_id is foreign key to Exam.

Candidate Keys
e_id
Primary Key
e_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE creates (
	e_id CHAR(8) NOT NULL,
	u_id CHAR(8) NOT NULL,
	PRIMARY KEY(e_id),
	FOREIGN KEY(e_id) REFERENCES Exam(e_id) ON UPDATE CASCADE
ON DELETE RESTRICT) ENGINE=INNODB;
```

2.15 grades_exam

Relational Model

grades_exam(t_id, s_id e_id, grade)
t_id is foreign key to Teacher(u_id).
s_id is foreign key to Student(u_id).
e_id is foreign key to Exam.

Candidate Keys
s_id e_id
Primary Key
s_id e_id

Functional Dependencies:
s_id e_id → grade

Normal Form
BCNF

Table Declaration

```
CREATE TABLE grades_exam (
        t_id VARCHAR(32) NOT NULL,
        s_id CHAR(8) NOT NULL,
        e_id CHAR(8) NOT NULL,
        grade INT NOT NULL,
        PRIMARY KEY( s_id e_id),
        FOREIGN KEY(s_id) REFERENCES Student(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(e_id) REFERENCES Exam(e_id) ON UPDATE CASCADE
    ON DELETE RESTRICT) ENGINE=INNODB;
```

2.16 lesson

Relational Model

lesson(l_id, l_name, level)

Candidate Keys
l_id

Primary Key
l_id

Functional Dependencies:
l_id → l_name, level

Normal Form
BCNF

Table Declaration

```
CREATE TABLE lesson (
        l_id CHAR(8),
        l_name VARCHAR(32)  NOT NULL,
        level VARCHAR(32) NOT NULL,
        PRIMARY KEY(l_id)) ENGINE=INNODB;
```

2.17 give

Relational Model

give(u_id, l_id)
u_id is foreign key to Teacher.
l_id is foreign key to lesson.

Candidate Keys
l_id
Primary Key
l_id

Functional Dependencies:
There are no functional dependencies.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE give (
        u_id VARCHAR(32) NOT NULL,
        l_id CHAR(8) NOT NULL,
        PRIMARY KEY( l_id),
        FOREIGN KEY(l_id) REFERENCES lesson(l_id) ON UPDATE CASCADE
ON DELETE RESTRICT) ENGINE=INNODB;
```

2.18 contains

Relational Model

contains(e_id, l_id)
e_id is foreign key to Exam.
l_id is foreign key to lesson.

Candidate Keys
e_id, l_id
Primary Key
e_id, l_id

Functional Dependencies:
There are no functional dependencies.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE contains (
        e_id CHAR(8) NOT NULL,
        l_id CHAR(8) NOT NULL,
        PRIMARY KEY(e_id, l_id),
        FOREIGN KEY(e_id) REFERENCES Exam(e_id) ON UPDATE CASCADE
ON DELETE RESTRICT,
        FOREIGN KEY(l_id) REFERENCES lesson(l_id) ON UPDATE CASCADE
    ON DELETE RESTRICT) ENGINE=INNODB;
```

2.19 enrolls

Relational Model

enrolls(u_id, l_id, status)
u_id is foreign key to Student.
l_id is foreign key to lesson.

Candidate Keys
u_id, l_id
Primary Key
u_id, l_id

Functional Dependencies:
u_id, l_id → status

Normal Form
BCNF

Table Declaration

```
CREATE TABLE enrolls (
        u_id CHAR(8) NOT NULL,
        l_id CHAR(8) NOT NULL,
        status VARCHAR(32),
        PRIMARY KEY(u_id, l_id),
        FOREIGN KEY(u_id) REFERENCES Student(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(l_id) REFERENCES lesson(l_id) ON UPDATE CASCADE
    ON DELETE RESTRICT) ENGINE=INNODB;
```

2.20 Meeting

Relational Model

Meeting(m_id, date, link)

Candidate Keys
m_id, date

Primary Key
m_id

Functional Dependencies:
m_id →date, link

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Meeting (
        m_id CHAR(8) NOT NULL,
        date DATE  NOT NULL,
        link VARCHAR(32)  NOT NULL,
        PRIMARY KEY(m_id)) ENGINE=INNODB;
```

2.21 requests

Relational Model

requests(m_id, s_id, n_id)
m_id is foreign key to Meeting.
s_id is foreign key to Student(u_id).
n_id is foreign key to Native(u_id).

Candidate Keys
m_id
Primary Key
m_id

Functional Dependencies:
There are no functional dependencies.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE requests (
        m_id CHAR(8) NOT NULL,
        s_id CHAR(8) NOT NULL,
        n_id CHAR(8) NOT NULL,
        PRIMARY KEY(m_id),
        FOREIGN KEY(m_id) REFERENCES Meeting(m_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.22 holds

Relational Model

holds(u_id, <u>m_id</u>)
u_id is foreign key to Native.
m_id is foreign key to Meeting.

Candidate Keys
m_id
Primary Key
m_id

Functional Dependencies:
There are no functional dependencies.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE holds (
        u_id CHAR(8),
        m_id CHAR(8),
        PRIMARY KEY(m_id),
        FOREIGN KEY(m_id) REFERENCES Meeting(m_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.23 grades_meeting

Relational Model

grades_meeting(u_id, m_id, grade)
u_id is foreign key to Native.
m_id is foreign key to Meeting.

Candidate Keys
m_id
Primary Key
m_id

Functional Dependencies:
m_id → grade

Normal Form
BCNF

Table Declaration

```
CREATE TABLE grades_meeting (
	u_id CHAR(8),
	m_id CHAR(8),
	grade FLOAT,
	PRIMARY KEY( m_id),
	FOREIGN KEY(m_id) REFERENCES Meeting(m_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.24 Words

Relational Model

Words(w_id, word, definition, example_usage)

Candidate Keys
w_id

Primary Key
w_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Words (
        w_id CHAR(8) NOT NULL,
        word VARCHAR(32)  NOT NULL,
        definition VARCHAR(32)  NOT NULL,
        example_usage VARCHAR(32)  NOT NULL,
        PRIMARY KEY(w_id)) ENGINE=INNODB;
```

2.25 taught_in

Relational Model

taught_in(w_id, l_id)
w_id is foreign key to Words.
l_id is foreign key to lesson.

Candidate Keys
w_id, l_id
Primary Key
w_id, l_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE taught_in (
        w_id CHAR(8) NOT NULL,
        l_id CHAR(8) NOT NULL,
        PRIMARY KEY(w_id, l_id),
        FOREIGN KEY(w_id) REFERENCES Words(w_id) ON UPDATE
CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(l_id) REFERENCES lesson(l_id) ON UPDATE CASCADE
ON DELETE RESTRICT) ENGINE=INNODB;
```

2.26 defines

Relational Model

defines(u_id, w_id)
u_id is foreign key to User.
w_id is foreign key to Words.

Candidate Keys
w_id
Primary Key
w_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE defines (
        u_id VARCHAR(32) NOT NULL,
        w_id CHAR(8) NOT NULL,
        PRIMARY KEY( w_id),
        FOREIGN KEY(w_id) REFERENCES Words(w_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.27 creates_phrasal

Relational Model

creates_phrasal(pre_id, post_id, new_meaning)
pre_id is foreign key to Words(w_id).
post_id is foreign key to Words(w_id).

Candidate Keys
pre_id
Primary Key
pre_id

Functional Dependencies:
pre_id → new_meaning

Normal Form
BCNF

Table Declaration

```
CREATE TABLE creates_phrasal (
        pre_id CHAR(8) NOT NULL,
        post_id CHAR(8) NOT NULL,
        new_meaning VARCHAR(32) NOT NULL,
        PRIMARY KEY(pre_id),
        FOREIGN KEY(pre_id) REFERENCES Words(w_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.28 Certificate

Relational Model

Certificate(certificicate_id, date, definition, u_id)

Candidate Keys
certificicate_id, date, definition, u_id

Primary Key
certificicate_id, date, definition, u_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Certificate (
        certificate_id CHAR(8) NOT NULL,
        date DATE  NOT NULL,
        definition VARCHAR(32)  NOT NULL,
        u_id CHAR(8) NOT NULL,
        PRIMARY KEY(certificicate_id, date, definition, u_id),
        FOREIGN KEY(u_id) REFERENCES Teacher(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.29 approves

Relational Model

approves(certificicate_id, date, definition, u_id, approval_date)
certificate_id is foreign key to Certificate.
date is foreign key to Certificate.
definition is foreign key to Certificate.
u_id is foreign key to Teacher.

Candidate Keys
certificicate_id, date, definition, u_id
Primary Key
certificicate_id, date, definition, u_id

Functional Dependencies:
certificicate_id, date, definition, u_id →approval_date

Normal Form
BCNF

Table Declaration

```
CREATE TABLE approves(
        u_id CHAR(8) NOT NULL,
        certificicate_id CHAR(8) NOT NULL,
        date DATE NOT NULL,
        definition VARCHAR(32) NOT NULL,
        PRIMARY KEY(certificicate_id, date, definition, u_id),
        FOREIGN KEY(certificicate_id, date, definition) REFERENCES
Certificate(certificicate_id, date, definition) ON UPDATE CASCADE ON DELETE
RESTRICT,
        FOREIGN KEY(u_id) REFERENCES Teacher(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT,) ENGINE=INNODB;
```

2.30 Activities

Relational Model

Activities(a_id, date, description, s_id)
s_id is foreign key to Student(u_id)

Candidate Keys
a_id, date, description, s_id

Primary Key
a_id, date, description, s_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE Activities (
        a_id CHAR(8) NOT NULL,
        date DATE  NOT NULL,
        description VARCHAR(32)  NOT NULL,
        s_id CHAR(8) NOT NULL,
        PRIMARY KEY(a_id, date, description, s_id),
        FOREIGN KEY(s_id) REFERENCES Student(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.31 view

Relational Model

view(a_id, date, description, s_id, t_id)
a_id is foreign key to Activities.
date is foreign key to Activities.
description is foreign key to Activities.
s_id is foreign key to Activities.
t_id is foreign key to Teacher(u_id)

Candidate Keys
a_id, date, description, s_id, t_id
Primary Key
a_id, date, description, s_id, t_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE view(
        t_id CHAR(8) NOT NULL,
        a_id CHAR(8) NOT NULL,
        date DATE NOT NULL,
        description VARCHAR(32) NOT NULL,
        s_id CHAR(8) NOT NULL,
        PRIMARY KEY(a_id, date, description, s_id, t_id),
        FOREIGN KEY(a_id, date, description,s_id) REFERENCES Activity(a_id,
date, description, s_id) ON UPDATE CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(t_id) REFERENCES Teacher(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.32 corresponds

Relational Model

corresponds(lesson_id,name)
lesson_id is foreign key to lesson(l_id)
name is foreign key to Language(name)

Candidate Keys
lesson_id
Primary Key
lesson_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE corresponds(
        lesson_id CHAR(8) NOT NULL,
        name CHAR(8) NOT NULL,
        PRIMARY KEY(lesson_id),
        FOREIGN KEY(lesson_id) REFERENCES lesson(l_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.33 language

Relational Model

Language(<u>name</u>)

Candidate Keys
l_id
Primary Key
l_id

Functional Dependencies:
l_id→ name

Normal Form
BCNF

Table Declaration

```
CREATE TABLE language(
        name  VARCHAR(32) NOT NULL,
        PRIMARY KEY(name)) ENGINE=INNODB;
```

2.34 learns

Relational Model

learns(s_id, name, level)
name is foreign key to language(l_id)
s_id is foreign key to student(u_id)

Candidate Keys
s_id, name
Primary Key
s_id, name

Functional Dependencies:
s_id, name → level

Normal Form
BCNF

Table Declaration

```
CREATE TABLE learns(
        name CHAR(8) NOT NULL,
        s_id CHAR(8) NOT NULL,
        PRIMARY KEY(name, s_id),
        FOREIGN KEY(name) REFERENCES Language(name) ON UPDATE
CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(s_id) REFERENCES Student(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

2.35 knows

Relational Model

learns(name, u_id)
name is foreign key to Language(name)
u_id is foreign key to Employee(u_id)

Candidate Keys
name, u_id
Primary Key
name, u_id

Functional Dependencies:
There is no functional dependency.

Normal Form
BCNF

Table Declaration

