Omni

System Design Document Team 6

Table of Contents

Introduction

Purpose of This Document

This document outlines the design for the Omni chat system. As a refresher, Omni is a decentralized chat platform; the frontend and web server are together known as “Omnichat” and are not a core part of the Omni system, but are provided as a reference for others who may want to build an Omni chat interface.

References

Omni System Requirements Specification Document

- Havens, Micah, et al. “System Requirements Specification.” 20 Oct. 2022, https://github.com/Skid-Team-6/Omnidocs/blob/main/Omni%20System%20Requirements%20Specification.pdf

Omni Project Proposal

- Havens, Micah, et al. “Project Proposal.” 12 Oct. 2022, https://github.com/Skid-Team-6/Omnidocs/blob/main/Omni%20Proposal.pdf

System Architecture

Architectural Design

When the user’s web browser loads the URL of our application, Express.js, our web server of choice, will respond with the page requested. It also notifies the session manager in case of, say, a POST message containing a request to create a new account or to log in. The session manager interfaces with a database that tracks user accounts. It is **up to the web server** to keep track of user accounts - though the web server must communicate with the Omni server when a new account has been created or a user logs in, so that Omni can keep track of which user sent what message, ban/timeout functionality, et cetera. Once the user is on the chat page, Socket.IO will notify the session manager of a new socket connection. The session manager will interface with Socket.IO to communicate with the user. If the user sends a message, for example, Socket.IO will notify the session manager, which then notifies the Omni server.

