

I. Group Information:

MSSV	Họ tên	Email liên lạc
20127003	Hoàng Quốc Bảo	20127003@student.hcmus.edu.vn
20127531	Trương Trọng Khánh	20127531@student.hcmus.edu.vn

II. Feature description:

A. Login:

1. Backend:

- Nhận email, password qua request body. Sau đó sử dụng bcrypt hash password vừa được nhận rồi so sánh với password trong database. Sau đó sử dụng JWT để tạo 1 refresh token mới và lưu lại vào database. Tiếp tục sử dụng JWT để cấp access token mới với nội dung token là id của refresh token mới vừa được tạo trên database và gửi access token này cho người dùng.

```
app.post('/login', async function(req, res, next) {
  let reqData = req.body;

  if (typeof(req.body) === "string") {
    reqData = JSON.parse(req.body);
  }

  const email = reqData['email'];
  const password = reqData['password'];

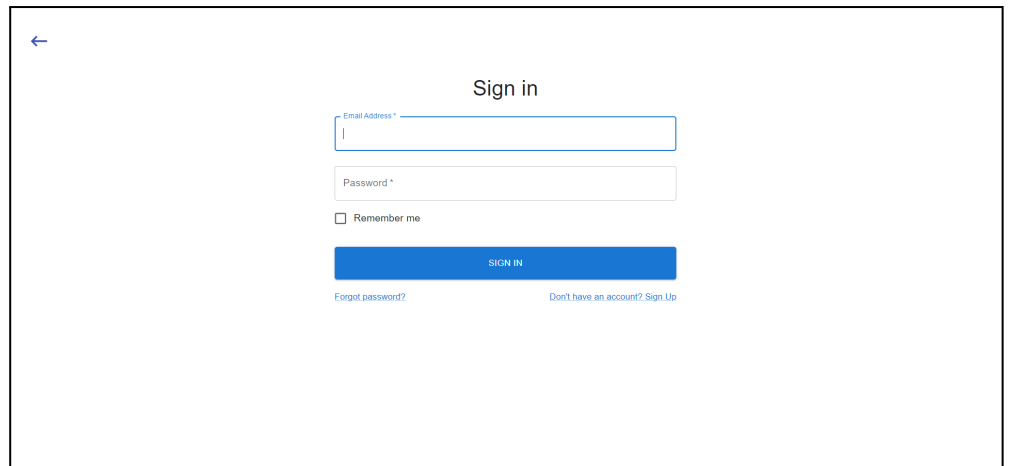
  const id = await authenPassword(email, password);

  if (id) {
    const refreshTokenId = await createRefreshToken(id);
    const accessToken = jwt.sign(
      { "refresh_token_id": refreshTokenId },
      process.env.ACCESS_TOKEN_SECRET_KEY, { expiresIn: '10s' }
    );
    res.json({ 'access_token': accessToken });
  }
  else {
    res.status(200).json({ "message": "Password or email is not correct" });
  }
});
```

```
async function authenPassword(email, password) {
  const sql = "SELECT id, `password` FROM `user` WHERE email = ?";
  const params = [email];
  const result = await databaseQuery(dbConnection, sql, params);

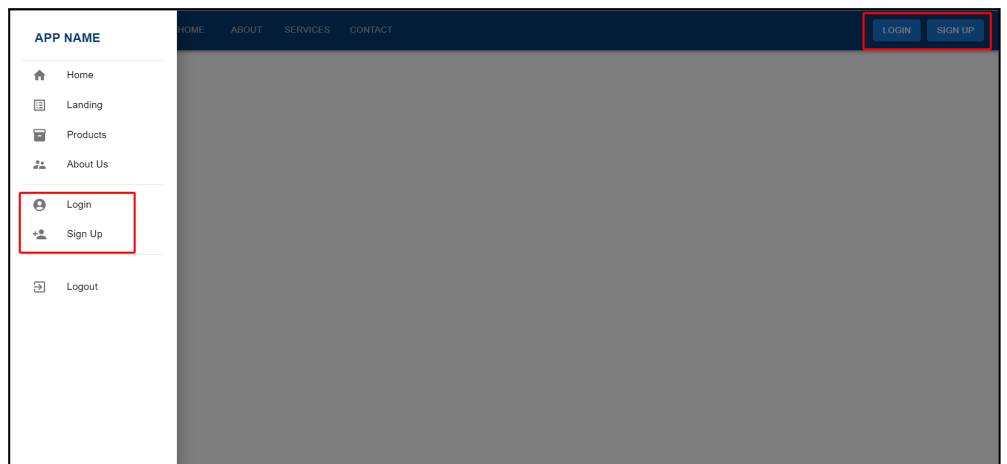
  if (result.length > 0) {
    const hash = result[0]['password'];
    const id = result[0]['id'];
    const res = await bcrypt.compare(password, hash).catch(err => console.log(err.message));
    if (res) return id;
  }
  return null;
}
```

2. Frontend:

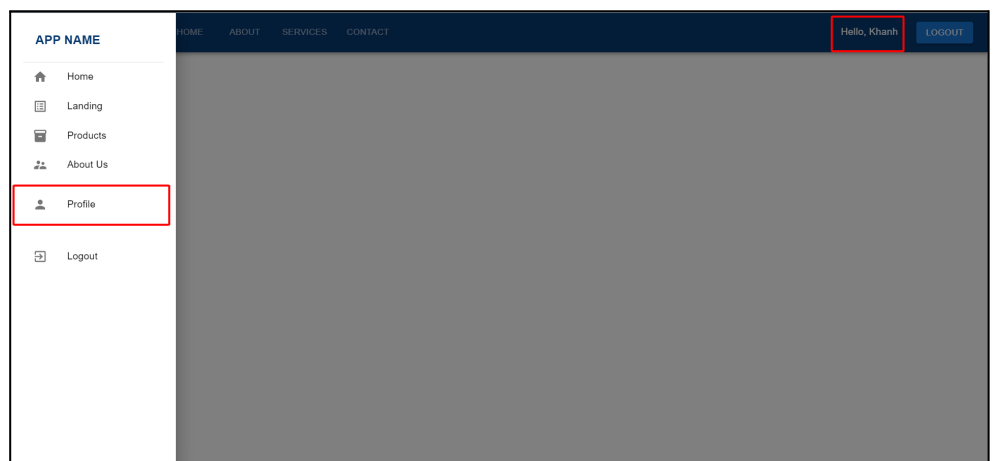


A screenshot of a 'Sign in' form. At the top left is a blue back arrow. The title 'Sign in' is centered. Below it are two input fields: 'Email Address *' and 'Password *'. A 'Remember me' checkbox is below the password field. A blue 'SIGN IN' button is centered below the checkbox. At the bottom, there are two links: 'Forgot password?' on the left and 'Don't have an account? Sign Up' on the right.

- Pre login:



- After login:



B. Logout:

1. Backend:

- Kết nối tới database và đánh dấu lại cột isRevoke của refresh token là true để không thể lấy access token mới từ refresh token này. Truy cứu để refresh token bằng

bằng refresh token id được mã hóa từ access token người dùng gửi về.

