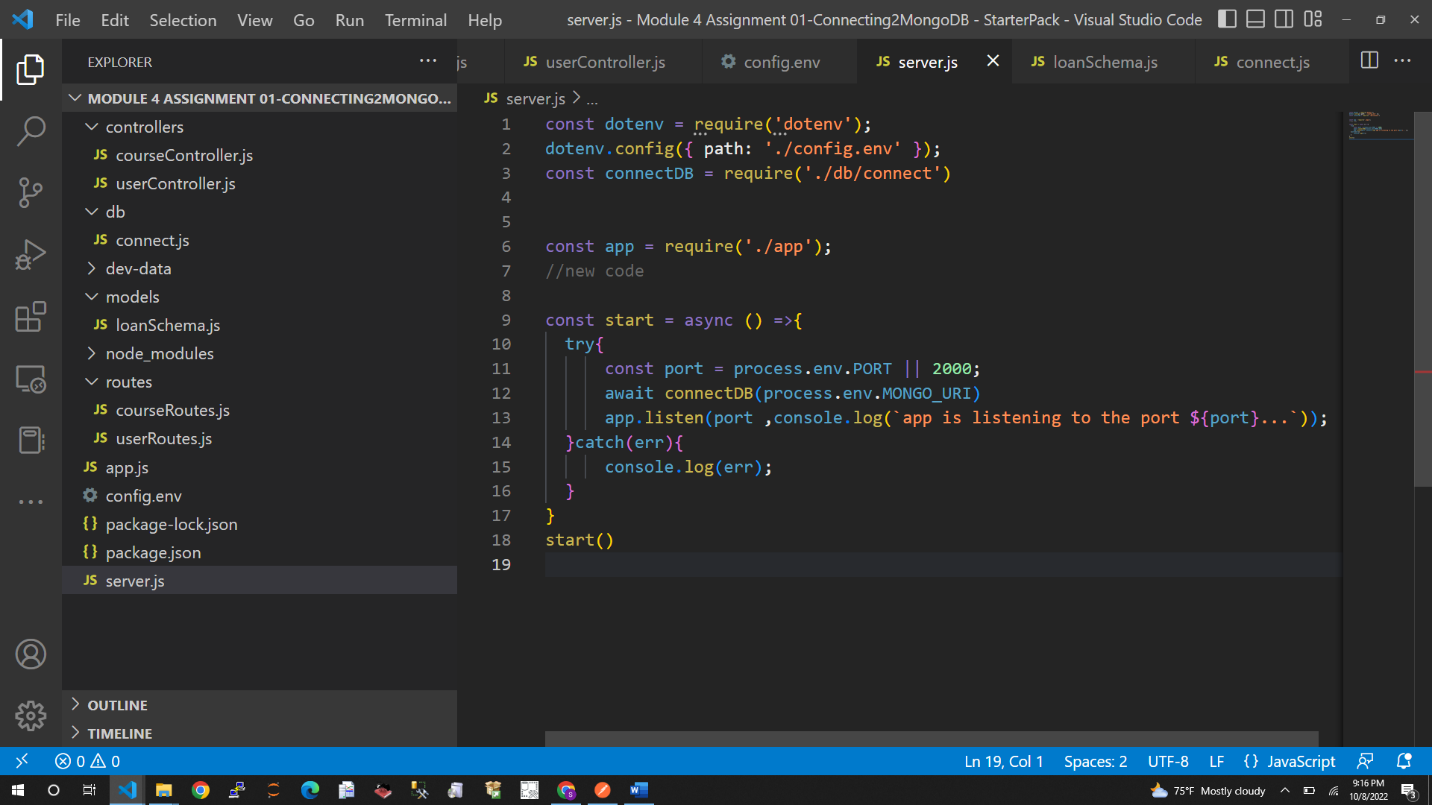
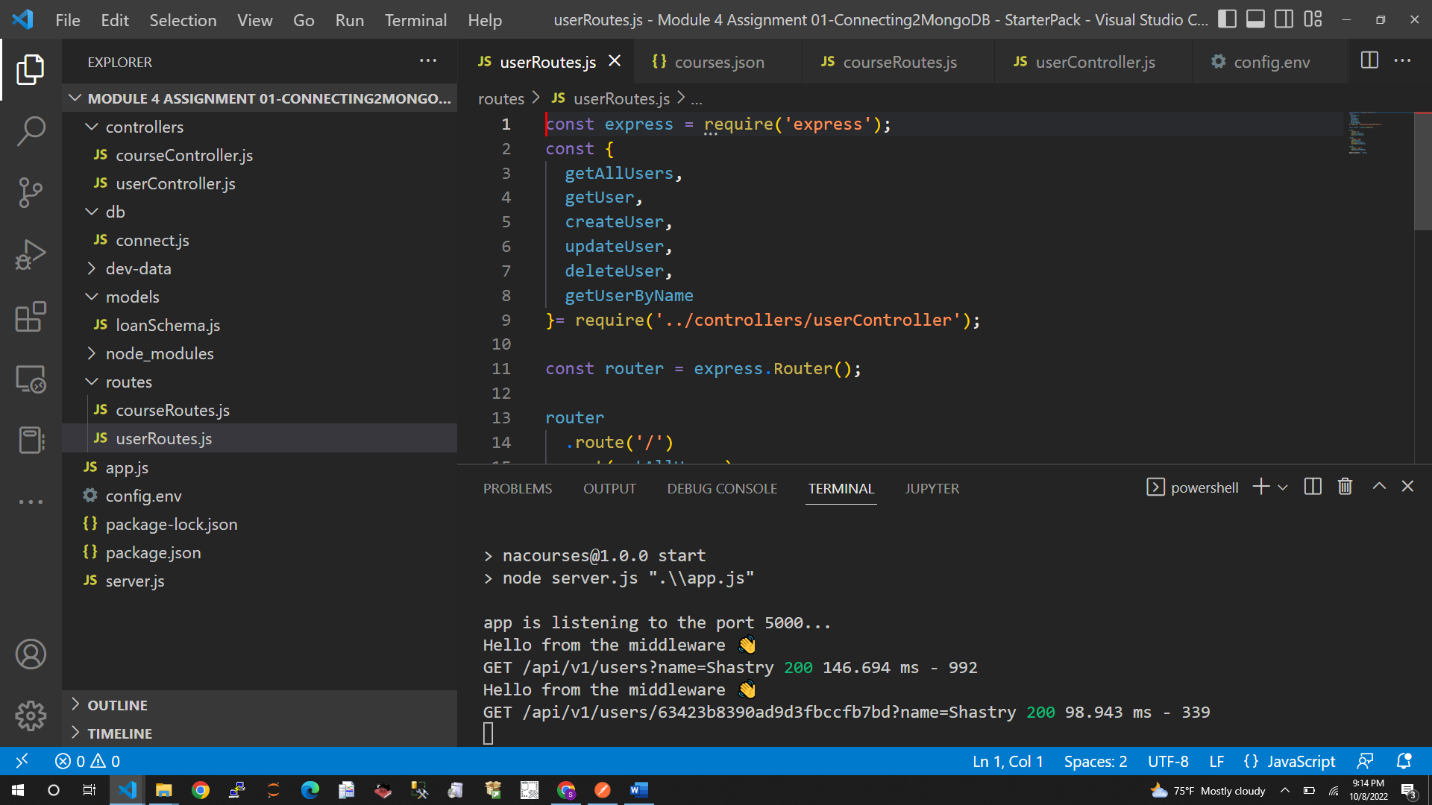
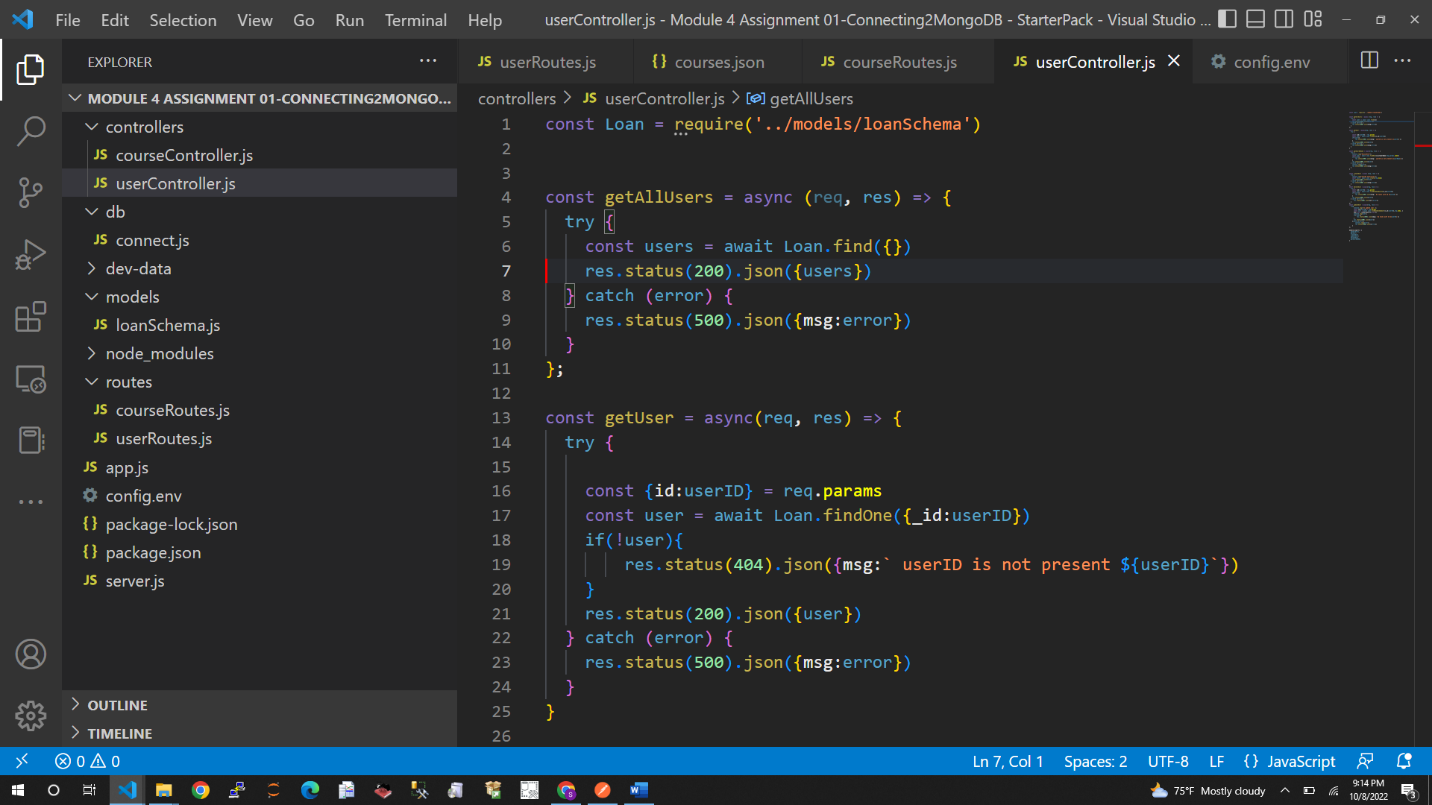
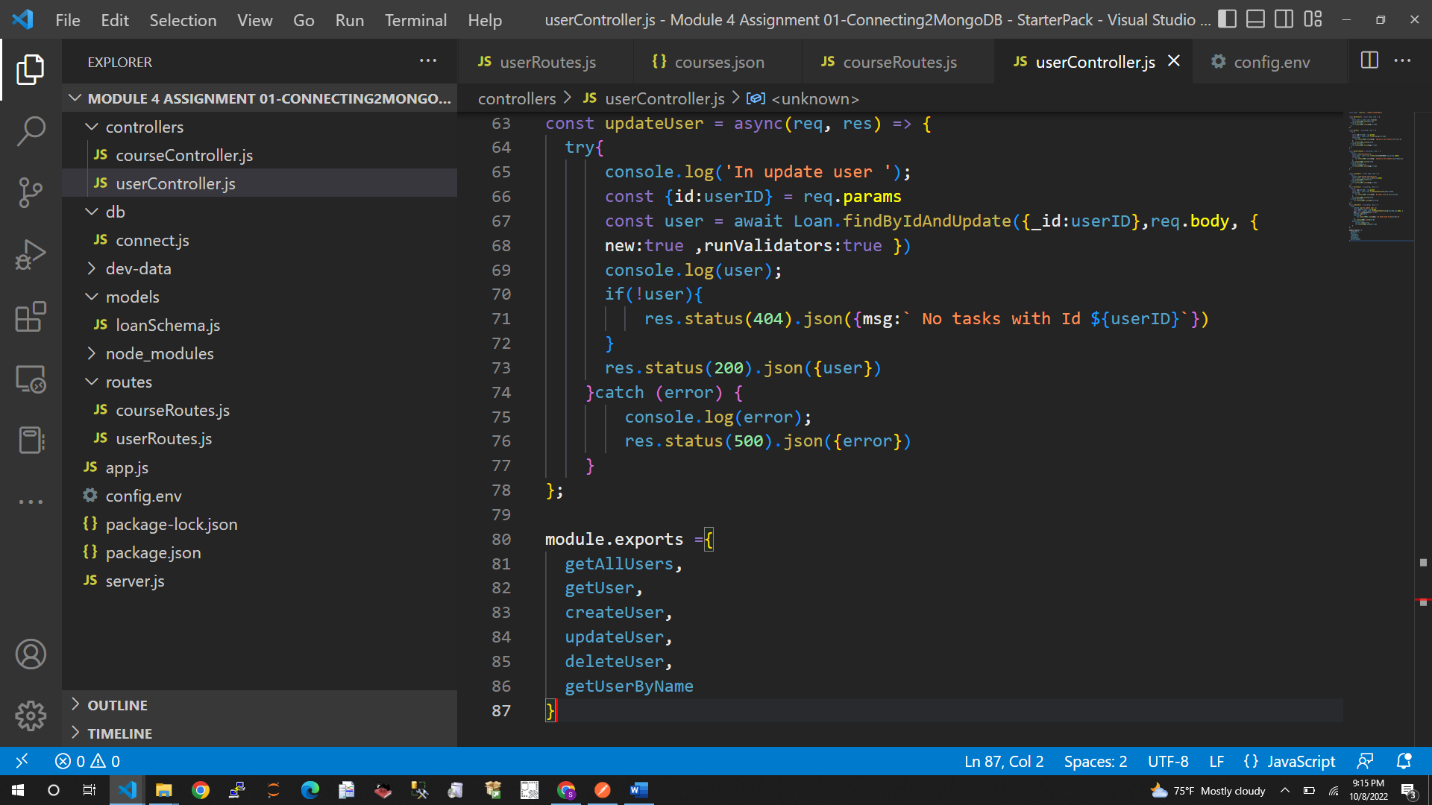
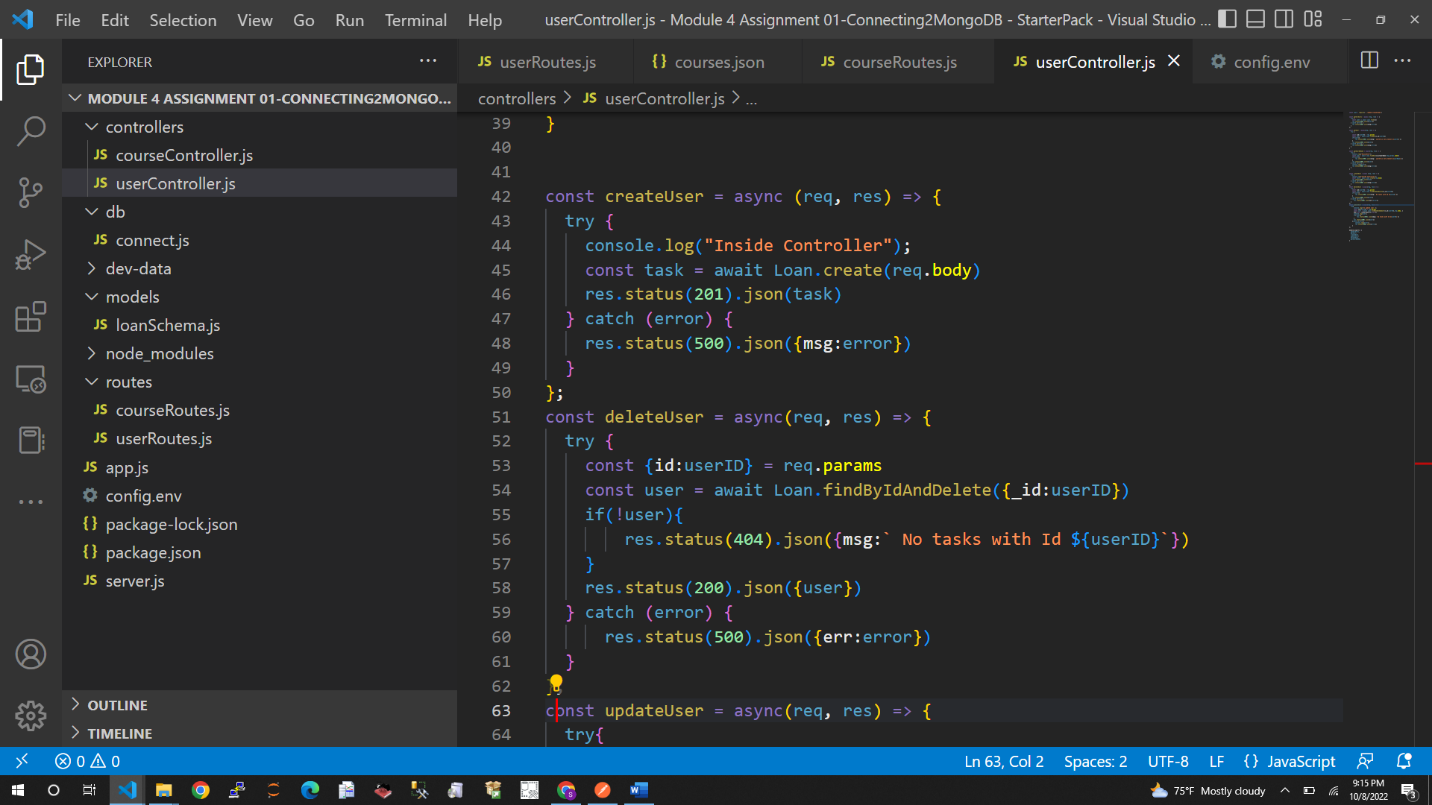
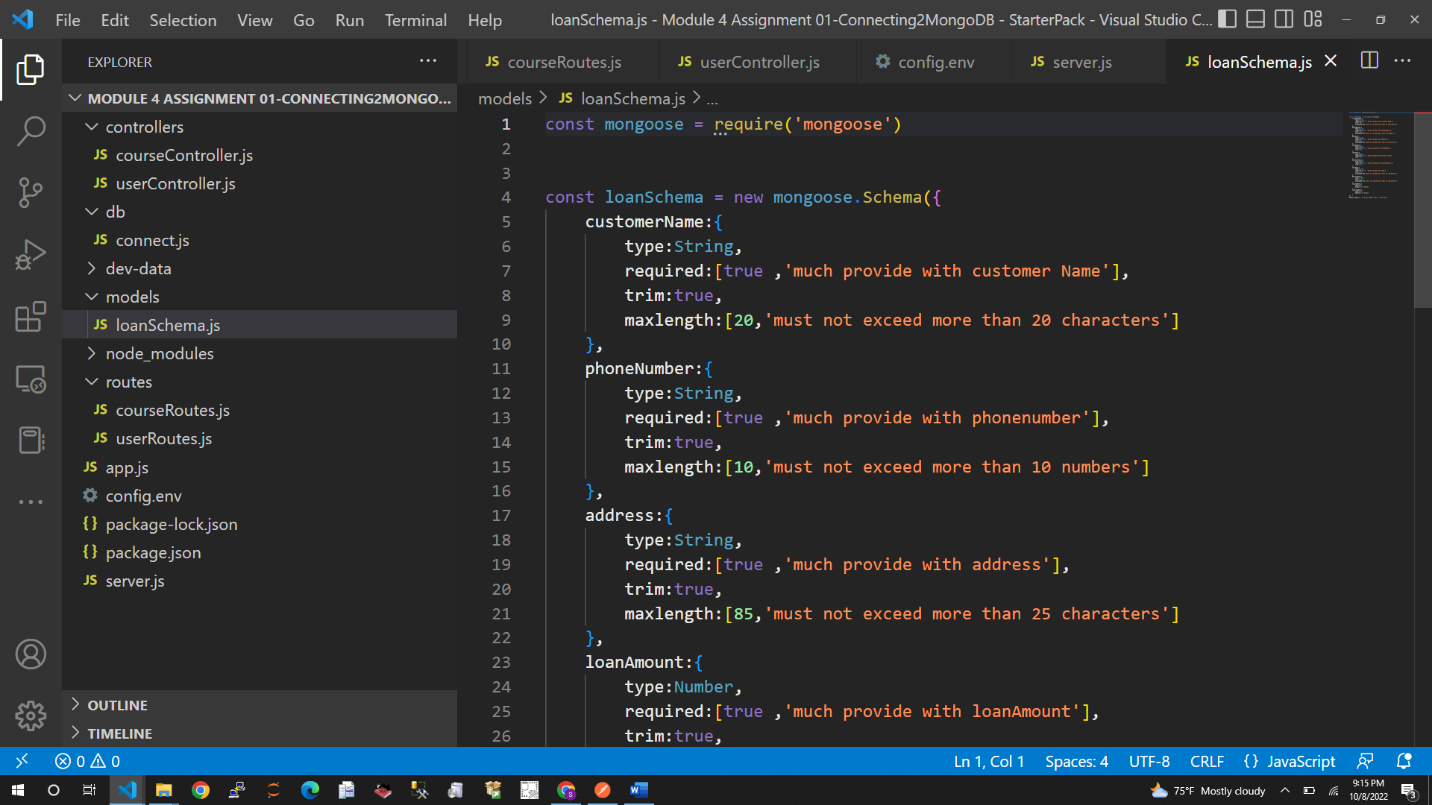
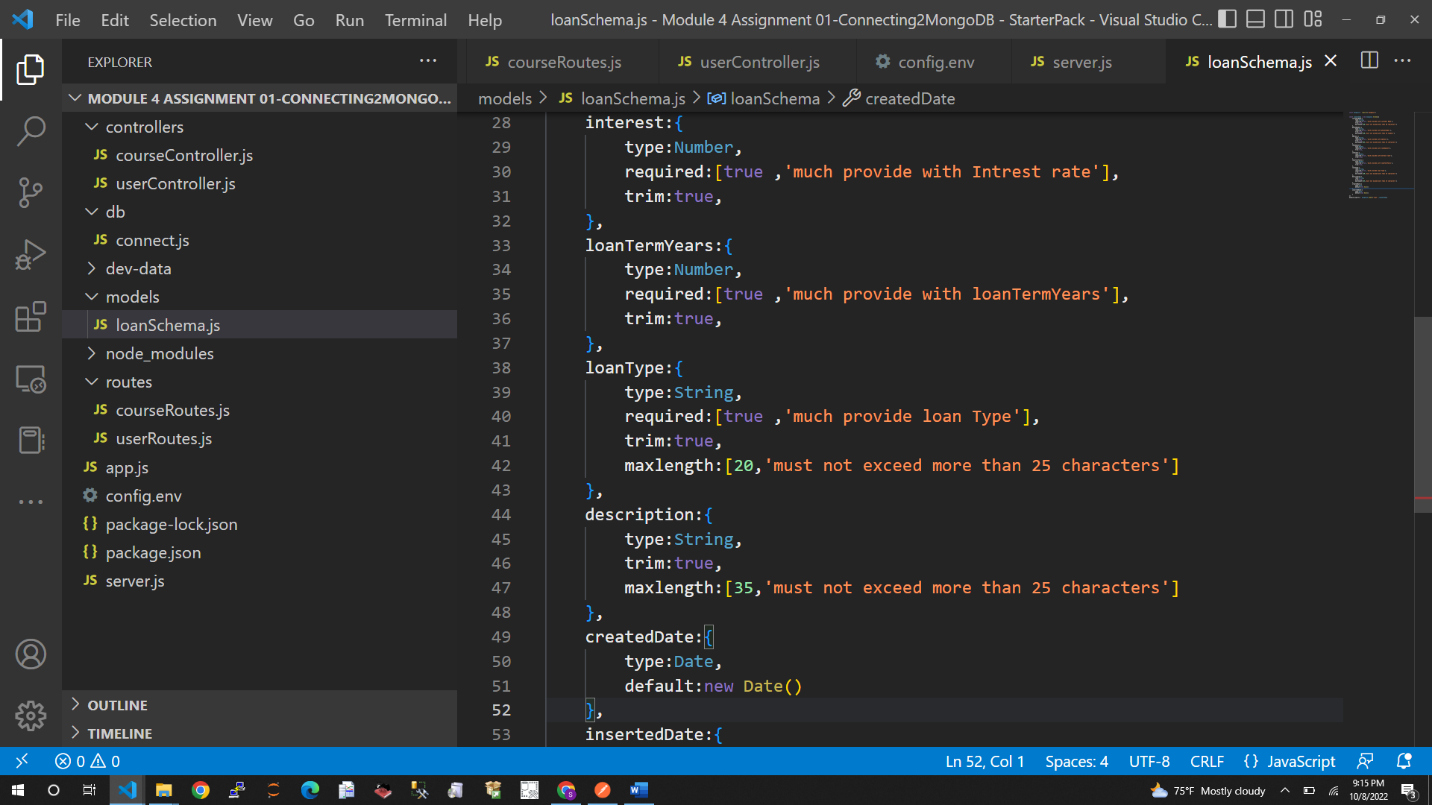
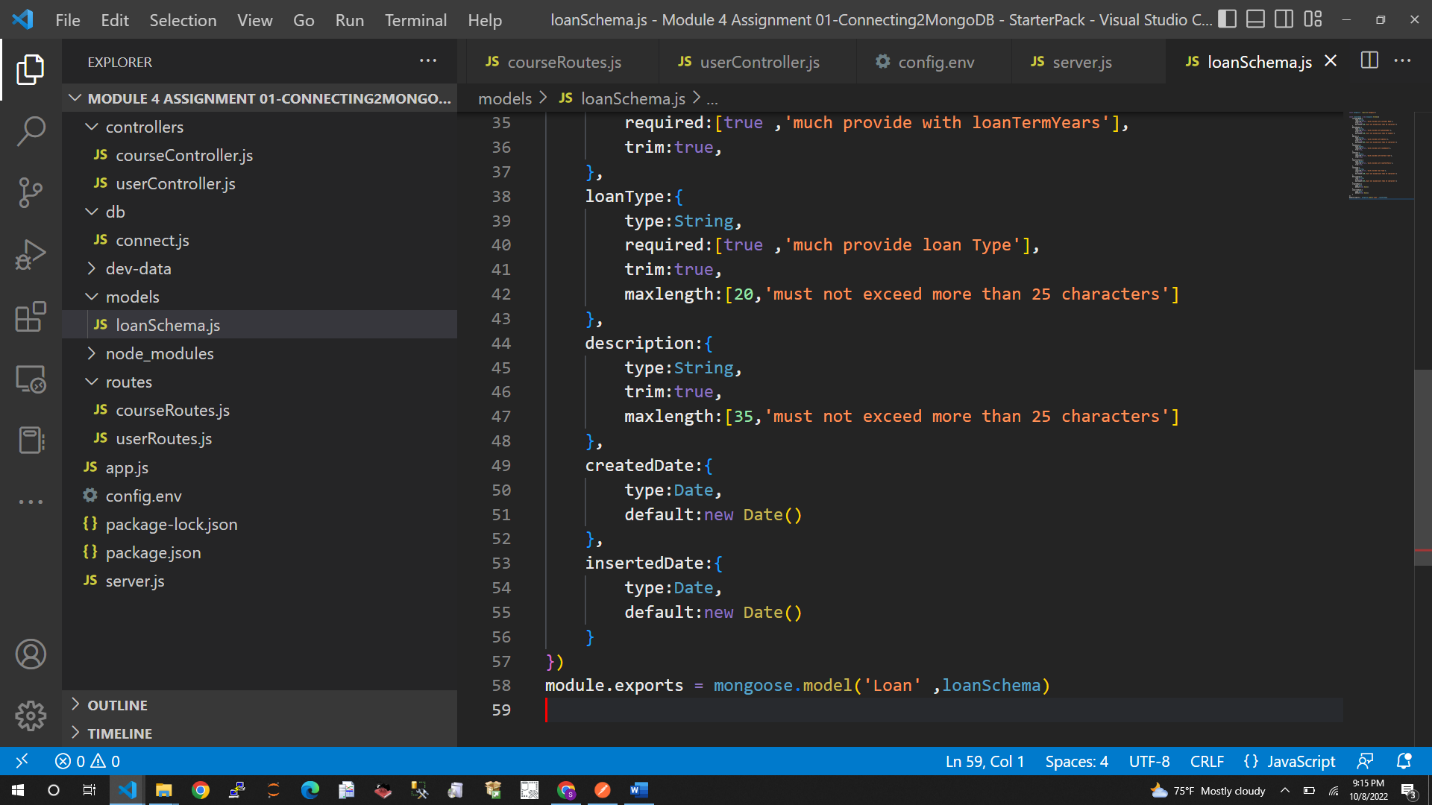
**Swetha Shree Byllahali Ananthaswamy**

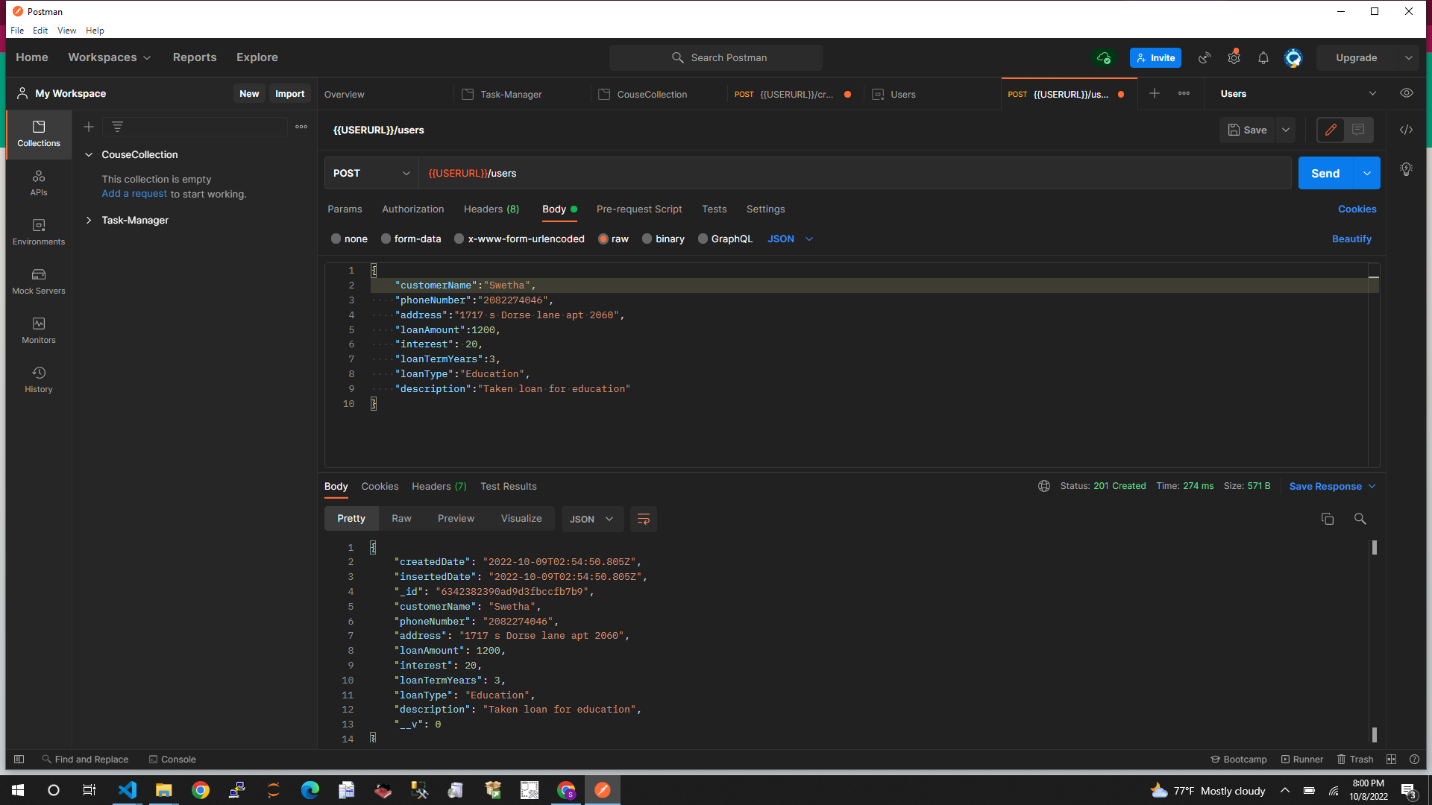
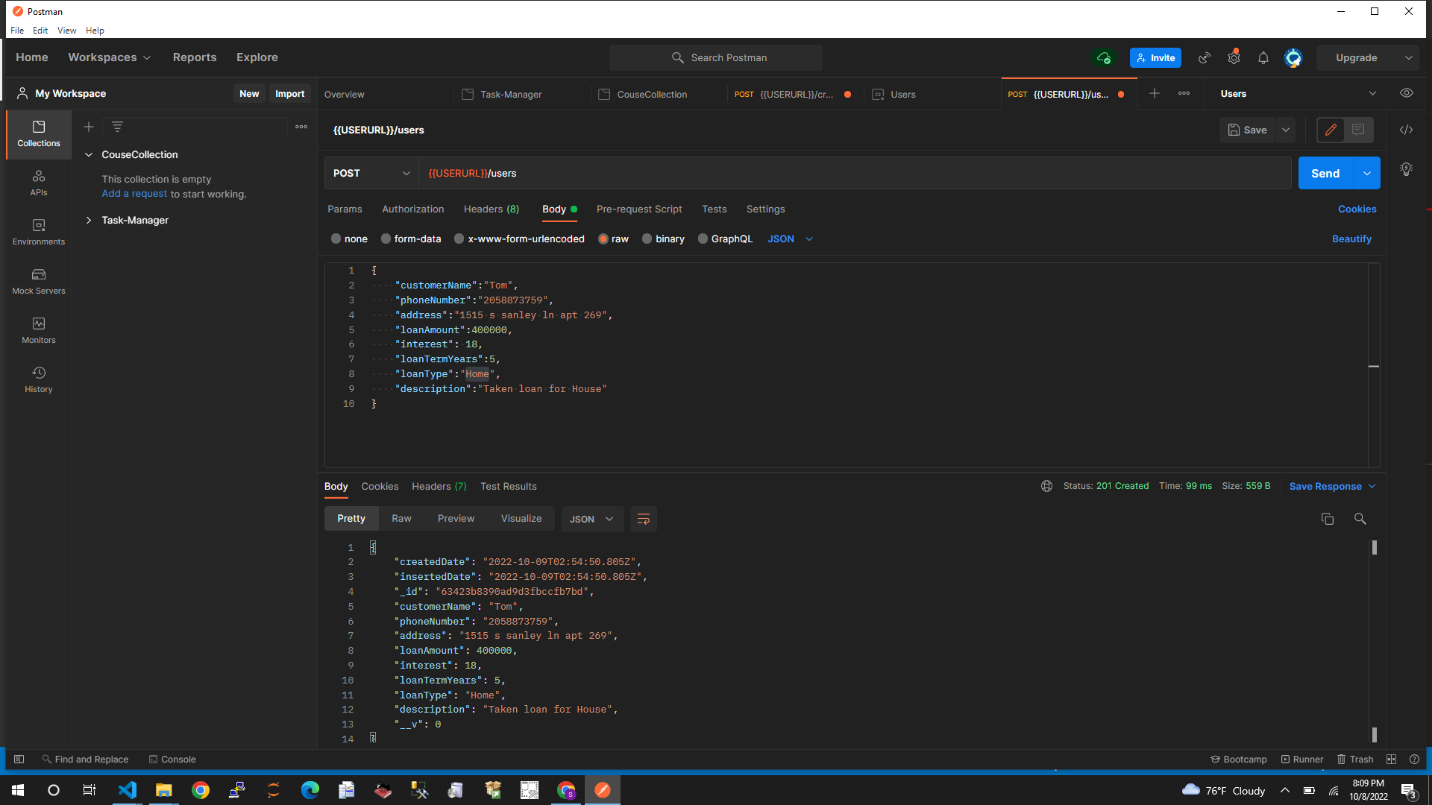
