

## Splash



## Login Screen



Application login page. Requirements:

- Validate all entered data. Min pass length - 4 symbols.

For user login send POST request to `http://54.76.2.13:9998/auth/login` with following parameters:

`email = "user email", password = "user password"`

In case of successfully login service returns JSON with user data:

```
{
  user_id: 11
  email: "user@user.com"
  name: "User"
  pass: "12345678"
  age: 19
  visibility: 1
  profile_pic: "http://54.76.2.13:9998/avatar/bSo35UrA.png"
}
```

## Signup screen



Enter your info:



Name:

Email

Password

Re-enter password

Application signup page. Requirements:

Validate all entered data. Min pass length - 4 symbols.

For user signup send POST request to `http://54.76.2.13:9998/auth/signup` with following parameters:

email = "user email", name = "user name", password = "user password", age = "user age"

In case of successfully registration service returns id of new user in response.

Response sample:

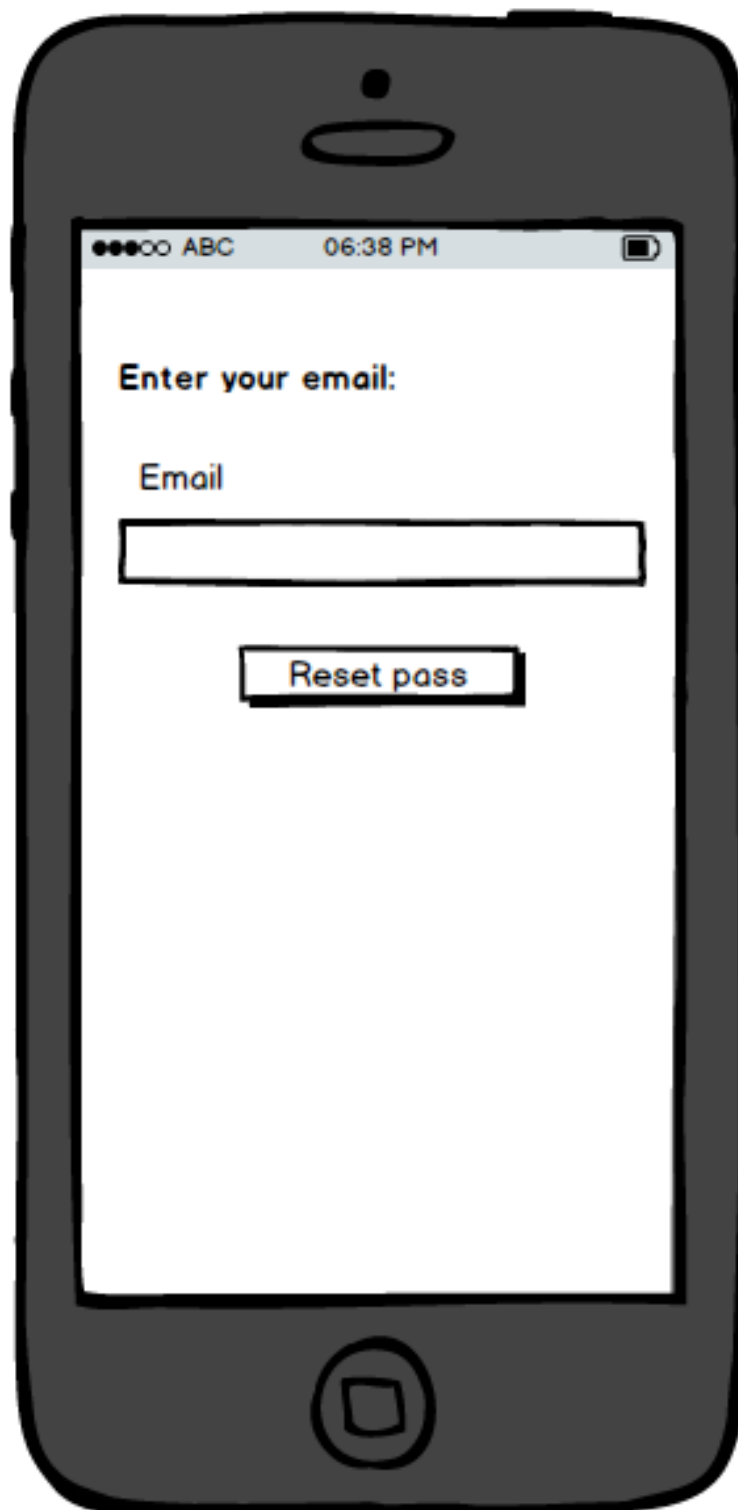
```
{
  user_id: 20
  email: "test@user.com"
  name: "Test user"
  pass: "pass"
  age: 25
  visibility: 1
  profile_pic: ""
}
```

