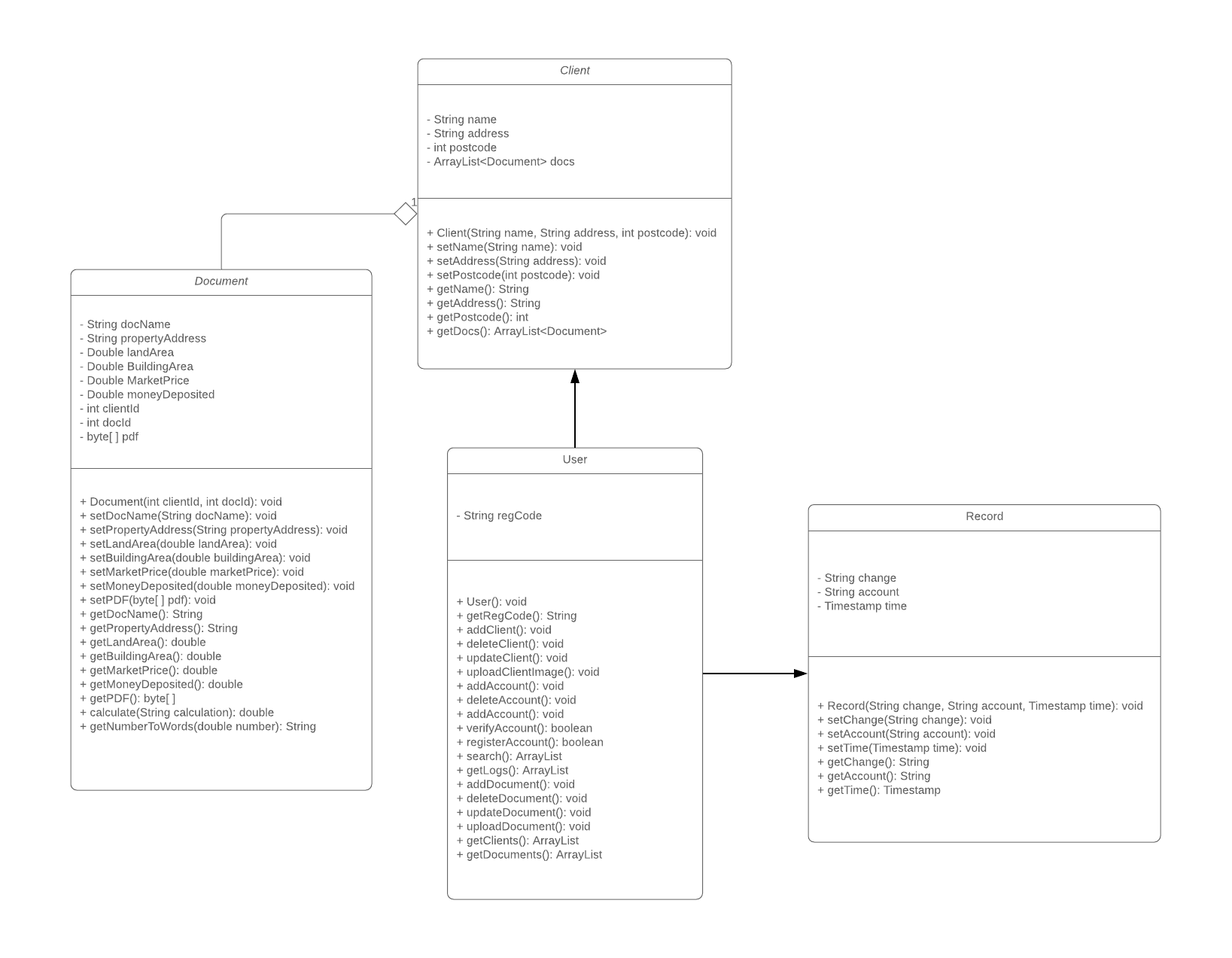
**Criterion B: Design**

**UML**

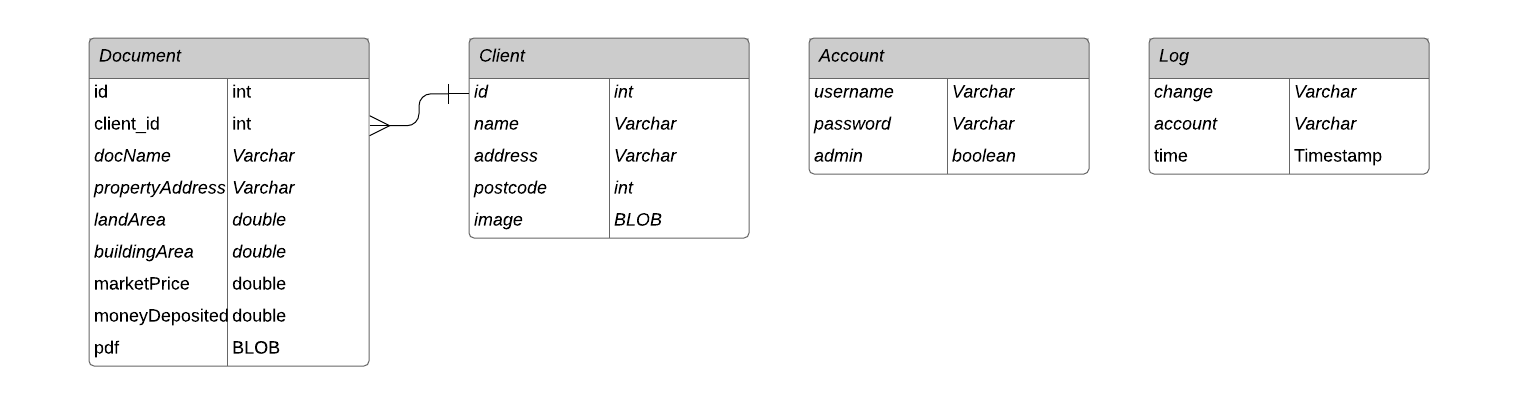
*Class Relationships*

**

The user class is used to interact with the database and uses client, document, and record objects when communicating or passing data to the JavaFX class. However, the user class is its own stand alone entity. The record, client, and document classes are each classes that defines an object. The client has an ArrayList of type document as each client has multiple documents which are indicated in the array of documents (docs) shown in the uml for the clients class. Both the client and document classes are objects used when pulling data from the respective ‘client’ and ‘document’ tables in the database. Record is used to represent objects that are later displayed in the logs page and used in the User class when pulling from the ‘log’ table of the database.

**Database**

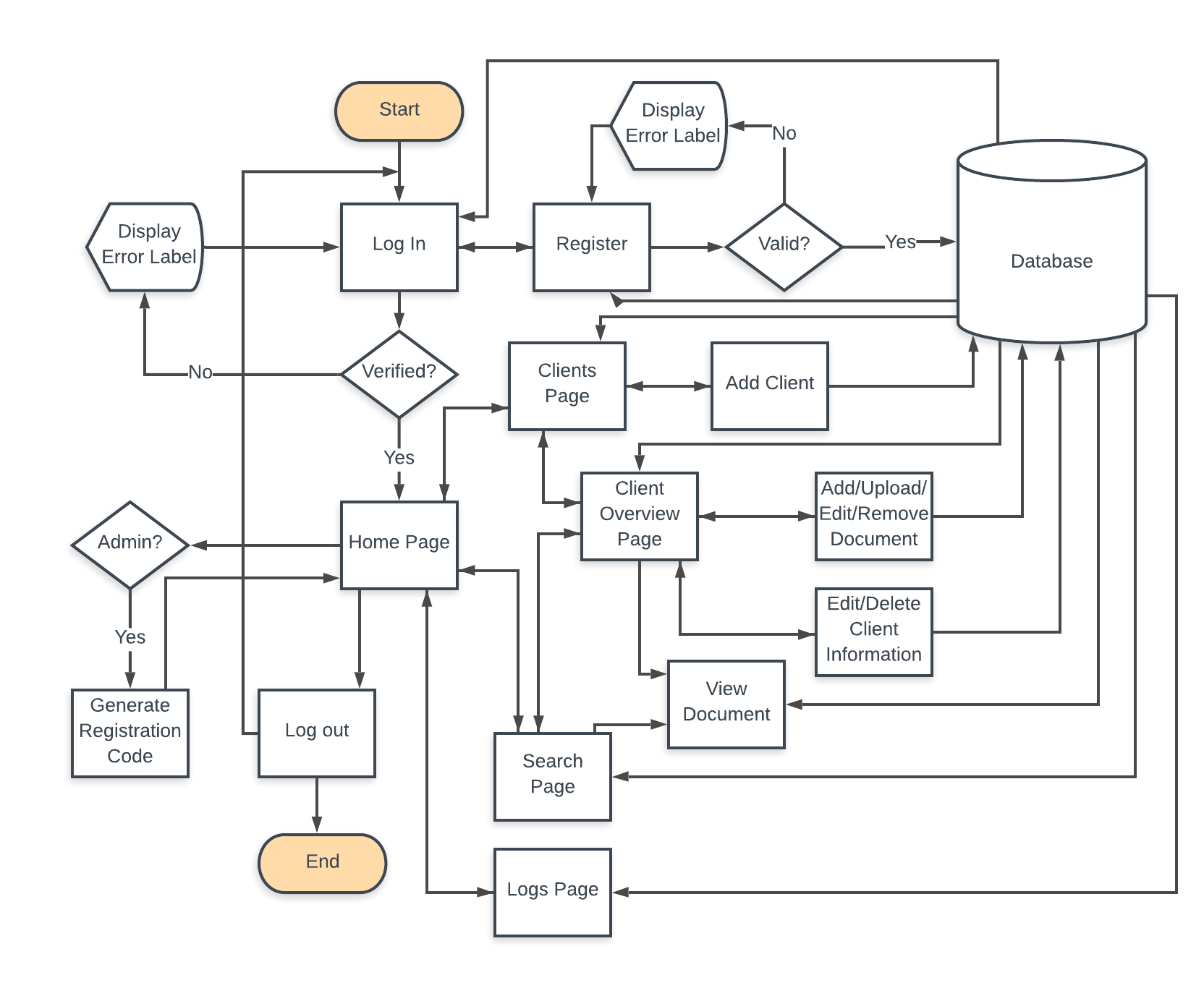
*Entity Relationship Diagram*



* A relational database will be used in order to represent the relationships between each entity. There will be 4 tables in the database, each representing a different entity.
* The “account” table is dedicated in keeping track of the registered accounts that will be able to access the application. “Account” is its own standalone entity as the data is not associated with any of the other entities. The “account” table is accessed to verify users logging in and also updated when new accounts are made in the register page.
* The “log” table is dedicated to keeping track of all of the changes made into the application. The “change” column keeps track of what specific change is made while the “account” indicates who made the change and the “time” column indicates when the change is made. While “account” is used in this table, the “log” table is not actually associated with the “account” table, making it a standalone entity.
* The “client” table has a one-to-many relationship with the “document” table as one client can have many documents. To indicate this relationship, the “client” table has a primary key “id” which is then set as the foreign key for the “document” table, indicated as “client\_id”.

**System Flowchart**

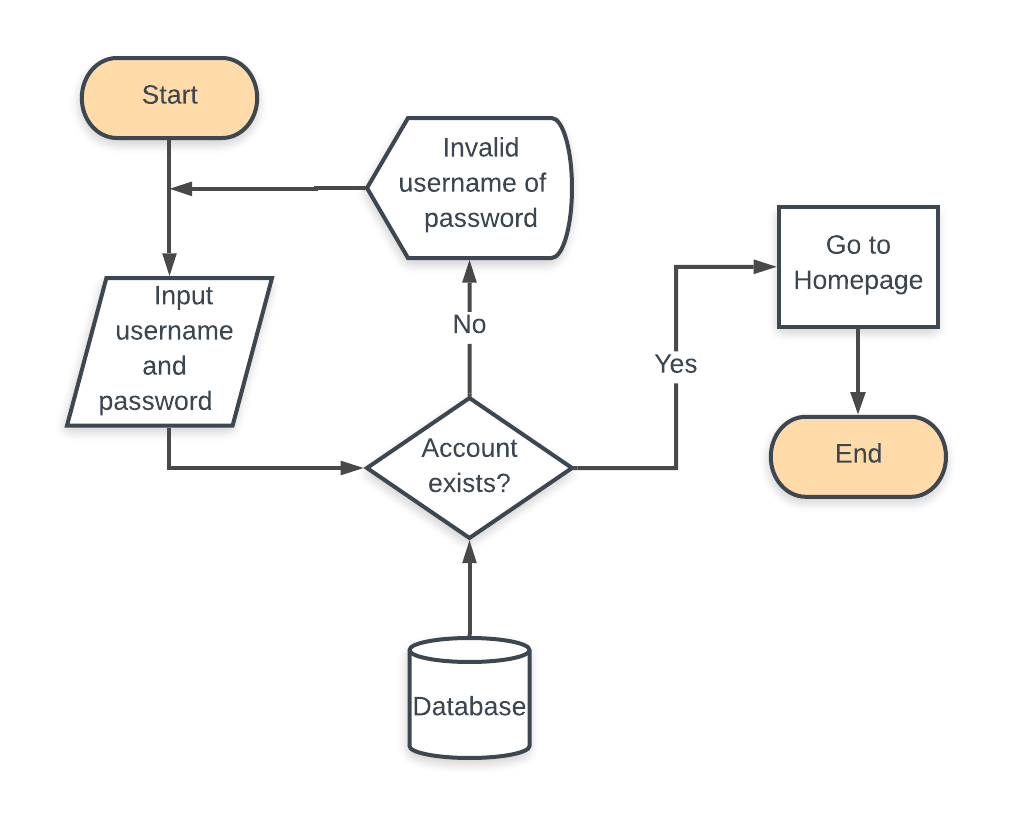
*Overview of the App*

**

* The app opens with a login page where user enters their username and password and verified by the database
  + If the account is verified, the user can access the home page
  + If the account is not verified, an error will be displayed and user will not be logged in to the home page
* From the login page, user can also go to the register page to register new accounts which requires a valid registration code and input fields to be filled out properly
  + If an error occurred because the inputs are invalid, an error label will be displayed and the account will not be registered
  + If the inputs are valid, the new account will be registered in the database and user will go back to register page
* If the user is an admin, user can generate a registration code
* User can go to the logs page from the home page which will display the recent changes made on the app from the database
* User can go to the clients page which will list all clients in the database
* User can add a client from the clients page which will then be saved in the database
* User can also go to the client overview page from clients page or search page
  + User can add/upload/edit/delete documents in the client overview page and these modifications will be updated in the database
  + User can also edit/delete client information from the client overview page
* From the homepage, user can go to search page to search for clients and/or documents
  + User can go to client overview page from search page by selecting a client
  + User can view documents from the search page when they click on a searched document
* User can log out from the home page which will take them back to the login page