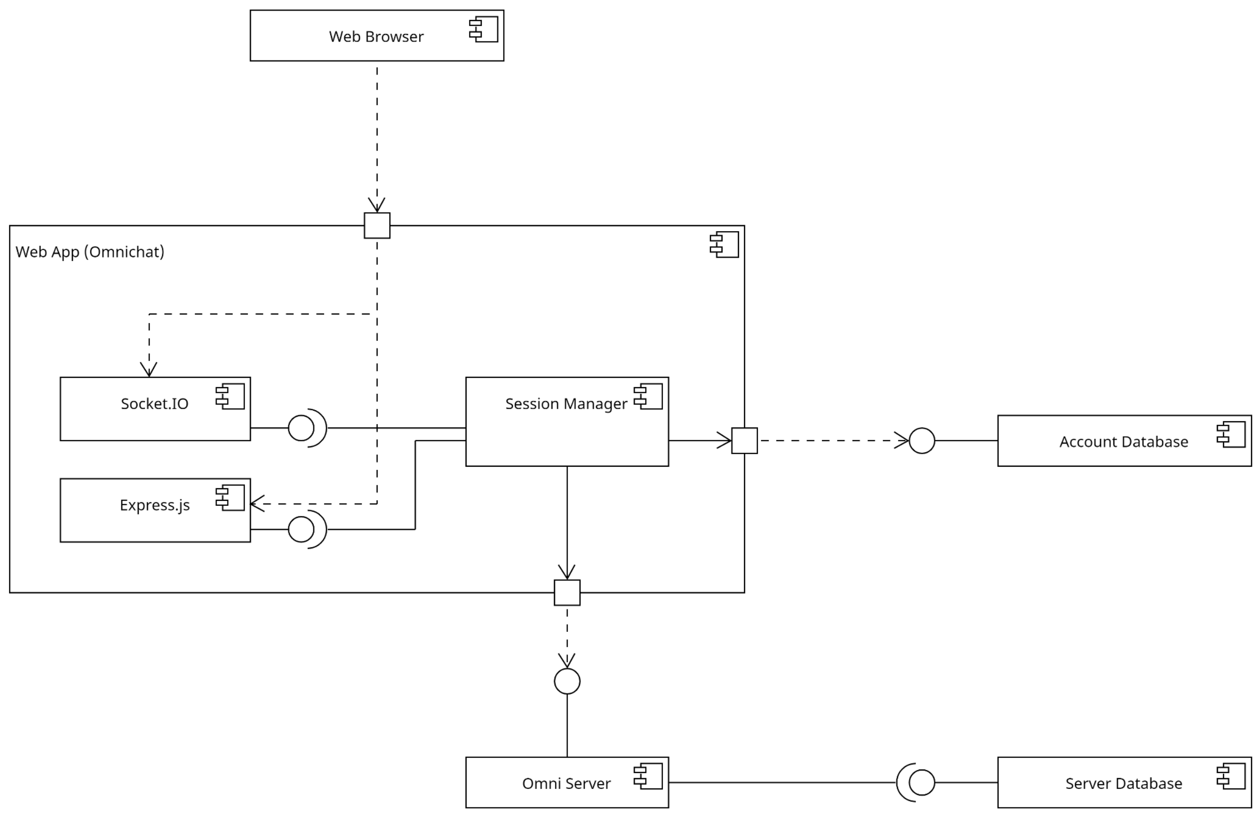


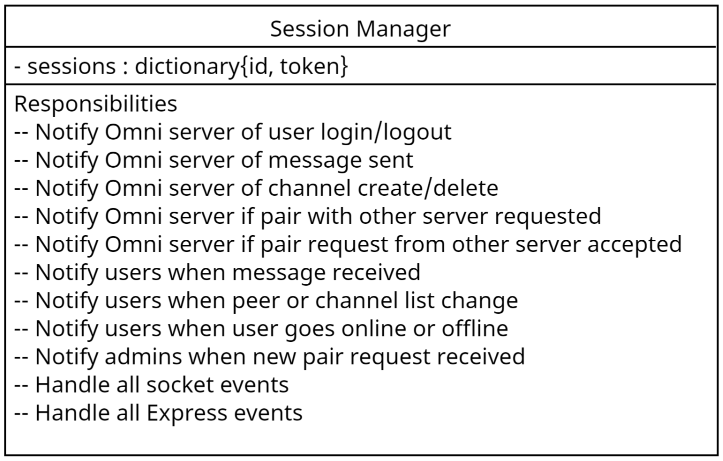
Figure 1. Component overview

As stated before, the Omni server ultimately **does not track user accounts**. It tracks some data, like the username and whether the user is an administrator. However, passwords, login methods like single sign-on, two-factor authentication, and so forth must be handled by the front-facing web application. When the web server notifies Omni of a new user account creation, Omni will generate a unique identifier for that user and store it along with their metadata. This identifier will be the **only** piece of data used in all communications between the web application and the Omni server when referencing a particular user. Users must never be referenced by username via the interface between the web server and Omni.

Though the Omni server is shown as not being part of the web application, it is important to note that because Omni is written in Node.js, the web application must load the Omni server, likely with the “require” function, just as it would with Express.js and Socket.IO. However, the Omni server is independent of the web application and the web server in its functionality - that is to say that it may exist independently of a web application; in theory, it could be loaded by any Node.js program, e.g. one that provides a Telnet interface to Omni, or something similar. It is therefore useful to keep the Omni server separate in the mental model and diagram.

