EXPRESS.JS FOR APIs ROADMAP

  **Routing**:

* Basic route handling
* Route parameters and dynamic routes
* Route query parameters
* Route middleware
* Route nesting and modularization

 **Middleware**:

* Built-in middleware (e.g., **express.json()**, **express.urlencoded()**)
* Custom middleware functions
* Error handling middleware
* Third-party middleware (e.g., authentication middleware, logging middleware)

 **Request and Response**:

* Accessing request headers, query parameters, and body
* Response methods (**send**, **json**, **status**, etc.)
* Handling file uploads
* Working with cookies and sessions

 **Error Handling**:

* Basic error handling using try-catch
* Implementing error-handling middleware
* Handling asynchronous errors and promise rejections
* Sending appropriate error responses (status codes, error messages)

 **Database Integration**:

* Connecting to a database (e.g., MongoDB, MySQL, PostgreSQL)
* Performing CRUD operations (Create, Read, Update, Delete)
* Working with database libraries (e.g., Mongoose, Sequelize)
* Database migrations and schema management

 **Authentication and Authorization**:

* Implementing user registration and login
* Using JWT (JSON Web Tokens) for authentication
* Protecting routes with authentication middleware
* Role-based authorization and permission handling

 **Validation**:

* Validating request data (e.g., input validation, schema validation)
* Using validation libraries (e.g., Joi, Validator.js)
* Handling validation errors and providing meaningful responses
* Custom validation middleware

 **API Documentation**:

* Documenting API endpoints and request/response schemas
* Using Swagger or other API documentation tools
* Adding descriptions and examples to endpoints
* Generating API documentation from code annotations

 **Testing**:

* Writing unit tests for individual functions and modules
* Integration testing of API endpoints
* Using testing frameworks (e.g., Mocha, Chai, Supertest)
* Mocking dependencies and using test doubles

 **Security**:

* Cross-Site Scripting (XSS) prevention
* Cross-Site Request Forgery (CSRF) protection
* Implementing rate limiting and request throttling
* Securely storing sensitive information (e.g., passwords, API keys)