Tesseract Software Designs

Arch Dental Associates

Adam Franzen

COSC 6000 FA2 2016

Contents

[The Problem 4](#_Toc469449377)

[Background Information 4](#_Toc469449378)

[Problem Description 4](#_Toc469449379)

[Users / Roles 4](#_Toc469449380)

[Requirements 5](#_Toc469449381)

[Functional 5](#_Toc469449382)

[Non-Functional 7](#_Toc469449383)

[Site Information 8](#_Toc469449384)

[Site Diagram 8](#_Toc469449385)

[Features and Benefits 8](#_Toc469449386)

[UI Screen Prints 9](#_Toc469449387)

[Screen Prints / Reports 19](#_Toc469449388)

[Project Design 20](#_Toc469449389)

[ER Diagram 20](#_Toc469449390)

[Database Schema 21](#_Toc469449391)

[Data Dictionary 22](#_Toc469449392)

[Appointment Table 22](#_Toc469449393)

[Appointment Service Table 23](#_Toc469449394)

[Clinic Table 24](#_Toc469449395)

[Entity Table 25](#_Toc469449396)

[Household Table 26](#_Toc469449397)

[Household Person Table 27](#_Toc469449398)

[Provider Table 28](#_Toc469449399)

[Provider Qualification Table 29](#_Toc469449400)

[Relationship Table 30](#_Toc469449401)

[Roles Table 31](#_Toc469449402)

[Room Table 32](#_Toc469449403)

[Service Table 33](#_Toc469449404)

[Service Clinic Table 34](#_Toc469449405)

[Users Table 35](#_Toc469449406)

[Use Cases (Essential Form) 36](#_Toc469449407)

[Use Cases (Essential Form) 37](#_Toc469449408)

[Use Cases (Expanded Form) 43](#_Toc469449409)

[Domain / Concept Model (UML) 48](#_Toc469449410)

[System Sequence Diagrams (UML) 49](#_Toc469449411)

[Class Diagrams (UML) 51](#_Toc469449412)

[Deployment Diagram (UML) 53](#_Toc469449413)

[Glossary 54](#_Toc469449414)

[Conclusion 55](#_Toc469449415)

[Program Implementation 56](#_Toc469449416)

[Program Structure 56](#_Toc469449417)

[Program Structure 58](#_Toc469449418)

[Presentation Layer 58](#_Toc469449419)

[Business Layer 175](#_Toc469449420)

[Business Objects 194](#_Toc469449421)

[Communication Layer 213](#_Toc469449422)

[Data Layer 234](#_Toc469449423)

[Utility Classes 286](#_Toc469449424)

[SQL Create/Drop 292](#_Toc469449425)

[Project Management Section 299](#_Toc469449426)

[Weekly Status Reports / Activities & Time 299](#_Toc469449427)

[End of Week 1 299](#_Toc469449428)

[End of Week 2 299](#_Toc469449429)

[End of Week 3 300](#_Toc469449430)

[Total Time Spent 303](#_Toc469449431)

[Project Management 303](#_Toc469449432)

[Difficulties Encountered 303](#_Toc469449433)

[Technical Reports Used To Solve Difficulties 303](#_Toc469449434)

[Other Technical Information 303](#_Toc469449435)

# The Problem

## Background Information

Arch Dental Associates (from now on referred to as “Arch”) is a chain of dental practices owned by a small group of partners. They have been in business for the past 25 years and are a recognized and accomplished dental establishment within the communities in which they reside. Although they do have multiple practices, they still consider themselves to be a small operation and try to take care of their employees just as though they were a family-run business.

## Problem Description

Arch has approached us with a request to help them to modernize their business processes, and to ease and simplify the use of day-to-day operations. They are looking to replace an age-old paper system of operating the business. Since they have multiple locations and multiple possible administrators, we have concluded that the best design for them would be a web-based application. This will help them to coordinate their Dental Practice and make their business more efficient from any location. It will be of upmost importance that the user interface for this application be highly user friendly and intuitive to use given that many of the users will have limited computer experience.

## Users / Roles

* User
  + These are the patients who receive services
  + Could be a primary account holder
  + Could be the primary account holder’s dependents
* Office Staff
  + An office worker or provider/dentist/hygienist
* Administrator
  + Main owner(s) of practice
  + Would typically be the owner of the practice but could also be a trusted user within the company

# Requirements

## Functional

* User (Patients)
  + Primary account holder
  + View/edit household account info
  + View/cancel appointments
  + Payments
  + Services
  + Designate primary insurance holder
  + Add/edit dependents
* Office Staff
  + An office worker or provider/dentist/hygienist
  + Create/view/update individual client account info
  + Create/view/update appointments
  + Process payments
* Administrator
  + Main owner(s) of practice
  + Would typically be the owner of the practice but could also be a trusted user within the company
  + Create/view/update any screen
  + Create/view/update office staff and providers
  + Create/view/update clinics
  + Create/view/update rooms
  + Create/view/update services
  + Create/view/update qualifications
  + Create/view/update accepted insurance

Arch would like to move to a modern/computerized system to allow their patients access to some portion of their account from home, to ensure rooms in each clinic are adequately utilized, and not double-booked, to ensure the time of each provider is adequately utilized, and not double-booked, and lastly, to provide a paperless (or at least significantly less paper) method for day-to-day business and accounting.

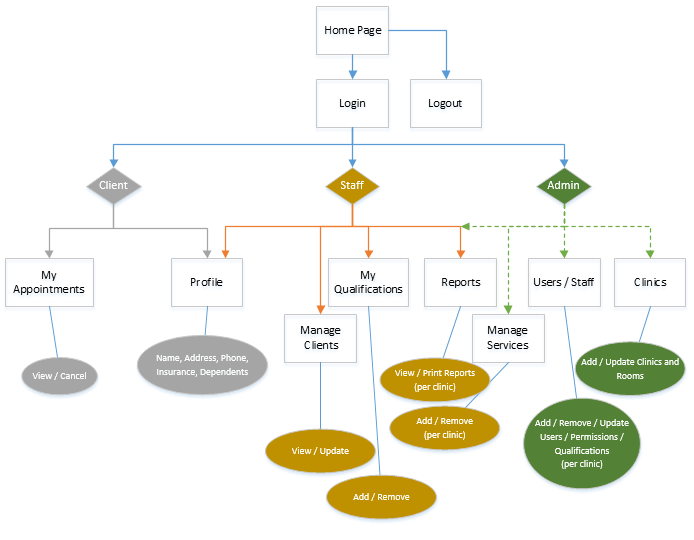
* The system shall support the following users: Client, Office, Admin
  + Client
    - The system shall designate a Primary account holder (a head of household)
    - The system shall allow the Client to view/edit household info such as:
      * Designate the household insurance company
      * Dependent information and relationship to primary
      * Birthdate of each person in household
      * Appointments coming / past and details including
        + Services rendered
        + Price
        + Provider Name
    - The system shall provide a user with a unique account number
  + Office
    - Office Staff, Doctors, Hygienists
    - The system shall allow all staff to create / update any Client account info
    - The system shall allow all staff to create / update appointments
    - The system
  + Admin
    - Main owners/partners of practice (could also be Doctors)
    - The system shall allow Admin to create/view/update any screen
    - The system shall allow Admin to create/view/update office staff and clients
    - The system shall allow Admin to create/view/update clinics
    - The system shall allow Admin to create/view/update rooms in a clinic
    - The system shall allow Admin to create/view/update services offered by a clinic
    - The system shall allow Admin to create/view/update designate which office users are providers
* The system shall provide a username / password to every user and allow user the ability to change them
* The system shall provide a unique numeric ID to each user
* The system shall obfuscate all user passwords
* The system shall track contact information for all users (name, address, phone)
* The system shall allow all staff to make appointments.
* The system shall keep track of all services (and costs for each service) a clinic provides.
  + A provider does not need to be qualified in order to perform a service the clinic offers
  + Designate a standard rate for each service
  + Designate a standard length of time to complete each service
* The system shall allow for more than one dentist and/or hygienist to provide services
* The system shall record services each provider is certified for
* The system shall allow office users and administrators to make appointments for services on a specific time/date
* The system shall allow clients to view/update their profile, and view/cancel appointments
* The system shall allow a patient to receive services from multiple providers during a visit
* The system shall track if an appointment is kept/missed/canceled
* The system shall, for each appointment, reserve a room for each service provided, note individual service lengths
* The system shall generate a single bill for an entire household
* The system shall allow Office user creating an appointment to alter standard pricing with billing reflecting both the standard and actual rates
  + The actual rate should never be higher than the standard rate
* The system shall provide the ability to generate an overall bill for all services outstanding, or provide an itemized bill for an individual appointment
  + Should contain the date of service, list of services, name of provider, and costs
* The system shall generate a bill on a monthly basis for each household
* The system shall generate a report containing all households, names, addresses, phones, person ID, dependents and relationships
* The system shall generate a report showing insurance coverage for all households including household ID, household primary account holder name, insurance company ID and name.
* The system shall generate a report containing all patients in order by person ID, name, birthdate, and also contain their insurance company and policy number
* The system shall generate a report show itemized billings for all households with household ID, primary account holder name, person ID, patient name, service, and cost ordered by primary account holder name, patient name, and billing date
* The system shall generate a report showing the total cost of all services received for each household
* The system shall generate a report showing each provider and services qualified to render
* The system shall generate a report all future appointments sorted by patient name, appointment date/time, length of service, phone – grouped by each patient
* The system shall generate a report list all services performed by each provider in order by provider name and include the service ID, service description, and cost
* The system shall generate a report, that for a given date, lists the total amount of services each provider rendered – ordered by provider name

## Non-Functional

* The system shall work only via web browser
* The system shall be optimized for the Google Chrome web browser
* The system shall have a Web API back end written in C# to support .NET 4.0
* The system shall have an HTML5 front end written in Angular JS
* The system shall have a modern/organized/consistent usability
* The system shall take into account multiple users working at the same time in order to best optimize for performance
* The system shall leverage an Oracle database for data storage
* The system shall support Microsoft IIS

# Site Information

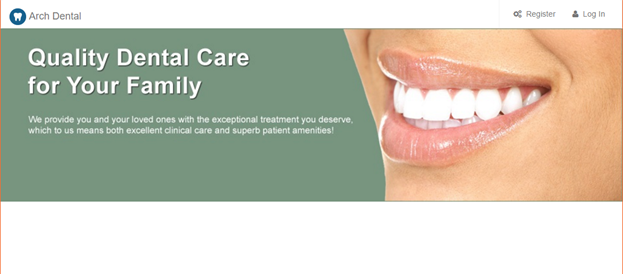
## Site Diagram

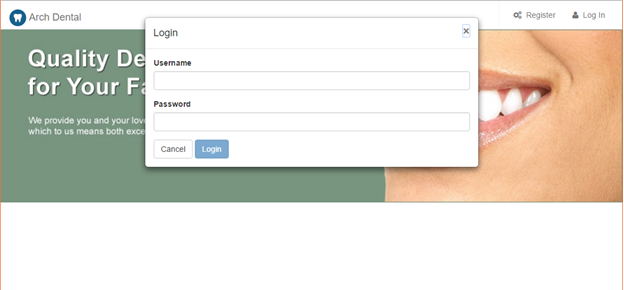


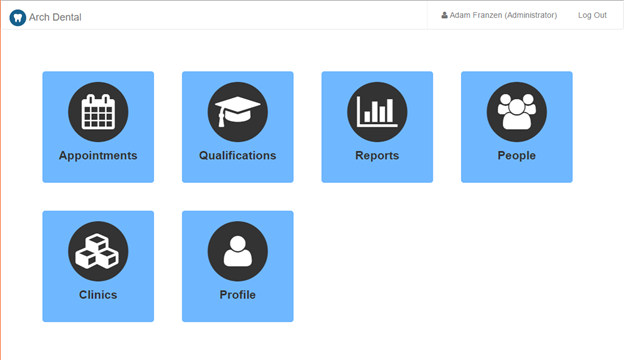
## Features and Benefits

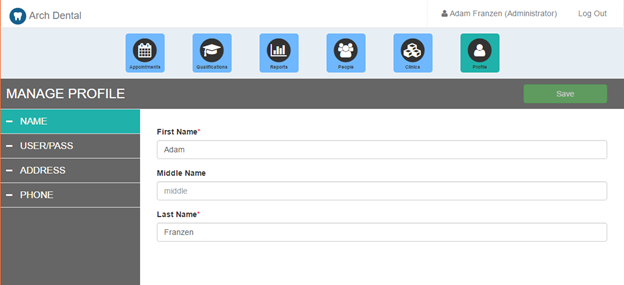
|  |  |
| --- | --- |
| ***Feature*** | ***Benefit*** |
| Appointments accessible from any computer | * Makes doing business easier and more intuitive, more organized, constantly up-to-date, and free of paper * Clients can view/cancel appointments online |
| Payments tracked by computer | * Time better optimized. Location of records and balancing books happens in a flash |
| Real-world, up-to-date data | * Room bookings always accurate * Provider bookings always accurate and fully optimized |
| Clients can manage their data online | * Lighter workload on employees when clients use online services   + Lower call instances for payments and billing questions   + Lower call instances for appointment time/date/service/cancellation questions |

