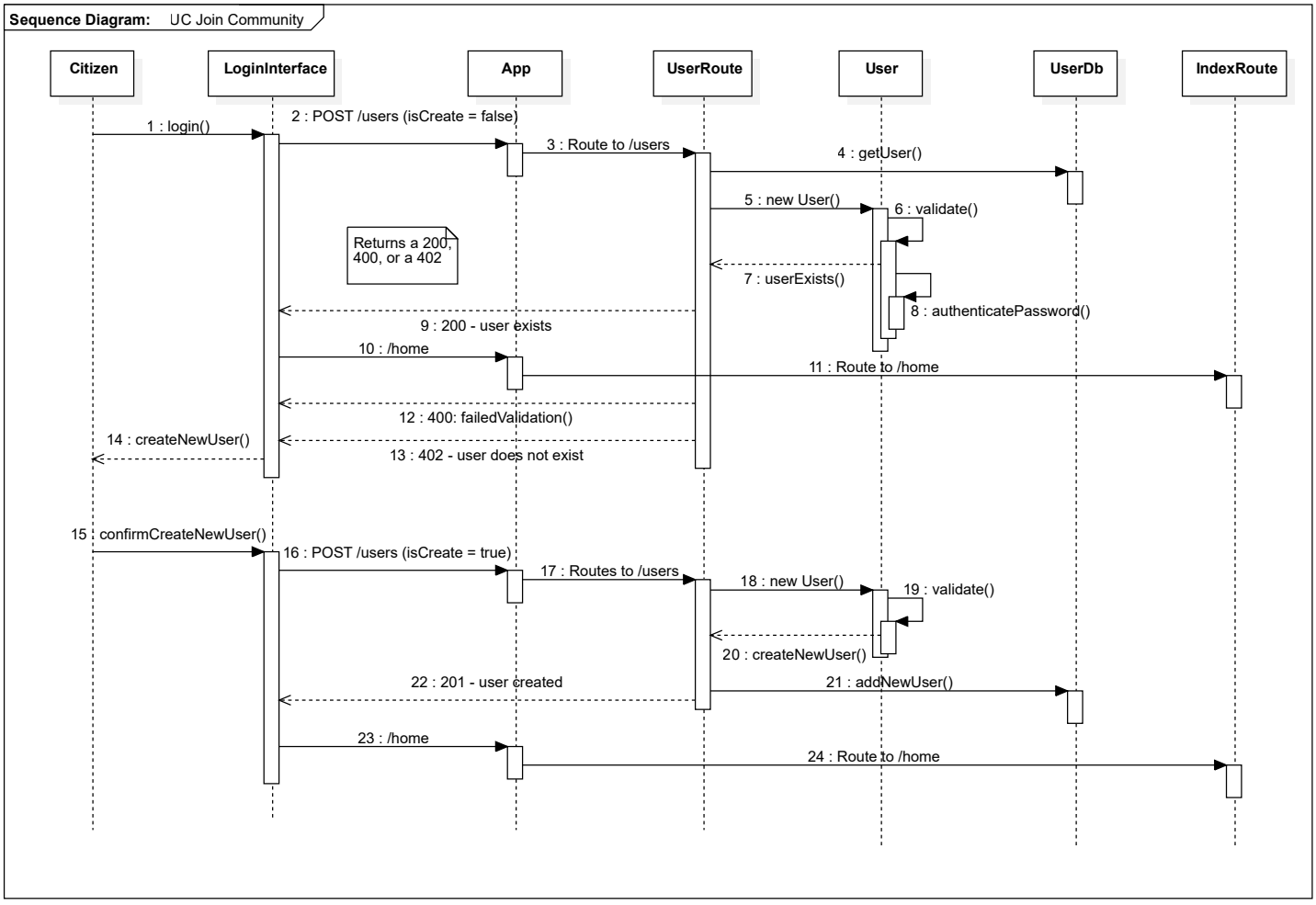
UC: Join Community Team: Justice League (SB5)

