Prof. Kaiser
Continuous Integration and Coverage - Coverage Report
SYNC(Ruicong Xie-rx2119, Tianci Zhong-tz2278, Yue Cen-yc3171, Jiaxin Su-js4722)
December 10th

Coverage Report

Dec.16th
Here is the summary about which file in which folder and how much each of the files are covered by our unit test.

* app folder
    * server.js - 100%
* models
    * commentModel.js - 100%
    * likeModel.js - 100%
    * requestModel.js - 100%
    * urlModel.js - 100%
    * userModel.js - 100%
    * userUrlModel.js - 100%
* route folder
    * auth.js - 100%
    * feed.js - 100%
    * index.js - 100%
    * rank.js - 93.1%
    * redirect.js - 100%
    * rest.js - 100%
* services folder
    * authService.js - 95.74%
    * config.js - 100%
    * rankUrlService.js - 91.53%
    * statsService.js - 88.52%
    * urlService.js - 84.48%
    * userUrlService.js - 82.05%

The test cases with pink highlight below are the ones that we did not cover.

I. router folder

* rank.js - 93.1%

```
router.get('/saveUrlClicks', function(req, res) {
    rankUrlService.saveUrlClicks(function(err, data) {
        Iif (err != null) {
            res.status(403).json(
                {'message': 'Save all Urls\' click information to Redis
failed'});
            return;
        }
        res.json({'message': 'Success'});
    });
});
```

