Project Catawba Area Agency On Aging Database (CAAAD)

System Manager Documentation

Catawba Area Agency on Aging

Winthrop University CSCI475 Team # 2

Justin Carmona, Jordan Douglas, AJ Ellis, Preston Fisher, Walker Immel, Christian Morris, Donnie Sullivan, Mario Delgado

Computer Science & Quantitative Methods Department
College of Business Administration
WINTHROP UNIVERSITY

Table of Contents

1. Overview

1.1 Current Functional Requirements

1.1.1 The system will contain a database of clients and their associated parameters.

1.1.1.1 Input: N/A

1.1.1.2 Output: N/A

1.1.2 The system will allow the user to import records to the database from Excel files.

1.1.2.1 Input: Client records contained in Excel files. (See Section 5.4)

1.1.2.2 Output: N/A

1.1.3 The system will allow manual input of client records into the database by the user.

1.1.3.1 Input: User input, client parameters that make up a client record.

1.1.3.2 Output: N/A

1.1.4 The system must allow modification of existing client parameters within the database by the user.

1.1.4.1 Input: User input, modifications to existing client parameters.

1.1.4.2 Output: modified record.

1.1.5 The system must allow deletion of previously entered client records from the database.

1.1.5.1 Input: User will identify the record to be deleted

1.1.5.2 Output: N/A

1.1.6 The system will prompt the user to resolve duplicate client records.

1.1.6.1 Input: A client record which already exists in the database.

1.1.6.2 Output: Duplicate record prompt.

1.1.7 The system must be able to filter the database by parameters.

1.1.7.1 Input: User input, filtering client parameters.

1.1.7.2 Output: Filtered database

1.1.8 The system will allow the user to be able to generate reports.

1.1.8.1 Input: client parameters to filter for in the database

1.1.8.2 Output: Generated reports based on user filtered client parameters.

1.2 Current Non-Functional Requirements

1.2.1 The system must be accessible through the organization's intranet system.

1.2.2 The system will only be accessible by the organization's staff members.

1.2.3 The system will emphasize ease of use with a simple interface as tested and verified by the customer.

1.2.4 The system will take at most 10 seconds to return all applicable results when filtering by client parameters.

1.2.5 The system must have the storage capacity to accommodate all existing client data.

1.2.6 The system will be able to scale for at least double the initial amount of client records given enough storage is provided.

1.3 Requirements Traceability Matrix

Project Catawba Area Agency on Aging Database					Team 2		
Requirements Information and Status							
Req. ID	Req. Desc.	Ref. ID	Date Estab.	Validation Method	Status	Status Date	Prod. Date
1	Database	2.2.1	Oct.22	Inspection	Complete	Apr. 23	Apr.23
2	Import Records	2.2.2	Oct.22	Testing	Complete	Apr. 23	Apr.23
3	Input Records	2.2.3	Oct.22	Testing	Complete	Apr. 23	Apr.23
4	Modify Records	2.2.4	Oct.22	Testing	Complete	Apr. 23	Apr.23
5	Delete Records	2.2.5	Oct.22	Testing	Complete	Apr. 23	Apr.23
6	Resolve Duplicate	2.2.6	Oct.22	Testing	Complete	Apr. 23	Apr.23
7	Filter Database	2.2.7	Oct.22	Testing	Complete	Apr. 23	Apr.23
8	Generate	2.2.8	Oct.22	Testing	Complete	Apr.	Apr.23

	Reports					23	
9	Intranet	2.3.1	Oct.22	Inspection	Initial	Apr. 23	Apr.23
10	Access Control	2.3.2	Oct.22	Inspection	Complete	Apr. 23	Apr.23
11	Ease of Use	2.3.3	Oct.22	Inspection	Complete	Apr. 23	Apr.23
12	Query Time	2.3.4	Oct.22	Testing	Complete	Apr. 23	Apr.23
13	Storage	2.3.5	Oct.22	Inspection	Initial	Apr. 23	Apr.23
14	Scalable	2.3.6	Oct.22	Inspection	Complete	Apr. 23	Apr.23

1.4 Data Dictionary

Field Name	**Field Description**
Last Name	Client's Last Name
First Name	Client's First Name
Address	Client's Address
Apt Number	Client’s Apt Number
PO Box	Client’s PO Box
City	Client's City
State	Client's State
Zip Code	Client's Zip Code
Phone Number	Client's Phone Number which can be used to contact the client
Date of Birth	Client's Date of Birth
Income	Client's Income

Marital Status	Client's Marital Status
Size of Household	Client's Size of Household
Medicare	Client's Medicare
Medicaid	Client's Medicaid
Comment	Client's Comment
Information, Referral and Assistance (IR&A)	Is the client receiving Information, Referral and Assistance (IR&A) services?
Medicare Counseling (SHIP)	Is the client receiving Medicare Counseling (SHIP) services?
Caregiver	Is the client receiving Caregiver services?
Home Delivered Meals	Is the client receiving Home Delivered Meals services?
Congregate Meals	Is the client receiving Congregate Meals services?
Home Care	Is the client receiving Home Care services?
Evidenced Based Health Promotion/Disease Prevention and Transportation	Is the client receiving Evidenced Based Health Promotion/Disease Prevention and Transportation services?
Legal Service	Is the client receiving Legal Service services?
Consumer Choice	Is the client receiving Consumer Choice Program?
Transportation	Is the client receiving Transportation?
Date Admitted	Date the client was admitted.

1.5 Project Specific Terms
The definitions of Terms that are specific to this project and its requirements are listed below.

Client: The individual person whose information is being used in the database.

Client parameter: See glossary entry for “parameter”.

Client record: See glossary entry for “record”.

CSV file: a text file in which the data fields are separated by commas

Developer: Team 2: Justin Carmona, Jordan Douglas, AJ Ellis, Preston Fisher, Walker Immel, Christian Morris, Donnie Sullivan

Duplicate record: A record in the database that has the same [Name] and [Address] as another record.

Excel Files: An Excel file (.xls, .xlsx, etc) containing client parameter data.

Intranet: A local or private computer network.

NYI: Not Yet Implemented - feature not demonstrated in the current prototype.

Parameter: An individual field associated with a client representing one of the following data types:

1) Last Name
2) First Name
3) Address
4) City
5) State
6) ZIP Code
7) Phone Number
8) Date of Birth
9) Income
10) Marital Status
11) Size of Household
12) Medicare
13) Medicaid
14) Social Worker
15) In-Home Services Received
16) Emergency Contact Name
17) Emergency Contact Phone Number
18) Emergency Contact’s Relationship to Client
19) Comment
20) Information, Referral and Assistance (IR&A)
21) Medicare Counseling (SHIP)
22) Caregiver
23) Ombudsman
24) Assessments
25) Home Delivered Meals
26) Congregate Meals
27) Home Care
28) Evidenced Based Health Promotion/Disease Prevention and Transportation
29) Legal Service

Record: A collection of parameters that is associated with a client in the database.

Report: Output of filtered client data by specific parameter(s) as a CSV file and presented in a readable manner in our program UI.

User: Average Catawba Area Agency On Aging staff member.

<u>Admin:</u> Higher Up Catawba Area Agency On Aging staff member or IT personnel.

1.6 Known issues

The Known issues with the current version of the system are listed below

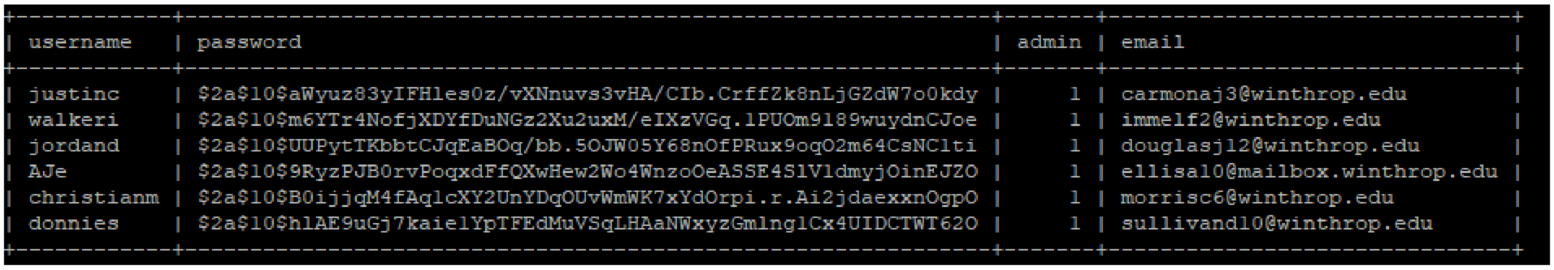
- The system has trouble displaying properly on some different monitors and screens but these are are only screens that are sub 1080p in resolution

2. Database Description

2.1 Database and table description

Our System has one main mysql database from which it pulls and uploads the different information that it needs to function. This database will be hosted on a server owned by The Catawba Area on Aging Agency and will be accessed using the organization intranet system. Within this mysql database there are two main tables:

2.2 Login Table



```
+------------+--------------------------------------------------------------+-------+-------------------------------+
| username   | password                                                     | admin | email                         |
+------------+--------------------------------------------------------------+-------+-------------------------------+
| justinc    | $2a$10$aWyuz83yIFHles0z/vXNnuvs3vHA/CIb.CrffZk8nLjGZdW7o0kdy |     1 | carmonaj3@winthrop.edu        |
| walkeri    | $2a$10$m6YTr4NofjXDYfDuNGz2Xu2uxM/eIXzVGq.1PUOm9189wuydnCJoe |     1 | immelf2@winthrop.edu          |
| jordand    | $2a$10$UUPytTKbbtCJqEaBOq/bb.5OJW05Y68nOfPRux9oqO2m64CsNC1ti |     1 | douglasj12@winthrop.edu       |
| AJe        | $2a$10$9RyzPJB0rvPoqxdFfQXwHew2Wo4WnzoOeASSE4S1V1dmyjOinEJZO |     1 | ellisa10@mailbox.winthrop.edu |
| christianm | $2a$10$B0ijjqM4fAq1cXY2UnYDqOUvWmWK7xYdOrpi.r.Ai2jdaexxnOgpO |     1 | morrisc6@winthrop.edu         |
| donnies    | $2a$10$h1AE9uGj7kaie1YpTFEdMuVSqLHAaNWxyzGm1ng1Cx4UIDCTWT62O |     1 | sullivand10@winthrop.edu      |
+------------+--------------------------------------------------------------+-------+-------------------------------+
```

*note that the data inside the table is just example data

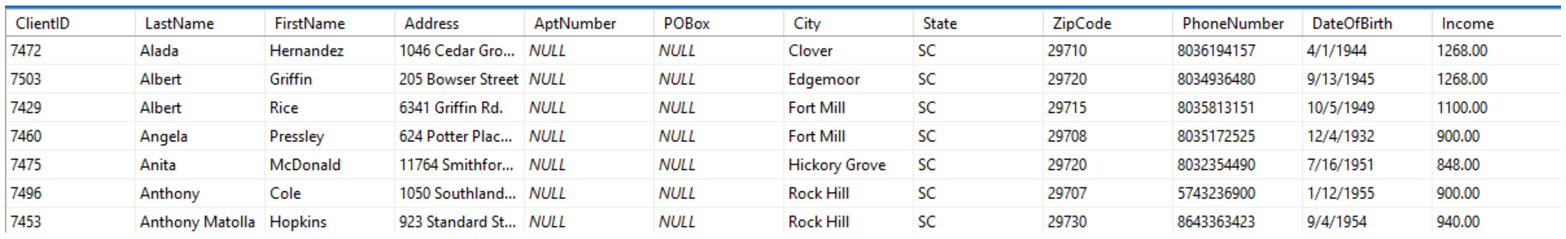
- The login table contains a list of a user and admin accounts that have been created. Each account is defined by the following categories:

Field Name	**Field Description**	**Type/Length**	**Required**
Username	User's account username	VARCHAR(50)	Yes
Password	User's account password	VARCHAR(75)	Yes
admin	Indicates if an account has admin privileges if	INT	yes

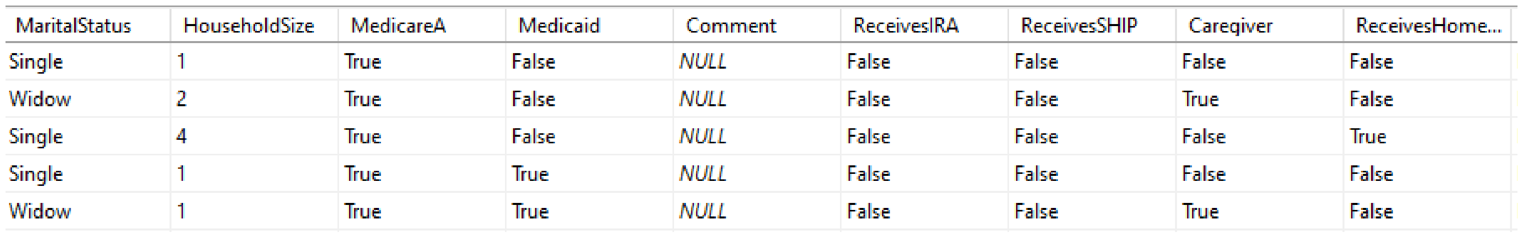
	account has admin privileges the number in the column will be a 1 if the account does not have admin privileges the number will be a 0		
email	User email address	VARCHAR(50)	yes

- Passwords are encrypted using a salted hashing algorithm known as Bycrpt for increased security these hashed versions of passwords are created locally by the program and then decrypted by the program whenever they are pulled from the database

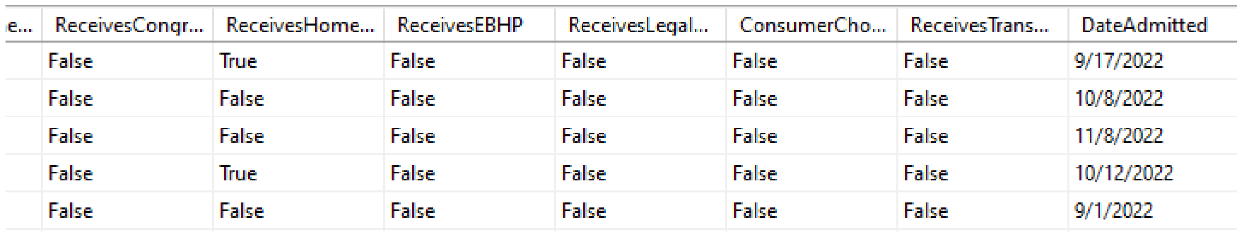
2.3 Client Demographic Table



ClientID	LastName	FirstName	Address	AptNumber	POBox	City	State	ZipCode	PhoneNumber	DateOfBirth	Income
7472	Alada	Hernandez	1046 Cedar Gro...	*NULL*	*NULL*	Clover	SC	29710	8036194157	4/1/1944	1268.00
7503	Albert	Griffin	205 Bowser Street	*NULL*	*NULL*	Edgemoor	SC	29720	8034936480	9/13/1945	1268.00
7429	Albert	Rice	6341 Griffin Rd.	*NULL*	*NULL*	Fort Mill	SC	29715	8035813151	10/5/1949	1100.00
7460	Angela	Pressley	624 Potter Plac...	*NULL*	*NULL*	Fort Mill	SC	29708	8035172525	12/4/1932	900.00
7475	Anita	McDonald	11764 Smithfor...	*NULL*	*NULL*	Hickory Grove	SC	29720	8032354490	7/16/1951	848.00
7496	Anthony	Cole	1050 Southland...	*NULL*	*NULL*	Rock Hill	SC	29707	5743236900	1/12/1955	900.00
7453	Anthony Matolla	Hopkins	923 Standard St...	*NULL*	*NULL*	Rock Hill	SC	29730	8643363423	9/4/1954	940.00



MaritalStatus	HouseholdSize	MedicareA	Medicaid	Comment	ReceivesIRA	ReceivesSHIP	Caregiver	ReceivesHome...
Single	1	True	False	NULL	False	False	False	False
Widow	2	True	False	NULL	False	False	True	False
Single	4	True	False	NULL	False	False	False	True
Single	1	True	True	NULL	False	False	False	False
Widow	1	True	True	NULL	False	False	True	False



e...	ReceivesCongr...	ReceivesHome...	ReceivesEBHP	ReceivesLegal...	ConsumerCho...	ReceivesTrans...	DateAdmitted
	False	True	False	False	False	False	9/17/2022
	False	False	False	False	False	False	10/8/2022
	False	False	False	False	False	False	11/8/2022
	False	True	False	False	False	False	10/12/2022
	False	False	False	False	False	False	9/1/2022

*note that the data inside the table is just example data

- The Client Demographic table contains a list of client Catawba's clients that are made of the following categories and types:

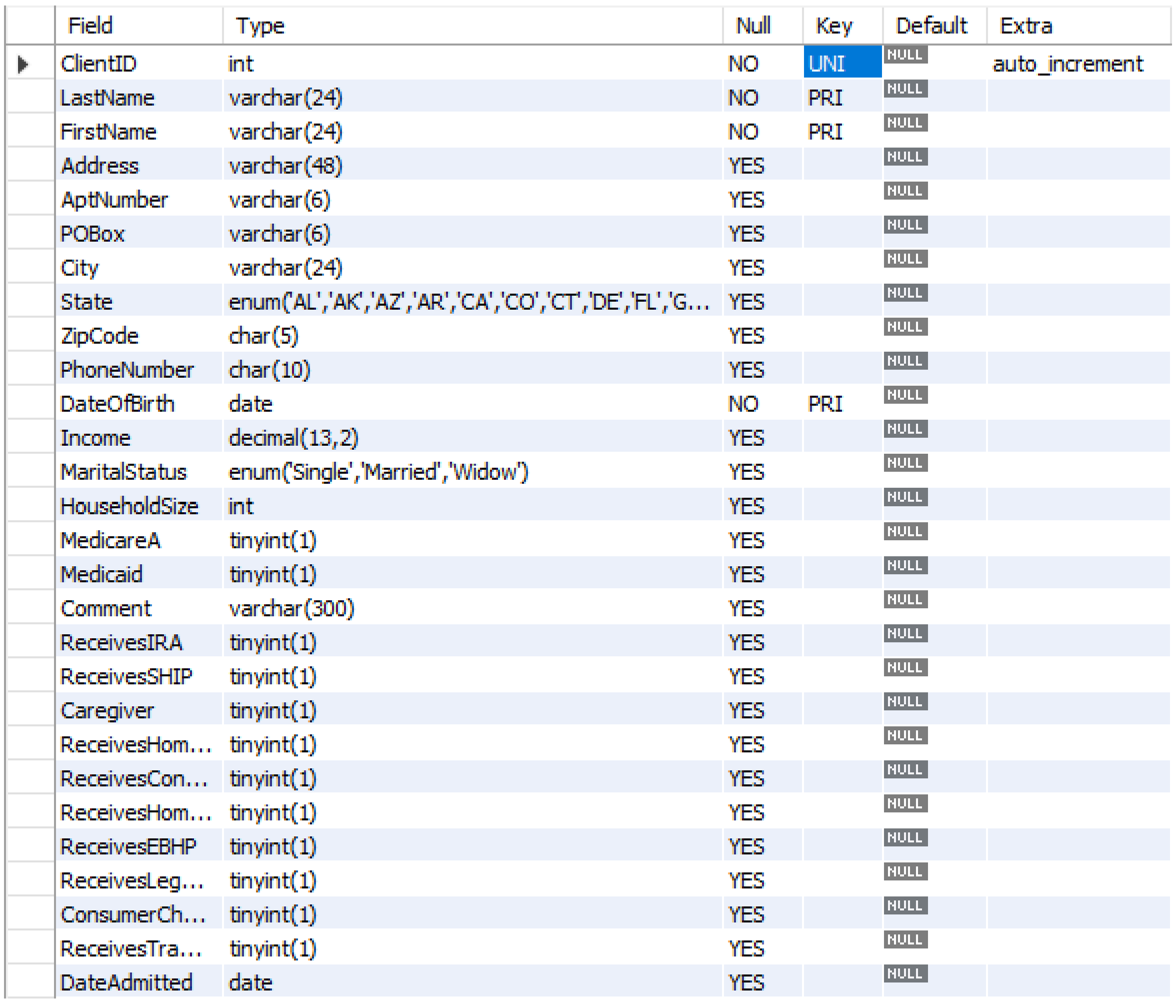
Field Name	**Field Description**	**Validation**	**Type/Length**	**Required**
ClientID	Client's ID	No	INT AI	Yes
LastName	Client's Last Name	Yes	VARCHAR(24) PK	Yes
FirstName	Client's First Name	Yes	VARCHAR(24) PK	Yes
Address	Client's Address	Yes	VARCHAR(48)	No
AptNumber	Client's Apt Number	Yes	VARCHAR(6)	
POBox	Client's PO Box	Yes	VARCHAR(6)	
City	Client's City	Yes	VARCHAR(24)	No
State	Client's State	No	ENUM('AL',...,'WY')	No
ZipCode	Client's Zip Code	Yes	CHAR(5)	No
PhoneNumber	Client's Phone Number which can be used to contact the client	Yes	CHAR(10)	No
DateOfBirth	Client's Date of Birth	Yes	DATE PK	Yes
Income	Client's Income	Yes	DECIMAL(13,2)	No

Field Name	**Field Description**	**Validation**	**Type/Length**	**Required**
MaritalStatus	Client's Marital Status	No	ENUM('Single','Married','Widow')	No
HouseholdSize	Client's Size of Household	Yes	INT	No
MedicareA	Client's Medicare	No	TINYINT(1)	No
Medicaid	Client's Medicaid	No	TINYINT(1)	No
Comment	Client's Comment	No	VARCHAR(300)	No
ReceivesIRA	Is the client receiving Information, Referral and Assistance (IR&A) services?	No	TINYINT(1)	No
ReceivesSHIP	Is the client receiving Medicare Counseling (SHIP) services?	No	TINYINT(1)	No
Caregiver	Is the client receiving Caregiver services?	No	TINYINT(1)	No
ReceivesHomeMeals	Is the client receiving Home Delivered Meals services?	No	TINYINT(1)	No
ReceivesCongregateMeals	Is the client receiving Congregate Meals services?	No	TINYINT(1)	No
ReceivesHomeCare	Is the client receiving Home Care services?	No	TINYINT(1)	No
ReceivesEBHP	Is the client receiving Evidenced Based Health Promotion/Disease Prevention and Transportation services?	No	TINYINT(1)	No
ReceivesLegalService	Is the client receiving Legal Service services?	No	TINYINT(1)	No