```
CREATE TABLE knows(
        name CHAR(8) NOT NULL,
        u_id CHAR(8) NOT NULL,
        PRIMARY KEY(name, u_id),
        FOREIGN KEY(name) REFERENCES Language(name) ON UPDATE
CASCADE ON DELETE RESTRICT,
        FOREIGN KEY(u_id) REFERENCES Employee(u_id) ON UPDATE
CASCADE ON DELETE RESTRICT) ENGINE=INNODB;
```

FUNCTIONAL COMPONENTS

3.1. Algorithms

3.1.1 Class creation and enrollment related algorithms

Students will search for language class based on his/her level and the language that s/he wants. Students can not enroll in a class higher than his/her level in the corresponding language. Admin will view some general information about the system. Many students can use this system and many students can choose the same language and they can take the same lesson. It should be restricted that the same email cannot sign up again. (For uniqueness)

3.1.2 Logical Requirements

Logical errors should be eliminated. Our program has a date attribute and they are separated into two. One of them should take the current time of day to tell us what is the starting point of this attribute. The other one will be determined for later. Students will understand how much time they have. This kind of date should be checked for the date should be later than the current time. Moreover, enrollment dates for students should be the earliest date rather than activity, homework or exam dates. Some certificate dates can be later or earliest.

3.2 Data Structures

Alphabetic, numeric and date types make up our system. Numeric types are INT and FLOAT. For strings, CHAR and VARCHAR types are used. CHAR is generally used for ID's and VARCHAR is used for long strings and these strings are not predetermined. Finally, we used the DATE type for the relevant date information.

3.3 Use Cases

3.3.1 Teacher

-Create Account: A teacher can create accounts with a name, password and email. Also, the user type should be selected as a teacher. The email should be unique for every user.

-Login: A teacher can login with their emails and passwords. When a teacher login successfully, the system can be used. Lesson, Homework, Exam, Grading, Dictionary and Profile pages will be active.

-View Course Request: Teachers can see students' requests to join a particular lesson. Students' names will be shown and which course they want. Teachers can accept the requests any time. Also, teachers can create lessons by choosing language and level.

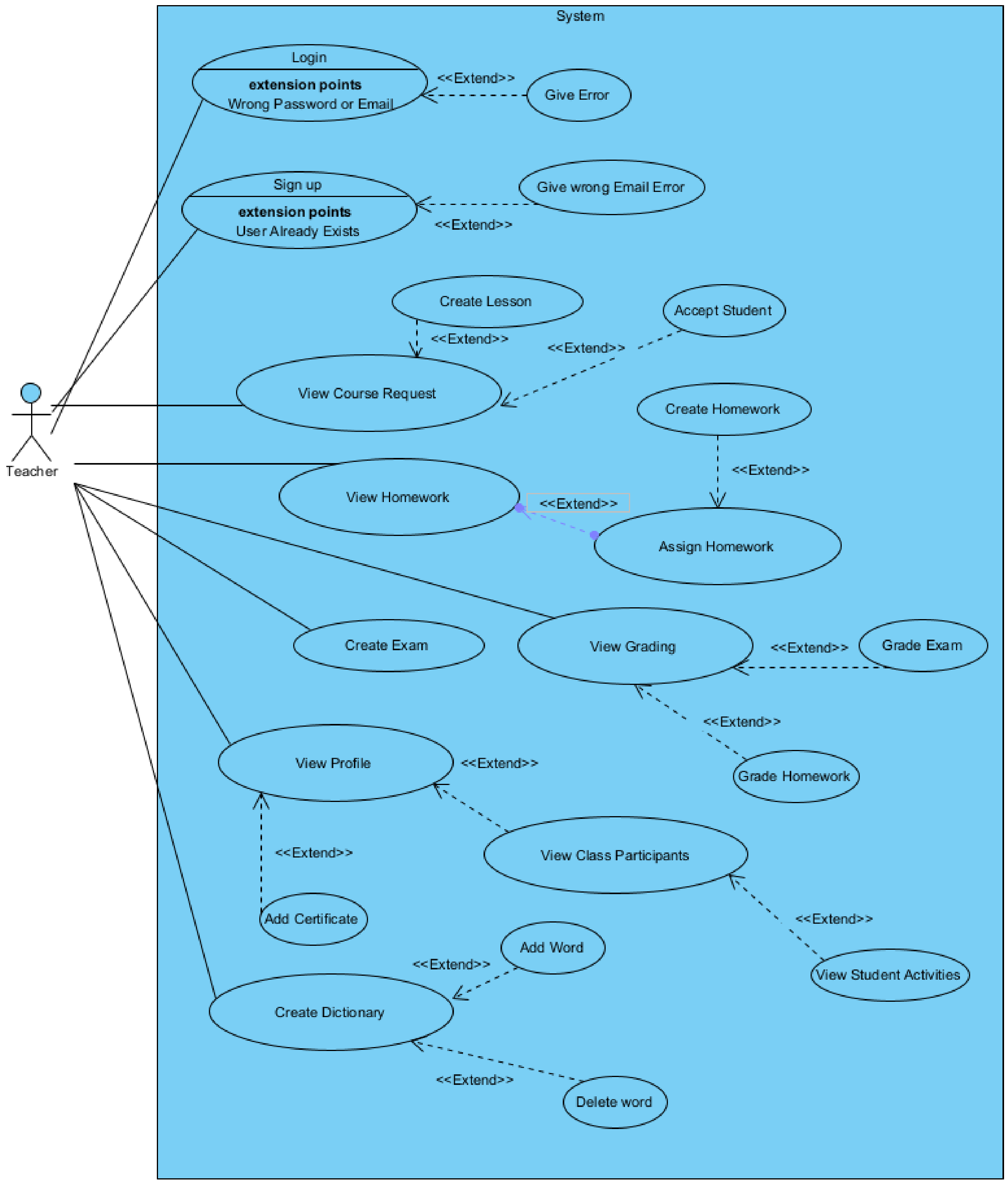
-View Homework: Teachers can see the information of the students from the student list and the desired student can be assigned homework.While the homework is given to the student, the date and description of the homework can be seen by the teachers. Teachers can also create another homework by entering new information.

-Create Exam: Teachers can create exams by choosing a date.

-View Grading: Teachers can grade students' homework or exams and the grades are given can be viewed as a list.

-View Profile: Teachers can add certificates and see the certificate they added on their profile page. In addition, they can see the number of participants in the courses they give and the activity of the desired student can be viewed.

-Create Dictionary: Teachers can see the words they add according to the selected lesson. New words can be added by entering a word, an explanation and an example and the desired word can be removed.


System
Login
extension points
Wrong Password or Email
<<Extend>>
Give Error
Sign up
extension points
User Already Exists
Give wrong Email Error
<<Extend>>
Create Lesson
<<Extend>>
Accept Student
<<Extend>>
View Course Request
Create Homework
<<Extend>>
Teacher
View Homework
<<Extend>>
Assign Homework
Create Exam
View Grading
<<Extend>>
Grade Exam
<<Extend>>
Grade Homework
View Profile
<<Extend>>
View Class Participants
<<Extend>>
Add Certificate
<<Extend>>
View Student Activities
Add Word
<<Extend>>
Create Dictionary
<<Extend>>
Delete word

3.3.2 Student

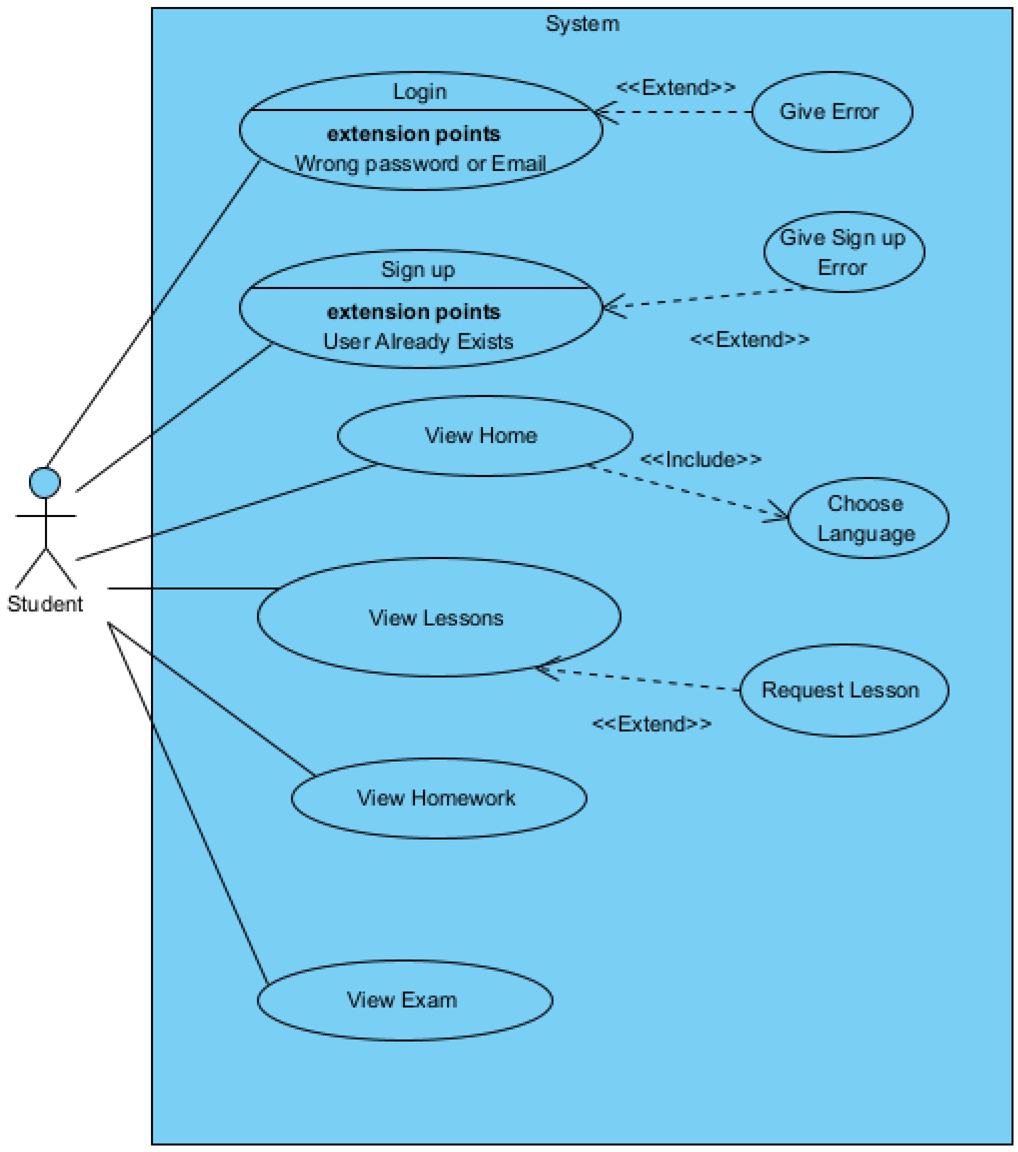
-Create Account: Student can create accounts with a name, password and email. Also, the user type should be selected as a student. The email should be unique for every user.

-Login: Students can login with their emails and passwords. When students login successfully, they have to choose language in order to use the system. After choosing a language Lesson, Homework, Exam, Meeting and Profile pages will be active.

-View Lesson: Students can send enrollment requests by choosing a language, level, and teacher and see the status of their requests.

-View Homework: Students can see the names of the given homework, the deadline, the total contribution and the grade,

-View Exam: The results, dates and grades of the exams taken by the students can be seen.



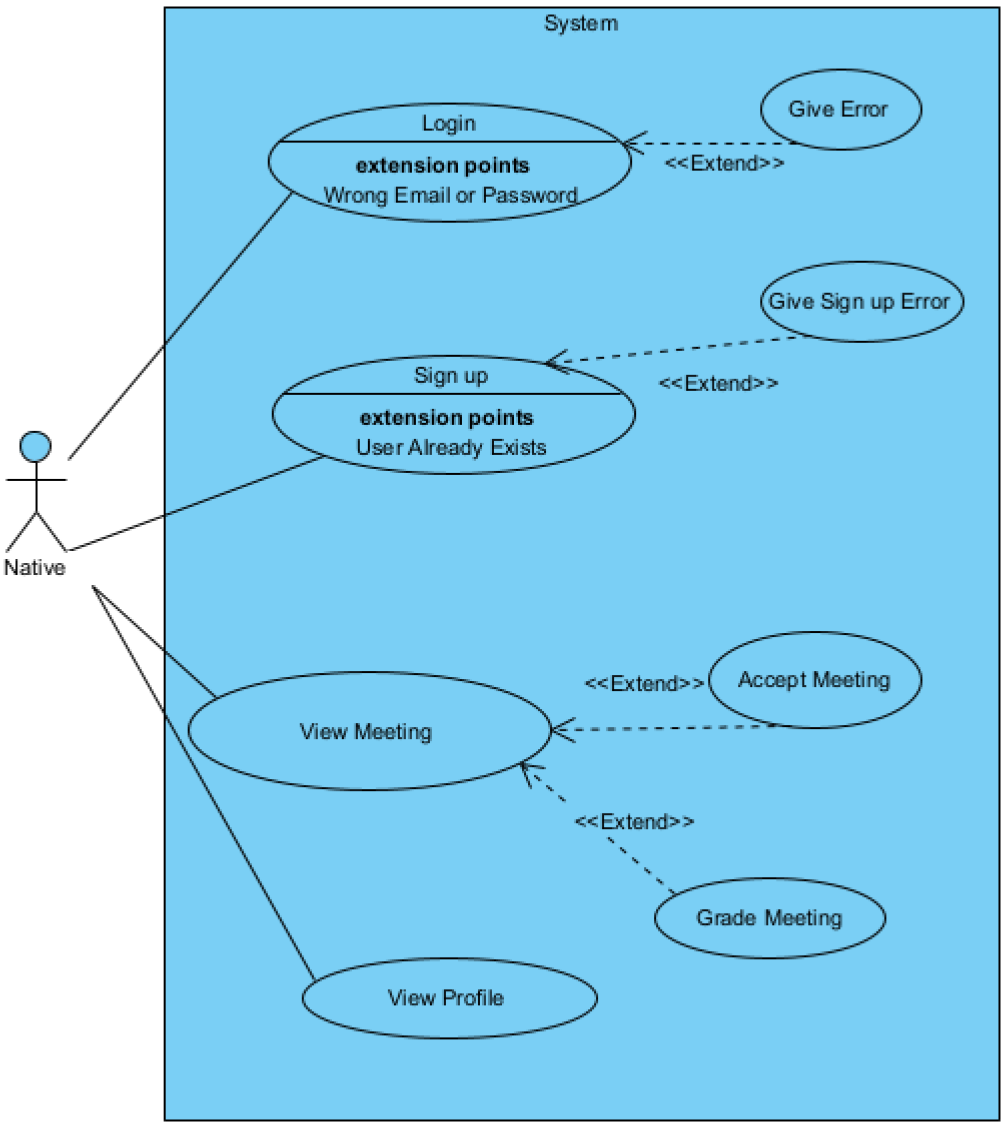
3.3.3 Native

-Create Account: Natives can create accounts with a name, password and email. Also, the user type should be selected as a native. The email should be unique for every user.

-Login: Natives can login with their emails and passwords. When Natives login successfully, they can use the system Meeting and Profile pages will be active.

-View Meeting: Natives can see and accept meeting requests from students. They can give grades to the meeting after the meeting.

-View Profile: Natives can see their profiles that including the nationality and country of the native



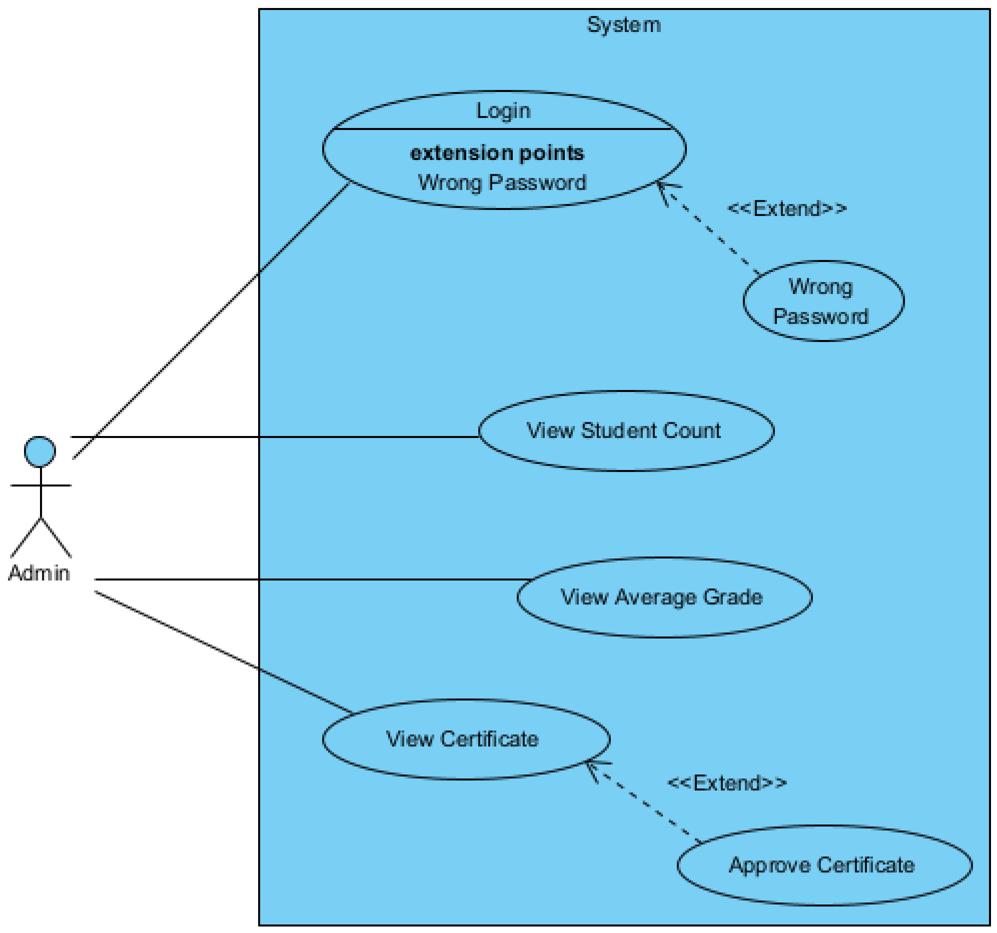
3.3.4 Admin

-Login: Admins can login with their emails and passwords. When admins login successfully, they can use the system Lesson and Certificate pages will be active.

-View Student Count: Admins can see the language of the lessons taught by the teachers and the number of students from the list of teachers.

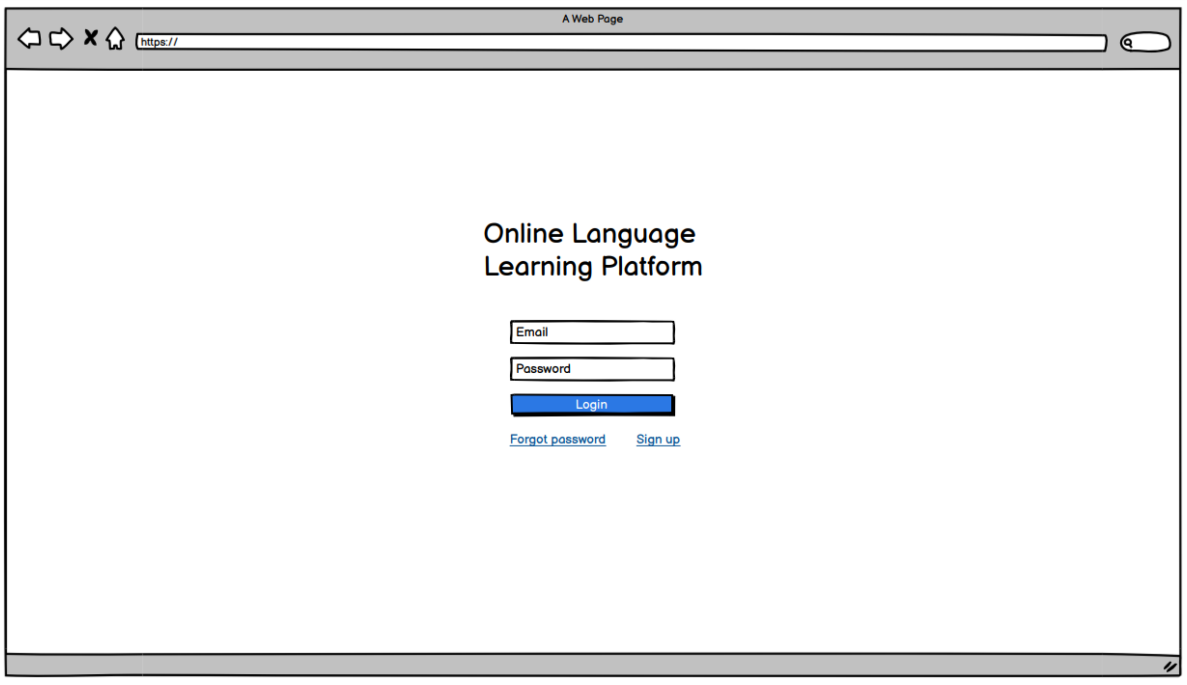
-View Average Grade: From the list of teachers, admins can see the average grade in the courses of the lessons taught by the teachers.

-View Certificate: Admins can check and accept the certificates that are added by teachers, and when they approve the certificate the approval date can be seen.



THE USER INTERFACE DESIGN AND CORRESPONDING SQL STATEMENTS

1) LOGIN PAGE



INPUTS

@email, @password

PROCESS

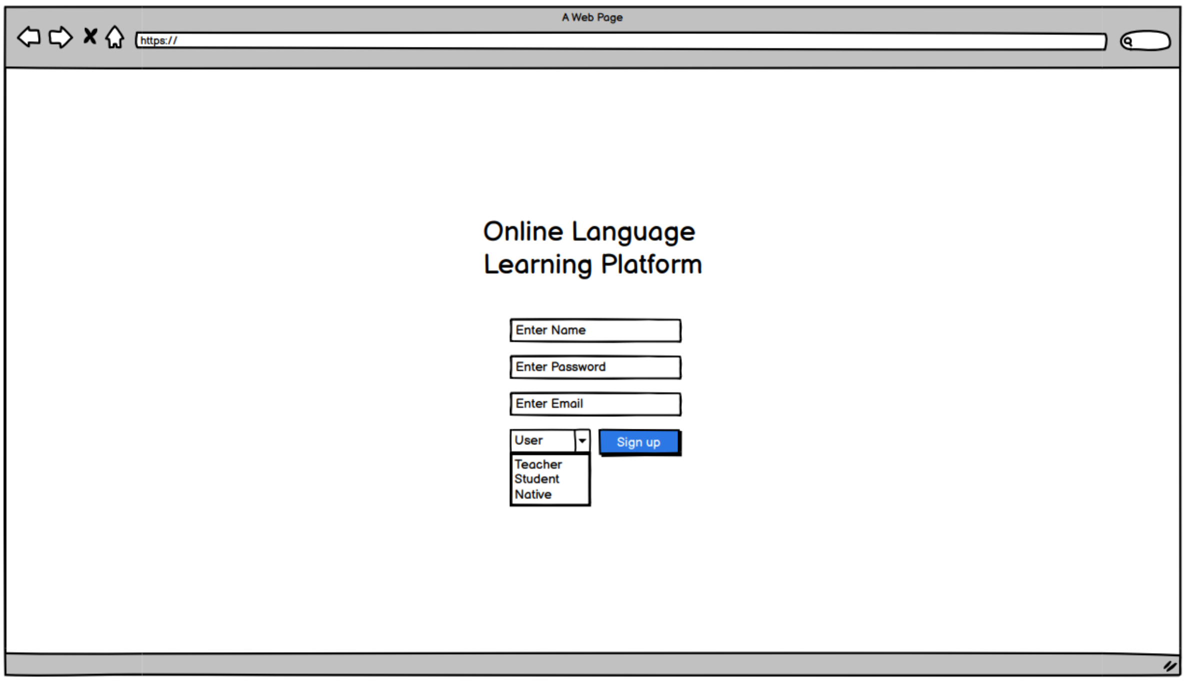
This page is for users that have accounts on the system. Users that do not have an account should go to the "sign up" page. Users will login the system with their email and password. After login to the system, the user will reach the home page. Admins, students, employees (natives and teachers) can use this page to login.

SQL QUERY

For login button,

