# CS 166: Project Phase 3 Requirements

February 19, 2016

## 1 Introduction

In this phase you are provided a package including SQL schema, dataset, and a template user interface in Java. The dataset consists of data records that should be loaded into your database system. The template user interface in Java shows you how to connect to the Postgres database and how to execute SQL queries.

Additional extra points will be awarded to projects that use GUI interfaces and properly handle exceptional situation by providing user-friendly error messages.

Please follow the below steps to get started:

1. Make sure that your environment is setup correctly: variables $DB_NAME, $PGDATA, $PGPORT are defined, Postgres instance is started correctly

2. Download project.zip from iLearn

3. In the download directory execute the following command:

   ```
   unzip project.zip
   ```

   You will see that a directory named "project" will be created. This directory contains all necessary files to get started. More specifically it contains the following things:

   - **project/data** - contains the files, which will be used to populate your database
   - **project/sql/src/create_tables.sql** - SQL script creating the database relational schema. It also includes the commands to drop these tables.

- **project/sql/src/create_indexes.sql** - SQL script which creates database indexes. Initially is empty, you should add all your indexes to this file.
- **project/sql/src/load_data.sql** - SQL script for loading the data in your tables. The script loads each text file into the appropriate table. **Note that the file paths have to be changed to absolute paths in order to make it work**.
- **project/sql/scripts/create_db.sh** - shell script, which you should to setup your database.
- **project/java/src/Messenger.java** - A basic java User Interface to your Postgres database. You should modify this program and write all your SQL-specific code there.
- **project/java/scripts/compile.sh** - compiles&runs your java code.
- **project/java/lib/pg73jdbc3.jar** - The Postgres JDBC driver, which is necessary for your Java code.

4. Change path to data files in **project/sql/src/load_data.sql**. Use absolute paths to avoid ambiguity. After that your load statements should look like this:

```
COPY USER_LIST
FROM '/home/user/project/data/usr_list.csv'
WITH DELIMITER ';';
```

5. Execute **project/sql/scripts/create_db.sh** to load your database

6. Execute **project/java/scripts/compile.sh** to compile and run your Java client.

## 1.1 Java console application

If this is the first time you work with Java there is no need to be worried. You are provided with a template client written in Java. You are expected to extend this basic client and add the functionality, required for a complete system. An Introduction to Java/JDBC can also be found in your Textbook (Sections 6.2 6.3 of Database Management Systems (Third edition)).

In this phase we basically want to create an interactive console for non-sql users (i.e. users of out Online Messenger). You should therefore pay special attention to making the interface as intuitive as possible for regular user.

In order to run the template Java program you should first start the Postgres database as it was described in the Postgres Manual (or the quick start guide). Then you should execute shell script *project/java/scripts/compile.sh*

The script will compile and run Java source for the file *project/java/src/Messenger.java*. If you will add other Java source files to your project, please modify the script accordingly.

# 2 Functionality of the Online Messenger

Below you will find the basic requirements for the functionality of the system. On top of them you may implement as many additional features as you want. Throughout the project you should keep in mind that Messengers users are not SQL-aware. You should therefore make the user interface as intuitive as possible. We initially planned to provide functionality in the following three groups:

- Users
- Chats
- Messages

Below we provide the basic operations you must implement in your system.

1. Users
    - New User Registration: when a new user comes to the system, he/she can setup a new account through your interface, by providing necessary information.
    - User Login/Logout: user can use his/her username and password to login into the system. Once he/she logs in, a session will be maintained for him/her until logout (an example has been provided in the Java source code).
    - Delete Ones Own Account: once logging in, a user can select to delete his/her account. The system will check whether there are any other records referring to this account, like the ownership on some groups or publications. If there is, the system will return information stating that there are linked information for this account so the delete cannot be processed. Otherwise, this account will be deleted and the user will leave the system.

- Browse Contact\Block list. For contacts, list user status messages (if any) should be displayed here.
- Add\Delete other users from Contact\Block list.
- Browse list of current chats and start a new one by selecting initial members. Depending on the number of members chat should be marked as private or group. Chats are displayed in the chronological order of their last update time.

2. Chats

   - Initial sender of the chat is able to add\delete members of the chat. He can also delete the whole chat, in this case all chat messages will be destroyed.
   - Chat viewer should show last 10 messages from the chat. There should also be a possibility to load earlier messages. This loads will be retrieved from the database in batches of 10.
   - Messages in the chat are displayed in chronological order of their creation timestamp
   - A new chat should appear in user's chat list whenever he becomes a new member of the chat. All previous chat messages should be viewable by the user.

3. Messages

   - All chat members should be able to create a new message and edit\delete their own messages.
   - Each message should indicate an author, creation date and it's text.

# 3 Extra Credit

1. Good User Interface.

   A good user interface will bring more users to your application, and also more points in this phase. A user interface is good if it is:

   - Easy for users to explore features;
   - Robust in exceptional situations, like unexpected inputs;
   - Graphic interface supporting all required features.

2. Triggers and Stored Procedures.

   Instead of processing the workflow step by step, triggers and stored procedures can be used to handle a sequence of SQL command and to execute these sequences on some table events (inserts, updates, etc). For example, when a message has a self-destruction timestamp defined you can write a trigger which will be deleting this message and all related with it information. You can also find other opportunities for triggers and stored procedures.

   To submit your triggers and stored procedures with your project, please include them in the following location *project/sql/src/triggers.sql*

3. Performance Tuning

   The performance can be improved if index are used properly. **You will receive points only if the index will actually be used in your queries**. You should defend why you have chosen to use an index at some particular table. For your submission, you should put index declarations in *project/sql/src/create_indexes.sql*

4. Any Other Fancy Stuff...

   Please feel free to show any of your ideas to improve your project! The only thing required will be clear documentation!

# 4 Submission Guideline

1. Project Report

   You should provide a high level description (1-2 pages) of your implementation. You should describe which part each of you was working on and any problems\findings, you found along your way. In this document you should also include whatever is asked in the below individual descriptions.

   You should submit a single zip archive (**1 submission per group!**) via iLearn at least 1 hour before your presentation and notify TA about that via email. The available slots for the presentation will be posted through iLearn later.

2. Files

   The following files should be submitted:

(a) Create Tables, Bulkload Data Scripts.

If you have any schema or tables modifications you should include your changes into the files and leave necessary comments for them. Modify the following files: *project/sql/src/create_tables.sql*, *project/sql/src/load_data.sql*

(b) System Implementation.

Submit your source file(s). You should make sure that your code can compiles and run successfully. **Any special requirement for compiling and running should be stated clearly in your project report, or a README file which comes with your source code**. Please put all your source code within the *project/java/src* directory.

(c) Other scripts, like triggers, stored procedures, and indexes.

You should provide descriptions for these features and include all your scripts within the *project/sql/src* directory.