

**Keller-Williams Family Check-In**

**Technical Design Document**

**version 1.0**

**Design team: Patrick Bailey, Josh Garnick, Aaron Flager**

**Table of Contents**

**1 Introduction ………………………………………………… 4**

1.1 Purpose ………………………………………………. 4

1.2 Scope ………………………………………………… 4

1.3 Document Organization ……………………………… 5

1.4 Audience ……………………………………………… 5

**2 Design Overview ……………………………………………. 6**

2.1 Approach ……………………………………………… 6

2.2 Architectural Goals and Constraints ………………….. 6

2.3 Guiding Principles …………………………………….. 7

2.3.1 Functionality First …………………………….. 7

2.3.2 Scalable ……………………………………….. 7

2.3.3 Responsive Design …………………………….. 8

2.3.4 Easy to Use …………………………………….. 8

2.3.5 Connecting KW Family ………………………… 8

2.4 Design Patterns …………………………………………. 8

2.5 Design Principles ……………………………………….. 9

**3 Use Cases ……………………………………………………… 10**

**4 Application Architecture ……………………………………… 16**

4.1 MVC Design …………………………………………… 16

4.1.1 Model …………………………………………… 17

4.1.2 View …………………………………………….. 17

4.1.3 Controller ……………………………………….. 17

4.2 Data Structure …………………………………………… 18

**5 Application Implementation ………………………………… 19**

**6 Database Architecture ……………………………………….. 21**

**7 Assumptions and Constraints ……………………………….. 23**

**Appendix A: Acronyms …………………………………………… 24**

**Appendix B: Products and Tools ………………………………… 25**

**Appendix C: Data Dictionary …………………………………….. 28**

**1. Introduction**

**1.1. Purpose**

The purpose of this document is to detail the technical characteristics of the Keller-Williams Family Check-In (KWFCI) system by providing an overview for implementation and design.

Primary focuses of the KWFCI system include:

* Organizing the interactions necessary to keep connected to significant Keller-Williams brokers.
* Facilitate more effective communication through automated and web based services.
* Provide feedback on current office task status through automated alerts and notifications.

This document will not address every circumstance of user implementation or installation. When possible, clarification of intent and assumptions will be provided.

As development progresses this document may be adapted and updated to accommodate any changes in implementation.

**1.2 Scope**

The KWFCI will allow Keller-Williams staff members to sign in to a single web application interface.

The interface will have the ability to record broker interactions the staff enter into the system, with specified parameters being tracked.

The interface will be able to refer to past recorded broker interactions.

Tracking office tasks is available through the interface, with tasks assigned to specific or general staff.

Notifications on the status of upcoming or past due tasks are displayed and their priority changed automatically as their due date approaches.

Staff will be able to use the interface to send mass email to a specified collection of brokers/staff using the interface.

Administrative management of the database is available through an admin interface that is separate from the staff interface.

**1.3 Document Organization**

This document is organized into the following sections:

|  |  |
| --- | --- |
| Introduction | Information related to this document. Includes purpose, scope, organization, and audience descriptions. |
| Design Overview | Defines the approach taken to implement the KWFCI system. This includes design goals and philosophies as well as technical details. |
| Use Cases | Details regarding the KWFCI usage by invested actors. |
| Application Architecture | Information explaining the design architecture of the KWFCI system. |
| Application Implementation | Directory structure and relevant details about how the KWFCI system is organized. |
| Database Architecture | Specifics of the database arrangement and architecture as it relates to the implementation of the KWFCI. |
| Assumptions and Constraints | A listing of assumptions made during the development process by the implementation team, as well as any constraints to be considered. |
| Appendix A | Acronym and abbreviation definitions. |
| Appendix B | Products and tools used. |
| Appendix C | Data dictionary of the system. |

**1.4 Audience**

The intended audience for this document include KWFCI design and implementation stakeholders, interface design team members, and technical guidance advisers.

**2. Design Overview**

**2.1 Approach**

The development of the Keller-Williams Family Check-In (KWFCI) system will occur via a multiple phase process:

* *Requirements Phase* – an initial gathering of the necessities of the system through communication with the Keller-Williams staff, creative impressions on possible development from the implementation team, and utilizing similar established architectures to format an initial impression of requirements. First development of design document.
* *System Design Phase* – the implementation team refines development requirements into a working theory prototype. Prioritization of features is established to work within the time frame available for development. Design document changed in accordance with implementation adjustments.
* *Construction Phase* – refinement and beginnings of implementation for the system. The implementation team uses established guidelines to ensure progress towards the completion of the project. Limited design document adjustments, only in accordance with shifts of priority from Keller-Williams staff or unexpected necessary adaptations. Agile implementation practices are used to account for changes in scope.
* *Implementation Phase* – quality assessment and training interactions between implementation team and Keller-Williams staff. Ensure that that project is fulfilling the requirements previously discussed to a level of satisfaction that the Keller-Williams staff is comfortable utilizing. Finalization of any changes to the design document.

