1 Purpose of the project

The overall goal of this project is to implement a single Java program that will back a new messaging application. The core of such a platform is a database system. In order to realize this goal, you will need to work as a member of a team which designs and develops a relatively large database application.

You must implement your program using Java, PostgreSQL, and JDBC. The first phase of the project will only involve generating schemas for all of your tables and populating them with example data.

This project focuses on the database backing the messaging application and not on the user interface. Hence, no graphical user interface should be produced for this project.

2 Phase 0 Specifics

2.1 Assemble teams

Organize yourselves into a team of 2-3 people and create a private project repository on GitHub. Add all team members as contributors to the repository. Further add your instructor and your TA as contributors to the repository (instructor and TA GitHub usernames can be found on the class website). Create a single file named team_members.txt file in your repository listing each team member’s GitHub username, Pitt ID (e.g., “ulf4”), and full name. Be sure to push team_members.txt file to your group’s repository by the deadline.

Be sure to share only 1 repository per team with the instructor/TA to avoid grading confusion.

3 Phase 1 Specifics

3.1 The database schema and example data

You should choose appropriate data types to make up the attributes of each relation that you create to represent miniworld that is described as follows. Be sure to state any assumptions that you make as SQL comments.

For each user in the messaging system, you will need to store their username, full name, and email.

Each user will also maintain a contact list. Any user can add any other user to their contacts. This addition is not automatically reciprocated.

Users can also belong to group chats. Your system should keep track of a name for each group chat on the platform, as well as its membership.

Finally, you should store messages. A message can be sent to either a single user or to a single group chat (exclusive or, you do not need to keep track of message sent to a user and also a group chat). You should keep track of the message text, the sender, and the date sent for each message. You can assume that messages are constrained to be less than 100 characters.
Once you have created a schema and integrity constraints for storing all of this information, you should generate sample data to insert into your tables. Generate the data to represent at least 100 users with an average contact list length of 10, 10 group chats, and 300 messages.

4 Submission

Your project will be collected by the TA via the GitHub repository that you have shared. To turn in your code, you must do three things by the deadline:

1. Make a commit to your project repository that represents what should be graded as your group’s submission for that phase. The message for this commit should be “Phase $X$ submission” where “$X$” is 0 or 1.

2. Push that commit to the GitHub repository that you have shared with the instructor and TA.

3. Send an email to the instructor and the TA with the title “[CS1555] Project phase $X$ submission” that includes a link to your repository on GitHub and the Git commit ID for the commit that should be graded. The commit ID is a 40 character string (a SHA-1 hash) and can be found on GitHub or as part of the output of the “git log” command.

Be sure to send this email to the instructor and TA by the deadline. Pushing the code to GitHub is not enough! If you do not email the instructor and TA a commit ID as specified above by the deadline, your submission for that phase will be considered late and you will receive a 0 for that phase.

5 Grading

The end result project will be graded on correctness (including transaction usage), robustness (error-checking, i.e., it produces user-friendly and meaningful error messages) and readability. Programs that fail to compile or run or connect to the database server earn zero and no partial points.

One secondary goal of this project is to help you start to incorporate feedback into a larger project. Hence, the TA will assign you a grade and give feedback on each phase of the assignment. By incorporating feedback given by the TA into later phases, you can earn back some of the points lost on earlier phases. For phase 1 and 2, the TA will assign you a general letter grade assessing the overall quality of your submission. Due to the fact that this grade can be improved through incorporating feedback, treat this letter grade as more of an indicator of how much work you need to do to improve this portion of the term project than what you have earned for that portion of the project.

Note that points deducted from phases 1 and 2 for issues that illustrate a lack of time or thought put into the project (e.g., unhandled database or compilation errors, schemas that fall far short of being able to handle the data needed to represent the mini world mentioned above, or inappropriate assumptions made in defining integrity constraints) cannot be earned back and will negatively affect your overall grade.

Because of this approach, you may find it to your benefit to submit phases 1 and 2 early. The sooner you notify the TA that you have completed a phase, the sooner the TA can begin grading that phase and provide you with feedback.
Be careful to ensure that you have addressed all the requirements of a phase when sending a submission email. Once the TA receives a submission email, the commit ID contained within that email is the one that will be graded for that phase (i.e., your group cannot send 2 submission emails for the same phase). Your submission for phase 3 will be considered your final submission for the term project. DO NOT send a submission email to the TA for phase 3 until you are completely done with your project and are satisfied that you have addressed all feedback from the TA.