软件配置与运维文档

1. 配置管理

1.1 配置项清单

- `package.json`:元数据文件，一般用来定义项目所需的依赖、脚本、版本信息等
- `setting.py`：Django 项目的核心配置文件
- `requirement.txt`：列出Django 项目所需的 Python 包及其版本

1.2 配置文件位置

- `package.json` 位于 `front_end` 项目根目录下
- `setting.py` 位于 `back_end_and_database\your_project_name\your_project_name` 路径下
- `requirement.txt` 位于 `back_end_and_database` 项目根目录下

1.3 配置说明

- `package.json`
  - `scripts`: 定义了可运行的脚本命令，常用于自动化任务
  - `dependencies`: 项目运行时所需的依赖包
  - `devDependencies`: 项目开发时所需的依赖包
- `setting.py`
  - 定义了项目的基本设置、应用程序、数据库、国际化、静态文件处理等关键配置
  - 主要修改 `DATABASES` :数据库配置，配置PostgreSQL 数据库信息
- `requirement.txt`
  - `asgiref`：ASGI（Asynchronous Server Gateway Interface）规范的参考实现
  - `django-cors-headers`：Django 项目的跨域资源共享（CORS）支持
  - `djangorestframework`：Django REST framework，用于构建 RESTful API
  - `psycopg2`：PostgreSQL 数据库的 Python 客户端
  - `setuptools`：Python 包安装和分发工具
  - `Django`:Django Web 框架，用于构建 Web 应用程序
  - `wheel`:Python 包的二进制分发格式
  - `sqlparse`:SQL 解析器，用于解析和格式化 SQL 语句
  - `tzdata`:时区数据，用于处理时区相关的功能

2. 版本控制

2.1 代码仓库

- GitHub 仓库地址：https://github.com/JerryZ2002/SoftwareEngineeringGroupProject/tree/main/src

2.2 分支策略

- **主分支（main）**：用于存放稳定版本的代码。
- **开发分支（develop）**：用于日常开发，功能完成后合并到主分支。

```
# 创建 develop 分支
git branch develop
# 推送 develop 分支到远程仓库
git push origin develop
# 切换到develop分支
git checkout develop
# 日常开发
git pull origin develop
...
git push origin develop
# 合并分支
git checkout main
git merge develop
```

3. 持续集成

3.1 自动化测试用例缺失情况

由于项目仅使用黑盒测试且手动测试功能，因此在持续集成过程使用：

- **手动测试**：开发者需要在每次提交代码前进行手动测试，包括功能测试、回归测试等
- **代码审查**：强制进行代码审查，让其他团队成员检查代码质量，发现潜在的问题和错误，提高代码质量。

3.2 持续集成流程

本项目的持续集成流程主要包括以下步骤：

1. **代码提交**：开发者完成代码编写后，将代码提交到版本控制系统（使用 Git）的develop分支。
2. **手动测试**：开发者在提交代码前进行手动测试，确保代码符合功能要求、质量标准等。
3. **代码审查**：提交代码后，其他团队成员进行代码审查，发现潜在问题并提出改进建议。
4. **代码集成**：经过代码审查通过后，将代码集成到主干分支，进行手动集成测试，验证代码的兼容性和稳定性。
5. **部署个人电脑**：本项目未统一服务器部署应用，因此由各成员手动部署代码到各自电脑环境中

4. 部署和运维计划

4.1 基本环境

- Python 3.9+，Node.js(参考v20.13.1), PostgreSQL15.*

4.2 本地部署

1. 克隆代码仓库

```
git clone git@github.com:JerryZ2002/SoftwareEngineeringGroupProject.git
```

2. 进入项目目录

```
cd src
```

3. 安装依赖

- **前端**

```
cd front_end
npm install
npm run dev
```

- **后端**

```
cd back_end_and_database
conda create -n new_env python=3.9
pip install -r requirements.txt
cd your_project_name
python manage.py runserver
```

4.3 配置数据库

使用 PostgreSQL 作为数据库。安装 PostgreSQL 并创建数据库。

1. **创建数据库**:

   ```
   createdb runoobdb
   ```

2. **应用数据库迁移**:

   ```
   python manage.py migrate
   ```

4.4 运行

- **前端**

```
cd front_end
--- # 开发环境
npm run dev
--- # 生产环境
npm run build
cd dist
npm install -g http-server
http-server
```

- **后端**

```
cd back_end_and_database/your_project_name
activate new_env
python manage.py runserver
```

4.5 运维计划

1. **更新依赖**：定期更新项目依赖项，以获取最新的功能和安全补丁

```
pip install --upgrade -r requirements.txt
npm update
```

2. **数据备份**：定期备份数据库，以防数据丢失

```
pg_dump runoobdb > backup.sql
```

3. **日志记录**：定期查看 Django 和 Vue 的日志文件，以便及时发现和解决问题
4. **未来计划**：逐步引入自动化测试，减少手动测试的工作量，提高代码质量。