# Project - Hotel

## 1 - Project overview

Akkor Hotel ltd is a company which aim to provide you with the best experience to book an hotel everywhere on the globe. You need to be able to handle all tasks related to finding the place, booking it, pay online and reschedule if necessary.

You also need to provide a good looking, UX focused interface which works both on desktop but also mobile.

This project can be done in a group of 2 people. **You are free to use the frontend/backend stack you want**

You will have to defend it in a 15 min presentation.

## 2 - Features

You need to develop multiples functionalities

- User management solution
    - Create, Read, Update, Delete user
    - User is at least {id, email, pseudo, password, role}
    - Normal users cannot read information about another user - but an employee can check information
    - You can create a new user even without being logged
    - You can only update yourself (other users cannot update you EXCEPT if admin)
    - You can only delete yourself (other users cannot delete you)
    - login/logout solution
- Authentication need to be setup
    - Different solutions can be used, jwt token is recommended (see tips for more information)
    - All Read endpoints of hotel data are non-logged/anonymous
    - All Write endpoints need the request to be authenticated (stateless)
- A hotel endpoint
    - List all hotel and allow you to sort by date, name, location with a limit (default limit is 10 but can be changed with a parameter)
    - Create, Read, Update, Delete hotel
    - Hotel is at least {id, name, location, description, and picture_list}

    - Only an admin can create, update and delete an hotel
- A Booking endpoint

    - Create, Read, Update, Delete booking
    - Only a logged user can create, update or delete a booking
    - A logged user only see his own booking
    - Admin should be able to find your booking with the ID (or your name/email)
- It is important to provide good feedback to the users using the API so you will need to implement a simple type solution/validation
- On the same note, you need to use the valid HTTP code when returning information to the user

You are also required to provide a full testing suite both for the frontend and backend

- You are expected to test all the feature in your code

    - Handle unit testing of the functions
    - Work around integration/functional testing
    - Validate the core user tunnel with e2e tests.
- For the test, it is expected to both handle the normal use case, but also edge cases
    - For example, e2e tests should try to register with a normal/valid email but also with a wrong email and check that it displays an error
    - For unit tests, you should test how your database is working if you insert an amount lower than 0 and what's the result.
- It is also expected to provide a basic GitHub pipeline for the CI/CD part
    - Pull Requests need to be reviewed by at least one person and not open comments to be able to merge. This should also trigger a pipeline to run all tests and allow the merge only if everything is green
    - When merging to the main branch it should trigger a full pipeline with at least
        - run all tests
        - check for security
        - Build the solution
        - "Deploy" the solution (no need to really deploy just have a job with an "echo")

## 3 - Tips

Authentication:

- You do not need a full OAuth solution, just a way to generate a token
- You can use PassportJS and the strategy Passport-local-mongoose
- To test a jwt token you can use jwt.io

Validation/type solution

- Joi, Yup and AJV are working almost in a same way (Node.js)
- Only test the users input
- It can be interesting to build a middleware for that !

Documentation

- The documentation can be based on swagger/openapi standard
- For an easy writing your can use editor.swagger.io
- Try to define the endpoint with both the input value (url, body…) and the possibles returned values (success, error…)

## 4 - Deliverables

You need to submit an archive containing the source code of your project and the design of the API done with swagger/openAPI (or other standard solution - in that case precise which one).

You also need to invite your professor to your private repository (GitHub or other solution your choose)

With a readme for the installation steps and to be able to run your project: **if the project is not runnable a 0 grading can be attributed!**

The end date is the March 5th at 23h59 - with an oral defense shortly after

## 5 - Graded items

Frontend: 2pts

Backend: 2pts

Testing:

- User management: 5pts  (2pts frontend - 3pts backend)

- Hotel endpoint: 5pts  (3pts frontend - 2pts backend)
- Booking: 4pts (2pts frontend - 2pts backend)

CI/CD pipeline: 2pts (+2pts bonus if everything work perfectly)

Not providing a documentation  result on negative points (up to -5)

**Up to 50% of points for each category can be lost to code quality and/or security. You have to work as you would do it in a company with impact for customers or finance !**