Project – OTT Watchlist Project (Backend)

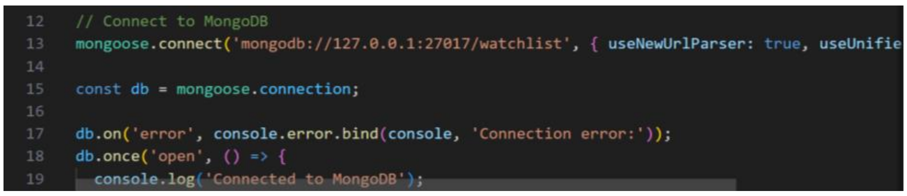
Name – Kanishk Kant

Tools Used

- Javascript
- Mongoose
- MongoDB
- Body parser (library)
- Express
- NodeJS

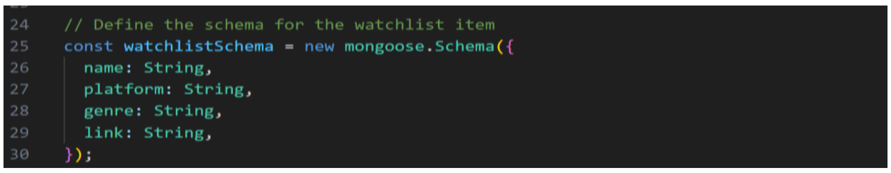
Code explanation

This line of code starts the connection to MongoDB database



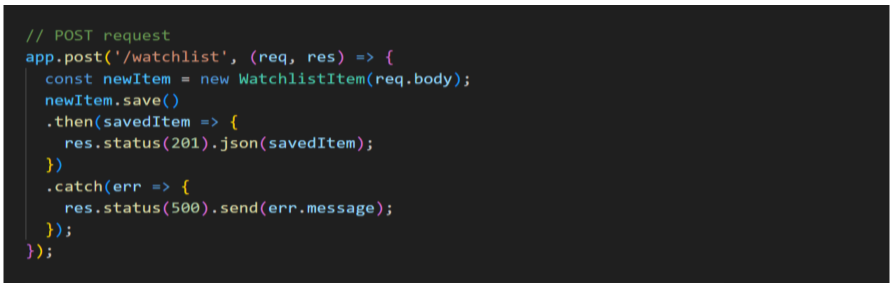
```javascript
// Connect to MongoDB
mongoose.connect('mongodb://127.0.0.1:27017/watchlist', { useNewUrlParser: true, useUnifie

const db = mongoose.connection;

db.on('error', console.error.bind(console, 'Connection error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
```

This code defines schema inside the database called as watchlist



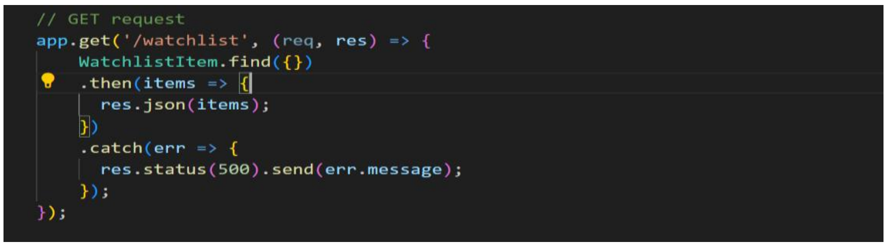
```javascript
// Define the schema for the watchlist item
const watchlistSchema = new mongoose.Schema({
  name: String,
  platform: String,
  genre: String,
  link: String,
});
```

