TEAM - 8 Requirements for Final Proj EVOL 2:

1. Functional Requirements

1.0 Stand-Alone Admin App

1. Stand-Alone app will process 3 csv files
    a. All students in the university (considered as initialization file)
        i. This will have pre-existing csv file containing the student password (random unique strings) , netid, emai, full name.
    b. All courses offered in the university (considered as initialization file)
    c. Lastly, csv file that maps students to courses
2. We will assume CSV file is all correct.
3. This app will parse the csv files and feed the database.
4. This app will never talk to the users directly. Its purpose is to build a database for all existing users (faculty & students) within the university
4. The user admin will not need authentication to run.
5. User passwords (excluding the admin) will need to be hashed and salted. The rest of the fields in the 3 csv files will not need to be encrypted.
7. The admin user just knows whose netid/email/password to remove or update or add.
8. We assume student is going to register to verbally tells the user admin to drop or add Admin drops and enrolls students to courses. The user admin therefore doesn't need to see the list of the currently enrolled students in a course to enroll a student in a course correctly. Same idea for dropping a student. The user admin will always correctly drop a student.
9. User will have the option to add a student to the University not just to a course.

1.1.a Server/Client Model App

1. Server will accept connection requests from multiple clients and respond to requests as needed concurrently.
2. We will be using TCP/IP sockets.
3. C/S app will send weekly attendance report. This attendance report will contain
    1. Only weekly attendance record (not cumulative)
    2. Cumulative attendance grade (If a student is marked as attended then it is 100% for that lecture. If tardy 80% for the lecture. If absent 0%)
4. Each course sends weekly report so if the student is enrolled in 5 courses, they receive 5 emails each week unless they decide to pause the weekly email-receiving function from the CLI.
5. If student chooses to not receive weekly email for a particular course, the server will not send email to that student for that particular course.

6. Anytime student's attendance record changes, the server will send an email to the student even if that student has paused their email subscription for a particular course.

1.2 Database

1. We will save salted and hashed student/professor user-authentication passwords
2. Any other data in the database will not be encrypted
3. Client sends request to server, and server will talk to the database.
4. Client should never be querying the database directly.
5. We will save all historical information in the database. No information will be deleted unless it was overwritten such client changing passwords.
6. We will keep track of attendance grades in the database and cumulative grades in the Attendance Record class

1.3 Authentication (Yixin)

1. Students or faculty will need to be authenticated to login to the server.
2. We assume students and faculty just know the password.
3. We give users the option to change their passwords. But we don't force password change at first login.
4. NetID is the username and the password is the password
5. Users cannot change their netID

1.3 Courses

1. Courses are added by the App admin via csv. And we assume this CSV is file always correct. See 1.0 for more info
2. Each course will have at least one section
3. Each section will be related to one teacher object in memory /entry in db
4. Atomic unit is the section and each section will have just one teacher

1.3 Teacher Activities on the Server

1. Teacher will select a course to take attendance for take attendance one by one fot each student enrolled in a course.
2. Teacher is able to change historical attendance records even if the student is no longer actively enrolled in a course.
3.Teacher will have the option to see particular students attendance records for a Course by searching by their netID.
4. Teacher will have the option to select a specific attendance sheet and see all students attendance report.

1.3 Student Activities on the Server

1. Students will have the option to see all the attendance records for a single course (displayed on the terminal and have the option to get it emailed to them)
2. If a student wants attendance reports for all the courses they are enrolled in, they will have to put in a request for each one of courses.
3. Students should be able to change notification (this is the weekly report) preferences. E.g., for ECE551 a student don't want notification but they can want a weekly notification for ECE580. Notification is an email. The default is to send a weekly email.
4. Student will have the option to change their preferred name. Their name change will be reflected in all the classes that they have enrolled in.
5. Student will have the option to change their login password. Student password must at least be 8 characters long.

2. Non-functional Requirements

1.1 Security

1. User password will be hashed and salted
2. User's other information will not be encrypted
3. Client/Server communication channel will not encrypted.
4. We assume no one is intercepting the conversation between the client and server.
5. Username and passwords will be sent as plain text to login from client to server. The server will salt and hash the password to check against the User Admin's saved hash in the database.

1.2 Reliability

1. User to C/S server communication should be resilient against data loss. How we ensure is up to the team. If the connection is dropped midway, our database should have the correct info (be in a valid state)

1.3 Portability

1. N/A

1.4 Constraints

2. Can use JDBC. Cannot use Jakarta or Hibernates.

TLDR:

Student Dos and Don'ts:

- Can change its password
- Can change their preferred name
- Can choose to select which courses to receive a weekly email from
- Can have the attendance report emailed to them on one-off basis

Teacher:

- Take attendance
- Edit previous attendance
- View attendance reports for a specific student

Admin Add:

- Parse all students in the university (considered as initialization file)
- Parse all courses in the university (considered as initialization file)
- Parse file mapping student to courses
- Enroll late enrollees (registered with school) to course
- Enroll late enrollees (not registered with school) in to course
- Drop student from a course