Software Design Specification

for

Campus Ministry Software

Version 1.0 approved

Prepared by Zahara Kazmi

Loyola University Maryland

February 27, 2017

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

The Software Design document is produced to describe the design of the software system we are developing. This document will provide details for how the software will be built and aid in the development. The document will be a graphical and narrative description of the software design including architectural style, use case model, sequence diagram and class diagrams

1.1 Goals and objectives

This project will include a Web interface with will be connected to a database as well as a Machine Learning Database for Event Planning. The software will include use for Campus Ministry Staff, Interns and Guest users. The database will store information for Staff, interns, hours worked, projects working on and program planning. The interns and Staff will interact with database via Web interface. The Machine Learning Algorithm will be used to query for program planning for specified amount of attendance.

1.2 Statement of scope

This Software Design Document is to explain in detail an overall architecture of the software as well detailed description of the software processes. This document will work as a proof of concept for the building of software and base level functionality understanding for the client. This document will include diagrams and architecture which include a high-level design as well as modules and components design. This document will be used in conjunction with the SRS document and expanding on the classes created and classes to be developed.

1.3 Software context

This Software will enable Campus Ministry to store and transform data into information to support work flow. This software will replaces previous work flow of using a file system. This software will allow staff and interns to have concurrent access to the database as well to guest in order to enter information. The database structure will improve integrity of the data as well providing a single location access point for all staff and guest members from any location. Data redundancy and inconsistency would be removed and data will come directly from source instead being manually entered by interns. It will increase the security and reliability of the office in collected information. Improvement in the various qualities will improve data quality as well thus resulting in improved and faster work flow.

Another aspect of this software is the Machine Learning Algorithm which will be providing suggestions for planning future events based on information from previous event. This will allow interns to have targeted audience to improve quality of the events.

1.4 Major constraints

Machine Learning Algorithm implementation of this software does not have data to be trained on and tested on. The office has already combining ed their collected data into one file and lost data for over the years. Thus, the algorithm needs to be confirmed via pseudo-data or use unsupervised learning algorithm. The Algorithm will be confirmed based on accuracy for the specified algorithm and its functionality to work with the different features.

1.5 References

In order to develop this document several different online resources are being implemented for illustrations and graphical representation. Draw.io is used to create and implement the graphical representation of software. A Table of Figures is included to reference all the various figures within this document.

2. Data design

This section provides a detailed description of all the data structure which will be implemented within this software. The software includes a Database and the users will be entering information via web interface thus we have several different structures where data is being passed from one point to another.

2.1 Internal software data structure

The internal data structures include structures which are passed between the components. Since the users will be entering data into the web forms, that data would needs to be read in and processed to be formatted for entry into MySQL database. The Web Interface component of the software is responsible for reading in the data entered by user from the web forms and it will send the data to connected Server program for processing. The program will process what the information by correctly formatting into MySQL data entry and inserting into MySQL.

2.2 Global data structure

The Global Data Structures for this software include query result selection entered by user. The user will enter the information into web form for query and this is then sent to the Program. From there depending on type of query either results are returned after running on MySQL or user send data to run Machine Learning Algorithm with those constraints. The data from this web form would be available to all the class within the program, sent to MySQL and Machine Learn Algorithm as well as HTML form.

2.3 Temporary data structure

For this software, the temporary data structure includes the data file which would be created when a query is ran on the database and the results for it will be returned by Java Program via file. Although it is possible to display the results without creating a file, since one of the requirements is that user must be able to download query results in CSV format, creating the file completes this requirement. If the user chooses to not download the file, it will be deleted regardless. Another aspect of the database which would use a temporary file would be when user runs the Machine Learning Algorithm. Whenever the algorithm is requested, first the program will run MySQL query and pull data from database which is saved in file in specific format. The data is the preprocessed as needed and divided into training, verifying and testing. During preprocessing, a new file is created and after algorithm is completed then the results are returned to Java Program via file which are then displayed to user. All the file created in processing Machine Learning Algorithm are deleted once results are displayed to user.

2.4 Database Description

The database to be implemented is described here in details using an Entity relationship diagram, Relational Schema, functional Dependencies and a Data dictionary. These figures will describe exactly what type of data will be accepted into database and how it will process and relate to other models. The reference figure below shows the different symbols and their meaning within the figures. For each figure which references the figure, it will be clearly stated that it references the EER Key Figure.

