MC Chat

Haki S., Gary L., Thomas S., Louie C.

**Software Requirements Specification**

**Document**

**Version: (1.0)** **Date: 01/22/2024**

**Table of Contents**

**1.  Introduction**

*1.1  Purpose*

**2.  System Overview**

*2.1  System Users*

*2.2  Operational Constraints*

**3.  User Requirements**

*3.1  Functional User Requirements*

*3.2  Non-Functional User Requirements*

**4.  System Requirements**

*4.1 Functional System requirements*

*4.2 Non-Functional System Requirements*

**5.  Diagrams**

*5.1 Use Case Diagrams*

*5.2 Sequence Diagram*

*5.3 Mock Diagram*

**1.  Introduction**

This document is intended to provide an understanding for the specifications, design and implementation of the MC Chat mobile application, an instant messaging application which will allow Manhattan College students to connect with each other and faculty via text and multimedia messages. Different features will be available, such as group chats, friend request systems, AI-assisted responses etc. As for now, this software product will be developed as a school project so there are no clients besides the developers themselves however it is made with the goal in mind to have this application acquired by the Manhattan College institute in the future.

**1.1 Purpose**

The purpose of this mobile application is to make communications among students and/or faculty easier and more efficient. Examining the limitations of using traditional emails for communications, this application aims to offer a real-time platform for more efficient and organized communications between students and faculty members.

**2. System Overview**

The key features of the system are:

* Users will be required to log in or register for a new account with their Manhattan College email address before using the system.
* Upon logging in, users will be presented with the main window of the application.
* Along the top of the main window will be a bar with options. Users will have the ability to change between individual chats / group chats and chat rooms, change their online status, open their profile, open application settings, or view incoming friend requests.
* Within the top bar of the main window, new options will appear depending on whether the user has individual chats or group chats / chat rooms selected. By selecting individual chats, the user can find/add other users, start new chats, and see incoming friend requests. By selecting group chats/chat rooms, the user can find/add/create group chats/chat rooms and see incoming group chat requests.
* The left side of the main window will be a list of ongoing chats. Individual chats or group chats/chat rooms will be displayed depending on what type of chat the user has selected to view.
* The right side of the main window will be a chat window which will be populated based on the chat the user currently has selected. There will be the ability to send text messages, voice messages, or start a voice/video chat.

**2.1  System Users**

The target user base for this application will be all active Manhattan College students and faculty. The majority of them however will be the students rather than faculty, so mostly young adults and teenagers. Either case all users can be categorized into two categories:

* Faculty: Manhattan College professors, adjuncts or other teaching staff looking to have a communications medium with the students they teach. Sample reasons for use could be: making class announcements, answering lecture related questions, sharing notes or other useful resources etc.
* Students: People enrolled in Manhattan College classes looking to communicate with other peers or professors. Sample reasons for use could be: Staying updated with professors about class announcements, exchanging lecture notes, casual chatting with peers etc.

**2.2  Operational Constraints**

The following constraints will apply to the software product:

* The system will be available as a mobile application so the users will need two things to access this product: working mobile device, and a working mobile connection.
* At the registration page, users will be required to have a unique Manhattan College email address which will be used for account verification.
* Users will be required to open their Manhattan College email inbox and verify the MC Chat account via an email the application sends to their address. Doing this ensures the completion of the registration process.
* The database used to keep track of all users and their chats will be hosted through FireBase services and it is the developers responsibility to run and maintain the database.

**3.  User Requirements**

This section describes the business needs for MC Chat that will be expected from the user. This will include any functional and non-functional user requirements.

**3.1  Functional User Requirements**

**Registration and Logging in**

* Users must be able to register using their Manhattan College email addresses and a valid password.
* Users must be able to login with their previously registered Manhattan College email address and password.
* Users must be able to select a *‘Forgot My Password’* option to reset their password.

**Profile Management**

* After registration, students can set up and edit their profiles, including Major, Graduation Year, etc.
* After registration, faculty can set up and edit their profiles, including Department, Title, Role, etc.
* Users will have the ability to enable or disable voice messages
* Users will have the ability to enable or disable live-video/voice chat.
* Users will have the ability to enable or disable a profanity filter.