## UI Screen Prints



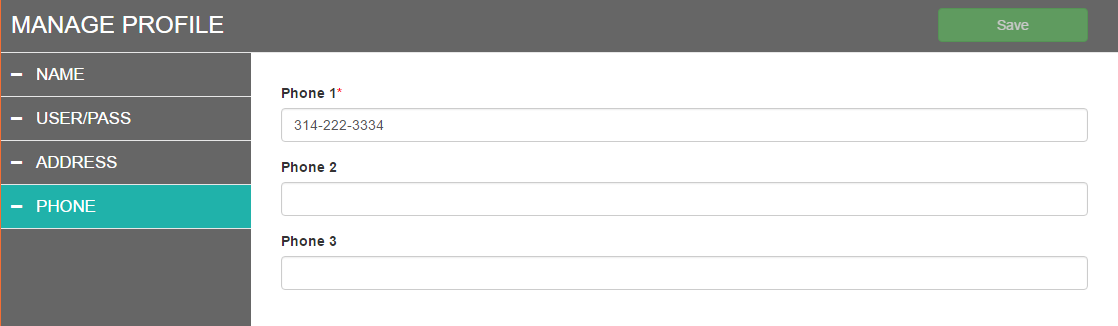


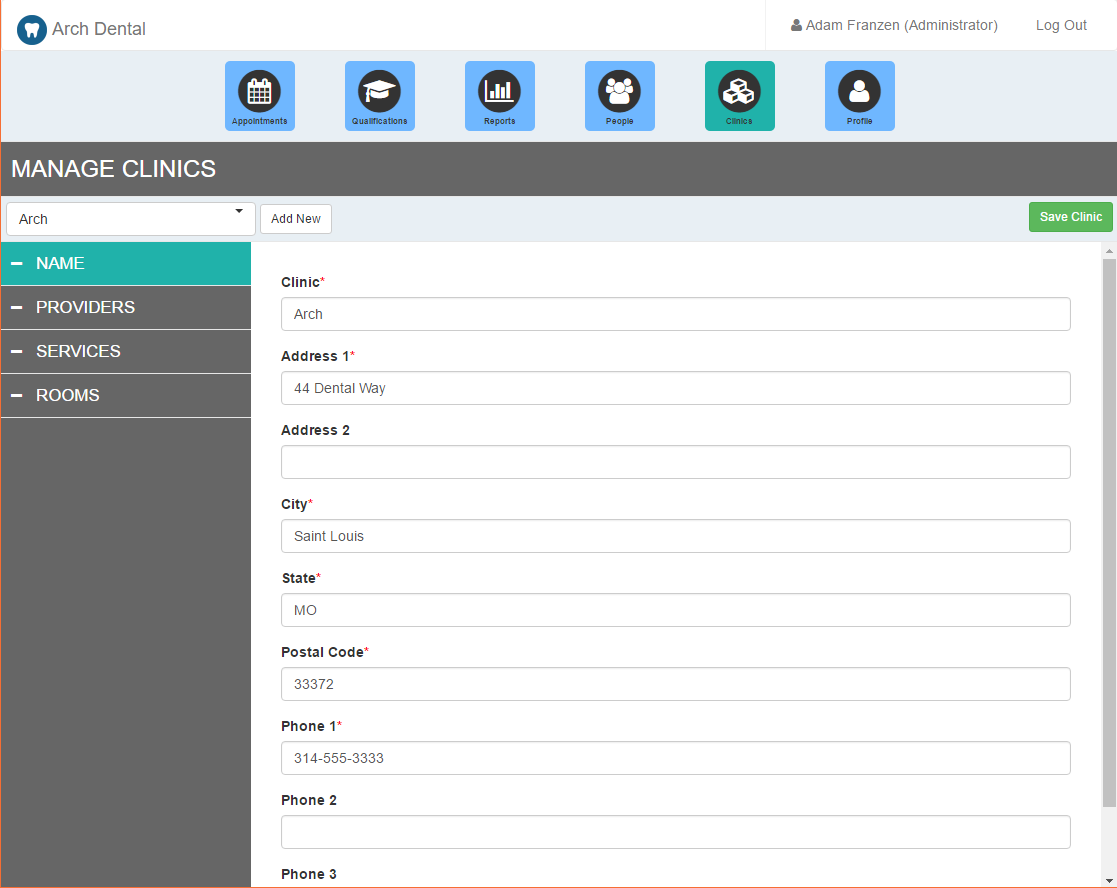


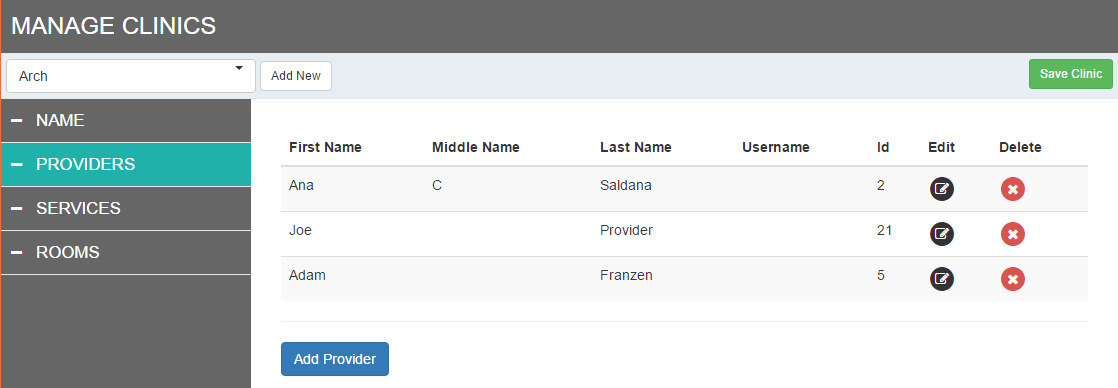


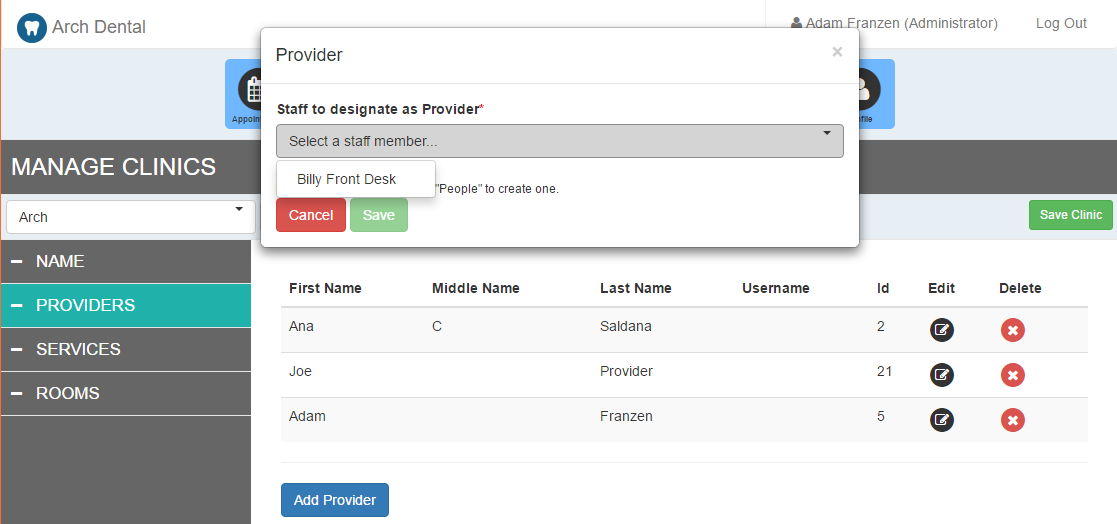


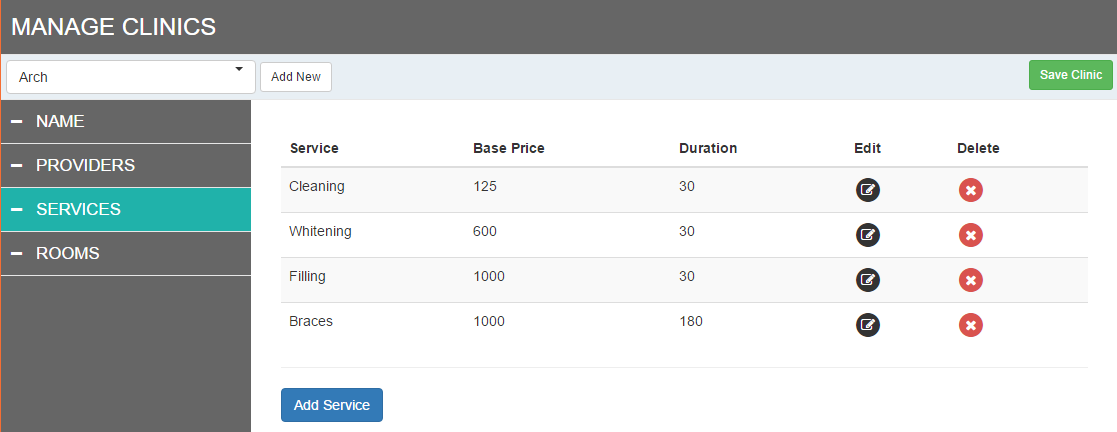


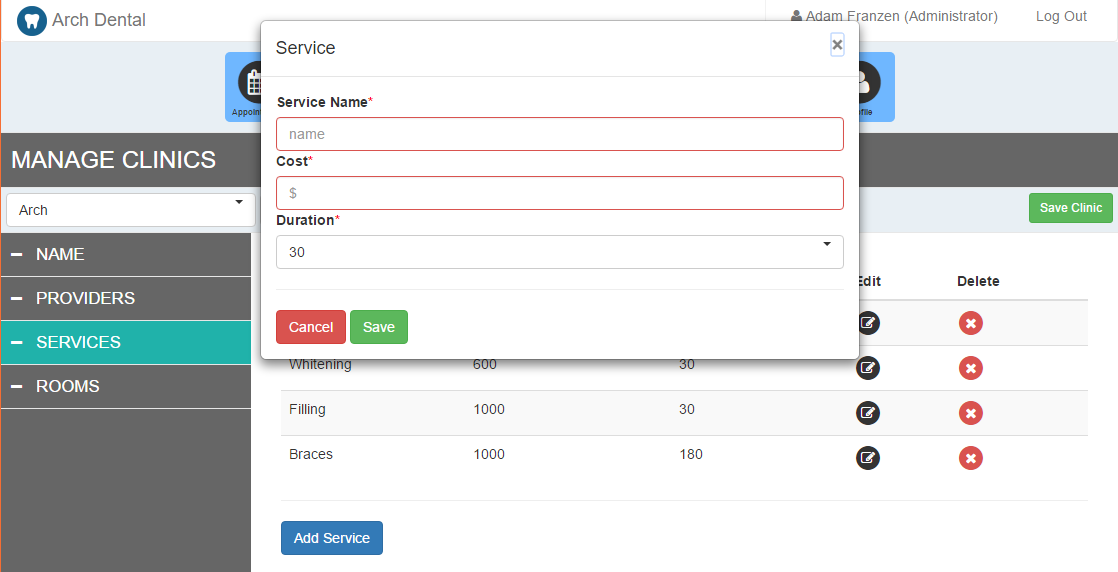


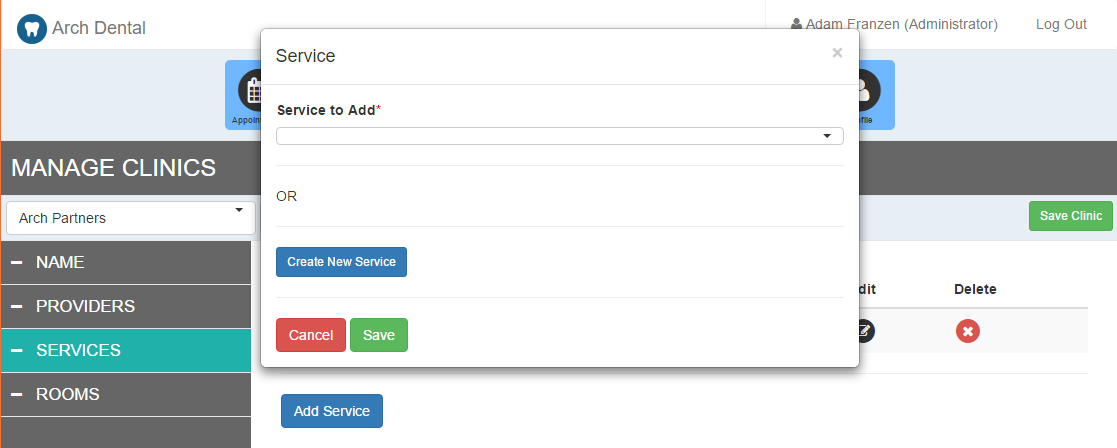


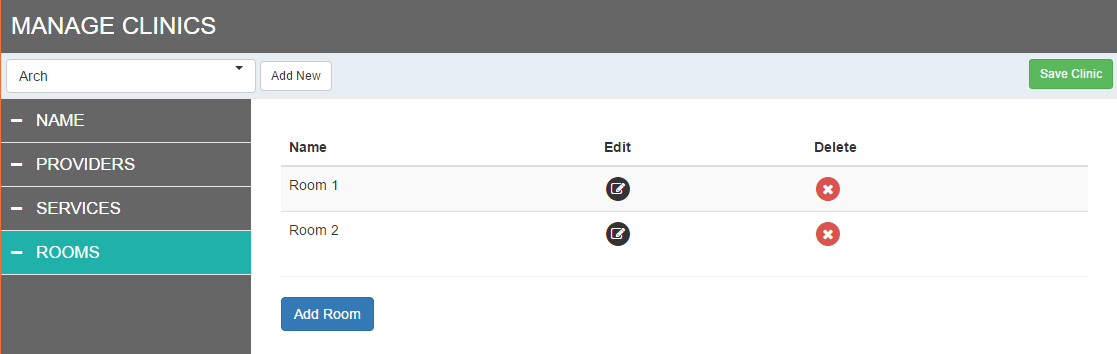


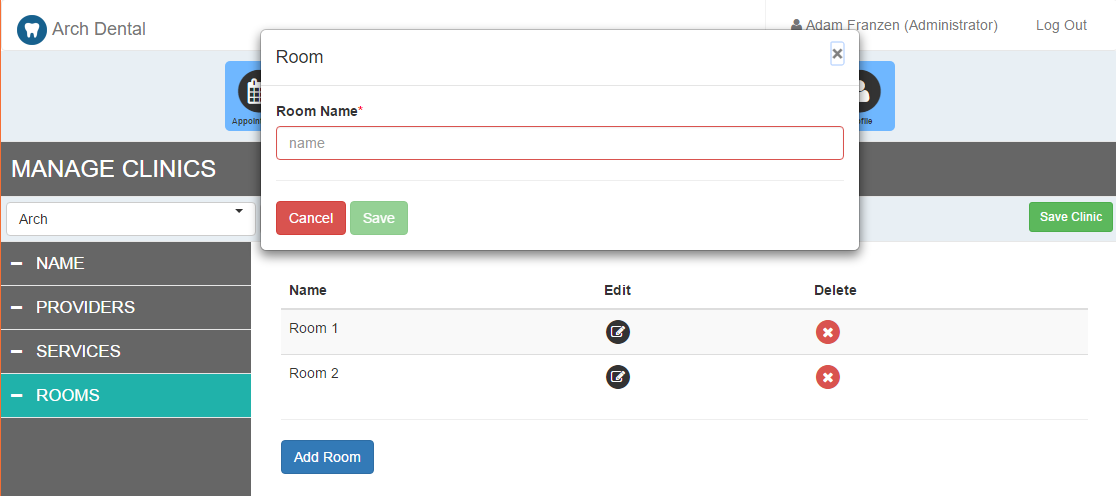


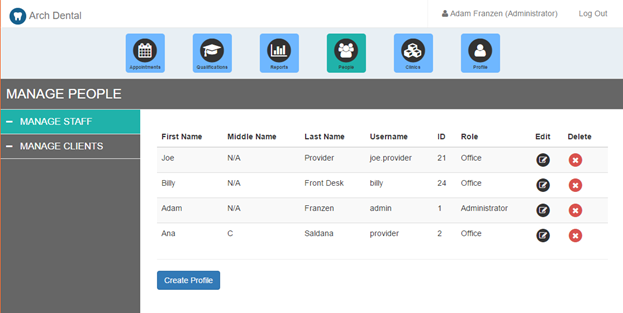


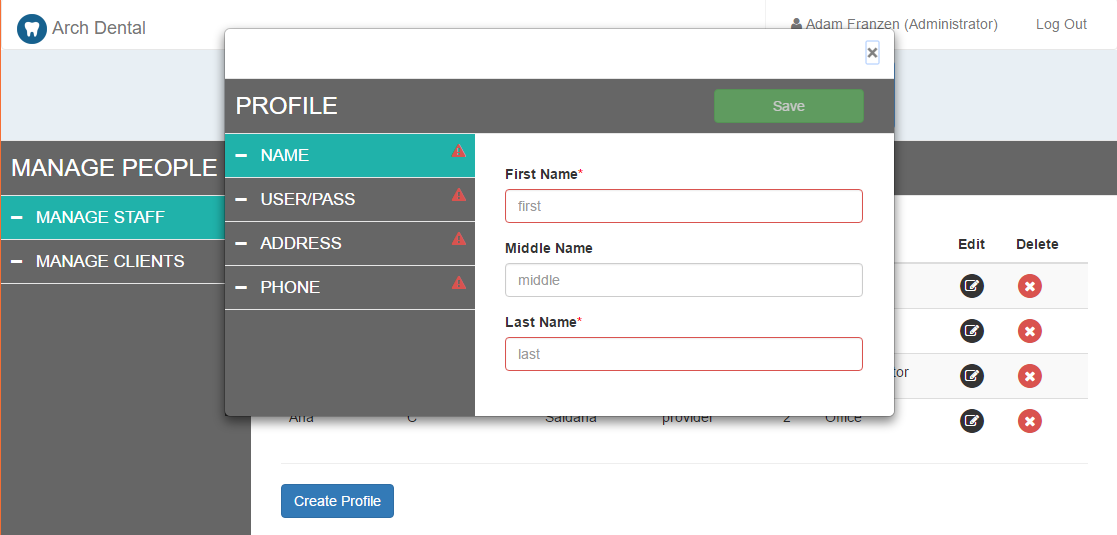


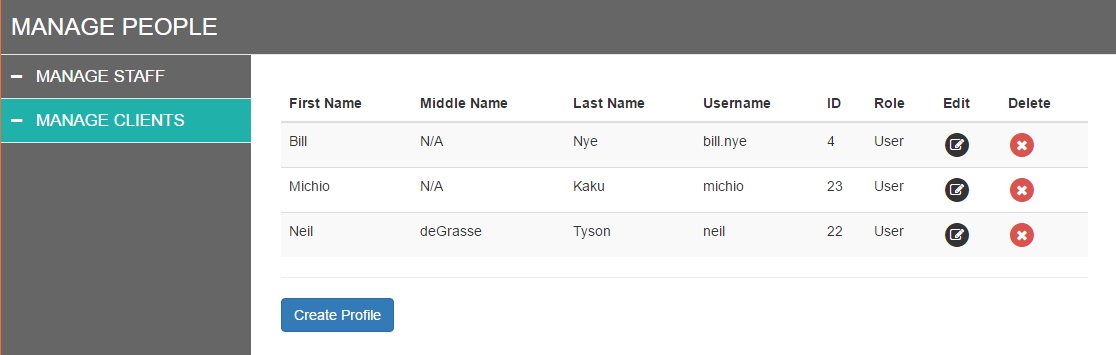


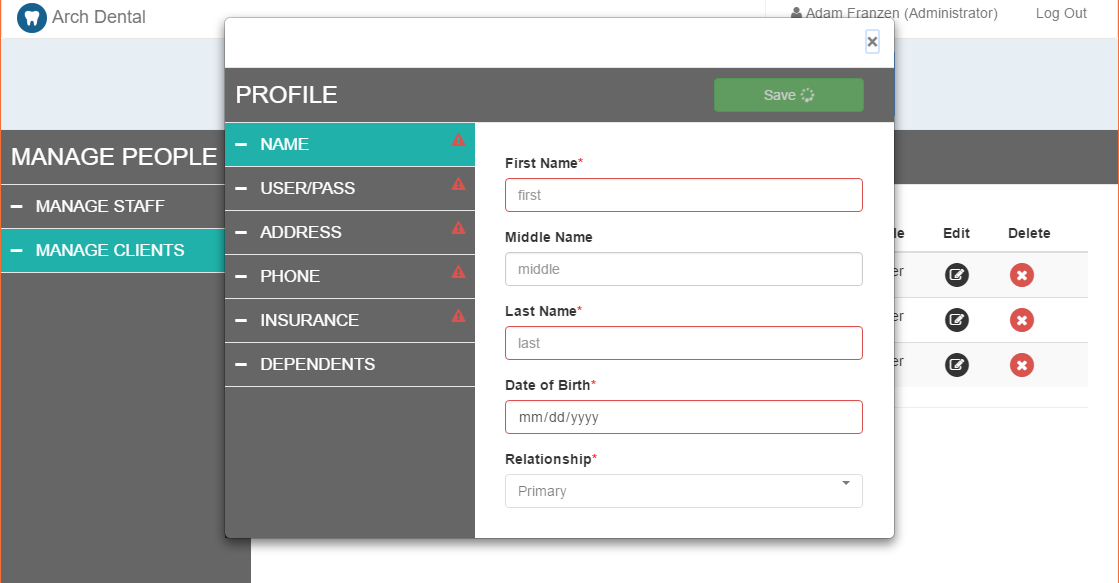


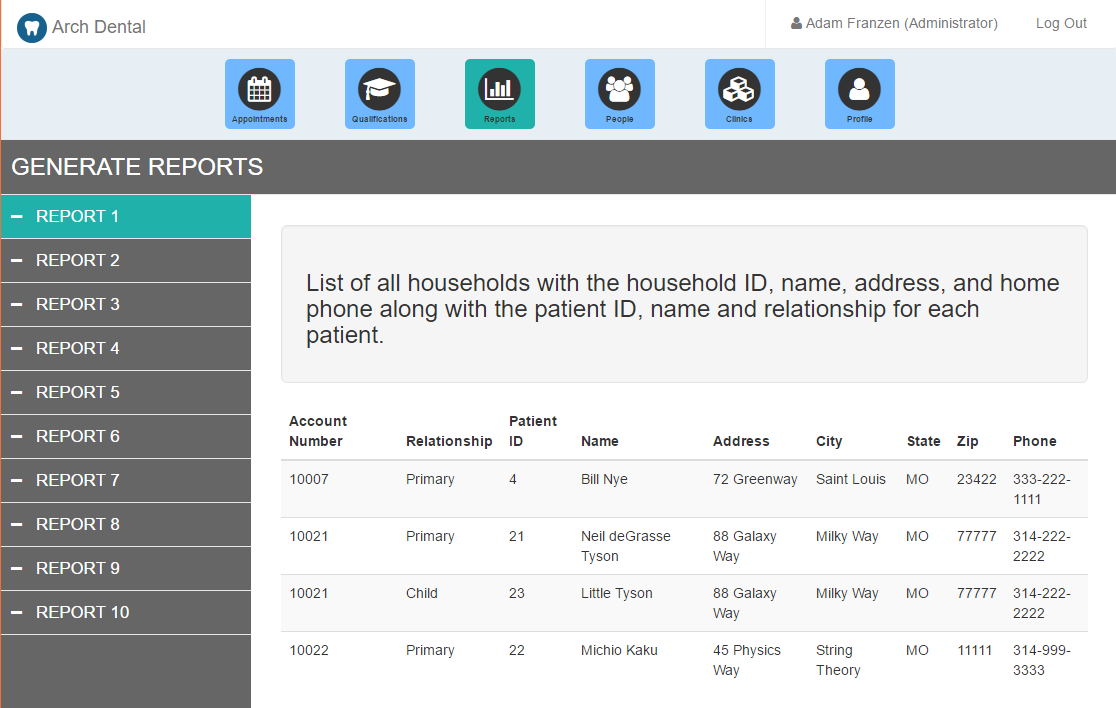


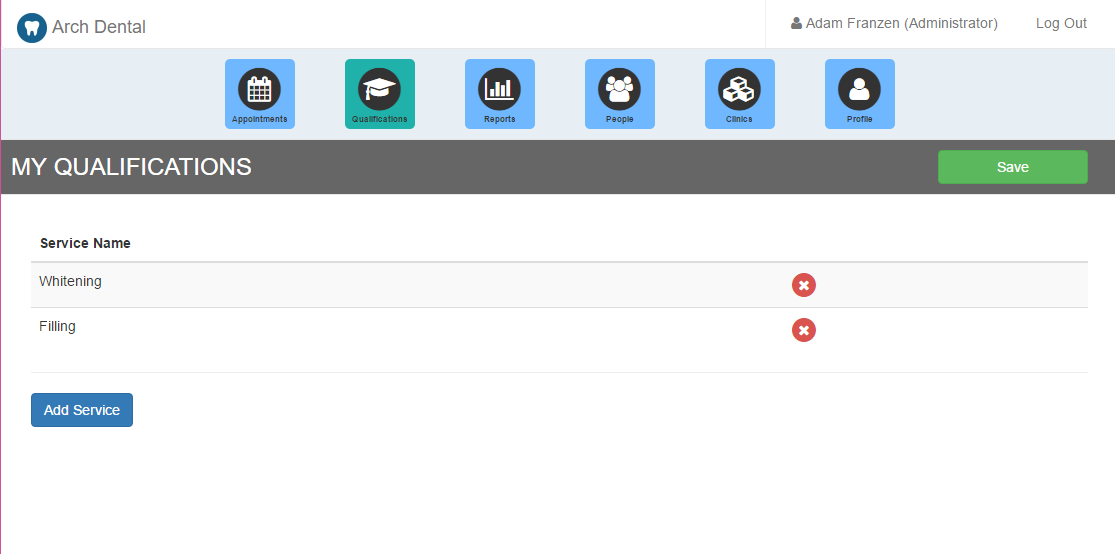


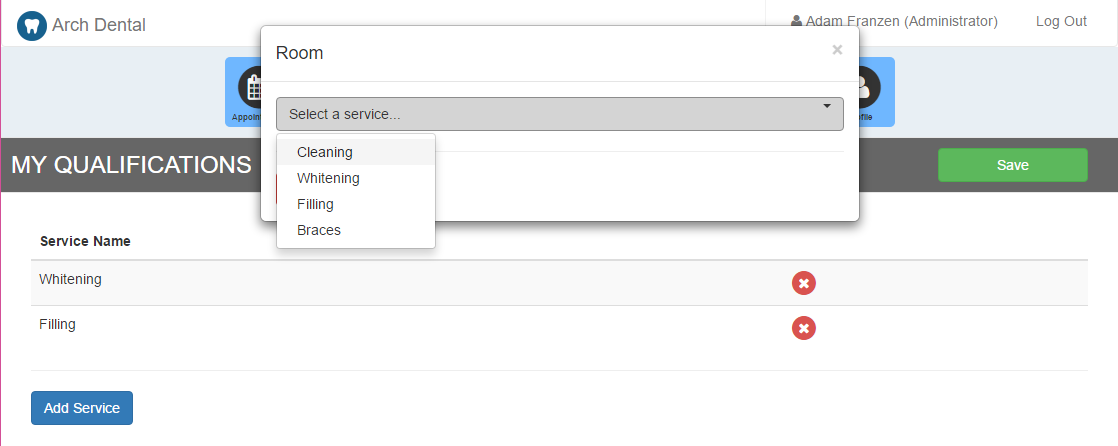


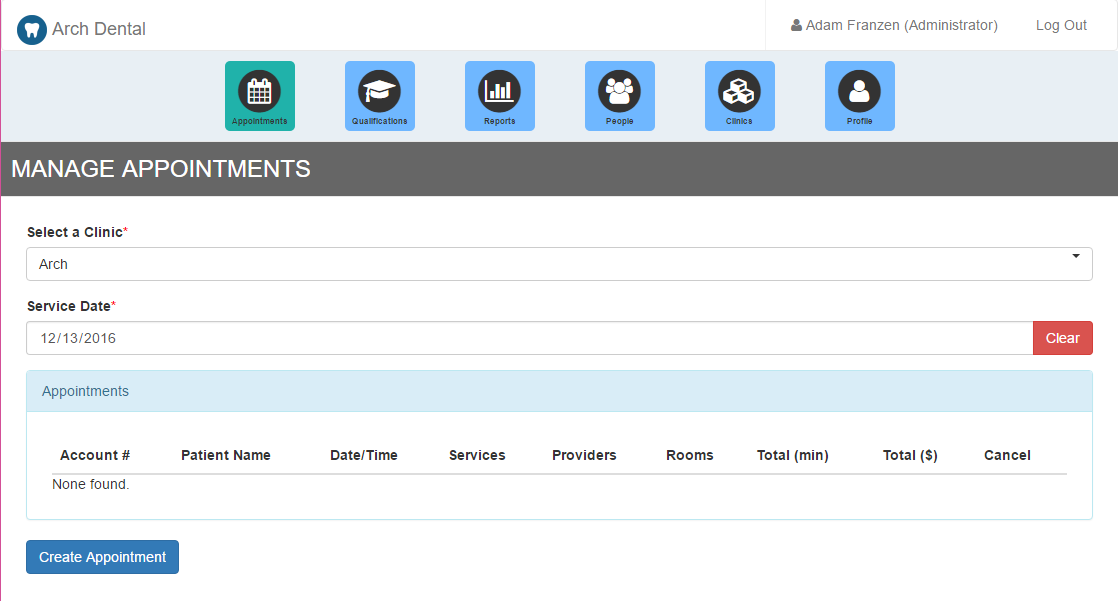


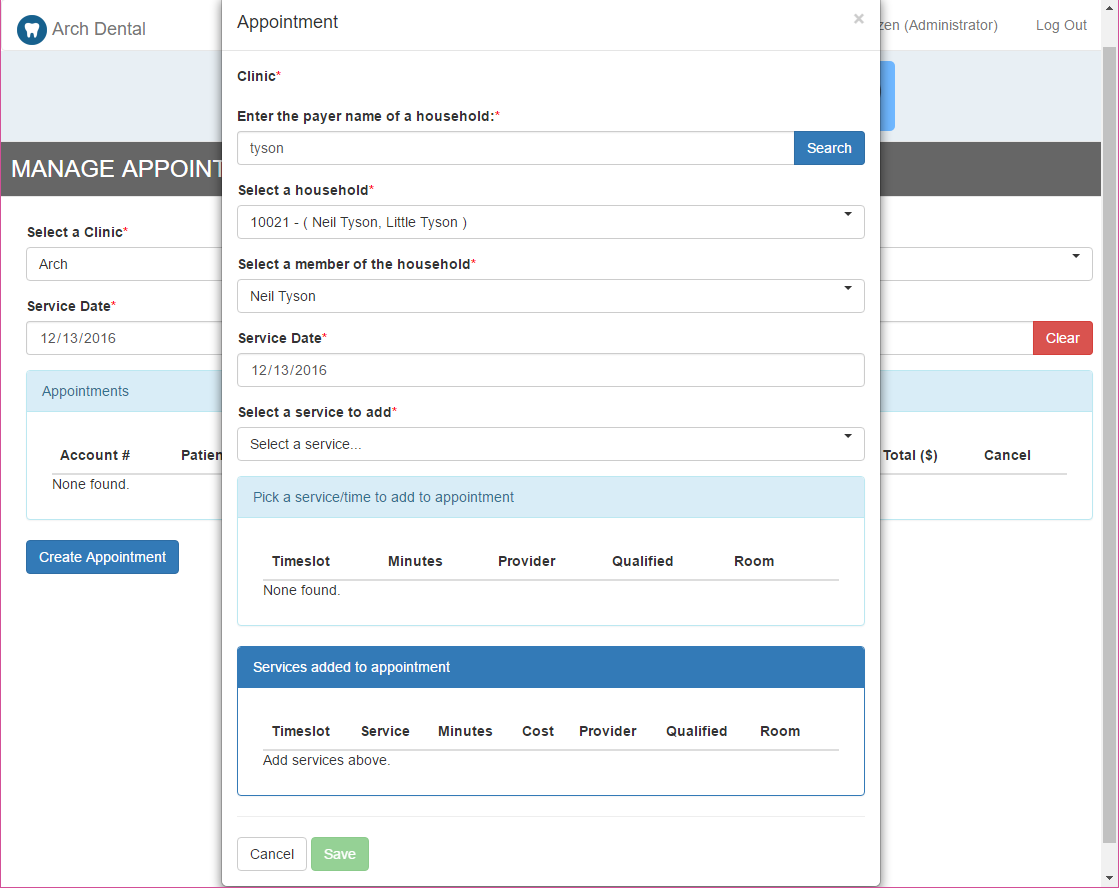










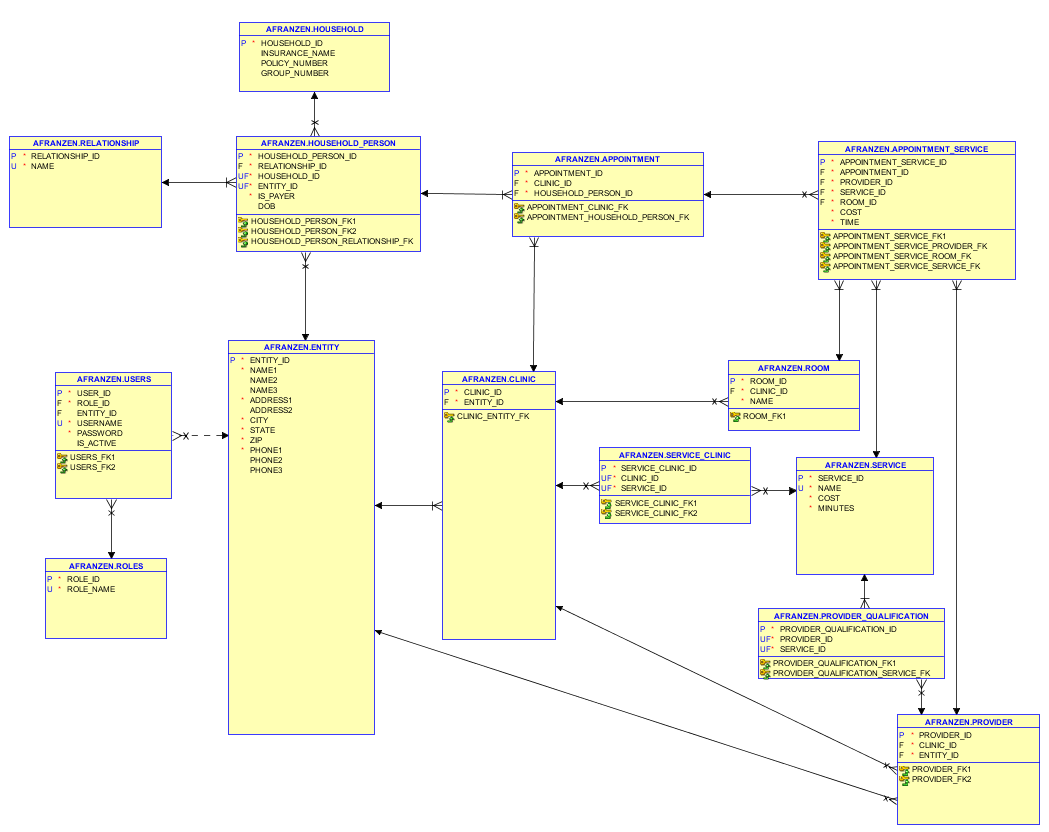


## Screen Prints / Reports

TODO

# Project Design

## ER Diagram



## Database Schema



## Data Dictionary

### Appointment Table

Table Description

This table contains data about the appointments at a Clinic.

Other Tables Referenced

Clinic, Household Person

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| APPOINTMENT\_ID | Appointment  Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| CLINIC\_ID | Clinic Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Clinic  Delete – Cascades |
| HOUSEHOLD\_PERSON\_ID | Person being served Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Household Person  Delete – Cascades |

### Appointment Service Table

Table Description

This table contains data about the services of an appointment. When they start, how long they will take, who they are performed by, how much they cost.

Other Tables Referenced

Appointment, Room, Service, Provider

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| APPOINTMENT\_SERVICE\_ID | Appointment  Service Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| APPOINTMENT\_ID | Appointment  Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| PROVIDER\_ID | Provider Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Clinic  Delete – Cascades |
| SERVICE\_ID | Service Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Household Person  Delete – Cascades |
| ROOM\_ID | Room Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Room  Delete – Cascades |
| COST | Price of Service | NUMBER | 6, 2 | 9(6,2) | 0.00 – 9999.00 | YES |  |  |
| TIME | Start Time | DATE |  | >= 08:00  <= 06:00 | 1-9999 | YES |  |  |

### Clinic Table

Table Description

This table contains data to identify a unique Clinic.

Other Tables Referenced

Entity

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| CLINIC\_ID | Clinic Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| ENTITY\_ID | Entity Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Entity  Delete – Cascades |

### Entity Table

Table Description

This table contains name and contact information about a person in a household, a user in the system, an employee/provider of a clinic, or a clinic.

Other Tables Referenced

None

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| ENTITY\_ID | Entity  Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| NAME1 | Name | NUMBER | 50 | X(50) |  | YES |  |  |
| NAME2 | Name | VARCHAR2 | 50 | X(50) |  |  |  |  |
| NAME3 | Name | VARCHAR2 | 50 | X(50) |  |  |  |  |
| ADDRESS1 | Address | VARCHAR2 | 50 | X(50) |  | YES |  |  |
| ADDRESS2 | Address | VARCHAR2 | 50 | X(50) |  |  |  |  |
| CITY | City | VARCHAR2 | 50 | X(50) |  | YES |  |  |
| STATE | State | VARCHAR2 | 2 | X(2) |  | YES |  |  |
| ZIP | Zip | VARCHAR2 | 10 | X(10) |  | YES |  |  |
| PHONE1 | Phone | VARCHAR2 | 15 | X(15) |  | YES |  |  |
| PHONE2 | Phone | VARCHAR2 | 15 | X(15) |  |  |  |  |
| PHONE3 | Phone | VARCHAR2 | 15 | X(15) |  |  |  |  |

Other Notes

Zip and Phone are intentionally any type of character to allow for hyphens, extensions, or other possible needed special characters.

### Household Table

Table Description

This table contains a unique key for each household the system and information about that home’s Insurance.

Other Tables Referenced

Insurance

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| HOUSEHOLD\_ID | Household Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| INSURANCE\_NAME | Insurance Company Name | VARCHAR2 | 50 | X(50) |  | YES |  |  |
| POLICY\_NUMBER | Insurance Policy Number | VARCHAR2 | 50 | X(50) |  | YES |  |  |
| GROUP\_NUMBER | Insurance Group Number | VARCHAR2 | 50 | X(50) |  | YES |  |  |

### Household Person Table

Table Description

This table contains data that describes a person who resides in (or is associated with) a household.

Other Tables Referenced

Relationship, Insurance, Entity

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| HOUSEHOLD\_PERSON\_ID | Household Person  Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| RELATIONSHIP\_ID | Relationship Name | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Relationship  Delete – Cascades |
| HOUSEHOLD\_ID | Household Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Household  Delete – Cascades |
| ENTITY\_ID | Entity Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Entity  Delete – Cascades |
| IS\_PAYER | Flag for whether primary account holder | CHAR | 1 | A(1) | ‘Y’ or ‘N’ | YES |  |  |
| DOB | Date of Birth | DATE |  | >= 00:01  <= 23:59 | 1-9999 | YES |  |  |

### Provider Table

Table Description

This table contains data about employees of a who are also providers.

Other Tables Referenced

Clinic, Entity

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| PROVIDER\_ID | Provider  Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| CLINIC\_ID | Clinic Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Clinic  Delete – Cascades |
| ENTITY\_ID | Entity Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Entity  Delete – Cascades |

### Provider Qualification Table

Table Description

This table contains data about the services each provider is qualified to perform.

Other Tables Referenced

Provider, Service

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| PROVIDER\_QUALIFICATION\_ID | Provider Qualification  Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| PROVIDER\_ID | Provider Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Provider  Delete – Cascades |
| SERVICE\_ID | Service Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Service  Delete – Cascades |

### Relationship Table

Table Description

This table contains a list of possible relationships a person can have to the primary (payer) account holder.

Other Tables Referenced

None

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| RELATIONSHIP\_ID | Relationship Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| NAME | Relationship Title | VARCHAR2 | 50 | X(50) |  | YES |  |  |

### Roles Table

Table Description

Contains system roles a user could have

Other Tables Referenced

None

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| ROLE\_ID | Role Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| ROLE\_NAME | Name of Role | VARCHAR2 | 50 | X(50) |  | YES |  |  |

### Room Table

Table Description

This table contains data about rooms in a Clinic.

Other Tables Referenced

Clinic

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| ROOM\_ID | Room  Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| CLINIC\_ID | Clinic  Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Clinic  Delete – Cascades |
| NAME | Clinic Name | VARCHAR2 | 50 | X(50) |  | YES |  |  |

### Service Table

Table Description

This table contains data services available in a Clinic.

Other Tables Referenced

Clinic

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| SERVICE\_ID | Service  Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| NAME | Service Name | VARCHAR2 | 50 | X(50) |  | YES |  |  |
| COST | Service Cost | NUMBER | 6, 2 | 9(6,2) | 0.00 – 9999.00 | YES |  |  |
| MINUTES | Time to Complete | NUMBER | 35 | 9999 | 1-9999 | YES |  |  |

### 

### Service Clinic Table

Table Description

This table contains data services available in a Clinic.

Other Tables Referenced

Clinic, Service

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| SERVICE\_CLINIC\_ID | Primary Key | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| SERVICE\_ID | Service  Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Service  Delete – Cascades |
| CLINIC\_ID | Clinic Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Clinic  Delete – Cascades |

### Users Table

Table Description

Contains user information, the role they possess, a link to their identifying/contact info, username, and password.

Other Tables Referenced

Role, Entity

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ATTRIBUTE NAME | CONTENTS | TYPE | LENGTH | FORMAT | RANGE | REQ’D | KEY | REFERENCED TABLE |
| USER\_ID | User Number | NUMBER | 35 | 9999 | 1-9999 | YES | PK |  |
| ROLE\_ID | Role Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Role  Delete – Cascades |
| ENTITY\_ID | Entity Number | NUMBER | 35 | 9999 | 1-9999 | YES | FK | Entity  Delete – Cascades |
| USERNAME | Username | VARCHAR2 | 50 | X(50) |  | YES |  |  |
| PASSWORD | Password | VARCHAR2 | 500 | X(500) |  | YES |  |  |

Use Cases (Essential Form)



## Use Cases (Essential Form)

|  |  |
| --- | --- |
| **Use Case Name:** | Start IIS and load app for first time – no users present yet |
| **Actors:** | Company Owner |
| **Scenario:** | Company just turned software on for the first time. How do they login if there is no admin to assign elevated roles etc? System checks to see if any users exist. If not, we assume the first user to login is the owner. Login button hidden and Register button shown. Whatever login is created is made an Admin by default. |

|  |  |
| --- | --- |
| **Use Case Name:** | Login |
| **Actors:** | Anyone |
| **Scenario:** | Someone comes to the home page and they are met with info about the company, and options to login or register. User clicks Login and is taken to Landing Page and is given tiles/links to where they can go in the system based upon their Role. |

|  |  |
| --- | --- |
| **Use Case Name:** | Logged In. At Landing Page |
| **Actors:** | User |
| **Scenario:** | The following Tiles/Links are displayed: Appointments, Billing, and Profile. User redirected to Landing Page if tries to go to a URL that is not permitted to based upon Role. |
| **Notes:** | Billing has been removed due to time constraints |

|  |  |
| --- | --- |
| **Use Case Name:** | Logged In. At Landing Page |
| **Actors:** | Office |
| **Scenario:** | The following Tiles/Links are displayed: Appointments, Billing, Qualifications, Reports, People, and Profile. User redirected to Landing Page if tries to go to a URL that is not permitted to based upon Role. |
| **Notes:** | Billing has been removed due to time constraints |

|  |  |
| --- | --- |
| **Use Case Name:** | Logged In. At Landing Page |
| **Actors:** | Admin |
| **Scenario:** | All available Tiles/Links are displayed. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Profile Page |
| **Actors:** | User |
| **Scenario:** | All tabs are shown (Name, User/Pass, Address, Phone) to edit profile – with the addition of the Dependents and Insurance tabs and ability to collect Date of Birth. Only when editing a User profile should we see these two additional tabs. Can modify any field except for Role (User) and Relationship (Primary). Save button disabled if form not valid or if no changes have been made. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Profile Page |
| **Actors:** | User, Admin |
| **Scenario:** | Name, User/Pass, Address, Phone tabs are shown. Save button disabled if form not valid or if no changes have been made. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Appointments Page |
| **Actors:** | User |
| **Scenario:** | Can see all own appointments by date. Can cancel individual appointments for themselves or any dependent in their household. Create appointment functionality disabled (simply hide button) since only Company personnel should be creating appointments. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Appointments Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Can see all company appointments by date (past/future/present/none) and selected clinic. Ability to cancel anyone’s appointment. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Appointments Page – Create |
| **Actors:** | Office, Admin |
