Y2 Project - StudentHub

C00296913 - Reanielle Broas

Description

Student Hub provides an interface for student's, professor's, and admin's to manage notes, courses, grades, and timetables within the system. It aids the transfer of notes from professor to student(s). It makes it easier for them to have access to information when needed, such as timetables and grades.

IMPORTANT LOG-IN INFO FOR DEMO:

emails:		passwords:
student@email.com	~~	pass123
professor@email.com	~~	prof456
admin@email.com	~~	admin789

PERMISSIONS/REQUIREMENTS:

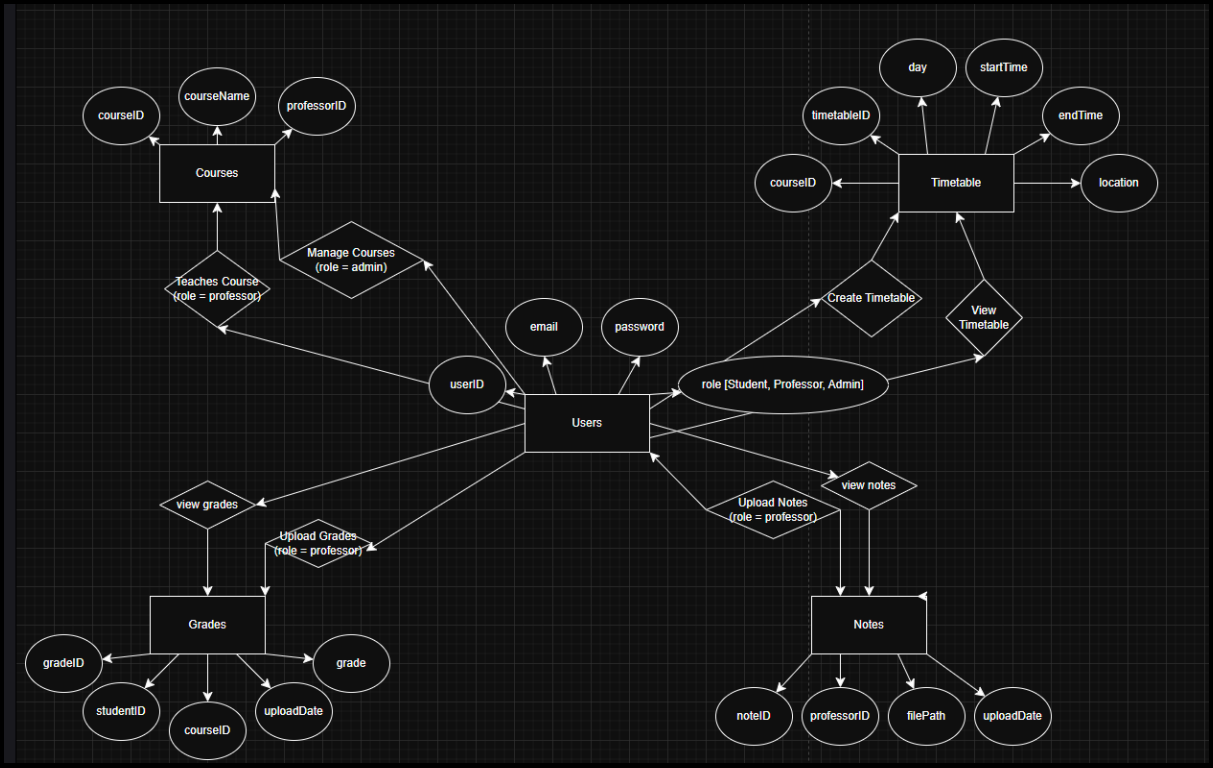
Student can ONLY ACCESS in dashboard:
- View Notes
- View Grades
- View Timetable

Professors HAVE ACCESS to:
- View Notes
- View Grades
- View Timetable
- Upload Notes
- Upload Grades
- Upload Timetable
- Manage Hub (Coming Soon)

Admins HAVE ACCESS to:
- Everything
- Manage Users

ER DIAGRAM

C00296913 - Reanielle Broas



SQL DOCUMENTATION

C00296913 - Reanielle Broas

Database Used: SQLite

CREATE SQL:

CREATE USERS

```
C:\Users\quan2\OneDrive\Desktop\sqlite-tools-win-x64-3490100>sqlite3 studenthub.db
SQLite version 3.49.1 2025-02-18 13:38:58
Enter ".help" for usage hints.
sqlite> CREATE TABLE Users(
(x1...> userID TEXT PRIMARY KEY,
(x1...> email TEXT UNIQUE,
(x1...> password TEXT,
(x1...> role TEXT CHECK(role IN('Student', 'Professor', 'Admin'))
(x1...> );
sqlite>
```