If user has selected photo send POST request to `http://54.76.2.13:9998/user/edit-user-photo`:

userId = "user id", file = "file with new photo"

After successfull signup show popup "Congratulations!!! You have successfully signed up!"

## Forgot pass screen

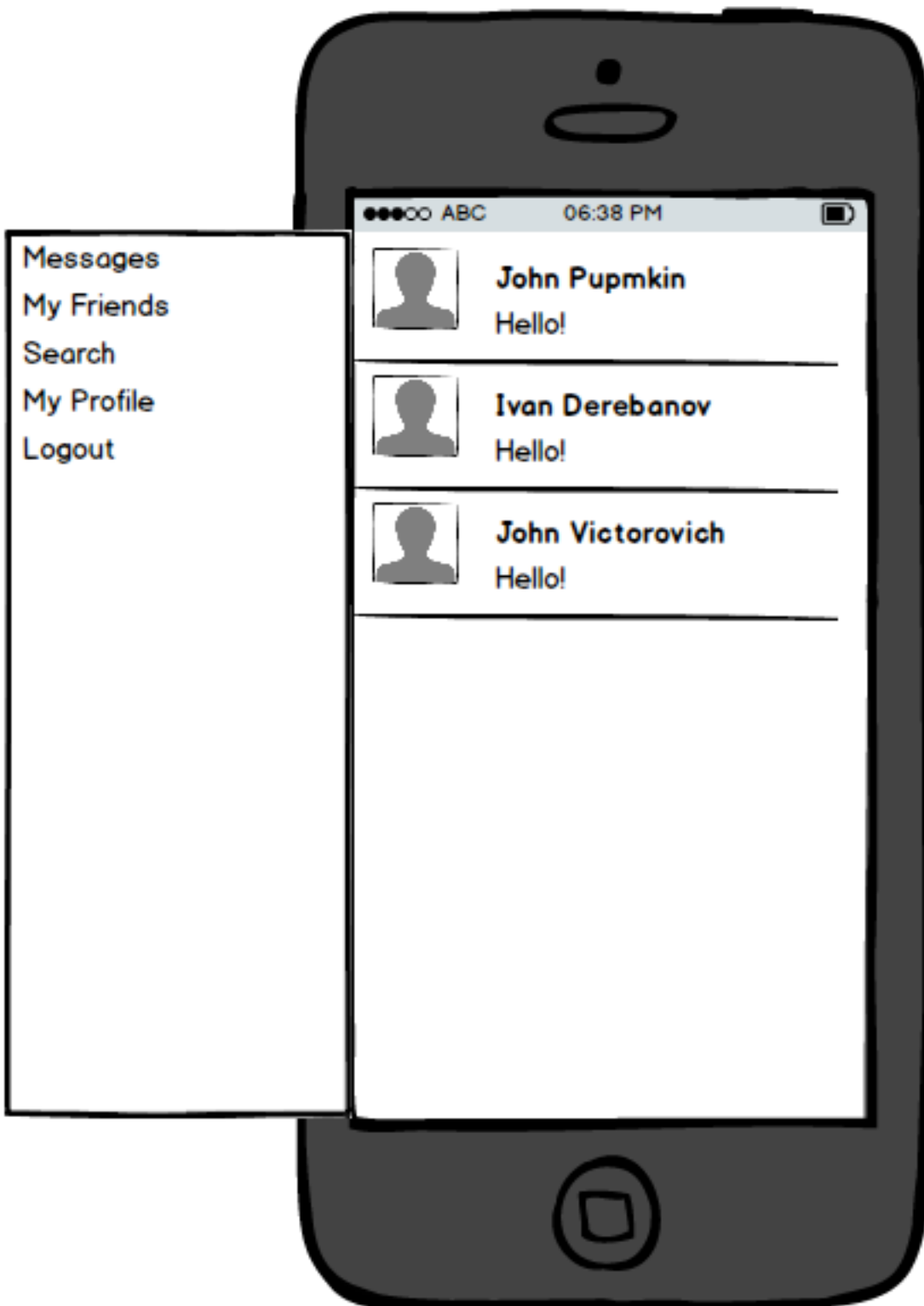


For user signup send POST request to <http://54.76.2.13:9998/auth/reset-pass> with following parameters:  
email = "user email"

In response you will receive new user password:

```
{  
  new_pass: "tc50fp"  
}
```