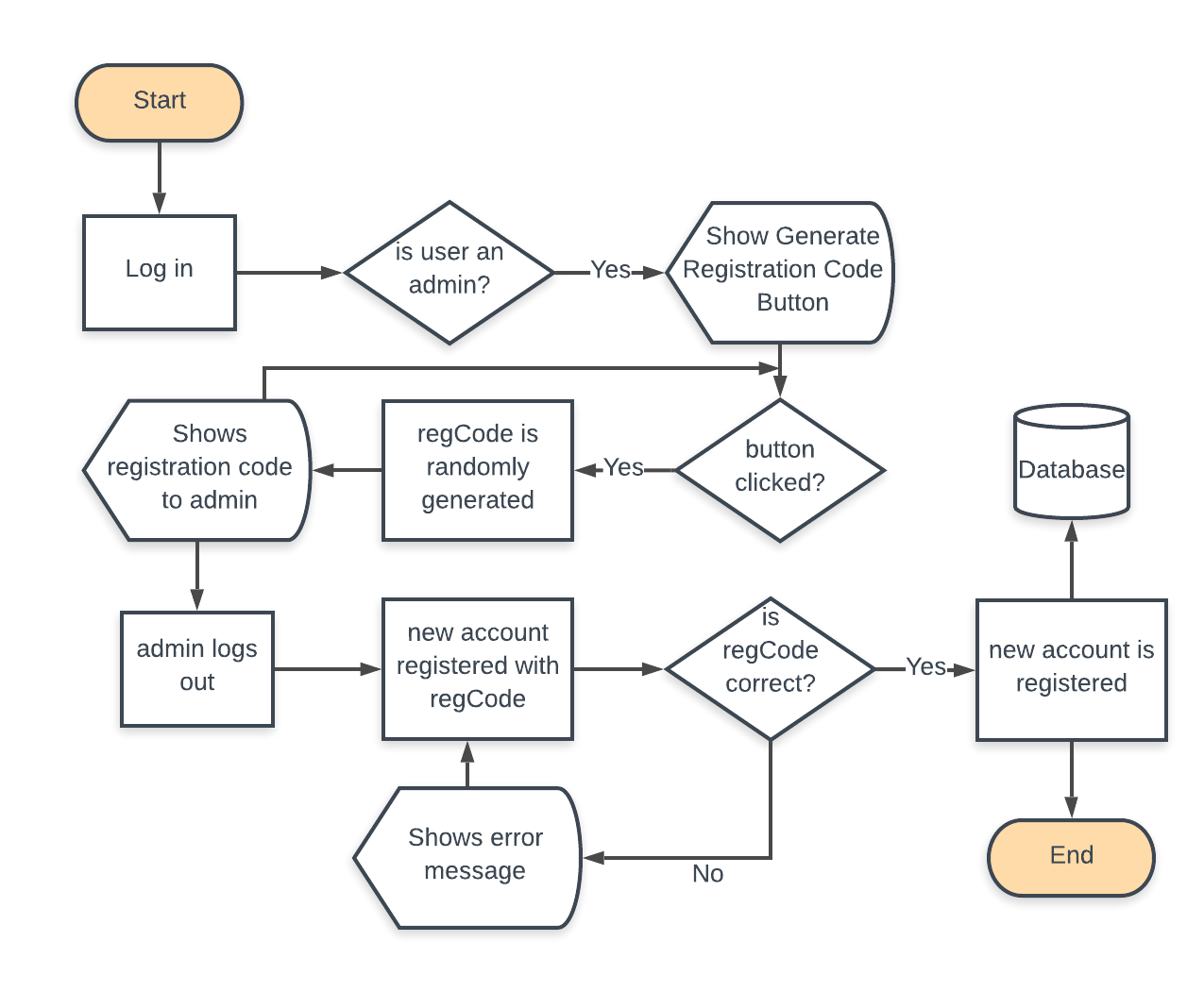
**Algorithm Flowchart**

*Sign in Page*



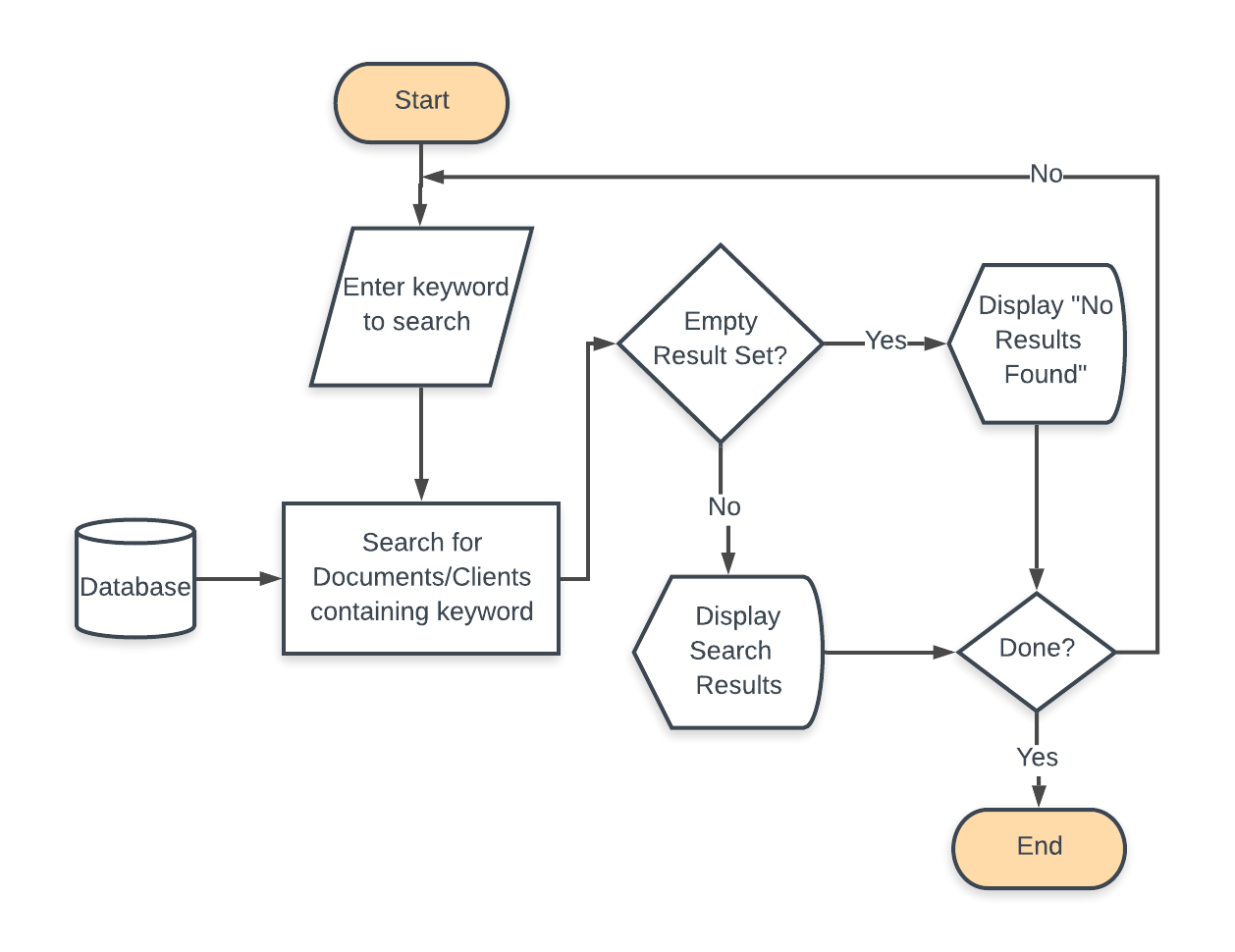
* This algorithm is executed as soon as the user click the ‘enter’ button on the sign in page and entered their username and password which serves as the input/parameters for this method
* The method will then check the accounts table in the database to check if the account exists
* If the account exists, the user will then be redirected to the homepage where they can access all the functionalities of the app.
* If not, an error message will show that will ask the user to enter the right username and password

*Register Page with Registration Code*

**

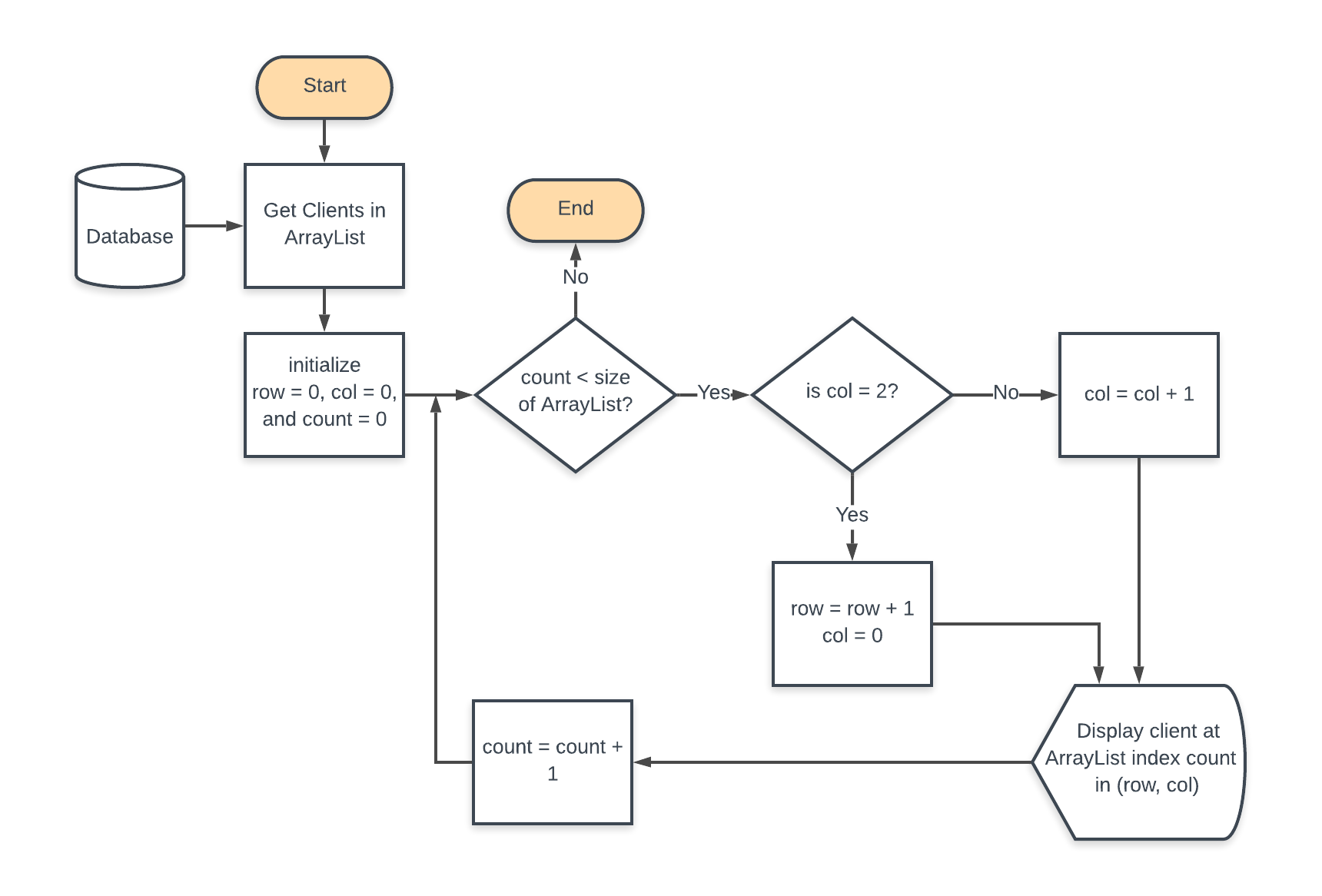
* To add new users so that they can log in to the application, an admin will need to sign in first to get the registration code
* If the user who logs in is an admin, they will have an additional button on their home page to generate a registration code
* The user then clicks on the button to randomly generate the registration code. For every time the button is clicked, the application will randomly generate a new registration code and displayed to the user.
* Admin then logs out and goes to the registration page to register a new account with the registration code.
* If the registration code is incorrect or not all of the input fields are filled out, an error message will show up, prompting the user to check their registration form.
* If the registration code entered is correct, the account is then registered to the database.

*Finding documents/clients in Search page*



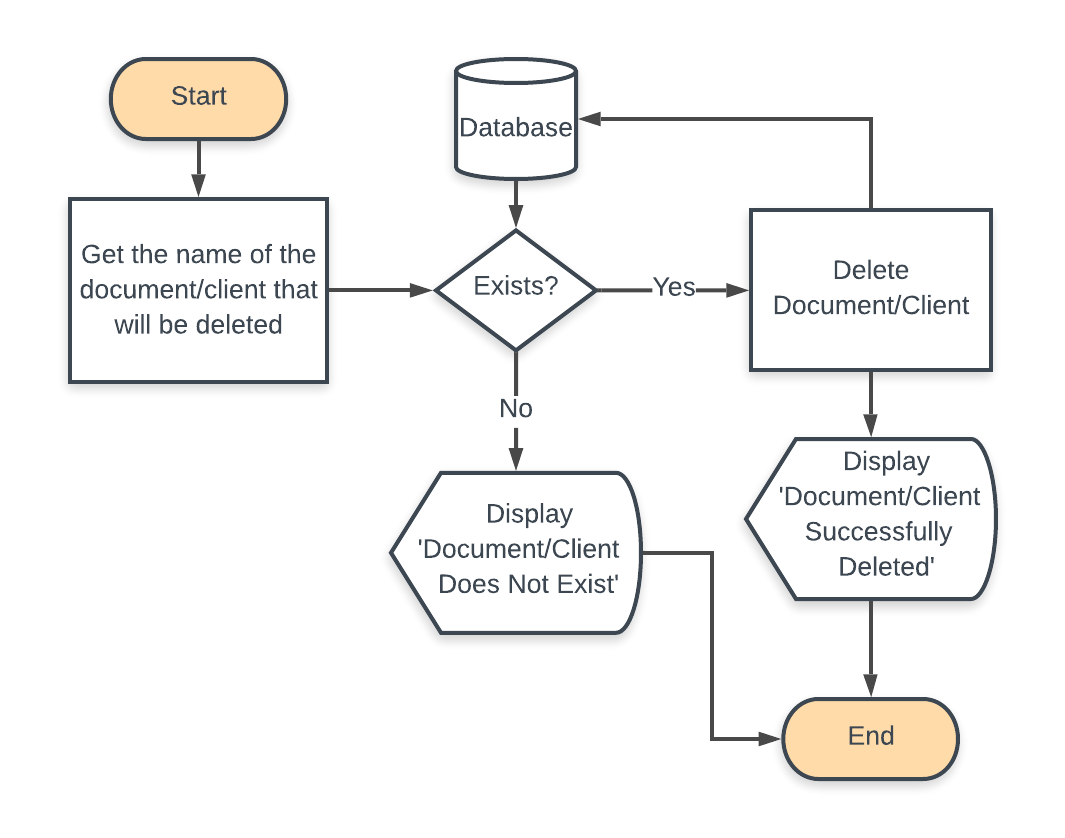
* This algorithm will be called in the search page for every time the user enters something in the search bar
* It will access both the documents and clients table to find any documents or clients who’s title/names contain the keyword entered by the user
* If there are any documents or clients that’s title/name contains the keyword entered by the user, then all of these documents/clients will be shown below the search bar that the user can click on
* If no documents/clients are found, then it will show an error message saying ‘No Results Found’

*Show Clients in Rows of Three*

**

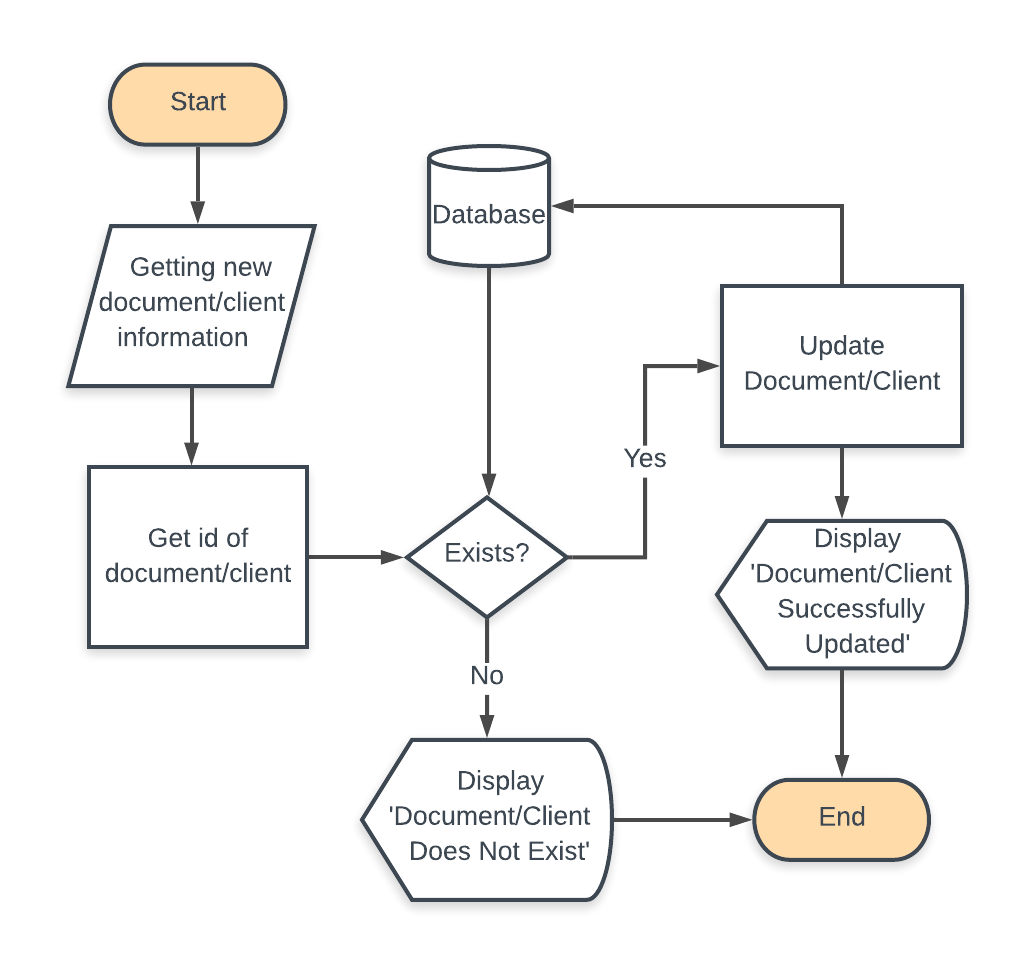
* This algorithm is run every time the user visits the clients page
* Purpose of algorithm is to display the clients in rows of three (*See Mockup for Clients Page below)*
* It will first get the list of clients from the database
* Loop structure will be used to iterate through the list and every time there are already three clients displayed in the row (by checking if the column or col equals to 2), it will start displaying the clients in the next row

*Removing Document/Client*

****

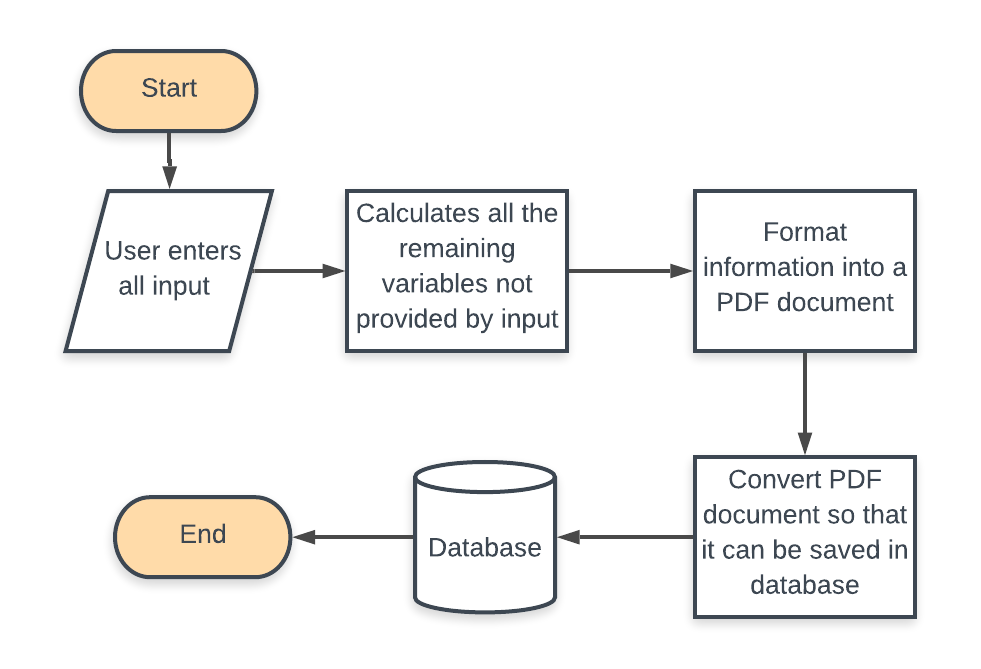
* This algorithm is called every time a user clicks on the ‘remove document’ or ‘remove client’ button in the client overview page
* When the user clicks on the button that corresponds to the document or client they want to delete, the title of the document/client will then be cross-checked in the database using either the document or client id to make sure that if the document/client exists
* If the document/client exists, the document/client will then be deleted from the database and a popup window displaying the message ‘Document/Client Successfully Deleted’ will then appear
* If the document/client does not exist, a popup window with the message ‘Document/Client does not exist’ will appear.