**Messaging**

* Users will be able to send and receive private text and voice messages.
* Users will be able to privately voice/video chat.
* The system must support group chat creation with options for an admin to set permissions.
* Admin will be able to select other users to be administrators/moderators in the chat room.
* The system must support chat room creation with options for an admin to set permissions.

**User Directory Access**

* Users will have access to a user directory and the ability to send friend requests to other users.

**Friend Requests**

* The application will allow users to send, receive, and manage friend requests.

**Media Sharing**

* Users must be able to share images, videos, audio messages, files, and emojis.

**Content Moderation**

* Users will have the ability to block other users.
* Users will have the ability to report inappropriate text messages, voice messages, and/or media. After reporting a message and/or media, the message and/or media will be obscured for the reporting user.
* Messages and/or media that has been reported 3 times (from 3 different users) will automatically be obscured for all users and a system/room administrator will review the content.
* If the system administrator deems their messages and/or media to be in violation of the EULA, the user account responsible for said messages/media sharing will be locked for a period of time. The punishment for subsequent violations will be progressively harsher.
* Public chat rooms will automatically have the profanity filter enabled, without the option for disabling.
* Users will be unable to send text messages containing racial, sexual, or other slurs.

**AI Assistant for Messaging**

* An AI tool will be available to suggest generic AI-generated responses to messages.

**Notifications**

* The user may receive real-time notifications for messages, friend requests, and other relevant alerts.

**3.2  Non-Functional User Requirements**

**Usability**

* The interface will be user-friendly and intuitive, suitable for diverse user groups.

**Performance**

* The application will load quickly and handle real-time messaging without significant delays.

**Security**

* User data must be securely stored and transmitted.
* The application must comply with data protection laws and privacy standards.

**Data Integrity**

* The system must ensure the accuracy and integrity of data, especially in user profiles and messaging.

**4.  System Requirements**

This section will outline the technical specifications and constraints that the proposed system must adhere to in order to meet the user requirements mentioned above. These requirements include hardware, software, performance, security, and other technical aspects that the system needs to fulfill.

**4.1  Functional System Requirements**

**Registration and Logging In**

* The user will be prompted to either register or login.
* When attempting to register, the system will authenticate whether the user’s entered email address is a valid Manhattan College email address (checking for proper email format), the entered email address is not already registered, and their entered password meets industry-standard security requirements.
* Upon successfully entering valid credentials when registering, an email containing a link will be sent to the user’s entered email address. The user will be prompted to click on this link to complete the registration process. If the link is not clicked within 1 hour, the account will not be created and the user will need to restart the registration process.
* Upon successfully clicking the confirmation link, the user will complete the registration process, the account will be saved to the appropriate table in the database, and the user will be automatically logged into the system and be brought to the main window.
* When attempting to login, the system will authenticate whether the user’s entered email address is already registered and whether the user’s entered password is valid against what is stored for the specified user in the database.
* When attempting to login, the user will be presented with a *‘Forgot My Password’* option.
* Upon clicking the *‘Forgot My Password’* option, an email containing a link will be sent to the user’s email address. The user will be prompted to click on the link to continue the ‘Forgot my password’ process
* Upon successfully clicking the link, the user will be prompted to input a new password. The system will authenticate whether the user’s entered password meets industry-standard security requirements.
* Upon successfully entering valid credentials in the *‘Forgot My Password’* process, the user will be logged in and brought to the main window. The user’s password will be updated in the database to reflect the changed password for future logins.
* Upon successfully entering valid credentials when logging in, the user will be brought to the main window.
* Upon entering invalid credentials in either the registration, login, or *‘Forgot My Password’* process, the user will be shown a relevant error message.
* Upon entering invalid credentials in the login process 5 times in a row, the account associated with the entered email address will be locked for a period of 1 hour. This lock will be reflected in the database for the specified time.
* When the lock timer expires, the database will update to allow the user to attempt to log in again.
* If the user attempts to login while their account is locked, an appropriate error message will be displayed.