**2.2 Architectural Goals and Constraints**

The intent of the KWFCI system is to provide an accessible and efficient organizational interface for the Keller-Williams staff to keep track of communication between them and brokers which require specific personal interactions.

The KWFCI system will provide services to enhance organization, including, but not limited to:

* Recording significant details of specific communication events between staff and brokers.
* Providing alerts to bring attention to significant events that are time sensitive.
* Enable multiple brokers to be notified at once about future meetings or significant information.
* Sort and organize detailed information about communication exchanges.
* Track upcoming tasks to be accomplished by office staff.

The KWFCI system will be designed keeping scalability and region specific adaptations as possible future adjustment requirements. Implementation is designed to facilitate better organization of communication, but the majority of details of such communications are provided by Keller-Williams staff and are subject to any errors that may occur during initial data entry. The scope of access to the KWFCI system will be dependent upon what hardware services the Keller-Williams staff decides to utilize for hosting.

**2.3 Guiding Principles**

The guiding principles for the KWFCI system provide a foundation for functionality and design. These provide a standard of quality to be met in order for a feature to be considered for implementation. The guiding principles provided here are in addition to design best practices and ethical standards which will be maintained by the implementation team.

**2.3.1 Functionality First**

The final goal of the KWFCI system is to provide organization and ease of understanding for the Keller-Williams staff to ensure that communication remains strong and positive between them and the brokers. With this in mind, aesthetic considerations will be taken, but only when the functionality of the system remains intact or enhanced by aesthetic changes.

**2.3.2 Scalable**

The intent of initial design for the KWFCI system is to provide an effective tool for local Keller-Williams staff. The scope of staff which can utilize the system has the potential to increase, so the system will be designed with this in mind. Local focus initially with the possibility of a broader implementation at a later date.

**2.3.3 Responsive Design**

The KWFCI system will provide efficient and productive feedback to the users of the system. Design parameters which would limit the functionality and responsiveness of the system will be adjusted accordingly.

**2.3.4 Easy to Use**

The intended users of the KWFCI system are the Keller-Williams office staff who will be interacting with brokers. The design of the system will reflect the intended user audience and provide clear and meaningful interfaces to allow for helpful organization of communication between Keller-Williams staff and brokers. Ease of use is determined by input from the intended users of the system as well as any significant stakeholders.

**2.3.5 Connecting Keller-Williams Family**

The KWFCI system is intended to enhance and clarify interactions between Keller-Williams staff and brokers. If any part of the system is inhibiting or restriction the ability to allow communication between Keller-Williams staff and brokers it will be reassessed and modified accordingly.

**2.4 Design Patterns**

The implementation team will follow a Model, View, Controller (MVC) design structure when creating the KWFCI system. The MVC design structure will allow for separation of concerns for data identification, user interface, and system interaction. In addition, the implementation team will utilize design elements that effectively enhance the MVC design process.

The design structure elements specifically being focused on for implementation are:

* *Model* – the logic for the system's data.
* *View* – display of the system's user interface.
* *Controller* – component to handle system interactions.
* *Unit* *Testing* – tests to ensure proper functionality of system components.
* *Interfaces* – allow for wider definition of objects without immediate implementation.
* *Templates –* guided structures to ensure consistency. This includes modals.
* *DRY* *approach –* 'do not repeat yourself' as a means of enhancing system efficiency.

**2.5 Design Principles**

The KWFCI system is designed to be utilized through a web browser that can support a web application utilizing modern client-server communications for validation, processing, and display.

Any Keller-Williams business rules will be programmed into the system, as communicated by Keller-Williams staff to the implementation team. Support for keeping business rule implementation up to date is at the discretion of whoever the Keller-Williams staff employs to continue to provide maintenance for the KWFCI system after initial design.

**3. Use Cases**

The KWFCI Use Case Actors are identified and defined as:

* *KWFCI System*
* *New Broker*
* *Broker In Transition*
* *Transfer Broker*
* *Staff member*
* *Administrator*

**The KWFCI System**

This actor represents the system itself. Most automated tasks will be the result of event triggers and state processes defined by the developers for specific tasks. The automatic generation of user requirements on a new broker, the application of a staff member’s unique user ID to an interaction, and the tracking of staff logging in/out are examples of the System’s duties.

