# SET09103

# Advanced Web Technologies Coursework 2

# Messenger - Development Report

Written by: Piotr Kubicki

Date: 30/11/2016

## Table of Contents

## Introduction

The purpose of this project is to develop online web communicator that allow users to send messages between them. Application should work in real-time. Program must provide login facility to give users possibility to access their own accounts. Users should not be able to send messages to people outside of their contact list, to prevent amount of spam that may be send through the application. User should have possibility to send invitations to the people outside of their list to allow them to add new contacts to their list. Number of invitations send to one person should be limited to prevent amount of spam. User should also have possibility to block other users. Application should also provide searching facility that will display users matching search query. Program should keep a record of communication between users to allow users read messages previously send.

## Requirements

### Functional

Application must:

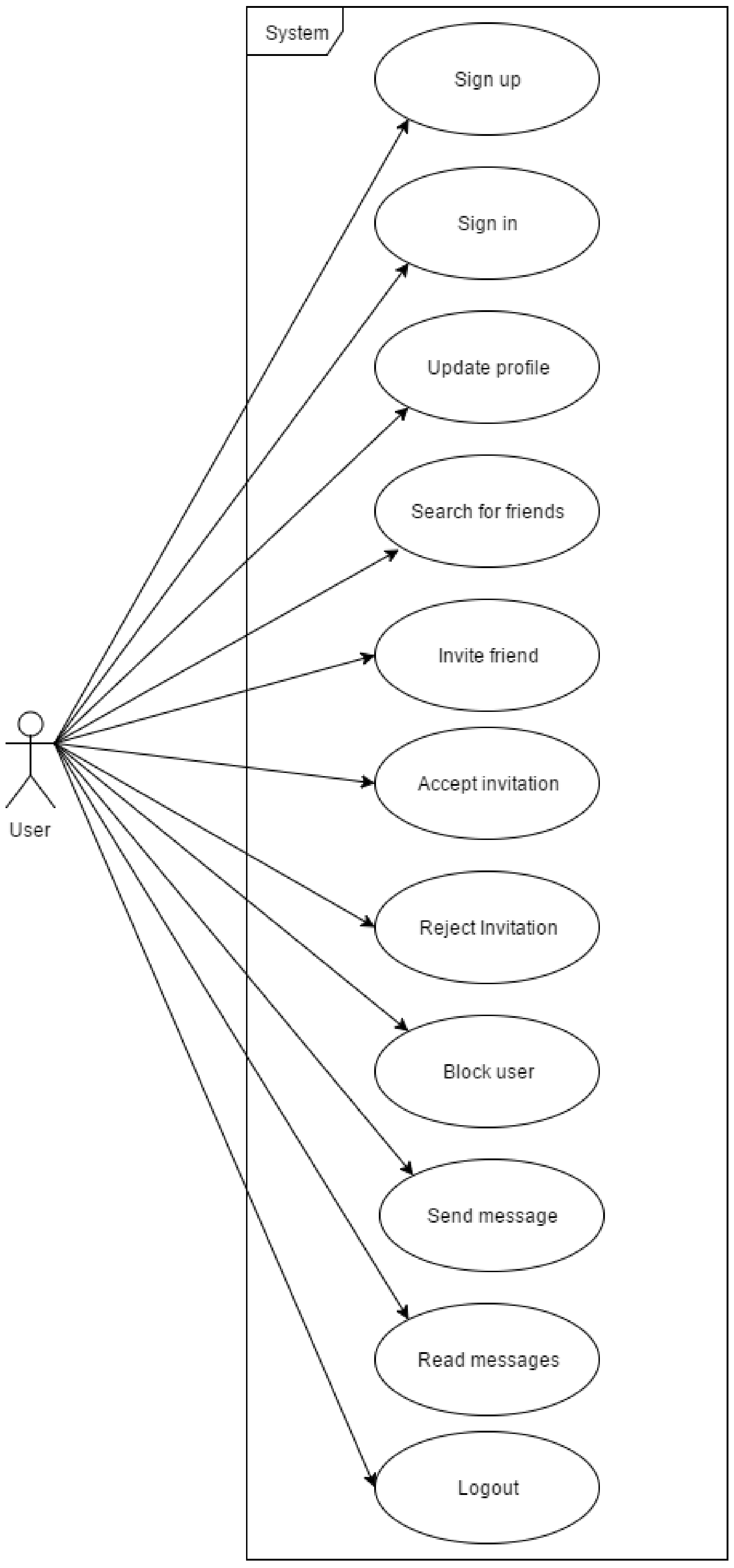
- Allow user to signup
- Allow user to login
- Allow user to find new contacts using searching facility
- Allow user to send invitations
- Allow user to accept or reject invitation
- Allow user to block another user
- Allow user to send message to another user from contact list
- Communicate contact account status (Online, Offline)
- Keep communication records
- Inform user about new messages
- Communicate errors

