Design Document

CSCE 361 – Fall 2017

Honors Project

Colton Harper

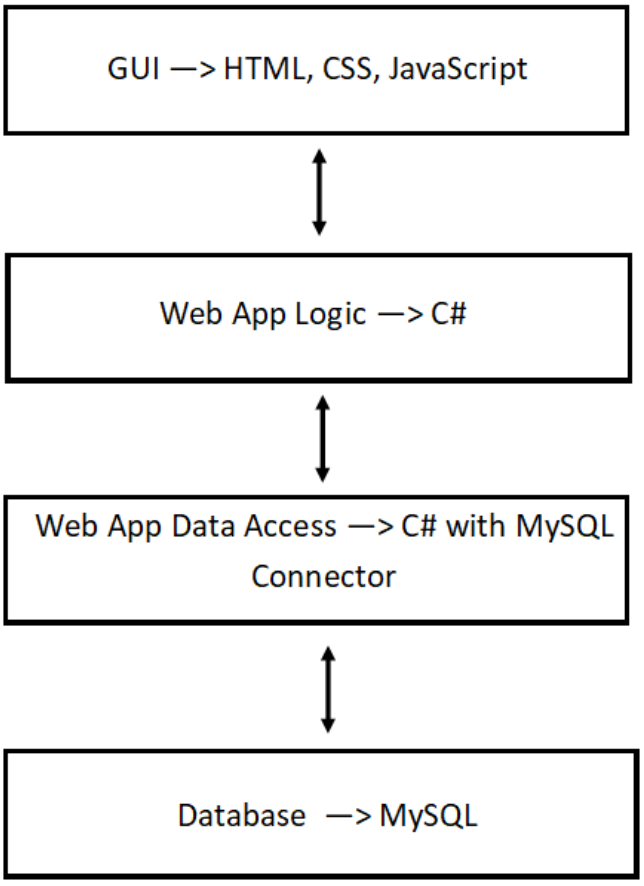
Contents

1. Introduction

This design document aims to provide details about the high-level architecture and entity relations that will be used for the Lincoln Dev Studio software system. More specifically, this document presents the architecture, entity diagrams to visualize the parts of the system. Furthermore, this document contains details regarding the database tables and the relationships therein. The intended audience of this document is system architects and software engineers who will implement and maintain the system described below.

2. Architecture

2.1. Introduction



The Lincoln Dev Studio system will implement a layered model for its high-level architectural design. The layers included in the model for this system include the GUI layer (top layer), the web app logic layer (second layer), the web app data access layer (third layer) and the database layer (fourth/lowest layer). The users will directly interact with the GUI layer. The GUI layer will store information to and retrieve information from the database for the user through the web app logic layer and the web app data access layer.

2.2. Modules

2.2.1. GUI Layer

The principal function of the GUI layer is to provide a simple user interface for the users to interact with. Users will be able to interact with a web form. The web form will allow users to input their basic information required by the Lincoln Dev Studio services and submit this information for storage on the database. On a hidden web page, users with the exact URL will be able to use the GUI to view the community group information stored in the database.

HTML, CSS, and JavaScript will be used to create the web pages as specified in the requirement specification. When a user performs an action on this system front-end, the information will be sent down one layer, to the web app logic layer, where it will be handled.

2.2.2. Web App Logic Layer

The web app logic layer is responsible for performing all the business logic for the system. When information is being retrieved from the database and pushed up to the GUI layer, this layer will parse the database information passed by the web app data access layer into C# objects. Then it will perform the business logic on the objects and finally pass the information to the GUI layer. In the case that the user has performed an action that stores information to the database, the web app logic layer will also perform the necessary business logic on the information and then store pass this information to the web app data access layer. This will also be responsible for sending a confirmation email to the user upon request submission.

2.2.3. Web App Data Access Layer

The primary function of the web app data access layer is to query the database for data that is requested by the top layers of the systems. The web app logic layer will inform the web app data access layer what information it needs, and the web app data access layer is responsible for retrieving the information from the database. The web app data access layer is also responsible for creating appropriate statements to store information it receives from the web app logic layer. This layer will use the MySQL Connector/Net to connect to the database for information storage and retrieval purposes.

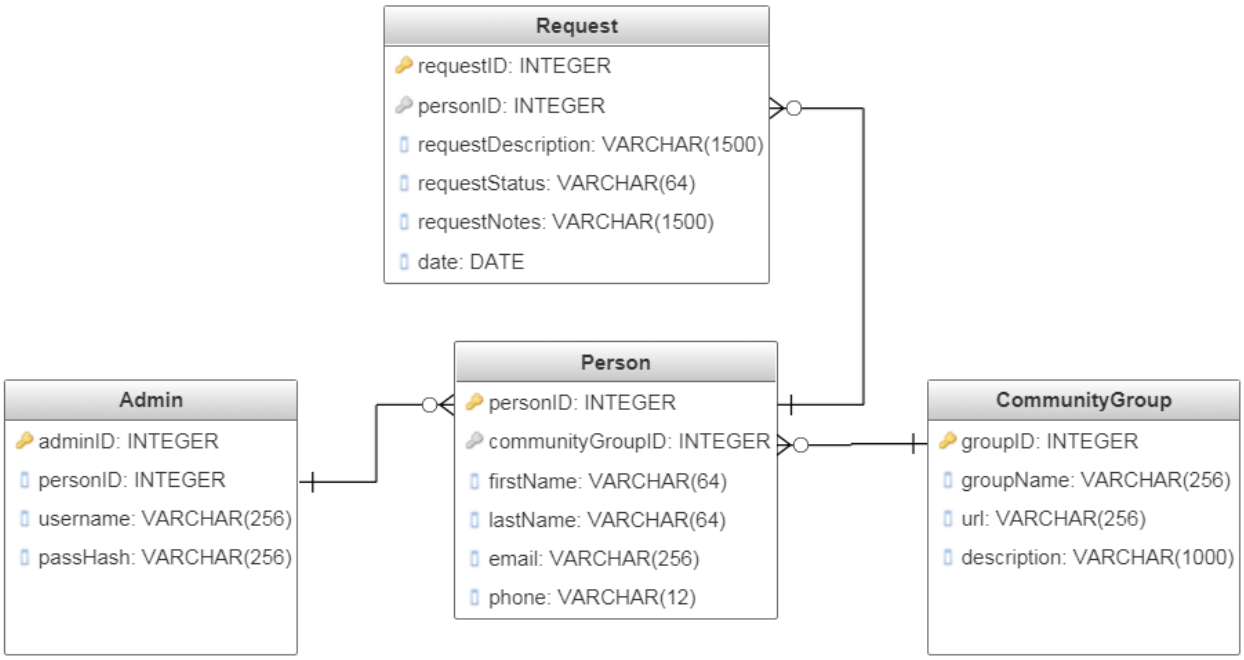
2.2.4. Database

The lowest layer, the database layer, will store all the data submitted by the community groups. This database will be developed using MySQL and will be hosted on the UNL CSE server. The web app data access layer will be the only layer that can access the database layer.

3. Class Diagrams

3.1. Database

3.1.1. Database Schema



3.1.2. Schema Description

3.1.2.1. Person Table

The Person table contains basic personal information for individual users. The information stored in the Person table includes the person's community group affiliation (communityGroupID), the person's first name (firstName), last name (lastName), email address (email), and phone number (phone). A person submits a request and is associated to a community group.

3.1.2.2. Community Group Table

The Community Group table stores the basic community group information, which includes the community group's name (groupName), web address (url), and a brief description of the service they offer (description).

3.1.2.3. Request Table

The Request table stores information about the request a user submits to the database. The information stored in the request table includes a reference to the person who is the primary contact or submitted the request (personID), a description of the community group request (requestDescription), the status of the request (requestStatus), and any notes added to the entry by special access users (requestNotes).

3.1.2.4. Admin Table

The Admin table stores the username (username), password (password), and a reference to the person who is associated to those credentials (personID).

