UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Web Applications A.Y. 2021-2022

Homework 1 – Server-side Design and Development

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: 22 April, 2022

Group TORE	Project We are implementing a Learning Management System (LMS)	
Last Name	**First Name**	**Badge Number**
Ahmed	Saad	2044003
Bahrami	Sepide	2043887
Hansen	Marit Fredrikke	2062124
Rao	Abdul Moeed	2044017
Rehman	Abdul	2048675
Sohail	Mohammad Muzammil	2043886
Soomro	Dodo Khan	2051143
Sultanova	Aliia	2041422
Zabalawi	Iyad	2041407

1 Objectives

TORE is a Learning Management System (LMS). It's purpose to empower departments with training and development for the learners, to provide ways of professor-student communication, sharing the information and making the studying process easier. All courses are offered through the platform. We would like to represent an application because audiences take most naturally to an LMS when it adopts modern solutions that people are used to working with. More traditional modes like paper tests and in-person instructor-led training just don't hit the mark for modern learners these days. Our team will include crucial part for any LMS as application management of profiles, signing up, logging in, enrolling/quitting a course, uploading files, the ability to register courses and to share course materials, to make announcements.

2 Main Functionalities

TORE- Learning Management System will allow students and teachers to interact with each other in an easy way to share study material and enquiries creating a central platform. Major functionalities will be as following

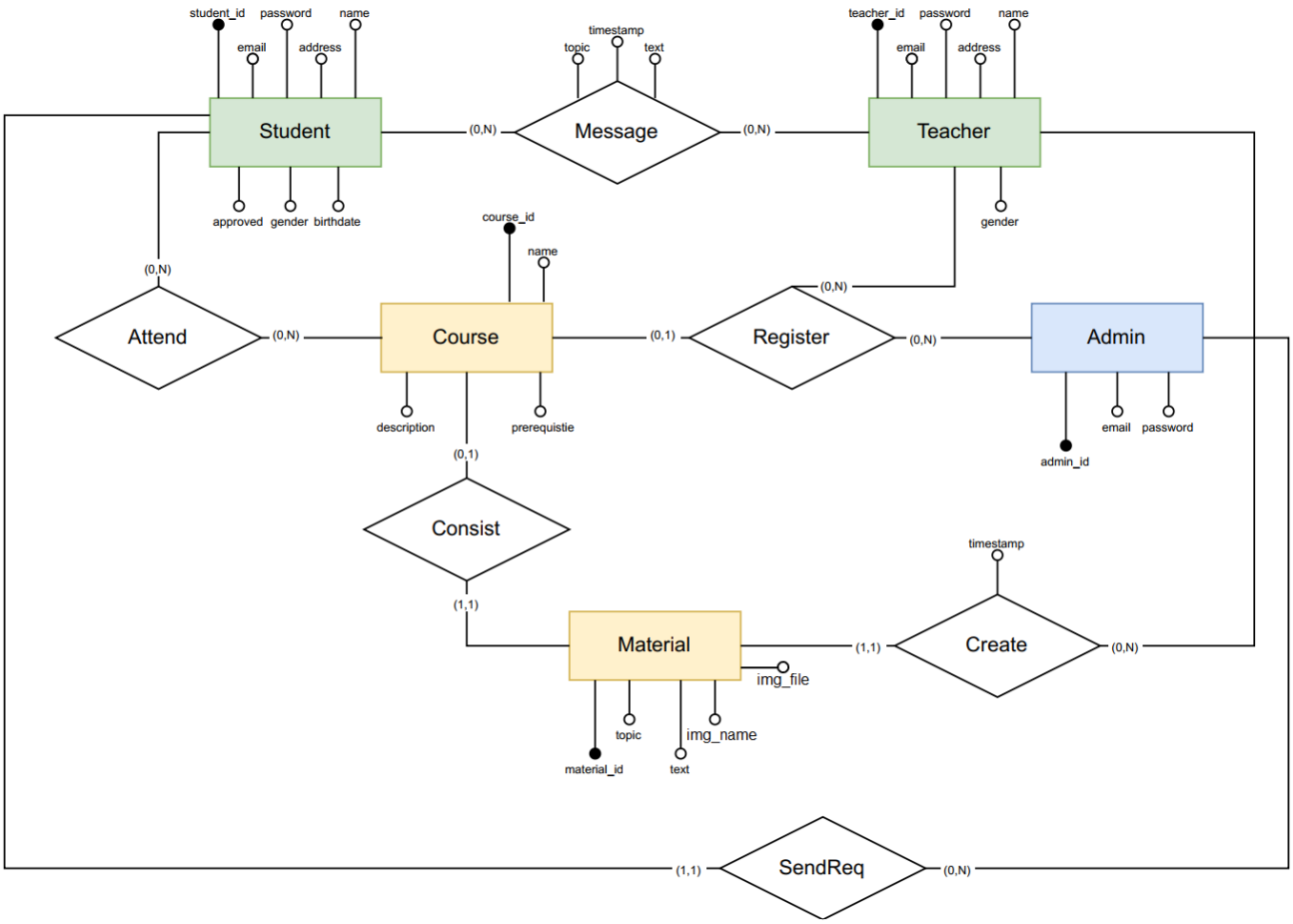
1. **Main Area**. Accessible to everyone interested in knowing about the system and will be used for registering new users and signing up existing users.
2. **Admin Area**. This area will only be accessible to users with admin rights. It will hold following functionalities.
    - Approval of all users registering for the system.
    - Maintenance of profiles according to type of account.
    - Maintenance of Database.
    - Assignment of tasks to teacher accounts.
    - Maintenance of subject allotment to teachers.
3. **Faculty Area**. This area will be restricted for the use of teachers only and will perform following
    - Each teacher will be able to access his own specified subjects.
    - Teacher will have the authority to manage students and course contents.
    - Management of student affairs including enrolment.
    - Replying to the comments/ questions made by the students.
4. **Students Area**. This area will be accessible to all users registered as students and will offer following functionalities
    - Enrolment to subjects/ courses.
    - Viewing contents shared by teachers of each course.
    - Submitting questions/ commenting on course contents.

Major roles of different users will be following

1. Admin. Responsible for over all management and approval of all users registering.
2. Teachers. Will be able to deal with subjects allotted by admin and enrol students.
3. Students. Can register and enrol into subjects offered after approval from admin.