### Non-functional

- Must be finished before deadline (30/11/2016)
- Use appropriate colour scheme
- User friendly

## Design Plan

### Use case diagram



### Use cases scenarios

User begin conversation with friend

**Actors:** User1, User2

**Use case overview:** User1 want to begin conversation with User2

**Trigger:** User1 send message to User2

**Precondition 1:** Server is up and running

**Precondition 2:** User1 is logged in

**Precondition 3:** User2 is logged in

**Precondition 4:** User1 is friend to User2

**Basic flow:** Begin conversation between User1 and User2

**Description:** In this scenario Users are friends and are both logged in to the application.

1. User1 send message to User1 and User2 common room.
2. Server received message, saved it to file and emit it to User1 and User2 common room.
3. Clients of both users received message and displayed it in conversation box

**Termination outcome:** Any user close conversation window.

## Backend design

### Database design and data storage

#### Users

Users table will store id, full name, username, avatar, password hash, type and date of birth. Date of birth is stored to help other users identify their friends. Type row will be used to identify account type like normal user or admin.

#### Rooms

Rooms table will store room id, first user id, second user id, and room name. This table will be used to track relations between users, and store the name of the room they use to exchange messages.

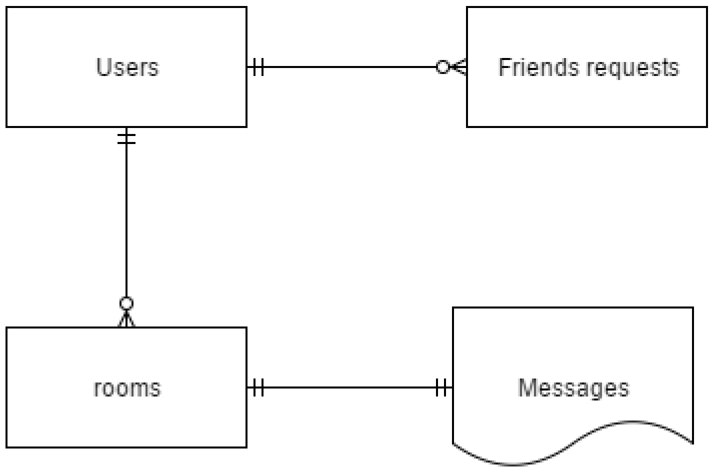
#### Friend requests

Friend requests table will store request id, sender id, receiver id and status flag. Status will be used to check if request has been seen by the target user and track their responses. In addition, if user reject three request from another user. User who send those requests won't be able to send more requests to that user.

#### Messages

In addition to database, application will also store users messages in JSON files. Every group and room will have accompanying file named the same as room. Those files will hold conversation history between users and groups. To help store them more secured, those files will be encrypted. Every message stored in file will contain following data, user id, message id, date, message and read flag. Read flag will be used to check if receiver view the message already.

## Database diagram



## Application requirements

In order provide real-time functionally this application must use additional module to support Flask framework. This additional module used in Messenger application is Flask-SocketIO. Another important software used here is socket.io that support real-time connections on client side. Unfortunately, in order to install and store socket.io locally Node.js package manager is required. After many tries to make Node.js working on Levinux platform, including tries to build it from the source files, it does not work. Only alternative found require to use content delivery network (CDN). That is why this application in order to work require internet connection all the time.

## Application functionality

### Sign up

When user create new account application must be able to store provided password securely. To do that application use bcrypt module that generate password hash and add salt to it to improve its security. That password hash together with added salt is then stored in database. In case if an unauthorised person get access to the passwords stored in database it will not be able to retrieve real passwords.

### Sign in

When user tries to login and get access to account must be authenticated. To do client send user username and password to the server. On server side application look for the password assigned to the given username and use bcrypt to encrypt just received password with salt stored in user password from database and compare resulting hash with the one taken from database. If both hashes are the same, application generate token. Token contain username, time when was created and token key generated using universally unique identifiers (uuid). Token are stored in tokens file located in src/var/tokens/tokens.json. Newly created token is added to the tokens file also token key and username are stored in session giving user possibility to get authorised access to the application.