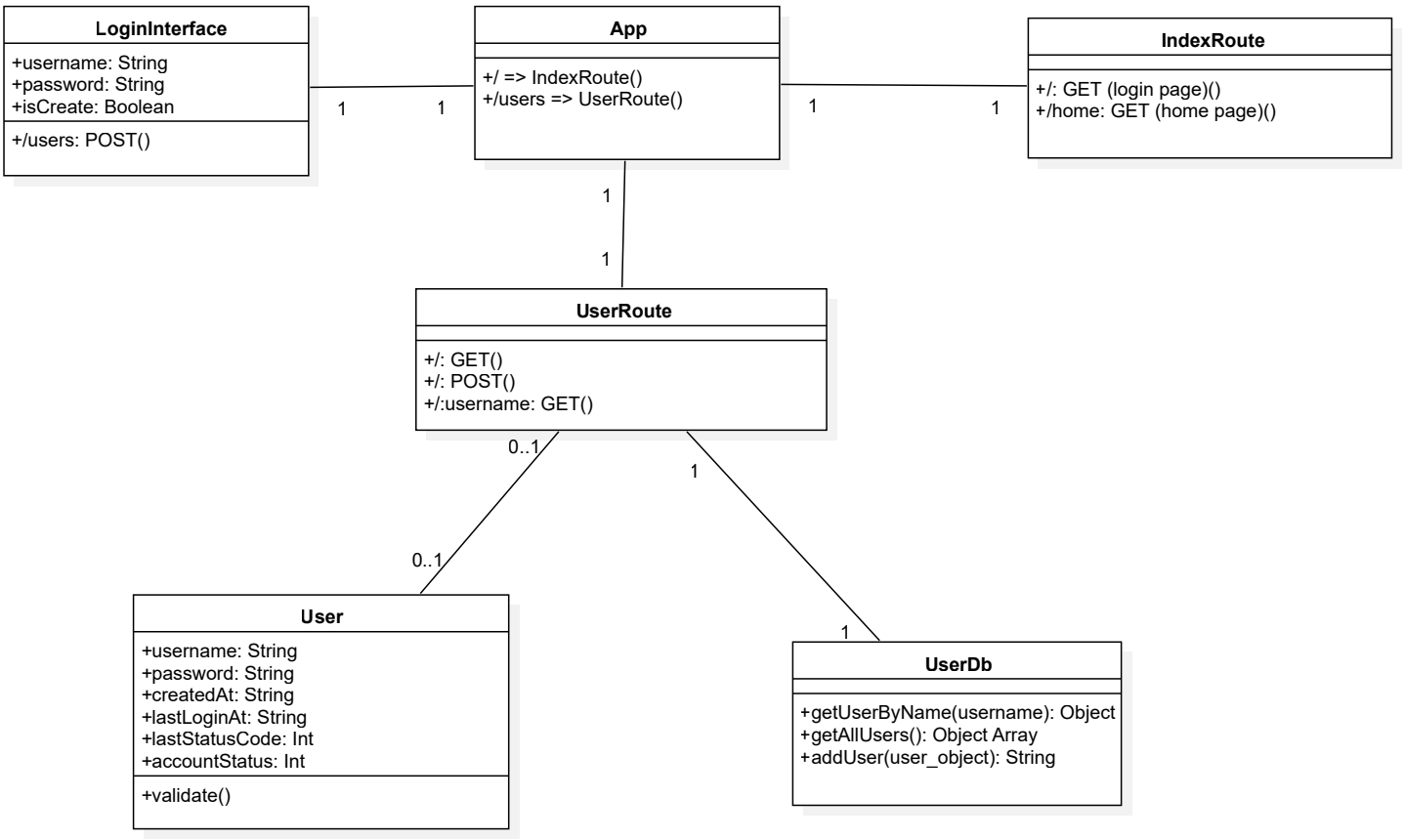
Analysis Classes

Entity Classes: User
Boundary Classes: UserDb, LoginInterface
Control Classes: App, UserRoute, IndexRoute

Basic Flow (Sequence Diagram)



Basic Flow (Class Diagram)