II. services folder

* authService.js - 95.74%

```
var getUser = function(req, callback) {
    if (!(req.headers && req.headers.authorization)) {
        callback({'_id': -1});
        return;
    }

    var header = req.headers.authorization.split(' ');
    var token = header[1];
    var payload = jwt.decode(token, config.tokenSecret);
    var userId = payload.sub;

    userModel.findById(userId, function(err, user) {
        Iif (err) {
            // console.log(err);
            callback(err);
            return;
        }
        // console.log('user: ' + user);
        callback(user);
    });
};
```

```
* rankUrlService.js - 91.53%

var getTopKUrls = function(k, callback) {
    redisClient.hgetall(hashClick, function(err, data) {
        var queue = new PriorityQueue(function(a, b) {
            return a.clicks - b.clicks;
        });
        for (var key in data) {
            queue.enq({shortUrl: key, clicks: data[key]});
        }
        var result = [];
        for (var i = 0; i < k; i++) {
            if (queue.isEmpty()) {
                break;
            }
            result.push(queue.deq());

        }
        callback(result);
    });
};

var saveUrlClicks = function(callback) {
    UrlModel.find({}, function(err, data) {
        Iif (err != null) {
            callback(err);
            return;
        }

        var count = 0;
        // console.log("length:" + data.length);
        for (var i = 0; i < data.length; i++) {
            var shortUrl = data[i].shortUrl;
            getUrlClicks(shortUrl, function(shortUrl, clicks) {
                var obj = {};
                obj[shortUrl] = clicks;
                // console.log(shortUrl, clicks);
                redisClient.hmset(hashClick, obj, function(err, data) {
                    Iif (err != null) {
                        callback(err);
                        return;
                    }
                });
                count++;
                // console.log(count);
                if (count === data.length) {
                    callback(err);
                }
            });
        }
    });
};
```

```
* statsService.js - 88.52%
var logRequest = function(shortUrl, req) {
    var reqInfo = {};
    reqInfo.shortUrl = shortUrl;
    reqInfo.referer = req.headers.referer || 'Unknown';
    reqInfo.platform = req.useragent.platform || 'Unknown';
    reqInfo.browser = req.useragent.browser || 'Unknown';
    var ip = requestIp.getClientIp(req);
    var geo = geoip.lookup(ip);
    Iif (geo) {
        reqInfo.country = geo.country;
    } else {
        reqInfo.country = 'Unknown';
    }
    reqInfo.timestamp = new Date();

    redisClient.keys('*', function(err, data) {
        console.log('data: ' + data);
        if (data.length > 0) {
            Eif (data.indexOf(shortUrl) > -1) {
                var request = new RequestModel(reqInfo);
                request.save(function(err, data) {
                    Iif (err != null) {
                        console.log(err);
                        return;
                    }
                    // console.log('request saved: ' + data);
                    Eif (data.shortUrl != 'undefined'
                        && data.shortUrl != 'favicon.ico') {
                        rankUrlService.updateUrlClicks(data.shortUrl,
                            function(url, data) {
                                Iif (url != shortUrl) {
                                    console.log(url);
                                }
                        });
                    }
                });
            }
        } else {
            // var reg = /[abc]*\/[abc]*/i;
            // RequestModel.remove({shortUrl: reg}, function(err, data) {
```

```
            //     console.log(data);
            // });
            UrlModel.find({shortUrl: shortUrl}, function(err, data) {
                var request = new RequestModel(reqInfo);
                request.save(function(err, data) {
                    Iif (err != null) {
                        console.log(err);
                        return;
                    }
                    // console.log('request saved: ' + data);
                    Eif (data.shortUrl != 'undefined'
                        && data.shortUrl != 'favicon.ico') {
                        rankUrlService.updateUrlClicks(data.shortUrl,
                            function(url, data) {
                                Iif (url != shortUrl) {
                                    console.log(url);
                                }
                            });
                    }
                });
            })
        }
    });
};
```

```
* urlService.js - 84.48%
