COSA INC.

Technical User Manual

International Student Management System

Version 1.1
June 10, 2022

Cyberbot Team Members:
Anu Treesa George
Mohammmad Jahanseir
Moises Melgarejo
Nhat Nguyen
Nilkumar Patel
Sasidhar Rekhapalli

Table of Contents

Overview

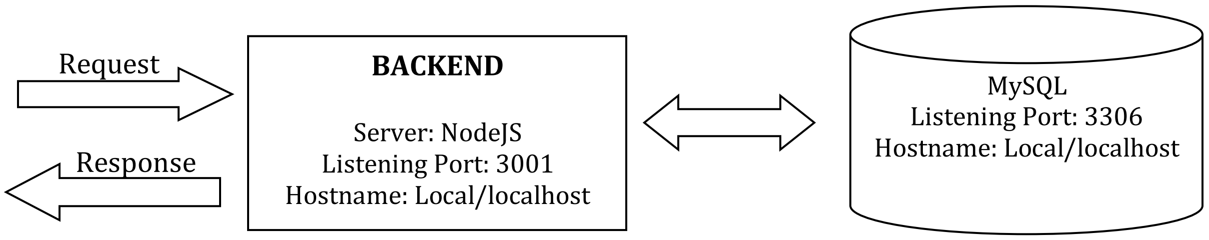
The Technical User Manual was created to cover all information about important modules in the International Student Management System

Product Use Description

This document gives detailed information the way it was developed, highlight important aspects from development side, the project structure, database, frontend and backend development and installations.

1.0 Backend

We use backend as the bridge for our connectivity between database and apis used in front end. Server receives request from frontend and can accepts parameters through routes. These parameters are mostly ids. Server excutes CRUD operations (Create, Read, Update and Delete) to manage data. Server send a response after excuting a specific CRUD operation. The image shows a general idea about data processing when a route is invoked. This section we will describe the implementation of the backend project.



We developed the server in Node.js. See Apendix A to install this Node.js runtime environment as well as the ISMS project backend.

1.1 ISMS Backend Project

To structure the backend project, we modularized server functions into folders. Each function has a particular behavior in the server as following,

Module Folder	Description
controllers	This folder is destined to insert methods with CRUD operations and send responses.
controllers/update	This folder is destined to uploaded files.
db	This folder is destined to make the connectivity with MySql Database. We set up the host, user, password and database information to created a pool connection.
routes	This folder is destined to define routes to get requests.

1.1.1 Server

Server.js is the file that defines the listening port, URLs, Upload File managemet and Database. To developed backend we utilized the following libraries,

1. Name: *Express*
   Version: "^4.18.1"
   Description: Express is a framework to do the server side functionality. This package provides plugins, template code, middleware packages, and routing functionality for faster and efficient web development.
2. Name: *MySQL*

Version: "^2.18.1"
Description: My-Sql package is used to create database connection with our application to My-sql database.

3. Name: *MySQL2*
Version: "^2.3.3"
Description: Updated version of MySQL for recent versions of MySQL.
4. Name: *Cors*
Version: "^2.8.5"
Description: Provide Connect/Express middleware that can be used to enable CORS with various options. For this project, the server credentials.
5. Name: *Express-FileUpload*
Version: "^1.4.0"
Description: Manage uploaded files and storage into the server.
6. Name: Bcrypt
Version: "^5.0.1"
Description: Bcrypt is a password hashing module.
7. Name: Passport
Version: "^0.6.0"
Description: Passport is authentication middleware for Node.js.
8. Name: Passport-local
Version: "^1.0.0"
Description: Passport strategy for authenticating with a username and password in Node.js application.

**Information taken from web site respectively.

1.1.2 Controller

As well as Server, Controller is the important backend module and we put all CRUD operations. In controller we have four main sections: student, user, conversation and login. For each part we have some modules to access data and run quries. For student and user we get all or one row of table in database, also create, update and delete are important modules in this part. In conversation like other parts, in addition we have update file as well.

Student	User	Conversation	Login
- getAllStudents - getStudentById - createStudent - addStudent - updateStudent - deleteStudent	- getAllUsers - getUserById - getUsersView - createNewUser - addUser - updateUser - deleteUser	- getConversation - getConversationByConsID - createConversation - updateConversation - updateFile	- resetPassword - login

1.1.3 Routes

We created all modules and export it in the controller, in this part those modules are import and assign in the proper route. Each CRUD operation assign in specific route and user directed to that route(addtress) to access to the system.
routes:

- router.get("/getallstudent", controller.getAllStudents);
- router.get("/getstudent/:id", controller.getStudentById);
- router.post("/newstudent", controller.createStudent);
- router.post("/addstudent", controller.addStudent);
- router.put("/updatestudent/:id", controller.updateStudent);
- router.delete("/deletestudent/:id", controller.deleteStudent);
- router.get("/getconversationid/:id", controller.getConversation);
- router.get("/getconversationbyconsid/:id", controller.getConversationByConsID);
- router.post("/newconversation", controller.createConversation);
- router.put("/updateconversation/:id", controller.updateConversation)
- router.get("/user/getuser", controller.getAllUsers);
- router.get("/user/getuser/:id", controller.getUserById);
- router.get("/user/getUsersView", controller.getUsersView);
- router.post("/register", controller.createNewUser);
- router.post("/user/adduser", controller.addUser);
- router.post("/login", controller.login);
- router.put("/user/updateuser/:id", controller.updateUser);
- router.delete("/deleteuser/:id", controller.deleteUser);
- router.put("/resetpassword/:id", controller.resetPassword);
- router.post("/updateFile/:id", controller.updateFile);

1.1.4 Login Server Side

We have to hightlight main functions from the security implementation. This part covers hashing and salting for encrypting passwords.

Bcrypt.hash(): -Bcrypt.hash is the hashing function allows us to build a password security platform that scales with computation power and always hashes every password with a salt.

This takes two arguments one is password to be hashed and another salt rounds and hashes the password accordingly.

SaltRounds: - salt rounds mean the cost factor. The cost factor controls how much time is needed to calculate a single Bcrypt hash. In this application we used salt rounds of 10 which means that many hashing rounds are done.

Bcrypt.compareSync(): - This function takes only two arguments and returns a Boolean value true or false. In this application we are comparing the user enter password with the hashed password saved in the database for the match.

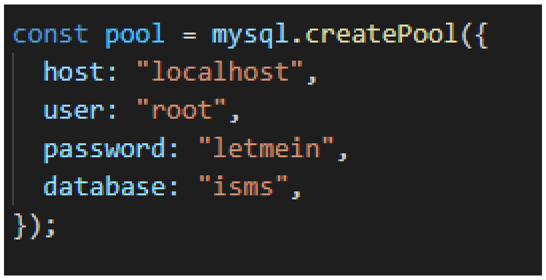
To setup, we follow these steps,

1. In “route.js” file in users’ routes there is route with post “/login” calls the controller function called “login.”
2. Most of the login work is done in “controller.js” file function called “login”
3. First, we try to connect to database and search for username with SQL query.
4. Save the password from the database if the username found.
5. Then using bcrypt.compareSync function we compare with passed in password with already hashed password in database.
6. If matched, we send back response to frontend using route.

1.1.5 Database Connection

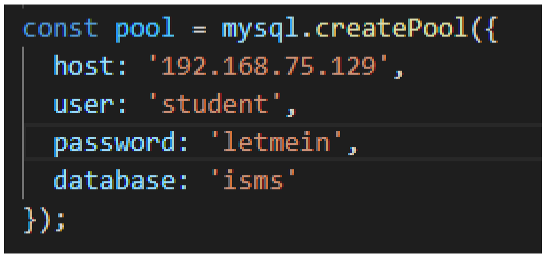
We created a relational database for storing data. Database manager is set up with correct parameters to get connectivity. Host, User and Password vary depending on MySQL setup initial parameters. As local computer, we used localhost, root as user and isms as database. If database is in the other server or on the Internet we use IP(Internet Protocol) address in the host name. All database information located in db/connect.js.

