Software Design Group 27

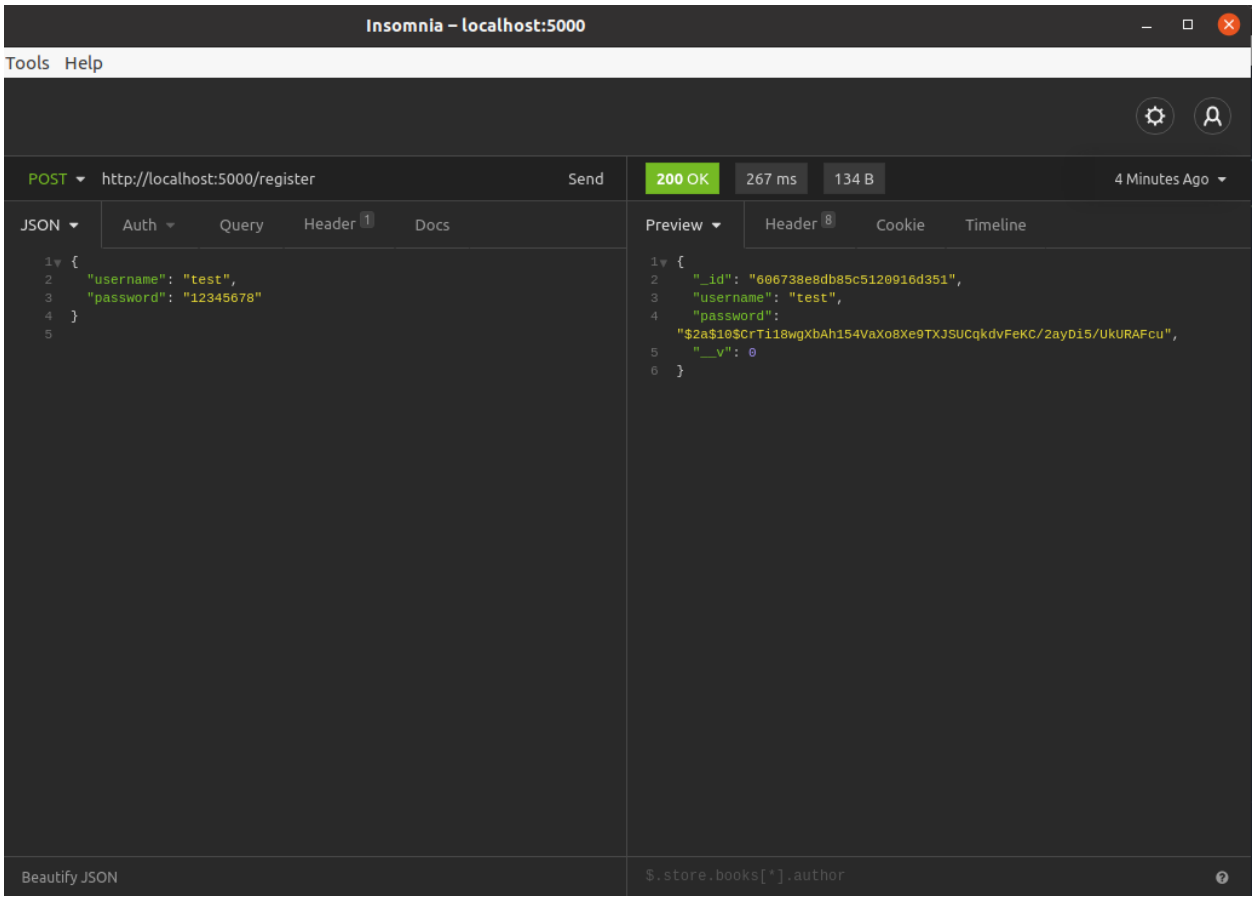
Mariana Villalobos | Gerardo Lopez | Jeff Onyemachi

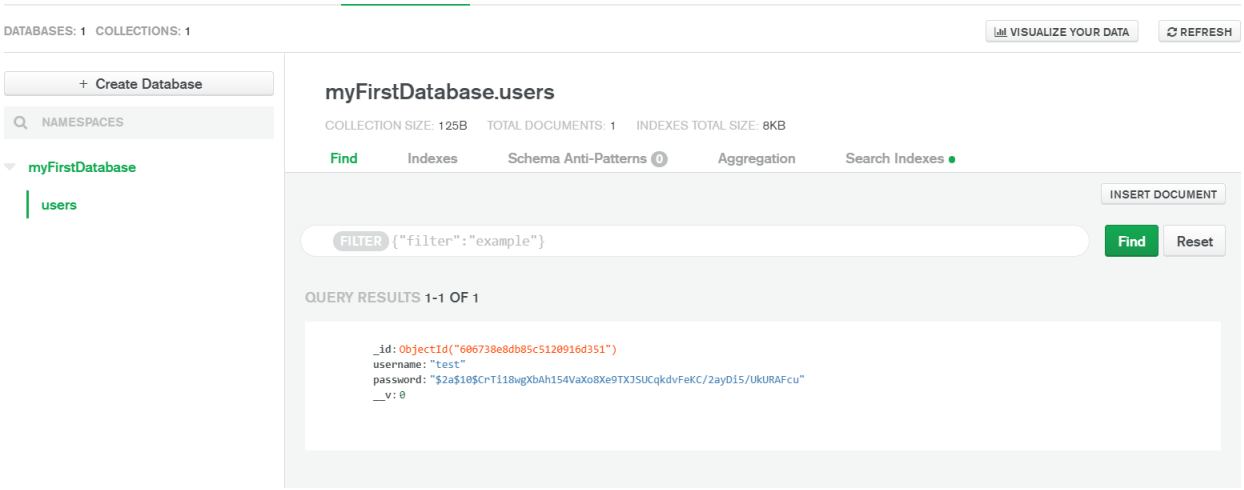
Answer these questions:

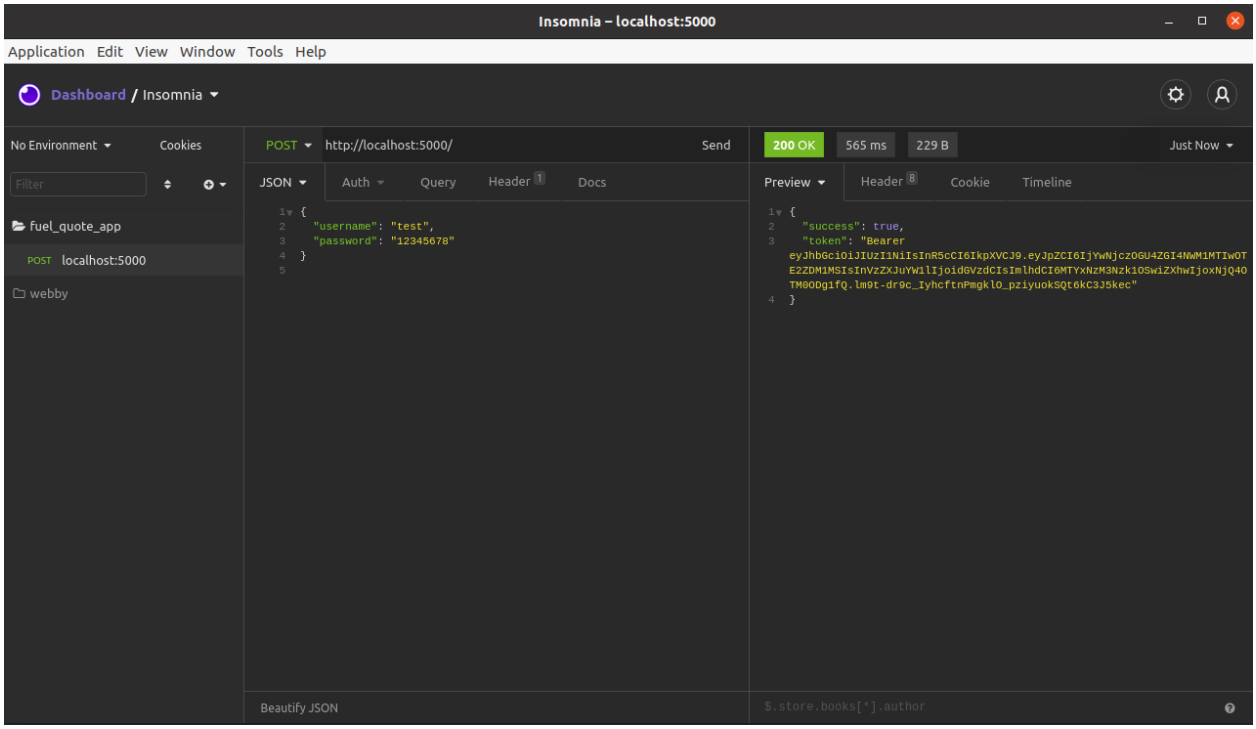
1. Provide link to GitHub repository for TAs to view the code.(5 points)
https://github.com/jerry-lopez/Software-Design-Group27

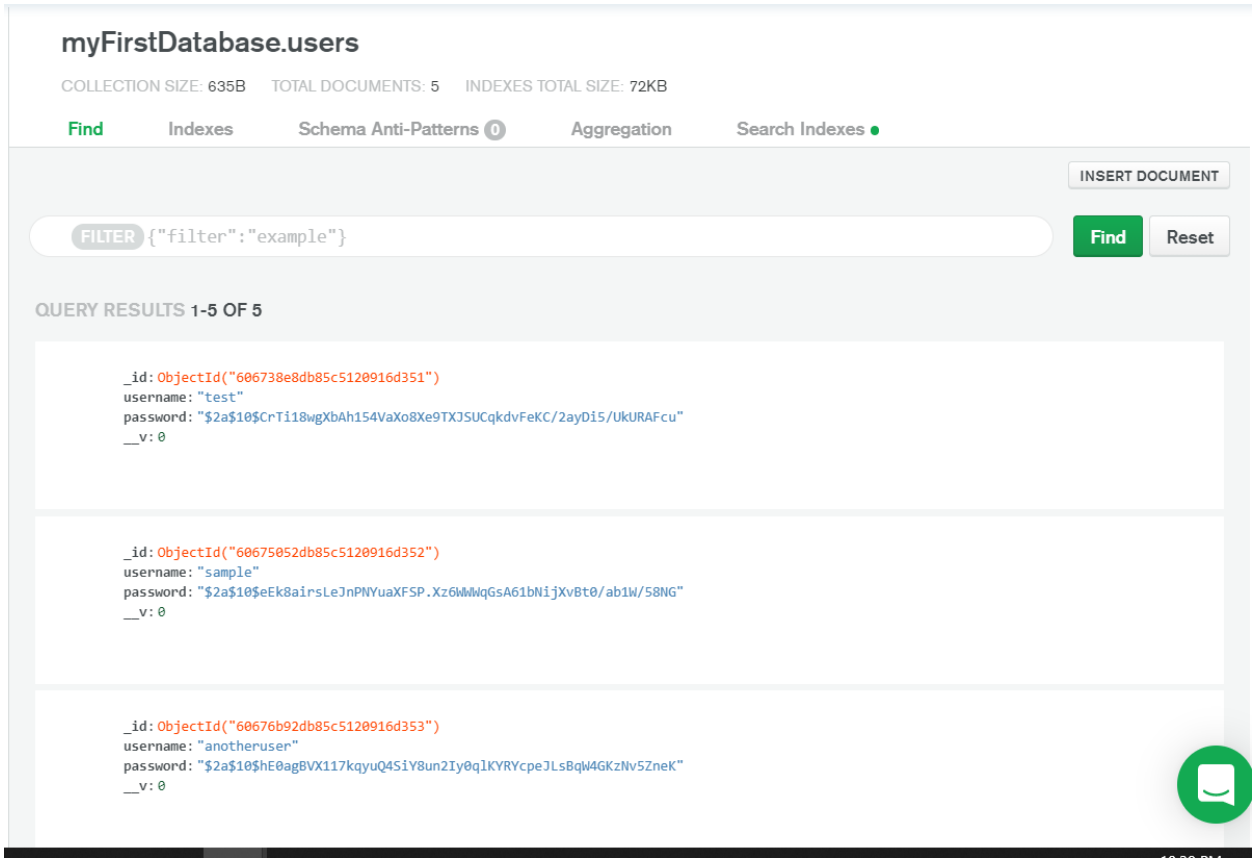
2. Provide SQL statements to create database.(3 points)

