### Software Engineering- CSC 4350 Spring 2017

An encryption and decryption system for message communication

**ADEPT**

**A**mani Konduru

Benjamin Garber (**D**aniel)

**E**dward Bull

**P**aul David Utesch

**T**eam

**2/28/2017**

Table of Contents

[Software Engineering- CSC 4350 Spring 2017 1](#_Toc476061413)

[I. Software Architecture 3](#_Toc476061414)

[II. Requirements Traceability Matrix (RTM) 3](#_Toc476061415)

[III. A Category Interaction Diagrams 6](#_Toc476061416)

[IV. Class interface 13](#_Toc476061417)

[V. WSD 15](#_Toc476061418)

[VI. Gantt Chart 15](#_Toc476061419)

[VII. Glossary 17](#_Toc476061420)

[VIII. Rationale 18](#_Toc476061421)

# Software Architecture

* SQLite and PostgreSQL
* Eclipse
* OS is Linux - Ubuntu

Ubuntu 16.04.1 x64 512 mb RAM 20gb storage

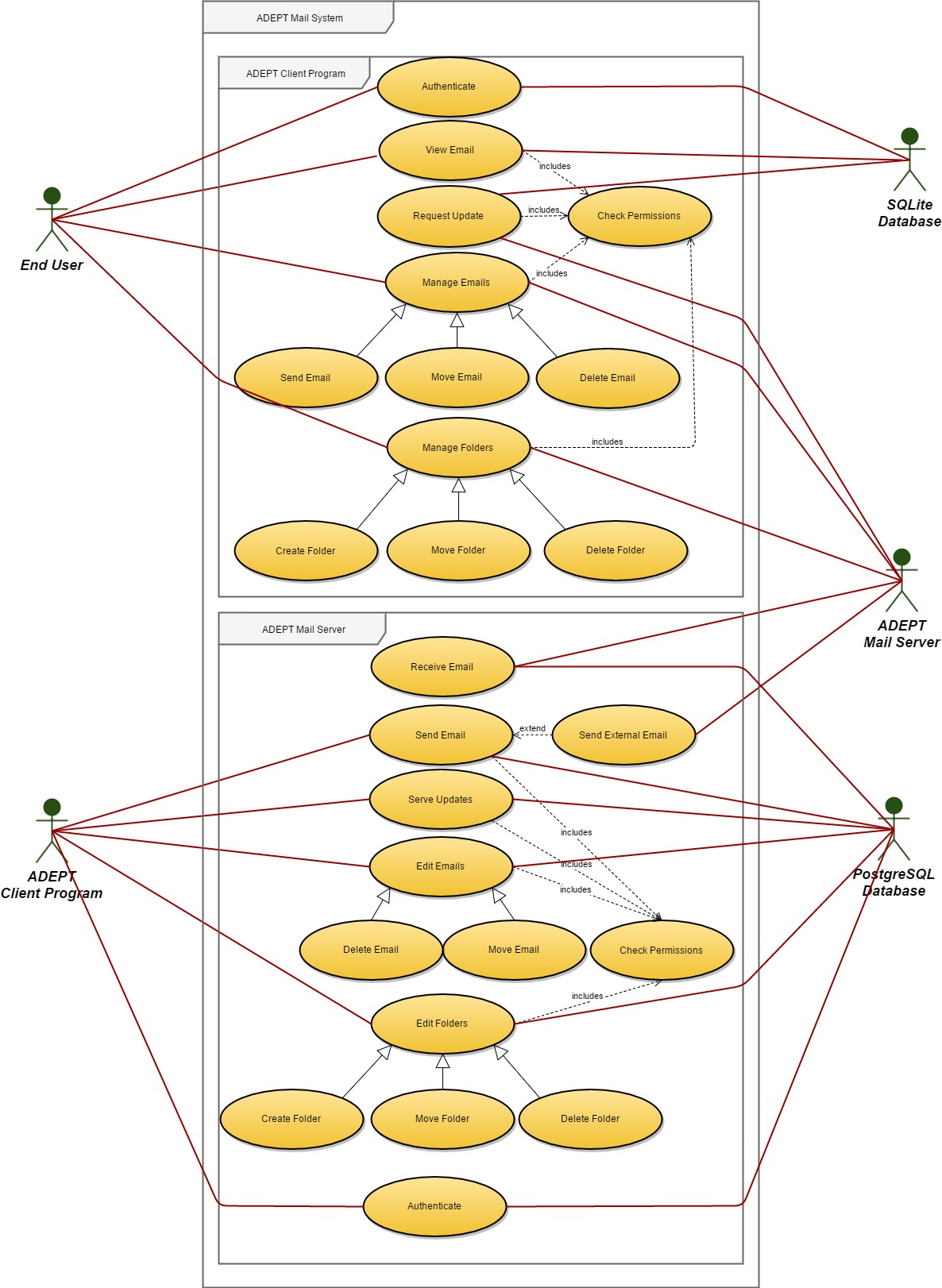
* Cloud based virtual machine-Digital Ocean droplet