*Update Client/Document*

****

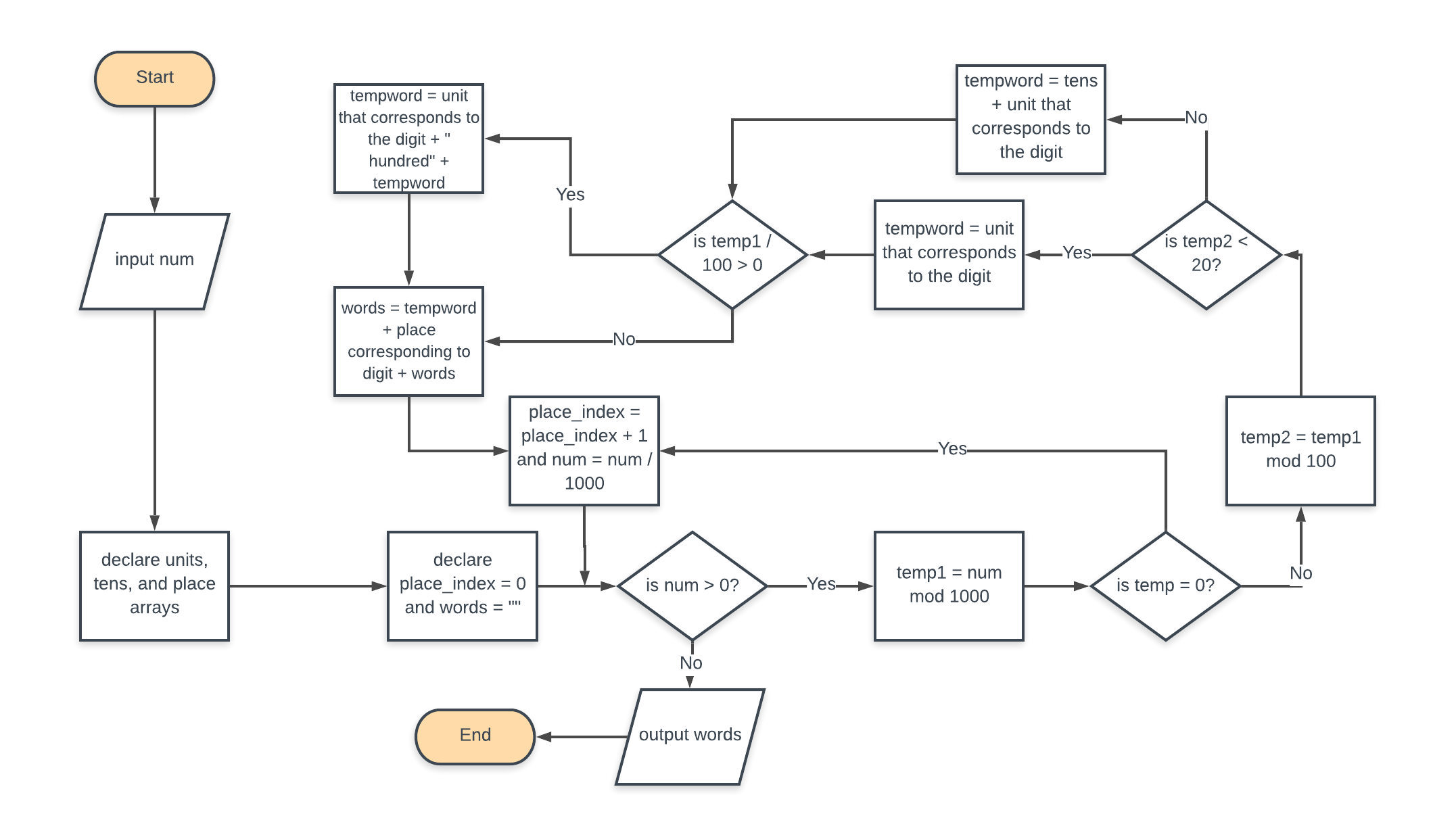
* User will fill out a form to update client/document and the input will be used as the new document/client information
* The program will then save the id of the document/client at which is being modified
* This id is then double checked in the database to see if the client/document exists
* If it does not exists, it will display an error message
* If it exists, the algorithm will then update the client/document information in the database and display a success message

*Making Documents*



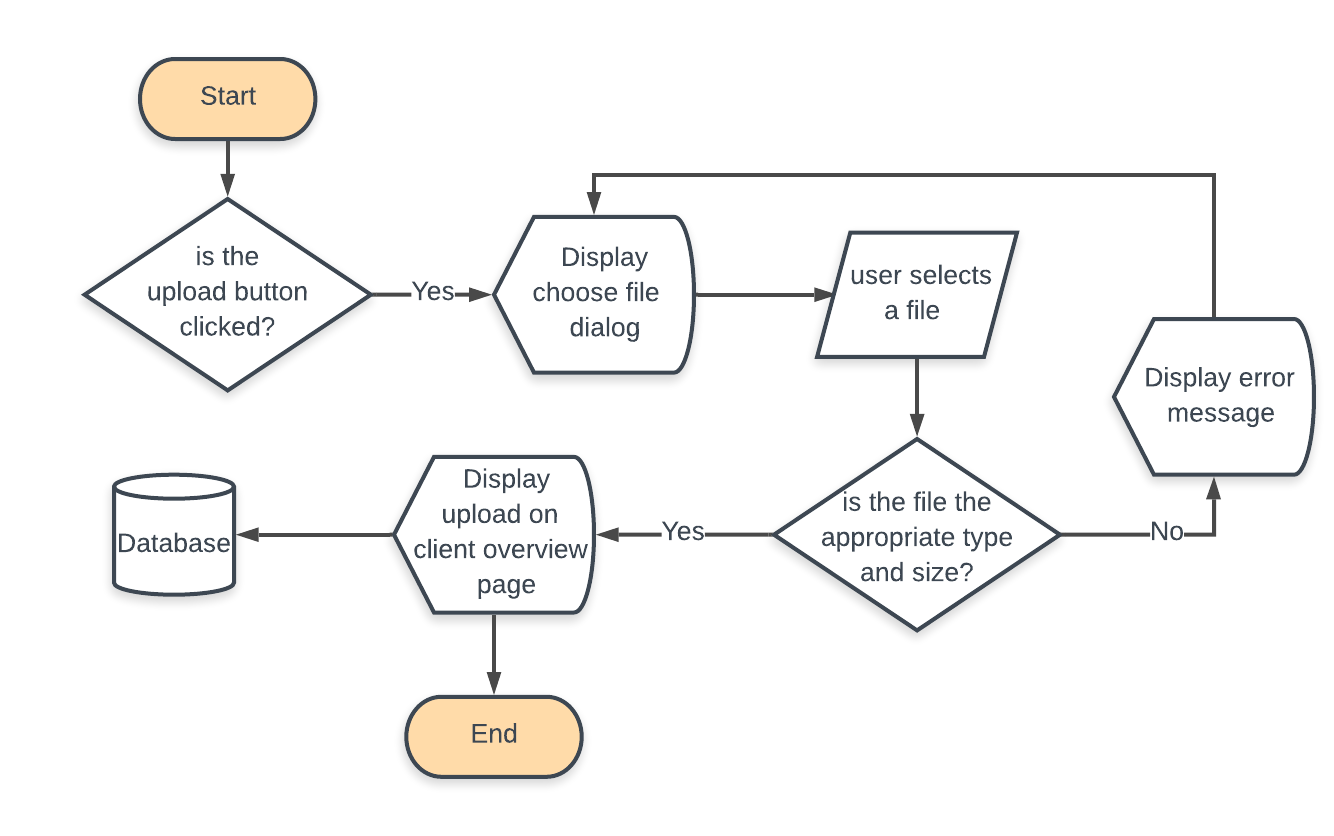
* This algorithm is needed to make a complete Tax Form document based on the inputs provided by the user
* User will enter all of the necessary input and that input will be used to calculate remaining variables that can be determine based on the input values (such as the Tax Object Sales Value)
* All of the information and calculated values is translated and structured into a PDF document
* Since PDF documents cannot be saved as it is in the database, it needs to be converted into a data structure that will support the PDF so it can be stored in the database

