Jesse Tiemens Software Engineering Capstone Documentation

Task 3

# B. Functionality Attributes

* Inheritance example:
  + The React framework, that this application uses heavily, is built on the concept of inheritance, as all React components that are created automatically inherit from the base ReactComponent class.
* Polymorphism example:
  + The ServerChannel component exemplifies polymorphism, as it is a single defined component that renders differently depending on properties passed during construction. Specifically, it changes based on the type property of the channel object that is passed to its properties. It looks up which icon to render next to its label based on which type of channel is passed, it will render a Hash icon for a text channel, a Mic icon for an audio channel, and a Video icon for a video channel.
* Encapsulation example:
  + All modals are encapsulated in a modal provider that is applied to the root layout document.
* Search functionality with multiple row results and displays:

A screenshot of a computer

Description automatically generated

* A database component with the functionality to securely add, modify, and delete the data:
  + Messages are stored in the database, and can be interacted with using the following methods:
    - **Add**: Typing a message into the chat box on the bottom of the screen and sending it adds it to the database.
    - **Modify**: Messages that you have previously sent can be modified by clicking the edit button in the top-right corner of the message display, which will update the database as well as send out a signal on a web socket to update the message on all other users’ screens.
    - **Delete**: All users can use the delete button in the top-right corner of the message that they have sent, and moderators/administrators can delete anyone’s messages. This does not technically delete the message from the database, as it will only remove the content of the message, and still show up to other members like so:
    - A screen shot of a computer

      Description automatically generated
    - For examples of actual deletion from the database, that happens when channels or servers are deleted.
* Ability to generate reports with multiple columns, multiple rows, date-time stamps, and title:
  + Visiting /generate-reports will provide these tables to report servers and profiles that exist on the application.

