ECS713P Functional Programming

Sai Vivek Rambha (230585403)

Summary:

The goal of this project is to describe the concurrent operations and thread-based features of a Haskell application.

Compiling and Running the app:

- Stack Build
- Stack Run

How Does app work?

This application simulates a social network by creating ten users and sending them random messages, each represented by a thread.

Application Flow:

1. Remove an existing file “all_messages.txt” which is where all the messages will be logged.
2. Creates user with predefined usernames in the Utils.hs.
3. Starts concurrent threads for each user and sends random messages to random users.
4. Appends messages to all_messages.txt.
5. Tracks total messages sent and stops after 100 messages.
6. Displays total number of messages send.
7. Finds and displays the user with most number of messages.
8. Shows all Users with number of messages received.

Mvars Used: used to manage message data, access to the file and keeps track of all the messages.

1. User Messages: Each user has an MVar [Message] to store and manage messages concurrently.
2. File Lock: ensure that only one thread writes to the txt file.
3. Message counter: tracks the total number of messages sent and stopping the process after 100 messages.

Issues Faced:

When closely observing the txt file I found that occasionally the users are attempting to send messages to themselves, which was unrealistic. This was solved by improving the user selection process. Another issue was concurrent file writing, where incomplete messages were logged, to solve this I added file lock. The file caused a problem because it was only updated the old file. To solve this, I used to remove the file if it already existed and create a new file for each run.

Extra Functionalities:

A feature has been added that determines the maximum number of messages delivered and retrieving user information for the user who has received highest number of messages. The output also includes transaction information. Each user is assigned a list of customised messages chosen at random before transmitting them. We can determine which user has been using the social media platform the longest with the help of this function.

Design Choice:

Main.hs: It is responsible for setting up used data and user activities. Main functions include creating new user, sending messages between users, appending these messages in a txt file and provides the total messages received.

Message.hs: This module defines the structure and data types of messages which has sender name and message content. It also displays the message as a string.

User.hs: It has the data structure for a user. Each user is identified by name and a Mvar which holds a list of messages.

Utils.hs: This contains list of names for the users.

UserAnalytics.hs: This module identifies which user has sent the most number of messages and displays the user. It also handles custom messages that can be sent.

ExceptionHandlers.hs: This module handles the file exception where if the file is already present it deletes the old file and creates a new one.