Field Name	**Field Description**	**Validation**	**Type/Length**	**Required**
Consumer Choice	Is the client receiving Consumer Choice Program?	No	TINYINT(1)	No
Transportation	Is the client receiving Transportation?	No	TINYINT(1)	No
Date Admitted	Date the client was admitted.	Yes	DATE	No

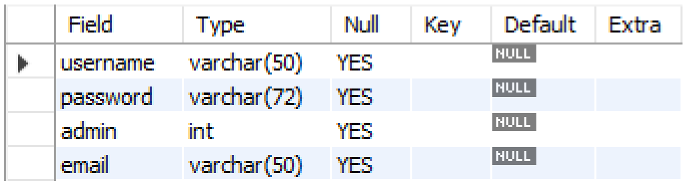
2.4 Database Initialization and Connection

The database used for connection with the software is configured for MySQL v.8.0.32. The database consists of two tables: **login** and **CLIENT_DEMOGRAPHICS**. The following schema can be used to create all fields associated with the **CLIENT_DEMOGRAPHICS** table:



	Field	Type	Null	Key	Default	Extra
▶	ClientID	int	NO	UNI	NULL	auto_increment
	LastName	varchar(24)	NO	PRI	NULL	
	FirstName	varchar(24)	NO	PRI	NULL	
	Address	varchar(48)	YES		NULL	
	AptNumber	varchar(6)	YES		NULL	
	POBox	varchar(6)	YES		NULL	
	City	varchar(24)	YES		NULL	
	State	enum('AL','AK','AZ','AR','CA','CO','CT','DE','FL','G...	YES		NULL	
	ZipCode	char(5)	YES		NULL	
	PhoneNumber	char(10)	YES		NULL	
	DateOfBirth	date	NO	PRI	NULL	
	Income	decimal(13,2)	YES		NULL	
	MaritalStatus	enum('Single','Married','Widow')	YES		NULL	
	HouseholdSize	int	YES		NULL	
	MedicareA	tinyint(1)	YES		NULL	
	Medicaid	tinyint(1)	YES		NULL	
	Comment	varchar(300)	YES		NULL	
	ReceivesIRA	tinyint(1)	YES		NULL	
	ReceivesSHIP	tinyint(1)	YES		NULL	
	Caregiver	tinyint(1)	YES		NULL	
	ReceivesHom...	tinyint(1)	YES		NULL	
	ReceivesCon...	tinyint(1)	YES		NULL	
	ReceivesHom...	tinyint(1)	YES		NULL	
	ReceivesEBHP	tinyint(1)	YES		NULL	
	ReceivesLeg...	tinyint(1)	YES		NULL	
	ConsumerCh...	tinyint(1)	YES		NULL	
	ReceivesTra...	tinyint(1)	YES		NULL	
	DateAdmitted	date	YES		NULL	

The **login** table can be created by referring to the schema below:



Field	Type	Null	Key	Default	Extra
username	varchar(50)	YES		NULL	
password	varchar(72)	YES		NULL	
admin	int	YES		NULL	
email	varchar(50)	YES		NULL	

The software will attempt to make a connection on **localhost** to the database **my_morrisc6_template1** using the username **my.morrisc6** and the password **team2**. All connection strings use SSL transfer security, so the data is encrypted while in transit. The connection string used within code segments that require database connection is:

"server=localhost;userid=my.morrisc6;password=team2;database=my_morrisc6_template1;SslMode=Preferred;persistsecurityinfo=True"

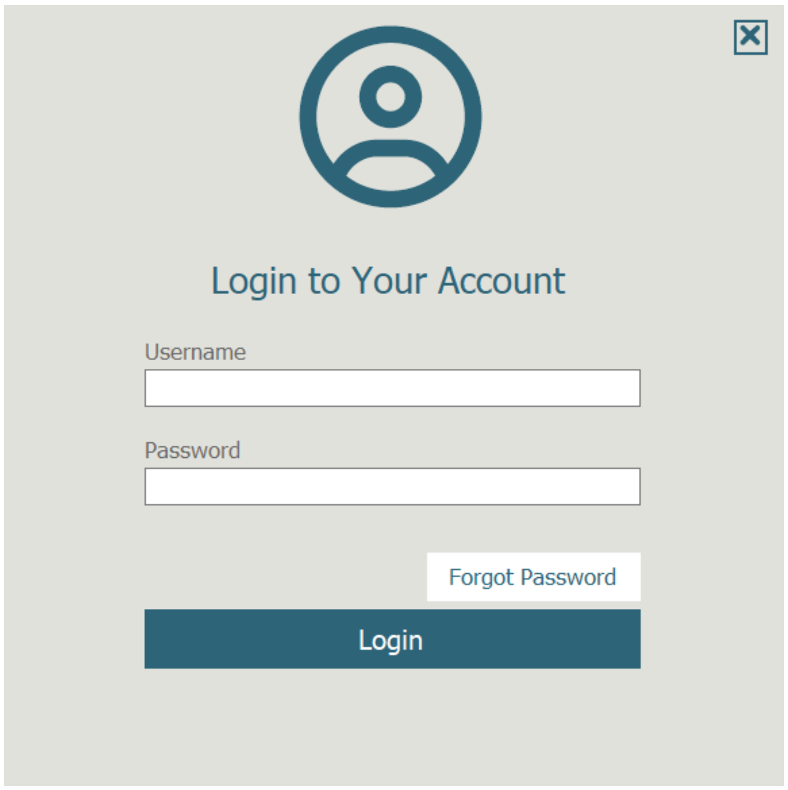
The database name as well as the username have kept their initial names to be used as a trace to contact a member of the group if needed. These connections can be changed, but would need to be altered in the following forms:

- EditClientForm
- EditUserForm
- ImportForm
- LoginForm
- NewClientForm
- NewUserForm
- ReportForm

The backup of database files should be handled by the server administrator, and is outside the scope of the production software. In order to manually backup the data (not the database or schema files) the export functionality of the software can be used. For more information on this, please refer to the user manual.

3. Code Documentation

3.1 LoginForm.cs

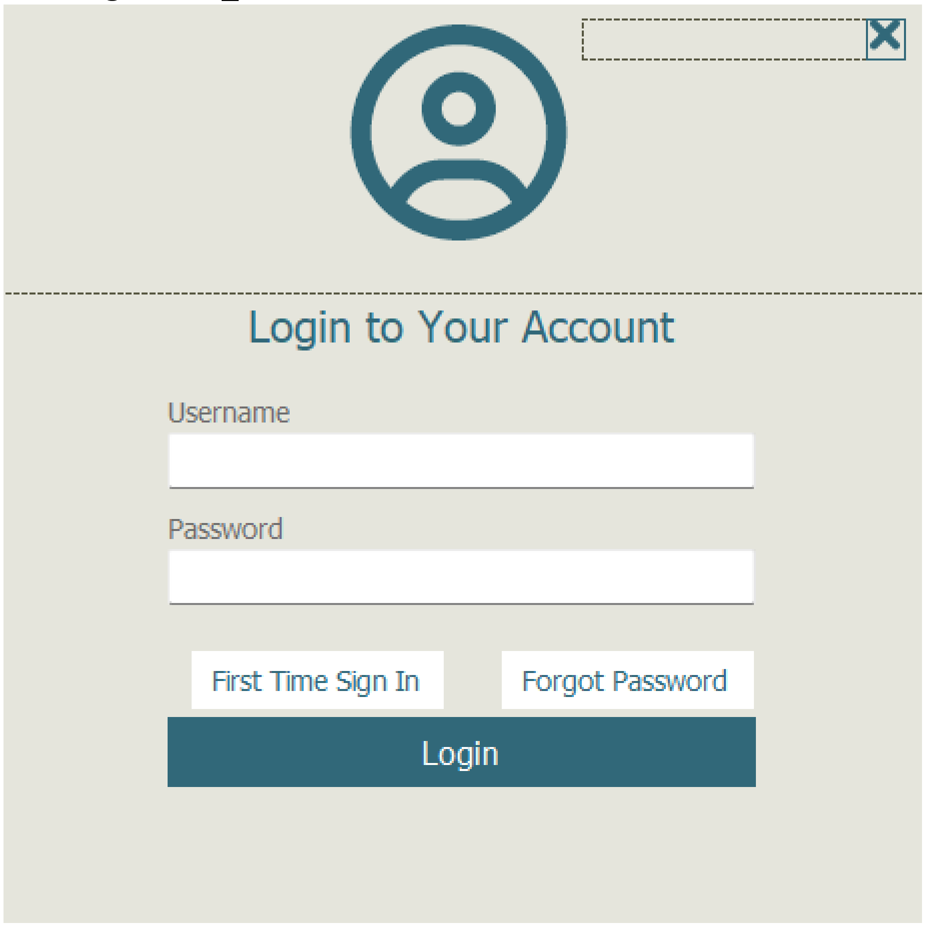


- The function of LoginForm.cs is to be the starting page for the CAAAD program; where the user can sign into their account and move to the main menu page, click the forgot password to begin the process of resetting their password for their account and if it is the first time the 'CAAAD program has been run and the login table in the database is empty allow the user to create a single admin user to sign into the program with. To accomplish these actions the LoginForm uses four main event functions:
    - LoginForm_Load
    - LoginBtn_Click
    - ForgotBtn_Click
    - firstTimebtn_Click

- The Login page references the following important variables/structures:
    - **string Myconn -** connection string containing information used to connect to the MySql database. Note to connect to a different database you would change the information in this string.

- **MySqlConnection conn** - used to hold the connection string for connecting to the Mysql database
- **MySqlCommand cmd** - used to hold the query to be run on the database
- **MySqlDataAdapter sda** - used to run the cmd query
- **MySqlDataReader reader** used to parse through specific parts of the data pulled from the database
- **DataTable dt** - holds the results of the query to be looked through
- **Admin** - integer var used to hold admin status that is pulled from the database
- **match**- boolean var used to hold the status of weather or not the password the user entered matches the one in the database.
- **UsernameTxt.Text** - holds the username entered by the user
- **PasswordTxt.Text** - holds the password entered by the user
- **firstTimebtn.Visible**- controls the visibility of the First Time Sign in Button
- **MainMenu f2** -links to the next page in the program known as MainMenu.cs
- **f2.urName** - public string var inside of MainMenu that holds the value of **UsernameTxt.Text** so that it can be used in the MainMenu page
- **f2.isAdmin** - public int var inside of MainMenu.cs that holds the value of **Admin** for use in MainMenu.cs
- **ForgorPassForm f3** -links to the forgot password page known as ForgorPassForm.cs
- **NewUserForm f4**-links to page known as NewUserForm.cs where new accounts are created

- ***3.1.1 LoginForm_Load***



- The LoginForm_Load function runs when LoginForm.cs is loaded. When this function is run it connects to the client database and checks to see if the login table has any accounts in the table of the database using **MySqlConnection conn, MySqlCommand cmd,MySqlDataAdapter sda ,MySqlDataReader reader** and **DataTable dt**. If no accounts are found a new button called **First Time Sign In** will appear on the login form This button is used for the firstTimebtn_Click function which will be discussed later.