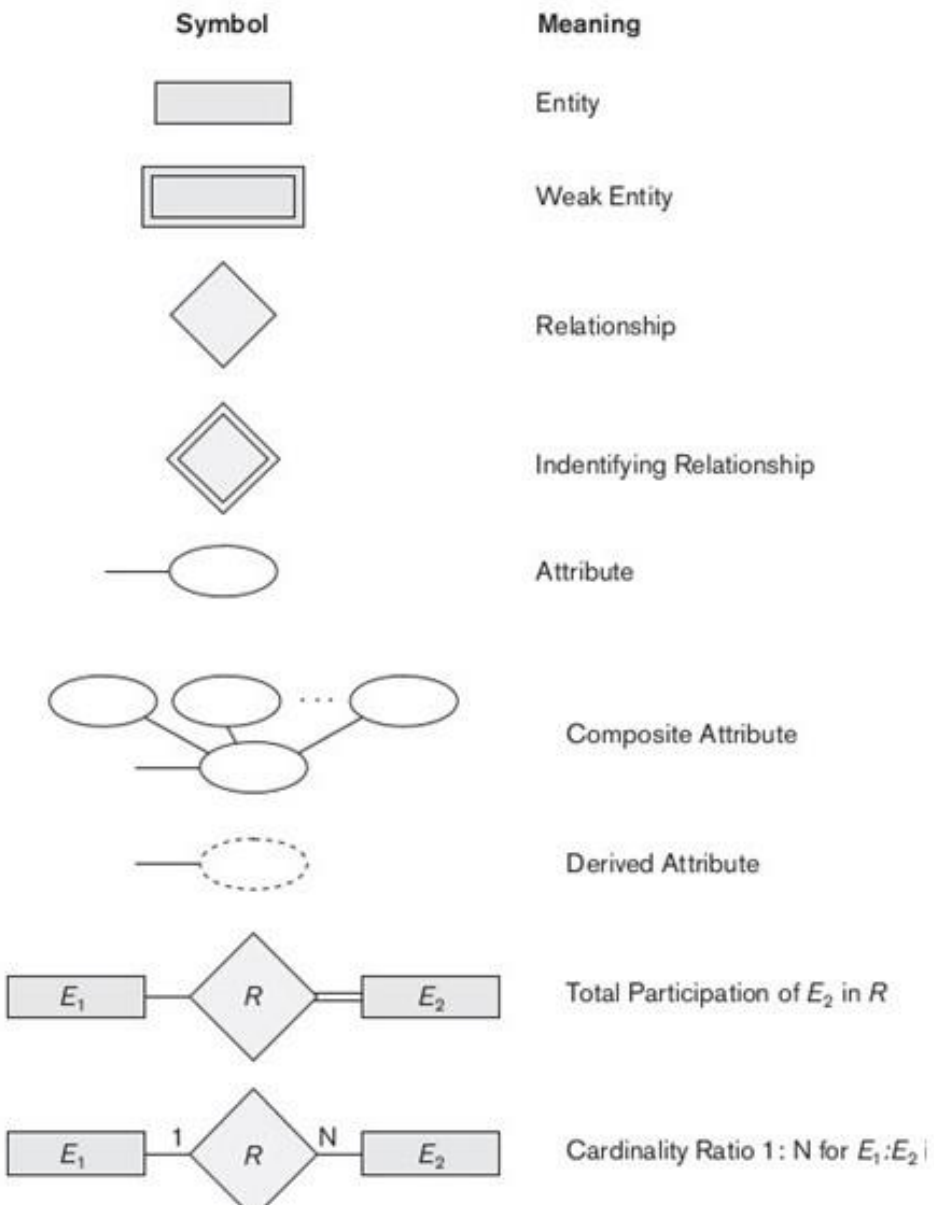

Figure 1 Entity Relation Reference Diagram

2.4.1 Entity Relationship Diagram

The Campus Ministry Database has been designed into an Entity relationship diagram and displayed below. The EERD shows all the different entities which are involved within this database and how they relate to one another. Entity diagram references EER Key figure for symbols.

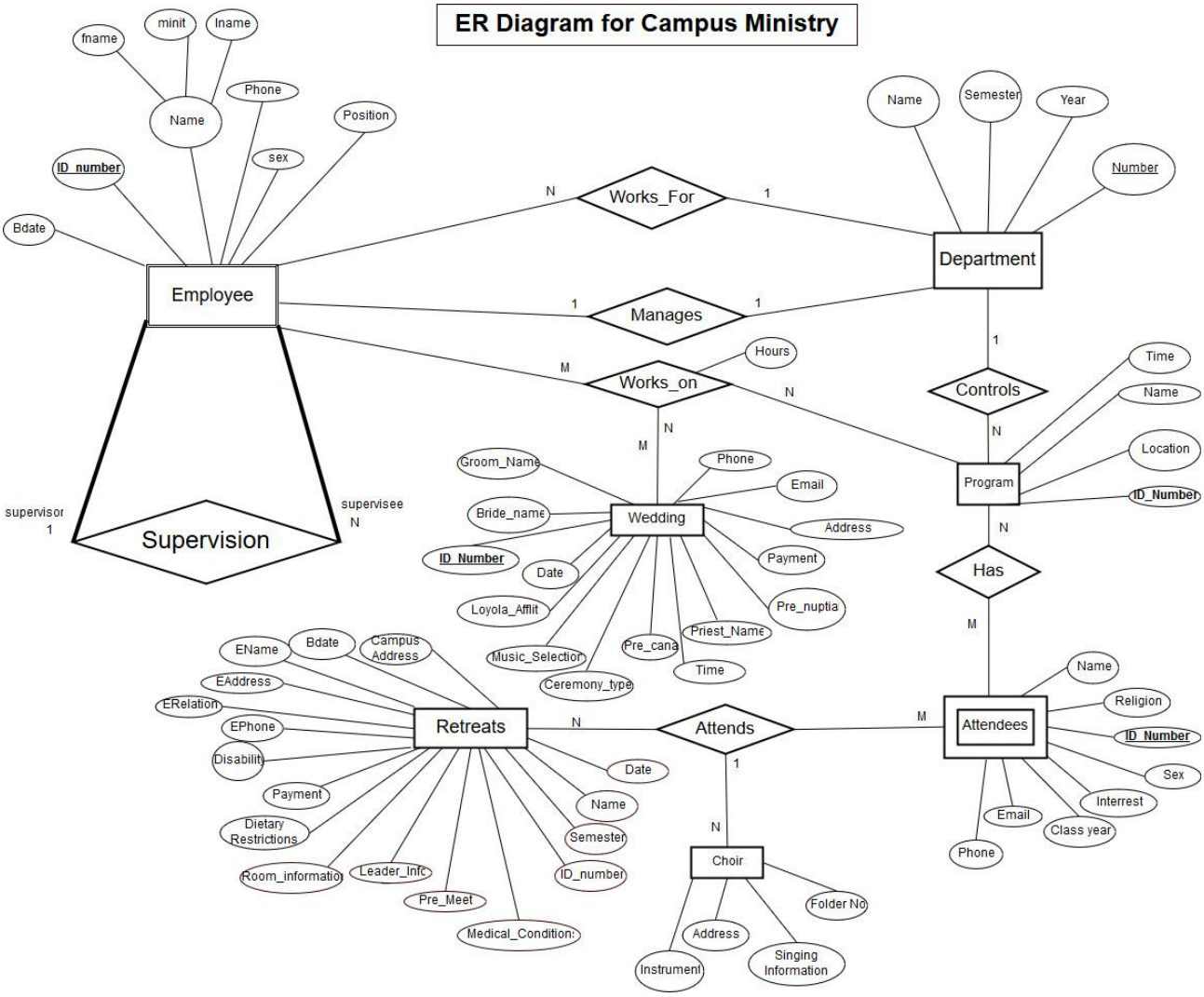

Figure 2 Entity Relationship Diagram for Campus Ministry Database

2.4.2 Relational Schema

The relational schema takes the ER diagram and dives further into more detail by explaining each entity and its relation to other entities. Below, the schema is divided up by each entity and each one is displayed by itself.

