Debugging Cheat Sheet

What is wrong with my code? If you are getting no results, a different result or a test is not going the way you intended, check the following:

Resetting the database after every test (or code change in the seed.js or schema.prisma)

Reset the database completely: npx prisma db push --force-reset
Fill the database: npx prisma db seed
Run the application: npm run dev

Application should be running

Can be seen by the yellow and green text

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/index.js`
Server is listening on port 3000
```

Running the tests

```
❏ users
↳ Create a new user
  POST http://0.0.0.0:3000/login [200 OK, 432B, 27ms]
  POST http://0.0.0.0:3000/users [500 Internal Server Error, 289B, 167ms]
  3. Response status code is 400
```

Search this text in the 'Bookings API.json' or 'Bookings API_negative.json file which data is being used

If the tests fail, copy the 'Create a new user' and search for it in the files, most likely a file called 'Bookings API.json' will open now.

Copy the data from the test over in Postman and see if you can get any result there

Does the user you want to GET or DELETE exist in the database? Try running a GET on http://0.0.0.0:3000/users first to see if the user is actually there.

Read the original terminal The terminal where you've run the 'npm run dev' command, check if there are any error messages

Example of an error

```
info: POST /users. status: 200. Duration: 1ms {"service":"booking-api"}
PrismaClientValidationError:
Invalid `prisma.user.create()` invocation:

{
  data: {
        password: undefined,
        name: undefined,
        email: undefined,
        phonenumber: undefined,
        pictureURL: undefined,
+   username: String
  }
}

Argument `username` is missing.
        at Cn (node_modules\@prisma\client\runtime\library.js:116:5888)
        at _n.handleRequestError
(node_modules\@prisma\client\runtime\library.js:123:6510)
        at _n.handleAndLogRequestError
(node_modules\@prisma\client\runtime\library.js:123:6188)
        at _n.request (node_modules\@prisma\client\runtime\library.js:123:5896)
        at async l (node_modules\@prisma\client\runtime\library.js:128:10871)
        at async \src\routes\users.js:44:18 {
  clientVersion: '5.10.2'
}
```

How to read these errors

Ignore everything related to the 'node_modules' text, and only focus on the other lines. The most important ones:

prisma.user.create

Search for this in your files to see where you are calling this

password: undefined,
It seems that there is no data being provided at all in this create user part

Argument `username` is missing.
Is the username being provided by the test? Is this a negative test that does not provide a username? Try posting data yourself with Postman

at async \src\routes\users.js:44:18
The error occurs on line 44 of the file src\routes\users.js

Other errors

Check the 'file order'

1. index.js
2. src\routes\properties.js
3. src\services\properties\getPropertyById.js

(Temporarily) remove the login from the router

router.put("/:id", auth, async (req, res, next) => {   try {

Change this into: router.put("/:id", async (req, res, next) => {   try {

Not getting any error messages at all?

Disable each line of Sentry in the 'index.js' file (but keep the other entries containing 'log')

Constraint errors

Use onDelete and onUpdate (see Prisma documentation)

Use a lot of console.log

Check which values you are working with, and where, is the value what you expect it to be?

```
router.post("/", auth, async (req, res, next) => {
    try {
          const { username, password, name, email, phonenumber, pictureURL } =
              req.body;
              console.log("req.body:",req.body)
              console.log("username:",username)
              // Add more entries

          const user = await createUser(
              username,
              password,
              name email,
              phonenumber,
              pictureURL
          );
          res.status(201).json(user);
 } catch (error) {
   next(error);
 }
});
```