**Church Database Manual**

Although the proposed project idea was for the database to be managed by the administration only, we eventually decided to allow members to sign up and manage their own accounts as well, this is so each member can know how much they owe or paid and know their preferred-type of payment and number of family members enlisted in the church.

**The Guide:**

To get access to our home page, please use the follow URL to get to our home page. <http://192.168.64.2/ChurchDatabase/DatabaseM/Churchlogin/index.php>

Since we use Xampp program to host our system, this will direct you to our home page.

The home page – here we take you to a log-in or sign-up window, where administration or members may login and new members may be added by administrators or members may sign up themselves.

**The Sign up menu:** here the user can create their own unique login. The required fields are: Firstname, Lastname, E-mail, Username, and password. Once signed up the user may access the Login menu by logging in with their username and password.

**The Login:**

Once logged in, the user will be granted “New Application”, “Update”, and “Sort by” features.

* **New Application:** Here user is granted with the ability to add a new Person, Member, or Employee application.
  + Require fields are: Person ID (which is set to auto increment), First Name, Last Name, Address, SSN, E-Mail, check whether is a Member or Employee. Then Sign up.
  + Required fields are: Member ID, Renewal Date, Payment Type, Family Count, Date Joined, and Application in process. Then Sign up.
  + Required Fields are: Person ID (reference to Person), Date of Application, Title, and Resume and Application on file.
* **Update:** User may make changes or update information for Person, Member, or Employee.
  + **Member Update -** Require fields are: Person ID (which is set to auto increment), First Name, Last Name, Address, SSN, E-Mail, check whether is a Member or Employee. Then update.
  + **Employee Update -** Required fields are: Member ID, Renewal Date, Payment Type, Family Count, Date Joined, and Application in process. Then Update.
  + **Person Update -** Required Fields are: Person ID (reference to Person), Date of Application, Title, and Resume and Application on file.
* **Sort by:** Use may sort or search the database by First Name, Last Name, Title, New Address, Old Address, and ID for each field.
  + ***Results will display according to desired search or sort choice.***

**Church Database Report**

**Does the project work as proposed?**

The finished program works as intended but does not include all work as purposed due to time constraints and lack of prior knowledge in specific areas- mostly language, such as HTML, CSS, and PHP. Another notable reason is this is the very first complete database we’ve designed and put together, perhaps with more hands-on experience going forward, the next database will be a better product.

**Do the user interfaces contain all of the features proposed?**

The user interface contains most of the features proposed. Our original idea to allow SQL commands did not materialize, again due to time constraints and lack of knowledge.

**Do the management interfaces contain all of the features proposed?**

Yes, the management interface contains all of the features proposed. This one is important as usability is a big priority of ours throughout the whole project.

**Is the application easy to understand?**

The application is simple and easy to understand, again because usability was priority.

**Is navigation through various features easy to understand?**

Navigating is easy to understand because our interface is simple, and purpose is unpretentious.

**Is the code well documented?**

Like any other program or application, the intent is always to have a code that is well documented for future referencing but due to a lack of time, we were not able to document all code as we would have liked. Also we learned that file naming is important even though file location access is usually sufficient.

**Are the SQL commands well-formed and easily understood?**

Our SQL commands are well formed and easy to understand, we did our best to simplify and limit complex situations as time is another issue here.

**Does the code and SQL appear easy to maintain?**

Our code and SQL is easy to maintain for the reasons that our focus was to keep things simple and still accomplished what we set out to do as far as the usability in features we originally proposed early on.

**Church SQL**

**THE SELECT and INSERT Statements**

**Person:**

$sql = "SELECT \* FROM Person WHERE PersonID='$PersonID'";

* This sql will select from person where the Person ID in person table is equal to the person ID in our add person form. To prevent user from entering a Person ID already in use, an “if-else” statement was used here.

$sql = "INSERT INTO Person (PersonID,FirstName, LastName , Address ,Zip, Phone, Email , Personnel\_Type , DateModified) VALUES ('$PersonID','$first','$last','$Address', '$Zip', '$Phone', '$Email','$Type','$Date');"

* After the if-else statement has pass all the error checks then it will insert new information into the Person Table, thus creating a new Person with a unique Person ID.

**Employee: *Note, the following SQL query executes and link data, first it will execute one, then it will link the next SQL query.***

$sql = "INSERT INTO Employee (PersonID,DateHired, Title , Resume) VALUES ('$PersonID','$DateHired','$Title','$Resume');";

* This SQL query will insert data into the Employee table once user presses a button on the Employee form. Then direct it to Assignments Table.

$sql = "INSERT INTO Assignment (EmployeeID,AssignmentName, AssignmentDate) VALUES ('$EmployeeID','$AssignmentName','$AssignmentDate');";

* This SQL query allows user to insert into the Assignment table after the form input is entered and submitted. Then it will direct it to Project input form.

$sql = "SELECT \* FROM Project WHERE ProjectID='$ProjectID'";

$sql = "INSERT INTO Project (ProjectID, AssignmentID, ProjectName, StartDate,EndDate) VALUES ('$ProjectID','$AssignmentID','$ProjectName','$StartDate','$EndDate');";

* This will allow user to input Project as long as there does not exist a repeating Project ID. If a Project ID already exists it will direct them to error page, but if successful, user will be directed to Salary Employee and Hourly Employee options.

$sql2 = "INSERT INTO HourlyEmployee (HE\_ID,Hourly\_Pay) VALUES ('$HE','$Hourly\_Pay');";

* Insert data into HourlyEmployee once complete and submitted, it will direct user to the menu page.