Department

- Name
- Semester
- Year
- Number

Program

- Time
- Date
- Location ID
- {Has(Attendees)}
- {Belongs(Department_Number)}

Choir

- Singing_Information
- Folder_no
- Address
- Instrument

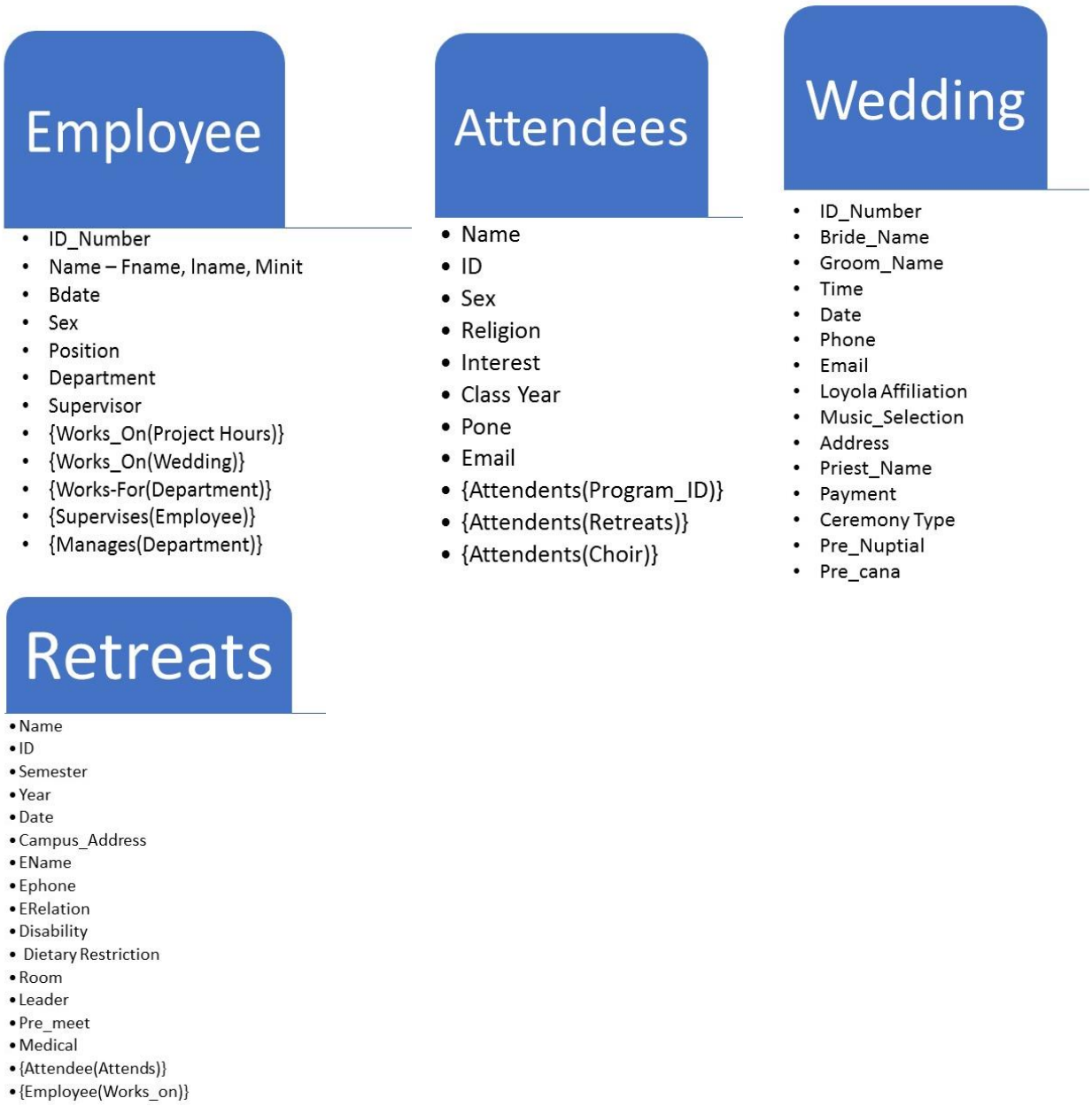

Figure 3 Functional Schema for Entities Department, Choir, Program, Employee, Attendees, Retreats and Wedding

In the above figure, each bullet represents single attribute for each entity and relationship with other objects is defined using brackets following by relationship and entity it has relationship with,

2.4.3 Relationship Schema

The relationship schema shows how each entity is related to other entities by displaying its relations as attributes. Below, the relationship schema for Campus Ministry Database. This figure references EER Key figure for symbols.

