Software Engineering- CSC 4350 Spring 2017

An encryption and decryption system for message communication

ADEPT

Amani Konduru

Benjamin Garber (Daniel)

Edward Bull

Paul David Utesch

Team

4/18/2017

Table of Contents

I. Introduction

Topic: A secure mail server and client pair

The Adept Mail system will be composed of two parts, a server and client. The first part, the Adept Mail Server, will listen on specified ports for IMAP and SMTP communication. It will be able to receive and store emails between its list of authenticated users. All network communication will be secured via SSL/TLS, and client requests will be authenticated via an IMAP authentication exchange. The Adept Mail Server will support multiple concurrent connections and will use a PostgreSQL database for storage. The Adept Mail Server will be decoupled from the database, so that multiple Adept Mail Servers could communicate with the same database or database system

The second part, the Adept Mail Client, will interact with a user via either a CLI or GUI interface. The client can authenticate, update local storage of emails, and manage their email account on the Adept Mail Server where the canonical storage of their emails will take place. The Adept Mail Client will communicate over SSL/TLS for security, and all requests will be made in properly formed IMAP or SMTP exchanges as appropriate. Local storage will be encrypted and only decrypted upon viewing. Unencrypted emails will not be stored in anything but RAM during the process.

While interoperability with other mail servers may not be feasible as a student project in a single semester, by adhering to the IMAP and SMTP protocol definitions in a minimally compliant fashion we can demonstrate how interoperability is accomplished in the real world. Additionally, while a truly secure program may also not be feasible (often for professional teams as well), we can demonstrate the fundamentals about how network connections, passwords, local data storage, and SQL queries can be secured.

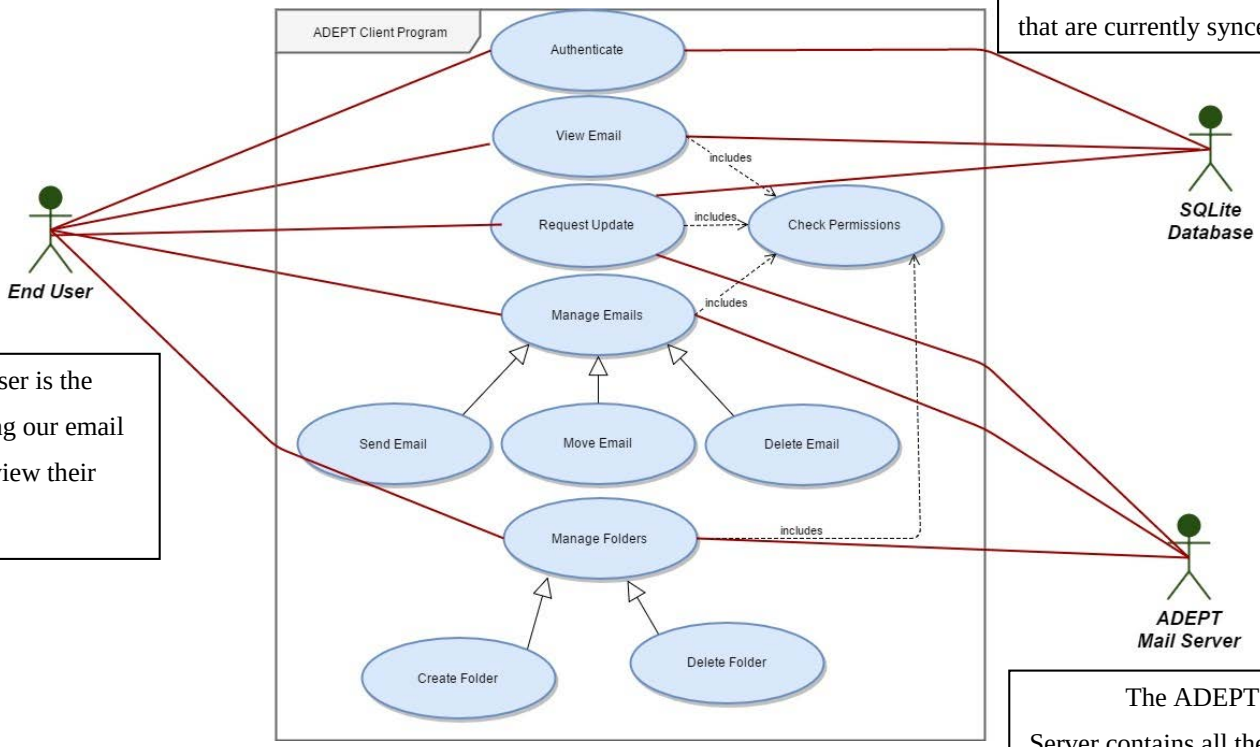
II. Requirements Traceability Matrix (RTM)

Entry	Specification	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 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 to other Adept Mail Servers upon an authenticated request from that user.	SW	2	Send 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 the user.	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

III. Use Case, Sequence and Interaction diagrams

Use Case Diagrams

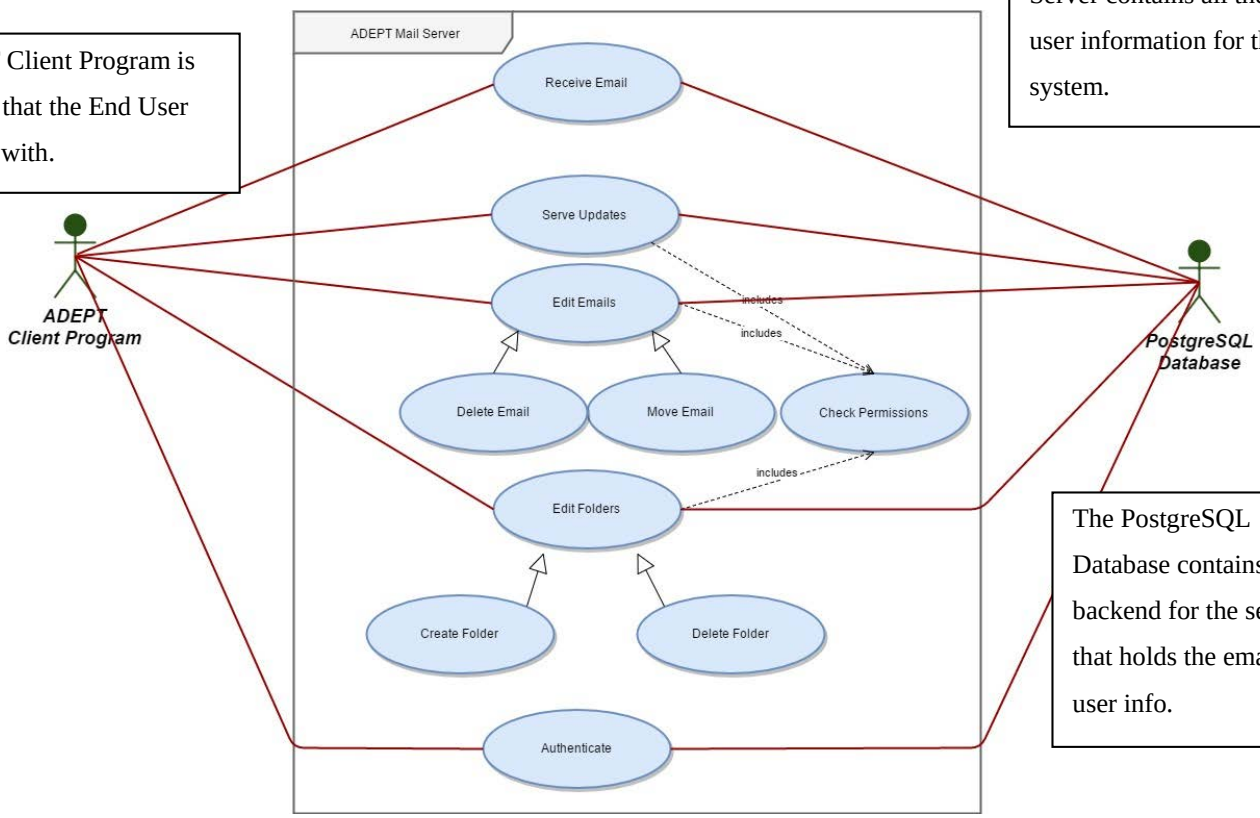
The SQLite database contains the user information and all their emails that are currently synced



The End User is the person using our email system to view their emails.

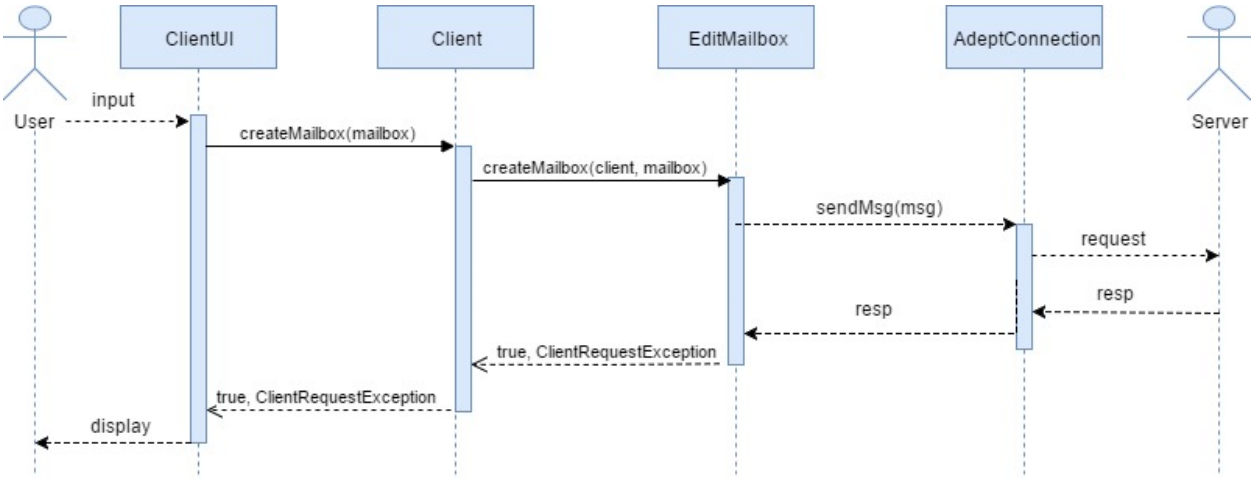
The ADEPT Mail Server contains all the emails and user information for the entire system.

