MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Hiểu và cài đặt Authentication
- ✓ Đăng ký, đăng nhập và mã hoá password
- ✓ Authentication
- ✓ Validation

PHẦN I

Bài 1 (3 điểm)

Viết RESTFUL API thực hiện đăng ký thành viên

Bước 1: Tạo model user

```
const Sequelize = require('sequelize');

const sequelize = require('../util/database');

const User = sequelize.define('tblUsers', {
  id: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    allowNull: false,
    primaryKey: true
  },
  email: {
    type: Sequelize.STRING,
    required: true
  },
  password: {
    type: Sequelize.STRING,
    required: true
  },
  typeUser: {
    type: Sequelize.INTEGER,
    required: true
  }
},
{ timestamps: false }
```

```
);
module.exports = User;
```

Bước 2: Xây dựng controller thực hiện thêm user

```
onst User = require('../models/user');
const bcrypt = require('bcryptjs');
const jwt = require("jsonwebtoken");

exports.createUser = (req, res, next) => {
  const email = req.body.email;
  const password = req.body.password;
  const confirmPassword = req.body.confirmPassword;
  const typeUser = req.body.typeUser;
  console.log(email);
  User.findOne({where: { email: email }})
  .then(user => {
    if (user) {
      console.log(user.email);
      return res.status(400).json({message:"Email da ton tai"});
    }
    return bcrypt.hash(password, 12);
    })
  .then(hashedPassword=>{
    const user = new User({ email: email, password: hashedPassword,typeUser:typeUser });
    return user.save();
  })
  .then(user => {
      res.status(201).json({
        message: 'Thêm thành công thành viên!',
        user: user
      });
    })
  .catch(err => res.status(400).json(err))
};
```

Bước 3: Thực hiện route

```
const express = require('express');

const userController = require('../controllers/auth');

const router = express.Router();
// POST /blog/post
router.post('/register', userController.createUser);
router.post('/login/', userController.login);
```

```
module.exports = router;
```

Bước 4: Test API với Postman

Bài 2 (2 điểm)

Viết REST API xử lý đăng nhập và trả về token

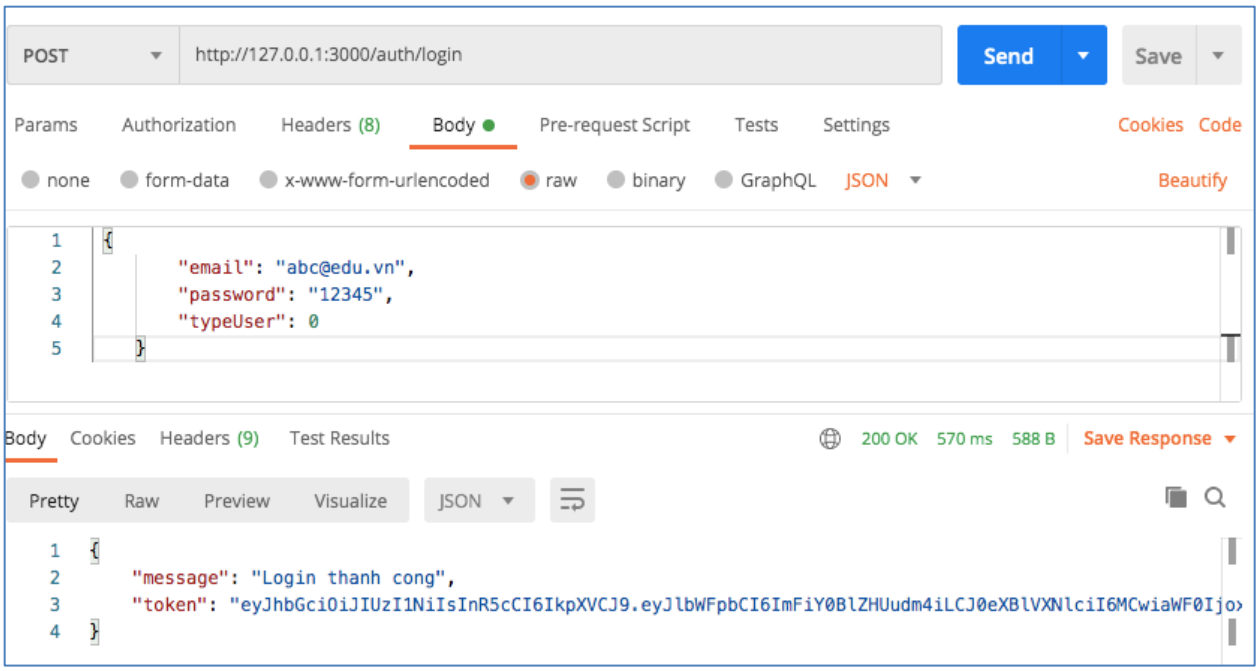
Bước 1: Viết API

```
exports.login = (req, res, next) => {
  const email = req.body.email;
  const password = req.body.password;

  User.findOne({where: { email: email }})
  .then(user => {
    if (!user) {
      return res.status(400).json({message:"Email khong ton tai"});
    }
    return Promise.all([bcrypt.compare(password, user.password),user]);
    })
  .then(result=>{
    const isMatch=result[0];
    const user=result[1];

    if(!isMatch) return res.status(400).json({message:"Password khong khop"})
    const payload={
      email:user.email,
      typeUser:user.typeUser
    }
    console.log(payload);
    return jwt.sign(payload,"FptPolyTechnic",{expiresIn:3600})
  })
  .then(token=>{
    console.log(token);
    res.status(200).json({message:"Login thanh cong",token})
  })
  .catch(err => res.status(400).json(err))
};
```