var getShortUrl = function(longUrl, callback) {
    // This part has been handled in the front-end, hence comment it.
    /* if (longUrl.indexOf('http') == -1) {
     longUrl = "http://" + longUrl;
     } */
    redisClient.get(longUrl, function(err, shortUrl) {
        if (shortUrl) {
            // console.log('using cache');
            callback({
                status: 'ok',
                shortUrl: shortUrl,
                longUrl: longUrl
            });
        } else {
            validateUrl(longUrl, function(output) {
                if (output.status !== 'ok') {
                    callback(output);
                } else {
                    UrlModel.findOne({
                        longUrl: longUrl
                    }, function(err, data) {
                        Eif (data) {
                            callback({
                                status: 'ok',
                                shortUrl: data.shortUrl,
                                longUrl: data.longUrl
                            });
                            redisClient.set(data.shortUrl, data.longUrl);
                            redisClient.set(data.longUrl, data.shortUrl);
                        } else {
                            generateShortUrl(function(shortUrl) {
                                var url = new UrlModel({
                                    shortUrl: shortUrl,
                                    longUrl: longUrl
                                });
                                url.save(function() {
                                    callback({
                                        status: 'ok',
                                        shortUrl: shortUrl,
                                        longUrl: longUrl
                                    });
                                    redisClient.set(shortUrl, longUrl);
                                    redisClient.set(longUrl, shortUrl);
                                });
                            });
                        }
                    });
                }
            });
        }
    });
};

var getLongUrl = function(shortUrl, callback) {
    redisClient.get(shortUrl, function(err, longUrl) {
        if (longUrl) {
            // console.log("byebye mongo " + longUrl + " end");
            callback({
                status: 'ok',
                shortUrl: shortUrl,
                longUrl: longUrl
            });
        } else {
            UrlModel.findOne({
                shortUrl: shortUrl
            }, function(err, data) {
                Iif (err) {
                    console.log(err);
                    callback({
                        status: 'failed',
                        message: err
                    });
                    return;
                }

                if (data) {
                    // console.log('data: ' + data);
                    callback({
                        status: 'ok',
                        shortUrl: shortUrl,
                        longUrl: data.longUrl
                    });
                    // console.log(data);
                    redisClient.set(shortUrl, data.longUrl);
                    redisClient.set(data.longUrl, shortUrl);
                } else {
                    callback({
                        status: 'failed',
                        shortUrl: shortUrl,
                        message: 'The short URL does not exist.'
                    });
                }
            });
        }
    });

};
```

```
* userUrlService.js - 82.05%
var getFeed = function(pageSize, lastId, isPublic, userId, callback) {
    // console.log('lastId: ' + lastId);
    pageSize = parseInt(pageSize);
    var countQuery = {isDeleted: {$ne: true}};
    var actualQuery = {isDeleted: {$ne: true}};
    // console.log('userId: ' + userId);
    // console.log('lastId: ' + lastId);
    if (userId !== -1) {
        countQuery['userId'] = userId;
        actualQuery['userId'] = userId;
    }
    if (lastId !== '-1') {
        actualQuery['_id'] = {$lt: lastId};
    }
    countQuery['public'] = isPublic;
    actualQuery['public'] = isPublic;

    userUrlModel.find(countQuery).count(function(err, count) {
        var json = {'status': 'ok', 'count': count, 'data': []};
        userUrlModel.find(actualQuery).sort({'_id': -1}).limit(pageSize).
            exec(function(err, data) {
                Iif (err) {
                    callback(err);
                    return;
                }

                json.data = data;
                callback(json);
            });
    });
};

var getPostById = function(postId, callback) {
    redisClient.get(postId.toString(), function(err, post) {
        if (post) {
            // console.log("using cache: " + post);
            var json = JSON.parse(post);
            callback(json);
        } else {
            userUrlModel.findById(postId, function(err, postInDb) {
                Iif (err) {
                    callback(err);
                    return;
                }
                callback(postInDb);
                // console.log("stringify: " + JSON.stringify(postInDb));
                redisClient.set(postId.toString(), JSON.stringify(postInDb));
            });
        }
    });
};

var removePost = function(postId, userId, callback) {
    if (userId === -1) {
        callback({'status': 'failed', 'message': 'Not logged in.'});
    } else {
        getPostById(postId, function(post) {
            if (userId != post.userId) {
                callback({'status': 'failed', 'message': 'Not authorized.'});
            } else {
                userUrlModel.update({_id: postId}, {
                    isDeleted: true
                }, function(err, affected, resp) {
                    Iif (err) {
                        callback(err);
                        return;
                    }
                    redisClient.del(postId.toString());
                    callback({'status': 'ok'});
                });
            }
        });
    }
};

var getNumberOfLikes = function(postId, callback) {
    var json = {'status': 'ok', 'data': {}};

    likeModel.find({postId: postId}).count().exec(function(err, count) {
        Iif (err) {
            json['status'] = 'failed';
            json['data'] = err;
            callback(json);
            return;
        }

        json['data']['count'] = count;
        callback(json);
    });
};

var hasLiked = function(postId, userId, callback) {
    var json = {'status': 'ok', 'data': {}};

    likeModel.find({postId: postId, userId: userId}).count().
        exec(function(err, count) {
            Iif (err) {
                json['status'] = 'failed';
                json['data'] = err;
                callback(json);
                return;
            }
            json['data']['hasLiked'] = count > 0;
            callback(json);
        });
};

var getCommentById = function(commentId, callback) {
    redisClient.get(commentId.toString(), function(err, comment) {
        Iif (comment) {
            // console.log("using cache: " + post);
            var json = JSON.parse(comment);
            callback(json);
        } else {
            commentModel.findById(commentId, function(err, commentInDb) {
                Iif (err) {
                    callback(err);
                    return;
                }
                callback(commentInDb);
                // console.log("stringify: " + JSON.stringify(postInDb));
                redisClient.set(commentId.toString(),
                    JSON.stringify(commentInDb));
            });
        }
    });
};

var removeComment = function(commentId, userId, callback) {
    if (userId == -1) {
        callback({'status': 'failed', 'message': 'Not logged in.'});
    } else {
        getCommentById(commentId.toString(), function(comment) {
            if (userId != comment.userId) {
                callback({'status': 'failed', 'message': 'Not authorized.'});
            } else {
                commentModel.update({_id: commentId}, {
                    isDeleted: true
                }, function(err, affected, resp) {
                    Iif (err) {
                        callback(err);
                        return;
                    }

                    redisClient.del(commentId.toString());
                    callback({'status': 'ok'});
                });
            }
        });
    }
};

var getComments = function(postId, callback) {
    var json = {'status': 'ok', 'data': {}};

    commentModel.find({postId: postId, isDeleted: {$ne: true}}).sort({_id: 1}).
        exec(function(err, comments) {
            Iif (err) {
                json['status'] = 'failed';
                json['data'] = err;
                callback(json);
                return;
            }

            json['data'] = comments;
            callback(json);
        });
};

var getNumberOfComments = function(postId, callback) {
    var json = {'status': 'ok', 'data': {}};

    commentModel.find({postId: postId, isDeleted: {$ne: true}}).count().
        exec(function(err, count) {
            Iif (err) {
                json['status'] = 'failed';
                json['data'] = err;
                callback(json);
                return;
            }

            json['data']['count'] = count;
            callback(json);
        });
};
```

Dec.10
Here is the summary about which file in which folder and how much each of the files are covered by our unit test.

```
* app folder
   * server.js - 100%
* models
   * commentModel.js - 100%
   * likeModel.js - 100%
   * requestModel.js - 100%
   * urlModel.js - 100%
   * userModel.js - 100%
   * userUrlModel.js - 100%
* route folder
   * auth.js - 88.24%
   * feed.js - 91.01%
   * index.js - 100%
   * rank.js - 92.31%
   * redirect.js - 93.57%
   * rest.js - 77.78%
* services folder
   * authService.js - 60.66%
   * config.js - 100%
   * rankUrlService.js - 87.72%
   * statsService.js - 85%
   * urlService.js - 52.54%
   * userUrlService.js - 75.8%
```

We will go through each of the non-100% files and see which part of the codes are not covered. The uncovered lines has red highlight.

I. route folder
* auth.js - 88.24%

```
router.post('/login', jsonParser, function(req, res) {
   authService.login(req.body.email, req.body.password, function(json) {
       Eif (json.status !== 200) {
           res.status(json.status).send({message: json.message});
       } else {
            res.send({token: json.token, user: json.user});
       }
   });
});
```

```
router.post('/reg', jsonParser, function(req, res) {
   // console.log(req.body);
   authService.reg(req.body.email, req.body.password, req.body.fullname,
       function(json) {
           Eif (json.status === 409) {
               res.status(409).send({message: json.message});
           } else {
                res.send({token: json.token});
           }
       });
});
```

* feed.js - 91.01%

```
// remove comment
router.post('/post/removeComment', jsonParser, function(req, res) {
   authService.getUser(req, function(user) {
       var userId = user._id;
       Iif (userId != -1) {
            userUrlService.removeComment(req.body.commentId, userId,
                function(data) {
                    if (data.status === 'ok') {
                        res.json(data);
                   } else {
                        res.status(403).send(data);
                   }
               });
       } else {
           res.status(403).
               send({'status': 'failed', 'message': 'Not logged in.'});
       }
   });
});
```

```
// remove post
router.post('/post/removePost', jsonParser, function(req, res) {
   authService.getUser(req, function(user) {
       var userId = user._id;
       Iif (userId != -1) {
            userUrlService.removePost(req.body.postId, userId, function(data) {
                if (data.status == 'ok') {
                    res.json(data);
               } else {
                    res.status(403).send(data);
               }
           });
       } else {
           res.status(403).
               send({'status': 'failed', 'message': 'Not logged in.'});
       }
   });
});
```

* rank.js - 92.31%

```
router.get('/saveUrlClicks', function(req, res) {
   rankUrlService.saveUrlClicks(function(err, data) {
       Iif (err != null) {
            res.status(403).json(
               {'message': 'Save all Urls\' click information to Redis
failed'});
            return;
       }
       res.json({'message': 'Success'});
   });
});
```