### User authentication

In order to use application user must be logged in or have valid token key. Every time user request any action that require authentication server will take username and token key stored in session and look for matching token in tokens.json file. If matching token will be found, user will get access to the required functionality otherwise will be informed that re logging is required and will be redirected to the login page. During the process of token matching function will also remove expired tokens found. Every token is valid for 10 minutes but token can be refreshed by user when any action that require authentication was triggered. That means every time when user will send new message, search for friends or send friend request token time value will be updated to current time. From the other side when user go away without logging out from the app or stop using them for any other reason, token will expire and be deleted. When that happens user will receive information that for security reason re logging is required and will be redirected to the login page.

### Connecting

When user connect to the application connect listener is triggered and function associated with then is executed. This function retrieved all rooms associated with the user from database and place user in each of them. It also emits signals to each user from friends list to inform them that is now online. When friends' client receive signal from the user it will change user status to online and emit signal back to the server requesting to inform that friend is online as well. When server receive returning signal it will emit another signal to the user. When users' client receive that signal it will mean that friend is online and will indicate that information to the user.

In this event it is also important for the server to check if user do not have any awaiting friend request or un readied messages.

To check if any friend request awaiting to be answered, server will look into database for requests associated with the user and if any un answered request was found, it means have status equals 0, it will emit signal with payload containing first un answered request.

To check if user have any messages that awaiting to be seen, application look for all messages files associated with the users' rooms and look for any message with status 1. If that occurs, user is added to the list. When all files are checked, list of users who tries to contact with the user is send.

### Disconnecting

When user disconnecting from the application either by logging out or for any other reason, it is important to let all friends that user is offline.

To do that application will retrieve list of all friends and emit appropriate signal to each of them.

In this stage it was found that application can sometimes create recursive call of disconnect event when user tries to emit signal inside disconnect function. To prevent that it is important to make sure that user leave all rooms where signal will be emitted to first and that includeSelf argument for emit function has been set to False

### Sending/receiving message

One of the most important function that is available in this application is possibility to send and receive messages. Application also store all messages to allow users read them when required.

When user want to send any message to another user, client must emit signal to the server containing message and their recipient. Server take this data and look for the room associated with sender and recipient of the message as well as json file. When those are found server saves message into json file with additional data such as sender username and time when message was send and emit signal containing correctly formatted message to the room.

When client receive information about new message it displays message in appropriate box if exists and making associated user plate to blink informing user that new message arrived. However, if user have associated messages box opened and active at this time user plate will not change their behaviour.

### Sending/receiving friend requests

All users must have possibility to send and receive friend requests.  User can only send messages to the users that appears on their friends list. To add another user to friend list user must either send friend request that will be accepted by another user or accept request received.

In order to send friend request user must know friend username or find him/her using search box. Then client take user username and send it to the server that do multiple steps to ensure that user have rights to bother another user with friend requests. Friend request is valid when username is not same as recipient username, recipient do not have awaiting request from that user already, recipient did not reject three requests in the past or recipient did not blocked requests from that user.

Valid request is next saved into database and appropriate signal emitted to the recipient who will receive it immediately is online or as soon as show up in application. Request will be displayed to the user as long as user do not select one of three options.

When server receive answer it will check if user do not have any more awaiting request and send one at the time until no more exists. At this stage user must select any option to stop be bothered by request but in the future it would be recommended to add option that allow to add request into a waiting list where user can select request that will be answered at any convenient time later.

## Frontend design

### Colours

Application make use of different shades of purple colour, colour scheme has been generated using Adobe Kuler app. According to article found on color wheel pro website (2015), this colour is associated with royalty and symbolise power, luxury and ambition.

In addition to purple login and signup pages will contain gold colour used in background to support them and highlight its royalty.

In most cases fonts displayed over purple background will be in white colour to make them more readable. Messages from the other side will be written in black over white background to make them more friendly for the user's eyes.

