Software Design Specification: Athena

OlympuSoft
Norman Maclennan
Stephen Failla
Gregory LeBlanc

Revision History:
First Draft: 3/3/2010
Final Edit: 3/27/2010
Submitted: 3/28/2010

Table of Contents

1 Introduction

1.1 Purpose of this Document

The Software Design Specification defines and describes the detailed development requirements for the software. This specification will identify and describe the design requirements established by the software engineer(s) that are conducive to the design and testing of the software and its specified requirements.

1.2 Scope of the Development Project

This scope of the development project can be described in three sections:

1.2.1 Project Name

Athena

1.2.2 Functionality Overview

Athena will be a standalone Java-based communication application that will allow users to exchange encrypted instant messages through a secure central server. Athena will have the ability to process encrypted messages in real time. Athena will have a constant connection to a database, and users will be able to observe the status of other users through a contact list in the main window of the application.

Athena will give users the option to create a user name, password, and encryption key pair for data security. This option will be available through the Athena web site as well as the program's interface.

The software interface will consist of an offline authentication interface and an online communication interface, with a subsystem of menus and windows exclusive to the online interface, available only to the active registered user. The software interface look and feel of Athena will be customizable to the user. Basic appearance and layout characteristics will have multiple options for active users.

1.2.3 Benefits

Athena users will have the protection of advanced encryption during all online communication sessions. Athena's level of encryption will provide substantial data protection and confidentiality for all users.

The security aspects of Athena will not complicate the user experience. The interface of Athena will remain simple and intuitive. Athena's platform compatibility will allow for simple and efficient source code collaboration and a global appeal to users of all platforms.

1.3 Definitions, Acronyms, and Abbreviations

See Appendix A

1.4 References

1. *Class HashTable*. (n.d.). Retrieved 3 15, 2010, from http://java.sun.com/j2se/1.4.2/docs/api/java/util/Hashtable.html

2. OlympuSoft. (2010). *Software Requirements Specifications: Athena.* Boston: OlympuSoft.

3. *Software Design Specification*. (n.d.). Retrieved 3 11, 2010, from http://www.cmcrossroads.com/bradapp/docs/sdd.html

4. *Software Design Specification and Software Module Specification*. (n.d.). Retrieved 3 05, 2010, from http://sdesmedt.wordpress.com/2006/08/22/software-design-specification-and-software-module-specification/

5. *The GNU Crypto Project*. (n.d.). Retrieved 3 20, 2010, from GNU.org: http://www.gnu.org/software/gnu-crypto/

1.5 Major Software Requirements

The software will require the Java Runtime Environment to function correctly on the user system. Java Standard Edition 5 or greater is strongly recommended for optimal application performance. The Aegis Server architecture will require the MySQL database language to be installed to successfully communicate with the Athena Database. This database communication will also require the JDBC library as a subsystem of the Java development package. The Client architecture package will not require these database interaction tools since all communication with the database will be handled by the server.

The GNU-Crypto library will be required by both the Server and Client package architectures. This encryption-oriented library will interact with Java to implement data encryption and decryption, as well as secure key pair generation to ensure a secure connection between Athena users.

1.6 Design Constraints and Limitations

The user must have an active internet connection in order for the application to initiate an online session with the server. The Authentication Interface will be available, but no communication-related modules or components will be available, and any attempts to communicate with the server will be refused until an internet connection is established.

The Server responsible for database management, user connection, and communication activity must also be online and active in order for successful data communication and encrypted connections to be established. This will be the responsibility of the Athena server maintenance team. No user manipulation or maintenance of the server will be necessary.

There are no specific hardware constraints for the software. It is recommended that the target system for the software has at least 512MB of RAM and a 1.5-gHz processor or better. While these are not required hardware specifications and the application may be able to run

on less efficient settings, it is recommended for optimal system performance, reliability, and functionality.

1.7 Changes to Requirements

The following changes will be made to requirements that were discussed in the Software Requirement Specification for the Athena software:

Requirement	Change To Requirement	Rationale
Send File	Removal	Requirement is not of high priority, and must be removed to effectively meet active deadlines of higher priority requirements.
Send One-time-address Email	Removal	Requirement is not of high priority, and must be removed to effectively meet active deadlines of higher priority requirements.
Send Permanent-address Email	Removal	Requirement is not of high priority, and must be removed to effectively meet active deadlines of higher priority requirements.

1.8 Overview of Document

Section 1 of this document identifies the purpose and scope of the software and the document and lists all definitions, acronyms, and references. Any changes, such as removal or addition of requirements are also noted in this section.

Section 2 is an overview of the design specifications for major data structures within the software, as well as the file structures and data elements that make up these fundamental data structures.

Section 3 contains an overview and explanation of major system architectures, modules, and component relationships, represented primarily through architectural diagrams. The process flow of data between architectures and major design components is also identified by these diagrams.

Section 4 contains a detailed description of major design components through a specific component analysis template. This template identifies the detailed design requirements for each component listed. Interface design and interaction is discussed in Section 5.

2 Data Design

Athena's overall data design will create a better understanding for the detailed analysis of architecture and components later in the document.

Athena will be designed in the Java language. It will be supplemented by a MySQL database and the JDBC and GNU-Crypto libraries. It will be divided into three distinct categories:

1. Front End User Interface

The GUI will be developed and implemented in Java using the Swing API. It will serve as the means of interaction between registered Athena users. It will interface with the database through the server when the user logs in or creates a user name and account. More information on the GUI user experience, including screenshots, can be found in Section 5 of this document.

2. Database

Athena's database will be implemented in a secure local MySQL environment. The database will be named *Athena_Auth* and will store the user name, hashed password (using SHA-256), first name, last name, age, and email address of a registered user. The database will be created and accessed locally on the same machine as the server. Because the authentication traffic does not have to travel over the public domain, this will provide an additional layer of security and privacy to the user. The “Create” and “Update” SQL statements will be implemented in the internal methods of the Athena software through the user of different software modules.

3. Internal Methods

The internal methods will be the subordinate functions of Athena that allow users to register, communicate with other users, change system preferences, add and remove users in a contact list, change Athena's theme and connect to the Server and Database (using JDBC). The internal methods will collectively associate the three main system architectures in the software, as these architectures will share the functionality of several component modules and data.

2.1 Data Objects and Resultant Data Structures

Athena will utilize Java’s hash table data structures to store key value data pairs. This will allow the server to correctly identify the socket, data stream, and availability corresponding to a specific user name. The Java socket structure will be used primarily for inter-computer communication, using Java’s *DataOutputStream* class to transfer user message data through a particular socket. More information on the use of this class is explained in the comments for source code for the software.

All messages sent in the communication interface will be stored in a *String* data type retrieved from Java plaintext text fields in various interface windows. All input passwords will be stored in encoded Java password fields for additional security. Before messages are encrypted and sent to the server, they will be transferred to a Java byte array structure to enable the necessary modules to process the data and successfully encrypt and decrypt user information.

2.2 File and Database Structures

The software will integrate a single database to store user account information. All other information generated by users will be temporary and dynamically linked to active communication sessions. When a session is terminated by a user, all account-unrelated information will be terminated. The account information database structure is as follows:

Database Name	Fields	File Structure
Athena_Auth	• user_id • user_name • password (SHA-256) • first_name • last_name • age • email_address	contacts.xml

Below is a sample *contacts.xml* file structure. This file will store all contact names and corresponding public keys for contacts that have been manually added to a particular user's contact list.

Associated public keys will be stored with contact names to be used as a parameter for initiating an encrypted data communication session with the corresponding contact. Contact private keys will not be shared or stored in the *contacts.xml* file to preserve user security and privacy.

File Structure: contacts.xml

Sample variables shown

```
<contactlist>
    <group>
        "group_default"
        <contact>
            <name>
                "user_name"
            </name>
            <pubkey>
                "user_pubkey"
            </pubkey>
        </contact>
    </group>
</contactlist>
```

Below is a sample *athena.conf* file structure. This file will formatted to Windows configuration file standards, but will use a cross-platform file extension for easier compatibility. This configuration file will be responsible for storing all customizable settings from the preference interface of a user's application. This file will be loaded every time a user logs into the Athena communication interface, allowing the user to extend personalized application settings across communication sessions. If the user has not saved any personalized settings, a default configuration will be loaded.

File Structure: athena.conf

Standard [.ini] file layout*
Sample variables shown

[General Options]
system_tray = 1
esc_tab = 0
spell_check = 0

[Encryption Options]
;1=RSA
;2=<Undetermined encryption method>
encryption_type=1

[Notification Options]
enable_notifications = 1
enable_sounds = 0

[Formatting Options]
font_face = "Times New Roman"
font_bold = 0
font_italic = 0
font_underline = 0

[Theme Options]
active_theme = 2

2.2.1 External File Structure

Athena will follow a standard application and file installation format. Below is a list of the default installer locations corresponding to operating system:

Windows: **C:\ProgramFiles\AthenaChat**

Linux: **/usr/lib/athenachat/**

MAC: **Applications/AthenaChat**

2.2.2 Global Data

Athena will store user account data in the local secure database on the server as discussed above. No data will be accessed publicly, and all user account information transmission will be encrypted between subsystem architectures. All internet traffic will be encrypted with the most up to date government standard.

No actual data elements will be global. This would contradict the concept behind Athena software that security and privacy insurance is the highest priority.

2.2.3 File and Data Cross Reference

Athena's data flow structure will allow the three major system architectures to collectively process and transfer encrypted messages over the Internet. The following table

of application processes provides a description of cross-referenced data between architectures:

Basic Process	Architectures Involved	Cross-Reference Description
Logging on to Athena	Server Thread Aegis Server Client Application	- User enters user name and password - Data is transferred to the **Server Thread** where it is verified with the database via the **Aegis Server** - The result of the verification is sent back to the **Client Application** - The user is logged on
Sending Messages	Server Thread Client Application	- User enters the message - The message is transferred to the **Server Thread** and redirected to the recipients' **Client Application** for the other user to read
Adding a Contact	Server Thread Client Application	- User enters the user name to add to the contact list - This user name is sent to the **Server Thread** where it is added to the user's updated contact list in the **Client Application** **-** It is also added to the user's local contacts file
Logging off of Athena	Server Thread Client Application	- The user clicks the Disconnect button The user's user name is sent to the **Server Thread** - The **Server Thread** sends out the user name to all related online contacts and updates **Client Application** contact lists to reflect user logoff

3 System Architecture Description

3.1 Overview of Modules / Components

The software will be designed strongly around scalability and reusability. A system of framework architectures that can be changed, updated, and reused easily, especially as an open source application, is essential for a successful and widely accepted design product. Any future changes to the system architecture will simply involve the modification of modules within data structures to enhance functionality and user experience. Security will also be a fundamental aspect of the software. Communication between architectures will be secure and private for every client.

There are three major architectural components to the software: the Aegis Server, the Server Thread, and the Client Application. These structures will be designed in two Java packages, a Server class package and a Client class package. The Server package will contain the Aegis Server and Server Thread architectures, while the Client package will contain the Client Application architecture.

3.1.1 Aegis Server Architecture

There are no exclusive components within the Aegis Server architecture. The Aegis Server will access the Athena Database as an external interface to the software. The Athena Database will contain user account information, but the client will not have the ability to directly manipulate this database information. The only ability for the client will be the option to create a new account, in which new information will be automatically added to the database by the Aegis Server. All user data manipulation is processed through the Client Application components.

3.1.2 Server Thread Architecture

The Server Thread architecture is a structural link between the Aegis Server and Client Application architectures. The Server Thread is the universal means of communication between the Aegis Server and Client Application. When the Client Application requests connection to the Aegis Server, a new instance of the Server Thread framework is created and assigned a socket connection that will allow communication between the Aegis Server and Client Application.