3 Data Logic Layer

3.1 Entity-Relationship Schema



The ER schema contains 5 main entities:

- Admin: This entity consists of the system admin and its primary key is an integer with auto-increment. Also, it has email(username) and password for logging into system as the admin user.
- Student: This entity is another type of the available users in the system. The primary key is an integer with auto-increment and in addition, there are some attributes used for logging into system (email, password) and personal information (name, gender, birth-date, address) for registering in the system. There is another Boolean attribute named as 'approved' which its value is set to true after the admin accepts the student's registration request.
- Course: This entity is the collection of courses available in the system for the students to enrol in. The primary key is an integer with auto-increment and the other attributes are used to save some information about each course.
- Teacher: Another type of users is the teacher entity which are registered (signed up) into system by the admin. Then one or more courses can be assigned to them.

- Material: Each course may have some materials (homeworks, announcements, etc). Therefore, they are all kept by having an integer auto-increment id for each material and their topic, text, and attachments if available are saved. Attachment can only be of maximum one image per each material posted.

3.2 Other Information

The n-ary relationship between Admin, Teacher and Course allows the admin to register teachers in the database and to assign one or more courses to each of the teachers. Thus, there would be no conflicts since admin does the assignment task.
Also each student, after filling in the registration form must wait to be accepted by the Admin (approved=true) and then can log into the system, enrol in courses, send messages to teachers and see the course materials available.

4 Presentation Logic Layer

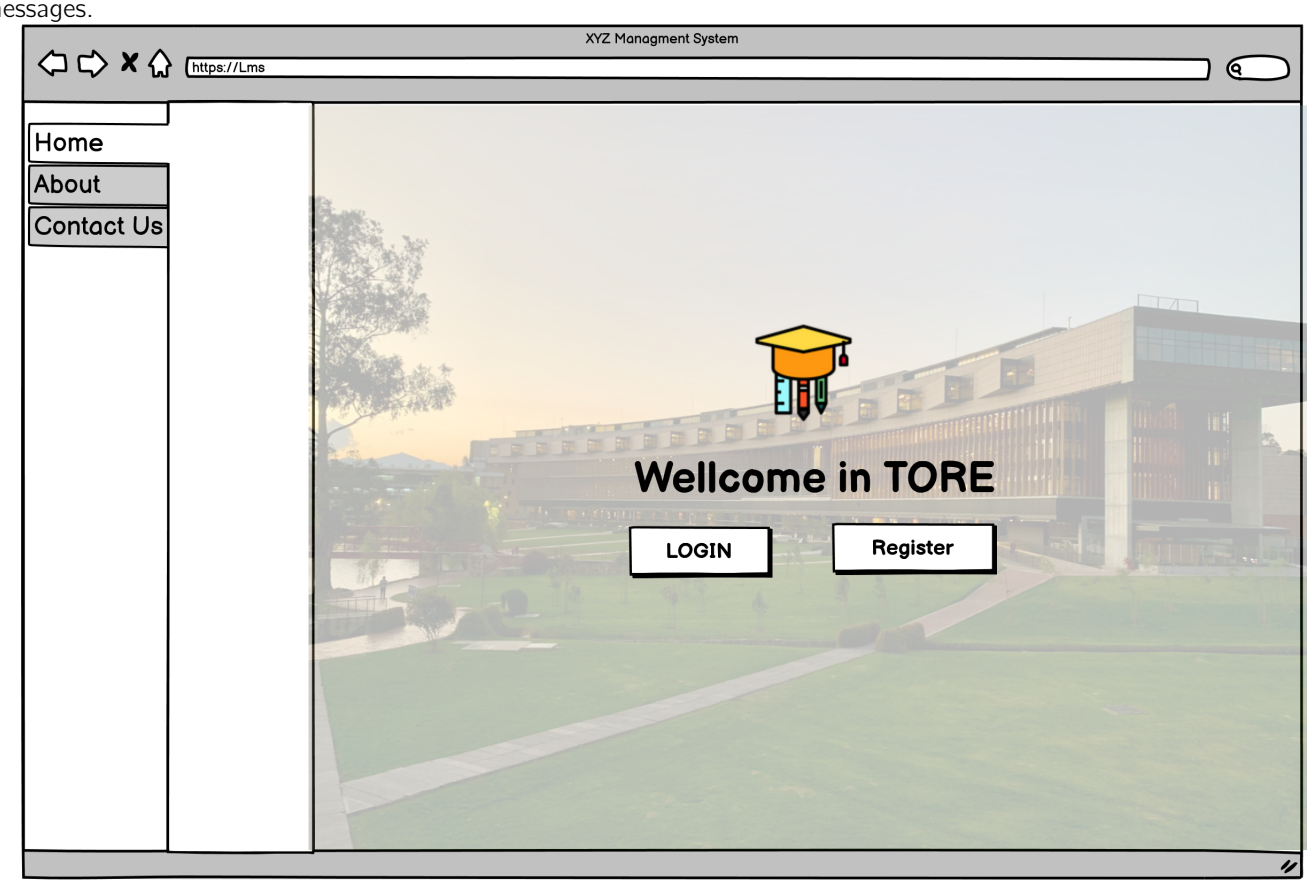
The website will be divided into the following pages:

- Homepage: contains the main area with the sections AboutUs and ContactUs. Allows users to login or register to the system through LOGIN and REGISTER buttons, respectively. Developed via jsp.
- Login page: allows the users login to the LMS. Developed via jsp.
- Registration page: allows students to register in the LMS.
- Administration page: allows to administrate users' accounts. Written in HTML.
- Teacher page: allows users to search for maintenance events. written in HTML with the support of javascript.
- List students page: allows to insert a new maintenance event. Written in HTML.
- List teachers page: allows to upload a file containing the measurements. Written in HTML.
- Teacher dashboard page: allows to redirect users toward pages handling specific resources. Written in HTML.
- Available teachers page:
- About Us page:
- Contact Us page:
- Post material page:
- RegisterWireframePage:
- RequestCoursePage:
- StudentDashboardPage:
- SendQueryTeacherPage:
- SubjectAreaPage:
- 2 pages each to handle insertion and update of parks, rides and models. Written in HTML.
- A support page which displays messages upon completion of specific tasks, such as the removal or update of resources.

