🍜 Reatl — Reels-Style Food Ordering App

MERN app where users browse short food videos and order instantly.

MVP: Reels feed, one-tap ordering, real-time order status.

📃 Reatl — Overview

Short vertical, fast-consumption UX like Reels/Shorts where users browse short food videos (or micro-ads) and can instantly order the dish shown. Emphasis: snackable discovery → 1-tap ordering.

🗃️ File Structure:

```
Reatl App
    |---> Frontend
    |---> Backend
```

🧍 **Normal User (Customer)**

Browse Food Content (Reels or list view) + Place Orders + User Login/Logout + Food List/Items View

🏪 **Food Partner (Restaurant / Vendor)**

Upload and manage dishes & reels.

🌐 Backend Building Roadmap

We'll start **Building our Backend First**.

✅ Initiate npm:

```
npm init -y
```

We'll get a file named `package.json` in our folder.

✅ Server Creation:

Install Express

```
npm i express
```

✅ Folder Creation:

```
backend
    |--node_modules
    |--src
    |    |--app.js ☑
    |--server.js ☑
    |--package.json
    |--package-lock.json
```

☑ Server Creation: We'll create the server in `app.js`

```
const express = require('express')
//server's instance is created
const app = express()

app.get('/',(req,res)=>{
    res.send('Welcome to the Reatl API')
})

module.exports = app
```

1. We'll Create a simple Express.js server through `const express = require('express')`
2. Instance of the express server is created - `const app = express()`
3. Dummy Route Created through - `app.get('/', (req, res) => { res.send('Welcome to the Reatl API')})`
4. Exporting the app to the server.js file. We'll use this app in the server.js file. `module.exports = app`

☑ Start Server: We'll start the server in `server.js`. Server is running on port 3000.

Importing the app from the app.js file.

```
const app = require('./src/app')

app.listen(3000, () => {
    console.log('Server is running on port 3000');
});
```

☑ Refresh Server if new changes occur in any way

```
npx nodemon server.js
```

☑ Will you Postman API to test

GET: `http://localhost:3000/` -> Send ➡

☑ Server Connetion with Database

Using Community edition of MongoDB

- Have to install `npm i mongoose`
- Create `db.js` inside `db` folder in `src` ➡ `/src/db/db.js`

```
const mongoose = require('mongoose')


function connectDB(){
    mongoose.connect("mongodb://localhost:27017/reatl")
    .then(()=>{
        console.log('☑ Connected to MongoDB');
    })
    .catch((err) => {
        console.log("✖ Error connecting to MongoDB", err);
    });
}
```

1. We'll Connect to the MongoDB database through `const mongoose = require('mongoose')`
2. We'll code how the database will be connected to the server through `function connectDB()`
3. `mongodb://localhost:27017` (The default port for MongoDB is 27017.) & name of the database - `reatl`
4. If the connection is successful, the following messages will be printed to the console or not. (`err` -> Error)
5. Until the function is getting called it won't connect our database to the server as all the logics are written here.
6. We'll call the function in `server.js` file.