This is the POST request



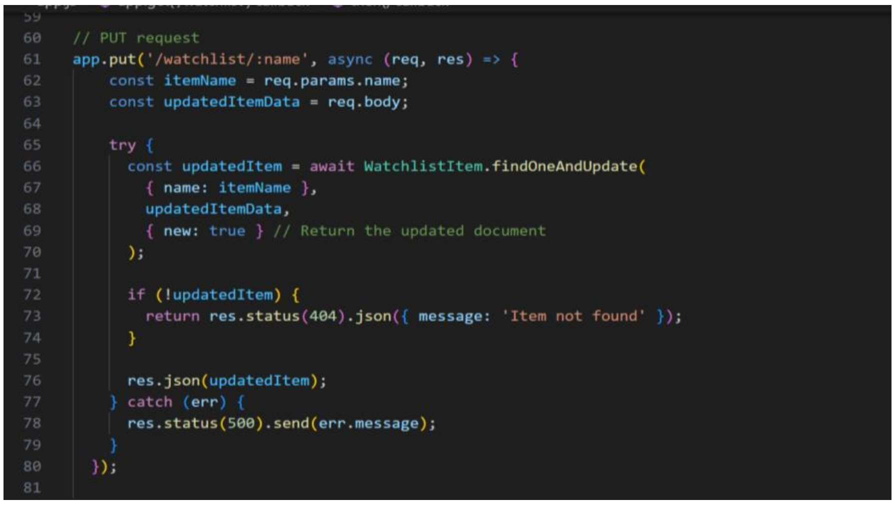
```javascript
// POST request
app.post('/watchlist', (req, res) => {
  const newItem = new WatchlistItem(req.body);
  newItem.save()
  .then(savedItem => {
    res.status(201).json(savedItem);
  })
  .catch(err => {
    res.status(500).send(err.message);
  });
});
```

This is the GET request



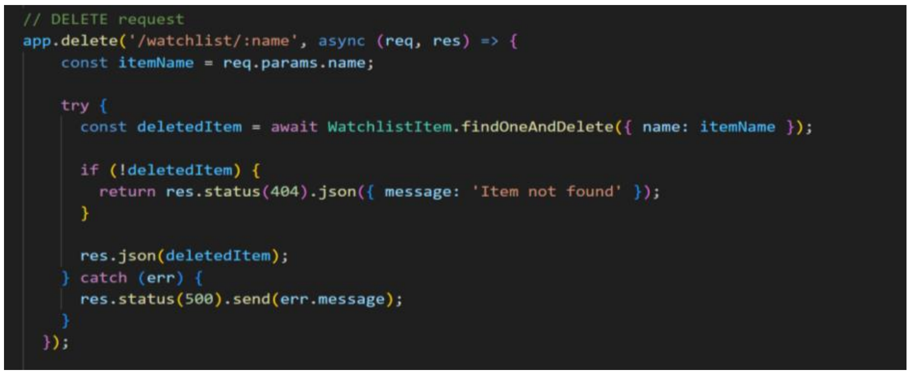
```
// GET request
app.get('/watchlist', (req, res) => {
    WatchlistItem.find({})
    .then(items => {
      res.json(items);
    })
    .catch(err => {
      res.status(500).send(err.message);
    });
});
```

This is the PUT request



```
// PUT request
app.put('/watchlist/:name', async (req, res) => {
    const itemName = req.params.name;
    const updatedItemData = req.body;

    try {
      const updatedItem = await WatchlistItem.findOneAndUpdate(
        { name: itemName },
        updatedItemData,
        { new: true } // Return the updated document
      );

      if (!updatedItem) {
        return res.status(404).json({ message: 'Item not found' });
      }

      res.json(updatedItem);
    } catch (err) {
      res.status(500).send(err.message);
    }
  });
```

This is the DELETE request



```
// DELETE request
app.delete('/watchlist/:name', async (req, res) => {
    const itemName = req.params.name;

    try {
      const deletedItem = await WatchlistItem.findOneAndDelete({ name: itemName });

      if (!deletedItem) {
        return res.status(404).json({ message: 'Item not found' });
      }

      res.json(deletedItem);
    } catch (err) {
      res.status(500).send(err.message);
    }
  });
```

<u>How to runserver ?</u>

First make sure the URL on which MongoDB is running is correct. Run the mongoDB server and then run the server first navigating to the correct directory (where app.js is located). The server can be started by typing "node app.js" in the terminal.