4.1 Login Form

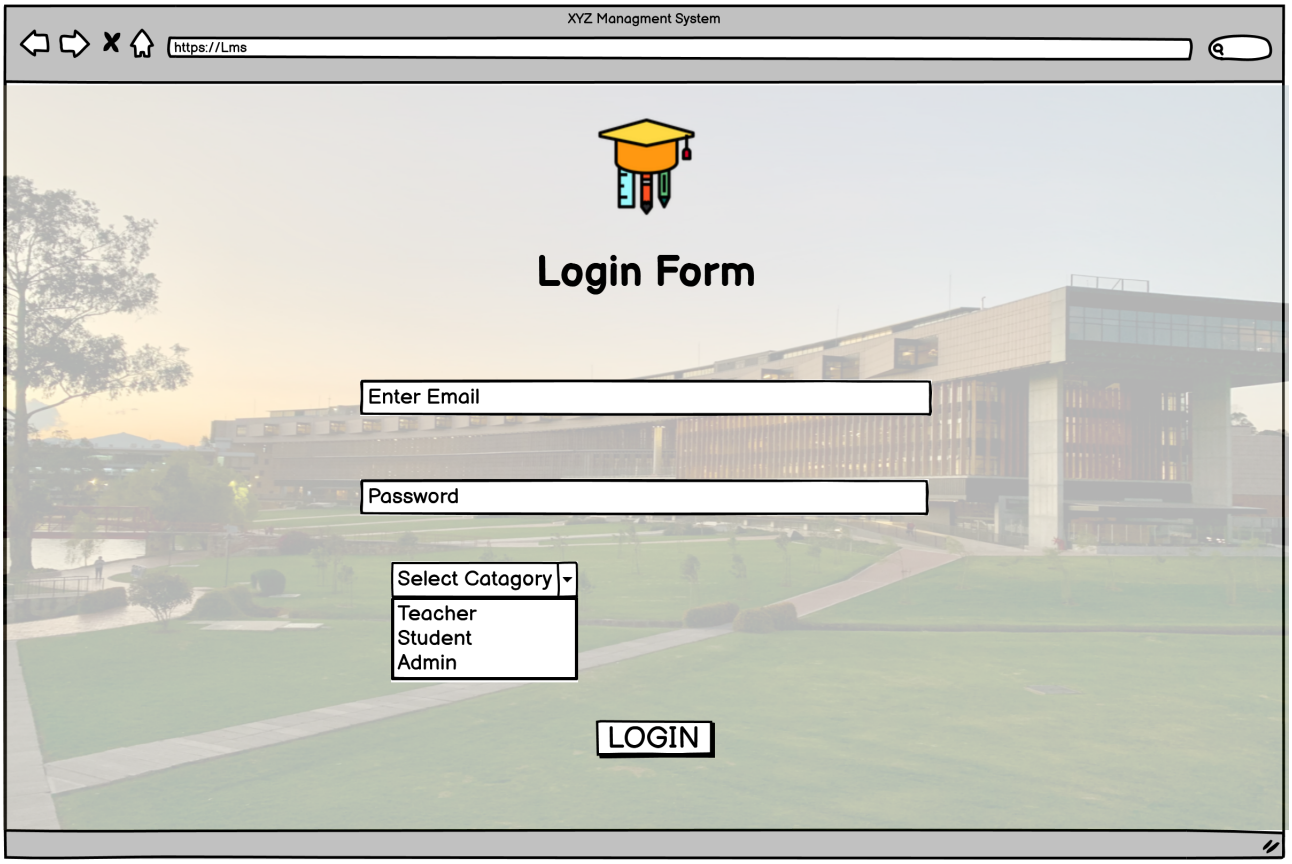
4.2 Homepage

The homepage contains the LOGIN button for the users who has already registered and using the system and the Register button for new users. Pressing the REGISTER button the user will go to the student registration page. The main area of the homepage will contain the information sections as About and Contact Us. There will be the description of the TORE project, contact links for our social media pages and the contact form for sending us messages.



