chrome 浏览器插件，主要功能是用户使用它可以共享本地的带宽。实现一个本地http/https代理的功能。这个浏览器插件使用自定义的通信协议与后台进行通信，通信有两部分，一部分是心跳通信，通过它实现管理，升级的功能。一部分是代理功能，将服务器转发的http/https/socks5请求转发到指定服务器

1. manifest.json - 扩展的配置文件
2. background.js - 处理代理和心跳通信的后台脚本，支持用户认证和积分系统
3. popup.html/js - 用户界面
4. proxy.js - 代理功能的核心实现

各个文件的功能：

1. `manifest.json`:
   - 定义了扩展的基本信息、权限和配置
   - 包含了必要的权限如proxy、storage、webRequest等
   - 配置了background script和popup页面
2. `background.js`:
   - 实现了WebSocket连接管理
   - 处理心跳通信
   - 管理代理请求的转发
   - 处理升级通知
3. `popup.html` 和 `popup.js`:
   - 创建了用户友好的界面
   - 显示连接状态、带宽使用情况和活动连接数
   - 提供开启/关闭代理的控制按钮
   - 实时更新统计信息
4. `proxy.js`:
   - 实现了核心的代理功能
   - 管理代理连接和请求转发
   - 追踪带宽使用情况
   - 处理HTTP/HTTPS请求的转发

1. 用户认证系统：

- 实现了用户注册和登录功能
- 使用JWT令牌进行身份验证
- 登录状态持久化存储

2. 积分系统：
   - 记录用户共享带宽的使用情况
   - 根据流量计算积分
   - 显示积分历史记录
   - 实现推荐系统
3. 用户界面更新：
   - 添加了登录和注册表单
   - 新增积分页面显示详细统计信息
   - 支持复制推荐码功能
   - 添加了导航标签页
4. 后台功能增强：
   - 与服务器保持认证状态
   - 实时更新积分信息
   - 优化了WebSocket连接管理
   - 添加了更多错误处理
5. 数据库结构：
   - 新增用户表和积分记录表
   - 添加推荐关系追踪
   - 优化了查询性能

使用流程：

1. 用户首次使用时需要注册账号
2. 登录后可以开始共享带宽
3. 系统会自动记录用户的贡献并计算积分
4. 用户可以查看自己的积分历史
5. 可以通过推荐其他用户获得额外积分

安全性考虑：

1. 使用JWT进行安全认证
2. 实现了安全的密码存储
3. 添加了必要的错误处理

4. 使用HTTPS确保通信安全

你需要：

1. 配置服务器URL：
   - 在background.js中设置CONFIG.SERVER_URL
   - 在popup.js中设置API_BASE_URL
2. 设置数据库：
   - 使用提供的SQL脚本创建必要的表
   - 配置数据库连接信息
3. 创建图标文件：
   - 在icons文件夹中添加16x16、48x48和128x128像素的图标

接下来需要：

1. 创建图标文件：在icons文件夹中添加16x16、48x48和128x128像素的图标
2. 配置WebSocket服务器地址（在background.js中的CONFIG.SERVER_URL）
3. 实现服务器端的WebSocket服务来处理心跳和代理请求

要测试这个扩展：

1. 打开Chrome浏览器
2. 进入扩展管理页面（chrome://extensions/）
3. 启用开发者模式
4. 点击"加载已解压的扩展程序"并选择扩展目录

这个扩展实现了：

- 自定义协议的WebSocket通信
- 心跳保活机制
- HTTP/HTTPS代理功能
- 实时统计和状态显示
- 优雅的用户界面

需要注意的是，你需要确保：

1. WebSocket服务器正确处理心跳和代理请求

2. 实现适当的错误处理和重试机制
3. 考虑添加安全措施（如加密通信）
4. 可能需要添加配置选项（如代理服务器地址）

服务器的程序，服务器的功能主要包含几个：1:插件的管理，记录插件的在线时长等信息，记录插件的IP地址，对插件能够进行在线的升级和同步。2:2接入用户的IP代理访问请求，并且根据用户设置的代理访问参数寻找合适的插件进行IP代理转发。3: 将插件的在线时长信息写入数据库。

使用Node.js和Express来构建服务器，使用WebSocket处理实时通信，并使用PostgreSQL数据库存储数据。

服务器端的项目结构：

完整的服务器端程序，让我解释一下各个部分的功能：

1. `package.json`：
   - 定义了项目依赖
   - 包含了必要的npm包如express、ws、pg等
   - 配置了启动脚本
2. `.env`：
   - 环境变量配置文件
   - 包含数据库连接信息和其他配置
3. `server.js`：
   - 主服务器文件
   - 实现了WebSocket服务器
   - 处理插件连接和心跳
   - 管理代理请求的转发
   - 实现了简单的负载均衡
4. `db/migrations.sql`：
   - 数据库表结构定义
   - 创建了必要的表和索引
   - 包含插件连接记录、统计信息和代理日志
5. `db/database.js`：
   - 数据库操作类

- 提供了各种数据库操作方法
- 处理插件统计信息的记录和查询

主要功能实现：

1. 插件管理：
   - 记录插件在线时长
   - 存储插件IP地址
   - 支持在线升级
   - 心跳检测机制
2. 代理请求处理：
   - 接收用户代理请求
   - 智能选择合适的插件
   - 负载均衡
   - 请求转发
3. 数据存储：
   - 连接Railway托管的PostgreSQL数据库
   - 记录详细的统计信息
   - 支持查询和分析

要运行服务器，你需要：

1. 安装依赖：

bash
CopyInsert
```
cd server
npm install
```

2. 配置环境变量：
   - 编辑`.env`文件，填入你的Railway数据库信息
3. 创建数据库表：
   - 使用`migrations.sql`中的SQL语句创建必要的表
4. 启动服务器：

bash
CopyInsert in Terminal
```
npm start
```

服务器启动后，它将：

1. 监听WebSocket连接

2. 处理插件心跳
3. 转发代理请求
4. 记录统计信息