| **Scenario:** | Step-by-step process.  1. Search for a household by payer (primary) name. This search is a wildcard search. If entering one word, assumes last name search. If two words entered, first name / last name. If three words, first / middle / last. Not case sensitive.  2. Select a household based on search result (results should show account #, and name of everyone in house).  3. Select a member of the house the appointment is for.  4. Select a date for service.  5. Select the desired service.  6. A long list is presented of all available times for the day by provider, room, length of time, and whether provider is qualified for service. Add one or more.  7. Added services/times are pushed into a new list showing what they are. These may be removed individually.  8. Added services/times show the standard cost – which is modifiable.  9. Save/Create should only be visible if form is valid and service(s)/time(s) have been selected.  10. Save will push new appointment into Appointments Page list. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Qualifications Page |
| **Actors:** | Office, Admin |
| **Scenario:** | User presented with a list of services they deem themselves qualified for. Can remove them individually or add new ones. Can only add services they don’t already have, and services that have been assigned to a Clinic previously. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | 10 reports should be made available to execute based upon requirements |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 1  List of all households with the household ID, name, address, and home phone along with the patient ID, name and relationship for each patient. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 2  List of the insurance coverage for all households by household ID, household name, insurance company ID and company name. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 3  List all patients in alphabetical order by patient ID, name, and date of birth along with the name of the insurance company and policy number. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 4  Itemized billings for all households with the household ID, household name, patient ID, patient name, service received, and the cost of the service. Show the output in alphabetical order by household name, patient name and billing date. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 5  Itemized billings for all households with the household ID, household name, patient ID, patient name, service received, and the cost of the service. Show the output in alphabetical order by household name, patient name and billing date. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 6  List each provider with all services he or she is qualified to render. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 7  List each service available with all providers who are qualified to offer this service. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 8  List all future appointments by name of patient, appointment date and time, estimated length of service, and contact home phone number. Dates and times should be in calendar order |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 9  For a given date, list all services provided by each provider in alphabetical order by name of the provider. Show the service ID, service description and cost of service. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Reports Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Report 10  For a given date, list the total amount of services each provider rendered. Show in alphabetical order by the provider’s name. |

|  |  |
| --- | --- |
| **Use Case Name:** | At People Page |
| **Actors:** | Office |
| **Scenario:** | Only Manage Client profiles available |

|  |  |
| --- | --- |
| **Use Case Name:** | At People Page |
| **Actors:** | Admin |
| **Scenario:** | Can manage staff or clients |

|  |  |
| --- | --- |
| **Use Case Name:** | At People Page |
| **Actors:** | Office, Admin |
| **Scenario:** | Shown a list of profiles in the system regardless of Clinic. Can Edit or Delete profiles individually. |

|  |  |
| --- | --- |
| **Use Case Name:** | At People Page – Create/Edit |
| **Actors:** | Office, Admin |
| **Scenario:** | Same rules apply as the Profile Page (see profile page use cases) when managing a profile (managing a client/household gets extra fields). |

|  |  |
| --- | --- |
| **Use Case Name:** | At Clinics Page |
| **Actors:** | Admin |
| **Scenario:** | First need to select a Clinic to edit.  Can create/edit/delete Clinics that are in the system.  Tabs give ability to designate Clinic name, address, phone, providers, services, and rooms. Can edit/delete providers, services, and rooms individually. Can create/add new providers, services, and rooms. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Clinics Page – Add Provider |
| **Actors:** | Admin |
| **Scenario:** | A pick list is shown only for people with the Office Role that have not been designated as a Provider yet. Adding a new Provider that is not in the list requires visiting the People Page to add them. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Clinics Page – Add Service |
| **Actors:** | Admin |
| **Scenario:** | If there are services provided by another clinic that are not in the selected one, a dropdown list is provided to pick from. Can also opt to create a completely new service. |

|  |  |
| --- | --- |
| **Use Case Name:** | At Clinics Page – Add Room |
| **Actors:** | Admin |
| **Scenario:** | Can enter the name for a new room. |

|  |  |
| --- | --- |
| **Use Case Name:** | Any error occurs on the back end |
| **Actors:** | User, Office, Admin |
| **Scenario:** | All exceptions or validation errors on the back end should be caught, handled, and bubbled up and displayed gracefully to the user. |

|  |  |
| --- | --- |
| **Use Case Name:** | Modals |
| **Actors:** | User, Office, Admin |
| **Scenario:** | All modals should have a similar look and feel. Cancel and Save buttons should always be at the bottom in the same order. Unless the modal has left-nav tabs – then the save button should be at the top-right. Every modal popup should have a close “X” button available and should disappear if clicking away from it. If a modal gets too big for the screen, a scroll should be present to allow it to be brought into view by the user. |

|  |  |
| --- | --- |
| **Use Case Name:** | Front-End Validation |
| **Actors:** | User, Office, Admin |
| **Scenario:** | All fields that are required, have a max length, or “selected” requirement should highlight themselves with a Red border when not valid. All pages with a left-nav should show a Warning Icon if any content within its tab is invalid. Save buttons should always be disabled if a body of work is invalid. |

## Use Cases (Expanded Form)

|  |  |  |
| --- | --- | --- |
| **USE CASE #** | 1 | |
| **Goal in Context** | User wants to create an appointment | |
| **Scope & Level** | Primary Task | |
| **Preconditions** | User is already registered and logged in | |
| **Success End Condition** | Appointment is created | |
| **Failed End Condition** | Appointment is not created | |
| **Primary/Secondary Actors** | Office, Admin | |
| **Trigger** |  | |
| **DESCRIPTION** | **Step** | **Action** |
|  | 1 | At Landing Page |
|  | 2 | Click Appointments Tile |
|  | 3 | Select “Create Appointment” |
|  | 4 | Search for a household by payer (primary) name. This search is a wildcard search. If entering one word, assumes last name search. If two words entered, first name / last name. If three words, first / middle / last. Not case sensitive. |
|  | 5 | Select a household based on search result (results should show account #, and name of everyone in house). |
|  | 6 | Select a member of the house the appointment is for. |
|  | 7 | Select a date for service. |
|  | 8 | Select the desired service. |
|  | 9 | A long list is presented of all available times for the day by provider, room, length of time, and whether provider is qualified for service. Add one or more. |
|  | 10 | Added services/times are pushed into a new list showing what they are. These may be removed individually. |
|  | 11 | Added services/times show the standard cost – which is modifiable. |
|  | 12 | Save/Create should only be visible if form is valid and service(s)/time(s) have been selected. No steps above can be skipped. |
|  | 13 | Save will push new appointment into Appointments Page list. |
| **EXTENSIONS** | **Step** | **Branching Action** |
|  | 1 | None |
| **SUB-VARIATIONS** | **Step** | **Branching Action** |
|  | 1 | None |

|  |  |  |
| --- | --- | --- |
| **USE CASE #** | 2 | |
| **Goal in Context** | User wants to add a Client to the system | |
| **Scope & Level** | Primary Task | |
| **Preconditions** | User is already registered and logged in | |
| **Success End Condition** | Client is added and can log into profile | |
| **Failed End Condition** | Client is not added | |
| **Primary/Secondary Actors** | Office, Admin | |
| **Trigger** |  | |
| **DESCRIPTION** | **Step** | **Action** |
|  | 1 | At Landing Page |
|  | 2 | Click People Tile |
|  | 3 | Select “Create Appointment” |
|  | 4 | Ensure “Manage Clients” is selected on the left |
|  | 5 | Click Create Profile |
|  | 6 | Enter information in Name tab |
|  | 7 | Enter information in the User/Pass tab |
|  | 8 | Enter information in the Address tab |
|  | 9 | Enter information in the Phone tab |
|  | 10 | Enter information in the Insurance tab |
|  | 11 | Enter information in the Dependents tab (optional) |
|  | 12 | Click Save |
|  | 13 | Save will push new Client into list |
| **EXTENSIONS** | **Step** | **Branching Action** |
|  | 1 | None |
| **SUB-VARIATIONS** | **Step** | **Branching Action** |
|  | 1 | None |

|  |  |  |
| --- | --- | --- |
| **USE CASE #** | 3 | |
| **Goal in Context** | User wants to add new Staff to the system | |
| **Scope & Level** | Primary Task | |
| **Preconditions** | User is already registered and logged in | |
| **Success End Condition** | Staff is added and can log into profile | |
| **Failed End Condition** | Staff is not added | |
| **Primary/Secondary Actors** | Admin | |
| **Trigger** |  | |
| **DESCRIPTION** | **Step** | **Action** |
|  | 1 | At Landing Page |
|  | 2 | Click People Tile |
|  | 3 | Select “Create Appointment” |
|  | 4 | Ensure “Manage Clients” is selected on the left |
|  | 5 | Click Create Profile |
|  | 6 | Enter information in Name tab |
|  | 7 | Enter information in the User/Pass tab |
|  | 8 | Enter information in the Address tab |
|  | 9 | Enter information in the Phone tab |
|  | 12 | Click Save |
|  | 13 | Save will push new Staff into list |
| **EXTENSIONS** | **Step** | **Branching Action** |
|  | 1 | None |
| **SUB-VARIATIONS** | **Step** | **Branching Action** |
|  | 1 | None |

|  |  |  |
| --- | --- | --- |
| **USE CASE #** | 4 | |
| **Goal in Context** | User wants to create a Clinic | |
| **Scope & Level** | Primary Task | |
| **Preconditions** | User is already registered and logged in | |
| **Success End Condition** | Clinic is created and has Providers, Services, and Rooms | |
| **Failed End Condition** | Staff is not added | |
| **Primary/Secondary Actors** | Admin | |
| **Trigger** |  | |
| **DESCRIPTION** | **Step** | **Action** |
|  | 1 | At Landing Page |
|  | 2 | Click Clinics Tile |
|  | 3 | Select a Clinic from the dropdown (if one exists). If not, click “Add New” |
|  | 4 | Ensure “Manage Clients” is selected on the left |
|  | 5 | Click Create Profile |
|  | 6 | Enter information in Name tab |
|  | 7 | Add at least one Provider in the Providers tab |
|  | 8 | Add at least one Service in the Services tab |
|  | 9 | Add at least one Room in the Rooms tab |
|  | 10 | Click Save (upper right since this is a side-nav scenario) |
|  | 11 | Save will push new Clinic into dropdown |
| **EXTENSIONS** | **Step** | **Branching Action** |
|  | 1 | None |
| **SUB-VARIATIONS** | **Step** | **Branching Action** |
|  | 1 | None |

|  |  |  |
| --- | --- | --- |
| **USE CASE #** | 5 | |
| **Goal in Context** | User wants to manage profile | |
| **Scope & Level** | Primary Task | |
| **Preconditions** | User is already registered and logged in | |
| **Success End Condition** | Profile is updated | |
| **Failed End Condition** | Profile is not modified | |
| **Primary/Secondary Actors** | User, Office, Admin | |
| **Trigger** |  | |
| **DESCRIPTION** | **Step** | **Action** |
|  | 1 | At Landing Page |
|  | 2 | Click Profile Tile |
|  | 3 | Select a Clinic from the dropdown (if one exists). If not, click “Add New” |
|  | 4 | Enter desired information into Name tab |
|  | 5 | Enter desired information into User/Pass tab |
|  | 6 | Enter desired information into Address tab |
|  | 7 | Enter desired information into Phone tab |
|  | 8 | <Branching Action 1> |
|  | 9 | Click Save (upper right since this is a side-nav scenario) |
| **EXTENSIONS** | **Step** | **Branching Action 1** |
|  | 1 | If logged in as a Client Role, more tabs are available |
|  | 2 | Enter information into Insurance tab |
|  | 3 | Enter information into dependents tab (optional) |
| **SUB-VARIATIONS** | **Step** | **Branching Action** |
|  | 1 | None |