Our application is developed and tested in local computer. To create a Database pool connection, we set up the following parameters,



```
const pool = mysql.createPool({
  host: "localhost",
  user: "root",
  password: "letmein",
  database: "isms",
});
```

Be careful for addressing, because many issue cause by changing database system’s ip address and forget to update information in the connect.js file. If database is not is same computer, find the ip address and update ip in this section. For example if database address is 10.10.10.10, instead of 192.168.75.129 (shown in the figure below) put your 10.10.10.10.



```
const pool = mysql.createPool({
  host: '192.168.75.129',
  user: 'student',
  password: 'letmein',
  database: 'isms'
});
```

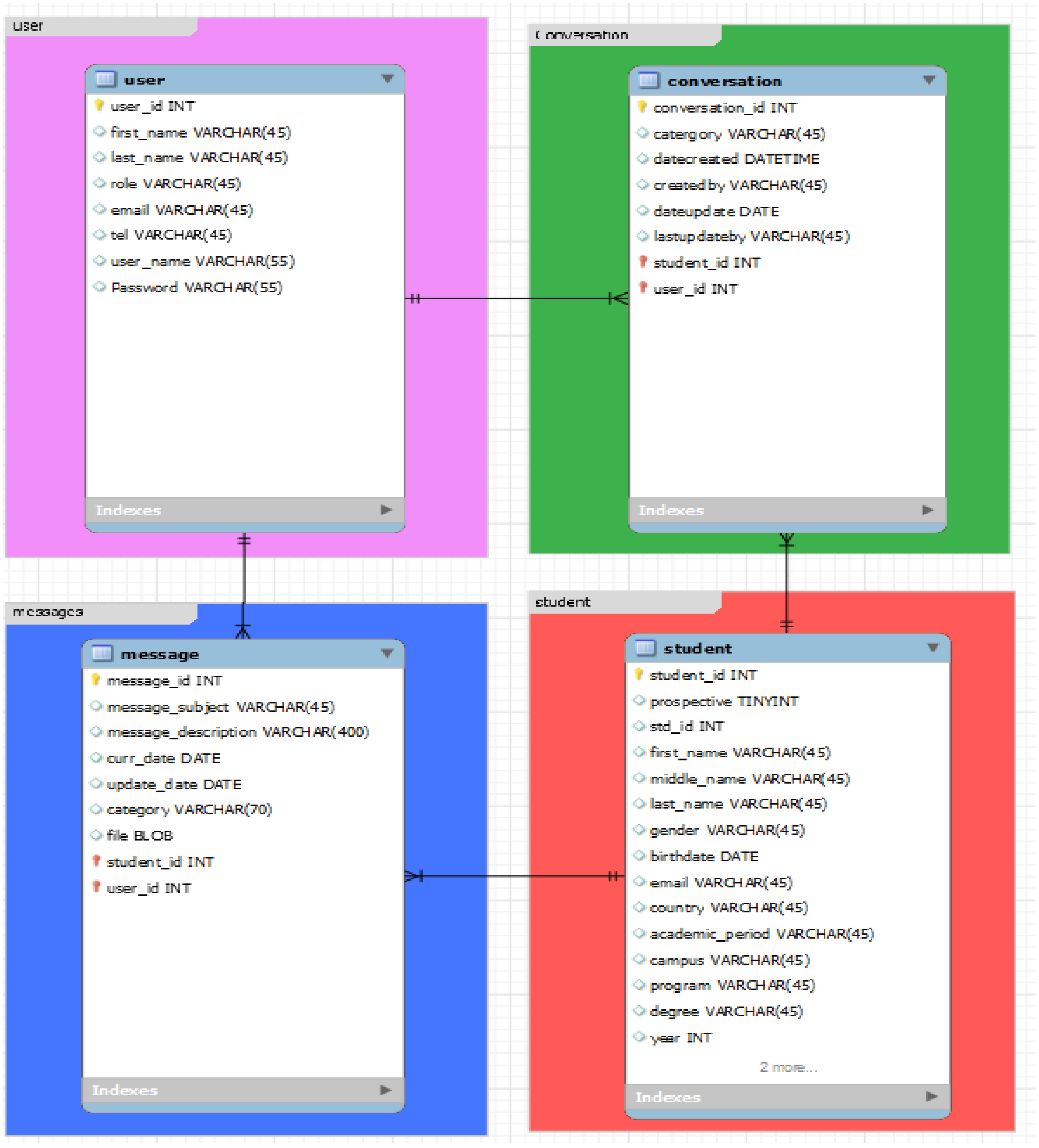
2.0 Database

To manage all data from students, we design a relational database. This database was standarized and normalized taking the raw data as model and the requirements to complete the users idea. Raw data was provided in Excel Spreadsheets which is the current data.

** See Appendix A to install MySQL database manager and scripts for installing ISMS database and data.

2.1 Entity Relationship Diagram

The following ER diagram corresponds to ISMS system. We created four tables and each table has a unique field or primary key and foreing key to make it relational with to other tables.



2.2 Tables

Four tables keep all required data for the system. User and student tables are the primary and crucial for this project. All communication and notes are stored in message and conversation tables.

Student Table
Stores the international student's information.

Field	Type
student_id	INT, PRIMARY KEY
prospective	BOOLEAN
std_id	INT
first_name	VARCHAR(45)
middle_name	VARCHAR(45)
last_name	VARCHAR(45)
gender	VARCHAR(45)
birthdate	DATE
email	VARCHAR(55)
country	VARCHAR(20)
academic_period	VARCHAR(45)
campus	VARCHAR(45)
program	VARCHAR(45)
degree	VARCHAR(45)
year	INT
graudate_ind	VARCHAR(45)
enroll	VARCHAR(45)

User table
Stores the user's information including user_ name and password as part of security matter to access into the system.

Field	Type
user_id	INT, PRIMARY KEY
first_name	VARCHAR(45)
last_name	VARCHAR(45)
role	VARCHAR(45)
email	VARCHAR(55)
tel	VARCHAR(45)
user_name	VARCHAR(55)
password	VARCHAR(255)

** Password field has a 255 characteres due to encrypts password with a long string.

Message Table
Stores messages from users from student's situation.

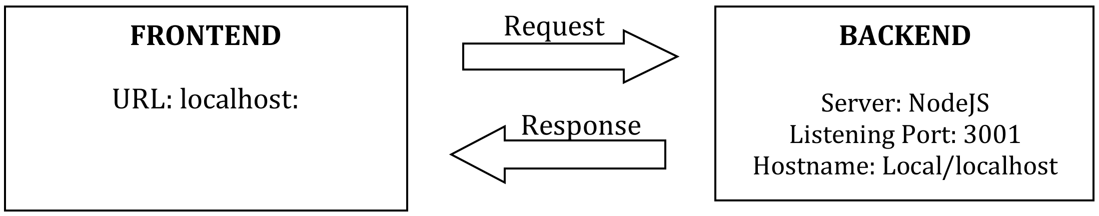
Field	Type
message_id	INT, PRIMARY KEY
message_subject	VARCHAR(45)
message_description	VARCHAR(400)
curr_date	DATE
update_date	DATE
category	VARCHAR(70)
file	BLOB
student_id	INT, FOREING KEY
user_id	INT, FOREING KEY

Conversation Table
Stores conversations made from emails or any other communication way.

