

BÁO CÁO DẶN - TUẦN 8

Họ và Tên: Lý Chí Công

MSV: B22DCCN088

Bộ môn: Thực tập cơ sở

1. Thông tin dự án:

Tên dự án: Hệ thống quản lý sân bóng

Công nghệ sử dụng: ReactJS (frontend), Node.js + Express.js (backend), MongoDB (database)

Mục tiêu chung: Xây dựng hệ thống quản lý sân bóng, bao gồm đặt sân, thanh toán, quản lý lịch sử sử dụng.

2. Mục tiêu tuần 8

- Hoàn thiện các API cơ bản cho các đối tượng: users, bookings, fields, field_status.
 - Tích hợp hệ thống xác thực và phân quyền người dùng bằng **JWT**.
 - Cấu hình **CORS** cho phép frontend truy cập.
 - Tìm hiểu cách triển khai xác thực qua email và tích hợp cổng thanh toán.
-

3. Kết quả đạt được

API cơ bản đã hoàn thiện

- **Users:** Đăng ký, đăng nhập, cập nhật thông tin, tìm kiếm, xóa.

```
Source code > Web-sanbong > BE > src > routes > JS authroutes.js > ...
4  const authValidation = require("../validations/auth.validation")
5
6  const router = express.Router()
7
8  /**
9   * @route POST /api/auth/register
10  * @desc Register a new user
11  * @access Public
12  */
13  router.post("/register", validate(authValidation.register), authController.register)
14
15  /**
16  * @route POST /api/auth/login
17  * @desc Login user
18  * @access Public
19  */
20  router.post("/login", validate(authValidation.login), authController.login)
21
22  /**
23  * @route POST /api/auth/logout
24  * @desc Logout user
25  * @access Public
26  */
27  router.post("/logout", validate(authValidation.logout), authController.logout)
28
29  /**
30  * @route POST /api/auth/refresh-tokens
31  * @desc Refresh auth tokens
32  * @access Public
33  */
34  router.post("/refresh-tokens", validate(authValidation.refreshTokens), authController.refreshTokens)
35
36  module.exports = router
37
```

Fields: Tạo, cập nhật, xóa, lấy danh sách sân.

```
6  const constants = require("../config/constants")
7
8  const router = express.Router()
9
10  /**
11  * @route GET /api/fields
12  * @desc Get all fields
13  * @access Public
14  */
15  router.get("/", fieldController.getFields)
16
17  /**
18  * @route GET /api/fields/:fieldId
19  * @desc Get field by ID
20  * @access Public
21  */
22  router.get("/:fieldId", validate(fieldValidation.getFieldById), fieldController.getFieldById)
23
24  /**
25  * @route POST /api/fields
26  * @desc Create a new field
27  * @access Private (Admin only)
28  */
29  router.post(
30    "/",
31    authenticate,
32    authorize(constants.roles.ADMIN),
33    validate(fieldValidation.createField),
34    fieldController.createField,
35  )
36
37  /**
38  * @route PUT /api/fields/:fieldId
39  * @desc Update field
40  * @access Private (Admin only)
41  */
```

Field status: Theo dõi trạng thái sân theo ngày/giờ.

```

9
10 /**
11  * @route GET /api/field-status/date/:date
12  * @desc Get field status by date
13  * @access Public
14  */
15 router.get(
16   "/date/:date",
17   validate(fieldStatusValidation.getFieldStatusByDate),
18   fieldStatusController.getFieldStatusByDate,
19 )
20
21 /**
22  * @route GET /api/field-status/:fieldId/date/:date
23  * @desc Get field status by field ID and date
24  * @access Public
25  */
26 router.get(
27   "/:fieldId/date/:date",
28   validate(fieldStatusValidation.getFieldStatusByFieldAndDate),
29   fieldStatusController.getFieldStatusByFieldAndDate,
30 )
31
32 /**
33  * @route PUT /api/field-status/:fieldId/date/:date
34  * @desc Create or update field status
35  * @access Private (Admin/Manager)
36  */
37 router.put(
38   "/:fieldId/date/:date",
39   authenticate,
40   authorize(constants.roles.ADMIN, constants.roles.MANAGER),
41   validate(fieldStatusValidation.createOrUpdateFieldStatus),
42   fieldStatusController.createOrUpdateFieldStatus,
43 )
44