Mapping

LoginInterface

lib/userDb.js

```
'use strict';

var mongoose = require('mongoose');
var user = require('../models/user.js');
var pw = require('../lib/password.js');

var UserModel = mongoose.model('User', user.userSchema);

class userDb {
    constructor(){
    }

    getUserByName(username, callback){
        UserModel.findOne({'username': username}, functio
n(err, user){
            if (err) {
                console.log(err);
                callback(err, false);
            } else {
                console.log("user found: " + user);
                callback(err, user);
```

```
            }
        });
    }

    //getUserById(uid, callback){
    //  UserModel.findOne({'_id': uid}, function(err, use
r){
    //      if (err) {
    //          console.log(err);
    //          callback(err, false);
    //      } else {
    //          console.log("user found: " + user);
    //          callback(err, user);
    //      }
    //  });
    //}

    getAllUsers(callback){
        UserModel.find({}, function(err, users){
            if (err) {
                console.log(err);
                callback(err, false);
            } else {
                console.log("users found: " + users);
                callback(err, users);
            }
        });
    }
```

```
    updateUserAccountStatus(username, status, callback){
        UserModel.update({'username': username},{
            accountStatus: status
        }, function(err, updated_user){
            if (err) {
                console.log(err);
                callback(err, false);
            } else {
                console.log('user successfully logged_out
');
                callback(err, updated_user)
            }
        });
    }

    updateLastLogin(username, status, callback){
        UserModel.update({'username': username},{
            accountStatus: status,
            lastLoginAt: new Date()
        }, function(err, updated_user){
            if(err){
                console.log(err);
                callback(err, false);
            } else {
                console.log('user successfully logged_in'
);
                callback(err, true);
```

```
            }
        });
    }


    addUser(data, callback){
        var user = new UserModel();
        user.username = data.username;
        user.password = pw.createDBPassword(data.password
);
        user.status = data.status;
        user.type = data.type;

        user.save(function(err){
            if (err) {
                console.log("error in creating user");
                console.log(err);
                callback(err, false);
            } else {
                console.log("user successfully created");
                callback(err, user._id);
            }
        });
    }

}

module.exports = userDb;
```

App

app.js

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var pug = require('pug');
var esession = require('express-session');

// Load routes
var routes = require('./routes/index');
var users = require('./routes/users');
var messages = require('./routes/messages');
var test = require('./routes/test');

var config = require('./config');

var app = express();
var http = require('http').Server(app);
var io = require('socket.io')(http);

// view engine setup
app.set('views', path.join(__dirname, 'views'));
```

```
app.set('view engine', 'pug');

// uncomment after placing your favicon in /public
//app.use(favicon(path.join(__dirname, 'public', 'favicon
.ico')));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

//if there is a connection, set up for socket stuff
io.on('connection', function(socket){
  app.set('socket', socket);
});

//Setup express session
app.use(esession({
    secret: config.sess_secret,
    resave: true,
    saveUninitialized: true
}));

// Routes //
app.use('/', routes);

app.use('/users', users);
app.use('/messages',messages);
```

```
// TEST ROUTE //
app.use('/test', test);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handlers

// development error handler
// will print stacktrace
if (app.get('env') === 'development') {
  app.use(function(err, req, res, next) {
    res.status(err.status || 500);
    res.render('error', {
      message: err.message,
      error: err
    });
  });
}

// production error handler
// no stacktraces leaked to user
app.use(function(err, req, res, next) {
```

```
  res.status(err.status || 500);
  res.render('error', {
    message: err.message,
    error: {}
  });
});

//LIVE SERVER, LIVEEEE!!
console.log('Server going live...');
http.listen(8080, function(){
  console.log("listening on http://localhost:8080");
});
console.log('Magic is happening on port 8080. \n  Now ope
n http://localhost:8080/ in your browser!');

module.exports = app;
```

IndexRoute

routes/users.js

```
var express = require('express');
var router = express.Router();

var Db = require('../lib/db');
var User = require('../lib/user');
var userDb = require('../lib/userDb')
var pw = require('../lib/password');

/* Retrieve all users */
router.get('/', function(req, res, next) {
  //see if user is logged in
  if (!req.session.username){
    res.redirect(302, '/');
  } else {
    var db = new Db();
    var userDbInst = new userDb();

    //start db connection
    db.start(function(connection){
      //Get all users
      userDbInst.getAllUsers(function(err, user_list){
        //sort by alphabetical order
        user_list.sort(function(a,b){
```

```
            return a.username.localeCompare(b.username);
          });
          //now by online status
          user_list.sort(function(a,b){
            return a.accountStatus - b.accountStatus;
          });

          console.log(user_list);

          db.close(connection, function(ret){
            //TODO: will do a res.render with data in the f
uture
            res.render('directory', {
              username: req.session.username,
              user_list: user_list,
            });
          });
        });
      });
    }
});

/* Register/Login a user */
router.post('/', function(req, res, next){
  var rb = req.body;
  var username = rb.username;
  var password = pw.decrypt(rb.ciphertext, rb.key, rb.iv)
;
```

```
  var isCreate = rb.isCreate;

  var client = new User(username, password);
  client.validate(isCreate, function(err, err_list, matchedUser){
    if(err) {
      console.log(err);
      console.log(err_list);
      res.status(err_list.httpCode).send({errors: err_list})
    } else {
      var sess = req.session;
      if(err_list.httpCode==200 || err_list.httpCode==201)
        sess.username = req.body.username;
      res.status(err_list.httpCode).send({errors: err_list})
    }
  });

});