#### Colours codes

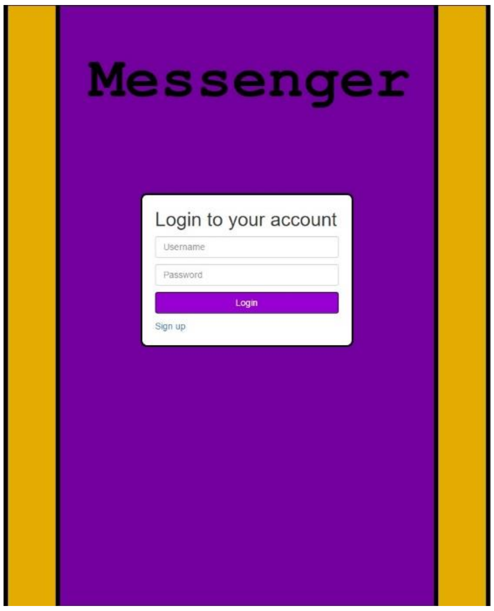
| Purpose | RGB | Hex | |
|---|---|---|---|
| Main | 153 0 210 | #9900D2 | 🟪 |
| | 60 0 83 | #3C0053 | 🟪 |
| | 176 65 218 | #B041DA | 🟪 |
| | 67 25 83 | #431953 | 🟪 |
| | 116 0 159 | #74009F | 🟪 |

## Screens design

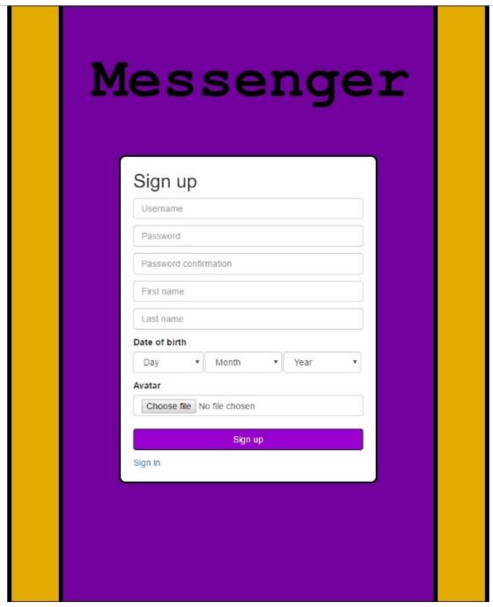
Code used to change default colours of Bootstrap4 navbar has been found in Stackoverflow (2013) online forum and can be found inside src/static/css/navbar.css file.