```
SELECT u_id, email, password
FROM User
Where email = @email AND password = @password;
```

2) SIGN UP PAGE



INPUTS

@name, @password, @email

PROCESS

In this page, users should enter valid email and password because they will use them for login. User should specify his/her user type. They will be seperated with this information as Student, Teacher or Native.

SQL QUERY

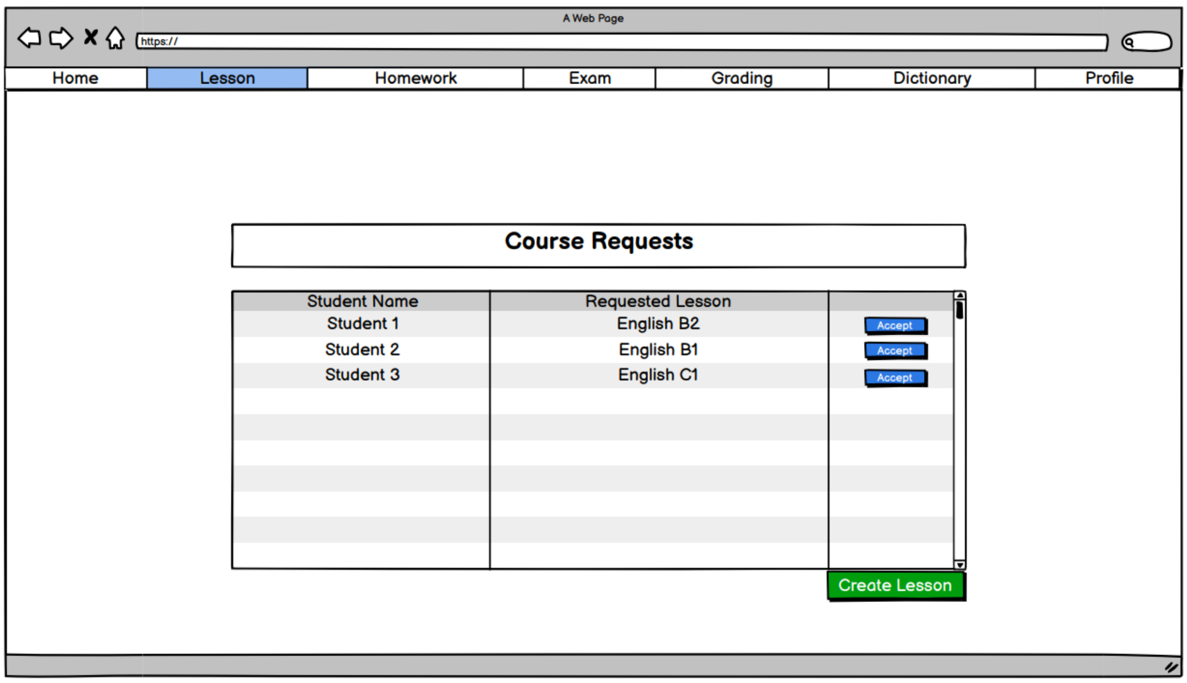
When Signup button pressed, if Student is selected,
INSERT INTO Student(u_id, email, name, password, enrollment_date)
VALUES(u_id, @email, @name, @password, enrollment_date)

When Signup button pressed, if Teacher is selected,
INSERT INTO Teacher(u_id, email, name, password, nationality, eligible_level)
VALUES(u_id, @email, @name, @password, nationality, eligible_level)

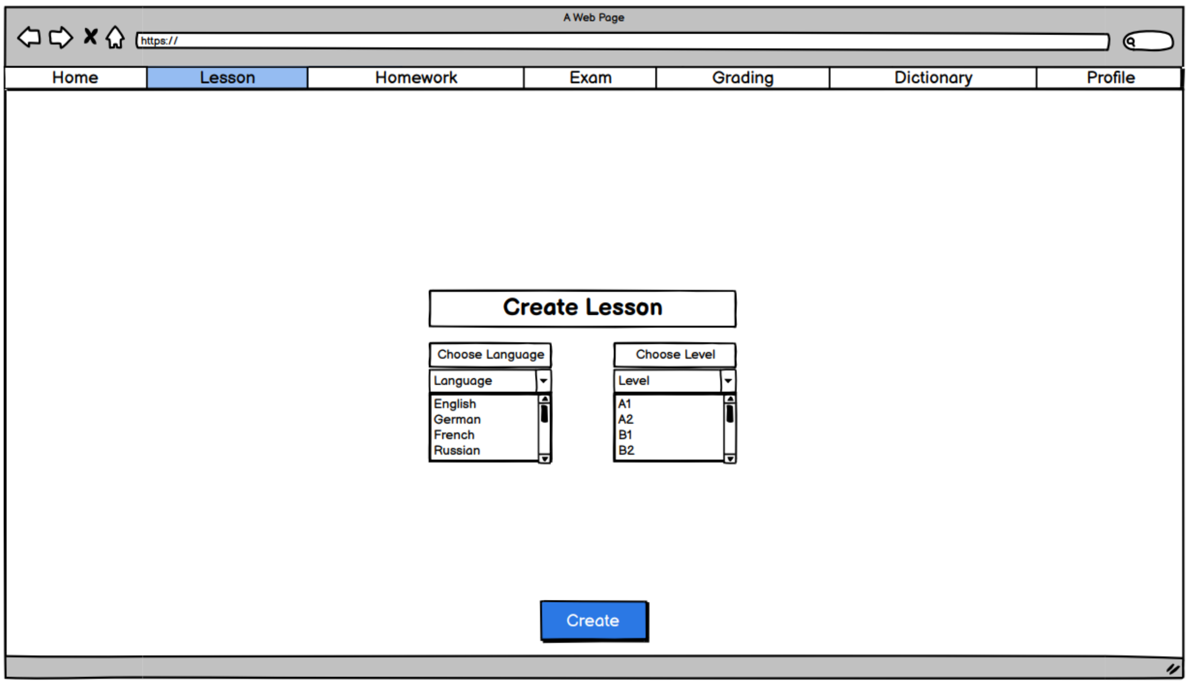
When Signup button pressed, if Native is selected,
INSERT INTO Native(u_id, email, name, password, nationality, country)

VALUES(u_id, @email, @name, @password, nationality, country)

3) LESSON PAGE FOR TEACHER



Student Name	Requested Lesson	
Student 1	English B2	Accept
Student 2	English B1	Accept
Student 3	English C1	Accept



INPUTS

@new_lan
@new_level

PROCESS

On this page, teachers can see lesson requests. The lesson name, and student's name can be seen.
On accept button corresponding student will be added to the course
Also teachers can add new lesson associated with them

SQL QUERY

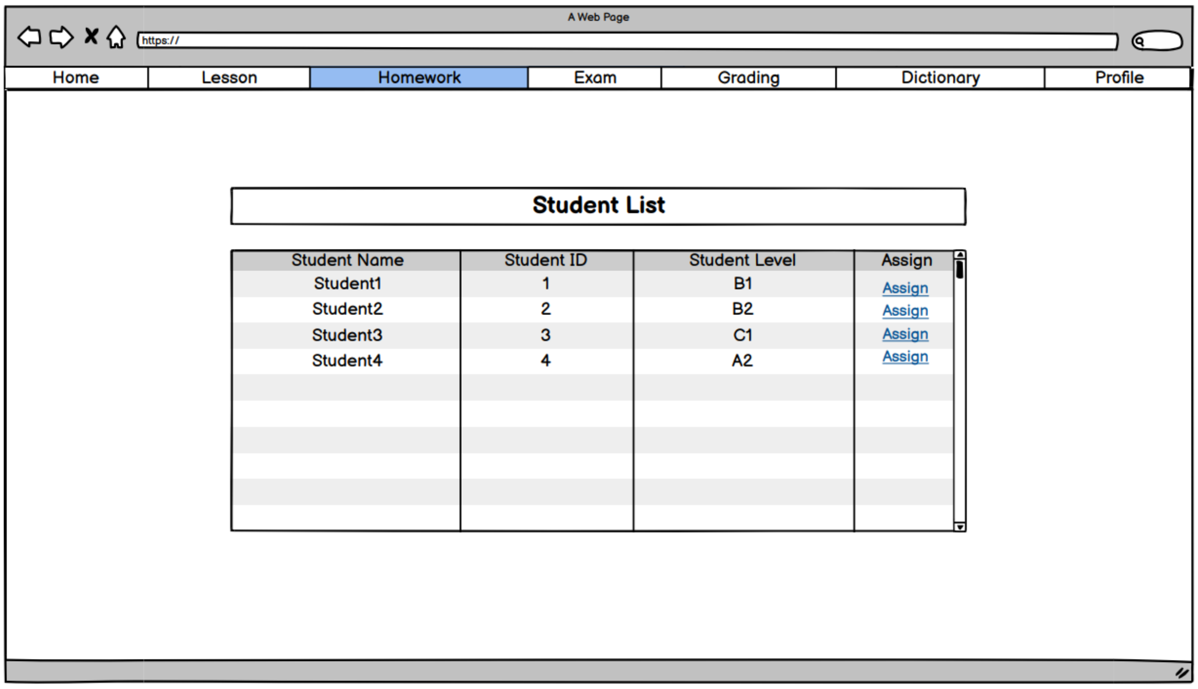
```
SELECT u.name, l.l_name
FROM Student s, User u, lesson l, enrolls e
Where e.status = waiting AND s.u_id=u.u_id AND e.u_id = s.u_id AND
        l.l_id = e.l_id