The ADEPT Client Program is the frontend that the End User will interact with.

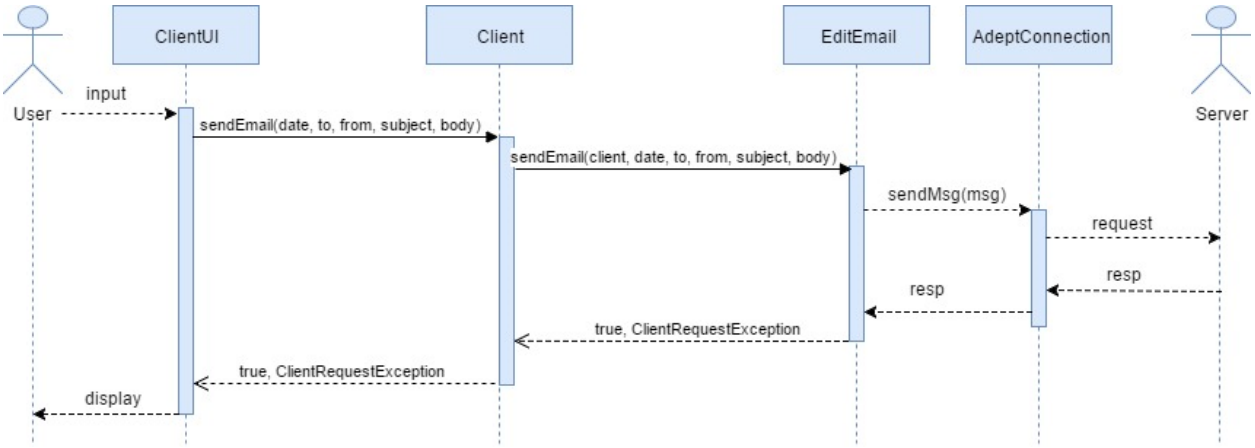


The PostgreSQL Database contains is the backend for the server that holds the emails and user info.

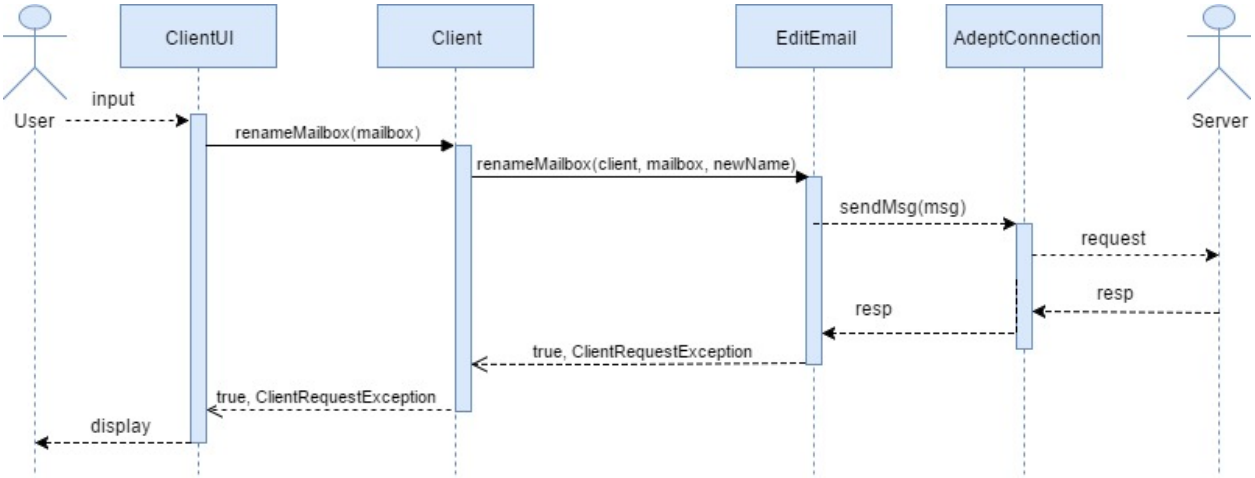
Sequence Diagrams



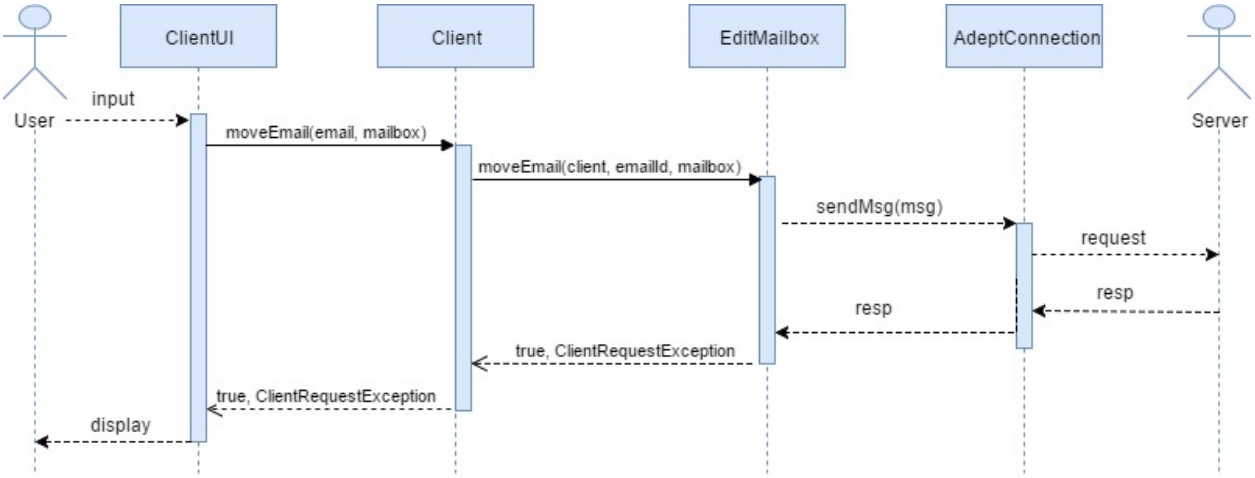
Here, the user wants to create a new mailbox. Using the UI to select the appropiate action, the user will be prompted for a mailbox name. After pressing enter, a new mailbox will be created will the provided string as the name.



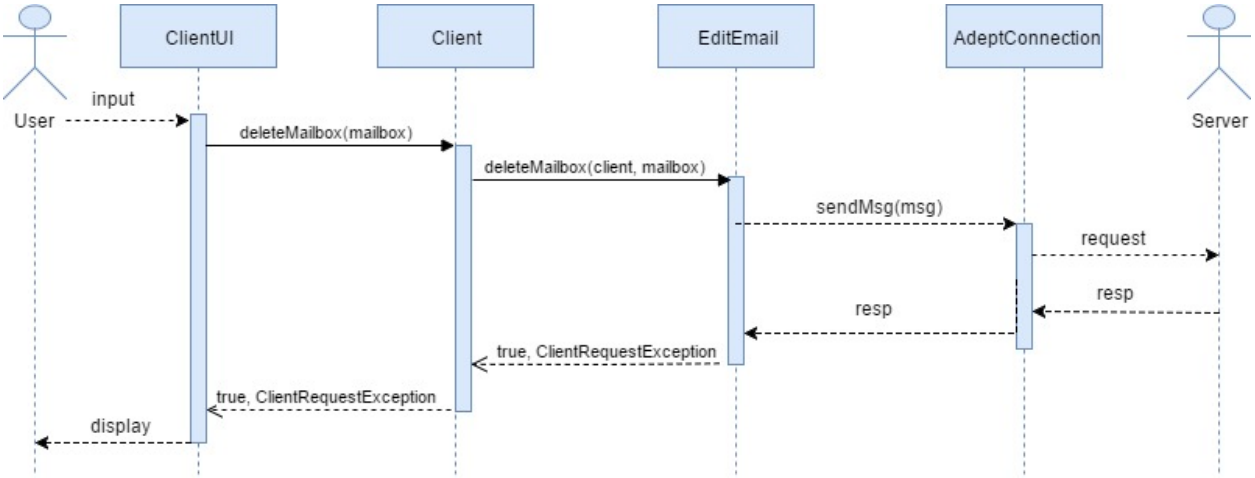
Here, the user wants to send an email. Using the UI to select the appropriate action, the user will be prompted for an email address to send to, a subject and a body. Upon pressing send, an email will be sent to that user's mailbox if they are in the server.



