Documentation for Merging Custom Changes in Open-WebUI with the Latest Updates

This document outlines the process and considerations for integrating custom modifications made to Open-WebUI (open-webui-test-1) with the latest updates from the upstream Open-WebUI (open-webui-github-1) repository. It provides a structured approach to ensure seamless merging and preservation of existing functionality while incorporating new features and improvements.

This document also captures the step-by-step debugging and resolution process for issues encountered while merging and integrating features in the open-webui-test-1 project. The primary focus was on fixing API route handling for CasePrediction and LegalSearch endpoints, resolving frontend/backend connectivity issues, and ensuring smooth application functionality.

Server Implemented on:
(ec2-13-236-50-121.ap-southeast-2.compute.amazonaws.com)

(Site:http://13.236.50.121:5173) - Frontend
(Site:http://13.236.50.121:8080/) - Backend

1) Initial Setup and Background:

- **The project (open-webui-test-1) is based on SvelteKit for frontend and FastAPI for the backend.**

- **Features involved:**

- **LegalSearch: A search interface utilizing legal embeddings.**
- **CasePrediction: A predictive model API integrated with HuggingFace.**

2) Merge Context:

- **Merging features from the "open-webui-test-1" branch with "open-webui" from GitHub.**

- **Create a Backup:**

  If you're concerned about potential issues during the merge, create a backup of both directories:

- **cp -r ~/open-webui-test-1 ~/open-webui-test-1-backup**
- **cp -r ~/open-webui-github-1 ~/open-webui-github-1-backup**

Step 1: **Ensure Clean Working Directories:**

- **Navigate to the open-webui-test-1 folder to ensure there are no uncommitted changes:**

  cd open-webui-test-1/
  `git status`

- **If there are uncommitted changes, either commit or stash them:**

➔ **To commit:**

git add .
git commit -m "Save work before merging latest changes"

➔ **To stash:**

git stash

- **Repeat the same check for open-webui-github-1:**

  cd open-webui-github-1
  `git status`

  Step 2: **Add open-webui-github-1 as a Remote**

- **You need to pull the latest changes from the original GitHub repo (open-webui-github-1) into open-webui-test-1.**

- **Navigate to the open-webui-test-1 directory:**

  cd open-webui-test-1

- **Add the open-webui-github-1 as a remote:**

  git remote add upstream ~/open-webui-github-1

- **Verify the remotes:**

  git remote -v

➔ **You should see an entry for upstream pointing to ~/open-webui-github-1.**

Step 3: Fetch the Latest Changes from open-webui-github-1

- **Fetch the latest changes from the open-webui-github-1 repo:**

  git fetch upstream

- **Check out your current branch in open-webui-test-1 (which is main)**

  git checkout main

- **Merge the latest changes from open-webui-github-1 into your branch:**

  git merge upstream/main

- **Resolve any merge conflicts that arise. Use a merge tool or manually edit conflicting files. Once resolved:**

  git status (Conflicting files will be marked as "both modified")

  git add <file_with_conflicts>

  git commit

- **For each conflicting file, run the following commands to accept changes only from upstream/main (i.e., open-webui-github-1):**

- **To keep only the changes from open-webui-github-1 for a specific file:**

  git checkout --theirs <file_with_conflict>

  Replace <file_with_conflict> with the actual filename.

- **Once resolved, mark the file as resolved:**

  git add <file_with_conflict>

- **If you want to resolve all conflicts by keeping only the changes from open-webui-github-1:**

  Keep theirs (changes from upstream/main) for all conflicts:

  git checkout --theirs .

  Add all resolved files:

  git add .

git commit -m "Merged upstream/main, keeping latest changes from open-webui-github-1"

Step 4: Test the Merged Code

- **Build and run the application in open-webui-test-1 to ensure everything works correctly with the latest changes:**

- **cd open-webui-test-1:**
- **sudo npm i**
- **sudo npm ci**
- **sudo npm run build**
- **sudo npm run dev**

- **cd open-webui-test-1/backend:**
- **pip install -r requirements.txt**
- **bash start.sh**

Step 5: Push the Merged Changes to Your Repository

- **Once the merge is complete and tested, push the changes to your remote repository:**

git push origin main

Step 6: To verify if the Merge is completed or not

- **Check the Git Log: Run:**

git log --oneline upstream/main

- **Look for a merge commit that mentions the Merge branch 'main' of upstream.**
- **Ensure a Clean Working Directory: Run:**

git status

The output should say:

On branch main

nothing to commit, working tree clean

➔ **If both checks confirm no pending actions, the merge is complete.**

Step 7: Check and Fix Permissions for the Project Directory

sudo chown -R ubuntu:ubuntu ~/open-webui-test-1

- **If the Issue Persists:**

  rm -rf .svelte-kit build dist

  rm -rf node_modules

  npm install

➔ **Run the Build Again**

3) **Permission Errors During Build & Issues that we faced:**

   Observations:

   Build failed with errors related to directory permissions:

   Error: EACCES: permission denied, rmdir 'build/_app'

   Actions Taken:

- **Checked Directory Permissions:**
- **Verified ownership:**

  ls -ld build build/_app

- **Updated permissions:**

sudo chown -R ubuntu:ubuntu build

- **Increased Node.js Heap Size:**

  Exported NODE_OPTIONS to increase memory:

- **export NODE_OPTIONS="--max-old-space-size=4096"**
- **source ~/.bashrc**

- **Rebuild the Application**

  There

  We also noticed that when routing to /legalsearch and /caseprediction they both have their respective +page.svelte to render their page but The layout behind "legalsearch" and "caseprediction" route (the dark background, sidebar, and other UI elements) is coming from a parent layout file or a global layout that wraps our +page.svelte in the /src/routes/(app)/ folder structure which hinders the rendering of /legalsearch and /caseprediction.