UPDATE enrolls e
SET status = "approved"
WHERE e.u_id = @u_id AND e.l_id = @l_id

INSERT INTO lesson(l_id, l_name, level)
VALUES(@l_id, @new_lan, @new_level)


INSERT INTO corresponds(lesson_id, language_id)
VALUES(@lesson_id, @language_id)
```

4) HOMEWORK PAGE FOR TEACHER



Student List

Student Name	Student ID	Student Level	Assign
Student1	1	B1	Assign
Student2	2	B2	Assign
Student3	3	C1	Assign
Student4	4	A2	Assign

INPUTS

-

PROCESS

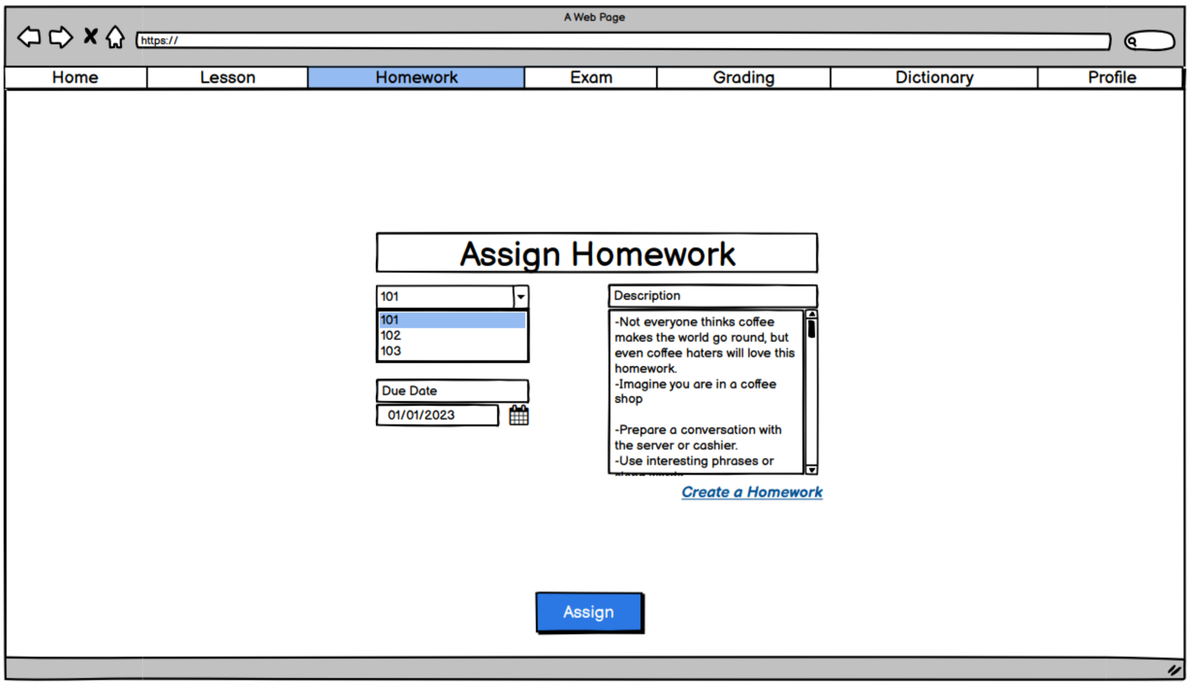
On this page, teachers can see students' information such as student name, student ID, student level. Teachers can assign homeworks in this page by pressing the “Add” button. When they press the “Add” button, they will reach the “Assign Homework” page.

SQL QUERY

```
SELECT S.name, S.u_id, req.level
FROM Student S, request_class req
WHERE S.u_id = req.u_id AND req.t_id = @u_id;