3.1.3 Client Application Architecture

The Client Application architecture contains all four of the Athena system design components. The four design components that make up the system are the Authentication Interface, Communication Interface, Communication Interface Menu, and Preference Interface. The Authentication, Communication, and Preference Interfaces will have the ability to communicate with the Server Thread framework. Any time one of these components requests access to the Athena Database or Aegis Server, the Server Thread will be used to process these requests. As a result, many modules used by these major design components will interact with the Server Thread architecture. It is important to note that all modules that interact with the Server Thread are called through the Client Application architecture's various components.

3.2 Structure and Relationships

The following diagram (Figure 3.2.1) is a Relational Systems Diagram description of the different actors that use the Athena Software System. Users and Administrators use the software directly through the GUI. Server Maintenance will take place periodically to effectively maintain a high quality of software and ensure that successful communication through Athena systems can be sustained. As explained in Section 1.5, the Athena software will make use of the Java, JDBC, and GNU-Crypto frameworks, as well as the Athena Database designed in MySQL. The software code will include low-level modules for integrating these frameworks. For more information on the included development libraries, please reference Section 1.4 and comments in the source code for the software.

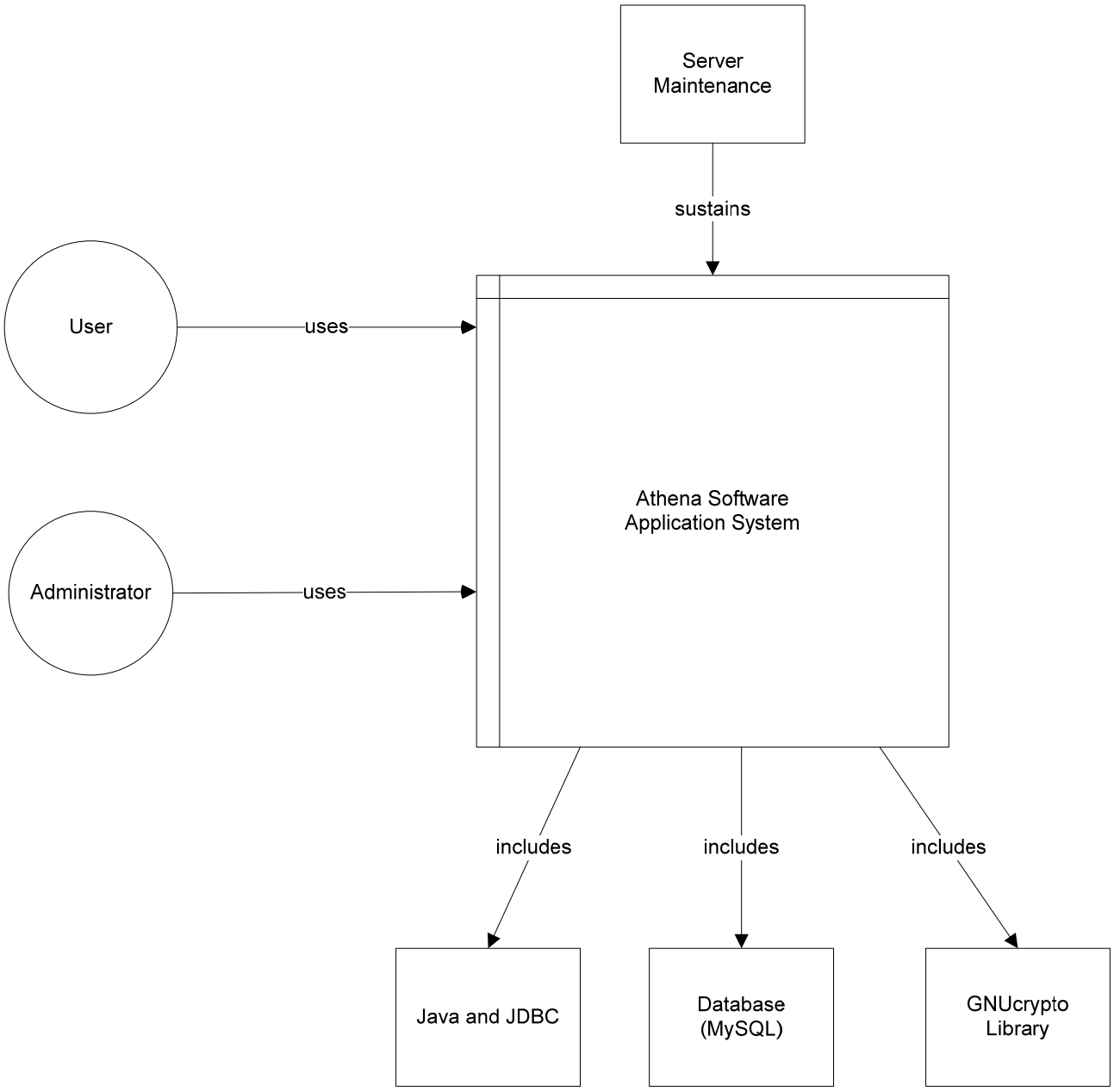
Server
Maintenance
sustains
User
uses
Athena Software
Application System
Administrator
uses
includes
includes
includes
Java and JDBC
Database
(MySQL)
GNUcrypto
Library

Figure 3.2.1

The diagram below (Figure 3.2.2) is an Architectural Context Diagram for the Athena software model. This diagram supplements the descriptions and relationships of architecture and major design components discussed in Section 3.1. The Athena software model is shown as the central system with Server and Client subsystems. The appropriate architecture and component relationships and dependencies are shown as well.

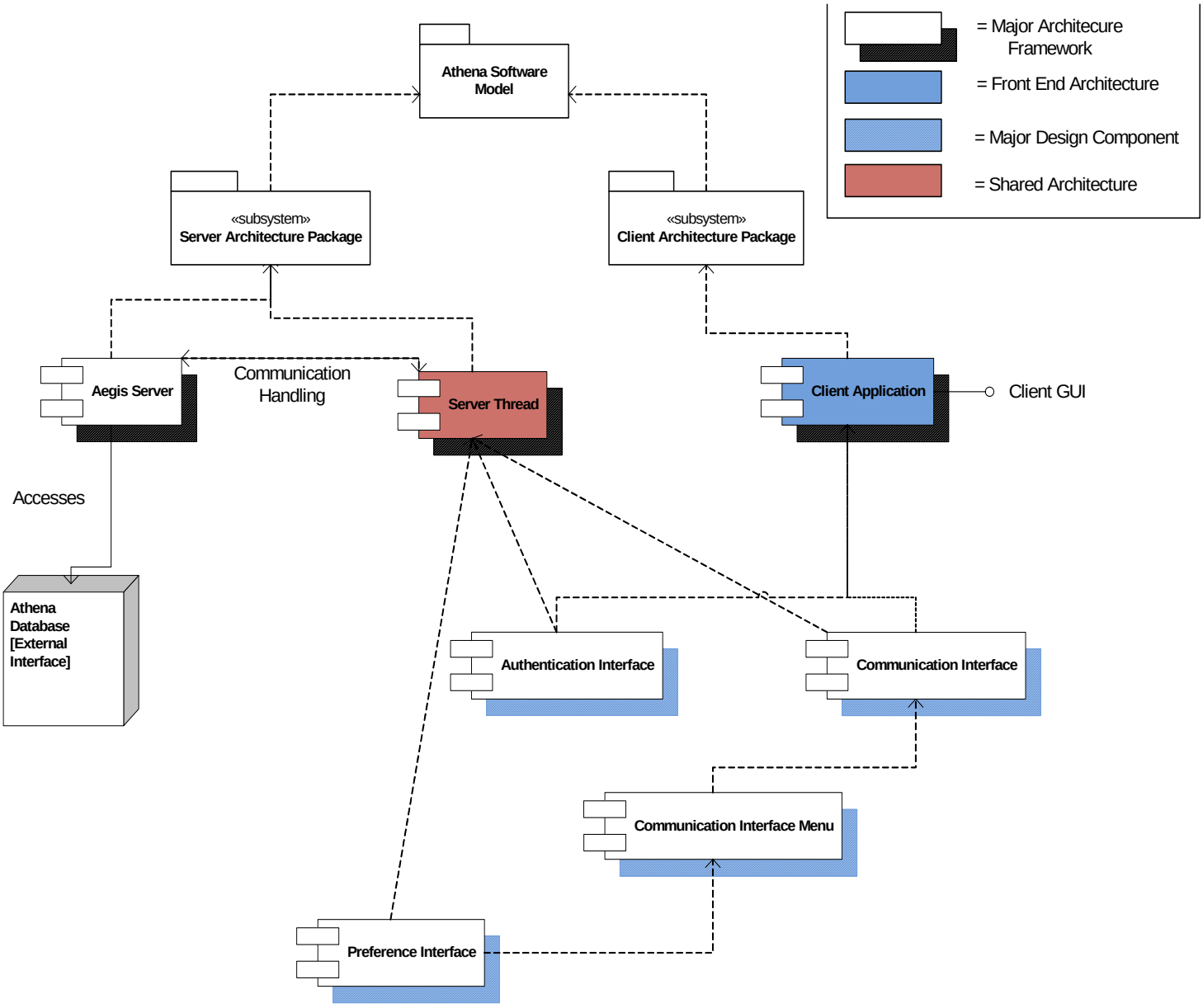

Figure 3.2.2

4 Detailed Description of Components

4.1 Component Template Description

4.n.1. Identification, Type and Purpose

Identification	Type	Purpose

4.n.2. Function

Function 1 Input [Associated Module]	Function 1 Description
Function 2 Input [Associated Module]	Function 2 Description
Function x Input [Associated Module]	Function x Description

4.n.3. Subordinates, Modules and Dependencies

Subordinates and Modules	Dependencies
SRS Requirements Satisfied	

4.n.4. Processing

Process 1	Process 1 Description	Process 1 Pseudocode
Process 2	Process 2 Description	Process 2 Pseudocode
Process x	Process x Description	Process x Pseudocode

4.n.5. Resources and Data

Resources	Data

4.2 Design Description of Communication Interface

4.2.1 Identification, Type and Purpose

Identification	Type	Purpose
Communication Interface	Subsystem of Client Application	The conversation area sub-component will contain all of the tabs for the user's active conversations. It will have a tabbed display with a single dedicated conversation in each tab. Each tab will contain a text area where incoming messages will be displayed and outgoing messages will be recorded, requiring all user-to-user conversation to be synchronized. It will also have a text field where users can input messages to be sent to intended recipients. The contact list sub-component will provide users with an interface that can be used to store preferred recipients for message communication. This interface will be designed as a list of contacts that are connected to the server at the time. The contact list will allow the user to choose recipients for specific messages. The Communications Interface will provide an interface to one of the main requirements of the program: the ability to send and receive messages between other users.