## Domain / Concept Model (UML)



## System Sequence Diagrams (UML)

Create Appointment



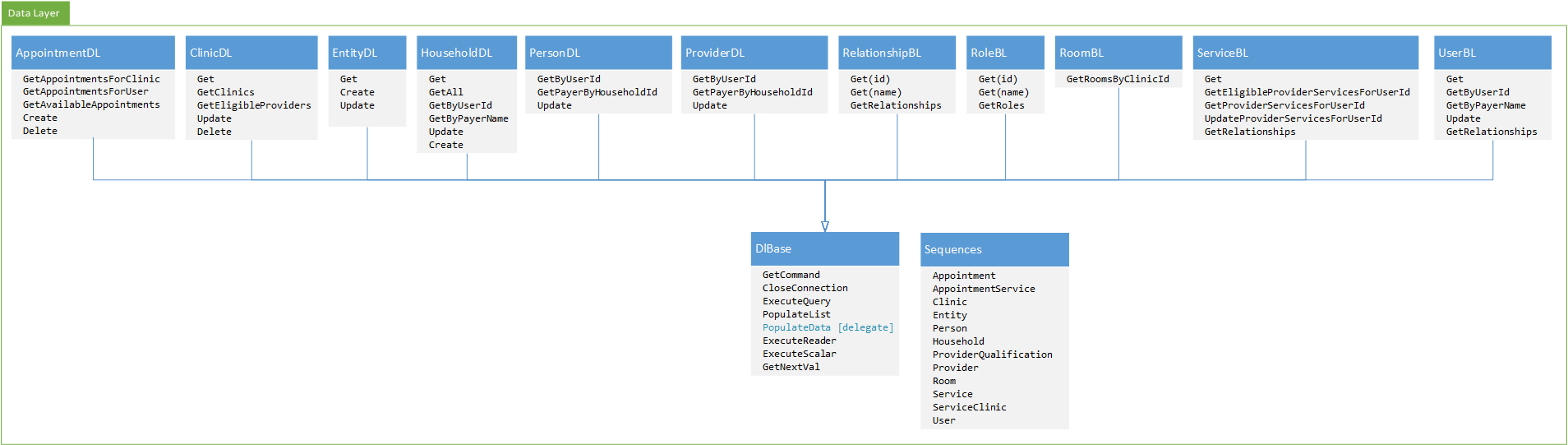
Create Household



## Class Diagrams (UML)









## Deployment Diagram (UML)



## Glossary

* Arch – Arch Dental Associates.
* UML – Unified Modeling Language – A common way to represent dataflows, structures, etc in a graphical way that can be easily understood.
* Use Case -  A list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal.
* Database – A computer program that is a common place for storing data. Software applications can use it as a common place to store information to recall later.
* SPA – Single Page Application – A web development methodology where rendering duties, and all front end (web browser) functionality is pushed into a user’s browser.
* CRUD – Create / Read / Update / Delete

# Conclusion

Creating this application has been fun, exhausting, challenging, interesting, and enjoyable. Fun because it is really great to watch a software application grow and mature as my ideas fall out of my fingers. It really brings me great satisfaction to get to be part of that creative process – and have it be the career that I work in every day. Exhausting, obviously, due to the time constraints we had, and my desire to create something of good quality. Challenging, as with any new project, as I stumbled along initially to find a way to create the type of project I had in mind. I really wanted to create something that resembled what I do professionally – but rarely get the opportunity to start from scratch. Interesting, because each new set of challenges really makes me flex my brain capacity to the limit at times. And when I finally find the answer, the solution to the bug that has been bothering me, and make that cool thing work – it brings about a great amount of satisfaction for me.

In reality, I may have gone a bit overboard with the complexity of this application. But, to be honest, I just really had it in my mind to produce the best quality of code that I could – something I would be willing to show my employer, and really put forth my best effort for this Master’s program.

I have really enjoyed this class, and my choice to participate in the Master’s program at Webster. And I really appreciate that the school gave me an opportunity to take it on.

# Program Implementation

## Program Structure

I believe the program structure is best illustrated by Class Diagrams and Domain/Concept Diagram from previous pages. What I tried to show here is a layered approach to my application. A layered approach can certainly be implemented with just about any application type – whether it be an ASP.NET WebForms application, or a SPA (single page application) as I have created.

But the nice thing about the SPA is that it completely separates the front end from the back end. This takes a mountain of web-page rendering load off of web servers by pushing it up into the client’s browser. Historically, web-page compilation is usually done on the server-side. All controls (inputs, buttons, tables, etc), where they belong on the page, injection/binding of values into those controls, rendering of tables, was all done by the server. The final rendered HTML was then pushed to the browser for display. Any user interaction (post-backs) required that some or all of the web page markup be sent back to the server, parsed by the server to extract changes, process events the user has triggered, re-render the new results, and then finally push the whole web-page back to the client’s browser. This happens every single time a post-back is done, which can create a very heavy load on the web-server. Especially as more and more users come online. This process was necessary in the past because browsers weren’t powerful enough, and computers didn’t have the kind of processing power to do all of this work. But with modern browsers and computers, all of the rendering work can be pushed to the browser. The server hands the user all of the HTML, CSS, and JavaScript files they will need (either all up front, or piece-meal – depends on the application and requirements) and then the browser takes over from there. The only time it has to hit the web-server is to ask for data. And the request/response is very light-weight compared to full-post backs. It is possible for users to even change pages without getting a single piece data from the web-server after the initial page load. All of this opens up the door for being able to do more for less in the web-server side, and being able to do some really cool things in the client’s browser. Lastly, it makes developing web-applications, maintaining them, and enhancing them really fast (once you have your framework laid out). In my SPA, each page you see on the screen has a unique HTML file, and a unique JavaScript file (or controller) that is directly tied to it. This isolation ensures that no JavaScript that was written for one HTML page will bleed over into another HTML page or another JavaScript file’s logic. Since JavaScript is notorious for causing developers to pull their hair out when trying to figure out why a global variable is getting set from some unknown/forgotten code somewhere, this separation is vital to the success of the application.

That concludes the first layer. The UI, or Presentation Layer. The next layer is the Communication Layer. This is typically some kind of SOAP or REST service back-end. This layer is responsible for one thing only, receive requests for data from the client, translate them into objects that lower layers can understand, and forward on the request. Once the lower layers are done, it will take the response, translate if back into something the browser can understand, and send it off.

The next layer down is what I have called the Business Layer. Here is where all of the logic happens. For example, if I want to update a household, it is here in this layer (and more specifically, the Household business layer class) that I would make the call to update the main Household properties, and then have logic to loop through each dependent to see which ones from the request are new, which were deleted, and which were updated. Once I have that information I can then call the business layer for a “Person” (the object for a dependent) and tell it to create, update, or delete each dependent in the request. When a business layer wants to query or persist information to the database, it has to do that via the next layer down, the Data Layer.

Inside the data layer resides all of the common code for all CRUD operations on the database. There is a single class that directly maps to nearly every table in the database. Each data layer class inherits a common base class. This base class provides common functionality for accessing oracle. Things like getting a next sequence, ExecuteQuery, ExecuteScalar, and ExecuteReader, creating and closing a connection. Each data layer class has mappings for how to translate data out of that table. The mappings are contained in a delegate function which is passed down to the base class when calling ExecuteReader.

It can be noticed easily that for each main unit of work to be done (manage appointment, manage user, etc) there is an object representing that unit of work in each layer. This makes it very easy for a developer to know where to look for functionality they need when writing code as every similar bit of functionality is contained in a logical/layered location. This separation of concern allows for highly maintainable and enhanceable code.

This style of code also makes the need for endless comments much less – almost to the point of being unnecessary, since the code should document itself by the layer you are in, and smart naming of functions, properties, classes, etc. It should be intuitive where to look next when adding the next thing, or trying to understand it.

## Program Structure

### Presentation Layer

#### confirm.html

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

<h4 class="modal-title">Confirm</h4>

</div>

<div class="modal-body">

<div class="form-group">

Are you sure you want to do this?

</div>

<div>

<button ng-click="close()" class="btn btn-default">Cancel</button>

<button class="btn btn-success" ng-click="confirm()">

Confirm

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</button>

</div>

</div>

#### confirm.less

.people {

}

#### ConfirmCtrl.ts

(function (angular) {

'use strict';

angular.module('clinic')

.controller('ConfirmCtrl', ConfirmCtrl);

function ConfirmCtrl($scope, $uibModalInstance, params) {

$scope.confirm = function () {

$scope.isLoading = true;

params.confirm().then(function () {

$scope.isLoading = false;

$scope.close();

});

};

$scope.close = function () {

$uibModalInstance.dismiss();

};

}

} ((<any>window).angular));

#### login.html

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

<h4 class="modal-title">Login</h4>

</div>

<div class="modal-body">

<div class="form-group">

<label>Username</label>

<input ng-model="username" type="text" class="form-control" />

</div>

<div class="form-group">

<label>Password</label>

<input ng-model="password" type="password" class="form-control" />

</div>

<div>

<button ng-click="close()" class="btn btn-default">Cancel</button>

<button ng-disabled="!isPageValid()" ng-click="login()" class="btn btn-primary">

Login

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</button>

</div>

</div>

#### LoginCtrl.ts

(function (angular) {

'use strict';

angular.module('clinic')

.controller('LoginCtrl', LoginCtrl);

function LoginCtrl($scope, $uibModalInstance, params) {

$scope.username = '';

$scope.password = '';

$scope.isLoading = false;

$scope.isPageValid = function () {

return $scope.username.length > 0 &&

$scope.password.length > 0 &&

!$scope.isLoading;

};

$scope.close = function () {

$uibModalInstance.dismiss();

};

$scope.login = function () {

$scope.isLoading = true;

var data = {

username: $scope.username,

password: $scope.password

};

params.submit(data).then(function () {

$scope.isLoading = false;

$scope.close();

});

};

}

} ((<any>window).angular));

#### register.html

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

<h4 class="modal-title" id="myModalLabel">Register</h4>

</div>

<div class="modal-body">

<div class="form-group required">

<div ng-if="shouldGatherHouseholdId()">

<label class="required">Account #</label>

<input ng-model="model.householdId" type="number" class="form-control" />

</div>

<label class="required">Username</label>

<input ng-model="model.username" type="text" class="form-control" />

<label class="required">Password</label>

<input ng-model="model.password" type="password" class="form-control" />

<label class="required">Confirm Password</label>

<input ng-model="model.passConf" type="password" class="form-control" />

<label ng-if="!passwordsMatch()" class="text-warning small">Passwords don't match</label>

<div ng-if="shouldGatherRole()">

<label class="required">Role</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="true">

<span class="message">**{{**getRoleMessage()**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="role **in** roles" ng-click="setRole(role)"><a>**{{**role.Name**}}**</a></li>

</ul>

</div>

</div>

</div>

<div>

<button ng-click="close()" class="btn btn-default">Cancel</button>

<button ng-click="register()" ng-disabled="!isPageValid()" class="btn btn-primary">

**{{**("Register")**}}**

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</button>

</div>

</div>

#### RegisterCtrl.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('RegisterCtrl', RegisterCtrl);

function RegisterCtrl($scope, userService, householdService, settings, params) {

$scope.model = {

householdId: null,

username: '',

password: '',

passConf: '',

selectedRole: null

};

$scope.isLoading = false;

$scope.isAdmin = (settings.User && settings.User.Role && settings.User.Role.Name === "Administrator") ? true : false;

$scope.roles = [];

function init() {

$scope.isLoading = true;

userService.getRoles().then(function (roles) {

$scope.roles = roles;

if ($scope.isEditMode) {

if ($scope.shouldGatherHouseholdId()) {

householdService.GetByUserId($scope.settings.user.Id).then(function (household) {

$scope.model.householdId = household.Id;

$scope.isLoading = false;

});

} else {

$scope.model.selectedRole = \_.find($scope.roles, function (role) {

return role.Id === $scope.settings.user.Role.Id;

});

$scope.isLoading = false;

}

} else {

$scope.isLoading = false;

}

});

}

$scope.passwordsMatch = function () {

return $scope.model.password.trim() === $scope.model.passConf.trim();

};

$scope.shouldGatherRole = function () {

return $scope.isAdmin && !$scope.shouldGatherHouseholdId();

};

$scope.shouldGatherHouseholdId = function () {

return settings.AdminExists && params.gatherHouseholdId;

};

$scope.setRole = function (role) {

$scope.model.selectedRole = role;

};

$scope.getRoleMessage = function () {

return $scope.model.selectedRole ? $scope.model.selectedRole.Name : 'Select a role...';

};

$scope.isPageValid = function () {

return $scope.passwordsMatch() &&

$scope.model.password.length > 0 &&

$scope.model.username.length > 0 &&

(!$scope.shouldGatherHouseholdId() || ($scope.shouldGatherHouseholdId() && $scope.model.householdId > 0)) &&

(!$scope.shouldGatherRole() || ($scope.shouldGatherRole() && $scope.model.selectedRole)) &&

!$scope.isLoading;

};

$scope.close = function () {

params.close();

};

$scope.register = function () {

$scope.isLoading = true;

var data = {

isEditMode: $scope.isEditMode,

householdId: $scope.model.householdId,

username: $scope.model.username,

password: $scope.model.password,

role: $scope.model.selectedRole

};

params.submit(data);

};

init();

}

} ((<any>window).angular));

#### header.html

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" ng-click="goHome()">

<div>

<img src="Images/logo.png" />

<span>Arch Dental</span>

</div>

</a>

</div>

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav navbar-right" ng-if="!settings.User">

<li class="divider-vertical"></li>

<li>

<a ng-click="register()"><i class="fa fa-cogs"></i>&nbsp Register</a>

</li>

<li ng-if="settings.AdminExists">

<a ng-click="login()"><i class="fa fa-user"></i>&nbsp Log In</a>

</li>

</ul>

<ul class="nav navbar-nav navbar-right" ng-if="settings.User">

<li class="divider-vertical"></li>

<li>

<a ng-click="goHome()">

<i class="fa fa-user"></i>

<span>**{{**getName()**}}**</span>

</a>

</li>

<li>

<a ng-click="logout()">

&nbsp Log Out

</a>

</li>

</ul>

</div>

</div>

</nav>

#### HeaderCtrl.ts

var toastr;

var \_baseUrl;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('HeaderCtrl', HeaderCtrl);

function HeaderCtrl($q, $rootScope, $scope, $uibModal, userService, settings, $location) {

$scope.settings = settings;

$scope.getName = function () {

if ($scope.settings.User) {

var name = ($scope.settings.User.FirstName ? $scope.settings.User.FirstName : '') +

($scope.settings.User.LastName ? ' ' + $scope.settings.User.LastName + ' ' : '') +

($scope.settings.User.Role ? ('(' + $scope.settings.User.Role.Name + ')') : '');

return name.trim();

}

return '';

};

$scope.goHome = function () {

if ($scope.settings.User) {

// go to tile page

} else {

// go to home page

}

};

$scope.login = function () {

$uibModal.open({

templateUrl: \_baseUrl + 'app/header/login/login.html',

controller: 'LoginCtrl',

resolve: {

params: function () {

return {

submit: function (data) {

var dfd = $q.defer();

userService.login(data).then(function (response) {

dfd.resolve();

$location.path('/home');

});

return dfd.promise;

}

}

}

}

});

};

$scope.register = function () {

$uibModal.open({

templateUrl: \_baseUrl + 'app/header/login/register.html',

controller: 'RegisterCtrl',

resolve: {

params: function () {

return {

gatherHouseholdId: true,

submit: function (data) {

var dfd = $q.defer();

userService.register(data).then(function (response) {

dfd.resolve();

$location.path('/home');

});

return dfd.promise;

}

}

}

}

});

};

$scope.logout = function () {

userService.logout().then(function () {

$location.path('/');

});

};

}

} ((<any>window).angular));

#### headerDirective.ts

var \_baseUrl;

(function (angular) {

'use strict';

angular.module('clinic').directive('appNav', appNav);

function appNav() {

return {

restrict: 'E',

replace: true,

scope: true,

templateUrl: \_baseUrl + 'app/header/header.html',

controller: 'HeaderCtrl'

};

};

}((<any>window).angular));

#### newAppointment.html

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

<h4 class="modal-title" id="myModalLabel">Appointment</h4>

</div>

<div class="modal-body appointment">

<div ng-if="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div>

<div class="form-group">

<label class="required">Clinic</label>

<span ng-model="model.Clinic.Name" />

</div>

<div class="form-group" ng-disabled="selectedHousehold">

<label class="required">Enter the payer name of a household:</label>

<div class="input-group">

<input ng-model="householdSearchName" type="text" placeholder="<Last>, <First Last>, <First Middle Last>"

class="form-control" ng-class="{'ng-invalid': !isHouseholdSearchValid()}" />

<span class="input-group-btn">

<button class="btn btn-primary" ng-disabled="!isHouseholdSearchValid()" ng-click="householdSearch()">Search</button>

</span>

</div>

</div>

<div class="form-group" ng-if="householdResults.length">

<label class="required">Select a household</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**getHouseholdSelectText(selectedHousehold)**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="household **in** householdResults" ng-click="setHousehold(household)"><a>**{{**getHouseholdSelectText(household)**}}**</a></li>

</ul>

</div>

</div>

<div ng-if="selectedHousehold">

<div class="form-group">

<label class="required">Select a member of the household</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**selectedMemberMessage()**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="member **in** selectedHousehold.People" ng-click="setMember(member)"><a>**{{**member.FirstName + ' ' + member.LastName**}}**</a></li>

</ul>

</div>

</div>

<div class="form-group">

<label class="required">Service Date</label>

<input type="date" ng-model="serviceDate" ng-change="serviceDateChanged(serviceDate)" class="form-control" placeholder="MM/dd/yyyy" min="**{{**getToday()**}}**" required />

</div>

<div class="form-group">

<label class="required">Select a service to add</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**selectedServiceMessage()**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="service **in** model.Clinic.Services" ng-click="searchServiceAvailability(service)"><a>**{{**service.Name**}}**</a></li>

</ul>

</div>

</div>

<div ng-if="serviceDate && model.Person">

<div class="panel panel-info">

<div class="panel-heading">Pick a service/time to add to appointment</div>

<div class="panel-body">

<table class="table table-striped">

<thead>

<tr>

<th>Timeslot</th>

<th>Minutes</th>

<th>Provider</th>

<th>Qualified</th>

<th>Room</th>

<th></th>

</tr>

</thead>

<tbody>

<tr ng-repeat="availableService **in** availableTimesForService">

<td><span>**{{**availableService.StartTimeString**}}**</span></td>

<td><span>**{{**availableService.Service.Minutes**}}**</span></td>

<td><span>**{{**availableService.Provider.FirstName + ' ' + availableService.Provider.LastName**}}**</span></td>

<td><span>**{{**availableService.IsQualified ? 'Yes' : 'No'**}}**</span></td>

<td><span>**{{**availableService.Room.Name**}}**</span></td>

<td class="table-btn">

<span class="fa-stack" ng-click="addServiceToAppointment(availableService)" style="cursor: pointer">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-plus fa-stack-1x fa-inverse"></i>

