6 Degrees

Design Document

CSCI 4830-- Software Engineering

Group 3-- Anna Buhman, Joshua Wiley, Douglas Martin, Justin Hamilton

Table of Contents

[1. Introduction 3](#_Toc276415782)

[1.1 Purpose 3](#_Toc276415783)

[1.2 Terms 3](#_Toc276415784)

[2. Architecture Description 4](#_Toc276415785)

[3. Decomposition Description 5](#_Toc276415786)

[3.1 Objects 5](#_Toc276415787)

[3.2 Logic 5](#_Toc276415788)

[3.3 User Interface 6](#_Toc276415789)

[4. Dependency Description 8](#_Toc276415790)

[4.1 Object 8](#_Toc276415791)

[4.2 Logic & User Interface 8](#_Toc276415792)

[5. Interface Description 10](#_Toc276415793)

[5.1 Object 10](#_Toc276415794)

[5.2 Logic 10](#_Toc276415795)

[5.3 User Interface 16](#_Toc276415796)

[6. Detailed Design 18](#_Toc276415797)

[6.1 Source File Organization 18](#_Toc276415798)

[6.2 Algorithms 19](#_Toc276415799)

[6.3 Interface Screenshots 20](#_Toc276415800)

# 1. Introduction

## 1.1 Purpose

The purpose of this report is to explain and document the design of the 6 Degrees application. 6 Degrees is an online application that consists of a user interface, logic, and database objects. The user interface allows the user to interact with the 6 Degree website (such as adding 6 Degree 'friends' and creating a connection path) as well as allowing interacting between Facebook and the 6 Degree website (such as allowing 6 Degree to access a users Facebook ). The logic portion will help allow the user interaction, performing many tasks such as connecting with Facebook to allow access as well as gathering list of Facebook friends, finding connections between 'unfriended' people, and basic navigation. The database objects will store the list of people who are friends on 6 Degree along with Facebook information if the friend was synchronized from Facebook database API; additionally, any connection paths that is created with the connection logic between two people will be stored in the database.

## 1.2 Terms

* Facebook- A website that provides a social network for users to communicate
* 6 Degree Friend - a person who is in a list on the database for a specific person defining a connection between the user and the person
* Facebook Friend - a person who is friends on facebook with the user
* Friend Collection- A collection of user objects for a specific user to link to as friends
* Friend List- A friend collection
* Connection Path- a chain of friends with the beginning and starting point user defined to see a relationship of how closely the connection is between two people who are not 6 Degree friends
* Canvas - a JavaScript area in which Dojo will be used to fill in with the interface for many pages

# 2. Architecture Description

The 6 Degree website is using javascript for almost all of the development. There are two main branches within the 6 Degree website: Node and the web portion. Node.js is a event framework written for Javascript serverside application and programs. Node.js was chosen because it can perform tasks asynchronously, meaning that processes and tasks are able to perform in a way that allows processes to be performed before another has ended. This is especially necessary for the searching algorithm to increase speed and to make sure the server does not time-out.

To run the system with the node and web portions, the Dug server was used. This is a Javascript server written by Doug Martin, one of the 6 Degree developers. A file called config.json is used to help set up the and customize the server to run 6 Degrees.

Within the Node.js system, there are other entities which include services, utilities, clients, domains, services. These are all containers for sets of objects. Services contain the auth (which is used to authenticate users), user (which is used to interact with a user and the user’s friends and information), and image (which is used to interact with the image). The services call various utilities which help interact with the domain methods and client methods. Client contains methods that are used to interact with the social networks’ APIs. Domain methods are used to interact with the two different databases set up, user and profile picture. The domain uses the external library Mongoose to interact with the MongoDB database.

The web system is developed with dojo and Raphael, both external libraries. Raphael is used to create HTML 5 graphics easily for the connection path display. Dojo is used as a framework used to develop the front end web portion.

# 3. Decomposition Description

## 3.1 Node

The services contain methods that are called to interact with the logic node system. They are the primary function called, and then call other functions in utilities, clients, and domains to perform the task.

3.2.1 Services

* auth

This file contains the methods used for authentication. It will perform checkStatus, which tests if there is currently an account logged in via the session. It contains the method createUserFromFacebook which calls utilities to create a user with Facebook information. It also has the login and logout method, which stores or deletes the session variable if credentials are correct.

