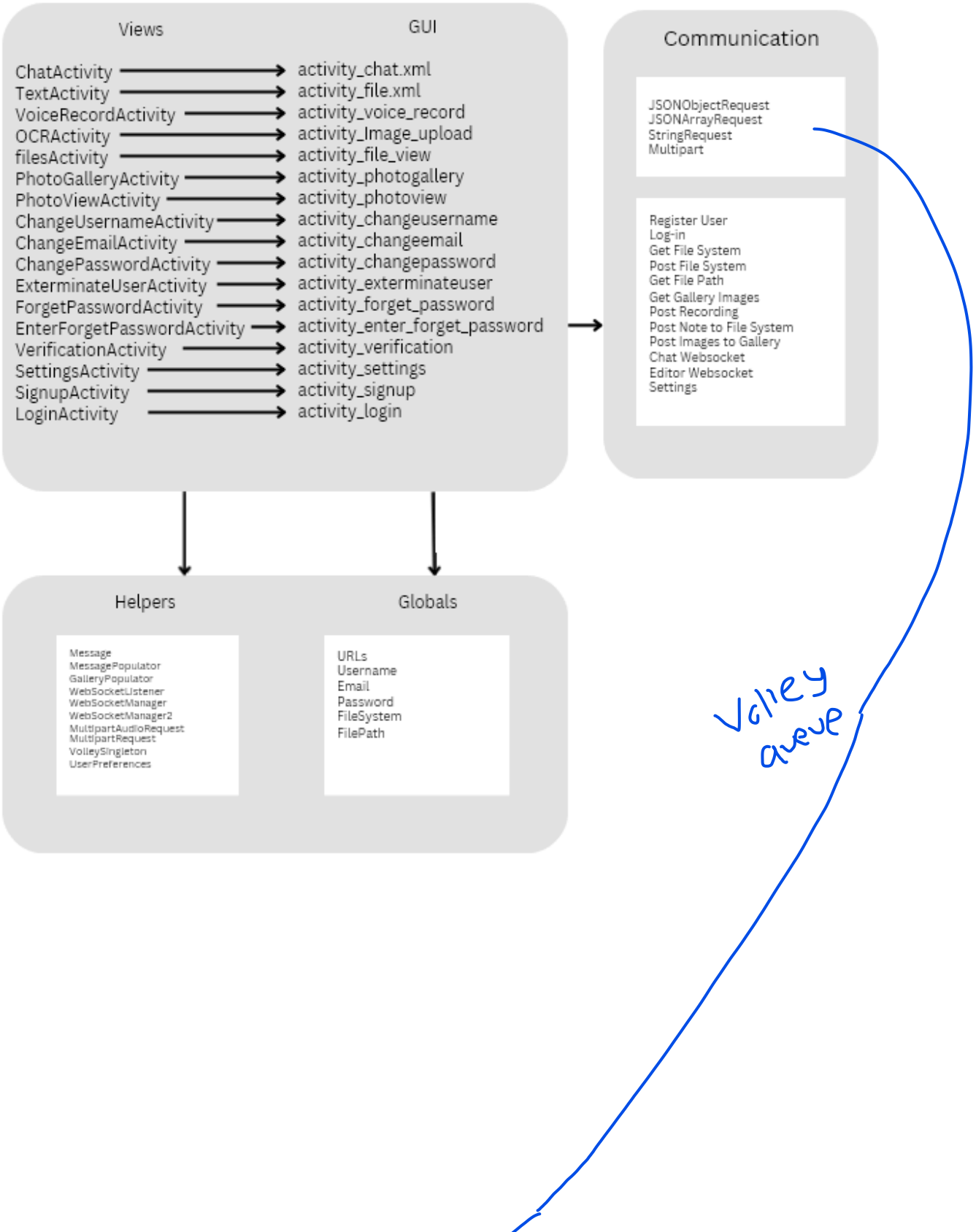
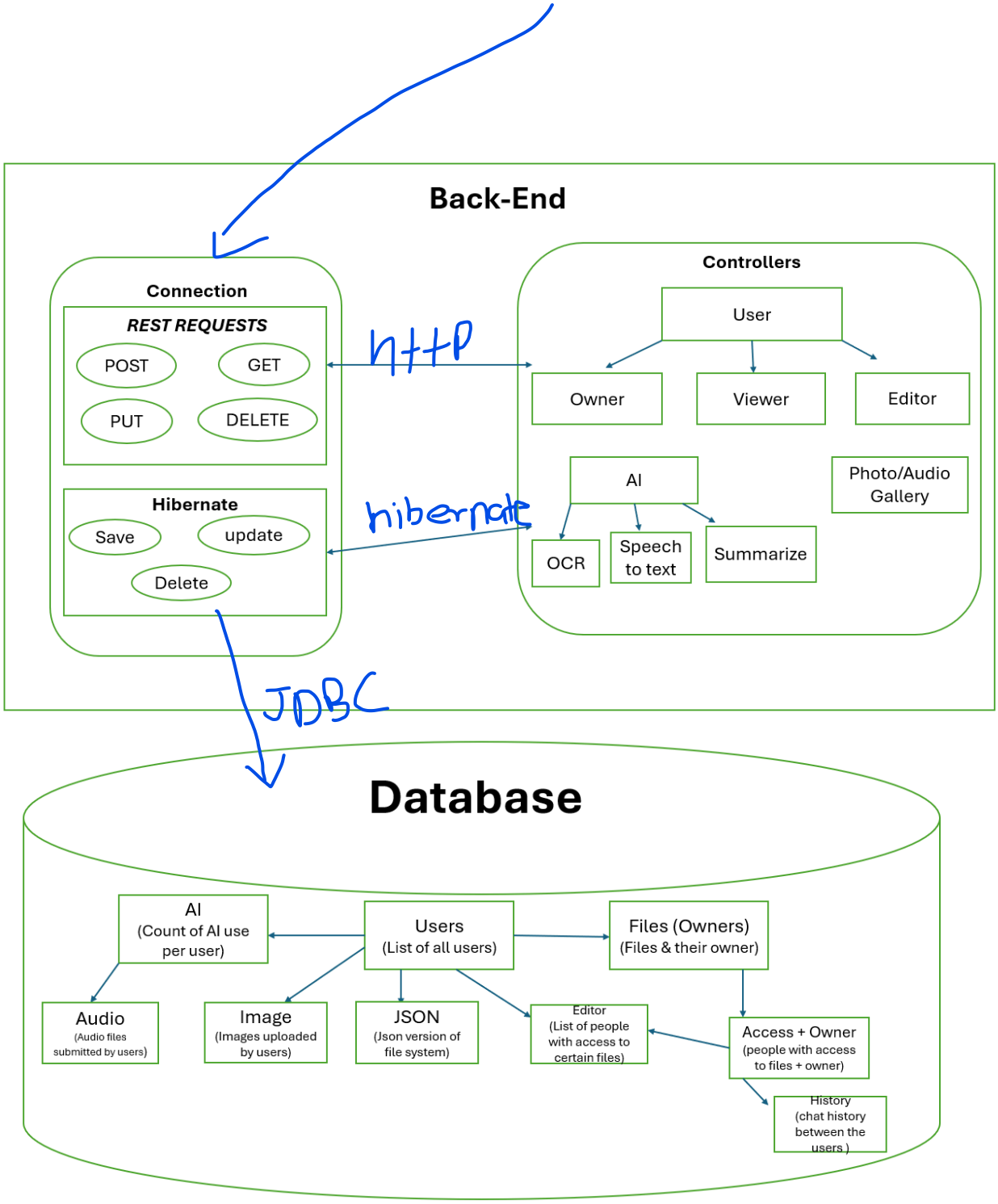
BLOCK DIAGRAMS