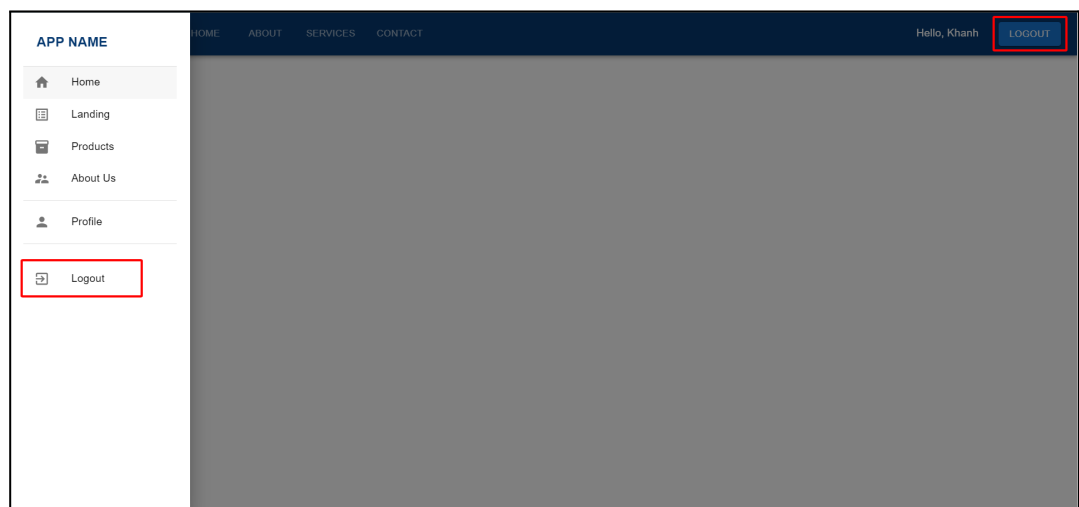
```
app.get('/logout', authToken, async function(req, res, next) {
  const authorizationHeader = req.headers['authorization'];
  const accessToken = authorizationHeader.split(' ')[1];

  jwt.verify(accessToken, process.env.ACCESS_TOKEN_SECRET_KEY, async (err, data) => {
    if (err) {
      res.status(403).json({ "Error": err });
      throw err;
    }
    const refreshTokenId = data['refresh_token_id'];
    const sql = "UPDATE refresh_authen SET is_revoked = 1 WHERE id = ?";
    const params = [refreshTokenId];

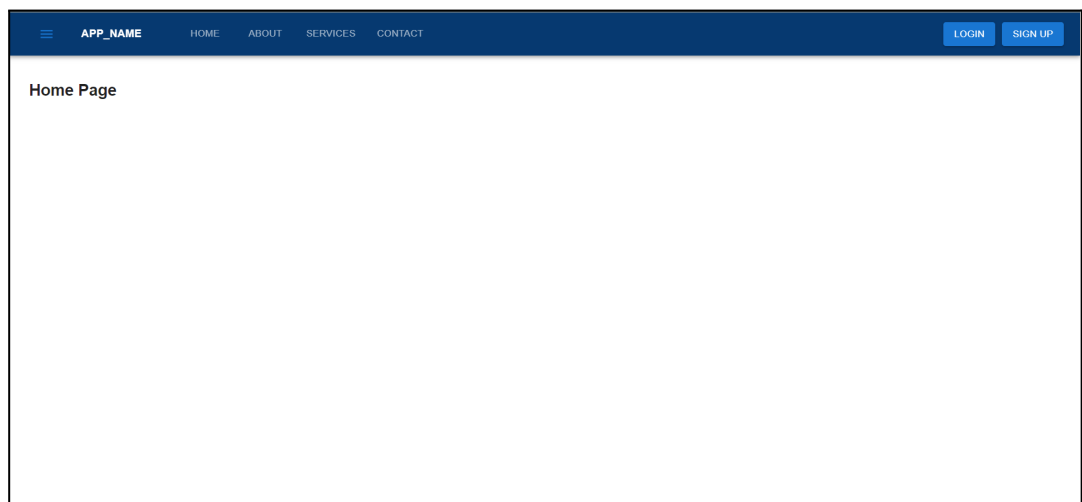
    await databaseQuery(dbConnection, sql, params).catch(err => res.status(500).json({ "Error": err }));

    res.json({ "message": "Logout successfully" });
  });
});
```

2. Frontend:



- After logout:



C. Register:

1. Backend:

- Sau khi kiểm tra access token hợp lệ, lấy thông tin người dùng từ request gửi về. Tạo id ngẫu nhiên cho người

dùng bằng uuidv4 sao đó kết nối đến database và lưu lại thông tin tạo tài khoản mới.

```
app.post('/register', async function(req, res, next) {
  let reqData = req.body;

  if (typeof(req.body) === "string") {
    reqData = JSON.parse(req.body);
  }

  const id = uuidv4();
  const email = reqData['email'];
  const firstName = reqData['first_name'];
  const lastName = reqData['last_name'];
  const password = reqData['password'];
  const phoneNumber = reqData['phone_number'];

  const sql = "SELECT * FROM `user` WHERE email = ?";
  const params = [email];

  const results = await databaseQuery(dbConnection, sql, params).catch(err => {
    console.err(err);
    res.status(500).json({ "message": "Server error" });
  });

  if (results.length > 0) {
    res.status(200).json({ "message": "Email already exist" });
  } else {
    const hashPassword = await bcrypt.hash(password, parseInt(process.env.SALT_ROUNDS)).catch(err => console.error(err.message));
    const sql = "INSERT INTO `user` (id, email, `password`, phone_number, first_name, last_name) VALUES (?, ?, ?, ?, ?, ?)";
    const params = [id, email, hashPassword, phoneNumber, firstName, lastName];
    databaseQuery(dbConnection, sql, params).catch(err => {
    });
    res.status(200).json({ "message": "Create success" });
  }
});
```

2. Frontend:



Sign up

First Name *

Last Name *

Email Address *

Password *

Confirm Password *

Phone Number *

SIGN UP

[Already have an account? Sign in](#)

- Information validate

Sign up

First Name *

Invalid first name

Last Name *

Invalid last name

Email Address *

Invalid email

Password *

8 to 24 characters.
Must include uppercase and lowercase letters, a number and a special character.
Allowed special characters: ! @ # \$ %

Confirm Password *

Passwords do not match

Phone Number *

Invalid phone number

SIGN UP

D. Show Profile:

1. Backend:

- Sau khi kiểm tra access token hợp lệ, lấy id người dùng bằng access token người dùng gửi về (decode access token lấy được refresh token id và truy vấn id người dùng bằng refresh token id trong database). Sau đó lấy id người dùng vừa tìm được để truy vấn thông tin người dùng

```

app.get('/profile', authToken, async function(req, res, next){
  const authorizationHeader = req.headers['authorization'];
  const accessToken = authorizationHeader.split(' ')[1];

  const refreshTokenId = jwt.decode(accessToken)['refresh_token_id'];

  const userIdSql = "SELECT user_id FROM refresh_authen WHERE id = ?";
  const userIdParams = [refreshTokenId];
  > const userId = await databaseQuery(dbConnection, userIdSql, userIdParams).catch(err => {
  });

  if (userId.length > 0) {
    const sql = "SELECT * FROM `user` WHERE id = ?"
    const params = [userId[0]["user_id"]];

    const results = await databaseQuery(dbConnection, sql, params).catch(err => {
      res.status(500).json({ "message": "Server error" });
    });

    res.status(200).json(results[0]);
  }
  > else {
    res.status(200).json({ "message": "Invalid access token" });
  }
});

```

2. Frontend:

APP_NAME

HOME

ABOUT

SERVICES

CONTACT

Hello, Khanh

LOGOUT

User Profile

First Name *

Khanh

Last Name *

Truong Trong

Email Address *

truongtrongkhanhcm@gmail.com

Phone Number *

0834775202

EDIT PROFILE

E. Edit Profile:

1. Backend:

- Lấy id người dùng tương tự với lấy profile. Sau đó sử dụng id người dùng vừa lấy được để truy vấn thay đổi thông tin người dùng đó với các thông tin mà API trả về

```

app.patch('/profile', authToken, async function(req, res, next) {
  let reqData = req.body;

  if (typeof(req.body) === "string") {
    reqData = JSON.parse(req.body);
  }

  const authorizationHeader = req.headers['authorization'];
  const accessToken = authorizationHeader.split(' ')[1];

  const refreshTokenId = jwt.decode(accessToken)['refresh_token_id'];
  const userIdSql = "SELECT user_id FROM refresh_authen WHERE id = ?";
  const userIdParams = [refreshTokenId];

  const userIdQueryResult = await databaseQuery(dbConnection, userIdSql, userIdParams).catch(err => {
    res.status(500).json({ "message": "Server error" });
  });

  const userId = userIdQueryResult[0]['user_id'];

  if (reqData['password']) {
    const sql = "UPDATE `user`"

    let setSql = "SET";

    const whereSql = "WHERE id = ?";

    const params = [userId];

    Object.keys(reqData).forEach(function(key){
      setSql += ` ${key} = "${reqData[key]}"`, `;`;
    });

    setSql = setSql.slice(0, -1);

    const result = await databaseQuery(dbConnection, `${sql} ${setSql} ${whereSql}`, params).catch(err => res.status(500).json({ "Error": err }));

    if (result.changedRows === 1) {
      res.status(200).json({ "messages": "Update user profile successfully" });
    } else {
      res.status(200).json({ "messages": "Update user profile fail (error query)" });
    }
  }
});

```

2. Frontend:

F. Refresh Token:

1. Backend:

- Sau khi xác thực access token vừa được nhận. Lấy refresh token id từ access token vừa nhận. Truy vấn refresh token từ refresh token id. Kiểm tra xem refresh token có hợp lệ không, nếu hợp lệ thì tạo access token mới cấp cho người dùng.

```

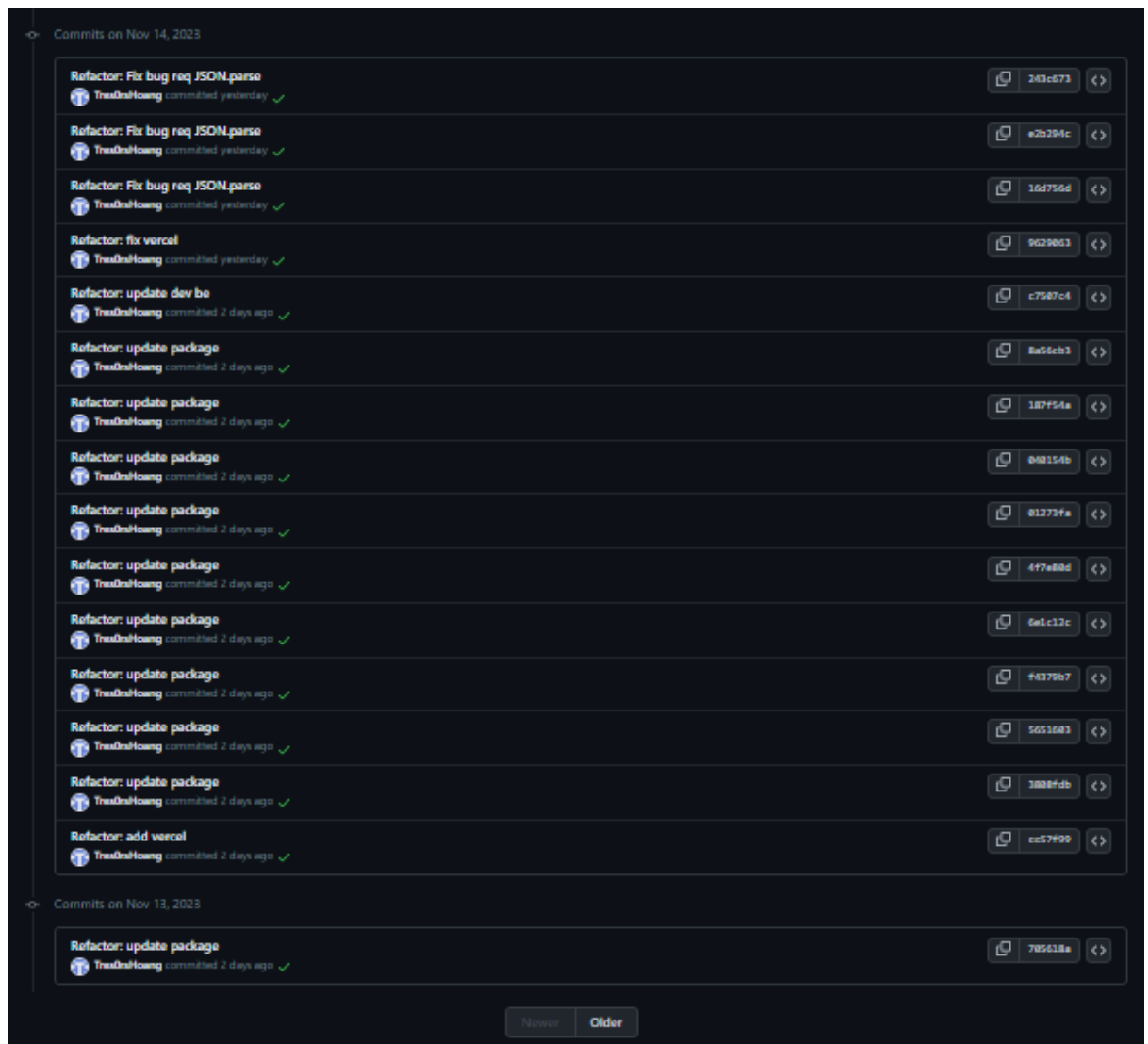
app.get('/refreshToken', authToken, async function(req, res, next) {
  const authorizationHeader = req.headers['authorization'];
  const accessToken = authorizationHeader.split(' ')[1];
  jwt.verify(accessToken, process.env.ACCESS_TOKEN_SECRET_KEY, async (err, data) => {
    if (err) {
      res.status(403).json({ "Error": err });
      throw err;
    }
    const refreshTokenId = data['refresh_token_id'];
    const sql = "SELECT * FROM refresh_authen WHERE id = ?";
    const params = [refreshTokenId];
    const result = await databaseQuery(dbConnection, sql, params).catch(err => console.err(err));
    if (result.length > 0) {
      const refreshToken = result[0]['token'];
      const refreshTokenRevoked = result[0]['is_revoked'];
      if (refreshTokenRevoked === 0) {
        jwt.verify(refreshToken, process.env.REFRESH_TOKEN_SECRET_KEY, (err, data) => {
          if (err) {
            res.status(403).json({ "Error": err });
            throw err;
          }
          const accessToken = jwt.sign(
            { "refresh_token_id": refreshTokenId },
            process.env.ACCESS_TOKEN_SECRET_KEY, { expiresIn: '5m' }
          );
          res.status(200).json({ "access_token": accessToken });
        });
      } else {
        res.status(403).json({ "message": "Refresh token is revoked" });
      }
    } else {
      res.status(403).json({ "Error": "Invalid Access Token" });
    }
  });
});