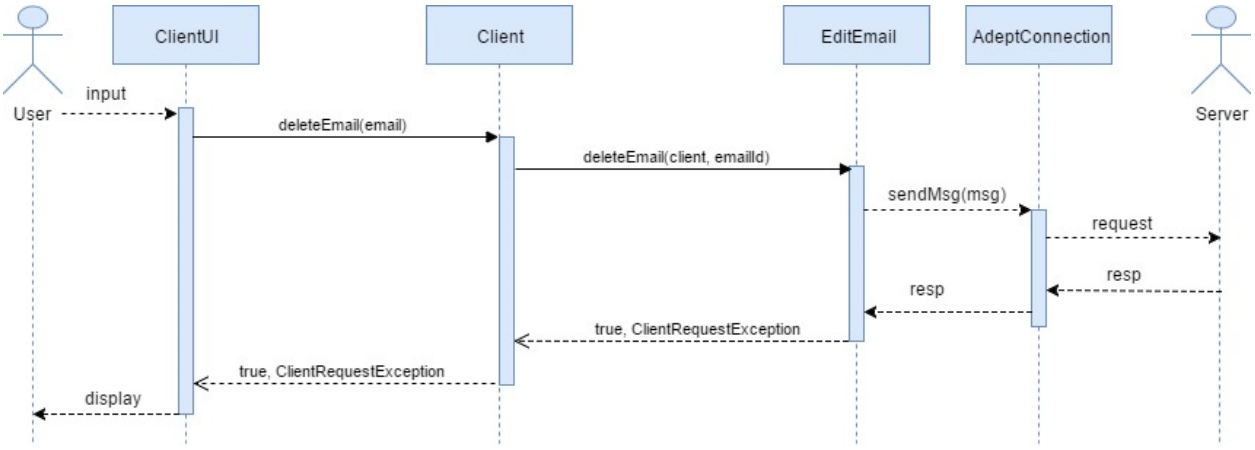
Here, the user wants to rename an existing mailbox. First the user must select the mailbox they wish to rename. The user will then be prompted to enter the new name for the mailbox. Upon pressing enter, the selected mailbox will be renamed to the string provided by the user. All emails will be moved to the new mailbox with it.



Here, the user wants to change the mailbox that an email resides in. First, the user must select the email they would like to move. Then, the user must select the appropriate UI element. They will be prompted for the name of the existing mailbox they would like to move the email to. Upon pressing enter, the email will be moved to the provided mailbox if the mailbox exists. If it doesn't, the operation will fail.

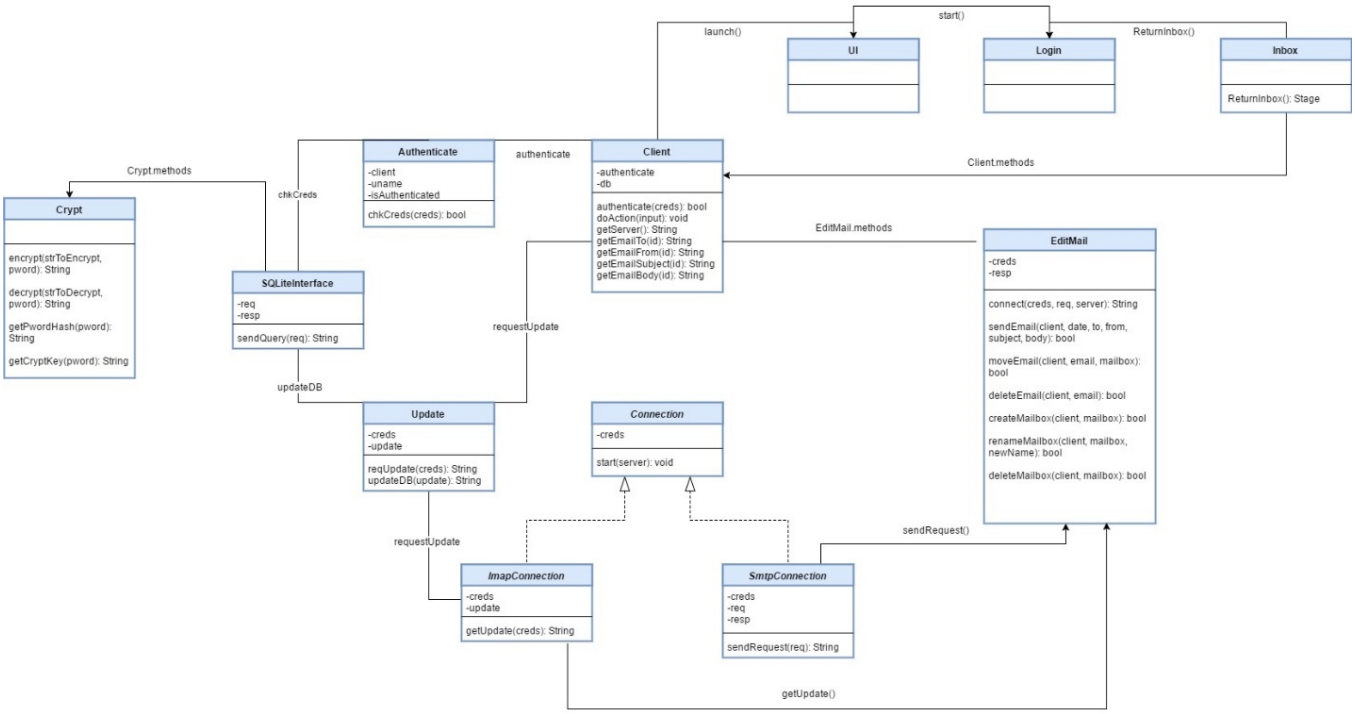


Here, the user wants to delete a mailbox. First, the user must select the mailbox they want to delete. Then, the user will select the appropriate UI element. The user will be prompted to confirm that they want to delete the mailbox. After pressing enter, the mailbox will be deleted.



Here, the user wants to delete an email. First, the user must select the email that they want to delete. Then, they will select the appropriate UI element. The user will be prompted for a confirmation. Upon confirming, the selected email will be deleted.

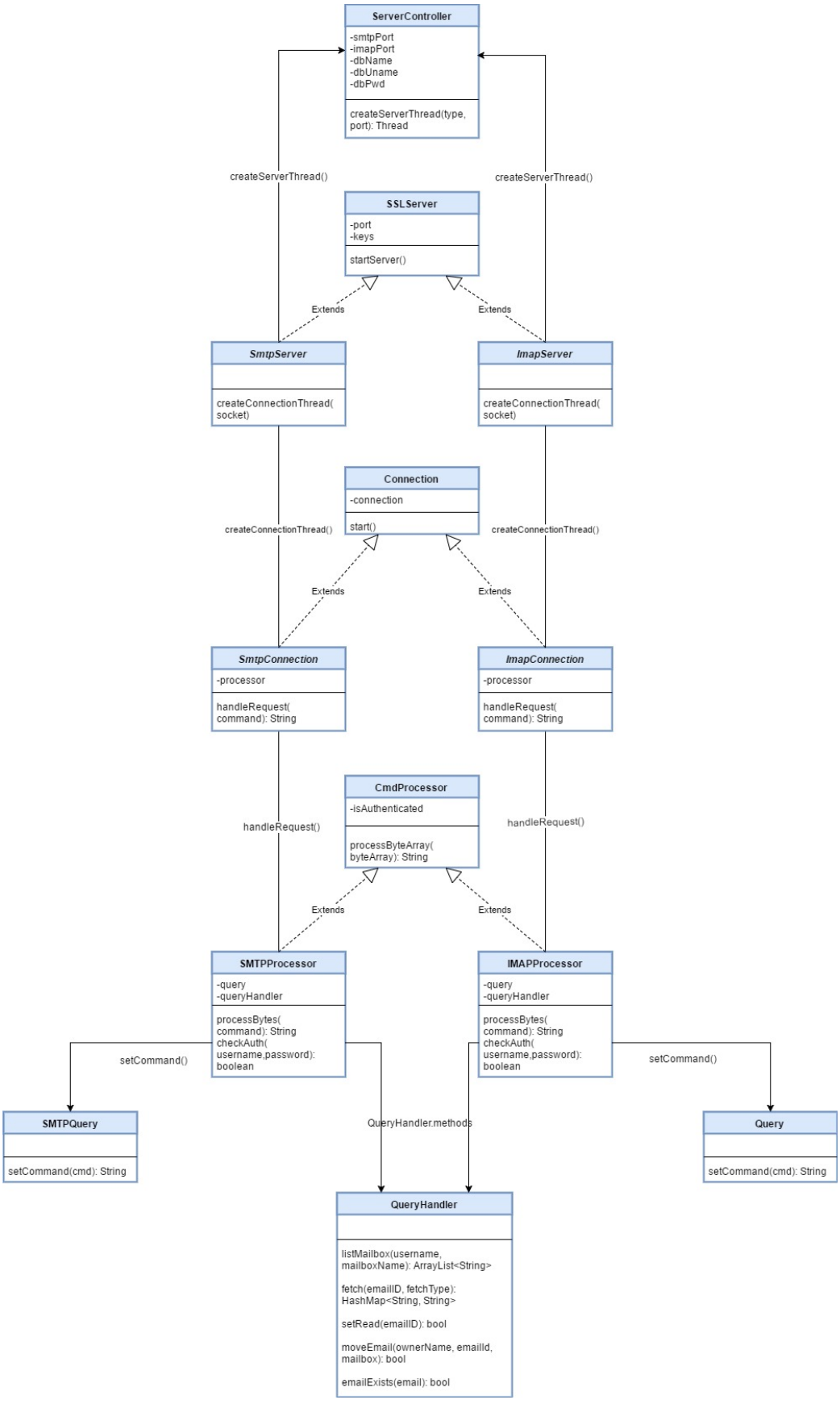
IV. Object Design



The Adept Mail Client class diagram shows the relationships between Adept Mail Client objects. The foundation of the system is the Client class. The UI class takes input from the user and calls the appropriate method from the Client Class. From there, the Client class can call Authenticate, Update, or EditEmail methods as appropriate.