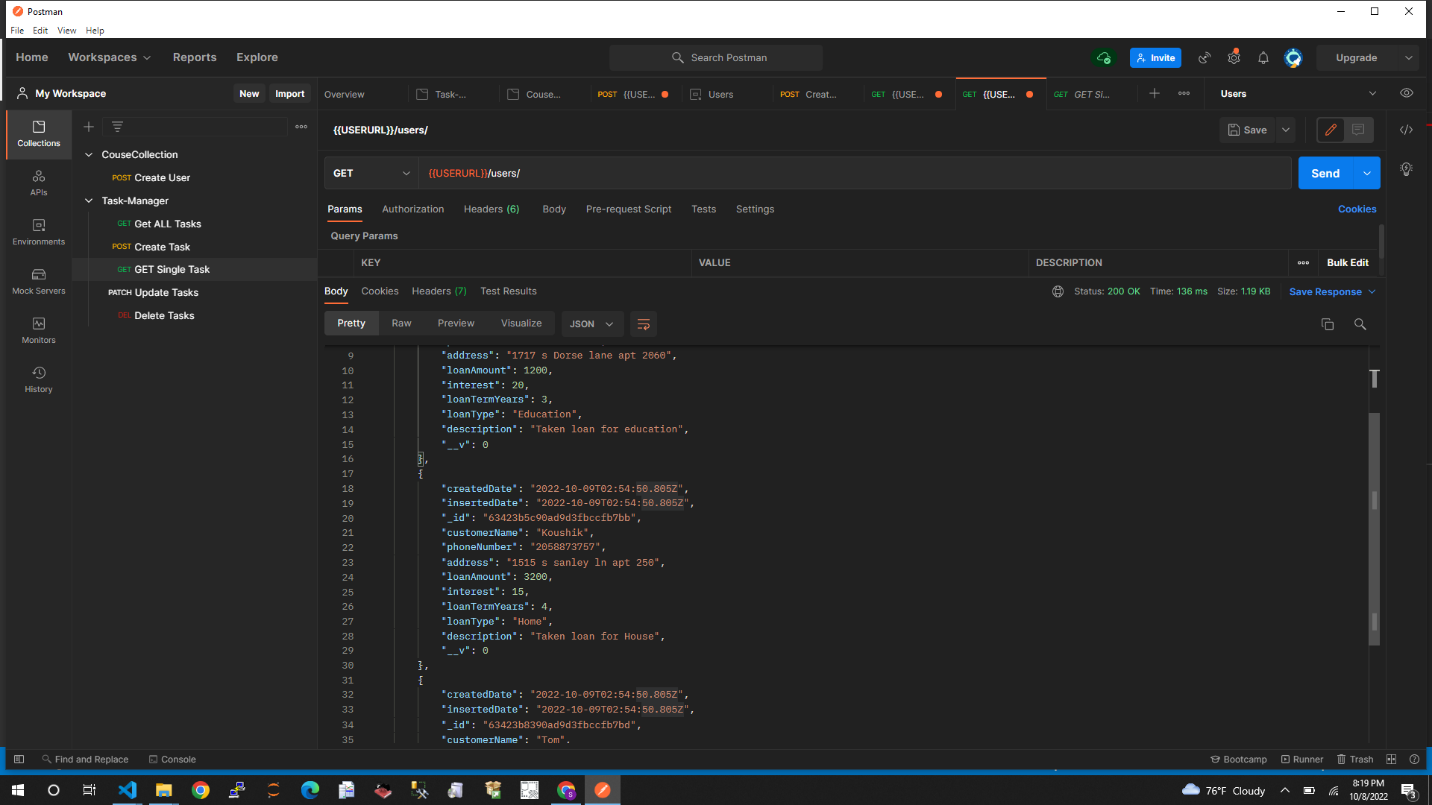
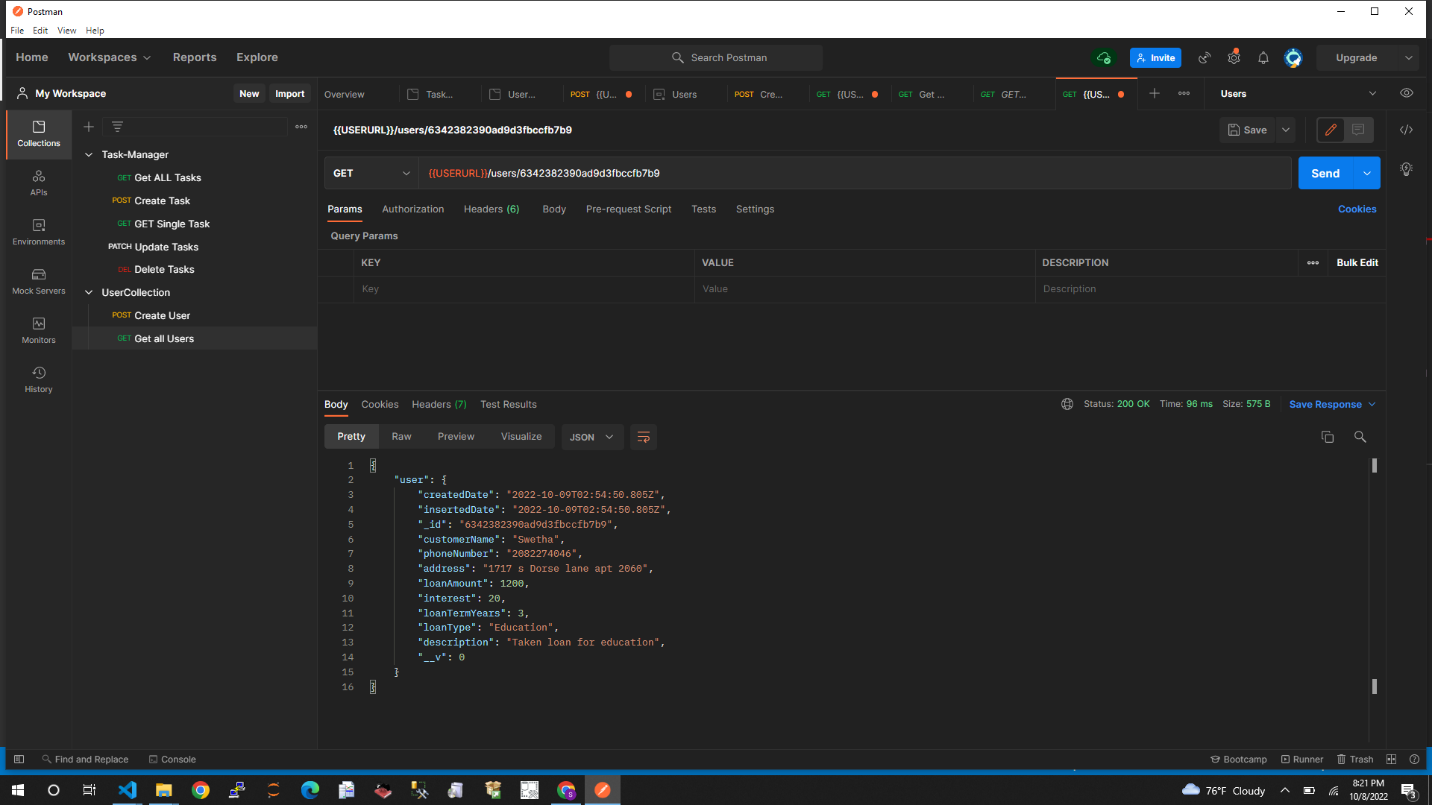
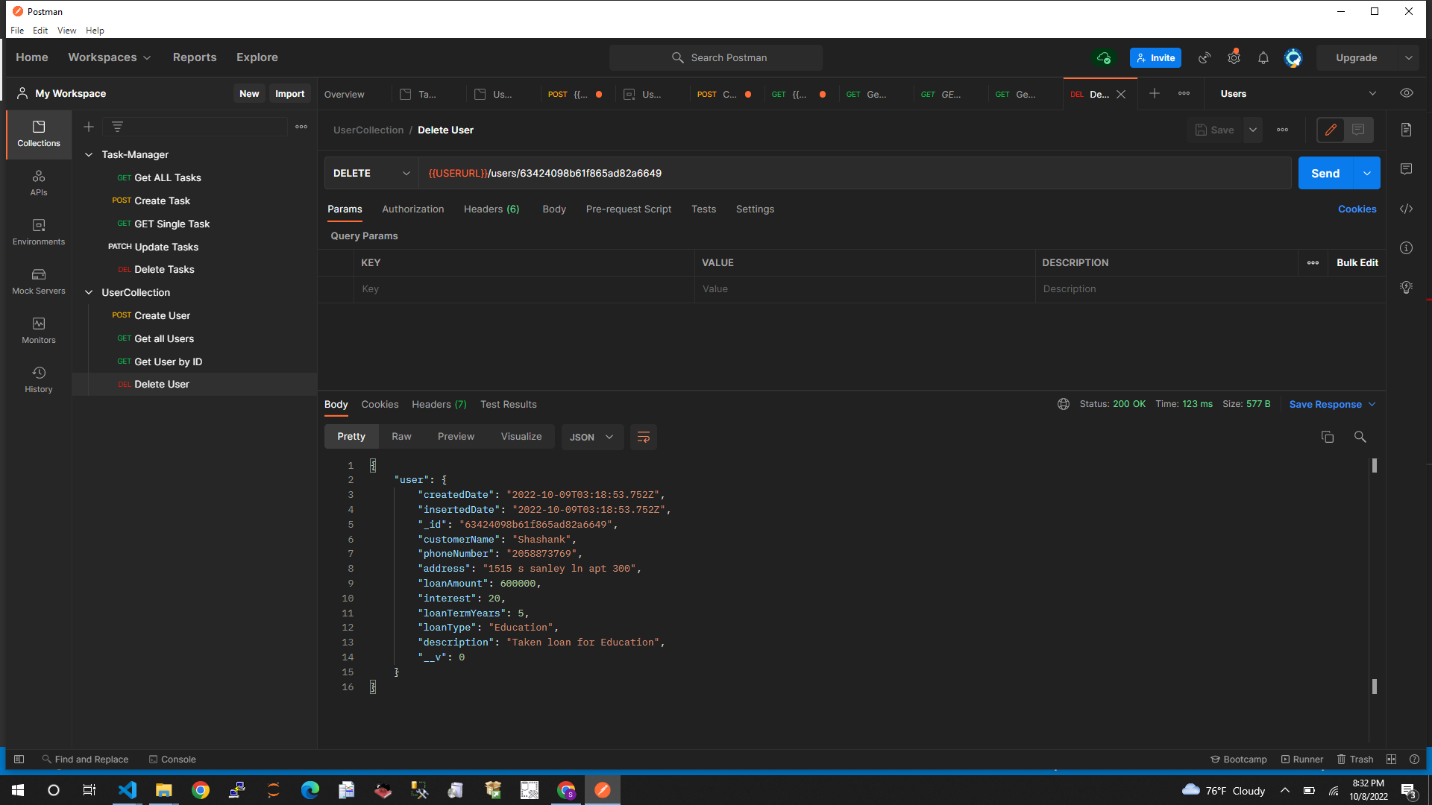
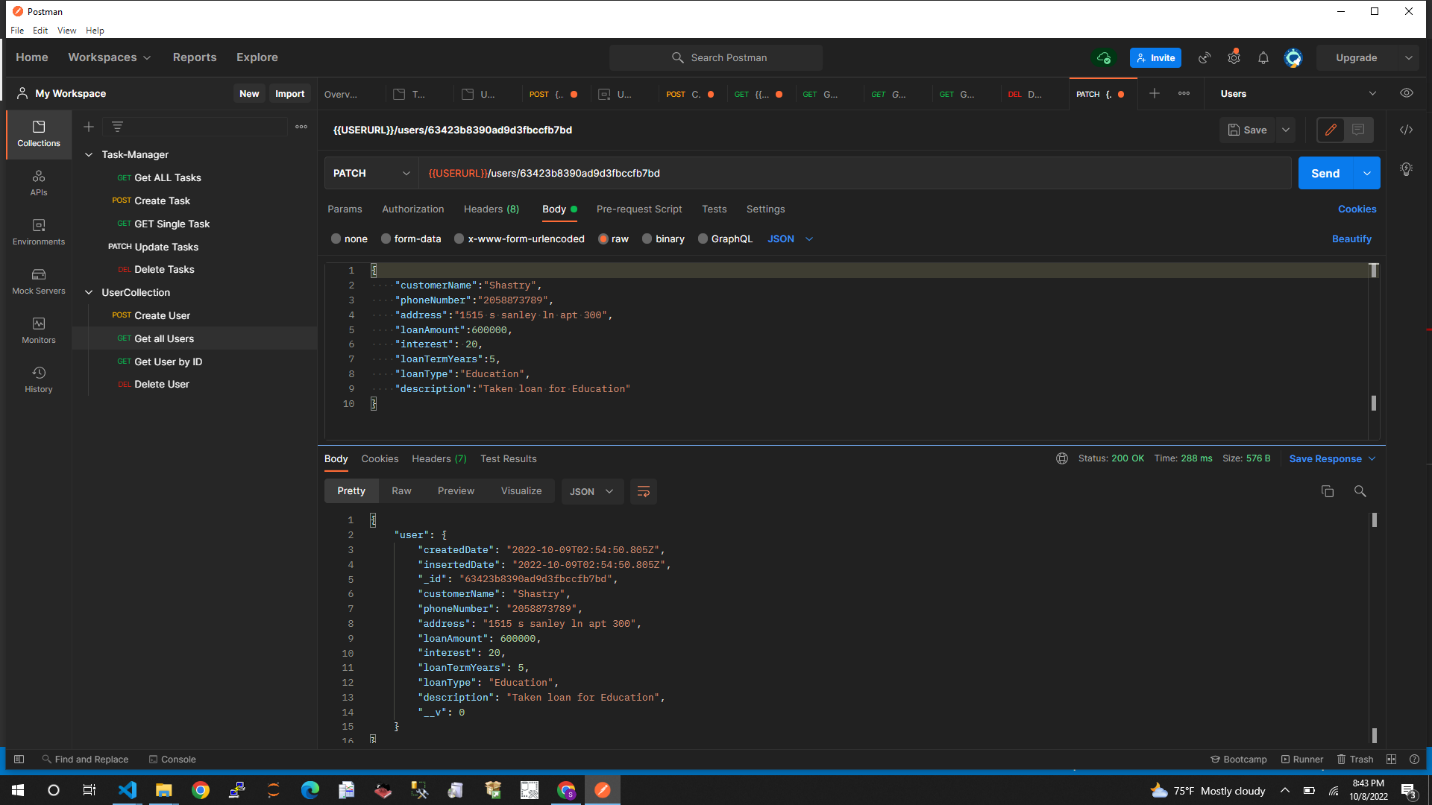
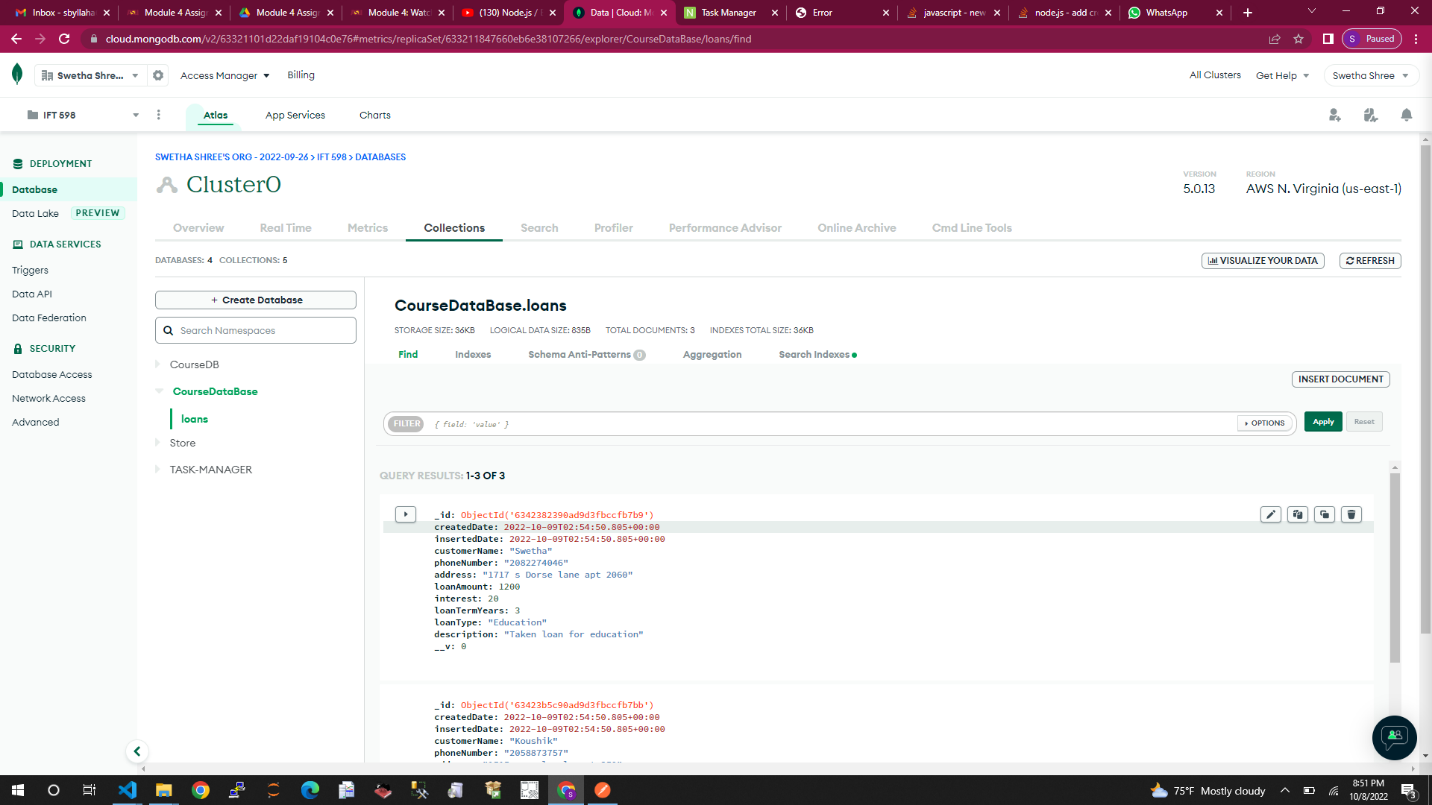
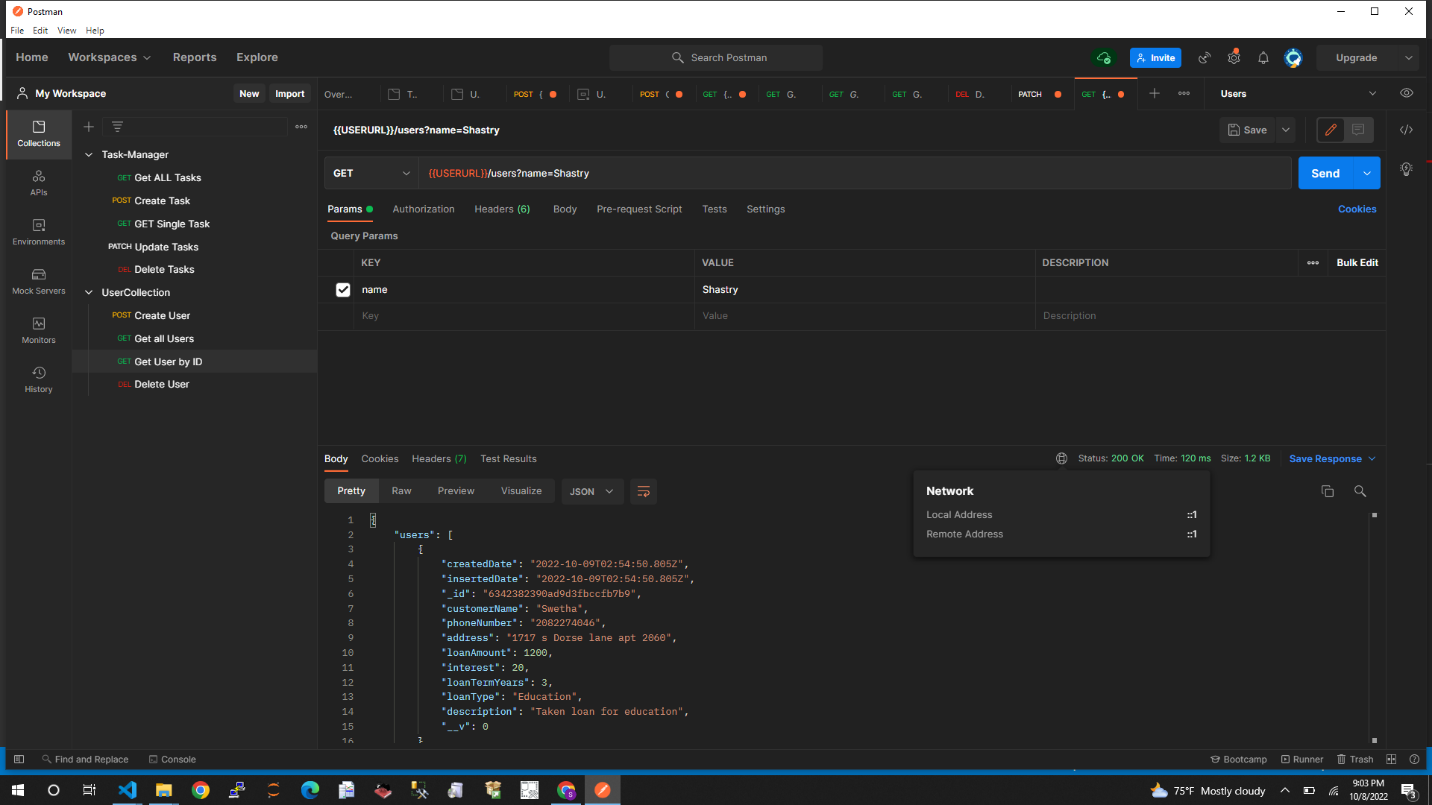
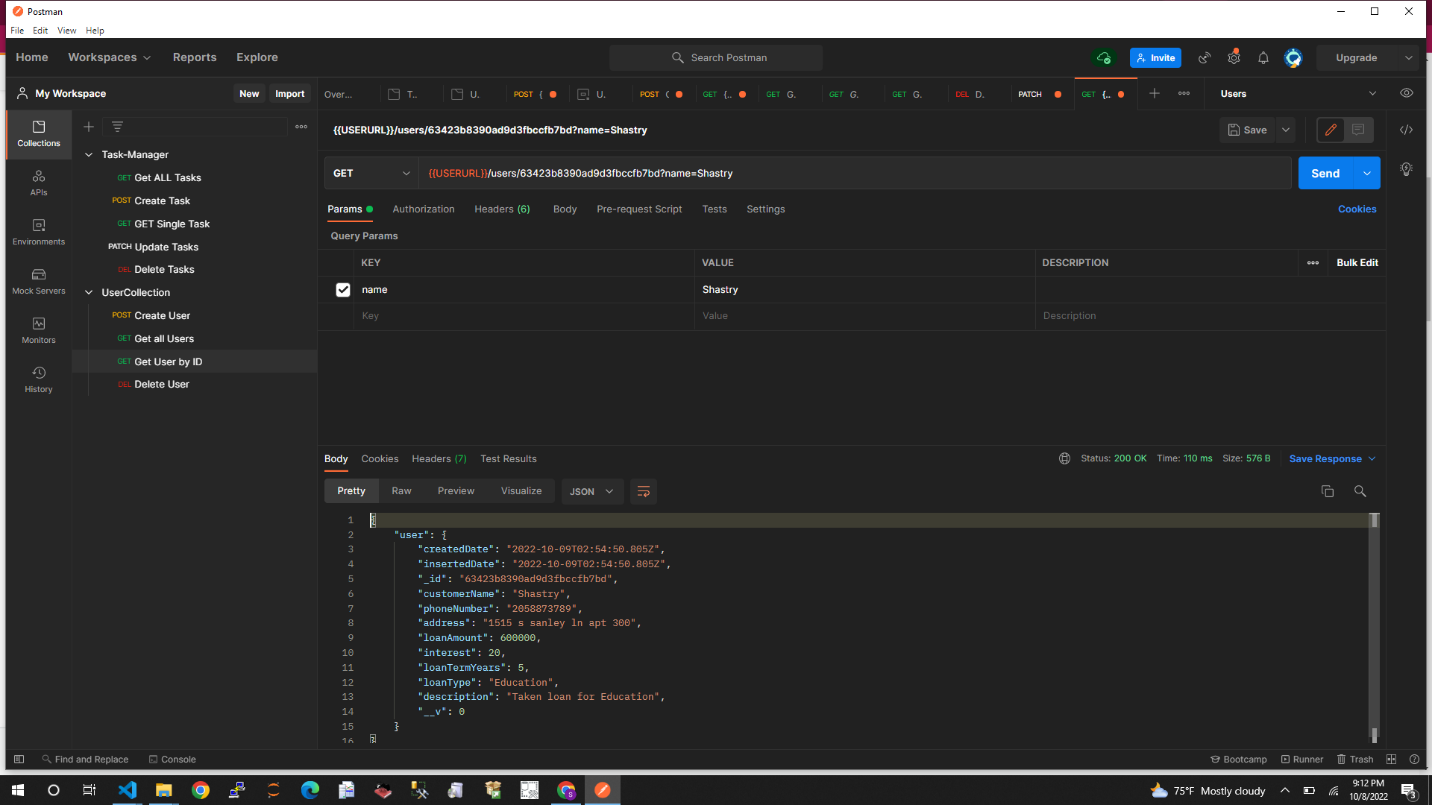
**Information Technology Management**

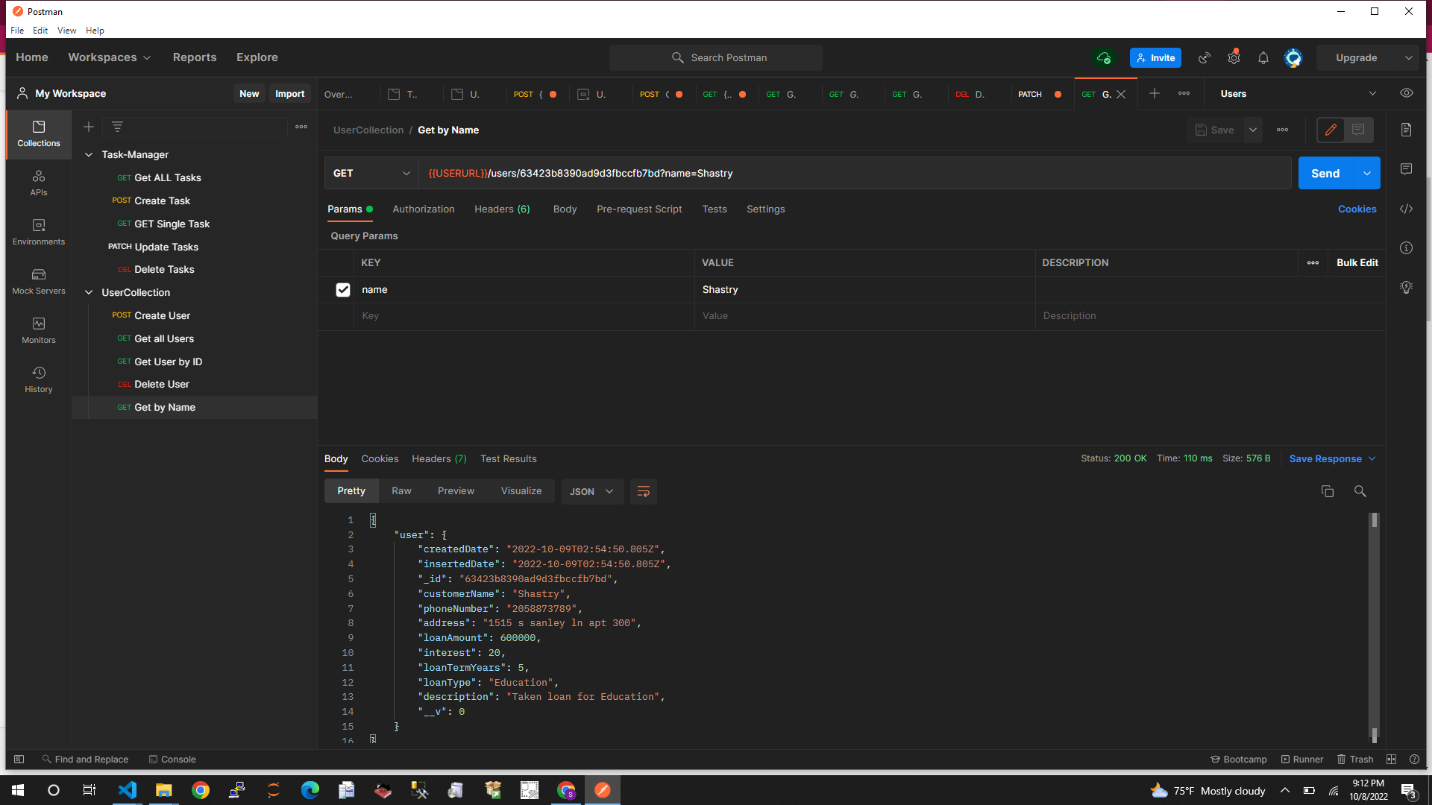