*Converting Numbers into Words*

****

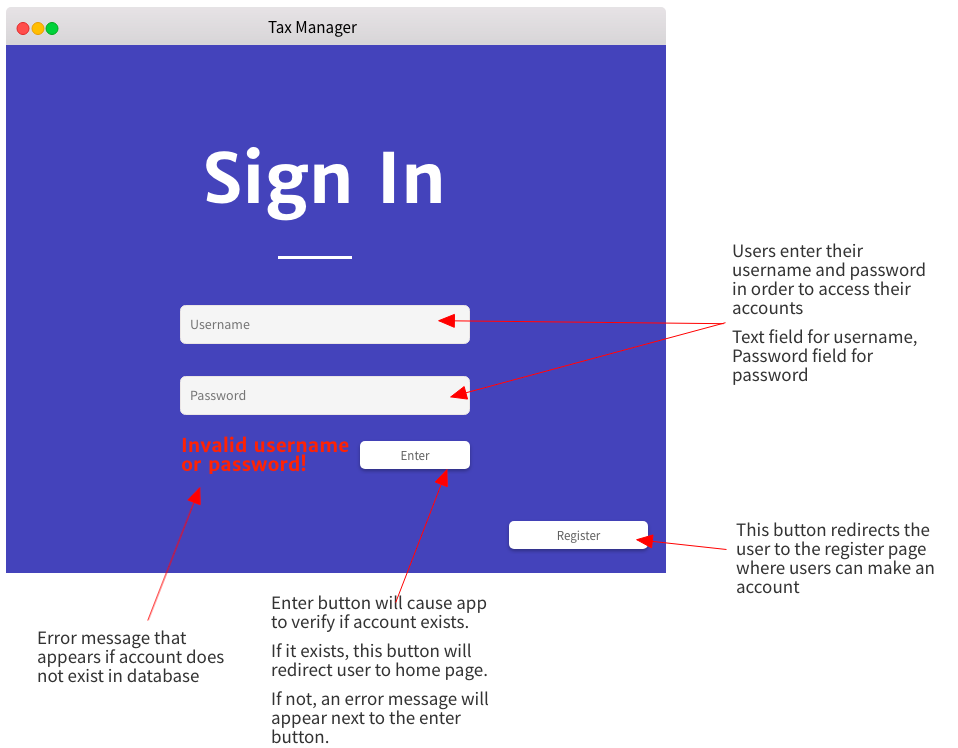
* This algorithm will be used in converting the MoneyDeposited value in integers into words, which is usually expected in a tax form
* Units, tens, and place are string arrays where the indices represent the word of the index for the units and tens array while the place arrays represent different place values in numbers. Below are examples of the contents of the array:
  + Units[9] = “nine”
  + Tens[3] = “thirty”
  + Place[2] = “thousand”
* Algorithm is called every time a document is made or when a document has been updated that changes the money deposited value, meaning the variable will need to be converted again.

*Upload Document/Image*

******

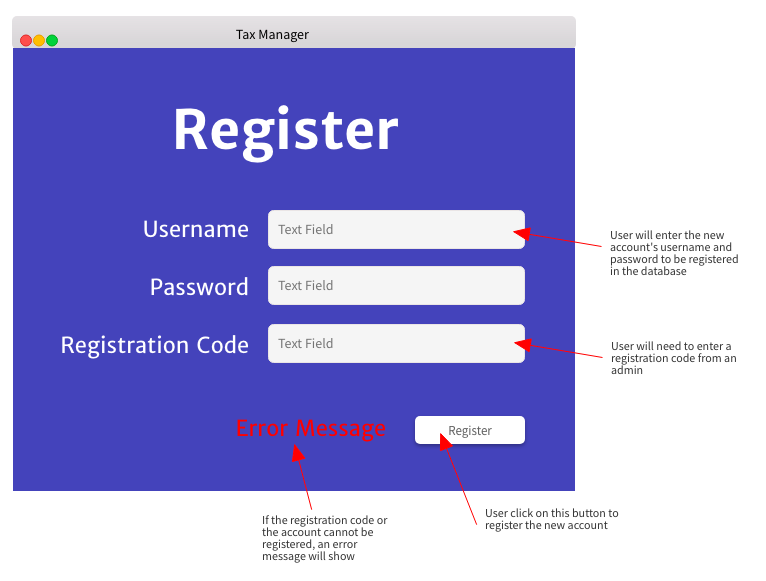
* The app checks if the user clicks on the upload button. If yes, a dialog for the user to choose a file will be displayed.
* User chooses a file.
* The app will then check if the file is of the appropriate type (.png or.jpg for image and .pdf for documents) and if the file is not too big.
* If the file is too big or not of the appropriate type, the user will then be shown the choose file dialog again to rechoose a file.
* If the file is approved, the image or the document will then be displayed on the client overview page and saved in the database.

**Mock Ups** (Made using Mockflow)[[1]](#footnote-0)

*Sign in Page*

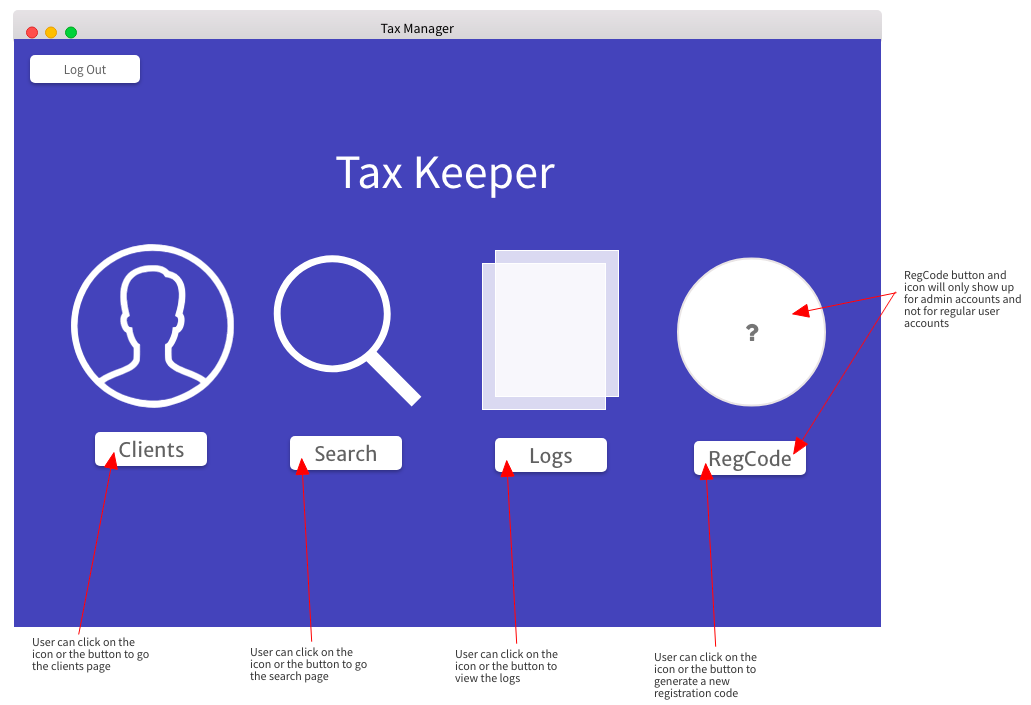
* Username is a text field and Password is a password field
* ‘Enter’ and ‘Register’ are both buttons
* ‘Sign in’ is a label

*Register Page*

**

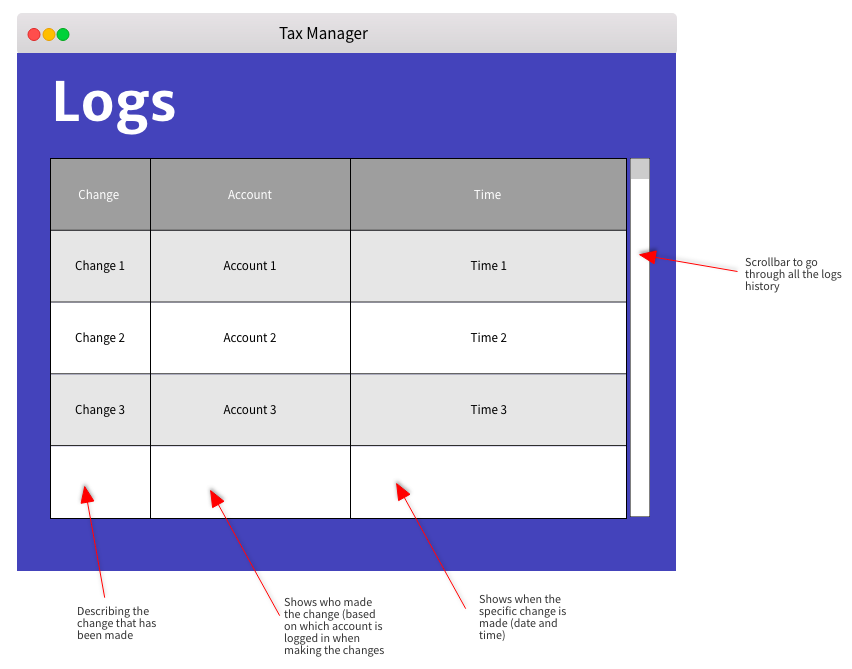
* “Register”, “Username”, “Password”, “Registration code”, “Error Message” are all labels
* User enters their desired username, password, and their registration code on available text fields
* The “register” button is clicked to register the new account