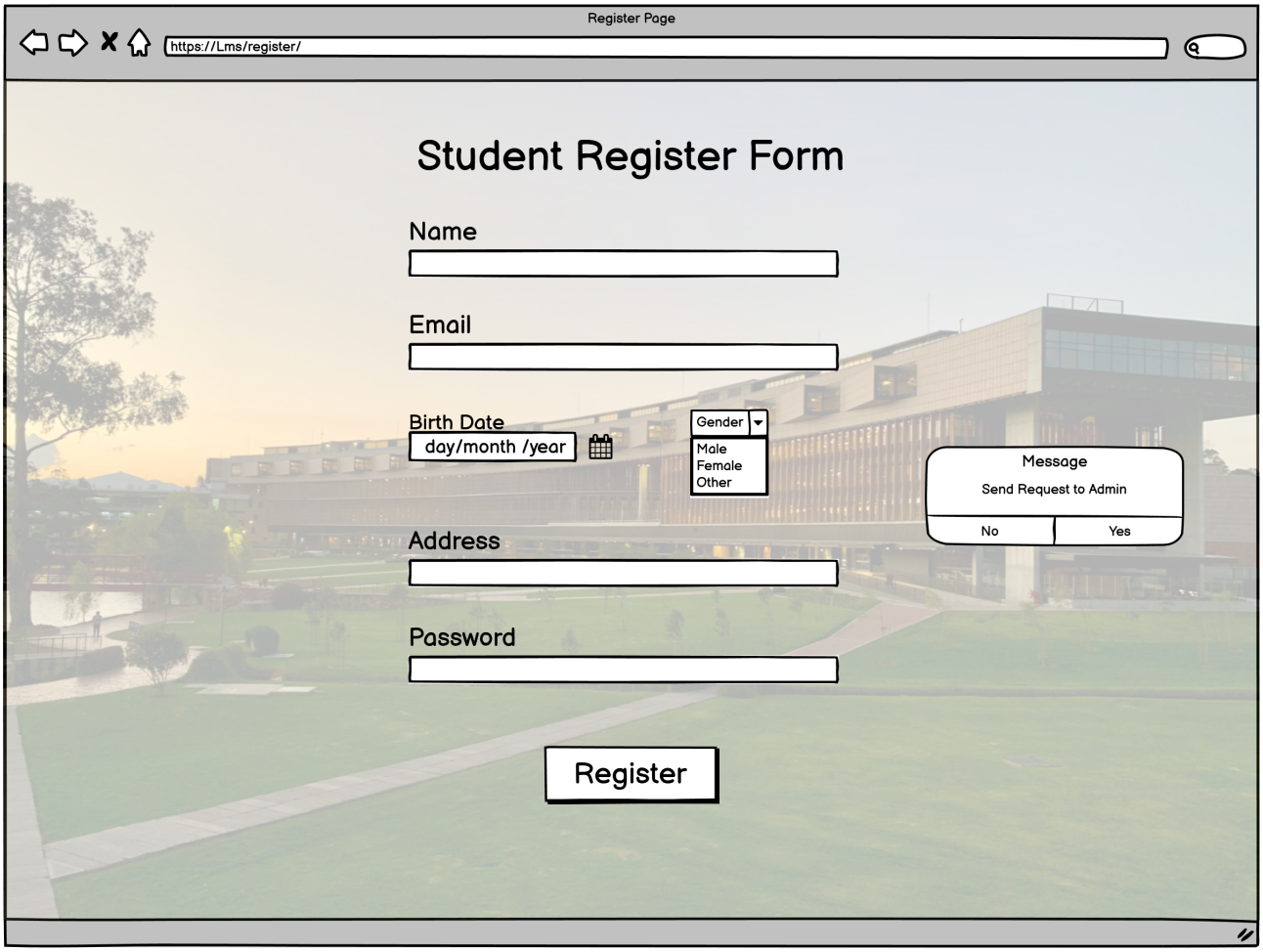
4.3 Login Page

The login page contains the form used to having an access to users own page inside the system. There are two input fields for e-mail address and the password, respectively. The user can select his/her status at the category field as teacher, student or admin. By pressing the LOGIN button the user will go to the personal page with all the information concerning his/her courses that are in progress or have already been finished.



4.4 Student registration page

The registration page contains a form for manual filling in the name, e-mail, address and password. The Gender field gives the option to choose between three variants and at the Birth Date field the date can be selected from the database. After the form is filled in the user submitting it and sending the register request to admin by clicking the Register button. Getting the register request from user the admin gives the access to the portal or decline it. If the register request is approved for the user he/she will have the ability to login to the portal.

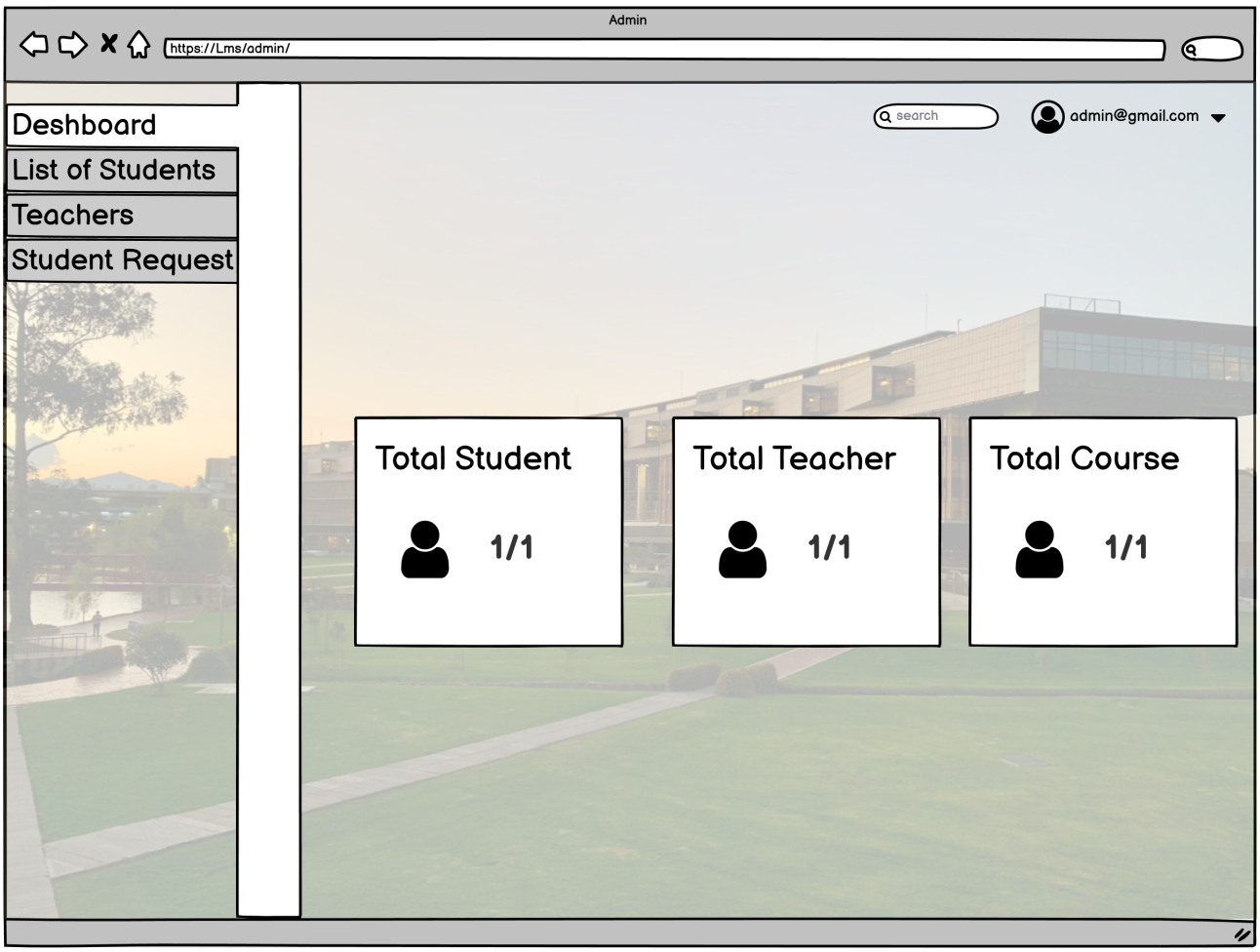


4.5 Admin Page

The admin page is divided into three sections: "search", "user details" and "update user details". Upon entering the page, only the "search" section will be filled.

