# Haskell Social Media Simulation Project Report

## Introduction

This Haskell project aims to simulate a simple social media platform where users can create accounts, send messages, and receive notifications. The following report outlines the technical challenges encountered, design decisions made, and the rationale behind these decisions.

## Design Decisions

**User and Message Modelling**: The User type was designed to encapsulate a username and a list of received messages, providing a clear representation of a social media user. The Message type captures both the sender and content of a message. The decision to use newtype for Username, Sender, Content, and Question was motivated by the need for type safety, ensuring that functions operating on these types could not be mistakenly supplied with plain strings. This design choice improves code clarity and reduces the risk of errors.

**Concurrency Choices**: MVar was chosen to manage shared state due to its thread-safe properties. It ensures that when one user (thread) sends a message, other user threads can safely read and update their state without causing race conditions. The MVar also serves as a mutable reference where concurrent threads can perform atomic read and write operations, which is crucial in a setting where multiple users interact simultaneously.

## Challenges Faced

**Concurrency Management**: One of the main challenges was ensuring that message sending and receiving did not result in lost or duplicate messages. The use of MVar helped address this, but careful attention was needed to manage the possibility of deadlocks. Implementing non-blocking patterns and ensuring that MVar operations were performed in the correct sequence was a critical part of the solution.

**User Experience**: Initially, users could not identify who sent them messages, leading to confusion. We addressed this by including sender information within the Message type and providing notifications that clearly display the sender's username.

## Additional Feature Implementation

**User Account Creation**: A feature was added to allow users to create their own accounts. Upon starting the application, users are prompted to sign up for the social media platform. This involved updating the Main module to handle user input and modify the user list dynamically.

**Message Notifications**: To enhance the user experience, the system was updated to notify users of new messages and display the sender's details. This required adjustments to the User module to handle the retrieval and display of messages in a user-friendly format.

## Conclusion

The project effectively showcases a multi-threaded Haskell application, focusing on type safety and concurrent state management through MVars. The inclusion of features like account creation and message notifications enhances user interaction. Future iterations could explore advanced UIs and user functionalities to mimic a real social media platform more closely. The project adeptly utilizes Haskell's type system and concurrency capabilities for a practical application.