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. The server is Node based, which is a javascript framework intended for web servers . This was decided because it can use multiple threads at a time, which is beneficial especially for the path finding logic. There are three main systems within 6 Degree: the user interaction, the logic, and the objects. The user interaction uses Dojo, which is a JavaScript toolkit that allows easy creation of user-interacting widgets. Using Dojo elements for the user to see and interact with, the controller for each page will send the data to individual functions in separate files that will use the user input in the logic.

The logic will be contained in functions within the system, all of which are not viewable to the user. There are three main categories of the logic. One is in charge of accessing Facebook through Facebooks APIs. It will connect to Facebook to add friends to the User object. The logic functions will also take care of managing the user objects, such as creating a user, adding friends to the collection, and edit user information. The last part of the logic will create connection paths between two people. This will search through each person's friend collection and continue with a breadth search of friends of friends to find a connection between two people.

The object will store the user information. This also includes the friend list collection. The logic will interact with this by accessing the user's variables and modifying them.

# 3. Decomposition Description

## 3.1 Objects

The objects holds information that is stored. The only object is the user object, which holds all the data.

3.1.1 User

The database will contain data the user has entered about themselves and from Facebook as well as the collection of friends. This is contained in the domain/User.js file.

## 3.2 Logic

The logic portion of the program handles adding user friends and calculating connection paths.

3.2.1 Create Account

The logic will create a new user object with input given into it for user information.

* domain/User.js

This file contains the basic constructor for the user object. It will be given input from the user and will create the object with appropriate parameters.

* Logon.js

This file contains a method that will call user constructor. It will pass the input from the interface and will validate the information is valid. It will also redirect the user to connect with Facebook after completing the basic account information.

3.2.2 Log in to Account

This logic will validate the login credentials and redirect to the user's home page.

* Logon.js

This file will contain all the logic that checks the password and email are valid.

3.2.3 Sync to Friends List

This will add or update the friends list of the user.

* Facebook.js

This is the basic API that will allow 6 Degrees to interact with Facebook. It will, after given Facebook credentials and permission, retrieve the list of names and Facebook ids for the friends of a person. The method called getFriends will be responsible for returning the friends.

* auth.js

This file will allow interaction between Facebook by granting 6 Degrees permission to access the Facebook user through the APIs. This also allows to update or create a new collection of friends (updating a blank collection).

* Logon.js

This file will call the Facebook.js getFriends method after a person makes a basic 6 Degree account when they are creating a new account.

3.2.4 Find Path

This can find a path between two users following a series of friends.

* \_BaseDao.js

This file features many methods that work together to help find the connection between the user and an unfriended person

## 3.3 User Interface

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 Object

The user object is the only storage object, therefor, the only dependency is the user object friend collection, which points to other user object files.

User 5

User 2

User 1

User 3

User 4

## 4.2 Logic & User Interface

Many of the logic 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 | User.js | User.js(2), BaseDAO.js, Auth.js, User.js(3), Logon.js, Service.js, Degrees.js, User.js(4) |
| Facebook.js | none | User.js(3), Logon.js |
| Friends.js | BaseDOA.js | None |
| User.js(2) | User.js | None |
| BaseDAO.js | User.js | Friends.js, User.js |
| Auth.js | User.js | None |
| User.js(3) | User.js, Facebook.js | None |
| Logon.js | User.js, Facebook.js | Degrees.js |
| Service.js | User.js | Degrees.js |
| Degrees.js | User.js, Logon.js, Service.js | None |
| User.js(4) | User.js, Profile.js | None |
| Profile.js | none | User.js(4) |

