

Queens College, CUNY, Department of Computer Science
Software Engineering
CSCI 370
Fall 2018

Instructor: Dr. Sateesh Mane

© Sateesh R. Mane 2018

due Sunday September 30, 2018

1 Project 1

1.1 General

- The overall project specification is to implement an email project.
- Various details of the project are listed below.
- The basic visualization is that of a client-server application. **See 1.2.**
- **The project must be self-contained.**
 1. It is not permitted to connect to Gmail or Yahoo, etc. servers.
 2. *Else what are you really doing?*
 3. The project must be 100% **your** work.
- **Trademark/copyright.** To avoid problems with trademark and copyright (no matter how minor the possibility), project submissions will be required to implement the following three domain names.
 1. `cq.edu`
 2. `yg.com` (It is available for sale!)
 3. `lnb.gov` (I used to work at Brookhaven National Lab `bnl.gov`.)

1.2 Client-server, database & GUI

- Client & Server

1. **The use of a client-server implementation is optional.**
2. The entire application may be “one program” not separate client and server programs.

- GUI

1. **There must be a visual display to users.**
2. This is part of the concept of a “user friendly” interface.
3. However the requirements for a “GUI” are not very strict.
4. Essentially, the term “GUI” means a minimal visual display.
5. *Contact me if you do not know how to write a GUI.*

- Database

1. **The use of a database is optional.**
2. **Use of SQL is optional.**
3. However, the application must be persistent across invocations.
4. Therefore the use of a persistent storage medium is mandatory.
5. It is acceptable if data is stored in text files instead of a database.
 - (a) The format of a text file should be in a way the looks like a database table.
 - (b) There must be a concept of a primary key.
 - (c) **A text file must not contain duplicate rows (violation of primary key).**
6. Two messages with the same sender, recipient, subject and text must be resolved and stored as different messages. (They should have different time stamps.)
 - (a) This requirement applies to storage of messages in either a text file or database.
 - (b) If a “reply” is sent to one of the two above messages, the application must identify which is the correct original message that the reply refers to.

1.3 Server

- Details of the server are left to the student.
- **The use of a server is optional.**
- **The items below describe “functionality” but can be implemented without a server.**
- Either a different server for each domain, or one server which supports all the domains.
- If there are multiple servers, the software architecture must explain how they communicate with each other.
- **A server must support client requests to create new email accounts for one of the above domains.**
- The client and server must respond gracefully (not crash or throw an unhandled exception) if an account name has already been registered by another client.
- The accounts must be persistent across invocations, i.e. it must be possible to switch off/shut down the server(s) and restart, and to verify all the email accounts are reloaded upon restart.
- Hence a persistent storage medium to store accounts is required.
- The details of the persistent storage medium (database or text file) are left to the student.

1.4 Client/GUI

- **All students/teams are expected to write a GUI, to display results.**
- **It is optional to implement the GUI as a “client” connecting to a server.**
- **The items below describe “functionality” but can be implemented without a server.**
- The messages must be persistent across invocations, i.e. it must be possible to switch off/shut down the client and restart, and to see all the emails upon restart.
- Hence a persistent storage medium to store messages is required.
- The details of the persistent storage medium (database or text file) are left to the student.
- The client must implement (i) an inbox folder, (ii) an outbox folder and (iii) a drafts folder.
 1. Messages in the drafts folder are read/write.
 2. Messages in the inbox and outbox folders are read only.
 3. We shall ignore the “cc” and “bcc” fields, implement “from” and “to” only.
 4. **The “to” field must support contain multiple recipient names.**
- **Suggestion**
 1. It may be a good idea to create a “Name” object, which contains a “nickname” as well as a full address.
 2. A Name object may contain a “nickname” `John` as well as the full address `John@domain`.
 3. Hence the “from” and “to” fields of an email display `John` not the full string `John@domain`.
 4. *This is a suggestion:* implementation details are left to the student.
- The client must implement “reply” and “forward” functions for emails.
- The client must implement a “send” function for an email in the drafts folder.
 1. If the “to” field is blank, an error message should be displayed.
 2. If the “to” field is not blank, a date stamp is inserted into the message (use the system clock) and the message is transferred to the outbox folder (and becomes read only).
 3. **Fraud protection: it must not be possible under any circumstances to edit the date stamp of an email.**
 4. If delivery fails because the “to” field not recognized, an error message should be displayed. (“Graceful handling of errors.”) The client/server should not crash or throw and unhandled exception.

1.5 Scalability/extensibility

- It should be perfectly obvious that I will ask students to implement a fourth domain name.
- Students must be able to demonstrate how easy is it to extend their design to accomodate an extra domain name.
- **An important part of the project grade is to demonstrate how flexible is the implementation, to support new/extra requirements.**

1.6 Update about user accounts (9/16/2018)

- The password should be **minimum 4 and maximum 12 characters.**
- The userid (including '@' and domain) should be **maximum 20 characters.**