## Screens

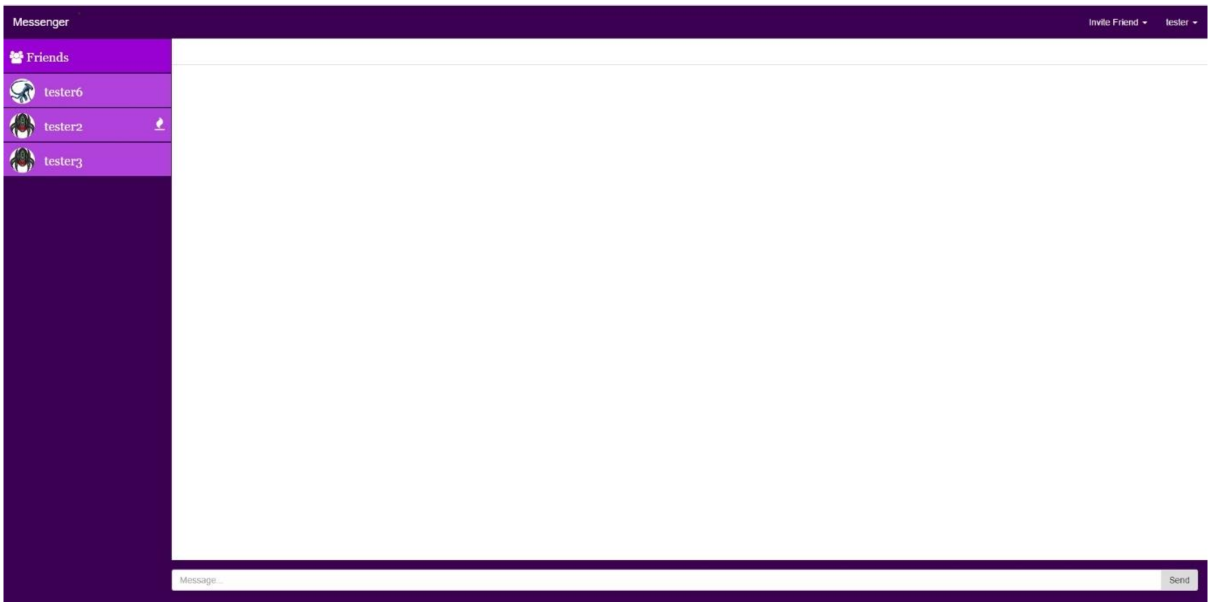
### Login screen



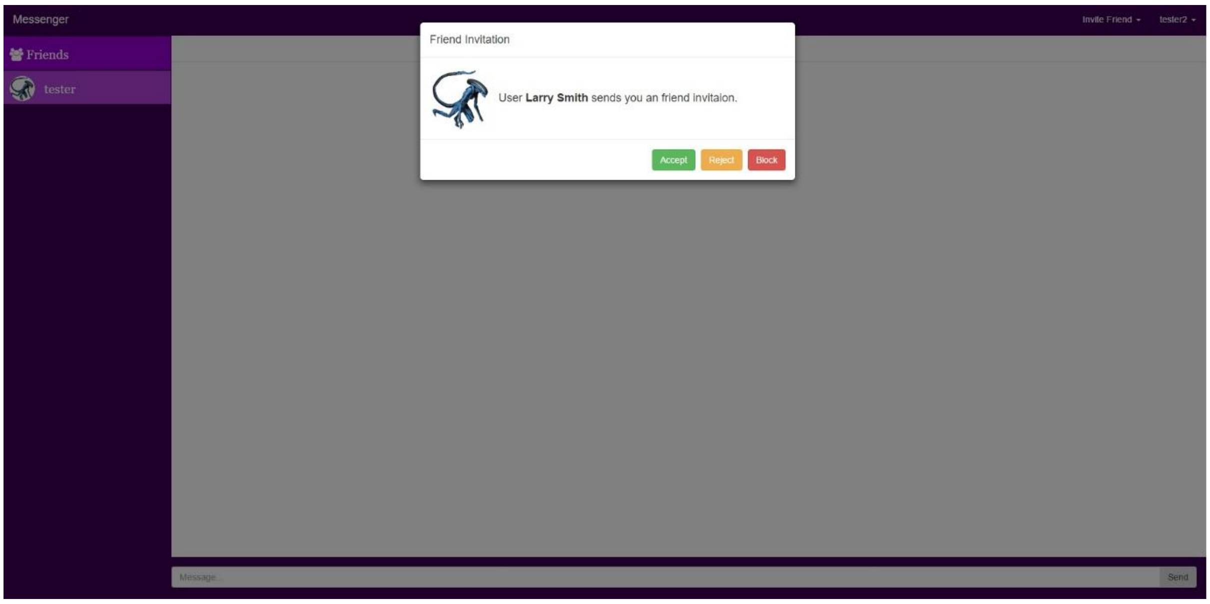
### Signup screen



## Dashboard

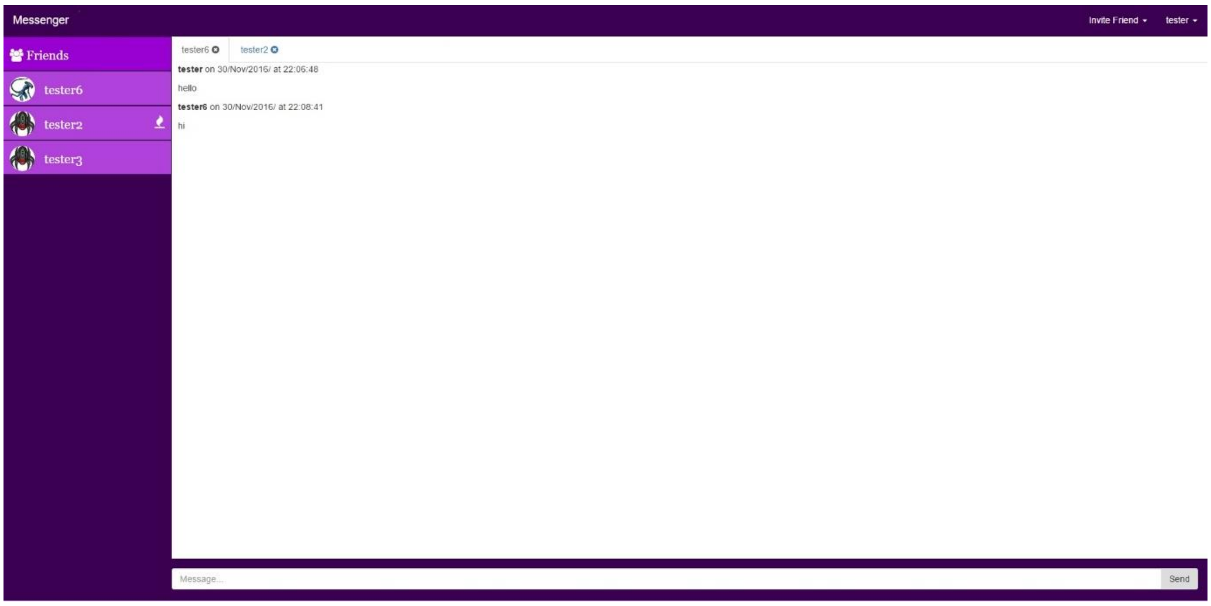


## Friend invitation



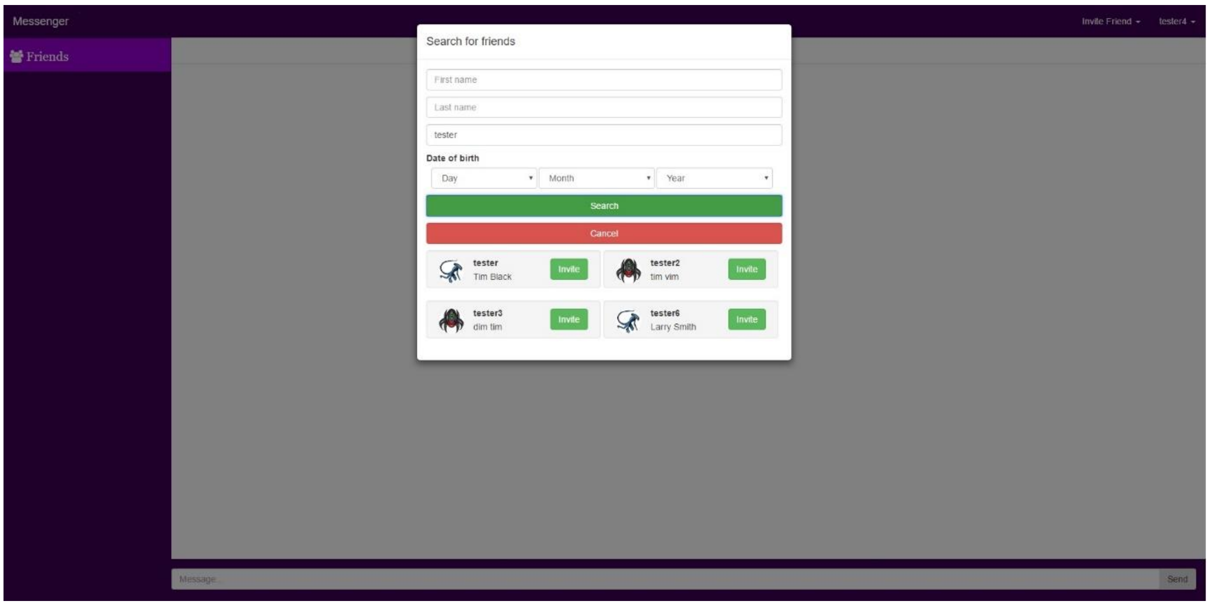
User **Larry Smith** sends you an friend invitaion.

## Tabs



**tester** on 30/Nov/2016/ at 22:06:48
hello
**tester6** on 30/Nov/2016/ at 22:08:41
hi

## Search



tester

| | |
|---|---|
| **tester** Tim Black | **tester2** tim vim |
| **tester3** dim tim | **tester6** Larry Smith |

## Tests

### Test plan

| ID | Test Case | Test Data | Expected Results |
|---|---|---|---|
| 1 | Crete new account using valid data | Username: tester6<br>Password: password<br>Password confirmation: password<br>Date of birth: 11/10/1990<br>First name: Larry<br>Last name: Smith | Account created successfully |
| 2 | Create new account using invalid data | Username: tester6<br>Password: password<br>Password confirmation: password<br>Date of birth: 11/10/1990<br>First name: Larry<br>Last name: Smith | Error message for already used username |
| 3 | Login using valid credentials | Username: tester<br>Password: password | User dashboard displayed |
| 4 | Login using invalid credentials | Username: tester<br>Password: pasword | Application redirected to login page, error message displayed |
| 5 | Change profile first name | New first name: Tom | First name changed |
| 6 | Change profile last name | New last name: Black | Last name changed |
| 7 | Change profile date of birth | New D.O.B: 21/12/1989 | Date of birth changed |
| 8 | Change profile avatar | Any .jpg .png file | User avatar changed |
| 9 | Invite friend using quick invite | Friend username: tester6 | Invitation send to tester6. User tester6 should be able to see it |
| 10 | Find friend using search box | User first name: Tim | User found |
| 11 | Send friend request to offline user | Any offline user | Invitation send. User should be able to see it when log in |
| 12 | Invite friend found using search box | Search friend username: tester<br>Invite any friend found | Invitation send. User should be able to see it when online |
| 13 | Invite user who is already your friend using quick invite | Friend username: tester6 | Error message displayed |
| 14 | Send friend request to yourself | n/a | Error message displayed |
| 15 | Accept friend invitation | n/a | New friend should be displayed in friend list |

