Moutour Mathematicians

Engagement App

Design Specifications 1.0

Russell A, Michael C, Devin F, Tyler H, Jaheim O
3-6-2025

Table of Contents

1. Introduction

1.1 Purpose

This document provides a high-level, graphical understanding of the different software elements that compose the Engagement app. It serves as the design blueprint on which the implementation and development of software components will be based.

1.2 Scope

This document includes diagrams, methods, class attributes, and relationships between elements that meet and describe the implementation of the requirements specified in the Requirements Document.

2. General Description

2.1 Product Goal

The goal of this application is to provide end users with an application that allows them to easily navigate, interact, and keep up with campus events and other happenings throughout the semester. It must also provide means for moderation, and a posting system that allows event organizers and student organizations to promote their events.

2.2 Development Profile

The application is a web service hosted on a machine located on the Ursinus College Campus known as Stratus. The application will function primarily in a client-server model, where the client (An end user's phone) interacts with a server (the Stratus) to perform the goals identified in the requirements documentation. The client will communicate with the server using a RESTful API implementation. The Database identified in the Requirements documentation will be implemented using ... postgreSQL

3. System Architecture Overview

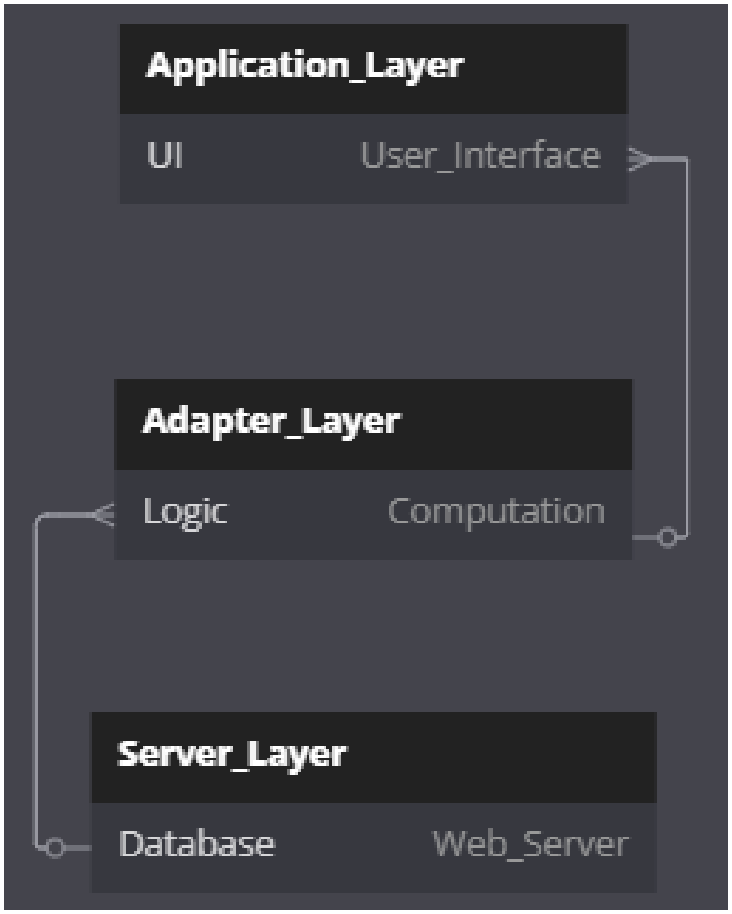

Figure 1 - Student Engagement App Architecture.

Each layer provides the application with what is needed for the user to utilize the functionalities of the app while saving state for multiple users simultaneously. For example, the Application Layer handles all user interactions, the Adapter Layer handles any necessary logic or computational processes on the user and post data, and the Server Layer is responsible for saving and organizing user and post data. The data is funneled through the adapter layer to be translate the readability from the Application Layer to the Server Layer and vice versa. The app will be created as a web app and will be able to be ported into simple skeletons in iOS and Android mobile devices.

4. Communication Between Layers

Each following section contains the design details including UML diagrams laying out the functional classes, functions, and screens of the Student Engagement app. The application consists of multiple layers each with its own set of screens, functions, or databases.

The following diagram provides an overview of how all the classes between and within the layers work together.

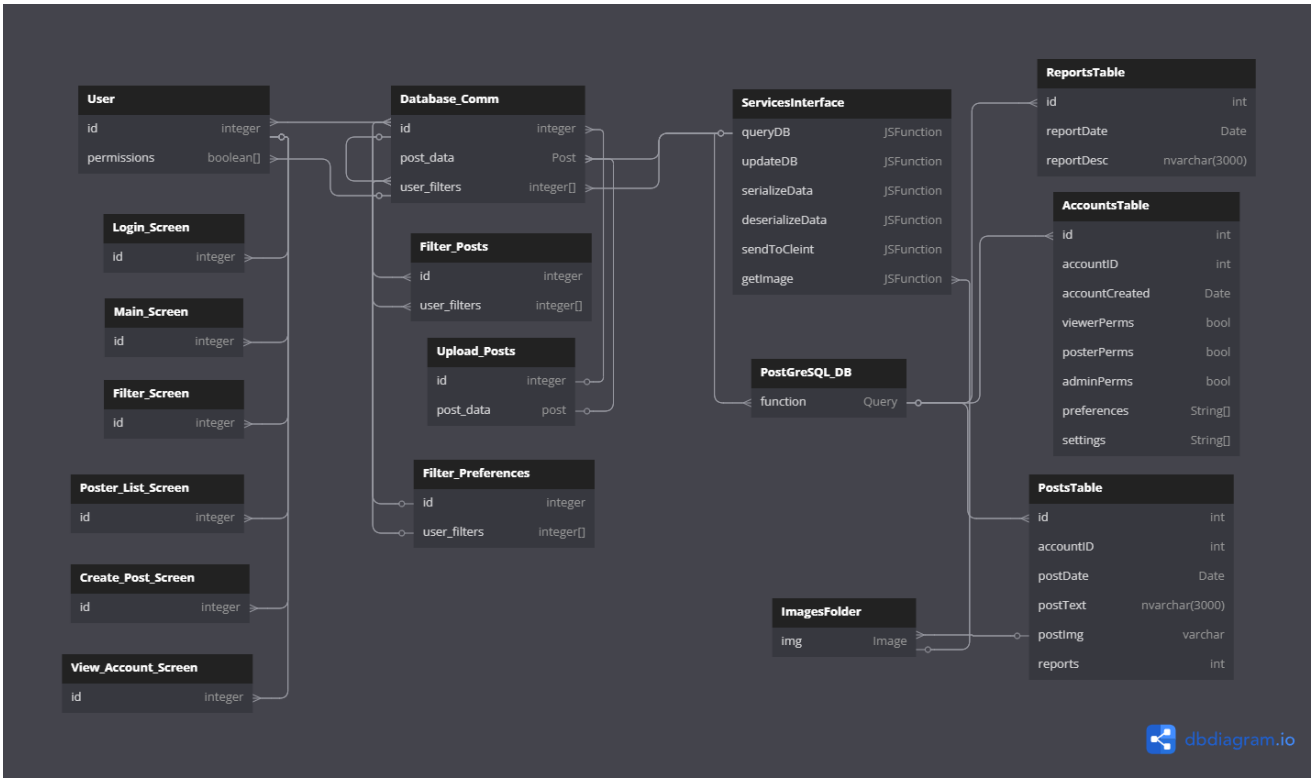

Figure 2 – Diagram of Communication Between Layers

5. Application Layer

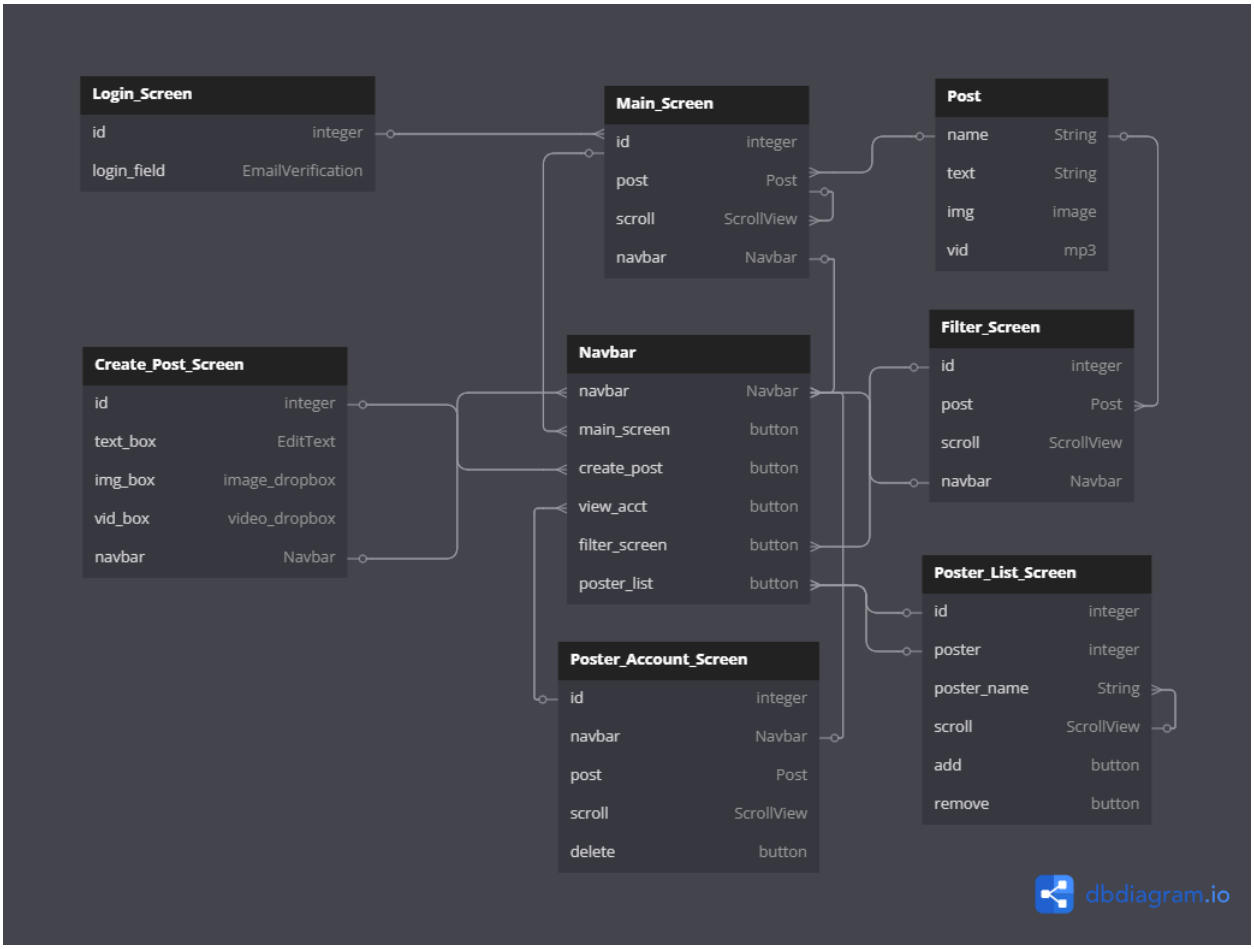

Figure 3 – Application Layer Diagram

5.1 Main_Screen

An interface that allows the user to scroll through several posts, with the navbar providing buttons to other screens.

5.1.1 id

The user's id associated with their account that allows for tailored data and to track their actions and available actions within the app.

5.1.2 post

A collection of text, images, and videos that posting accounts can send out to the application to appear in every users main screen and in their followers following screen.

5.1.3 scroll

A view that contains all of the posts that the user can scroll through.

5.1.4 navbar

A location within the main screen containing a collection of buttons to allow the user to navigate to multiple different screens within the application.

5.2 Login_Screen

Allows the user to login through their email, authenticating through an email verification field.

5.2.1 id

A unique numerical value associated with an email address (establishes the user id).

5.2.2 login_field

A container wherein an end user enters their account credential (an email address).

5.3 Create_Post_Screen

An interface that provides input fields for users to create and upload new posts, including text, and images.

5.3.1 id

A unique ID to identify the post.

5.3.2 text_box

A container where a poster will input the text they wish to include in their post.

5.3.3 img_box

A container where a poster will input an image they wish to include in their post.

5.3.4 vid_box

A container where a poster will input a video they wish to include in their post.

5.3.5 navbar

A button that allows the end user to return to the other pages of the application.

5.4 Post

An object that contains all of the elements of a post.

5.4.1 name

Author of the post.

5.4.2 text

A container wherein the post's text is displayed

5.4.3 img

A container wherein an image file is displayed

5.4.4 vid (aspirant)

A container wherein a video file is displayed

Displays an individual post, which can include a name, text, images, and audio.

5.5 Navbar

Provides buttons for accessing the different screens, including the main screen, create post screen, account screen, filter screen, and poster list screen.

5.5.1 navbar

A display container to hold the buttons to allow the users to traverse the application.

5.5.2 main_screen

An html button that takes the end user to **5.1**

5.5.3 create_post

An html button that takes the end user to **5.3**

5.5.4 view_acct

An html button that takes the end user to **5.5**

5.5.5 filter_screen

An html button that takes the end user to **5.8**

5.5.6 poster_list

An html button that takes the end user to **5.7**

5.6 Poster_Account_Screen

5.6.1 id

A unique numerical identifier associate with the posters account allowing them to identify their own posts.

5.6.2 navbar

A display container to hold the buttons to allow the users to traverse the application.

5.6.3 post

A collection of text, images, and/or videos sent by the poster to the application for the users to see with an associated user id.

5.6.4 scroll

A container for the user to be able to scroll through their own posts.

5.6.5 delete

A button to allow the poster to delete posts they no longer want visible.

5.7 Poster_List_Screen

A list of all poster accounts for the user to scroll through and select which accounts they wish to follow or unfollow.

5.7.1 id

The user's id associated with their account to link to their specific data.

5.7.2 poster

The poster account's user id.

5.7.3 poster_name

The name of the posting account associated with the id.

5.7.4 scroll

A container for the user to be able to scroll through all of the posters names.

5.7.5 add

A button for the user to be able to add any poster they do not already follow to their following list.

5.7.6 remove

A button for the user to be able to remove any poster they follow from their following list.

5.8 Filter_Screen

An interface that enables users to apply filters to customize the type of content displayed on the main screen.

5.8.1 id

The user's id allowing the application to determine which accounts they follow.

5.8.2 post

A collection of text, images, and/or videos that posters can send to the main screen and to the filter screen of users who follow them.

5.8.3 scroll

A container to store all of the posts from the accounts the user follows that the user can scroll through.

5.8.4 navbar

A container of buttons to allow the user to navigate through the various screens of the app.

6. Adapter Layer

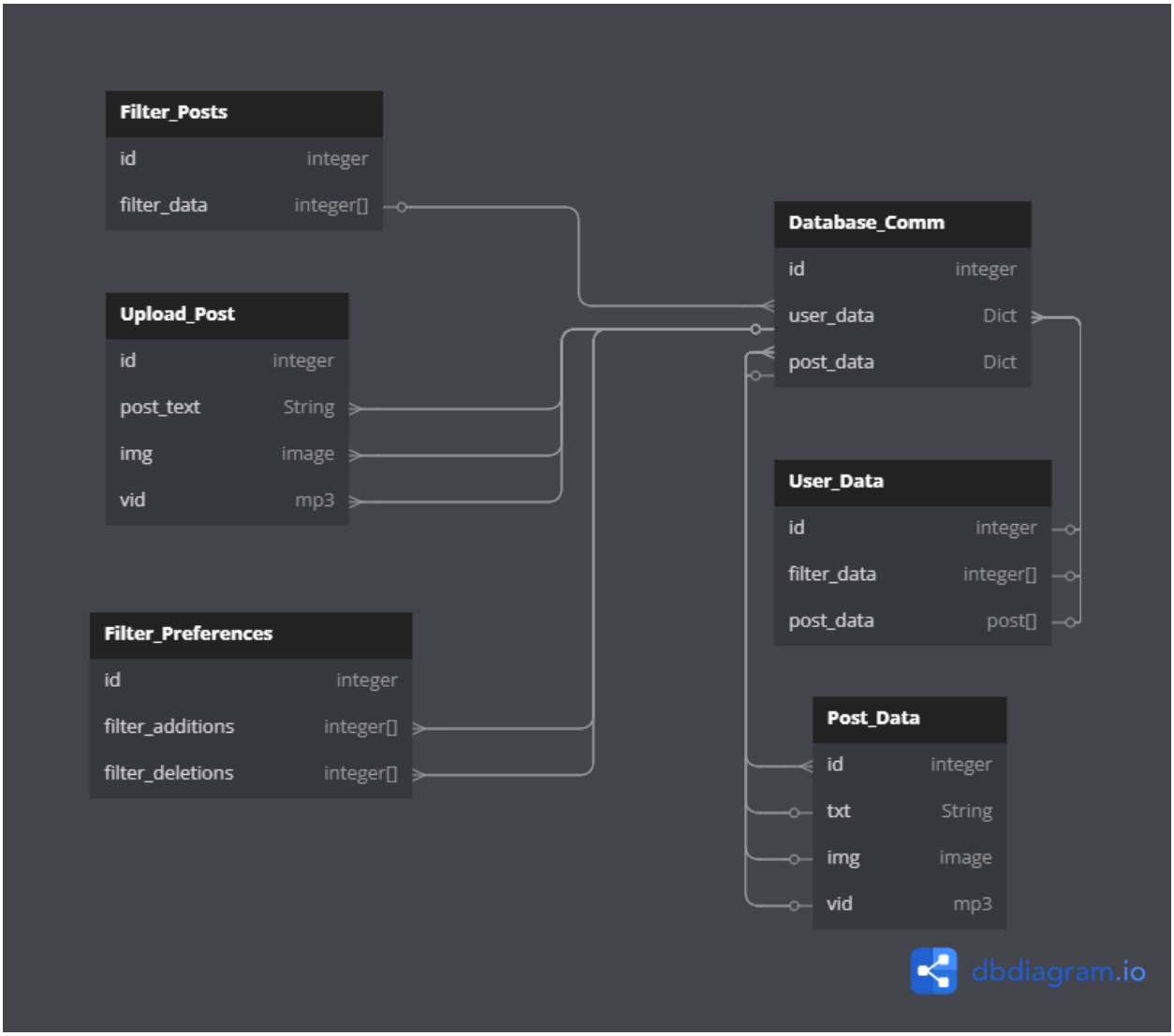

Figure 4 – Adapter Layer Diagram

6.1 Database_Comm

6.1.1 id

The user's unique id to allow for specific queries to the server to result in personalized data for each user.

6.1.2 user_data

A collection of data associated with the user id including the user's following list, permissions, and changes to the following list.

6.1.3 post_data

A collection of data associated with a post including the text, image, and/or video associated with the post as well as the poster user that created the post.

6.2 Filter_Posts

A functionality of the adapter layer to filter all posts into only the posts from the accounts that the specific user follows.

6.2.1 id

The specific user id whose following list the posts are being filtered to follow.

6.2.2 filter_data

The user's list of poster accounts that they follow.

6.3 Upload_Post

The function that translates the text, image, and/or video from the posters in the application layer to the databases in the server layer.

6.3.1 id

The id of the poster user who created the post.

6.3.2 post_text

The text contained within the post.

6.3.3 img

The image contained within the post (if applicable).

6.3.4 vid (aspirant)

The video contained within the post (if applicable).

6.4 Filter_Preferences

The translation between the application and server layer concerning user's additions or deletions from their following list.

6.4.1 id

The user's id whose following list is being edited.

6.4.2 filter_additions

All of the poster user's ids that are being added to the user's following list.

6.4.3 filter_deletions

All of the poster user's ids that are being removed from the user's following list.

6.5 User_Data

The individual user's specific data to allow for a personalized experience of the application.

6.5.1 id

The specific user's numerical id number to be associated with all other user data.

6.5.2 filter_data

The list of post user ids that the user wishes to follow to filter their posts into the filter screen.

6.5.3 post_data

If the user is a poster, this holds a list of all of the posts they have created and sent using the app.

6.6 Post_Data

The stored data of each post created within the app.

6.6.1 id

The id of the poster user who created the post.

6.6.2 txt

The text contained within the post.

6.6.3 img

The image contained within the post (if applicable).

6.6.4 vid (aspirant)

The video contained within the post (if applicable).