4.2.2 Function

Function Input [Associated Module]	Function Description
User name with which to start a conversation, chosen by the user via the contact list module [Conversation Area]	- User name is sent from the contact list to the conversation area - Module will either create new tab with text area and text field or set focus to existing conversation with chosen contact
Message from the current user to send to a dedicated recipient [Conversation Area]	- Message and recipient name are retrieved from text field corresponding to user's tab - Message and recipient data are output to encryption module for transmission to recipient - Message is decrypted and output on recipient conversation tab with user - Message also output on user screen to maintain conversation synchronization
Message being sent to user from another online contact [Conversation Area]	- Message is received on user's application - Message is sent to decryption module and displayed in plaintext on user's corresponding conversation tab - If no tab for contact exists on user interface, a new tab for contact is created and displayed
User clicks "X" icon on a conversation tab to close that tab [Conversation Area]	- Tab selected to be closed is retrieved and removed from user interface - All components and data within closed tab are destroyed permanently, tab no longer displayed
Contact list file "contacts.xml" from user machine in specific application directory path [Contact List]	- Contact list file is parsed and added to user's contact list object in the interface component - Process is transparent to user – occurs while component is being initialized
Contact user name to check connection status via server [Contact List]	- Contact name is sent to server and server sends message back to user application with connection status of contact - If contact is connected to server, contact name is displayed as visible in user contact list interface Otherwise, contact name is not displayed to signify offline status - Repeat for all contacts in contact list file
Message from server containing user name and status code of a contact when a contact logs on or off server [Contact List]	- If user contact list contains contact that has logged on/off, user contact list interface changes to reflect contact's status change - added to list if log on, removed from list if log off - If user contact list does not contain contact that has logged on/off, no changes will be made to user contact list interface
Contact name specified by user for addition to or removal from contact list [Contact List]	- If specified contact user name is added, contact name will appear in user contact list (if online) and be written to contacts.xml file - If specified contact user name is removed, contact name will no longer appear in user contact list (if online) and associated data will be removed from contacts.xml file

4.2.3 Subordinates, Modules and Dependencies

Subordinates and Modules	Dependencies
- Conversation area is housed in a tabbed pane - Tabs are created, destroyed, or manipulated based on various interactions - Each tab contains a text area for displaying the messages in the conversation - Each tab contains a text field that the user will use to send messages to the recipient - Contact list is a simple list connected to a hash table - GNU-Crypto library modules are used for encrypting and decrypting messages to/from the server and other users	- Component interacts with the GNU-Crypto encryption module when user sends a message to a recipient - Component interacts with the GNU-Crypto decryption module when retrieving and displaying messages from other users, as well as decrypting broadcast messages from the server for contact status changes.

SRS Requirements Satisfied				
Send Message	||	Receive Message	||	Contact List
View user Status	||	Add Contact	||	Remove Contact
Add Group	||	Remove Group	||	New Message Tab
Close Message Tab	||	Text View Area	||	Text Communication Area
Text Scroll Vertical				

4.2.4 Processing

Process	Description	Pseudocode
Create a Tab	There is a Java object that contains components and functions that each tab must use. When a tab is created, an instance of the object is created and added to a hash table for future reference.	GetClickedContactFromGUI(); CreateTab(); MapRecipientToTab();
Focus a Tab	When a tab needs to be focused, the hash table of all of active tabs is referenced. Each tab is mapped to an integer, and each integer corresponds to a "tab index". When the tab index is found, the tab is focused in the interface.	GetClickedTabFromGUI(); GetTabNumberofClickedTab(); FocusTabInGUI();

Update a Tab	Each tab has an object in it, containing various components and methods for updating the tab. For example, when a message is received in the conversation area from the decryption module, the sender is used to determine which tab the message must be displayed. Once the correct tab is chosen, the message is appended to the text area in that tab in the user's communication interface.	ReceiveMessage();
		DetermineSender();
		if(SenderHasTab()) {
		  AddToMessageArea();
		} else {
		  CreateTab();
		  AddToMessageArea();
		}

Remove a Tab	Each tab has an "X" button on it. When the "X" button is clicked, the tab is selected from the hash table. The object within the tab is selected and removed from the display and destroyed. Then the tab is removed from the tab pane.	GetInformationFromGUI(); GetClickedTabIndex(); DestroyTab(); RemoveFromHashTable(); RemoveTabFromGUI();

Process	Description	Pseudocode

Generate Contact List	When a user connects to the server, the program reads and parses the user's contacts.xml file to obtain the user names and public keys of the user's contacts. These user names are added to the GUI's contact list and matched with their respective public key. The program then queries the server for the connection status of each contact. If the contact is online, the contact is displayed on the list as online. Otherwise, the contact is added to the list but not displayed (offline).	ConnectToAegis();
		ReadContactFile();
		while(MoreContactsInFile) {
		QueryOnlineStatus();
		if(UserOnline()) {
		AddUserToContactList();
		MapPublicKeyToUser();
		}
		}

Update Contact Status	Once the user's contact list is created, the user will receive broadcast messages from the server with changes in contact connection information. When a contact logs on, the server sends out a broadcast message to all other known contacts to notify that the contact has connected. A similar message is sent when a contact disconnects. Each of these messages consists of a user name and a status code. When a user's program receives one of these messages, the contact list will either add or remove a particular contact from the online list, depending on the status code contained within the message.	ReceiveBroadcastMessage();
		GetUsername();
		MatchWithContactList();
		if(UserPresentOnList()) {
		UpdateStatus();
		} else {
		NoActionTaken;

Add Contact To Contact List	Once the user is connected to the server, the ability to add contacts to the contact list is now available by clicking a "plus" (+) icon under the contact list. The user will input the user name of the contact to add. The program verifies that the user name is valid (correct length, and format), and then queries the server for the new contact's information. The server responds with a message of whether or not the user name exists. If the user name does not exist, the program informs the user via an error message. If the user name does exist, the program requests the public key associated with the user name. When the program receives the public key from the database via the server, the user name and public key are added to the user's contact list and written to the contacts.xml file.	GetInfoFromGUI();
		GetUsernameFromUser();
		if(VerifyValidUsername()) {
		GetUserPublicKey();
		AddToContactList();
		WriteToContactsXML();
		} else {
		ErrorMessage();

Remove Contact From Contact List	The user is able to remove contacts from the contact list using a "minus" (-) icon under the contact list. The user selects one of the contact user names in the contact list and clicks the minus icon button. The program retrieves the user name of the selected contact and removes it from the contact list, as well as from the contacts.xml file.	GetInfoFromGUI(); DetermineSelectedUser(); RemoveFromContactList(); RemoveFromContactsXML();

4.2.5 Resources and Data

Resources	Data
- Negligible amount of CPU time when tabs are being created, destroyed, or updated - Memory in the form of RAM to store all of the sub-components in the communication interface, such as the text field, text area, and tabs - I/O channels via the mouse and keyboard - Java Swing library to render each interface sub-component.	- Each tab is mapped to a hash table, matching a tab to a remote user - The contact list is built using an external file named contacts.xml (see Section 2.1 for file structure) - The contact list will be modified using data supplied by the server if contacts connect or disconnect - The contact list can be modified via user input by adding and removing contacts from the list

4.3 Design Description of Authentication Interface

4.3.1 Identification, Type and Purpose

Identification	Type	Purpose
Authentication Interface	Subsystem of Client Application	The authentication interface component will allow Athena users to connect to the Aegis Server and access the communication interface. It will provide access to the primary features of the software. When a user runs the Athena program, the authentication interface is the first interface that is displayed. This component will provide the fundamental privacy and security verification needed to ensure the appropriate level of encryption in the communication interface.

4.3.2 Function

Function Input [Associated Module]	Function Description
User name, password, first name, last name, age, and email address for creating a new account [Create User]	- All input fields will be added to a new row of the local MySQL Athena database - Will allow user to register a user name for a new Athena account to communicate with the server and other registered users
User name and password for connecting to the Aegis Server [Login]	- Both fields are blank on initialization - Input data is sent to Aegis server to be checked against Athena database for account verification - If verification succeeds, communication interface will be displayed - If verification fails, error message will be displayed - User's connection status is changed to online and related contacts are notified
No inputs required to log off of Athena [Disconnect]	- User's socket and data stream connection will be terminated - User connection status will be changed to offline and related contacts will be notified - Communication interface is closed and user is returned to authentication interface

4.3.3 Subordinates, Modules and Dependencies

Subordinates and Modules	Dependencies
- The authentication interface uses subordinate Java classes for different modular processes - The ClientAddUser Java class is used when a user creates an account - The ClientLogin Java class is used when a user connects to the Aegis server - The ClientApplet Java class is used when a user disconnects from the Aegis server, as data is sent to related contacts with notification of updated connection status change - All other GUI-related components and modules cannot be accessed in the program without verification through this component	- This is the first component and interface accessed when running the Athena software, so no previous interface or component processes are necessary - The Athena database must exist for authentication data to be verified successfully - The Aegis server must be active and online for authentication data to be verified with the Athena database - All data fields must be populated when attempting to create an account or log on to the server – blank data will result in an error message - The Disconnect module is only available in the communication interface where the user is already connected – communication interface must be active for Disconnect module to be accessed
SRS Requirements Satisfied	
Create Account || User Name Entry || Password Entry Forgot Password || Connect || Disconnect Exit	

4.3.4 Processing

Process	Description	Pseudocode

Create a User	When creating a user, the create account window appears. After appropriate data fields are completed, info is sent to server for database verification. If requested user name exists, user is notified to choose a different name. Otherwise, data is stored in database and user is notified of successful account creation	{
		DatabaseConnection();
		sendInformationFromGUI();
		if (suppliedUserName exists) {
		ChooseNewName();
		} else {
		insertInfoIntoDatabase();
		}
		}

Connect to Server	When connecting to server, user name and password data is sent to server for database verification. If user name and password exist and match in database, user is directed to communication interface. If user name or password is incorrect, user is notified with error message and asked to enter correct information	{
		DatabaseConnection();
		sendInformationFromGUI();
		hashSuppliedPassword();
		if (suppliedUserName exists AND password matches passwordInDatabase) {
		allowLogin();
		displayCommInterface();
		} else {
		showErrorMessage();
		}

Process	Description	Pseudocode

Disconnect from Server	When disconnected from server, user connection status is sent to all related contacts to update appropriate contact lists. User socket and data stream connections are terminated and communication interface is closed.	{
		UpdateConnectionStatus();
		LogOffServer();
		}

4.3.5 Resources and Data

Resources	Data
- During authentication and interface window transition, a negligible amount of CPU and RAM is allocated during processing. - Resources are allocated primarily for Java Virtual Machine purposes. - Strength of active Internet connection on user machine will also determine varied processing speed of data authentication with server.	- All data fields for creating a user or connecting to server are stored in String variables and transferred using Java's DataOutputStream class to the server. - Dedicated data streams between users and server are mapped to corresponding socket for consistency of data transmission

4.4 Design Description of Communication Interface Menu

4.4.1 Identification, Type and Purpose

Identification	Type	Purpose
Communication Interface Menu	Subsystem of Client Application	The communication interface menu will be responsible for any user actions that are not part of the standard communication interface as described above. The communication interface menu is essentially a subsystem of the communication interface. The menu system will consist of five submenus, and each submenu will have an exclusive selection of system or interface-related button actions. The different submenus will allow the user to make changes to visual or system settings without having to navigate to a new or external window outside of the communication interface. The communication interface will remain active and no information in the interface will be lost when navigating between the menus. The communication interface menu component will enhance and expand the overall functionality and user experience of an online communication session. This component is required to provide access to the submenus and all of the modules included within each submenu. This component will inherit the requirements of all modules available within the submenus.

4.4.2 Function

Function Input [Associated Module]	Function Description
Change connection status, update personal status message on contact list, or exit application [File drop-down submenu]	- User will be able to select Disconnect, Status Message, or Exit from the File drop-down submenu. - Disconnect action will terminate the current active connection with the server. - Status Message action will allow the user to set a message that notifies other online users of current activity or location. - Exit action will terminate the Athena application entirely after prompting for confirmation.
Change account or interface settings related to current connected user [Edit drop-down submenu]	- User will be able to select Preferences or Change Password from the Edit drop-down submenu. - Preferences action will display a new window that contains an exclusive selection of additional options and actions related to the software. Any options or actions that are set in the Preferences window will be saved and loaded every time the user logs into the communication interface. - Preference settings will be relative to the user account linked to the online communication session. For a detailed description of the Preferences component, please see Section 4.5.The communication interface will remain open in the background. - Change Password action will allow the user to change the current password linked to the Athena account. The user will be asked to enter current password, enter a new password, and confirm the new password entry. Password entry will be verified for required format and length.
Adjust the view of the contact list in the communication interface [View drop-down submenu]	- User will be able to select only the Contact List View action from the View drop-down submenu. - Contact List View action will display a small checkbox window that will allow the user to select or de-select the “Show Contact List” option. When option is selected, contact list will appear in communication interface. If option is de-selected, contact list will not be shown in the communication interface.