* User

This file contains methods used to interact with a user or multiple users. There is a method to find a connection, send a message, get friend information, get user information, get friend list, and search for users.

* Image

This file contains methods that are used to interact with the profile image. One method is setProfilePic, which contains the logic to condition the picture to store. It also contains functions getProfilePic and get ProfileForUser to help return the picture that is stored. Lastly, two helper methods help size the image and help check mime types.

3.2.2 Util

The utilities are meant to help connect the functions of the services and the domain and clients.

* UserUtil.js

This file contains an object that holds many functions to assist with interacting with a user as well as function to save a profile picture.

3.2.3 Client

Clients provide the basic authentication and retrieval for social networks. It can be expanding by creating more files, each for a social network to interact with.

* Facebook.js

This provides a dojo class for interacting with Facebook. Methods help make an authenticate request, obtain the friend list, obtain information of the user, obtain the picture profile.

3.2.4 Domain

The domain contains models for the mongoose database.

* **User**

The database will contain data the user has entered about themselves and from Facebook as well as a hash collection of friends which are references to other users.

* **ProfilePic**

The profile picture database stores a user’s profile picture in binary within the database. This is done to allow faster transfer of data as well as fewer files.

Facebook

profilePic

user

UserUtil

User

Image

auth

## 3.2 Web

The user interface allows the user to interpret what is going on in the logic and database object portions of the program as well as to issue commands.

4.3.1 Login

This is the page where a person must log in.

* Login.html

This provides the basic form for the user to log on in html.

* Logon.js

This provides the interaction between the logic as well as creating the look and interaction the user has with the graphics.

4.3.2 User’s home page

This is a web page that shows the user’s data.

* index.html

Provides a basic html document that a canvas can be inserted for the user to see the graphical interface which is provided by other JavaScript files below.

* profile.js

Generates the JavaScript that is fed into the index.html. It will generate the menu, graphics, links, and all parts of the gui system.

4.3.3 User’s friends

This shows the user’s friend list

* home.html

Provides a basic html document that a canvas can be inserted for the user to see the graphical interface which is provided by other JavaScript files below.

* user.js

Creates the list of friends

* profile.js

Generates the JavaScript that is fed into the index.html. It will generate the menu, graphics, links, and all parts of the gui system.

4.3.4 Connection

This shows the connection path that the user has asked the program to create.

* home.html

Provides a basic html document that a canvas can be inserted for the user to see the graphical interface which is provided by other JavaScript files below.

* connection.js

This file calls the connection path logic and then displays it in the home.html file canvas.

# 4. Dependency Description

## 4.1 MongoDB

The user object and the profile picture are the only storage objects; therefore, the only dependencies are the user object friend collection, which points to other user objects as well as a user profile picture depen

User 5

User 2

User 1

User 3

User 4

userId

User

ProfilePic

## 4.2 Node

Many of the Node files call each other to perform a specific task, which is very common in object-oriented programming. Because of that, below is a list of the files with files that it calls methods in as well as files that call it in methods.

|  |  |  |
| --- | --- | --- |
| File | Calls | Is Called By |
| User.js | UserUtil.js | None |
| Auth.js | UserUtil.js | None |
| Image.js | userUtil.js | None |
| UserUtil.js | Facebook.js, User.js, profilePic.js | User.js, Auth.js, Image.js |
| Facebook.js | None | UserUtil.js |
| User.js | None | UserUtil.js |
| profilePic.js | None | UserUtil.js |