# 5. Interface Description

## 5.1 Object

5.1.1 User.js-- user object

**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

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

## 5.2 Logic

5.2.1 Facebook.js-- Gets friends from 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

constructor

**Input**

params - parameter values for the constructor

**Output**

return value

getFriends

**Input**

uid - user id

**Output**

def - path to functions

\_handleResponse

**Input**

def -  path to functions

response

5.2.2 Friends.js-- exports the user’s friends

Internal parameters--

dojo - relative url of dojo

BaseDAO - relative url of BaseDAO

Db - relative url of mongodb.Db

Server - relative url of mongodb.Server

5.2.3 User.js(2)-- makes the user

Internal parameters--

dojo - relative url of dojo

BaseDAO - relative url of BaseDAO

Db - relative url of mongodb.Db

Server - relative url of mongodb.Server

User - relative url of User

getUser

**Input**

id - user id

callback

createUser

**Input**

userObj - user object

callback

createFacebookUser

**Input**

userObj - user object

callback

updateFriends

**Input**

userID - user id

friendIDs - friend ids

verifyUser

**Input**

email

password

callback

sixDegrees

**Input**

seeker - person looking for it

target - end goal

callback

5.2.4 BaseDAO.js-- makes a database and collects things for it

Internal parameters--

dojo - relative url of dojo

Db - relative url of mongodb.Db

Server - relative url of mongodb.Server

Object ID - relative url of bson

constructor

**Input**

host

port

ensureIndex

**Input**

query

unique - whether it’s unique

callback

createPrimaryKey

**Output**

object id

getCollection

**Input**

callback

count

**Input**

query

callback

findBy

**Input**

query

options

callback

findByPaged

**Input**

query

page index

page size

callback

\_buildPagingOptions

**Input**

options

page index

page size

**Output**

options

findByPagedAndSort

**Input**

query

page index

page size

sort

callback

**Output**

return value

findAll

**Input**

callback

findById

**Input**

id

callback

findByObjectId

**Input**

id

callback

findOne

**Input**

query

callback

updateByObjectId

**Input**

id

modifier

callback

updateById

**Input**

id

modifier

callback

update

**Input**

query

modifier

callback

save

**Input**

doc

callback

5.2.5 auth.js-- creates a new user

Internal parameters--

UserDAO - relative url of User.UserDAO

fb - relative url of Facebook.FacebookClient

dojo - relative url of dojo

User - relative url of User.User

db - an object of class UserDAO

client - an object of class fb

APP\_ACCESS\_TOKEN - keeps the page secure

verifyUser - a function that verifies the user

exports.checkStatus

**Input**

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

exports.createUser

**Input**

user

req - path to functions

res - path to functions

retrieveAndCreateFriends

**Input**

cookie

user id

5.2.6 user.js(3)-- searches the graph

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

exports.getPersonInfo

exports.getInfo

**Input**

req - path to functions

res - path to functions

exports.getMessages

**Input**

recipient user id

req - path to functions

res - path to functions

exports.getFriends

**Input**

req - path to functions

res - path to functions

5.2.7 Logon.js-- logs on

constructor

\_showCreate

\_checkPasswords

**Output**

true

\_checkEmail

**Output**

true

\_onSubmit

**Input**

e - path to functions

\_onCreateSubmit

**Input**

e - path to functions

\_checkResponse

**Input**

res - path to functions

\_setUserAttr

**Input**

vals - values

5.2.8 Service.js-- holds several functions and urls to access them

findConnection

**Input**

endUser - the user at the end of the connection

**Output**

connection

CheckStatus

**Output**

status

getUserInfo

**Output**

info

logon

**Input**

creds - see if the user is authorized

**Output**

logon information

logout

**Input**

creds - user authorization

**Output**

logout information

createUser

**Input**

user - user to create

**Output**

create user information

## 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)
  + db
    - dao
      * \_BaseDAO.js
      * Friends.js
      * Users.js
    - domain
      * User.js
  + external-libs
    - bson ( a library mangoDB library for binary-encoded serialization of JSON-like documents)
  + lib
    - dojo (dojo library files)
    - dug-server (server specific
    - mongodb (mongoDB library files)
  + services
    - auth.js
    - user.js
  + 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)
  + 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.