Function Input [Associated Module]	Function Description
Change or extract encryption information related to the current session or user account [Encryption drop-down submenu]	- User will be able to select Generate New Key Pair or Export Current Key Pair from the Encryption drop-down submenu. - Generate New Key Pair action will allow the user to create a new encryption key pair for secured communication if the current key is in danger of being compromised or unreliable. For security purposes, the user account password will be required in order to confirm the encryption setting change. - Export Current Key Pair action will allow the user to extract the current encryption key pair to a local file that can be used with other encryption-based applications using the same encryption standards as the Athena software. For security purposes, the user account password will be required before exporting the key pair to a local file on the user's operating system.
Access important version and troubleshooting information related to the software [Help drop-down submenu]	- User will be able to select only the About Athena action from the Help drop-down submenu. - About Athena action will display a window with version information corresponding to the version of the software currently installed on the user's system. - About Athena window will also provide links to the Athena website for additional help and production information. The location of the user guide, if downloaded, will be displayed in the window as well.

4.4.3 Subordinates, Modules and Dependencies

Subordinates and Modules	Dependencies
- Communication interface menu will be a Java menu embedded in the communication interface window - Each submenu of this component will be a drop-down context menu with an exclusive selection of button actions related to the submenu category - Certain submenu button actions will display new window panels or message boxes when selected, depending on the scope of its functionality. - This component is used to access the Preferences design component. Detailed information regarding this component and its dependencies are described in Section 4.5.	- Communication interface must be initialized to access communication interface menu - Menu functionality will be independent from communication interface functionality - Successful account verification through the authentication interface must also occur before this component can be initialized and accessed
SRS Requirements Satisfied	
Disconnect || Status Message || Exit Preferences || Change Password || Contact List View Generate New Key Pair || Export Current Key Pair || About Athena	

4.4.4 Processing

After the communication interface menu is initialized with the communication interface window, all submenus are available for user interaction. When a user clicks a submenu, a drop down frame will appear on top of the communication interface with the corresponding list of button actions related to that submenu.

If at any time the user chooses a submenu and then clicks anywhere outside the drop-down list, the submenu will close and the focus will change to the clicked object or area.

Any time an action button is selected from a submenu, the drop-down list will close and the selected module will be activated in appropriate fashion. The selected submenu will only remain active and visible until the user makes a button action choice from that submenu or clicks outside of the drop-down list. The processing for each of the five submenus is as follows:

Process	Description	Pseudocode

Using the File drop-down submenu	The menu will listen for user to make a button selection. If user selects Disconnect button, program will prompt user to confirm choice. If choice is confirmed, user will be disconnected from server and communication interface will close. Otherwise, user will remain connected to server.
	If user selects Status Message button, program will prompt user to enter a new status message into a text box. After text is entered, status message will be posted to contact list of all related contacts. If no text is entered, error message will be displayed.
	If user selects Exit button, program will prompt user to confirm choice. If choice is confirmed, user will be disconnected from server and entire application will be terminated. Otherwise, user will remain connected and application will remain open and active.
		//Listen for user to make action button selection
		Case (SelectDisconnect()) {
		ConfirmSelection();
		If (User confirms choice)
		Call Disconnect() module;
		Else (User does not confirm choice)
		Return to communication interface;
		}
		Case (SelectStatusMessage()) {
		PromptForStatusMessage();
		If (User has successfully entered a new status message)
		Call StatusMessage() module;
		Else (No text was entered)
		Prompt user to enter text or cancel;
		}
		Case (SelectExit()) {
		ConfirmSelection();
		If (User confirms choice)
		Call Exit() module;
		Else (User does not confirm choice)
		Return to communication interface;
		}

Process	Description	Pseudocode

Using the Edit drop-down submenu	The menu will listen for user to make a button selection. If user selects Preferences button, program will initialize preference component window. Processing for this component is discussed in Section 4.5.
	If user selects Change Password button, program will prompt user to enter current password, a new password, and new password confirmation. If all data is entered correctly, module will process data. Otherwise, program will issue error message.
		//Listen for user to make action button selection
		Case (SelectPreferences()) {
		Run Preferences() component;
		}
		Case (SelectChangePassword()) {
		PromptForPasswordInfo();
		If (anyfields_blank) OR (anyfields_incorrect)
		Prompt user for correct/valid data;
		Else
		ChangePassword();
		}

Using the View drop-down submenu	The menu will listen for user to make a button selection. If user selects Contact List View button, a frame with a check box will appear to allow the user select if the contact list will be displayed in the communication interface. A checked box signifies a visible contact list. An empty box will hide contact list in the interface.	//Listen for user to make action button selection
		Case (SelectContactListView()) {
		ContactListView();
		}

Using the Encryption drop-down submenu	The menu will listen for user to make a button selection. If user selects Generate New Key Pair button, program will prompt user to verify account password for security purposes. If password is correct, new encryption key pair will be generated for account. Otherwise, an error message will be displayed regarding password entry.
	If user selects Export Key Pair button, program will prompt user to verify account password for security purposes. If password is correct, current encryption key pair will be exported to a local file in the Athena directory. Otherwise, an error message will be displayed regarding password entry.
		//Listen for user to make action button selection
		Case (SelectGenerateKeyPair()) {
		VerifyPassword();
		If (correct)
		GenerateNewKeyPair();
		Else
		Prompt user to enter correct password;
		}
		Case (SelectExportKeyPair()) {
		VerifyPassword();
		If (correct)
		ExportKeyPair();
		Else
		Prompt user to enter correct password;
		}

Using the Help drop-down submenu	The menu will listen for user to make a button selection. If user selects About Athena button, a new frame will be displayed with links and information regarding troubleshooting and general help topics for the software.	//Listen for user to make action button selection
		Case (SelectAboutAthena()) {
		AboutAthena();
		}

4.4.5 Resources and Data

Resources	Data
- When a user chooses a submenu or makes a choice from a submenu within the communication interface menu, a negligible amount of CPU time is allocated to the action process of the selected button. - Any time a button action initiates a new window or panel, a small block of virtual RAM will be allocated to that module until it is closed or destroyed. - All button actions will be triggered by mouse action. All data input for text fields within submenu modules will be entered with a keyboard. - Java Swing API used to render submenus and display action buttons within each menu. - Each submenu will have the ability to generate individual Java panels or frames depending on scope of particular action button modules.	- The communication interface menu is comprised of submenus with action buttons, so there is no data to be stored on the top level of the component. - Depending on action button selected, some modules will prompt user for data input, such as password or account verification. - Any data required for input will be input to text boxes and stored in temporary variables for processing until action is completed.

4.4 Design Description of Preference Interface

4.4.1 Identification, Type and Purpose

Identification	Type	Purpose
Preference Interface	Subsystem of Client Application	The preference interface menu will allow users to customize and personalize the Athena application experience. A series of settings and preference choices will be available in the interface for the user to choose from and save for future sessions. If preferences have been saved, they will be loaded to the user's application every time the program is opened. This will allow for a customized interface across communication sessions, without the need to re-apply settings every time the user connects to the server. The preference interface will allow the user to personalize the application to improve user convenience and experience.

4.5.2 Function

The preference interface will have a vertical list of five icons that serve as action buttons to switch between grouped sections of settings. The five icons will be labeled from top to bottom in the window as General, Notifications, Encryption, Formatting, and Theme. Each section will consist of one or more settings related to the title of the section. The functions of all preference interface modules are as follows:

Function Input [Associated Module]	Function Description
User clicks a check box labeled “Show Athena in System Tray” and clicks “Save” [General]	- The program will create an Athena logo icon and place it in the user’s system tray notification area. - Setting is written to the athena.conf file. - User can view and click on Athena system tray icon to show/hide Athena windows or quickly exit the program
User clicks a check box labeled “Allow ESC Key to Close a Tab” and clicks “Save” [General]	- The program will enable a listener for the escape key in the conversation area of the communication interface. - When escape (ESC) key is pressed, the currently focused conversation will be destroyed and the tab will be removed from the interface window. - Setting is written to athena.conf file.
User clicks a check box labeled “Enable Spell Check” and clicks “Save” [General]	- The external spell check library will be activated on all text entry fields of the conversation area. - Setting is written to athena.conf file. - User will be visually notified of any misspelled words in the text field of a particular conversation area.
User clicks a check box labeled “Enable Notifications” and clicks “Save” [Notifications]	- Each tab in conversation area will be modified to show a visual notification when a new message is received in that particular tab. - Setting is written to athena.conf file. - User will see a visual notification when new messages are received in conversation tabs that do not have active focus.
User clicks a check box labeled “Enable Sounds” and clicks “Save” [Notifications]	- Sound files will be loaded and the program will begin playing sounds when a new message is sent or received in a conversation tab. - Sound notifications will also play when a contact connects or disconnects from the server and changes status in the user’s contact list - Setting is written to athena.conf file.
User clicks a button labeled “Generate” under the “Generate New Encryption Key Pair” heading [Encryption]	- The program will contact the server and begin the key pair generation process. - User will be prompted to re-enter account password for security purposes. - Upon verification, program generates a new encryption key pair. - New public key is encrypted with old private key and sent to the server to replace the old public key in the database. - New private key will replace old private key on local machine. - User will see a notification of success when process completes, or an error message otherwise. - No settings written to athena.conf file

Function Input [Associated Module]	Function Description
User selects a new font style, font color, or font size in corresponding fields and clicks “Save” [Formatting]	- After settings are chosen, any active text areas in the conversation area of the user’s communication interface are modified to display the new font settings. - Settings are written to athena.conf file. - User will now see new font settings in all active text areas.
User selects a new theme and clicks “Save” [Theme]	- Setting is written to athena.conf file. - New theme style will not be activated immediately – style will be loaded next time user restarts application - User will be notified that application must be restarted for changes to take effect. If no restart is completed, current theme will remain until application is restarted. - Upon restart, new theme look and feel will be loaded to all interface components and subsystems.