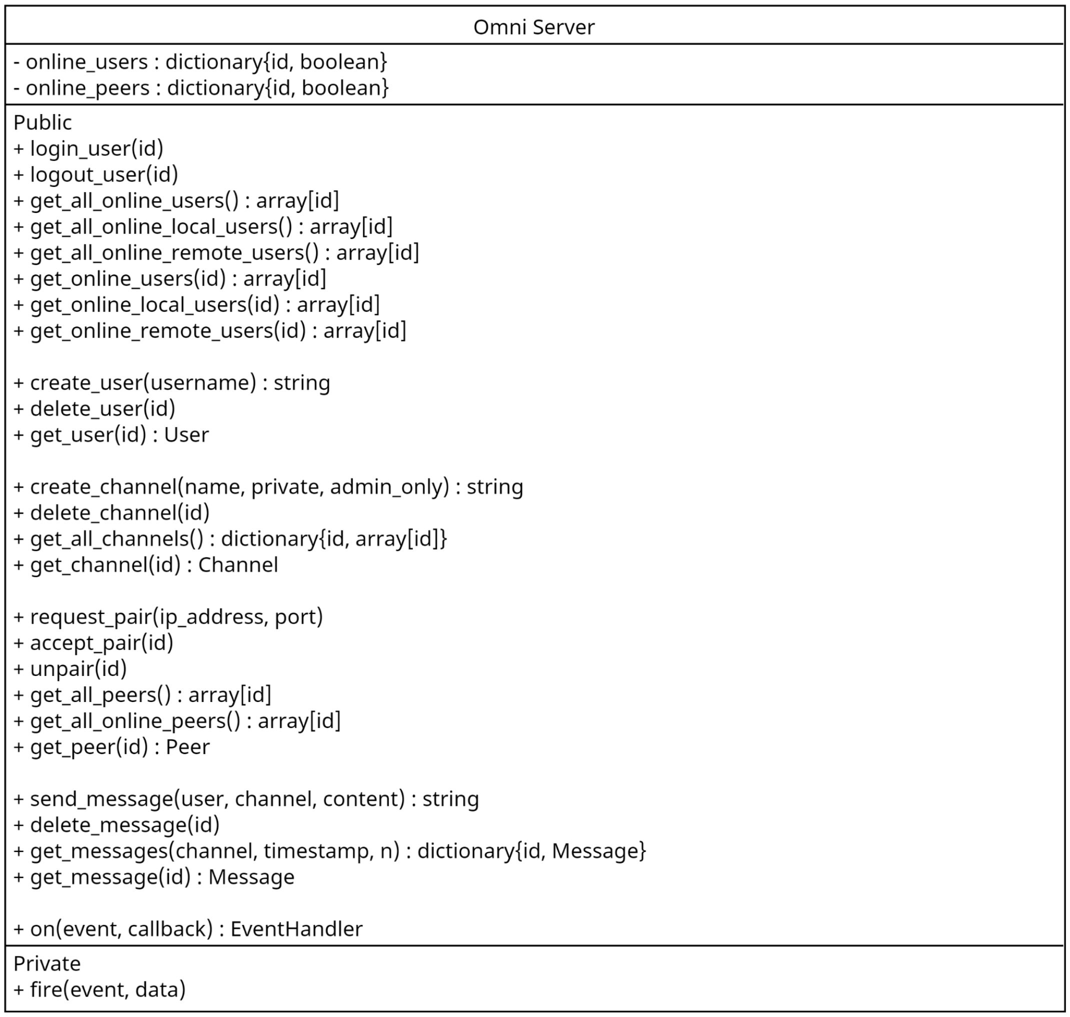
Decomposition

The two components that necessitate decomposition are the Session Manager and the Omni server. Their design is not yet finalized, as it must necessarily be flexible depending on how implementation proceeds. What follows are *tentative* diagrams, with some points intentionally left vague, and no doubt missing some functionality (mainly when it comes to communication with peers).



```
Session Manager
- sessions : dictionary{id, token}
Responsibilities
-- Notify Omni server of user login/logout
-- Notify Omni server of message sent
-- Notify Omni server of channel create/delete
-- Notify Omni server if pair with other server requested
-- Notify Omni server if pair request from other server accepted
-- Notify users when message received
-- Notify users when peer or channel list change
-- Notify users when user goes online or offline
-- Notify admins when new pair request received
-- Handle all socket events
-- Handle all Express events
```

Figure 2. Session Manager breakdown



```
Omni Server
- online_users : dictionary{id, boolean}
- online_peers : dictionary{id, boolean}
Public
+ login_user(id)
+ logout_user(id)
+ get_all_online_users() : array[id]
+ get_all_online_local_users() : array[id]
+ get_all_online_remote_users() : array[id]
+ get_online_users(id) : array[id]
+ get_online_local_users(id) : array[id]
+ get_online_remote_users(id) : array[id]

+ create_user(username) : string
+ delete_user(id)
+ get_user(id) : User

+ create_channel(name, private, admin_only) : string
+ delete_channel(id)
+ get_all_channels() : dictionary{id, array[id]}
+ get_channel(id) : Channel

+ request_pair(ip_address, port)
+ accept_pair(id)
+ unpair(id)
+ get_all_peers() : array[id]
+ get_all_online_peers() : array[id]
+ get_peer(id) : Peer

+ send_message(user, channel, content) : string
+ delete_message(id)
+ get_messages(channel, timestamp, n) : dictionary{id, Message}
+ get_message(id) : Message

+ on(event, callback) : EventHandler
Private
+ fire(event, data)
```

Figure 3. Omni server breakdown

Persistent Data Design

Database Descriptions

There are two databases between Omni and Omnichat. Omnichat holds an account database. In the account database, it tracks usernames, IDs, hashed passwords, a salt for the password hashes, and possibly some extraneous information such as account creation date. This data is laid out in the following figure.

