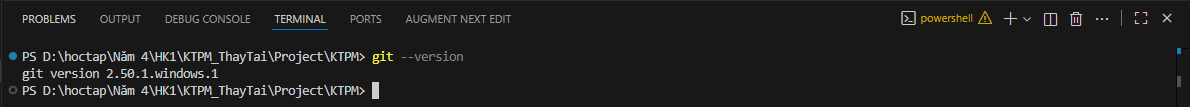
**Install Node.js và npm**  
**A black screen with a black background

AI-generated content may be incorrect.**

**Install Git**



**Install Docker**

A black screen with white text

AI-generated content may be incorrect.

Create development branch

A screenshot of a computer

AI-generated content may be incorrect.

**Install Dependencies**

cd BackEnd

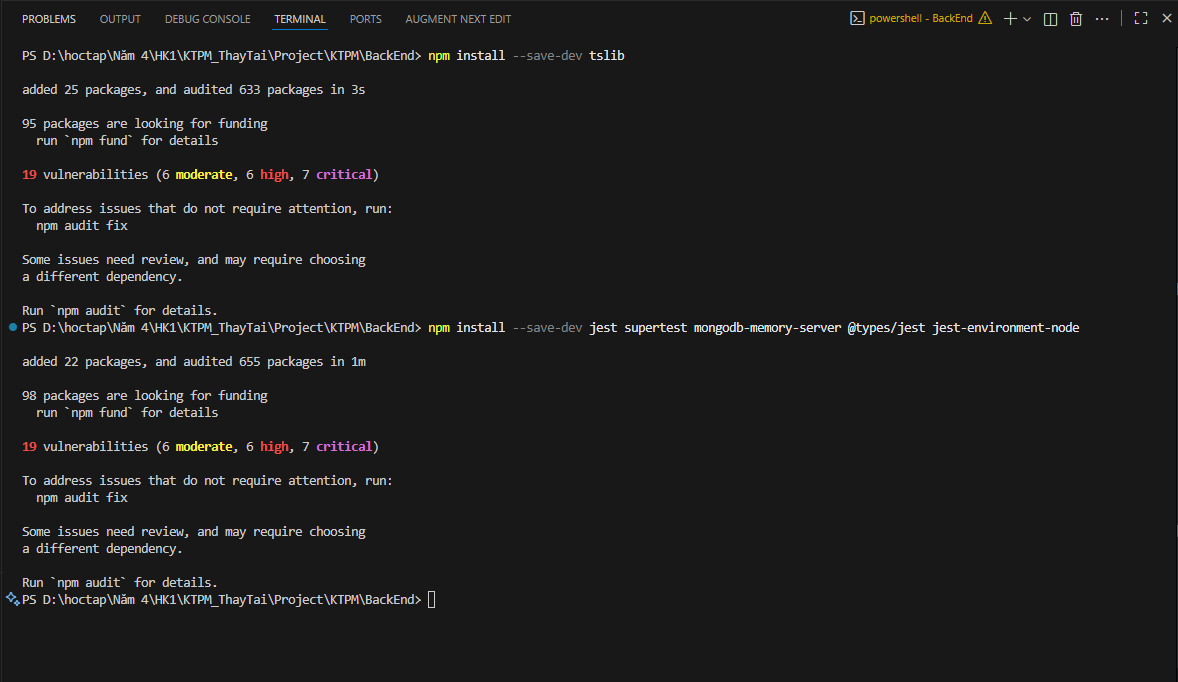
# Install production dependencies

npm install

# Install additional testing dependencies

npm install --save-dev jest supertest mongodb-memory-server @types/jest jest-environment-node

| **Gói Thư Viện** | **Mục Đích Chính** | **Vai Trò Trong Kiểm Thử** |
| --- | --- | --- |
| **jest** | **Framework kiểm thử** hàng đầu cho JavaScript. | Cung cấp bộ công cụ để chạy, tổ chức, và viết các bài kiểm thử (Unit/Integration). |
| **supertest** | Giúp **kiểm thử HTTP assertions** dễ dàng. | Được sử dụng để gửi các HTTP request đến API Express.js (Backend) và kiểm tra response (status code, body, headers). Cần thiết cho **Integration Tests** API. |
| **mongodb-memory-server** | Cung cấp một phiên bản **MongoDB ảo** chỉ chạy trong bộ nhớ (RAM). | Được sử dụng để tạo một database tạm thời, **sạch sẽ và độc lập** cho mỗi lần chạy kiểm thử mà không cần cài đặt MongoDB vật lý. Giúp tăng tốc độ và độ tin cậy của Unit/Integration Tests. |
| **@types/jest** | **Type definitions** cho Jest. | Cần thiết nếu dự án sử dụng TypeScript (cung cấp gợi ý code và kiểm tra kiểu). |
| **jest-environment-node** | Định cấu hình môi trường chạy test. | Đảm bảo rằng Jest chạy trong môi trường **Node.js** (chứ không phải trình duyệt), phù hợp cho việc kiểm thử Backend. |