4.5.3 Subordinates, Modules and Dependencies

Subordinates and Modules	Dependencies
- Preference component will interface with the GNU-Crypto encryption libraries when executing Encryption-related setting changes. - Java Swing libraries will be used to render icon images, check boxes, and various labels between setting tabs.	- The GNU-Crypto encryption library module must be installed with application for any encryption key pair processes to function correctly. - Communication interface menu must be initialized and accessible in order to access the preference interface. This interface is accessed from the Edit drop-down submenu in the communication interface menu.
SRS Requirements Satisfied	
Text Font Style || Text Font Color || Text Font Size Generate New Key Pair	

4.5.4 Processing

Process	Description	Pseudocode
Show System Tray Icon	When the user chooses this setting, the icon will be created and added to the user’s system tray. The system tray icon will have application functions to show, hide, or exit the application.	WritePreferenceChangesToFile(); CreateTrayIcon(); AddIconToTray();

Set ESC key to Close Tab	When the user chooses this setting, an action listener will be created in the user’s conversation area to listen for the ESC key. Whenever the key is pressed, the current conversation tab in focus will be removed from the interface and all data contained will be destroyed.	WritePreferenceChangesToFile();
		if (EscapePressed()) {
		GetFocusedTab();
		DestroyFocusedTab();
		}

Process	Description	Pseudocode

Enable Spell Check	When the user chooses this setting, the external spell check library will be activated for each text field within active conversation tabs. Any incorrect spelling will be located and user will be visually notified in text field area.	WritePreferenceChangesToFile();
		If (SpellingError) {
		  EditTextToShowError();
		}

Enable Visual Message Notification	When the user chooses this setting, sections of code will be activated to provide visual notification of new message activity in a particular conversation tab that the user has not yet acknowledged.
	If a message is received in a tab, the header of that tab will be changed to boldface and an asterisk symbol will be placed next to the header text.
		WritePreferenceChangesToFile();
		if(MessageReceived()) {
		  GetTabOfMessage();
		  BoldTabHeader();
		}

Enable Audio Message Notification	When the user chooses this setting, standard sound files will be loaded and sections of code are activated to play sound files when a contact changes connection status on a user's contact list, or when the user sends or receives a message in the conversation area.	WritePreferenceChangesToFile();
		if(MessageReceived() OR MessageSent() OR ContactConnect() OR ContactDisconnect()) {
		  PlaySound();
		}

Generate New Encryption Key Pair	When the user chooses this setting, account password verification will take place first for security and privacy purposes.
	When account is verified successfully, a new key pair will be generated using GNU-Crypto libraries. The new public key will be encrypted and sent to the server to replace the old public key in the database. The new private key is stored locally – no private keys are stored in database or sent to server.
		VerifyPassword();
		if(WrongPassword()) {
		  Exit();
		}
		GenerateKeys();
		EncryptPublicKey();
		SendKeyToServer();
		StorePrivateKey();

Font Style, Color, and Size Formatting	When the user chooses to change any of these settings, the font in all active text areas of the conversation area will be changed and user will be able to view conversation data with new font settings applied.	WritePreferenceChangesToFile(); UpdateFontFace(); UpdateFontSize(); UpdateFontStyle();

Change Athena Theme, also known as "Skin"	User will be able to select from a small selection of application themes.
	When the user changes this setting, the selected theme will be loaded to the program initialization process so that new theme will be activated upon restarting application.
	User will be prompted to restart application – if user confirms, application will restart and update immediately; otherwise, old theme will remain until application is restarted.
		WritePreferenceChangesToFile();
		PromptToRestartProgram();
		If (RestartNow()) {
		  Restart and ApplyNewTheme();
		} Else {
		  ApplyThemeOnNextRestart();
		}

4.5.5 Resources and Data

Resources	Data
- The preference component will use a negligible amount of CPU time when rendering new windows, applying settings updates, or generating new encryption keys. - Small amounts of RAM will be allocated to these different processes as they execute as well. - Keyboard and mouse input will be required to switch between preference interface sections and enter data when necessary. - A low-level output channel will be used to write preference data settings to the appropriate configuration file. - Java Swing API will be used to render all icons and interface components.	- The preference component will read and write data from the dedicated settings file in the appropriate application directory. - Configuration file is titled *athena.conf.* - GNU-Crypto libraries will be used when generating a new encryption key pair.

For more information regarding the detailed functional specifications of any modules or requirements listed above, please reference Section 3.2 of the Athena Software Requirements Specification document. This document will be available on the Athena product web site for viewing and download.

5 Interface Design

Detailed descriptions and screenshots of Athena's User Interface are illustrated in this section.

For more information on any of the user interface components below, please see Section 4 of this document.

Figure 5.1 Login Window

The login window will be presented to users when the application is started. Users will input an account user name and password and after verification will be connected to the server. Once users are connected, the communication interface will be displayed and accessible.

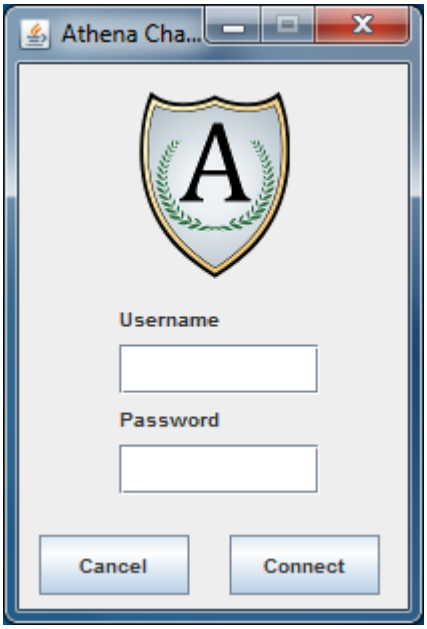

Figure 5.1

Figure 5.2 Communication Interface Window

This figure illustrates the default communication interface window after connecting to the server. There will be a single conversation tab in the communications area and two contacts currently connected. Each tab will be clearly labeled with the contact associated with that particular conversation. The contact list and conversation will automatically update when a contact connects or disconnects, or a message is sent or received.

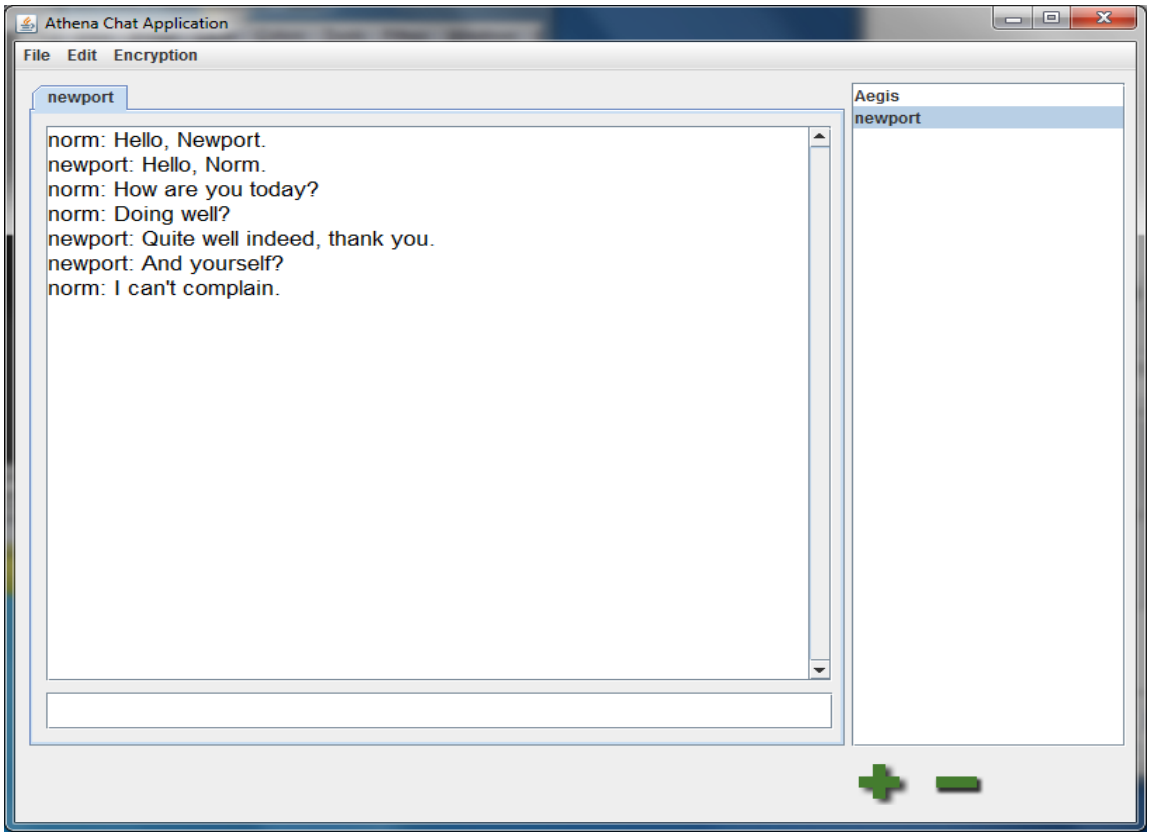

Figure 5.2

Figure 5.3 User Registration Window

The user registration screen a new user will need to access and fill out before they are able to log in. Users will provide the corresponding information to create the account. The encryption key pair is created "behind the scenes," transparent to the user. The account user name and public key are added to the database. Users are then able to login.

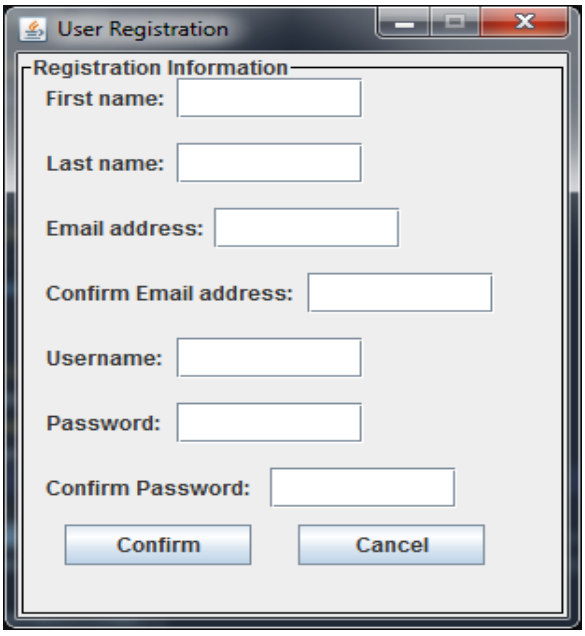

Figure 5.3

Figure 5.4 Communication Interface Menu

The "File" submenu will allow users to disconnect from the server, reconnect once they've been disconnected, and exit the program.

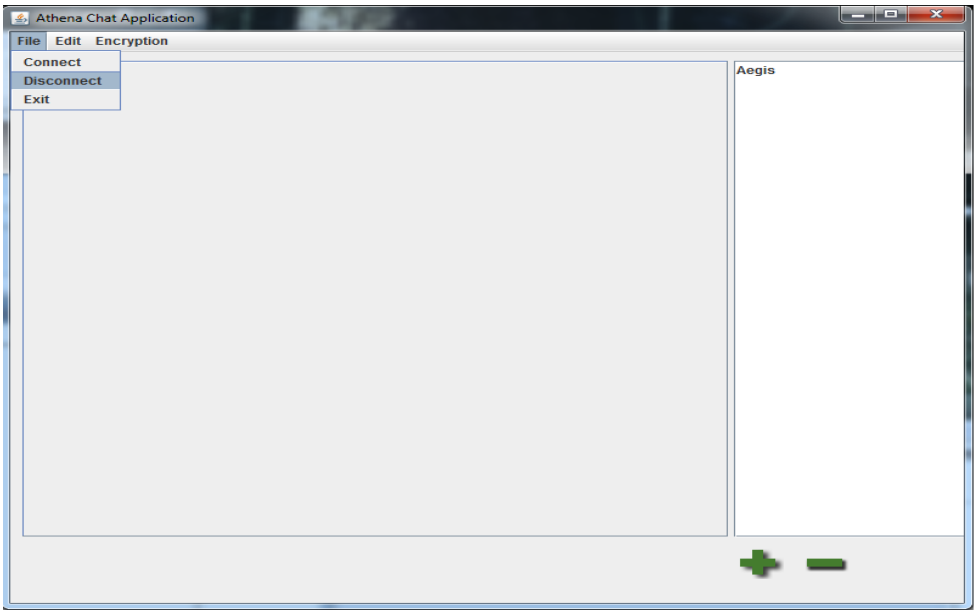

Figure 5.4

Figure 5.5 Communication Interface Menu Preference Access

The "Edit" submenu - users use this submenu to access the "Preferences" window.