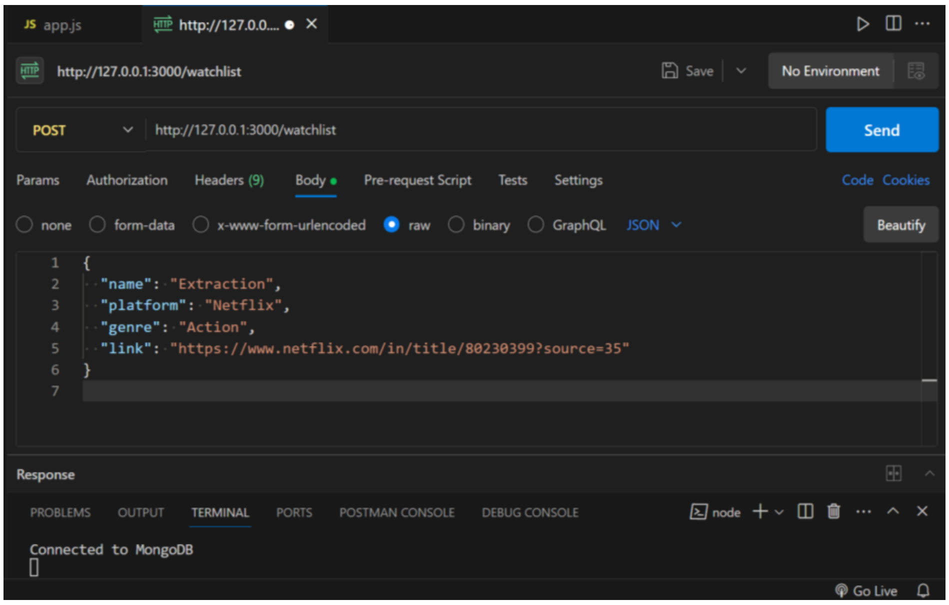
<u>Postman Result Example</u>

Here is an example of a request made to the server

POST request

Endpoint - http://127.0.0.1:3000/watchlist/

Before sending request



http://127.0.0.1:3000/watchlist

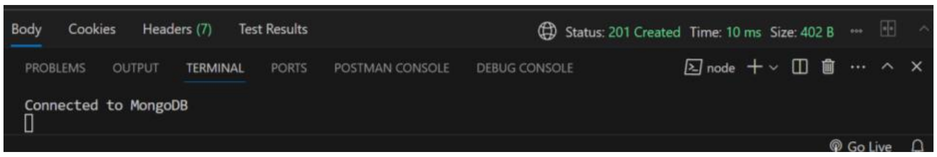
POST http://127.0.0.1:3000/watchlist

Body: none / form-data / x-www-form-urlencoded / [✓ raw] / binary / GraphQL — JSON

```
{
  "name": "Extraction",
  "platform": "Netflix",
  "genre": "Action",
  "link": "https://www.netflix.com/in/title/80230399?source=35"
}
```

```
Connected to MongoDB
```

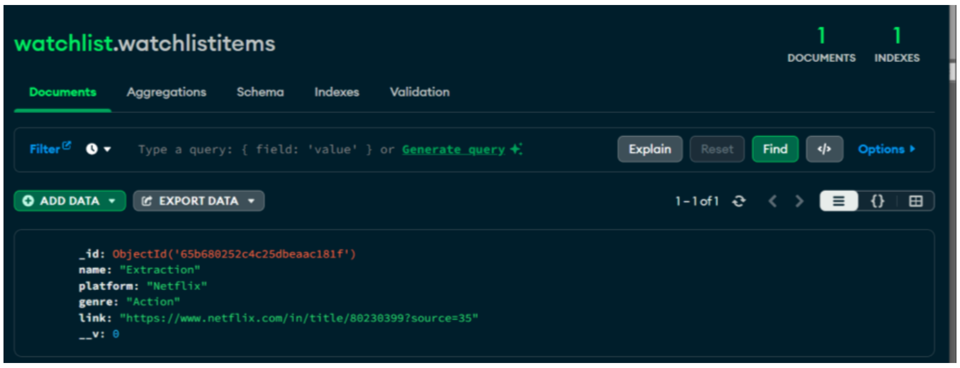
After sending request



Status: 201 Created Time: 10 ms Size: 402 B

```
Connected to MongoDB
```

DB results



watchlist.watchlistitems

1 DOCUMENTS 1 INDEXES

1-1 of 1

```
_id: ObjectId('65b680252c4c25dbeaac181f')
name: "Extraction"
platform: "Netflix"
genre: "Action"
link: "https://www.netflix.com/in/title/80230399?source=35"
__v: 0
```

Such results can be verified for all sorts of requests made to the server by using Postman.