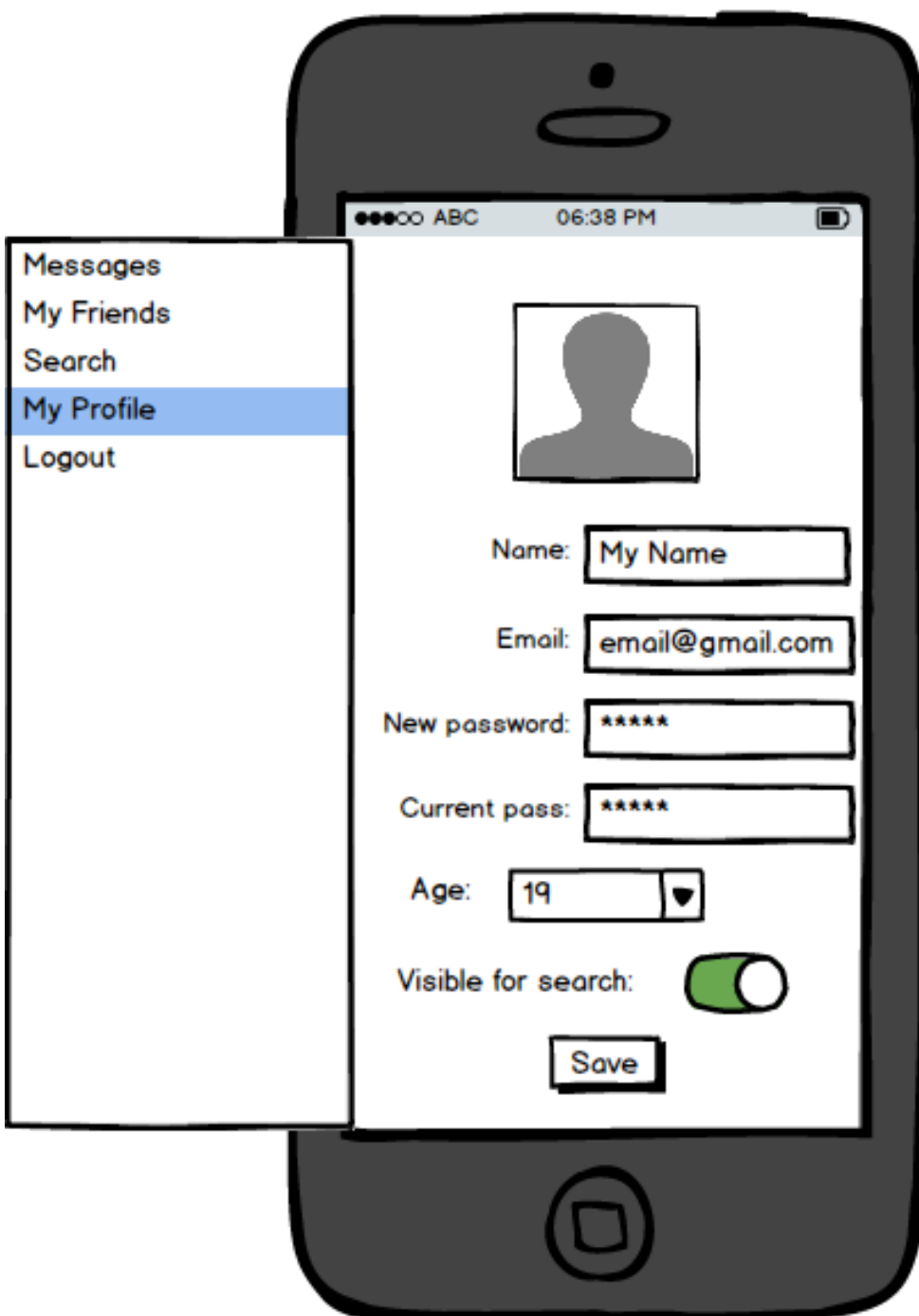
## Sliding menu



Add Sliding menu navigation, that can be accessed by left swipe.  
Also, user can be redirected on Login page if press Logout button.  
Be sure that menu can be accessed from any page that listed in it.

App will start from messages page, create UI without functionality for now.

## My Profile



User profile page.

If user clicks on picture, show popup that allows to choose image from gallery or take new photo from camera.

Fields are editable. Click on "Age" opens age picker.

Allow user to enter new pass only if he entered current password.

To save entered data user need to click on "Save" button.