For our database we used MongoDB, a NoSQL database. Here are the screenshots of the statements used to create the schemas with mongoose and mongoDB. Also below will be screenshots of the sample users in our mongoDB Atlas that were created using Insomnia (A tool for testing HTTP Requests)




DATABASES: 1 COLLECTIONS: 1
VISUALIZE YOUR DATA
REFRESH
+ Create Database
NAMESPACES
myFirstDatabase
users
myFirstDatabase.users
COLLECTION SIZE: 125B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 8KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER {"filter":"example"}
Find
Reset
QUERY RESULTS 1-1 OF 1
_id: ObjectId("606738e8db85c5120916d351")
username: "test"
password: "$2a$10$CrTi18wgXbAh154VaXo8Xe9TXJSUCqkdvFeKC/2ayDi5/UkURAFcu"
__v: 0


Insomnia – localhost:5000
Application Edit View Window Tools Help
Dashboard / Insomnia
No Environment
Cookies
Filter
fuel_quote_app
POST localhost:5000
webby
POST
http://localhost:5000/
Send
JSON
Auth
Query
Header 1
Docs
{
"username": "test",
"password": "12345678"
}
200 OK
565 ms
229 B
Just Now
Preview
Header 8
Cookie
Timeline
{
"success": true,
"token": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYwNjczOGU4ZGI4NWM1MTIwOTE2ZDM1MSIsInVzZXJuYW1lIjoidGVzdCIsImlhdCI6MTYxNzM3Nzk1OSwiZXhwIjoxNjQ4OTM0ODg1fQ.lm9t-dr9c_IyhcftnPmgklO_pziyuoksQt6kC3J5kec"
}
Beautify JSON
$.store.books[*].author



backend > models > JS User.js > ...

```
1   const mongoose = require("mongoose");
2   const Schema = mongoose.Schema;
3
4   const userSchema = new Schema({
5       username: {
6           type: String,
7           minlength: 4,
8           maxlength: 50,
9           required: true,
10          unique: true
11      },
12      password: {
13          type: String,
14          required: true,
15          minlength: 8
16      }
17  });
18
19  module.exports = User = mongoose.model("users", userSchema);
```

backend > models > JS ClientInfo.js > ...

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const clientSchema = new Schema({
    fullname: {
        type: String,
        required: true,
        maxlength: 50
    },
    addressOne: {
        type: String,
        required: true,
        maxlength: 100
    },
    addressTwo: {
        type: String,
        maxlength: 100
    },
    city: {
        type: String,
        required: true,
        maxlength: 100
    },
    state: {
        type: String,
        enum: ['CA', 'TX'],
        required: true,
        minlength: 2,
        maxlength: 2
    },
    zipcode: {
        type: Number,
        min: 5,
        max: 9,
        required: true
    }
});

module.exports = Client = mongoose.model("clientInfo", clientSchema);
```

3. Rerun the code coverage report and provide it. (2 points)

We sent an email regarding this part of the project.

4. IMPORTANT: list who did what within the group. TAs should be able to validate in GitHub, otherwise team members who didn't contribute will receive a ZERO.

Mariana Villalobos: Created validation files for input validation for registration and for Client Profile using Validator and isEmpty modules. Ran HTTP testing requests for registration and login verification. Attempted to do code coverage, ran into some issues and was not able to complete it in time.

Gerardo Lopez: Created MongoDB Cluster and connected it to the backend server. Added User authentication for Login and registration and state management with redux for node.js. Added Encryption and password hashing for user credentials before adding them to the database. Created database models using mongoose schemas for users and client information. Added private routes for components to only be reached by logged in users.