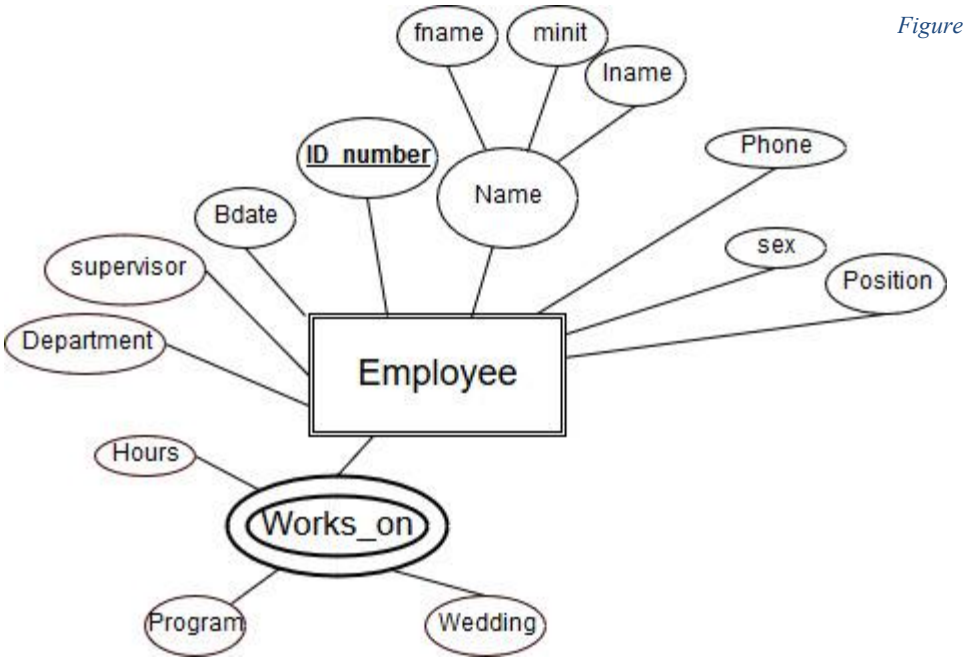

Figure 4 Relational Schema for Employee

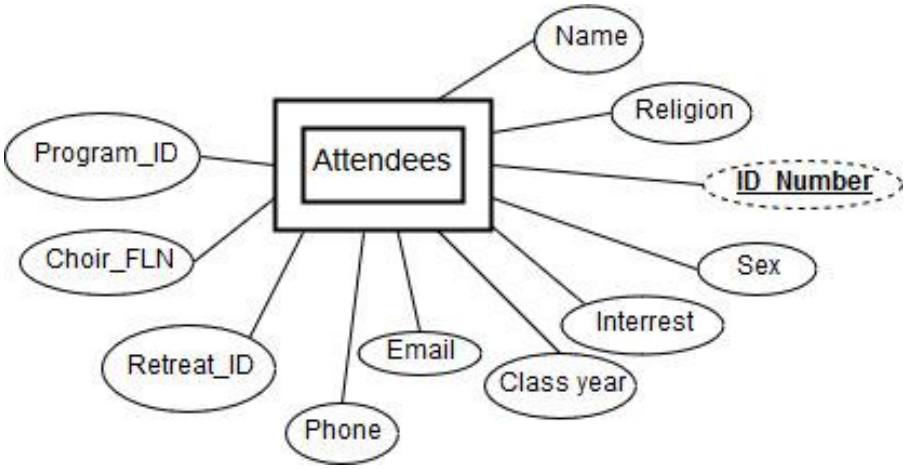

Figure 5 Relational Schema for Attendee

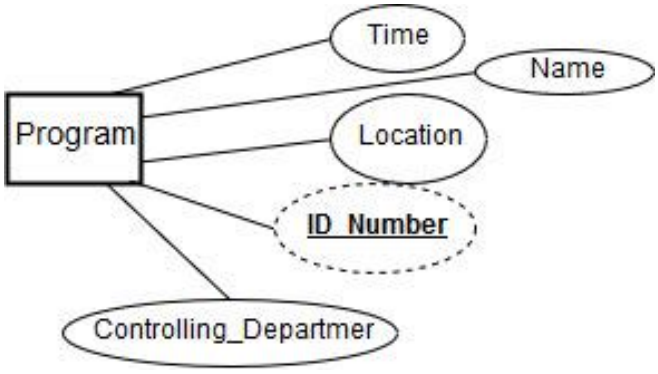

Figure 6 Relational Schema for Program

2.4.4 Data Dictionary

The data dictionary for Campus Ministry database states information for each variable within the database. This includes the variable type and its length. These variables reference the database as whole instead of each individual entity, allowing for repeated attributes to be stated at once.

Variable Name	Type	Length	Constraints
Fname	String	30	Characters only
Minit	Char	1	Characters only
Lname	String	30	Characters only
Sex	Char	1	Characters only
Bdate	String	10	Numbers and / only
Id_number	Integer	Varying from 2 to 16 – for different entities	Numbers only – automated by system
Semester	String	4	Two characters and two integers – FA17
Year	Integer	4	Numbers only
Name	String	50	Characters only
Religion	String	30	Characters only
Interest	String	60	Characters only
Phone	String	12	Numbers only and in format of xxx-xx-xxxx
Email	String	40	Characters only
Class_year	Integer	4	Numbers only
Hours	Float	4	Numbers only
Payment	Float	4	Numbers only
Date	String	10	Numbers only and / marks in format xx/xx/xxxx
Pre_meet	Boolean		Either yes or no for attending meeting
Dietary_restrictions	String	40	Characters only
Disability	String	40	Characters only
Leader_info	String	40	Characters only
Medical_condition	String	40	Characters only
Room_information	String	20	Characters only
Ceremoney_type	String	20	Characters only
Time	Float	4	Numbers only
Music_Selection	String	20	Characters only
Pre_nuptial	String	30	Characters only
Priest_Name	String	30	Characters only
Loyola affiliation	String	20	Characters only
Folder_no	Integer	2	Characters only
Instrument	String	30	Characters only
Singing_information	String	30	Characters only

Figure 7 Data Dictionary Table

3. Architectural and Component-Level Design

In this section, we will explain the overall architecture of this software as well as its individual components. This section will break down and explain the components in detail and presents limits, restrictions and implementation design features.

3.1 Program Structure

This software will be implemented using a Layered Architecture Structure in order to cater to all the different users and aspects of the software. This software includes web interface which is connected to database on the backend and has machine learning algorithm. The layered architecture will allow us to only show information to specific user which is pertinent to them and keep information hidden as necessary by user privileges. The layered architecture shows how the software as a whole would look and how the different parts will interact with one another.

3.1.1 Architecture Diagram

The figure below is the overall architecture diagram for the software. This only shows the broad over view for the architecture including servers and domains which will be used from already established applications such as MySQL.

