Name: Devanshu Surana
PRN: 1032210755
Roll No.: 23
Panel: C**, Batch:** C1

FSD Lab 06
Aim: Develop a set of REST API using Express and Node.

Objectives:

1. To define HTTP GET and POST operations.
2. To understand and make use of ‘REST’, ‘a REST endpoint’, ‘API Integration’, and ‘API Invocation’
3. To understand the use of a REST Client to make POST and GET requests to an API.

Theory:

1. REST API (Representational State Transfer API):
   REST, which stands for Representational State Transfer, is an architectural style for designing networked applications. A REST API is a set of rules and conventions for building and interacting with web services that adhere to the principles of REST. It uses a stateless communication model, meaning each request from a client contains all the information needed to understand and process the request.
2. Main Purpose of REST API:
   The main purpose of a REST API is to enable communication between systems on the web. It provides a standardized way for different applications to interact with each other over the internet. REST APIs use standard HTTP methods (GET, POST, PUT,

DELETE) to perform operations on resources, which are identified by URIs (Uniform Resource Identifiers). This simplicity and adherence to

standards make REST APIs widely adopted for building scalable and interoperable web services.

FAQ:

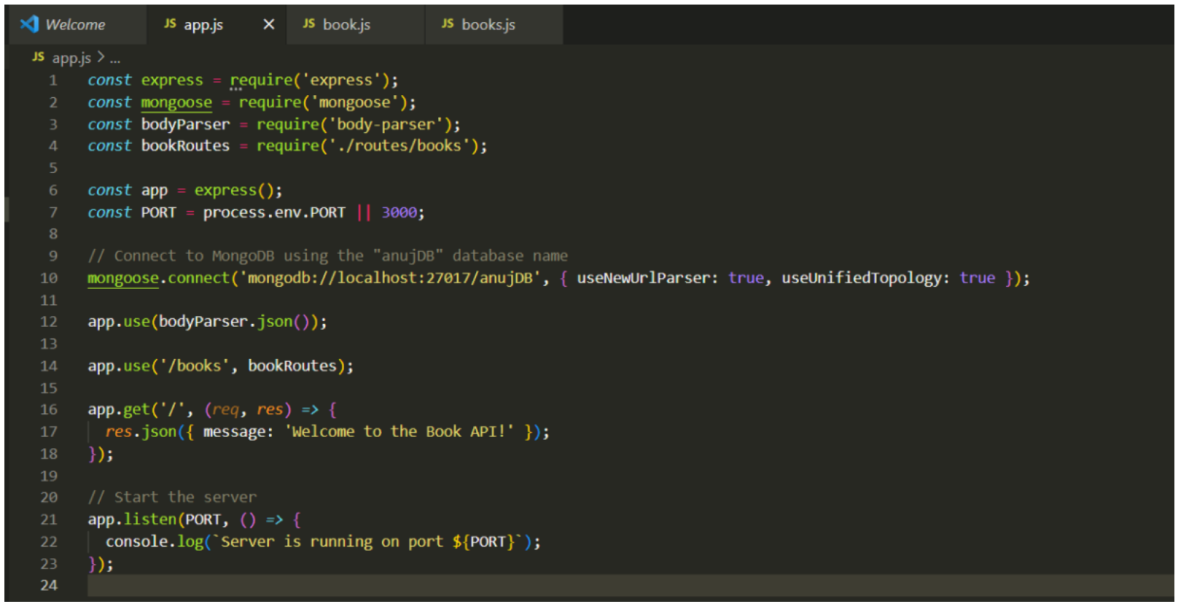
1. What are HTTP Request types? ANS: HTTP Request Types:

HTTP (Hypertext Transfer Protocol) defines several request methods, also known as HTTP verbs, to indicate the desired action to be performed on a resource. The main HTTP request types are:

GET: Retrieves data from a specified resource.
POST: Submits data to be processed to a specified resource.
PUT: Updates a resource or creates a new resource if it doesn't exist.
DELETE: Deletes the specified resource.
PATCH: Partially updates a resource.
HEAD: Same as GET but retrieves only the headers and not the actual data.
OPTIONS: Describes the communication options for the target resource.
TRACE: Performs a message loop-back test along the path to the target resource.