**New Broker**

A new broker is defined as a broker who has been newly added to the team without prior involvement with the company. New Brokers exist in the use cases as actors for representation purposes only and do not ever interact directly with the system.

**Broker in Transition**

A broker in transition is defined as a broker who is currently employed by Keller-Williams but due to extenuating circumstances has been inactive for some significant amount of time and needs reorientation into Keller-Williams systems, policies, and practices. A broker in transition does not have the same mandatory requirements that a new broker does and never interacts with the KWFCI system directly.

**Transfer Broker**

A transfer broker is a broker who has been moved from one Keller-Williams office to another. These broker’s require the same amount of interactions with a Keller-Williams staff as the other two types of brokers, but are already integrated into the Keller-Williams system and therefore do not require additional management. Like the other types of brokers, transfer broker’s never directly interact with the KWFCI system.

**Staff Member**

Staff members are any Keller-Williams employees whose job it is to assist brokers through the orientation period. The KWFCI’s primary user will be the Keller-Williams staff member, and it is for them that this system is being designed. Staff members are responsible for managing new brokers, brokers in transition, and transfer brokers through the KWFCI system.

**Administrator**

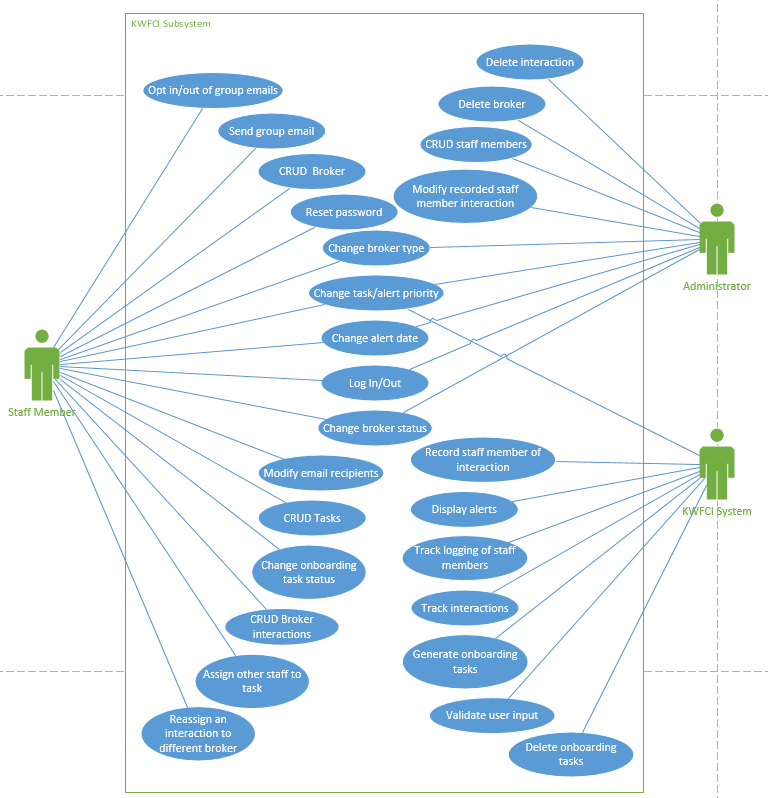
The administrator is a Keller-Williams employee who has been given authority to manage other Keller-Williams staff members. The administrator’s responsibilities include managing staff members and modifying details normal staff members shouldn’t, such as the unique user ID of a staff member who created an interaction. Other responsibilities will include permanently removing data from the database as needed, such as interactions and brokers whose information is no longer required to keep.