Any local changes to the Adept Mail Client storage goes through the SQLiteInterface class. Sensitive fields are first encrypted or hashed as appropriate via the Crypt class and then stored. Any remote requests (authentication and commands) go through either the ImapConnection class or the SmtpConnection class.

Below, the Adept Server class diagram shows the relationships between Adept Mail Server objects. The ServerController class is the starting point, and spawns an ImapServer and SmtpServer as separate threads. The Imap/SmtpServer classes then listen on their appropriate ports for incoming connections. When a connection is received, they spawn off a Connection object in a separate thread and then resume listening. The Imap/SmtpConnection objects create an Imap/SmtpProcessor object and then pass the Imap/SmtpProcessor any messages they receive over their active socket. Those messages are identified as commands by Query, parsed by Imap/SmtpProcessor, and then the appropriate method is called in QueryHandler. QueryHandler then constructs and executes a SQL query with the default PostgreSQL server as a target.


ServerController
-smtpPort
-imapPort
-dbName
-dbUname
-dbPwd
createServerThread(type, port): Thread
createServerThread()
createServerThread()
SSLServer
-port
-keys
startServer()
Extends
Extends
SmtpServer
createConnectionThread(socket)
ImapServer
createConnectionThread(socket)
Connection
-connection
start()
createConnectionThread()
createConnectionThread()
Extends
Extends
SmtpConnection
-processor
handleRequest(command): String
ImapConnection
-processor
handleRequest(command): String
CmdProcessor
-isAuthenticated
processByteArray(byteArray): String
handleRequest()
handleRequest()
Extends
Extends
SMTPProcessor
-query
-queryHandler
processBytes(command): String
checkAuth(username,password): boolean
IMAPProcessor
-query
-queryHandler
processBytes(command): String
checkAuth(username,password): boolean
setCommand()
setCommand()
SMTPQuery
setCommand(cmd): String
QueryHandler.methods
Query
setCommand(cmd): String
QueryHandler
listMailbox(username, mailboxName): ArrayList<String>
fetch(emailID, fetchType): HashMap<String, String>
setRead(emailID): bool
moveEmail(ownerName, emailId, mailbox): bool
emailExists(email): bool

V. Test Cases

Note: We have omitted the SendExternalEmailFunctionality and RenameMailboxFunctionality test cases due to time and development constraints.

Test-case Identifier	ServerConnectivity
Feature	RTM: 7, 8, 14
Feature Pass/Fail Criteria	The test passes if each request receives a properly formed protocol compliant response.
Means of Control	IMAP, SMTP direct connections
Data	A series of protocol specific commands (both well-formed and mal-formed) and their expected protocol-specific responses.
Test Procedure	Using a third party tool, the tester will create a series of concurrent connections to the server on both its IMAP and SMTP interfaces.
Special Requirements	An authenticated account on the test server.

Target: Our Digital Ocean VM at 138.197.104.156 running the latest version of server.jar to demonstrate internet connectivity.

Test 1: IMAP

Test 2: SMTP

As is visible from the openssl s_client information, SSL authentication was successful (if naïve) and the LOGIN command received the expected output on both ports. Note that we are using IMAP authentication on the SMTP port. This is an intentional time-saving implementation.

Test-case Identifier	SendFunctionality
Feature	RTM: 1, 5, 6, 7, 8, 11, 13, 14
Feature Pass/Fail Criteria	The test passes if emails and their associated mailboxes are sent from the client, received by the server, and stored in the database.
Means of Control	Client UI or Client CLI
Data	Placeholder content and names will be used for both the email and mailboxes.
Test Procedure	Using an authenticated account, the tester will use the client UI to send an e-mail to another authenticated account on the same server.
Special Requirements	Two authenticated accounts on the test server.

Pre-test condition: The email ebull@adept.com has no current emails before the test.

```
coredb=# select email_id, email, subject, mailbox from emails inner join users on emails.owner=users.user_id where email='ebull@adept.
com';
 email_id | email | subject | mailbox
----------+-------+---------+---------
(0 rows)
```

Then the test is run and verified in the Adept Mail Client CLI. The second email account ed@adept.com is used to send the email to ebull@adept.com.

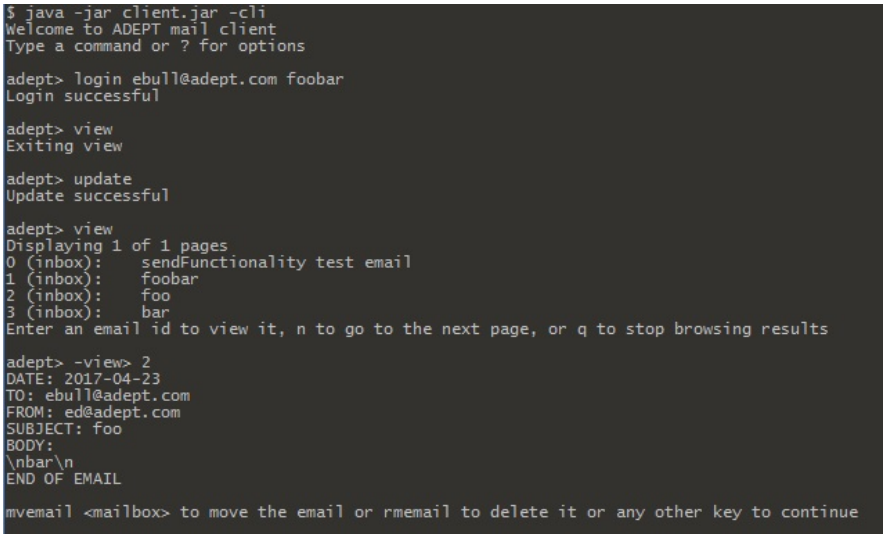
```
$ java -jar client.jar -cli
Welcome to ADEPT mail client
Type a command or ? for options

adept> login ebull@adept.com foobar
Login successful

adept> view
Exiting view

adept> logout
Logged out

adept> login ed@adept.com adept
Login successful

adept> send
Enter recipient (if multiple recipients, separate by commas)

adept> -send> ebull@adept.com
Enter subject

adept> -send> sendFunctionality test email
Enter body, terminated with a "." on a single line
The test passes if emails and their associated mailboxes are:
send from the client.
received by the server.
stored in the database.
.
Email sent

adept> logout
Logged out

adept> login ebull@adept.com foobar
Login successful

adept> view
Exiting view

adept> update
Update successful

adept> view
Displaying 1 of 1 pages
0 (inbox):    sendFunctionality test email
Enter an email id to view it, n to go to the next page, or q to stop browsing results

adept> -view> 0
DATE: 2017-04-23
TO: ebull@adept.com
FROM: ed@adept.com
SUBJECT: sendFunctionality test email
BODY:
The test passes if emails and their associated mailboxes are:\nsend from the client.\nreceived by the server.\nstored in the databas
e.\n
END OF EMAIL

mvemail <mailbox> to move the email or rmemail to delete it or any other key to continue
```

Post-test condition: The email ebull@adept.com has received the test email.

```
coredb=# select email_id, email, subject, mailbox from emails inner join users on emails.owner=users.user_id where email='ebull@adept.
com';
 email_id |      email      |           subject           | mailbox
----------+-----------------+-----------------------------+---------
       43 | ebull@adept.com | sendFunctionality test email |      17
(1 row)
```

Test-case Identifier	UpdateFunctionality
Feature	RTM: 4, 7, 8, 9, 10, 12, 13, 14, 15
Feature Pass/Fail Criteria	The test passes if the server receives the update request from the client, sends back e-mail data, and the client updates its local database with that data.
Means of Control	Client UI or Client CLI
Data	Placeholder content and names will be used for both the email and mailboxes.
Test Procedure	Using an authenticated account, the tester will use the client UI to request a mailbox update from the server. Once the update is complete, the tester will verify that all data has transferred correctly.
Special Requirements	An authenticated account on the test server. No emails stored on local storage.

Pre-test condition: By deleting the Adept Mail Client's local sqlite database, we can ensure a valid testing pre-condition. The emails do not show in the ebull@adept.com inbox.



Post-test condition: After running the update, the emails are visible in the ebull@adept.com inbox.