- ***3.1.2 LoginBtn_Click***
- The LoginBtn_Click function is run when the “login button” is clicked and its purpose is to check the credentials entered by the user if the user exists then check their admin privileges and then take them to the MainMenu.cs form. The system does 2 kinds of checks:
    - The system checks to see if the username and password are blank. If any combination of the fields are blank the program will notify the user with an error. If both fields are filled the program will then move on the next check.
    - The system will check to see if the supplied username and password are correct by connecting to the established database using **MySqlConnection conn,**

MySqlCommand cmd,MySqlDataAdapter sda ,MySqlDataReader reader and **DataTable dt**.

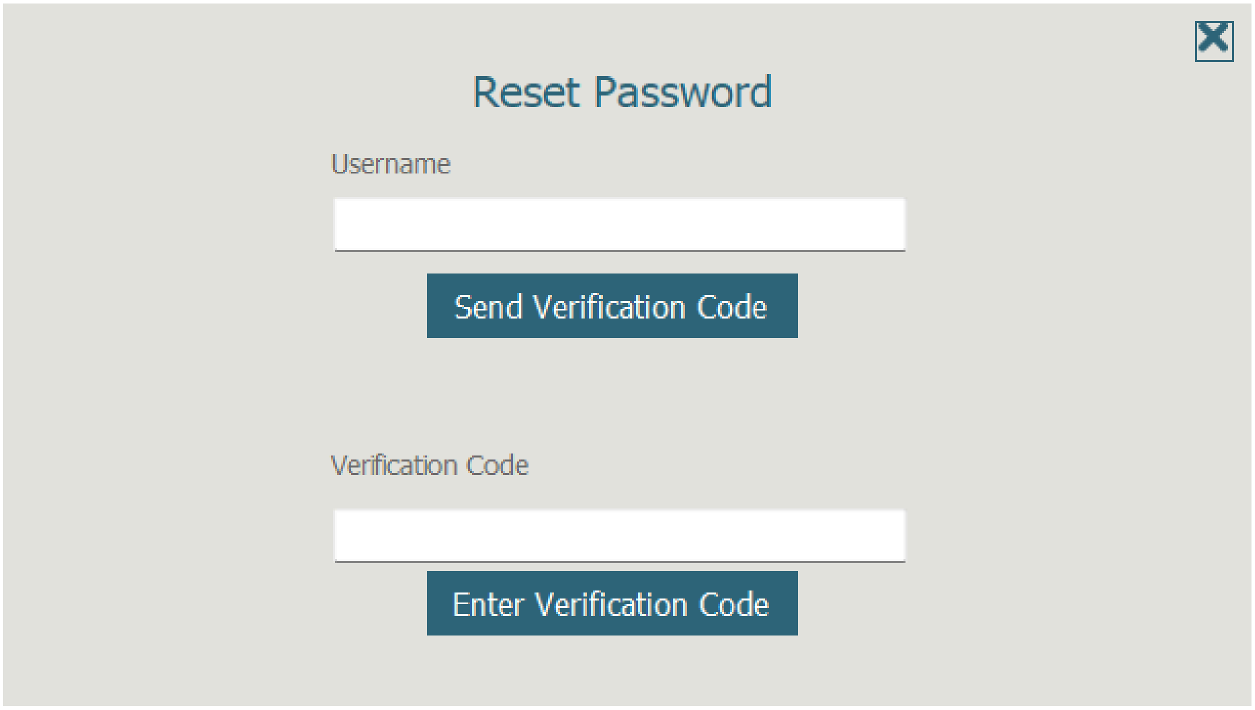
    - During this check the program also grabs the user account privileges using **MySqlDataReader reader** and determines if the user is an admin or just a regular user via an if and an else if statement. If Admin status is a 0 then the account is a regular user account. Else if Admin status is a 1 then the account is an admin account.
    - Note that the password is salted and Hash using the previously mentioned Bcrypt algorithm so when checking the entered password stored in **PasswordTxt.Text** the program decrypts the password found in the login table of the Mysql database. If the passwords match then the variable **Match** is set to true.

- If both Checks are successful the system will then and only then will the admin/user be logged into the program. From this point the system will assign the value of **UsernameTxt.Text** to **f2.urName** and the value of **Admin** to **f2.isAdmin** and then takes the user to the MainMenu page.

- ***3.1.3 ForgotBtn_Click***
- The ForgotBtn_Click function is run when the “forgot password” button is clicked at which point the program will close the login form and move the user to the ForgorPassForm.cs form. The only important variable referenced in this function is **ForgorPassForm f3** which links to the ForgorPassForm.cs form.

- ***3.1.3 firstTimeBtn_Click***
- The firstTimeBtn_Click function is run when the “First Time Sign In " button is clicked. The function starts off by closing the login form and setting **f4.firstTime** equal to true; This lets the system know that this is the first time the program is run. Then the system takes the user to a special version of the NewUserForm.cs form where the user creates the first account that all other accounts will be created from.This instance of the form is special because it only allows the user to create an admin account and not a regular user account.
  - Note this button can only be seen when ***LoginForm_Load*** is run and connects to the database and sees no existing accounts inside of the Login Table.

3.2 ForgorPassForm.cs



- The function of ForgorPassFrom.cs is to start the process of resetting a user's password by allowing the user to search for their account by their username and then send a verification code to the email address registered to that account. The verification code that is sent is a 6 character code consisting of letters and numbers that is generated using a Nuget package known as AAJGen. The email address containing the 6 digit code is sent using the FluentEmail.Core Nuget package and a gmail account. To accomplish this ForgorPassForm uses two main event functions:
    - sendCode_Click
    - entCode_Click

- To access the gmail account used to send these emails login into gmail using the following credentials:
    - Username:caad565@gmail.com
    - password :Team27832
    - Also uses 2 step verification (for more details ask your administrator)

- The ForgorPassWord form references the following important variables/structures:

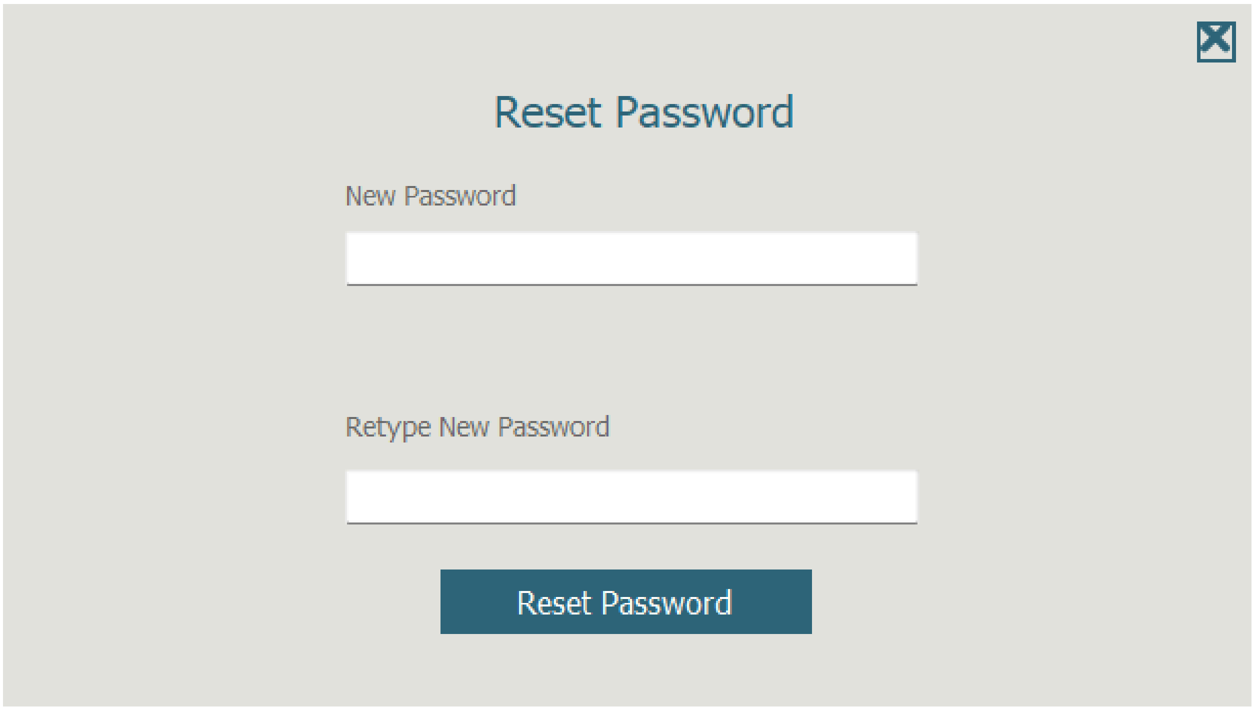
  - **email** - String var that is meant to hold the email address that is pulled from the login table of the database.
  - **UsernameTxt.Text** - holds the username entered by the user
  - **VerificationTxt.Text** - holds the verification code entered by the user
  - **sysemail** - string var that holds the email address of the gmail account that the verification code will be sent from.
  - **password** - string var that holds a special password form allotted to the program by the gmail account.
  - **ranNum** - global string var that holds the 6 character code to be sent in the email
  - **MimeMessage message** - holds the information that will be seen in the actual email sent to the user such as the who the email is from, who the email is going to, the subject line of the email address and the information contained in the body of the email address.
  - **SmtpClient client** - used to establish a connection with gmail's SMTP server, login to the previously mentioned gmail account using **sysemail** and **password** and send the message containing the verification code.

- ***3.2.1 sendCode_Click***
- The ForgotBtn_Click function is run when the "Send Verification Code" button is clicked and its main purpose is to send the verification code to the user's email address. To Accomplish this the program does the following:
  - program first checks if **UsernameTxt.Text** is empty and then checks if the username entered in **UsernameTxt.Text** actually exist in the login table of the database this is done using the same method as the login page.The only difference is this time is that it also pulls out the email under that the account instead of the admin privileges and stores that email in the **email** variable.
  - If the Username is not in the database, an error message is shown which says, "credentials not found. Please try again."
  - If the Username field is left blank an error message is shown which says, "username field is empty. Please try again."
  - If the username is found the program then generates the 6 character verification code and stores it inside **ranNum** and generates the message to be sent to via email and stores it in **MimeMessage message.**
  - Then the program connects to gmail's SMTP server, logs into the gmail CAAAD gmail account and sends the message using **SmtpClient client** and a series of try catch statements**.**

- ***3.2.2 entCode_Click***
- The entCode_Click function is run when the "enter verification code" button is clicked and checks to make sure that the verification code in **VerificationTxt.Text** is the same verification code that was sent out via email in the previous function. If the codes match the user is taken to the ResetPassForm.cs form.

- If the codes do not match the user will get the following error message: “verification codes do not match” and will not be taken to the ResetPassForm.cs form.

