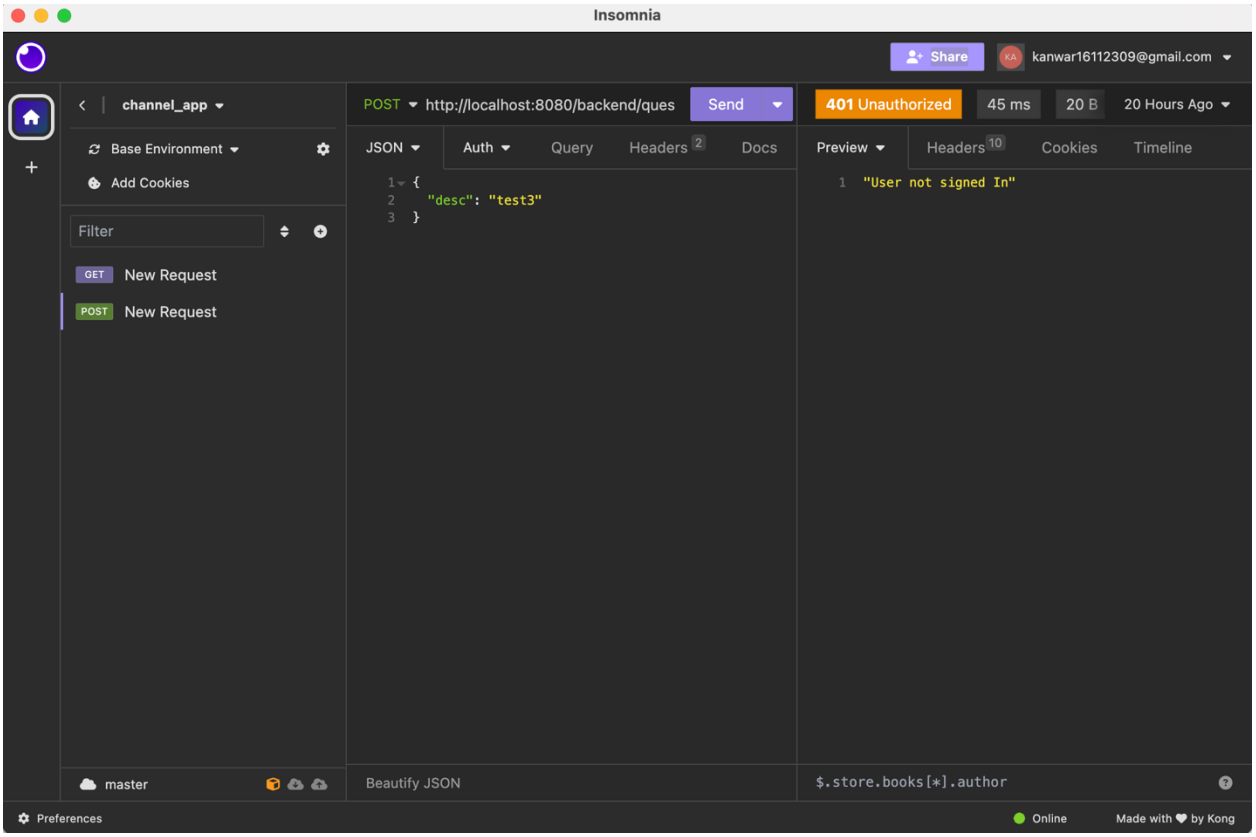
A combination of methods was used in testing the node.js code, different variations of loadtest command were used to achieve this. Initial tests were performed by sending POST requests using tools like Insomnia to ensure functionality and check backend data transfers.

Also manual testing was used “console.log()”when method are called and also printing out ok on browser when database is initialized.

Examples of Insomnia testing



Examples of endpoint testing I did

addQuestion Endpoint Testing:

1. **Single Request Test**
    - Completed Requests: 4879
    - Mean Latency: 5.6 ms
    - Effective rps (Requests Per Second): 488
    - Observation: The server handles single requests very efficiently with low latency.
2. **Large Data Test**
    - Completed Requests: 10

- Mean Latency: 25 ms
- Effective rps: 128
- Observation: Larger data size significantly increases latency suggesting a performance impact with larger payloads.

3. **Increased Number of Requests**
    - Completed Requests: 100
    - Mean Latency: 11.8 ms
    - Effective rps: 319
    - Observation: As the number of requests increases there's a moderate increase in latency but the server still maintains a good response rate.
4. **Increased Concurrency**
    - Completed Requests: 100
    - Mean Latency: 90.5 ms
    - Effective rps: 364
    - Observation: Higher concurrency leads to a significant increase in latency indicating potential bottlenecks under concurrent load.
5. **Stress Test**
    - Completed Requests: 1000
    - Total Errors: 40
    - Mean Latency: 379 ms
    - Effective rps: 493
    - Observation: Under extreme load latency dramatically increases and errors start to occur indicating limits of current server configuration.

getQuestions Endpoint Testing:

6. **Single Request Test**
    - Completed Requests: 1760
    - Mean Latency: 16.6 ms
    - Effective rps: 176
    - Observation: Single request handling is efficient with moderate latency.
7. **Increased Number of Requests**
    - Completed Requests: 100
    - Mean Latency: 26.2 ms
    - Effective rps: 145
    - Observation: Increased requests lead to a slight increase in latency but the server is still performing adequately.
8. **Increased Concurrency**
    - Completed Requests: 100
    - Mean Latency: 230.2 ms
    - Effective rps: 141
    - Observation: High concurrency significantly impacts performance with a substantial increase in latency.
9. **Stress Test**
    - Completed Requests: 1000
    - Mean Latency: 1196.4 ms

- Effective rps: 153
- Observation: Under heavy load latency becomes quite high which could affect user experience negatively