| | | | |
|---|---|---|---|
| 16 | Refresh page displaying friend invitation | n/a | Friend request should be displayed again |
| 17 | Reject friend request | n/a | Friend request should gone |
| 18 | Reject three invitations from one user | n/a | User should not be able to send any more friend requests to us |
| 19 | Block user who send you friend request | n/a | User should not be able to send any more friend requests to us |
| 20 | Online user status | n/a | Online status should be show inside user box when online |
| 21 | Offline user status | n/a | Offline status should be show inside user box when offline |
| 22 | Open friend tab | n/a | New tab with username should appear |
| 23 | Switch between multiple friend tabs | n/a | Displayed messages should change to display conversation with current user |
| 24 | Close friend tab | n/a | Conversation with user should be closed |
| 25 | Send message to online friend who not talking with him/her at the moment | n/a | New message should appear inside conversation box. Friend should see sender plate blinking |
| 26 | Send message to online friend who talking with him/her at the moment | n/a | New message should appear inside conversation boxes of both users |
| 27 | Send message to offline friend | n/a | New message should appear inside conversation box. Friend should see sender plate blinking when log in to application. |
| 28 | Click on blinking friend plate | n/a | Plate should stop blinking. Messages should appear inside messages box |
| 29 | Read messages history | n/a | Application should display all messages |
| 30 | Logout | n/a | Login screen displayed |
| 31 | Access page that require authentication when not authenticated | URL: '/' | User should be redirected to login page |
| 32 | Press back button after logout | n/a | User should be redirected to login page |

## Test results

| ID | Test Case | Test Data | Expected Results | Actual Results | Status | Actions |
|---|---|---|---|---|---|---|
| 1 | Crete new account using valid data | Username: tester6 Password: password Password confirmation: password Date of birth: 11/10/1990 First name: Larry Last name: Smith | Account created successfully | As expected | Pass | |
| 2 | Create new account using invalid data | Username: tester6 Password: password Password confirmation: pasword Date of birth: 11/10/1990 First name: Larry Last name: Smith | Error message for already used username | As expected | Pass | |
| 3 | Login using valid credentials | Username: tester Password: password | User dashboard displayed | As expected | Pass | |
| 4 | Login using invalid credentials | Username: tester Password: pasword | Application redirected to login page, error message displayed | As expected | Pass | |
| 5 | Change profile first name | New first name: Tom | First name changed | As expected | Pass | |
| 6 | Change profile last name | New last name: Black | Last name changed | As expected | Pass | |
| 7 | Change profile date of birth | New D.O.B: 21/12/1989 | Date of birth changed | As expected | Pass | |
| 8 | Change profile avatar | Any .jpg .png file | User avatar changed | As expected | Pass | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 9 | Invite friend using quick invite | Friend username: tester6 | Invitation send to tester6. User tester6 should be able to see it | As expected | Pass | |
| 10 | Find friend using search box | User first name: Tim | User found | As expected | Pass | |
| 11 | Send friend request to offline user | Any offline user | Invitation send. User should be able to see it when log in | As expected | Pass | |
| 12 | Invite friend found using search box | Search friend username: tester Invite any friend found | Invitation send. User should be able to see it when online | As expected | Pass | |
| 13 | Invite user who is already your friend using quick invite | Friend username: tester6 | Error message displayed | As expected | Pass | |
| 14 | Send friend request to yourself | n/a | Error message displayed | As expected | Pass | |
| 15 | Accept friend invitation | n/a | New friend should be displayed in friend list | As expected | Pass | |
| 16 | Refresh page displaying friend invitation | n/a | Friend request should be displayed again | As expected | Pass | |
| 17 | Reject friend request | n/a | Friend request should gone | As expected | Pass | |
| 18 | Reject three invitations from one user | n/a | User should not be able to send any more friend requests to us | As expected | Pass | |
| 19 | Block user who send you friend request | n/a | User should not be able to send any more friend requests to us | As expected | Pass | |
| 20 | Online user status | n/a | Online status should be show inside | As expected | Pass | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | user box when online | | | |
| 21 | Offline user status | n/a | Offline status should be show inside user box when offline | As expected | Pass | |
| 22 | Open friend tab | n/a | New tab with username should appear | As expected | Pass | |
| 23 | Switch between multiple friend tabs | n/a | Displayed messages should change to display conversation with current user | As expected | Pass | |
| 24 | Close friend tab | n/a | Conversation with user should be closed | As expected | Pass | |
| 25 | Send message to online friend who not talking with him/her at the moment | n/a | New message should appear inside conversation box. Friend should see sender plate blinking | As expected | Pass | |
| 26 | Send message to online friend who talking with him/her at the moment | n/a | New message should appear inside conversation boxes of both users | As expected | Pass | |
| 27 | Send message to offline friend | n/a | New message should appear inside conversation box. Friend should see sender plate blinking when log in to application. | As expected | Pass | |
| 28 | Click on blinking friend plate | n/a | Plate should stop blinking. Messages should appear | As expected | Pass | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | inside messages box | | | |
| 29 | Read messages history | n/a | Application should display all messages | As expected | Pass | |
| 30 | Logout | n/a | Login screen displayed | As expected but sometimes error message displayed | Pass | Need to be investigated Low level of importance |
| 31 | Access page that require authentication when not authenticated | URL: '/' | User should be redirected to login page | As expected | Pass | |
| 32 | Press back button after logout | n/a | User should be redirected to login page | After pressing back button user moved back to dashboard but token is removed preventing user from any actions | Failed | Solution required medium level of importance |

