Architecture Document

Version 1.0

In Preparation for:

Computer Science 4770

Authored by:

Kristan Hart
Keir Strickland-Murphy
Diego Zuluaga
Meishang Chen
Aizaz Iqbal

Table of Contents

1 Introduction

1.1 Purpose:

The purpose of this document is to present a detailed description of the designs of the Memorial University Student Network (MUNSN), created for Memorial University. As this is an architecture document, this is intended for consumption by programmers and other software professionals as a guideline for implementing the project.

1.2 Scope:

This document gives a detailed description of the software architecture of the website. It specifies the structure and design of some of the modules discussed in the SRS. It also displays some of the use cases. The class diagrams show how the programming team would implement the specific module.
The scope of this project includes the five members of Team C, Diego Zuluaga, Meishang Chen, Kristan Hart, Keir Strickland-Murphy, and Aizaz Iqbal. The features of the project will be implemented using Node.js, MongoDB for the databasing, and GitHub for version handling and distribution. The main goal of this project is to provide a social network for students in Memorial University. Students should be able to sign up, have a timeline, create groups, post lost & found items and create polls.

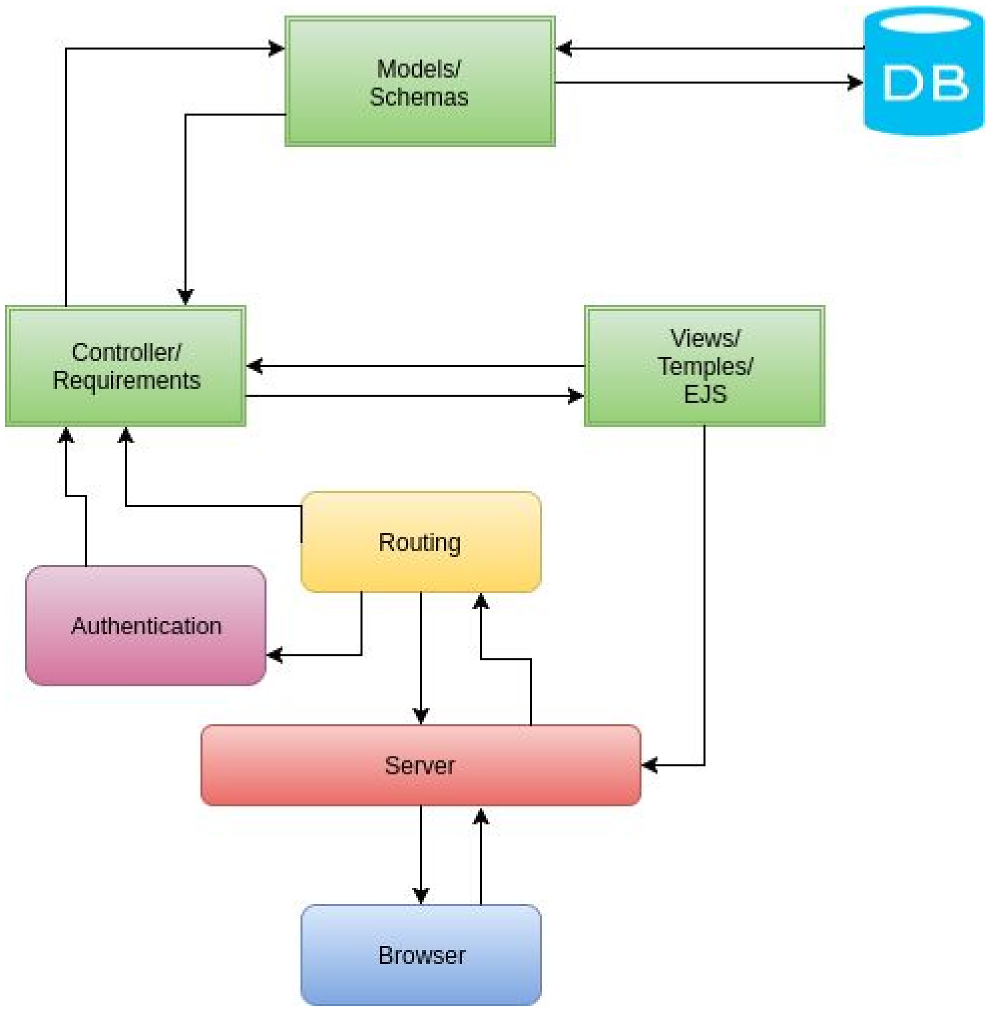
1.3 Definition & Abbreviations:

- MUNSSN - Memorial University of Newfoundland Student's Social Network, the name of the project being developed.
- User - Someone who has not yet signed up to the MUNSSN
- Member - Someone who has signed up for the MUNSSN

1.4 References:

1. MUNSSN Project Description
2. On the Criteria to be used in Decomposing Systems

2 Design of the System:



3 Decomposition:

Decomposition of the software allows for the functionality of the software to be placed into manageable working pieces. Our decomposition model follows the modularization 1 model presented in Parnas(1971), and represents a workflow model of modularization.

3.1 Module design

3.1.1 Template module

The template module provides the view of the website to the user through html and ejs, and allows a platform for the user to input information to the database and request information from the server.

