项目书

项目名称	TinyEditor 支持协同编辑
项目导师	vaebe
申请人	曹里林
邮箱	yinlin124@gmail.com

1. 个人背景

个人完成过商业小程序七号球鞋(React)、宠物商店(Vue) 等项目，对前端基础技术栈熟悉。(详情可见个人简历部分)。

目前在月新增 10w+ app 前端部门实习，对前端技术感兴趣，有一定的编码能力，熟练使用 git 协同，了解实际项目工作流。

2. 项目分析

2.1 项目要求

完成 TinyEditor 富文本编辑器的协同编辑功能前后端开发，极短时间内相应用户操作，并有离线保存功能以及文档持久化功能。完成项目测试以及文档。

2.2 技术实现

结合前端结合 `yjs`、`y-quill` 、`y-websocket` ，后端部分为 `y-websocket` 为核心实现协同开发功能。此套方案利用现有的开源框架 `Yjs` (`Yjs` 是一个高性能的 CRDT 框架，用于构建自动同步的协作应用程序)，完全符合项目要求，且为 `js` 原生实现符合 `TinyEditor` 理念，与框架无关。

2.2.1 协同实现

利用 `y-quill` 适配器将 `TinyEditor` 数据绑定到 `Yjs` 定义的数据结构 `Y.Doc` 上，后续本地或服务器有更新会修改此数据结构。`Y.Doc` 和服务器的通信使用 `y-websocket`，将本地编辑器信息和服务器同步

```JS
const ydoc = new Y.Doc();
const provider = new WebsocketProvider(
    'wss://demos.yjs.dev/ws',
    'OC-demo-YL',
    ydoc
);

const ytext = ydoc.getText('quill');
const awareness = provider.awareness;

new QuillBinding(ytext, quill, awareness);
```

2.2.2 用户感知

利用 `yjs` 中 `Awareness` 模块，将 `Tiny` 的光标模块开启后即可对应展示不同用户信息的状态。`provider. awareness` 是 Yjs 的一个独立模块，负责同步非核心文档数据，例如用户的在线状态、光标位置、姓名、颜色等。

```JS
quill = new TinyEditor(editorRef.value, {
    theme: 'snow',
    modules: {
        cursors: true,
        toolbar: TOOLBAR_CONFIG,
    },
});

const awareness = provider.awareness;
new QuillBinding(ytext, quill, awareness);
```

2.2.2 离线支持与本地持久化

`IndexeddbPersistence` 利用浏览器内置的 `IndexedDB` 数据库，将 `Y.Doc` 的状态本地存储在客户端。

```JS
const persistence = new IndexeddbPersistence('OC-demo-YL', ydoc);
```

2.2.3 用户处理加入和退出协同编辑等情况

利用 `yjs` 现有监听模块

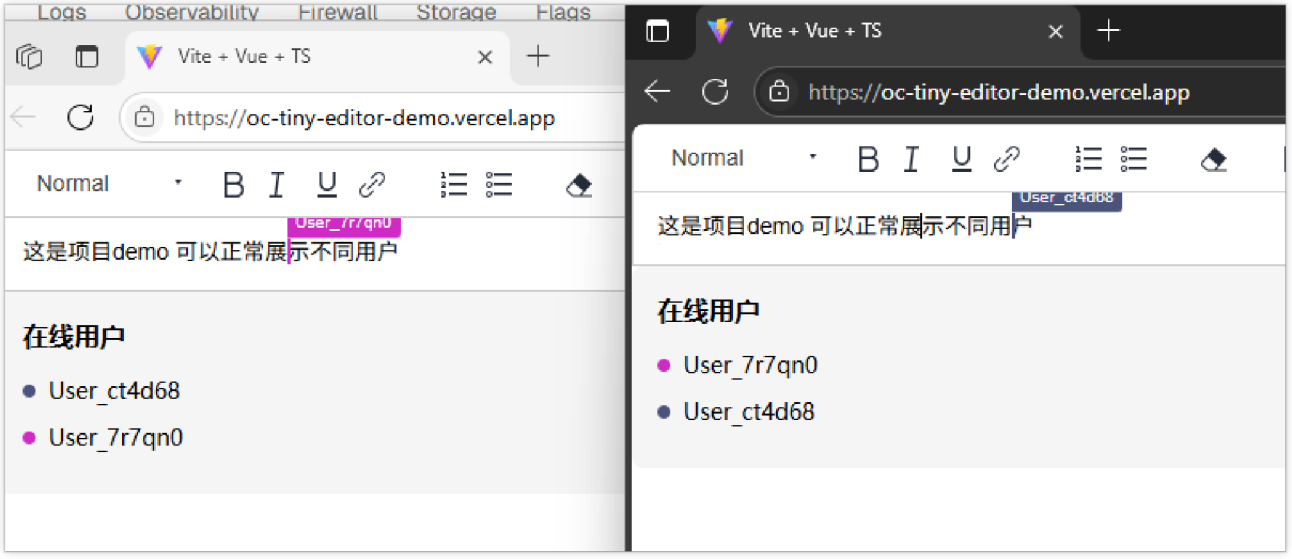
```
provider.on('status', callback); //  监听 WebSocket 连接的状态变化
awareness.on('update', callback) //  当任何客户端的 awareness 状态发生变化时回调（包括本地状态变化，以及其他客户端的状态变化）。
```

离线状态下也能正常处理

```JS
persistence.once('synced', callback) // 首次从 IndexedDB 加载完毕后触发
persistence.on('synced', callback) // 设定不同状态下回调
```

2.2.4 Demo 实现演示

利用以上技术配合 `TinyEditor` 搭建简易 Demo(链接在下)



> [Demo]

综上所述，`Yjs` 框架及其模块化生态系统与我们项目的核心需求高度契合。其出色的实时同步性能、原生的离线支持、以及灵活的扩展能力，确保项目能够满足所有功能和非功能性需求。

2.2.5 与后端服务的接口

结合现有 `y-websocket` 让其有持久化能力并与主服务器(负责原有的权限、创建编辑器文件等管理)协同工作。

```
const wss = new WebSocketServer({ port: 1234 });

// 初始化 MongoDB 持久化实例
const mdb = new MongodbPersistence('mongodb://localhost:27017/yjs-
docs', {
    collectionName: 'documents',
});

wss.on('connection', (conn, req) => {
    setupWSConnection(conn, req, {
        persistence: mdb, // 传入持久化实例
    });
});
```

主应用后端不直接处理 `Yjs updates`。它负责提供文档的“最新版本”给 y-websocket，并在 y-websocket 持久化文档后，可以获取或触发相关操作，便于后续用户扩展功能。

3. 项目开发时间计划

阶段	时间段	关键任务
前期准备	6.9-7.15	学习 `yjs` 官方文档和 `y-websocket` 源码，完成技术选型，构建出关键功能继承 demo
项目开发	7.16-8.20	完善 TinyEditor 与 Yjs 的双向绑定(包括 Tiny 中的各种扩展)，确保所有编辑操作的流畅同步。实现动态文档 ID 管理。集成 QuillCursors, 提供 Cursors 自定义显示信息等相关接口。强化离线保存机制的稳定性和用户体验。
项目进阶	8.21-9.10	优化状态监听，对连接断开、错误等情况给出用户友好提示。增强用户加入/退出协同会话的界面反馈,完善 TypeScript 类型声明。
项目验收	9.11-9.30	执行全面的功能测试（实时同步、离线、用户感知），确保系统稳定可靠。撰写详细的项目技术实现文档（突出 Yjs 各模块应用）、使用指南和部署说明。准备项目演示，突出协同编辑的核心功能和技术亮点。