Design Document for ScribbleShare

Group <1_SG_2>

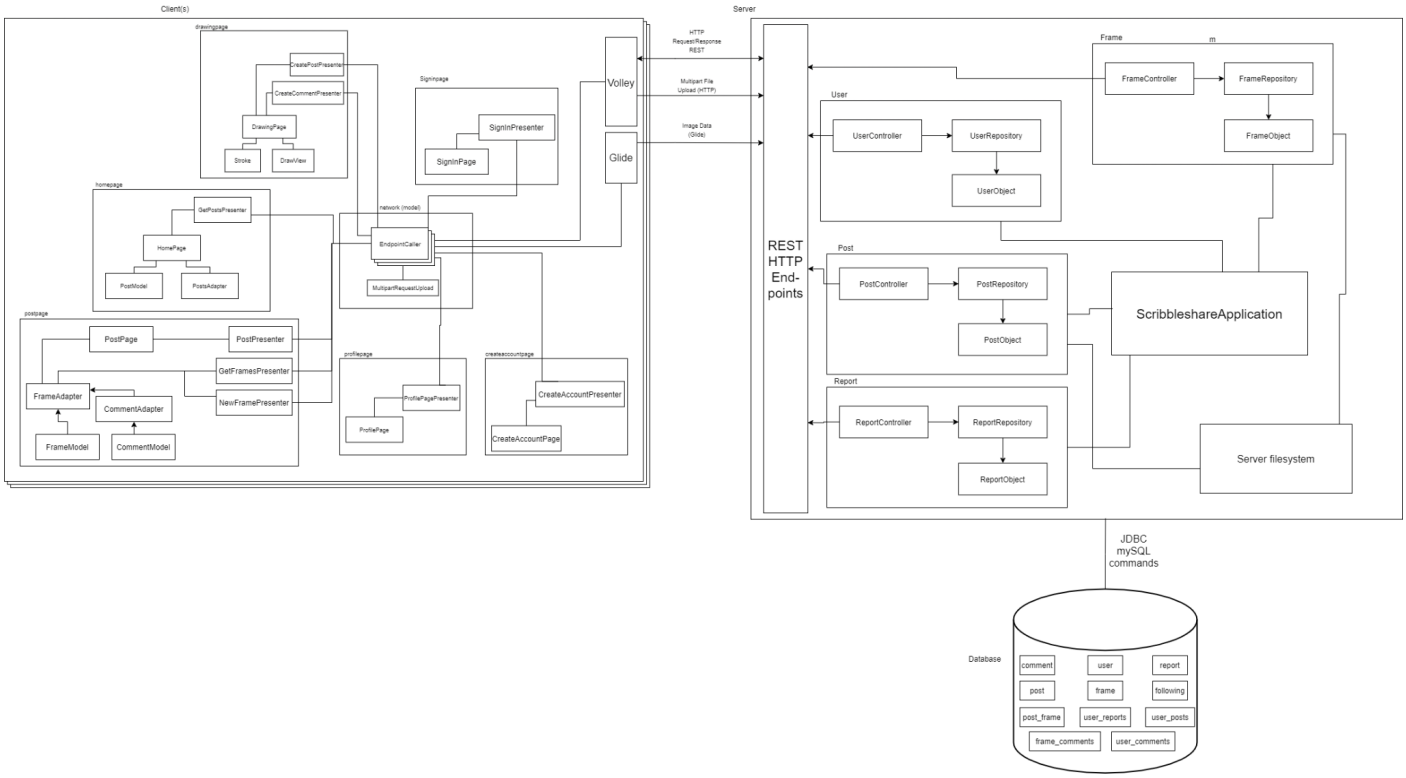
Member 1 Corbin Kems: 25% contribution

Member 2 Blake Fisher: 25% contribution

Member 3 Cole Langer: 25% contribution

Member 4 Jens Rasmussen: 25% contribution

PUT THE BLOCK DIAGRAM PICTURE ON THIS PAGE! (Create the picture using pencil or drawIO)



Use this third page to describe complex parts of your design.

Client

Our application is an android app which implements the Android class AppCompatActivity in various classes named "*Page" (for example DrawingPage, HomePage, etc). Each page communicates with at least one presenter, which has the responsibility of communicating with the EndpointCaller. A presenter implements IVolleyListener<T>, where T is the response type of the network request it will be listening for from the EndpointCaller. The EndpointCaller's responsibility is to create and process network requests, and to send each response to its correct presenter (which automatically casts it to the presenter's type T). The client uses two methods of creating requests. One uses volley to create HTTP requests which get JSON responses that are parsed in various locations in the app. The other type uses Glide, which makes necessary image data requests to the server to load various image data.

The structuring of our data enables us to easily compartmentalize our client application for both testing and maintenance. Presenters only need to implement the correct IVolleyListener, so they can easily be mocked or re-created with new presenter logic if needed. The endpoint caller can also easily be extended to support more types of requests, as it can be instantiated with any type T. This also allows for complete asynchronous network communication with the server, as all requests are listener based. This enables the client to be responsive and user friendly even when it's loading many images (which our app consists of many of) or other data such as a post feed or profile information.

Server

Our server uses a spring boot application to implement many REST HTTP endpoints. Each endpoint is defined by its respective controller class (for example, UserController, PostController, etc). These classes define the endpoints in every way, including HTTP request type (GET, PUT, DELETE, etc), input parameters, and return types. Each controller is connected to a repository, which is responsible for communicating with the database using JDBC mySQL commands. The repository can also be configured to provide custom SQL queries, which enables more complicated data processing, as custom queries can collect and manipulate data in any way.

A unique aspect of our server's backend is the use of the local filesystem to store images. Instead of encoding them to store on the local tables, when a new post/comment is uploaded to the server, it saves the file to a location on the server's file system, then saves that file path as a string in a column in the database. This is especially useful because it reduces the size of the database, making it more responsive. When a post/comment's image is needed to be retrieved, the file path is gathered from the database, and a multipart file is created to serve at the endpoint. This enables very simple image downloading on the client side, as it simply needs to create a multipart file download request (which is handled entirely by Glide in android) to load the image.

Database

Our database is a mySQL server which is entirely managed by spring boot. Most commands are also handled by JDBC from the server, but a few custom SQL queries have also been written for more complicated data processing.

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkbench)