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

3/7/2017

Table of Contents

Table of Contents

[A secure mail server and client pair 1](#_Toc480594217)

[I. Introduction 3](#_Toc480594218)

[II. Requirements Traceability Matrix (RTM) 4](#_Toc480594219)

[III. Use Case, Sequence and Interaction diagrams 6](#_Toc480594220)

[IV. Object Design 6](#_Toc480594221)

[V. Test Cases (document) 6](#_Toc480594222)

[VI. Rational (for the entire project) 7](#_Toc480594223)

[VII. Fictional Point Cost Analysis and COCOMO (with comparison and conclusions) 8](#_Toc480594224)

[VIII. Project Legacy 8](#_Toc480594225)

[IX. WSD 8](#_Toc480594226)

[X. Gantt Chart 9](#_Toc480594227)

[XI. Dictionary 10](#_Toc480594228)

[XXII. Resumes 12](#_Toc480594229)

[XXII. User Guide 13](#_Toc480594230)

# 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 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 |

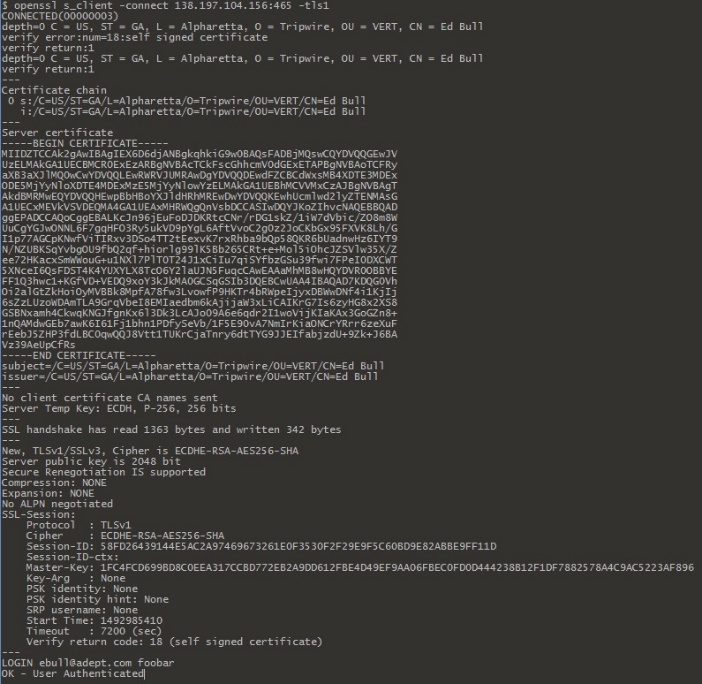
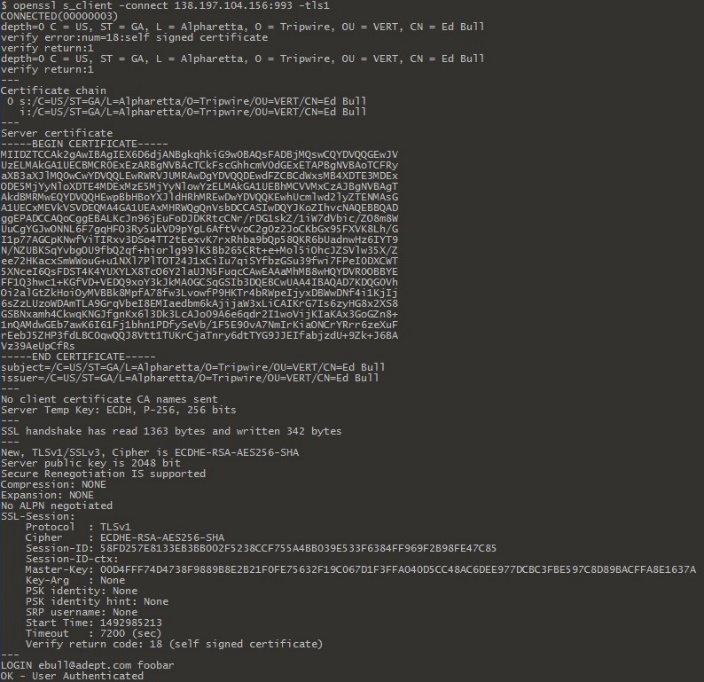
# III. Use Case, Sequence and Interaction diagrams

# IV. Object Design

# V. Test Cases (document)

|  |  |
| --- | --- |
| Test-case Identifier | ServerConnectivity |
| Feature | RTM: 4, 5, 7, 8 |
| 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. Once each connection is established, a series of commands will be sent and the responses compared against expected results. |
| Special Requirements | An authenticated account on the test server. |

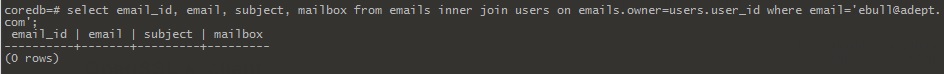
Target: Our Digital Ocean VM at 138.197.104.156 running the latest version of server.jar to demonstrate internet connectivity.

Test 1: IMAP Test 2: SMTP

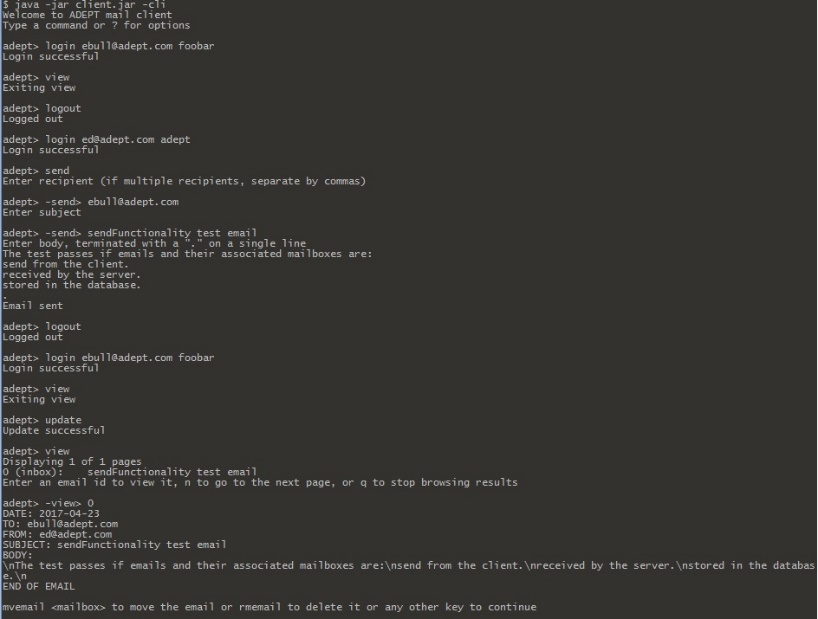
As is visible from the openssl s\_client information, SSL authentication was successful (if naïve) and the LOGIN command received the expected output on both ports. Note that we are using IMAP authentication on the SMTP port. This is an intentional time-saving implementation.

|  |  |
| --- | --- |
| Test-case Identifier | SendFunctionality |
| Feature | RTM: 1, 5, 7, 10, 11, 12, 13, 14, 15 |
| 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. |

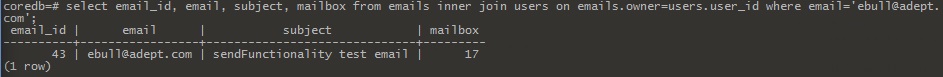
Pre-test condition: The email [ebull@adept.com](mailto:ebull@adept.com) has no current emails before the test.



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

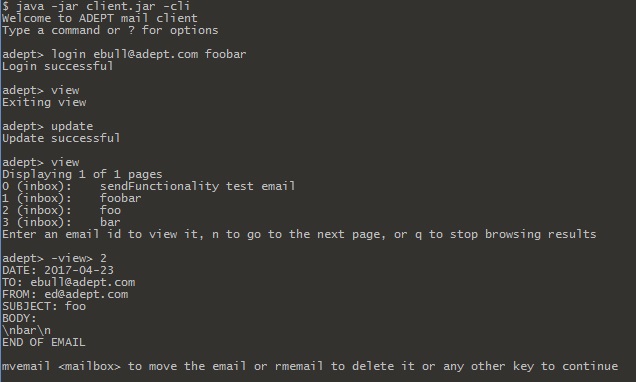


Post-test condition: The email [ebull@adept.com](mailto:ebull@adept.com) has received the test email.



|  |  |
| --- | --- |
| Test-case Identifier | UpdateFunctionality |
| Feature | RTM: 1, 7, 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. |