**Profile Management:**

* User accounts will have different user information fields: i.e; faculty will have fields such as department, title, role, etc. Student accounts will have fields such as major, minor, graduation date, etc.
* Users will be able to manage/update their profile information. When the user decides to change any information regarding their profile, the system will update the database to reflect the changes made, which will also update the UI to display the newly modified data.
* While logged in, the user will have an option to *‘Manage Profile’*
* When the user clicks on *‘Manage Profile’*, the system should switch the window that the user is viewing to a profile management window.
* This window will display all of the information for the profile, which is pulled from the database, and will allow the user to be able to make modifications to specified data fields (not including their name or email).
* From this page, users will have the ability to enable or disable the receiving of voice/video calls indefinitely or for a set period of time (1 hour, 2 hours, etc.)
* From this page, users will have the ability to enable or disable the receiving of voice messages.
* From this page, users will have the ability to enable or disable a profanity filter that will replace common inappropriate language in received text messages with asterisks in private/group chats.
* When the user makes any update to their profile and clicks *‘Submit’*, the system will update the appropriate fields in the database to reflect the changes to the user’s profile
* When the user makes any update to their profile, a *‘Cancel’* option will become available to undo any changes made.

**Messaging:**

**Individual/Group Messages:**

* From the main page, the UI will display multiple tabs, including a ‘*Messages*’ tab.
* When the user clicks this tab, the window will change to display all of the chats the user is currently a member of, as well as a button to *‘Create New Chat’.*
* If the user clicks a current chat, the chat thread will be displayed to the user with the chat history. The user will be able to send/receive messages from here.
* If the user clicks *‘Create New Chat’*, the UI will change to the create chat UI page with the necessary input fields
* The user will choose who to create a chat thread with from their current friends list or from the current *User Directory*. The user will be able to choose 1 or many different users to add to the chat.
* If the user that is selected is not in the current user’s friends list, the chat will also serve as a friend request for the specified user to either accept (opening a chat dialogue) or reject (denying the chat creation).
* Once the chat is created, the system will store the relevant information in the database and will keep the database updated as the chat progresses.
* Once a new chat has been opened, the User within the chat will have the ability to type a text message, record a voice message, attach multimedia, or start a voice/video call.

**Chat Rooms:**

* From the main page, the UI will display multiple tabs, including a ‘*Rooms*’ tab
* When the user clicks this tab, the UI will change to display all of the chat rooms that are currently created, as well as a button to *‘Create New Room’.*
* If the user clicks a currently created room, they will either enter the chat, or will request access from the admin of the room depending on how the room has been set up by the admin.
* If the user clicks the *‘Create New Room’* button, the UI will change to the create room UI, which will prompt the user to input the necessary information, such as who can access the room (public or request access), as well as who to initially add to the room (if anyone other than the user).
* Once the user finishes the room setup and clicks ‘*Submit*’, the new room will be created and saved in the database by the system, which will also update the list of rooms for all users.
* Once a new room has been opened, the users within the room will have the ability to type a text message, record a voice message, or attach multimedia.
* The system will handle updating the database as the chat progresses.
* In a room created by the user, the user will also have the option to *‘Modify*’ the room, either changing who can access the room, adding/removing members from the room, as well as deleting the room. These changes will be reflected within the database

**User Directory Access**

* When the user clicks the user directory button, a popup window will appear. The popup window will include a search bar/search filter options on top and a list of users below it.
* Within the popup window, the system will search for and populate the user list with all users within the database.
* The system will narrow the users populated within the user list based on user specified search filters.

**Friend Requests**

* Within the User Directory Access, the user will have the ability to send friend requests to specified users.
* Upon sending a friend request to a specified user, the system will send a notification to the specified user informing them of the friend request. The specified user will have the ability to accept or reject the friend request.
* Upon acceptance of the friend request, the system will add the specified user to the user’s friends list within the database. The user will then be able to start a chat with the specified user.
* Upon rejecting the friend request, the system will delete the friend request notification for the specified user. The user will be able to send the specified user a friend request again.

