Revised Project Idea:

Our idea for the ER diagram changed a bit after talking to Professor Riondatto. In particular, we realized that relationships can have their own attributes, which is why included the status attribute for the relationships between students and courses. We also realized that the existing particpation relationship between meeting and instructor was not sufficient to capture the complete database, which is why we extended another relationship between instructor and meetings. The same extended relationship applies towards students and meetings. These changes now completely describe our vision for this new relational database and will allow us to proceed forward with the software development of this project.

Role Assignments:

- **Kayikunmi**: Front-End Development Lead
- **Sydney**: Back-End Development Lead
- **Maria**: Project Manager (Oversees the flow of the project. Also responsible for front-end + SQL)
- **Cuong**: Full Stack Developer (Front + Back)

Questions:

1) Do we have to host our website on the server, or can we use AWS or anything else?
2) How common is it to have a website and a database hosted on the server?
3) Can we develop the website locally?
1) How extensive is the enrollment history of a student? (for Janna, customer)
2) How long are the notes? (for Janna, customer)
3) When can we expect some data?

Software Installed:

These are some of the softwares that we have installed onto the server. We have some questions/confusion with regards to what we will be hosting on the server (if its only the database or if its the entire website that we build). Depending on the answer to that question, the softwares we download might change.

- Postman: Need it to connect the database to the API we build
- Snap :
- Npm, NodeJS: Used for reactJS

Translation of the ER Schema into a Relational Schema:

ENTITIES:

CREATE TABLE **Students** (
  Email VARCHAR(100) NOT NULL,
  PRIMARY KEY (Email),
  First_Name VARCHAR(50) NOT NULL,
  Last_Name VARCHAR(50) NOT NULL,
  Preferred_Name VARCHAR(50) NOT NULL,
  Pronouns VARCHAR(50),
  Campus ENUM (UMass, Holyoke, Amherst, Hampshire) NOT NULL,
  Enrollment_History VARCHAR(50) NOT NULL,
  Academic_Career VARCHAR(50) NOT NULL,
  Graduation_Year VARCHAR(255) NOT NULL,
  Other_Notes TEXT(500),
  Phone VARCHAR(50),
 );

CREATE TABLE **Meetings** (
  Day DATE NOT NULL, → DATE doesn't require a length specification
  Time TIME NOT NULL, → TIME can take a length specification, but for number of decimals after seconds.
  Meeting_Type VARCHAR(20) NOT NULL,
  Location ENUM NOT NULL,
  PRIMARY KEY (Day,Time,Location)
)

CREATE TABLE **Courses**(
  scheduleNum INTEGER,
  courseNum CHAR(20),
  Semester CHAR(6),
  lang CHAR(15),
  Program, CHAR(4),
  ConversationsNum, INTEGER
  TutorialNum INTEGER,
  academicYear INTEGER,
  courseName CHAR(20)

```
        PRIMARY KEY (courseNum,semester,academicYear)
 )


CREATE TABLE Instructors (
    email VARCHAR(100) PRIMARY KEY,
    First_Name VARCHAR(50) NOT NULL,
    Last_Name VARCHAR(50) NOT NULL,
    Preferred_Name VARCHAR(50),
    Pronouns VARCHAR(50),
    Role VARCHAR(50),
    Academic_Career VARCHAR(50),
    Languages_Taught VARCHAR(255),
    Campus VARCHAR(50),
    Phone INT,
    Graduation_Year INT,
    Approved_to_Hire BOOLEAN,
    Paperwork_Status VARCHAR(50),
    Notes TEXT,  -- Combining Other Notes and Interview Notes
    Hiring_History TEXT
);
```

RELATIONSHIPS:

```
CREATE TABLE isRegistered(
        courseNum CHAR(20),
        Semester CHAR(6),
        academicYear INTEGER,
        status VARCHAR(50)
        email VARCHAR(100) NOT NULL
        FOREIGN KEY (courseNum)
            REFERENCES Courses,
        FOREIGN KEY (academicYear)
            REFERENCES Courses,
        FOREIGN KEY (Semester)
            REFERENCES Courses,
         FOREIGN KEY (Email)
            REFERENCES (Students)
```

```
        PRIMARY KEY (courseNum,semester,academicYear, email)
)

CREATE TABLE isEnrolled(
        courseNum CHAR(20),
        Semester CHAR(6),
        academicYear INTEGER,
        status VARCHAR(50)
        email VARCHAR(100) NOT NULL
        FOREIGN KEY (courseNum)
              REFERENCES Courses,
        FOREIGN KEY (academicYear)
              REFERENCES Courses,
        FOREIGN KEY (Semester)
              REFERENCES Courses,
         FOREIGN KEY (Email)
            REFERENCES (Students)
        PRIMARY KEY (courseNum,semester,academicYear, email)
)


CREATE TABLE participatesIn(
        role CHAR(20)
        email VARCHAR(100) NOT NULL
        day date
        time time
        location ENUM
        FOREIGN KEY (email)
              REFERENCES Students,
        FOREIGN KEY (day)
              REFERENCES Meetings,
         FOREIGN KEY (date)
            REFERENCES (Meetings)
        FOREIGN KEY (location)
            REFERENCES (Meetings)
        PRIMARY KEY (email,date,location,email)


CREATE TABLE Leads(
        role CHAR(20)
```

```
        email VARCHAR(100) NOT NULL
        day date
        time time
        location ENUM
        FOREIGN KEY (email)
             REFERENCES Instructors,
        FOREIGN KEY (day)
             REFERENCES Meetings,
         FOREIGN KEY (date)
            REFERENCES (Meetings)
        FOREIGN KEY (location)
            REFERENCES (Meetings)
        PRIMARY KEY (email,date,location,email)
)


CREATE TABLE taughtBy(
        role CHAR(20)
        courseNum CHAR(20)
        semester CHAR(6)
        academicYear INTEGER
        email VARCHAR(100)
        FOREIGN KEY (courseNum)
                REFERENCES Courses
        FOREIGN KEY (semester)
                REFERENCES Courses
        FOREIGN KEY (academicYear)
                REFERENCES Courses
        FOREIGN KEY (email)
                REFERENCES Instructors
        PRIMARY KEY ( courseNum, semester, academicYear, email)


CREATE TABLE SpireStatus(
        email CHAR(11),
        courseNum CHAR(20) NOT NULL,
        Status ENUM (enrolled, notEnrolled),
        PRIMARY KEY (email, courseNum),
        FOREIGN KEY (email) REFERENCES Students,
        FOREIGN KEY (courseNum) REFERENCES Courses
```

)

CREATE TABLE **When&Where** (
courseNum CHAR(20) NOT NULL,
semester CHAR(6) NOT NULL,
academicYear INTEGER NOT NULL,
day ENUM (mon,tues,wed,thurs,fri,sat,sun) NOT NULL,
time DATETIME NOT NULL,
location VARCHAR(100) NOT NULL,

FOREIGN KEY (courseNum) REFERENCES Courses,
FOREIGN KEY (academicYear) REFERENCES Courses,
FOREIGN KEY (semester) REFERENCES Courses,
FOREIGN KEY (day) REFERENCES Meetings,
FOREIGN KEY (time) REFERENCES Meetings,
FOREIGN KEY (location) REFERENCES Meetings,
PRIMARY KEY (courseNum,semester,academicYear, day, time, location)
)