```

2. Frontend:

III. Git commit history:

Merge pull request #5 from TieuDnsHoang/feature/Khanh trongkhanh11 committed 14 minutes ago ✖	Verified 901d98e <>
Merge branch 'master' of https://github.com/trongkhanh11/advanced_web... trongkhanh11 committed 18 minutes ago ✖	3ba1c22 <>
update edit profile trongkhanh11 committed 18 minutes ago ✖	3591a0f <>
Refactor: Update expired time of access token TieuDnsHoang committed 26 minutes ago ✔	64837a3 <>
Refactor: add database migrate TieuDnsHoang committed 1 hour ago ✔	c3c3e91 <>
Refactor: add database migrate TieuDnsHoang committed 1 hour ago	5aac218 <>
Merge branch 'master' of https://github.com/trongkhanh11/advanced_web... trongkhanh11 committed 1 hour ago ✖	ae9a7f8 <>
Refactor: add CORS TieuDnsHoang committed 1 hour ago ✔	ca8f254 <>
Refactor: update vercel TieuDnsHoang committed 1 hour ago ✖	dcb3906 <>
Refactor: update vercel.json TieuDnsHoang committed 2 hours ago ✔	8f8480c <>
call api trongkhanh11 authored and TieuDnsHoang committed 2 hours ago ✔	777e17e <>
update login, sign up pages + responsive navbar trongkhanh11 authored and TieuDnsHoang committed 2 hours ago	344292e <>
update web ui trongkhanh11 authored and TieuDnsHoang committed 2 hours ago	7983ec7 <>
Refactor: update dev be TieuDnsHoang committed 2 hours ago	cdf5eb <>
Refactor: Update vercel.json accept CORS TieuDnsHoang committed 2 hours ago ✔	b3eacbe <>
call api trongkhanh11 committed 2 hours ago ✔	e1499a3 <>
update login, sign up pages + responsive navbar trongkhanh11 committed 17 hours ago ✔	d931c68 <>
update web ui trongkhanh11 committed 19 hours ago ✔	f55bb08 <>
Refactor: Add update profile TieuDnsHoang committed yesterday ✔	2197f8d <>



IV. Instruction of database, environment setup:

A. Database: MySQL

- File import database được để trong thư mục database migration. Cấu trúc database bao gồm 2 bảng user (id, email, password, first_name, last_name), refresh_authen (id, user_id, token, is_revoked)
- Thông tin ssh của database:
 - Host: sql12.freeseqdatabase.com
 - User/Pass: sql12661265 / cFlhyA67Rd

B. Environment setup:

1. Backend:

```
ACCESS_TOKEN_SECRET_KEY = 1234567890
REFRESH_TOKEN_SECRET_KEY = 0987654321
SALT_ROUNDS = 13
DATABASE_HOST = sql12.freeseqldatabase.com
DATABASE_USER = sql12661265
DATABASE_PASS = cFlhyA67Rd
DATABASE_NAME = sql12661265
```

-
- Thông tin env bao gồm:
 - ACCESS_TOKEN_SECRET_KEY: được dùng để hash access token
 - REFRESH_TOKEN_SECRET_KEY: được dùng để hash refresh token
 - SALT_ROUNDS: được dùng để hash bcrypt
 - DATABASE_HOST: host của database
 - DATABASE_USER: user sử dụng database
 - DATABASE_PASS: password đăng nhập database
 - DATABASE_NAME: tên của database

V. Public host:

- Server: <https://advance-web-programing.vercel.app/>
- Site: <https://advance-web-programing.netlify.app/>