Question - 1

REST API Response

SCORE: **5 points**

REST API | Medium

Consider the following RESTful API endpoint and response.

```
GET /api/books?sort=desc&limit=10&offset=20 HTTP/1.1
Accept: application/json
```


```
{
  "count": 100,
  "next": "/api/books?sort=desc&limit=10&offset=30",
  "previous": "/api/books?sort=desc&limit=10&offset=10",
  "results": [
    {
      "id": 1,
      "title": "To Kill a Mockingbird",
      "author": "Harper Lee",
      "publication_date": "1960-07-11",
      "genre": "Novel"
    },
    {
      "id": 2,
      "title": "1984",
      "author": "George Orwell",
      "publication_date": "1949-06-08",
      "genre": "Dystopian Fiction"
    },
    {
      "id": 3,
      "title": "Brave New World",
      "author": "Aldous Huxley",
      "publication_date": "1932-06-11",
      "genre": "Dystopian Fiction"
    }
  ]
}
```


Which of the following statements is true about the endpoint and response?

- The endpoint is using a custom HTTP method. The response is paginated and sorted by the publication_date field in ascending order.
- The endpoint is using a standard HTTP method. The response is paginated and sorted by the title field in descending order.
- **The endpoint is using a standard HTTP method. The response is paginated and sorted by the publication_date field in descending order.**
- The endpoint is using a custom HTTP method. The response is paginated and sorted by the title field in ascending order.

Question - 2

SCORE: **5 points**

APIs: True or False?

API Easy

An API is a binary interface.

- True
- False

Question - 3

SCORE: **5 points**

Query Result

REST API Filtering Pagination Easy

Consider the following GET API endpoint.

```
/resources?filter=name:John,age:>25,address: NY&sort=name&page=2&size=10
```

What does the API call do?

It filters the resources by the specified fields and values, sorts the filtered resources by name in ascending order, and retrieves the second page of filtered resources, with a maximum of 10 resources per page.

It filters the resources by the specified fields and values, sorts the filtered resources by name in descending order, and retrieves the second page of filtered resources, with a minimum of 10 resources per page.

It filters the resources by the specified fields, sorts the filtered resources by the specified fields in ascending order, and retrieves all resources starting from the second page, with a maximum of 10 resources per page.

It filters the resources by the specified fields, sorts the filtered resources by the specified fields in descending order, and retrieves all resources ending at the second page, with a minimum of 10 resources per page.

Question - 4

SCORE: **5 points**

Error handling - I

REST API Easy

A client requested a resource through a RESTful API and received a "NOT FOUND" error message in response. Which statement best describes the error handling of this scenario?

- If a resource is not found, a 404 error is reported due to an error on the server side because the requested resource was not found.

If a resource is not found, a 404 error is reported due to an error on the client side because the requested resource does not exist but is still being accessed.

- If a resource is not found, a 504 error is reported due to an error on the client side.

If a resource is not found, a 504 error is reported due to an error on the server side.

Question - 5

SCORE: **5 points**

HTTP Methods

HTTP | HTTP methods | Easy

Consider a RESTful API that represents a job application service and the following code snippets.

```
# Code Snippet 1
POST /api/job_applications HTTP/1.1 Content-Type: application/json { "job_title": "Software Engineer",
"company_name": "HackerRank Inc.", "date_applied": "2023-02-18", "status": "Pending" }
```

```
# Code Snippet 2
GET /api/job_applications HTTP/1.1
```

```
# Code Snippet 3
GET /api/job_applications/123 HTTP/1.1
```

```
# Code Snippet 4
PUT /api/job_applications/123 HTTP/1.1 Content-Type: application/json { "status": "Interview Scheduled" }
```

Which of the following statements are true about HTTP methods and the code snippets?

- The first code snippet gets a list of job applications using the GET method, and the second code snippet creates a new job application using the POST method.
- The first code snippet creates a new job application using the POST method, and the second code snippet gets a list of job applications using the GET method.
- The third code snippet updates a specific job application using the PATCH method, and the fourth code snippet deletes a job application using the GET method.
- The third code snippet gets a specific job application using the GET method, and the fourth code snippet updates a specific job application using the PUT method.

Question - 6

SCORE: **5 points**

APIs in Express JS

Javascript | Medium | ExpressJS | Node.js

What is the console output when the GET request is sent to *http://localhost:5000/* using a web browser?

```
const express = require("express");
const app = express();

app.use((req, res, next) => {
  setTimeout(() => {
    console.log(`${req.method} request received.`);
  }, 3000);
});

app.get("/", (req, res) => {
  setTimeout(() => {
    console.log("You sent a GET request");
  }, 1000);
```

```
});

app.post("/", (req, res) => {
  res.send("You sent a POST request.");
});

app.listen(5000, () => {
  console.log("Server started on port 5000.");
});
```

- "Server started on port 5000."
- "GET request received."
- "You sent a GET request." and "GET request received."
- "GET request received." and "You sent a GET request."

Question - 7

Middleware

SCORE: **5 points**

Middleware | REST API | Medium

Consider a RESTful API that represents a social media platform, and the following code snippets.

```
const requestLogger = (req, res, next) => {
  console.log(`${req.method} ${req.path} - ${req.ip}`);
  next();
};

const authenticateUser = (req, res, next) => {
  const authToken = req.headers.authorization?.split(' ')[1];
  if (authToken) {
    // Authenticate the user using the auth token
    req.user = { id: 1234, username: 'johndoe' };
    next();
  } else {
    res.status(401).send('Unauthorized');
  }
};

const validatePostData = (req, res, next) => {
  const { text, mediaUrl } = req.body;
  if (text && mediaUrl) {
    next();
  } else {
    res.status(400).send('Invalid request data');
  }
};

const rateLimiter = (req, res, next) => {
  // Rate limit the request based on the client's IP address
  const ipAddress = req.ip;
  // ...
  next();
};
```

```
POST /api/posts HTTP/1.1
Content-Type: application/json
```

```
{
  "text": "Hello, world!",
  "mediaUrl": "https://example.com/image.jpg"
}
```

Which of the following code snippets illustrates the use of middleware in this API?