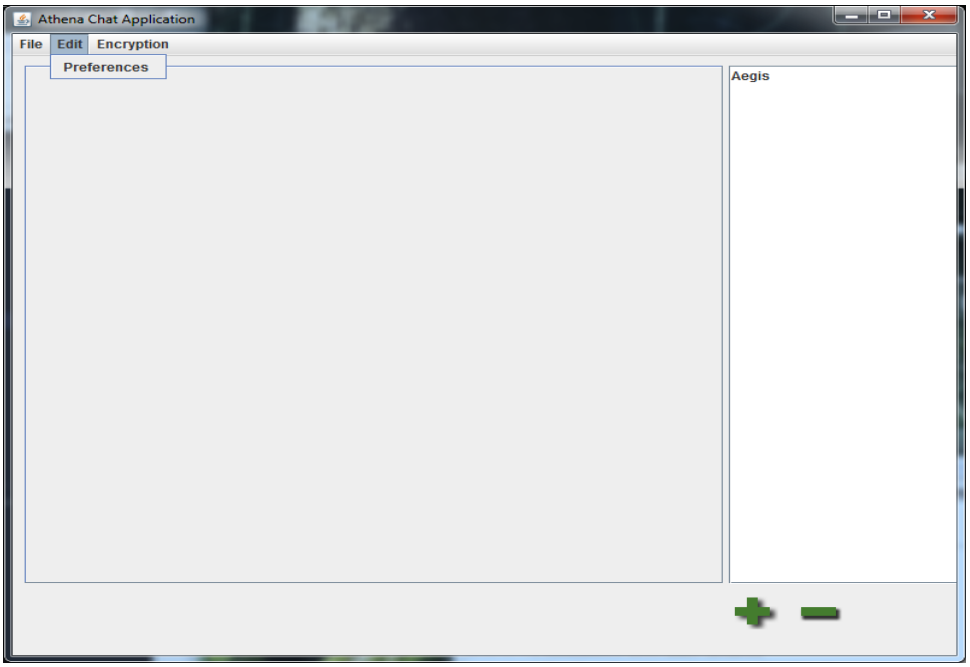

Figure 5.5

Figure 5.6 Preference Interface Window

The "General Options" area of the preferences window will allow the user to change key options for their experience. They can choose options such as whether or not to show an icon in the notification area (system tray), use escape to close a conversation tab, or enable spell check in the conversation area.

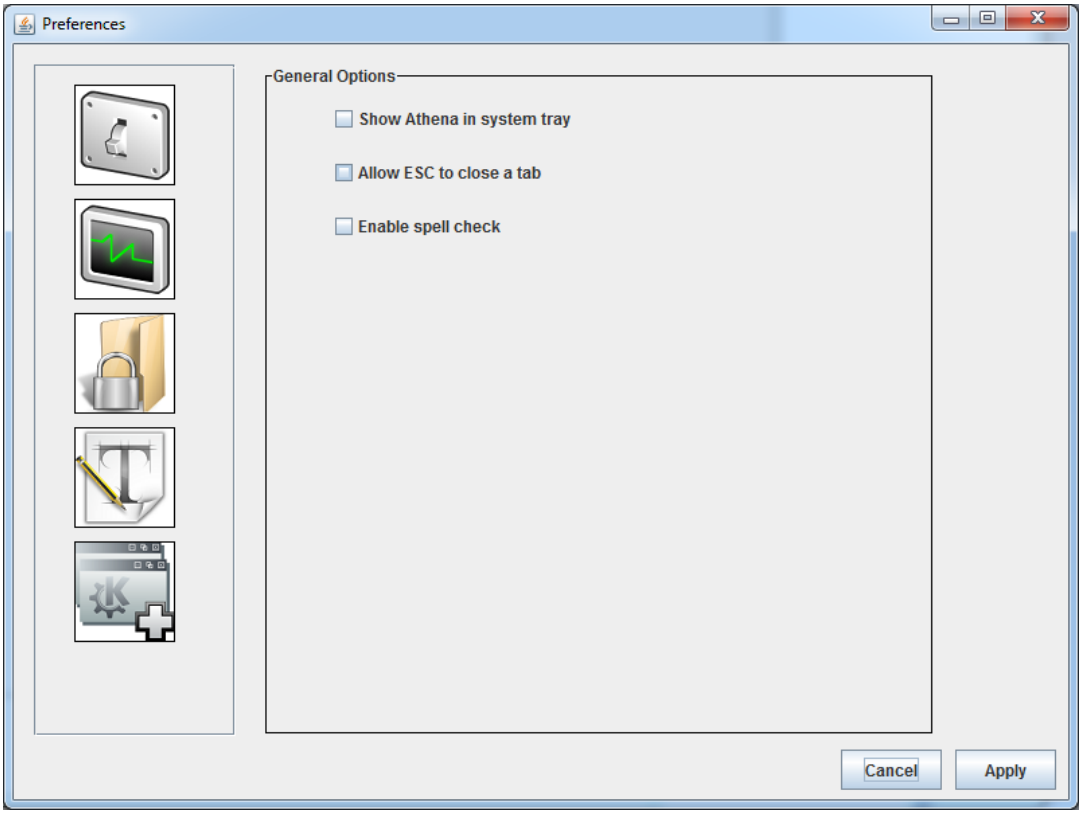

Figure 5.6

For more user interface screenshots and corresponding explanations, please refer to Appendix C.

Appendix A

Name	***Formal Definition***
Advanced Encryption Standard (AES)	A method of encryption using symmetric keys. The same key is used to encrypt and decrypt the target message. This method is faster than RSA, but an agreement on a symmetric key must first be established.
Application	Computer software designed to help the user to perform a single or multiple related specific tasks. Such programs are also called software applications, applications or apps.
Application Programming Interface (API)	This is a public "interface" that allows software to interact with other software and users.
Central Processing Unit (CPU)	The "brain" of the computer that executes the program's instructions, and performs all of the calculations for the encryption algorithms.
Client-Server Architecture	A common program architecture in which a user or users run a small "client" program which interacts with a main "server" to exchange information. The Internet is an example of client-server architecture.
Compiler	A software application used to convert source code into machine-language executable programs to be run by a computer.
Cross-platform	The ability for a program to run on different operating systems or computer types. For example, a cross-platform program is able to run on both Windows and Macintosh computers.
Diffie-Hellman	A method of generating one-time-use encryption keys (e.g. AES) secretively. Through the use of large prime numbers, users are able to agree upon and separately generate the same key without ever communicating over an unsecure connection.

Name	***Formal Definition***
Encryption	A method of hiding or obscuring information, generally using a mathematical algorithm. This information can only be recovered using a secret password or key.
File Transfer	A way of moving or copying a file from one location to another.
GNU	Stands for “GNU’s Not Unix.” GNU is a branding given to any number of free, open source software projects of the GNU Project.
GNU-Crypto	The GNU Cryptography Libraries. External Java classes used by Athena for encryption, decryption, and RSA key pair generation.
Hashing	A "one-way" algorithm. This is similar to encryption, but slightly distinctive in nature. A hashing algorithm will generate a unique string of characters of any data passed to it. These algorithms are designed to never generate the same string from different data, making it perfect for securely storing and verifying passwords.
Java	An object-oriented programming language that allows programmers to create applications on any operating system. Java uses a "Virtual Machine" to run its programs.
Java Virtual Machine (JVM)	Uses the java instructions generated by a Java compiler and translates them into machine code for execution. This makes java programs inherently cross-platform, because any java program will run on any computer capable of running the JVM.
JDBC	The Java Database Connectivity Libraries. External Java classes used by Aegis server to connect to the authentication and public key database.
Local-area Network (LAN)	A group of computers in a small physical area that are connected one another. LANs are smaller in scope than the Internet; the Internet is made up of a large group of interconnected LANs.
Machine Language	Instructions a program executes at a very low level that allows the application to run.

Name	***Formal Definition***
Man-in-the-middle Attack (MITM)	A common vulnerability in communications infrastructures in which an attacker can capture messages from one endpoint, and, after capturing the information, transparently relay the information to the intended recipient.
MySQL	A popular branch of the Structured Query Language and a Relational Database server. The Aegis database runs on the MySQL language.
Operating System (OS)	An interface between hardware and user that is responsible for the management and coordination of activities and the sharing of the resources of a computer. The OS acts as a host for computing applications that run on the machine.
Plaintext	A message or text that is unencrypted and in human-readable form.
Random-access Memory (RAM)	Temporary cache memory commonly referred to as virtual memory. When computer programs are run or files are opened, they are loaded into the RAM for later access when it is needed.
Relational Database	A collection of tables used to organize specific information. A company may use a relational database to keep information on their employees and customers.
Rivest Shamir Adleman (RSA)	A method of encryption using a key pair comprised of a public and a private key. A message encrypted with the public key can only be decrypted with the private key, and vice-versa. This ensures confidentiality and represents a digital signature. The acronym references the names of the method's original designers.
Specific Hashing Algorithm 256 (SHA-256)	Part of the "SHA-2" family of hash algorithms. It creates a hash of a message that is 256 bits long. A bit is the basic unit of information in computing (e.g. A one or a zero).

Appendix B

Figure B.1 Use Case

Use case for a simple session of Athena - this diagram shows a user logging on, encountering an error, the Athena Administrator fixing the error, then completing the session and finally logging off.

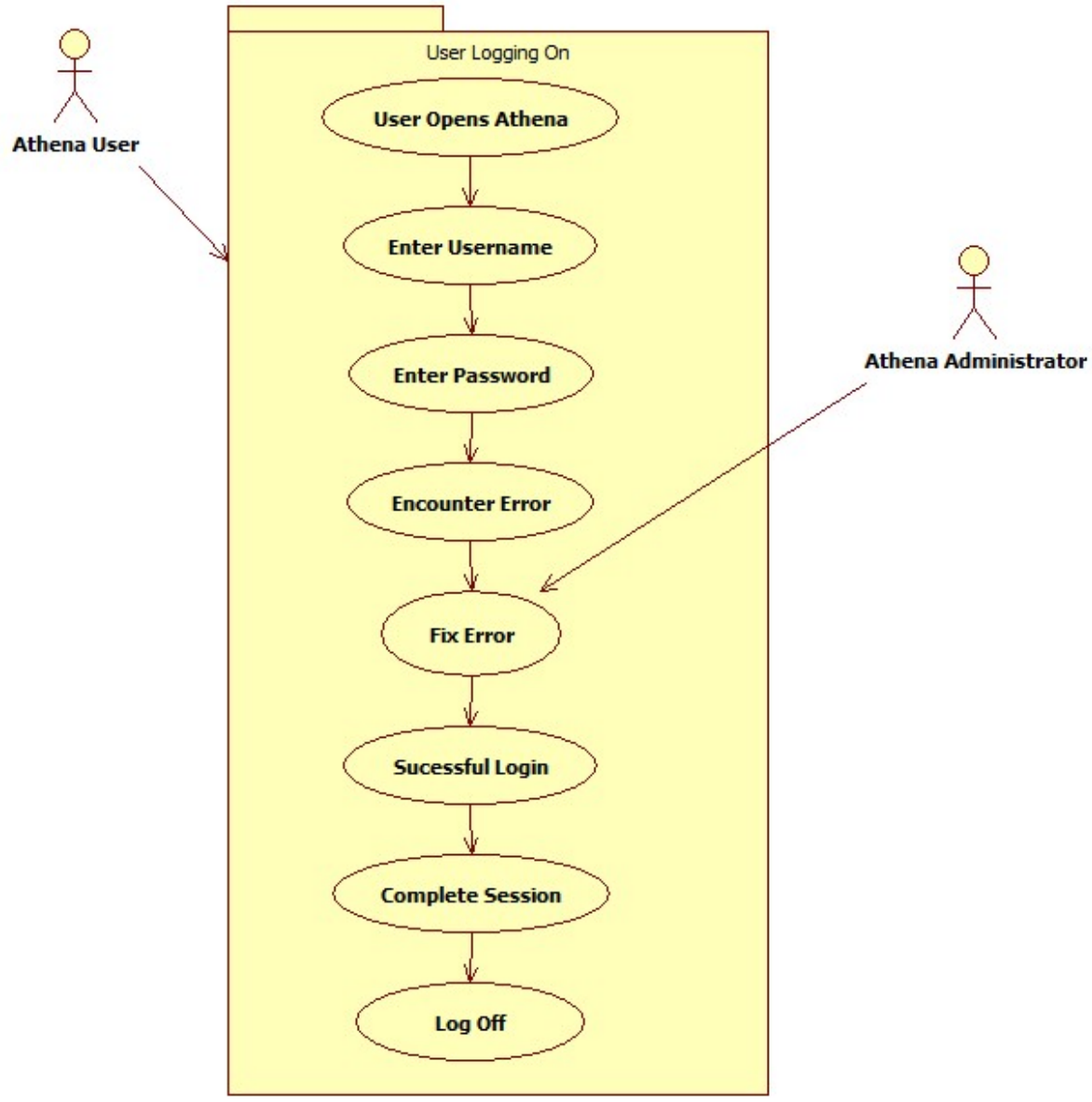

Figure B.1

Figure B.2 Use Case

Use case showing a user communicating to another user through the communication interface. The user types a message and sends it to the other user. They decide to add another contact to the contact list. Once the recipient of the message receives it and replies, the message is shown in the user's contact list. The user then encounters an error in which case the Athena Administrator corrects the problem.