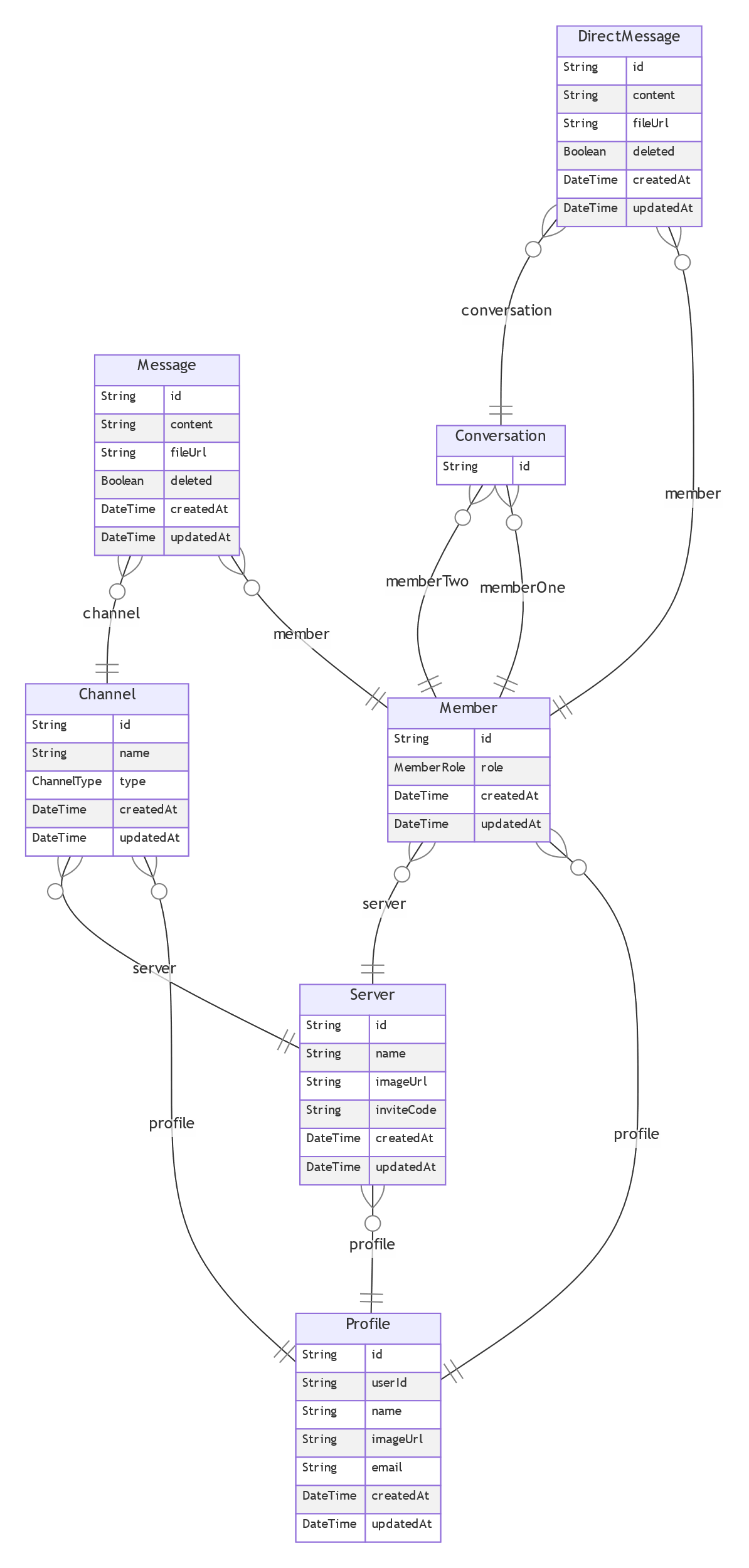
A screenshot of a computer

Description automatically generated

* Validation functionality:
  + A user is allowed to upload attachments to messages in the form of only image formats or pdfs, validation code is provided to prevent any other file format from being uploaded.
* Industry-appropriate security features:
  + The entire website is protected by an authorization middleware process. Accessing any page on the website will redirect to a sign-in/sign-up page so that unauthorized access is impossible. Further authorization on a server-by-server basis is provided through administrator and moderator roles that can be assigned by server owners and administrators.
* Design elements that make the application scalable:
  + The fact that we have chosen cloud solutions for both hosting the server and the database is inherently scalable. Design elements that the application itself utilizes to promote scalability include:
    - Any area on the user interface that has a variable amount of content is placed within a ScrollArea component.
    - Messages on servers and direct message conversations will obviously rack up many items as people chat. There is an inherent pagination to the way the client loads messages. It only loads the most recent page’s worth of messages, based on the viewport height of the device. As the user scrolls up, it will load more messages progressively. This allows for a smooth user experience that scales with the number of messages in each location.
    - As servers grow, there will end up being many users and channels. There is a search component for quickly locating elements, so that the scale of the server does not impact a user’s ability to locate a particular element quickly and easily.
* A user-friendly functional GUI:
  + Our GUI is user-friendly, as it is trying to replicate a GUI that is already user-friendly. However, it can also be argued that our GUI is user-friendly as no functionality is more than a couple of clicks away. Our GUI is also functional, as every interaction that is available to the user have all been tested thoroughly to ensure functionality.

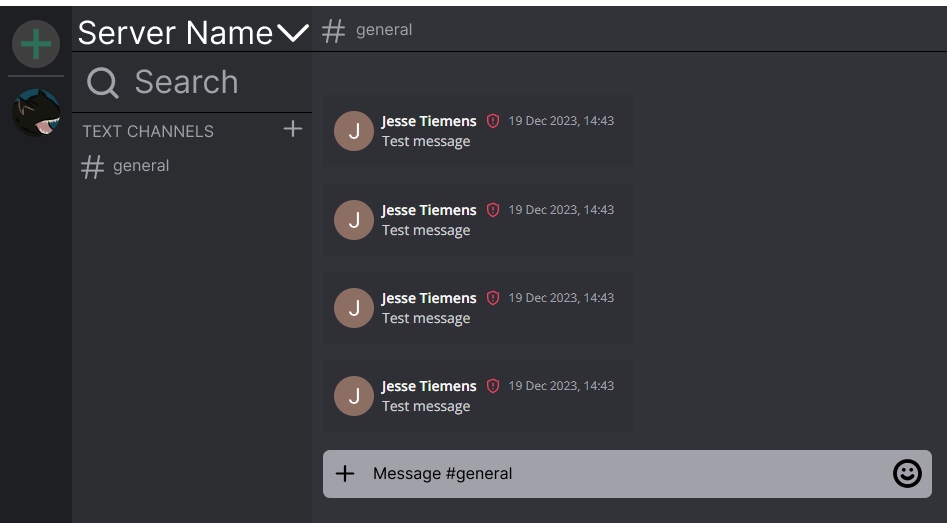
# C. Other Documentation

## Class Diagram



## Design Diagram

Low-fidelity wireframe of main view:



## Web App Host Location

<https://ele-discord-clone.up.railway.app/>

## GitLab Repository Location

<https://gitlab.com/wgu-gitlab-environment/student-repos/jtieme2/d424-software-engineering-capstone>

## User Setup/Running Maintenance Guide

## User Perspective Application Use Guide

1. Visit the website.
2. Log in or register.
3. Create your own server or click on an invitation link you’ve received from a friend.
4. If you created your own server, if you click the server’s name and click Invite People to open a window that will generate an invitation link you can copy to your clipboard and send to your friends.
5. You can see which servers you are in on the left-most navigation panel of the screen, clicking on servers will allow you to navigate to that server.
6. The next panel will show you the available channels in that server, selecting a text channel will change the main section of your screen to show the messages that users have sent in that channel, with an input for you to type and send your own messages at the bottom.
7. Other channel options include audio and video channels, clicking on one of those will change the main section of your screen to be a media room where you can talk to and see the users that are currently in those rooms.
8. When you are done with the application, you can click your user avatar in the bottom-left of the screen and select sign out.

# D. Testing Documentation

# E. Panopto Video Recording