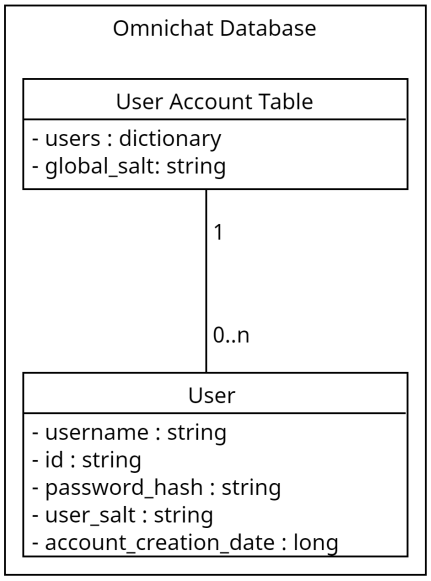


Figure 4. Omnichat database diagram

Omni holds a user table, a channel table, a message table, and a peer table. The user table contains user objects indexed by their ID. A user object includes a username, user ID, and administrator status. The channel table holds channel objects indexed by their ID; each has a name, channel ID, whether the channel is private to the local server, and whether the channel is administrator-only. The message table contains message objects, also indexed by their ID; each message has an associated channel ID, the ID of the user who sent it, and the message contents. The peer table contains information about peer servers, including their names and IP addresses. It also includes the ID of the local Omni server which will be sent to new peers. You will notice there is some redundancy between the Omni and Omnichat databases. This could be eliminated through, for example, procedure calls, but we see this as an unnecessary complication at this point in time.

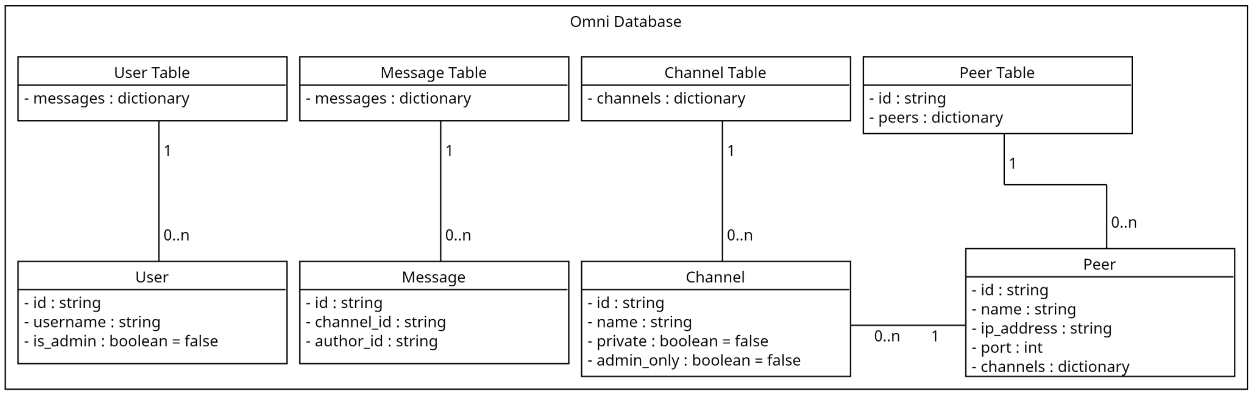


Figure 5. Omni database diagram

File Descriptions

No files will be used by Omni or Omnichat for configuration or data storage. Everything will be managed through the databases described above.

Requirements Matrix

Logged Out Users

Requirement	Component
Do not load any messages channels or other users when a user visits a chat page.	Session Manager
Redirect users to the login page when they click the login button.	Web Browser
Redirect users to the create account page when they click the create account button.	Web Browser
Redirect users to the chat page and grant the ability to use the chat interface when they have completed the signup or login process.	Session Manager
Reply with “username in use” error when the account create request received from the user contains a username that has already been taken.	Session Manager
Reply with “invalid passphrase” error when the account create request received from the user contains a username that has already	Session Manager

been taken.	
Create user account, save user account securely in database, create session for user, add user to list of online users, reply with session token when account create request received from user, all fields meet requirements.	Session Manager
Reply with authentication error when login request received from the user contains invalid credentials.	Session Manager
Create session for user, reply with session token when login request from user contains valid credentials.	Session Manager
Reply with “user is not logged in” error when logout request is received from the user.	Session Manager
Reply with authentication error when message is received from the user.	Session Manager
Reply with authentication error when the user requests a backlog of messages in the channel.	Session Manager
Reply with authentication error when the user requests the list of online users in the channel.	Session Manager