SELECT s.name, s.u_id, learns.level
FROM enrolls e, Student s, lesson l, give g, learns
WHERE s.u_id = e.u_is AND e.l_id = l.l_id AND l.l_id = g.l_id AND
        learns.u_id = s.u_id AND learns.name = @language AND
        g.u_id = @u_id;
```

5) ASSIGN HOMEWORK PAGE FOR TEACHER



-Not everyone thinks coffee makes the world go round, but even coffee haters will love this homework.
-Imagine you are in a coffee shop

-Prepare a conversation with the server or cashier.
-Use interesting phrases or

INPUTS

@h_id

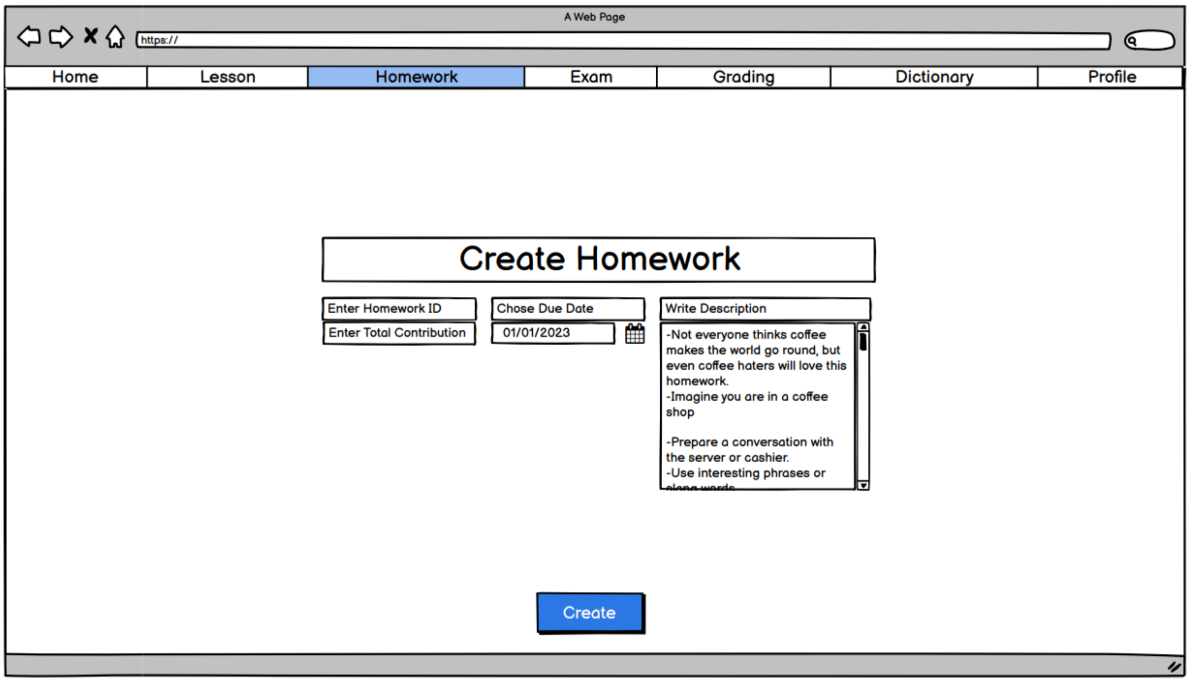
PROCESS

On this page, teachers can assign homework to students. In description part, explanation of the homework will be shown. Teachers will choose the ID of homework, they will see the due date for homework and total contribution. When they press the “Assign” button, information will be stored in the database. After pressing the “Create homework” button, teachers will reach the create homework page.

SQL QUERY

When assign button is pressed
INSERT INTO assign(h_id, s_id)
VALUES (@h_id, @s_id, @u_id)

6) CREATE HOMEWORK PAGE FOR TEACHER



-Not everyone thinks coffee makes the world go round, but even coffee haters will love this homework.
-Imagine you are in a coffee shop

-Prepare a conversation with the server or cashier.
-Use interesting phrases or

INPUTS

@h_id, @due_date, @total_contribution, @description

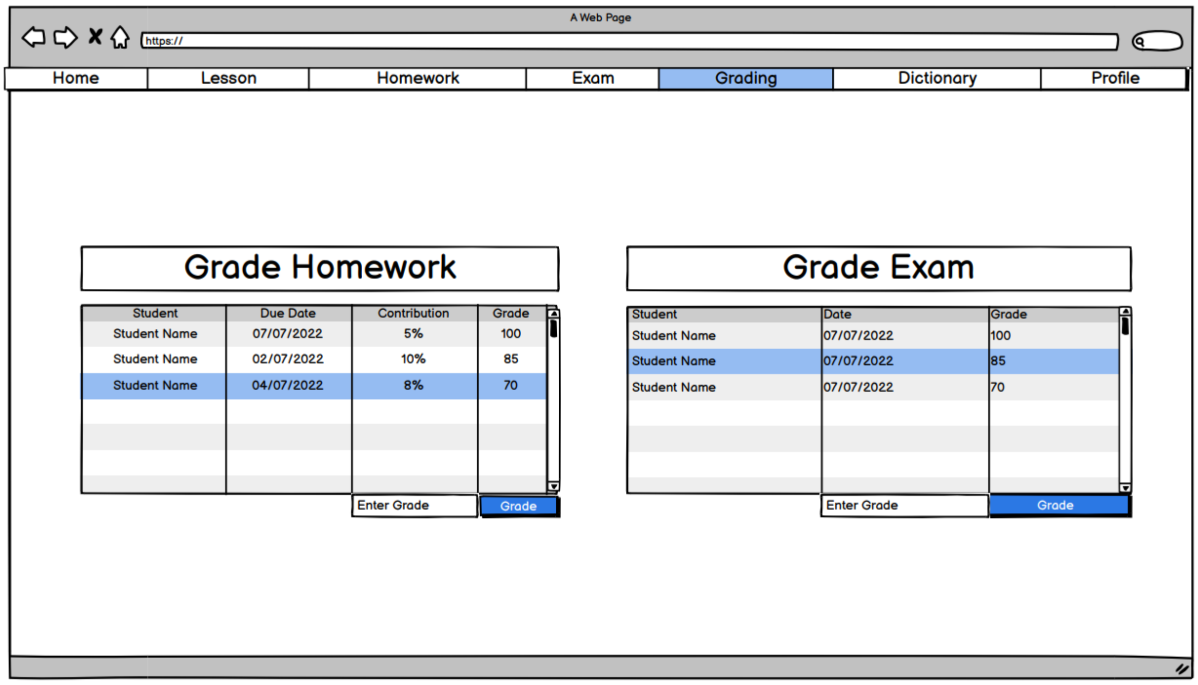
PROCESS

On this page, teachers can create homework for students. In the description part, teachers explain the homework that they want to create. Teachers will enter the ID of homework, a due date for homework and enter total contribution to assign the homework. When they press the “Done” button, information will be stored in the database.

SQL QUERY

When Done button pressed,
INSERT INTO Homework(h_id, due_date, total_contribution, description)
VALUES(@h_id, @due_date, @total_contribution, @description)

7) GRADE PAGE FOR TEACHER



Grade Homework

Student	Due Date	Contribution	Grade
Student Name	07/07/2022	5%	100
Student Name	02/07/2022	10%	85
Student Name	04/07/2022	8%	70

Grade Exam

Student	Date	Grade
Student Name	07/07/2022	100
Student Name	07/07/2022	85
Student Name	07/07/2022	70

INPUTS

@grade_exam, @grade_homework

PROCESS

On this page teacher can see the students who have assigned homeworks, and can give grades to the students

SQL QUERY

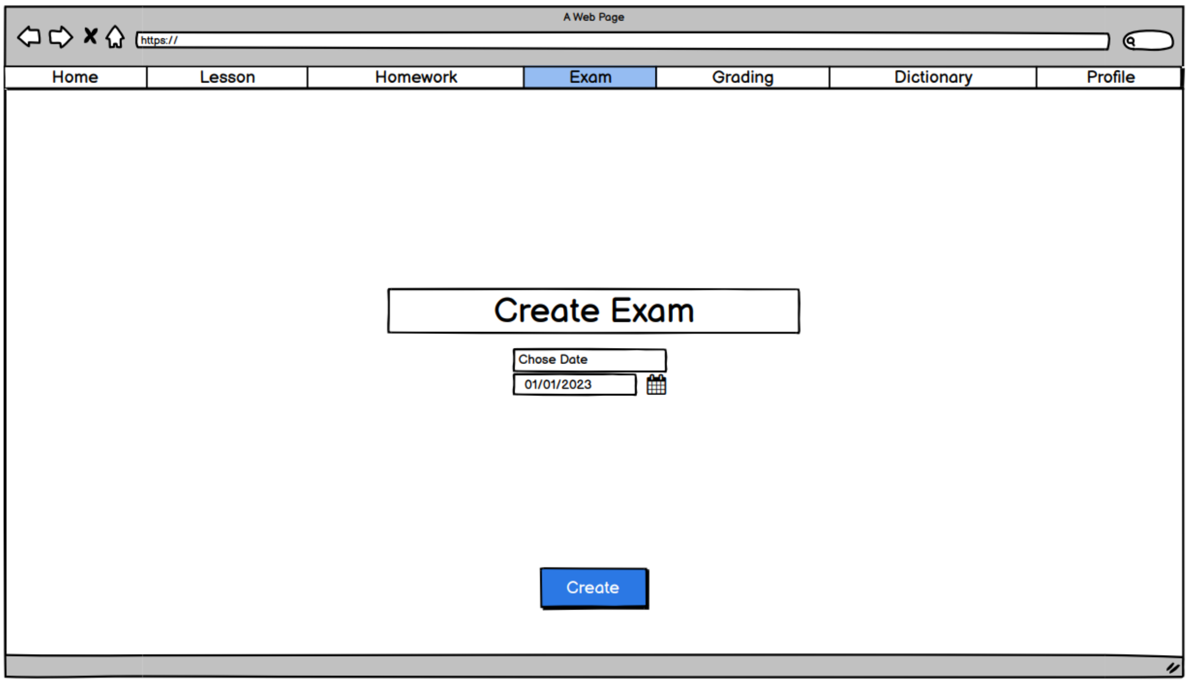
SELECT s.name, h.due_date, h.total_contribution, g.grade
FROM Student s, Teacher t, Grades g, Homeworks h
WHERE g.h_id = h.h_id and g.t_id = t.u_id and g. s_id = s.u_id

When Grade_homework button pressed,
INSERT INTO grades(t_id,s_id, h_id, grade)
VALUES(@t_id, @s_id, @h_id, @grade)

SELECT s.name, e.date, g.grade
FROM Student s, Teacher t, Grades_exam g, exam e
WHERE g.e_id = e.e_id and g.t_id = t.u_id and g. s_id = s.u_id

When Grade_exam button pressed,
INSERT INTO grades_exam(t_id,s_id, e_id, grade)
VALUES(@t_id, @s_id, @e_id, @grade)

8) CREATE EXAM PAGE FOR TEACHER



INPUTS

@date

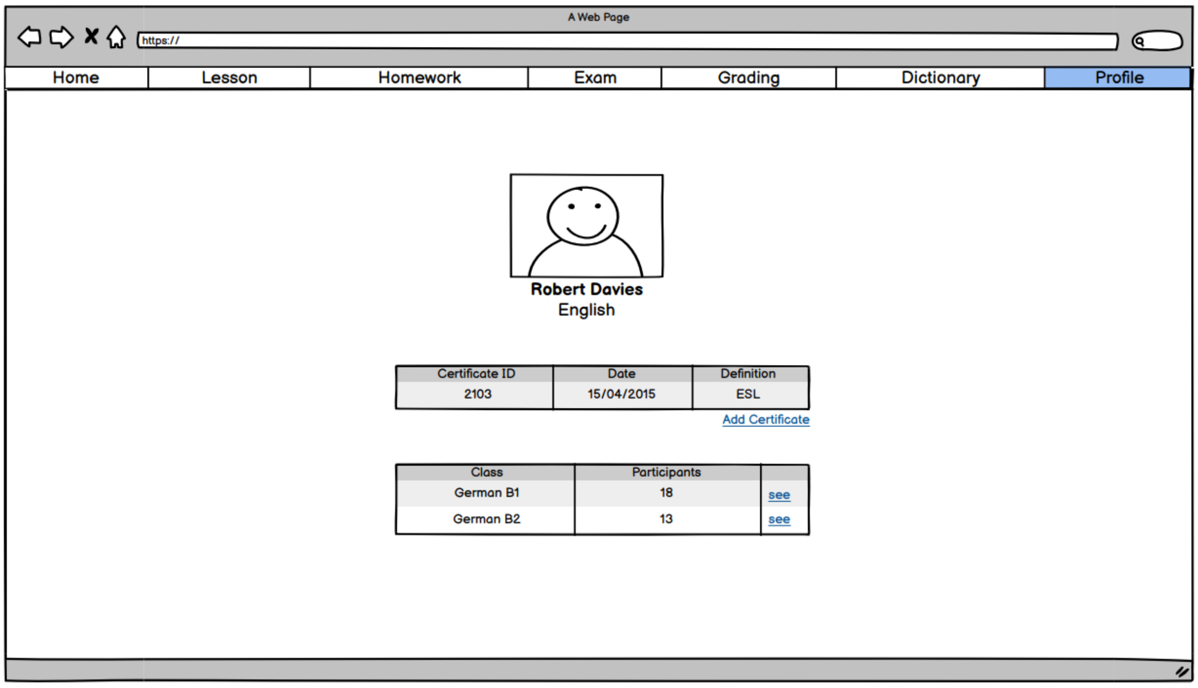
PROCESS

On this page teacher can see the students who have assigned homeworks, and can give grades to the students

SQL QUERY

When Create button pressed,
INSERT INTO Exam(date)
VALUES(@date)

9) TEACHER PROFILE PAGE



Certificate ID	Date	Definition
2103	15/04/2015	ESL

Class	Participants	
German B1	18	see
German B2	13	see

INPUTS

-

PROCESS