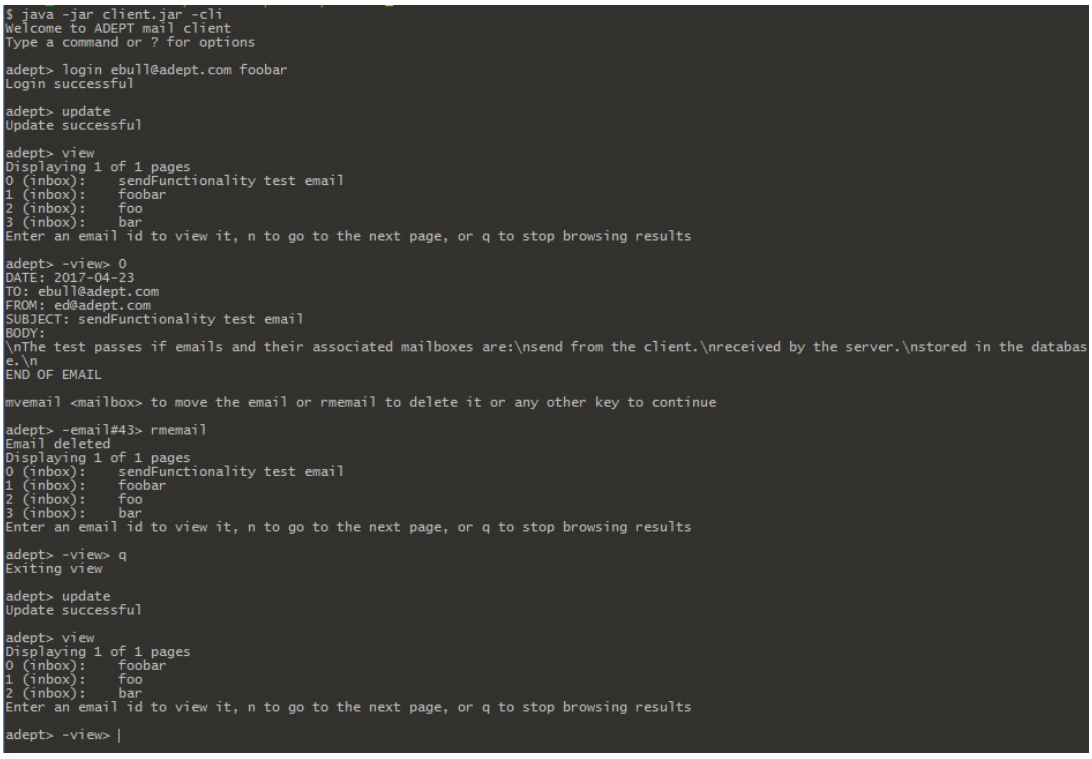
Test-case Identifier	MoveEmailFunctionality
Feature	RTM: 1, 2, 4, 8, 9, 12, 13, 14
Feature Pass/Fail Criteria	The test passes if the e-mail is moved from one mailbox to another mailbox, and then deleted.
Means of Control	Client UI or Client CLI
Data	Placeholder content and names will be used for both the email and mailboxes.
Test Procedure	Using an authenticated account, the tester will use the client UI to move an email from one mailbox to another mailbox. The tester will confirm the move.
Special Requirements	An authenticated account on the test server.

Pre-test condition: A valid email for ebull@adept.com in the "inbox" mailbox.

Post-test condition: The email is now in the "school" mailbox.

Test-case Identifier	DeleteEmailFunctionality
Feature	RTM: 1, 3, 4, 8, 9, 12, 13
Feature Pass/Fail Criteria	The test passes if the e-mail is moved from one mailbox to another mailbox, and then deleted.
Means of Control	Client UI or Client CLI
Data	Placeholder content and names will be used for both the email and mailboxes.
Test Procedure	Using an authenticated account, the tester will use the client UI to delete an e-mail. The tester will confirm the deletion.
Special Requirements	An authenticated account on the test server.

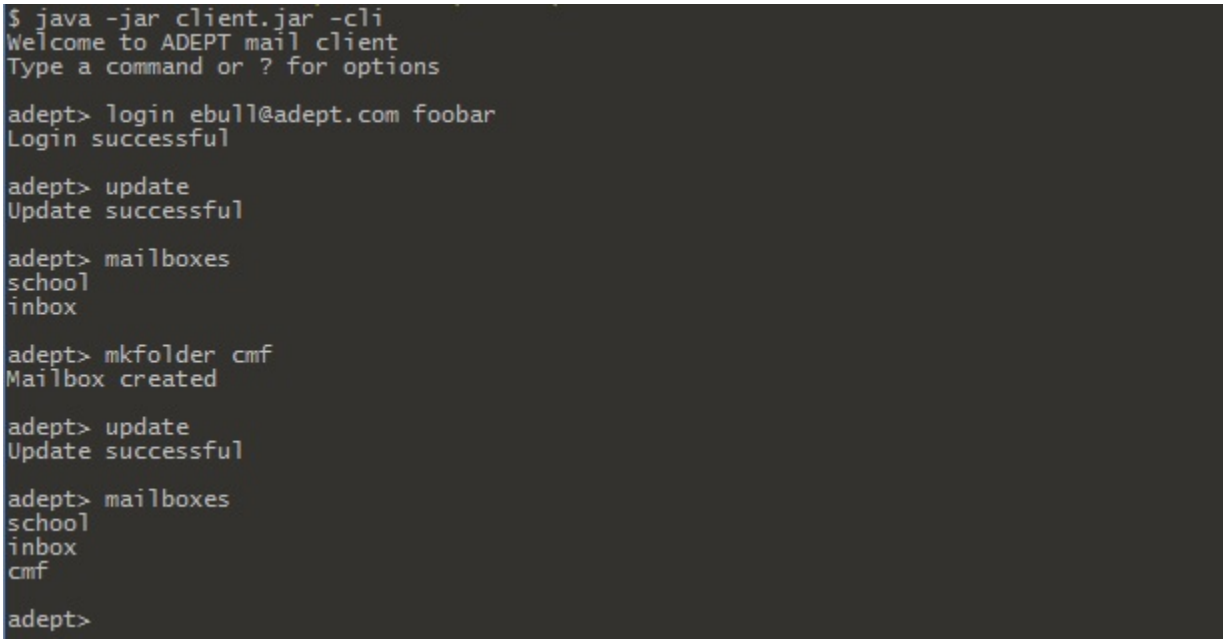
Pre-test condition: An email is in the ebull@adept.com mailbox.



Post-test condition: The email is no longer in the ebull@adept.com mailbox.

Test-case Identifier	CreateMailboxFunctionality
Feature	RTM: 1, 4, 8, 9, 12, 13
Feature Pass/Fail Criteria	The test passes if a new mailbox is created.
Means of Control	Client UI or Client CLI
Data	Placeholder content and names will be used for both the email and mailboxes.
Test Procedure	Using an authenticated account, the tester will use the client UI to create a new mailbox. The tester will confirm that the new mailbox is present.
Special Requirements	An authenticated account on the test server.

Pre-test condition: The account ebull@adept.com does not have the mailbox 'cmf'.



Post-test condition: The account ebull@adept.com has the mailbox 'cmf'.

Test-case Identifier	DeleteMailboxFunctionality
Feature	RTM: 1, 4, 8, 9, 12 13
Feature Pass/Fail Criteria	The test passes if the e-mail is moved from one mailbox to another mailbox, and then deleted.
Means of Control	Client UI or Client CLI
Data	Placeholder content and names will be used for both the email and mailboxes.
Test Procedure	Using an authenticated account, the tester will use the client UI to move an email from one mailbox to another mailbox. The tester will confirm the move, then delete the e-mail. The tester will confirm the deletion.
Special Requirements	An authenticated account on the test server.

Pre-test condition: The account ebull@adept.com has the mailbox 'cmf'.

```
$ java -jar client.jar -cli
Welcome to ADEPT mail client
Type a command or ? for options

adept> login ebull@adept.com foobar
Login successful

adept> update
Update successful

adept> mailboxes
school
inbox
cmf

adept> rmfolder cmf
Mailbox deleted

adept> update
Update successful

adept> mailboxes
school
inbox

adept>
```

Post-test condition: The account ebull@adept.com no longer has the mailbox 'cmf'.

Our test cases cover the following requirements.

- ServerConnectivity: 7, 8, 14
- SendFunctionality: 1, 5, 6, 7, 8, 11, 13, 14
- UpdateFunctionality: 4, 7, 8, 9, 10, 12, 13, 14, 15
- MoveEmailFunctionality: 1, 2, 4, 8, 9, 12, 13, 14
- DeleteEmailFunctionality: 1, 3, 4, 8, 9, 12, 13
- CreateMailboxFunctionality: 1, 4, 8, 9, 12, 13
- DeleteMailboxFunctionalilty: 1, 4, 8, 9, 12 13

The set of these requirements is the set of the RTM {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}. If these tests complete, we can say with confidence that we have tested the use cases in our system.

In a real world scenario, there would need to be further software testing, including unit testing and regression testing. Because a large numbers of errors can be reported between the Server and Client via the SMTP and IMAP protocols, significant handling would need to be incorporated in that respect.

In addition on the software side, the current implementation of the server would need to incorporate real logging functionality and alerts. Security penetration testing would be important to identify and sanitize possible SQL injection vectors.

Finally, implementation of the Adept Mail System would require significant hardware testing. Any Adept Mail Server would need to undergo stress testing to verify that it could withstand the estimated maximum number of user connections. There might need to be security testing on the hardware aspect as well (ex. using an Adaptive Security Appliance or firewall to defend against possible denial of service attacks).

VI. Rationale

The Adept Mail Server and Adept Mail Client will offer a secured email alternative for users interested in the confidentiality and integrity of their communications. Like competitive mail options, the Adept system will communicate over a secured TLS channel. That means that network sniffers and nodes between the Client and Server will not be able to intercept traffic as plain text. Unlike many other options, the Client will encrypt sensitive fields of its local storage and only decrypt those fields in RAM upon viewing. This ensures that even if the device on which the Client is installed is compromised physically or otherwise, the local storage won't be.

Our system will demonstrate some key points on security, including secure network communications, encryption, password hashing, and SQL sanitization. While our first iteration will be far from fully secure (many professional products are as well!) it will be an excellent learning experience and demonstration of techniques. The framework is in place for later improvements including parameterization of SQL inputs, TLS certificate verification, and more robust password hashing algorithms.

Another important feature of the Adept Mail System is protocol compliance. The Adept Mail Client communicates with the Server using minimally compliant but properly formed IMAP and SMTP commands over appropriate network ports. This means that extending the Adept Mail Client and Server for interoperability with third party mail clients and servers is already close at hand. Imagine sending emails to Gmail and Microsoft accounts via your own homebrewed solution! There are obstacles, of course, and full compliance is a tall order even for professional teams and products that have been established for decades. But the Adept Mail System demonstrates that TLS, IMAP, and SMTP are not black magic—it's achievable even for students to research and comply with standard communications protocols.