/* Retrieve a user's record */
router.get('/:username', function(req, res, next){

  if(!req.session.username) {
    res.redirect(302, '/');
  }
```

```
  else {
    var db = new Db();
    var userDbInst = new userDb();

    db.start(function(connection){
      userDbInst.getUserByName(req.params.username, funct
ion(err, user){
        db.close(connection, function(ret){
          if(err){
            console.log(err);
            res.status(500);
          } else {
            if(user == null)
              res.render('fourohfour', {
                err_msg: "You shouldn't be here."
              });
            else
              res.render('profile', {
                user: user
              });
          }
        });
      });
    });
  }
});

module.exports = router;
```

IndexRoute

routes/index.js

```
var express = require('express');
var router = express.Router();

var User = require('../lib/user');
var pw = require('../lib/password');

/* GET home page. */
router.get('/', function(req, res, next) {
  console.log(req.session);
  if(req.session.username)
    res.redirect(302, '/home');
  else {
    var salt = pw.createRandomString(10);
    res.render('login',{
      salt: salt
    });
  }
});

router.get('/home', function(req, res, next) {
  //see if user is logged in
  if (!req.session.username)
    res.redirect(302, '/');
```

```
  else {
    var justCreated = req.param('justCreated');
    res.render('index',{
      username : req.session.username,
      title : 'ESN',
      justCreated : justCreated
    });
  }
});

router.get('/logout', function(req,res,next){
  if (!req.session.username)
    res.redirect(302, '/');
  else {
    User.logout(req.session.username, function(err, succe
ss){
      if(success){
        req.session.username = null;
        //request succeeded, redirect to homepage now
        res.redirect(302, '/');
      } else {
        console.log("Can't log out.");
      }
    });
  }
});

module.exports = router;
```

User

lib/user.js

```
'use strict';

var config = require('../config.js');
var Db = require('../lib/db');
var userDb = require('../lib/userDb');
var pw = require('../lib/password');

class User {
  constructor(name, password, status, type) {
    this.username = name;
    this.password = password;

    //if status is empty, set as green i.e. 1
    if(!status)
      this.status = 1;
    else
      this.status = status;

    //if type is empty
    if(!type)
      this.type = 0;
    else
      this.type = type;
```

```
  }

  //remove your active status and set username in session
 to null
  static logout(username, callback){
    var db = new Db();
    var userDbInst = new userDb();
    var name = username;

    db.start(function(connection){
      userDbInst.updateUserAccountStatus(name, 1, functio
n(err, updatedUser){
        db.close(connection, function(db_err){
          if(!updatedUser || err){
            console.log("wasn't able to logout for some r
eason");
            callback(err, false);
          } else {
            callback(err, true);
          }
        })
      });
    });
  }

  userInputInvalid(username, password, errList) {
    if(username.length < 3) {
      console.log("Username must be at least 3 characters
```

```
 long");
      errList.httpCode = 422;
      errList.usernameLenInvalid = true;
    }
    if(config.bannedUserNames.indexOf(username.toLowerCas
e()) >= 0) {
      console.log("Username " + username + " is banned an
d cannot be used");
      errList.httpCode = 422;
      errList.usernameBanned = true;
    }
    if(password.length < 4) {
      console.log("Password must be at least 4 characters
 long");
      errList.httpCode = 422;
      errList.passwordLenInvalid = true;
    }

    return (errList.usernameLenInvalid || errList.usernam
eBanned || errList.passwordLenInvalid);
  }

  sendServerError(err, errList, callback) {
    console.log(err);
    errList.httpCode = 500; //Server Error
    callback(err, errList, null);
  }
```

```
  //validate if the username and pw is correct
  // if so check if it exists
  // otherwise create a new user
  validate(isCreate, callback) {

    var username = this.username;
    var password = this.password;

    //errList -
    var errList = {httpCode: 200,
                   usernameLenInvalid: false,
                   passwordLenInvalid: false,
                   usernameBanned: false
                  };

    if(this.userInputInvalid(username, password, errList)
) {
      callback(-1, errList, null);
    }
    else {
      //Check if User exists
      var db = new Db();
      var userDbInst = new userDb();
      var user = this;
      db.start(function(connection){
        userDbInst.getUserByName(username, function(err,
matchedUser) {
          if(err) {
```