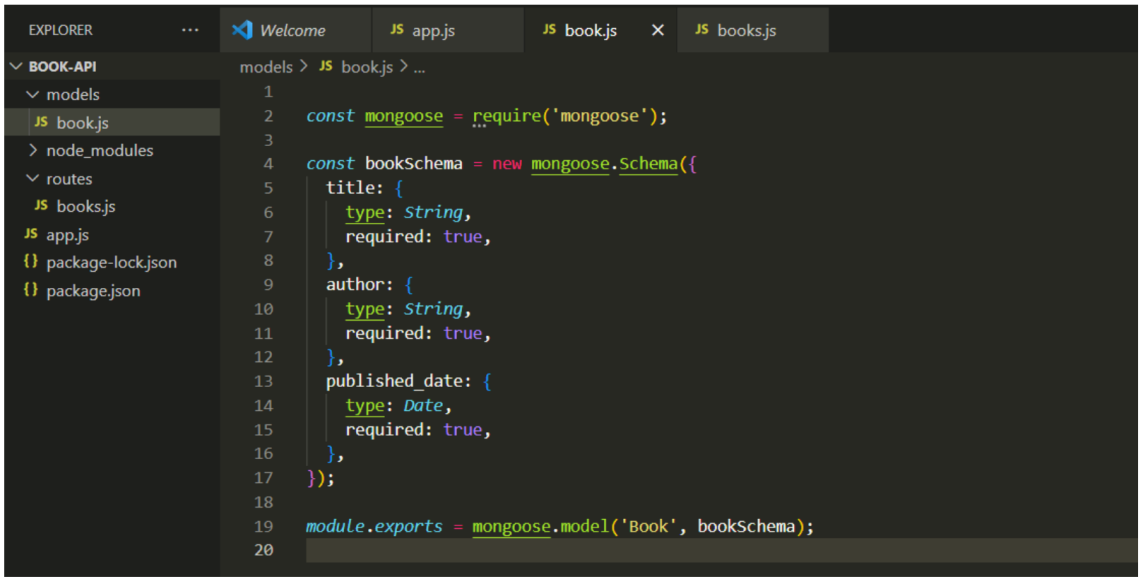
OUTPUT:

app.js



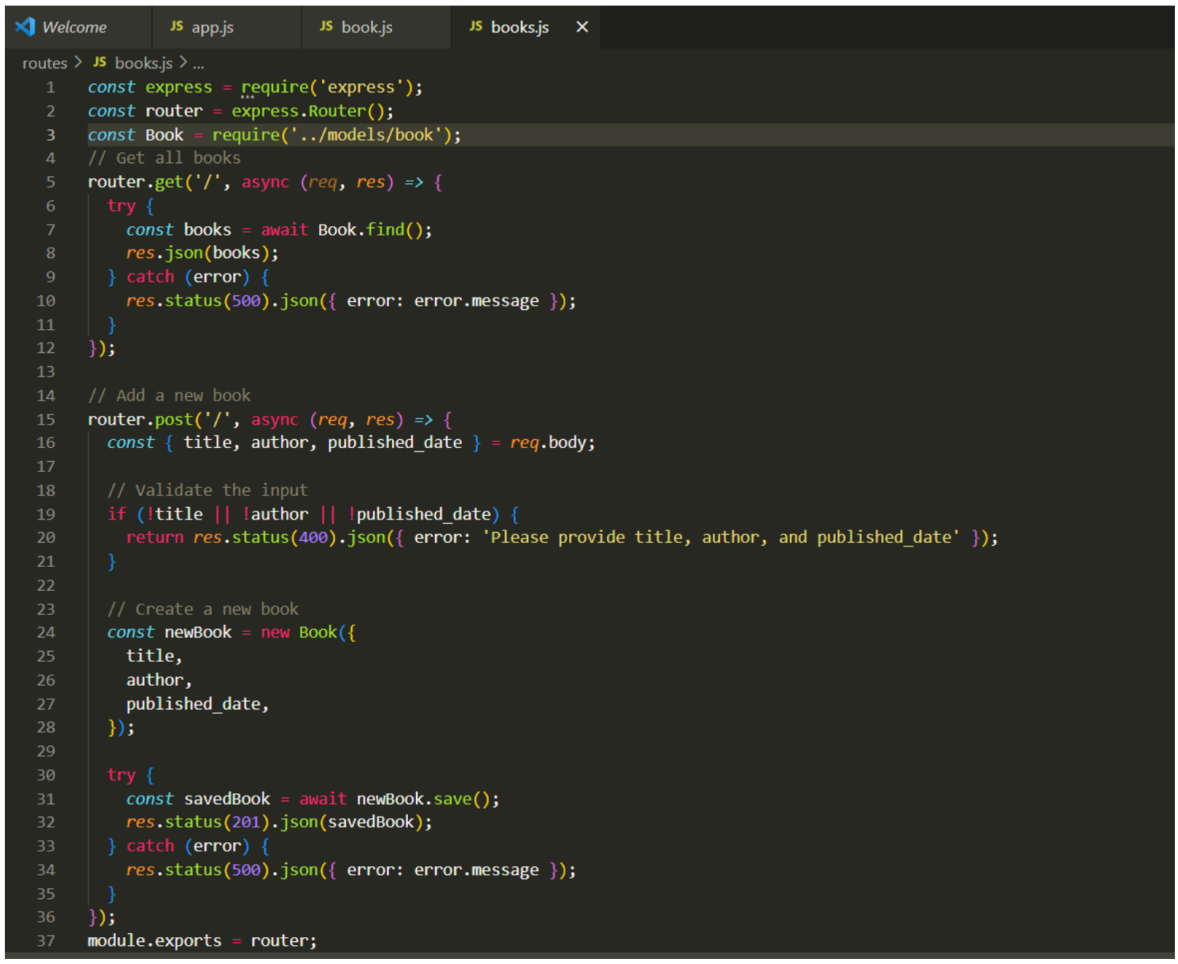
```
Welcome | app.js | book.js | books.js
app.js > ...
1  const express = require('express');
2  const mongoose = require('mongoose');
3  const bodyParser = require('body-parser');
4  const bookRoutes = require('./routes/books');
5
6  const app = express();
7  const PORT = process.env.PORT || 3000;
8
9  // Connect to MongoDB using the "anujDB" database name
10 mongoose.connect('mongodb://localhost:27017/anujDB', { useNewUrlParser: true, useUnifiedTopology: true });
11
12 app.use(bodyParser.json());
13
14 app.use('/books', bookRoutes);
15
16 app.get('/', (req, res) => {
17   res.json({ message: 'Welcome to the Book API!' });
18 });
19
20 // Start the server
21 app.listen(PORT, () => {
22   console.log(`Server is running on port ${PORT}`);
23 });
24
```