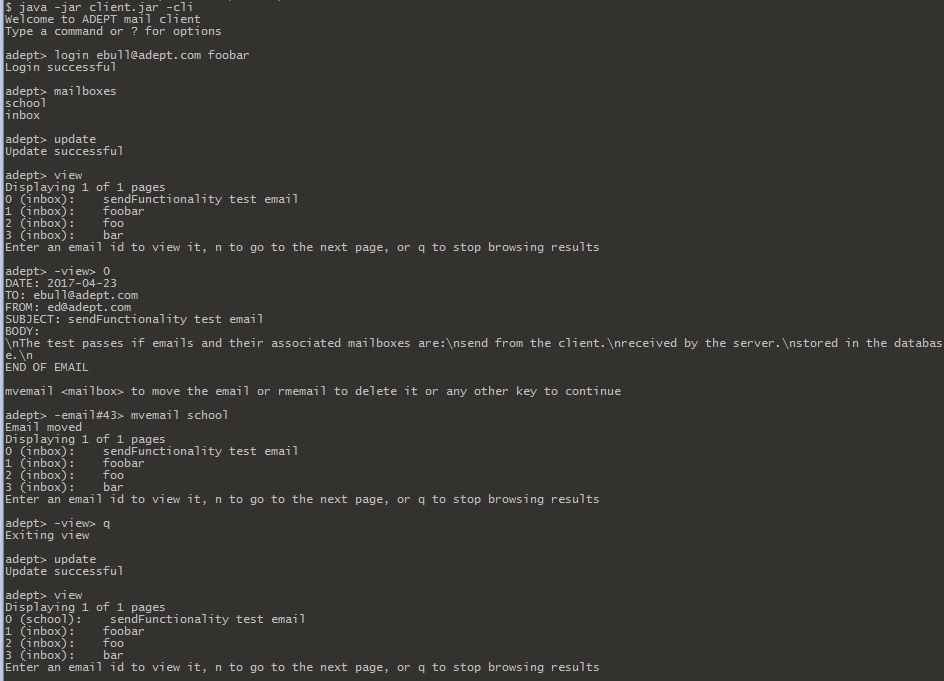
Pre-test condition: By deleting the Adept Mail Client's local sqlite database, we can ensure a valid testing pre-condition. The emails do not show in the [ebull@adept.com](mailto:ebull@adept.com) inbox.



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

|  |  |
| --- | --- |
| Test-case Identifier | MoveEmailFunctionality |
| Feature | RTM: 1, 2, 3, 7, 10, 12, 13, 14, 15 |
| 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. |

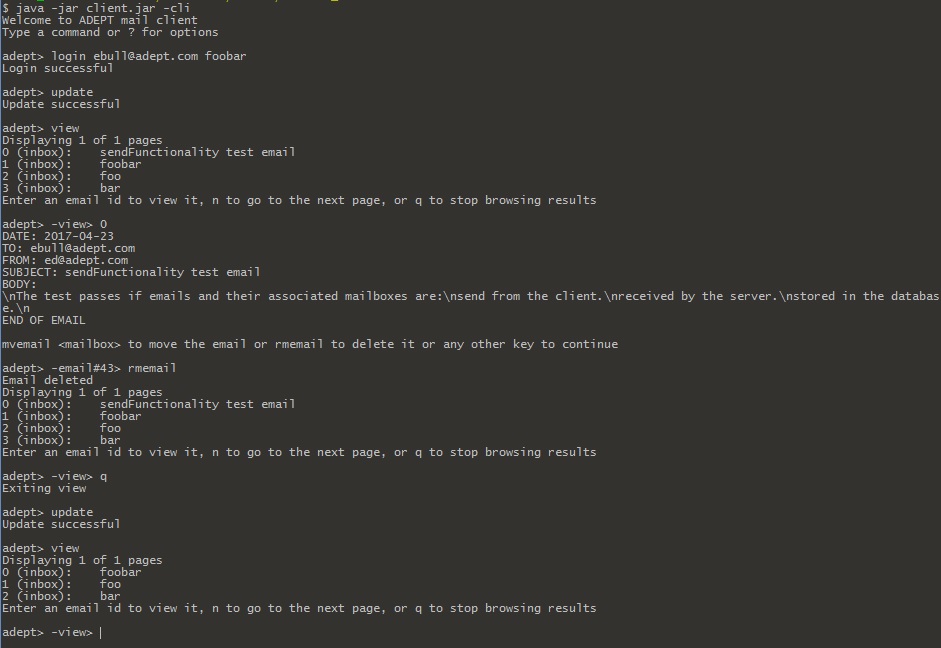
Pre-test condition: A valid email for [ebull@adept.com](mailto:ebull@adept.com) in the "inbox" mailbox.



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

|  |  |
| --- | --- |
| Test-case Identifier | DeleteEmailFunctionality |
| Feature | RTM: 1, 2, 3, 7, 10, 12, 13, 14, 15 |
| 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](mailto:ebull@adept.com) mailbox.