</span>

</td>

</tr>

</tbody>

</table>

<span ng-if="!availableTimesForService.length">None found.</span>

</div>

</div>

<div class="panel panel-primary">

<div class="panel-heading">Services added to appointment</div>

<div class="panel-body">

<table class="table table-striped">

<thead>

<tr>

<th>Timeslot</th>

<th>Service</th>

<th>Minutes</th>

<th>Cost</th>

<th>Provider</th>

<th>Qualified</th>

<th>Room</th>

<th></th>

</tr>

</thead>

<tbody>

<tr ng-repeat="addedService **in** model.AppointmentServices">

<td><span>**{{**addedService.StartTimeString**}}**</span></td>

<td><span>**{{**addedService.Service.Name**}}**</span></td>

<td><span>**{{**addedService.Service.Minutes**}}**</span></td>

<td><input type="number" ng-model="addedService.Service.Cost" /></td>

<td><span>**{{**addedService.Provider.FirstName + ' ' + addedService.Provider.LastName**}}**</span></td>

<td><span>**{{**addedService.IsQualified ? 'Yes' : 'No'**}}**</span></td>

<td><span>**{{**addedService.Room.Name**}}**</span></td>

<td class="table-btn">

<span class="fa-stack" style="color: red" ng-click="removeAddedService(addedService)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-times fa-stack-1x fa-inverse"></i>

</span>

</td>

</tr>

</tbody>

</table>

<span ng-if="!model.AppointmentServices.length">Add services above.</span>

</div>

</div>

</div>

</div>

<hr />

<button ng-click="close()" class="btn btn-default">Cancel</button>

<button ng-click="save()" ng-disabled="!isPageValid()" class="btn btn-success">

Save

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</button>

</div>

</div>

#### NewAppointmentCtrl.ts

var \_, toastr;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('NewAppointmentCtrl', NewAppointmentCtrl);

function NewAppointmentCtrl($scope, $timeout, appointmentService, householdService, clinicService, settings, params) {

$scope.model = {

Clinic: params.clinic,

Person: null,

AppointmentServices: []

};

$scope.isLoading = false;

$scope.householdSearchName = null;

$scope.selectedHousehold = null;

$scope.householdResults = [];

$scope.desiredServices = [];

$scope.selectedService = null;

$scope.serviceDate = new Date();

$scope.availableTimesForService = [];

function init() {

}

$scope.householdSearch = function () {

$scope.isLoading = true;

var names = \_.split($scope.householdSearchName, ' ', 3);

var params: any = {};

if (names.length === 0 || names.length > 3) {

$scope.isLoading = false;

return;

}

if (names.length === 1) {

params.LastName = names[0];

} else if (names.length === 2) {

params.FirstName = names[0];

params.LastName = names[1];

} else if (names.length === 3) {

params.FirstName = names[0];

params.MiddleName = names[1];

params.LastName = names[2];

}

householdService.getByPayerName(params).then(function (result) {

$scope.householdResults = result;

$scope.isLoading = false;

});

};

$scope.isHouseholdSearchValid = function () {

var names = \_.split($scope.householdSearchName, ' ');

return names.length > 0 && names.length <= 3;

};

$scope.selectedMemberMessage = function () {

return $scope.model.Person ? $scope.model.Person.FirstName + ' ' + $scope.model.Person.LastName : 'Select a member...';

};

$scope.selectedServiceMessage = function () {

return $scope.selectedService ? $scope.selectedService.Name : 'Select a service...';

};

$scope.setHousehold = function (household) {

$scope.selectedHousehold = household;

};

$scope.getHouseholdSelectText = function (household) {

if (!household) {

return 'Select a household...';

}

var names = \_.map(household.People, function (person) {

return person.FirstName + ' ' + person.LastName;

});

var nameString = \_.join(names, ', ');

return household.Id + ' - ' + '( ' + nameString + ' )';

};

$scope.setMember = function (member) {

$scope.model.Person = member;

};

$scope.serviceDateChanged = function (date) {

$scope.serviceDate = date;

\_.remove($scope.model.AppointmentServices, function () { return true; });

if ($scope.selectedService && $scope.serviceDate && $scope.model.Person) {

$scope.searchServiceAvailability($scope.selectedService);

}

};

$scope.isAdded = function (service) {

return $scope.model.AppointmentServices.length &&

\_.find($scope.model.AppointmentServices, function (aptSvc) { return aptSvc.Service.Name === service.Service.Name; });

};

$scope.removeServiceFromAppointment = function (addedService) {

$scope.isLoading = true;

\_.remove($scope.model.AppointmentServices, addedService);

$scope.searchServiceAvailability($scope.selectedService);

};

$scope.getToday = function () {

var today = new Date();

var dd: any = today.getDate();

var mm: any = today.getMonth() + 1;

var yyyy = today.getFullYear();

if (dd < 10) {

dd = '0' + dd

}

if (mm < 10) {

mm = '0' + mm

}

return yyyy + '-' + mm + '-' + dd;

};

$scope.searchServiceAvailability = function (service) {

$scope.selectedService = service;

$scope.isLoading = true;

appointmentService.getAvailableAppointments($scope.model.Clinic.Id, $scope.selectedService.Id, $scope.serviceDate)

.then(function (results) {

\_.each(results, function (result) {

result.StartTime = new Date(result.StartTimeString);

});

$scope.availableTimesForService = results;

$scope.isLoading = false;

});

};

$scope.addServiceToAppointment = function (availableService) {

if ($scope.isAdded(availableService)) {

toastr.warning('You can only add a service type once');

return;

}

if ($scope.model.AppointmentServices.length) {

var lastSvc = $scope.model.AppointmentServices[$scope.model.AppointmentServices.length - 1];

var correctNextStartTime = new Date(lastSvc.StartTime.getTime() + lastSvc.Service.Minutes \* 60000);

if (availableService.StartTime.getTime() !== correctNextStartTime.getTime()) {

toastr.warning('Next service should start at ' + correctNextStartTime.toLocaleTimeString());

return;

}

}

$scope.model.AppointmentServices.push(availableService);

};

$scope.removeAddedService = function (addedService) {

\_.remove($scope.model.AppointmentServices, addedService);

};

$scope.isPageValid = function () {

return $scope.model.Clinic &&

$scope.model.Person &&

$scope.model.AppointmentServices &&

$scope.model.AppointmentServices.length;

};

$scope.close = function () {

params.close();

};

$scope.save = function () {

$scope.isLoading = true;

params.submit($scope.model).then(function () {

$scope.isLoading = false;

});

};

init();

}

} ((<any>window).angular));

#### appointment.html

<home-nav></home-nav>

<div class="appointment page-wrapper">

<div class="banner">

Manage Appointments

</div>

<div class="page-content" style="padding: 25px;">

<div ng-if="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div class="form-group" ng-if="!isLoading && !isUser">

<label class="required">Select a Clinic</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**selectedClinic.Name**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="clinic **in** clinincs" ng-click="setClinic(clinic)"><a>**{{**clinic.Name**}}**</a></li>

</ul>

</div>

</div>

<div class="form-group">

<label class="required">Service Date</label>

<div class="input-group">

<input type="date" ng-model="serviceDate" ng-change="serviceDateChanged()" class="form-control" placeholder="MM/dd/yyyy" />

<span class="input-group-btn">

<button class="btn btn-danger" ng-click="clearDate()">Clear</button>

</span>

</div>

</div>

<div class="panel panel-info">

<div class="panel-heading">Appointments</div>

<div class="panel-body" style="margin: 25px;">

<table class="table table-striped">

<thead>

<tr>

<th>Account #</th>

<th>Patient Name</th>

<th>Date/Time</th>

<th>Services</th>

<th>Providers</th>

<th>Rooms</th>

<th>Total (min)</th>

<th>Total ($)</th>

<th>Cancel</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="appointment **in** appointments">

<td><span>**{{**appointment.Person.HouseholdId**}}**</span></td>

<td><span>**{{**appointment.Person.FirstName + ' ' + appointment.Person.LastName**}}**</span></td>

<td><span>**{{**getStartTime(appointment)**}}**</span></td>

<td><span>**{{**getServicesString(appointment)**}}**</span></td>

<td><span>**{{**getProvidersString(appointment)**}}**</span></td>

<td><span>**{{**getRoomsString(appointment)**}}**</span></td>

<td><span>**{{**getTotalTime(appointment)**}}**</span></td>

<td><span>**{{**getTotalPrice(appointment)**}}**</span></td>

<td class="table-btn red">

<span class="fa-stack" ng-click="delete(appointment)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-times fa-stack-1x fa-inverse"></i>

</span>

</td>

</tr>

</tbody>

</table>

<div ng-if="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<span ng-if="!appointments.length">None found.</span>

</div>

</div>

<div>

<button class="btn btn-primary" ng-click="openAppointment()" ng-if="!isUser">Create Appointment</button>

</div>

</div>

</div>

#### appointment.less

.appointment {

.panel {

max-height: 400px;

overflow-y: auto;

.panel-body {

margin: 25px;

}

}

}

.modal-appointment-modal-size {

width: 60% !important;

}

#### AppointmentCtrl.ts

var \_;

var \_baseUrl;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('AppointmentCtrl', AppointmentCtrl);

function AppointmentCtrl($scope, $uibModal, clinicService, appointmentService, settings) {

$scope.isUser = settings.User.Role.Name === "User";

$scope.isLoading = true;

$scope.clinics = [];

$scope.selectedClinic = {};

$scope.appointments = [];

$scope.serviceDate = $scope.isUser ? null : new Date();

function init() {

clinicService.getAll().then(function (clinics) {

$scope.clinics = clinics;

$scope.setClinic(clinics[0]);

});

}

$scope.setClinic = function (clinic) {

$scope.isLoading = true;

$scope.selectedClinic = clinic;

doSearch();

};

$scope.clearDate = function () {

$scope.serviceDate = null;

doSearch();

};

$scope.getStartTime = function (appointment) {

return appointment.AppointmentServices[0].StartTimeString;

};

$scope.getServicesString = function (appointment) {

var services = \_.map(appointment.AppointmentServices, function (aptSvc) {

return aptSvc.Service.Name;

});

return \_.join(services, ', ');

};

$scope.getProvidersString = function (appointment) {

var rooms = \_.map(appointment.AppointmentServices, function (aptSvc) {

return aptSvc.Provider.FirstName + ' ' + aptSvc.Provider.LastName;

});

return \_.join(rooms, ', ');

};

$scope.getRoomsString = function (appointment) {

var rooms = \_.map(appointment.AppointmentServices, function (aptSvc) {

return aptSvc.Room.Name;

});

return \_.join(rooms, ', ');

};

$scope.getTotalTime = function (appointment) {

var total = 0;

\_.each(appointment.AppointmentServices, function (apsSvc) {

total += apsSvc.Service.Minutes;

});

return total

};

$scope.getTotalPrice = function (appointment) {

var total = 0;

\_.each(appointment.AppointmentServices, function (apsSvc) {

total += apsSvc.Service.Cost;

});

return total

};

function doSearch() {

$scope.isLoading = true;

if ($scope.isUser) {

appointmentService.getAppointmentsForUser(settings.User.Id, $scope.serviceDate).then(function (appointments) {

$scope.appointments = \_.uniqBy(appointments, function (apt) {

return apt.Id;

});;

$scope.isLoading = false;

});

} else {

appointmentService.getAppointmentsForClinic($scope.selectedClinic.Id, $scope.serviceDate).then(function (appointments) {

$scope.appointments = \_.uniqBy(appointments, function (apt) {

return apt.Id;

});

$scope.isLoading = false;

});

}

}

$scope.openAppointment = function (origAppointment) {

var modal = $uibModal.open({

templateUrl: \_baseUrl + 'app/home/appointment/manage/newAppointment.html',

controller: 'NewAppointmentCtrl',

size: 'appointment-modal-size',

resolve: {

params: function () {

return {

appointment: origAppointment,

clinic: $scope.selectedClinic,

submit: function (appointment) {

return appointmentService.createAppointment(appointment).then(function (appointment) {

if (appointment.Id) {

$scope.appointments.push(appointment);

modal.close();

}

});

},

close: function () {

modal.close();

}

};

}

}

});

};

$scope.delete = function (appointment) {

$scope.isLoading = true;

appointmentService.deleteAppointment(appointment).then(function () {

\_.remove($scope.appointments, appointment);

$scope.isLoading = false;

});

};

init();

}

} ((<any>window).angular));

#### billing.html

<home-nav></home-nav>

<div class="billing page-wrapper">

<div class="banner">

Billing

</div>

<div class="page-content">

hello

</div>

</div>

#### billing.less

.billing {

}

#### BillingCtrl.ts

(function (angular) {

'use strict';

angular.module('clinic')

.controller('BillingCtrl', BillingCtrl);

function BillingCtrl($scope) {

}

} ((<any>window).angular));

#### provider.html

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

<h4 class="modal-title" id="myModalLabel">Provider</h4>

</div>

<div class="modal-body">

<div ng-show="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div ng-if="!isLoading">

<label class="required">Staff to designate as Provider</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**getProviderMessage()**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="user **in** users" ng-click="setUser(user)"><a>**{{**user.FirstName**}}**&nbsp;**{{**user.LastName**}}**</a></li>

</ul>

</div>

<br />

<small>Provider not in the list? Go to "People" to create one.</small>

</div>

<button class="btn btn-danger" ng-click="close()">Cancel</button>

<button class="btn btn-success" ng-disabled="isLoading || !selectedUser" ng-click="save()">Save</button>

</div>

#### provider.less

.provider {

}

#### ProviderCtrl.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('ProviderCtrl', ProviderCtrl);

function ProviderCtrl($scope, $uibModalInstance, params, clinicService) {

$scope.isEdit = params.provider && params.provider.Id ? true : false;

$scope.selectedUser = null;

$scope.isLoading = true;

$scope.users = [];

function init() {

clinicService.getEligibleProviders(params.clinic.Id).then(function (users) {

$scope.users = users;

if (params.provider) {

\_.each(users, function (user) {

if (user.EntityId === params.provider.EntityId) {

$scope.selectedUser = \_.cloneDeep(user);

}

});

}

\_.each(params.clinic.Providers, function (provider) {

\_.remove($scope.users, function (user) {

return provider.EntityId === user.EntityId;

});

});

$scope.isLoading = false;

});

}

$scope.getProviderMessage = function () {

return $scope.selectedUser ? ($scope.selectedUser.FirstName + ' ' + $scope.selectedUser.LastName) : 'Select a staff member...';

};

$scope.setUser = function (user) {

$scope.selectedUser = user;

};

$scope.close = function () {

$uibModalInstance.close();

};

$scope.save = function () {

params.submit($scope.selectedUser);

$scope.close();

};

init();

}

} ((<any>window).angular));

#### room.html

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

<h4 class="modal-title" id="myModalLabel">Room</h4>

</div>

<div class="modal-body">

<ng-form name="roomForm">

<div class="form-group">

<label class="required">Room Name</label>

<input type="text" ng-model="Name" placeholder="name" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

</ng-form>

<hr />

<button class="btn btn-danger" ng-click="close()">Cancel</button>

<button class="btn btn-success" ng-disabled="!isValid()" ng-click="save()">Save</button>

</div>

#### room.less

.room {

}

#### RoomCtrl.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('RoomCtrl', RoomCtrl);

function RoomCtrl($scope, $uibModalInstance, params) {

$scope.Name = '';

function init() {

if (params.room) {

$scope.Name = params.room.Name

}

}

$scope.isValid = function () {

return $scope.Name.length;

};

$scope.close = function () {

$uibModalInstance.close();

};

$scope.save = function () {

params.submit({ Name: $scope.Name });

$scope.close();

};

init();

}

} ((<any>window).angular));

#### service.html

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

<h4 class="modal-title" id="myModalLabel">Service</h4>

</div>

<div class="modal-body">

<div ng-show="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div ng-if="!isLoading && !shouldCreate()">

<label class="required">Service to Add</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**getServiceMessage()**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="service **in** services" ng-click="setService(service)"><a>**{{**service.Name**}}**</a></li>

</ul>

</div>

<hr />

<span>OR</span>

<hr />

<button class="btn btn-primary btn-sm" ng-click="startCreateNew()">Create New Service</button>

</div>

<ng-form ng-if="!isLoading" name="serviceForm" ng-show="shouldCreate()">

<div class="form-group">

<label class="required">Service Name</label>

<input type="text" ng-model="selectedService.Name" placeholder="name" class="form-control" ng-maxlength="50" ng-required="true" />

<label class="required">Cost</label>

<input type="number" ng-model="selectedService.Cost" placeholder="$" class="form-control" ng-maxlength="50" ng-required="true" />

<label class="required">Duration</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**getDurationMessage()**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="duration **in** durations" ng-click="setDuration(duration)"><a>**{{**duration**}}**</a></li>

</ul>

</div>

</div>

</ng-form>

<hr />

<button class="btn btn-danger" ng-click="close()">Cancel</button>

<button class="btn btn-success" ng-disabled="isLoading || !selectedService || !isValid()" ng-click="save()">Save</button>

</div>

#### service.less

.service {

}

#### ServiceCtrl.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('ServiceCtrl', ServiceCtrl);

function ServiceCtrl($scope, $uibModalInstance, params, clinicService) {

$scope.isEdit = params.service ? true : false;

$scope.isCreating = false;

$scope.selectedService = {};

$scope.isLoading = true;

$scope.services = [];

$scope.durations = [30, 60, 90, 120, 150, 180];

$scope.selectedService.Minutes = 30;

function init() {

clinicService.getAllServices().then(function (services) {

$scope.services = services;

if (!params.service) {

\_.each(params.clinic.Services, function (clinicService) {

\_.remove($scope.services, function (service) {

return clinicService.Name === service.Name;

});

});

} else {

$scope.selectedService = \_.cloneDeep(params.service);

}

$scope.isLoading = false;

});

}

$scope.shouldCreate = function () {

return $scope.isEdit || $scope.isCreating || !$scope.services.length;

};

$scope.startCreateNew = function () {

$scope.isCreating = true;

};

$scope.getServiceMessage = function () {

return $scope.selectedService ? ($scope.selectedService.Name) : 'Select a service...';

};

$scope.getDurationMessage = function () {

return $scope.selectedService ? ($scope.selectedService.Minutes) : 'Select duration...';

};

$scope.isValid = function () {

return (($scope.isEdit || $scope.isCreating) && $scope.selectedService.Name && $scope.selectedService.Cost && $scope.selectedService.Minutes) ||

(!$scope.isEdit && $scope.selectedService);

};

$scope.setService = function (service) {

$scope.selectedService = service;

};

$scope.setDuration = function (duration) {

$scope.selectedService.Minutes = duration;

};

$scope.close = function () {

$uibModalInstance.close();

};

$scope.save = function () {

params.submit($scope.selectedService);

$scope.close();

};

init();

}

} ((<any>window).angular));

#### clinics.html

<home-nav></home-nav>

<div class="clinics page-wrapper">

<div class="banner">

Manage Clinics

</div>

<div class="sub-banner">

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**selectedClinicMessage()**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="clinic **in** clinics" ng-click="setClinic(clinic)"><a>**{{**clinic.Name**}}**</a></li>

</ul>

</div>

<button class="btn btn-default btn-sm" ng-click="addNewClinic()">Add New</button>

<button class="btn btn-danger btn-sm right" ng-if="false && selectedClinic && selectedClinic.Id" ng-click="deleteClinic()">Delete Clinic</button>

<button class="btn btn-success btn-sm right" ng-if="selectedClinic" ng-disabled="!isValid() || !isModified()" ng-click="saveClinic()">Save Clinic</button>

</div>

<div class="page-content">

<div class="page-content-left" ng-if="selectedClinic">

<div class="page-content-left-tab" ng-repeat="tab **in** tabs" ng-click="setTab(tab)" ng-class="{ 'active': tab.id === curTab.id }" ng-if="selectedClinic">

<i class="fa fa-minus"></i>

<span>**{{**tab.title**}}**</span>

<i ng-if="isFormError(tab)" class="fa fa-exclamation-triangle"></i>

</div>

</div>

<div ng-show="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div ng-show="!isLoading && selectedClinic" class="page-content-right">

<ng-form ng-show="curTab.id === tabs[0].id" name="clinicForm">

<div class="form-group">

<label class="required">Clinic</label>

<input type="text" ng-model="selectedClinic.Name" placeholder="Name" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Address 1</label>

<input type="text" ng-model="selectedClinic.Address1" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label>Address 2</label>

<input type="text" ng-model="selectedClinic.Address2" class="form-control" ng-maxlength="50" />

</div>

<div class="form-group">

<label class="required">City</label>

<input type="text" ng-model="selectedClinic.City" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label class="required">State</label>

<input type="text" ng-model="selectedClinic.State" class="form-control" ng-minlength="2" ng-maxlength="2" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Postal Code</label>

<input type="text" ng-model="selectedClinic.Zip" class="form-control" ng-maxlength="10" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Phone 1</label>

<input type="text" ng-model="selectedClinic.Phone1" class="form-control" ng-maxlength="15" ng-required="true" />

</div>

<div class="form-group">

<label>Phone 2</label>

<input type="text" ng-model="selectedClinic.Phone2" class="form-control" ng-maxlength="15" />

</div>

<div class="form-group">

<label>Phone 3</label>

<input type="text" ng-model="selectedClinic.Phone3" class="form-control" ng-maxlength="15" />

</div>

</ng-form>

<ng-form ng-if="curTab.id !== tabs[0].id">

<table class="table table-striped">

<thead>

<tr>

<th ng-repeat="col **in** curTab.columns">**{{**col.colName**}}**</th>

<th>Edit</th>

<th>Delete</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="row **in** dataToDisplay">

<td ng-repeat="col **in** curTab.columns"><span>**{{**getValue(row, col)**}}**</span></td>

<td class="table-btn">

<span class="fa-stack" ng-click="edit(row)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-pencil-square-o fa-stack-1x fa-inverse"></i>

</span>

</td>

<td class="table-btn red">

<span class="fa-stack" ng-click="delete(row)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-times fa-stack-1x fa-inverse"></i>

</span>

</td>

</tr>

</tbody>

</table>

<span ng-if="!dataToDisplay.length">None found. Add least one required.</span>

<hr />

<button class="btn btn-primary" ng-click="edit()">**{{**curTab.button**}}**</button>

</ng-form>

</div>

</div>

</div>

#### clinics.less

.clinics {

.sub-banner {

.dropdown {

width: 250px;

display: inline-block;

}

.btn {

display: inline-block;

&.right {

margin-left: 15px;

float: right;