Services:

1. Implement receiving user profile data from server. Send GET request to `http://54.76.2.13:9998/user/{id}` (id - id of current logged user)

Response example:

```
{
  user_id: 20
  email: "test@user.com"
  name: "Test user"
  pass: "tc50fp"
  age: 25
  visibility: 1
  profile_pic: ""
}
```

Show user data on views.

2. To update user data send POST request to `http://54.76.2.13:9998/user/edit-user`

Params: `userId = "current user id"`, `name = "new name"`, `age = "new age"`, `visibility = "1/0"`, `email = "new email"`, `password = "old or new password"`

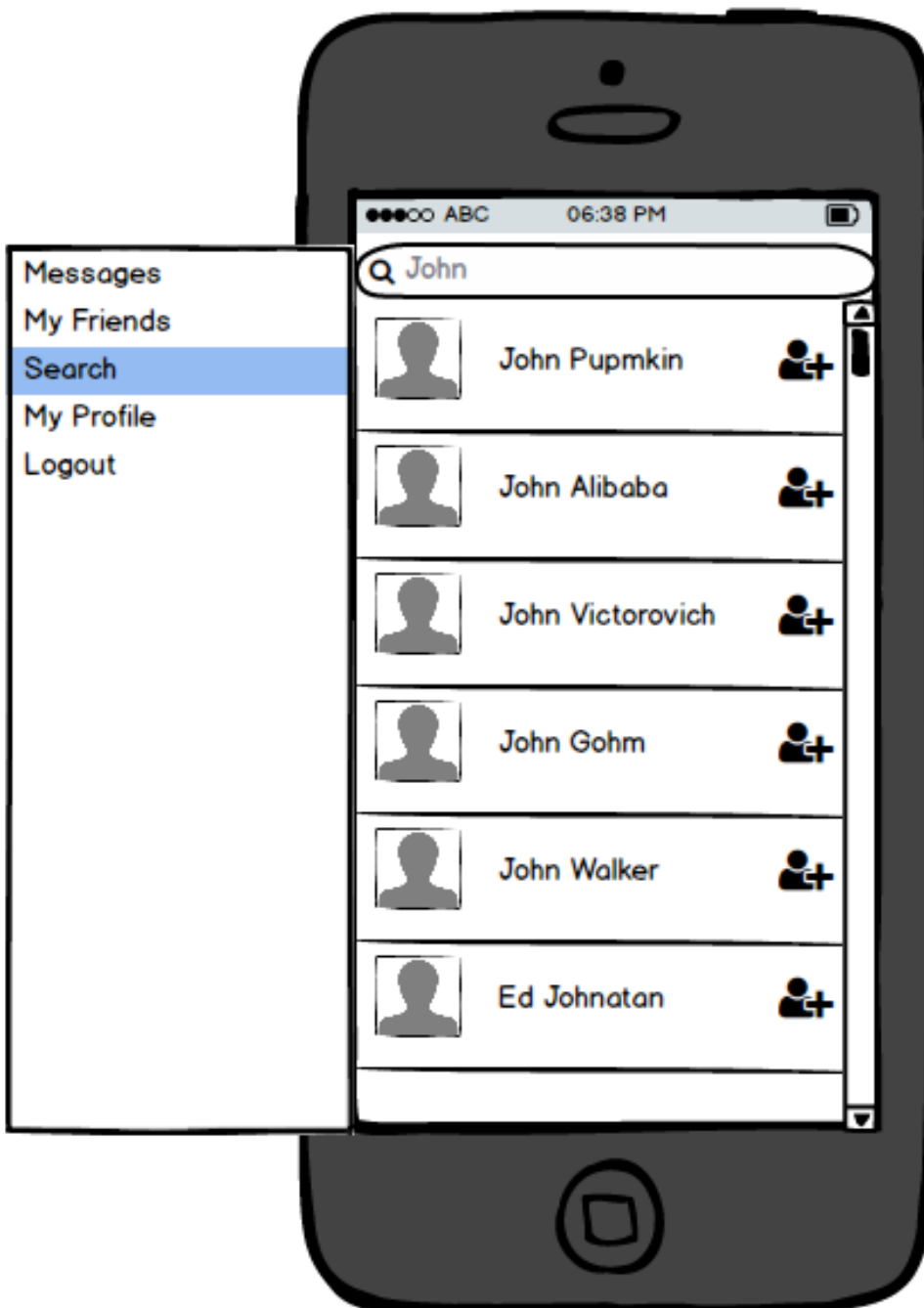
visibility 0 is used for hiding user from search