3.3 ResetPassForm.cs



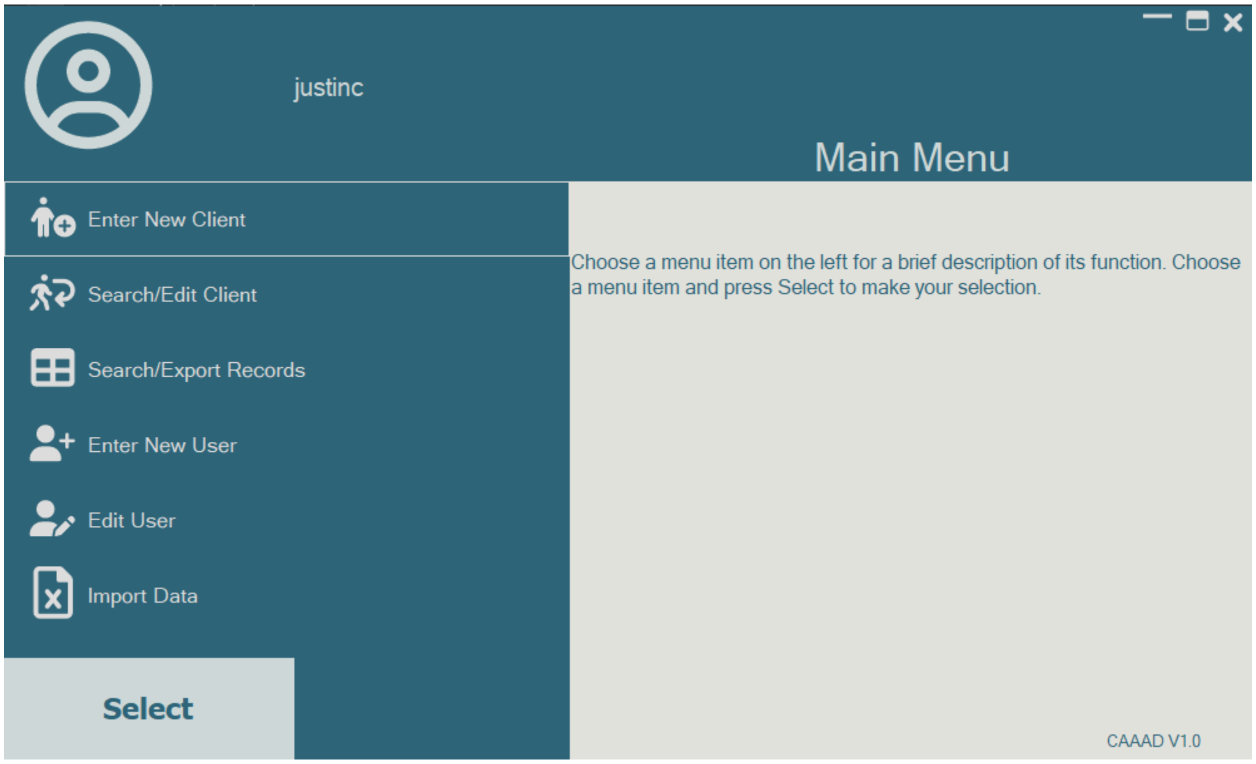
- The function of ResetPassFrom.cs is to finish the process of resetting a user's password by allowing them to actually reset their password now that they have verified themselves. It accomplishes this using the Resetp_Click function.

- The ResetPassForm.cs references the following important variables/structures:
    - **specpattern** -string var that holds the regular expression used to find special Characters such as “!@#$ etc”
    - **speChar** -int var that is used as a counter to hold the number of special characters found in a user's password.Note the initial value is set to 0.
    - **capLetter** -int var that is used as a counter to hold the number of capital letters found in a user's password. Note the initial value is set to 0.
    - **isnumb** -int var that is used as a counter to hold the number of numbers found in a user's password.Note the initial value is set to 0.
    - **counter** -int var that is used as a counter. Note the initial value is set to 0.
    - **NewPassTxt.Text** -holds the new password entered by the user
    - **PVerificationTxt.Text** -holds a copy of the verification code sent to the user’s email account.
    - **HashWord** - string var used to hold the encrypted version of the new password stored in **NewPassTxt.Text**
    - **LoginForm f2** - link to LoginForm.cs
    - **urName** - holds the username entered in ForgorPassForm.cs

- ***3.3.1 Resetp_Click***
- The Resetp_Click function is run when the “reset password button is clicked” and it does the following 5 types of checks to make sure the password is acceptable:
    - First the program checks the length of **NewPassTxt.Text** to see if the new password is less than 8 characters long.
        - If the new password is less than 8 characters long then the will throw the following error: “password is less than 8 characters. please enter a longer password”
    - If the new password is at least characters long then the program moves on to checking the number of special characters, capital letters, and numbers using a for loop with the **counter** variable and checks each character in the new password until it has reached the end of the new password. Inside the for loop exist if and else if statements that run when a special character, capital letter, and number is found and once they are found **capLetter**, **speChar**, and **isnumb** are incremented respectively.
    - Then us **capLetter**, **speChar**, and **isnumb** the program checks to see if there were at least 2 capital letters, at least 1 special character, and at least 2 numbers in the new password contained in **NewPassTxt.Text.**
        - If the new password does not meet the specified requirements you will receive the following error “Password does not contain enough special characters, numbers or capital letters. passwords must contain at least 2 capital letters and at least 1 special character and at least 2 numbers. please try again.”
        - If the new password does meet the requirements the system will then check to see if the password in **NewPassTxt.Text.** and **PVerificationTxt.Text** match.
            - If the passwords do not match you will receive the following error: “Your passwords do not match. please try again.”

    - If the passwords in **NewPassTxt.Text.** and **PVerificationTxt.Text** match then the program will encrypt the new password in **NewPassTxt.Text.** using the previously mentioned BCrypt hashing algorithm and then store the encrypted new password in the **HashWord** Variable.
    - The program will then establish a connection with the database using **MySqlConnection conn, MySqlCommand cmd, MySqlDataAdapter sda,** and **DataTable dt**.and change the password for the account with the same username stored in **urName.** Once this has been done the user will be prompted with the following message: “Password has been reset!” and they will then be taken back to the LoginForm where they can then login with their new password.

3.4 MainMenu.cs



Choose a menu item on the left for a brief description of its function. Choose a menu item and press Select to make your selection.

- The function of MainMenu.cs is to act as the homepage for the program from which you can access every other form from. To accomplish this main menus relies on the following main functions:
  - MainMenu_Load
  - And several _button_Click functions that are named after each button on the main menu
  - select_button_Click

- MainMenu references the following variables/structures
  - **urName** - public string var that contains the username passed to it from the login page
  - **isAdmin** - public int var that contains the admin status passed to it from the login page
  - **currentBtn** - private var that contains the user's selected menu option

- ***3.4.1 MainMenu_Load***
- MainMenu_Load function runs when MainMenu loads and uses the **urName** variable to dynamically show the username of the currently logged in user at the top left corner of

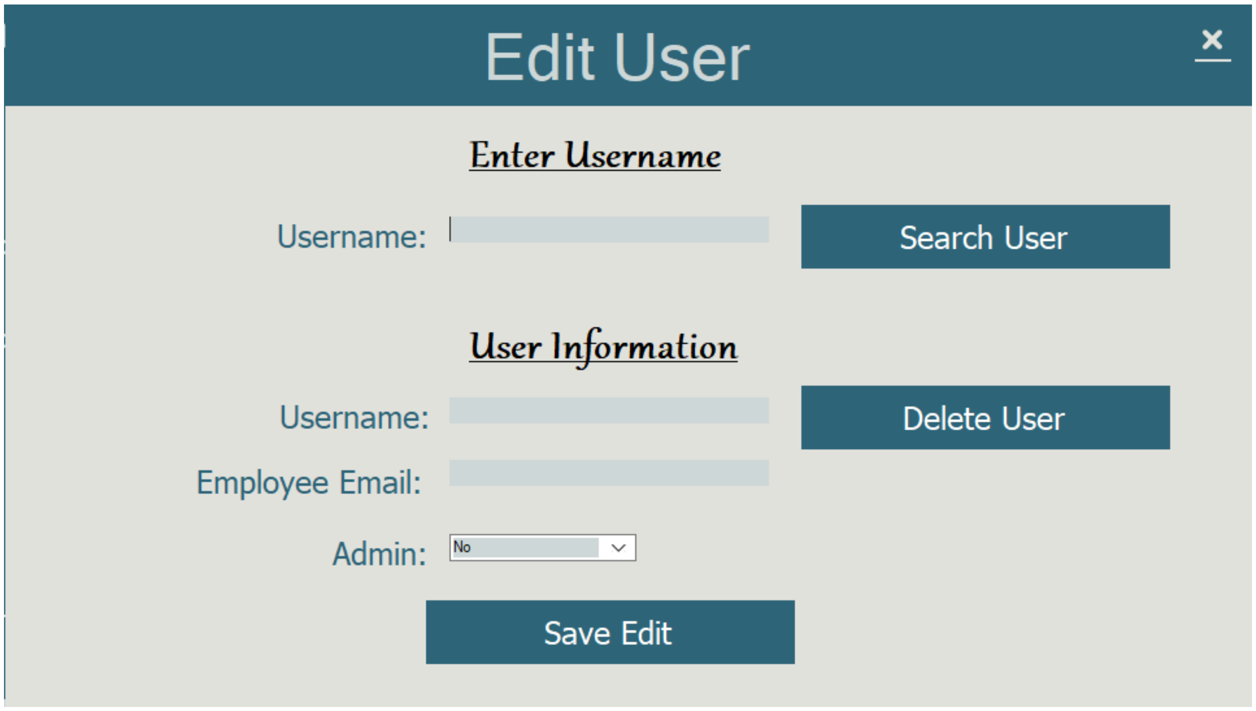
the main menu and the **isAdmin** variable to check whether or not the currently logged in user is an admin account.

- To do this the program checks to see if **isAdmin** is a 1 or 0
    - If **isAdmin** is a 1 then main menu will show the options for enter new user and create new user
    - If **isAdmin** is a 0 then main menu will not show the options for enter new user and create new user

- ***3.4.2 Several_button_click***
- As previously mentioned there are several _button_click functions in the main menu (newClient_button_Click, editClient_button_Click, etc) function all serve the same purpose so we will cover them here. These functions are run when any of the options on the main menu are clicked and when they are clicked they change the color of the clicked on button and the text description for the button so the user can see what that button does.