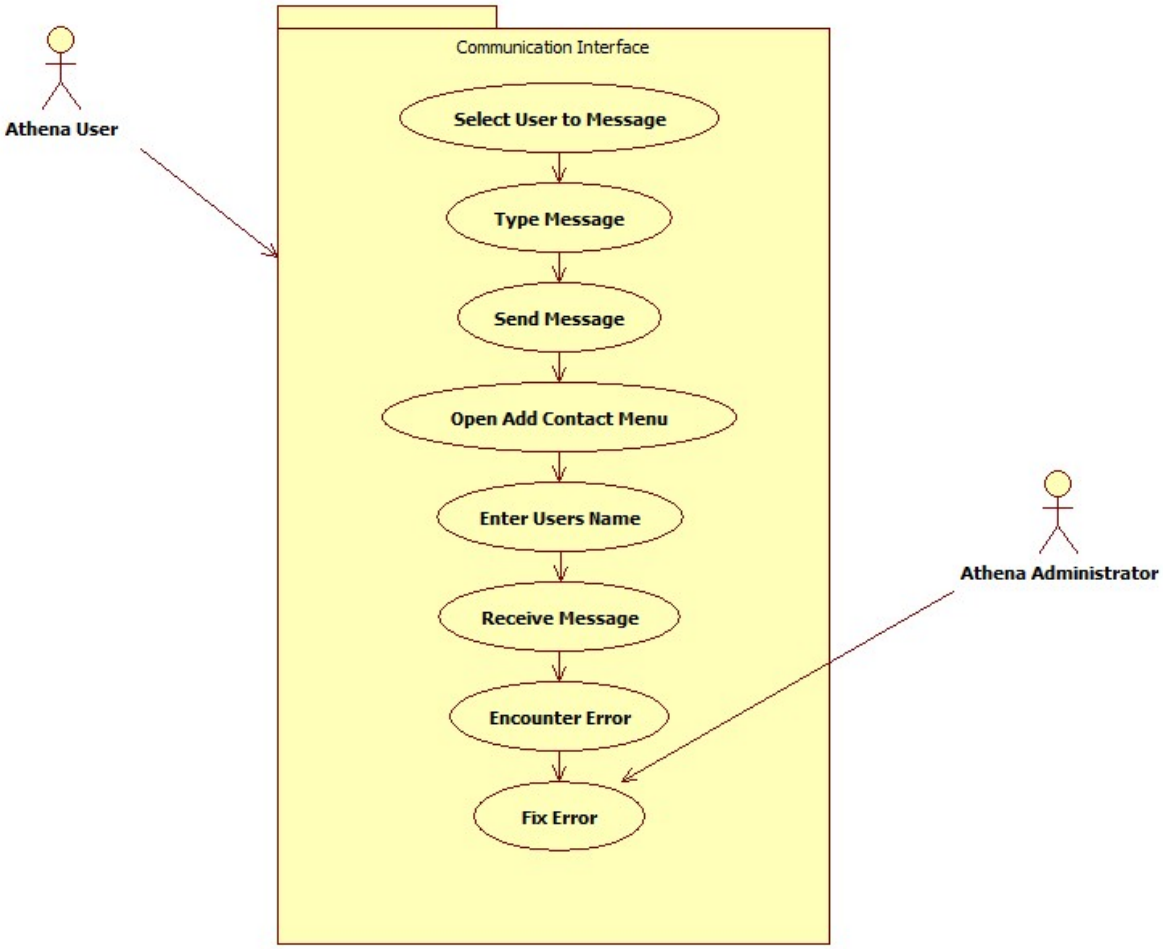

Figure B.2

Figure B.3 Use Case

Use case for changing preferences of Athena. The user opens the preferences window and changes a simple preference. The user encounters an error in which case the Athena Administrator fixes it. Once this occurs the preference window is closed.

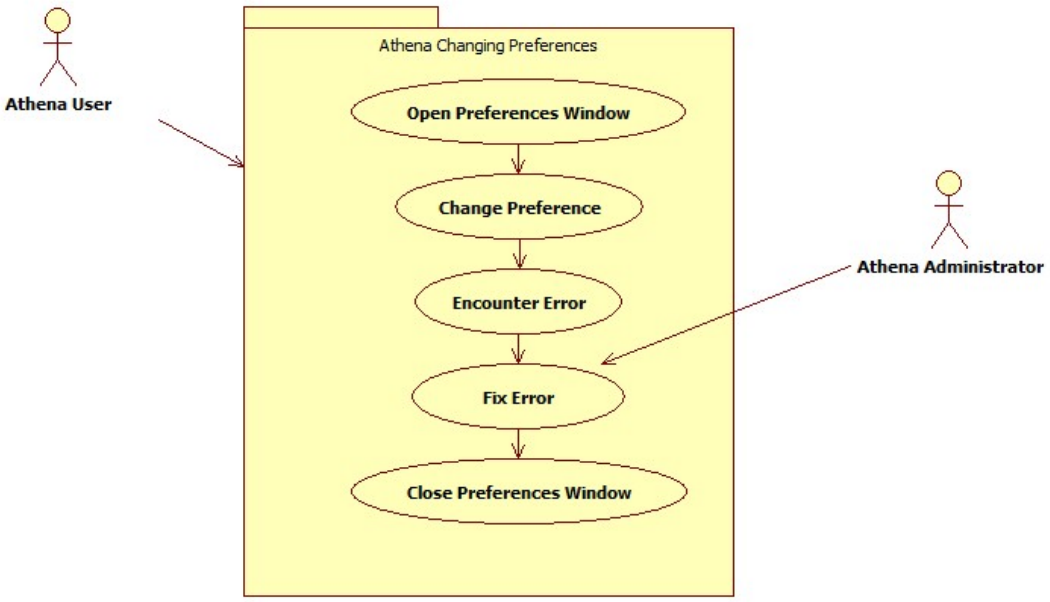

Figure B.3

Figure B.4 Use Case

Use case for a user creating a new user name. The user enters all required information, and may encounter an error, in which case the Athena Administrator fixes the error. The information is correctly sent and registered.

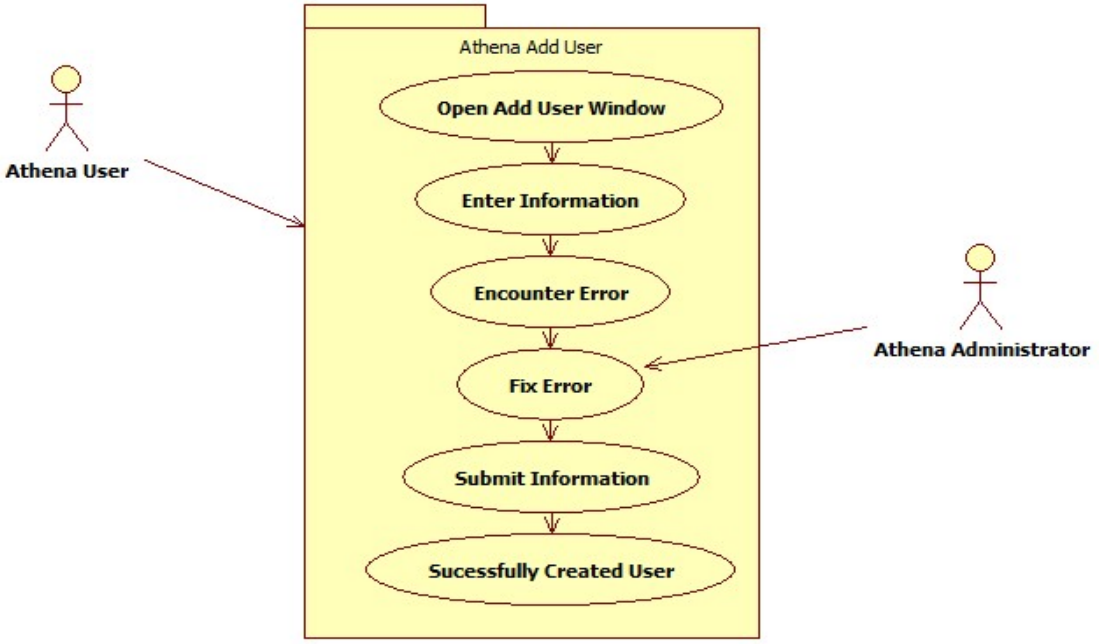

Figure B.4

Figure B.5 Class Diagram

The following diagram is a class diagram for the Athena software. All classes, modules, and associated data are listed for each class.

