Fantastic, it's good to know that your students already have some background knowledge and that you're using MongoDB for the database. Since your focus is more model-centric and the students are expected to have some familiarity with the concepts, you can dive right into the core components of the project. Below is a lesson plan targeted towards getting the project up and running, with a focus on coding:

### THINKIFIC Lesson Plan

---

#### Lesson 1: Project Setup and Initial Server Creation

1. \*\*Objective\*\*: Setting up the project and initializing the server.

2. \*\*Activities\*\*:

- Walkthrough of the existing folder structure.

- Initializing a new Node.js project with npm.

- Creating a basic Express server in `server.js`.

3. \*\*Code Along\*\*: Writing the code for the basic server setup.

#### Lesson 2: MongoDB Connection

1. \*\*Objective\*\*: Connect the Express server to MongoDB.

2. \*\*Activities\*\*:

- Introduction to the MongoDB driver or Mongoose.

- Code the MongoDB connection in `server.js`.

3. \*\*Code Along\*\*: Write the MongoDB connection code.

#### Lesson 3: Building the Models

1. \*\*Objective\*\*: Create MongoDB schemas and models for Users, Products, and Orders.

2. \*\*Activities\*\*:

- Discuss the schema designs.

- Create models in the `model` folder.

3. \*\*Code Along\*\*: Write the code for the User, Product, and Order models.

#### Lesson 4: Routes and Controllers for User

1. \*\*Objective\*\*: Implementing CRUD operations for User.

2. \*\*Activities\*\*:

- Setup routes in the `routes` folder.

- Create controller methods for CRUD operations in the `controller` folder.

3. \*\*Code Along\*\*: Implement the CRUD operations for the User model.

#### Lesson 5: Routes and Controllers for Products

1. \*\*Objective\*\*: Implementing CRUD operations for Products.

2. \*\*Activities\*\*:

- Review the Product model schema.

- Create controller methods for CRUD operations.

3. \*\*Code Along\*\*: Implement the CRUD operations for the Product model.

#### Lesson 6: Routes and Controllers for Orders

1. \*\*Objective\*\*: Implementing CRUD operations for Orders.

2. \*\*Activities\*\*:

- Discuss the relationship between Users and Orders.

- Create controller methods for CRUD operations.

3. \*\*Code Along\*\*: Implement CRUD operations for the Order model.

#### Lesson 7: Testing CRUD Operations

1. \*\*Objective\*\*: Test CRUD operations using Postman or a similar tool.

2. \*\*Activities\*\*:

- Demonstrate how to test CRUD operations.

- Debug common issues.

3. \*\*Code Along\*\*: Perform CRUD tests.

#### Lesson 8: Basic Authentication

1. \*\*Objective\*\*: Implement basic authentication for User login.

2. \*\*Activities\*\*:

- Add login and registration routes and controllers.

- Implement basic session handling or JWT.

3. \*\*Code Along\*\*: Code the authentication features.

#### Lesson 9: Basic Front-end Views

1. \*\*Objective\*\*: Create simple views to interact with the backend.

2. \*\*Activities\*\*:

- Introduce a template engine like EJS.

- Create basic forms and lists to display data.

3. \*\*Code Along\*\*: Build simple views for User, Product, and Order models.

#### Lesson 10: Deployment and Review

1. \*\*Objective\*\*: Deploy the application and review the course.

2. \*\*Activities\*\*:

- Demonstrate a simple way to deploy the project.

- Review what was learned and how to extend the project.

3. \*\*Code Along\*\*: Deploy the application.

---

Feel free to modify these lessons to better suit the needs and pacing of your students. Given that they have pre-existing knowledge, this plan aims to be hands-on and dives straight into building the project.

Certainly! Let's go ahead and start with Lesson 1, which is all about setting up the project and initializing the server. I'll assume you've already created a new directory for your project and you have Node.js installed,

**Lesson 1: Project Setup and Initial Server Creation**

Objective:

Setting up the project and initializing the server.

