# Chat.com Documentation for Assignment 2

## For ICT3813 Web Framework Development

**By Paul Collins s5198025**

**Git**

*Describe the layout of your git repository and the approach you took for version control.*

Git version control was used and published via Github.com. Two separate repositories were used for server and client code respectively to facilitate deployment. Simple version numbers were given to each repo. 3 Continuous Integration branches were set up: develop, test, production. While there was no dev-ops process involved this branch design enables the integration of a SDLC at a later point if required. All development commits were added to develop branch. Production branch is the client facing branch to be deployed. Test branch is used for integration and User Acceptance Testing. Experimental branches were used to verify new features. ‘Milestone’ branches were used as an easier alternative to retrieving prior versions of files at particular stages eg ‘git checkout milestone1 app.component.ts’ .

Modularization was incorporated into the project by subdividing code into smaller focused files where practicable facilitating multiple users being able to push and pull to the remote origin without merge conflicts.

Frequent commits were used with the convention ‘<branch-name>:<category><description (optional)> ‘. Git aliases were used to facilitate regular commits without requirements for individual naming eg ‘misc = "!git add . && git commit -m $(git branch-name)': miscellaneous updates"'. More significant commits were manually named.

**Data Structures**
Describe the main data structures used in the program. For example, how the users and groups are represented.

All entities with the exception of messages are stored in a mongdb database collection (‘dbcollection’). Each instance of an entity is stored as a document within this collection:

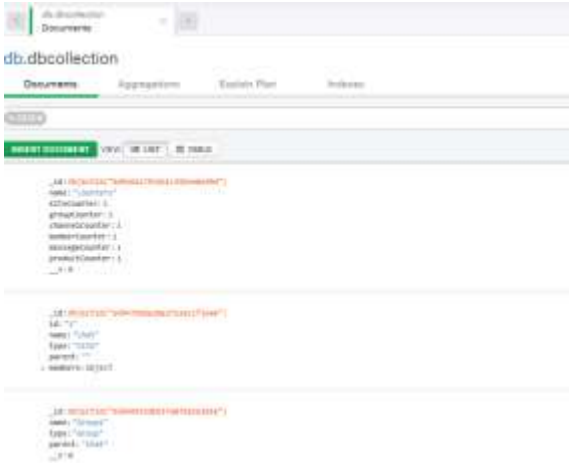


*Figure 1 dbcollection entities example*

The entities in this collection are able to be linked via an informal entity relationship where the ‘parent’ property acts as a foreign key that refers to the ‘name’ (primary key) property of another document. This forms a hierarchical ‘parent-child’ relationship.

| Collection | Entity/primary key | Parent/foreign key |
|---|---|---|
| dbcollection | Site | Null |
| dbcollection | Group | Site |
| dbcollection | Channel | Group |
| dbcollection | Member | Site |
| dbcollection | Member | Group |
| dbcollection | Member | Channel |
| Messages | Message | Channel |
| Messages | Message | Member |

These entities were mirrored in the server as Mongoosejs schemas and on the client side as both interface and class.

*Figure 2Example Mongoosejs Schema*

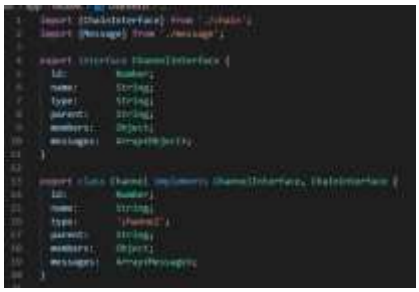

*Figure 3Example client side interface and class*

**Documentation - REST API**

The Angular front end should communicate with the Node.js server using a REST API. describe each route provided, parameters, return values, and what it does.

| url end point | Verb | Description |
|---|---|---|
| / | GET | Provides a html page with a summary of end points |
| api/group | Get | Retrieves object using query string arguments |
| api/group | Post | Inserts or updates using request body |
| api/channel | Get | Retrieves object using query string arguments |
| api/channel | Post | Inserts or updates using request body |
| api/member | Get | Retrieves object using query string arguments |
| api/member | Post | Inserts or updates using request body |
| api/message | Get | Retrieves object using query string arguments |
| api/message | Post | Inserts or updates using request body |
| api/groups | Get | Retrieves objects using query string arguments |
| api/channels | Get | Retrieves objects using query string arguments |
| api/members | Get | Retrieves objects using query string arguments |
| api/messages | Get | Retrieves objects using query string arguments |
| api/login | Post | Verifies username and password of site member |

**Documentation - Angular Architecture**

Angular Entities

| Type | Name | Description |
|---|---|---|
| Component | App | |
| Component | Account | |
| Component | Chat | |
| Component | Detail | |
| Component | Home | |
| Component | Login | |
| Component | Navbar | |
| Component | Register | |
| Service | Controller | Used as an abstraction layer between the persistence service (DbService) and the components |
| Service | Socket | Interface for socket.io. |
| Service | Db | Interface for the server application |