# Tests

This directory contains tests for both the API (server) and client (frontend) components of the Online Examination System. It also includes a simple load test script. Follow the instructions below to install dependencies and run the tests.

## Structure

tests/  
├── api/ # C# unit tests for the ASP.NET Core API  
│ ├── AuthControllerTests.cs  
│ ├── ExamsControllerTests.cs  
│ ├── ResultsControllerTests.cs  
│ ├── GradingServiceTests.cs  
│ ├── MonitoringHubTests.cs  
│ └── TestFixtures/  
│ └── TestDbContextFactory.cs  
├── client/ # TypeScript/React tests for the front‑end  
│ ├── components/  
│ │ ├── Timer.test.tsx  
│ │ ├── QuestionCard.test.tsx  
│ │ └── ResultTable.test.tsx  
│ ├── hooks/  
│ │ └── useTimer.test.ts  
│ └── pages/  
│ ├── LoginPage.test.tsx  
│ ├── ExamListPage.test.tsx  
│ └── ExamRoomPage.test.tsx  
├── load/ # Load testing scripts  
│ └── load\_test.js  
└── README.md

## Prerequisites

* **API tests** – Require the .NET 7 SDK. Install from [Microsoft .NET downloads](https://dotnet.microsoft.com/download). The API test project relies on the API code under api/ and uses the [InMemory provider](https://learn.microsoft.com/en-us/ef/core/miscellaneous/testing/in-memory) for quick setup. Run these tests from the repository root using dotnet test.
* **Client tests** – Use [Jest](https://jestjs.io/) and [React Testing Library](https://testing-library.com/). Install dev dependencies:
* cd client  
  npm install --save-dev jest @testing-library/react @testing-library/user-event @testing-library/react-hooks jest-environment-jsdom @testing-library/jest-dom
* Then run the tests with npm test. You may need to configure Jest and TypeScript support; refer to the official docs.
* **Load tests** – Use [k6](https://k6.io/) CLI to run the load\_test.js. Install k6 and run the script via:
* k6 run tests/load/load\_test.js --env API\_BASE\_URL=http://localhost:5000/api

## Running API Tests

From the root of the repository:

dotnet test

The test project will build the API and execute the unit tests defined in tests/api/. The TestDbContextFactory uses the in‑memory provider to isolate tests.

## Running Client Tests

Install the required dev dependencies as described above, then run:

cd client  
npm test

This will execute the tests located under tests/client/ using Jest. If you prefer Vite’s native testing with [Vitest](https://vitest.dev/), you can adapt the test configuration accordingly.

## Running Load Tests

Ensure the API is running locally (e.g., via docker-compose up) and then execute the load test script with k6:

k6 run tests/load/load\_test.js --env API\_BASE\_URL=http://localhost:5000/api

The script performs a simple login request followed by fetching the list of exams. Customize the payloads and endpoints to suit your performance testing goals.

## Notes

* The tests provided here are intentionally minimal and serve as examples of how to set up testing infrastructure. Extend them with comprehensive assertions, mock services and integration scenarios as your project evolves.
* To keep the repository free of secrets, place any sensitive configuration values (e.g., JWT secrets, database credentials) in an .env file based on the provided .env.example.