```
            this.sendServerError(err, errList, callback);
          }
          else {
            if(typeof matchedUser != 'undefined' && match
edUser) {
              //user exists, check the password
              console.log('user exists, check pw');
              if(pw.authenticate(password, matchedUser.pa
ssword)){
                console.log("password matches");
                userDbInst.updateLastLogin(username, 0, f
unction(err, sucess){
                  db.close(connection, function(close_err
){
                    console.log('user status has been udp
ated to logged in');
                    errList.httpCode = 200;
                    callback(err, errList, matchedUser)
                  });
                });
              } else {
                db.close(connection, function(close_err){
                  console.log('password fail');
                  errList.httpCode = 401;
                  callback( err, errList, null);
                });
              }
            }
```

```
            else {
              if(isCreate == "false") {
                db.close(connection, function(close_err){
                  if(err) {
                     this.sendServerError(err, errList, c
allback);
                  }
                  else {
                     console.log(username + " doesn\'t ex
ist; Sending 404 to client");
                     errList.httpCode = 404;
                     callback(err, errList, null);
                  }
                });
              }
              else {
                console.log("Creating new user " + userna
me);
                userDbInst.addUser(user, function(err, ui
d){
                  db.close(connection, function(close_err
){
                    if(err) {
                       this.sendServerError(err, errList,
 callback);
                    }
                    else {
                        if(err){
```

```
                            this.sendServerError(err, errLi
st, callback);
                        }
                        else {
                            console.log("Successfully added
 new user " + username);
                            errList.httpCode = 201;
                            callback(err, errList, matchedU
ser);
                        }
                    }
                  });
                });
              }
            }
          }
        });
      });
    }
  }

}

module.exports = User;
```

Mongoose User Model

```
/*
 * Model: User
 * Model for user collection
 */

var mongoose = require('mongoose');
var Schema = mongoose.Schema;

/*
*       username: string
*       password: string
*       status: 0: undefined, 1: ok (Green), 2: help (yel
low), 3: emergency (red)
*       type: 0: civilian, 1: admin, 2: coordinator
*       lastLoginAt is updated when they logout
*       accountStatus: 0: ACTIVE, 1: INACTIVE
*/
var userSchema = new Schema({
    username: { type: String, required: true, unique: tru
e },
    password: { type: String, required: true },
    createdAt: { type: Date, default: Date.now, required:
 true },
    updatedAt: { type: Date, default: Date.now, require:
true },
    lastLoginAt: { type: Date, default: Date.now, require
```

```
: true },
    status: { type: Number, require: true, default: 0 },
    type: { type: Number, required: true, default: 0 },
    accountStatus: { type: Number, required: true, defaul
t: 0}
});

module.exports = mongoose.model('User', userSchema);
```

UserDb

lib/userDb.js

```
'use strict';

var mongoose = require('mongoose');
var user = require('../models/user.js');
var pw = require('../lib/password.js');

var UserModel = mongoose.model('User', user.userSchema);

class userDb {
    constructor(){
    }

    getUserByName(username, callback){
        UserModel.findOne({'username': username}, functio
n(err, user){
            if (err) {
                console.log(err);
                callback(err, false);
            } else {
                console.log("user found: " + user);
                callback(err, user);
            }
        });
```

```
    }

    //getUserById(uid, callback){
    //  UserModel.findOne({'_id': uid}, function(err, use
r){
    //      if (err) {
    //          console.log(err);
    //          callback(err, false);
    //      } else {
    //          console.log("user found: " + user);
    //          callback(err, user);
    //      }
    //  });
    //}

    getAllUsers(callback){
        UserModel.find({}, function(err, users){
            if (err) {
                console.log(err);
                callback(err, false);
            } else {
                console.log("users found: " + users);
                callback(err, users);
            }
        });
    }

    updateUserAccountStatus(username, status, callback){
```

```
        UserModel.update({'username': username},{
            accountStatus: status
        }, function(err, updated_user){
            if (err) {
                console.log(err);
                callback(err, false);
            } else {
                console.log('user successfully logged_out
');
                callback(err, updated_user)
            }
        });
    }

    updateLastLogin(username, status, callback){
        UserModel.update({'username': username},{
            accountStatus: status,
            lastLoginAt: new Date()
        }, function(err, updated_user){
            if(err){
                console.log(err);
                callback(err, false);
            } else {
                console.log('user successfully logged_in'
);
                callback(err, true);
            }
        });
```

```
    }


    addUser(data, callback){
        var user = new UserModel();
        user.username = data.username;
        user.password = pw.createDBPassword(data.password
);
        user.status = data.status;
        user.type = data.type;

        user.save(function(err){
            if (err) {
                console.log("error in creating user");
                console.log(err);
                callback(err, false);
            } else {
                console.log("user successfully created");
                callback(err, user._id);
            }
        });
    }

}

module.exports = userDb;
```