If user photo was updated send POST request to

`http://54.76.2.13:9998/user/edit-user-photo:`

`userId = "user id"`, `file = "file with new photo"`

## Add Friends page




"Search" page allows user to find other people in application and add them to Friends list.

When user enters query in Search field, send GET request:

`http://54.76.2.13:9998/friends/search?id="id of current user"&query="search query"`

Send empty query to get all users.

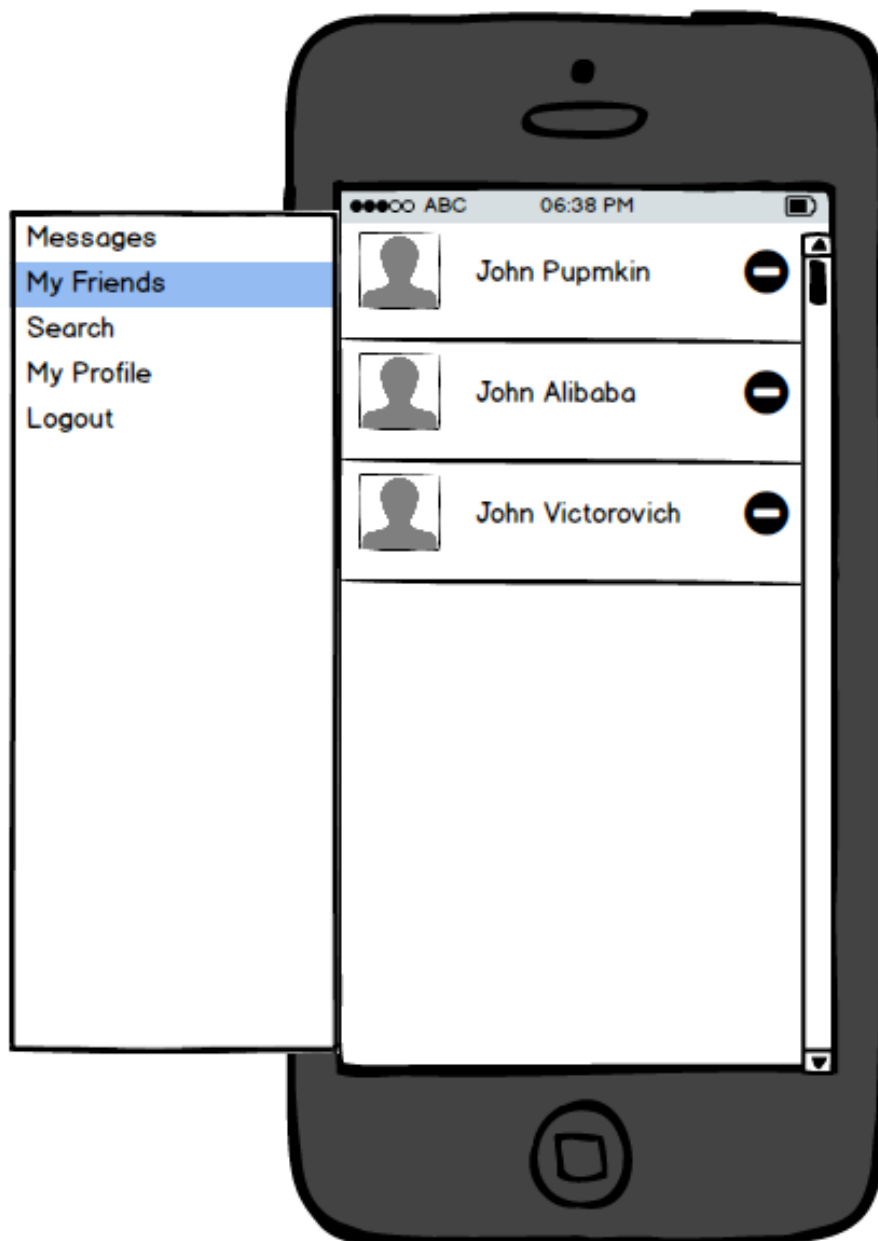
There will be list of found friends in response, that need to be displayed below Search widget.