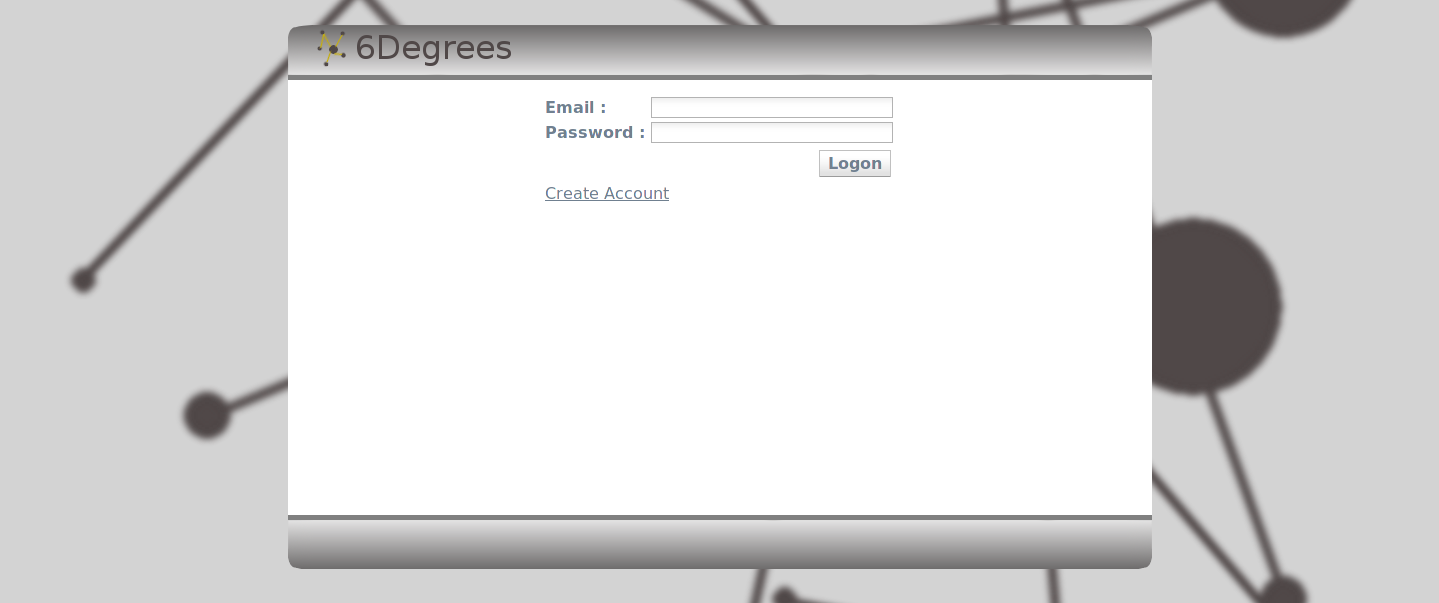
Mark every queued node with their “parent” node.

Increment a Search-Depth marker by 1.

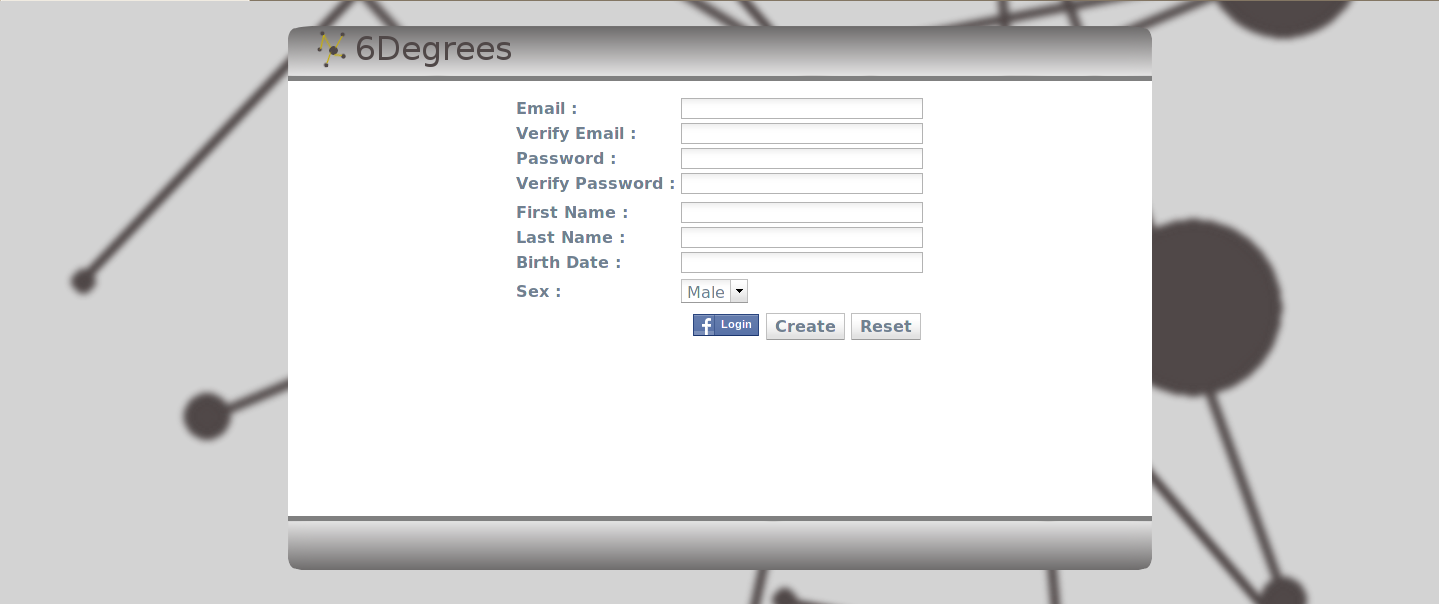
While Search-Depth marker is <= 6

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