3.1.1.1 Pseudocode

```
START
    load fields from HTML
    load style from CSS
    get data from server
    fill fields from data
    generate userInputHandlers

    userInputHandler (userInput) {
        //Text fields, buttons, etc
        handle userInput
    }
END
```

3.1.2 Server module

The server module provides the communication between the website to be routed through the controller and then either to the database or the views. Our software uses NodeJS for this module.

3.1.2.1 Pseudocode

```
Server
Start

http = require(“http”)
Server = createserver(){insert server parts here}
Server.listen(port)
Server.log(“message here”)

end
```

3.1.3 Authentication module

When users encounter the website they either have to login or signup. Before any further action can be taken the user needs to be authenticated. The user inputs the data through the ejs template using javascript, and this data get sent through the controller to be compared against the user data in the database.

3.1.3.1 Pseudocode

```
Start

GET request
if canBeHandled(request):
if checkValidation(info):
        return true
else:
        return false

End
```

3.1.4 Requirements module

The requirements module is the controller of the MVC. This controls the routing to the views, the authentication to the database, and schemas/database information to back to the server.

3.1.4.1 Pseudocode

```
Start
//Server Requirements
Set requirement 1
Set requirement 2
...
Set requirement late // These would be everything from express to the database
config

Connect MongoDB // or alternate DB

Require(session Manager)

//express setup
Use requirement 1
Use requirement 2
...
Use requirement Last

//Session Manager setup
Use requirement 1
Use requirement 2
...
Use requirement Last

//Ports
require(routes)(SessionManager)
```

3.1.5 Routes module

The routes are used to route information from the server module to the controller to be passed on to either the database or to the views and then onto the browser. The routing is written in javascript.

3.1.5.1 Psuedocode

```
START
    get request
    switch (request){
        case case1:
            handle request
            break

        case case2:
            handle request
            break

        // An arbitrary number of cases

        case caseN:
            handle request
            break

        default:
            default handle
            break
    }
END
```

3.1.6 Schemas module

Schemas module holds all of the structure of the database to be placed into the database module. Mongoose is the ODM used to create the schemas.

3.1.6.1 Pseudocode

```
Start

define schema
checkSchema(schema):
    export schema

End
```

3.1.7 Data module

The database module takes information from the schemas and passes the data back when requested. The database module uses the MongoDB language.

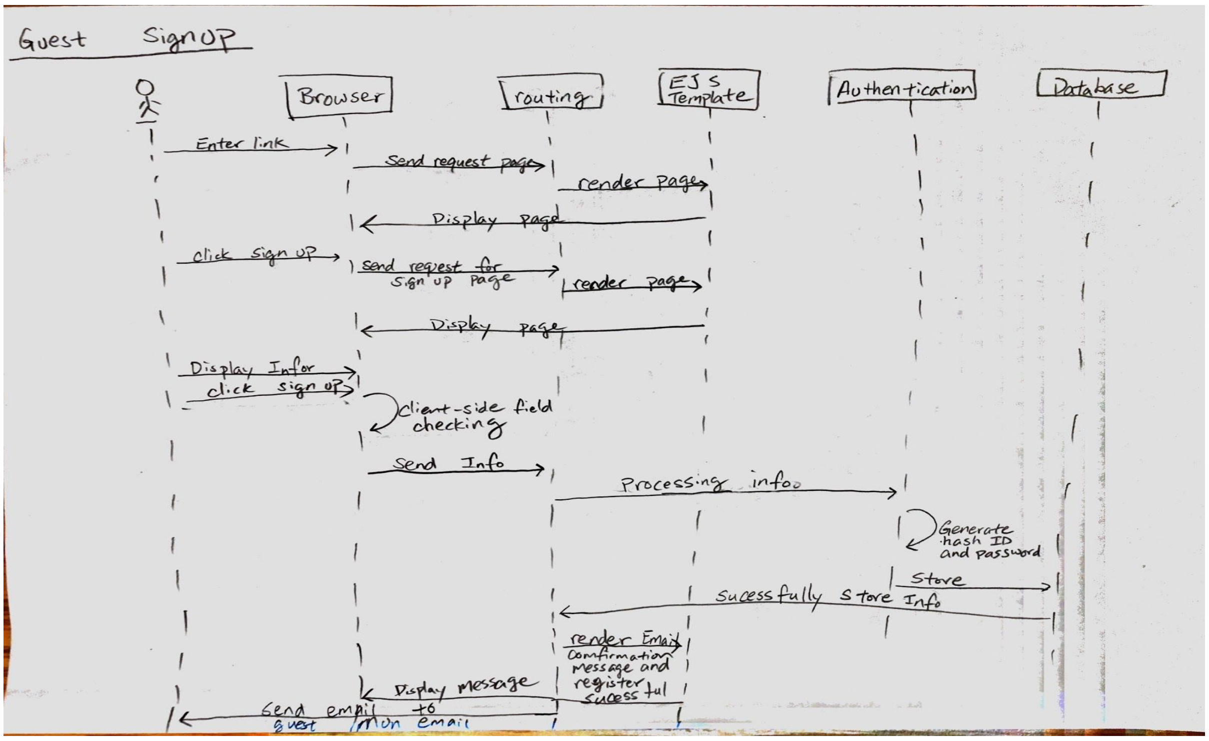
3.1.7.1 Psuedocode

```
Start

GET query

if found(query):
        return info
else:
        return fail

End
```