**Database Setup**

**3.1. MongoDB với Docker (Cho Phát Triển và Integration Tests)**

**# Start MongoDB container**

**docker run -d --name ktpm-mongodb-test -p 27017:27017 -e MONGO\_INITDB\_ROOT\_USERNAME=admin -e MONGO\_INITDB\_ROOT\_PASSWORD=password -e MONGO\_INITDB\_DATABASE=ktpm\_test mongo:7.0**

Mục đích của bước này là cung cấp một **môi trường cơ sở dữ liệu thực tế** và **ổn định** để bạn phát triển ứng dụng và chạy các bài kiểm thử tích hợp (Integration Tests).

| **Lệnh** | **Mục Đích** | **Chi Tiết** |
| --- | --- | --- |
| docker run -d --name ktpm-mongodb-test ... | **Khởi động Container MongoDB**. |  |
| -d | Chạy container ở chế độ **detached** (nền). |  |
| --name ktpm-mongodb-test | Đặt tên cho container để dễ quản lý. |  |
| -p 27017:27017 | **Ánh xạ cổng** (Port Mapping). | Mở cổng 27017 trên máy tính cục bộ của bạn để kết nối với cổng 27017 bên trong container MongoDB. |
| -e MONGO\_INITDB\_... | **Thiết lập biến môi trường**. | Tạo người dùng admin và đặt tên database khởi tạo là ktpm\_test. |
| mongo:7.0 | **Image Docker** được sử dụng. | Kéo và chạy phiên bản MongoDB 7.0. |
| docker ps | **Kiểm tra trạng thái**. | Xác minh rằng container đang chạy. |
| docker logs ktpm-mongodb-test | **Xem nhật ký**. | Kiểm tra xem MongoDB đã khởi động thành công bên trong container hay chưa. |

A screenshot of a computer program

AI-generated content may be incorrect.

| **Mục Đích** | **Kết Quả trong Ảnh** | **Ý Nghĩa** |
| --- | --- | --- |
| **Pull Image** | Pulling from library/mongo... Pull complete | Docker đã không tìm thấy image mongo:7.0 cục bộ, nên đã **tự động tải** image này từ Docker Hub (library/mongo) thành công. |
| **Status** | Status: Downloaded newer image for mongo:7.0 | Xác nhận việc tải image thành công. |
| **Container ID** | 08ae...7f83f1 | Đây là Container ID **đã được khởi tạo** thành công ở chế độ nền (-d). |
| **Lỗi** | Không có thông báo lỗi docker run nào xuất hiện. | Lệnh chạy container đã được thực thi thành công. |

**Verify MongoDB is running**

docker ps

docker logs ktpm-mongodb-test

**3.2. MongoDB Memory Server (Cho Unit Tests)**

Mục đích của phần này là cung cấp một **database ảo, nhanh chóng và độc lập** cho các bài **Unit Tests**. Đã cài đặt ở bước trước. Sẽ tự động start/stop trong test files

| **Đặc Điểm** | **Mô Tả** | **Lợi Ích** |
| --- | --- | --- |
| **Database trong bộ nhớ (RAM)** | mongodb-memory-server sẽ tải một phiên bản MongoDB nhị phân và chạy nó hoàn toàn trong RAM của máy tính. | Tốc độ cực nhanh vì không cần ghi/đọc từ ổ đĩa. |
| **Tự động start/stop** | Server sẽ được khởi tạo trong hook beforeAll và dừng lại trong afterAll hoặc afterEach của Jest. | Đảm bảo mỗi bài kiểm thử có một **database sạch** và độc lập, tránh xung đột dữ liệu giữa các test. |
| **Không cần Docker** | Server này hoạt động độc lập với Docker. | Giúp các bài Unit Test có thể chạy mà không cần Docker hoặc bất kỳ dịch vụ database nào khác đang chạy nền. |