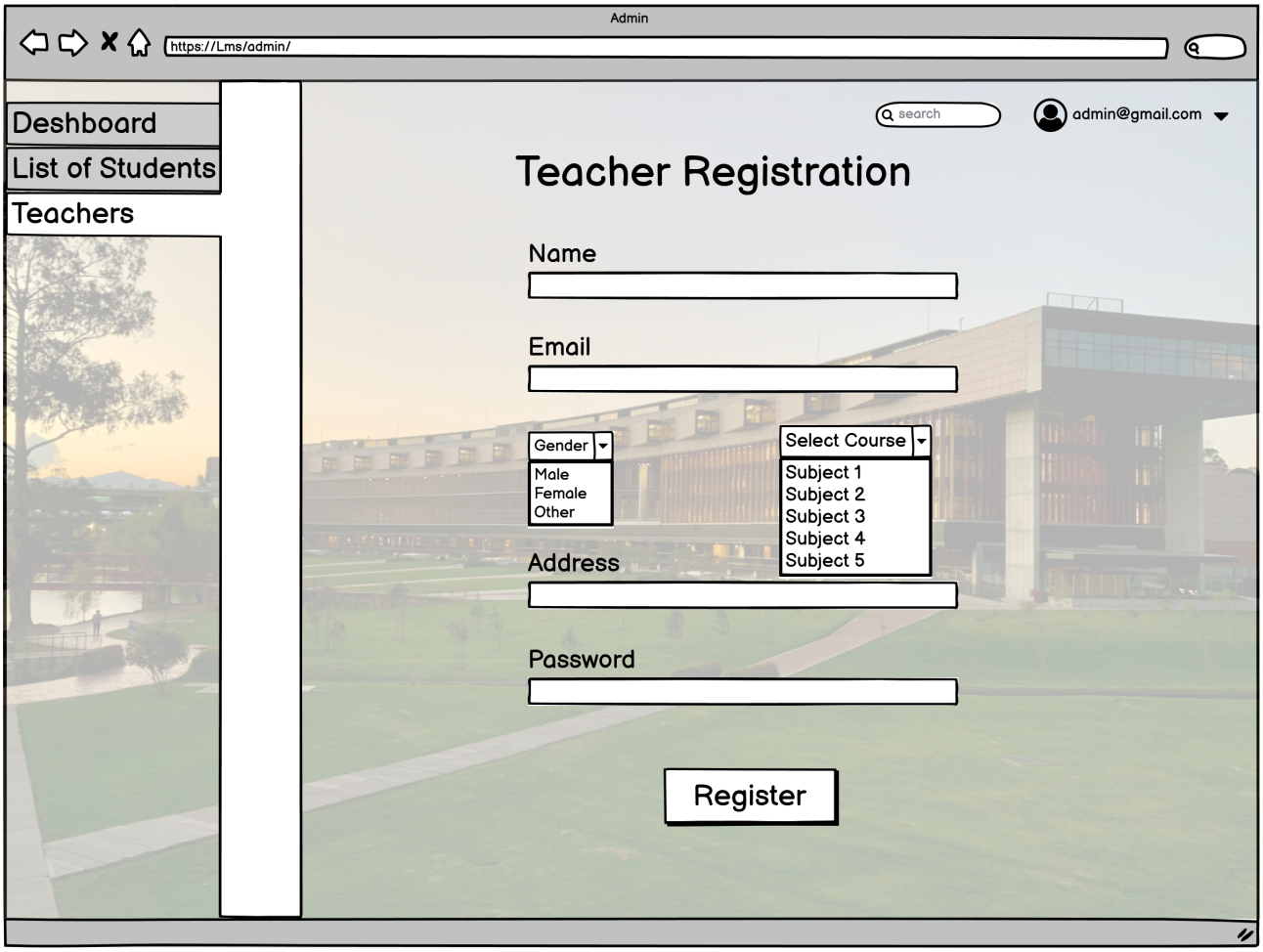
The admin page allows to search for users via email. The selector is filled automatically with the emails of all the registered users. Once the email of the user has been specified, by clicking on the "Search" button, the two following sections ("user details" and "update user details") are automatically filled with the information on the users. In particular, in the section "user details" is reported a table with the user email, first name, last name and role.

The section "update user details" contains a form, automatically filled with the data on the searched user. The admin can update such data. They can then click on either the button "update" which updates the user details, or the button "delete" which deletes the user from the database, revoking their possibility to login and therefore to access reserved areas of the website.



4.6 Teacher Page

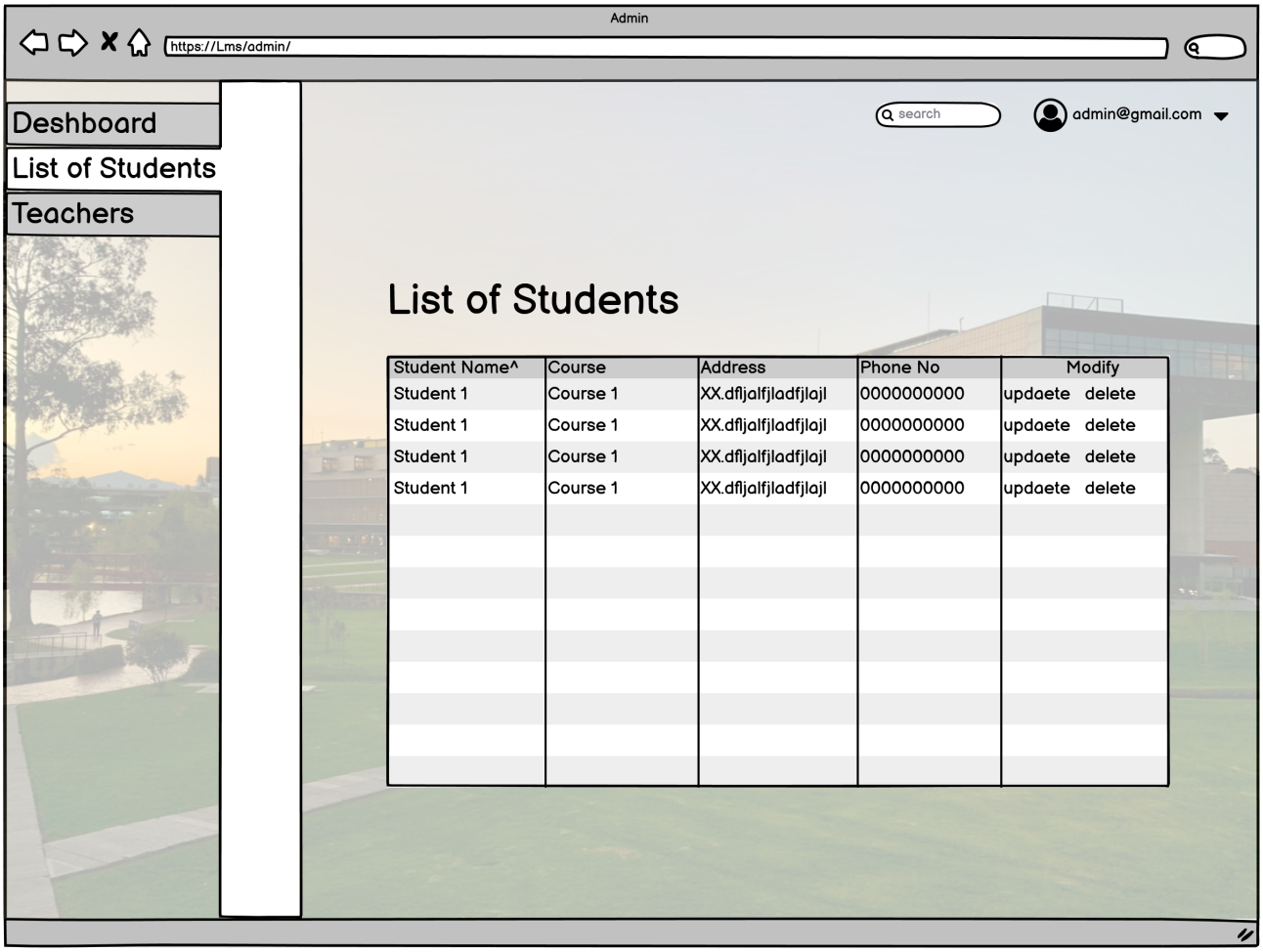
The general builder page is organized into folders. Each folder corresponds to a specific entity in the dataset, among those that can be managed by the builders. In particular, we have a folder for the parks, one for the rides and one for the models. By clicking on each folder, the user is presented with the interface to interact with the requested resource. In particular, there is a link that allows the user to reach the page for inserting a new resource and a selector that allows the user to select a specific resource based on its unique identifier and reach the page to edit it.