CREATE COURSES

```
sqlite> CREATE TABLE Courses (
(x1...> courseID INTEGER PRIMARY KEY AUTOINCREMENT,
(x1...> courseName TEXT,
(x1...> professorID TEXT,
(x1...> FOREIGN KEY (professorID) REFERENCES Users(userID)
(x1...> );
sqlite>
```

CREATE GRADES

```
sqlite> CREATE TABLE Grades(
(x1...> gradeID INTEGER PRIMARY KEY AUTOINCREMENT,
(x1...> studentID TEXT,
(x1...> courseID INTEGER,
(x1...> grade REAL,
(x1...> uploadDate TEXT,
(x1...> FOREIGN KEY (studentID) REFERENCES Users(userID),
(x1...> FOREIGN KEY (courseID) REFERENCES Courses(courseID)
(x1...> );
sqlite>
```

CREATE NOTES

```
sqlite> CREATE TABLE Notes(
(x1...> noteID INTEGER PRIMARY KEY AUTOINCREMENT,
(x1...> filePath TEXT NOT NULL,
(x1...> uploadDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
(x1...> , professorID TEXT,
(x1...> FOREIGN KEY (professorID) REFERENCES Users(userID)
(x1...> );
```

CREATE TIMETABLE

```
sqlite> CREATE TABLE Timetable(
(x1...> timetableID INTEGER PRIMARY KEY AUTOINCREMENT,
(x1...> courseID INTEGER,
(x1...> day TEXT,
(x1...> startTime TEXT,
(x1...> endTime TEXT,
(x1...> location TEXT,
(x1...> FOREIGN KEY (courseID) REFERENCES Courses(courseID)
(x1...> );
sqlite> .exit
```

.schema

```
sqlite> .schema
CREATE TABLE Users(
userID TEXT PRIMARY KEY,
email TEXT UNIQUE,
password TEXT,
role TEXT CHECK(role IN('Student', 'Professor', 'Admin'))
);
CREATE TABLE Courses (
courseID INTEGER PRIMARY KEY AUTOINCREMENT,
courseName TEXT,
professorID TEXT,
FOREIGN KEY (professorID) REFERENCES Users(userID)
);
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE Grades(
gradeID INTEGER PRIMARY KEY AUTOINCREMENT,
studentID TEXT,
courseID INTEGER,
grade REAL,
uploadDate TEXT,
FOREIGN KEY (studentID) REFERENCES Users(userID),
FOREIGN KEY (courseID) REFERENCES Courses(courseID)
);
sqlite> CREATE TABLE Notes(
(x1...> noteID INTEGER PRIMARY KEY AUTOINCREMENT,
(x1...> filePath TEXT NOT NULL,
(x1...> uploadDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
(x1...> , professorID TEXT,
(x1...> FOREIGN KEY (professorID) REFERENCES Users(userID)
(x1...> );
);
CREATE TABLE Timetable(
timetableID INTEGER PRIMARY KEY AUTOINCREMENT,
courseID INTEGER,
day TEXT,
startTime TEXT,
endTime TEXT,
location TEXT,
FOREIGN KEY (courseID) REFERENCES Courses(courseID)
);
sqlite>
```

INSERTING TEST DATA:

```
sqlite> INSERT INTO Users(userID, email, password, role) VALUES
   ...> ('TestStudent', 'student@email.com', 'pass123'
(x1...> ,'Student'),
   ...> ('TestProfessor', 'professor@email.com','prof456','Professor'),
   ...> ('A001', 'admin@email.com', 'admin789', 'Admin');
sqlite> INSERT INTO Courses(courseName, professorID) VALUES
   ...> ('OOP', 'TestProfessor'),
   ...> ('SADT', 'TestProfessor');
sqlite> INSERT INTO Grades(studentID, courseID, grade, uploadDate) VALUES\
   ...> ('TestStudent', 1, 85.4, '2025-03-09');
Parse error: unrecognized token: "\"
  des(studentID, courseID, grade, uploadDate) VALUES\ ('TestStudent', 1, 85.4, '
                                     error here ---^
sqlite> INSERT INTO Grades(studentID, courseID, grade, uploadDate) VALUES
   ...> ('TestStudent', 1, 85.4, '2025-03-09');
sqlite> INSERT INTO Notes(noteID, professorID, filePath) VALUES
   ...> ('1', 'TestProfessor','C:\Users\renel\OneDrive\Desktop\Y2_StudentHub\notetxt.txt');
sqlite> INSERT INTO Timetable (courseID, day, startTime, endTime, location) VALUES
   ...> (1, 'Monday', '09:00', '11:00'
(x1...> , 'Room A101');
```

