QUEEN MARY
UNIVERSITY OF LONDON

Haskell Individual Project of using threads and concurrent computation

Student Name: Nijaguna Darshana
Student Number: 230671566

QUEEN MARY
UNIVERSITY OF LONDON

Overview:

This project simulates a social network using Haskell. It creates 10 users with unique names and starts a web server. Each user periodically sends a message to a random user. The content of the message is a random quote from Shakespeare's Hamlet. The simulation stops sending messages when the counter reaches 100.

Design Decision:

User and Message Types

The User type represents a user in the social network. It has a username and a mailbox which is an MVar containing a list of Message objects. The Message type represents a message sent from one user to another. It contains the sender, receiver, content, and timestamp.

The decision to use MVar for the mailbox was made to ensure thread safety. Since multiple threads could potentially access and modify a user's mailbox at the same time, MVar provides a way to ensure that only one thread can access it at a time.

Difficulty: Yesod Library

The Yesod library was considered for this project due to its comprehensive features for building web applications. However, it was too large and complex for the scope of this project. Instead, a simpler approach was taken using the Warp library for the web server and Blaze for HTML generation.

Another issue was controlling the number of messages sent. This was solved by using a shared TVar to count the number of messages sent and stopping the simulation when the counter reaches 100.

Extension: An UI

The Web.hs module in this Haskell program is responsible for starting a web server and generating an HTML page that displays the messages sent between users in the simulated social network.

The startWebServer function is the entry point for this module. It takes a function getUsers as an argument, which when called, returns a list of User objects. The function starts a web server that listens on port 3000. When a request is received, the server calls getUsers to get the current list of users. For each user, it reads the user's mailbox by calling readMailbox, which returns a list of Message objects. These messages are then passed to the generateHtml function along with the list of users. The generateHtml function generates an HTML document that displays the messages for each user. It uses the Text.Blaze.Html5 library to generate the HTML. For each user, it creates a heading with the user's name and a list of messages. Each message is displayed with the sender's name, the content of the message, and the time the message was received. The generated HTML is then sent as the response to the request. The responseLBS function from the Network.Wai library is used to create the response. The status code is set to 200 (OK), the content type is set to "text/html", and the body of the response is the generated HTML.

Note: Regarding permissions for port 3000, it's important to note that some systems may require you to grant permissions to allow the program to listen on this port.