```

Bookings: Đặt sân, cập nhật lịch đặt, hủy đặt sân.

```

10 /**
11  * @route GET /api/bookings
12  * @desc Get all bookings
13  * @access Private (Admin/Manager)
14  */
15 router.get("/", authenticate, authorize(constants.roles.ADMIN, constants.roles.MANAGER), bookingController.getBookings)
16
17 /**
18  * @route GET /api/bookings/:bookingId
19  * @desc Get booking by ID
20  * @access Private
21  */
22 router.get("/:bookingId", authenticate, validate(bookingValidation.getBookingById), bookingController.getBookingById)
23
24 /**
25  * @route GET /api/bookings/user/:userId
26  * @desc Get bookings by user ID
27  * @access Private
28  */
29 router.get(
30   "/user/:userId",
31   authenticate,
32   validate(bookingValidation.getBookingsByUser),
33   bookingController.getBookingsByUser,
34 )
35
36 /**
37  * @route POST /api/bookings
38  * @desc Create a new booking
39  * @access Private
40  */
41 router.post("/", authenticate, validate(bookingValidation.createBooking), bookingController.createBooking)
42
43 /**
44  * @route PUT /api/bookings/:bookingId
45  * @desc Update booking

```

Phân quyền bằng JWT:

- Đã triển khai middleware xác thực và phân quyền người dùng (user, admin).
- Các route quan trọng chỉ cho phép truy cập khi có token hợp lệ.

```
const authenticate = async (req, res, next) => {
  try {
    const authHeader = req.headers.authorization

    if (!authHeader || !authHeader.startsWith("Bearer ")) {
      return apiResponse(res, 401, "Authentication required")
    }

    const token = authHeader.split(" ")[1]

    if (!token) {
      return apiResponse(res, 401, "Authentication required")
    }

    try {
      const decoded = jwt.verify(token, config.jwt.secret)

      const user = await userService.getUserById(decoded.sub)

      if (!user) {
        return apiResponse(res, 401, "Invalid token")
      }

      req.user = user
      next()
    } catch (error) {
      return apiResponse(res, 401, "Invalid token")
    }
  } catch (error) {
    next(error)
  }
}
```

Chưa hoàn thành

Xác thực người dùng qua Email (Mail Service): Chưa triển khai tính năng gửi email xác nhận hoặc đặt lại mật khẩu.

Tích hợp API thanh toán: Chưa viết được các API xử lý thanh toán qua ví điện tử hoặc ngân hàng.

4. Khó khăn và hướng giải quyết

Khó khăn	Hướng giải quyết
Thiết kế API thanh toán đòi hỏi tích hợp với bên thứ ba (e.g. Momo, ZaloPay)	Tạm hoãn, dự kiến sẽ xử lý sau khi hoàn thiện chức năng đặt sân
Chưa rõ cách gửi email xác nhận qua Node.js	Dự kiến dùng thư viện nodemailer và thiết lập tài khoản Gmail hoặc Mailtrap để kiểm thử
JWT cần xử lý nhiều trường hợp lỗi (hết hạn, token sai)	Đã tạo middleware xử lý token, sẽ tiếp tục tối ưu logic phản hồi lỗi

5. Mục tiêu tuần 9

- Triển khai xác thực qua email sử dụng nodemailer.
- Viết và hoàn thiện các API liên quan đến **thanh toán đơn giản** (giả lập).
- Kiểm thử toàn diện hệ thống booking: kiểm tra trùng lịch, trạng thái sân.
- Kết nối giao diện React với các API backend đã viết.
- Bổ sung phân trang và tìm kiếm nâng cao cho danh sách người dùng/sân.