**Media Sharing**

* In any message thread, the user will have an option to share different forms of media (files, videos, audio, emojis, etc.) by clicking the ‘*Add Attachment(s)’* button.
* When the user clicks this button,  a pop up will appear allowing them to choose the file(s) from their device to share.
* Once the user selects the file(s) to share and clicks ‘*Send’*, the file(s) will be stored in the database and the chat will be updated to reflect the attachment(s).

**Content Moderation**

* Within the user’s friend’s list or an existing individual/group chat and/or chat room, users will be able to select a *‘Block’* button for each user that will remove the selected user from the user’s friend’s list, deny the selected user from being able to message the user, deny the selected user from being able to send a friend request to the user, and remove the selected user’s messages and/or media from shared individual/group chats and/or chat rooms from the user’s view.
* Within an individual/group chat and/or chat room, the user will be able to select a *‘Report’* button for each message and/or media. The user will also be able to select the *‘Report’* button again to un-report the message and/or media. A reported message and/or media will be visually obscured for the user.
* Within a group chat and/or chat room, if a message and/or media is reported 3 times by other users, it will automatically be visually obscured for all users within the group chat and/or chat room.
* Users that have had their messages and/or media reported 3 times within individual/group chats by separate users will have their account flagged for review by a system administrator or chat room administrator, depending on where the reported messages and/or were sent.
* Users will receive progressively harsher punishments for each occurrence of their account being flagged and being found in violation of the EULA, ranging from a 1 hour account suspension to a permanent account suspension.
* Chat rooms will automatically have a profanity filter enabled, with common inappropriate language in text messages being replaced by asterisks.
* All text messages before being sent will be automatically checked for common racial, sexual, or other slurs. Text messages containing such content will be unable to be sent and the user will be informed that their message contains explicit content that is unable to be shared on the platform.

**AI Assistant for Messaging**

* Provide a button next to the text box where the user would type in their message in a chat. This button’s functionality will be to provide an AI-generated and relevant response to whoever the user is chatting with.
* This functionality will be achieved by using the OpenAI API which allows an application to communicate with their LLM model known as ChatGPT.
* Depending on the purchased plan from the developed team different models of their LLM can be chosen. Specifically for this project the intended model to be used is GPT-4-turbo which is currently the fastest and most efficient model that OpenAI has created.

**Notifications**

* Upon receiving a friend request, the system will display a notification for the user that they have a pending friend request.
* Upon having sent a friend request and having it accepted, the system will display a notification for the user that their friend request has been accepted.
* Upon receiving a message from a friend/group chat while not actively focused on the chat window, the system will display a notification for the user that they have an unread message.
* Upon having a message and/or media reported 3 times, the system will display a notification for the user that their content is being reviewed by a system/room administrator.
* Upon having their account suspended, the system will display a notification for the user that their account has been suspended for the specified amount of time.

**4.2  Non-Functional System Requirements**

**Usability**

* User friendly user interface which will be built with ReactJS. All application components will be easily accessible by the user and visibly distinguished from each other.
* Use up to date front-end design practices. Such practices include having easily readable fonts, large enough font sizes, color palettes with adequate contrast levels between any text, background and icon components.
* Smooth transitions between different pages with an intuitive design. Have a main menu that is visible at all times and does not take too much space on the screen. Chat section should be big enough to allow the user to comfortably read and write their message.

**Performance**

* Use lazy loading for deferred loading of non-essential components, and employ caching for improved user experience.
* Implement code splitting techniques for faster initial load times.
* Reduce network requests by optimizing assets and prioritize rendering efficiency.
* Compress CSS, JavaScript, and images to decrease their size and enable caching for static assets, thereby reducing server load and enhancing loading times.