Click on  will send POST request to `http://54.76.2.13:9998/friends/add` with following parameters:

`id="id of current user", friendId="id of user that you want to add to friends"`

User will appear on My Friends page in case of success response.

## My Friends




"My friends" page shows list of users's added friends.

To get friends list from server send GET request:

`http://54.76.2.13:9998/friends/list?id="id of current user"`

Display list that will be received in request.

Click on  will send POST request to `http://54.76.2.13:9998/friends/remove` with following parameters:

`id="id of current user"`, `friendId="id of user that you want to delete from friends"`

Remove friend from list on success response.

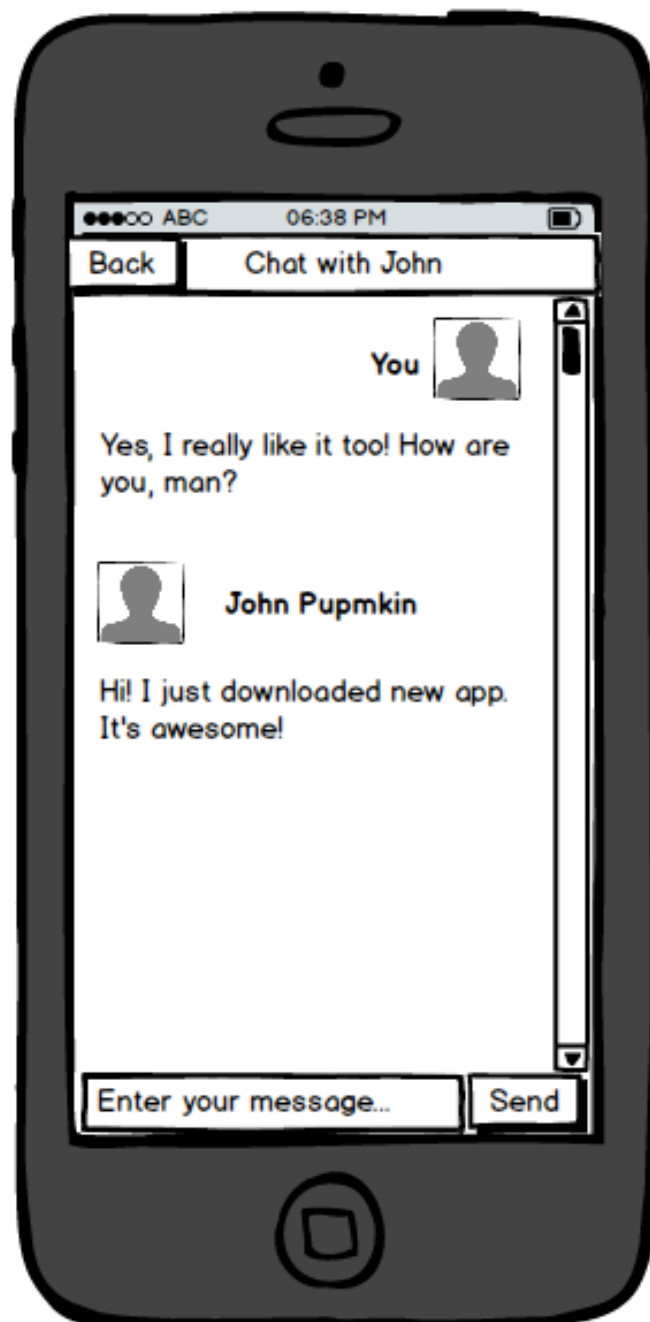
Click on element list will open chat page with specific friend.