}

}

}

}

#### ClinicsCtrl.ts

var \_;

var \_baseUrl;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('ClinicsCtrl', ClinicsCtrl);

function ClinicsCtrl($scope, $uibModal, $q, clinicService) {

$scope.isLoading = true;

$scope.dataToDisplay = [];

$scope.clinics = [];

$scope.selectedClinic = null;

$scope.tabs = [

{

title: 'Name', id: 'name'

},

{

title: 'Providers', id: 'providers', button: 'Add Provider',

columns: [

{ colName: 'First Name', valueName: 'FirstName' },

{ colName: 'Middle Name', valueName: 'MiddleName' },

{ colName: 'Last Name', valueName: 'LastName' },

{ colName: 'Username', valueName: 'Username' },

{ colName: 'Id', valueName: 'Id' }

]

},

{

title: 'Services', id: 'services', button: 'Add Service',

columns: [

{ colName: 'Service', valueName: 'Name' },

{ colName: 'Base Price', valueName: 'Cost' },

{ colName: 'Duration', valueName: 'Minutes' }

]

},

{

title: 'Rooms', id: 'rooms', button: 'Add Room',

columns: [

{ colName: 'Name', valueName: 'Name' }

]

}

];

function init() {

// load clinics

clinicService.getAll().then(function (clinics) {

if (!clinics || !clinics.length) {

$scope.isLoading = false;

return;

}

$scope.clinics = clinics;

$scope.selectedClinic = clinics[0];

$scope.setTab($scope.tabs[0]);

$scope.isLoading = false;

});

}

$scope.selectedClinicMessage = function () {

return $scope.selectedClinic ? $scope.selectedClinic.Name : 'Select a clinic...';

};

$scope.getValue = function (row, column) {

return row[column.valueName];

};

$scope.setTab = function (tab) {

$scope.curTab = tab;

$scope.dataToDisplay = $scope.curTab === $scope.tabs[1]

? $scope.selectedClinic.Providers

: $scope.curTab === $scope.tabs[2]

? $scope.selectedClinic.Services

: $scope.curTab === $scope.tabs[3]

? $scope.selectedClinic.Rooms

: [];

};

$scope.setClinic = function (clinic) {

$scope.selectedClinic = clinic;

$scope.selectedClinicOriginal = \_.cloneDeep(clinic);

$scope.setTab($scope.tabs[0]);

}

$scope.addNewClinic = function (obj) {

var newClinic = {

Providers: [],

Services: [],

Rooms: []

};

$scope.setClinic(newClinic);

$scope.setTab($scope.tabs[0]);

};

$scope.isValid = function () {

return $scope.clinicForm.$valid &&

$scope.selectedClinic.Providers &&

$scope.selectedClinic.Providers.length &&

$scope.selectedClinic.Services &&

$scope.selectedClinic.Services.length &&

$scope.selectedClinic.Rooms &&

$scope.selectedClinic.Rooms.length;

};

$scope.isModified = function () {

return !\_.isEqual($scope.selectedClinic, $scope.selectedClinicOriginal);

};

$scope.isFormError = function (tab) {

return (tab === $scope.tabs[0] && !$scope.selectedClinic.Name) ||

(tab === $scope.tabs[1] && (!$scope.selectedClinic.Providers || !$scope.selectedClinic.Providers.length)) ||

(tab === $scope.tabs[2] && (!$scope.selectedClinic.Services || !$scope.selectedClinic.Services.length)) ||

(tab === $scope.tabs[3] && (!$scope.selectedClinic.Rooms || !$scope.selectedClinic.Rooms.length));

};

$scope.saveClinic = function () {

$scope.isLoading = true;

clinicService.update($scope.selectedClinic).then(function (result) {

init();

});

};

$scope.deleteClinic = function () {

// confirm

$scope.isLoading = true;

clinicService.deleteClinic($scope.selectedClinic.Id).then(function () {

init();

$scope.isLoading = false;

});

};

$scope.delete = function (obj) {

var property = $scope.curTab === $scope.tabs[1]

? 'Provider'

: $scope.curTab === $scope.tabs[2]

? 'Service'

: 'Room';

var clinic = $scope.selectedClinic;

\_.remove(clinic[property + 's'], function (curObj) {

return obj === curObj;

});

};

$scope.edit = function (obj) {

$scope.curTab === $scope.tabs[1]

? openProvider(obj)

: $scope.curTab === $scope.tabs[2]

? openService(obj)

: openRoom(obj);

}

function openProvider(existingProvider) {

$uibModal.open({

templateUrl: \_baseUrl + 'app/home/clinics/provider/provider.html',

controller: 'ProviderCtrl',

resolve: {

params: function () {

return {

provider: existingProvider,

clinic: $scope.selectedClinic,

submit: function (provider) {

if (existingProvider) {

existingProvider.Id = provider.Id;

existingProvider.EntityId = provider.EntityId;

existingProvider.FirstName = provider.FirstName;

existingProvider.MiddleName = provider.MiddleName;

existingProvider.LastName = provider.LastName;

existingProvider.Username = provider.Username;

} else {

$scope.dataToDisplay.push({

Id: null,

EntityId: provider.EntityId,

FirstName: provider.FirstName,

MiddleName: provider.MiddleName,

LastName: provider.LastName,

Username: provider.Username

});

}

}

}

}

}

});

}

function openService(existingService) {

$uibModal.open({

templateUrl: \_baseUrl + 'app/home/clinics/service/service.html',

controller: 'ServiceCtrl',

resolve: {

params: function () {

return {

service: existingService,

clinic: $scope.selectedClinic,

submit: function (service) {

if (existingService) {

existingService.Id = service.Id;

existingService.Name = service.Name;

existingService.Cost = service.Cost;

existingService.Minutes = service.Minutes;

} else {

$scope.dataToDisplay.push({

Id: null,

Name: service.Name,

Cost: service.Cost,

Minutes: service.Minutes

});

}

}

}

}

}

});

}

function openRoom(existingRoom) {

$uibModal.open({

templateUrl: \_baseUrl + 'app/home/clinics/room/room.html',

controller: 'RoomCtrl',

resolve: {

params: function () {

return {

room: existingRoom,

clinic: $scope.selectedClinic,

submit: function (room) {

if (existingRoom) {

existingRoom.Name = room.Name;

} else {

$scope.dataToDisplay.push({

Id: null,

Name: room.Name

});

}

}

}

}

}

});

}

init();

}

} ((<any>window).angular));

#### people.html

<home-nav></home-nav>

<div class="profile page-wrapper">

<div class="banner">

Manage People

</div>

<div class="page-content">

<div class="page-content-left">

<div class="page-content-left-tab" ng-repeat="tab **in** tabs" ng-click="setTab(tab)" ng-class="{ 'active': tab === curTab }">

<i class="fa fa-minus"></i>

<span>**{{**tab.title**}}**</span>

</div>

</div>

<div ng-show="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div ng-show="!isLoading" class="page-content-right">

<table class="table table-striped">

<thead>

<tr>

<th>First Name</th>

<th>Middle Name</th>

<th>Last Name</th>

<th>Username</th>

<th>ID</th>

<th>Role</th>

<th>Edit</th>

<th>Delete</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="user **in** users">

<td><span>**{{**(user.FirstName ? user.FirstName : 'N/A')**}}**</span></td>

<td><span>**{{**(user.MiddleName ? user.MiddleName : 'N/A')**}}**</span></td>

<td><span>**{{**(user.LastName ? user.LastName : 'N/A')**}}**</span></td>

<td><span>**{{**user.Username**}}**</span></td>

<td><span>**{{**user.Id**}}**</span></td>

<td><span>**{{**user.Role.Name**}}**</span></td>

<td class="table-btn">

<span class="fa-stack" ng-click="edit(user)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-pencil-square-o fa-stack-1x fa-inverse"></i>

</span>

</td>

<td class="table-btn red">

<span class="fa-stack" ng-click="delete(user)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-times fa-stack-1x fa-inverse"></i>

</span>

</td>

</tr>

</tbody>

</table>

<span ng-if="!users.length">None found. Add one below.</span>

<hr />

<button class="btn btn-primary" ng-click="edit()">Create Profile</button>

</div>

</div>

</div>

#### people.less

.people {

}

#### PeopleCtrl.ts

var \_baseUrl;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('PeopleCtrl', PeopleCtrl);

function PeopleCtrl($scope, userService, $uibModal, $q, settings, $timeout) {

$scope.users = []

$scope.isLoading = false;

$scope.tabs = [];

$scope.curTab = {};

function init() {

if (settings.User.Role.Name === "Administrator") {

$scope.tabs.push({ title: 'Manage Staff', id: 'staff' });

}

$scope.tabs.push({ title: 'Manage Clients', id: 'clients' });

$scope.setTab($scope.tabs[0]);

}

$scope.setTab = function (tab) {

$scope.curTab = tab;

if ($scope.curTab.id === 'staff') {

$scope.getStaff();

} else {

$scope.getClients();

}

};

$scope.getStaff = function () {

$scope.isLoading = true;

userService.getStaff().then(function (data) {

$scope.users = data;

$scope.isLoading = false;

});

};

$scope.getClients = function () {

$scope.isLoading = true;

userService.getClients().then(function (data) {

$scope.users = data;

$scope.isLoading = false;

});

};

$scope.edit = function (user) {

settings.temp = {

profile: {

isHousehold: $scope.curTab === $scope.tabs[1],

user: user ? user : {},

close: function () {

$scope.setTab($scope.curTab);

modal.close();

}

}

};

var modal = $uibModal.open({

templateUrl: \_baseUrl + 'app/home/profile/profileModal.html',

controller: 'ProfileModalCtrl',

size: 'profile-modal-size'

});

};

$scope.delete = function (user) {

alert('todo');

}

init();

}

} ((<any>window).angular));

#### profile.html

<div class="profile">

<home-nav ng-if="!isModalEdit"></home-nav>

<div class="profile page-wrapper">

<div class="banner">

<span ng-if="!isModalEdit">Manage Profile</span>

<span ng-if="isModalEdit">Profile</span>

<button class="btn btn-success" ng-click="update()" ng-disabled="isLoading || !isValid() || !isDirty()">

Save

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</button>

</div>

<div class="page-content">

<div class="page-content-left">

<div class="page-content-left-tab" ng-repeat="tab in tabs" ng-click="setTab(tab)" ng-class="{ 'active': tab === curTab }">

<i class="fa fa-minus"></i>

<span>{{tab.title}}</span>

<i ng-if="isFormError(tab)" class="fa fa-exclamation-triangle"></i>

</div>

</div>

<div ng-show="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div ng-show="!isLoading" class="page-content-right is-modal">

<ng-form ng-show="curTab.id === 'user'" name="userForm">

<div class="form-group">

<label class="required">Username</label>

<input type="text" ng-model="user.Username" placeholder="user" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group required">

<label class="required">Password</label>

<input type="password" ng-model="user.Password" placeholder="enter new password..." class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div>

<label class="required">Role</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown" ng-disabled="!shouldGatherRole()">

<span class="message">{{getRoleMessage()}}</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="role in roles" ng-click="setRole(role)"><a>{{role.Name}}</a></li>

</ul>

</div>

</div>

</ng-form>

<ng-form ng-show="curTab.id === 'name'" name="nameForm">

<div class="form-group">

<label class="required">First Name</label>

<input type="text" ng-model="model.FirstName" placeholder="first" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label>Middle Name</label>

<input type="text" ng-model="model.MiddleName" placeholder="middle" class="form-control" ng-maxlength="50" />

</div>

<div class="form-group">

<label class="required">Last Name</label>

<input type="text" ng-model="model.LastName" placeholder="last" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div ng-if="isHousehold" class="form-group">

<label class="required">Date of Birth</label>

<input type="date" ng-model="householdPerson.DateOfBirth" class="form-control" placeholder="MM/dd/yyyy" max="{{getToday()}}" required />

</div>

<div ng-if="isHousehold">

<label class="required">Relationship</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown" ng-disabled="true">

<span class="message">{{getRelationshipMessage(householdPerson)}}</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="relationship in relationships" ng-click="setRelationship(relationship)"><a>{{relationship.Name}}</a></li>

</ul>

</div>

</div>

</ng-form>

<ng-form ng-show="curTab.id === 'address'" name="addressForm">

<div class="form-group">

<label class="required">Address 1</label>

<input type="text" ng-model="model.Address1" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label>Address 2</label>

<input type="text" ng-model="model.Address2" class="form-control" ng-maxlength="50" />

</div>

<div class="form-group">

<label class="required">City</label>

<input type="text" ng-model="model.City" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label class="required">State</label>

<input type="text" ng-model="model.State" class="form-control" ng-minlength="2" ng-maxlength="2" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Postal Code</label>

<input type="text" ng-model="model.Zip" class="form-control" ng-maxlength="10" ng-required="true" />

</div>

</ng-form>

<ng-form ng-show="curTab.id === 'phone'" name="phoneForm">

<div class="form-group">

<label class="required">Phone 1</label>

<input type="text" ng-model="model.Phone1" class="form-control" ng-maxlength="15" ng-required="true" />

</div>

<div class="form-group">

<label>Phone 2</label>

<input type="text" ng-model="model.Phone2" class="form-control" ng-maxlength="15" />

</div>

<div class="form-group">

<label>Phone 3</label>

<input type="text" ng-model="model.Phone3" class="form-control" ng-maxlength="15" />

</div>

</ng-form>

<ng-form ng-show="curTab.id === 'insurance'" name="insuranceForm">

<div class="form-group">

<label class="required">Company Name</label>

<input type="text" ng-model="household.InsuranceName" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Policy Number</label>

<input type="text" ng-model="household.PolicyNumber" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Group Number</label>

<input type="text" ng-model="household.GroupNumber" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

</ng-form>

<ng-form name="dependentsForm"></ng-form>

<ng-form ng-show="curTab.id === 'dependents'">

<div ng-show="!isAddingDependent">

<table class="table table-striped">

<thead>

<tr>

<th>First Name</th>

<th>Middle Name</th>

<th>Last Name</th>

<th>ID</th>

<th>Edit</th>

<th>Delete</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="person in household.People" ng-if="!person.IsPayer">

<td><span>{{(person.FirstName ? person.FirstName : 'N/A')}}</span></td>

<td><span>{{(person.MiddleName ? person.MiddleName : 'N/A')}}</span></td>

<td><span>{{(person.LastName ? person.LastName : 'N/A')}}</span></td>

<td><span>{{person.Id}}</span></td>

<td class="table-btn">

<span class="fa-stack" ng-click="editDependent(person)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-pencil-square-o fa-stack-1x fa-inverse"></i>

</span>

</td>

<td class="table-btn red">

<span class="fa-stack" ng-click="deleteDependent(person)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-times fa-stack-1x fa-inverse"></i>

</span>

</td>

</tr>

</tbody>

</table>

<span ng-if="(!household.People || household.People.length < 2)">None found. Add one below.</span>

<hr />

<button class="btn btn-primary" ng-click="editDependent({})">Add Dependent</button>

</div>

<ng-form ng-show="isAddingDependent" name="curDependentForm">

<button class="btn btn-primary" style="float: right; margin-left: 15px;" ng-click="saveDependent()" ng-disabled="curDependentForm.$invalid || !curDependent.Relationship">Save Dependent</button>

<button class="btn btn-default" style="float: right" ng-click="cancelEditDependent()">Cancel</button>

<br />

<div class="form-group">

<label class="required">First Name</label>

<input type="text" ng-model="curDependent.FirstName" placeholder="first" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label>Middle Name</label>

<input type="text" ng-model="curDependent.MiddleName" placeholder="middle" class="form-control" ng-maxlength="50" />

</div>

<div class="form-group">

<label class="required">Last Name</label>

<input type="text" ng-model="curDependent.LastName" placeholder="last" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Date of Birth</label>

<input type="date" ng-model="curDependent.DateOfBirth" class="form-control" placeholder="MM/dd/yyyy" max="{{getToday()}}" required />

</div>

<div>

<label class="required">Relationship</label>

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">{{getRelationshipMessage(curDependent)}}</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="relationship in relationships" ng-click="curDependent.Relationship = relationship"><a>{{relationship.Name}}</a></li>

</ul>

</div>

</div>

<br />

<div class="form-group">

<label class="required">Address 1</label>

<input type="text" ng-model="curDependent.Address1" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label>Address 2</label>

<input type="text" ng-model="curDependent.Address2" class="form-control" ng-maxlength="50" />

</div>

<div class="form-group">

<label class="required">City</label>

<input type="text" ng-model="curDependent.City" class="form-control" ng-maxlength="50" ng-required="true" />

</div>

<div class="form-group">

<label class="required">State</label>

<input type="text" ng-model="curDependent.State" class="form-control" ng-minlength="2" ng-maxlength="2" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Postal Code</label>

<input type="text" ng-model="curDependent.Zip" class="form-control" ng-maxlength="10" ng-required="true" />

</div>

<div class="form-group">

<label class="required">Phone 1</label>

<input type="text" ng-model="curDependent.Phone1" class="form-control" ng-maxlength="15" ng-required="true" />

</div>

<div class="form-group">

<label>Phone 2</label>

<input type="text" ng-model="curDependent.Phone2" class="form-control" ng-maxlength="15" />

</div>

<div class="form-group">

<label>Phone 3</label>

<input type="text" ng-model="curDependent.Phone3" class="form-control" ng-maxlength="15" />

</div>

</ng-form>

</ng-form>

</div>

</div>

</div>

</div>

#### profile.less

.profile {

height: 100%;

width: 100%;

}

#### ProfileCtrl.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('ProfileCtrl', ProfileCtrl);

function ProfileCtrl($q, $scope, settings, userService, householdService) {

$scope.settings = settings.temp.profile;

$scope.household = null;

$scope.householdPerson = {};

$scope.user = \_.cloneDeep($scope.settings ? $scope.settings.user : settings.User);

$scope.model = $scope.user;

$scope.isModalEdit = $scope.settings ? true : false;

$scope.isHousehold = (($scope.settings && $scope.settings.isHousehold) || (!$scope.settings && $scope.user.Role.Name === 'User')) ? true : false;

$scope.isLoading = true;

$scope.roles = [];

$scope.relationships = [];

$scope.originalDependents = [];

$scope.curDependent = null;

$scope.tabs = $scope.isHousehold

? [

{ title: 'Name', id: 'name' },

{ title: 'User/Pass', id: 'user' },

{ title: 'Address', id: 'address' },

{ title: 'Phone', id: 'phone' },

{ title: 'Insurance', id: 'insurance' },

{ title: 'Dependents', id: 'dependents' }

]

: [

{ title: 'Name', id: 'name' },

{ title: 'User/Pass', id: 'user' },

{ title: 'Address', id: 'address' },

{ title: 'Phone', id: 'phone' },

];

$scope.curTab = $scope.tabs[0];

function init() {

var rolesDfd = userService.getRoles();

var relationDfd = householdService.getRelationships();

var houseDfd = $q.defer();

if ($scope.isHousehold && $scope.user) {

householdService.getByUserId($scope.user.Id).then(function (household) {

houseDfd.resolve(household);

});

} else {

houseDfd.resolve();

$scope.model = $scope.user;

}

$q.all([rolesDfd, relationDfd, houseDfd.promise]).then(function (data) {

$scope.roles = data[0];

$scope.relationships = data[1];

$scope.household = data[2];

initUserInfo();

initHouseholdInfo();

$scope.isLoading = false;

});

}

function initUserInfo() {

if ($scope.isHousehold) {

$scope.user.Role = \_.find($scope.roles, function (role) { return role.Name === "User"; })

} else {

\_.remove($scope.roles, function (role) {

return role.Name === "User";

});

if ($scope.user && $scope.user.Role) {

$scope.user.Role = \_.find($scope.roles, function (role) { return role.Name === $scope.user.Role.Name; });

}

\_.merge($scope.model, $scope.user);

}

}

function initHouseholdInfo() {

if (!$scope.isHousehold) {

return;

}

if (!$scope.household) {

$scope.household = {};

}

if (!($scope.household.People && $scope.household.People.length)) {

$scope.household.People = [

{

IsPayer: true,

Relationship: \_.find($scope.relationships, function (relationship) {

return relationship.Name === "Primary";

})

}

]

}

$scope.dependents = \_.filter($scope.household.People, function (person) {

return person.IsPayer === false;

});

$scope.householdPerson = \_.find($scope.household.People, function (person) {

return person.IsPayer;

});

\_.each($scope.household.People, function (person) {

person.DateOfBirth = new Date($scope.household.People[0].DateOfBirthString);

});

$scope.model.FirstName = $scope.householdPerson.FirstName;

$scope.model.MiddleName = $scope.householdPerson.MiddleName;

$scope.model.LastName = $scope.householdPerson.LastName;

$scope.model.Address1 = $scope.householdPerson.Address1;

$scope.model.Address2 = $scope.householdPerson.Address2;

$scope.model.City = $scope.householdPerson.City;

$scope.model.State = $scope.householdPerson.State;

$scope.model.Zip = $scope.householdPerson.Zip;

$scope.model.Phone1 = $scope.householdPerson.Phone1;

$scope.model.Phone2 = $scope.householdPerson.Phone2;

$scope.model.Phone3 = $scope.householdPerson.Phone3;

$scope.originalDependents = \_.cloneDeep($scope.household.People);

}

$scope.getToday = function () {

var today = new Date();

var dd: any = today.getDate();

var mm: any = today.getMonth() + 1;

var yyyy = today.getFullYear();

if (dd < 10) {

dd = '0' + dd

}

if (mm < 10) {

mm = '0' + mm

}

return yyyy + '-' + mm + '-' + dd;

};

$scope.setTab = function (tab) {

$scope.curTab = tab;

};

$scope.isFormError = function (tab) {

return !$scope[tab.id + 'Form'].$valid;

};

$scope.setRole = function (role) {

$scope.user.Role = role;

};

$scope.getRoleMessage = function () {

return $scope.user && $scope.user.Role ? $scope.user.Role.Name : 'Select a role...';

};

$scope.setRelationship = function (relationship) {

$scope.householdPerson.Relationship = relationship;

};

$scope.shouldGatherRelationship = function () {

return $scope.settings.user ? true : false;

};

$scope.getRelationshipMessage = function (householdPerson) {

return householdPerson && householdPerson.Relationship ? householdPerson.Relationship.Name : 'Select a relationship...';

};

$scope.shouldGatherRole = function () {

return !$scope.isHousehold && $scope.settings && $scope.user.Id !== settings.User.Id;

};

$scope.cancelEditDependent = function () {

$scope.curDependent = null;

