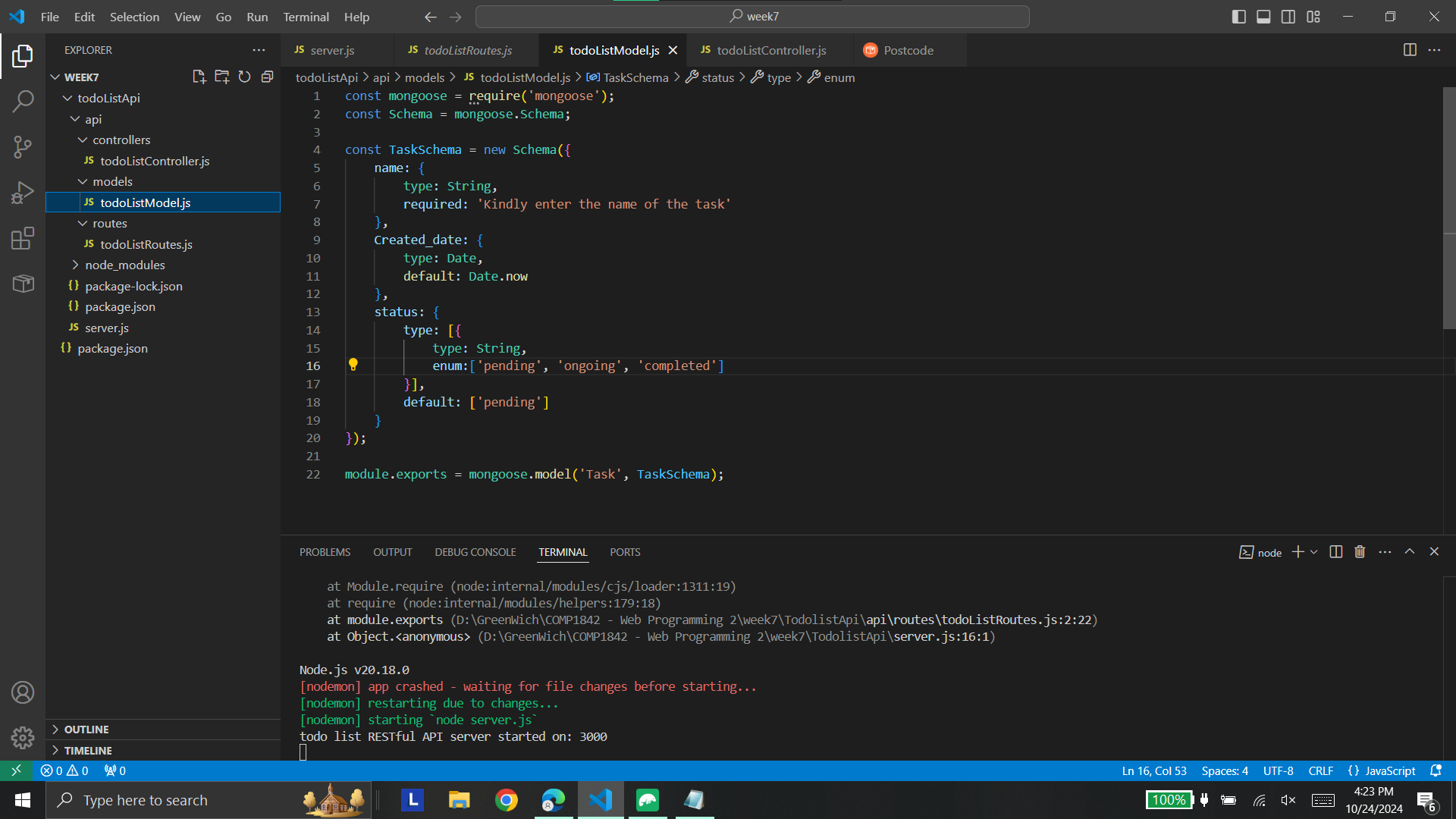
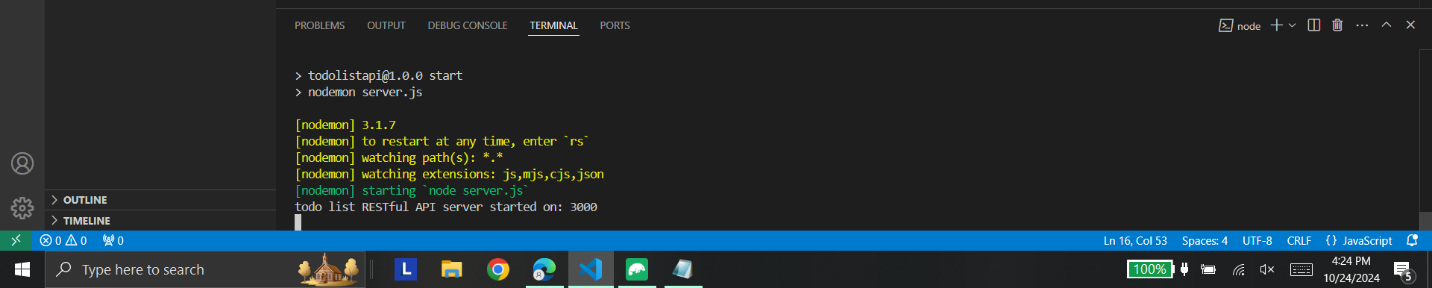
