Backend Syllabus

Prerequisites

Tier 1: Absolutely Essential (Non-Negotiable)

This is the core knowledge the syllabus assumes you already have.

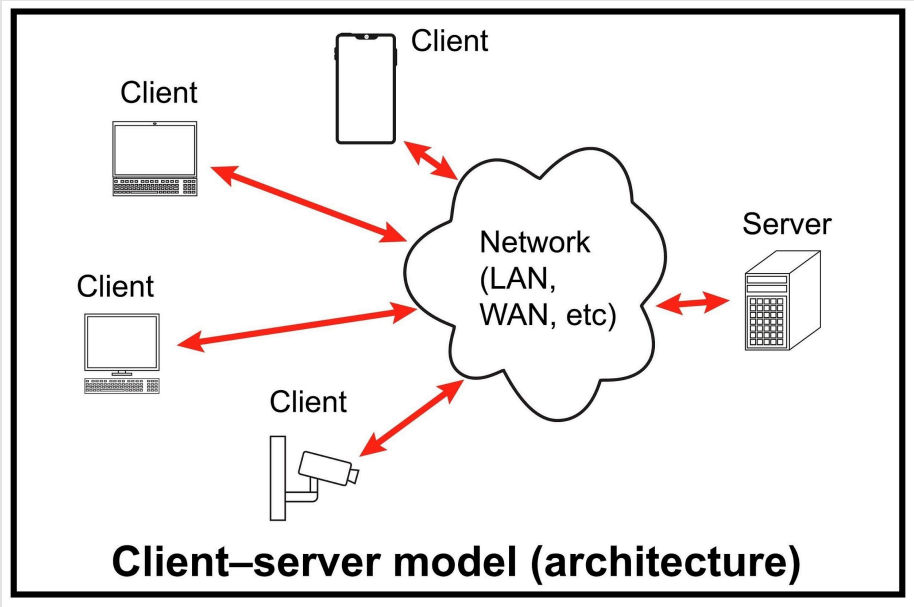
- **Modern JavaScript (ES6+):** The entire syllabus is built on Node.js, which is a JavaScript runtime. You must be comfortable with:
    - **Core Fundamentals:** Variables (`let`, `const`), Data Types, Operators, Loops (`for`, `while`), and Conditionals (`if/else`, `switch`).
    - **Functions:** How to declare functions (standard and arrow functions `() => {}`). Understanding "callbacks" is a huge plus.
    - **Data Structures:**
        - **Objects:** Creating object literals, accessing properties, and understanding `this` (though its behavior in Node.js is simpler than in browser JS).
        - **Arrays:** Creating arrays and using common methods like `.map()`, `.filter()`, `.find()`, and `.forEach()`.
    - **Asynchronous JavaScript (Crucial):** This is the single most important prerequisite. Backend development is full of asynchronous operations (like database queries and API calls). You *must* understand:
        - **Promises:** How to write and consume promises (`.then()`, `.catch()`, `.finally()`).
        - **`async/await`:** This is the modern standard used in almost all backend code. You must know how to use `async` functions and the `await` keyword to handle promises.
    - **ES6+ Features:**
        - **Modules:** Understanding how to `import`/`export` (ESM) or `require`/`module.exports` (CommonJS). The syllabus will heavily use this for file organization.
        - **Destructuring:** Easily pulling properties from objects and arrays.
        - **Spread/Rest Operators:** (`...`).

Tier 2: Core Web Concepts

You need to understand the "environment" your backend will operate in.

- **The Client-Server Model:** You must understand the basic architecture of the web.
    - What is a **Client** (e.g., browser, mobile app)?
    - What is a **Server** (the code you will be writing)?
    - What is a **Request** and a **Response**?

Backend Syllabus



- Shutterstock
- **HTTP Basics:** The syllabus mentions REST APIs, status codes, and request methods. You should know:
    - What the common **methods** are for: `GET`, `POST`, `PUT`, `DELETE`, `PATCH`.
    - What common **status codes** mean (e.g., `200` OK, `404` Not Found, `500` Internal Server Error).
- **Data Formats (JSON):** JSON (JavaScript Object Notation) is the language of modern APIs. You must know what JSON is and how to read and write it.
- **Basic HTML/CSS:** The syllabus includes EJS (a template engine), which mixes HTML with backend data. You don't need to be a frontend developer, but you must know basic HTML tags (`<div>`, `<form>`, `<p>`, etc.) to understand what you are sending to the browser.