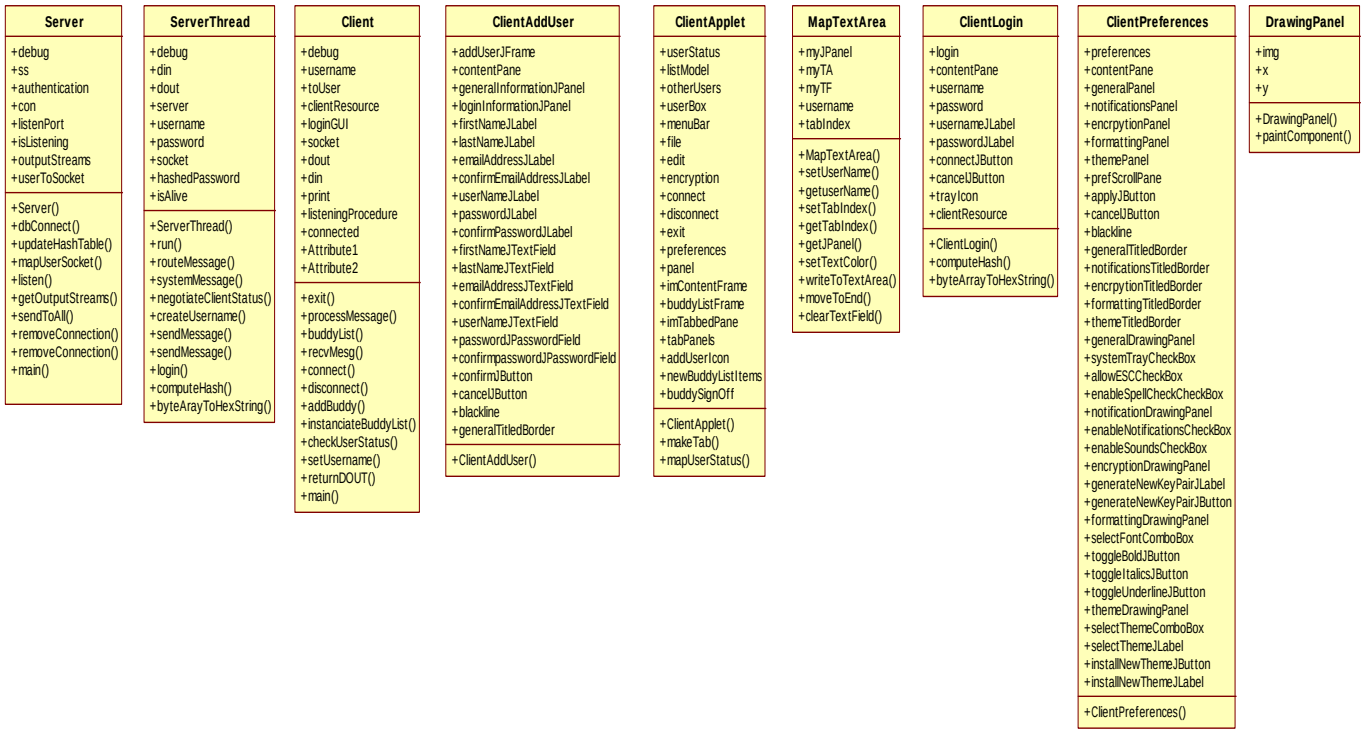

Figure B.5

Appendix C

Figure C.1 Notification Options

The "Notification Options" area of the Preferences window will allow users to toggle audio and visual notifications of events. Events include: New message, message sent, contact connects, contact disconnects.

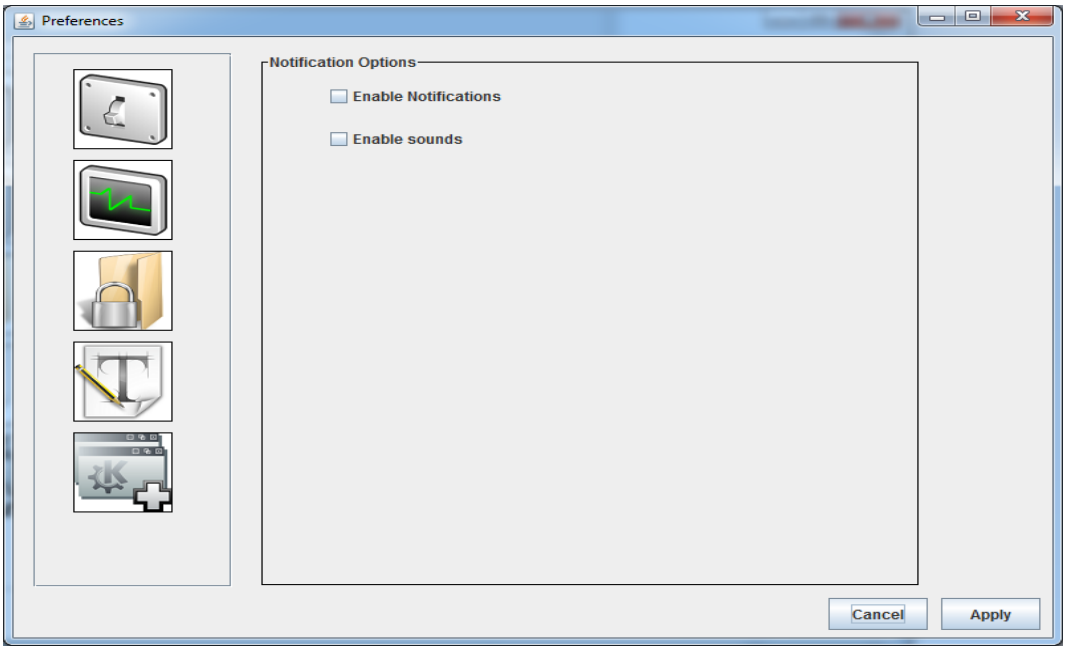

Figure C.1

Figure C.2 Formatting Options

The "Formatting Options" area will allow users to customize the fonts displayed in the communications area. Users can customize the font face, font size, and font style to personalized preference.

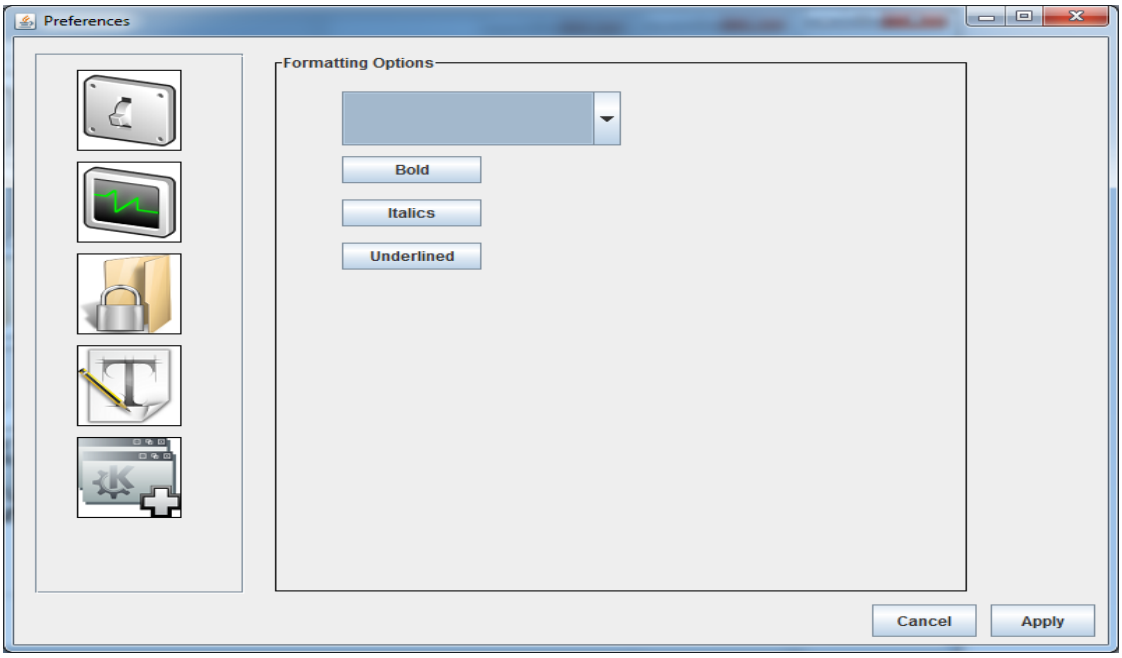

Figure C.2

Figure C.3 Encryption Options

The “Encryptions Options” in the preference window will allow users to generate a new encryption pair at any time. This is useful if a private key has been compromised.

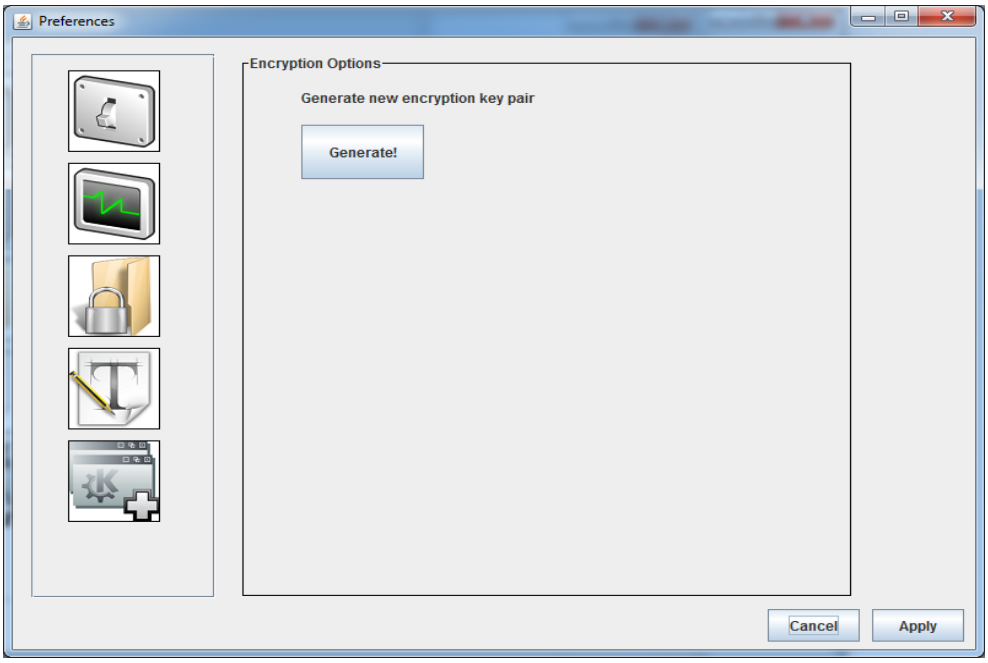

Figure C.3

Figure C.4 Theme Options

The "Theme Options" area of the preferences window will allow users to install and select custom themes that change the look and feel of Athena.

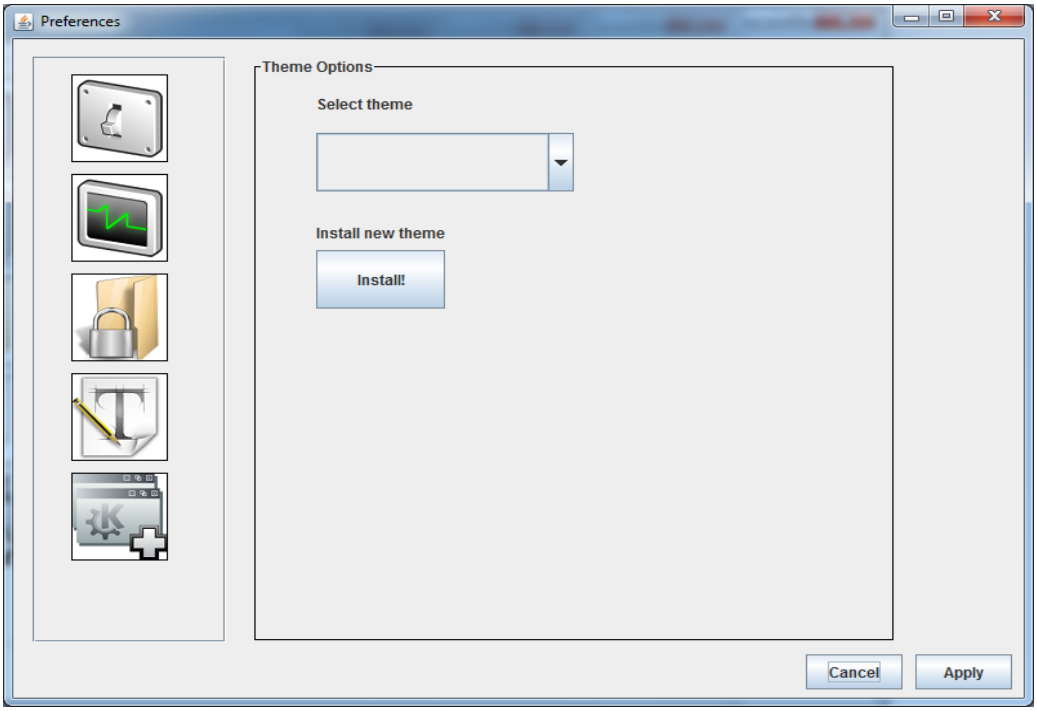
Preferences
Theme Options
Select theme
Install new theme
Install!
Cancel
Apply

Figure C.4