Field	Type
conversation_id	INT
category	VARCHAR(45)
datecreated	DATETIME
createdby	VARCHAR(45)
dateupdated	DATE
lastupdatedby	VARCHAR(45)
note	LONGTEXT
comments	LONGTEXT
sharedLink	LONGTEXT
subject	VARCHAR(45)
permission	VARCHAR(45)
file_upload	LONGTEXT
student_id	INT, FOREING KEY
user_id	INT, FOREING KEY

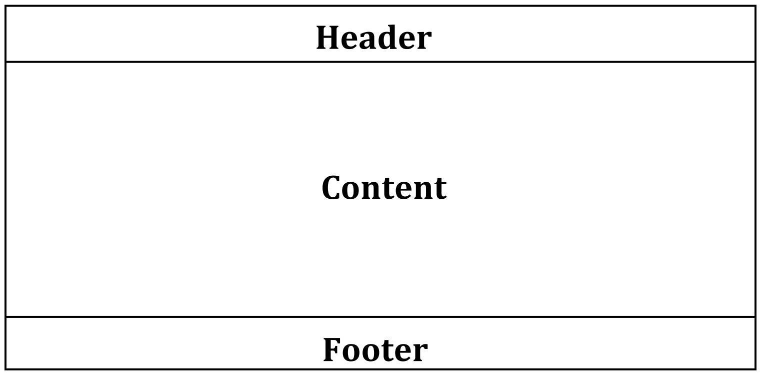
3.0 Frontend

Frontend is the view module in our system. It was implemented in React and created components to structure the application. User inputs data through forms and the application shows general and detailed information from database. Frontend generates requests to the server to process the inputted data into data base. This section we will describe the implementation of the frontend project.



** See Apendix A to install ISMS project frontend.

During the analysis and design of the UI prototype, we ended up with the following template for the whole system except the Loging page.This template will contain forms to insert data and tables to show data. It was practical for us to work on this way.



Where,

Header	Navbar.jsx
Footer	FootNav. jsx
Content	LoginView.jsx StudentPage.jsx BriefShowStudent.jsx AddStudent.jsx RegisterUser.jsx UserManagement.jsx AddConversation.jsx ResetPassword.jsx DetailDialog.jsx

Every single component has its own implementation. The practicity of having components makes the system more maintainable. If there is a situation to add more components or remove components, index.js in app folder can be updated to make such changes.

Login Component works in the different way, it does not have a template implementation because it contains two input data, User and Password as any other Login Page.

3.1 ISMS Frontend Project

When we create a react project, the batch process from npx generates automatically the project structure. Our implementation is based on this structure and we describe our setup in the following table.

Module Folder	Description
public	Contains index.html. This attached main file inserts the content.
source	Contains the developed application.
source/api	Contains the request routes to server.
source/app	This folder is destined to define routes to get requests.
source/components	Contains the main template components
source/css	Contains the style sheet for Login
source/images	Contains images used in the system.
source/pages	Contains the content components

To develope frontend we utilized the following libraries,

1. Name: *Bootstrap*
   Version: "^5.1.3"
   Description: Bootstrap is a potent front-end framework used to create modern websites and web apps, This package bootstrap is used to import files to use bootstrap classes.
   Usage: Using the import statement we can include the bootstrap package in our application.
2. Name: *React*
   Version: "^18.1.0"
   Description: React package is used to create interactive UIs. Design simple views for each state in our application, and React will efficiently update and render different page layouts.
   Usage: Using the import statement we can include the react package in our application.
3. Name: *React-bootstrap*
   Version: "^2.4.0"
   Description: React package is used to create interactive UIs. Design simple views for each state in our application, and React will efficiently update and render different page layouts.
   Usage: Using the import statement we can include the react-bootstrap package in our application.
4. Name: *React-dom*
   Version: "^18.1.0"
   Description: React package is a source for components, state, props.
   The main usage is mounting our application to the index.
   Usage: Using the import statement we can include the react-dom package in our application.
5. Name: *React-router-dom*
   Version: "^18.1.0"

Description: React-router- dom package is used to do dynamic routing within our app.
Usage: Using the import statement we can include the react-router-dom package in our application.

6. Name:*React-scripts*
Version: "^5.0.1"
Description: React-scripts package is used to start sets up the development environment and starts a server.
Usage: Using the import statement we can include the react-scripts package in our application.
7. Name: *Styled-Components*
Version: "^5.3.5"
Description: Styled-components package is used to add component level styles in our application
Usage: Using the import statement we can include the styled-components package in our application.

**Information taken from web site respectively

3.2 Components

The developed components with specific operation, the following table describes that operation.

Component	Operation
LoginView	Shows loging page
BriefShowStudent	Shows the detailed student's information
RegisterUser	Shows the form to input user's information data
StudentPage	Shows the students' basic information in a table with searching operation. This page has two options, to add new student or setup user's profiles (applicable to Admin).
UserManagement	Shows the users' basic information in a table. Option to add new User or Update user's role. This page is applicable to Admin
AddStudent	Shows a form to input user's information data.
AddConversation	Show a forms to add conversation to students
ResetPassword	Shows a form to reset password for users.
DetailDialog	Shows a table with Detailed dialog between Students and Users
Navbar.jsx	Show the Header, also shows the page status with the current user.
FootNav.jsx	Shows additional Saskatchewan Polytechnic Information.

Each component has Navbar.jsx and FootNav.jsx added

```
<Navbar />

    -   Content

<FootNav />
```

3.3 Login Page Setup

1. In “Login.jsx” file using function “handleLogin” we get the username and password from react state and send those as parameters to function in Auth.js file

2. In “Auth.js” file there is function called “postUserLogin” with post route “/login” sends the parameters username and password from the browser.

Appendix A – Installation Instructions

Development

We have to install the following software and tools for development.

Visual Studio Code
It's and IDE for coding intervencions. It was used for developing ISMS frontend/backend.

Web site	https://code.visualstudio.com/

NodeJS
To run the Server and execute the Application, install NodeJS. Follow the indications from the official Web site

Web site	https://nodejs.org/en/
Downdload and Installation Options	https://nodejs.org/en/download/
Documentation	https://nodejs.org/en/docs/

MySQL
To run the Dababase, Install MySQL . Follow the indications from the official Web site

Main Page	https://www.mysql.com/
Downdload and Installation Options	https://www.mysql.com/downloads/
Documentation	https://dev.mysql.com/doc/

IntelliJ
It's an IDE for coding intervensions. It was used for creating code, compile and execute programs in Java.

Web site	https://www.jetbrains.com/idea/

Java
It's a development kit. It was used for importing data program.

Web site	https://www.java.com/en/
Downdload and Installation Options	https://www.java.com/en/download/
Documentation	https://dev.java/learn/

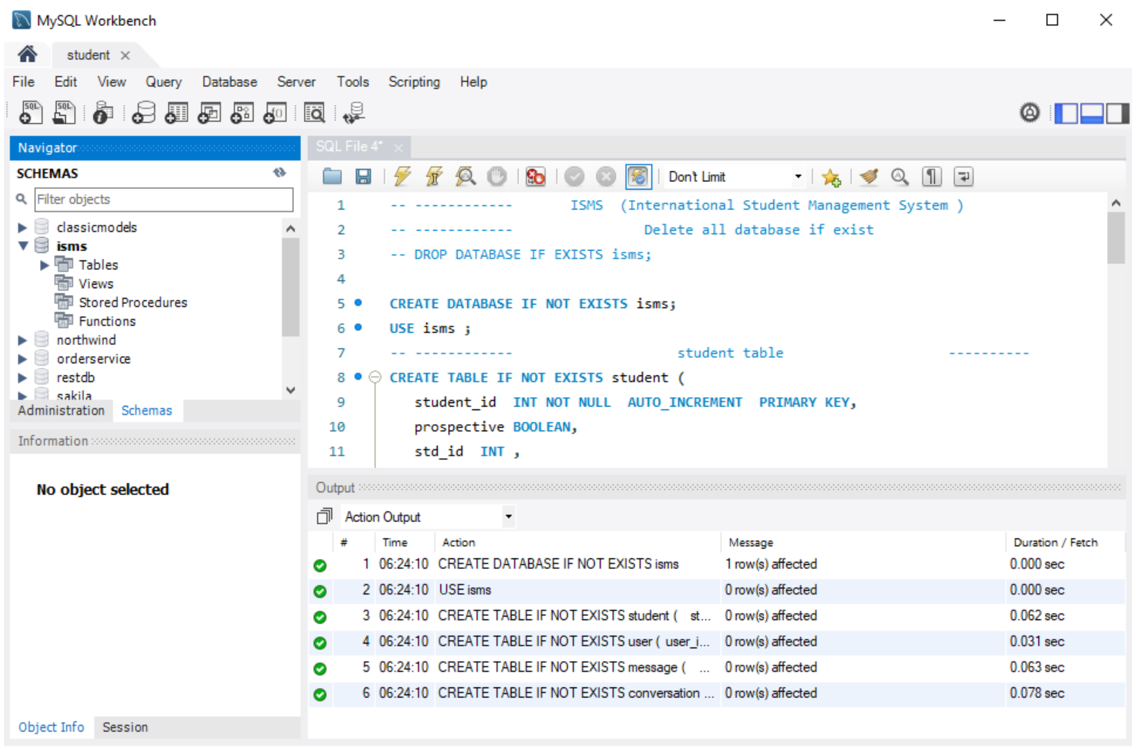
Installing ISMS

Installing Database

To install the data base, get theSQL scripts from ISMS shared folder.

