Bytexl’s guided project

Final Project report

|  |  |
| --- | --- |
| Name of the educator | Neetesh Parashar |
| Project title | UTube Subscriber Backend |
| Tools / platforms used | MERN |

About the project: <Max of 250 words>

This project manages subscriber data via a RESTful API application. To see subscriber data by ID, retrieve all subscribers, and retrieve certain attributes such subscriber names and their subscribed channels, it offers many endpoints. The application, which was created with Node.js and Express, is set up to efficiently process and react to a variety of API calls. The database is MongoDB, which is controlled by Mongoose and offers a customizable structure for subscriber data. This project might be used in situations like a membership-based organization or a content subscription business where it is necessary to track and manage a big number of subscribers.

System requirements: both software and hardware

· Node.js v14+

· MongoDB v4+

· npm (Node Package Manager)

Hardware

· Minimum 4 GB RAM

· Disk space for MongoDB (varies based on data volume)

Functional requirements:

1 Get a comprehensive list of every subscription.  
2 Use an ID to retrieve specific subscriber information.  
3 Provide the names of subscribers together with the channels to which they are subscribed.

4 Establish a test endpoint to confirm the functioning of the API.

User interface requirements if any:

No UI requirements as this is a backend-only application.

Inputs and Outputs:

· ****Inputs****: HTTP requests with parameters (e.g., subscriber ID)

· ****Outputs****: JSON-formatted data, such as subscriber details, names, and channels

List of subsystems:

1 Managing HTTP GET requests for subscriber data is the responsibility of API routing.  
2 Data Management: Uses Mongoose to manage CRUD activities.  
3 Middleware: Handles URL-encoded request data and JSON.

Other Applications relevant to your project: Mention in what different contexts this project can be used with other flavours.

This project might be modified for any application that needs organized, retrievable user data, such as customer management systems or subscription-based services.

Designing of Test cases:

Write the list of test cases and explain their functions

1 Fetch All Subscribers: Use a list of all subscribers (status 200) to confirm the answer.  
2 Fetch Subscriber by ID: Provide an error notice if the ID is incorrect and confirm respond with accurate subscriber data if the ID is legitimate.  
3 Fetch Subscriber Names and Channels: Verify that each subscriber's name and channel are the only fields included in the response.  
4 Dummy API: Verify that a test message verifies the route's functionality and activity.

Future Work:

· Add more advanced querying options (e.g., by subscription date or channel type).

· Integrate authentication for secure API access.

· Expand the system to handle more complex subscriber interactions, such as tracking subscription changes or payment history.

References: ((negative points if missing or inadequate)

· [Express.js Documentation](https://expressjs.com/" \t "_new)

· [Mongoose Documentation](https://mongoosejs.com/" \t "_new)

· [MongoDB Documentation](https://www.mongodb.com/docs/" \t "_new)

Reflection of the project creation:

* Describe the *technical challenges* you encountered in the development of your project

**Technical Challenges**: Handling errors and managing responses for various scenarios were key challenges. Additionally, structuring the data flow efficiently using Mongoose required careful planning

* Describe how your existing software engineering knowledge / techniques helped you to address those challenges

Utilizing Software Engineering Knowledge: It helped to have prior expertise with REST API architecture and Express error handling. Consistent data processing was made possible via the Mongoose schema architecture.

* What benefits did you individually experience while working on this project?

Personal Benefits: Acquired expertise in middleware development, asynchronous data flow management, and CRUD operation setup using MongoDB.

* Describe what other knowledge you feel might have helped you with the project development

**Additional Knowledge**: Knowledge of authentication protocols (e.g., JWT) and cloud deployment could enhance future development.