Backend Dev Lesson Plan

We rise by lifting others!

This lesson plan is to be followed in the main batch. This doc consists of in-class assignments that you should take up in the class. You do not need to necessarily cover all things in the same lecture as stated in the plan. It's okay if you have some topic in the next class (or the one after that). However, be comprehensive while teaching 1 topic and move to the next one only after all key concepts of that topic have been covered. Lectures in the video may sometime start after 1 hour of the doubts session, please seek the video accordingly. It may happen that not all key points are covered in the video link shared, please make sure you cover those in your lecture in that case.

Apart from this, make sure you spend at least 15 mins every week motivating the students. Letting them know that they definitely will get placed if they attend all classes and submit all assignments.

In general, try to be friendly, accepting & patient while answering doubts/ teaching concepts.

Lecture 1 - Node and Express - Backend System

- Key concepts to be covered-

```
    1. Intro to the backend - Client Server Architecture

    2. What is a single-threaded backend

    3. Components in Backend System: Server, DataBase

    4. SingleThreaded JS and Async Behavior

    5. Setting Up Node

    6. Advantages of Node.js

    7. Working Demo - Running Node environment

    8. Basic operations in the node REPL
```

Lecture 2- Web APIs

```
    1. Intro to Web APIs

    2. Create a hello world web api

    3. Access data from body return custom data
```

```
     4. Create a basic calculator API : Perform operations ADD/ SUBTRACT/
DIVIDE/MULTIPLY.

     5. Show status code 200

     6. Show exception: CANNOT DIVIDE BY ZERO with different status code
```

Lecture 3- NPM - Package Manager and Usage

- Key concepts to be covered-

```
    1. Intro to Npm

    2. Package.json and Npm

    3. Express - Server and Express Server

    4. Intro to Express.js

    5. Advantages of Express.js

    6. Working Demo of the calculator with express

    7. Calculator demo with Query string and params
```

Lecture 4- TP: Status codes, Testing using postman

- Key concepts to be covered-

```
 1. Status Codes, Postman, Demo of each API method, Testing using Postman: For
the calculator API formed earlier

 2. Storage: Intro to Storage, File Storage, FS Module

 3. Using FS Module, create a basic JSON file to be used as a database for a
todo app, and create CRUD APIs for ToDo App. Add todo, delete todo, update
todo, get 1 todo, get all todos. Show status code 201 in add todo

 4. Do discuss the thinking process behind API and data saving structure chosen
for the todo app.

 5. Highlight the cons of using system files for storage.

 6. Use postman exhaustively while building the todo APIs
```

Lecture 5- Intro to MongoDB

```
   1. Why Databases, what is a database, Two types of databases.

   2. Structure of NoSQL databases, documents, ids, relations, Concept of ORM,
mongoose installation, show mongodb compass. Use mongoDB atlas to create a simple
database

   3. Redo TODO app with database as backend. Show commonly used mongoose methods like
getOneById, getMany, update, delete.

   4. Show how responses can be sorted directly in the db call using orderby.
```

1. Talk about more complex features we will be adding to the todo app. Pagination, Authentication, sorting, edit with history. We will implement these in coming lectures

Lecture 6 - Authentication

```
  1. Concept of middleware in express. Create a request logger Using a middleware to
log request headers, body and url.

  2. Bcrypt installation, create a signup api that creates a hashed password and
stores the username and password in the users table in database

  3. Create a “basic” authentication setup that uses X-Acciojob token value as “Basic
Base64encoded<username:password>”. Create a middleware that gives a 401 status code in
case the header is not present/not verified.
```

Lecture 7- Implementing Todo features

- Use ORM and mongo capabilities extensively to implement: Pagination (limit & skip using query params), Authentication, sorting, edit with history. All in the existing todo application.

Lecture 8,9,10 - SQL and sequelize

1. UML Diagrams and thinking of db structure, datatypes in sql
2. CRUD SQL and **Joins**: Intro, Types (Left, Right, Inner, Outer)
3. Group by order by having,
4. Writing entities, defining relations, todo app structure and all exisitng ORM calls in sequelize

Lecture 11,12,13,14,15,16,17, 18, 19, 20 - Blogging Assignment

- Key concepts to be covered-

```
1. Blogging Site ( Course Final Project ).
```

Video of lecture1 Video of lecture2 Video of lecture3 Video of lecture4 Video of lecture5 Video of lecture6 Video of lecture7 Video of lecture8 Video of lecture9 Video of lecture10 Video of lecture11

Don’t:

1. Do not send response in HTML/XML
2. Do not send string responses
3. No db calls in the controllers

Do’s:

1. API’s should send JSON response (status, message, data)
2. API’s should be rate limited - 500ms = 2hits/sec

MVC - Model(Classes, Schemas), Views, Controller(Routes)

1. **Authentication (Session based auth) - DONE**
    1. Register - email(unique), username(unique), password, name
    2. Login - email/username, password
    3. Logout
2. **Create Tweet - Done**
    1. Only text data
    2. Limit of characters to 1000 max - Send error for limit exceeded
    3. DB Schema should store the creation_datetime and user_details of the user who tweeted it
    4. Schema - {title, text, creation_datetime, userId}
3. **Home Page - Done**
    1. All the tweets in descending order of time
    2. Paginate the API (Limit - 20) (see more or scroll - Frontend)
4. **My Tweets - Done**
    1. All my tweets in descending order of time
    2. Paginate the API (Limit 10)
5. **Edit Tweet - Done**
    1. Edit can only happen until 30 mins from tweeting
6. **Delete Tweet -Done**
    1. Allows users to delete tweets anytime

Database Collection/Tables:

1. User Details
2. Tweets
3. Sessions

Follow Up Tasks:

1. **Follow(Create):** Allows users to follow other user - **Done**
    1. DB entry following_user: userId, follower: myUserId, creation_datetime
2. **Followers List (Read) -** following_user: userId ***Pagination - *Done**
3. **Following List (Read) -** follower: userId - Pagination - **Done**
4. **Unfollow -** Deletes the follow entry from db - follower: userId - **Done**
5. **Bin - Done**

    1. Delete should not delete the item, it should move it to the bin
    2. isDeleted: true, deletion_datetime: time of deletion
    3. Update the read api’s to check for isDeleted: true
    4. Cron to delete the deleted tweets from db - Everyday to delete 30 days old tweets

Database:

1. Follow

Advanced Features:

1. Hashtags
    1. Array of 30 chars string stored in tweets schema - 20 hashtags at max
2. Trending (Top 10)
    1. Tweets on a particular hashtag being used most in last 3 hrs
3. Laugh/Like on tweets - Tweet schema will have a laughReaction, likeReaction keys / {type: laugh/like, tweetId, userId}
4. Comments - Nested tweets

Assignments:

Authentication - JWT Status

Calculator API Status

covid-aggregate Status

Creating a News API server Status

rate-limiting Status

School Administration Status

School Leaderboard Status

Search-college Status

Sharable URLS StatusBackend

To-do List StatusBackend