## cdraw.png

# 5. Interface Description

## 5.1 Node

**Services**

5.1.1 user.js--services for the user

Internal parameters--

dojo - relative url of dojo

UserDAO - relative url of User.UserDAO

MessagesDAO - relative url of Messages.MessagesDAO

fb - relative url of Facebook.FacebookClient

APP\_ACCESS\_TOKEN - helps keep the page secure

APP\_SECRET - helps keep the page secure

API\_KEY - helps keep the page secure

userDAO - an object of type UserDAO

messagesDAO - an object of type MessagesDAO

client - an object of type fb

exports.findConnection

**Input**

target - the end goal

req - path to functions

res - path to functions

exports.sendMessage

Input

to

message

req

res

exports.getInfo

**Input**

req - path to functions

res - path to functions

exports.getFriends

**Input**

name - name of friend

req - path to functions

res - path to functions

5.1.2 auth.js-- authenticate and creates new users

Internal parameters--

dojo - relative url of dojo

UserUtil - relative url of User.User

formidable - relative url to the formidable library

sys

fs

child\_process

exports.checkStatus

**Input**

req - path to functions

res - path to functions

exports.createUser

**Input**

user

req - path to functions

res - path to functions

exports.login

**Input**

email

password

req - path to functions

res - path to functions

exports.logout

**Input**

req - path to functions

res - path to functions

5.1.3 image.js-- interacts with the image

Internal parameters--

dojo - relative url of dojo

UserUtil - relative url of User.User

formidable - relative url to the formidable library

sys

fs

child\_process

exports.setProfilePic

**Input**

req - path to functions

res - path to functions

exports.getProfilePic

**Input**

id

req - path to functions

res - path to functions

exports.scale

**Input**

image

mime

width

height

callback

exports.mime

**Input**

mime

**UserUtil**

5.1.4 UserUtil.js-- connections functions from the domain, services, and clients

Internal parameters--

dojo - relative url of dojo

mongoose

fb

sys

mongo

fs

user

profilePicture

saveProfilePicture

**Input**

userId

data

mime

callback

constructor

**Input**

opts

sixDegrees

**Input**

seeker

target

callback

VerifyUser

**Input**

email

password

callback

createUser

**Input**

useJson

callback

removeUser

**Input**

userId

callback

findByFbId

**Input**

fbId

callback

findById

**Input**

userId

callback

getUserInfo

**Input**

userId

callback

addFriend

**Input**

userId

friendId

callback

addMessage

**Input**

from

to

message

callback

retrieveAndCreateFriends

**Input**

accessToken

User

setProfilePicture

**Input**

userId

data

mime

callback

getProfilePicture

**Input**

userId

callback

searchUsers

**Input**

query

callback

createFromFacebook

**Input**

userJson

accessToken

callback

**Clients**

5.1.5 Facebook.js-- Gets friends from Facebook by using the Facebook API and keeping a access\_token to validate the 6 Degrees application to Facebook.

Internal parameters--

http - where to find certain things

dojo - relative url of dojo.js

sys - where to find certain functions

queryString - the string form of the previous query

request - gets the node utilities file

constructor

**Input**

params - parameter values for the constructor

\_**makeRequest**

**Input**

url - url to make the request

params - parameters for the user to request

def - definition for the request return value

getFriends

**Input**

uid - user id

access\_token

**Output**

def - path to functions

getInfo

**Input**

uid - user id

access\_token

**Output**

def - path to functions

getProfilePic

**Input**

uid - user id

access\_token

**Output**

def - path to functions

\_handleResponse

**Input**

def -  path to functions

response

**Domain**

5.1.6 User.js-- user object. It is stored by a MongoDB database. It is a basic definition for storing the parameters, that Mongoose then allows utilities to allow interaction with the database.

**Internal parameters**--

\_id - User ID

fbId - Facebook ID

dateOfBirth - User’s date of birth

password - User’s password

sex - User’s sex

friends - User’s friends

messages - Messages for the user

name - User’s name

email - User’s e-mail

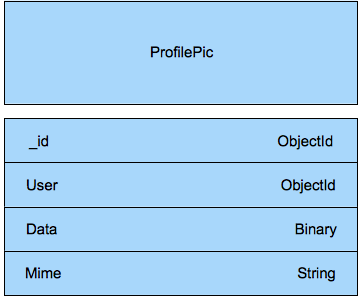
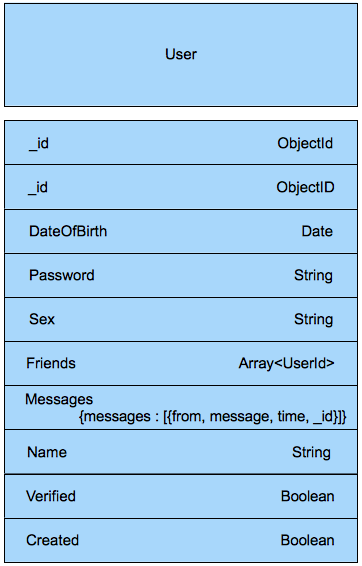
verified - whether or not the user is verified

created - whether or not the account is created

5.17 ProfilePic.js - profile picture object. It is stored by a MongoDB database. It is a basic definition for storing the parameters, that Mongoose then allows utilities to allow interaction with the database.