* redirect.js - 93.57%

```
module.exports = function RedirectRouter(io) {
   var router = express.Router();
   router.get('*', function(req, res) {
       var shortUrl = req.originalUrl.slice(1);
       urlService.getLongUrl(shortUrl, function(url) {
           Eif (url) {
               res.redirect(url.longUrl);
               statsService.logRequest(shortUrl, req);
               io.emit('shortUrlVisited', shortUrl);
           } else {
                res.sendFile('404.html', {
                   root: path.join(__dirname, '../public/views')
               });
           }
       });
   });
   this.router = router;
};
```

* rest.js - 77.78%

```
router.post('/urls', jsonParser, function(req, res) {
   var longUrl = req.body.longUrl;
   urlService.getShortUrl(longUrl, function(json) {
       Iif (json.status === 'ok') {
            authService.getUser(req, function(user) {
               // console.log('user: ' + user);
                if (user._id !== -1) {
                    var isPublic = typeof req.body.isPublic === 'undefined' ?
                       true : req.body.isPublic;
                    userUrlService.add(user._id, user.fullname, json.shortUrl,
                       longUrl, isPublic, function(data) {
                            console.log(data);
                       });
               }

                res.json(json);
           });
       } else {
           res.status(400).send(json);
       }
   });
});
```

II. services folder
* authService.js - 60.66%

```
var isAuthenticated = function(req, res, next) {
    if (!(req.headers && req.headers.authorization)) {
        return res.status(400).send({message: 'No Token.'});
   }

    var header = req.headers.authorization.split(' ');
    var token = header[1];
    var payload = jwt.decode(token, config.tokenSecret);
    var now = moment().unix();

    if (now > payload.exp) {
        return res.status(401).send({message: 'Token has expired.'});
   }

    userModel.findById(payload.sub, function(err, user) {
        if (!user) {
            return res.status(400).send({message: 'User does not exist.'});
       }
```

```
        req.user = user;
        next();
   });
};


var getUser = function(req, callback) {
   if (!(req.headers && req.headers.authorization)) {
       callback({'_id': -1});
       return;
   }

   var header = req.headers.authorization.split(' ');
   var token = header[1];
   var payload = jwt.decode(token, config.tokenSecret);
   var userId = payload.sub;

   userModel.findById(userId, function(err, user) {
       Iif (err) {
           // console.log(err);
            callback(err);
            return;
       }
       // console.log('user: ' + user);
       callback(user);
   });
};

var reg = function(email, password, fullname, callback) {
   userModel.findOne({email: email}, function(err, existingUser) {
       Eif (existingUser) {
           callback(
               {status: 409, message: {email: 'Email is already taken.'}});
           return;
       }

        var user = new userModel({
           email: email,
           password: password,
           fullname: fullname
       });

        bcrypt.genSalt(10, function(err, salt) {
            bcrypt.hash(user.password, salt, function(err, hash) {
                user.password = hash;

                user.save(function() {
                    var token = createToken(user);
                    callback({status: 200, token: token});
               });
           });
       });
   });
};

var login = function(email, password, callback) {
   userModel.findOne({email: email}, '+password', function(err, user) {
       Iif (!user) {
            callback(
               {status: 401, message: {email: 'This user does not exist.'}});
            return;
       }

       bcrypt.compare(password, user.password, function(err, isMatch) {
           if (!isMatch) {
               callback({
                   status: 401,
                   message: {password: 'The password is not correct.'}
               });
               return;
           }

           user = user.toObject();
           delete user.password;

           var token = createToken(user);
           callback({status: 200, token: token, user: user});
       });
   });
};
```

