**基于区块链的宠物保险系统**

**小组成员：林文杰、宋耀龙、胡文举、牛童、高忠辉**

**分工：林文杰（项目文档的撰写、编写智能合约、区块链的搭建与连接）**

**牛童（编写智能合约、区块链的搭建与连接）**

**高忠辉（编写智能合约、区块链的搭建与连接）**

**宋耀龙（前端与后端的项目代码实现）**

**胡文举（前端与后端的项目代码实现、数据库搭建）**

**基于区块链的宠物保险系统**

# **一、项目概述**

## **1.1项目背景与目标**

近年来，区块链技术的快速发展为各行各业带来了创新契机，尤其是在金融和保险领域。传统的宠物保险系统面临数据透明度低、处理效率低下、安全性不足等诸多挑战。为了解决这些问题，我们开发了一个基于区块链技术的宠物保险系统，旨在提供一个安全、高效和透明的解决方案。

本项目的主要目标包括：

* 提高数据透明度：利用区块链的透明性，使保险合同和理赔过程对所有相关方可追溯。
* 增强数据安全性：通过区块链的不可篡改性，确保宠物信息和保险数据的安全。
* 提升处理效率：借助智能合约自动化执行保险合同和理赔流程，减少人工干预，提高效率。
* 改善用户体验：提供用户友好的界面，方便宠物主人管理宠物信息和保险服务。

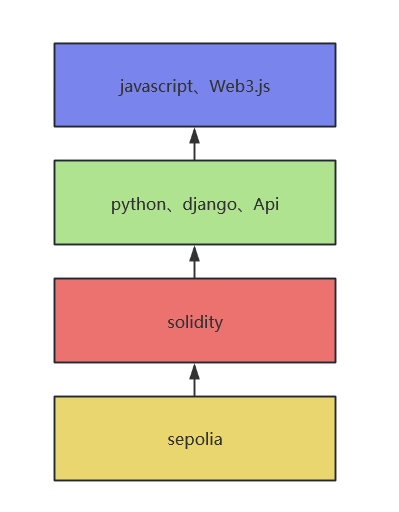
## **1.2项目范围**

本项目的实施范围涵盖以下几个方面：

* 系统架构设计：设计并实现系统的整体架构，包括前端、后端和区块链部分的交互。
* 智能合约开发：编写并部署用于管理保险合同和理赔流程的智能合约。
* 前端开发：开发用户界面，包括用户注册、登录、宠物信息管理和保险服务页面。
* 功能模块实现：包括用户注册与登录、宠物信息管理和保险服务与理赔功能。
* 测试与调试：对系统进行全面的功能和安全测试，确保系统稳定可靠。
* 部署：在Sepolia测试网上部署系统，并为未来的主网部署做好准备。

# **二、系统架构图**

下图展示了系统的整体架构，涵盖了用户界面层、后端服务层、智能合约层和区块链网络（Sepolia测试网）的相互关系及其功能：



1、用户界面层（前端）

* 技术栈: 使用React框架和Web3.js。
* 功能: 提供用户注册、登录、宠物信息管理和保险服务页面，处理用户交互并与智能合约进行通信。

2、后端服务层

* 技术栈: 使用Python和Django框架。
* 功能: 处理业务逻辑，提供API接口，管理与前端和区块链的交互，存储用户和宠物信息。

3、智能合约层

* 技术栈: 使用Solidity编写智能合约。
* 功能: 管理保险合同和理赔流程，确保合同的自动执行和数据的不可篡改。

4、区块链网络

* 网络: Sepolia测试网。
* 功能: 提供去中心化的账本，记录所有的交易和合同信息，确保数据的透明和安全。

# **三、技术栈**

## **3.1编程语言和框架**

1、Python:

* 主要用于后端服务的开发。
* 使用Django框架，它强调快速开发和清晰、实用的设计，适合于处理Web应用的后端逻辑和数据库交互。

2、 JavaScript:

* 负责前端开发，用于构建交互式用户界面。
* 主要利用React库，它支持高效的用户界面渲染和组件化的开发方式。
* Web3.js库用于实现前端与区块链智能合约的交互，管理区块链交易和账户信息。

3、Solidity:

* 用于智能合约的开发，是以太坊智能合约的主要编程语言。
* 管理保险合同的逻辑和处理理赔流程，运行在以太坊兼容的区块链上。

## **3.2测试网络和工具**

1、Sepolia测试网:

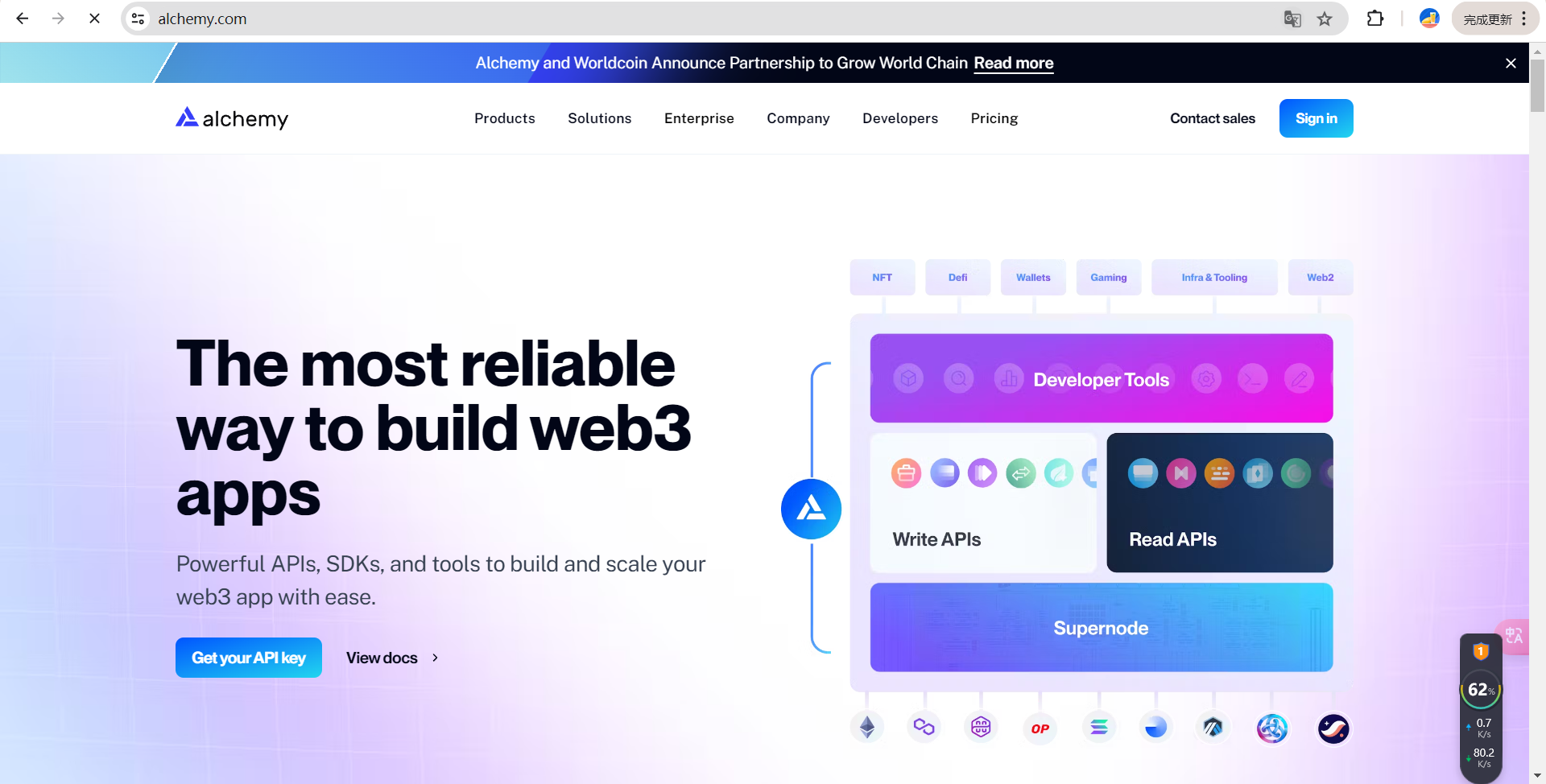
* 作为以太坊的官方测试网络之一，提供了一个接近真实环境的平台，用于智能合约的开发和测试。
* 允许开发者在无需消耗真实资产的情况下，测试智能合约的功能和性能。

2、开发和测试工具:

* MetaMask: 浏览器扩展，允许用户与区块链网络交互，是管理个人账户和进行交易的工具。



* Alchemy: 提供区块链开发必需的基础设施，如区块链节点访问、API和开发工具，支持快速、稳定的应用开发和运行。



# **四、智能合约**

## **4.1设计与实现**

智能合约是基于区块链的宠物保险系统的核心组成部分，负责执行保险合同的逻辑和处理理赔流程。以下是智能合约的设计与实现细节：

1、合同逻辑

* 保险合同管理：智能合约用于创建、管理和存储宠物保险合同的详细信息，如保险期限、保费、覆盖范围等。

2、数据存储

* 宠物信息：合约内存储宠物的基本信息，如种类、年龄和医疗历史，这些信息用于理赔时的验证。
* 交易记录：所有与保险合同和理赔相关的交易都在区块链上记录，确保数据的不可篡改性和透明度。

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract PetInsurance {

struct PolicyInfo {

string petName;

string petSpecies;

string policyName;

string policyTermYears;

}

mapping(string => PolicyInfo[]) private policiesByUsername;

event PolicyPurchased(string username, string petName, string petSpecies, string policyName, string policyTermYears);

function buyPolicy(

string memory username,

string memory petName,

string memory petSpecies,

string memory policyName,

string memory policyTermYears

) public {

policiesByUsername[username].push(PolicyInfo(petName, petSpecies, policyName, policyTermYears));

emit PolicyPurchased(username, petName, petSpecies, policyName, policyTermYears);

}

function getPolicies(string memory username)

public

view

returns (

string[] memory petNames,

string[] memory petSpecies,

string[] memory policyNames,

string[] memory policyTermYears

)

{

uint256 policyCount = policiesByUsername[username].length;

petNames = new string[](policyCount);

petSpecies = new string[](policyCount);

policyNames = new string[](policyCount);

policyTermYears = new string[](policyCount);

for (uint256 i = 0; i < policyCount; i++) {

PolicyInfo memory policy = policiesByUsername[username][i];

petNames[i] = policy.petName;

petSpecies[i] = policy.petSpecies;

policyNames[i] = policy.policyName;

policyTermYears[i] = policy.policyTermYears;

}

return (petNames, petSpecies, policyNames, policyTermYears);

}

}

3、安全机制

* 异常处理：智能合约包含异常检测和处理机制，确保在数据输入错误或操作非法时能够恰当地处理。

modifier validString(string memory str) {

require(bytes(str).length > 0, "String cannot be empty");

\_;

}

## **4.2部署与测试**

智能合约的部署和测试是确保系统功能正常和数据安全的关键步骤：

1、合约部署

* 网络选择：首先在Sepolia测试网进行部署，以检测合约功能和性能。
* 正式发布：在通过所有测试和验证后，合约将部署到以太坊主网，供实际使用。

2、测试策略

* 单元测试：对智能合约的每个独立功能进行测试，确保其按预期工作。
* 集成测试：测试智能合约与前端应用和后端服务的集成，确保整个系统的交互流程无误。
* 压力测试：模拟高负载和异常情况下的合约表现，确保在极端条件下的稳定性和安全性。

# **五、前端开发**

## **5.1页面设计与架构**

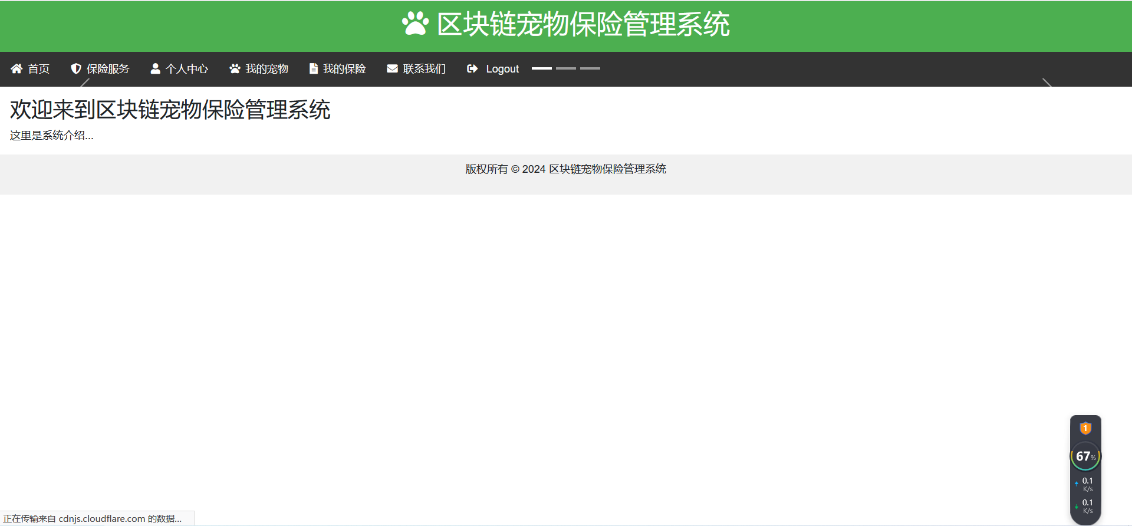
前端开发的主要任务是创建一个直观、响应式的用户界面，以提供良好的用户体验和有效的与区块链智能合约的交互。以下是前端开发的关键组成部分：

1、技术栈

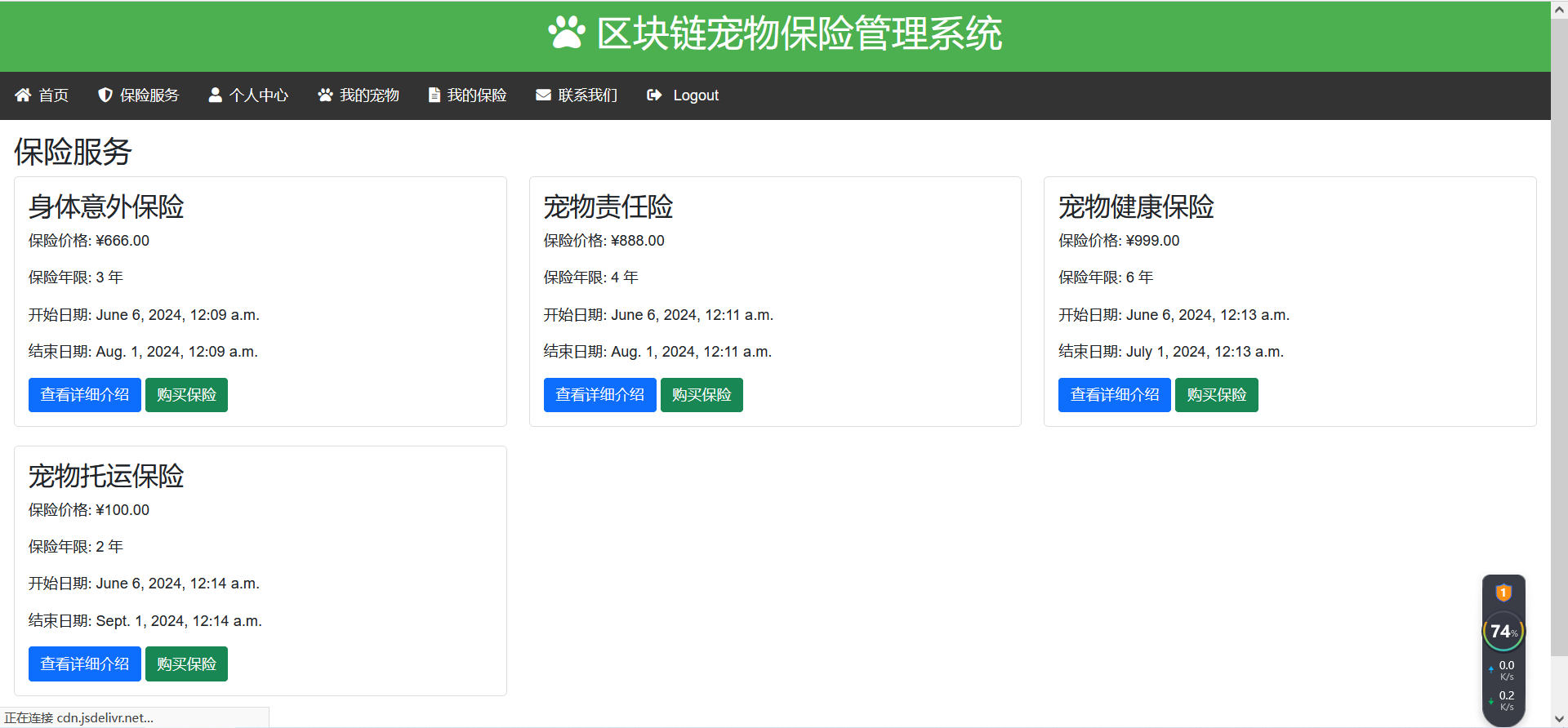
* 使用React框架开发，利用其组件化和高效渲染的特性来构建用户界面。
* Web3.js库用于与区块链智能合约交互，处理账户信息和交易。

2、界面组成

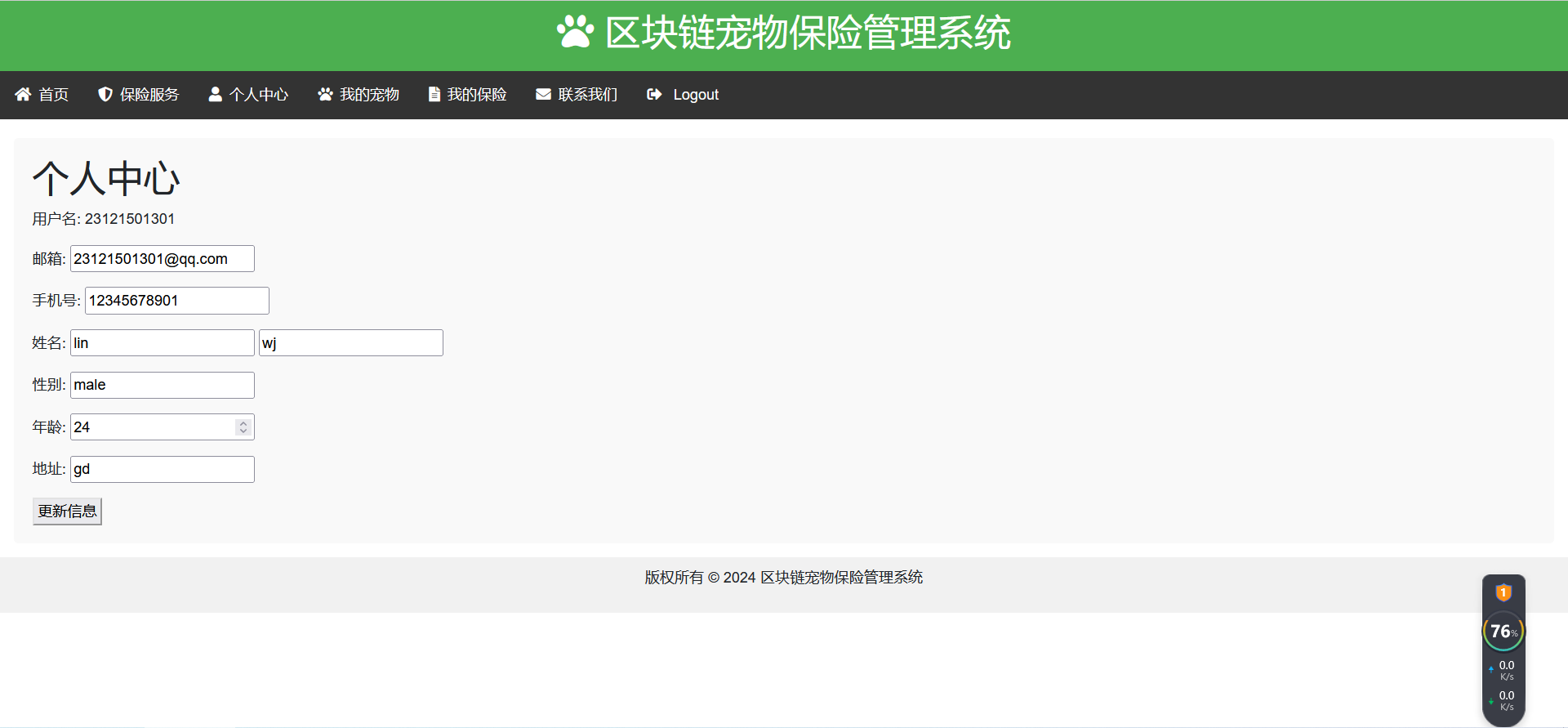
* 首页：提供系统概览，包括导航到各主要功能的入口。



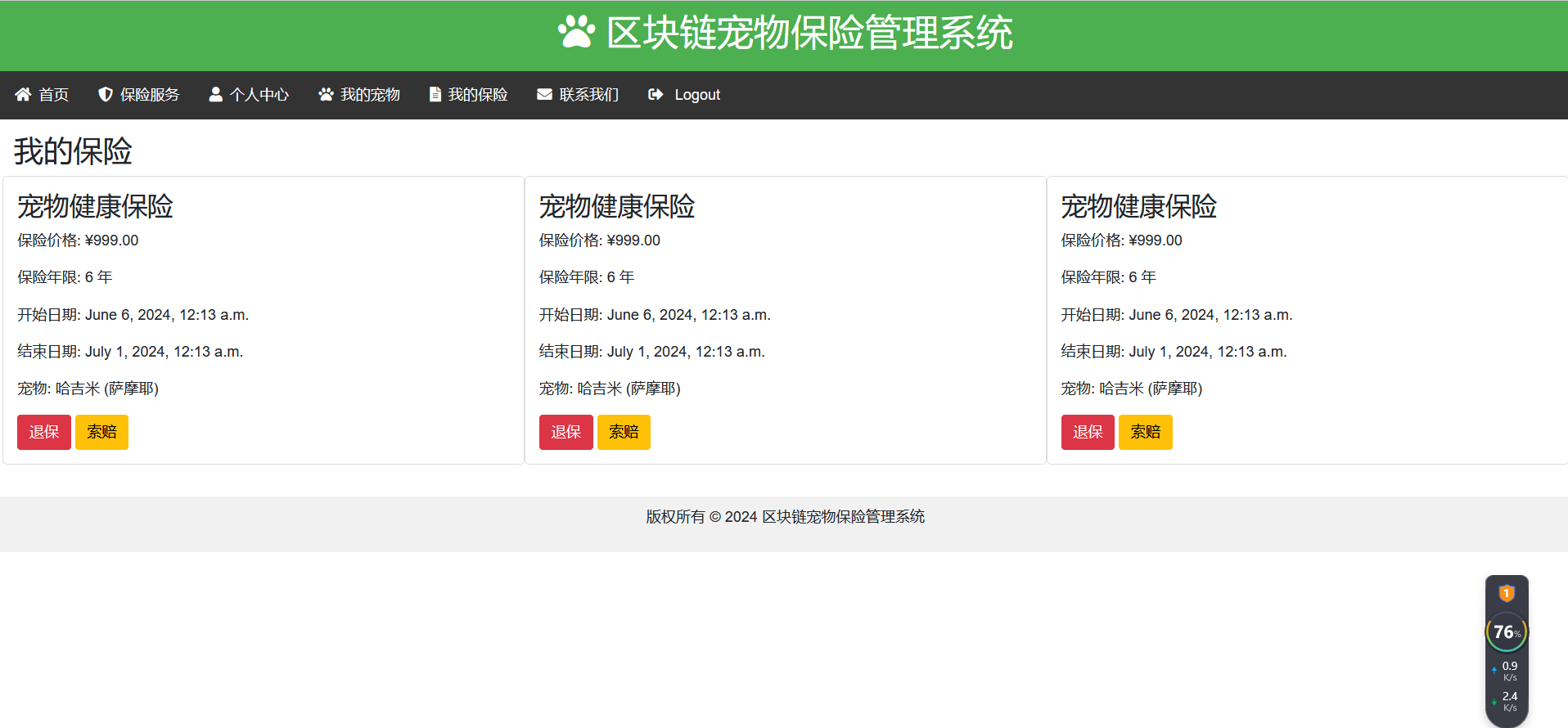
* 保险服务页面：用户可以浏览不同的保险产品，选择合适的保险计划并进行购买。



* 个人中心：用户可以管理个人信息的更新。



* 我的保险：用户可以查看和管理自己的保险合同，包括保单状态、理赔请求。



3、交互设计

* 界面设计考虑到易用性和可访问性，确保所有用户都能轻松地导航和操作。
* 交互动画和视觉反馈用于增强用户体验，如加载动画、按钮点击反馈等。

4、响应式设计

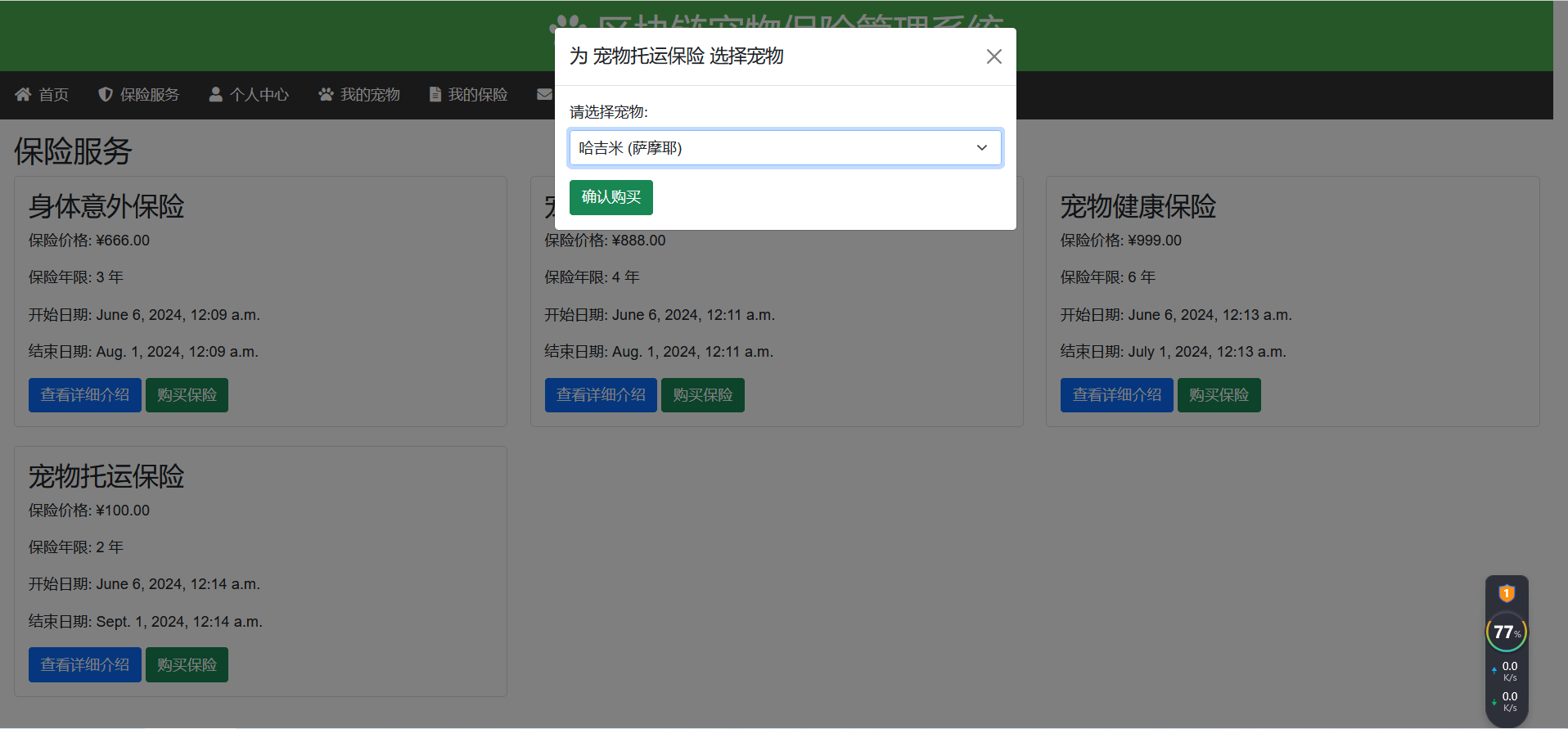
* 页面布局和元素设计能够适应不同设备和屏幕尺寸，包括桌面、平板和手机。
* 使用CSS Flexbox和Grid系统进行响应式布局设计，保证布局的灵活性和整洁性。

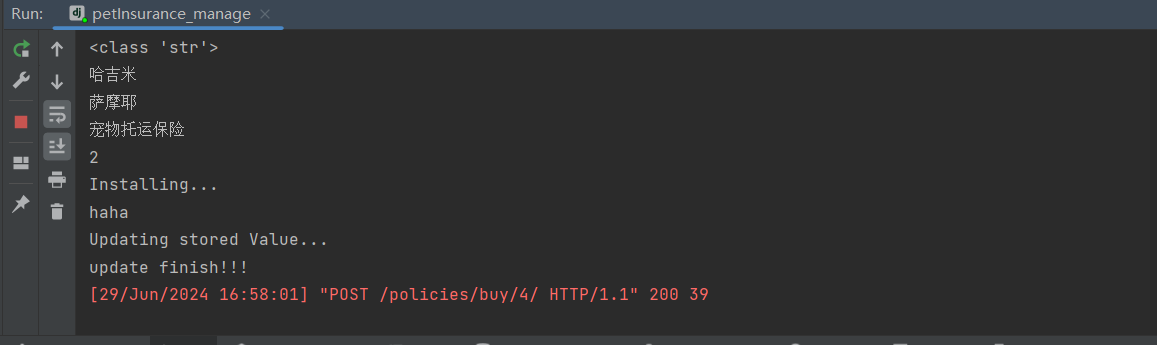
## **5.2前端与智能合约交互**

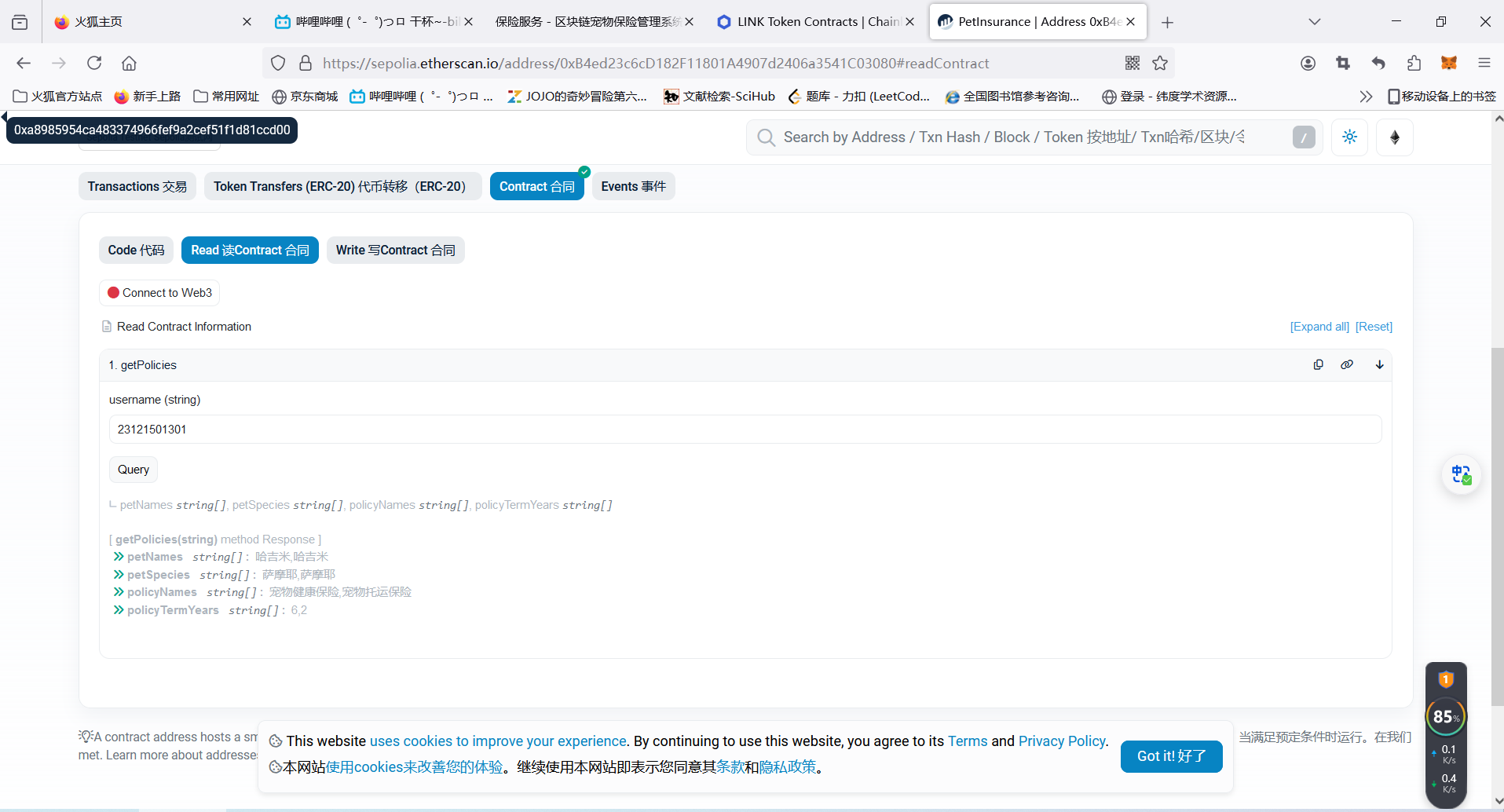
前端与智能合约的交互是系统功能的关键部分，确保用户操作能够转化为区块链上的有效交易：

1、交易发起

* 用户在前端界面进行的操作如购买保险、提交理赔等，将通过Web3.js生成相应的区块链交易。







2、状态更新

* 前端应用监听区块链事件，实时更新用户界面上的信息，如保单状态变更、理赔进度等。

# **六、功能模块**

## **6.1用户注册与登录**

1、用户注册

* 提供注册表单，用户需填写基本信息如姓名、电子邮箱和密码。
* 集成邮箱验证功能，确保用户输入的邮箱有效，并通过发送激活链接来完成注册流程。
* 使用加密技术保护用户密码和个人信息安全。

2、用户登录

* 用户通过邮箱和密码登录系统。
* 支持忘记密码和密码重置功能，通过用户注册的邮箱发送重置链接。
* 引入多因素认证（如手机短信验证码），增强账户安全。

## **6.2宠物信息管理**

1、信息录入

* 用户可以添加或更新其宠物的详细信息，如宠物名称、种类、年龄和健康记录。

2、信息查看

* 用户可以查看和管理自己所有宠物的信息。

## **6.3保险服务**

1、保险产品浏览

* 展示不同类型的宠物保险产品，包括详细的保险覆盖内容、价格和购买条件。
* 用户可以根据宠物类型和需求比较不同保险计划。

2、购买保险

* 用户可以选择合适的保险计划，并通过在线支付方式购买保险。
* 交易信息将通过智能合约处理，确保交易的透明和不可篡改。

# **七、部署**

## **7.1环境部署**

1、开发环境

* 配置开发环境，包括安装必要的软件和工具（如web3、solcx、json等）。
* 设置本地开发服务器和数据库，以便开发和测试。

2、测试环境

* 搭建与生产环境相似的测试环境，用于执行系统测试。
* 配置Sepolia测试网，用于智能合约的部署和测试。

3、生产环境

* 配置生产服务器，包括安全设置和性能优化。
* 部署智能合约到以太坊主网，确保所有功能正常运行。
* 设置监控和日志记录系统，用于跟踪系统状态和及时响应潜在问题。

## **7.2测试网部署**

1、智能合约部署

* 在Sepolia测试网上部署智能合约，验证合约功能和交互逻辑。
* 使用pycharm工具进行合约编译、测试和部署。

2、前端和后端部署

* 在测试环境中部署前端和后端应用，确保它们能够与测试网上的智能合约正确交互。
* 进行端到端的集成测试，确保所有组件协同工作无误。

## **7.3主网部署（计划）**

1、准备阶段

* 确保所有测试都已通过，并且没有未解决的重大问题。
* 通知所有用户有关即将发生的部署和可能的服务中断。

2、部署操作

* 在以太坊主网上部署智能合约。
* 更新前端和后端应用，确保它们指向主网合约地址。
* 进行彻底的安全审查和性能优化。

3、监控与优化

* 持续监控系统性能和用户反馈。
* 根据需要进行优化和更新，以提高系统性能和用户体验。

# **八、文档与资源**

## **8.1用户手册**

**8.1.1使用指南**

1、注册与登录：

* 打开系统首页，点击“注册”按钮。
* 填写姓名、邮箱和密码，点击“提交”。
* 收到激活邮件后，点击邮件中的激活链接完成注册。
* 返回系统首页，输入邮箱和密码，点击“登录”进入系统。

2、管理宠物信息：

* 登录后，导航到“个人中心”页面。
* 点击“添加宠物”，填写宠物的名称、种类、年龄等信息，并上传宠物照片和医疗记录。
* 点击“保存”完成宠物信息的添加。

3、购买保险：

* 在导航栏中选择“保险服务”。
* 浏览不同的保险产品，点击感兴趣的产品查看详细信息。
* 选择合适的保险计划，点击“购买”，按照提示完成支付。

4、查看保险状态：

* 在“个人中心”页面，可以查看所有保单的状态和历史记录。
* 点击某个保单，可以查看详细的保险条款和理赔记录。

## **8.2开发者指南**

**8.2.1系统概述**

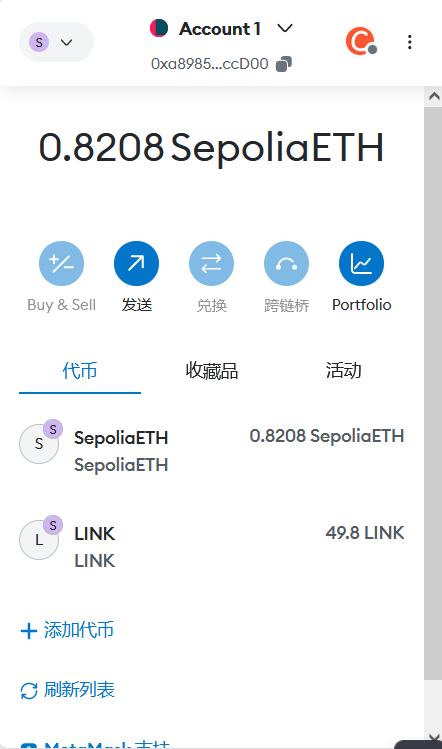
1、架构说明：本系统包括前端、后端、智能合约和区块链网络四个主要部分。

* 前端使用javascript和Web3.js。
* 后端使用Python和Django框架。
* 智能合约使用Solidity编写，部署在Sepolia测试网上。
* 区块链网络通过Alchemy提供的节点进行交互。

2、技术栈：前端（javascript, Web3.js），后端（Python, Django），智能合约（Solidity），区块链网络（Sepolia测试网, Alchemy）。

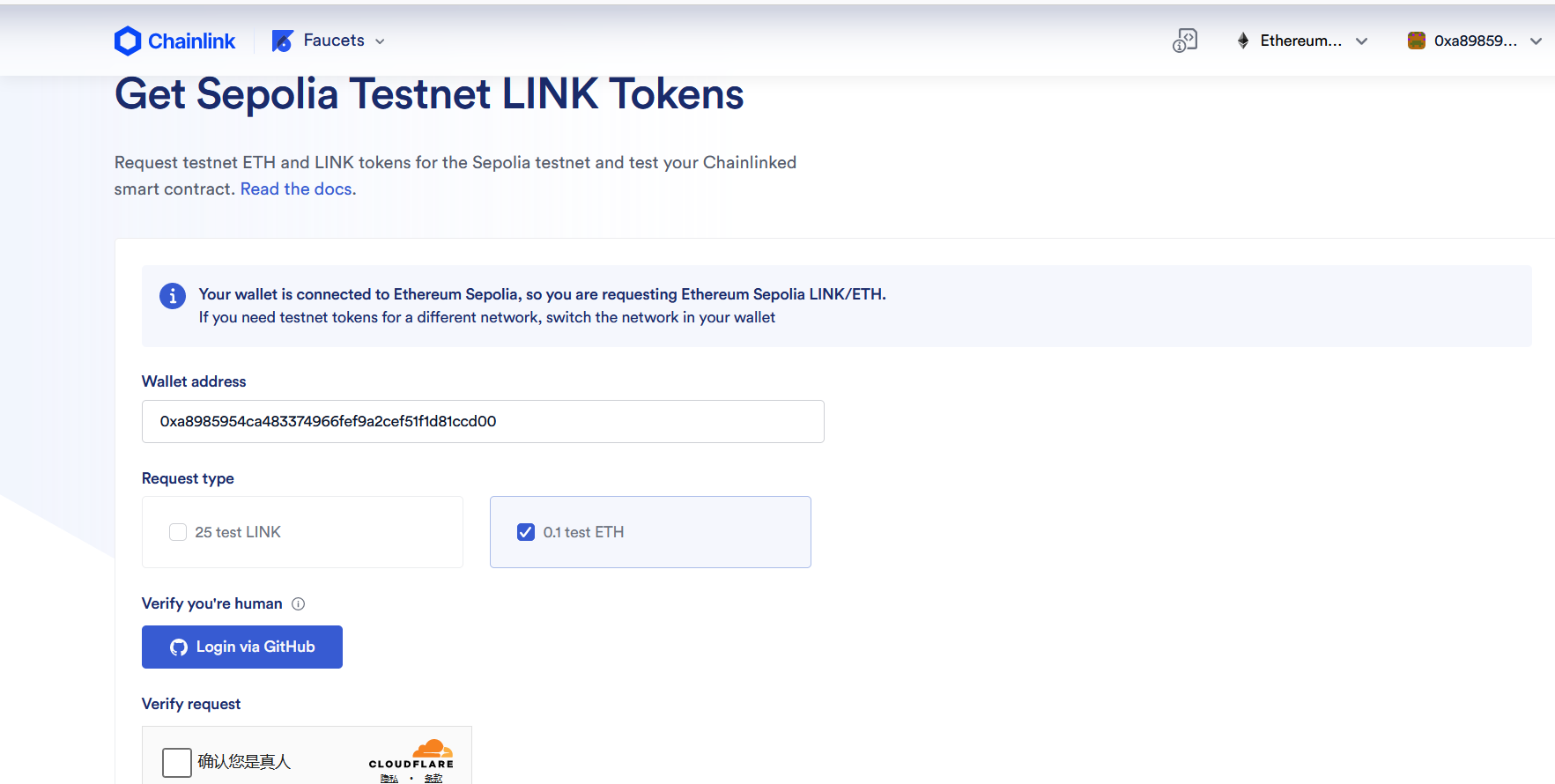
3、使用注意：

（1）部署合约与调用合约需要消耗小狐狸钱包中的测试币：

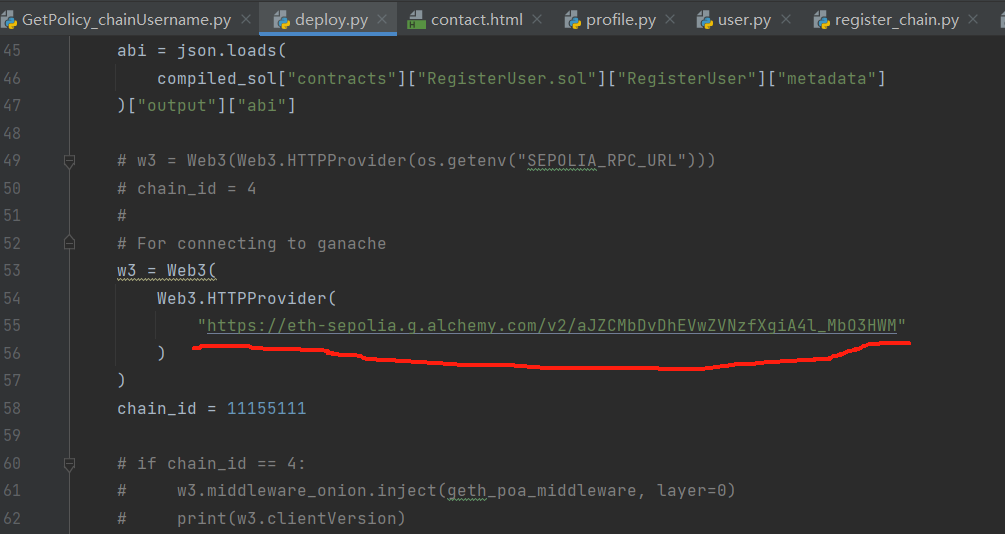


测试币在如下水龙头网址中可免费获取，一开始大概两天领取0.25，后面变成两天领取0.1：

<https://faucets.chain.link/sepolia>



（2）现在使用的节点是我个人创建的，想自己重新创建节点可注册登陆Alchemy，将sepolia测试网址<https://sepolia.etherscan.io/token/0x779877A7B0D9E8603169DdbD7836e478b4624789>输入其中，进行节点的创建，节点创建后返回的便是如下红线的链接。



**8.2.2安装与配置**

环境配置与部署：

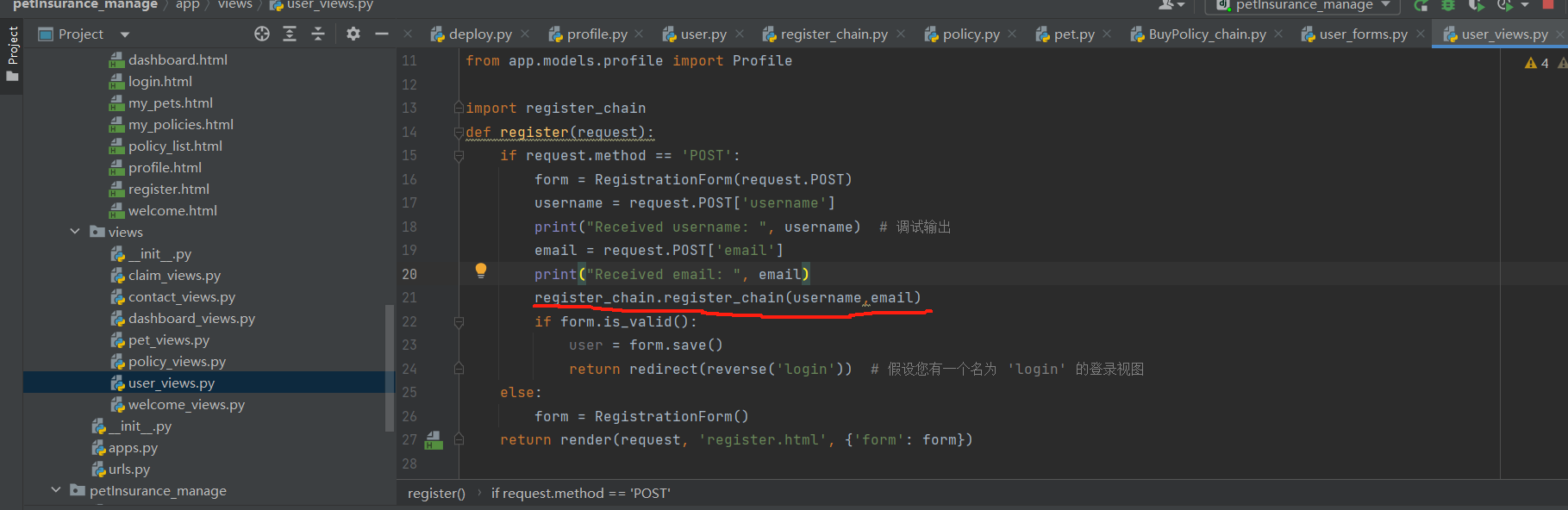
* 操作系统：windows10。
* 依赖包：web3,Python 3.11, solcx、django等。
* 插件：solidity、docker
* 克隆项目仓库：git@github.com:BlueEyeHeart/Blockchain-based-Pet-Insurance-System.git。
* 启动服务器：在petInsurance\_manage目录下运行项目

## **8.3 API文档**

1、接口列表：

* 用户注册：.venv/BlockChain/scripts/register\_chain.py，参数：\_username,\_email，(将用户名及邮箱部署到链上)

案例图：



* 购买保险：.venv/BlockChain/scripts/BuyPolicy\_chain.py，参数：\_username,\_petName,\_petSpecies,\_policyName,\_policyTermYears（将保险信息部署到链上）

案例图：



* 获取所购买的信息：.venv/BlockChain/scripts/GetPolicy\_chainUsername.py，参数：\_username，返回：\_username,\_petName,\_petSpecies,\_policyName,\_policyTermYears。

