**Homework – REST API #2**

This homework is splitted into three parts:

* Database part
* REST API part
* ExpressJS part

This is the 2nd and 3rd ExpressJS parts.

In this homework we are going to write an express.js RestFul API server to simulate the Friend Circle in WeChat with MongoDB.

Part I: Database Part – You can ignore now

**Database Design**

* Start running MongoDB in your local or in your preferred cloud enviroment (e.g. MLab), create a new database called wechat.
* In database wechat, we will have three collections:
  + user - to store users
  + post - to store posts in Friend Circle
  + like - to store likes for posts
* Every document in user collection should have:
  + id (You can use the default \_id created by mongoDB)
  + name
  + gender
  + location
  + bio
* Every document in post collection should have:
  + id (You can use the default \_id created by mongoDB)
  + content
  + created\_at - the time when you create this post
  + userId - which user this post belongs to
* Every document in like collection should have:
  + id (You can use the default \_id created by mongoDB)
  + userId - who makes the like
  + postId - which post this like belongs to
* **MMake Schemas for those three collections in your express server**

After we have the basic structure and design for what our database will look like, it's time to create the Schemas in express server using mongoose.

* Create Schema for user collection.
* Create Schema for post collection.
* Create Schema for like collection.

**Create Models based on Schemas**

Once you have the schema ready, you need to create three different Models using mongoose based on those three Schemas. Ideally, you should export those Modals using module.exports to make each one a reusable Javascript module.

**Part II – ExpressJS App**

**Use Models to define express routes to handle different http request**

Now it's a time to write express routes handlers which will respond to http requests with the Models we have. Below is an example of how to create a route to handler GET request at "/users":

// Define a handler to handle GET request at '/'

app.get('/users', function(req, res) {

// use find() method in the User model to get all user documents

UserModel.find({}, function(err, users) {

// If there is an issue to get result, throw the error

if (err) throw err;

// once you get all users, use res.status(xxx).json({..})

// to send all the users back as the response for the http request

res.status(200).json({users});

});

});

**Part III – REST API**

User routes

* GET /users: Get all users from user collection
* GET /users/:userId: Get a single user document based on the userId route parameter
* POST /users: Create a new user document in user collection of MongoDB. When you send POST request, you need to put everything you need for user creating (name, gender, etc.) in the request body.
* PUT /users/:userId: Modify a single user document based on the userId route parameter. When you send PUT request, you need to put everything you want to modify (name, gender, etc.) in the request body.
* DELETE /users/:userId: Delete a single document from user collection based on the userId route parameter.

**Posts Routes**

* GET /posts: Get all posts from post collection
* GET /posts/:userId: Get a single post document based on the postId route parameter
* POST /posts: Create a new post document in post collection of MongoDB. When you send POST request, you need to put everything you need for post creating (content, etc) in the request body.
* PUT /posts/:postId: Modify a single post document based on the postId route parameter. When you send PUT request, you need to put everything you want to modify (content, etc) in the request body.
* DELETE /posts/:postId: Delete a single document from post collection based on the postId route parameter.

**Likes Routes**

* GET /likes: Get all likes from like collection
* GET /likes/:userId: Get a single like document based on the likeId route parameter
* POST /likes: Create a new like document in like collection of MongoDB. When you send POST request, you need to put everything you need for like creating (userId, postId) in the request body.
* PUT /likes/:likeId: Modify a single like document based on the likeId route parameter. When you send PUT request, you need to put everything you want to modify (userId, postId) in the request body.
* DELETE /likes/:likeId: Delete a single document from like collection based on the likeId route parameter.