- const app = express();
  app.use(requestLogger);
  app.use(authenticateUser);
  app.use(validatePostData);
  app.use(rateLimiter);

- app.post('/api/posts', (req, res) => {
  const { text, mediaUrl } = req.body;
  // ...
  });

- app.post('/api/posts', validatePostData, (req, res) => {
  const { text, mediaUrl } = req.body;
  // ...
  });

- app.post('/api/posts', rateLimiter, (req, res) => {
  const { text, mediaUrl } = req.body;
  // ...
  });

Question - 8

SCORE: **5 points**

SetTimeout

Functions | Javascript | events | Easy | Loops | Array | Javascript Timers

What is the output of this JavaScript code snippet?

```
const numbers = [1, 2, 3, 4, 5];

for (var i = 0; i < numbers.length; i++) {
  setTimeout(() => {
    console.log(`Number: ${numbers[i]}`);
  }, i * 1000);
}
```

- Number: 1
  Number: 2
  Number: 3
  Number: 4
  Number: 5

Number: undefined
Number: undefined
Number: undefined
Number: undefined
Number: undefined

Number: 5
Number: 5
Number: 5
Number: 5
Number: 5

No output will be displayed.

Question - 9

NodeJS: Event Ordering

SCORE: **5 points**

Node.js | Easy | EventEmitter

What is the console output from this code snippet?

```
const EventEmitter = require("events");
const event = new EventEmitter();

event.on("second", () => {
  console.log("Second event");
});

event.on("first", () => {
  console.log("Welcome!");
});

event.emit("first");

event.on("first", () => {
  event.emit("second");
  console.log("First Event");
});

event.emit("first");
```

```
Second event
First Event
Welcome!
Second event
First Event
Welcome!
```

- ```
  Welcome!
  Second event
  First Event
  Welcome!
  Second event
  First Event
  ```

- ```
  Welcome!
  Welcome!
  Second event
  First Event
  ```

Question - 10

Async Methods

SCORE: **5 points**

async await | Map | Javascript | Easy | Async Programming | ES6 Features | Javascript Timers

What is the output of this JavaScript code snippet?

```
const asyncOperations = [
    async () => {
        await new Promise(resolve => setTimeout(resolve, 200));
        return 1;
    },
    async () => {
        await new Promise(resolve => setTimeout(resolve, 1000));
        return Promise.reject(new Error("An error occurred"));
    },
    async () => {
        await new Promise(resolve => setTimeout(resolve, 100));
        return 2;
    },
    async () => {
        await new Promise(resolve => setTimeout(resolve, 500));
        return 3;
    },
];

(async () => {
    const resultMap = new Map();

    const processOperation = async (index, operation) => {
        try {
            const result = await operation();
            resultMap.set(index, { status: "fulfilled", value: result });
        } catch (error) {
            resultMap.set(index, { status: "rejected", reason: error.message });
```

```
        }
    };

    await Promise.all(asyncOperations.map((operation, index) => processOperation(index, operation)));

    console.log(Array.from(resultMap.entries()).sort((a, b) => a[1].value - b[1].value).map(entry => entry[1]));
})();
```

- [ ] [
{ status: "fulfilled", value: 2 },
{ status: "fulfilled", value: 1 },
{ status: "fulfilled", value: 3 },
{ status: "rejected", reason: "An error occurred" },
]

- [ ] [
{ status: "fulfilled", value: 1 },
{ status: "fulfilled", value: 2 },
{ status: "fulfilled", value: 3 },
{ status: "rejected", reason: "An error occurred" },
]

- [ ] [
{ status: "rejected", reason: "An error occurred" },
{ status: "fulfilled", value: 1 },
{ status: "fulfilled", value: 2 },
{ status: "fulfilled", value: 3 },
]

- [ ] [
{ status: "fulfilled", value: 1 },
{ status: "rejected", reason: "An error occurred" },
{ status: "fulfilled", value: 2 },
{ status: "fulfilled", value: 3 },
]

Question - 11

SCORE: **5 points**

REST Basics

Easy

REST is an architecture to provide access to data over a network through an API. Which of the following are true?

- [ ] REST is strictly a client-server interaction type meaning that the client performs requests and the server sends responses to these requests.

- [ ] REST is a server-server interaction meaning that both sides can make requests and send responses to requests.

- [ ] In REST architecture, a properly designed access endpoint should not specify actions as a part of the resource URI. Instead, actions should be specified using appropriate protocol methods such as GET, POST, PUT, and DELETE over HTTP.

- [ ] REST responses are not capable of specifying any caching related information regarding the accessed resource. Caching must be resolved with other mechanisms.

Question - 12

SCORE: **5 points**

Write Array To File

Node.js Easy

```
const arr = ["apple", "mango", "banana"];
```

Given an array, which of the following commands opens the file "demo.txt" (or creates it if the file does not exist) and writes the contents of the array to the file?

- fs.writeFileSync("demo.txt", arr);
- fs.writeFileSync("demo.txt", JSON.stringify(arr));
- fs.writeFile("demo.txt", arr, (err) => {
  console.log(err);
  });
- fs.write("demo.txt", JSON.stringify(arr));