rubriks

GIT
Criteria
Commit messages The messages of your commits need to be descriptive and follow the conventions that are described in https://www.conventionalcommits.org/en/v1.0.0/
Branch usage Branches need to be used following the git flow methodology, using feature branches, bugfix branches and release branches. This in order to allow for CI/CD
Clean history Use rebasing in order to create a clean history, so earlier messages that are not up to conventions are not left in the project

Conventions
Criteria
Naming All of the files need to be named correctly and descriptively. The name of the file needs to indicate what it is for.
Folders The project needs to be build up in a consistent and logical fashion
Separation of concerns Files and functions should be separated by their functions

Documentation	
Criteria	
Function documentation The entire project should be accurately documented, including params and returns	
endpoints a full documentation needs to be available regarding all the possible endpoints in the system. This includes body, and returns	

Open source	
Criteria	
README The readme should contain all the necessary elements to describe the project, including information to run it and history	
License and other documents The project should contain a code of conduct, an up to date changelog containing the releases, a contribution guideline and a license file	

Docker
Criteria
Build and run The project should be build-ready on cloning the repository, with clear instructions regarding the .env file and others in the README. The entire project should run upon following these instructions, without any installation
Dockerised Every service should run in a docker container, with descriptive names. This includes database systems and storing mechanisms
Deployed [extra] The docker setup should be deployed on the school servers, and published on the docker hub

Testing
Criteria
Test driven development The project should be created using the TDD guidelines, where tests are defined before development is started, which is visible in the git history
End to end tests The project should contain one part of an end to end tests, which shows that you understand complete testing of data retention and deletion across the project
Unit and integration tests The project should contain unit tests for each function, and integration tests for the endpoints and functionalities. I need to see that you understand all the principles

CI/CD
Criteria
Deploy on PR Make sure the project deploys when a merge or PR to main (or development) is triggered
Container management Use the docker hub to keep your containers on the server up to date, including the build frontend, or run all the tests before pushing a container to the hub.
testing / production Make sure there is a different setup for the production and development branch

[o] Threejs and shaders
Criteria
Frontend Create a frontend using threejs to visualise an interactive scene
Shaders Use shaders to create textures on interactive objects
Performance Make sure the site runs without too much use of resources. Don't load what isn't necessary etc.

[o] Communication protocols
Criteria
Use of communication The project needs to communicate with external hard- or software, using relevant communication protocols. Use a separate docker service to achieve this.
Clean entry and exit The service needs to cleanly exit, and initiate connections. Either on demand or on startup. Make sure the connection resets when something goes wrong.
Data transmission practices Make sure the transmitted data contains exactly what is needed, with regards to performance and overview

[o] React and webpack
Criteria
Built frontend The frontend needs to be compiled and build using webpack. It needs to run in a docker container, and start with docker-compose
React The frontend needs to be made in a modular fashion, using correct and up-to-date functions like useEffect and state management
Best practices The frontend needs to adhere to the best practices regarding style and performance. Use libraries to help you