Software Integration Project

Given the attached project (app.zip), your tasks are the following **(in this order)**:

- Set it up as a **GitHub** repository
- Add Eslint and Prettier
- Add Husky: (all commits in your project should follow this convention)
    - Make sure to have a hook that checks if there's any TS or lint errors whenever you commit (This hook should also run prettier to properly format the code)
    - Ensure that commits follow the Conventional Commits specification
        - format: <type>: <description> (NOTE: no need to apply a scope)
        - recommended types: ['build', 'chore', 'ci', 'docs', 'feat', 'fix', 'perf', 'refactor', 'revert', 'style', 'test']
- Set up 4 different environments (dev, release, pprod & prod, with 4 mongoDB clusters and 4 PostgreSQL DBs for each)
- Convert **all** Javascript code to Typescript
- Add Unit tests using **Jest** for **all** services, models and middlewares (success & **all** error cases)
- Add Integration tests with **Supertest** for all the routes present in the application (success & **all** error cases)
- Add CI/CD pipelines at every step (you need to have **1 workflow for PRs** & **1 workflow for Deployments**)
    - Testing, building, and linting should be checked at **every pull request**
    - Testing, building, and linting should be re-checked at **every push** + **deployment**.

Deliverables:

- Only 1 submission per team
- A Zip of your project + github repository url
- A postman collection with all the routes + 4 different environments with the different variables
    - Routes should have saved examples for all possible use cases
- A recorded live demo with the different members of the team explaining what they worked on
    - Each member should set up at least one environment -> DBs + pipelines + hosted server. **Video should not exceed 5mins**.
    - Each member should work on at least on 1 controller + route + model + middleware

- One Member should oversee the demonstration of husky, linter, pipelines and that the project is hosted (by making at least 1 request)

NOTES:

- Initial code base should not be changed
- All merges should be done via **pull requests**
- Branches created should be properly named and should follow the following format:
  - <type>/<description>
  - recommended types: ['build', 'chore', 'ci', 'docs', 'feat', 'fix', 'perf', 'refactor', 'revert', 'style', 'test']

- The compilerOptions in tsconfig.json and the rules of your eslint should be exactly what you have below. (You may add more rules to make it stricter but not less)

- CI/CD pipelines should be done using Github Actions
- Only EC2 instances should be used

- How to add the code to Github ? You could choose one of these approaches:

  - Add the code feature by feature (in JS or TS)

  or

  - Add the code all at once in JS, then proceed with converting it to typescript

Any submission past the deadline will not be accepted.

Typescript:

Inside the **compilerOptions** of your **tsconfig.json** you should have the following

```
{
    "incremental":false,
    "target":"es2016",
    "module":"commonjs",
    "rootDir":"./",
    "baseUrl":"./",
    "resolveJsonModule":true,
```

```
    "declaration":true,
    "sourceMap":true,
    "outDir":"dist",
    "removeComments":true,
    "allowSyntheticDefaultImports":true,
    "esModuleInterop":true,
    "forceConsistentCasingInFileNames":true,
    "noImplicitAny":true,
    "noUnusedLocals":true,
    "noUnusedParameters":true,
    "noFallthroughCasesInSwitch":true,
    "skipLibCheck":true
  }
```

Eslintrc:

The rules to add inside of your eslintrc

```
  "rules":{
    "@typescript-eslint/interface-name-prefix":"off",
    "@typescript-eslint/explicit-function-return-type":"error",
    "@typescript-eslint/explicit-module-boundary-types":"off",
    "@typescript-eslint/no-explicit-any":"warn",
    "@typescript-eslint/no-unused-vars":"error",
    "no-console":"warn",
    "no-restricted-syntax":[
      "error",
      "ForInStatement",
      "LabeledStatement",
      "WithStatement"
    ]
  }
```