**4. Testing Framework Setup ( tương tự mục 4 trong ENVIRONMENT\_SETUP\_GUIDE.md):**

**4.1. Jest Configuration (Backend)**

| **File** | **Mục đích** | **Vai trò chính** |
| --- | --- | --- |
| BackEnd/jest.config.js | **Cấu hình Jest**. | Định nghĩa cách Jest hoạt động, bao gồm: |
|  |  | \* **testEnvironment: 'node'**: Chỉ định môi trường chạy test là Node.js, phù hợp cho Backend. |
|  |  | \* **setupFilesAfterEnv**: Chỉ định file thiết lập database/môi trường chạy trước khi các test file được thực thi. |
|  |  | \* **collectCoverageFrom**: Xác định những file nào cần thu thập dữ liệu coverage (controllers, models, utils, middlewares). |
|  |  | \* **forceExit, clearMocks**: Đảm bảo môi trường test được dọn dẹp sạch sẽ sau mỗi lần chạy. |

4.2. Vitest Configuration (Frontend)

| **File** | **Mục đích** | **Vai trò chính** |
| --- | --- | --- |
| FrontEnd/vite.config.js | **Cấu hình Vitest** (thông qua file cấu hình Vite). | Định nghĩa cách Vitest hoạt động cho Frontend React: |
|  |  | \* **environment: 'jsdom'**: Mô phỏng môi trường DOM của trình duyệt (cần thiết cho React Component Testing). |
|  |  | \* **globals: true**: Cho phép sử dụng các hàm test toàn cục (describe, test) mà không cần import. |
|  |  | \* **setupFiles**: Chỉ định file thiết lập môi trường giả lập (mock) trình duyệt. |
|  |  | \* **coverage**: Thiết lập định dạng báo cáo coverage (text, json, html). |

Các bước trong mục **"4. Testing Framework Setup"** này nhằm mục đích **thiết lập và cấu hình hoàn chỉnh** môi trường kiểm thử **Unit Testing** và **Integration Testing** cho cả Backend (sử dụng Jest) và Frontend (sử dụng Vitest) của dự án.

Nó là bước then chốt để đảm bảo các bài kiểm thử có thể chạy tự động, độc lập và cung cấp báo cáo coverage chính xác.

**⚙️ Chi Tiết Về Mục Đích Của Từng Bước**

**4.1. Jest Configuration (Backend)**

| **File** | **Mục đích** | **Vai trò chính** |
| --- | --- | --- |
| BackEnd/jest.config.js | **Cấu hình Jest**. | Định nghĩa cách Jest hoạt động, bao gồm: |
|  |  | \* **testEnvironment: 'node'**: Chỉ định môi trường chạy test là Node.js, phù hợp cho Backend. |
|  |  | \* **setupFilesAfterEnv**: Chỉ định file thiết lập database/môi trường chạy trước khi các test file được thực thi. |
|  |  | \* **collectCoverageFrom**: Xác định những file nào cần thu thập dữ liệu coverage (controllers, models, utils, middlewares). |
|  |  | \* **forceExit, clearMocks**: Đảm bảo môi trường test được dọn dẹp sạch sẽ sau mỗi lần chạy. |

**4.2. Vitest Configuration (Frontend)**

| **File** | **Mục đích** | **Vai trò chính** |
| --- | --- | --- |
| FrontEnd/vite.config.js | **Cấu hình Vitest** (thông qua file cấu hình Vite). | Định nghĩa cách Vitest hoạt động cho Frontend React: |
|  |  | \* **environment: 'jsdom'**: Mô phỏng môi trường DOM của trình duyệt (cần thiết cho React Component Testing). |
|  |  | \* **globals: true**: Cho phép sử dụng các hàm test toàn cục (describe, test) mà không cần import. |
|  |  | \* **setupFiles**: Chỉ định file thiết lập môi trường giả lập (mock) trình duyệt. |
|  |  | \* **coverage**: Thiết lập định dạng báo cáo coverage (text, json, html). |

**4.3. Test Setup Files (Môi trường ảo)**

Phần này tạo ra các file thiết lập cần thiết để cô lập môi trường test, đảm bảo tính **độc lập** và **tái tạo** của các bài kiểm thử.