Order	Script	Description
1	All-in-One-Script-V5.sql	Creates DB and Tables
2	ImportStudents.sql	Inserts Students Data
3	CreateUser.sql	Inserts Users Data
4	CreateConversation.sql	Insert Coversations

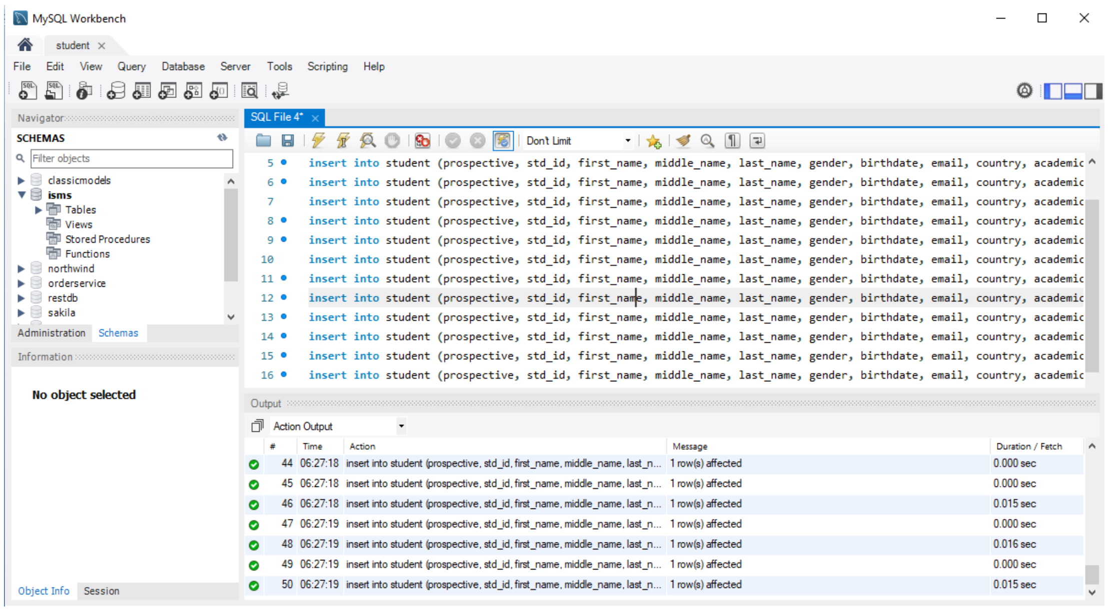
Open All-in-One-Script-V5.sql and execute the selected script. This script contains creation of database, tables and relationships.



```
-- ------------        ISMS  (International Student Management System )
-- ------------                 Delete all database if exist
-- DROP DATABASE IF EXISTS isms;

CREATE DATABASE IF NOT EXISTS isms;
USE isms ;
-- ------------                     student table               ---------
CREATE TABLE IF NOT EXISTS student (
   student_id  INT NOT NULL  AUTO_INCREMENT  PRIMARY KEY,
   prospective BOOLEAN,
   std_id  INT ,
```

```
#  Time      Action                                    Message            Duration / Fetch
1  06:24:10  CREATE DATABASE IF NOT EXISTS isms        1 row(s) affected  0.000 sec
2  06:24:10  USE isms                                  0 row(s) affected  0.000 sec
3  06:24:10  CREATE TABLE IF NOT EXISTS student (  st...  0 row(s) affected  0.062 sec
4  06:24:10  CREATE TABLE IF NOT EXISTS user ( user_i...  0 row(s) affected  0.031 sec
5  06:24:10  CREATE TABLE IF NOT EXISTS message (   ...   0 row(s) affected  0.063 sec
6  06:24:10  CREATE TABLE IF NOT EXISTS conversation ...  0 row(s) affected  0.078 sec
```

In a Query tab, type ***use isms;*** *this action access to the new ISMS database and make the database ready for insertions. Open* ImportStudents.sql and execute the selected script. This script contains the insertion of data from Students. This script was created from a batch process that reads a the current raw data and converts a SQL insert script. All students will be added into Student table.

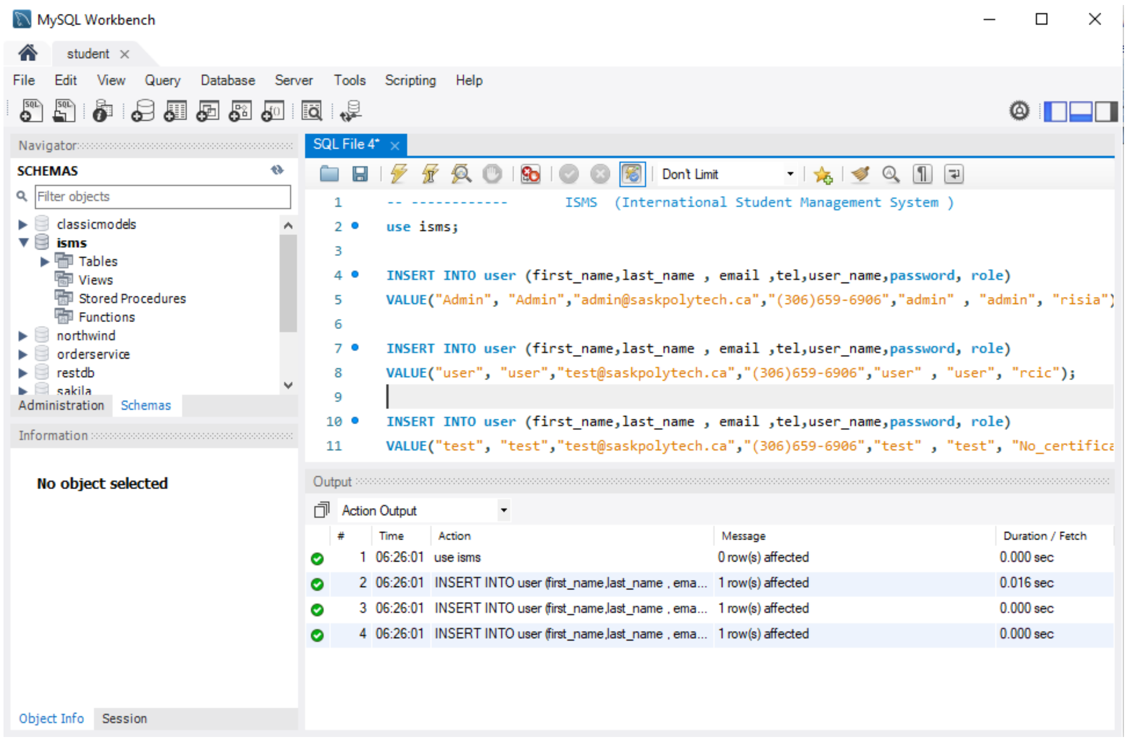


```
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
insert into student (prospective, std_id, first_name, middle_name, last_name, gender, birthdate, email, country, academic
```

No object selected

	#	Time	Action	Message	Duration / Fetch
✓	44	06:27:18	insert into student (prospective, std_id, first_name, middle_name, last_n...	1 row(s) affected	0.000 sec
✓	45	06:27:18	insert into student (prospective, std_id, first_name, middle_name, last_n...	1 row(s) affected	0.000 sec
✓	46	06:27:18	insert into student (prospective, std_id, first_name, middle_name, last_n...	1 row(s) affected	0.015 sec
✓	47	06:27:19	insert into student (prospective, std_id, first_name, middle_name, last_n...	1 row(s) affected	0.000 sec
✓	48	06:27:19	insert into student (prospective, std_id, first_name, middle_name, last_n...	1 row(s) affected	0.016 sec
✓	49	06:27:19	insert into student (prospective, std_id, first_name, middle_name, last_n...	1 row(s) affected	0.000 sec
✓	50	06:27:19	insert into student (prospective, std_id, first_name, middle_name, last_n...	1 row(s) affected	0.015 sec

We have test data for Users , this data will help us to login into the system. Open CreateUser.sql and execute the selected script. It will insert users into user table.