Logged In Users

Requirement	Component
Inform the web server of user logout, clear the session, and refresh the page when the user clicks the logout button.	Web Browser
Clear messages pane, retrieve messages in selected channel from cache or from server, populate messages pane. Clear online users list, retrieve online users from cache or from server, populate online users pane when the user selects a channel.	Web Browser
Make the send button visible and send action	Web Browser

available when the user types a message in the message box.	
Inform web server of message send, add message to the message pane, clear the message box when the user presses the send button or uses the send action	Web Browser
Update the channel list when a channel list update is received from the web server.	Web Browser
Update the online user list when an online user list update is received from the web server.	Web Browser
Add the message to the message pane when the message received from the web server is in the currently selected channel.	Web Browser
Add the message to the local cache of messages, to be displayed if the user chooses to open the channel when the message received from the web server is in a channel that is not currently selected.	Web Browser
Delete user from list of online users, delete session token, reply with successful logout message when logout request is received from the user.	Session Manager
Reply with “invalid channel” error when message received from the user and the user does not have access to the channel.	Session Manager
Reply with “invalid channel” error when the message received from the user is in a channel that is not a valid channel.	Session Manager
Forward message and user ID to Omni server when the message received from the user has access to the channel.	Session Manager
Reply with “invalid channel” error when the user requests a backlog of messages in the channel and the user does not have access to the channel.	Session Manager
Reply with “invalid channel” error when the user requests a backlog of messages in the channel and the channel is not a valid channel.	Session Manager

Reply with a backlog of the last 50 messages which have time stamps equal to or previous to *t* when the user requests a backlog of messages in the channel from before timestamp *t* and the user has access to the channel.	Session Manager
Reply with “invalid channel” error and the user requests the list of online users in the channel and the user does not have access to the channel.	Session Manager
Reply with “invalid channel” error when the user requests the list of online users in the channel and the channel is not a valid channel.	Session Manager
Reply with a list of online users in the channel when the user requests a list of online users in the channel and the user has access to the channel.	Session Manager
Send message to user when the Omni server requests a message be sent to a user.	Session Manager
Send message to all users specified by Omni server when the Omni server requests a message be sent to a group of users.	Session Manager
Send a channel list update to online users who have access to the channel when the Omni server replies to a channel creation or deletion request with a success message.	Session Manager
Send a channel list update to online users who have access to the channel when the Omni server informs that a peer has created or deleted a channel.	Session Manager
Send an online user list update to online users when the Omni server informs that a user has gone online or offline.	Session Manager
Send a channel list update to online users who have access to the peer when the Omni server informs that a peer has gone online or offline.	Session Manager
Send notification to server administrators through frontend when the Omni server informs that a new pair request has been	Session Manager

received.	
Send notification to server administrators through frontend when the Omni server informs of a successful pair.	Session Manager

Omni Server

Reply with a list of users to forward the message to, excluding any users in the ban list when the user’s message is received from the web server and the destination is a private channel.	Omni Server
Reply with a list of users to forward the message to (excluding any users in the ban list), inform peers of a new message when the user’s message is received from the web server and the destination is a private channel.	Omni Server
Reply with an empty array when the user’s message is received from the web server and the user is in the ban list or the time-out list.	Omni Server
Update list of online users and inform peers of a user list change when the web server informs of a user going online or offline.	Omni Server
Reply with a list of channels hosted on the local Omni server when the web server requests a list of local channels.	Omni Server
Reply with a list of channels hosted on peer Omni servers when the web server requests a list of remote channels.	Omni Server
Reply with a list of channels hosted locally or on peer Omni servers when the web server requests a list of all channels.	Omni Server
Reply with a list of users online with access to the channel specified when the web server requests a list of online users in a channel.	Omni Server

Reply with “invalid channel” error when the web server requests a list of online users in an invalid channel.	Omni Server
Reply with a list of paired Omni servers when the web server requests a list of peer servers.	Omni Server
Reply with “invalid channel” error when the web server requests a list of sent messages in an invalid channel	Omni Server
Reply with an array of messages in the specified channel, starting from the timestamp specified by the web server and going back *n* messages when the web server requests a list of *n* sent messages in a channel from before timestamp *t.*	Omni Server
Add a user to the ban list when the web server requests a ban of a user.	Omni Server
Add a user to the time-out list, schedule the user to be removed from the time-out list at a time *s* seconds in the future when the Web server requests a time-out of a user for *s* seconds.	Omni Server
Reply with “channel exists” error when the web server requests creation of a new channel and the channel name already exists in the list of local channels.	Omni Server
Create a new channel object internally, reply with a success message when the web server requests creation of a new channel and either the channel name does not exist in the list of local channels or the channel is private.	Omni Server
Create a new channel object internally, reply with a success message, and send channel list updates to peers when the web server requests creation of a new channel and either the channel name does not exist in the list of local channels or the channel is public.	Omni Server
Reply with “invalid channel” error when the web server requests the deletion of an invalid channel.	Omni Server
Delete channel object internally, purge all	Omni Server