Tier 3: Essential Developer Tools

These are the tools you'll use every day.

- **Using the Command Line (Terminal/CLI):** You must be comfortable navigating your computer without a GUI. You should know how to:
    - Navigate folders (`cd`).
    - List files (`ls` or `dir`).
    - Create folders (`mkdir`).
    - Run basic commands.
- **NPM (Node Package Manager):** Since the syllabus uses Node.js, you must know what `npm` is and how to use it to:
    - Initialize a project (`npm init`).
    - Install packages (`npm install <package_name>`).
    - Understand the purpose of `package.json`.

Backend Syllabus

- **Basic Git & GitHub:** The syllabus mentions Git configuration. You should know the basic workflow:
    - `git init`
    - `git add`
    - `git commit`
    - `git push`

Backend Syllabus

Backend Syllabus

Module 1: The Foundation - Node.js & Basic Servers

This module covers the absolute basics: what Node.js is, how to run it, and how to create a server without any frameworks.

- **1. Starting with Node.js - The Beginning** 🏁
  - Introduction to Node.js and Getting Our Tools - Node.js LTS, Postman, Editor
  - Setting up the Tools for our Environments
  - Running script with nodejs - “Namaste Duniya”
  - NPM Basics | Installing Packages.
  - Creating and Managing package.json.
- **2. Creating Server - Writing Our First Server** 📱
  - What is Server and how it works?
  - Setting Up Our First Node.js Server using HTTP
  - Serving A Response to the Browser and Understanding Responses.
  - Routing in HTTP Servers.
  - Understanding Status Code - 1XX , 2XX , 3XX ,404 - Not Found , 200 - success , 500 - Internal Server error , 422 - Invalid Input , 403 - the client does not have access rights to the content , etc.
  - Installing Nodemon for Automatic Server Restarts.

Module 2: The Framework - Express.js Core Concepts

This module introduces the Express framework, its underlying architecture (MVC), and its most critical concepts: middleware and error handling.

- **3. Some talk on Different Architectures** 
  - Different Architectures in backend like MVC and SOA.
  - Understanding MVC Architecture Model , View ,Control.
  - MVC in the context of REST APIs.
- **4. Web Framework - Express.js** 🚀
  - what is Express.js and why to use it.
  - Setting Up Express Server .
  - Returning Response from the server.
  - Using Query Parameters and URL Parameters.
  - HTTP Request - Some Important part of requests , Different Types of Requests - Get , Post , PUT , Patch , Delete.
- **6. Middleware in Express.js (one of my favorite)** 🐵
  - Understanding the middleware in express.
  - Implementing middleware with express.
  - Different types of middleware : builtIn middleware , third-party middleware ,custom middleware .
  - Different level of middleware : Application-Level , Router-Level .
- **16. Error handling in express** 🛑

- Basic Error Handling in Express next() .
- Catching Specific Errorstry &catch .
- Creating Util Class for Error Handling.

Module 3: Server-Side Rendering & Static Files

This module focuses on the "V" in MVC: how to serve dynamic HTML pages using a template engine and how to serve static assets like CSS and images.

- **5. Template Engine - EJS** 🚜
  - What is Template Engine and What is the use of Template Engine.
  - Template Engine Option - Handlebars , EJS , Pug , jade but We'll use EJS .
  - Setting Up Template Engine - Installed EJS template engine.
  - Rendering Our First Page using EJS and Some important syntax - <%= %> , <% %> , <%- %>.
  - Loop statement, Conditional statement and Locals in views - EJS.
  - Accessing the Static Files Inside EJS file.
  - *Includes concept from section 4: Serving Static Files with express.static()*
- **7. Handling file with Express** 📁
  - Understand Multer and its usecase?
  - Uploading file with multer.
  - Understanding Memory and Disk Storage.
  - Accessing uploaded file req.file.
  - Working with express.static.
  - Using Cloudinary or Imagekit for Real-time media processing APIs and Digital Asset Management.

Module 4: Databases - From Basics to Optimization (MongoDB)

This is a comprehensive module on the database. It starts with the basics of MongoDB and Mongoose and moves directly into advanced optimization techniques.