* rankUrlService.js - 87.72%

```
var getUrlClicksCached = function(shortUrl, callback) {
    redisClient.hget(hashClick, shortUrl, function(err, data) {
        Iif (err == null && data == null) {
             getUrlClicks(shortUrl, function(shortUrl, data) {
                 callback(shortUrl, data);
            });
        } else {
            callback(shortUrl, data);
        }
    });
};

var getTopKUrls = function(k, callback) {
    redisClient.hgetall(hashClick, function(err, data) {
        var queue = new PriorityQueue(function(a, b) {
            return a.clicks - b.clicks;
        });
        for (var key in data) {
            queue.enq({shortUrl: key, clicks: data[key]});
        }
        var result = [];
        for (var i = 0; i < k; i++) {
            Iif (queue.isEmpty()) {
                 break;
            }
            result.push(queue.deq());

        }
        callback(result);
```

```
   });
};

var saveUrlClicks = function(callback) {
   UrlModel.find({}, function(err, data) {
       Iif (err != null) {
            callback(err);
            return;
       }

       var count = 0;
       // console.log("length:" + data.length);
       for (var i = 0; i < data.length; i++) {
           var shortUrl = data[i].shortUrl;
           getUrlClicks(shortUrl, function(shortUrl, clicks) {
               var obj = {};
               obj[shortUrl] = clicks;
               // console.log(shortUrl, clicks);
               redisClient.hmset(hashClick, obj, function(err, data) {
                   Iif (err != null) {
                        callback(err);
                        return;
                   }
               });
               count++;
               // console.log(count);
               if (count === data.length) {
                   callback(err);
               }
           });
       }
   });
};
```

* statsService.js - 85%

```
var logRequest = function(shortUrl, req) {
   var reqInfo = {};
   reqInfo.shortUrl = shortUrl;
   reqInfo.referer = req.headers.referer || 'Unknown';
   reqInfo.platform = req.useragent.platform || 'Unknown';
   reqInfo.browser = req.useragent.browser || 'Unknown';
   var ip = req.headers['x-forwarded-for'] ||
       req.connection.remoteAddress ||
        req.socket.remoteAddress ||
        req.connection.socket.remoteAddress;
   var geo = geoip.lookup(ip);
   Iif (geo) {
        reqInfo.country = geo.country;
   } else {
       reqInfo.country = 'Unknown';
   }
   reqInfo.timestamp = new Date();
   var request = new RequestModel(reqInfo);
   request.save(function(err, data) {
       Iif (err != null) {
            console.log(err);
            return;
       }
       Iif (data.shortUrl.indexOf('/') === -1 && data.shortUrl.indexOf('_')) {
            rankUrlService.updateUrlClicks(data.shortUrl, function(err, data) {
                if (err != null) {
                    console.log(err);
                }
           });
       }
   });

};
```

```
* urlService.js - 52.54%
```

```
var getShortUrl = function(longUrl, callback) {
   // This part has been handled in the front-end, hence comment it.
   /* if (longUrl.indexOf('http') == -1) {
    longUrl = "http://" + longUrl;
    } */
   redisClient.get(longUrl, function(err, shortUrl) {
       if (shortUrl) {
           console.log('using cache');
           callback({
               status: 'ok',
               shortUrl: shortUrl,
               longUrl: longUrl
           });
       } else {
           validateUrl(longUrl, function(output) {
               Eif (output.status !== 'ok') {
                   callback(output);
               } else {
                    UrlModel.findOne({
                       longUrl: longUrl
                   }, function(err, data) {
                        if (data) {
                            callback({
                               status: 'ok',
                               shortUrl: data.shortUrl,
                               longUrl: data.longUrl
                           });
                            redisClient.set(data.shortUrl, data.longUrl);
                            redisClient.set(data.longUrl, data.shortUrl);
                       } else {
                            generateShortUrl(function(shortUrl) {
                                var url = new UrlModel({
                                   shortUrl: shortUrl,
                                   longUrl: longUrl
                               });
                                url.save(function() {
                                    callback({
                                       status: 'ok',
                                       shortUrl: shortUrl,
                                       longUrl: longUrl
                                   });
                                    redisClient.set(shortUrl, longUrl);
                                    redisClient.set(longUrl, shortUrl);
                               });
                           });
                       }
                   });
               }
           });
       }
   });
};

var getLongUrl = function(shortUrl, callback) {
   redisClient.get(shortUrl, function(err, longUrl) {
       if (longUrl) {
           // console.log("byebye mongo " + longUrl + " end");
           callback({
               status: 'ok',
               shortUrl: shortUrl,
               longUrl: longUrl
           });
       } else {
           UrlModel.findOne({
               shortUrl: shortUrl
           }, function(err, data) {
               Iif (err) {
                    console.log(err);
                    callback({
                       status: 'failed',
                       message: err
                   });
                    return;
               }

               Iif (data) {
                   // console.log('data: ' + data);
                    callback({
                       status: 'ok',
                       shortUrl: shortUrl,
                       longUrl: data.longUrl
                   });
                   // console.log(data);
                    redisClient.set(shortUrl, data.longUrl);
                    redisClient.set(data.longUrl, shortUrl);
               } else {
                   callback({
                       status: 'failed',
                       shortUrl: shortUrl,
                       message: 'The short URL does not exist.'
                   });
               }
           });
       }
   });

};

var generateShortUrl = function(callback) {
    UrlModel.count({}, function(err, num) {
        callback(convertTo62(num + 1));
   });
};

var convertTo62 = function(id) {
    var code = 'abcdefghijklmnopqrstuvwxzy';
    code += code.toUpperCase();
    code += '0123456789';
    var shortUrl = '';
    while (id > 0) {
        shortUrl += code[(id - 1) % 62];
        id = parseInt(id / 62);
   }
    return shortUrl;
};
```