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | **Use Case** | **Type** | **Description** |
| Administrator | CRUD\* Staff Members | External | - Create, Retrieve, Update, and Delete staff members |
|  | Log In/Out | External | - Log the administrator into and out of the system |
|  | Modify recorded staff member of interaction | External | - Change the userID of the Staff Member applied to an interaction |
|  | Delete interaction | External | - Remove an interaction from the database |
|  | Delete Broker | External | - Remove a broker from the database |
| System | Track logging of members | State | - Create a timestamp in the database automatically each time a staff member or Administrator logs into or out of the system |
|  | Track interactions | State | - Add to count of interactions out of 10 each time a new interaction is saved |
|  | Generate Requirements list | Trigger | - When new Broker is created automatically apply list of onboarding tasks to new Broker |
|  | Display Alerts | State | - Display all alerts whose due-date is within defined period (i.e. 1 day, 1 week, etc.) on Staff Member’s Alerts page. Display critical alerts on all pages |
|  | Record staff member of interaction | State | - Apply unique employee userID to interaction to each interaction Staff Member creates |
|  | Generate onboarding tasks | Trigger | - Create an individual task of type “onboarding” for each item on the requirements list for a new broker when a new broker is created and assign them to this broker. |
|  | Validate user input | State | - Ensure proper error messages are displayed when input formatting is incorrect |
|  | Delete onboarding tasks | Trigger | - Automatically delete all onboarding tasks for a new broker when all existing tasks are marked as complete |
| Staff Member | CRUD\* Broker | External | - Create, Retrieve, Update, and Delete\*\* Brokers |
|  | CRUD\* Broker Interactions | External | - Create, Retrieve, Update, and Delete\*\* Interactions between Brokers and Staff Members |
|  | Change Broker Status | External | - Change whether a Broker’s status is “Active”, “Away”, or “Inactive” |
|  | Send group email | External | - Send email to all Brokers whose “Receive Group Emails” property is set to “true” |
|  | Modify email recipients | External | - Change a Broker’s “Receive Group Email” property to “true” or “false” |
|  | CRUD\* Tasks | External | - Create, Retrieve, Update, and Delete Tasks |
|  | Change Onboarding task status | External | - Mark a new Broker’s onboarding task as complete or incomplete |
|  | Opt in/out of group emails | External | - Staff members change their own setting for receiving group emails or not on settings page |
|  | Reset password | External | - Staff member sends themselves an email with link to reset their password |
|  | Change broker type | External | - Change the type of a broker to “New Broker”, “Broker In Transition”, or “Transfer” |
|  | Assign staff to task | External | - Change the staff member assigned to a task or assign to no one |
|  | Reassign interaction to different broker | External | - Change which broker an interaction is assigned to |
|  | Change task/alert priority | External | - Manually select a priority from 1 to 5 |
|  | Change alert date | External | - Change or clear the date the system will alert staff members about this task |

**\*** CRUD is an abbreviation for the common set of database functions that must exist for each entity within a database. Those functions are Create/Retrieve/Update/Delete. It is being used here to shorten the list of repetitive use cases present with each entity within the KWFCI Database by turning 4 use cases into 1.

**\*\*** A staff member will have the ability to archive brokers and interactions, but not remove them from the database. This action simulates the Delete function of CRUD, but the actual removal of these objects from the database is the administrator’s responsibility.

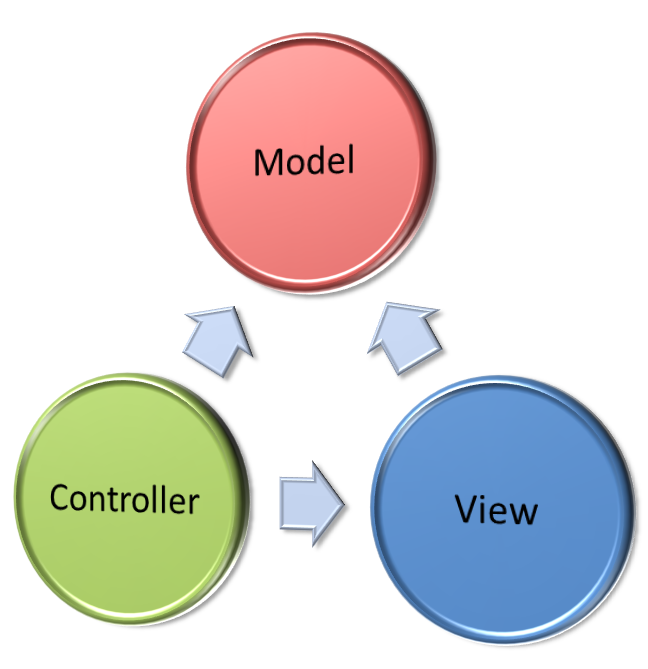
**KWFCI Use Case Diagram:**



**4. Application Architecture**

**4.1 Model, View, Controller (MVC) Design**

The Keller-Williams Family Check-In (KWFCI) system will be designed utilizing the MVC architecture for organization and development. Separation of concerns for the program will be clarified using the *Model, View,* and *Control* design structure.



*Flow of concerns between* Model*,* View*, and* Control

**4.1.1 Model**

The model is a representation of the system data. It is not the actual data, but rather an interface to the data. The model allows the program to pull data from the database without becoming entangled with the intricacies of the underlying database. The model provides an abstraction layer with the database, thus enabling the model to be reused for multiple databases.