**Backend Test Setup (BackEnd/tests/setup.js)**

File này thiết lập **Database ảo cho Backend Tests**:

* Sử dụng **mongodb-memory-server** để tạo một instance MongoDB tạm thời, chỉ chạy trong bộ nhớ (RAM).
* Sử dụng các hooks beforeAll, afterAll, và afterEach để:
  + **Khởi tạo** kết nối database trước tất cả các test (beforeAll).
  + **Đóng và dừng** database sau tất cả các test (afterAll).
  + **Xóa sạch** dữ liệu trong tất cả các collection sau *mỗi* test (afterEach), đảm bảo các test độc lập với nhau.

**Frontend Test Setup (FrontEnd/src/tests/setup.js)**

File này thiết lập **môi trường trình duyệt ảo** cho Frontend Tests:

* Chứa các **mocks** cho các API trình duyệt không có sẵn trong môi trường Node.js/JSDOM.
* **window.matchMedia**: Mock cho các truy vấn media (thường được dùng bởi các thư viện UI như Ant Design, Material UI).
* **global.IntersectionObserver**: Mock cho API theo dõi khả năng hiển thị của phần tử (thường dùng cho lazy loading).
* import '@testing-library/jest-dom' giúp mở rộng các hàm expect trong Vitest để dễ dàng kiểm tra DOM (ví dụ: toBeInTheDocument()).

**5.Setup github**

**6. Docker Development Environment**

**6.1 Start Development Environment (Sử Dụng Docker)**

Phần này tập trung vào việc khởi động toàn bộ kiến trúc (Backend, Frontend, Database) bằng Docker Compose, mô phỏng gần nhất với môi trường Production/Staging

| **Lệnh** | **Mục đích** | **Lợi ích** |
| --- | --- | --- |
| docker-compose up -d | **Khởi động tất cả dịch vụ** (Backend, Frontend, MongoDB). | Chạy tất cả các dịch vụ trong các container độc lập ở chế độ nền (-d). |
| docker-compose ps | **Kiểm tra trạng thái dịch vụ**. | Xác minh rằng tất cả các container đều đang ở trạng thái Up (đang chạy). |
| docker-compose logs -f [service] | **Xem nhật ký (logs)** theo thời gian thực. | Giúp bạn theo dõi quá trình khởi động và gỡ lỗi (debug) cho từng dịch vụ (backend hoặc frontend). |

**6.2 Development Workflow (Làm Việc Hàng Ngày)**

Phần này cung cấp các tùy chọn để chạy ứng dụng và các bài kiểm thử, phù hợp cho quy trình làm việc hàng ngày của lập trình viên.

Khởi chạy Server Phát Triển

| **Lệnh** | **Mục đích** | **Khi nào sử dụng** |
| --- | --- | --- |
| cd BackEnd && npm run dev & | Khởi động **Backend server cục bộ**. | Khi bạn muốn chỉnh sửa code và thấy kết quả thay đổi ngay lập tức (thường có Hot Reload). Yêu cầu phải có MongoDB Docker (hoặc dịch vụ database khác) đang chạy. |
| cd FrontEnd && npm run dev & | Khởi động **Frontend server cục bộ**. | Tương tự, để phát triển giao diện người dùng với Hot Module Replacement. |
| docker-compose up | **Khởi động mọi thứ bằng Docker**. | Thay thế cho các lệnh npm run dev cục bộ nếu bạn muốn cả Backend và Frontend chạy bên trong container, mô phỏng môi trường giống CI/CD hơn. |

Chạy Kiểm Thử (Testing)

| **Lệnh** | **Mục đích** | **Vai trò** |
| --- | --- | --- |
| cd BackEnd && npm test | **Chạy tất cả Unit/Integration Tests** cho Backend. | Đảm bảo logic và API Backend hoạt động chính xác trước khi commit code. |
| cd FrontEnd && npm test | **Chạy tất cả Unit/Integration Tests** cho Frontend. | Kiểm tra các components và logic phía Client. |
| npm run test:watch | **Chạy test Backend ở chế độ Watch**. | Tự động chạy lại các bài test liên quan mỗi khi bạn lưu file, giúp phát triển dựa trên test (TDD). |
| npm run test:ui | **Chạy test Frontend ở chế độ UI**. | Cung cấp giao diện đồ họa (web interface) để xem kết quả test, debug dễ dàng hơn. |