*Home Page*



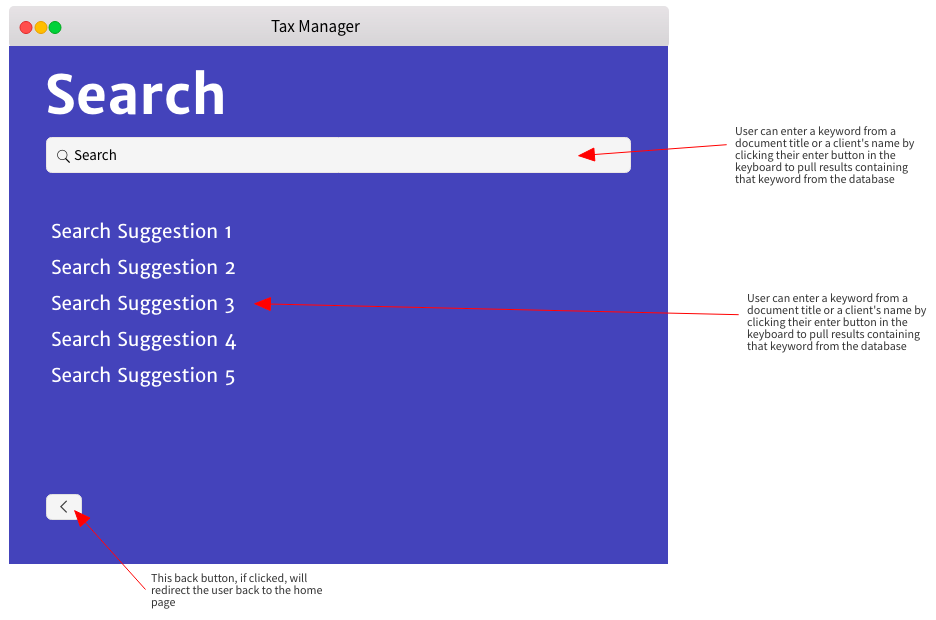
* “Log out”, “Clients”, “Search”, “Logs”, and “RegCode” along with the visuals above them are buttons that redirect to their corresponding pages which are the Log in page, Clients page, the search page, the logs page, and the regCode page respectively
* The RegCode button only shows up if the user that logs in is an admin as they are the only one who can generate a new registration code
* If the user who logs in is not an admin, the buttons available are only the log out button, the clients button, the search button, and the logs button.
* Title of the page “Tax Keeper” is a label

*Logs Page*

**

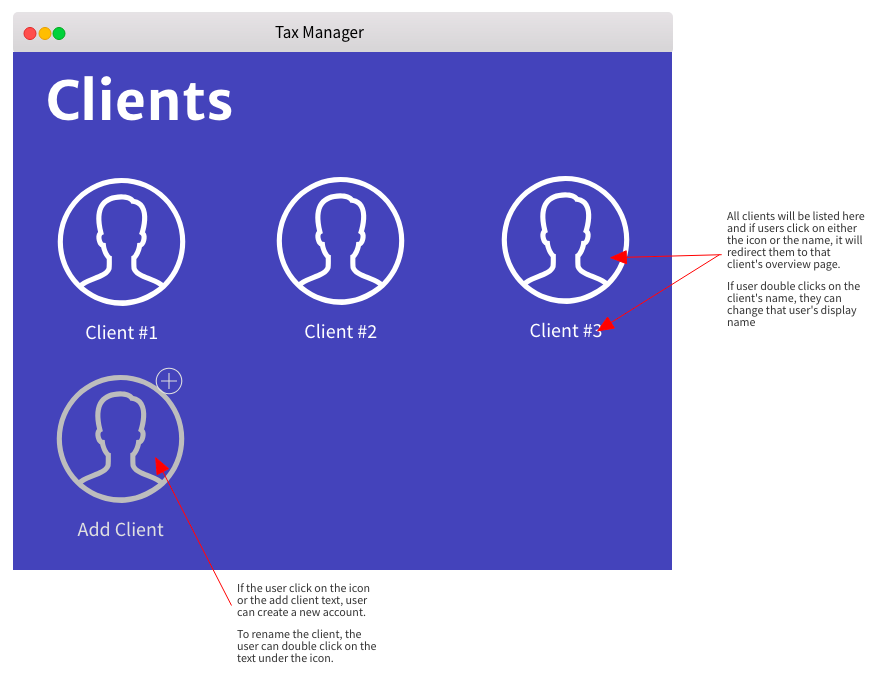
* The title “Logs” is a label
* Data will be represented in a table using TableView
* A scrollbar for the table will be provided so that user can go through all of the contents on the table
* Table is organized by time with the most recent change displayed at the top

*Search Page*

**

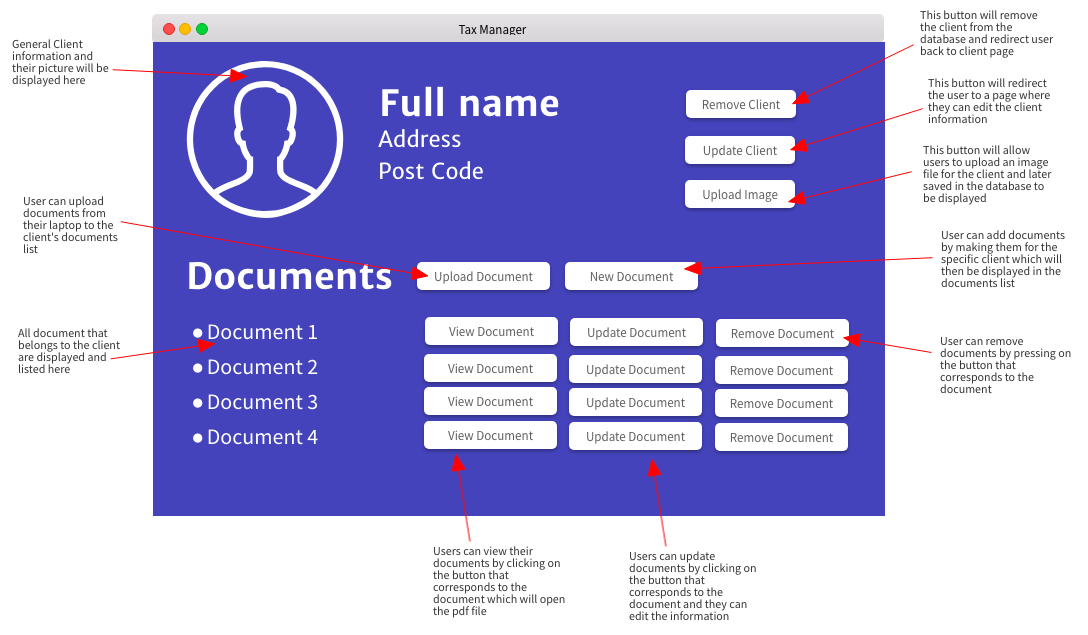
* The title ‘Search’ is a label
* Arrow in the bottom left corner is a back button that redirects the user to the home page
* The user enters the keyword to document/client that they are searching for in a text field and when they press their ‘enter’ key in their keyboard, it will provide search suggestions
* Search suggestions are displayed in labels that are clickable

*Clients Page*

**

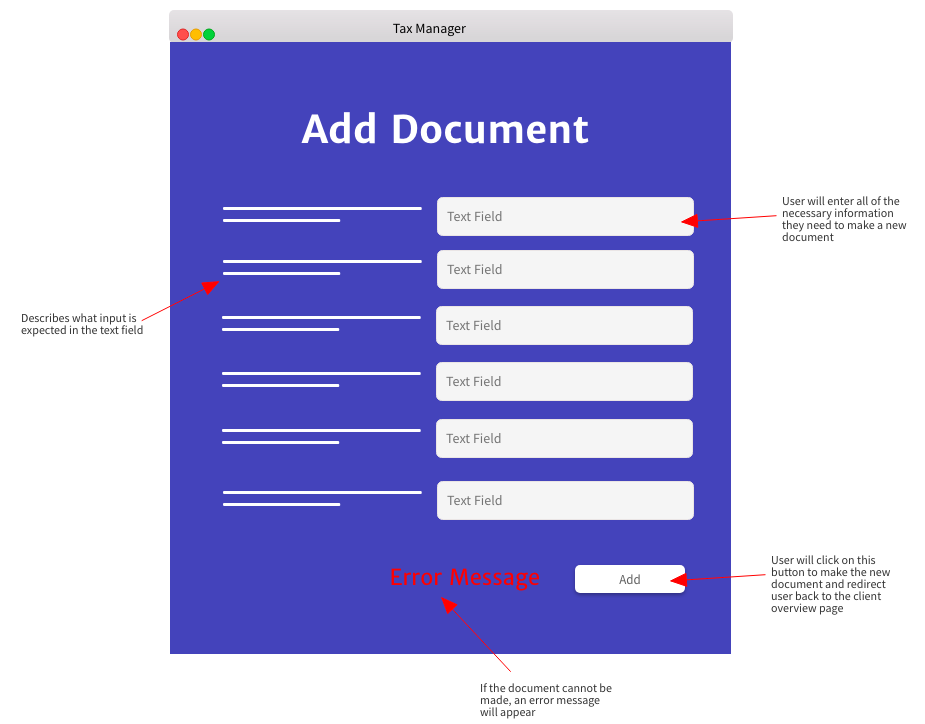
* The ‘Clients’ text on the top will be a label
* All the clients’ profile are all buttons (including add client)