**4.1.2 View**

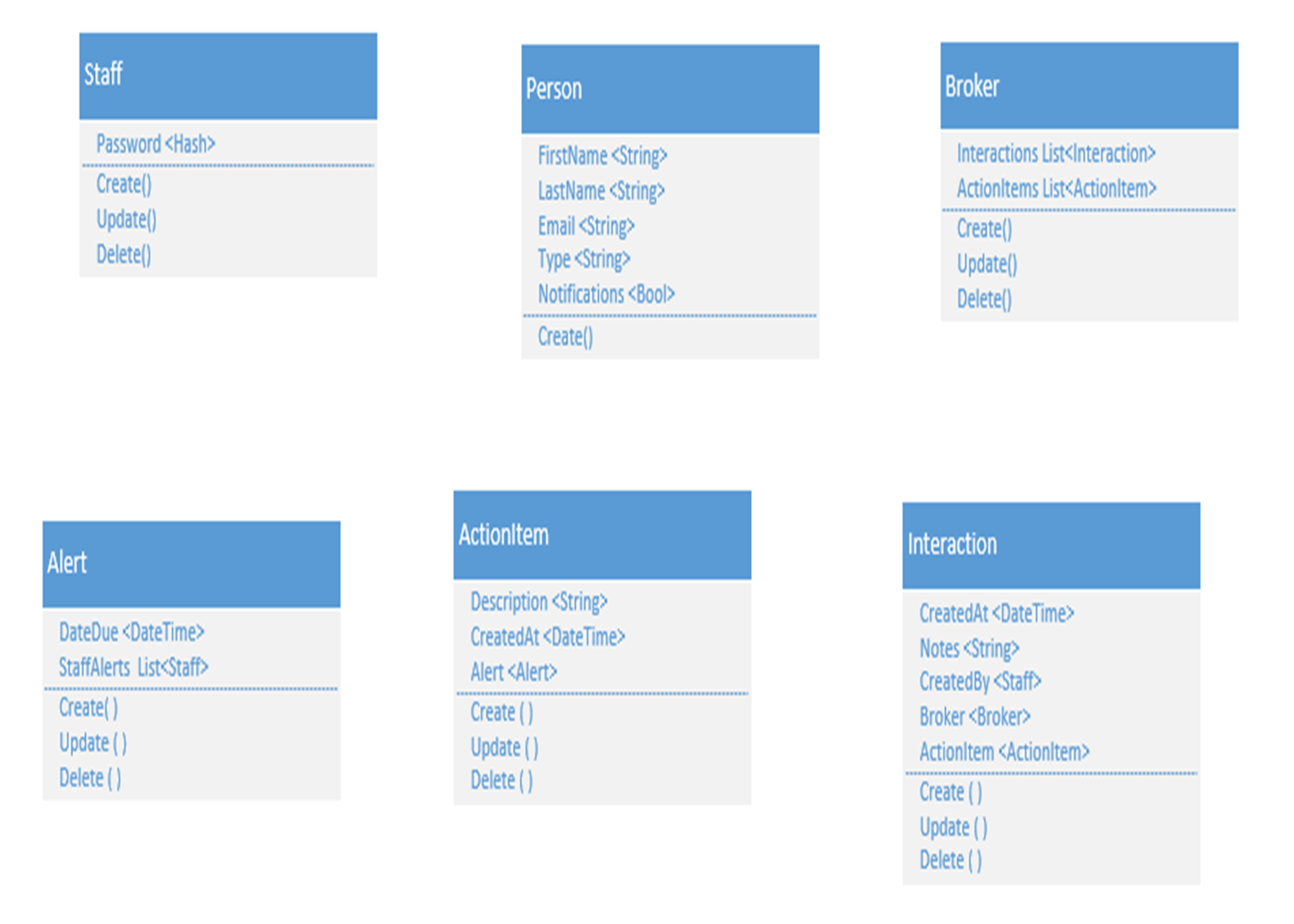
The view represents what the user sees. This includes the HTML and CSS used to house the data displayed to the user. It is the presentation layer of the program model. On a user's machine, the view would be what is displayed in a web browser for the KWFCI system. The view may also act as an interface to collect user input.

**4.1.3 Controller**

The controller controls the flow and processing of information between the model and the view. It utilizes programmed logic to dictate what information is pulled from the database via the model, which can then be passed on to the view. It can also collect information from the user via view input and implement business logic on that input; either by changing the view, modifying data through the model, or both. Calculations and processing is most often managed by the controller.

**4.2 Data Structure**

An initial framework of the data in the KWFCI database was organized into the following class diagram. As the project has progressed the foundation outlined below has been advanced to accommodate the changing needs of the system requirements, but the following diagram can provide insight into initial implementation perspectives.