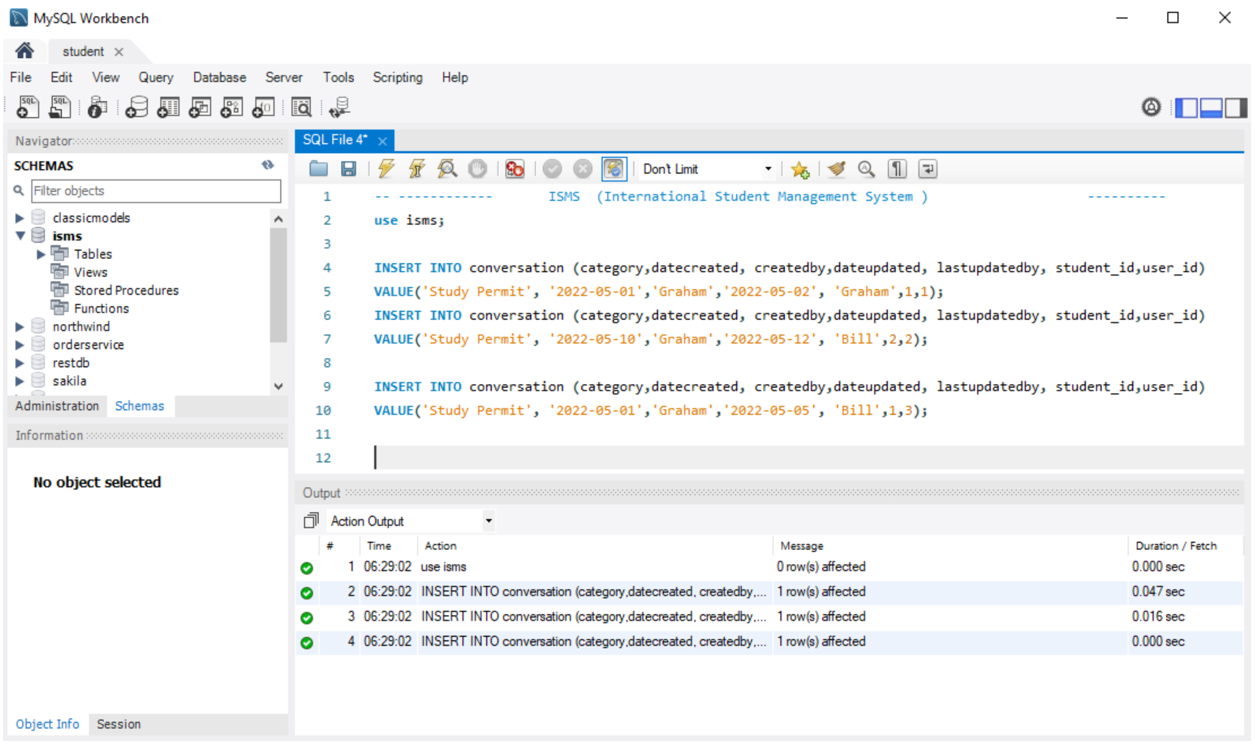


```
-- ------------       ISMS  (International Student Management System )
use isms;

INSERT INTO user (first_name,last_name , email ,tel,user_name,password, role)
VALUE("Admin", "Admin","admin@saskpolytech.ca","(306)659-6906","admin" , "admin", "risia")

INSERT INTO user (first_name,last_name , email ,tel,user_name,password, role)
VALUE("user", "user","test@saskpolytech.ca","(306)659-6906","user" , "user", "rcic");


INSERT INTO user (first_name,last_name , email ,tel,user_name,password, role)
VALUE("test", "test","test@saskpolytech.ca","(306)659-6906","test" , "test", "No_certifica
```

No object selected

	#	Time	Action	Message	Duration / Fetch
✓	1	06:26:01	use isms	0 row(s) affected	0.000 sec
✓	2	06:26:01	INSERT INTO user (first_name,last_name , ema...	1 row(s) affected	0.016 sec
✓	3	06:26:01	INSERT INTO user (first_name,last_name , ema...	1 row(s) affected	0.000 sec
✓	4	06:26:01	INSERT INTO user (first_name,last_name , ema...	1 row(s) affected	0.000 sec

Open CreateConversation.sql and execute the selected script. This script will insert test conversation from users with students.



```
-- ------------        ISMS  (International Student Management System )          ----------
use isms;

INSERT INTO conversation (category,datecreated, createdby,dateupdated, lastupdatedby, student_id,user_id)
VALUE('Study Permit', '2022-05-01','Graham','2022-05-02', 'Graham',1,1);
INSERT INTO conversation (category,datecreated, createdby,dateupdated, lastupdatedby, student_id,user_id)
VALUE('Study Permit', '2022-05-10','Graham','2022-05-12', 'Bill',2,2);

INSERT INTO conversation (category,datecreated, createdby,dateupdated, lastupdatedby, student_id,user_id)
VALUE('Study Permit', '2022-05-01','Graham','2022-05-05', 'Bill',1,3);
```

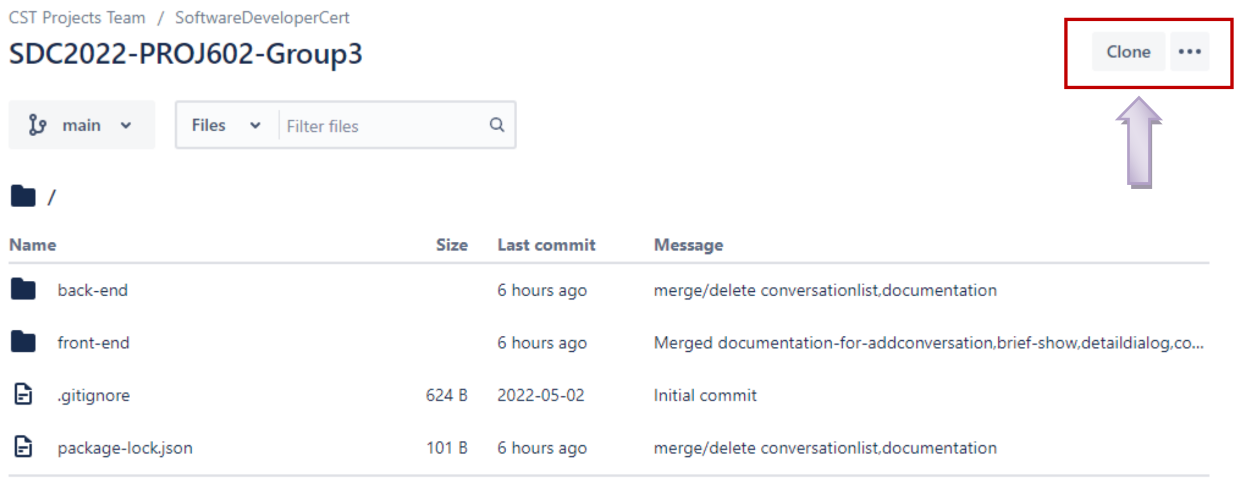
#	Time	Action	Message	Duration / Fetch
1	06:29:02	use isms	0 row(s) affected	0.000 sec
2	06:29:02	INSERT INTO conversation (category,datecreated, createdby,...	1 row(s) affected	0.047 sec
3	06:29:02	INSERT INTO conversation (category,datecreated, createdby,...	1 row(s) affected	0.016 sec
4	06:29:02	INSERT INTO conversation (category,datecreated, createdby,...	1 row(s) affected	0.000 sec

Clone Application Project from Bitbucket Repository

To install the application we have to clone our project into Visual Studio Code. First we have to login into Bitbucket and find the SDC2022-PROJ602-Group3.