On this page, teachers can see their profile. This profile will include information about the teacher. Teacher can see his/her classes' information such as class name and participant count. The teacher can see his/her certificate(s). With clicking the "Upload a Certificate" button, s/he can upload his/her new certificate. When pressing the "see" button, the teacher can reach the lesson information page related to the chosen lesson. Moreover, the teacher can also see his/her profile picture and his/her name.

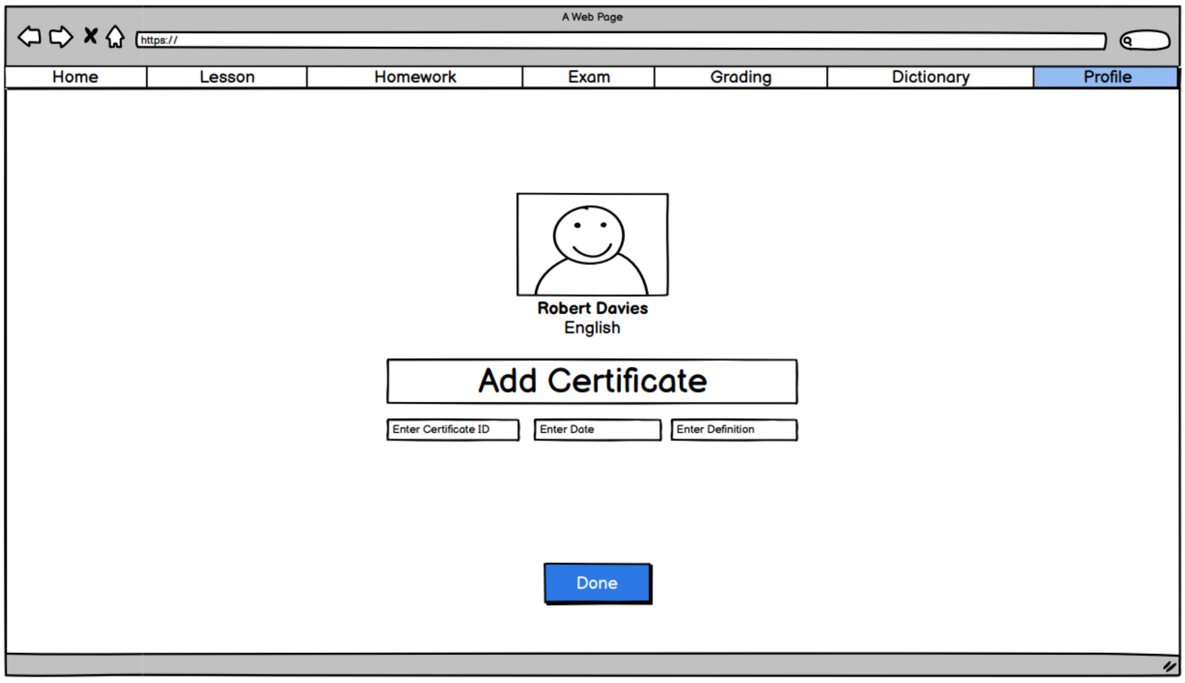
SQL QUERY

For teacher's name,
SELECT t.name
FROM Teacher t
WHERE t.u_id = @u_id;

For certificate,
SELECT c.certificate_id, c.date, c.definition,
FROM Certificate c, Teacher t
WHERE c.u_id = t.u_id

For class information,
SELECT l.l_id, count(*)
FROM lesson l, give g
WHERE l.u_id = t.u_id AND t.u_id = @u_id
GROUP BY (l.l_id)

10) ADD CERTIFICATE PAGE FOR TEACHER



INPUTS

@id, @date, @def

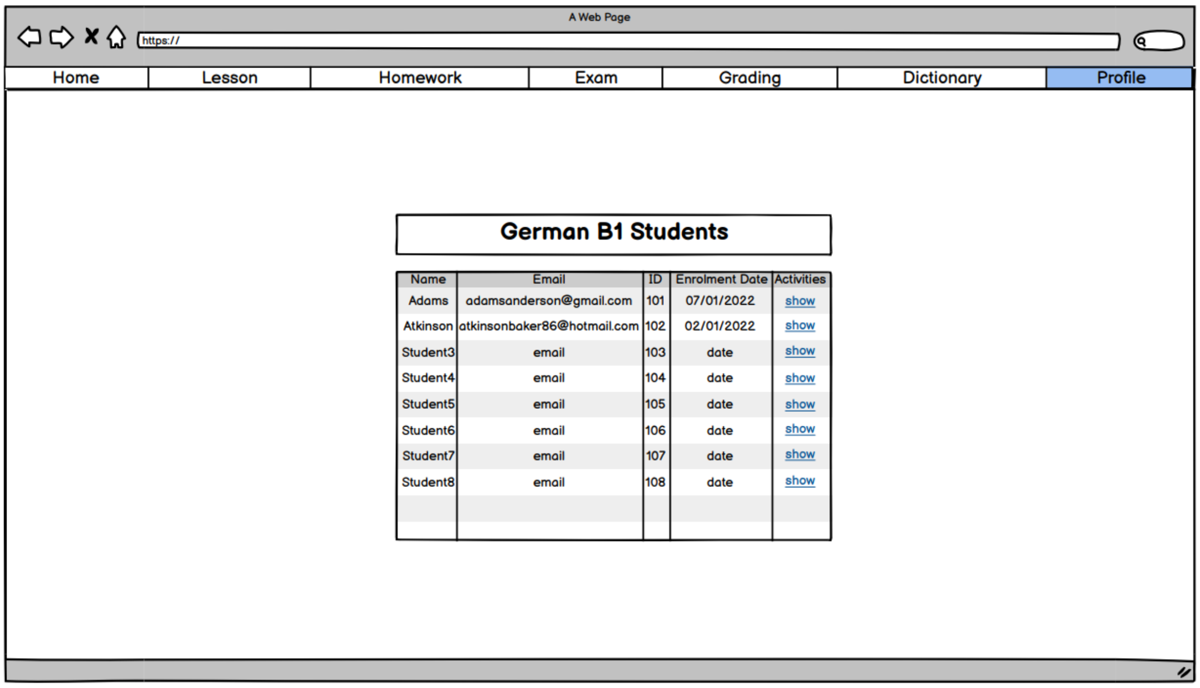
PROCESS

On this page teacher can add new certificate to their profile

SQL QUERY

When Add button pressed,
INSERT INTO Certificate(certificate_id,date, definition)
VALUES(@id, @date, @def, @u_id)

11) CLASS PAGE FOR TEACHER



German B1 Students

Name	Email	ID	Enrolment Date	Activities
Adams	adamsanderson@gmail.com	101	07/01/2022	show
Atkinson	atkinsonbaker86@hotmail.com	102	02/01/2022	show
Student3	email	103	date	show
Student4	email	104	date	show
Student5	email	105	date	show
Student6	email	106	date	show
Student7	email	107	date	show
Student8	email	108	date	show

INPUTS

-

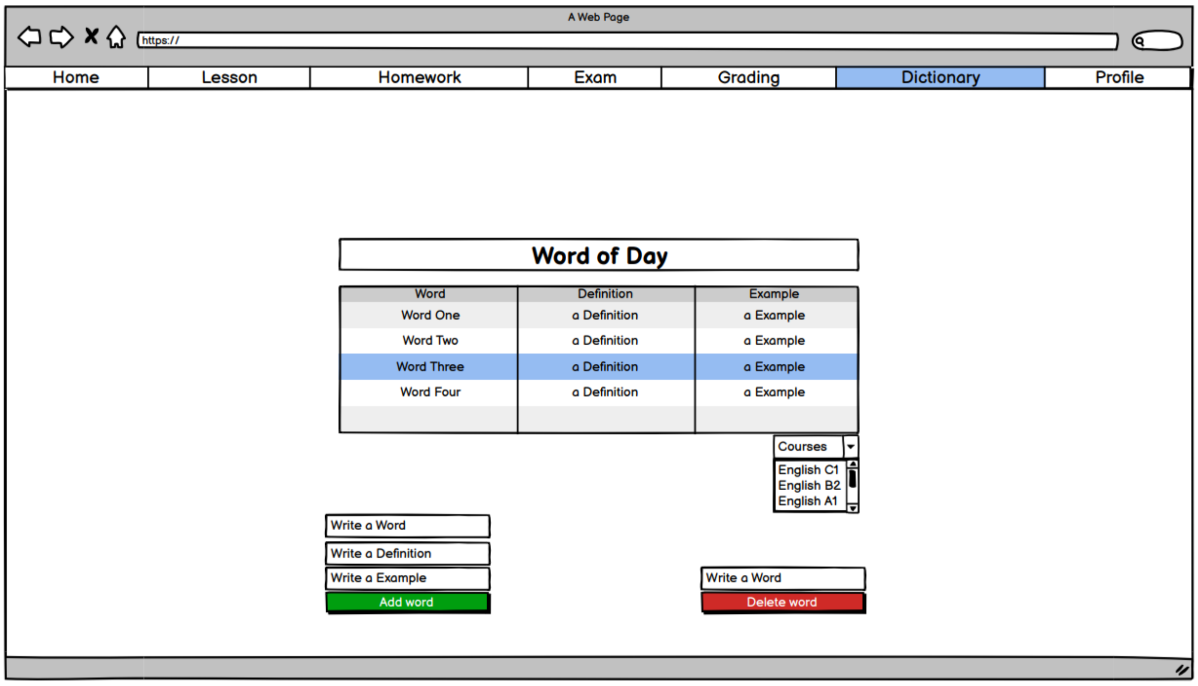
PROCESS

On this page, teachers can see all student information in her/his class. Teacher will get his/her students' name, email, ID and enrollment date. When "show" button is pressed, they will reach activities.

SQL QUERY

For class page,
SELECT s.name, s.email, s.u_id, s.enrollment_date
FROM Student s, enrolls, e
WHERE s.u_id = e.u_id AND e.l_id = @l_id

12) DICTIONARY PAGE FOR TEACHER



Word	Definition	Example
Word One	a Definition	a Example
Word Two	a Definition	a Example
Word Three	a Definition	a Example
Word Four	a Definition	a Example

INPUTS

@lesson, @new_word, @new_def, @new_example, @del_word

PROCESS

On this page, teachers can arrange “Word of day” related to each course by choosing the lesson. They can add a new word with pressing the “Add word” button. Moreover, they can delete the word that is selected.

SQL QUERY

For dictionary page

SELECT w.word, w.definition, w.example_usage
FROM Words w, taught_in t, lesson l, defines d
WHERE w.w_id = t.w_id AND t.l_id = l.l_id AND d.w_id = w.d_id AND d.u_id = @u_id

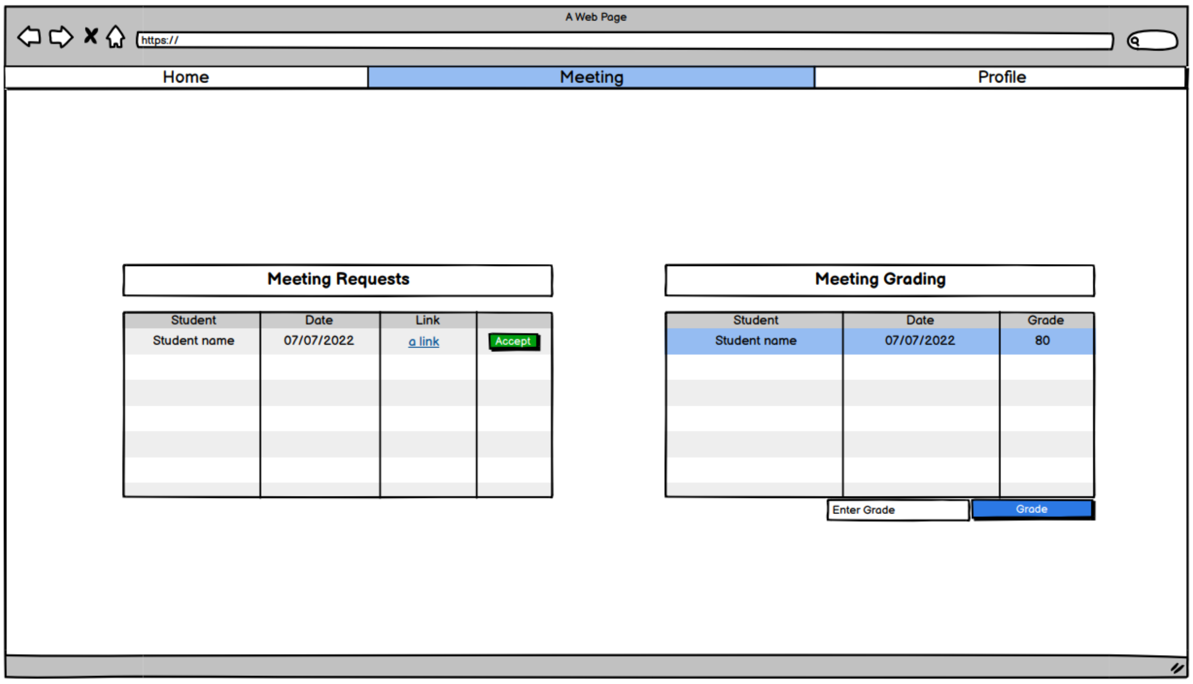
Inserting a word

INSERT INTO Words WHERE words.word = @del_word

Deleting a word

DELETE FROM Words
VALUES(@new_word, @new_def, @new_example)

13) MEETING PAGE FOR NATIVES



Student	Date	Link	
Student name	07/07/2022	a link	Accept

Student	Date	Grade
Student name	07/07/2022	80

INPUTS

@grade, @link

PROCESS

On this page, Native users can see meeting requests from students, students' name, date and the meeting link. When they click the "accept" button, they will go to the meeting. Moreover, after meeting Native users can grade with pressing the "Grade" button. Natives can also see the students' name, date, the meeting link and grade in the grading table.

SQL QUERY

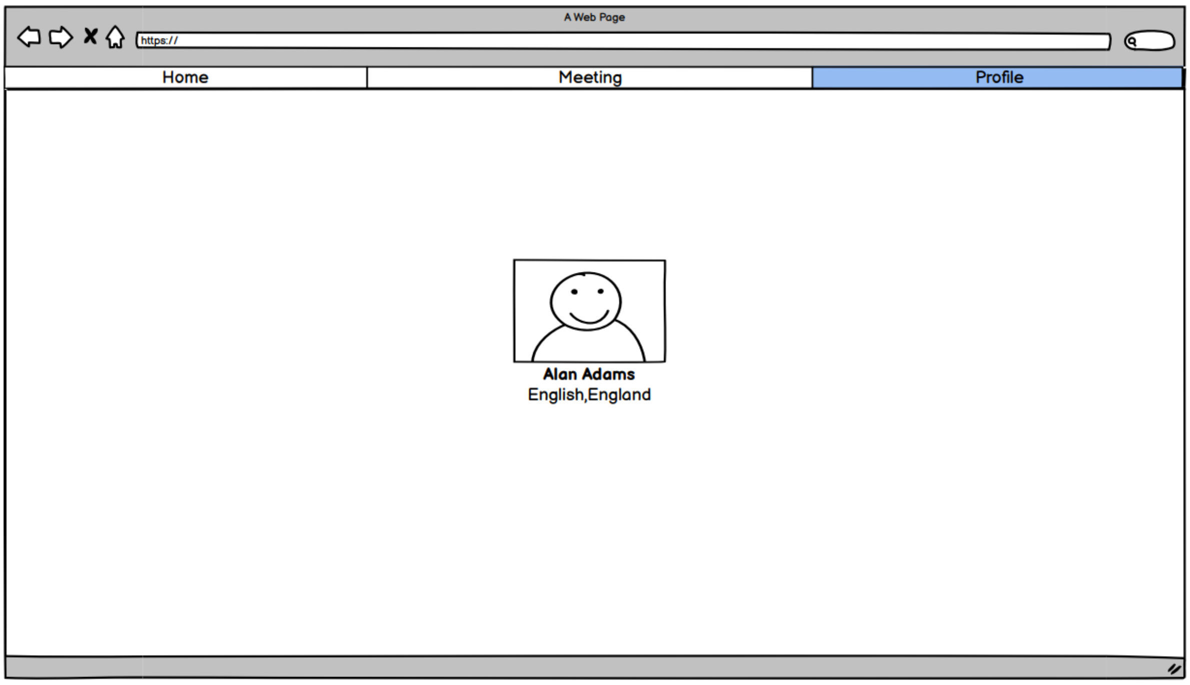
For grading table,