# Requirements Traceability Matrix (RTM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entry**  **#** | **System Specification text** | **Type** | **Build** | **Use Case Name** | **Category** |
| 1 | The Adept Mail Server shall store user e-mails in a database. | SW | 1 | Send Email, Edit Emails | Server |
| 2 | The Adept Mail Server shall move user e-mails between user-designated mailboxes upon an authenticated request from that user. | SW | 2 | Edit Folders | Server |
| 3 | The Adept Mail Server shall delete user-designated e-mails from its database upon an authenticated request from that user. | SW | 2 | Delete Emails | Server |
| 4 | The Adept Mail Server shall serve user data when authenticated requests are received from the Adept Mail Client via a minimally compliant IMAP protocol. | SW | 1 | Serve Updates | Server |
| 5 | The Adept Mail Server shall send user emails from other Adept Mail Servers upon an authenticated request from that user via a minimally compliant SMTP protocol. | SW | 2 | Send Email, Send External Email | Server |
| 6 | The Adept Mail Server shall receive user emails from other Adept Mail Servers via a minimally compliant SMTP protocol. | SW | 2 | Receive Email | Server |
| 7 | The Adept Mail Server shall encrypt all incoming and outgoing connections using the TLS 1.2 standard. | SW | 1 | Receive Email, Send External Email, Serve Updates, Edit Emails, Edit Folders, Authenticate | Server |
| 8 | The Adept Mail Server shall support multiple concurrent connections. |  | 1 | Receive Email, Serve Updates, Edit Emails, Edit folders | Server |
| 9 | The Adept Mail Client shall request user email data from the Adept Mail Server via a minimally compliant IMAP protocol. | SW | 1 | Request Update | Client |
| 10 | The Adept Mail Client shall store user email data locally in a local database. | SW | 1 | Request Update | Client |
| 11 | The Adept Mail Client shall send user emails to the Adept Mail Server via a minimally compliant SMTP protocol. | SW | 1 | Send Email | Client |
| 12 | The Adept Mail Client shall provide a graphical user interface to allow users to generate requests and view their emails. | SW | 2 | Authenticate, view Email, Manage Emails, Manage Folders | Client |
| 13 | The Adept Mail Client shall require local authentication from any user before executing local requests. | SW | 1 | Authenticate | Client |
| 14 | The Adept Mail Client shall provide remote authentication to the Adept Mail Server prior to executing any requests. | SW | 1 | Manage Emails, Manage Folders | Client |
| 15 | The Adept Mail Client shall locally encrypt and decrypt the subject and body of every email it sends and receives, respectively, using symmetric-key block encryption based on a user provided password. | SW | 2 | Request Update | Client |

