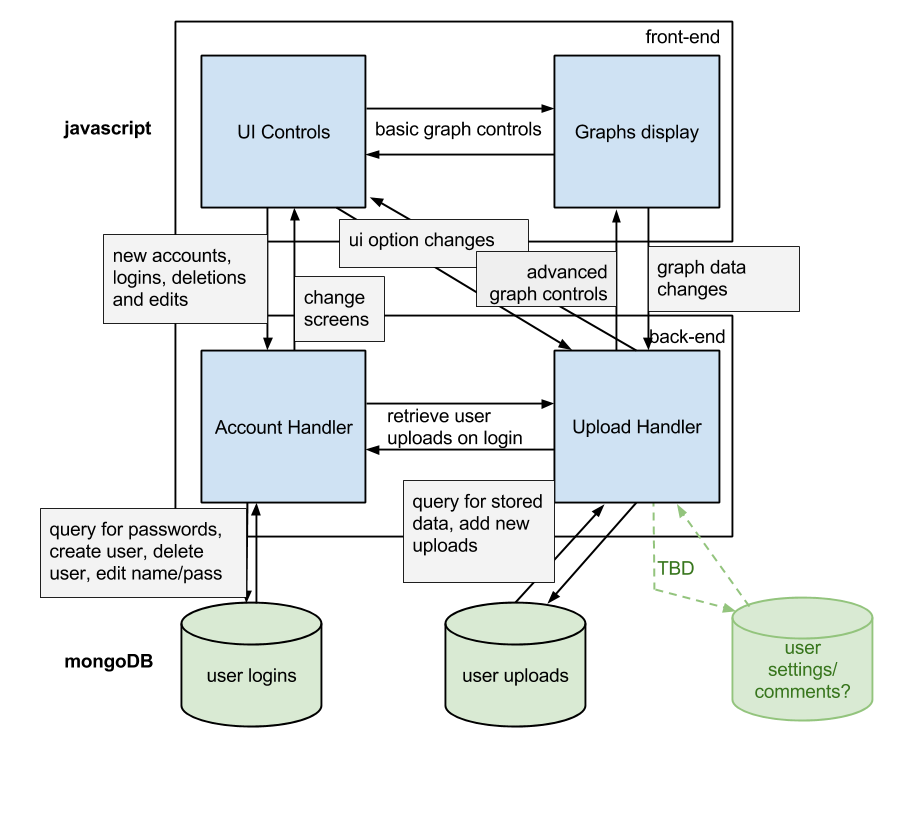
DATABASE:

Logins: each user’s username, password, userID

Uploads: runs uploaded (UID;runid;measure;query;value)

Settings/etc: for other uploaded files that may eventually be handled by the uploader (comments for runs, qrels maybe, etc)

STAGES:

1. UPLOAD:

i. parse file (read first line and go from there, probably?)

a. if NOT a trec results file, place data in "settings"/etc table and return? (later implement display for comments, queries, etc)

ii. place data in table

iii. extract data from table

iv. display extracted data in graphs

2. CREATE ACCOUNT:

i. add uid, username and pass to db

ii. add tables for uid in settings and uploads?

iii. turn on functionality

3. LOGIN TO ACCOUNT:

i. pull up uploads and settings

ii. read contents, display uploads if settings say so

iii. turn on functionality

4. CHANGE GRAPH

i. read db if necessary

ii. extract relevant data

iii. display in graphs

iv. select by voting in a table?

ERRORS: user - malformed upload file, invalid login, connection problems, etc

internal - couldn’t access database, connection problems, etc

LIBRARY CHOICE: Google Charts vs D3

* D3 is more flexible, won’t break if unsupported/upgraded, maybe more suited for this task
* Charts seems a bit easier to use, less of a learning curve

Things not decided: how it will deal with ‘other’ uploads or the ability to add query details, comments etc, the exact mechanism of ‘ranking’, exactly how/where graph data is changed (probably: range sliders and select/deselect on the graph like in the main example [here](http://nvd3.org/index.html), add/remove runs overall on a separate table which definitely needs to call the upload handler to access stored runs, some radial choices/tick boxes for display by measure, query, add/remove graph, etc)

Some things may depend a little on the accepted format of data by the graph library - i.e. how much reformatting needs to be done by the upload handler to move data from the uploaded file or the database to the graph input. Charts can query some databases itself I think, D3 may need some extra stuff tacked on

CLASSES/functions/etc:

1. Handlers:

a. LoginHandler (both login and account creation) [SERVER]

i. accept username and pass

ii. produce + send query

iii. receive response

iv. send update ui/login error message to client

b. UploadHandler [SERVER]

i. accept file

ii. parse file

iii. produce entry for database

iv. format for graph handler

v. send entry/reformat to db/graph handler

c. GraphHandler (won’t be detailed here until library is decided) [CLIENT?]

i. accept data

ii. produce graphs

iii. may have some minor controls to modify its own display

d. ControlHandler [SERVER]

i. accept input

ii. class input (measure, data request, ranking request)

iii. produce + send query (if necessary)

iv. produce ranking, reformat data (if necessary)

v. return data to graph or UI handler

e. UIHandler [CLIENT]

i. accept changed settings

ii. display changes

iii. send selections/entries to server

2. Messages/Events:

a. Client -> Server:

i. login details - Login(username, pass)

ii. new account details - NewUser(username, pass)

iii. changed account details - ChangeUser(username, pass)

iv. upload - Upload(results)

v. graph control - GetData(runs, measures, queries???)

vi. ranking control - GetOrder(runs, measures???)

b. Server -> Database:

i. login details - query(Username, Pass = x?)

ii. new account details - addDocument(Username, Pass, generated UID)

iii. delete account - deleteDocument(Username, Pass)

iii. change account details - updateDocument(Username, Pass, NewPass)

iv. upload - addDocument(UID, Parsed File)

v. get data - query(UID, runid)

vi. delete - deleteDocument(UID, runid)

c. Server -> Client:

i. appearance update - change(control, to)

ii. graph update - change(graph, values)

d. Database -> Server:

i. login result - Login(Pass)

ii. new account creation/deletion/update confirmation - true

iii. upload/deletion confirmation - true

iv. get data - return(runid, data)