**5. Application Implementation**

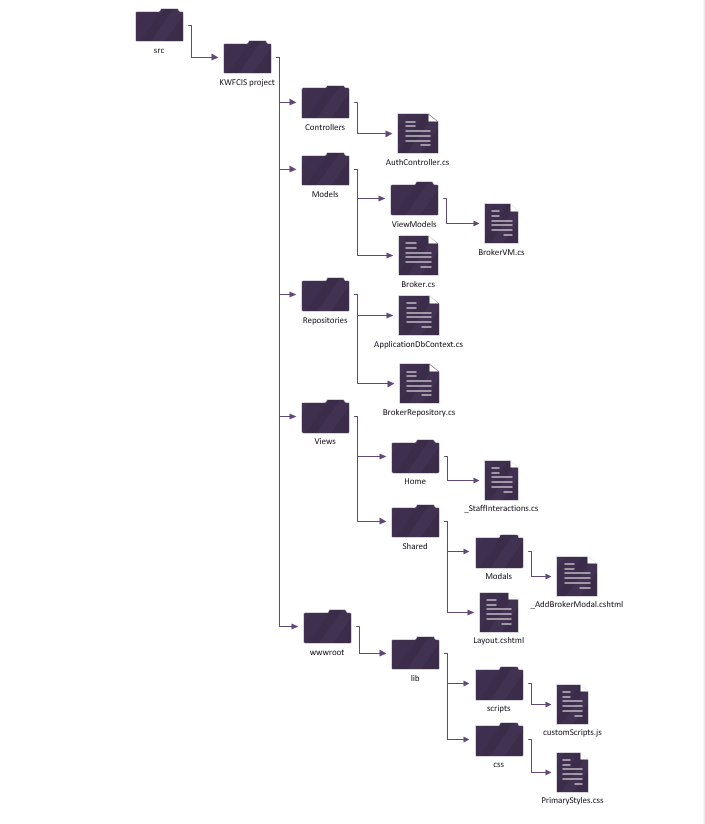
The Keller-Williams Family Check-In (KWFCI) system components are arranged utilizing *Models, Views,* and *Controllers* to separate content into a specialized system of organization.

The following figure is a high level directory diagram detailing an example of the logical structure of the KWFCI system in accordance with the MVC framework. This diagram represents the necessary hierarchy the MVC architecture requires to provide the seamless interaction of file dependencies by use of proper naming conventions.

The directory diagram displays several folders named according to MVC framework principles. The design follows proper naming conventions, utilized by Visual Studio, to allow for a separation of concerns for system content. For instance, in the following directory diagram we can see that all KWFCI controller classes are contained within the controllers folder.

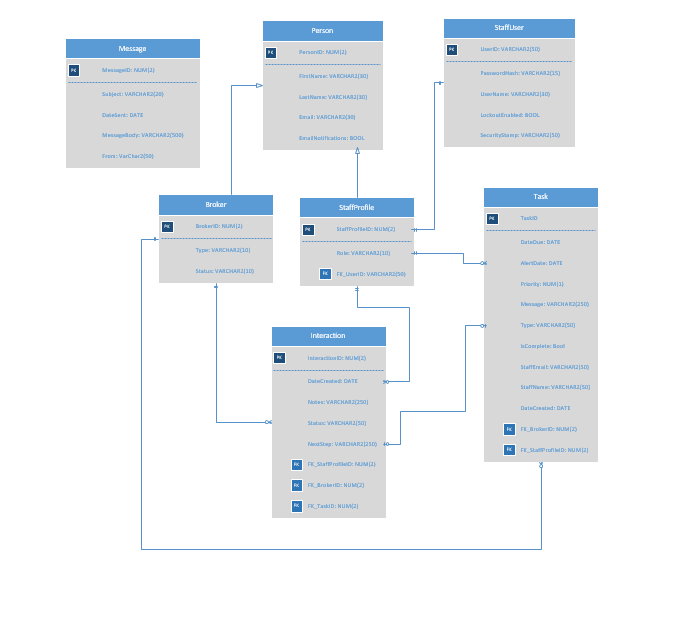
This specific diagram utilizes single files to represent the type of file that would exist in each specific section of the file structure. For example, src/KWFCIS project/Views/Shared/Modal could contain many modal files besides \_AddBrokerModal.cshtml, but for brevity only one sample representative file has been shown.

**Directory Diagram Figure**



**6. Database Architecture**

The Keller-Williams Family Check-In (KWFCI) system will utilize a Microsoft SQL Server database as a means of storing data within data tables. The following structure is current as of version 1, 6/13/2017, though it may be subject to change as an evolving structure based on client needs.



***KWFCI Entity-Relationship Diagram***

The preceding ERD graphically illustrates the KWFCI system's information entities and the relationships between those entities. The ERD provided is a conceptual and representational model of data used to represent the entity framework infrastructure.

