BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 8

XÂY DỰNG NỀN TẢNG MUA BÁN TRANH CHO NGHỆ SĨ KỸ THUẬT SỐ

Giảng viên hướng dẫn	: Kim Ngọc Bách
Họ và tên sinh viên	: Nguyễn Hoàng
Mã sinh viên	: B22DCVT209
Lớp	: E22CQCN01 - B

I. Tiến độ

- Đã viết các Move package và deploy thành công trên localhost

Cấu trúc dữ liệu và logic cơ bản:

```
module package::core {
    use sui::object;
    use sui::tx_context;
    use sui::address;

    // Property struct for key-value pairs
    public struct Property has copy, drop, store {
        key: vector<u8>,
        value: vector<u8>,
    }

    public struct UserProfile has key {
        id: object::UID,
        address: address,
        username: vector<u8>,
        profile_picture: address,
        bio: vector<u8>,
        is_2fa_enabled: bool,
        is_suspended: bool,
        created_at: u64,
    }

    public struct NFTMetadata has store {
        name: vector<u8>,
        description: vector<u8>,
        image_url: vector<u8>,
        properties: vector<Property>,
        royalties: u64,
        watermark: vector<u8>,
    }

    public struct NFT has key {
        id: object::UID,
        creator: address,
        metadata: NFTMetadata,
        current_owner: address,
        price: u64,
        is_for_sale: bool,
        is_auction: bool,
        created_at: u64,
        last_updated: u64,
    }
```

```
public struct Collection has key {
    id: object::UID,
    name: vector<u8>,
    description: vector<u8>,
    creator: address,
    nfts: vector<object::UID>,
    created_at: u64,
}

public struct TransactionRecord has key {
    id: object::UID,
    nft_id: object::UID,
    seller: address,
    buyer: address,
    price: u64,
    timestamp: u64,
    royalty_paid: u64,
}

public struct Comment has key {
    id: object::UID,
    nft_id: object::UID,
    author: address,
    content: vector<u8>,
    created_at: u64,
}

public struct Notification has key {
    id: object::UID,
    recipient: address,
    content: vector<u8>,
    created_at: u64,
    is_read: bool,
}

public fun create_user(
    username: vector<u8>,
    profile_picture: address,
    bio: vector<u8>,
    ctx: &mut tx_context::TxContext
): UserProfile {
    UserProfile {
        id: object::new(ctx),
        address: tx_context::sender(ctx),
        username,
            profile_picture,
            bio,
            is_2fa_enabled: false,
            is_suspended: false,
            created_at: tx_context::epoch(ctx),
        }
    }

    public fun update_profile(
        profile: &mut UserProfile,
        new_bio: vector<u8>,
        new_profile_picture: address,
        ctx: &tx_context::TxContext
    ) {
        assert!(tx_context::sender(ctx) == profile.address, 0);
        profile.bio = new_bio;
        profile.profile_picture = new_profile_picture;
    }

    public fun enable_2fa(profile: &mut UserProfile, ctx:
&tx_context::TxContext) {
        assert!(tx_context::sender(ctx) == profile.address, 0);
        profile.is_2fa_enabled = true;
    }

    public fun disable_2fa(profile: &mut UserProfile, ctx:
&tx_context::TxContext) {
        assert!(tx_context::sender(ctx) == profile.address, 0);
        profile.is_2fa_enabled = false;
    }

    public fun suspend_user(profile: &mut UserProfile, ctx:
&tx_context::TxContext) {
        profile.is_suspended = true;
    }

    public fun unsuspend_user(profile: &mut UserProfile, ctx:
&tx_context::TxContext) {
        profile.is_suspended = false;
    }

    public fun mint_nft(
        creator: address,
        metadata: NFTMetadata,
        price: u64,
        is_auction: bool,
        ctx: &mut tx_context::TxContext
    ): NFT {
        NFT {
            id: object::new(ctx),
            creator,
            metadata,
            current_owner: creator,
            price,
            is_for_sale: true,
            is_auction,
            created_at: tx_context::epoch(ctx),
            last_updated: tx_context::epoch(ctx),
        }
    }

    public fun mint_nft_with_metadata_fields(
        creator: address,
        name: vector<u8>,
        description: vector<u8>,
        image_url: vector<u8>,
        properties: vector<Property>,
        watermark: vector<u8>,
        royalties: u64,
        price: u64,
        is_auction: bool,
        ctx: &mut tx_context::TxContext
    ): NFT {
        let metadata = NFTMetadata {
            name,
            description,
            image_url,
            properties,
            royalties,
            watermark
        };
        mint_nft(creator, metadata, price, is_auction, ctx)
    }

    public fun update_nft_price(nft: &mut NFT, new_price: u64, ctx:
&tx_context::TxContext) {
        assert!(tx_context::sender(ctx) == nft.current_owner, 0);
        nft.price = new_price;
        nft.last_updated = tx_context::epoch(ctx);
    }

    public fun transfer_nft(nft: &mut NFT, new_owner: address, ctx:
```

```
&tx_context::TxContext) {
        assert!(tx_context::sender(ctx) == nft.current_owner, 0);
        nft.current_owner = new_owner;
        nft.last_updated = tx_context::epoch(ctx);
    }

    public fun create_collection(
        name: vector<u8>,
        description: vector<u8>,
        creator: address,
        ctx: &mut tx_context::TxContext
    ): Collection {
        Collection {
            id: object::new(ctx),
            name,
            description,
            creator,
            nfts: vector::empty(),
            created_at: tx_context::epoch(ctx),
        }
    }

    public fun add_nft_to_collection(
        collection: &mut Collection,
        nft_id: object::UID,
        ctx: &tx_context::TxContext
    ) {
        assert!(tx_context::sender(ctx) == collection.creator, 0);
        vector::push_back(&mut collection.nfts, nft_id);
    }

    public fun create_transaction_record(
        nft_id: object::UID,
        seller: address,
        buyer: address,
        price: u64,
        royalty_paid: u64,
        ctx: &mut tx_context::TxContext
    ): TransactionRecord {
        TransactionRecord {
            id: object::new(ctx),
            nft_id,
            seller,
            buyer,
            price,
            timestamp: tx_context::epoch(ctx),
            royalty_paid,
        }
    }

    public fun calculate_royalties(
        sale_price: u64,
        royalty_percentage: u64
    ): u64 {
        (sale_price * royalty_percentage) / 100
    }

    public fun get_nft_royalties(nft: &NFT): u64 {
        nft.metadata.royalties
    }

    public fun remove_nft(nft: &mut NFT, reason: vector<u8>, ctx:
&tx_context::TxContext) {
        // TODO: Implement NFT removal logic
    }

    public fun update_royalty_percentage(new_percentage: u64, ctx:
&tx_context::TxContext) {
        // TODO: Implement royalty update logic
    }

    public fun update_transaction_fee(new_fee: u64, ctx:
&tx_context::TxContext) {
        // TODO: Implement fee update logic
    }

    // Helper functions
    public fun get_username(profile: &UserProfile): vector<u8> {
        profile.username
    }

    public fun get_profile_picture(profile: &UserProfile): address {
        profile.profile_picture
    }

    public fun get_bio(profile: &UserProfile): vector<u8> {
        profile.bio
    }

    public fun is_2fa_enabled(profile: &UserProfile): bool {
        profile.is_2fa_enabled
    }
    public fun is_suspended(profile: &UserProfile): bool {
        profile.is_suspended
    }

    public fun get_nft_name(nft: &NFT): vector<u8> {
        nft.metadata.name
    }

    public fun get_nft_description(nft: &NFT): vector<u8> {
        nft.metadata.description
    }

    public fun get_nft_image_url(nft: &NFT): vector<u8> {
        nft.metadata.image_url
    }

    public fun get_nft_properties(nft: &NFT): vector<Property> {
        nft.metadata.properties
    }

    public fun get_nft_watermark(nft: &NFT): vector<u8> {
        nft.metadata.watermark
    }

    public fun is_nft_for_sale(nft: &NFT): bool {
        nft.is_for_sale
    }

    public fun is_nft_auction(nft: &NFT): bool {
        nft.is_auction
    }

    public fun get_collection_name(collection: &Collection): vector<u8>
{
        collection.name
    }

    public fun get_collection_description(collection: &Collection):
vector<u8> {
        collection.description
    }

    public fun get_collection_nfts(collection: &Collection):
&vector<object::UID> {
        &collection.nfts
    }
}
```

Quản trị hệ thống

```
module package::admin {
    use sui::object;
    use sui::tx_context;
    use sui::address;
    use package::core;

    public struct AdminSettings has key {
        id: object::UID,
        admin_address: address,
        permissions: vector<u8>,
        last_login: u64,
    }

    public fun create_admin(
        admin_address: address,
        permissions: vector<u8>,
        ctx: &mut tx_context::TxContext
    ): AdminSettings {
        AdminSettings {
            id: object::new(ctx),
            admin_address,
            permissions,
            last_login: tx_context::epoch(ctx),
        }
    }

    public fun verify_admin(
        admin: &AdminSettings,
        ctx: &tx_context::TxContext
    ): bool {
        tx_context::sender(ctx) == admin.admin_address
    }

    public fun suspend_user(
        admin: &AdminSettings,
        profile: &mut core::UserProfile,
        ctx: &tx_context::TxContext
    ) {
        assert!(verify_admin(admin, ctx), 0);
        core::suspend_user(profile, ctx);
    }
    public fun unsuspend_user(
        admin: &AdminSettings,
        profile: &mut core::UserProfile,
        ctx: &tx_context::TxContext
    ) {
        assert!(verify_admin(admin, ctx), 0);
        core::unsuspend_user(profile, ctx);
    }

    public fun remove_nft(
        admin: &AdminSettings,
        nft: &mut core::NFT,
        reason: vector<u8>,
        ctx: &tx_context::TxContext
    ) {
        assert!(verify_admin(admin, ctx), 0);
        core::remove_nft(nft, reason, ctx);
    }

    public fun update_royalty_percentage(
        admin: &AdminSettings,
        new_percentage: u64,
        ctx: &tx_context::TxContext
    ) {
        assert!(verify_admin(admin, ctx), 0);
        core::update_royalty_percentage(new_percentage, ctx);
    }

    public fun update_transaction_fee(
        admin: &AdminSettings,
        new_fee: u64,
        ctx: &tx_context::TxContext
    ) {
        assert!(verify_admin(admin, ctx), 0);
        core::update_transaction_fee(new_fee, ctx);
    }
}
```

Hệ thống đấu giá và giao dịch:

```
module package::marketplace {
    use sui::object;
    use sui::tx_context;
    use sui::address;
    use package::core;

    public struct Auction has key {
        id: object::UID,
        nft_id: object::UID,
        highest_bid: u64,
        highest_bidder: address,
        end_time: u64,
        reserve_price: u64,
        created_at: u64,
    }

    public fun create_auction(
        nft_id: object::UID,
        reserve_price: u64,
        end_time: u64,
        ctx: &mut tx_context::TxContext
    ): Auction {
        Auction {
            id: object::new(ctx),
            nft_id,
            highest_bid: 0,
            highest_bidder: tx_context::sender(ctx),
            end_time,
            reserve_price,
            created_at: tx_context::epoch(ctx),
        }
    }

    public fun place_bid(
        auction: &mut Auction,
        bidder: address,
        amount: u64,
        ctx: &tx_context::TxContext
    ) {
        assert!(amount > auction.highest_bid, 0);
        assert!(tx_context::epoch(ctx) < auction.end_time, 1);
        auction.highest_bid = amount;
        auction.highest_bidder = bidder;
    }

    public fun finalize_auction(
        auction: &mut Auction,
        nft: &mut core::NFT,
        ctx: &mut tx_context::TxContext
    ): u64 {
        assert!(tx_context::epoch(ctx) >= auction.end_time, 0);
        assert!(auction.highest_bid >= auction.reserve_price, 1);

        let _royalties = core::calculate_royalties(
            auction.highest_bid,
            core::get_nft_royalties(nft)
        );

        core::transfer_nft(nft, auction.highest_bidder, ctx);
        core::update_nft_price(nft, 0, ctx);
        // Không được phép truy cập trực tiếp vào trường private
        // nft.is_for_sale = false;
        // Sử dụng hàm core::is_nft_for_sale hoặc
core::update_nft_price(nft, 0, ctx) để cập nhật trạng thái bán nếu cần

        auction.highest_bid
    }

    public fun record_transaction(
        nft_id: object::UID,
        seller: address,
        buyer: address,
        price: u64,
        royalty_paid: u64,
        ctx: &mut tx_context::TxContext
    ): core::TransactionRecord {
        core::create_transaction_record(
            nft_id,
            seller,
            buyer,
            price,
            royalty_paid,
            ctx
        )
    }
}
```

Quản lý NFT:

```
module package::nft {
    use sui::tx_context;
    use package::core;

    public fun mint_nft(
        creator: address,
        name: vector<u8>,
        description: vector<u8>,
        image_url: vector<u8>,
        properties: vector<core::Property>,
        watermark: vector<u8>,
        royalties: u64,
        price: u64,
        is_auction: bool,
        ctx: &mut tx_context::TxContext
    ): core::NFT {
        core::mint_nft_with_metadata_fields(
            creator,
            name,
            description,
            image_url,
            properties,
            watermark,
            royalties,
            price,
            is_auction,
            ctx
        )
    }

    public fun update_nft_price(
        nft: &mut core::NFT,
        new_price: u64,
        ctx: &tx_context::TxContext
    ) {
        core::update_nft_price(nft, new_price, ctx)
    }

    public fun transfer_nft(
        nft: &mut core::NFT,
        new_owner: address,
        ctx: &tx_context::TxContext
    ) {
        core::transfer_nft(nft, new_owner, ctx)
    }

    public fun get_nft_name(nft: &core::NFT): vector<u8> {
        core::get_nft_name(nft)
    }

    public fun get_nft_description(nft: &core::NFT): vector<u8> {
        core::get_nft_description(nft)
    }

    public fun get_nft_image_url(nft: &core::NFT): vector<u8> {
        core::get_nft_image_url(nft)
    }

    public fun get_nft_properties(nft: &core::NFT):
vector<core::Property> {
        core::get_nft_properties(nft)
    }

    public fun get_nft_watermark(nft: &core::NFT): vector<u8> {
        core::get_nft_watermark(nft)
    }

    public fun is_nft_for_sale(nft: &core::NFT): bool {
        core::is_nft_for_sale(nft)
    }

    public fun is_nft_auction(nft: &core::NFT): bool {
        core::is_nft_auction(nft)
    }

    public fun get_nft_royalties(nft: &core::NFT): u64 {
        core::get_nft_royalties(nft)
    }
}
```

Quản lý người dùng và bộ sưu tập:

```
module package::user {
    use sui::object;
    use sui::tx_context;
    use sui::address;
    use package::core;

    public fun create_user(
        username: vector<u8>,
        profile_picture: address,
        bio: vector<u8>,
        ctx: &mut tx_context::TxContext
    ): core::UserProfile {
        core::create_user(username, profile_picture, bio, ctx)
    }

    public fun update_profile(
        profile: &mut core::UserProfile,
        new_bio: vector<u8>,
        new_profile_picture: address,
        ctx: &tx_context::TxContext
    ) {
        core::update_profile(profile, new_bio, new_profile_picture,
ctx);
    }

    public fun enable_2fa(profile: &mut core::UserProfile, ctx:
&tx_context::TxContext) {
        core::enable_2fa(profile, ctx);
    }

    public fun disable_2fa(profile: &mut core::UserProfile, ctx:
&tx_context::TxContext) {
        core::disable_2fa(profile, ctx);
    }

    public fun get_username(profile: &core::UserProfile): vector<u8> {
        core::get_username(profile)
    }

    public fun get_profile_picture(profile: &core::UserProfile): address
{
        core::get_profile_picture(profile)
    }

    public fun get_bio(profile: &core::UserProfile): vector<u8> {
        core::get_bio(profile)
    }

    public fun is_2fa_enabled(profile: &core::UserProfile): bool {
        core::is_2fa_enabled(profile)
    }

    public fun is_suspended(profile: &core::UserProfile): bool {
        core::is_suspended(profile)
    }

    public fun create_collection(
        name: vector<u8>,
        description: vector<u8>,
        ctx: &mut tx_context::TxContext
    ): core::Collection {
        core::create_collection(name, description,
tx_context::sender(ctx), ctx)
    }

    public fun add_nft_to_collection(
        collection: &mut core::Collection,
        nft_id: object::UID,
        ctx: &mut tx_context::TxContext
    ) {
        core::add_nft_to_collection(collection, nft_id, ctx)
    }

    public fun get_collection_name(collection: &core::Collection):
vector<u8> {
        core::get_collection_name(collection)
    }

    public fun get_collection_description(collection:
&core::Collection): vector<u8> {
        core::get_collection_description(collection)
    }

    public fun get_collection_nfts(collection: &core::Collection):
&vector<object::UID> {
        core::get_collection_nfts(collection)
    }
}
```

II. Mục tiêu tuần 9

- Tạo dần frontend cho các Move package