```
SELECT s.name, m.date, g.grade
FROM Student s, request r, Meeting m, Native n, grades_meeting g
WHERE s.s_id = r.s_id AND n.n_id = r.n_id AND m.m_id = r.m_id
AND g.m_id = m.m_id AND g.n_id = n.n_id
```

For request table,

```
SELECT s.name, m.date,m.link
FROM Student s, request r, Meeting m, Native n
WHERE s.s_id = r.s_id AND n.n_id = r.n_id AND m.m_id = r.m_id
```

14) PROFILE PAGE FOR NATIVES



INPUTS

-

PROCESS

On this page, users can see their profile picture, their name and their country.

SQL QUERY

```
SELECT n.name, n.country
FROM Native n, Native na
WHERE n.n_id = na.n_id
```

15) HOME PAGE FOR STUDENT

INPUTS

@language

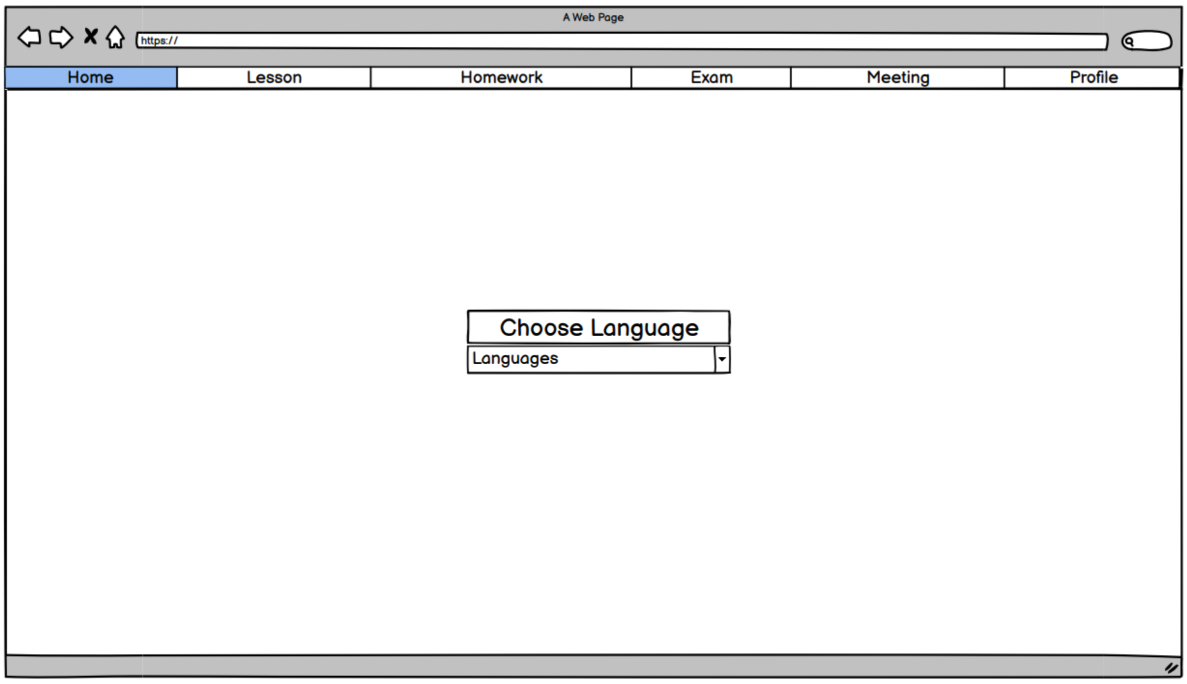
PROCESS

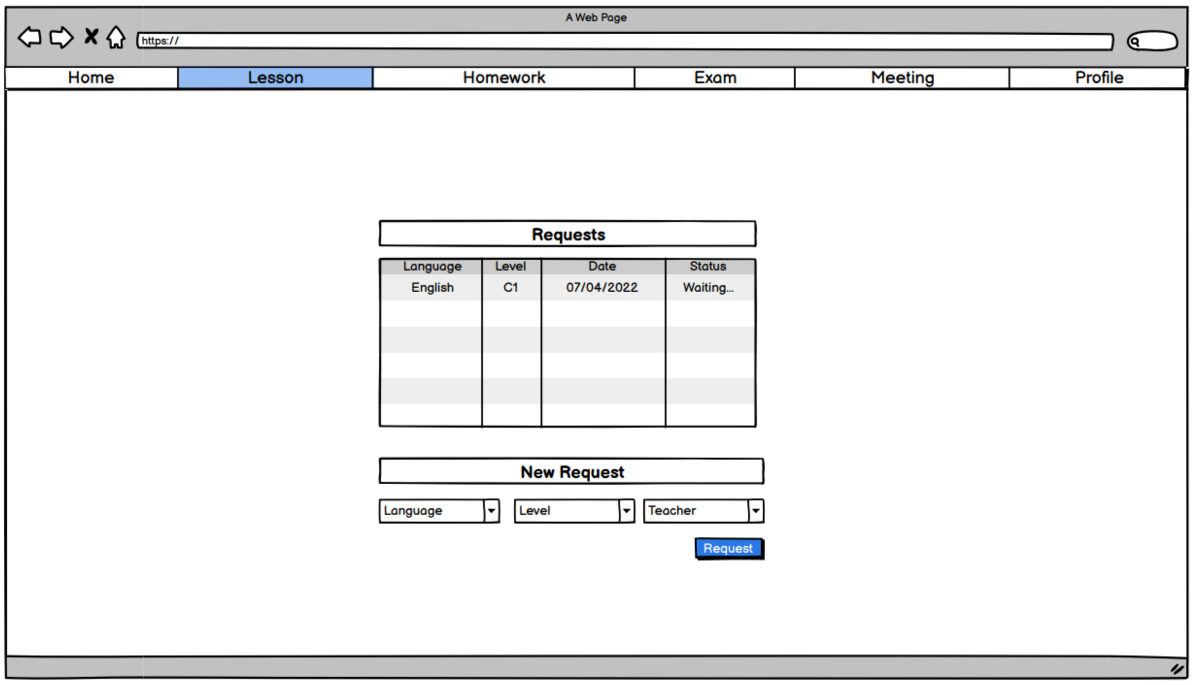
On this page, Student users should choose the language that they want to learn. After they choose the language they can move to the other pages.

SQL QUERY

```
SELECT DISTINCT name
FROM Language
```

16) LESSON PAGE FOR STUDENT





Language	Level	Date	Status
English	C1	07/04/2022	Waiting...

INPUTS

@language
@level
@t.name

PROCESS

On this page, Student users can create new lesson requests by choosing a language, his/her level, and a teacher. After pressing the “request” button, the request will be stored in the database. Students can see their requests and their status on the request table.

SQL QUERY

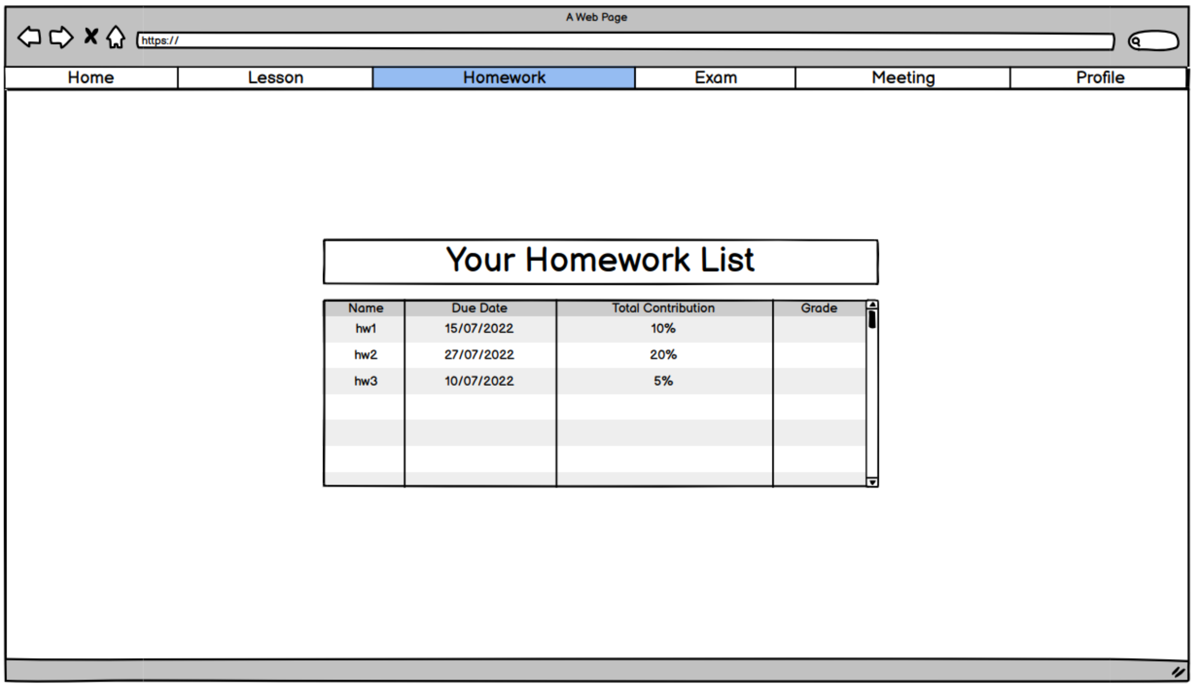
For request table

```
SELECT l.name, le.level, s.enrollment_date, e.status
FROM Language l, learns le, Student s, enrolls e, lesson les
WHERE l.name = le.name AND s.s_id= le.s_id AND s.s_id = e.s_id AND les.l_id =
e.l_id
```

For new request