### Web browsers tests

Application has been tested using Chrome, Mozilla Firefox, Internet Explorer and Edge web browsers and all functionality looks to work fine. However, some visual elements might by displayed incorrectly using Internet Explorer and Edge web browsers.

## Enhancements

### Administration accounts

At this moment application supports only one type of accounts however, by all that time it was designed to contain admins and super admin accounts. Those accounts would allow to provide more control over application and provide user support.

### Groups

At this moment communication is only one-to-one communication is provided but in the future application should all users to create groups and where multiple users can be invited. That will allow bigger group of users collaborate over projects and tasks, or just chit chat in bigger group.

### Files exchange

At this stage application allow to send text messages and nothing else but in the future application supposed to give users possibility to upload files as it is done by similar applications.

## Project evaluation

This application has been created to provide easy to use, secured communication tool. Program should allow users to safely exchange messages between their friends or co-workers.

Application allow users to create accounts that require minimum personal information required to keep them separated from other users and allow them to be found be their friends. Program stores user first and last name together with date of birth and avatar image that might be but not must their personal photography. At this moment application stores this data in unchanged form but in the future those should be encrypted to improve security. However, application do not gather any data that could be used to contact with user by any other way then through the application.

Colours scheme used within that app my not suite all users. It also looks bit different depends from screen.

To ensure that user have rights to use requested application functions, program generate token key that is stored by certain amount of time on server and session cookie file. If application will be left unused and token expire user will be redirected to the login page and asked to login again. That solution may provide some inconvenience in user experience but is necessary for security reason.

This application is unfinished and contain many areas with room to improvements but considering all factors such as new unknown technologies used, time given for development, concurrent projects development and many more, can be considered as successfully approach. Basic functionality that was defined at the beginning has been developed and seems to work fine however, more tests with remote server and bigger group of users using application in the same time is required.

## Personal evaluation

Work over this application required lot of time spend on planning and research. In order to provide real-time functionality, it was required to undertake detailed research and lot of testing. Without those this project could fail in later stage when undertaking completely new task would be not possible because of time constrain. To find required information and solutions many software documentation has been looked such as flask-socketio, jinja2, socket.io and python.

Successfully most from the planed functionality has been provided and tested on time. Every idea that did not work has been replaced with another often better idea created by using experience taken from previous failures.

Time during the development has been well managed between multiple projects that have has to be undertaken simultaneously and personal life without taking unnecessary compromises.

Summarising work over this project was great fun. It allowed to develop current skills as well as gain many new including use of new technologies and combining them with already possessed knowledge.

## Resources

Flask-socketio - https://flask-socketio.readthedocs.io/en/latest/

Font-awesome - http://fontawesome.io/

JQuery - https://jquery.com/

Bootstrap3 - http://getbootstrap.com/

Python 2.7 - https://docs.python.org/2.7/

Adobe Kuler - https://color.adobe.com/create/color-wheel/

## References

Color-wheel-pro. (2015). Color meaning. Retrieved November 30, 2016 from http://www.color-wheel-pro.com/color-meaning.html

zessx. (2013, 30). Change navbar color in Twitter Bootstrap3. Retrieved October 20, 2016 from http://stackoverflow.com/questions/18529274/change-navbar-color-in-twitter-bootstrap-3