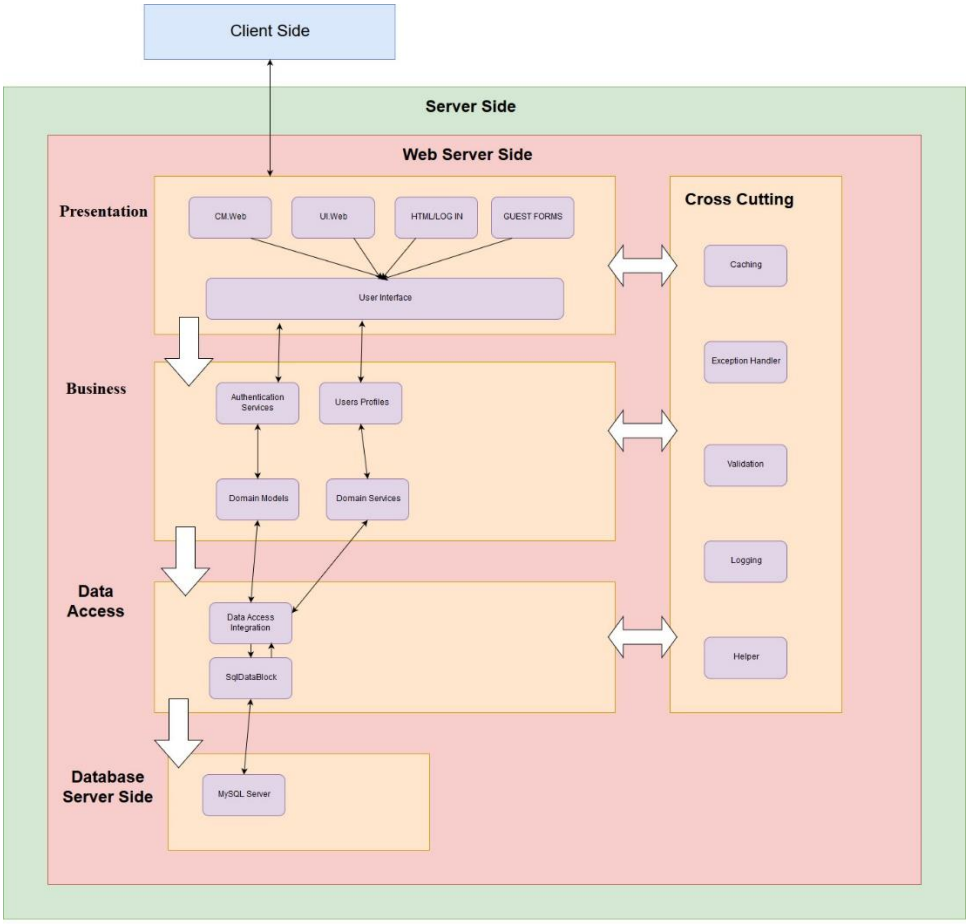

Figure 8 Software Architecture Diagram

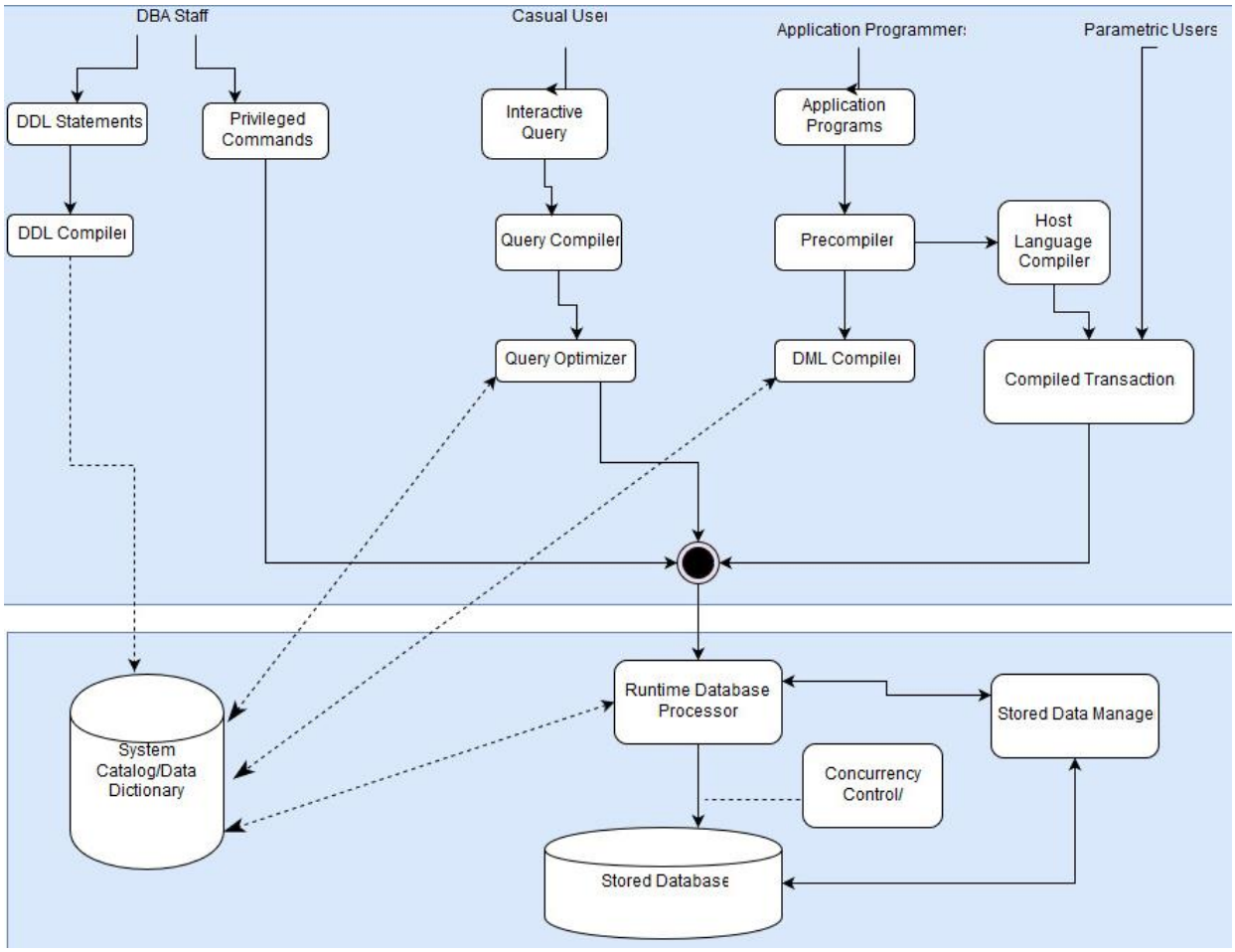

Figure 9 Detailed Architecture of the Software showing variation of all the users.

3.1.2 Alternatives

The figure above is a very broad overview of the different layers of how the different layers will be interacting. Using the layered architecture allows the use of encapsulation and information hiding to only allow authorized users to access to the specified information. Implementing a layered architecture has many benefits such as the software's flexibility in long lasting and adapt to changes over time. The second figure is a more detailed architecture for the software and how much of the software all the different users will get to interact with. As shown in the figure above, we have different users interacting with different applications depending on the type. The upper part of the layer is interface and involves users who are giving input to the software. The second layer is the processing and database portion which takes the user input and verifies that it agrees with data dictionary and processes it in database as needed.

3.2 Component Description

This section of the software will explain the various components of the software and their functionality within the software. A UML class diagram below shows all the components and how

they relate to one another. Following the diagram, we have explained each of the components in detail.