$scope.isAddingDependent = false;

};

$scope.editDependent = function (dependent) {

$scope.isAddingDependent = true;

$scope.curDependent = dependent;

if ($scope.householdPerson && !$scope.curDependent.FirstName) {

\_.merge($scope.curDependent, $scope.model);

$scope.curDependent.FirstName = '';

$scope.curDependent.MiddleName = '';

$scope.curDependent.LastName = '';

$scope.curDependent.DateOfBirth = null;

$scope.curDependent.Relationship = null;

$scope.curDependent.Id = null;

$scope.curDependent.EntityId = null;

}

}

$scope.saveDependent = function () {

$scope.household.People.push($scope.curDependent);

$scope.cancelEditDependent();

};

$scope.isDirty = function () {

var val = ($scope.userForm.$dirty ||

$scope.nameForm.$dirty ||

$scope.addressForm.$dirty ||

$scope.phoneForm.$dirty ||

$scope.insuranceForm.$dirty ||

($scope.isHousehold && !\_.isEqual($scope.originalDependents, $scope.household.People)));

return val;

};

$scope.isValid = function () {

var val = ($scope.userForm.$valid &&

$scope.nameForm.$valid &&

$scope.addressForm.$valid &&

$scope.phoneForm.$valid &&

(

!$scope.isHousehold ||

($scope.isHousehold && $scope.insuranceForm.$valid && $scope.household.People.length > 0) // one is the primary

));

return val;

};

$scope.update = function () {

$scope.isLoading = true;

\_.merge($scope.user, $scope.model);

userService.update($scope.user).then(function (user) {

$scope.user = user;

if ($scope.isHousehold) {

var personId = $scope.householdPerson.Id;

\_.merge($scope.householdPerson, $scope.model);

$scope.householdPerson.Id = personId;

$scope.householdPerson.EntityId = $scope.user.EntityId;

householdService.update($scope.household).then(function (household) {

$scope.household = household;

$scope.householdPerson = \_.find($scope.household.People, function (person) {

return person.IsPayer === true;

});

if ($scope.settings) {

$scope.settings.close();

} else {

init();

$scope.isLoading = false;

}

});

} else {

if ($scope.settings) {

$scope.settings.close();

} else {

init();

$scope.isLoading = false;

}

}

});

};

init();

}

} ((<any>window).angular));

#### profileDirective.ts

var \_baseUrl;

(function (angular) {

'use strict';

angular.module('clinic').directive('profile', profile);

function profile() {

return {

restrict: 'E',

replace: true,

scope: true,

templateUrl: \_baseUrl + 'app/home/profile/profile.html',

controller: 'ProfileCtrl'

};

};

}((<any>window).angular));

#### profileModal.html

<div class="profile-modal">

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

</div>

<div class="modal-body">

<profile></profile>

</div>

</div>

#### profileModal.less

.profile-modal {

.modal-body {

padding: 0 !important;

}

}

.modal-profile-modal-size {

width: 60% !important;

}

#### ProfileModalCtrl.ts

(function (angular) {

'use strict';

angular.module('clinic')

.controller('ProfileModalCtrl', ProfileModalCtrl);

function ProfileModalCtrl($scope, settings, $uibModalInstance) {

$scope.close = $uibModalInstance.close;

}

} ((<any>window).angular));

#### qualification.html

<div class="modal-header">

<button ng-click="close()" type="button" class="close"><span>&times;</span></button>

<h4 class="modal-title" id="myModalLabel">Room</h4>

</div>

<div class="modal-body">

<div ng-show="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div class="dropdown" ng-show="!isLoading">

<button class="btn btn-default dropdown-toggle" type="button" data-toggle="dropdown">

<span class="message">**{{**getServiceMessage()**}}**</span>

<span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li ng-repeat="service **in** services" ng-click="setService(service)"><a>**{{**service.Name**}}**</a></li>

</ul>

</div>

<hr />

<button class="btn btn-danger" ng-click="close()">Cancel</button>

<button class="btn btn-success" ng-disabled="!isValid()" ng-click="save()">Save</button>

</div>

#### qualification.less

.qualification {

}

#### QualificationCtrl.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('QualificationCtrl', QualificationCtrl);

function QualificationCtrl($scope, $uibModalInstance, params, userService, settings) {

$scope.selectedService = null;

$scope.services = [];

$scope.isLoading = true;

function init() {

userService.getEligibleQualifications(settings.User.Id).then(function (services) {

$scope.services = services;

$scope.isLoading = false;

});

}

$scope.getServiceMessage = function () {

return $scope.selectedService ? ($scope.selectedService.Name) : 'Select a service...';

};

$scope.isValid = function () {

return $scope.selectedService;

};

$scope.setService = function (service) {

$scope.selectedService = service;

};

$scope.close = function () {

$uibModalInstance.close();

};

$scope.save = function () {

params.submit($scope.selectedService);

$scope.close();

};

init();

}

} ((<any>window).angular));

#### qualifications.html

<home-nav></home-nav>

<div class="qualifications page-wrapper">

<div class="banner">

My Qualifications

<button class="btn btn-success" ng-click="update()" ng-disabled="isSaveDisabled()">

Save

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</button>

</div>

<div class="page-content" style="padding: 30px;">

<table class="table table-striped">

<thead>

<tr>

<th>Service Name</th>

<th></th>

</tr>

</thead>

<tbody>

<tr ng-repeat="service **in** services">

<td><span>**{{**service.Name**}}**</span></td>

<td class="table-btn red">

<span class="fa-stack" ng-click="delete(service)">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-times fa-stack-1x fa-inverse"></i>

</span>

</td>

</tr>

</tbody>

</table>

<span ng-if="!services.length">

None found. Add one below. <br /><br />

Note: Make sure you are a designated by the Admin as a Provider. <br />

Services you add here will be available to any Clinic you work in that provides that service.

</span>

<hr />

<button class="btn btn-primary" ng-click="addService()">Add Service</button>

</div>

</div>

#### qualifications.less

.qualifications {

}

#### QualificationsCtrl.ts

var \_baseUrl;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('QualificationsCtrl', QualificationsCtrl);

function QualificationsCtrl($scope, $uibModal, settings, userService) {

$scope.services = null;

$scope.originalServices = null;

$scope.isLoading = true;

function init() {

userService.getQualifications(settings.User.Id).then(function (qualifications) {

$scope.services = qualifications;

$scope.originalServices = \_.cloneDeep(qualifications);

$scope.isLoading = false;

});

}

$scope.isModified = function () {

return !\_.isEqual($scope.services, $scope.originalServices);

};

$scope.isSaveDisabled = function () {

return $scope.isLoading || !$scope.isModified();

};

$scope.addService = function () {

$uibModal.open({

templateUrl: \_baseUrl + 'app/home/qualifications/qualification/qualification.html',

controller: 'QualificationCtrl',

resolve: {

params: function () {

return {

submit: function (serviceToAdd) {

if (!\_.find($scope.services, function (service) { return service.Name === serviceToAdd.Name; })) {

$scope.services.push(serviceToAdd);

}

}

}

}

}

});

};

$scope.delete = function (delService) {

\_.remove($scope.services, function (service) { return service === delService });

};

$scope.update = function () {

$scope.isLoading = true;

userService.updateQualifications(settings.User.Id, $scope.services).then(function () {

$scope.isLoading = false;

});

};

init();

}

} ((<any>window).angular));

#### reports.html

<div class="reports">

<home-nav></home-nav>

<div class="page-wrapper">

<div class="banner">

<span>Generate Reports</span>

</div>

<div class="page-content">

<div class="page-content-left">

<div class="page-content-left-tab" ng-repeat="tab **in** tabs" ng-click="setTab(tab)" ng-class="{ 'active': tab === curTab }">

<i class="fa fa-minus"></i>

<span>**{{**tab.title**}}**</span>

</div>

</div>

<div ng-if="isLoading">

<i class='fa fa-spinner fa-spin' ng-if="isLoading"></i>

</div>

<div ng-if="!isLoading" class="page-content-right is-modal">

<div class="well well-lg"><h3>**{{**curTab.message**}}**</h3></div>

<div ng-repeat="input **in** curTab.requiredInputs" class="form-group">

<label class="required">**{{**input.title**}}**</label>

<input ng-if="input.type === 'date'" type="date" ng-model="input.value" class="form-control" placeholder="MM/dd/yyyy" ng-disabled="true" />

</div>

<div ng-if="curTab.requiredInputs">

<button class="btn btn-sm" ng-click="doSearch()">Search</button>

</div>

<table class="table table-striped">

<thead>

<tr>

<th ng-repeat="col **in** data[0]">**{{**stripUnder(col.Key)**}}**</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="row **in** data">

<td ng-repeat="element **in** row"><span>**{{**element.Value**}}**</span></td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

</div>

#### reports.less

.reports {

}

#### ReportsCtrl.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.controller('ReportsCtrl', ReportsCtrl);

function ReportsCtrl($scope, reportService) {

$scope.isLoading = true;

$scope.data = null;

function init() {

$scope.setTab($scope.tabs[0]);

}

$scope.tabs = [

{

title: 'Report 1', id: '1', getter: reportService.generateAllHouseholdReport,

message: 'List of all households with the household ID, name, address, and home phone along with the patient ID, name and relationship for each patient.'

},

{

title: 'Report 2', id: '2', getter: reportService.getHouseholdAndInsurance,

message: 'List of the insurance coverage for all households by household ID, household name, insurance company ID and company name.'

},

{

title: 'Report 3', id: '3', getter: reportService.getAllPatientsAndInsurance,

message: 'List all patients in alphabetical order by patient ID, name, and date of birth along with the name of the insurance company and policy number.'

},

{

title: 'Report 4', id: '4', getter: reportService.getAllBilling,

message: 'Itemized billings for all households with the household ID, household name, patient ID, patient name, service received, and the cost of the service.Show the output in alphabetical order by household name, patient name and billing date.'

},

{

title: 'Report 5', id: '5', getter: reportService.getHouseholdTotalCosts,

message: 'List the total cost of all services received for each household.'

},

{

title: 'Report 6', id: '6', getter: reportService.getProvidersAndServices,

message: 'List each provider with all services he or she is qualified to render.'

},

{

title: 'Report 7', id: '7', getter: reportService.getServicesAndProviders,

message: 'List each service available with all providers who are qualified to offer this service.'

},

{

title: 'Report 8', id: '8', getter: reportService.getFutureAppointmentsByPatient,

message: 'List all future appointments by name of patient, appointment date and time, estimated length of service, and contact home phone number.Dates and times should be in calendar order'

},

{

title: 'Report 9', id: '9', getter: reportService.getAllServicesProvided,

message: 'For a given date, list all services provided by each provider in alphabetical order by name of the provider.Show the service ID, service description and cost of service.'

},

{

title: 'Report 10', id: '10', getter: reportService.getTotalServicesForProviders,

requiredInputs: [{ title: 'ServiceDate', type: 'date', value: null }],

message: 'For a given date, list the total amount of services each provider rendered.Show in alphabetical order by the provider’s name.'

}

];

$scope.setTab = function (tab) {

$scope.isLoading = true;

$scope.curTab = tab;

if (tab.requiredInputs) {

$scope.isLoading = false;

return;

} else {

$scope.doSearch();

}

};

$scope.doSearch = function () {

$scope.isLoading = true;

var params = null;

if ($scope.curTab.requiredInputs) {

params = {};

\_.each($scope.curTab.requiredInputs, function (input) {

params[input.title] = input.value;

});

}

$scope.curTab.getter(params).then(function (response) {

$scope.data = response;

$scope.isLoading = false;

});

}

$scope.stripUnder = function (str) {

return \_.replace(str, '\_', ' ');

}

init();

}

} ((<any>window).angular));

#### home.html

<div ng-class="isNavMode ? 'home-nav' : 'home'">

<div class="tile" ng-repeat="tile **in** tiles" ng-style="{'background-color': tile.color}" ng-click="goTo(tile.route)" ng-class="getClass(tile)">

<div class="tile-inner" ng-mouseover="tile.open = true" ng-mouseleave="tile.open = false">

<div>

<span class="fa-stack fa-3x">

<i class="fa fa-circle fa-stack-2x"></i>

<i class="fa fa-stack-1x fa-inverse" ng-class="tile.icon"></i>

</span>

</div>

<div>

<span>**{{**tile.name**}}**</span>

</div>

</div>

</div>

</div>

#### home.less

.tile-mixin (@size, @margin, @padding, @fontSize) {

.tile {

width: @size;

height: @size;

text-align: center;

margin: @margin;

padding: @padding;

display: inline-block;

font-size: @fontSize;

font-weight: bold;

cursor: pointer;

border-radius: 5px;

&.active {

background-color: lightseagreen !important;

}

&:hover {

background-color: lightseagreen !important;

}

.tile-inner {

height: 100%;

width: 100%;

}

}

}

.home-nav {

width: 100%;

padding: 5px;

text-align: center;

border-bottom: 1px solid #e7e7e7;

background-color: rgba(23,97,142,0.1);

.tile-mixin(70px, 5px 25px, 5px, 0.6em);

}

.home {

width: 100%;

padding: 50px;

.tile-mixin(200px, 25px, 10px, 1.5em);

}

#### HomeCtrl.ts

(function (angular) {

'use strict';

angular.module('clinic')

.controller('HomeCtrl', HomeCtrl);

function HomeCtrl($scope, settings, homeTiles, $location, $route) {

$scope.tiles = homeTiles.getTiles();;

$scope.goTo = function (hash) {

$location.path(hash);

};

$scope.getClass = function (tile) {

return $route.current.$$route.originalPath === tile.route ? 'active' : '';

};

}

} ((<any>window).angular));

#### homeDirective.ts

var \_baseUrl;

(function (angular) {

'use strict';

angular.module('clinic').directive('homeNav', homeNav);

function homeNav() {

return {

restrict: 'E',

replace: true,

scope: true,

templateUrl: \_baseUrl + 'app/home/home.html',

controller: 'HomeCtrl',

link: function (scope, elem, attr) {

scope.isNavMode = true;

}

};

};

}((<any>window).angular));

#### homeTiles.ts

var \_: any;

(function (angular) {

'use strict';

angular.module('clinic')

.factory('homeTiles', homeTiles);

var admin = 'Administrator',

office = 'Office',

user = 'User';

function homeTiles($q, ajaxService, settings) {

var tiles = [

{

name: 'Appointments',

icon: 'fa-calendar',

color: '#6fb7ff',

route: '/home/appointment',

permissions: [user, office, admin]

},

//{

// name: 'Billing',

// icon: 'fa-dollar',

// color: '#6fb7ff',

// route: '/home/billing',

// permissions: [user, office, admin]

//},

{

name: 'Qualifications',

icon: 'fa-graduation-cap',

color: '#6fb7ff',

route: '/home/qualifications',

permissions: [office, admin]

},

{

name: 'Reports',

icon: 'fa-bar-chart',

color: '#6fb7ff',

route: '/home/reports',

permissions: [office, admin]

},

{

name: 'People',

icon: 'fa-users',

color: '#6fb7ff',

route: '/home/people',

permissions: [office, admin]

},

{

name: 'Clinics',

icon: 'fa-cubes',

color: '#6fb7ff',

route: '/home/clinics',

permissions: [admin]

},

{

name: 'Profile',

icon: 'fa-user',

color: '#6fb7ff',

route: '/home/profile',

permissions: [user, office, admin]

}

];

function getTiles() {

if (settings.User) {

return \_.filter(tiles, function (tile) {

return \_.find(tile.permissions, function (perm) {

return perm === settings.User.Role.Name;

});

});

}

return [];

}

return {

getTiles: getTiles

};

}

} ((<any>window).angular));

#### landing.html

<div>

<img style="width: 100%" src="Images/banner.jpg" />

</div>

#### LandingCtrl.ts

(function (angular) {

'use strict';

angular.module('clinic')

.controller('LandingCtrl', LandingCtrl);

function LandingCtrl($scope) {

}

} ((<any>window).angular));

#### settings.ts

var \_: any;

(function (angular) {

'use strict';

angular.module('clinic')

.factory('settings', settings);

function settings($q, ajaxService, $location) {

var settings: any = {};

settings.temp = {};

settings.getSettings = function(force = false) {

var dfd = $q.defer();

if (force || Object.keys(settings).length === 3) {

return ajaxService.post("GlobalSettings", "GetSettings").then(function (response) {

\_.merge(settings, response);

settings.isDev = $location.absUrl().indexOf('localhost') ? true : false;

dfd.resolve(settings);

});

} else {

dfd.resolve(settings);

}

return dfd.promise;

}

settings.clearTemp = function () {

if (settings) {

settings.temp = {};

}

};

settings.getSettings();

return settings;

}

} ((<any>window).angular));

#### ajaxService.ts

var \_, toastr;

(function (angular) {

'use strict';

angular.module('clinic')

.factory('ajaxService', ajaxService);

function ajaxService($q) {

var curUser;

function post(controller, method, data, blockToastErrors) {

var dfd = $q.defer();

$.ajax({

url: "Controllers/" + controller + "Controller.asmx/" + method,

type: "POST",

dataType: "json",

contentType: "application/json; charset=utf-8",

data: data ? angular.toJson(data) : '{}',

success: function (result) {

var response = result.d ? JSON.parse(result.d) : null;

if (response && response.errorMessages) {

if (!blockToastErrors) {

\_.each(response.errorMessages, function (message) {

toastr.error(message.Message, message.Field);

});

}

}

dfd.resolve(response);

},

error: function (e) {

dfd.reject(e);

}

});

return dfd.promise;

}

return {

post: post

};

}

} ((<any>window).angular));

#### appointmentService.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.factory('appointmentService', appointmentService);

function appointmentService($q, ajaxService, settings) {

function getAppointmentsForClinic(clinicId, date) {

var params = {

Id: clinicId,

ServiceDate: date

};

var dfd = $q.defer();

ajaxService.post("Appointment", "GetAppointmentsForClinic", params).then(function (results) {

dfd.resolve(results);

});

return dfd.promise;

}

function getAppointmentsForUser(userId, date) {

var params = {

Id: userId,

ServiceDate: date

};

var dfd = $q.defer();

ajaxService.post("Appointment", "GetAppointmentsForUser", params).then(function (results) {

dfd.resolve(results);

});

return dfd.promise;

}

function getAvailableAppointments(clinicId, serviceId, date) {

var params = {

ClinicId: clinicId,

ServiceId: serviceId,

ServiceDate: date

};

var dfd = $q.defer();

ajaxService.post("Appointment", "GetAvailableAppointments", params).then(function (results) {

dfd.resolve(results);

});

return dfd.promise;

}

function createAppointment(appointment) {

var appt = \_.clone(appointment);

appt.Person.DateOfBirth = new Date(appt.Person.DateOfBirthString);

\_.each(appt.AppointmentServices, function (svc) {

delete svc.Id;

});

var dfd = $q.defer();

ajaxService.post("Appointment", "Create", appt).then(function (result) {

dfd.resolve(result);

});

return dfd.promise;

}

function deleteAppointment(appointment) {

var appt = \_.clone(appointment);

appt.Person.DateOfBirth = new Date(appt.Person.DateOfBirthString);

\_.each(appt.AppointmentServices, function (svc) {

svc.StartTime = new Date(svc.StartTimeString);

});

var dfd = $q.defer();

ajaxService.post("Appointment", "Delete", appt).then(function (results) {

dfd.resolve(results);

});

return dfd.promise;

}

return {

getAppointmentsForClinic: getAppointmentsForClinic,

getAppointmentsForUser: getAppointmentsForUser,

getAvailableAppointments: getAvailableAppointments,

createAppointment: createAppointment,

deleteAppointment: deleteAppointment

};

}

} ((<any>window).angular));

#### clinicService.ts

(function (angular) {

'use strict';

angular.module('clinic')

.factory('clinicService', clinicService);

function clinicService($q, ajaxService, settings) {

function getAll() {

var dfd = $q.defer();

ajaxService.post("Clinic", "GetAll").then(function (clinics) {

dfd.resolve(clinics);

});

return dfd.promise;

}

function update(data) {

var dfd = $q.defer();

ajaxService.post("Clinic", "Update", data).then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function deleteClinic(id) {

var dfd = $q.defer();

ajaxService.post("Clinic", "Delete", { Id: id }).then(function () {

dfd.resolve();

});

return dfd.promise;

}

function getEligibleProviders(clinicId) {

var dfd = $q.defer();

ajaxService.post("Clinic", "GetEligibleProviders", { Id: clinicId }).then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getRooms(clinicId) {

var dfd = $q.defer();

ajaxService.post("Clinic", "GetRooms", { Id: clinicId }).then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getAllServices(clinicId) {

var dfd = $q.defer();

ajaxService.post("Clinic", "GetAllServices").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

return {

getAll: getAll,

update: update,

deleteClinic: deleteClinic,

getEligibleProviders: getEligibleProviders,

getRooms: getRooms,

getAllServices: getAllServices

};

}

} ((<any>window).angular));

#### householdService.ts

var \_;

(function (angular) {

'use strict';

angular.module('clinic')

.factory('householdService', householdService);

function householdService($q, ajaxService, settings) {

function get(id) {

var dfd = $q.defer();

ajaxService.post("Household", "Get", { Id: id }).then(function (household) {

dfd.resolve(household);

});

return dfd.promise;

}

function getByUserId(userId) {

var dfd = $q.defer();

ajaxService.post("Household", "GetByUserId", { UserId: userId }).then(function (household) {

dfd.resolve(household);

});

return dfd.promise;

}

function getByPayerName(params) {

var dfd = $q.defer();

ajaxService.post("Household", "GetByPayerName", params).then(function (results) {

dfd.resolve(results);

});

return dfd.promise;

}

function update(household) {

var dfd = $q.defer();

ajaxService.post("Household", "Update", household).then(function (input) {

dfd.resolve(household);

});

return dfd.promise;

}

function getRelationships(household) {

var dfd = $q.defer();

ajaxService.post("Household", "GetRelationships", household).then(function (household) {

dfd.resolve(household);

});

return dfd.promise;

}

return {

get: get,

getByUserId: getByUserId,

getByPayerName: getByPayerName,

update: update,

getRelationships: getRelationships

};

}

} ((<any>window).angular));

#### reportService.ts

(function (angular) {

'use strict';

angular.module('clinic')