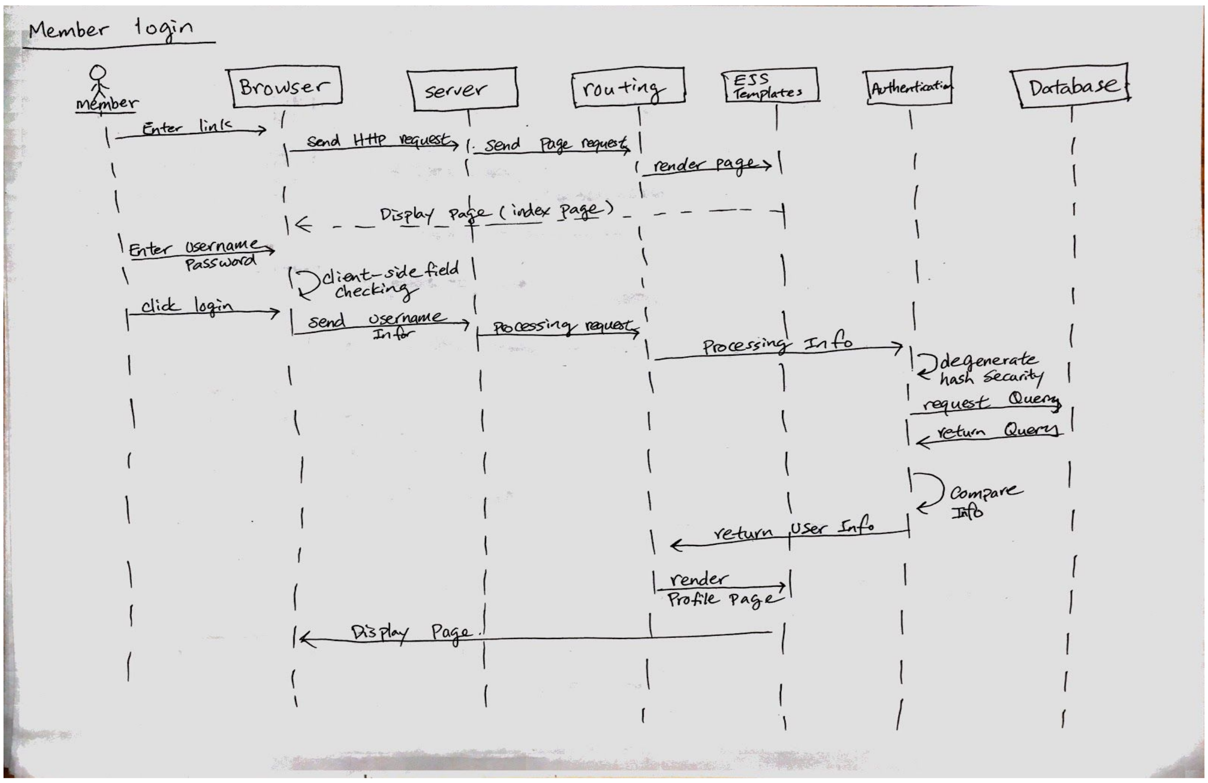
4 Sequence Diagrams of System Functionality

The following sequence diagrams further decompose the modules into how the user will interact with the software. The browser is only included to show at what level the user will interact with the software.

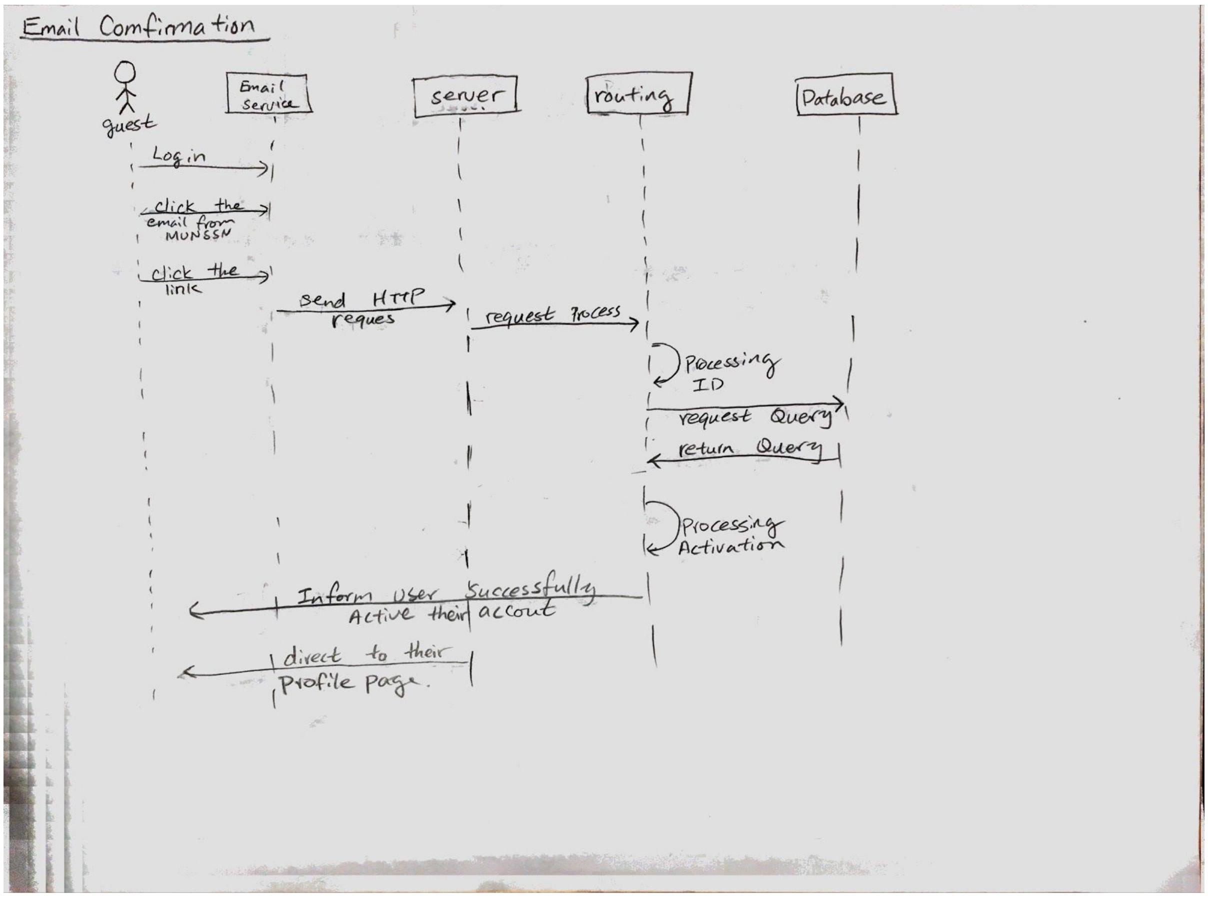
4.1 Guest signup:



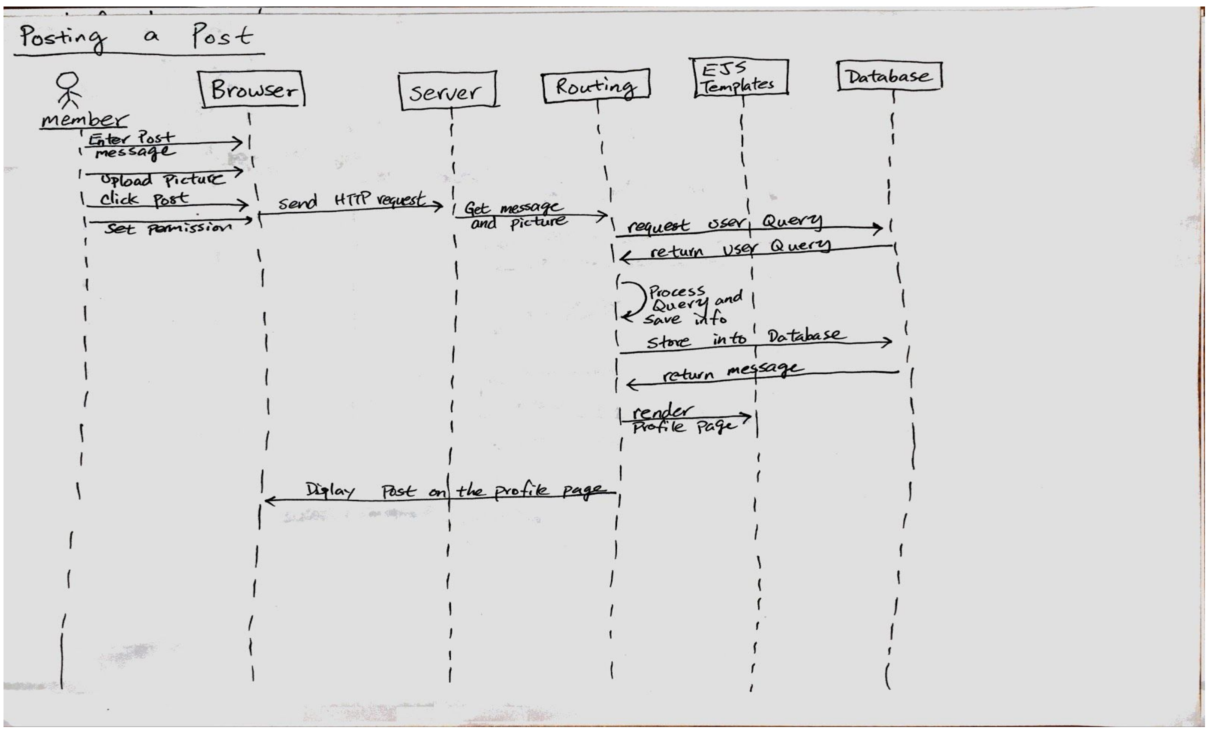
4.2 Member Login:



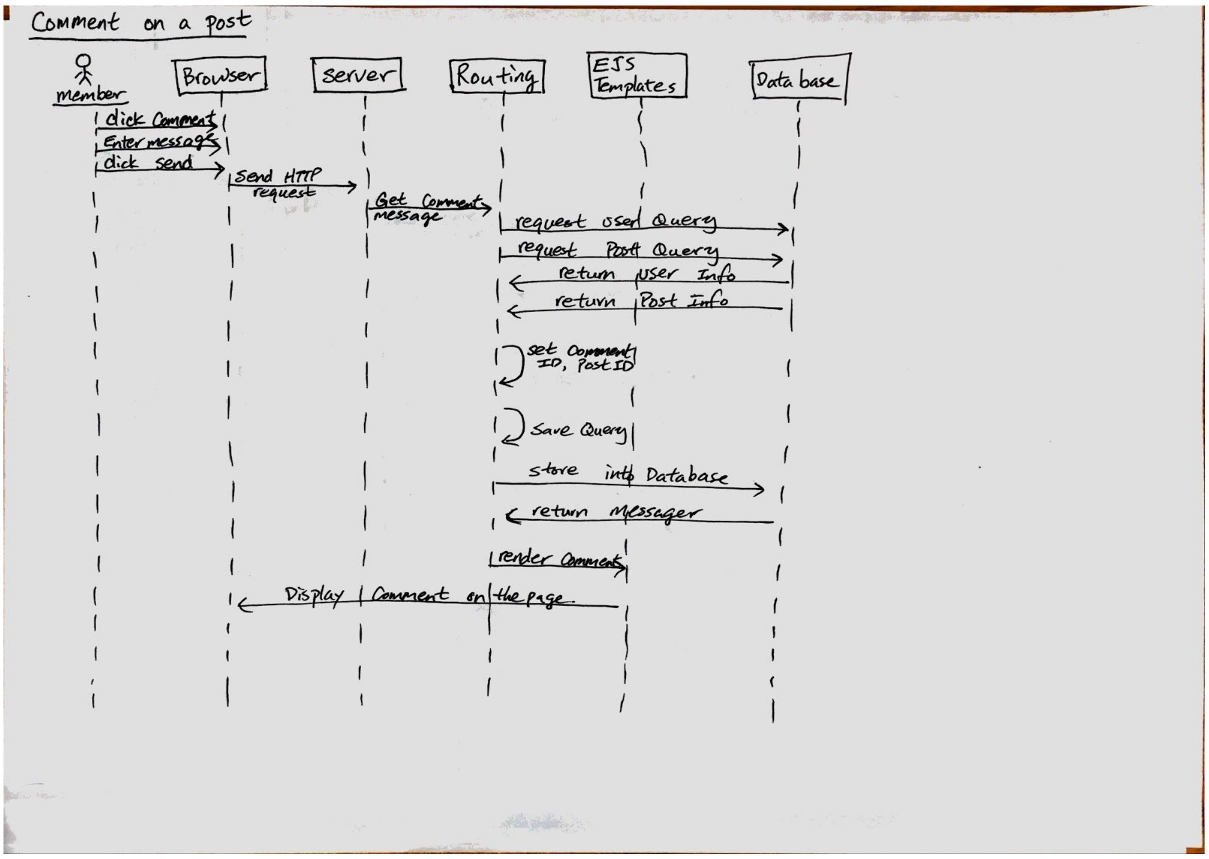
4.3 Email Confirmation:



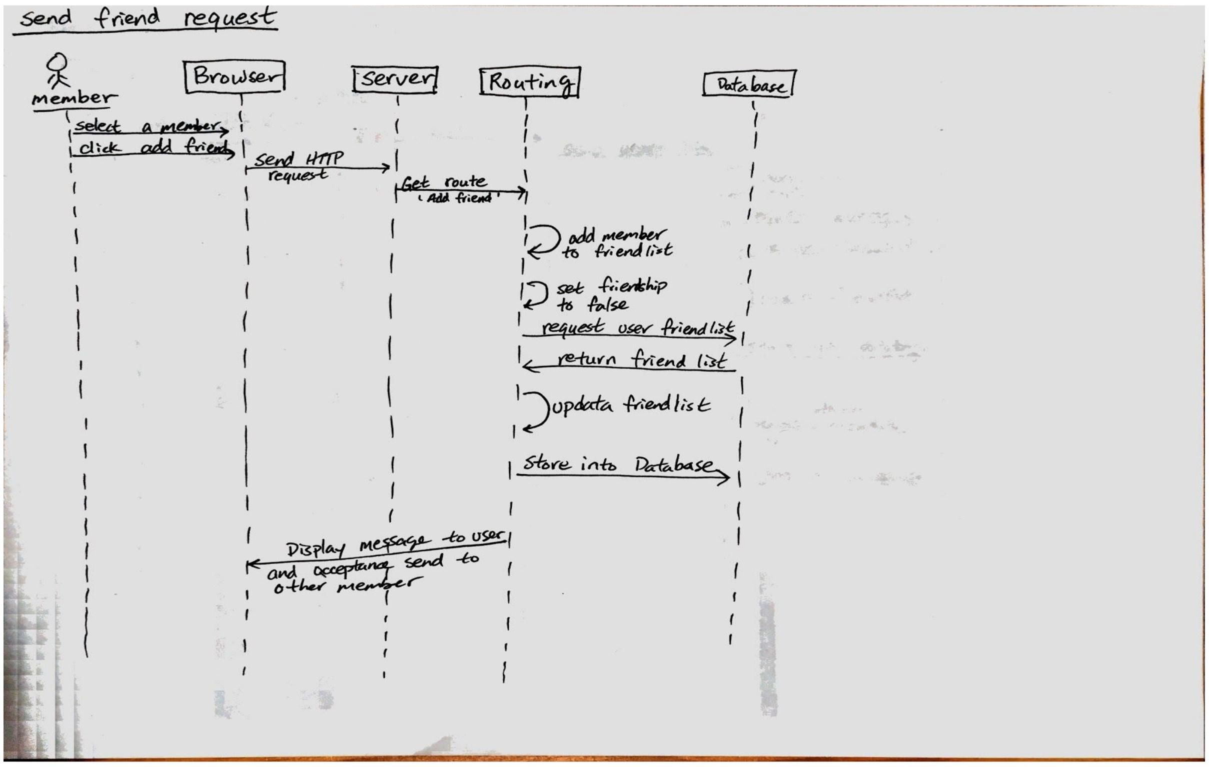
4.4 Posting a Post:



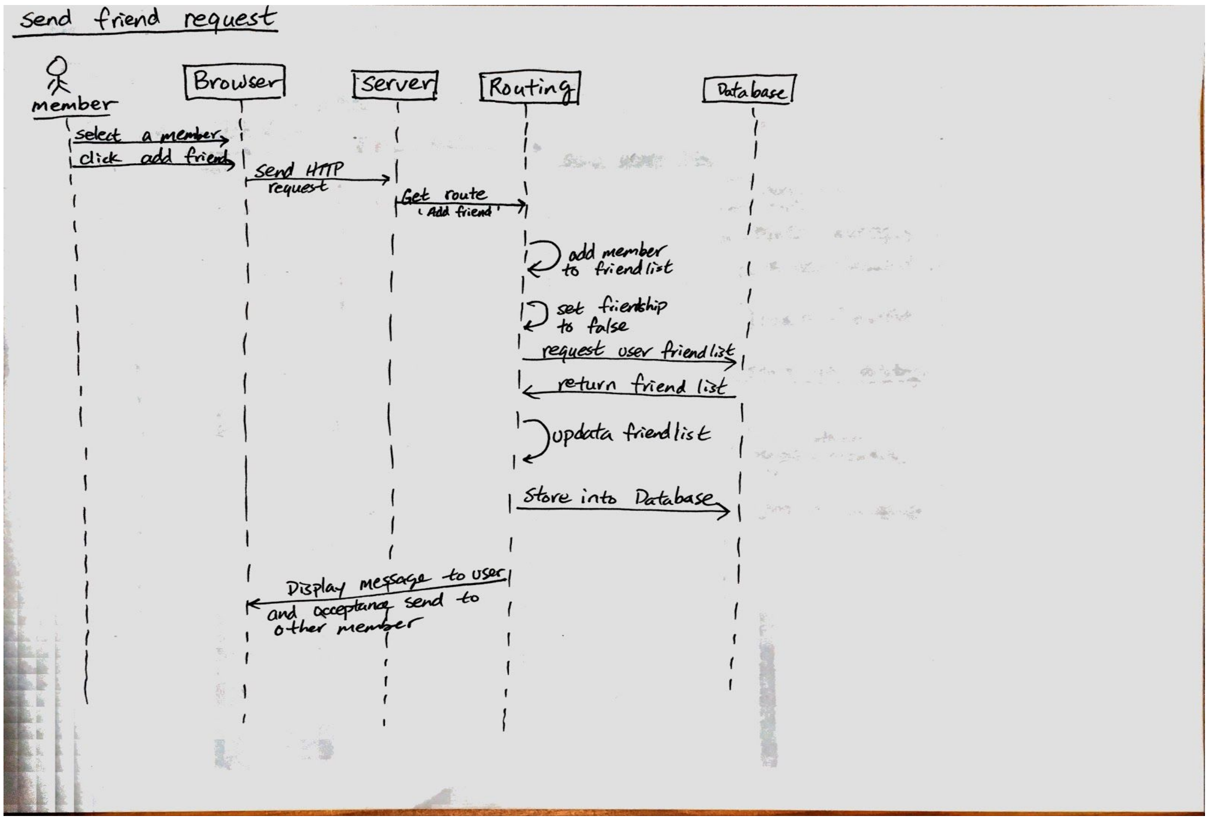
4.5 Commenting on a Post:



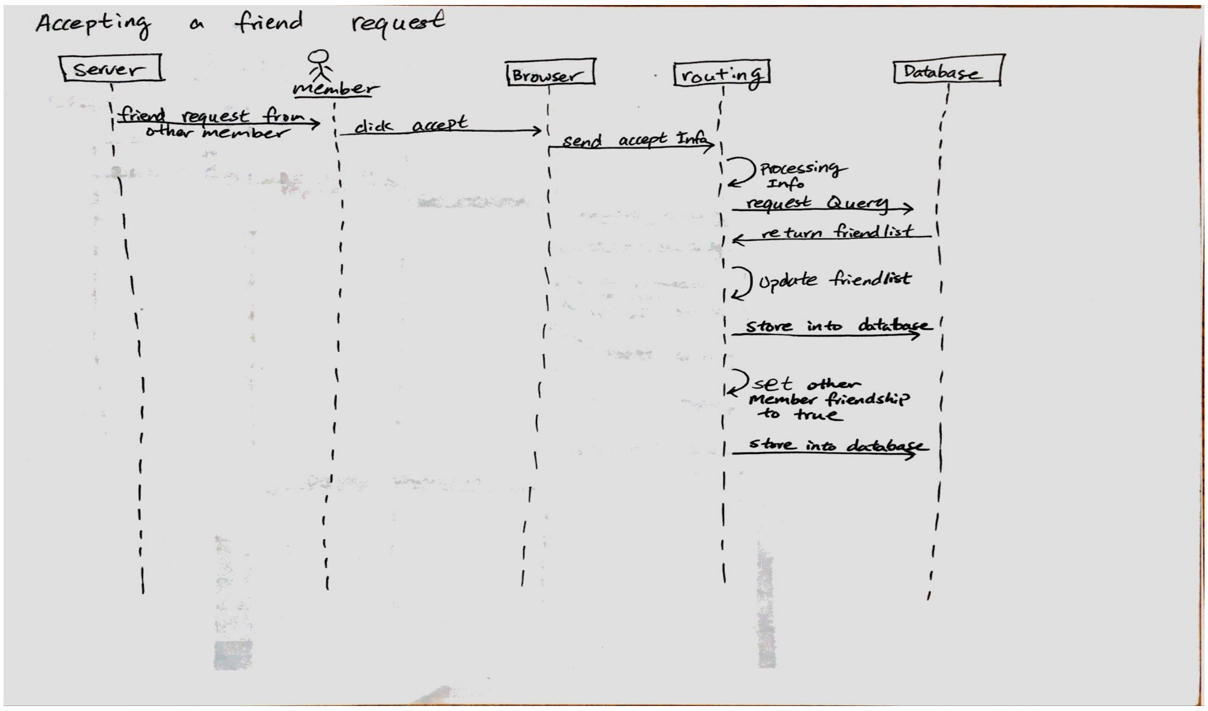
4.6 Reply to a Comment:



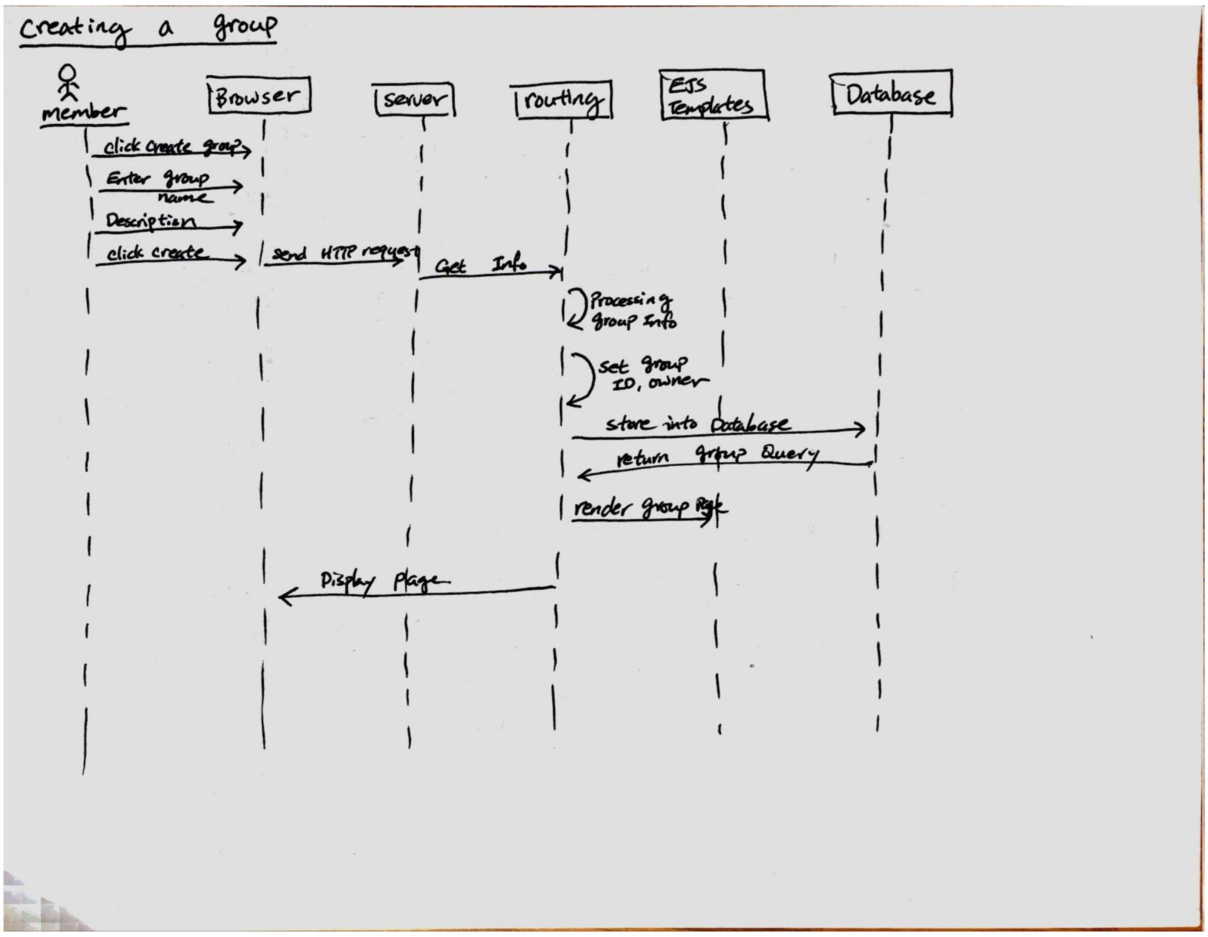
4.7 Sending a Friend Request:



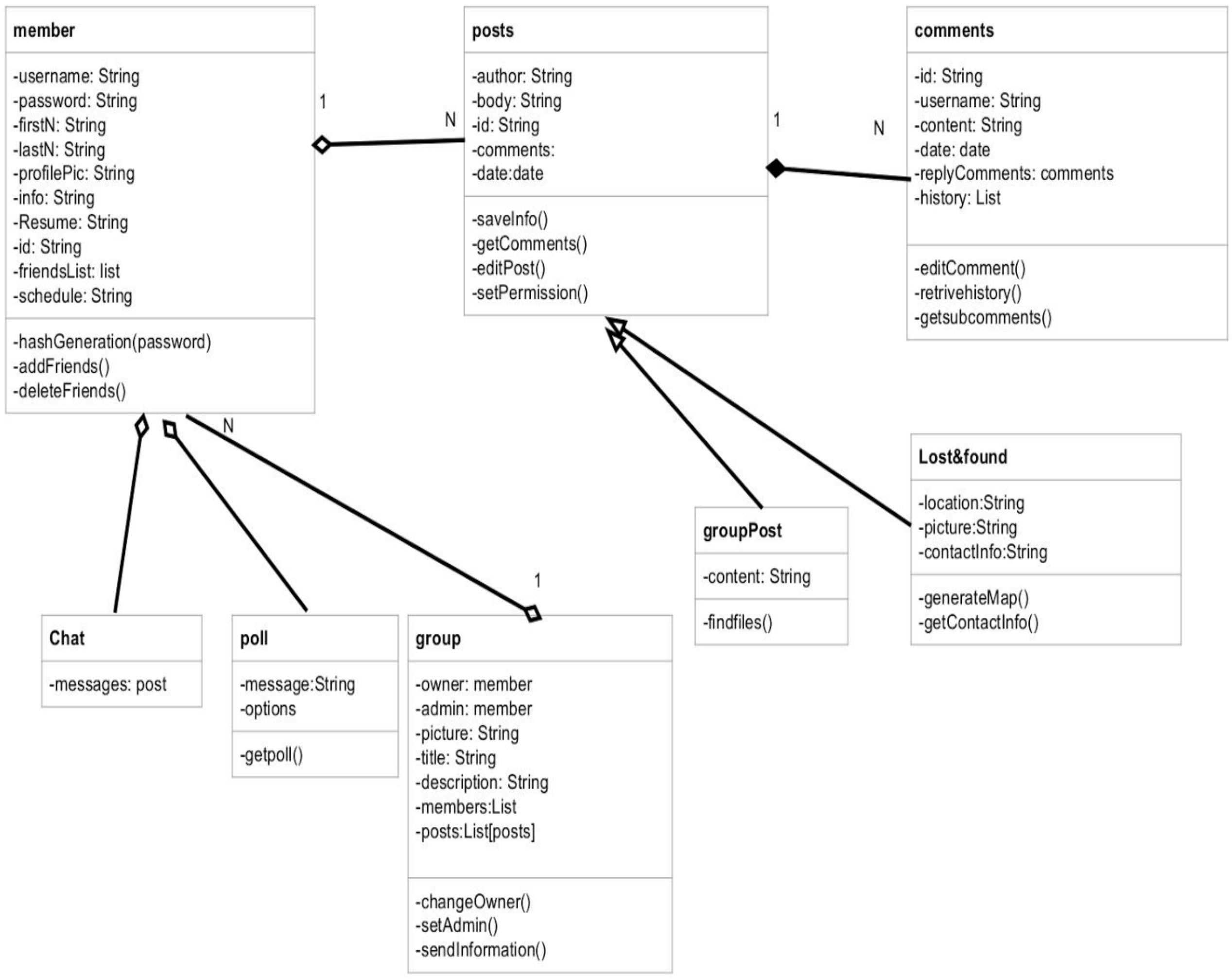
4.8 Accepting a Friend Request:



4.9 Creating a Group:



5 Class Diagram for System Requirements:


member
-username: String
-password: String
-firstN: String
-lastN: String
-profilePic: String
-info: String
-Resume: String
-id: String
-friendsList: list
-schedule: String
-hashGeneration(password)
-addFriends()
-deleteFriends()
posts
-author: String
-body: String
-id: String
-comments:
-date:date
-saveInfo()
-getComments()
-editPost()
-setPermission()
comments
-id: String
-username: String
-content: String
-date: date
-replyComments: comments
-history: List
-editComment()
-retrivehistory()
-getsubcomments()
Lost&found
-location:String
-picture:String
-contactInfo:String
-generateMap()
-getContactInfo()
groupPost
-content: String
-findfiles()
Chat
-messages: post
poll
-message:String
-options
-getpoll()
group
-owner: member
-admin: member
-picture: String
-title: String
-description: String
-members:List
-posts:List[posts]
-changeOwner()
-setAdmin()
-sendInformation()
1
N
1
N
N
1