4.7 List of students Page

The "search maintenance page" contains a form used to filter out maintenance events. The user can select the ride to which the searched maintenance events refer to. They can filter events with two checkboxes to select either only events planned, not planned or both.

There are also two date input fields, they can be used to filter maintenance events in order to select only maintenance events occurring between two specific dates.



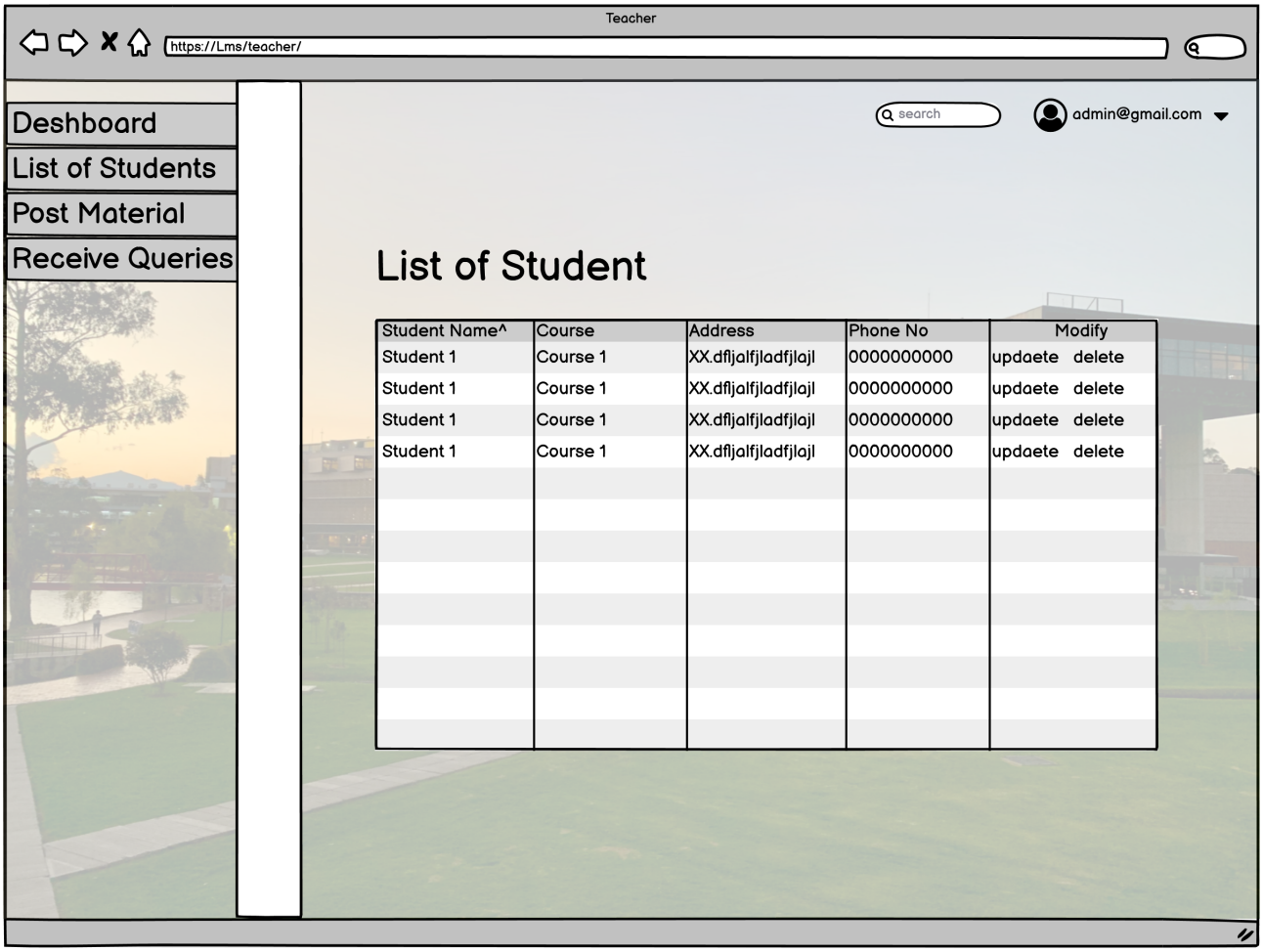
4.8 List of enrolled students Page

The Insert Rides Page contains the form necessary to insert a new ride.

There is a static and a dynamic part of the page. In the static part, it is possible to insert the description of the ride, the park it is going to be deployed in and its model. Both the park input and the model input are going to be filled automatically, based on the data available in the database.