In the KWFCI database the design team will be utilizing inheritance to reduce potential redundancy between the STAFF and BROKER tables. A table, PERSON, is created to represent common data between the two tables. The STAFF and BROKER tables now inherit attributes from the PERSON table in addition to their own independent attributes.

**7. Assumptions and Constraints**

The design of the Keller-Williams Family Check-In (KWFCI) system is designed with specific constraints being considered. During the development the implementation team has made assumptions during the process, of which several are included below.

|  |
| --- |
| Assumptions and Constraints |
| Use of the system is designed to be applied via web browsers which are considered up to date as of 6/13/2017. Separate custom software will not be necessary to utilize the KWFCI. |
| The KWFCI system will be hosted at the expense of the Keller-Williams company so that it may be accessed using HTTP connections. Keller-Williams staff will be responsible for determining which services they would like to purchase to enable hosting services. |
| Maintenance of the KWFCI system will not be the responsibility of the implementation team beyond the initial establishment of the system. Further maintenance required for the system will be the responsibility of the Keller-Williams staff to determine an expert to manage the KWFCI. |
| Business rules, and other encoded specifications, will be established by the implementation team upon initial creation of the KWFCI. After initial implementation, it will be the responsibility of the Keller-Williams staff expert to make any changes to the KWFCI. |
| Initial scope of the KWFCI is intended for the Eugene area Keller-Williams company and any further expansions of the program to include application for other regions is considered beyond the scope of the initial development of the KWFCI. |
| Regular access to Keller-Williams staff, for the purpose of communicating clarifications, are available for the implementation team or currently interested stakeholders. Response from the Keller-Williams staff will take no longer than seven days from the initial inquiry when possible. |

**Appendix A: Acronyms**

|  |  |
| --- | --- |
| AJAX | Asynchronous JavaScript and XML |
| CRUD | Create/Retrieve/Update/Delete |
| CSS | Cascading Style Sheets |
| DBMS | Database Management System |
| DRY | 'Don't Repeat Yourself' Philosophy |
| ERD | Entity Relationship Diagram |
| HTML | Hypertext Markup Language |
| IDE | Integrated Development Environment |
| KWFCI | Keller-Williams Family Check-In |
| MVC | Model/View/Controller design |
| ORM | Object Relational Mapping |

**Appendix B: Products and Tools**

The following software components will be utilized in the Keller Williams Family Check-In System. New versions of software may be released during the development of the system. The implementation of these new versions will be evaluated on an individual basis in determining if and when they will be implemented. Due to the small life cycle of this development process it is unlikely that new versions of systems will be implemented unless expressly required by the foundational tools.

|  |  |  |  |
| --- | --- | --- | --- |
| **Software/Tool** | **Version** | **Source** | **Description** |
| Microsoft Visual Studio Professional 2015 | 14.0.25431.01 | www.visualstudio.com | Primary IDE for C# development |
| Microsoft SQL Server 2014 | 12.0.2569.0 | https://www.microsoft.com/en-us/download/details.aspx?id=53168 | DBMS used with Entity Framework |
| Microsoft ASP.Net Core Tools | (Preview2) 14.1.21111.0 | https://www.asp.net/ | Cross-Platform framework for Visual Studio |
| JQuery | 3.1.1 | https://code.jquery.com/jquery-3.1.1.min.js | JavaScript library used to manipulate and traverse HTML |
| JQuery-Validate | 1.16.0 | https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.16.0/jquery.validate.min.js | One of 3 necessary components for client-side form validation |
| JQuery-Validation-  Unobtrusive | 3.2.6 | https://cdnjs.cloudflare.com/ajax/libs/jquery-validation-unobtrusive/3.2.6/jquery.validate.unobtrusive.min.js | One of 3 necessary components for client-side form validation |
| Html | 5 | Bundled with browsers and IDE | Primary language used for client-side programming |
| Entity Framework | 6.1.3 | https://www.nuget.org/packages/EntityFramework | ORM Package for Visual Studio |
| Bootstrap | 3.3.7 | www.getbootstrap.com | CSS Framework for styling client-side code |
| ASP.Net Identity packages | 2.1.0 | https://blogs.msdn.microsoft.com/webdev/2014/08/05/announcing-rtm-of-asp-net-identity-2-1-0/ | Asp.Net Core package for adding management of application users |
| XUnit | 2.2.0 | NuGet Package Manager 2.12 or later | Tools for testing application within IDE |
| Google Chrome | windows - 54.0.2840.99 | https://www.google.com/chrome/ | Primary client browser for viewing application |
| Mozilla Firefox | 51.0.1 | https://www.mozilla.org/en-US/firefox/new/?f=102 | Alternative client browser for viewing application |
| Kestrel | 1.0.1 | NuGet Package Manager | Simulated hosting environment for local development |
| .Net framework | 4.6.2 | https://www.microsoft.com/en-us/download/details.aspx?id=53345 | Windows based application development framework |
| Microsoft Azure |  | https://azure.microsoft.com | Web-bases application hosting software |
| NuGet package Manager | 3.5.0 | Bundled with Visual Studio 2015 | Content delivery network for visual studio tools and packages |
| List.js | 1.5.0 | http://listjs.com/ | Adds ajax filtering capability to UL/LI list structures |
| Bootstrap Datepicker | 1.6.4 | https://uxsolutions.github.io/bootstrap-datepicker/?markup=input&format=&weekStart=&startDate=&endDate=&startView=0&minViewMode=0&maxViewMode=4&todayBtn=false&clearBtn=false&language=en&orientation=auto&multidate=&multidateSeparator=&keyboardNavigation=on&forceParse=on#sandbox | Adds flyout calendar UI for date inputs |
| jQuery DataTables | 1.10.15 | https://datatables.net/ | Adds Filtering/Sorting/Paging capabilities to table structures |
| MailKit | 1.16.1 | https://www.nuget.org/packages/MailKit/ | Adds server email functionality for things like resetting passwords |
| Doorbell.io |  | https://doorbell.io/ | Adds Feedback functionality for our testers |