Send GET request to `http://54.76.2.13:9998/messages/get-conversation?id="id of current user"&friendId="id of friend"`

There will be `conversationId` in response, that will be used to manage chat later.



## Chat screen



Chat page allows users to make conversations with each-others. To get old messages and show them to user, we need to make GET request to `http://54.76.2.13:9998/messages/conversation/{conversationId}`

New messages must appear at top of list.

Update messages when user opens this page, every 20 seconds,

To send message send POST request:

`http://54.76.2.13:9998/messages/send`

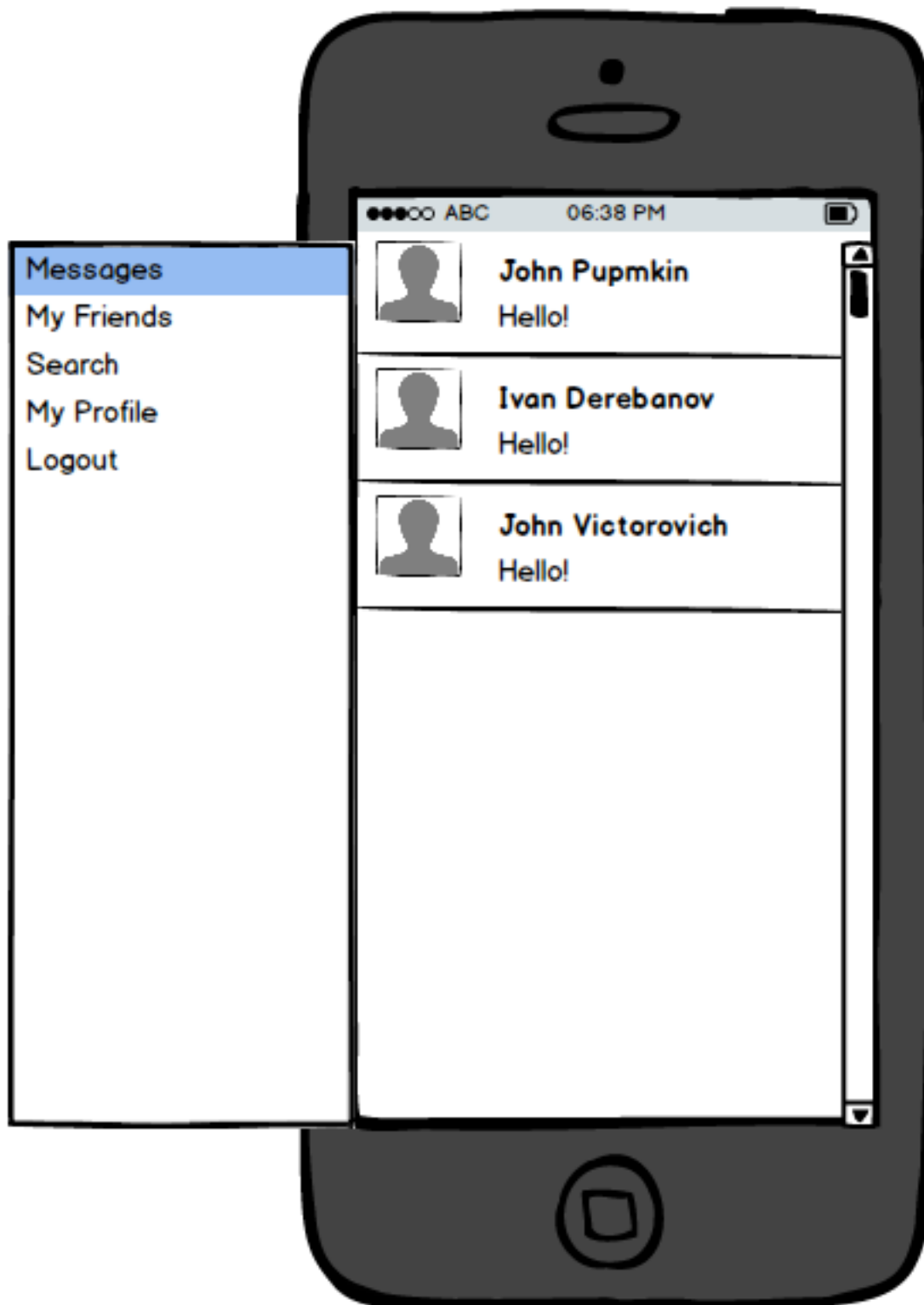
Params:

`receiverId = "id of friend",`

`senderId = "id of current user",`

`message = "some text"`

## Messages



Messages page shows all user's conversations.

To get this list send GET request:

[http://54.76.2.13:9998/messages/list?userId="id of current user"](http://54.76.2.13:9998/messages/list?userId='id of current user')

Click on list element opens Chat page with this conversation.

## Additional

1. Add social networks login possibility (Facebook, twitter). When user logs-in into social network, call <http://54.76.2.13:9998/auth/social-login> with following parameters:  
email = "user email", name = "user name", social = "true", age = "user age"  
In this case user will be saved into DB at first login by social network.

2. Add possibility to add coordinates to messages in Chat. To do this, attach MAP button and send lat="latitude", long="longitude" with message.

To send message send POST request:

<http://54.76.2.13:9998/messages/send-coord>

Params:

receiverId = "id of friend",

senderId = "id of current user",

message = "some text"

lat="latitude",

long="longitude"

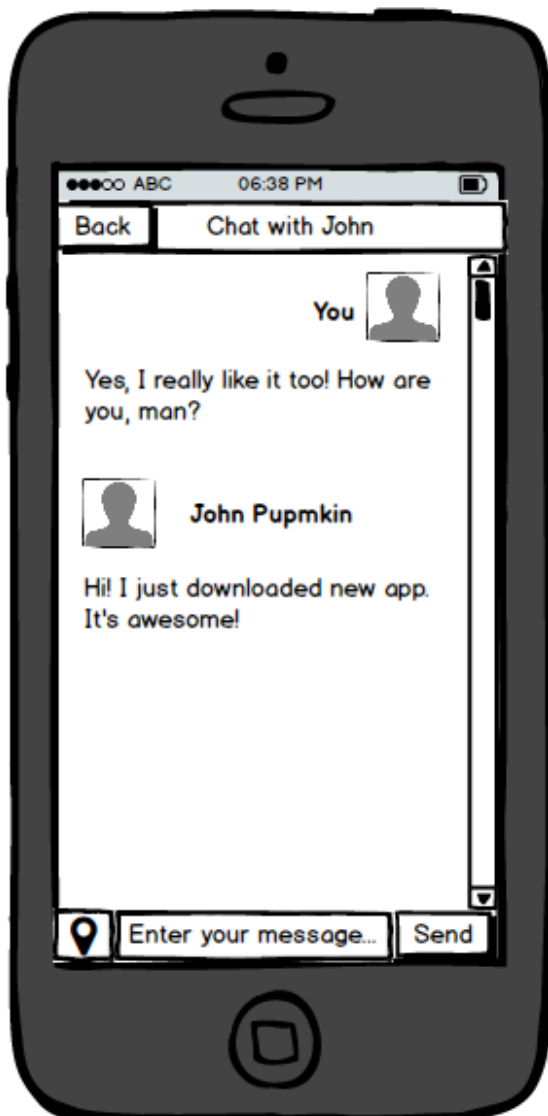
3. Add Map page into menu.

Attach Google Map there. After user changes map camera, get map center coordinates and call GET request into:

<http://54.76.2.13:9998/user/get-map-posts?userId=current user id&lat=latitude&long=longitude>

You will receive posts near this coordinates, show them as pins on map. Show popup with message after click on pin.

2



3