The functionality goals of the Adept Mail System are:

- To allow users to send and receive emails with other Adept Mail Clients.
- To allow users to manage their mails—including creating and deleting mailboxes and moving and deleting emails.
- To allow users to securely decrypt and view their stored encrypted emails.
- To allow the creation of local user accounts, each of which specifies its own parent Adept Mail Server, port settings, and secret key.
- To serve concurrent requests to the Adept Mail Server as quickly as possible.

VII. Fictional Point Cost Analysis and COCOMO

Please enter the size in **SLOC**: 5000

Software Development Mode

The software project needs to be described with one of three development modes. These modes range from the familiar to the ambitious, tightly constrained development projects.

- Choose **Organic** for relatively small teams developing software in a highly familiar, in-house environment.
- Choose **Semi-Detached** when the team members have some experience related to some aspects of the system under development but not others and the team is composed of experienced and inexperience people.
- Choose **Embedded** if the project must operate within a strongly coupled complex of hardware, software, regulations, and operational procedures, such as real-time systems.

Choose one: ◉ **Organic** ○ **Semi-Detached** ○ **Embedded**

Software Development Cost Drivers

There are four categories of cost drivers that are found significant performance predictors for a software development project. Each category has several cost drivers. Each driver has a possible rating from Very Low (VL) to Extra High (XH). The ratings are used in the model to adjust the baseline development effort estimation up or down.

For **HELP** on each cost driver, select the driver name.

Product Attributes

○ VL ○ L ○ N ◉ H ○ VH ○ XH : Required Reliability
○ VL ○ L ○ N ○ H ◉ VH ○ XH : Database Size
○ VL ○ L ○ N ◉ H ○ VH ○ XH : Product Complexity

Computer Attributes

○ VL ○ L ○ N ○ H ◉ VH ○ XH : Execution Time Constraint
○ VL ○ L ○ N ○ H ◉ VH ○ XH : Main Storage Constraint
○ VL ○ L ○ N ○ H ◉ VH ○ XH : Virtual Machine Volatility
○ VL ○ L ◉ N ○ H ○ VH ○ XH : Computer Turnaround Time

Personnel Attributes

○ VL ○ L ◉ N ○ H ○ VH ○ XH : Analyst Capability
○ VL ○ L ○ N ◉ H ○ VH ○ XH : Applications Experience
○ VL ○ L ◉ N ○ H ○ VH ○ XH : Programmer Capability
○ VL ○ L ○ N ◉ H ○ VH ○ XH : Virtual Machine Experience
○ VL ○ L ◉ N ○ H ○ VH ○ XH : Programming Language Experience

Project Attributes

○ VL ○ L ◉ N ○ H ○ VH ○ XH : Modern Programming Practices
○ VL ◉ L ○ N ○ H ○ VH ○ XH : Use of Software Tools
○ VL ◉ L ○ N ○ H ○ VH ○ XH : Required Development Schedule

COCOMO II - Constructive Cost Model

Model(s): COCOMO
Monte Carlo Risk: Off
Auto Calculate: Off

Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	5000						
Reused	500	0	0	50	5		
Modified	500	20	20	50	5	50	0

Software Scale Drivers

Precedentedness	Nominal	Architecture / Risk Resolution	Low	Process Maturity	High
Development Flexibility	Very High	Team Cohesion	High		

Software Cost Drivers

Product		Personnel		Platform	
Required Software Reliability	Low	Analyst Capability	High	Time Constraint	Nominal
Data Base Size	Low	Programmer Capability	High	Storage Constraint	Nominal
Product Complexity	High	Personnel Continuity	Low	Platform Volatility	Low
Developed for Reusability	Nominal	Application Experience	Nominal	**Project**	
Documentation Match to Lifecycle Needs	High	Platform Experience	High	Use of Software Tools	High
		Language and Toolset Experience	High	Multisite Development	Nominal
				Required Development Schedule	High

Maintenance Off

Software Labor Rates
Cost per Person-Month (Dollars): 0
Calculate

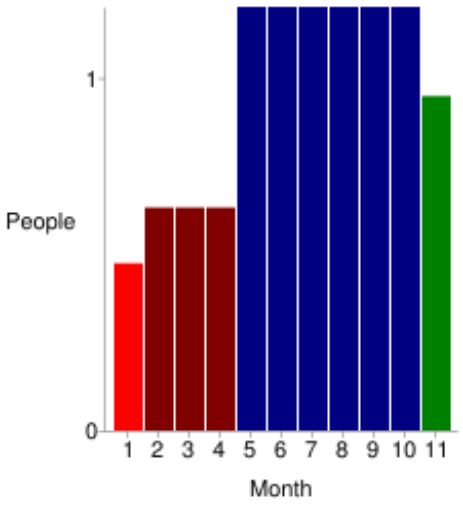
Results

Software Development (Elaboration and Construction)

Effort = 10.1 Person-months
Schedule = 10.2 Months
Cost = $0

Total Equivalent Size = 5270 SLOC

Staffing Profile



Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.6	1.3	0.5	$0
Elaboration	2.4	3.8	0.6	$0
Construction	7.7	6.4	1.2	$0
Transition	1.2	1.3	0.9	$0

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.3	0.8	0.2
Environment/CM	0.1	0.2	0.4	0.1
Requirements	0.2	0.4	0.6	0.0
Design	0.1	0.9	1.2	0.0
Implementation	0.0	0.3	2.6	0.2
Assessment	0.0	0.2	1.8	0.3
Deployment	0.0	0.1	0.2	0.4

Your output file is http://csse.usc.edu/tools/data/COCOMO_March_25_2017_06_50_29_414519.txt

Given the time period of 10.2 months, our group could have feasibly finished this project in its entirety if we were full dedicated to the task at hand. However, due to all members having either full-time jobs, being full-time students, or both, we were not able to complete the entirety project to its full potential. In an ideal situation we all have to code and we need at lease couple more people to code with more time.

VIII. WSD

Name	Role
Amani Konduru	Project manager, back-end (PostgreSQL), documentation
Benjamin Garber (Daniel)	Server developer (IMAP LIST, SMTP MAIL TO), documents
Edward Bull	Server developer, Client developer (except GUI), back-end (SQLite3), back-end (PostgreSQL), documents
Paul David Utesch	GUI developer and back-end (PostgreSQL)

IX. Gantt Chart

Title	ID	Participants	Status	Estimate Pts.
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	bgarber	Future	1
Server SmtpClient	S-01021	bgarber	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
Document 4 #2	S-01063	akonduru	Accepted	1
Document 4 #4	S-01064	Putesch	Accepted	1
Document 4 #5	S-01065	Ed Bull	Accepted	1
Document 4 #9	S-01066	Ed Bull,Putesch,akonduru	Accepted	1
Document 4	S-01067	Ed Bull,akonduru	Accepted	1
Document 6 #3	S-01068	Ed Bull	In Progress	2
Document 6 #4	S-01069	bgarber	In Progress	1
Document 6 #5	S-01070	akonduru	In Progress	2

Document 6 #9	S-01071	akonduru	In Progress	2
Set up PostgresSQL test server	S-01053	akonduru	In Progress	2
Implement Test Database	S-01074	akonduru	Accepted	2
Server ServerController	S-01075	Ed Bull	Accepted	2
Server SmtpServer	S-01076	Ed Bull	Accepted	2
Server CmdProcessor	S-01079	bgarber	Accepted	2
Server QueryGenerator	S-01078	Ed Bull	Accepted	2
Server SmtpClient	S-01077	Ed Bull	Accepted	2
Doc6 #3	S-01068	bgarber	Accepted	1
Doc6 #4	S-01069	akonduru	Accepted	1
Doc6 #5	S-01070	akonduru	Accepted	1
Doc6 #9	S-01071	Ed Bull	Accepted	1
Final Document	S-01072	Ed Bull, akonduru, bgarber	In Progress	2
Presentation	S-01073	Ed Bull, akonduru	In Progress	2

X. Resumes

407-902-5450 • EBULL@KNIGHTS.UCF.EDU
EDWARD BULL
10975 WITTENRIDGE DR. • ALPHARETTA, GA 30022

EDUCATION

Georgia State University **100/120 Credits**

- Bachelor of Sciences in Computer Science

University of Central Florida **2008 - 2010**

- Master of Arts in English, Dolores A. Auzenne Fellowship Recipient

SKILLS

Python, Java, C, HTML/CSS/Javascript, PHP. Familiar with Unix/Linux, Windows, and VMware ESXi environments.

EXPERIENCE

VERT Intern **2014 - present**

Tripwire, Inc

- **Role:** Set up testing environments, QA IP360 rules, develop IP360 rules
- **Highlights:**
  - Development tasks, especially remote vulnerability detection (ManageEngine)
  - NVIDIA video card detection
  - Created scripts to accelerate tasks (e.g., CPE names, fixing SPM reference links)

Instructor of Record **2009 – 2013**

University of Central Florida

- **Role**: Sole instructor for a dozen English courses including Composition and Creative Writing
- **Highlight**: Piloted a new curriculum based on research about skill transfer and cognitive theory.