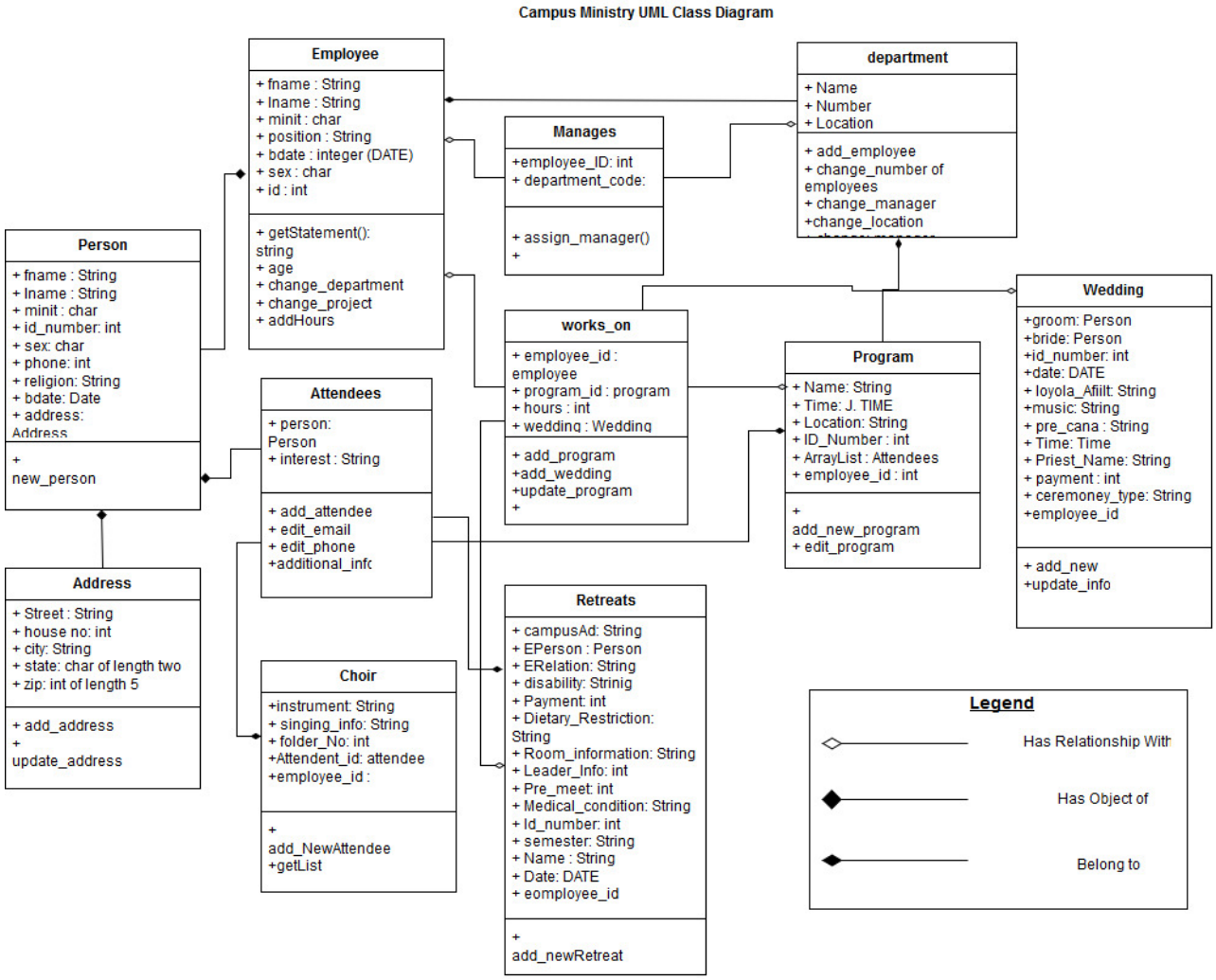

Figure 10 UML Class Diagram for Components

3.2.1 Description of Classes

The UML diagram above shows all the different components of the overall database. These components include all the various classes within the database and the information contained and collected within the database.

3.2.1.1 Processing Narrative for Component

As depicted in the UML diagram, each class has a relationship with another class. There are three types of relationships one object can have with another. One type of relationship one object can have with another is a has a relationship. In the above UML, an example is that a department and Employee have management relationship. Another relationship an object can have is “has an object of” which means that the class has the object as variable in within itself. In the description above, Employee has a person object within its class as a variable. Lastly, we have a belongs relationship, an example would be Employee belongs to a department. These relationship translates into how the classes communicate with one another and which class is able to invoke other classes.

3.2.1.2 Component Interface Description

The Interface for these classes will be implemented via forms and the staff members must go through authentication process before they are able to become an employee and use the software as an Employee. The interface description is included in the interface description for the overall software further ahead in the document.

3.2.1.3 Sub-Component Details of Classes

A sub component of Classes is the person class. The person class will be generic class to be used which has basic information which can be collected for a person.

3.2.1.3.1 Sub-Component Processing Detail

The person class will be implemented in Employee, Attendee and Wedding. The component will collect basic information about person but the information will be implemented and stored in database in connection to other classes within this software. For example, an Employee will have all the information for a person but a person cannot be directly entered but instead it must be an Employee who can be entered the system.

3.2.1.3.1.1 Restrictions and Limitations

As stated in the Processing detail, the Person class is used only as a Parent class and can only be used as with other classes within the software. A Person cannot be entered by themselves within the database and there will be no entity Person but instead an Employee with valid position and more information required of the class can be included.

3.2.1.3.1.2 Design Constraints

The person class is only a helper class and it must be implemented in the most simple and basic design possible. It should be programmed in least amount of lines of code and will simply have constraints for different type of information for a person and methods to be able to access those values later.

3.2.1.4 Sub-Component Details of Classes

Another sub-component captured from the UML diagram are the Employee, Department, Choir, Wedding, Retreats and Program classes.

3.2.1.4.1 Sub-Component Processing Detail

This sub-component captures the entities from the Entity Relation Diagram and transform them into classes along with their relationship to other classes. They capture, store and process the same exact and specific information defined in the EERD.

3.2.1.4.1.1 Interface Description

These classes will be implemented using web forms with the users will be filling out and Employees will access the information via their login. The interface description and example of the forms are provided in the interface section.

3.2.1.4.1.2 Restrictions and Limitations

As discussed in Processing details, these classes have been created directly from entities thus the information the values contained in these classes will be defined by those entities. All of the variables to be stored in these classes and processed are defined in the data dictionary table.

3.2.2 Description of Machine Learning Algorithm

3.2.2.1 Processing Narrative for Component

The Machine Learning Algorithm will be used for Staff to get prediction for when an event should be held if they want to have specific attendance. The algorithm will return a time of day, between morning, afternoon and evening along with the day of the week. For example

3.2.2.2 Component Interface Description

The Machine Learning Algorithm will involve minimal interface. The only input necessary from user is the attendance which will be accepted from user via Web form and the value returned will be displayed to the user.

3.2.2.3 Sub-Component Details

A sub-component of the MLA (Machine Learning Algorithm) is the implementation Algorithm being used for MLA. We will be using a multi-layered perceptron to model the features and run the algorithm.

3.2.2.3.1 Sub-Component Processing Detail

The Multi-Layered Perceptron will allow the implementation of different features such as attendance, program time, location and type. Depending on the type of prediction user is looking for that feature will be weighted heavier than others.

3.2.2.3.1.1 Interface Description

There will be no interface implementation for this sub-component as all the work will be done in the backend and the user will only be getting a result value.

3.2.2.3.1.2 Algorithmic Model

A Multi-layered Perceptron will be implemented and it will be process the data in batch mode. Various types of algorithms within the MLP class will be used to test which function works better for our data values.

3.2.2.3.1.3 Restrictions and Limitations

This algorithm is limited due to the data points a available for testing and verification. Since the previous data has not been kept for the modeling, the reliability for the work has to depend on new collected data.

3.2.2.3.1.4 Performance Issues

In general, MLP works for algorithm that need to process more features but it cannot be verified as being better since the data to test the algorithm is not available.

3.2.2.3.1.5 Design Constraints

For the MLA, testing and verification data is not available thus sample data needs to be acquired in order to test the algorithm.

3.3 Software Interface Description

The software is being implemented using HTML and CSS. All the users except for the Admin will be interacting with the software via Web Interface. More details about how the user interface is being implemented is explained in User Interface design description.

4. User Interface Design

4.1 Description of the User Interface

All the users interacting with the software will be using this interface to connect with the software and only the administrator has access to server side if necessary for them to check anything. The UI includes several guest forms to be accessed by un-priviliged users and login information system UI for privileged users. In the figures below, some of the sample forms and pages are displayed and their implementation is explained.

4.1.1 Screen Images

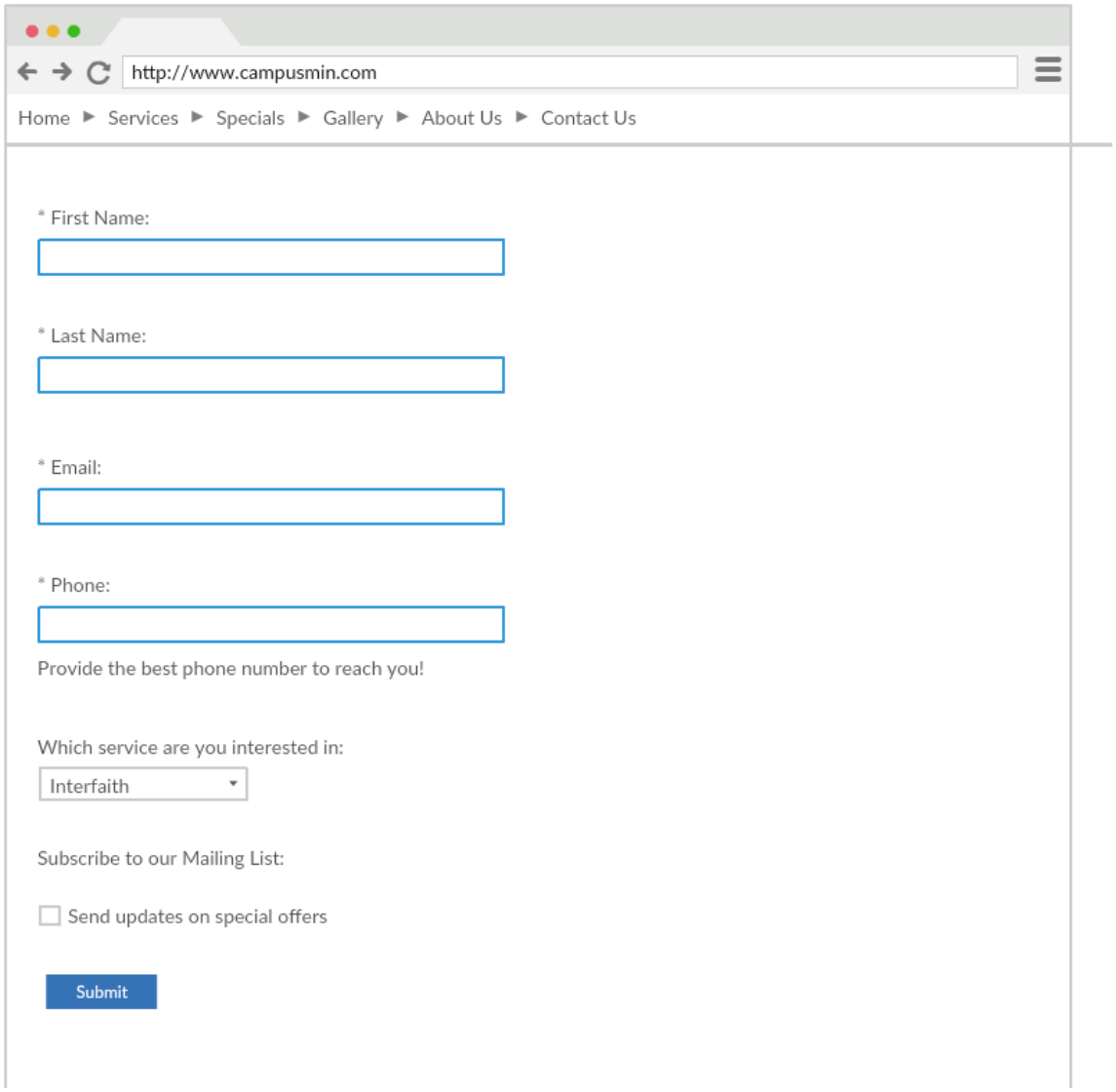

Figure 11 Campus Minstry Attendee Contact Form

The software will implement several different web forms and the template for the forms in general will be as displayed in the implementation above. It will have text field for user to enter information right under the question they are being asked and a submit button to confirm their submittion.

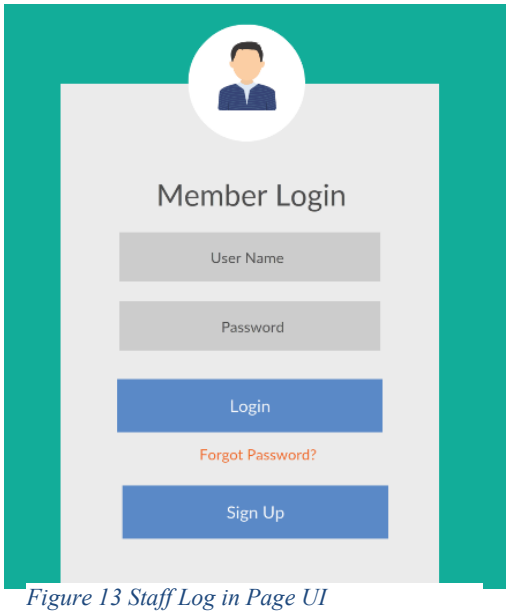

Figure 13 Staff Log in Page UI

The employee login page will be implemented in the style of the template above. It will give user options to login and sign up if that is what is needed.

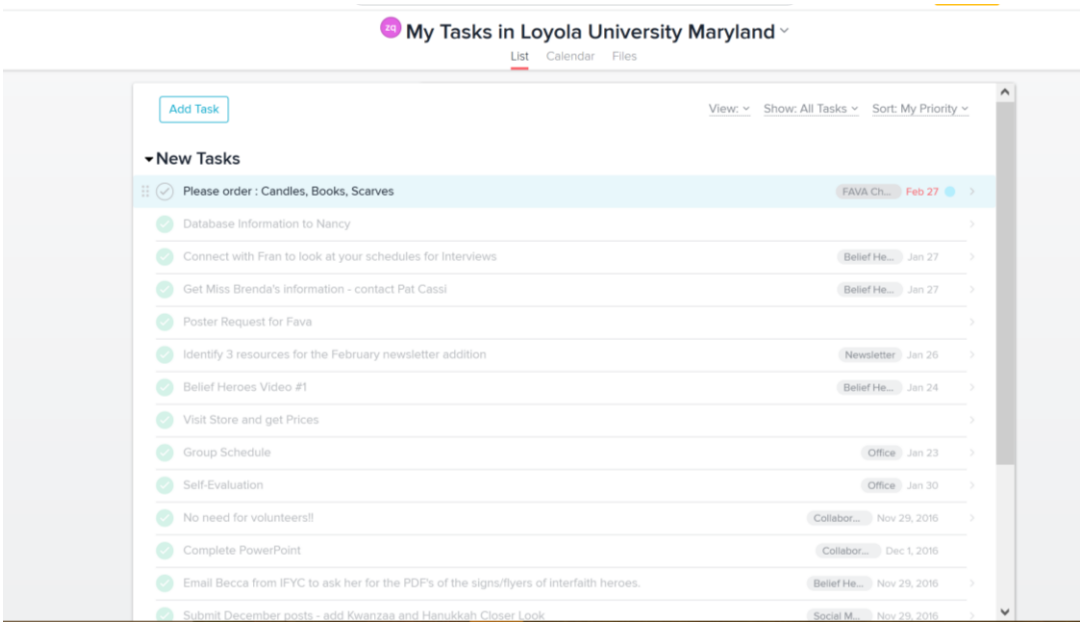

Figure 12 Employee Projects View Page

The page above displays a sample of what the employee would be seeing if they had listed all their programs and it would also show the amount of hours spent on that task.

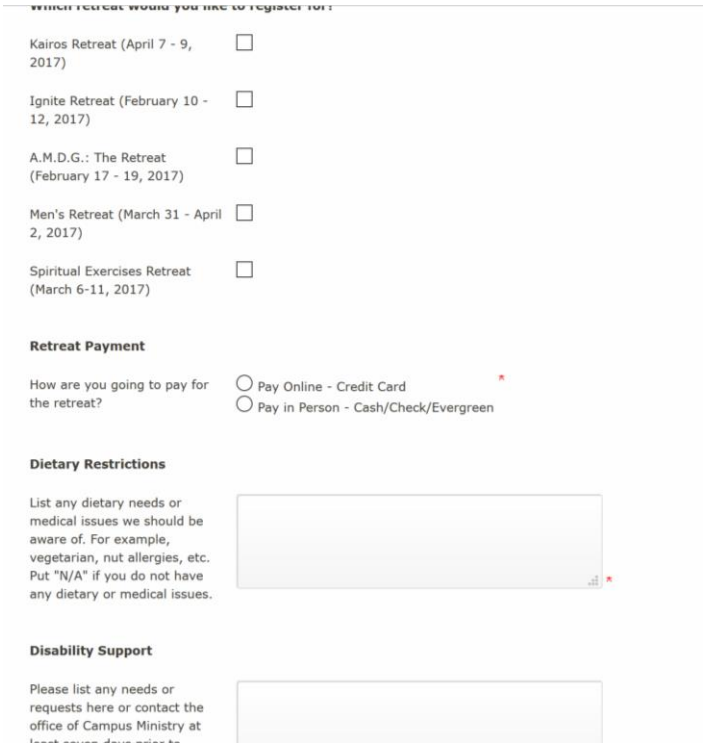

Figure 14 Retreat Sample Form

4.1.2 Objects and Actions

The user will have the choice to do different actions depending on where they currently are within the interaction. Since many of users will be guest, they will be primarily interacting with single page or form which will end with a submit button once they have completed the form with all the required information.

4.2 Interface Design Rules

In order to enhance the usability of this software we will be implementing several different rules with in the interface. The interface must implement universally used button and symbols in order to allow different groups of users to understand the actions which need to be completed. For example, a back button should be implemented using a back arrow instead of forward arrow. The software must have completion of action for the various forms user interacts with – there should be submit button or some sort of confirmation that the information has in fact been submitted and is completed. The interface must be allow user to recover easily from errors and retain information needed without user having to remember from one screen to another. The interface must be symmetrical in its implementation and not have un-aligned fields.

4.3 Components Available

The interface will be implemented using HTML and CSS, thus we will have a variety of different styles and options to choose from. Any kind of implementation is possible with the two options. Both of these languages together give the power to the designer to design as needed by the users and by the software. We can make the forms as detailed as needed or as broad. For example, as discussed above, a date will string of length 10, so we can simply have a calendar for user to pick

from or have them insert date. The interface can be implemented using as many components necessary to make the interface within the design rules as well understandable within one sitting.

4.4 User Interface Development System Description

The process to develop the user interface will be start to with guest forms such as the Contact form, Wedding forms and Retreat forms. Since those forms are individual entries and do not impact the other parts of the interface, once they are completed. Black Box Testers will test the forms to understand how easy it is for them to use them to evaluate what needs to be changed. Once a general style is selected from working on the guest forms, then employee interface will be created using the standards from the testers for guest forms.

5. Restrictions, Limitations, and Constraint

One of the components of this software is the Machine Learning Algorithm which will be implemented to find when a event should be held to plan for specific amount of attendance. Although this component will be implemented, its performance cannot be verified and tested with real data because it is not available by the Client. The Client has already mixed the data into one big excel folder which takes away pertinent information such as event time, attendees and day. Without real data from previous events, the system can only predict on sample data until database is able to accumulate enough data from usage over time.

6. Appendices

6.1 Requirements Traceability Matrix

The table below lists all the requirements and explains how the different components cover the functional requirements set by the Client.

Requirement	Component	Rational
Database Functional Requirements: REQ-1 to REQ-10	1	Component 1, the different classes created cover all the requirements within the database. • A new staff member can be added • Interns work on projects, retreats and weddings. • Staff supervise interns • Retreats have list of members, semester, year and date. • Bride and Groom person have wedding at Campus Ministry – collect different information about Bride and Groom.

		• Supervisor can check intern hours.
Web Interface: REQ 11-20	User Interface and Component 1	These functional requirements under web interface are covered by user interface such as: • Log in page exists • Guest forms for Choir, Wedding, Interested Participants and event attendance. • Queries results • Hours updated and viewed. • Overview of projects. • Download query results.
Machine Learning Algorithm: REQ 21-27	Component 2 and User interface	The functional requirements under Machine learning are covered by component 2 and its sub components. • Pull data from previous events • Create weights and preprocess data to add weights. • Run algorithm on data to train, test and verify. • Return results to user - a day and time of day given to user on interface.

6.2 Packaging and Installation issues

The Client is still working to confirm where the server and database will be implemented. Initially it was suggested that it would be implemented on a physical machine but due to maintenance concern, the prospects for Cloud hosting are being explored. The actual installation is very simple once the location has been decided.

6.3 Design Metrics

The software is being designed to implement the database and interface first followed by Machine Learning Algorithm thus more part of the document was spent on database than on MLA. The figures presented in this document will be used to implement the software. More components and data structure might become necessary as we move further in implementation but as they are included in the software, this document will also be updated along with figures and data.

6.4 Supplementary Information

6.4.1 Architecture Design

This section includes some sample information which shows smaller components and part of the architecture and possible help for the implementation.

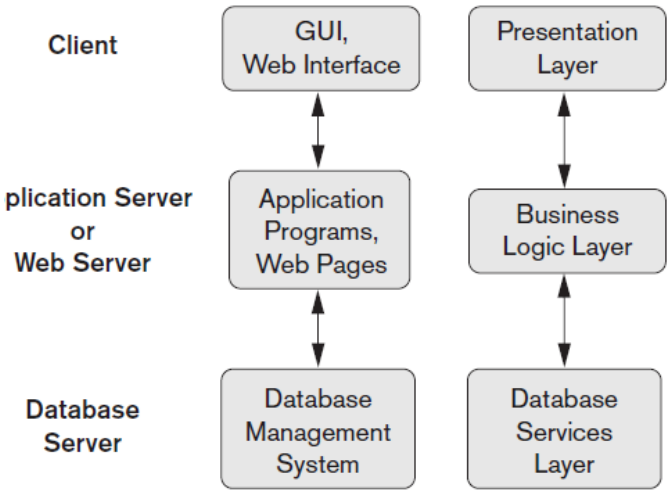

Figure 15 Overview of Architecture Layers

6.4.2 DBMS Type

This Database is Homogeneous DBMS because the same database will be used for all the different forms and interfaces. The database is a special purpose database and it can only be used by Campus Ministry because much of information being implemented is only used campus ministry.