messages from said channel from database, and reply with a success message when the web server requests the deletion of a private channel.	
Delete channel object internally, purge all messages from said channel from database, send channel list update to peers, reply with a success message when the web server requests the deletion of a public channel.	Omni Server
Purge all messages from the specified channel from the database when the web server requests a purge of all messages in a private channel.	Omni Server
Purge all messages from the specified channel from the database, inform peers of a channel purge when the web server requests a purge of all messages in a public channel.	Omni Server
Reply with “invalid channel” error when the web server requests a purge of all messages in an invalid channel.	Omni Server
Request the web server to send the message to local users with access to the channel when the peer informs of a new message in a channel.	Omni Server
Update the cached channel list for the peer and inform the web server of a peer channel list change when the peer informs of a channel list change.	Omni Server
Purge the local copy of all messages in the specified channel when the peer informs of a channel purge.	Omni Server
Update the cached list of online users for the peer and inform the web server of an online user list change when the peer informs of an online user list change.	Omni Server
Update list of online peers and inform the web server of a peer status change when the peer informs that it has gone online or will go offline.	Omni Server
Delete all data related to the peer, including all channels, messages, and online users,	Omni Server

from the cache/database, and inform the web server of peer going offline when the peer sends an unpair.	
Send list of channels and list of online users to the peer, inform the web server of the successful pair, inform the web server of the online peer, inform the web server of the peer channel list and online users when the peer accepts a pair request.	Omni Server
Inform peers that this server is now online when the Omni server is brought online.	Omni Server
Inform peers that this server is now offline when the Omni server is being shut down.	Omni Server
Inform the web server that a pair request was received when the pair request is received from the Omni Server.	Omni Server
Inform peer of the unpair, delete all data related to the peer, including all channels, messages, and online users, from the cache/database, and inform the web server of peer going offline when the web server requests an unpairing with the peer.	Omni Server
Inform peer of pair request acceptance, send peer list of channels and online users, await list of channels and online users from peer when the web server accepts a peer request.	Omni Server

Appendix A - Customer and Contractor Agreement

The customer and Team including: Micah Havens, Scott Devere, C.J. Commodore, Adnaan Dasoo, and Josh Martin are agreeing to the implementation of Omni in accordance with the information listed in the information above. The team and the customer are agreeing that everything listed above is acceptable and sufficient for the task that the customer needs. If future changes need to be made to this document all members of the team will meet with the customer to explain what needs to be changed and why, and upon agreement the changes will follow.

Dated Signatures:

- Micah Havens, 10/29/22, X____MH____
- Scott Devere, 10/29/22, X_______SD_____

- C.J. Commodore, 10/29/22, X______CC______
- Adnaan Dasoo, 10/29/22, X______AD______
- Josh Martin, 10/29/22, X________JM____

Customer Area:

Customer Comments:	
Date:__________	Signature:X___________

Appendix B - Team Review Sign-off

All members, including Micah Havens, Scott Devere, C.J. Commodore, Adnaan Dasoo, and Josh Martin, have reviewed the system requirements specification document for our software, named "Omni". Each team member has reviewed this document for accuracy and completeness in all parts, including text, diagrams, bullets, charts, and tables.

Dated Signatures:

- Micah Havens, 10/29/22, X____MH____
- Scott Devere, 10/29/22, X_______SD_____
- C.J. Commodore, 10/29/22, X_____CC_______
- Adnaan Dasoo, 10/29/22, X______AD______
- Josh Martin, 10/29/22, X_______JM_____

Appendix C - Document Contributions

- Micah Havens
  - Worked on: System Architecture, Persistent Data Design
  - Percentage estimate: 40%
- Scott Devere
  - Worked on: Appendices and Requirements Matrix
  - Percentage estimate: 25%
- C.J. Commodore
  - Worked on: Requirements Matrix
  - Percentage estimate: 10%
- Adnaan Dasoo
  - Worked on: Proofreading and editing
  - Percentage estimate: 10%
- Josh Martin
  - Worked on: References, ~Appendix A
  - Percentage estimate: 15%