7. Server Layer

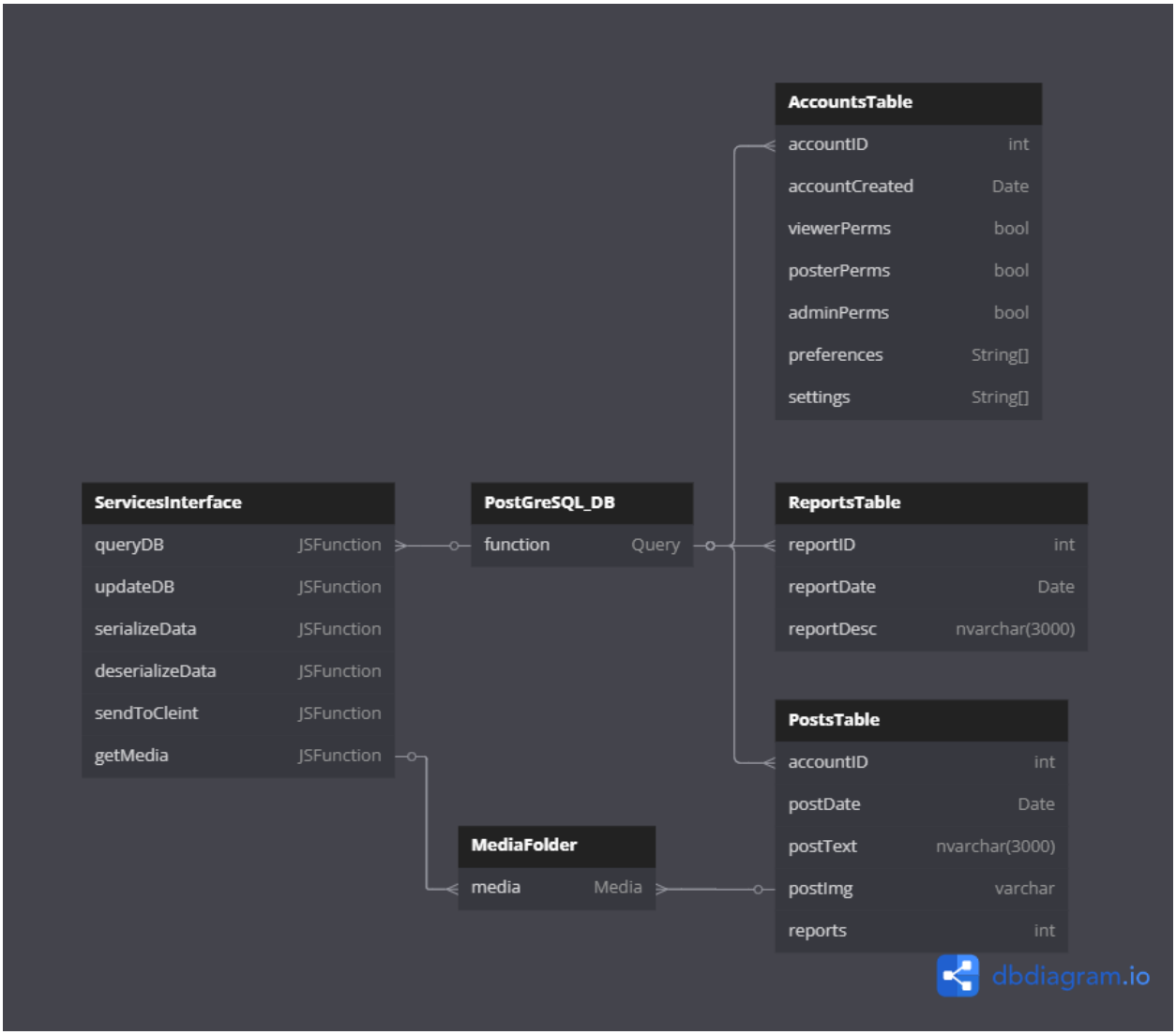

Figure 5 – Server Layer Diagram

7.1 ServicesInterface

The services interface is composed of all functions that operate on server resources.

7.1.1 queryDB

This method will take deserialized data, parse it, and query the Database. It will return the results of the query. Expressly for SELECT queries.

7.1.2 updateDB

This Method will take deserialized data, parse it, and add entries to the appropriate table in the Database.

7.1.3 serializeData

serializeData prepares the output of QueryDB or updateDB for transmission back to the client.

7.1.4 deserializeData

deserializeData prepares incoming data for utilization by other functions the server provides.

7.1.5 sendToClient

sendToClient uses HTTP to transmit serialized data back to the client over the network.

7.1.6 getMedia

The media data stored in the DB is merely a pointer to where the media itself is on the server machine. This function takes that location and retrieves the resource, as part of data serialization.

7.2 AccountsTable

The AccountsTable holds a record for each unique account that is registered for the service

7.2.2 accountID (int)

Unique numerical Identifier for an account

7.2.3 accountCreated (Date)

Date that an account was created

7.2.4 viewerPerms (Boolean)

A Boolean that determines if an account has viewer permissions.

7.2.5 posterPerms (Boolean)

A Boolean that determines if an account has posting permissions.

7.2.6 adminPerms (Boolean)

A Boolean that determines if an account has admin permissions.

7.2.7 preferences (string[])

A string array that stores the end user's post preferences.

7.2.8 settings (string[])

A string array that stores the end user's client settings.

7.3 ReportsTable

7.3.1 reportID (int)

unique report identifier

7.3.2 reportDate (Date)

date the report was created.

7.3.3 reportDesc (nvarchar(3000))

A field that contains a description of the offense reported by the end user.

7.4 PostsTable

PostsTable is a table that holds all posts and the information they contain.

7.4.2 accountID (int)

accountID refers to the account that created the post.

7.4.3 postDate (Date)

postDate is the date a post was created.

7.4.4 postText (nvarchar(3000)

postText is the text included in the post itself.

7.4.5 postImg (varchar)

postImg is a text value that points to an image file located on the server.

7.4.6 reports (int)

An integer that counts the number of reports made against a post.

7.5 MediaFolder

This folder is where all post media is stored (video / image)

7.5.1 media

A media file in .png, .jpeg, .gif, .mp3, .mp4 formats

7.6 PostGreSQL_DB

The Database instance on the server, which contains the tables PostsTable, ReportsTable, and AccountsTable.