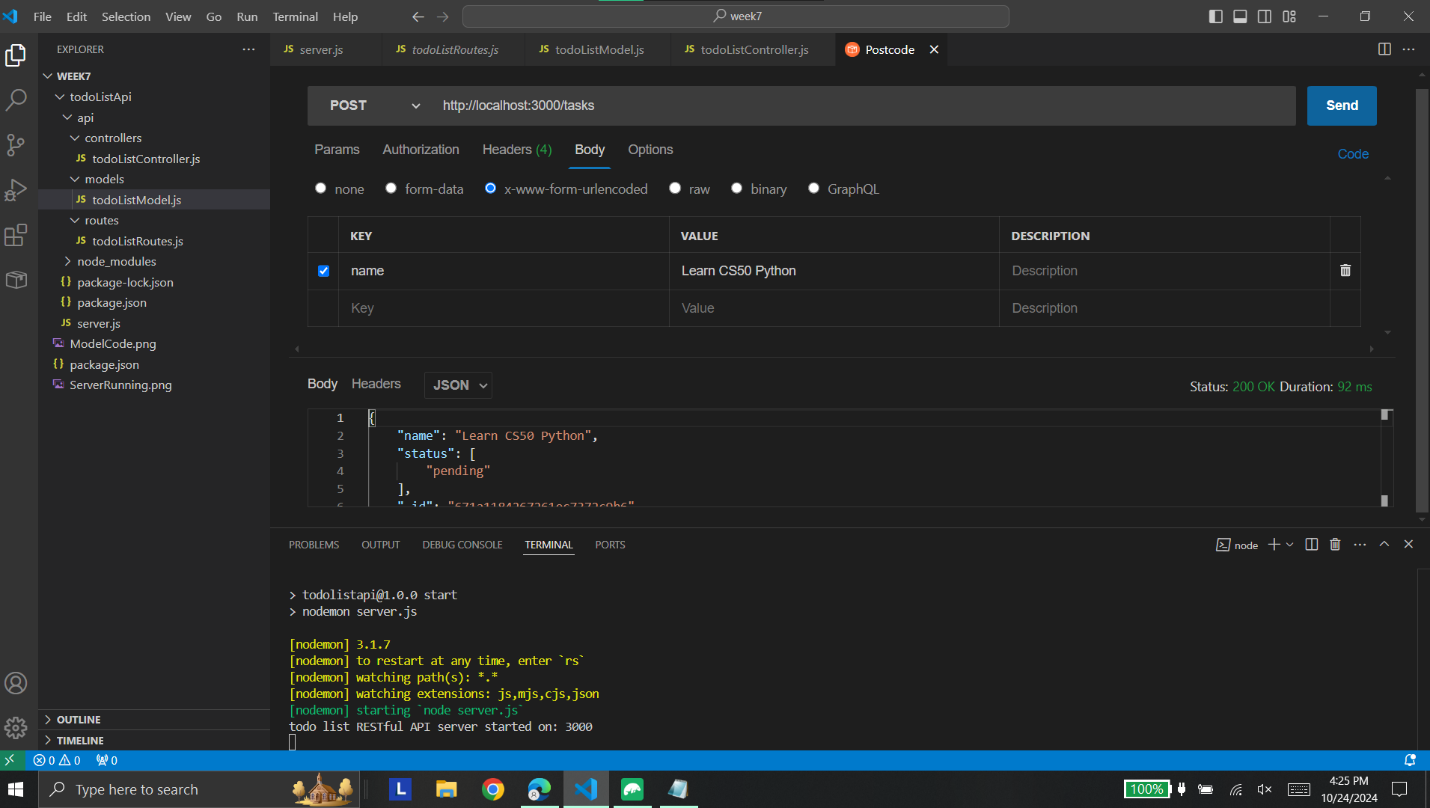
## Model Code



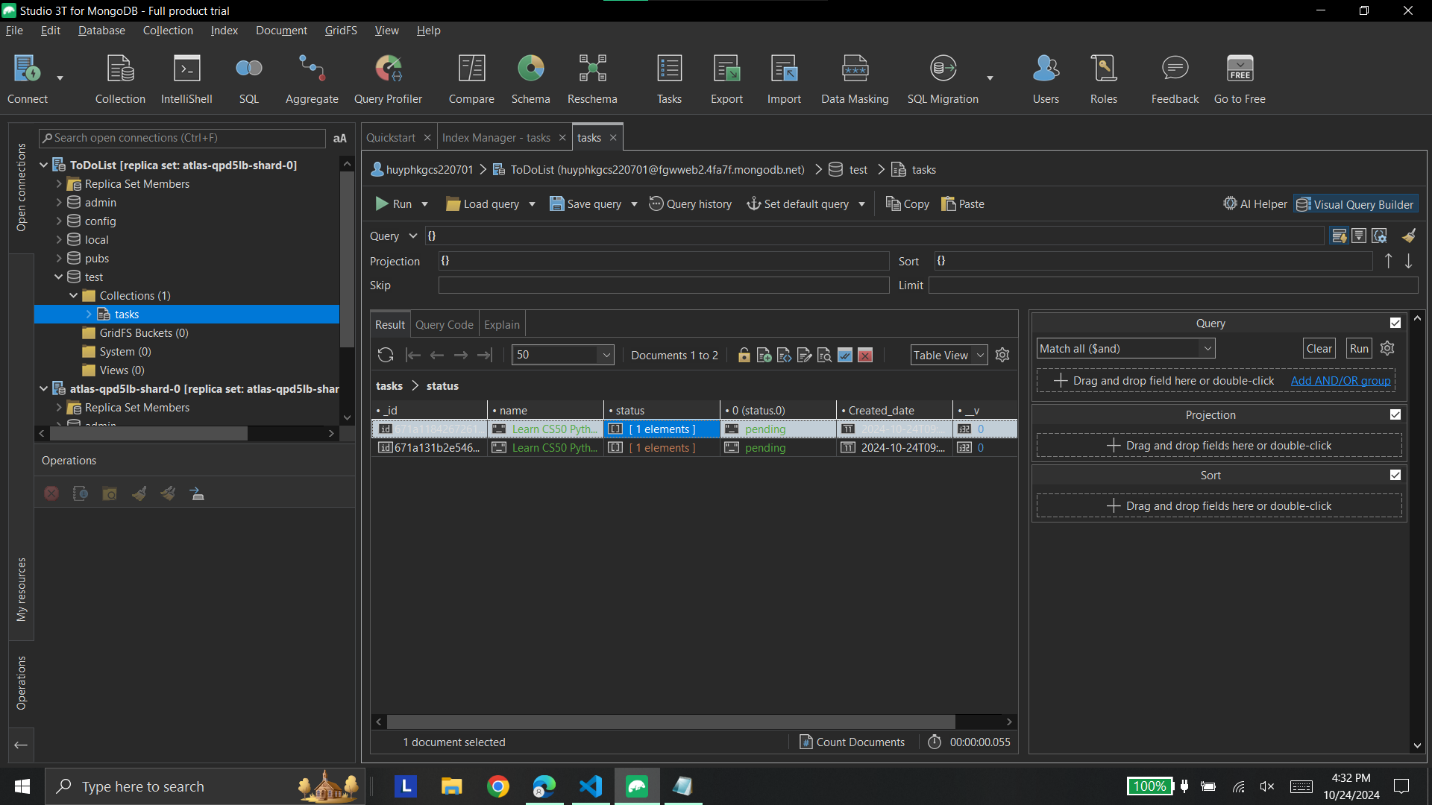
## Server running on with Nodemon on Port 300



## API connected in Postcode showing some data from database



## Data in MongoDB



In the development of the API, the todoListModel.js file is where the data model for the application is defined using Mongoose, a library that acts as a bridge between the application and MongoDB. The Schema class from Mongoose is used to structure the data. In this case, the TaskSchema contains three key attributes: name, date, and status, each having specific data types and basic validation. Once the schema is set, it is exported as a model to be used throughout the project.

The routes for the API are set up in todoListRoutes.js, where different endpoints are defined for tasks, such as /tasks and /tasks/:taskId. Each endpoint is linked to HTTP methods (e.g., GET, POST, PUT, DELETE), allowing the application to handle requests accordingly. These routes call respective controller functions that handle the logic of interacting with the database.

The controller functions, defined in todoListController.js, perform operations such as creating, reading, updating, and deleting tasks using various Mongoose methods like find, findById, and save. Each function sends responses in JSON format, making the API suitable for web and mobile applications.

Finally, the server.js file ties everything together, connecting the database, routes, controllers, and models. The Body Parser middleware is used to handle incoming data in POST and PUT requests, transforming it into a usable format for the application.

Testing the API with Postman involves sending requests to the defined endpoints and checking the database for correct functionality.