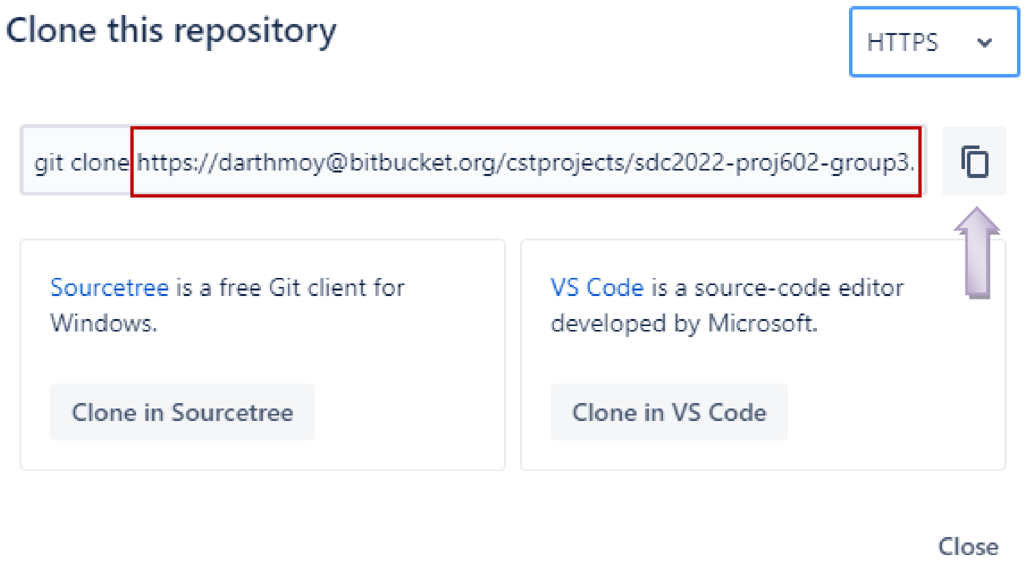
Clone SDC2022-PROJ602-Group3

Once you are at the SDC20200-PROJ602-Group3 folder, click on Clone button.

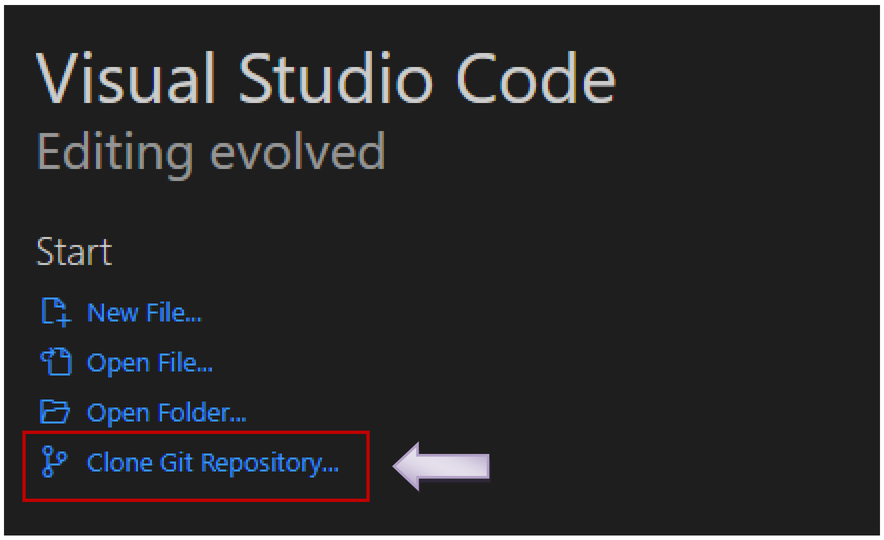


Name	Size	Last commit	Message
back-end		6 hours ago	merge/delete conversationlist,documentation
front-end		6 hours ago	Merged documentation-for-addconversation,brief-show,detaildialog,co...
.gitignore	624 B	2022-05-02	Initial commit
package-lock.json	101 B	6 hours ago	merge/delete conversationlist,documentation

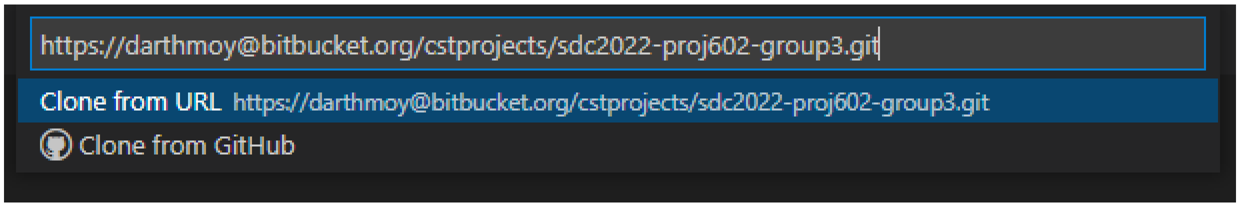
A popup screen will shows up with the provided URL to clone. There are more options to clone this project using other alternatives. Copy the URL shown starting from https:…


Clone this repository
HTTPS
git clone https://darthmoy@bitbucket.org/cstprojects/sdc2022-proj602-group3.
Sourcetree is a free Git client for Windows.
Clone in Sourcetree
VS Code is a source-code editor developed by Microsoft.
Clone in VS Code
Close

Open Visual Studio Code and click on “Clone Git Repository”



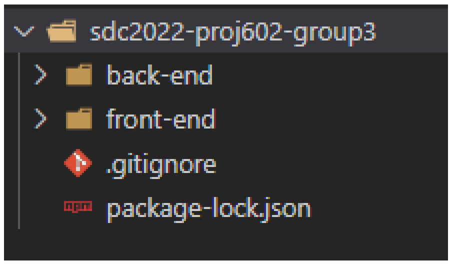
A popup input text shows up from the top of the screen. Paste the URL you copied from Bitbucket and press enter.



```
https://darthmoy@bitbucket.org/cstprojects/sdc2022-proj602-group3.git
```