# A Category Interaction Diagrams

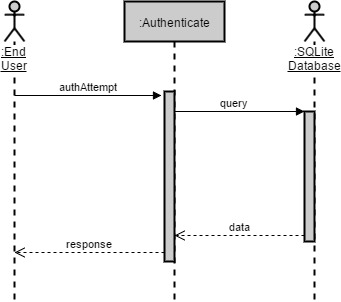
3.1. Use Case Diagram



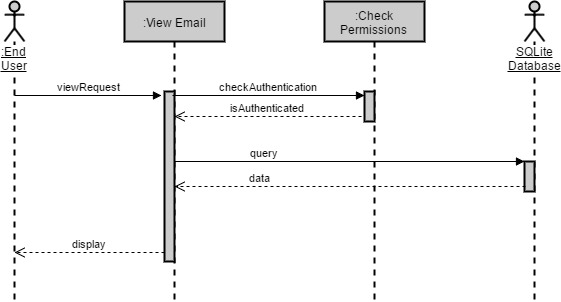
**3.2. Interaction Diagrams**

**3.2.1. ADEPT Client Program Interaction Diagrams**

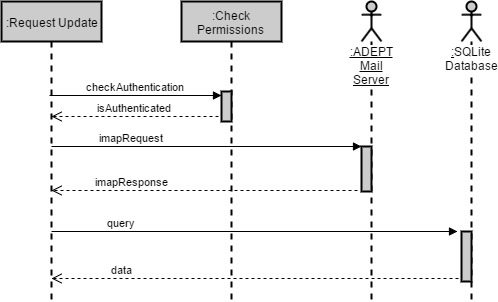
3.2.1.1. Authenticate



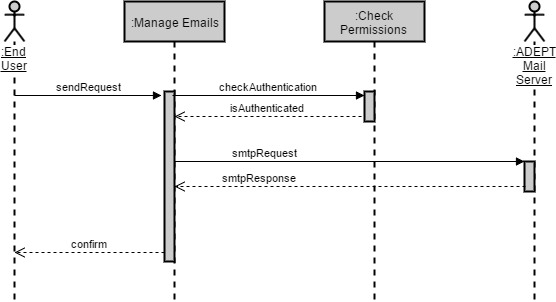
3.2.1.2. View Email



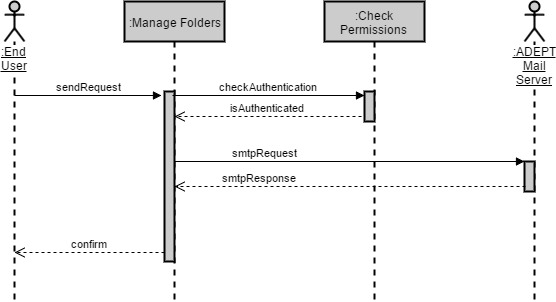
3.2.1.3. Request Update



3.2.1.4. Manage Emails

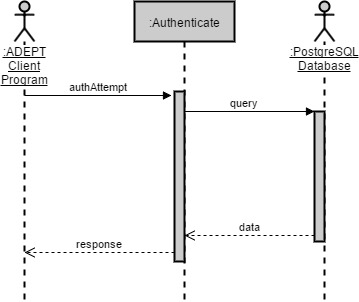


3.2.1.5. Manage Folders

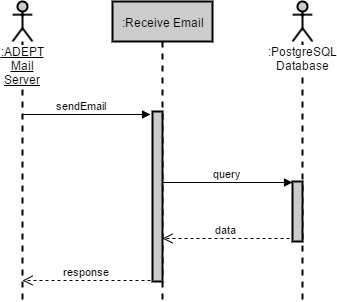


3.2.2. ADEPT Mail Server Interaction Diagrams

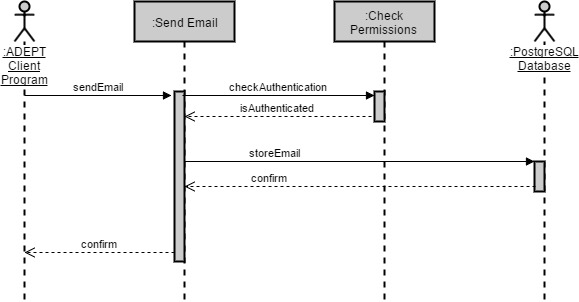
3.2.2.1. Authenticate



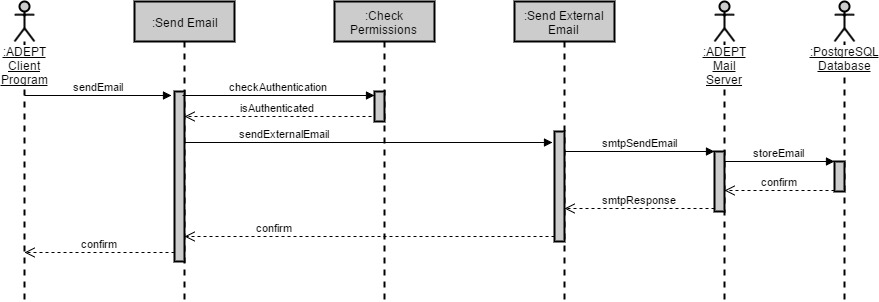
3.2.2.2. Receive Email



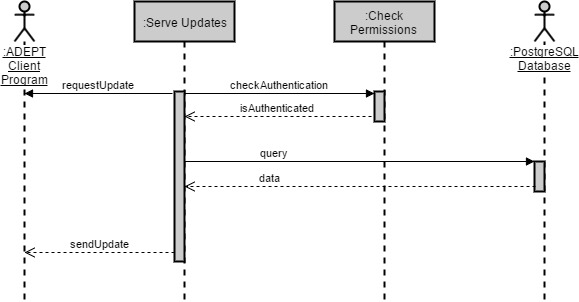
3.2.2.3. Send Email



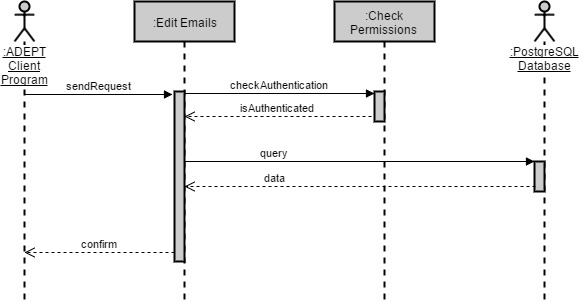