```
SELECT l.name, le.level, t.name
FROM Language l, learns le, Teacher t
```

17) HOMEWORK PAGE FOR STUDENT



Your Homework List

Name	Due Date	Total Contribution	Grade
hw1	15/07/2022	10%	
hw2	27/07/2022	20%	
hw3	10/07/2022	5%	

INPUTS

-

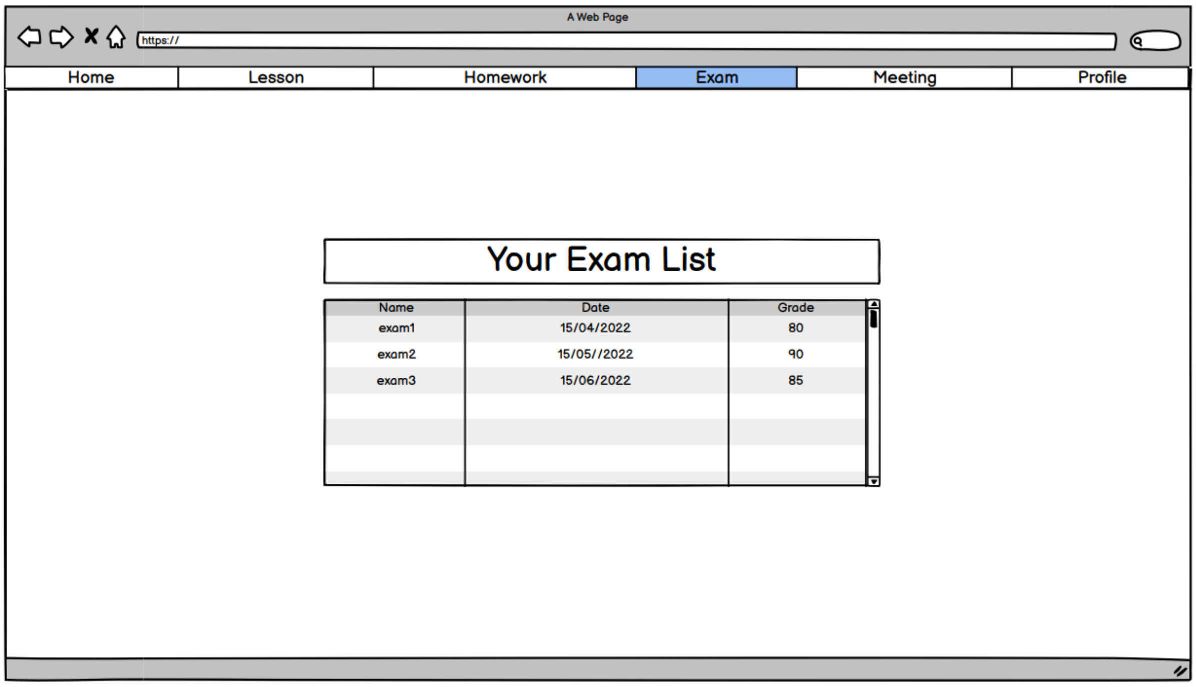
PROCESS

On this page, Student users can see their assigned homeworks and their due dates, homeworks' total contributions and their grades.

SQL QUERY

```
SELECT h.h_id, h.due_date, h.total_contribution, g.grade
FROM Homeworks h, Grades g
WHERE g.h_id = h.h_id
```

18) EXAM PAGE FOR STUDENT



Your Exam List

Name	Date	Grade
exam1	15/04/2022	80
exam2	15/05//2022	90
exam3	15/06/2022	85

INPUTS

-

PROCESS

On this page, Student users can see their exam information and their date's, and their grades.

SQL QUERY

```
SELECT e.e_id, e.date, g.grade
FROM Exam e, grades_exam g
WHERE g.e_id = e.e_id
```

19) LESSON PAGE FOR ADMIN

INPUTS

-

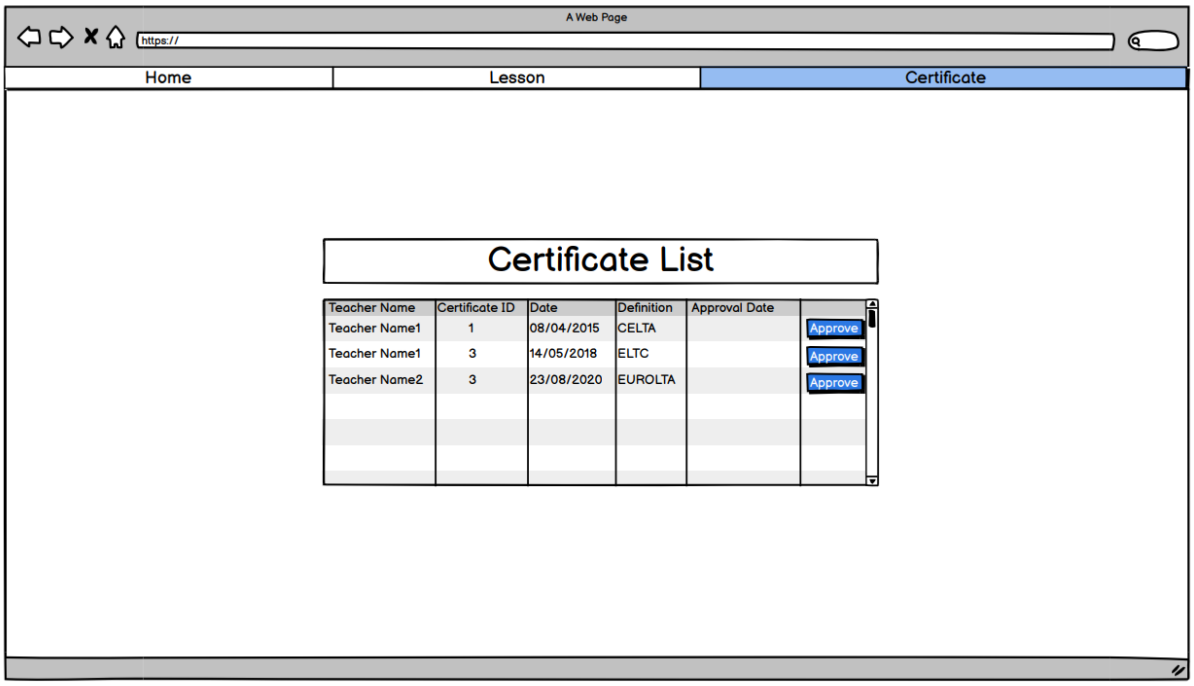
PROCESS

On this page, Admin users can see the Teacher list. This table will include teachers' name, language that the teacher teaches, student count in this class and the average grade.

SQL QUERY

```
SELECT t.name, l.name, count(s_id), avg(grade)
FROM Teacher t, Language l, Student s, learns le, grades_exam g, Knows k
WHERE t.t_id = k.u_id AND l.name = le.name AND s.s_id = le.s_id AND s.s_id =
g.s_id AND k.name = l.name
GROUP BY (t.name,l.name)
```

20) CERTIFICATE LIST PAGE FOR ADMIN



Certificate List

Teacher Name	Certificate ID	Date	Definition	Approval Date	
Teacher Name1	1	08/04/2015	CELTA		Approve
Teacher Name1	3	14/05/2018	ELTC		Approve
Teacher Name2	3	23/08/2020	EUROLTA		Approve

INPUTS

-

PROCESS

On this page, Admin users can see Certificate list. Admin can reach some information such as teacher name, certificate ID that belongs to the teacher, certificate date, definition of certificate, status of the certificate. When the “Approve” button is pressed, the approval date will be written on the approval date column.

SQL QUERY

```
SELECT t.name, c.certificate_id, c.date, c.definition, a.approval_date
FROM Certificate c, Teacher t, approves a
WHERE c.t_id = t.t_id AND a.certificate_id = c.certificate_id AND a.date = c.date
AND a.definition = c.definition AND a.t_id = c.t_id
```

ADVANCE DATABASE COMPONENTS

5.1 Views

Approved_lessons: This view is used for listing the approved enrollments by the instructor

```
CREATE VIEW approved_lesson(name, l_name)
AS SELECT s.name, l.l_name
FROM (enrolls e, student s, lesson l)
WHERE e.u_id = s.u_id AND e.l_id = l.l_id AND e.status = “approved”
```

Analytics_lessons: This view will be used for listing the analytics for every lesson in the system

```
CREATE VIEW analytics_lesson( t_name, l_name, count, avg_grade)
AS SELECT t.name, l.name, count(s_id), avg(grade)
FROM Teacher t, Language l, Student s, learns le, grades_exam g, Knows k
WHERE t.t_id = k.u_id AND l.name = le.name AND s.s_id = le.s_id AND
        s.s_id = g.s_id AND k.name = l.name
GROUP BY (t.name, l.name)
```

enroll_infos: This view is used for listing the information of courses of a corresponding teacher.

```
CREATE VIEW enroll_infos(l_id, count) AS (
SELECT l.l_id,  count( *)
FROM lesson l, give g
WHERE l.u_id = t.u_id AND t.u_id = @u_id
GROUP BY (l.l_id)
)
```

5.2 Triggers

- When a enrollment request is accepted by a teacher, corresponding relations,which are enrolls, request_class, and activities, will be updated
- When a homework assigned to a student, student activity relation and have relation will be updated
- When a certificate approved, corresponding relation which is approves will be updated
- When a teacher give grade to a homework, corresponding relations are updated which is grades
- When a teacher give grade to a exam, corresponding relations are updated which is grades_exam

- When a native give grade to a meeting, corresponding relations are updated which is grades_meeting

5.3 Constraints

- Student can not enroll in a same course twice
- Student can not request meeting from a native if the native holds a meeting in that time slot
- Student can not enroll in a course which is higher than his level in that language
- Teacher can not create a lesson which is higher than his/hers eligible level

5.4 Reports

Teachers can see how many students are enrolled in their lessons.

```
CREATE VIEW enroll_infos(l_id, count) AS (
SELECT l.l_id,  count( *)
FROM lesson l, give g
WHERE l.u_id = t.u_id AND t.u_id = @u_id
GROUP BY (l.l_id)
)
```

IMPLEMENTATION

We used the dijkstra server's InnoDB to implement our database system. Furthermore, PHP, JavaScript, HTML and CSS used for UI (User Interface) and our website's functionalities.

PROBLEMS FACED DURING IMPLEMENTATION

We faced with several problems during implementation of the system. First of all we created the user id's in the user, employee, teacher, native, student, and admin tables as characters. By changing that to integer values and defining them as a auto increment we solve our database side problem in registration. However, we still got errors since we could not manage to pull the user type from frontend successfully so we decided to divide registration page into three parts, for student, native, and teacher. Also we added a new attribute to the user which store what type of user is it.

USER MANUAL

User Manuel for Login and Register Pages

In login page, users are waited to enter their username and password regardless of the user type. In order to login, user should hit the register button and than enter an username, password, email, and their user type which could be teacher, student or native. After hitting the register button they will redrected to the login page.

After login, teachers will be redirected to the lesson page which displays the enrollment requests and a create lesson button. Natives will be redirected to meeting page which displays the meeting requests and meeting grading tables. Students will be directed to home page where they can choose a learning language, and finally admin will be redirected to lesson page where they can view the analysis of the lessons.

User Manuel for Teacher

After login, teacher will be redirect to the lesson page where they can see the class request which sended by the students. They can accept the request by clicking on the accept button right next to the request and also they can create new lesson by clicking to the create lesson page. After the create lesson button clicked, teacher will be redirected to the create lesson page where they can choose the language and level for their new lesson.

Also teacher can assign homework and create homework in homework page. In the homework page, teacher can view all the students that are enrollen into their courses and can assign homework by clicking onto the assign button in the corresponding column. After the click action, teacher will be directed to the assign homework page and by selecting the pre-created homework and specifying a due date for the homework, he/she can assign a homework. If teacher wants to create a new homework, he/she should click on the create a homework button which will redirect them to the create homework page. In this page they can create homework by entering a due date, total contribution and a description.

In the Exam page, teacher can create an exam by specifying a due date. In the Grade page, teacher can grade both exam and a homework of a student. To do that they have to provide an exam id or homework id alongside with a grade to that corresponding assignment. Int the dictionary page, teacher can see their already created words also they can create new words with a definition and provide an example usage of that word. They can also delete a word from their dictionary. In the profile page they can see their lessons and how many student enrolled into that course. By clicking the button right next to a lesson, they can view the class. The also can upload a certificate to their profile in profile page.

User Manuel for Native

After login, native will be redirect to the meeting page where they can see the meeting request which sended by the students and also they can grade the accepted meetings. They can accept the request by clicking on the accept button right next to the request and for grading they have to provide the meeting id and grade for that meeting. Also natives can view their profiles in their profile page

User Manuel for Student

After login, student will be redirected to the home page where they can select a language to learn. After selecting the language, they will be redirected to the lesson page where they can see their unapproved class requests. Also, they can add a new request by choosing a level and a teacher on this page. On the homework page, the student can view their homework list, their names, description, and total contribution. Also, they can view the grade that is given to homework if it's eligible.

On the exam page, they can view their exams, dates, and a given grade to that exam if it's eligible. On the meeting page, they can send a meeting request to a native by selecting a native speaker and a date.

User Manuel for Admin

After login, admin will be redirected to lesson page where they can view the analytics of the lessons. In the certificate page, they will see the unapproved certificates and their details. By clicking the corresponding approve, they can approve a certificate.

Work Done

In this project, all the group members deal with both the front-end and the back-end of the project. Detailed work-sharing given below

Selahattin Cem Öztürk:
- Creating the ER diagram
- Writing the create table queries
- Writing the Views
- Writing the frontend of the teacher side
- Writing the login and register pages' front-end and back-end
- Writing the sql queries for teacher
- Finding the extra features

Lamia Başak Amaç:
- Writing the create table queries
- Writing the student register page's front-end and back-end
- Writing the frontend of the student side
- Writing the sql queries for student
- Connection to the database

Alper Kandemir:
- Creating mockups
- Writing the lnative register page's front-end and back-end
- Writing the front-end of the native side
- Writing the sql queries for native
- Connection to the database