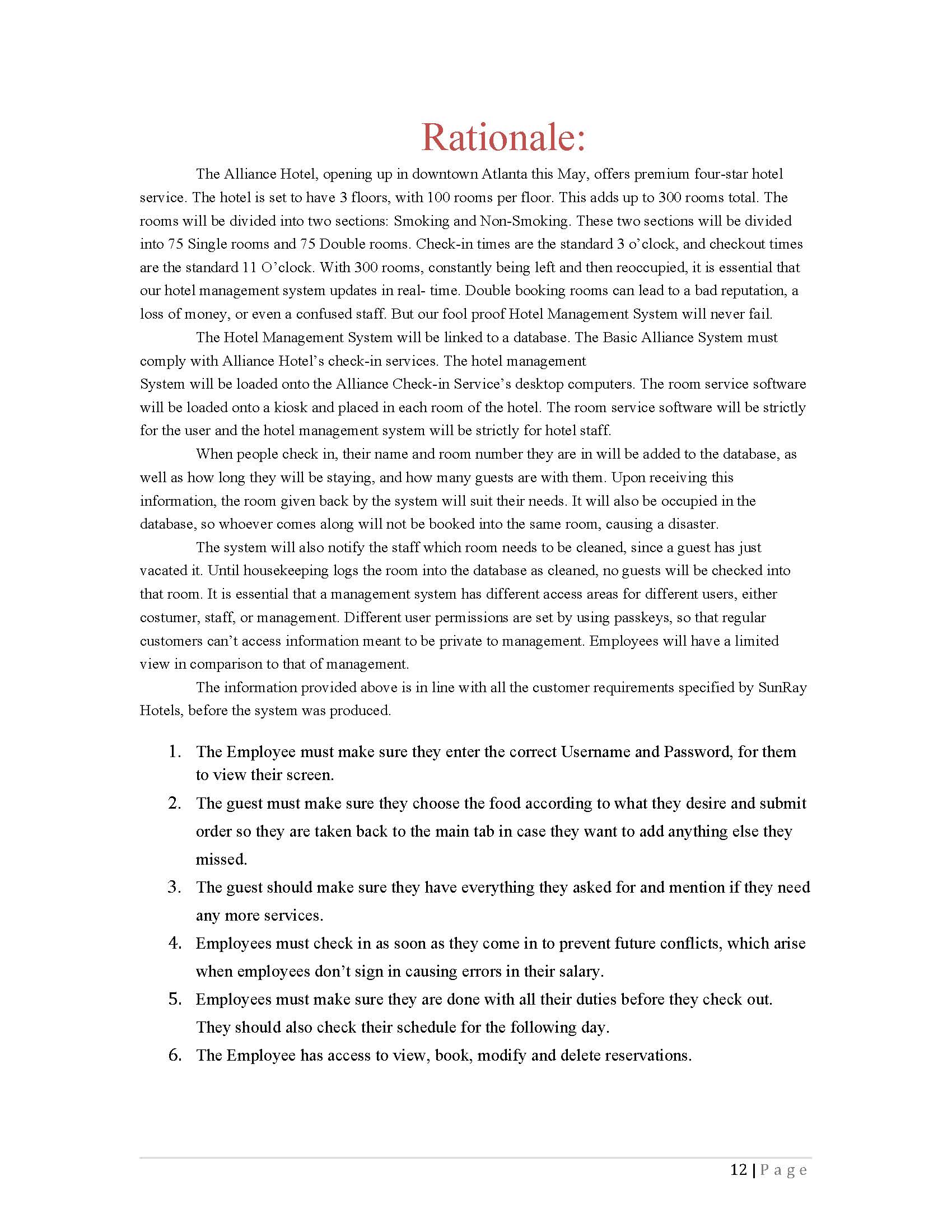
Post-test condition: The email is no longer in the [ebull@adept.com](mailto:ebull@adept.com) mailbox.

|  |  |
| --- | --- |
| Test-case Identifier | CreateMailboxFunctionality |
| Feature | RTM: 1, 2, 3, 7, 10, 12, 13, 14, 15 |
| 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 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. |

|  |  |
| --- | --- |
| Test-case Identifier | DeleteMailboxFunctionality |
| Feature | RTM: 1, 2, 3, 7, 10, 12, 13, 14, 15 |
| 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. |

|  |  |
| --- | --- |
| Test-case Identifier | RenameFolderFunctionality |
| Feature | RTM: 1, 2, 3, 7, 10, 12, 13, 14, 15 |
| 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. |

# VI. Rationale (for the entire project)



# VII. Fictional Point Cost Analysis and COCOMO (with comparison and conclusions)

# VIII. Project Legacy

# IX. WSD

|  |  |
| --- | --- |
| Name | Role |
| Amani Konduru | Project manager, tester, document handler, and Back-End developer (PostgresSQL) |
| Benjamin Garber (Daniel) | Java Developer |
| Edward Bull | Main Java Developer  -Coded the Server and Client prototypes into  classes that can will be used in the GUI |
| Paul David Utesch | Create the GUI and test its functionality |

# X. 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 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 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.

# XXII. Resumes



# XXII. User Guide