- ***3.4.3 Select_button_click***
- The Select_button_click function when the big select button at the bottom of the page is clicked. During this process the system checks to what menu button has been clicked on and stored in **currentBtn**. This is done using a series of if-else statements that check the value of **currentBtn** and then take the user to a page stored in **currentBtn.**

3.5 EditUserForm.cs



- The function of EditUserForm.cs is an option exclusive to admins accounts in the system. You can select this function through the main menu page for the CAAAD program; where after the admin signs into their account it is among the options located on the left hand side. This allows the admin to edit users that are currently in the system by looking for the user based on their username, and allowing the admin to change that user's Username, Employee Email, and Admin Status. It accomplishes this by use of three main event functions:

  - SearchUser_Click
  - DeleteUser_Click
  - SaveBtn_Click

- The Edit User page references the following important variables/structures:
  - **urName1 -** used to hold the name of the currently logged in user
  - **AdminPrivlages -** used to hold whether or not user will have admin privileges
  - **string Myconn(2)** - connection string containing information used to connect to the MySql database. Note to connect to a different database you would change the information in this string.
  - **MySqlConnection conn(2)** - used to hold the connection string for connecting to the Mysql database
  - **MySqlCommand cmd(1,2)** - used to hold the query to be run on the database
  - **MySqlDataReader reader** used to parse through specific parts of the data pulled from the database
  - **MySqlDataAdapter sda(1,2)** - used to run the cmd query

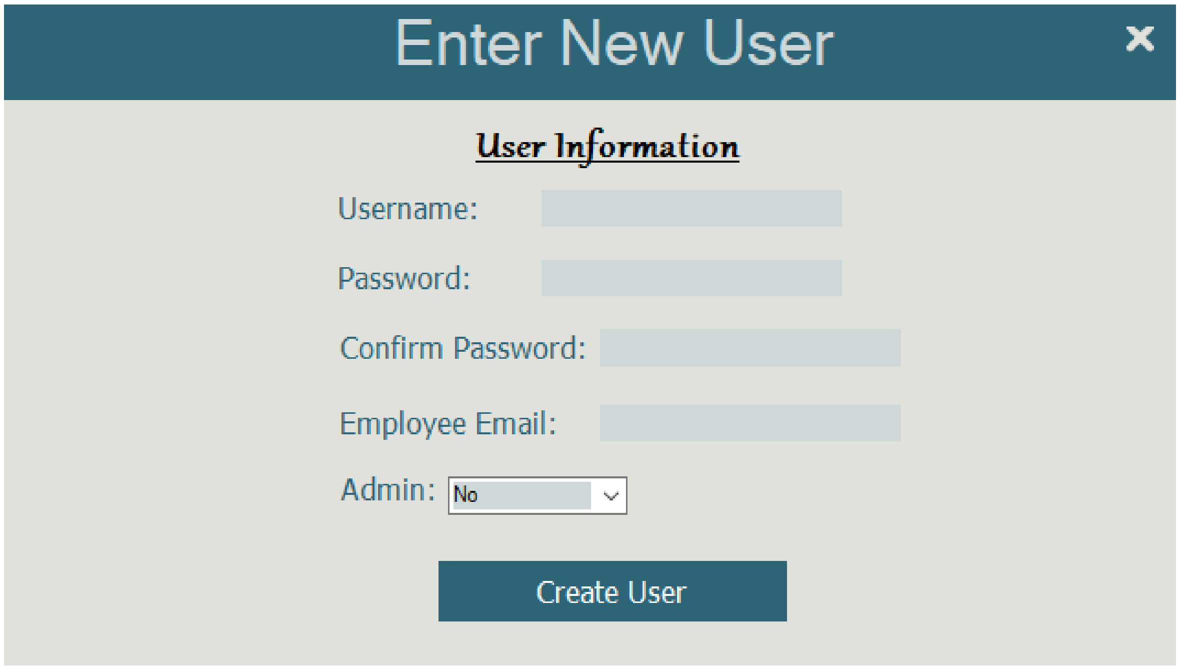
  - **DataTable dt(1,2)** - holds the results of the query to be looked through
  - **UsernameTxt.Text** - holds the username entered by the user
  - **UserNameTxt2.Text** - holds the username entered into the database/ to enter into the database
  - **EmplEmailTxt.Text** - holds the email entered into the database/ to enter into the database
  - **UserExist** - holds if the username is already used or not
  - **emailUsed** - holds if the email is already used or not

- ***3.5.1 SearchUser_Click***
- The SearchUser_Click function is run when the “Search User” button is clicked and its purpose is to check if the Username entered matches a Username in the database which is being connected through using the **MySqlConnection conn, MySqlCommand cmd, MySqlDataAdapter sda, MySqlDataReader reader,** and **DataTable dt**. This check brings forth two possible paths.
  - If the Username is in the database the corresponding data attached to that Username is input from the database to the appropriate fields (Username via **UserNameTxt2.Text**, Employee Email via **EmplEmailTxt.Text**, Admin via **AdminPrivlages**)
  - If the Username is not in the database, an error message is shown which says, “User not found. please try again.”

- ***3.5.2 DeleteUser_Click***
- The DeleteUser_Click function is run when the “Delete User” button is clicked and its purpose is to delete a user from the database, which is an irrevocable action. It connects to the database using the **MySqlConnection conn, MySqlCommand cmd, MySqlDataAdapter sda, MySqlDataReader reader,** and **DataTable dt**. The first thing that is checked when the button is clicked is if **UserNameTxt.Text** is equal to **urName1** stopping the active user from deleting themself. After this check is complete a pop-up box asking for confirmation of deletion, giving the user two options.
  - If the user confirms that they would like to delete the user from the database, the appropriate SQL command is run to do so. Subsequently all the fields are cleared bringing the Page to the same state it was at when first opened.
  - If the user doesn’t confirm that they would like to delete the user from the database the pop-up box closes and takes them back to the Edit User Page, which is in the state it was before they clicked the “Delete User” button.

- ***3.5.3 SaveBtn_Click***
- The SaveBtn_Click function is run when the “Save Edit” button is clicked and its purpose is to commit the made changes of a user’s information to the database through use of he **MySqlConnection conn(2), MySqlCommand cmd(1,2), MySqlDataAdapter sda(1,2), MySqlDataReader reader,** and **DataTable dt(1,2)**.
  - The code first checks if the Username was changed by comparing the Username searched by and the Username under the User Information section, **UserNameTxt.Text** and **UserNameTxt2.Text**, if they are not the same the program searches the database to see if the new Username, **UserNameTxt2.Text**, is already taken. If so, a pop-up notifies the user of the conflict stating,”Username is already in use please pick another username”, if not, the checks continue. After exiting the pop-up box they are taken back to the Edit User Page, which is in the state it was before they clicked the “Save Edit” button.
  - The next check is to see if the email is already taken by looking for entries in the database that match the email given, **EmplEmailTxt.Text**, if it does match a pop-up notifies the user of the conflict stating,”email is already in use. please pick another email”. After exiting the pop-up box they are taken back to the Edit User Page, which is in the state it was before they clicked the “Save Edit” button.
  - After these checks are complete, the information in the database is updated and the user is notified of these changes via a pop-up box which states, “User changes have been saved!”. After exiting the pop-up box they are taken back to the Edit User Page, which is in the state it was before they clicked the “Save Edit” button.

3.6 NewUserForm.cs



- The NewUserForm class is used to create a new user account in the application. It provides a form that allows the user to enter the required information to create a new user account. You can select this function through the main menu page for the CAAAD program; where after the admin signs into their account it is among the options located on the left hand side. This allows the admin to edit users that are currently in the system by looking for the user based on their username, and allowing the admin to change that user's Username, Employee Email, and Admin Status. It accomplishes this by use of the main event function:

    - CreateUser_Click

- Dependencies: The NewUserForm class depends on the following:
    - The MySql.Data.MySqlClient namespace for connecting to the MySQL database.
    - The BCrypt.Net library for hashing the password entered by the user.
    - The System.Windows.Forms namespace for creating the user interface.

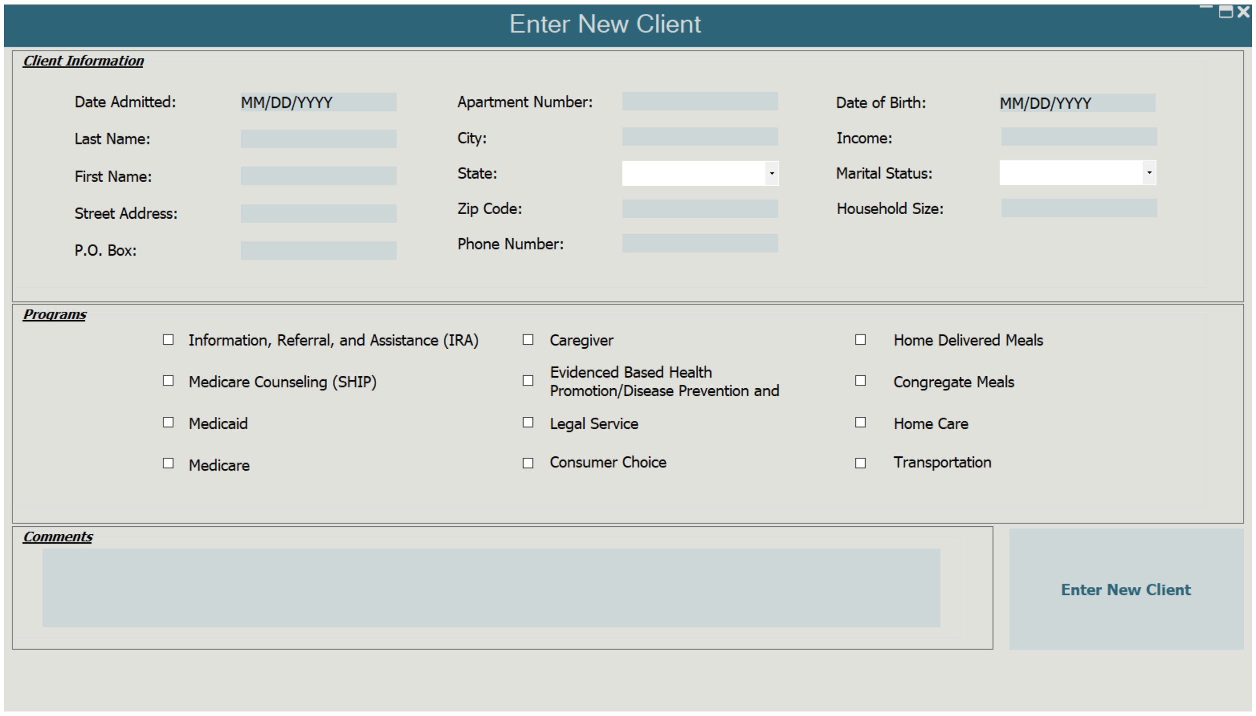
- The New User page references the following important variables/structures:
    - **AdminPrivlages** - used to indicate whether the user will have admin privileges or not. A 1 for this value means yes, while a 0 means no.
    - **UserNameTxt.Text** - holds the entered username
    - **PasswordTxt.Text** - holds the entered password
    - **ConfirmPassTxt.Text** - holds the confirmed password entered
    - **EmpEmailTxt.Text** - holds the entered email

- ○ **HashWord -** holds the password after being hashed using BCrypt
- ○ **emailRegex -** holds the regular expression used to check whether or not the entered email is considered valid.

3.6.1 CreateUser_Click

- The CreateUser_Click function is run when the "Create User" button is clicked after the user enters the required information to create a new user account, including the username, password, email, and whether the new user account will have admin privileges or not. The application checks whether the entered username and email are already in use in the database which is being connected to using the **MySqlConnection conn, MySqlCommand cmd, MySqlDataAdapter sda, MySqlDataReader reader,** and **DataTable dt**. This check brings forth two possible paths
    - If either the username or email is already in use, the application displays an error message to the user and prompts them to enter a different username or email.
    - If the entered username and email are not already in use, the application hashes the password entered by the user and stores the new user account in the database. The application displays a success message to the user and closes the NewUserForm.