- **8. Beginning of Database Basics (Bohot km theory)** 🗄️
  - Relational and non-relational Databases : mongodb & mysql .
  - What is MongoDB? Why Use It?
  - Installing Compass and Understand how to access DB using terminal.
  - Setting Up MongoDB Locally and in the Cloud.
  - Understanding Datatypes Collections and Documents.
  - Connecting MongoDB to Node.js with Mongoose .
  - Database Relations - One to One , One to Many OR Many to One , Many to Many , Polymorphic .
  - Handling Relationships with Mongoose (populate).
- **10. Database Optimization for Fast response** 🧘

- Indexing for Performance with MongoDB :- Single-Field Indexes , Compound Indexes , Text Indexes ,Wildcard Indexes.
- Best practice with Indexing explain().
- Learning MongoDB Aggregation.
- Comparison Operators - [$eq , $ne , $lt , $gt , $lte , $gte , $in , $nin]
- Logical Operators - [$not , $and , $or and $nor]
- Array[$pop, $pull, $push and $addToSet]
- Stages in Aggregation pipeline :- $match , $group ,$project ,$sort ,$lookup.
- Creating Database on Local and Atlas
- Creating parallel pipeline with $facet .
- Learning MongoDB Operators.
- Understanding Different types of Operators :- Comparison ,Regex ,Update ,Aggregation.

Module 5: Building the REST API

With Express and Databases covered, this module combines them to build a full REST API, including validation and security best practices.

- **9. API Development(REST)** ⛓
  - What is a REST API?
  - Versioning in RESTful APIs - /v1/
  - Using Postman for API Testing and developing - Send Requests , Save Collections , Write Tests .
  - Understanding and Working With Status code , 2xx (Success) , 4xx (Client Errors) , 5xx (Server Errors) .
  - Validating API Inputs Using libraries like express-validator or Sanitization .
  - Security Handling - Rate Limiting with express-rate-limit ,XSS Attack , CSRF Attack , DOS Attack.
  - *Includes middleware concepts from section 6: Handeling Errors and Security with middleware : Error-Handling , Helmet , CORS.*

Module 6: Authentication & Authorization

This module is dedicated to the critical feature of securing the API and managing user access.

- **13. Authentication and Authorization** 🪪
  - Difference Between Authentication & Authorization
  - Working with Passwords and Authentication - Cookie Authentication , OAuth Authentication
  - Understanding Session and Token Authentication.
  - Implementing JWT Authentication :- jsonwebtoken JWT_SECRET.
  - Securing user password with bcrypt hashing salt.
  - Role-Based Access Control (RBAC).

- Authenticating user with Express middleware .
- Understanding Passport.js and its usecase?
- Glancing through and Installing Passport.js
- Setting up Passport.js - passport-local, local-strategy , google-OAuth
- express-sessions and using passport for authentication.

Module 7: Advanced Features & Performance

These modules cover topics that build on top of a functional application, adding real-time capabilities and performance optimization.

- **14. Working Real time communication : WebSockets and socket.io** 💬
  - Understanding WebSockets protocol for realtime applications?
  - Learning handshake ,Persistent connection ,Bidirectional communication ,HTTP polling .
  - Understanding difference between WebSocket Vs Socket.io.
  - Working with socket.io for realtime applications.
  - Understanding usage ofRooms in Socket.io.
  - Understanding Middleware in Socket.io.
- **15. Working With Caching - Local and Redis** 🍄
  - What is Caching and How to cache data locally?
  - What is Redis?
  - Why Use Redis for Caching?
  - Implementing Redis Caching in Node.js.
  - Advanced Redis Features TTL ,Complex Data Structures , Pub/Sub.

Module 8: Production & Maintenance

The final module covers "real-world" practices: making your code maintainable, logging issues, and testing.

- **12. Production Wala Project Structure and Configuration** 🗼
  - Understanding the Basic Structure of application.
  - Learning File Naming Conventions, Git Configuration,
  - Understanding Important Folders :- src/ ,config/ ,routes/ ,utils/ .
  - Role of package.json , ENV and .gitignore .
  - Production Environment - PM2 , Error & Response Handling Configuration , CORS Configuration , async-handler.js.
  - Using and Configuring ESLint and Prettier for code formatting.
  - Testing APIs using Postman.
- **11. Logging Backend : Express.js**
  - Why is Logging Important?
  - Setting Up Logging with Libraries winstone ,Pino ,Morgan .
  - Different mode of morgan ,dev ,short ,tiny .

    - Error Handling and Logging.
- **17. Testing Tools 🛠️**
    - Understanding Unit-Testing With Jest.
    - Cross Browser Testing and Why Is It Performed?
    - What Is Web Testing? and How to Test a Website.