:Activities

\*\*Initialize a new Node.js Project with NPM\*\* .1

:Open the terminal, navigate to your project folder, and run

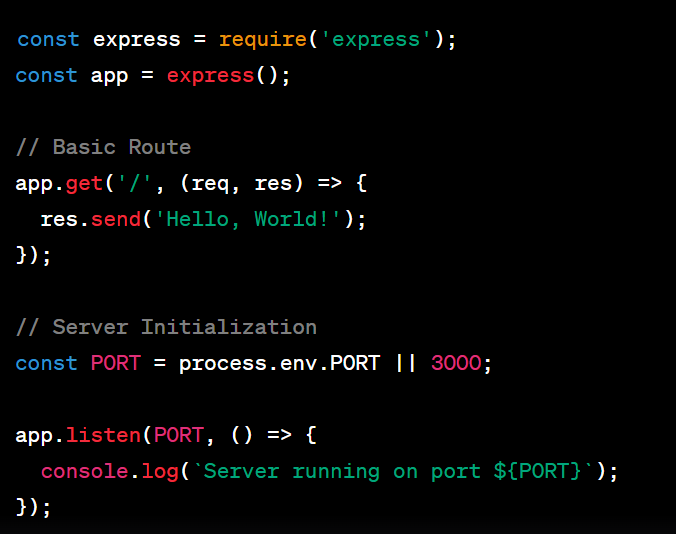
**npm init -y**

This will create a `package.json` file that keeps track of all your project's dependencies.

\*\*Install Express\*\*. 2

Next, let's install the Express framework.

**npm install express**

\*\*Create `server.js` File\*\*.1

And that's it for Lesson 1! This sets up a basic Express server, and you can move on to connecting to MongoDB in the next lesson. Feel free to go through the code with your students, ensuring they understand each step.

Certainly! Let's proceed to Lesson 3 where you'll work on setting up your MongoDB schemas

and models.

**Lesson 2: integrate MongoDB**

**תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי**

**Npm install dotenv**

**Lesson 3: Building the Models**

#### Objective:

Create MongoDB schemas and models for Users, Products, and Orders.

#### Activities:

1. \*\*Install Mongoose\*\*

Mongoose is a popular library for MongoDB and Node.js that simplifies interactions with your database. To install Mongoose, run:

**npm install mongoose**

2. \*\*Create Model Files\*\*

In your `model` directory, create three files:

- `User.js`

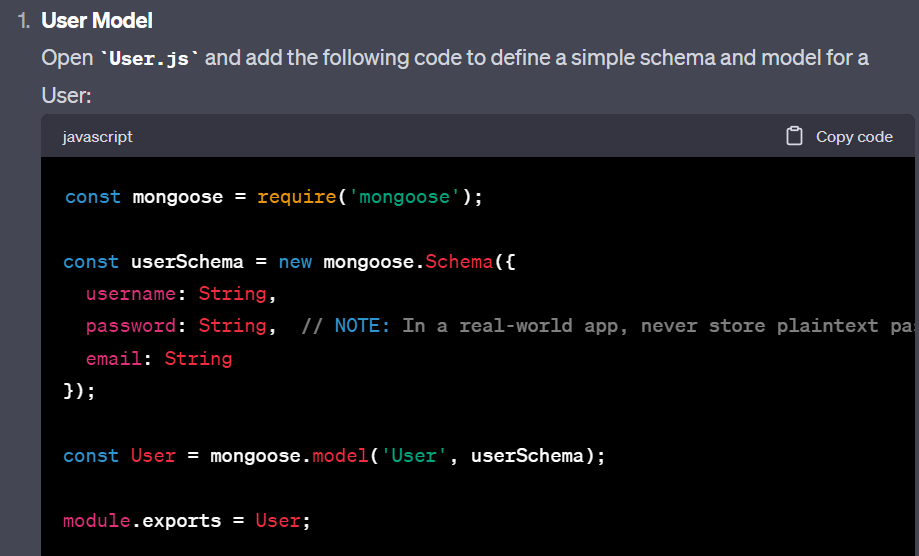
- `Product.js`

- `Order.js`

#### Code Along: Building the Models

**1. \*\*User Model\*\***

Open `User.js` and add the following code to define a simple schema and model for a User:



**2. \*\*Product Model\*\***

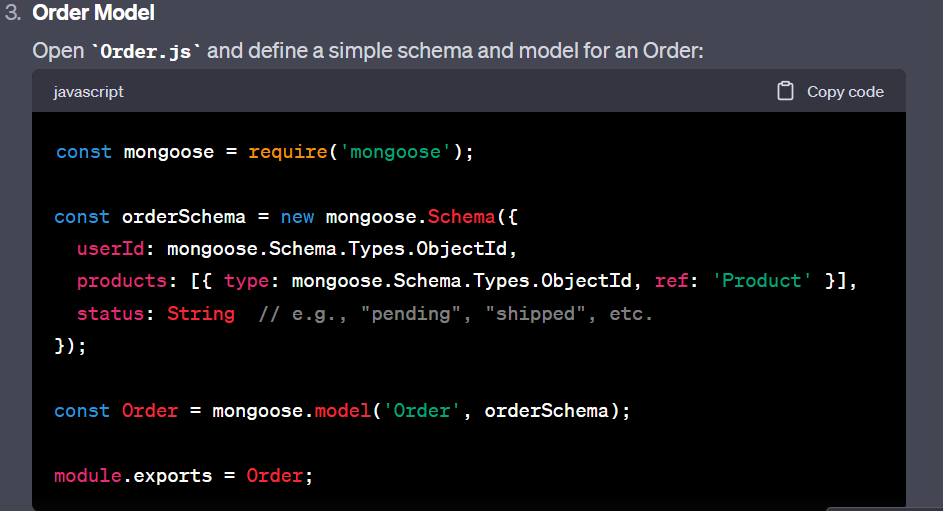
Open `Product.js` and define a simple schema and model for a Product:

תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי

**3. \*\*Order Model\*\***

Open `Order.js` and define a simple schema and model for an Order:



In each model, we define a schema that specifies the fields we expect to have in each collection in the MongoDB database. Once the schema is defined, we then create a model using `mongoose.model()`.

We're also exporting the models to use them later in controllers and routes. Note that in a real-world application, you'd also want to add data validation, but for the scope of this project, we're keeping it simple.

Great! Let's move on to Lesson 4, where you'll start creating routes and controllers to handle CRUD operations for the User model.

**Lesson 4: Routes and Controllers for User**

#### Objective:

Implementing CRUD operations for the User model.

#### Activities:

**1. \*\*Setup User Controller\*\***

In your `controller` directory, create a new file named `userController.js`.

**2. \*\*Setup User Routes\*\***

In your `routes` directory, create a new file named `userRoutes.js`.

#### Code Along: Implementing CRUD Operations for User

**1. \*\*User Controller\*\***

Open `userController.js` and import the User model at the top:

תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי2. **\*\*User Routes\*\***

Open `userRoutes.js` and set up your routes:

**3. \*\*Connect Routes to Main Server\*\***

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטיOpen `server.js` and import the user routes:

This should set up all the CRUD operations for the User model. You can test these operations by running your server and using a tool like Postman to make API requests.

**Lesson 5: Routes and Controllers for Product and Order**

1. \*\*Setup Product and Order Controllers\*\*

Create two new files in your `controller` directory:

- `productController.js`

- `orderController.js`

2. \*\*Setup Product and Order Routes\*\*

**1. \*\*Product Controller\*\***

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי**2. \*\*Order Controller\*\***

Open `orderController.js` and import the Order model:

const Order = require('../model/Order');

```

3. \*\*Product Routes\*\*

Open `productRoutes.js` and set up your routes:

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

4. \*\*Order Routes\*\*

Open `orderRoutes.js` and set up your routes:

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

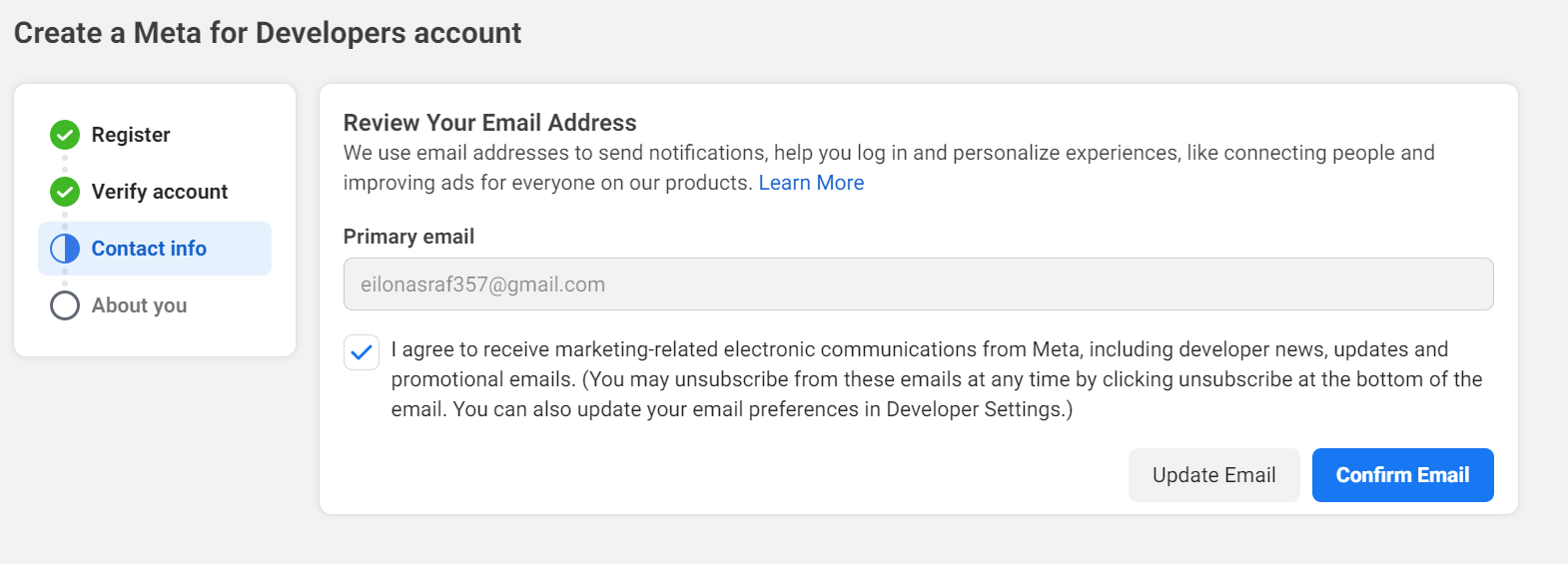
5. \*\*Connect All Routes to Main Server\*\*

Open `server.js` and import the new routes:

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

After completing this lesson, you should have a fully functioning backend with CRUD operations for Users, Products, and Orders. You can run your server and test these CRUD operations using a tool like Postman or your front-end application.

FACEBOOK API-   


תמונה שמכילה טקסט, צילום מסך, תוכנה, מספר

התיאור נוצר באופן אוטומטי

npm install passport-facebook

npm install fb

תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי**Lesson 6: Login, Logout, Signup, Admin**

**Npm install crypt's**

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי

**Lesson 7: Basic Front-end views**

**npm install ejs**

**npm install cors**

**npm install express-fileupload**

**npm install express mongoose dotenv path cors express-fileupload ejs**

**npm install express-session**

**npm install method-override**

**npm install**

**npm install multer**

* **בורסהnpm install axios**