3.2.2.4. Send External Email



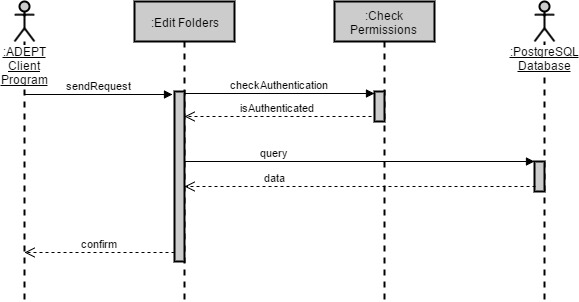
3.2.2.5. Serve Updates



3.2.2.6. Edit Emails

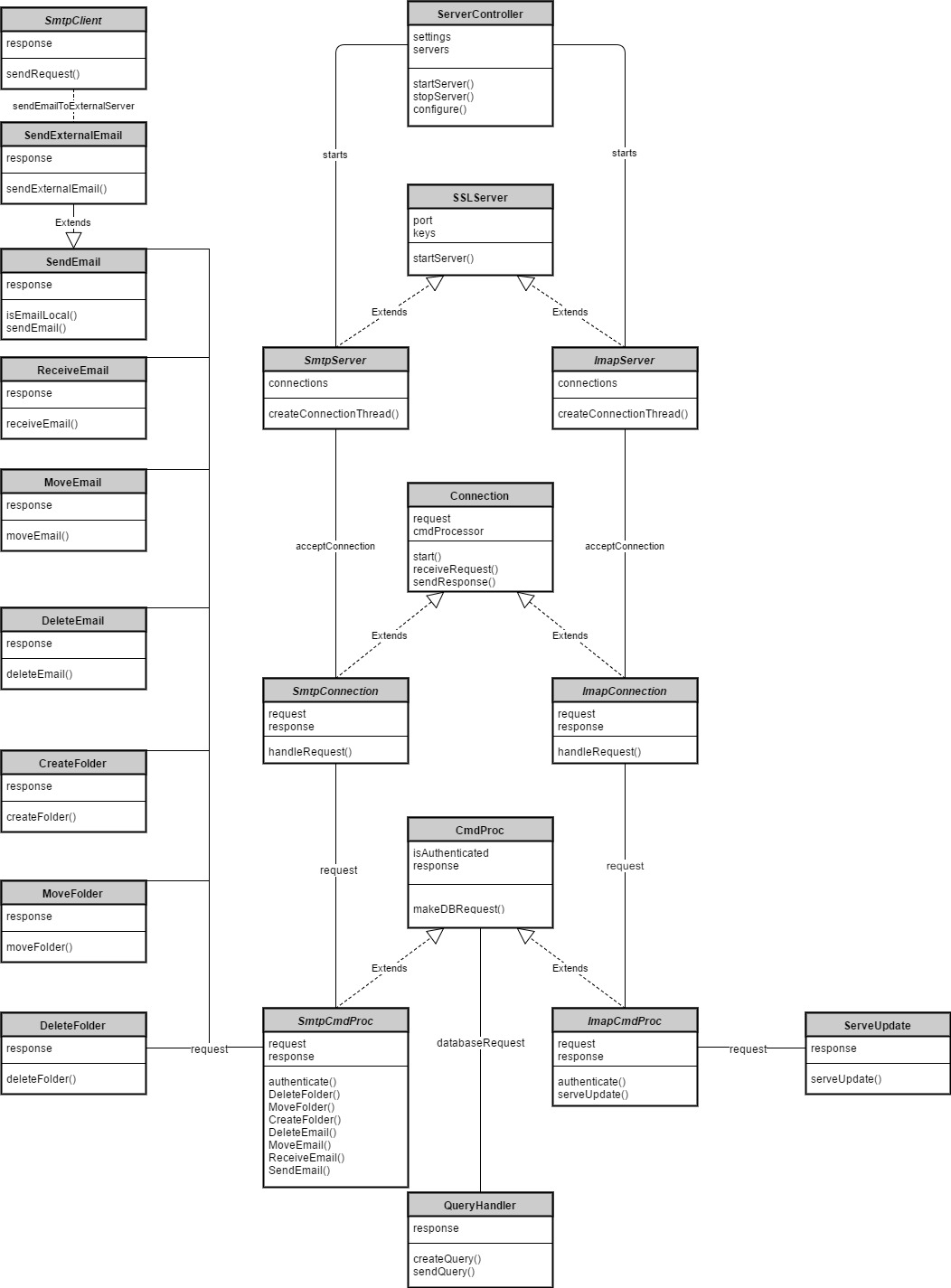


3.2.2.7. Edit Folders

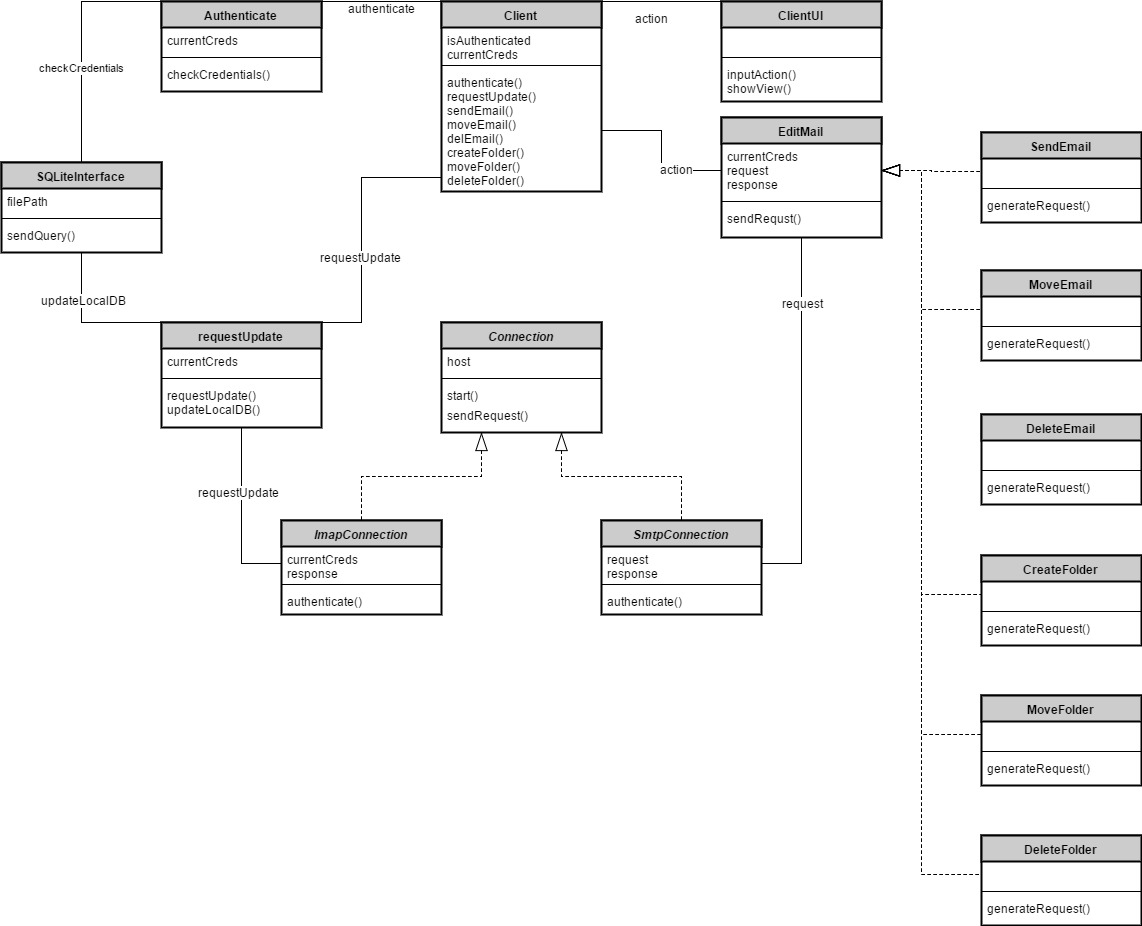


# Class interface

4.1 Server Class Diagram



4.2 Client Class Diagram



# WSD

|  |  |
| --- | --- |
| **Name** | **Role** |
| **Amani Konduru** | Project manager and tester, Document handler, Java Programmer, Database Structure (PostgresSQL schemas) |
| **Benjamin Garber (Daniel)** | Code the GUI that utilizes Ed's code base and document the code for JavaDocs |
| **Edward Bull** | Code the Server and Client prototypes into classes that can will be used in the GUI program and document with JavaDocs |
| **Paul David Utesch** | Create testing scenarios for testing the GUI functionality and J-Unit |