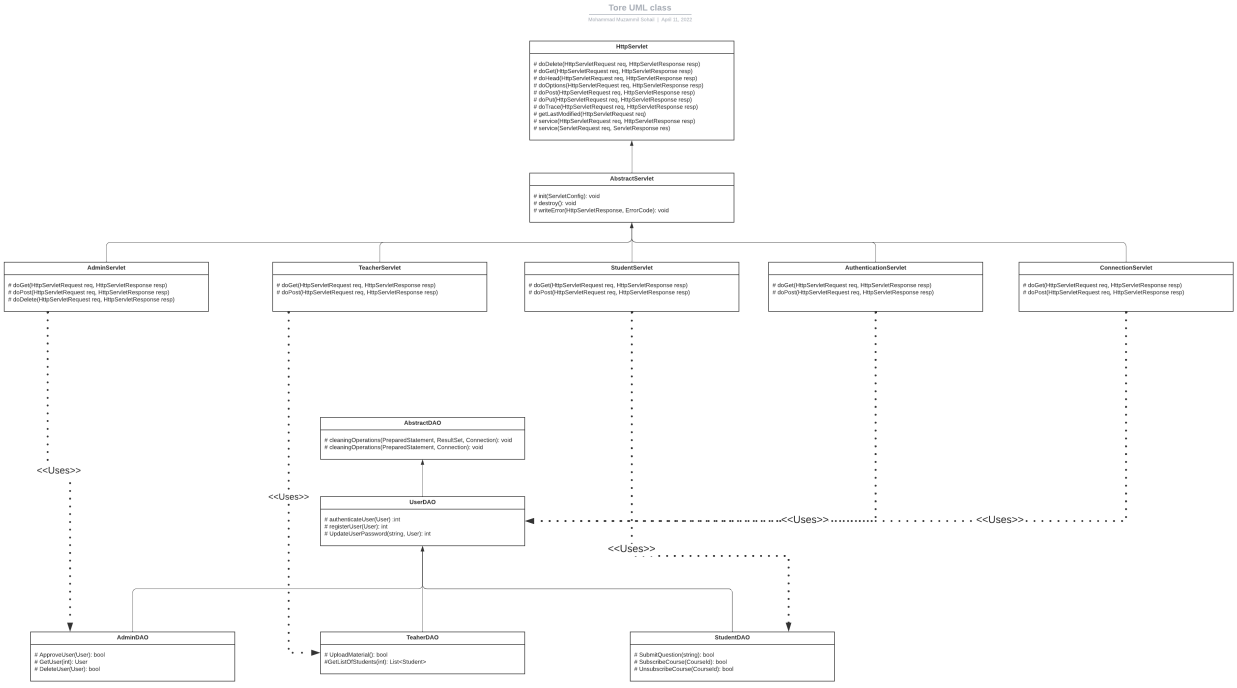
Dynamic aspects of the page regard the possibility of adding (and removing) on the fly additional devices that are mounted on the ride. By clicking on a "+" button, a new device form will be inserted in the device form area. There is also a "-" button, which will allow the user to remove devices. Note that, each device form will have its own "-" button. This way, the user can remove specific devices, without the need for additional effort that might stem from the removal of the last inserted device.

Upon completion, if the insertion ended correctly, the user is redirected to the homepage.


Teacher
https://Lms/teacher/
Deshboard
List of Students
Post Material
Receive Queries
search
admin@gmail.com
List of Student
Student Name^ Course Address Phone No Modify
Student 1 Course 1 XX.dfljalfjladfjlajl 0000000000 updaete delete
Student 1 Course 1 XX.dfljalfjladfjlajl 0000000000 updaete delete
Student 1 Course 1 XX.dfljalfjladfjlajl 0000000000 updaete delete
Student 1 Course 1 XX.dfljalfjladfjlajl 0000000000 updaete delete

5 Business Logic Layer

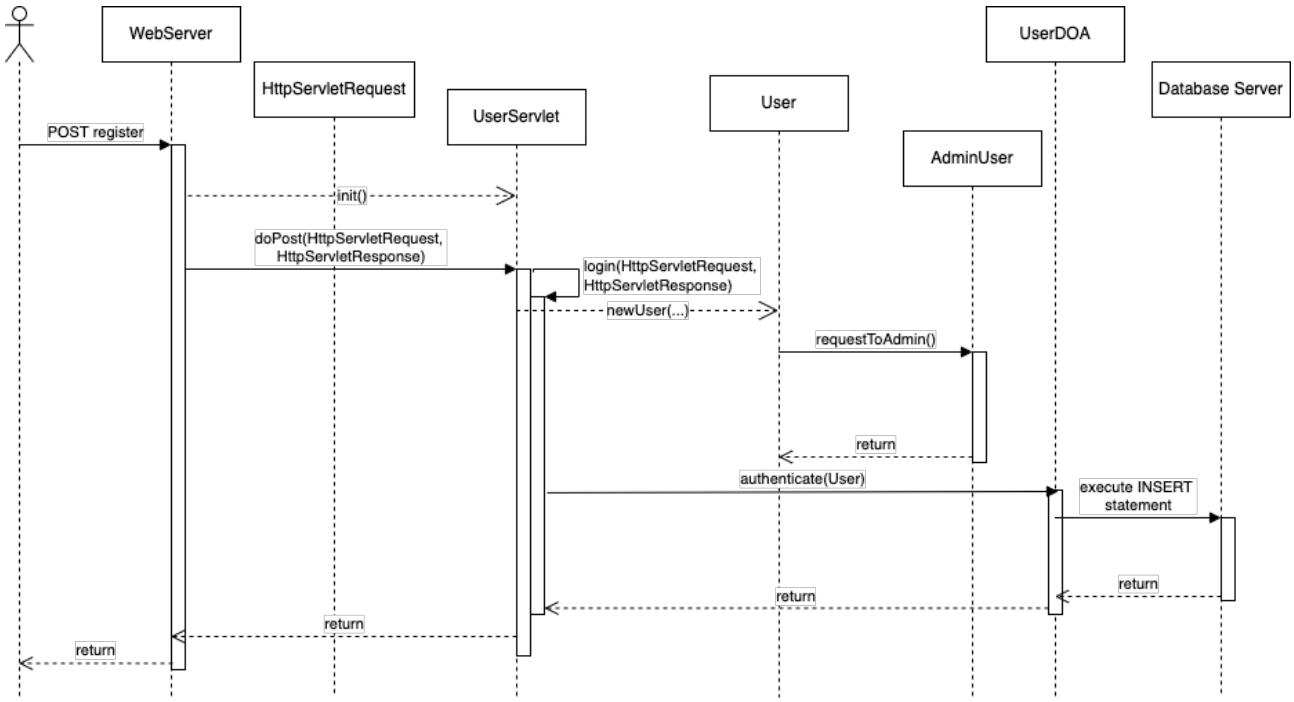
5.1 Class Diagram



The class diagram contains (some of) the classes used to handle functionalities of the Admin, Teacher and Student in our system. AdminServlet, TecherServlet and StudentServlet are subclasses of the Java's HttpServlet class. They are used to request functionalities accessible by the repective user. The Connection Servlet is used to make connections to the database and consequently exchange data from it. The Authentication Servlet is used to authenticate each of the respective users.

