##### National University of Computer & Emerging Sciences, Karachi



##### Computer Science Department

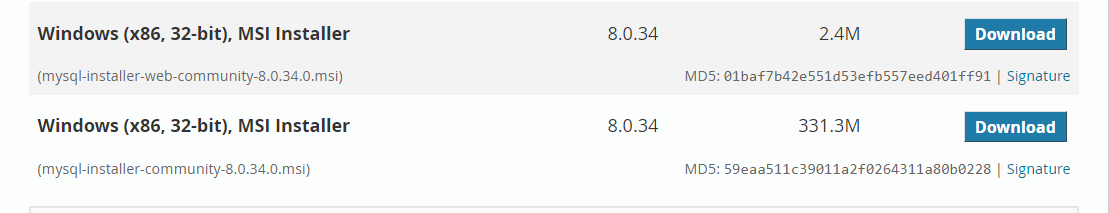
**Spring 2023, Lab Manual – 07**

|  |  |
| --- | --- |
| **Course Code: CL-2005** | **Course: Database Systems Lab** |
| **Instructor(s):** | **Shakir Hussain** |

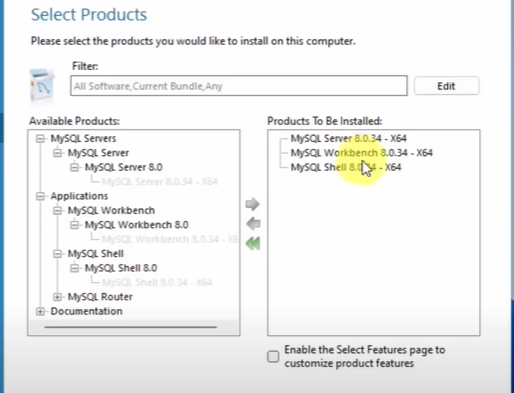
# **CONNECTIVITY**

* **My SQL (NO ORM)**

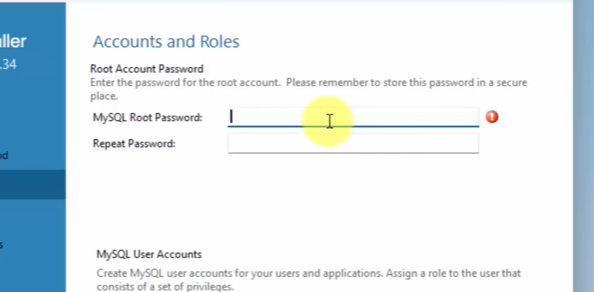
Goto <https://dev.mysql.com/downloads/installer/>, and download the 2nd file.



Once downloaded, run the installer, and choose custom option. From within the custom option, select MySQL version, MySQL workbench, MySQL Shell and install them.



Once done, continue with the installation until you reach the password interface and set a password for your “sys” user (make it easier to remember).



Once downloaded, you must add MySQL to the path in environmental variables. For that

* Press windows key and type “environment”.
* Go to the “edit the system environment variables” and select “Environment variables”, go to path as shown below, select it and click edit. Once inside, select new and copy & paste the address to MySQL bin folder there.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Back-end connectivity (Node & express)**

Goto the project folder(backend) and run,

**npm install mysql2**

Create an SQL connection pool with

**const pool = mysql.createPool({**

**host: 'localhost',**

**user: 'your\_mysql\_user',**

**password: 'your\_mysql\_password',**

**database: 'your\_database\_name',**

**connectionLimit: 10,**

**});**

You can test the connection with

**pool.getConnection((err, connection) => {**

**if (err) {**

**console.error('Error connecting to MySQL:', err);**

**} else {**

**console.log('Connected to MySQL database');**

**connection.release(); // Release the connection when done**

**}**

**});**

Once connected, you can make request to the database like

try {

        const [rows] = await pool.execute('SELECT \* FROM user');

        res.json(rows);

    } catch (err) {

    console.error('Error executing query:', err);

    res.status(500).send('Error executing query');

    }

This will return all users stored within the specific database that you’ve provided details of, in the pool above.

**With ORM**

**ORM** stands for object relational mapping. Basically, it’s a tool that allows you to get rid of queries and tables and instead automates the whole work for you where you can just define a class or model in the back-end and it will automatically create that table for you directly from the back-end without any SQL queries.

The ORM that SQL based languages usually use is called sequelize. Following are the steps to adding ORM to your back end.

* **npm install sequelize mysql2** (if not mysql2 is installed already).
* Create a file db.js or config.js where you’re gonna set up the connection as follows:

**const { Sequelize } = require('sequelize');**

**const sequelize = new Sequelize('your\_database\_name', 'your\_mysql\_user', 'your\_mysql\_password', {**

**host: 'localhost',**

**dialect: 'mysql',**

**});**

**module.exports = sequelize;**

* As mentioned above, you can create models for your tables now. Below mentioned code shows the implementation for the user model:

const { DataTypes } = require('sequelize');

const sequelize = require('../db'); // Import the Sequelize configuration

const User = sequelize.define('User', {

 id: {

    type: DataTypes.INTEGER,

    primaryKey: true,

 },

  full\_name: {

    type: DataTypes.STRING,

    allowNull: false,

  },

  email: {

    type: DataTypes.STRING,

    allowNull: false,

    unique: true,

  },

});

module.exports = User;

* Once done, you can use this model in backend to create and retrieve data from you database table “Users”.

**Postgresql**

For postgresql, the process is quite similar. Download postgres, pgadmin and its essential and set up the environment variables then for the connection, follow the below mentioned steps:

* **npm install pg**
* in your config or db.js, do the following:

const { Client } = require('pg');

const db = new Client({

  host: "localhost",

  user: "postgres", //youruser

  port: "5432", //yourport

  password: "shakir", //yourpassword

  database: "postest", //yourdb

});

module.exports = db;

* Once done, you can import this db modules and use **db.connect()** to connect the DB to the back-end and then make queries to it like,

try {

        const users = await db.query('SELECT \* FROM users');

        res.json(users.rows);

    } catch (error) {

        console.error(error);

        res.status(500).json({ error: 'Internal server error' });

    }

For Python & Django, you can connect in the same manner once your initial indes.py and config.py are setup.