**Security**

* FireBase will be used to store user data and chat logs. FireBase has support for using transport encryption between the client and the nodes, and between the nodes in the cluster. Encrypting traffic ensures that no one can “sniff” sensitive data on the network.
* Comply with data protection and privacy laws by:

1. Limiting data collection to what is absolutely necessary for the app's functionality. Such data includes the user's first name, last name, school email, profile pictures, friend lists. All the other data such as passwords and chat logs will be encrypted at storage and non-accessible by anyone else.
2. Allow users to erase all records of their data, therefore allow the user to delete their account and all information related to their account.
3. Notifying all users in case of a data breach as required by consumer data privacy laws.

**Data Integrity**

* Set up automatic database backups to ensure data integrity and reduce data loss.
* Minimize human error by ensuring user accounts are created with existing and valid email addresses.
* Minimize database access to a small part of the development team.
* As part of normal operation, FireBase maintains a running log of events, including entries such as incoming connections, commands run, and issues encountered. Generally, log messages are useful for diagnosing issues, monitoring your deployment, and tuning performance.

**5. Diagrams**

This section will contain diagrams outlining and detailing the use, interactions, and components of the system.

**5.1 Use Case Diagrams**

**Overall Use Case Diagram**

**Registration Use CaseA diagram of a website

Description automatically generated**

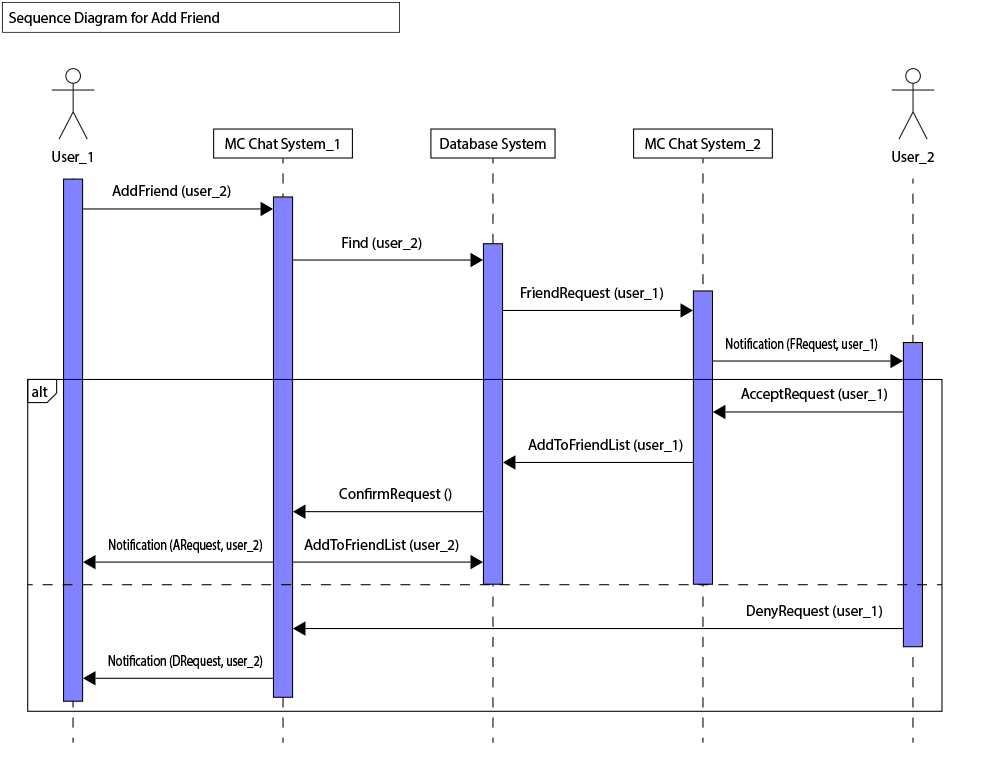
A diagram of a company

Description automatically generated

**A diagram of a login system

Description automatically generatedA diagram of a software system

Description automatically generated5.2 Sequence Diagrams**



**A screenshot of a login box

Description automatically generatedA black and white cover with white text

Description automatically generatedA green and white logo

Description automatically generated5.3 Mock Diagrams**

A screenshot of a chat

Description automatically generatedA screenshot of a chat

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated