Design Document for Coll8

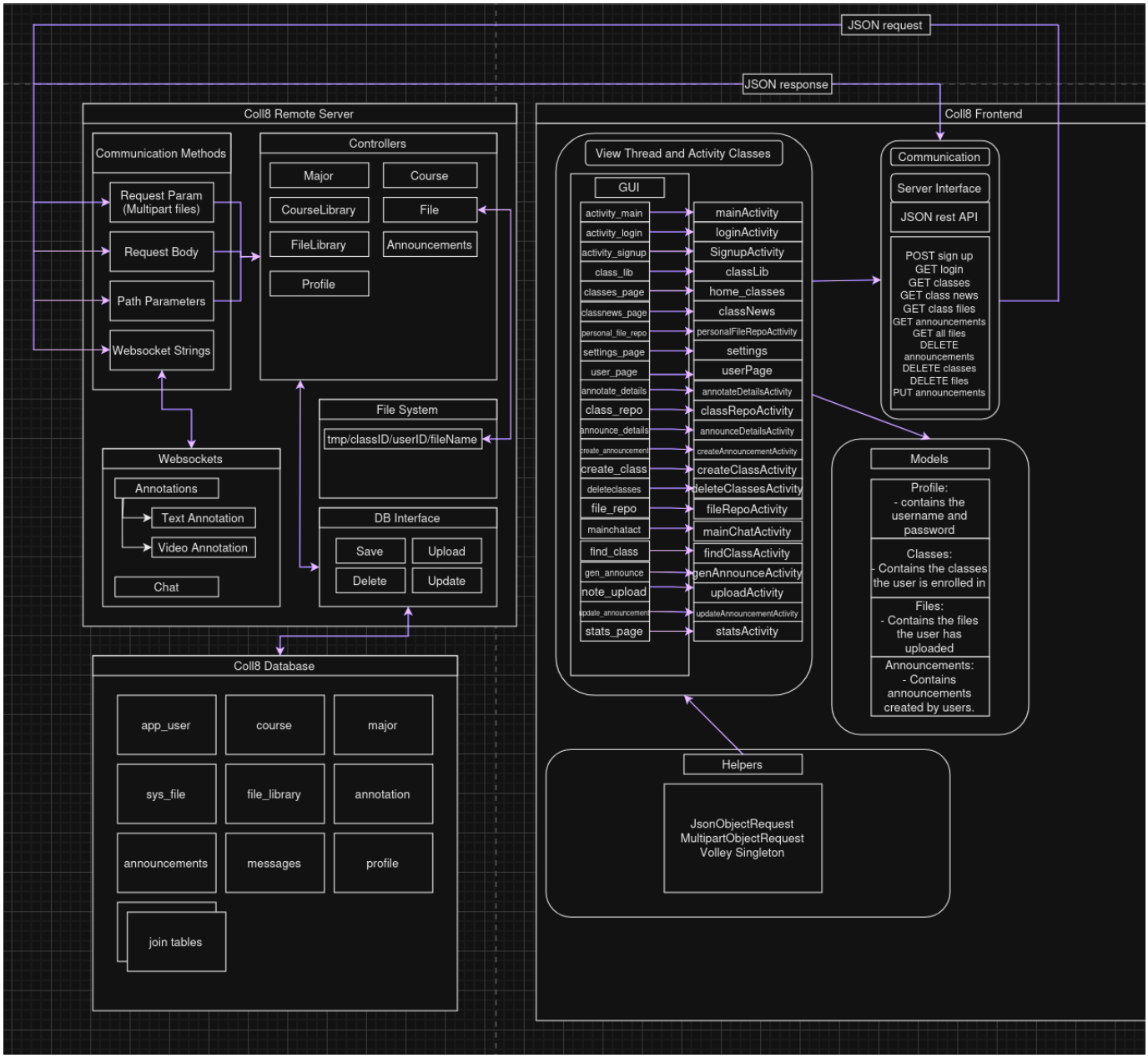
Group 4_Rasel_3

Simon Weydert: 25% contribution

Soma Germano: 25% contribution

Ayan Asim: 25% contribution

Aisha Mohammed: 25% contribution


JSON request
JSON response
Coll8 Remote Server
Communication Methods
Request Param (Multipart files)
Request Body
Path Parameters
Websocket Strings
Controllers
Major
Course
CourseLibrary
File
FileLibrary
Announcements
Profile
File System
tmp/classID/userID/fileName
Websockets
Annotations
Text Annotation
Video Annotation
Chat
DB Interface
Save
Upload
Delete
Update
Coll8 Database
app_user
course
major
sys_file
file_library
annotation
announcements
messages
profile
join tables
Coll8 Frontend
View Thread and Activity Classes
GUI
activity_main
activity_login
activity_signup
class_lib
classes_page
classnews_page
personal_file_repo
settings_page
user_page
annotate_details
class_repo
announce_details
create_announcement
create_class
deleteclasses
file_repo
mainchatact
find_class
gen_announce
note_upload
update_announcement
stats_page
mainActivity
loginActivity
SignupActivity
classLib
home_classes
classNews
personalFileRepoActtivity
settings
userPage
annotateDetailsActivity
classRepoActivity
announceDetailsActivity
createAnnouncementActivity
createClassActivity
deleteClassesActivity
fileRepoActivity
mainChatActivity
findClassActivity
genAnnounceActivity
uploadActivity
updateAnnouncementActivity
statsActivity
Communication
Server Interface
JSON rest API
POST sign up
GET login
GET classes
GET class news
GET class files
GET announcements
GET all files
DELETE announcements
DELETE classes
DELETE files
PUT announcements
Models
Profile:
- contains the username and password
Classes:
- Contains the classes the user is enrolled in
Files:
- Contains the files the user has uploaded
Announcements:
- Contains announcements created by users.
Helpers
JsonObjectRequest
MultipartObjectRequest
Volley Singleton

FRONTEND

Getting files from the database:

For a personal file repo, the user can go to the settings and the personal file repo. This will display files that the user himself/herself has uploaded.

- This will display all the files the user uploads in a text format.
- *Multipart Request:* Sends a POST request to fetch files related to a specific course and populates the filesList TextView with file details if the request is successful.

AnnotationDetails:

Utilizes the following:

- Buttons for download, delete, home, saveAnnotation, and addComment.
- Textview to display fileTitle or name.And a VideoView to display a video a video files for users to interact with
- A dynamic LinearLayout : annotationContain

Given that a user selects a file in the repos, they will be taken to this page. For some files the user will be allowed to download, otherwise they can add annotations. There is a universal delete button that can be utilized at any time for the given file.

UploadFile:

This feature utilizes

- Buttons to go back home, insert file, and submit. Followed by EditText boxes such as courseNum, courseAbbreviation, and file name.

This page is more complex because it requires reading and writing from Storage to access files, along with documenting the file type which should be appended to the filename. The wrong filename will lead to rejection of submission, along with other restrictive parameters. The request used here is a Multipartrequest, which requires Byte Mapping of the file to be sent to the backend.

BACKEND

Communication

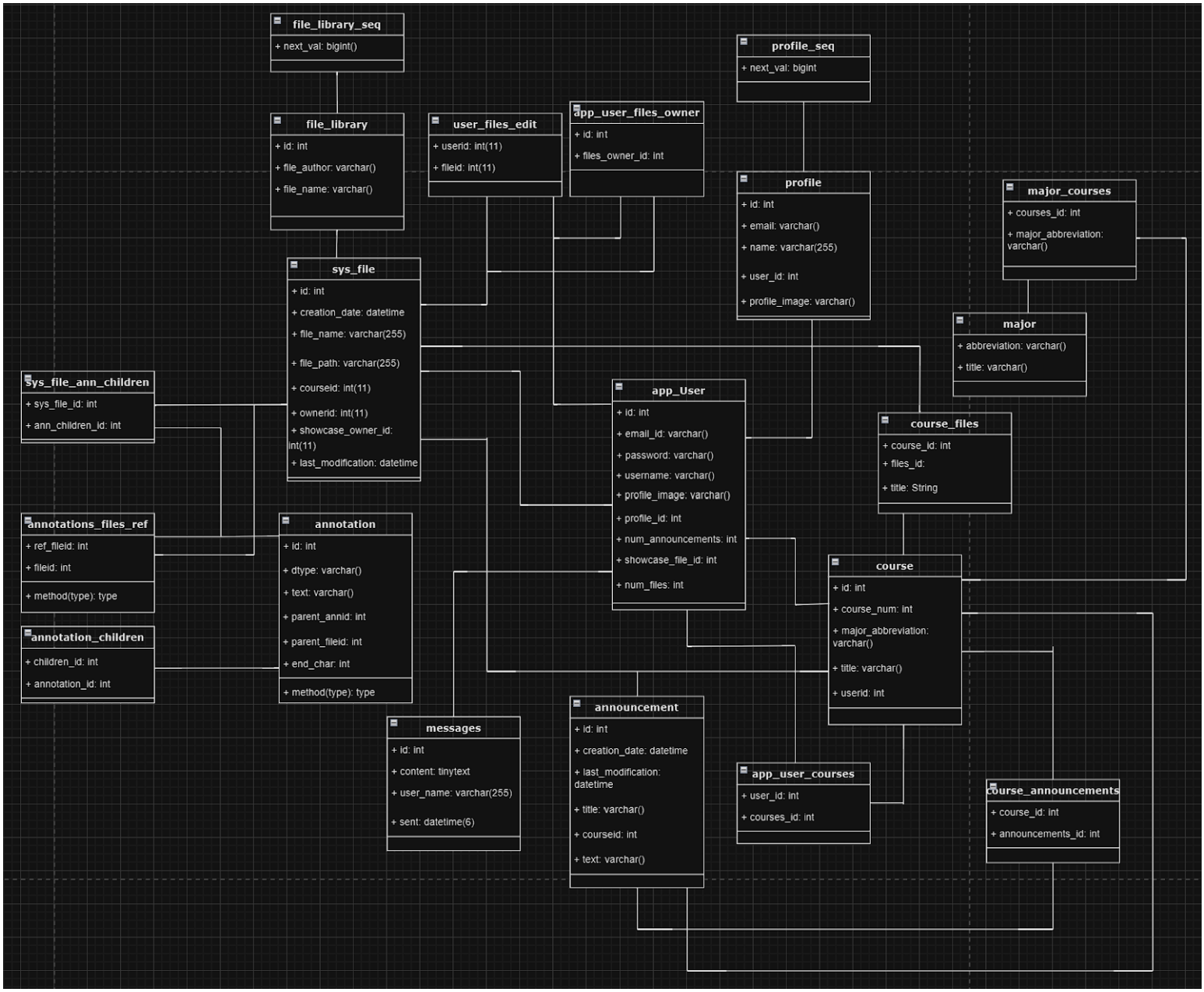
The four mappings used to interface with the system are POST, PUT, DELETE, and GET. These four employ the use of the following methods to communicate data:

- *Path Parameters*: Used in a URL to specific data. May only be basic types, such as integers and strings. These are named, and are typically used to denote IDs or usernames.
- *Request Body*: Used to send one long string which can be parsed as JSON. Commonly used to requests that need to update an object or reference several objects.
- *Request Param*: Another kind of name-value pair scheme, but this time, the value is a file or a string. Used mostly for uploading, updating, and searching for files from the file controller.
- *Websocket Strings*: A websocket connection can only send strings back and forth, which may be interpreted as JSON. The only two controllers that utilize websockets are the Annotation and Chat controllers.

Complex Controllers

Controllers may interact with both the filesystem and the database.

- *File controller*: Create and delete files on the file system. For each file, create a SysFile object in the database that describes the file and its metadata, such as its owner and any annotations made on it.
- *Course*: A course acts as a repository of files for a group of users. As such, each course has a set of files. Each course also has a unique major / course number pair.
- *User*: A user object details a user and all their actions, including all files, annotations, and announcements they've uploaded. They also have a list of joined courses.


file_library_seq
+ next_val: bigint()
profile_seq
+ next_val: bigint
file_library
+ id: int
+ file_author: varchar()
+ file_name: varchar()
user_files_edit
+ userId: int(11)
+ fileId: int(11)
app_user_files_owner
+ id: int
+ files_owner_id: int
profile
+ id: int
+ email: varchar()
+ name: varchar(255)
+ user_id: int
+ profile_image: varchar()
major_courses
+ courses_id: int
+ major_abbreviation: varchar()
sys_file
+ id: int
+ creation_date: datetime
+ file_name: varchar(255)
+ file_path: varchar(255)
+ courseId: int(11)
+ ownerId: int(11)
+ showcase_owner_id: int(11)
+ last_modification: datetime
major
+ abbreviation: varchar()
+ title: varchar()
sys_file_ann_children
+ sys_file_id: int
+ ann_children_id: int
app_User
+ id: int
+ email_id: varchar()
+ password: varchar()
+ username: varchar()
+ profile_image: varchar()
+ profile_id: int
+ num_announcements: int
+ showcase_file_id: int
+ num_files: int
course_files
+ course_id: int
+ files_id:
+ title: String
annotations_files_ref
+ ref_fileid: int
+ fileid: int
+ method(type): type
annotation
+ id: int
+ dtype: varchar()
+ text: varchar()
+ parent_annid: int
+ parent_fileid: int
+ end_char: int
+ method(type): type
course
+ id: int
+ course_num: int
+ major_abbreviation: varchar()
+ title: varchar()
+ userid: int
annotation_children
+ children_id: int
+ annotation_id: int
messages
+ id: int
+ content: tinytext
+ user_name: varchar(255)
+ sent: datetime(6)
announcement
+ id: int
+ creation_date: datetime
+ last_modification: datetime
+ title: varchar()
+ courseid: int
+ text: varchar()
app_user_courses
+ user_id: int
+ courses_id: int
course_announcements
+ course_id: int
+ announcements_id: int