**IFT-598 Middleware Prog & Database Sec**

**Professor: Dinesh Sthapit**

**Due Date: 10/08/2022**







* Write the importance of schema definition and the need for schema definition for a sanitized documents in respect to MongoDB NoSQL database.  Also mentions the datatypes that are available in MongoDB.

A "Schema" might be challenging to comprehend at first, but it's rather simple. In object-oriented programming, a "Class" is like a database schema. A Schema is a blueprint for building objects (MongoDB refers to them as "documents") in a database, whereas a class is like a plan for building objects in a program. We inform the database about the structure of the documents we intend to enter (they will have a name property of "String" type, an admin property of "Boolean" type, etc.), and the database may then perform some validation on our behalf.

NoSQL databases often have a lot of flexibility because they don't impose any structure and let us store anything in them. (Relational databases, on the other hand, frequently enforce that all entries in the database have a uniform appearance. Therefore, good luck changing anything in your data since you'll have to alter everything that's already there.

However, Mongoose builds on the conventional "Wild West" approach of MongoDB by adding a layer of structure. Without having to develop a ton of boilerplate code us, this enables us to undertake further validation to make sure that our customers aren't uploading false data to our database.

In the userSchema shown above, you may also see examples of collections or more sophisticated datatypes. For instance, emailAddress will be an array of strings. A user is able to add numerous email addresses to their account information in this way.

But what if the complexity of the second level of data is increased? To further appreciate one of the crucial design choices you'll have to make as a MongoDB database administrator, let's look at the example below.

* [String](https://mongoosejs.com/docs/schematypes.html#strings)
* [Number](https://mongoosejs.com/docs/schematypes.html#numbers)
* [Date](https://mongoosejs.com/docs/schematypes.html#dates)
* [Buffer](https://mongoosejs.com/docs/schematypes.html#buffers)
* [Boolean](https://mongoosejs.com/docs/schematypes.html#booleans)
* [Mixed](https://mongoosejs.com/docs/schematypes.html#mixed)
* [ObjectId](https://mongoosejs.com/docs/schematypes.html#objectids)
* [Array](https://mongoosejs.com/docs/schematypes.html#arrays)
* [Decimal128](https://mongoosejs.com/docs/api.html#mongoose_Mongoose-Decimal128)
* [Map](https://mongoosejs.com/docs/schematypes.html#maps)

**Reference:**

**https://mongoosejs.com/docs/guide.html**