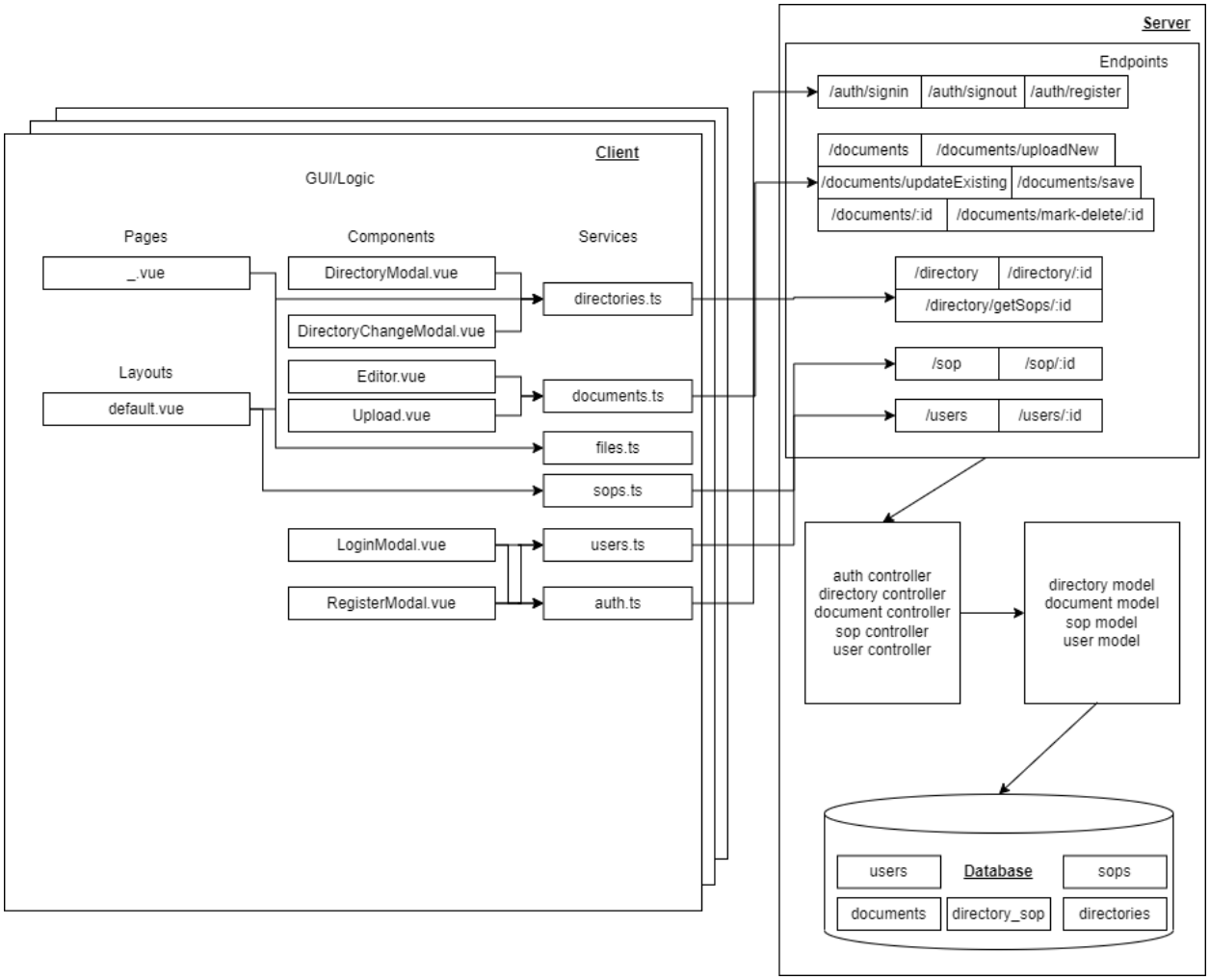
Block diagram



GUI (main features): When a visitor enters the website, they can see the sidebar. This is represented by default.vue. They can register an account if they need to (RegisterModal.vue), or if they have an account, they can sign in (LoginModal.vue). The first action uses the endpoint /auth/register and the second uses /auth/signin. Whether signed in or not, a user can view all the directories and SOPs. These are gathered from /directory and /directory/getSops. When a user clicks an SOP, its contents are fetched from /files, where Express.js serves the files statically to the frontend page _.vue. A signed in user can create directories from DirectoryModal.vue, which will call /directory. Note this is a POST request. A signed in user can upload documents through the Upload.vue component, which calls on /documents/uploadNew. Alternatively, they can create a document without uploading too. Additionally, users can either edit an existing document or directly upload a new version (up-version) inside Editor.vue, calling /documents/save and /documents/updateExisting respectively. They can mark a document version for deletion using /documents/mark-delete. Apart from that, another feature signed in users can do is move SOPs between directories, which is activated through DirectoryChangeModal.vue, and this calls /sops/changeDirectory. Admins have the ability to delete document versions and delete a whole SOP.

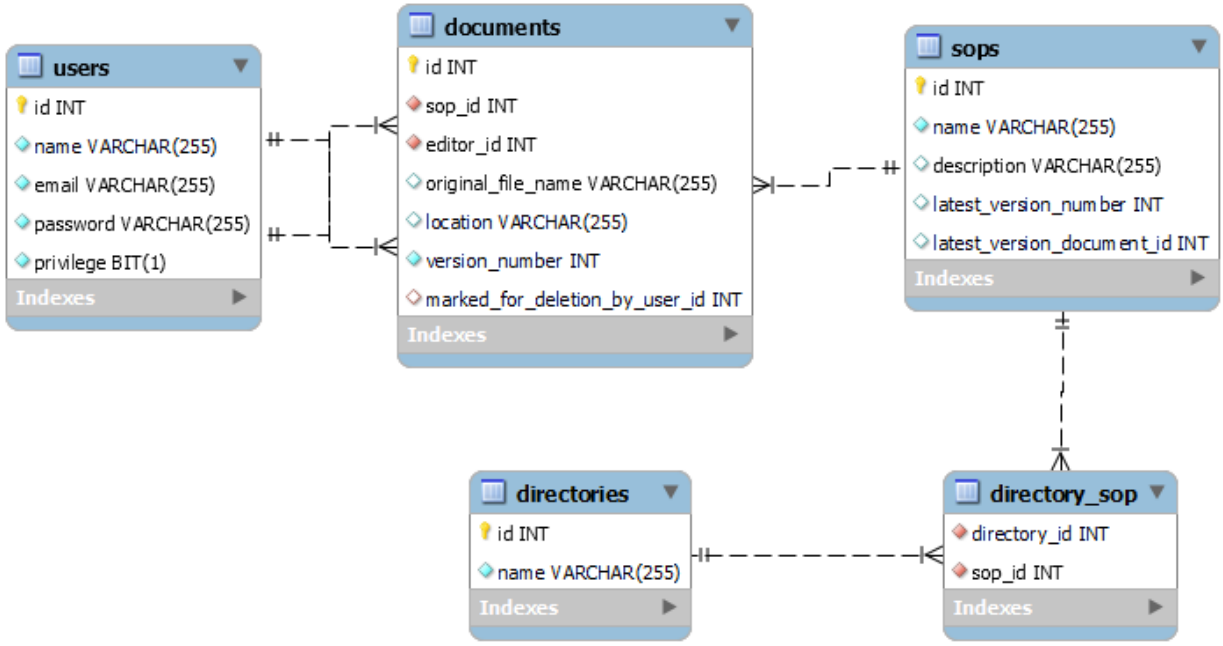
Server Side (main routes):

- /auth/preregister takes an email to preregister
- /auth/register takes a name, email, and password as credentials
- /auth/login takes an email and password, and this password becomes encrypted using the bcrypt library; the password is not encrypted as it is sent to the backend though
- /documents will return all documents or a specific one is specified with an id
- /documents/mark-delete will mark a document version for deletion and keep track of the user who marked it
- /documents/uploadNew requires a file and directory. It first creates an entry in the sop table. When an SOP entry is created, we create a sub-directory for future documents/revisions of that SOP to reside. Specifically, this location is ${STORAGE_DIR}/${directoryName}/${sop_id}. Additionally, an entry in the directory_sop table is created. Next, we create an entry in the documents table. This involves using pandoc to convert the file to .html and storing the files at the directory mentioned earlier.
- /documents/updateExisting will create a new document at the SOP's directory, create a new entry in the documents table, and update the SOP's latest version.
- /documents/save is responsible for updating a document when its content is edited from Quill. This creates a new document at the respective directory, creating a new entry in the documents table, and updating the SOP's latest version.

- /sops is used for returning an SOP given an id. It can also return all SOPs.
- /sops/changeDirectory will change the directory of the SOP. First, it moves the directories like so: ${STORAGE_DIR}/${old directory}/${sop_id} → ${STORAGE_DIR}/${new directory}/${sop_id}. Then, the directory_sop table is updated for that SOP. Then, for all documents with that SOP id, we update the location column in the documents table.
- /directory can list a directory or create a directory, which all it takes is a name. /directory/getSops will return all SOPs inside that directory
- /users has several methods. It can be used to create a user, list all users, or update a user

Database: We have a users table. An important non-trivial column is privilege. It is a bit. 0 means not admin and 1 means admin. We use this for admin capabilities: deleting an SOP version, deleting an SOP entirely, granting registration access to a user. We have a directories table so that we can easily keep track of where documents are, being able to list them and change where they belong. An SOPs table is useful because it is what is displayed on the sidebar. An important column is latest_version_document_id. Its purpose is self-evident. Another table we have is directory_sop. This table manages the special relationship that directories have sops and sops have directories. Specifically, this allows us to get the SOPs from a directory. The final table we have is the documents table. This contains important information like the SOP id that it belongs to, the editor/user who edited that document, the location of the document, and the version number. This allows us to have records of metadata for the document and to be able to control where documents are located and their relationship to an SOP.

DB Schema


users
id INT
name VARCHAR(255)
email VARCHAR(255)
password VARCHAR(255)
privilege BIT(1)
Indexes
documents
id INT
sop_id INT
editor_id INT
original_file_name VARCHAR(255)
location VARCHAR(255)
version_number INT
marked_for_deletion_by_user_id INT
Indexes
sops
id INT
name VARCHAR(255)
description VARCHAR(255)
latest_version_number INT
latest_version_document_id INT
Indexes
directories
id INT
name VARCHAR(255)
Indexes
directory_sop
directory_id INT
sop_id INT
Indexes