$sql = "INSERT INTO SalaryEmployee (SE\_ID,Salary\_Pay) VALUES ('$SE','$Salary\_Pay');";

* Insert data into SalaryEmployee once complete and submitted, it will direct to menu page.

**Member**:

$sql = "SELECT \* FROM Member WHERE MemberID='$MemberID'";

$sql = "INSERT INTO Member ( MemberID,RenewalDate, Payment\_Type, FamilyCount, DateJoin, Application) VALUES('$MemberID','$RenewalDate','$Payment\_Type', '$FamilyCount', '$DateJoin', '$Application');";

* This will check for member ID information, also checks for repeating ID to avoid duplicates. If the Member ID unique, data will be inserted into the Member table and user will be directed to Recurring charge page.

$sql = "SELECT \* FROM RecurringCharge WHERE   AccountNumberID ='$ AccountNumberID '";

$sql = "INSERT INTO RecurringCharge (AccountNumberID, Card\_Number , Exp\_Year, BankAccount ,RouteNum, ModifiedDate , Exp\_Month ) VALUES('$AccountID','$Card\_Num','$Exp\_Year','$BankAcc','$RouteNum', '$ModifiedDate', '$Exp\_Mon');";

* Number ID will be checked for duplicate. If no duplicated it will insert data into Recurring Charge table and direct it to Payment pages.

$sql = "SELECT \* FROM Payment WHERE PaymentID='$PaymentID' OR AccountNumberID !='$AccountnumID';";

$sql = "INSERT INTO Payment (PaymentID, MemberID , AccountNumberID , Pay , Amount ) VALUES ('$PaymentID','$ID','$AccountnumID','$Pay','$Amount');";

* Number ID will be checked for duplicate. If no repeat it will insert data into Payment table and direct it to the menu.

**Users**:

$sql = "SELECT \* FROM users WHERE user\_uid='$uid' OR user\_email='$uid'";

* Here, it will select users where the UID equal to some variable . This SQL will use to test the password that user enter in the home page.

$sql = "SELECT \* FROM users WHERE user\_uid='$uid'";

$sql = "INSERT INTO users (user\_first, user\_last, user\_email, user\_uid, user\_pwd)VALUES ('$first', '$last', '$email', '$uid', '$hashedPwd');";

* From the Users table and an if-else statement is to check if the Sign up page has all information entered or if it has any empty variable, and if the information entered is valid. Also, for security, allow the password to be hashing so if someone else views the database they won’t be able to identify unique user password for login. Once everything is properly entered according to code and is valid, values will be inserted into tables.

**The UPDATE Statment:**

**Employee update:**

$sql = "UPDATE Employee

        SET PersonID= '$PersonID',

            DateHired= '$DateHired',

            Title='$Title',

            Resume= '$Resume'

        WHERE PersonID = '$PersonID';";

* User is allowed to make changes to Employee information.

**Member update:**

 $sql = "UPDATE Member

        SET RenewalDate= '$RenewalDate',

            Payment\_Type= '$Payment\_Type',

            FamilyCount='$FamilyCount',

            DateJoin= '$DateJoin',

            Application= '$Application'

        WHERE MemberID = '$MemberID';";

* User is allowed to make changes to Member information.

**Person update:**

$sql = "UPDATE Person

        SET PersonID= '$PersonID',

            FirstName= '$first',

            LastName='$last',

            Address= '$Address',

            Zip= '$Zip'

            , Phone= '$Phone'

            , Email= '$Email'

            , Personnel\_Type= '$Type'

            , DateModified = '$Date'

        WHERE PersonID = '$PersonID';";

* User is allowed to make changes to Person information.

**Sort Statement:**

**Old Address Sort:**

$sql = "SELECT P.\*, PA.\* FROM Person P, PriorAddress PA

WHERE P.PersonID = PA.PersonID ORDER BY Address\_Old;";

* This will sort by the old address which is using the prior Address table to sort

**New Address Sort:**

$sql = "SELECT P.\*, PA.\* FROM Person P, PriorAddress PA

WHERE P.PersonID = PA.PersonID ORDER BY P.Address;";

* This will sort by new address that user input in person table by set the Person ID in Person equal to the Person ID in PriorAddress to match up the person information

**Sort by First Name:**

$sql = "SELECT \* FROM Person WHERE PersonID = PersonID ORDER BY FirstName ASC;";

* Sort Person first name in A-Z of the first name

**Sort by Last Name:**

$sql = "SELECT \* FROM Person WHERE PersonID = PersonID ORDER BY LastName DESC;";

* Sort by Last Name in descend order of Z-A

**Sort by Title:**

$sql = "SELECT E.\*, P.\* FROM Employee E, Person P WHERE P.PersonID = E.PersonID ORDER BY E.Title ASC;";

* Sort by Employee Title in Ascend order of A-Z and where Person PersonID equal to the employee Person ID

**Sort by Employee ID:**

$sql = "SELECT E.\*, P.\* FROM Employee E, Person P WHERE P.PersonID = E.PersonID ORDER BY E.EmployeeID;";

* This will sort it by Employee ID where the Person Person ID equal to the Employee Person ID

**Sort Member ID:**

$sql = "SELECT M.\*,P.\* FROM Member M, Person P WHERE M.PersonID = P.PersonID ORDER BY M.MemberID;";

* Sort by Member Id where the member person id equal the Person Person ID, so it doesn’t repeat the data once it select information in the table

**Sort by Person ID:**

$sql = "SELECT \* FROM Person WHERE PersonID = PersonID ORDER BY PersonID;";

* Sort by Person ID where Person ID equal the same ID number in Person table