**Appendix C: Data Dictionary**

Included is a data dictionary for the Keller-Williams Family Check-In system's database:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TABLE\_NAME | TABLE COMMENTS | COLUMN\_NAME | DATA\_TYPE | DATA\_LENGTH | COLUMN COMMENTS |
| PERSON | Generic Person Information | PERSONID | NUMBER |  | PERSON\_ID\_SEQ |
| PERSON | Generic Person Information | FIRSTNAME | VARCHAR | 15 |  |
| PERSON | Generic Person Information | LASTNAME | VARCHAR | 15 |  |
| PERSON | Generic Person Information | EMAIL | VARCHAR | 50 |  |
| PERSON | Generic Person Information | ROLE | VARCHAR | 10 |  |
| PERSON | Generic Person Information | NOTIFICATIONS | BOOL | 1 |  |
| BROKER | Broker Information | BROKERID | NUMBER |  | BROKER\_ID\_SEQ |
| STAFF | Staff Information | STAFFID | NUMBER |  |  |
| STAFF | Staff Information | PASSWORD | VARCHAR | 20 |  |
| INTERACTION | Interaction with a Broker Information | INTERACTIONID | NUMBER |  | INTERATION\_ID\_SEQ |
| INTERACTION | Interaction with a Broker Information | CREATEDAT | DATETIME |  |  |
| INTERACTION | Interaction with a Broker Information | NOTES | VARCHAR | 60 |  |
| INTERACTION | Interaction with a Broker Information | STAFFID | NUMBER |  | STAFF\_ID\_SEQ |
| INTERACTION | Interaction with a Broker Information | BROKERID | NUMBER |  | BROKER\_ID\_SEQ |
| INTERACTION | Interaction with a Broker Information | ACTIONITEMID | NUMBER |  | ACTIONITEM\_ID\_SEQ |
| ALERT | Setting an Alert for an Interaction | ALERTID | NUMBER |  | ALERT\_ID\_SEQ |
| ALERT | Setting an Alert for an Interaction | DATEDUE | DATETIME |  |  |
| ALERT | Setting an Alert for an Interaction | MESSAGE | VARCHAR | 60 |  |
| ALERT | Setting an Alert for an Interaction | PRIORITYLEVEL | NUMBER | 5 |  |
| STAFFALERTS | Notifying staff of alert | STAFFID | NUMBER |  | STAFF\_ID\_SEQ |
| STAFFALERTS | Notifying staff of alert | ALERTID | NUMBER |  | ALERT\_ID\_SEQ |
| ACTIONITEM | Staff follow up item for a broker | ACTIONITEMID | NUMBER |  | ACTIONITEM\_ID\_SEQ |
| ACTIONITEM | Staff follow up item for a broker | DESCRIPTION | VARCHAR | 50 |  |
| ACTIONITEM | Staff follow up item for a broker | CREATEDAT | DATETIME |  |  |
| ACTIONITEM | Staff follow up item for a broker | BROKERID | NUMBER |  | BROKER\_ID\_SEQ |
| ACTIONITEM | Staff follow up item for a broker | ALERTID | NUMBER |  | ALERT\_ID\_SEQ |

Authors: Aaron Flager, Josh Garnick, Patrick Bailey