3.2. Class information

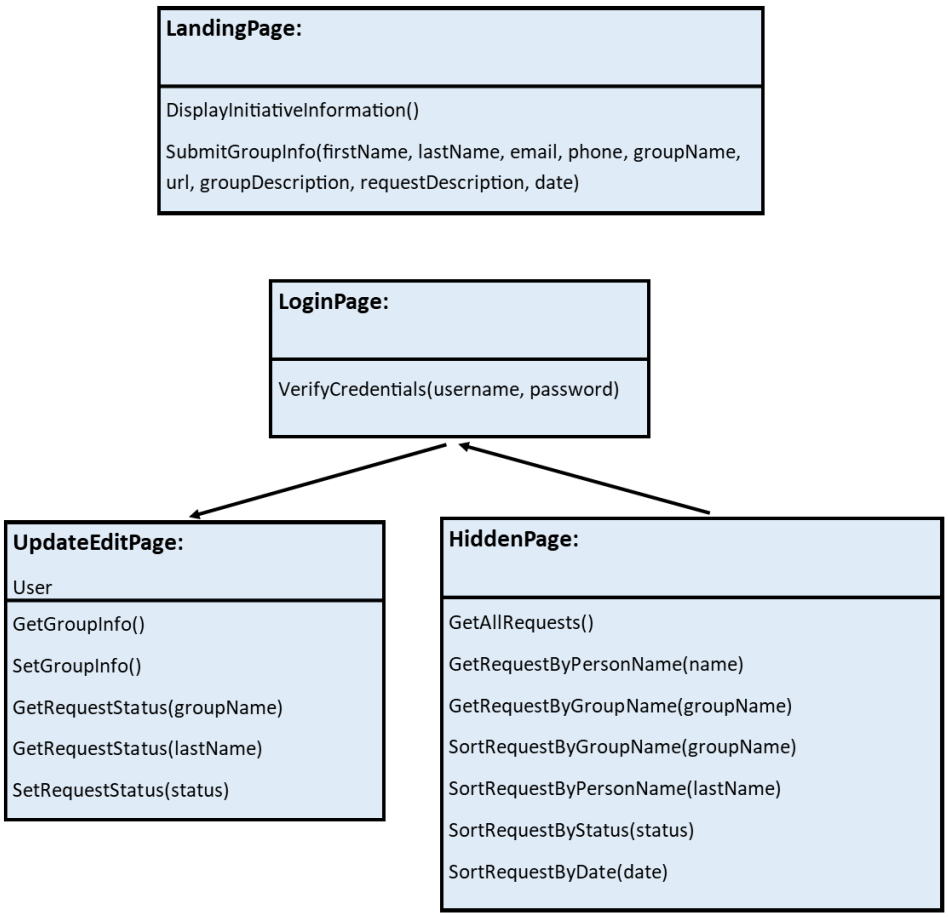
The data from the database will be represented as C# objects in the web app logic layer. An entry in the table will be represented by an object. The objects will have attributes that correspond to the data fields for its respective data table. For instance, a person class will be created. The attributes of a person include their group affiliation, first and last name, email address, and phone number. The web app logic layer will use these objects to generate the proper information for the GUI.

3.3. GUI Layer

There will be four web pages that website will consist of. These pages include the general web form page, the hidden information view page, the login page, and the restricted access page for editing information. The landing page will display basic information about the software development initiative and other pertinent information. There will also be a web form on the landing page. The web form will have instructions regarding what information should be inputted into the web form. The user will have an option to press a button to submit the information entered in the web form to the database.

Users with the URL will be able to navigate to another webpage where they will be able to view the information stored in the database. The user will have different viewing options. The user may search for a community group by group name or contact first or last name. The users may also sort the information displayed on the webpage. The sort options will consist of sort by group name, sort by contact last name, sort by submission date, and sort by status.

Some users will be given access to a restricted web page. On the aforementioned hidden webpage, there will be a login button. This login button will take the user to a log-in page where users will be prompted to enter their account username and password. After their account information is submitted, the information is compared with the credentials stored in the database. If the information is not an exact match, a message will be displayed to inform the user that the information does not match a valid account. If there is an exact match, the user will be redirected to the edit page. On this edit page, users will have the ability to add notes to, edit the status of, or remove entries in the database.


LandingPage:
DisplayInitiativeInformation()
SubmitGroupInfo(firstName, lastName, email, phone, groupName, url, groupDescription, requestDescription, date)
LoginPage:
VerifyCredentials(username, password)
UpdateEditPage:
User
GetGroupInfo()
SetGroupInfo()
GetRequestStatus(groupName)
GetRequestStatus(lastName)
SetRequestStatus(status)
HiddenPage:
GetAllRequests()
GetRequestByPersonName(name)
GetRequestByGroupName(groupName)
SortRequestByGroupName(groupName)
SortRequestByPersonName(lastName)
SortRequestByStatus(status)
SortRequestByDate(date)