The AdminDAO, TeacherDAO and StudentDAO are subclasses of UserDAO because some of the functionalities are common in each of the users. Theses DAO classes are used to obtain access to the desired type of user in the system.

5.2 Sequence Diagram



The sequence diagram shows how the registration of a new user. The project uses the MVC pattern in the development. User, AdminUser, UserDOA, and DatabaseServer are the models. WebServer and HttpServletRequest interact as views, while UserServlet is the controller. The sequence starts with a user that sends a POST request for registration to the WebServer. A UserServlet is then initialised and executes methods for doPost() and login(). The user makes a call to the User model with newUser() and the User model asks for a request to the model AdminUser to approve the new user. After the approval of the new user, the UserServlet sends an authenticate(User) to UserDOA for the model to INSERT the new user data into the DatabaseServer model.

5.3 REST API Summary

The list of the rest API for this homework are still "in progress" because we have to decide according to the frontend where is better to use a REST call or not. Anyway, there are some resources that are just available with REST that are reported in the next table.

The rest API is studied to experiment with multiple different approaches to the REST paradigm. More in detail, endpoints associated with rides and devices are developed in a more traditional way, with the entirety of the information needed to satisfy the request directly in the URI. Other endpoints require additional information (such as the type of operation required) as additional parameters in the request.

Part of the URIs are filtered through different filters. S indicates that only users with the role of student can access to the endpoint, T indicates that only T can access the endpoint, A that only administrators can request the specified URI to the server.

URI	Method	Description	Filter
admin/	POST	Go to the administrator section. Only admin can see access the infomation	A
admin/dashboard/	POST	Go to the administrator dashboard. Only admin login.	A
login/	POST	Go to login section for teacher and students only	S and T
register/	POST	Go to registration section for teacher and students only	S and T
home/	POST	Go to homepage	S and T
about-us/	POST	Go to about us	S and T
contact-us/	POST	Go to contact us	S and T
(user)/dashboard/	GET	Go to the administrator dashboard. Student and teacher can see access the infomation.	S and T
(user)/search/(course-name)	GET	Search for the course	S and T
(user)/(course-id)	GET	Go to the course for information	S and T
(user)/(course-id)/enroll	POST	Enroll in a course	S
(user)/(course-id)/unenroll	POST	Un-enroll in a course	S
teacher/(course-id)/post	GET	Post material for a course	T
student/(teacher-id)/message	POST	Message a teacher	T

Table 2: Describe in this table your REST API

5.4 REST Error Codes

Error Code	HTTP Status Code	Description
Error Code Identifier	Corresponding HTTP Status Code	Description of the error

Table 3: Describe in this table your REST API

5.5 REST API Details

In our project, we have 10 different resource types that our web application handles during its functioning. Following are resources.

Homepage content

The homepage contains an aggregation of different pieces of information. In particular, the Homepage has login and register pages, which allow users to log in themselves and register as well.

- URL: `Homepage/overview`
- Method: `GET`
- URL Parameters: `No Parameters are required in the URL`
- Data Parameters: `No Data is passed when requiring this method`
- Success Response: `Code: 200`

Login Page

- URL: `Homepage/LMS/register`
- Method: `POST`
- URL Parameters:
  - `Name`: {string}
  - email = {string}
  - birth_date = {string}
  - gender = {string}
  - address = {string}
  - password = {string}
- Success Response: `Code: 200`
  Content: The user is redirected to the login page.
- Error Responses:
  - Code: 400 BAD_REQUEST
    Content: `{\error": {\code": -102, \message" : \One or more input fields are empty."}}`
    When: some of the parameters are missing or are empty string.
  - Code: 409 CONFLICT
    Content: `{\error": {\code": -107, \message" : \Email Already Used."}}`
    When: The user is trying to get register with the same email address that is already present in database.
  - Code: 400 $BAD_REQUEST$
    $Content$: `{\code": -108, \message" : \Password is Weak."}}`
    $When$: $EnteredPasswordisweak$.
- Code: 500 $INTERNAL_SERVER_ERROR$
  $content$: `{\code": -999, \message" : \Internal Error."}}`
  $When$: $ifthereisaSQLExceptionoraNamingException, thiserrorisreturned$.

Admin Page

It contains the full details of Student and Teacher information in the database, there is a page that contains information about students and their enrolled courses, and there is also a separate page for Teacher's' information and their teaching courses.

1.View Students:

Admin can View all registered student.

- **URL:** `/admin/student-list`
- **Method:** `GET`
- **Data Parameters:** `No Data is passed when requiring this method.`

- **Success Response:** `Code: 200`
  Upon success, the servlet returns a JSON object with the required information.

  Content:
  {
  "data":
  [{
  "Name": "XYZ"
  "email":"xyz@gmail.com"
  "address": \Padova, PD 35010"
  "Courses List": \Machine Learning, Foundation of Database, Search Engine, Web Application"
  }]
  }

- **Error Responses:** `Code: 500 INTERNAL_SERVER_ERROR`
  Content: {\error": {\code": -999, \message" : \Internal Error."}}
  When: if there is a SQLException or a NamingException, this error is returned.

2. View Courses:

Admin can View all courses.

- **URL:** `/admin/courses`
- **Method:** `GET`
- **Data Parameters:** `No Data is passed when requiring this method.`
- **Success Response:** `Code: 200`
  Upon success, the servlet returns a JSON object with the required information.

  Content:
  {
  "data":
  [{
  "Name": "XYZ"
  "email":"xyz@gmail.com"
  "address": \Padova, PD 35010"
  "Courses List": "Machine Learning, Foundation of Database, Search Engine, Web Application"
  }]
  }

- **Error Responses:** `Code: 500 INTERNAL_SERVER_ERROR`
  Content: {\error": {\code": -999, \message" : \Internal Error."}}
  When: if there is a SQLException or a NamingException, this error is returned.

6 Group Members Contribution

Ahmed Saad

Bahrami Sepide

Hansen Marit Fredrikke

Rao Abdul Moeed

Rehman Abdul

Sohail Mohammad Muzammil

Soomro Dodo Khan

Sultanova Aliia

Zabalawi Iyad