REFERENCES

Farhan Jiva, Tripwire VERT: fjiva@tripwire.com
Craig Young, Tripwire VERT: cyoung@tripwire.com
Lane Thames, Tripwire VERT: lthames@tripwire.com
Kristina Kopic, Content Specialist at the Ruderman Family Foundation: kristina@rudermanfoundation.org

AMANI KONDURU

akonduru2@student.gsu.edu | 404-960-5971
www.linkedin.com/in/amani-konduru | 2507 Lawrenceville Hwy
www.GitHub.com/Aamani1 | Apt 2, Decatur GA 30033

EDUCATION

Georgia State University Honors College, Atlanta, GA — Expected May 2019
Bachelor of Science, Computer Science GPA: 3.71
Dean's List (Fall '15-Present)

Relevant Courses: Java Programming, Software Engineering, Web Programming, Database Systems, Data Structures

TECHNICAL SKILLS

Programming	Java, C, PHP, HTML, CSS, Unix Shell Programming, and MySQL
Operating Systems	Windows, Unix, MacOS
Database Management	SQL Server
Other	VersionOne (Agile Software), GitHub, Slack, and Mathematica
Languages	Telugu, and Hindi

RELATED EXPERIENCES

Georgia State University HackGSU, Atlanta,GA — Sep 2016-Present
Lead

- Work for the sponsorship division. Communicate with sponsors and negotiate on what they could provide.
- Allocate and craft the budget of $30,000+ for spring '16 and fall '17 Hackathon.
- Participated and developed an android application called Central in a group in spring '16 Hackathon and is published on Devpost.

Nivasoft Inc. Monroe, NJ — May-Aug 2014
Intern

- Explored on how a small information technology company functions.
- Qualified and quantified data that was provided to record by using Excel and ensured employee records were complete and reconciled accounts.

LEADERSHIP EXPERIENCES

Georgia State University Computer Science Club, Atlanta, GA — Sep 2015-Present
Co-President

- Coordinate meetings and events for 200+ members. Discuss finances, events, growth and improvement.
- Host many programming events for more than 40+ attendees. Survey and analyze success of each event.
- Lectured 130+ students in web development.
- Improve and boost social media and other accounts- Facebook, Google, GroupMe and Slack.

Georgia State University Women in Technology, Atlanta, GA — Sep 2016-Present
Vice President of Programs

- Initiated and established Women in Technology (WIT) campus at Georgia State University (GSU).
- Marketing and organizing events for WIT on GSU Atlanta Campus for 25,000+ students.
- Enable women in STEM to attain opportunities, so they can succeed and thrive in technology fields.
- Create opportunities for professional development through mentorship, insight and help shape a career path.

Georgia State University Student Alumni Association, Atlanta, GA — Aug 2015-Present
Board of Director

- Organize events for the entire campus to help students succeed life after college.
- Help students meet 100+ Georgia State Alumni and organize Lunch and Learn workshops.

BENJAMIN GARBER

- 4221 Kinsmon Way, Marietta GA 30062
- C: 404-824-3317 | bdanielgarber@gmail.com

SUMMARY OF WORK EXPERIENCE

- Technology Troubleshooter for a Small Business
- Assisted Management in Reorganization of Online Files
- Customer Service – Front Desk / Personal Assistance
- Sales Representative

EDUCATION & ACHIEVEMENTS

GEORGIA STATE UNIVERSITY – 08/2014 to Present
HONORS COLLEGE
Atlanta, Georgia

***Major*: Computer Science**

***Graduation*: May 2018 (expected)**

- Dean's List
  - 5 Semesters
- Hope Scholarship Recipient
- Honors College Scholarship Recipient
- Zell Miller Scholarship (Full Tuition)

NOTABLE UNIVERSITY CLASSES TAKEN

- *Design & Analysis: Algorithms (4520)*
- *Fundamentals of Game Design (4821)*
- *Mobile App Development (4360)*
- *Data Structures (3320)*
- *Introduction to Compilers (4340)*
- *Operating Systems (4320)*
- *Assembly Programming (3210)*
- *Computer Architecture (4210)*
- *Programming Language Concepts (4330)*
- *Web Programming (4370)*
- *Physics I (2211k)*
- *Physics II (2212k)*

Paul David Utesch

1475 Willow Lake Drive | Atlanta, GA 30329 | (678-710-5102) | pdutesch@gmail.com

Education

Georgia State University, Atlanta GA — Expected Graduation 2017
Bachelor of Science, Computer Science — GPA 3.39
Databases & Knowledge-Base Systems

Work Experience

MakeMyDeal.com, Atlanta GA — August 2015 - current
Database architect/ETL manager

- Managed old ETL tasks and deployed new ones
- Developed integration techniques for 3rd parties over API and FTP
- Optimized and Streamlined existing queries and tables
- Created SSIS and Jasper reports

MakeMyDeal.com, Atlanta GA — January 2015 – May 2015
Frontend Developer

- Troubleshot issues with our widget loading and appearing correctly
- Deployed an ELK stack

Federal Reserve Bank of Atlanta, Atlanta, GA — May 2014 – January 2015
IT intern

- Moved client computers to storage for renovations and restored setup when complete
- Troubleshot client's computer issues
- Managed loaner laptops and virtual machines

Total Computer Solutions, Greensboro, NC — June 2013 - July 2013
IT intern

- Performed remote maintenance on client computers
- Visited customer sites to backup information and to organize computers
- Prepared, delivered, and installed new computers at client location
- Repaired computers to working condition
- Salvaged useful parts from unrecoverable computers

Skills

- Java programming language
- SQL querying
- SSIS and Talend ETL tools
- Windows and Linux operating systems
- Computer Hardware
- Type speed average: 80 WPM

XI. Dictionary

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 Encryption:** 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.

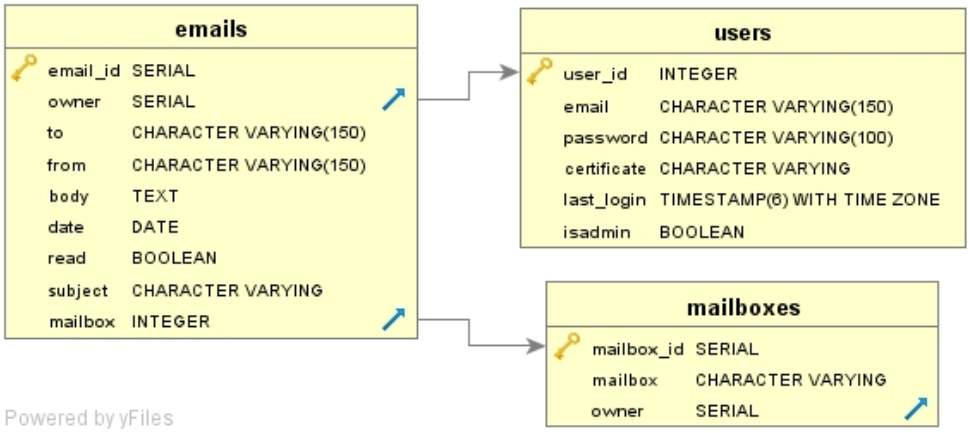
XII. Database

We used two databases: PostgreSQL and SQLite. The metadata is stored in a simple PostgreSQL database, with the user_id as a primary key. The client program has some basic login functionality and a SQLite database was used to store usernames and passwords as well as the user's local emails and mailbox structure. Both are open source and were best for our project from the limited options that are available. PostgreSQL is good for concurrency and was used for our server to support concurrent connections and operations. SQLite is easy to set up and work with, so it's an ideal solution for our client program. Digital Ocean virtual machine "droplet" was used for testing our server database. Digital Ocean droplets are inexpensive, reliable, and can be snapshotted for easy rollbacks should something go wrong during development. Our Digital Ocean runs on Ubuntu for easy terminal access over SSH.

PostgreSQL was used to manage the server database. Database 'coredb' was created to maintain the Adept mail server. The database coredb has three tables: users, mailboxes and emails. Currently the server is running on a command line prompt using sequel queries. The users table contains six attributes: user_id, email, password, certificate, last_login and isadmin. The user_id attribute is the primary key for this table. This allows us to distinguish each person in the Adept server. The user_id has a serial datatype so it auto increments as each user registers. The email attribute is unique for each user and lower case index was added to ignore case. The email attribute uses character varying datatype which limits the email to hundred and fifty characters. The password attribute also users character varying and limits the password of the user to hundred characters. Password-hash was supposed to be implemented, but was unable to due to time constraints. Certificate attribute is set to character varying with no limit it stores the SMPT certificate. The users table contains a last_login attribute. A timestamp attribute with time zone datatype was used which allows the users to see when they last logged in. Unlike other database PostgreSQL allows us to implement date and time values very easily and efficiently. The last attribute in the users table was the isadmin attribute which is a boolean attribute, which flags to see if the user is an admin or not. All the attributes except for certificate are not NULL. The mailboxes table consists of three attributes: mailbox_id, mailbox, and owner. The attribute mailbox_id is the primary key and is being auto incremented. The attribute mailbox being the name of the mailbox such as inbox, spam, etc. The owner attribute of this table determines that different users could have the same mailbox name but it might belong to some other owner. Therefore,

owner attribute is linked to user_id in the users table as a foreign key. The mailbox_id and owner are not NULL. The emails table consists of nine attributes: email_id, owner, mailbox, date, to, from, subject, body and read. Where email_id is the primary key and is auto incremented. The owner attribute here is also linked to the user_id in the users table as a foreign key. The mailbox attribute is linked to mailbox attribute in the mailboxes table as a foreign key. Together the database keeps track of the user data by their user_id, owner, and mailbox_id attributes along with their written emails.

