🍳 Overcooked-Style Kitchen Server (For User Study)

Welcome! This fake Overcooked-style kitchen server is part of a **user study**. This server simulates a multiplayer cooking game where players submit dish orders, and the system processes them with **intentional latency**, **parallel execution**, and **detailed logs**.

🎯 Study Instructions

Please follow the setup steps below. During the study, you will interact with the logs produced by this server using our VS Code extension.

> ⚠️ **Do not open or modify anything in the `please-do-not-open/` folder.**
> This folder contains internal logic meant to remain hidden for the purpose of the study.

> ✅ **Only open `open-when-requested/` when you are instructed to do so.**
> This folder contains your assigned tasks (e.g., `task1.md`, `task2.md`).

📁 Project Structure

```
please-open-here/
├── server.js              # Main server logic (you may view this)
├── example.js             # Sample client that sends a request to the server
├── do-not-modify.js       # Script to send custom requests
├── server-backup.js       # Backup script for you to start the second task
├── please-do-not-open/    # 🔒 Internal logic (DO NOT OPEN)
└── open-when-requested/   # ✅ Task instructions (open when asked)
    ├── task1.md
    └── task2.md
```

🛠️ Setup Instructions

Please execute the following commands to install dependencies and compile the server.

1. Set up the client folder

```
cd ./please-open-here
npm run setup
```

2. Start the server

```
node ./server.js
```

You should see:

```
🍳 Overcooked Kitchen running at http://localhost:3000
```

▶️ Try Submitting a Sample Order

In a **separate terminal**, run the example client:

```
cd ./please-open-here
node ./example.js
```

This will simulate a player placing an order like `burger` and `salad`. You will see a sequence of logs printed in the terminal representing each step of dish preparation.

📦 API Summary (optional)

You are not required to interact directly with these endpoints, but here's what they do for reference:

`POST /order`

Submit an order with player ID and dish names.

❓ Need Help?

If anything doesn't work or you have questions during the session, please ask the study organizer.

Thanks for participating and helping us improve developer tools! 🙌

🧩 High-Level Flow (Call Stack + Async)

When an order is placed, the async function chain looks like this:

```
[async POST handler]
   └── prepareOrder(orderId, player, dishes)
         └── Promise.all(dishes.map(...))
               └── prepareDish(dish, orderId)
                     └── recipes[dish](orderId)
                           └── (prepareBurger OR prepareSalad)
```

All major operations inside `prepareOrder`, `prepareDish`, and the recipe functions (`prepareBurger`, etc.) are asynchronous — many of them simulate latency with `await sleep(...)`.