Bước 2: Test với phương thức POST với postman



PHẦN II

Bài 3 (2 điểm)

Xây dựng RESTFUL API Authenticate

Bước 1: Tạo middleWare authen

Lab07_api > middleware > JS auth.js > ...

```
const jwt = require("jsonwebtoken");
exports.authenticate = (req, res, next) => {
  const token=req.header('token');
  if(!token) return res.status(401).json({message:"Chưa được phép truy cập, cho cung cấp token"});
  jwt.verify(token,"FptPolyTechnic")
  .then(decoded=>{
      req.user=decoded;
      next();
  })
};
```

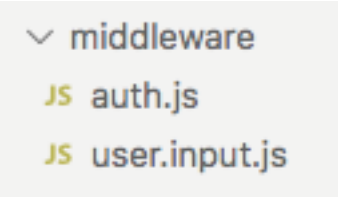
Bước 2: sử dụng middleware trên cho route

```
router.get('/private/',authenticate, userController.testAuth);
```

Bước 3: Test với postman

Bài 4 (2 điểm)

Xây dựng middleware kiểm tra tính hợp lệ dữ liệu nhập vào khi thực hiện đăng ký user.




```
Lab07_api > middleware > JS user.input.js > ...
  1   const lad=require('lodash');
  2   const validator=require('validator');
  3
  4   const {User}=require('../models/user');
  5   exports.checkInput = async (req, res, next) => {
  6       let errors={};
  7       const email =lad.get(req.body,"email","");
  8       const password =lad.get(req.body,"password","");
  9       const password2 = lad.get(req.body,"confirmPassword","");
 10       const typeUser = lad.get(req.body,"typeUser","");
 11       if(validator.isEmpty(email))
 12           errors.email="Phải nhập Email";
 13       if(lad.isEmpty(errors)) return next();
 14       return res.status(400).json(errors)
 15   }
```

- Sử dụng validate trong route

```
// POST /blog/post
router.post('/register',checkInput, userController.createUser);
router.post('/login/', userController.login);
router.get('/private/',authenticate, userController.testAuth);
```

Bài 5 (1 điểm)

Test REST API: sinh viên thực hiện gọi tất cả các API đã thực hiện ở bài 3,4 với front end.

*** Yêu cầu nộp bài:

SV nén file (*hoặc share thư mục google drive*) bao gồm các yêu cầu đã thực hiện trên, nộp LMS đúng thời gian quy định của giảng viên. KHÔNG NỘP BÀI COI NHƯ KHÔNG CÓ ĐIỂM.

---Hết---