3.7 NewClientForm.cs


Enter New Client

Client Information

Date Admitted:	MM/DD/YYYY	Apartment Number:		Date of Birth:	MM/DD/YYYY
Last Name:		City:		Income:	
First Name:		State:		Marital Status:	
Street Address:		Zip Code:		Household Size:	
P.O. Box:		Phone Number:			

Programs

☐ Information, Referral, and Assistance (IRA)	☐ Caregiver	☐ Home Delivered Meals
☐ Medicare Counseling (SHIP)	☐ Evidenced Based Health Promotion/Disease Prevention and	☐ Congregate Meals
☐ Medicaid	☐ Legal Service	☐ Home Care
☐ Medicare	☐ Consumer Choice	☐ Transportation

Comments

- The New Client Form is designed to capture and store information about a new client. The form includes fields for basic client information such as name, address, phone number, date of birth, income, and marital status. Additionally, there are fields for specific services the client receives, such as home care and transportation services. All users can select this function through the main menu page for the CAAAD program. The form achieves this through the use of its main event function:

- enterClient_button_Click

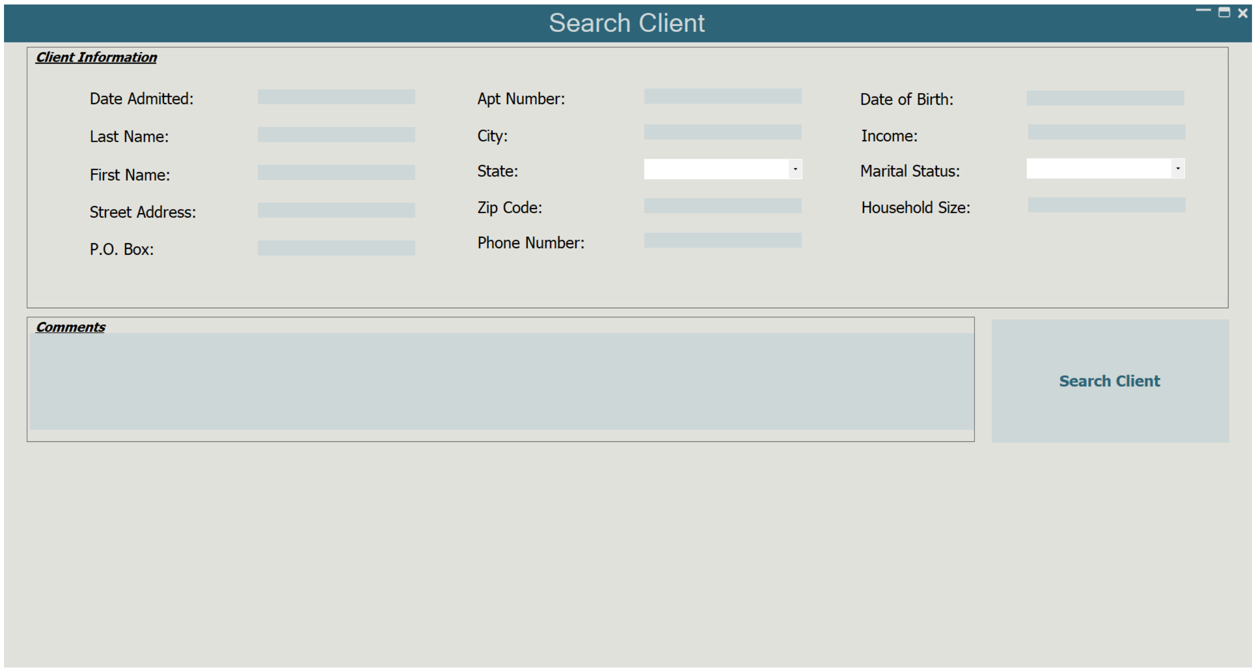
- The New Client page references the following important variables/structures:
    - **enterDobTxt.Text** - holds the date of birth for the client, entered by the user into the form
    - **enterDateTxt.Text** - holds the date that this client is entered into the form
    - **enterLastTxt.Text** - holds the last name of the client, entered by the user into the form
    - **enterFirstTxt.Text** - holds the first name of the client, entered by the user into the form
    - **enterAddressTxt.Text** - holds the address of the client, entered by the user into the form
    - **enterPOTxt.Text** - holds the P.O. box of the client, entered by the user into the form
    - **enterApartmentTxt.Text** - holds the apartment information of the client,entered by the user into the form
    - **enterCityTxt.Text** - holds the city in which the client resides, entered by the user into the form
    - **enterStateCbox.Text** - holds the state in which the client resides, chosen via dropdown by the user on the form (options are “SC” and “NC”)
    - **enterZipTxt.Text** - hold the ZIP code in which the client resides, entered by the user into the form
    - **enterPhoneTxt.Text** - holds the phone number of the client, entered by the user into the form
    - **enterIncomeTxt.Text** - holds the income of the client, entered by the user into the form
    - **enterMaritalStatusCbox.Text** - holds the marital status of the client, chosen via dropdown by the user on the form (options are “single,” “married,” and “widowed”)
    - **enterHouseholdTxt.Text** - holds the household size of the client, entered by the user into the form
    - **recMedicareCbox.Checked** - indicates whether or not a client receives Medicare via a checkbox chosen on the form by a user
    - **recMedicaidCbox.Checked** - indicates whether or not a client receives Medicaid via a checkbox chosen on the form by a user
    - **recIRACbox.Checked**
    - **recShipCbox.Checked** - indicates whether or not a client receives Medicare Counseling (SHIP) via a checkbox chosen on the form by a user
    - **recCaregiverCbox.Checked** - indicates whether or not a client receives caregiver services via a checkbox chosen on the form by a user

  - **recDeliveredMealsCbox.Checked** - indicates whether or not a client receives delivered meals via a checkbox chosen on the form by a user
  - **recCongregateMealsCbox.Checked** - indicates whether or not a client receives congregate meals via a checkbox chosen on the form by a user
  - **recHomeCareCbox.Checked** - indicates whether or not a client receives home care via a checkbox chosen on the form by a user
  - **recEbhpCbox.Checked** - indicates whether or not a client receives evidence based health promotion/disease prevention via a checkbox chosen on the form by a user
  - **recLegalServicesCbox.Checked** - indicates whether or not a client receives legal services via a checkbox chosen on the form by a user
  - **recConsumerChoiceCbox.Checked** - indicates whether or not a client receives consumer choice services via a checkbox chosen on the form by a user
  - **recTransportationCbox.Checked** - indicates whether or not a client receives transportation services via a checkbox chosen on the form by a user

3.7.1 enterClient_button_Click

- The enterClient_button_Click function is run when the “Enter New Client” button is clicked after the user enters the required information to enter a new client, including first name, last name, date of birth, and any other fields that the user wishes to enter. The application then writes the client to the database which is being connected through using the **MySqlConnection conn, MySqlCommand cmd, MySqlDataAdapter sda, MySqlDataReader reader,** and **DataTable dt.**

3.8 EditClientForm.cs



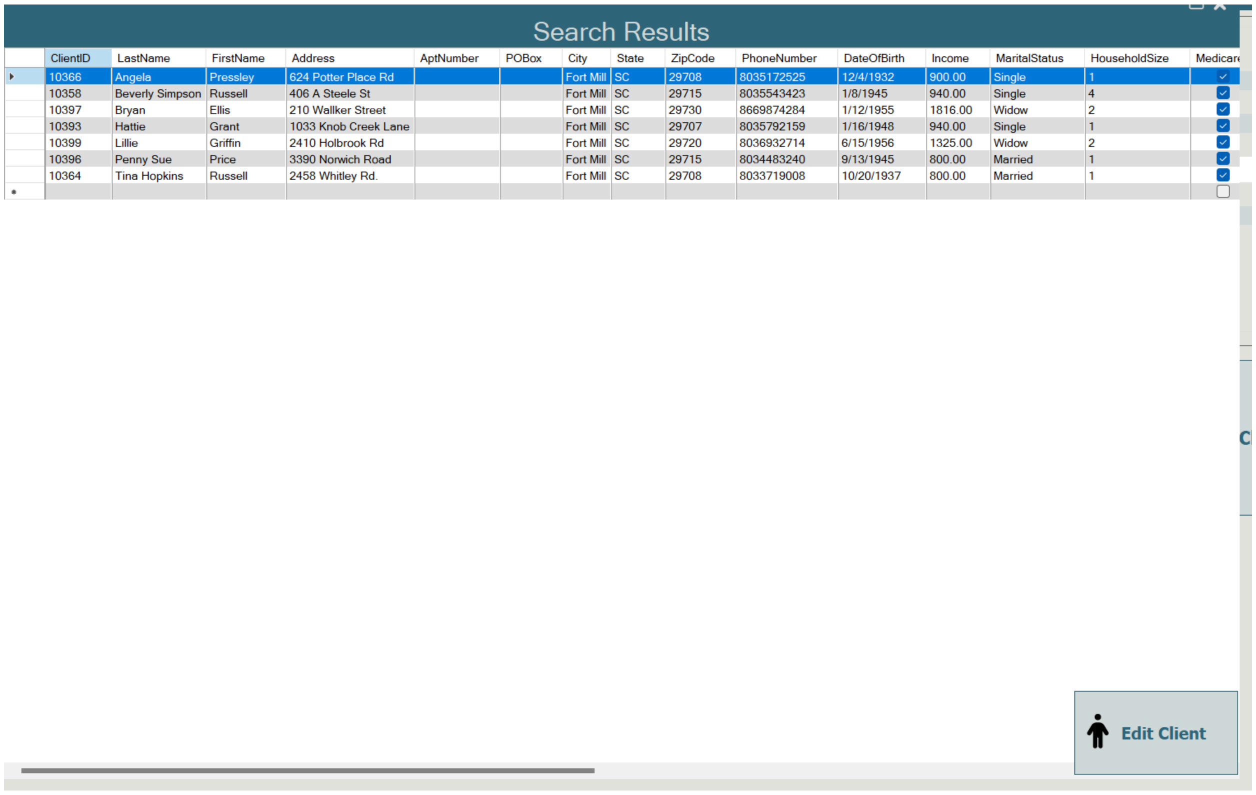
- The main purpose of the EditClientForm.cs is to allow the user to search for an existing client in the database and then make edits to the different client fields and comments. This page also allows the user to delete a client from the DB. To accomplish this EditClientForm.cs uses 2 main functions:
  - enterClient_button_Click
  - deleteClient_btn_Click

- The Edit Client page references the following important variables/structures:
  - **enterDobTxt.Text** - holds the date of birth for the client, entered by the user into the form
  - **enterDateTxt.Text** - holds the date that this client is entered into the form
  - **enterLastTxt.Text** - holds the last name of the client, entered by the user into the form
  - **enterFirstTxt.Text** - holds the first name of the client, entered by the user into the form
  - **enterAddressTxt.Text** - holds the address of the client, entered by the user into the form
  - **enterPOTxt.Text** - holds the P.O. box of the client, entered by the user into the form
  - **enterApartmentTxt.Text** - holds the apartment information of the client,entered by the user into the form
  - **enterCityTxt.Text** - holds the city in which the client resides, entered by the user into the form

- **enterStateCbox.Text** - holds the state in which the client resides, chosen via dropdown by the user on the form (options are “SC” and “NC”)
- **enterZipTxt.Text** - hold the ZIP code in which the client resides, entered by the user into the form
- **enterPhoneTxt.Text** - holds the phone number of the client, entered by the user into the form
- **enterIncomeTxt.Text** - holds the income of the client, entered by the user into the form
- **enterMaritalStatusCbox.Text** - holds the marital status of the client, chosen via dropdown by the user on the form (options are “single,” “married,” and “widowed”)
- **enterHouseholdTxt.Text** - holds the household size of the client, entered by the user into the form

3.8.1 enterClient_button_Click

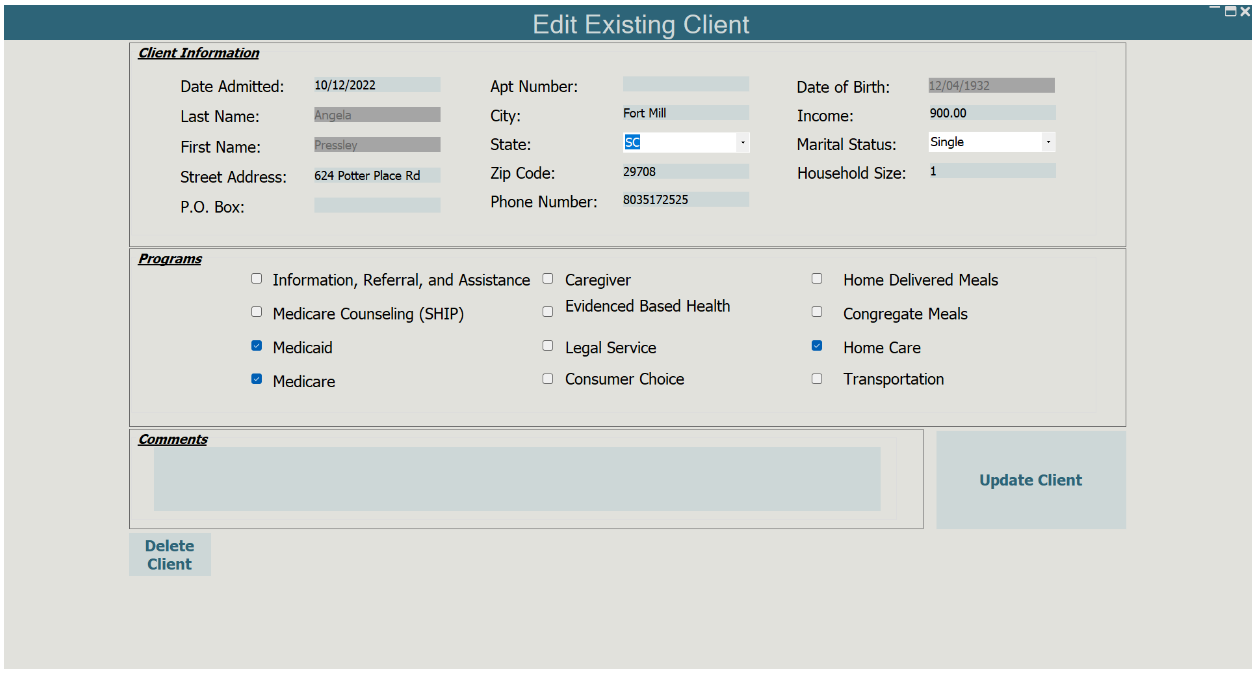
- The enterClient_button_Click function is run when the “Search Client” button is clicked after the user enters the required information to enter a new client, including first name, last name, date of birth, and any other fields that the user wishes to enter. The application then searches for the client entry in the database which is being connected through using the **MySqlConnection conn, MySqlCommand cmd, MySqlDataAdapter sda, MySqlDataReader reader,** and **DataTable dt.** But this program then brings up the following menu containing the results of the query:



Search Results

	ClientID	LastName	FirstName	Address	AptNumber	POBox	City	State	ZipCode	PhoneNumber	DateOfBirth	Income	MaritalStatus	HouseholdSize	Medicare
▸	10366	Angela	Pressley	624 Potter Place Rd			Fort Mill	SC	29708	8035172525	12/4/1932	900.00	Single	1	✓
	10358	Beverly Simpson	Russell	406 A Steele St			Fort Mill	SC	29715	8035543423	1/8/1945	940.00	Single	4	✓
	10397	Bryan	Ellis	210 Wallker Street			Fort Mill	SC	29730	8669874284	1/12/1955	1816.00	Widow	2	✓
	10393	Hattie	Grant	1033 Knob Creek Lane			Fort Mill	SC	29707	8035792159	1/16/1948	940.00	Single	1	✓
	10399	Lillie	Griffin	2410 Holbrook Rd			Fort Mill	SC	29720	8036932714	6/15/1956	1325.00	Widow	2	✓
	10396	Penny Sue	Price	3390 Norwich Road			Fort Mill	SC	29715	8034483240	9/13/1945	800.00	Married	1	✓
	10364	Tina Hopkins	Russell	2458 Whitley Rd.			Fort Mill	SC	29708	8033719008	10/20/1937	800.00	Married	1	✓
*															☐

From this point the user clicks on the client they wish to enter and then hits the Edit Client button and they will be taken to the following page.



Edit Existing Client

Client Information

Date Admitted:	10/12/2022	Apt Number:		Date of Birth:	12/04/1932
Last Name:	Angela	City:	Fort Mill	Income:	900.00
First Name:	Pressley	State:	SC	Marital Status:	Single
Street Address:	624 Potter Place Rd	Zip Code:	29708	Household Size:	1
P.O. Box:		Phone Number:	8035172525		

Programs

- ☐ Information, Referral, and Assistance
- ☐ Medicare Counseling (SHIP)
- ☑ Medicaid
- ☑ Medicare
- ☐ Caregiver
- ☐ Evidenced Based Health
- ☐ Legal Service
- ☐ Consumer Choice
- ☐ Home Delivered Meals
- ☐ Congregate Meals
- ☑ Home Care
- ☐ Transportation

Comments

The user will now be able to edit the different parameters of the selected client. Once they have finished the edits they want to make the user will click the update client button at which point the program will establish a connection to the database using **String**

MyConnection, **MySqlConnection MyConn2, MySqlCommand comm.** And submit the changes to the data base

- ***3.8.2 deleteClient_Btn_Click***
- This function is run when the user wishes to delete the selected client and clicks on the Delete Client button at which point the program will connect establish a connection to the database using **String MyConnection**, **MySqlConnection MyConn2, MySqlCommand comm** and query the database for and remove the select client.
    - If the deletion was successful you will see the following message: “Record successfully deleted.”
    - If the deletion was not successful you will see the following error message “Could not delete record.\nWas it already deleted?”

3.9 exportCsvBtn_Click

- This is the “Export” function or the “Create CSV” button in the **DataViewForm.cs**
- Asks the user to select a save location for the exported CSV file. If the save location is valid, calls the **ExportFile** function which will save a .CSV file with the information displayed in the datatable UI. The file name will be based on the current system time in the "MM-dd-yyyy_hh-mmtt" format.
- **ExportFile**: This function is a big loop that will write the searched data from the **DataViewForm.cs** into the .CSV file. First it writes the column headers, there are 26 columns in the database at the time this manual was created. After all headers are placed, a new loop begins which writes values to the table N times where N is equal to the amount of rows in the datatable. The last column in each row forgoes a comma. For each cell of data, the program attempts to determine if the data is a date or BOOL. If it is detected as one of those data types when it is formatted accordingly, if not then the data is added in unformatted. This process will repeat until the end of the datatable is reached. Once completed successfully the message “Successfully exported” will be displayed. If an error occurs, “Export error” will be displayed instead.

3.9 ImportForm.cs

The import form handles all functionality related to file selection, opening, and querying data files into the database. It has two primary functions, and several smaller functions that serve as wrappers into the primary functions.

3.9.1 filename_txt_KeyDown

This function is called whenever a key is pressed in the textbox **filename_txt**. Its only purpose is to detect whenever the user presses the enter key, when it will call *ImportFile* with the contents of the textbox.

3.9.2 import_btn_Click

This function is called whenever the user clicks on the button **import_btn**. It serves as the precondition for *ImportFile* and will call it with the contents of textbox **filename_txt**.

3.9.3 browse_btn_Click

This function is called whenever the user clicks on the button **browse_btn**. It launches the windows file select dialog to get the path of the source file to be imported. Using the methods available in the namespace **System.Windows.Forms**, it shows a dialog box that defaults to the special folder “Documents”. The dialog box uses a default filter of “*.csv|CSV files” but also allows for selection of the filter “*.*|All files”

The dialog response is verified as being equivalent to **DialogResult.OK**. Upon successful dialog return, the function will assign the filename to textbox **filename_txt** and then call *ImportFile* using the contents of that textbox.

3.9.4 ImportFile

This function is called by *filename_txt_KeyDown, import_btn_Click,* and *browse_btn_Click*. It first checks that a filename has been provided. Shortly after, the function copies the contents of the file to a string and verifies that there is content in the file. After these preliminary checks have been passed, the function loops through each row and performs queries using the information within that row of the file. Each value is checked using similar checks to those made in *NewClientForm.cs* which is documented in section 3.6. The records are also checked to contain the primary key requirements similar to section 3.6. Any errors that are found are stored in the string array **badQueries** and the number of attempted and successful queries are stored in the integers **attemptedQueries succesfulQueries**, respectively.

As the row strings are parsed, any occurrence of double quotes are removed from the values in order to correctly check them. After each of these checks, the query is attempted and the above variables are updated. Once the file loop has been completed, the function updates the listbox **output_lb** with any errors that were detected. It finally updates the **processedCount_lbl** with the total number of successful imports and notifies the user of completion using a message box as documented in the **System.Windows.Forms** namespace.

The function may throw the following exceptions, which are handled by the function itself by showing another message box with the corresponding error message:

- ArgumentException
    - "No file path specified"
        - Cause: The filename argument has no length
        - Resolution: Exit the function
    - "No content detected in file: '<FILENAME>'"
        - Cause: The file has no non-whitespace content
        - Resolution: Exit the function
    - "Record must have 26 fields"
        - Cause: The file does not have enough commas to signify 26 fields on the current line
        - Resolution: Do not attempt a query for the row, document the error in **badQueries** and increment **attemptedQueries**
    - "Last name required"
        - Cause: The file does not contain a value in the LastName position on the current line
        - Resolution: Do not attempt a query for the row, document the error in **badQueries** and increment **attemptedQueries**
    - "First name required"
        - Cause: The file does not contain a value in the FirstName position on the current line

        - Resolution: Do not attempt a query for the row, document the error in **badQueries** and increment **attemptedQueries**
    - "DoB required"
        - Cause: The file does not contain a value in the DateOfBirth position on the current line
        - Resolution: Do not attempt a query for the row, document the error in **badQueries** and increment **attemptedQueries**
- DataException
    - "Empty row detected"
        - Cause: The file does not contain any values for any fields on the current line
        - Resolution: Do not attempt a query for the row