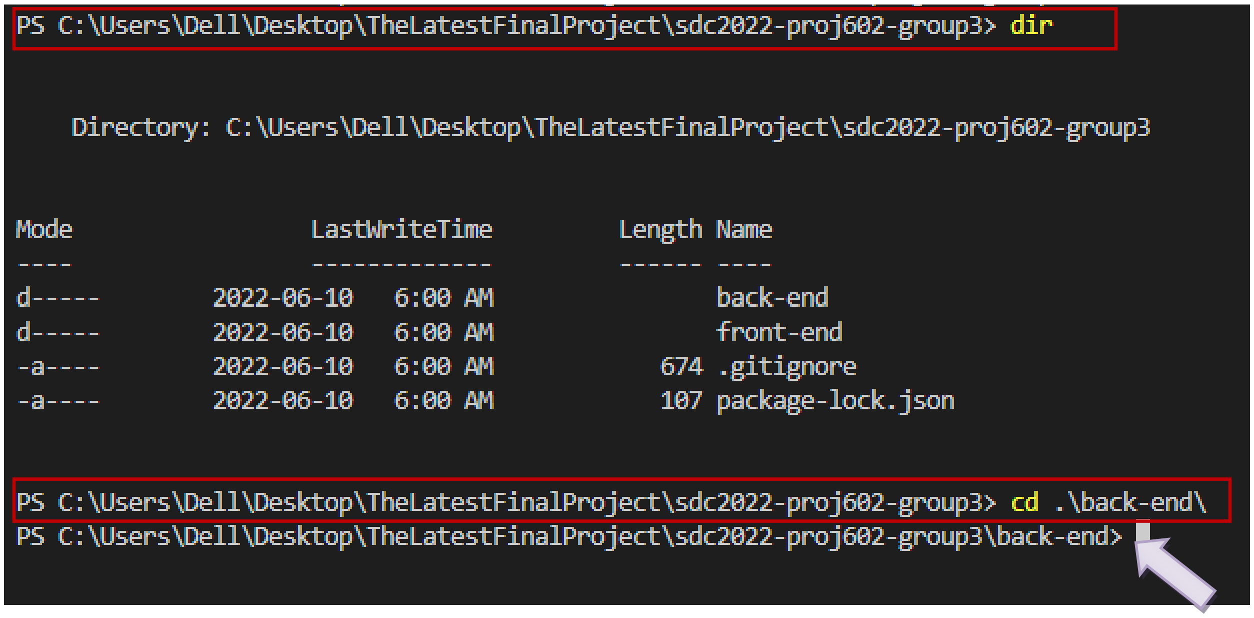
Clone from URL https://darthmoy@bitbucket.org/cstprojects/sdc2022-proj602-group3.git
Clone from GitHub

The project will download from repository. The IDE will notify that the complete project is ready to be used. You will see that back-end and front-end are now available. Browse around the project to identify the projects structure.



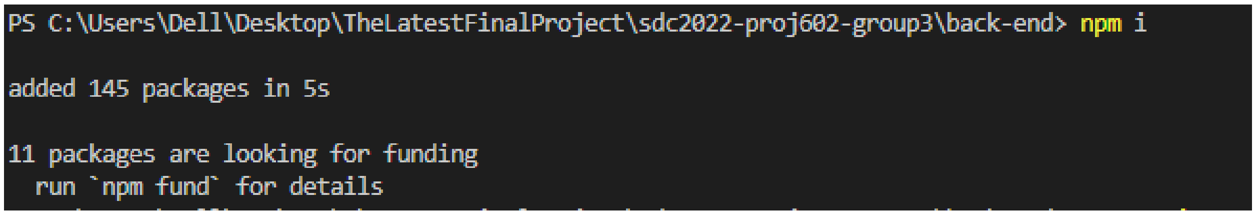
Installing Backend

Open new terminal in Visual Studio Code and type **dir**, you will see both projects. To install backend, type **cd back-end**, this action will take you to the backend directory.



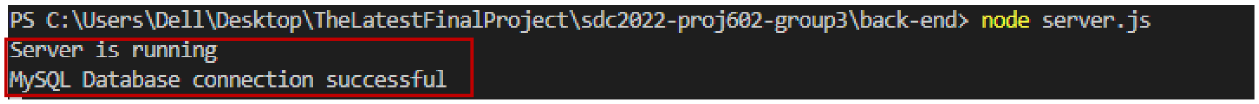
```
PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3> dir


    Directory: C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2022-06-10   6:00 AM                back-end
d-----        2022-06-10   6:00 AM                front-end
-a----        2022-06-10   6:00 AM            674 .gitignore
-a----        2022-06-10   6:00 AM            107 package-lock.json


PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3> cd .\back-end\
PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3\back-end>
```

Install server app by typing **npm i**. This action will install the node modules and libraries used.



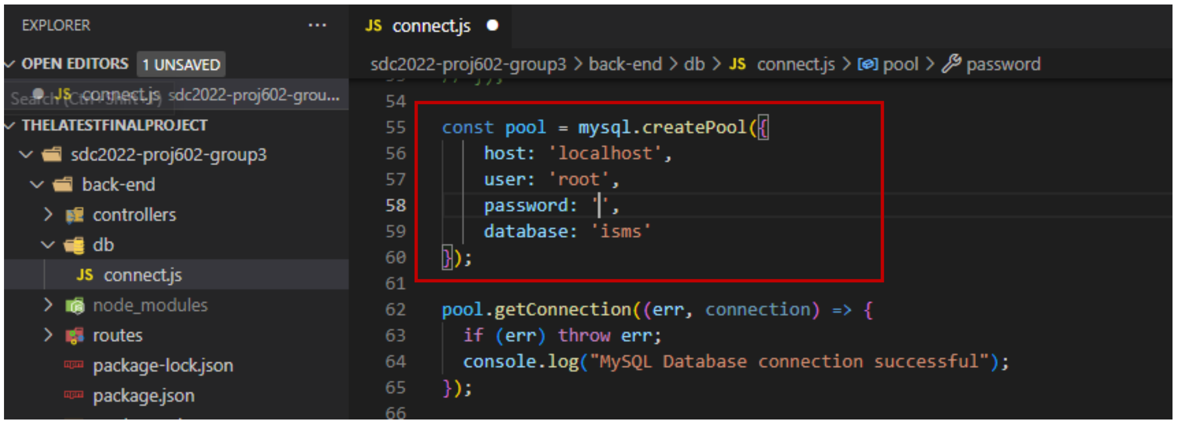
```
PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3\back-end> npm i

added 145 packages in 5s

11 packages are looking for funding
  run `npm fund` for details
```

To run the server, type either **node server.js** or **npm start**. To verify that the server is running properly, two lines will appear **Server is Running / MySQL Database connection successful**. Server is running and ready and the MySQL Database is connected.



```
PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3\back-end> node server.js
Server is running
MySQL Database connection successful
```

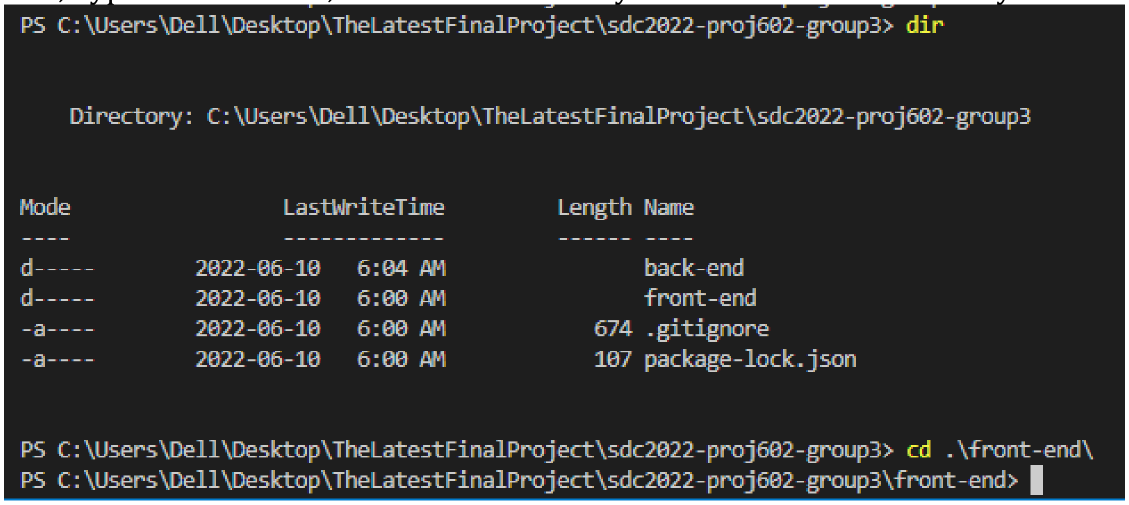
Note, if an exception threw instead of those two lines, check the database connection and change correct information.



```
const pool = mysql.createPool({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'isms'
});

pool.getConnection((err, connection) => {
  if (err) throw err;
  console.log("MySQL Database connection successful");
});
```