# Gantt Chart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Create V1 Stories | S-01001 | Ed Bull,akonduru | Done | 1 |
| Document 2 - Requirements Elicitation | S-01017 | akonduru | Done | 1 |
| Learn how to use VersionOne | S-01022 | Ed Bull,bgarber,akonduru | Done | 1 |
| Set up group Github | S-01018 | Ed Bull | Done | 1 |
| Plan Server Database And Write CreateTables SQL Script | S-01019 | Putesch,akonduru | Future | 1 |
| Set up PostgresSQL test server | S-01009 | akonduru | In Progress | 1 |
| Implement Test Database | S-01020 | akonduru | Future | 1 |
| Server ServerController | S-01002 | Ed Bull | In Progress | 1 |
| Server SmtpServer | S-01003 | Ed Bull | In Progress | 1 |
| Server ImapServer | S-01004 | Ed Bull | In Progress | 1 |
| Server SmtpConnection | S-01005 | Ed Bull | In Progress | 1 |
| Server ImapConnection | S-01006 | Ed Bull | In Progress | 1 |
| Server CmdProcessor | S-01007 | bgarber | In Progress | 1 |
| Server QueryGenerator | S-01008 |  | Future | 1 |
| Server SmtpClient | S-01021 |  | Future | 1 |
| Document 2 Title Page | S-01037 | akonduru | Done | 1 |
| Document 2 Problem Statements | S-01038 | Ed Bull | Done | 1 |
| Document 2 RTM | S-01039 | Ed Bull | Done | 1 |
| Document 2 WSD | S-01040 | akonduru | Done | 1 |
| Document 2 Gantt | S-01041 | akonduru | Done | 1 |
| Document 2 Dictionary | S-01042 | Ed Bull,bgarber | Done | 1 |
| Set up PostgresSQL test server | S-01043 | akonduru | In Progress | 2 |
| Implement Test Database | S-01044 | akonduru | Future | 2 |
| Server ServerController | S-01045 | Ed Bull | In Progress | 2 |
| Server SmtpServer | S-01046 | Ed Bull | In Progress | 2 |
| Server ImapServer | S-01047 | Ed Bull | In Progress | 2 |
| Server SmtpConnection | S-01048 | Ed Bull | In Progress | 2 |
| Server ImapConnection | S-01049 | Ed Bull | In Progress | 2 |
| Server CmdProcessor | S-01050 | bgarber | In Progress | 2 |
| Server QueryGenerator | S-01051 |  | Future | 2 |
| Server SmtpClient | S-01052 |  | Future | 2 |
| Collate Document 3 | S-01024 | akonduru | In Progress | 2 |
| Document 3 Title Page | S-01025 | akonduru | Accepted | 2 |
| Document 3 RTM (5 columns) | S-01027 | Ed Bull | Accepted | 2 |
| Document 3 Use Cases and Int. Diagrams | S-01028 | Ed Bull | Done | 2 |
| Document 3 Function Point Analysis | S-01029 | bgarber | Done | 2 |
| Document 3 Database To Be Used | S-01030 | akonduru | Future | 2 |
| Document 3 Updated WSD | S-01031 | akonduru | Done | 2 |
| Document 3 Updated Gantt Chart | S-01032 | Ed Bull | In Progress | 2 |
| Document 3 Dictionary | S-01033 | Ed Bull | Accepted | 2 |
| Document 3 Use Cases Rationale | S-01034 | Ed Bull | Done | 2 |
| Document 3 horizontal prototype | S-01035 | Putesch | In Progress | 2 |
| Doc4 #2 | S-01063 | akonduru | In Progress | 2 |
| Doc4 #4 | S-01064 | Putesch | Done | 2 |
| Doc4 #5 | S-01065 | Ed Bull | In Progress | 2 |
| Doc4 #9 | S-01066 | EdBull,Putesch,akonduru | In Progress | 2 |

# Glossary

1. **Encryption:** the process of converting data into a code, to prevent unauthorized access. Encryption is the process of transforming data into an unreadable, encrypted form. The transformation is done using one of several cryptographic algorithms that leverage computationally difficult mathematical problems to make reversing the transformation difficult if not effectively impossible
2. **Symmetric Encryption**: Symmetric Encryption uses a key or set of keys to both encrypt and decrypt data. If data is to be shared between two parties, they must both have the key or keys to decrypt or encrypt the data.
3. **Asymmetric Encryption:** Asymmetric Encryption, also known as Public Key Encryption, is a type of encryption where anyone in possession of a public key can encrypt a message. That message can then only be decrypted with a private key. This method is often used for identity authentication because it is computationally expensive. Once authentication is completed, communications will then often transition into symmetric encryption after generating a symmetric encryption key.
4. **End to End Encryptio**n: Only the communicating users can read the messages.
5. **SSL/TLS (Secure Sockets Layer / Transport Layer Protocol):** TLS and the now- deprecated SSL it is based on are network security protocols meant to secure client-server connections using both symmetric encryption for data transfer and asymmetric encryption for identity authentication. While there are many options that can be set in an SSL/TLS session, the foundation of the protocols lie in using encryption to authenticate the identities of the connected parties and to secure the privacy of the data transferred between them.
6. **Server:** a server program awaits and fulfills requests from client programs, which may be running in the same or different computers.
7. **Client:** requesting program or user.
8. **Socket:** Is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.
9. **SMTP protocol:** Simple Mail Transfer Protocol. It is an Internet standard for electronic mail (email) transmission. SMTP was first defined by RFC 821 and updated in RFC 5321.
10. **IMAP protocol:** Internet Message Access Protocol. Itis an Internet standard protocol used by e-mail clients to retrieve e-mail messages from a mail server over a TCP/IP connection. IMAP is defined by RFC 3501.
11. **TCP/IP :** IP (Internet Protocol) is the basic communication language or protocol of the ozInternet. It can also be used as a communications protocol in a private network (either an intranet or an extranet). TCP (Transmission Control Protocol) is layered on top of IP to provide certain network control and data validation features for many internet communications.