* userUrlService.js - 75.8%

```
var add = function(userId, fullname, shortUrl, longUrl, isPublic, callback) {
   Eif (userId !== -1) {
       var userUrl = new userUrlModel({
           userId: userId,
           fullname: fullname,
           shortUrl: shortUrl,
           longUrl: longUrl,
           timestamp: Date.now(),
           public: isPublic
       });
       userUrl.save();
       callback(userUrl);
   } else {
        callback({message: 'No userId.'});
   }
};

var getFeed = function(pageSize, lastId, isPublic, userId, callback) {
   // console.log('lastId: ' + lastId);
   pageSize = parseInt(pageSize);
   var countQuery = {isDeleted: {$ne: true}};
   var actualQuery = {isDeleted: {$ne: true}};
   // console.log('userId: ' + userId);
   // console.log('lastId: ' + lastId);
   if (userId !== -1) {
       countQuery['userId'] = userId;
       actualQuery['userId'] = userId;
   }
   Iif (lastId !== '-1') {
        actualQuery['_id'] = {$lt: lastId};
   }
   countQuery['public'] = isPublic;
   actualQuery['public'] = isPublic;

   userUrlModel.find(countQuery).count(function(err, count) {
       var json = {'status': 'ok', 'count': count, 'data': []};
       userUrlModel.find(actualQuery).sort({'_id': -1}).limit(pageSize).
           exec(function(err, data) {
               Iif (err) {
                    callback(err);
                    return;
               }

               json.data = data;
               callback(json);
           });
   });
};

var getMeta = function(url, callback) {
   url = decodeURIComponent(url);
   var client = new MetaInspector(url, {timeout: 5000});

   var json = {'result': {'status': 'ok'}, 'meta': {}};

   client.on('fetch', function() {
       json['meta']['url'] = url;
       json['meta']['rootUrl'] = client.rootUrl;
       json['meta']['title'] = client.title;
       json['meta']['description'] = client.description;
       json['meta']['image'] = client.image;

       // console.log(client.images);
       var length = client.images.length;
       json['meta']['images'] = [];
       for (var i = 0; i < length; i++) {
           json['meta']['images'].push(client.images[i]);
       }

       callback(json);
   });

   client.on('error', function(err) {
        console.log(err);
        json['result']['status'] = 'failed';
        json['result']['error'] = err;
        json['meta']['url'] = url;
        callback(json);
   });

   client.fetch();
};

var getPostById = function(postId, callback) {
   redisClient.get(postId.toString(), function(err, post) {
       if (post) {
           // console.log("using cache: " + post);
           var json = JSON.parse(post);
           callback(json);
       } else {
           userUrlModel.findById(postId, function(err, postInDb) {
               Iif (err) {
                    callback(err);
                    return;
               }
               callback(postInDb);
               // console.log("stringify: " + JSON.stringify(postInDb));
               redisClient.set(postId.toString(), JSON.stringify(postInDb));
           });
       }
   });
};

var removePost = function(postId, userId, callback) {
   if (userId === -1) {
       callback({'status': 'failed', 'message': 'Not logged in.'});
   } else {
       getPostById(postId, function(post) {
           Iif (userId != post.userId) {
                callback({'status': 'failed', 'message': 'Not authorized.'});
           } else {
               userUrlModel.update({_id: postId}, {
                   isDeleted: true
               }, function(err, affected, resp) {
                   Iif (err) {
                        callback(err);
                        return;
                   }
                   redisClient.del(postId.toString());
                   callback({'status': 'ok'});
               });
           }
       });
   }
};

var getNumberOfLikes = function(postId, callback) {
   var json = {'status': 'ok', 'data': {}};

   likeModel.find({postId: postId}).count().exec(function(err, count) {
       Iif (err) {
            json['status'] = 'failed';
            json['data'] = err;
            callback(json);
            return;
       }

       json['data']['count'] = count;
       callback(json);
   });
};

var hasLiked = function(postId, userId, callback) {
   var json = {'status': 'ok', 'data': {}};

   likeModel.find({postId: postId, userId: userId}).count().
       exec(function(err, count) {
           Iif (err) {
                json['status'] = 'failed';
                json['data'] = err;
                callback(json);
                return;
           }

           json['data']['hasLiked'] = count > 0;
           callback(json);
       });
};

var addComment = function(postId, userId, fullname, message, callback) {
   Iif (userId == -1) {
        callback({'status': 'failed', 'message': 'Not authorized.'});
   } else {
       getPostById(postId, function(post) {
           var comment = new commentModel({
               userId: userId,
               fullname: fullname,
               postId: postId,
               shortUrl: post.shortUrl,
               message: message,
               isDeleted: false,
               timestamp: Date.now()
           });
           comment.save(function() {
               callback({'status': 'ok', 'data': comment});
           });

       });
   }
};

var getCommentById = function(commentId, callback) {
   redisClient.get(commentId.toString(), function(err, comment) {
       Iif (comment) {
           // console.log("using cache: " + post);
            var json = JSON.parse(comment);
            callback(json);
       } else {
           commentModel.findById(commentId, function(err, commentInDb) {
               Iif (err) {
                    callback(err);
                    return;
               }
               callback(commentInDb);
               // console.log("stringify: " + JSON.stringify(postInDb));
               redisClient.set(commentId.toString(),
                   JSON.stringify(commentInDb));
           });
       }
   });
};

var removeComment = function(commentId, userId, callback) {
   if (userId == -1) {
       callback({'status': 'failed', 'message': 'Not logged in.'});
   } else {
       getCommentById(commentId.toString(), function(comment) {
           Iif (userId != comment.userId) {
                callback({'status': 'failed', 'message': 'Not authorized.'});
           } else {
               commentModel.update({_id: commentId}, {
                   isDeleted: true
               }, function(err, affected, resp) {
                   Iif (err) {
                        callback(err);
                        return;
                   }

                   redisClient.del(commentId.toString());
                   callback({'status': 'ok'});
               });
           }
       });
   }
};

var getComments = function(postId, callback) {
   var json = {'status': 'ok', 'data': {}};

   commentModel.find({postId: postId, isDeleted: {$ne: true}}).sort({_id: 1}).
       exec(function(err, comments) {
           Iif (err) {
                json['status'] = 'failed';
                json['data'] = err;
                callback(json);
                return;
           }

           json['data'] = comments;
           callback(json);
       });
};

var getNumberOfComments = function(postId, callback) {
   var json = {'status': 'ok', 'data': {}};

   commentModel.find({postId: postId, isDeleted: {$ne: true}}).count().
       exec(function(err, count) {
           Iif (err) {
                json['status'] = 'failed';
                json['data'] = err;
                callback(json);
                return;
           }

           json['data']['count'] = count;
           callback(json);
       });
};
```