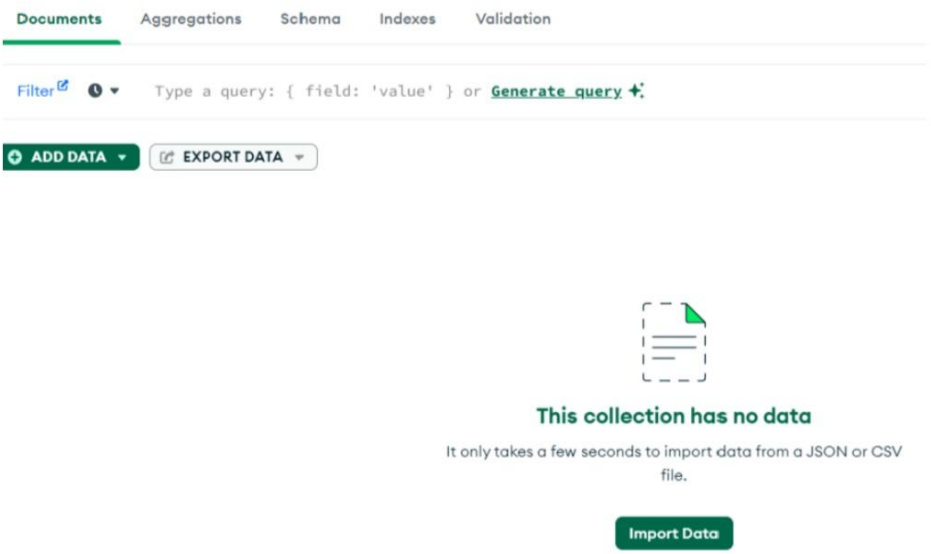
models/book.js

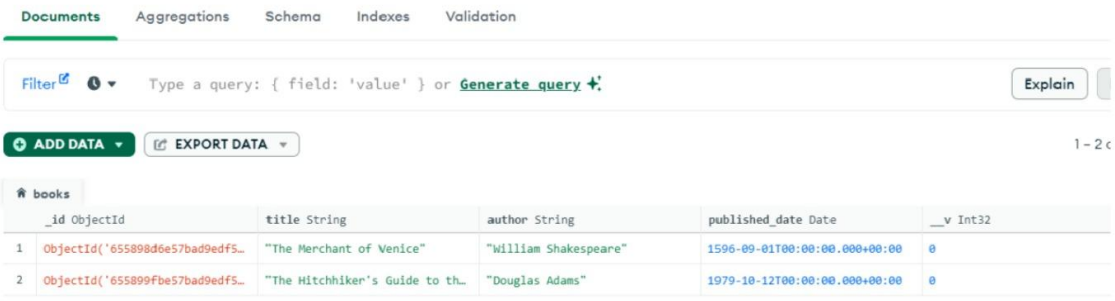


```
EXPLORER
BOOK-API
  models
    book.js
  node_modules
  routes
    books.js
  app.js
  package-lock.json
  package.json
Welcome | app.js | book.js | books.js
models > book.js > ...
1
2  const mongoose = require('mongoose');
3
4  const bookSchema = new mongoose.Schema({
5    title: {
6      type: String,
7      required: true,
8    },
9    author: {
10     type: String,
11     required: true,
12   },
13   published_date: {
14     type: Date,
15     required: true,
16   },
17 });
18
19 module.exports = mongoose.model('Book', bookSchema);
20
```

routes/books.js







	_id ObjectId	title String	author String	published_date Date	__v Int32
1	ObjectId('655898d6e57bad9edf5…	"The Merchant of Venice"	"William Shakespeare"	1596-09-01T00:00:00.000+00:00	0
2	ObjectId('655899fbe57bad9edf5…	"The Hitchhiker's Guide to th…	"Douglas Adams"	1979-10-12T00:00:00.000+00:00	0