*Client Overview Page*

**

* ‘Remove Client’, ‘Upload Document’, ‘New Document’, and ‘Remove Document’ are all buttons
* ‘Documents’, ‘Full Name’, image, ‘Address’, and ‘Post Code’ are all labels
* Each document in the documents list will all be clickable labels
* When removing a document or client, a confirmation page or dialog will be prompted

*Add or Update Document Form*



* “Add Document”, “Error Message”, and input descriptors are all labels
* Users will enter all their inputs in the provided text field
* “Add” is a button that will allow the user to create the document

**Software Development Framework**

The Agile System Development (ASD) Framework will be used because this app will have many features that can be divided into different components and handled like mini projects, which ASD offers. I plan to break the app down by functionality and work on each functionality at a time, so for instance, by working on the login and registration page first before moving on to the home page.This will help improve the quality of the product as each component is given as much attention as needed. With this framework, it will be easier to adjust to change as there isn’t a strict plan that will need to be followed and it will be easier to extend the app further by adding more functionalities that wasn’t previously considered. ASD also emphasizes communications between stakeholder and therefore, I will work closely with the client by frequently updating her with the most recently added functionality of the app and contacting her whenever I have questions regarding the app.

**Testing Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Action to be Tested | Test Method | Expected Output |
| 1 | New accounts for a user can only be registered with a randomly generated registration code made by the admin once they log in to their account first. New accounts are then saved in the ‘user’ table of the database. | Try to register new accounts in the register page with three possible scenarios:   1. Not entering anything on the registration code field 2. Entering an incorrect registration code 3. Entering the correct admin password | New users are only registered if the correct admin password is entered in the registration code field of the register page. New user’s credentials (username and password) are entered into the database and can be used to log in to the app. If the registration code is left blank or entered incorrectly, it will not register the account and an error label is displayed. |
| 2 | Users can log in to their accounts by entering their username and password which will then be checked whether this account is registered in the database. | The possible input scenarios for the log in mechanism are:   * Not entering anything in the username and password field * Entering the wrong username but correct password * Entering the correct username but wrong password * Entering the correct username and password | Users can only log in if both the username and password is entered correctly. Otherwise, there should be an error message that says “Invalid”. |
| 3 | Users can easily search for a document or a client in the database through a search bar based on the title of the documents or names of the clients. | The possible input scenarios for the search mechanism are:   * Not entering anything on the search field * Entering a non-existing client or document * Entering an incomplete document name or client name * Entering a keyword that is both in a client’s name and document’s name | Search results are displayed below the search bar even if the keyword entered is not the complete name of a client or a document. Searching a non-existing client or document should display the message “Search results not found”. If nothing is entered on the search bar, then there should be nothing displayed. Search results will need to display all clients and documents whose name contains the keyword. |
| 4 | Clients have their own profiles which contain all of their documents | * Client Overview page can be opened by selecting a specific client in the clients page * Client Overview page can be opened from the search page when a client is selected | The client overview page can be opened through the search and clients page. It should feature a picture of the client, an upload button for the picture, button to delete client, all of the client’s information (client’s name, address, etc.), button to edit the client, a list of all the clients’ documents, and buttons to view/edit/update/add/delete documents in the client overview page. |
| 5 | Users can create new client profiles, edit and delete them. Changes made to the client will be reflected in the database. | In the clients page, users should be able to click on the “add client” button. User should check for the following scenarios:   * Click on the “cancel” button before entering anything on the fields * Click on the “cancel” button with fields filled out * All fields are entered and the button “register account” is clicked   In the overview page, there will be the edit and delete buttons. User should test for the following scenarios:   * “Delete” button is clicked but user selects “no” button in the confirmation dialog * “Delete” button is clicked and the user selects “yes” in the confirmation dialog * “Update” button is clicked, the user modifies a few information, but user clicks on the “cancel” button * “Update” button is clicked, the user modifies a few details and clicks on the “Update” button | The add client button in the clients page should redirect the user to a form where the user can input data about the new client. Clients will only be created if all fields are entered and the “register” button is clicked. Otherwise, the “cancel” button will only redirect the user back to the clients page and discard all of the information entered in the add client form. In the overview page, the “Delete” button for the client should show the user a dialog that asks the user to confirm whether they want to delete the client. If the user clicks on “no”, the user should be redirected back to the client overview page. If the user clicks on “yes”, the user should be redirected to the clients page with that specific client deleted or erased from the list. When the user clicks on “update”, the client's data will only change if the user clicks on the “update” button. If user clicks “cancel”, all information entered is discarded and the client’s information remains unchanged. |
| 6 | Users can easily create a new document for a client by entering the required inputs. The inputs will then be added to the ‘document’ table of the database. The app will programmatically complete aspects of the document that can be calculated. | The user should click on the “add document” button and test for the following scenarios:   * Entering the fields and clicking “cancel” * Entering the fields and clicking “add document” | If the user clicks on “cancel”, the user will be redirected back to the overview page and the data entered is discarded. If the user clicks “add document” the user will be redirected to the overview page and a new document is created and displayed in the overview page and databasel. |
| 7 | Documents should be saved and presented/displayed in a PDF format when opened. Documents should also be saved in the database. | * Check on the database on the data type format at which the pdf is saved under * User clicks on “view” button for any of the documents in the overview page or by clicking the document in the search page | PDFs should be saved in the database table in a data type that will be easy to access when the app runs. When clicking the “view” button or choosing a document through the search page, it should open as a PDF document. |
| 8 | Users can delete, read, and edit existing documents. When deleting or updating the documents, these changes will then be reflected in the ‘document’ table of the database. | * User should click on the “delete” button for a document and click “no” on the dialog * User should click on the “delete” button for a document and click “yes” on the dialog * User should click on the “view” button for a document * User should click on the “update” button for a document and enter the fields but click “cancel” * User should click on the “update” button for a document and enter the fields and click “update” | Clicking the “delete” button and clicking “yes” on the dialog will erase the corresponding document from the database table and also on the overview page. However, if the user clicks “no” on the dialog, the document will not be erased and the dialog is closed. The “view” button will open the PDF document on the desktop. The “update” button will redirect the user to a form where users can fix the inputs for the document. Clicking the “cancel” button will redirect users back to the overview page and discard all of the information entered previously. Entering all the fields and clicking “update” will modify the information on the PDF document and all the calculations. Once updated, the user can click on “view” and notice the changes that have been made. |
| 9 | Users can upload documents to the app. The newly uploaded documents will then be stored in the ‘document’ table of the database. | In the overview page, the user clicks on the “upload document” and select any file that they can select. User should try uploading a small document (one that is similar to the one that the app makes) and another larger document (greater than 100KB) | The “upload document” button will open a window where users will be able to select a file. Users are limited to only select PDF files. Once they have selected the file, the document name and the file itself will be saved in the database and displayed in the overview page. If the size of the document is too big, an error dialog will appear instead and the file will not be uploaded to the database. |
| 10 | Each client can have a picture identifying them and featured in the Clients Page, Overview Page, and when searched in the Search Page. This picture will be saved in the database. | The user should click on the “upload image” button where they can choose an image for the client. The user should try uploading an image with a small size and another image with a bigger size. | Once the button is clicked, a window is opened where users can choose an image. Users can only select images that are .jpg, .jpeg, or .png. Once they have selected an image, it is displayed instead of the placeholder image and saved in the database. If the image is too big, an error dialog will appear instead and the image will not be uploaded to the database or displayed in the overview page. |
| 11 | A log page will outline all of the changes made in the application by detailing what is changed, who changed it, and when. | User should try using different accounts and do the following operations:   * Adding a new client * Updating client information * Deleting client * Adding document * Upload document * Delete document * Adding a new user | When the user opens the log page, which can be accessed from the home page, there should be a table outlining recent changes starting from the most recent change. The changes should be described in the “change” column, the account making the change should be listed in the “account” column, and the date and time should be listed in the “time” column. If the user makes a new change in the application, it should be immediately reflected on the logs page. |

1. Produle. “MockFlow - Wireframe Tools, Prototyping Tools, UI Mockups, UX Suite.” *Mockflow.Com*, 2009, mockflow.com/. [↑](#footnote-ref-0)