4_rasel_2: Naveen Prabakar, Yi Yun Khor, Nicholas Wang, Jamey Nyugen

FRONT END PICTURE GOES HERE:




Back-End
Connection
REST REQUESTS
POST
GET
PUT
DELETE
Hibernate
Save
update
Delete
http
hibernate
Controllers
User
Owner
Viewer
Editor
AI
OCR
Speech to text
Summarize
Photo/Audio Gallery
JDBC
Database
AI
(Count of AI use per user)
Users
(List of all users)
Files (Owners)
(Files & their owner)
Audio
(Audio files submitted by users)
Image
(Images uploaded by users)
JSON
(Json version of file system)
Editor
(List of people with access to certain files)
Access + Owner
(people with access to files + owner)
History
(chat history between the users)

Front-end Doc:

Login:

- A screen activity that takes user input, email and password, and upon valid input, takes the user to the main menu. (Files Activity)
    - EditText: Username (email)
    - EditText: Password
    - Button: Login
    - Button: Signup
    - Button: Forgot

FilesActivity:

- Serves as the apps main home page. From here, users can create new files, aka *notes*, or scroll to see their already existing notes, which could also be sorted into folders. Users can share individual notes to other users to see. A user can also access the OCR scanner here.
    - EditText: New Folder Name
    - EditText: Share To: username
    - Button: New Folder
    - Button: New File
    - Button: Delete (a file)
    - Button: Share
    - Button: OCR Scanner

TextActivity:

- Serves as the main view activity for each note. A user(s) can type their notes here, while also being able to ask a prompt to an AI and return a summarized answer. From here, they can also access the voice recorder, which would let them record any speech needing to be transcribed, and save that transcription as a note. They can save the note under any name they wish.
    - EditText: editor - the display that the user interacts with when they type
    - EditText: mainText - used to display the post-rendered text of markwon
    - Markwon: markwon editor used for rich text
    - EditText: input prompt here (for AI)
    - TextView: aiText
    - Button: Summarize (the given prompt for AI)
    - Button: Voice – to be sent to the voice activity
    - Button: Live Chat – to be sent to a chatbox on the specific note

Additional Features:

- Photo Gallery
    - Keeps hold of all pictures submitted, and can parse through the content of each picture and create a new note on the contents transcribed.
    - Uses the OCR Activity to upload new pictures.
- Voice Recorder
    - Can use the microphone of the device to record and has the ability to transcribe speech to text. The transcription can also be used to create a new note.
- Settings

Backend- Doc:

- **Communication :**
  - The backend usings various REST-API mappings to communicate effectively with the front end regarding several features & tasks
  - POST : Post is used whenever something new is created such as a new entry into the database. (For example, signing up, creating a file would all be posts requests in our code)
  - GET : Get is used whenever we are need to retrieve something from the database and sent it to the front end to display or for other purposes needed (An example would be getting the an already created file)
  - PUT: Put is used whenever we are updating an entry that already exists in the table. (An example would be updating a document and saving it)
  - DELETE: Delete is used whenever something is being removed from the database, particularly a record (An example would be deleting a profile or deleting a document)

- **Controllers:**
  - SignUp : The controller that keeps track of all the people that have signed up for the app. SignUp has one to many relationships with other controllers such as Files, Access, Image, Audio, & AI.
  - Login: The controller that keeps track of the people that login into the app. Login uses the SignUp table which can be accessed through the email and password. If the user forgot the password, we will send the email to their email for verification code and reset the password.
  - Files: The controller keeps track of the names of the files and the owner of the file. It holds a many to many relationship with the Access table.
  - Access: The controller keeps track of the ids of people who have access to a certain document and the owner. It holds a many to many relation with the Access table.
  - Image: The controller keeps track of all the uploaded pictures of the users and is able to get it back for later use. It's used to translate images to text to add to the document (OCR).
  - Audio: The controller keeps track of all the uploaded audio files of the users and is able to get it back for later use. It's used to translate audio to text to add to the document (Speech to Text) entity. This is the one to many relationship in which a user can have multiple audio files.
  - SummarizeAI: The controller takes in user prompts based on the text document and returns a summarized version of the content that user parsed in.
  - Json : the controller keeps track of JSON string representations of the file system. It holds a one to one relationship with SignUp as one user can only have one file system.

- **Additional:**
    - Editor : This is not part of the controllers, but it is a join table created by the many to many relationship between Files & Access. The join table shows a list of users and the documents they have access to.
    - History : This is not part of the controllers, but it is a table that keeps track of one's chat history during a live chat (websockets) also with AI bot in different file groups. It is a many to many relationship since the user can have many chats in different file groups.

Database Schema