**Internal parameters**

properties - array of the data, user, and mime type



## 5.3 User Interface

5.3.1 Degrees.js-- checks the status of the user

checkStatus

formatDate

**Input**

dateString - the current date

includeTime - whether to include the time

**Output**

return the date formatted as specified

5.3.2 user.js-- gets the user info

Init

5.3.3 Profile.js--  template for the profile

\_setUserAttr

**Input**

user - user to set attributes of

\_showEdit

\_hideEdit

\_edit

# 6. Detailed Design

## 6.1 Source File Organization

* node
  + clients
    - Facebook.js
  + configuration (configuration for JSON, a data-interchange format)
  + domain
    - User.js
    - ProfilePic.js
  + external-libs
    - bson ( a library mangoDB library for binary-encoded serialization of JSON-like documents)
  + lib (external libraries)
    - dojo (dojo library files)
    - dug-server (server specific
    - mongodb (mongoDB library files)
    - formidable
    - mongoose
    - node-utils
    - node\_hash
  + services
    - auth.js
    - user.js
    - image.js
  + util
    - UserUtil
  + index.js
* web
  + css
    - 6Degrees.css
  + images (folder containing images for the gui)
  + js
    - degrees
      * layout (dojo toolkit specific directory)
      * templates
        + Logon.html
      * themes
        + Logon.css
      * user
        + layout (dojo toolkit specific directory)
        + templates

Profile.html

* + - * + themes

Profile.css

* + - * + Profile.js
      * degrees.js
      * Logon.js
      * Service.js
      * user.js
    - dojo (dojo libraries and files for the interface)
    - raphael (raphael graphic engine)
    - degrees.profile.js
  + error.html
  + home.html
  + index.html
  + logon.html
  + unauthorized.html

## 6.2 Algorithms

Basic pseudo code for the path-finding logic, which is based on Breadth-First Search.

Initialize an empty queue.

Place Target node onto the queue.

Do:

Dequeue top of queue and hold it.

If held node is seeker node:

Return list of connections from current to target node.

Else:

Queue all friends of held node not on queue.

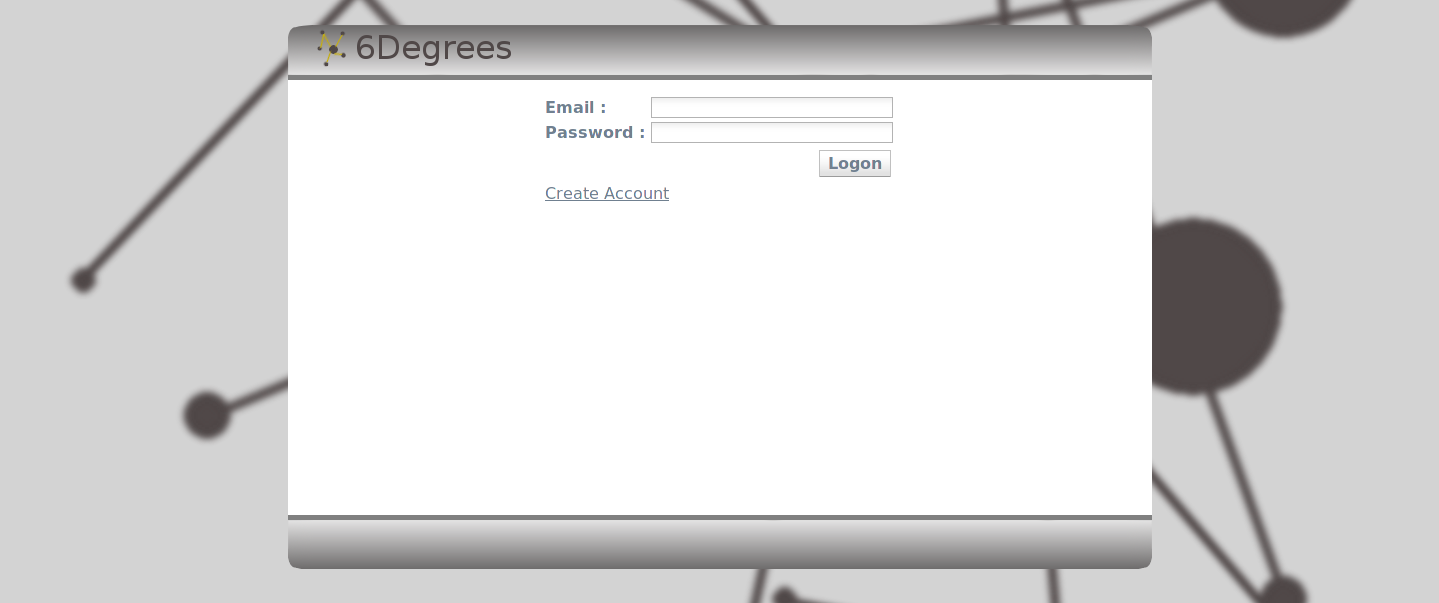
Mark every queued node with their “parent” node.

