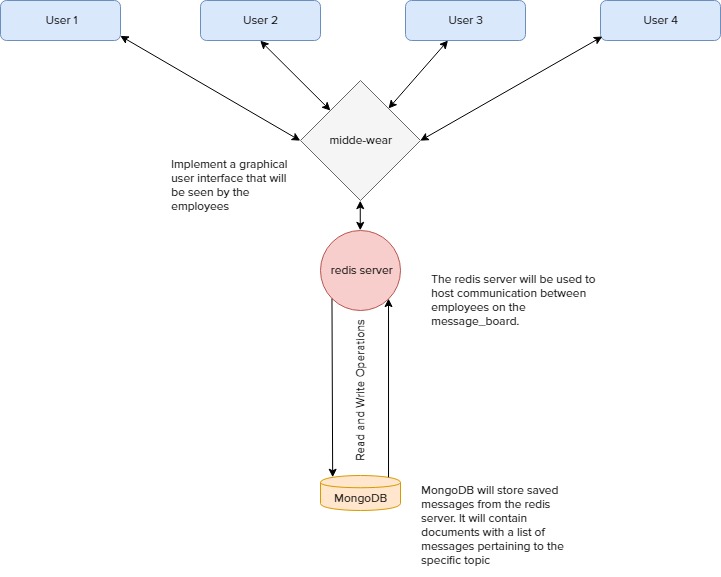
P1 Task 2: Report

Raymond Zhu

923008555

Project 1 Task 2 took me approximately 6 hours to complete.

It has come to my attetion that using redis will be optimal option for communication between employees. The benefit of using redis over such a database like MongoDB is that redis provides faster performance (enables our employees to receive messages through the message\_board extremely fast), gives us the opportunity to use rich data types, atomic operations, and many other functions. In the diagram represnted below, we would not want our employees to chat through command prompts and so we will need to develop an application and/or graphical user interface that will be seen by the employees. Redis will be used as the primary database for communication and MongoDB will be used as our backup. If our server were to encounter any issues, we will also be able to recover data from MongoDB. Additonally, some information that are discussed between employees may be sensitive information. The option of saving to cold storage will be given (saved to MongoDB). To ensure that our redis server is reliably fast, and able to service many employees, messages stored on our redis database will be removed after a certain period or if space is limited. The resulting information will be transferred to our mongo database (if not already).



Looking at the figure above, employees (known as users) will access message boards through a graphical user interface and send/receive messages through our redis server. Depending on how the interface is designed, we can list our message boards similar to the looks of an online forum. Selecting a message board will send select requests to our server to host the given employee. The user will be shown a blank chat log once they join in and they also have the option to scroll up to previously old messages. This will send in read requests to our server and display older messages to the user (this operation may be slow since messages may need to be recovered from our mongo database). For any exceptions that occur during user interaction with the server, the user will be directed to a safe state and errors will be logged on our server so our developers will know what issue occurred. And so, every interaction that a user can make will correspond to the functions that will be made to our server.

A simple prototype to show user interaction with a redis server has been created. The prototype contains a simple terminal interaction using a local server. This will give you all a basic understanding of the core functionality with using redis. A readme on how to run the prototype has been included.

The tradeoff between availability and consistency is important for a distributed system to function properly. Let us consider each employee as node in a distributed system. Every employee should have their own box and each box comes with a set of memory, storage, etc. that a business provides. For every box that communicates with the redis server, chat logs can be stored locally on boxes to provide consistency checks. If we are to implement a distributed system, it will be better off to sacrifice consistency for availability. It will be ideal to provide reliable service to our employees even though disruptions in our server may cause problems. During information storage, our server can then communicate with nodes present at the time to validate corrupted messages and ensure correct messages are copied into cold storage.