  In SvelteKit, layout inheritance allows you to define reusable layouts at higher levels of the routing hierarchy. Let me explain how this works and where this layout might be defined.

- **What Controls the Layout?**

- **Parent Layout File:**

  In our case, the layout is defined in src/routes/(app)/+layout.svelte.

  This layout wraps all child pages within the (app) group, including /legalsearch and /caseprediction.

- **Sidebar or Navigation:**

  If there's a persistent sidebar or navigation, it's part of the +layout.svelte file located in src/routes/(app) or even higher (e.g., src/routes/+layout.svelte).

- **How to Check Which File Controls the Layout**

  Look for +layout.svelte:

➔ **Navigate to src/routes/(app) and open the +layout.svelte file:**

Add a simple Debug: Parent Layout inside +layout.svelte and reload your browser to confirm the file controlling the layout.

We created a route-specific layout for the /testing route to test its functionality. Within the (app) directory, there is a dedicated +layout.svelte file and a +page.svelte file, which act as parent components for the routes under this directory. Additionally, in the routes directory, there is a global +layout.svelte file serving as the overarching parent component, with the other components functioning as its child components.

In the newer version of open-webui, the rendering of child pages like legalsearch/+page.svelte depends on the logic and components defined in parent +layout.svelte files. Ensure that the legalsearch page is compatible with the parent layout's structure and dependencies.

In the updated version of open-webui, routing conventions may have been updated to support nested layouts or stricter route definitions. Verify that the legalsearch route is correctly placed within the directory structure and adheres to the latest routing standards. Align the page code with any changes in components, props, or APIs introduced in the newer version.

- **To Apply the Stashed Changes (That is any of the changes that you made in the respective file) into your remote server which is open-webui-test-1**

- **Reconnect to the Server Log in to your remote server and navigate to your project directory:**

  cd open-webui-test-1

- **Check Your Current Branch Confirm that you're on the correct branch (main):**

  git branch

  If you're not on main, switch to it:

  git checkout main

- **View Your Stash List Double-check which stash you need to apply (recommended to only use the latest stash):**

  git stash list

➔ **Your list should still look like:**

stash@{0}: WIP on main: ca21ccc3c Merge remote-tracking branch 'upstream/main'

stash@{1}: WIP on main: ca21ccc3c Merge remote-tracking branch 'upstream/main'

stash@{2}: WIP on master: a567efb commit

- **Apply the Most Recent Stash:**

git stash apply stash@{0}

Note: Only Apply Stash Once: Once you've applied a stash, you don't need to apply it again unless you re-stash it.

- **Stage Your Changes Add the changes to the staging area:**

  git add .

- **Commit Your Changes**

  git commit -m "Describe your changes here"

- **(Optional) Push Your Changes to Remote**

  git push origin main

- **After committing, check your Git status to confirm your working directory is clean:**

  git status

  You should see:

  On branch main

  nothing to commit, working tree clean

  Note: Backup folders for open-webui-test-2 and open-webui-test-3 the Older version of open-webui is also backed up on the server mentioned above

  In the old open-webui-main server there was an issue we faced nothing related to the latest open-webui-test-1 but if incase someone faces it follow these steps:

❖ **Resolution of Permission and Port Configuration Issues in Open-WebUI:**

During the integration process, a user encountered a permission issue that restricted their ability to log in and access necessary privileges. To resolve this, Saif and I performed the following steps:

- **Database Cleanup:**

  We navigated to the data directory and deleted the following files:

➔ **data.db**
➔ **webui.db**
➔ **chroma.db**

- **Backend Initialization:**

  After confirming that the data directory was correctly mounted in the volumes section of the docker-compose-dev.yaml file, we ran the bash start.sh script within the backend directory. This allowed the application to start fresh, enabling the user to log in and access the required privileges.

- **Frontend Port Configuration:**

  Inside the docker-compose-dev.yaml file, we noted that the frontend was configured to run on port 3000. We updated the port configuration from 8080 to 3000. However, when accessing the application on port 3000, the page displayed a spinner and did not load properly. Meanwhile, running the backend using bash start.sh on port 8080 rendered the application fully functional with all features intact.

- **Troubleshooting Rendering Issues:**

  We attempted to resolve the issue by taking the original Svelte files from the open-webui repository and integrating them into open-webui-main. Despite running the necessary Docker commands, the changes were not reflected on either port 3000 or 8080. However, port 8080 continued to function correctly, rendering the application with all features.

- **Ibrahim Sultan**