Increment a Search-Depth marker by 1.

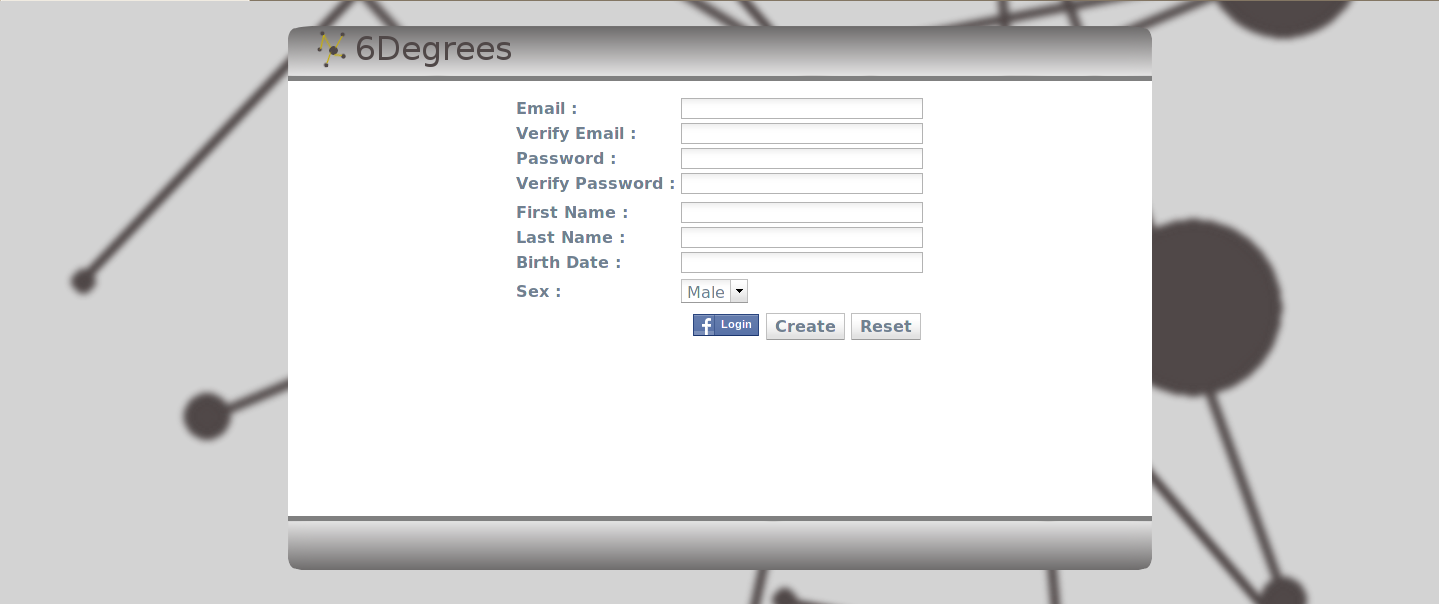
While Search-Depth marker is <= 6

Traditionally, a while loop is used to keep performing the search. However, this may lock up the server system. Therefore, callbacks are used. Callbacks are a reference to an executable (part) of code.

## 6.3 Interface Screenshots



6.3.1 Logon Screen



6.3.2 Create Account Screen



6.3.3 Current Home Screen