Here is the detailed functionality and API design for the \*\*Free Software Sharing\*\* module and the summaries for \*\*Code Service & Product Build\*\* and \*\*Marketplace for Resources & Services\*\* modules, translated into English:

---

# PHASE 1: Free Software Sharing

## 1. Main Features

### 1.1. Software Management

- Display software list by category, platform, type (free, paid)

- Search and filter software by name, category, platform

- View software details: description, images, download link, ratings, comments

- Add new software (admin or approved users only)

- Approve user-submitted software (admin)

### 1.2. User Management

- Register, login (JWT supported)

- User profile management

- History of downloaded software, ratings, comments

### 1.3. Ratings and Comments

- Users can rate software (1-5 stars)

- Write comments and feedback on software

- Comment management (admin can delete or approve)

### 1.4. Notifications

- Send email or notifications about new software updates or approvals

---

## 2. API Design (RESTful)

### 2.1. User Registration & Login

| Method | URL | Description | Request Body | Response |

|--------|-------------------|--------------------------|------------------------------------------------|--------------------------------------------|

| POST | /api/auth/register | Register a new account | `{ "name": "string", "email": "string", "password": "string" }` | `{ "message": "Registration successful" }` or error |

| POST | /api/auth/login | Login and get JWT token | `{ "email": "string", "password": "string" }` | `{ "token": "jwt\_token", "user": {...} }` or error |

---

### 2.2. Software Management

| Method | URL | Description | Request Body (POST/PUT) | Response | Access Control |

|--------|----------------------|------------------------------|------------------------------------------------|----------------------------------|---------------------|

| GET | /api/softwares | Get software list (with filters) | Query params: `category`, `platform`, `search` | List of software | Public |

| GET | /api/softwares/:id | Get software details by ID | - | Software detail | Public |

| POST | /api/softwares | Add new software | `{ "name": "", "description": "", "category\_id": "", "platform": "", "download\_link": "" }` | New software or pending approval | User/Admin |

| PUT | /api/softwares/:id | Update software | Same as POST | Updated software | Admin or creator |

| DELETE | /api/softwares/:id | Delete software | - | Deletion confirmation | Admin |

---

### 2.3. Ratings and Comments

| Method | URL | Description | Request Body | Response | Access Control |

|--------|--------------------------|-----------------------------|------------------------------------------------|-------------------------------|-----------------|

| POST | /api/reviews | Add rating/comment | `{ "target\_type": "software", "target\_id": "id", "rating": 4, "comment": "..." }` | Success message | Logged-in users |

| GET | /api/reviews/:software\_id | Get software reviews | - | List of ratings and comments | Public |

| DELETE | /api/reviews/:id | Delete rating/comment | - | Deletion confirmation | Admin or owner |

---

### 2.4. User Management

| Method | URL | Description | Request Body | Response | Access Control |

|--------|----------------|---------------------------|-------------------------------|-----------------------------|-----------------|

| GET | /api/users/:id | Get user info | - | User info | User or Admin |

| PUT | /api/users/:id | Update user info | `{ "name": "", "email": "" }` | Updated user info | User or Admin |

---

## 3. Sample JSON Request/Response

### Example: Get software list

\*\*Request:\*\*

`GET /api/softwares?category=1&platform=windows&search=editor`

\*\*Response:\*\*

```json

{

"softwares": [

{

"id": 101,

"name": "Super Editor",

"description": "Free photo editing software",

"category\_id": 1,

"platform": "windows",

"download\_link": "https://example.com/download/super-editor",

"rating": 4.5

}

],

"total": 25

}

```

---

## 4. Main Database Model (PostgreSQL)

| Table | Key Fields | Description |

|------------|---------------------------------------------|----------------------------------|

| users | id, name, email, password\_hash, role, created\_at | User info and roles |

| softwares | id, name, description, category\_id, platform, download\_link, created\_by, status, created\_at | Software info |

| categories | id, name, parent\_id | Software categories |

| reviews | id, user\_id, target\_type, target\_id, rating, comment, created\_at | Ratings and comments |

---

## 5. Password & Authentication

- Use \*\*bcrypt\*\* to hash passwords before storing.

- Authenticate users with \*\*JWT\*\* tokens.

- Middleware to verify JWT on protected routes.

---

# PHASE 2: Code Service & Product Build Module Summary

## 1. Main Goals

- Manage full workflow from client project request, developer quoting, to product completion and payment.

- Support direct chat between client and developer.

- Manage secure payments with escrow.

- Showcase featured portfolios to build trust and assist client decisions.

---

## 2. Core Features

### 2.1. Project Management

- Clients create detailed project requests.

- Developers view projects and submit quotes.

- Track project status, progress, and contracts.

- Support milestone or full payment.

- Manage feedback after project completion.

### 2.2. Quote Management

- Developers send quotes per project.

- Clients accept or reject quotes.

### 2.3. Chat System

- Real-time messaging stored per project.

- New message notifications.

### 2.4. Payment Management

- Payments via Stripe, PayPal, etc.

- Payment status tracking and escrow release.

### 2.5. Portfolio Showcase

- List fixed and user-uploaded portfolios.

- Clients can view, rate, and comment.

- Build trust with positive feedback.

---

## 3. API Design (RESTful)

| Feature | Method | URL | Description | Access Control |

|--------------------|-----------------|----------------------------------|---------------------------------|----------------------------|

| Project Management | POST/GET/PUT/DELETE | /api/projects, /api/projects/:id | Create, view, update, cancel projects | Client, Developer, Admin |

| Quote Management | POST/GET/PUT | /api/projects/:id/quotes, /api/quotes/:quote\_id | Send, view, update quotes | Developer, Client, Admin |

| Chat System | POST/GET | /api/projects/:id/messages | Send, get messages | Client, Developer, Admin |

| Payment Management | POST/GET/PUT | /api/projects/:id/payments, /api/payments/:payment\_id | Create, view, update payments | Client, Developer, Admin |

| Portfolio | POST/GET/PUT/DELETE | /api/portfolios, /api/portfolios/:id | Manage portfolios | Developer, Seller, Admin, Public |

| Portfolio Reviews | POST/GET | /api/portfolios/:id/reviews | Add, view portfolio reviews | User, Public |

| Static Portfolios | GET | /api/static-portfolios | Get fixed portfolios | Public |

---

## 4. Main Database Model (PostgreSQL)

- `users`: id, username, email, password\_hash, role (client/developer/admin), created\_at

- `projects`: id, client\_id, title, description, requirements, budget, deadline, status, created\_at

- `quotes`: id, project\_id, developer\_id, price, timeline, message, status, created\_at

- `messages`: id, project\_id, sender\_id, content, created\_at

- `payments`: id, project\_id, payer\_id, amount, payment\_method, status, escrow\_release, transaction\_id, created\_at

- `portfolios`: id, developer\_id, title, description, images, demo\_link, technologies, created\_at

- `portfolio\_reviews`: id, portfolio\_id, user\_id, rating, comment, created\_at

\*\*Relationships:\*\*

`users` 1-N `projects`

`projects` 1-N `quotes`, `messages`, `payments`

`users` 1-N `portfolios`

`portfolios` 1-N `portfolio\_reviews`

---

## 5. Authorization & Security

- Login required for project, quote, chat, payment actions.

- Roles:

- Client: create projects, view quotes, chat, pay

- Developer: view projects, send quotes, chat, manage portfolios

- Admin: full management of projects, payments, portfolios, users

- User: can review portfolios

---

# PHASE 3: Marketplace for Resources & Services with Escrow Payment

## 1. Requirements Analysis

### 1.1. Main Goals

- Build an e-commerce platform for users (sellers) to sell resources/services.

- Buyers can search, order, and pay securely via escrow.

- Manage orders from creation to completion or cancellation.

- Transparent rating and feedback system to boost seller reputation and service quality.

### 1.2. Main Actors

- \*\*Seller\*\*: Lists resources/services.

- \*\*Buyer\*\*: Purchases resources/services.

- \*\*Admin\*\*: Manages the entire system.

### 1.3. Core Features

- Sellers add, edit, delete products/services.

- Buyers browse, order, and pay.

- Escrow holds payment until buyer confirms receipt.

- Order management: status, history, cancellations.

- Ratings for products and sellers.

- Notifications on order/payment status via email or system.

---

## 2. System Design

### 2.1. Main Data Model

| Table | Description |

|---------------|------------------------------------------|

| users | User info (buyer, seller, admin) |

| products | Resources/services listed for sale |

| orders | Order info (buyer, product, status) |

| order\_items | Details of products in orders (if multiple)|

| payments | Payment info, escrow status |

| reviews | Product and seller reviews |

| notifications | Order/payment status notifications |

---

### 2.2. Order Statuses

- Pending

- Processing

- Shipped/Delivered

- Completed

- Cancelled

---

### 2.3. Main Workflow

1. Seller lists products.

2. Buyer searches and places orders.

3. Buyer pays; money held in escrow.

4. Seller delivers product/service.

5. Buyer confirms receipt; escrow released.

6. Buyer reviews product and seller.

7. Admin oversees orders and dispute handling.

---

## 3. API Design (RESTful)

| Method | URL | Description | Request Body/Params | Access Control |

|--------|-------------------------|---------------------------------|-------------------------------------------|---------------------|

| POST | /api/products | Seller adds new product | `{ title, description, price, images, category }` | Seller (authenticated) |

| GET | /api/products | Get product list | Query params: `category`, `search`, `seller\_id` | Public |

| GET | /api/products/:id | Get product detail | - | Public |

| PUT | /api/products/:id | Update product | Same as POST | Seller/Admin |

| DELETE | /api/products/:id | Delete product | - | Seller/Admin |

| POST | /api/orders | Buyer creates order | `{ product\_id, quantity, shipping\_info }` | Buyer |

| GET | /api/orders/:id | Get order detail | - | Buyer/Seller/Admin |

| PUT | /api/orders/:id/cancel | Cancel order | - | Buyer/Seller/Admin |

| POST | /api/payments | Make escrow payment | `{ order\_id, payment\_method, amount }` | Buyer |

| GET | /api/payments/:id/status | Get payment status | - | Buyer/Seller/Admin |

| POST | /api/orders/:id/confirm | Buyer confirms receipt, release escrow | - | Buyer |

| POST | /api/reviews | Add product/seller review | `{ order\_id, rating, comment }` | Buyer |

| GET | /api/reviews/:product\_id | Get product reviews | - | Public |

---

## 4. Simplified Database Schema (PostgreSQL)

```sql

CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(100),

email VARCHAR(100),

password\_hash VARCHAR(255),

role VARCHAR(20), -- buyer, seller, admin

created\_at TIMESTAMP DEFAULT NOW()

);

CREATE TABLE products (

id SERIAL PRIMARY KEY,

seller\_id INT REFERENCES users(id),

title VARCHAR(255),

description TEXT,

price DECIMAL(10,2),

images JSONB,

category VARCHAR(100),

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

CREATE TABLE orders (

id SERIAL PRIMARY KEY,

buyer\_id INT REFERENCES users(id),

status VARCHAR(20) DEFAULT 'Pending',

total\_amount DECIMAL(10,2),

shipping\_info JSONB,

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

CREATE TABLE order\_items (

id SERIAL PRIMARY KEY,

order\_id INT REFERENCES orders(id),

product\_id INT REFERENCES products(id),

quantity INT DEFAULT 1,

price DECIMAL(10,2)

);

CREATE TABLE payments (

id SERIAL PRIMARY KEY,

order\_id INT REFERENCES orders(id),

amount DECIMAL(10,2),

payment\_method VARCHAR(50),

status VARCHAR(20), -- pending, completed, failed, refunded

escrow\_release BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT NOW()

);

CREATE TABLE reviews (

id SERIAL PRIMARY KEY,

order\_id INT REFERENCES orders(id),

product\_id INT REFERENCES products(id),

buyer\_id INT REFERENCES users(id),

rating INT CHECK (rating BETWEEN 1 AND 5),

comment TEXT,

created\_at TIMESTAMP DEFAULT NOW()

);

```

---

## 5. Authorization & Security

- \*\*Seller\*\*: Manage products, view own orders.

- \*\*Buyer\*\*: Create orders, pay, confirm receipt, review.

- \*\*Admin\*\*: Manage all products, orders, payments, handle disputes.

- Authentication via JWT or sessions.

- Role-based access control on endpoints.

- Secure payment and personal data handling.

---

## 6. Extensions & Notes

- Integrate popular payment gateways (Stripe, PayPal) with escrow support.

- Notification system (email, push).

- Dispute management.

- Sales and revenue analytics.

- UX optimization for buyers and sellers.

---

If you want, I can help with backend sample code (Express.js), frontend (React), or detailed parts. Where would you like to start?