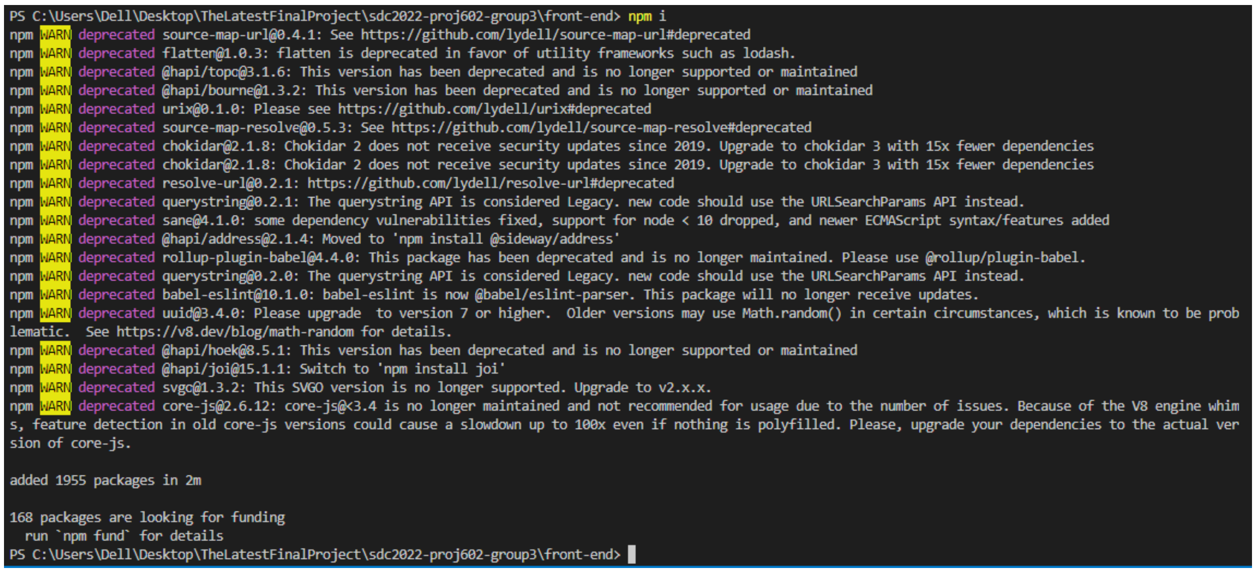
Installing Frontend

Open new terminal in Visual Studio Code and type **dir**, you will see both projects. To install frontend, type **cd front-end**, this action will take you to the frontend directory.



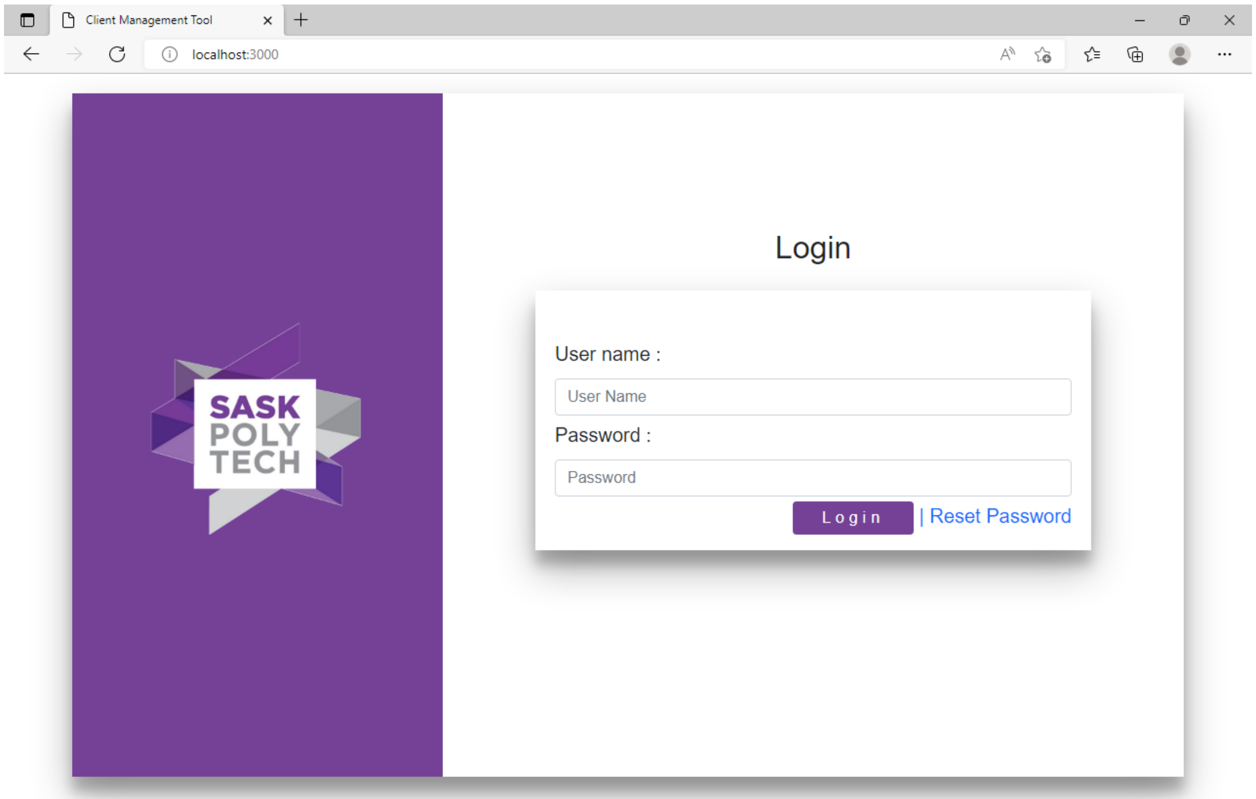
```
PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3> dir


    Directory: C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2022-06-10   6:04 AM                back-end
d-----        2022-06-10   6:00 AM                front-end
-a----        2022-06-10   6:00 AM            674 .gitignore
-a----        2022-06-10   6:00 AM            107 package-lock.json


PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3> cd .\front-end\
PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3\front-end>
```

Install app by typing **npm i**. This action will install the node modules and libraries used. It might show warns from deprecated modules, this situation won't affect to run our application.



```
PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3\front-end> npm i
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated flatten@1.0.3: flatten is deprecated in favor of utility frameworks such as lodash.
npm WARN deprecated @hapi/topo@3.1.6: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated @hapi/bourne@1.3.2: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated chokidar@2.1.8: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x fewer dependencies
npm WARN deprecated chokidar@2.1.8: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x fewer dependencies
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated querystring@0.2.1: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
npm WARN deprecated sane@4.1.0: some dependency vulnerabilities fixed, support for node < 10 dropped, and newer ECMAScript syntax/features added
npm WARN deprecated @hapi/address@2.1.4: Moved to 'npm install @sideway/address'
npm WARN deprecated rollup-plugin-babel@4.4.0: This package has been deprecated and is no longer maintained. Please use @rollup/plugin-babel.
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
npm WARN deprecated babel-eslint@10.1.0: babel-eslint is now @babel/eslint-parser. This package will no longer receive updates.
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances, which is known to be prob
lematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated @hapi/hoek@8.5.1: This version has been deprecated and is no longer supported or maintained
npm WARN deprecated @hapi/joi@15.1.1: Switch to 'npm install joi'
npm WARN deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.
npm WARN deprecated core-js@2.6.12: core-js@<3.4 is no longer maintained and not recommended for usage due to the number of issues. Because of the V8 engine whim
s, feature detection in old core-js versions could cause a slowdown up to 100x even if nothing is polyfilled. Please, upgrade your dependencies to the actual ver
sion of core-js.

added 1955 packages in 2m

168 packages are looking for funding
  run `npm fund` for details
PS C:\Users\Dell\Desktop\TheLatestFinalProject\sdc2022-proj602-group3\front-end>
```

To run the application, type cd type either **npm start**. To verify that the application is running a browser will show up with the Login page.


Client Management Tool
localhost:3000
SASK POLY TECH
Login
User name :
User Name
Password :
Password
Login
| Reset Password