DATABASE TABLES:

SELECT * FROM USERS

```
sqlite> SELECT * FROM Users;
userID         email                password  role
-------------  -------------------  --------  ---------
TestStudent    student@email.com    pass123   Student
TestProfessor  professor@email.com  prof456   Professor
A001           admin@email.com      admin789  Admin
sqlite>
```

SELECT * FROM COURSES

```
sqlite> SELECT * FROM Courses;
courseID  courseName  professorID
--------  ----------  -------------
1         OOP         TestProfessor
2         SADT        TestProfessor
sqlite>
```

SELECT * FROM GRADES

```
sqlite> SELECT * FROM Grades;
gradeID  studentID    courseID  grade  uploadDate
-------  -----------  --------  -----  ----------
1        TestStudent  1         85.4   2025-03-09
```

SELECT * FROM NOTES

```
sqlite> SELECT * FROM Notes;
1|TestProfessor|C:\Users\renel\OneDrive\Desktop\Y2_StudentHub\notetxt.txt|2025-04-11 12:02:36
```

SELECT * FROM TIMETABLE

```
sqlite> SELECT * FROM Timetable;
timetableID  courseID  day     startTime  endTime  location
-----------  --------  ------  ---------  -------  ---------
1            1         Monday  09:00      11:00    Room A101
sqlite>
```

INTERESTING CODE SNIPPETS

C00296913 - Reanielle Broas

FILE CHOOSER

```
// file chooser
            JTextField filePathField = new JTextField(30);
            filePathField.setEditable(false);
            filePathField.setMaximumSize(new Dimension(400, 30));
            JButton browseButton = new JButton("Browse");

            // file chooser functionality
            browseButton.addActionListener(e -> {
                JFileChooser fileChooser = new JFileChooser();
                int result = fileChooser.showOpenDialog(this);
                if (result == JFileChooser.APPROVE_OPTION) {
                    File selectedFile = fileChooser.getSelectedFile();
                    filePathField.setText(selectedFile.getAbsolutePath());
                }
            });
```

FILE UPLOAD FUNCTIONALITY

```
// upload button functionality
            uploadButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    String filePath = filePathField.getText();
                    if (filePath.isEmpty()) {
                        JOptionPane.showMessageDialog(p5UploadNotes.this,
"Please select a file to upload.", "Error", JOptionPane.ERROR_MESSAGE);
                        return;
                    }

                    // validate file format
                    if (!isSupportedFormat(filePath)) {
                        JOptionPane.showMessageDialog(p5UploadNotes.this,
"Unsupported file format. Please upload a PDF, DOCX, or TXT file.", "Error",
JOptionPane.ERROR_MESSAGE);
                        return;
```

```
                    }

                    // upload file to database
                    if (uploadNotesToDatabase(professorID, filePath)) {
                        JOptionPane.showMessageDialog(p5UploadNotes.this,
"Notes uploaded successfully!");
                        filePathField.setText(""); // clear the file path
field
                    } else {
                        JOptionPane.showMessageDialog(p5UploadNotes.this,
"Failed to upload notes. Please try again.", "Error",
JOptionPane.ERROR_MESSAGE);
                    }
                }
            });
```

DAY/TIME ERROR HANDLING

```
// Validate day
            String[] validDays = {"Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday", "Sunday"};
            boolean isValidDay = false;
            for (String validDay : validDays) {
                if (validDay.equalsIgnoreCase(day)) {
                    isValidDay = true;
                    break;
                }
            }
            if (!isValidDay) {
                JOptionPane.showMessageDialog(this, "Invalid day. Please
enter a valid day (e.g., Monday, Tuesday).", "Input Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }

            // Validate time format
            if (!isValidTime(startTime)) {
                JOptionPane.showMessageDialog(this, "Invalid start time.
Please enter a valid time in HH:MM format (e.g., 09:30).", "Input Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            if (!isValidTime(endTime)) {
                JOptionPane.showMessageDialog(this, "Invalid end time.
Please enter a valid time in HH:MM format (e.g., 17:45).", "Input Error",
JOptionPane.ERROR_MESSAGE);
                return;
```

```
}
```