# Rationale

**Software Architecture Rationale**

**Database**: We will use two databases PostgreSQL and SQLite. The metadata will be used to store the encrypted message in a simple PostgreSQL database, with the owner as a primary key. The client program will have some basic login functionality and a SQLite database will be used to store usernames and a hashed/salted password. Both are open source and are best for our project from the limited options that are available. PostgreSQL is good for concurrency and SQLite is easy to function with. We are in the process of setting up the database. SQLite and PostgreSQL

**Eclipse:** easy fictional usage and everyone has experience with it.

**Linux – Ubuntu:** Allows terminal usage.

**Cloud based virtual machine-Digital Ocean droplet:** Don’t have to worry about storage it’s on the cloud and easy to access at any given point.

**Class Diagram Rationale**

**Server Rationale**

1. Server Controller: The ServerController object configures the server on startup and spawns the SmtpServer and ImapServer objects.
2. SSLServer: The SSLServer abstract class is the foundation for the SmtpServer and ImapServer objects. An SSLServer listens for traffic over a secured SSL socket.
3. SmtpServer: The SmtpServer object listens for and accepts SmtpConnection objects.
4. ImapServer: The ImapServer object listens for and accepts ImapConnection objects.
5. Connection: The Connection abstract class is the foundation for the SmtpConnection and ImapConnection objects. A Connection represents an incoming request over the listening socket, parsed into a byte array.
6. SmtpConnection: The SmtpConnection objects input requests into the system and pass those requests to SmtpCmdProc objects.
7. ImapConnection: The ImapConnection objects input requests into the system and pass those requests to the ImapCmdProc objects.
8. CmdProc: The CmdProc abstract class forms the foundation of the SmtpCmdProc and ImapCmdProc objects. It will process requests and then execute the appropriate action based on the format of the request.
9. SmtpCmdProc: The SmtpCmdProc objects parse requests according to the SMTP protocol.
10. ImapCmdProc: The ImapCmdProc objects parse requests according to the IMAP protocol.
11. QueryHandler: The QueryHandler class constructs queries and receives responses from the PostgreSQL database attached to the server.
12. ServeUpdate: The ServerUpdate class executes an IMAP request for an update.
13. DeleteFolder, MoveFolder, CreateFolder, DeleteEmail, MoveEmail, ReceiveEmail, and SendEmail classes all execute their respective action by calling the appropriate method in QueryHandler.
14. SendExternalEmail: The SendExternalEmail class extends the SendEmail class in the situation that an email destination address is not local. This class sends an email to the destination server by invoking the SmtpClient class and passing the request on.
15. SmtpClient: The SmtpClient object connects to an external socket and sends an email in an SMTP formatted request.

**Client Rationale**

1. Client: The Client class starts the program and receives input from ClientUI and translates that input into actions based on its various connected classes.
2. Authenticate: The Authenticate class receives credentials from Client and checks those credentials against a query to the SQLiteInterface class.
3. SQLiteInterface: The SQLiteInterface class processes and sends queries to the local SQLite database and responds with data as its result
4. RequestUpdate: The RequestUpdate class is periodically called by the Client class. It requests updates from a spawned ImapConnection object and then updates the local database with that data by invoking the SQLiteInterface class.
5. Connection: The Connection abstract class is the foundation for the ImapConnection and SmtpConnection objects. It opens a connection to an external listening server over the relevant protocol.
6. ImapConnection: Used to connect to the server to request updates for the RequestUpdate class.
7. SmtpConnection: Used to connect to the server to make requests for the EditMail class.
8. EditMail: The EditMail class takes actions from the Client class and calls the appropriate class to execute those actions, then responds with a status to be displayed.
9. SendEmail, MoveEmail, DeleteEmail, CreateFolder, MoveFolder, DeleteFolder classes all execute their respective actions by forming and then sending a request to the SmtpConnection class. The response is then sent back to Client.