Relational diagram for the PostgreSQL database 'coredb'



Sample data for the users table

user_id [PK] integer	email character varying	password character varying	certificate character varying	last_login timestamp with time zone	isadmin boolean
1	amani@gmail.com	amani	dhj1jn2oj	2017-04-21 00:33:54+00	true
2	amani@adept.com	amani	adcde	2017-04-21 00:33:54+00	false
3	ebull@adept.com	foobar	dsfalkjfdlkasf	2017-04-21 16:04:33.929...	false
4	daniel@adept.com	foobar	adcdefeef	2017-04-21 00:33:54+00	false
5	paul@adept.com	paul	xyz	2017-04-21 00:33:54+00	false
6	ed@adept.com	adept	hmmm	2017-04-21 00:33:54+00	true

Sample data for the mailboxes table

mailbox_id [PK] integer	mailbox character varying	owner integer
14	inbox	1
15	inbox	2
17	inbox	3
18	school	1
19	school	3

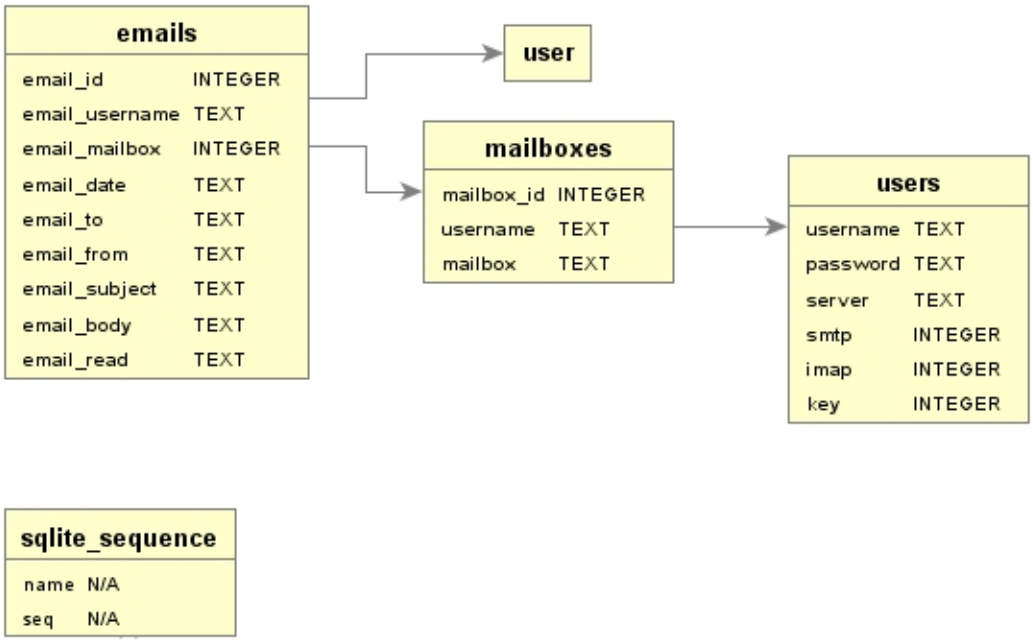
Sample data for the emails table

email_id [PK] integer	owner integer	to character varying	from character varying	body text	date date	read boolean	subject character varying	mailbox integer
35	1	amani@gmail.com, amani@adept....	ebull@adept.com	\ndoes it work?\n	2017-04-22	true	first multiple recipient	14
36	2	amani@gmail.com, amani@adept....	ebull@adept.com	\ndoes it work?\n	2017-04-22	false	first multiple recipient	15
37	1	amani@gmail.com	ebull@adept.com	\nfoo\n	2017-04-23	false		14
38	1	amani@gmail.com	ebull@adept.com	\nso let's hurry this up\n	2017-04-23	true	it's almost 3 am so i'm going to go to bed	14
39	1	amani@gmail.com	ebull@adept.com	\nfdasljkdlkfsjalkjdafs	2017-04-23	false	foobar	14
40	1	amani@gmail.com	ebull@adept.com	dsjaklflkjljkasdfljkasd\na...	2017-04-23	false	foobar	14
44	3	ebull@adept.com	ed@adept.com	\nthis is the body of an ...	2017-04-23	false	foobar	17
45	3	ebull@adept.com	ed@adept.com	\nbar\n	2017-04-23	true	foo	17
46	3	ebull@adept.com	ed@adept.com	\nfoo\n	2017-04-23	false	bar	17
47	1	amani@gmail.com	amani@gmail.com	\ntest\n	2017-04-24	false	test	14
48	1	amani@gmail.com	amani@gmail.com	\ntest2\n	2017-04-24	false	test2	14
49	1	amani@gmail.com	amani@gmail.com	\nllamas\n	2017-04-24	false	Presentation	14

SQLite

SQLite was used to manage the client database. Database ‘localdb’was created to maintain the Adept mail client. The database ‘localdb’ has three tables: users, mailboxes and emails. The users table contains six attributes: username, password, server, smtp, imap, and key. The username attribute is the primary key for this table. This allows us to distinguish each person in the Adept client.

Relational diagram for the PostgreSQL database ‘localdb’



Sample data for the users table

username	password	server	smtp	imap	key
Filter	Filter	Filter	Filter	Filter	Filter
amani@gmail.com	bIXffJNW19n5G9h3mDgt58...	localhost	465	993	ABC
ebull@adept.com	rXdwzAgN1AcmoKQ6Vt6bM...	localhost	465	993	ABC

Sample data for the emails table

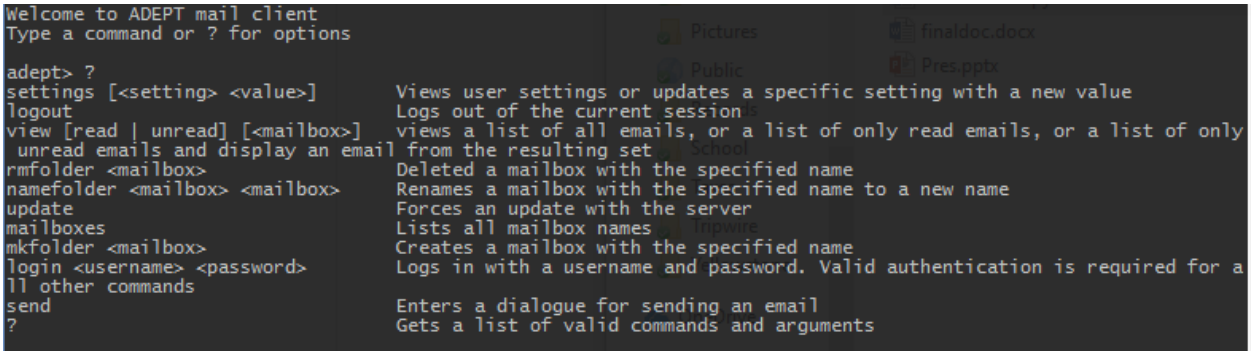
email_id	email_username	email_mailbox	email_date	email_to	email_from	email_subject	email_body	email_read
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
35	amani@gmail.com	1	2017-04-22	6enATeyUr+1...	iRfkg+4Vwi01...	L8lxXgWRh...	WGO6CzWr...	t
38	amani@gmail.com	1	2017-04-23	mPLkzyrT9JP...	iRfkg+4Vwi01...	/CVAqYh56u...	X5XteCtUy1...	f
39	amani@gmail.com	1	2017-04-23	mPLkzyrT9JP...	iRfkg+4Vwi01...	f4YcokYVVE...	TB3GilXUTg...	f
40	amani@gmail.com	1	2017-04-23	mPLkzyrT9JP...	iRfkg+4Vwi01...	f4YcokYVVE...	OMfHCk+Blj...	f
44	ebull@adept.com	3	2017-04-23	oX/xdfcQ4Irl0...	v5cJ/F/pvTWVz...	lJ3f89X4VUA...	eptcEMu9m...	f
45	ebull@adept.com	3	2017-04-23	oX/xdfcQ4Irl0...	v5cJ/F/pvTWVz...	j1tWqB+kl0r...	CHlIjo5YMQ...	t
46	ebull@adept.com	3	2017-04-23	oX/xdfcQ4Irl0...	v5cJ/F/pvTWVz...	j5VFrQeVI95...	OPmHeBC1...	f

Sample data for the mailboxes table

mailbox_id	username	mailbox
Filter	Filter	Filter
1	amani@gmail.com	inbox
3	ebull@adept.com	inbox
4	amani@gmail.com	school
5	ebull@adept.com	school
6	ebull@adept.com	cmf

XIII. User Guide

CLI:



1. The user must login using the login command before any other commands will execute. The user can log out using the logout command.
2. The user can use the settings command to change user settings, including the server IP address, IMAP and SMTP ports, and user key.
3. The update command must be run manually to update local storage of emails. Changes are not updated automatically.
4. The view command can be used to enter view mode. From there, emails can be selected and modified with the rmemail and mvemail commands.
5. The mailboxes command can be used to list all mailboxes belonging to the current user.

6. The rmfolder, namefolder (unimplemented) and mkfolder commands can modify mailboxes for the current user.
7. The send command enters a dialogue to send an email. Follow the prompts to populate the recipient, subject, and body.

GUI:

1. Login: Enter your username and password and press ‘Submit’.
2. Inbox: View your current emails. Press ‘Update’ to receive new emails. Press ‘New Email’ to bring up the New Email prompt.
3. New Email: Enter an email to send to, the subject, and the body for your email.

XIV. Project Legacy

Things that went well

1. The core functionality of the client and server worked well and consistently. We met almost all functionality requirements in the client.
2. We all learned about plenty of different software’s and applications and how to work with them.

Things that were okay

1. The IMAP update is set up to send all commands with individual authentications. Although this works, it is a very slow process.
2. The SSL connection does not involve certificates. Although this helps to encrypt the communication between the client and the server, it leaves the system open to a man in the middle attack.

Things that went badly

1. The GUI lacked a large portion of the functionality of the client.
2. We were not able to expand the server to allow for communications with external mail servers.