.factory('reportService', reportService);

function reportService($q, ajaxService, settings) {

function generateAllHouseholdReport() {

var dfd = $q.defer();

ajaxService.post("Report", "GenerateAllHouseholdReport").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getHouseholdAndInsurance() {

var dfd = $q.defer();

ajaxService.post("Report", "GetHouseholdAndInsurance").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getAllPatientsAndInsurance() {

var dfd = $q.defer();

ajaxService.post("Report", "GetAllPatientsAndInsurance").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getAllBilling() {

var dfd = $q.defer();

ajaxService.post("Report", "GetAllBilling").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getHouseholdTotalCosts() {

var dfd = $q.defer();

ajaxService.post("Report", "GetHouseholdTotalCosts").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getProvidersAndServices() {

var dfd = $q.defer();

ajaxService.post("Report", "GetProvidersAndServices").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getServicesAndProviders() {

var dfd = $q.defer();

ajaxService.post("Report", "GetServicesAndProviders").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getFutureAppointmentsByPatient() {

var dfd = $q.defer();

ajaxService.post("Report", "GetFutureAppointmentsByPatient").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getAllServicesProvided() {

var dfd = $q.defer();

ajaxService.post("Report", "GetAllServicesProvided").then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

function getTotalServicesForProviders(serviceDate) {

var dfd = $q.defer();

ajaxService.post("Report", "GetTotalServicesForProviders", { Date: serviceDate }).then(function (response) {

dfd.resolve(response);

});

return dfd.promise;

}

return {

generateAllHouseholdReport: generateAllHouseholdReport,

getHouseholdAndInsurance: getHouseholdAndInsurance,

getAllPatientsAndInsurance: getAllPatientsAndInsurance,

getAllBilling: getAllBilling,

getHouseholdTotalCosts: getHouseholdTotalCosts,

getProvidersAndServices: getProvidersAndServices,

getServicesAndProviders: getServicesAndProviders,

getFutureAppointmentsByPatient: getFutureAppointmentsByPatient,

getAllServicesProvided: getAllServicesProvided,

getTotalServicesForProviders: getTotalServicesForProviders

};

}

} ((<any>window).angular));

#### userService.ts

(function (angular) {

'use strict';

angular.module('clinic')

.factory('userService', userService);

function userService($q, ajaxService, settings) {

function login(data) {

var dfd = $q.defer();

ajaxService.post("User", "Login", data).then(function (user) {

settings.getSettings(true).then(function () {

dfd.resolve();

});

});

return dfd.promise;

}

function logout() {

var dfd = $q.defer();

ajaxService.post("User", "Logout").then(function () {

settings.getSettings(true).then(function () {

dfd.resolve();

});

});

return dfd.promise;

}

function register(data) {

var dfd = $q.defer();

ajaxService.post("User", "Register", data).then(function (user) {

settings.getSettings(true).then(function () {

dfd.resolve();

});

});

return dfd.promise;

}

function update(data) {

var dfd = $q.defer();

ajaxService.post("User", "Update", data).then(function (user) {

settings.getSettings(true).then(function () {

dfd.resolve(user);

});

});

return dfd.promise;

}

function remove(data) {

var dfd = $q.defer();

ajaxService.post("User", "Remove", data).then(function (user) {

settings.getSettings(true).then(function () {

dfd.resolve(user);

});

});

return dfd.promise;

}

function getStaff() {

var dfd = $q.defer();

ajaxService.post("User", "GetStaff").then(function (data) {

dfd.resolve(data);

});

return dfd.promise;

}

function getClients() {

var dfd = $q.defer();

ajaxService.post("User", "GetClients").then(function (data) {

dfd.resolve(data);

});

return dfd.promise;

}

function getRoles() {

var dfd = $q.defer();

ajaxService.post("User", "GetRoles").then(function (data) {

dfd.resolve(data);

});

return dfd.promise;

}

function getEligibleQualifications(userId) {

var dfd = $q.defer();

ajaxService.post("User", "GetEligibleQualifications", { UserId: userId }).then(function (data) {

dfd.resolve(data);

});

return dfd.promise;

}

function getQualifications(userId) {

var dfd = $q.defer();

ajaxService.post("User", "GetQualifications", { UserId: userId }).then(function (data) {

dfd.resolve(data);

});

return dfd.promise;

}

function updateQualifications(userId, qualifications) {

var dfd = $q.defer();

ajaxService.post("User", "UpdateQualifications", { UserId: userId, Services: qualifications }).then(function (data) {

dfd.resolve(data);

});

return dfd.promise;

}

return {

login: login,

logout: logout,

register: register,

update: update,

remove: remove,

getStaff: getStaff,

getClients: getClients,

getRoles: getRoles,

getEligibleQualifications: getEligibleQualifications,

getQualifications: getQualifications,

updateQualifications: updateQualifications

};

}

} ((<any>window).angular));

#### app.ts

var \_baseUrl;

(function (angular) {

'use strict';

var clinic = angular.module('clinic', [

'ngRoute',

'ngAnimate',

'ngSanitize',

'ui.bootstrap',

'ui.router',

'ui.select'

]);

clinic.config(routeConfig);

function routeConfig($routeProvider: ng.route.IRouteProvider): void {

var base = 'adamfranzen71';

$routeProvider

.when('/', {

name: 'landing',

templateUrl: \_baseUrl + 'app/landing/landing.html',

controller: 'LandingCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home', {

name: 'home',

templateUrl: \_baseUrl + 'app/home/home.html',

controller: 'HomeCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home/appointment', {

name: 'appointment',

templateUrl: \_baseUrl + 'app/home/appointment/appointment.html',

controller: 'AppointmentCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home/billing', {

name: 'billing',

templateUrl: \_baseUrl + 'app/home/billing/billing.html',

controller: 'BillingCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home/people', {

name: 'people',

templateUrl: \_baseUrl + 'app/home/people/people.html',

controller: 'PeopleCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home/clinics', {

name: 'clinics',

templateUrl: \_baseUrl + 'app/home/clinics/clinics.html',

controller: 'ClinicsCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home/profile', {

name: 'profile',

templateUrl: \_baseUrl + 'app/home/profile/profile.html',

controller: 'ProfileCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home/qualifications', {

name: 'qualifications',

templateUrl: \_baseUrl + 'app/home/qualifications/qualifications.html',

controller: 'QualificationsCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home/services', {

name: 'services',

templateUrl: \_baseUrl + 'app/home/services/services.html',

controller: 'ServicesCtrl',

resolve: {

factory: checkRouting

}

})

.when('/home/reports', {

name: 'reports',

templateUrl: \_baseUrl + 'app/home/reports/reports.html',

controller: 'ReportsCtrl',

resolve: {

factory: checkRouting

}

})

.otherwise('/');

}

function checkRouting($q, settings, $route, $location) {

settings.clearTemp();

var dfd = $q.defer();

settings.getSettings(true).then(function () {

if (!settings.User && $route.current.$$route.name !== 'landing') {

$location.path('/');

} else if (settings.User && $route.current.$$route.name === 'landing') {

$location.path('/home');

}

dfd.resolve();

});

return dfd.promise;

}

} ((<any>window).angular));

#### site.less

@import url('../Content/font-awesome');

@import url('../Content/bootstrap/bootstrap');

@import url('../Content/toastr');

@import url('../app/home/home');

@import url('../app/home/appointment/appointment');

@import url('../app/home/billing/billing');

@import url('../app/home/people/people');

@import url('../app/home/clinics/clinics');

@import url('../app/home/clinics/provider/provider');

@import url('../app/home/clinics/room/room');

@import url('../app/home/clinics/service/service');

@import url('../app/home/profile/profile');

@import url('../app/home/profile/profileModal');

@import url('../app/home/qualifications/qualifications');

@import url('../app/home/qualifications/qualification/qualification');

@import url('../app/home/reports/reports');

@headerHeight: 50px;

@titleHeight: 50px;

@sidebarWidth: 200px;

@sidebarPadding: 20px;

body {

position: fixed;

top: 0;

bottom: 0;

left: 0;

right: 0;

background-color: white !important;

input.ng-invalid {

border: 1px #d9534f solid !important;

}

.main-view {

height: 100%;

width: 100%;

}

.modal-backdrop {

display: none !important;

}

.navbar {

margin-bottom: 0 !important;

background-color: white;

img {

height: 30px;

cursor: pointer;

}

a {

cursor: pointer;

}

}

.page-title {

padding: 5px;

position: absolute;

top: 0;

left: @sidebarPadding;

right: 0;

height: @titleHeight;

}

.panel-body {

padding: 0 !important;

}

.sidebar {

padding: @sidebarPadding 0;

position: absolute;

top: @titleHeight;

left: @sidebarPadding;

width: @sidebarWidth;

bottom: 0;

}

.sidebar-sibling {

padding: 0 0 @sidebarPadding @sidebarPadding;

margin: @sidebarPadding;

position: absolute;

top: @titleHeight;

left: @sidebarWidth;

right: 0;

bottom: 0;

}

.content {

position: fixed;

top: @headerHeight;

bottom: 0;

right: 0;

left: 0;

overflow-y: auto;

}

.borderless {

border: none !important;

border-color: transparent !important;

-webkit-box-shadow: none !important;

box-shadow: none !important;

}

.navbar .divider-vertical {

height: 50px;

margin: 0 9px;

border-right: 1px solid #ffffff;

border-left: 1px solid #f2f2f2;

}

.navbar-inverse .divider-vertical {

border-right-color: #222222;

border-left-color: #111111;

}

@media (max-width: 767px) {

.navbar-collapse .nav > .divider-vertical {

display: none;

}

}

.dropdown {

button {

width: 100%;

span.message {

float: left;

}

span.caret {

float: right;

}

}

}

label.required:after {

content: "\*";

color: red;

}

.table {

margin-bottom: 0 !important;

}

.table-empty {

padding: 10px;

}

.panel-footer.button {

cursor: pointer;

}

.page-wrapper {

height: 100%;

width: 100%;

.banner {

width: 100%;

text-transform: uppercase;

border-bottom: 1px solid #e7e7e7;

color: white;

font-weight: 500;

font-size: 24px;

padding: 10px;

background-color: rgba(0, 0, 0, 0.6);

.btn {

margin-right: 20px;

float: right;

width: 150px;

}

}

.sub-banner {

width: 100%;

padding: 5px;

border-bottom: 1px solid #e7e7e7;

background-color: rgba(23,97,142,0.1);

}

.page-content {

width: 100%;

height: 100%;

overflow-y: auto;

display: table;

.red {

color: #d9534f;

}

.table-btn {

cursor: pointer;

}

.page-content-left {

width: 250px;

height: 100%;

display: table-cell;

vertical-align: top;

background: rgba(0, 0, 0, 0.6);

.page-content-left-tab {

width: 100%;

border-bottom: 1px solid #e7e7e7;

text-transform: uppercase;

color: white;

padding: 10px;

cursor: pointer;

span {

font-size: 1.2em;

margin-left: 10px;

}

.fa-exclamation-triangle {

color: #d9534f;

float: right;

}

&.active {

background-color: lightseagreen !important;

}

&:hover {

background-color: lightseagreen !important;

}

}

}

.page-content-right {

height: calc(~"100% - 180px");

overflow-y: auto;

padding: 30px;

&.is-modal {

height: inherit !important;

overflow-y: inherit !important;

display: table-cell;

}

}

}

}

margin: 0;

background-color: #EDEFED;

}

### Business Layer

#### AppointmentBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class AppointmentBL

{

public List<Appointment> GetAppointmentsForClinic(int clinicId, DateTime? date)

{

return DataLayer.AppointmentDL.GetAppointmentsForClinic(clinicId, date)

.OrderBy(x => x.AppointmentServices[0].StartTime)

.ThenBy(x => x.Person.LastName)

.ThenBy(x => x.Person.FirstName)

.ToList();

}

public List<Appointment> GetAppointmentsForUser(int userId, DateTime? date)

{

return DataLayer.AppointmentDL.GetAppointmentsForUser(userId, date)

.OrderBy(x => x.AppointmentServices[0].StartTime)

.ThenBy(x => x.Person.LastName)

.ThenBy(x => x.Person.FirstName)

.ToList();

}

public void Create(Appointment appointment)

{

DataLayer.AppointmentDL.Create(appointment);

}

public void Delete(Appointment appointment)

{

DataLayer.AppointmentDL.Delete(appointment);

}

public List<AppointmentService> GetAvailableAppointments(int clinicId, int serviceId, DateTime date)

{

var desiredService = DataLayer.ServiceDL.Get(serviceId);

var consumedTimes = new List<DateTime>();

var allStartTimes = new List<DateTime>();

var minBetween = 30;

var openTime = new DateTime(date.Year, date.Month, date.Day).AddHours(8); // 1st appointment @ 8am

var curTime = openTime; // this is not a reference assignment - .NET creates a copy

var closeTime = openTime.AddHours(8 + 10); // Last appointment ends by 6pm

// making a list of half-hour blocked datetimes

while (curTime != closeTime)

{

allStartTimes.Add(curTime);

curTime = curTime.AddMinutes(minBetween);

}

var existingAptSvcs = DataLayer.AppointmentDL.GetExistingAppointmentServices(clinicId, date);

var takenAptSvcs = new List<AppointmentService>();

var allProviders = DataLayer.ProviderDL.GetProvidersByClinicId(clinicId);

var allRooms = DataLayer.RoomDL.GetRoomsByClinicId(clinicId);

// create a massive list of every possible combination of appointment time, provider, room, service in a given day

var availableAppointments = new List<AppointmentService>();

allStartTimes.ForEach(time =>

allProviders.ForEach(provider =>

allRooms.ForEach(room =>

availableAppointments.Add(new AppointmentService

{

Cost = desiredService.Cost,

Provider = provider,

Room = room,

Service = desiredService,

StartTime = time

})

)

)

);

// making takenAptSvcs a list of appointment times consumed for every half hour each existing appointment occupies a room

foreach (var time in allStartTimes)

{

var aptSvcsAtCurTime = existingAptSvcs.Where(x => x.StartTime == time).ToList();

foreach (var aptSvc in aptSvcsAtCurTime)

{

for (var curMinutes = 0; curMinutes < aptSvc.Service.Minutes; curMinutes += minBetween)

{

takenAptSvcs.Add(new AppointmentService

{

Id = aptSvc.Id,

Cost = aptSvc.Cost,

Provider = aptSvc.Provider,

Room = aptSvc.Room,

Service = aptSvc.Service,

StartTime = aptSvc.StartTime.AddMinutes(curMinutes)

});

}

}

}

// add times that will end after close of business to the taken list

availableAppointments.RemoveAll(avail =>

avail.StartTime.AddMinutes(avail.Service.Minutes) > closeTime

);

foreach (var takenAptSvc in takenAptSvcs)

{

DateTime takenEndTime = takenAptSvc.StartTime.AddMinutes(takenAptSvc.Service.Minutes);

availableAppointments.RemoveAll(avail =>

avail.Provider.Id == avail.Provider.Id &&

avail.Service.Id == avail.Service.Id &&

avail.Room.Id == avail.Room.Id &&

// if available timeslot starts or ends in the middle of a taken timeslot

(

( avail.StartTime > takenAptSvc.StartTime &&

avail.StartTime < takenEndTime )

||

( avail.StartTime.AddMinutes(avail.Service.Minutes) > takenAptSvc.StartTime &&

avail.StartTime.AddMinutes(avail.Service.Minutes) < takenEndTime )

)

);

}

return availableAppointments

.OrderBy(x => x.StartTime)

.ThenBy(x => x.IsQualified)

.ThenBy(x => x.Provider.FirstName)

.ThenBy(x => x.Provider.LastName)

.ThenBy(x => x.Room.Name)

.ToList();

}

}

}

#### BusinessLayer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public static class BusinessLayer

{

public static UserBL UserBL = new UserBL();

public static HouseholdBL HouseholdBL = new HouseholdBL();

public static PersonBL PersonBL = new PersonBL();

public static RoleBL RoleBL = new RoleBL();

public static EntityBL EntityBL = new EntityBL();

public static RelationshipBL RelationshipBL = new RelationshipBL();

public static ClinicBL ClinicBL = new ClinicBL();

public static ServiceBL ServiceBL = new ServiceBL();

public static RoomBL RoomBL = new RoomBL();

public static ReportBL ReportBL = new ReportBL();

public static AppointmentBL AppointmentBL = new AppointmentBL();

}

}

#### ClinicBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class ClinicBL

{

public Clinic.BO.Clinic Get(int id)

{

return DataLayer.ClinicDL.Get(id);

}

public List<Clinic.BO.Clinic> GetClinics()

{

return DataLayer.ClinicDL.GetClinics();

}

public void Delete(int id)

{

DataLayer.ClinicDL.Delete(id);

}

public List<User> GetEligibleProviders(int clinicId) {

var allStaff = BusinessLayer.UserBL.GetStaff();

if (clinicId > 0)

{

var eligibleUsers = new List<User>();

var clinic = Get(clinicId);

foreach (var potential in allStaff)

{

if (!clinic.Providers.Any(existing => existing.EntityId == potential.EntityId))

{

eligibleUsers.Add(potential);

}

}

return eligibleUsers;

}

else

{

return allStaff;

}

}

public void Update(Clinic.BO.Clinic clinic)

{

if (clinic.EntityId == null)

{

BusinessLayer.EntityBL.Create(clinic);

}

else

{

BusinessLayer.EntityBL.Update(clinic);

}

if (clinic.Id.HasValue)

{

DataLayer.ClinicDL.Update(clinic);

}

else

{

DataLayer.ClinicDL.Create(clinic);

}

var existingClinic = BusinessLayer.ClinicBL.Get(clinic.Id.Value);

var newProviders = clinic.Providers.Where(x => !existingClinic.Providers.Any(existing => existing.Id == x.Id)).ToList();

var removedProviders = existingClinic.Providers.Where(existing => !clinic.Providers.Any(cur => cur.Id.Value == existing.Id.Value)).ToList();

var newServices = clinic.Services.Where(x => !x.Id.HasValue).ToList();

var updatedServices = clinic.Services.Where(x => x.Id.HasValue).ToList();

var removedServices = existingClinic.Services.Where(existing => !updatedServices.Any(cur => cur.Id.Value == existing.Id.Value)).ToList();

var newRooms = clinic.Rooms.Where(x => !x.Id.HasValue).ToList();

var updatedRooms = clinic.Rooms.Where(x => x.Id.HasValue).ToList();

var removedRooms = existingClinic.Rooms.Where(existing => !updatedRooms.Any(cur => cur.Id.Value == existing.Id.Value)).ToList();

newProviders.ForEach(x => DataLayer.ProviderDL.Create(clinic.Id.Value, x));

removedProviders.ForEach(x => DataLayer.ProviderDL.Delete(clinic.Id.Value, x));

newServices.ForEach(x => DataLayer.ServiceDL.AddToClinic(x, clinic));

updatedServices.ForEach(x => DataLayer.ServiceDL.Update(x));

removedServices.ForEach(x => DataLayer.ServiceDL.DeleteFromClinic(x, clinic));

newRooms.ForEach(x => DataLayer.RoomDL.Create(clinic.Id.Value, x));

updatedRooms.ForEach(x => DataLayer.RoomDL.Update(x));

removedRooms.ForEach(x => DataLayer.RoomDL.Delete(x));

}

}

}

#### EntityBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class EntityBL

{

public Entity Get(int id)

{

return DataLayer.EntityDL.Get(id);

}

public void Create(Entity entity)

{

DataLayer.EntityDL.Create(entity);

}

public void Update(Entity entity)

{

DataLayer.EntityDL.Update(entity);

}

}

}

#### HouseholdBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class HouseholdBL

{

public Household Get(int id)

{

return DataLayer.HouseholdDL.Get(id);

}

public List<Household> GetAll() {

return DataLayer.HouseholdDL.GetAll();

}

public Household GetByUserId(int userId)

{

return DataLayer.HouseholdDL.GetByUserId(userId);

}

public List<Household> GetByPayerName(string firstName, string middleName, string lastName)

{

return DataLayer.HouseholdDL.GetByPayerName(firstName, middleName, lastName);

}

public void Create(Household household)

{

DataLayer.HouseholdDL.Create(household);

}

public void Update(Household household)

{

if (household.Id.HasValue)

{

DataLayer.HouseholdDL.Update(household);

}

else

{

DataLayer.HouseholdDL.Create(household);

}

household.People.ForEach(person =>

BusinessLayer.PersonBL.Update(household.Id.Value, person)

);

}

}

}

#### PersonBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class PersonBL

{

public Person GetByUserId(int userId)

{

return DataLayer.PersonDL.GetByUserId(userId);

}

public Person GetPayerByHouseholdId(int householdId)

{

return DataLayer.PersonDL.GetPayerByHouseholdId(householdId);

}

public void Update(int householdId, Person person)

{

if (person.EntityId == null)

{

BusinessLayer.EntityBL.Create(person);

}

else

{

BusinessLayer.EntityBL.Update(person);

}

if (!person.Id.HasValue)

{

DataLayer.PersonDL.Create(householdId, person);

}

else

{

DataLayer.PersonDL.Update(householdId, person);

}

}

}

}

#### RelationshipBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class RelationshipBL

{

public Relationship Get(int id)

{

return DataLayer.RelationshipDL.Get(id);

}

public Relationship Get(string name)

{

return DataLayer.RelationshipDL.Get(name);

}

public List<Relationship> GetRelationships() {

return DataLayer.RelationshipDL.GetRelationships();

}

}

}

#### ReportBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Dynamic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class ReportBL

{

/// <summary>

/// List all households with the household ID, name, address, and home phone along with the

/// patient ID, name and relationship for each patient.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GenerateAllHouseholdReport()

{

return DataLayer.ReportDL.GenerateAllHouseholdReport();

}

/// <summary>

/// List the insurance coverage for all households by household ID, household name, insurance

/// company ID and company name.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetHouseholdAndInsurance()

{

return DataLayer.ReportDL.GetHouseholdAndInsurance();

}

/// <summary>

/// List all patients in alphabetical order by patient ID, name, and date of birth along with the

/// name of the insurance company and policy number.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetAllPatientsAndInsurance()

{

return DataLayer.ReportDL.GetAllPatientsAndInsurance();

}

/// <summary>

/// Show itemized billings for all households with the household ID, household name, patient ID,

/// patient name, service received, and the cost of the service.Show the output in alphabetical

/// order by household name, patient name and billing date.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetAllBilling()

{

return DataLayer.ReportDL.GetAllBilling();

}

/// <summary>

/// List the total cost of all services received for each household.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetHouseholdTotalCosts()

{

return DataLayer.ReportDL.GetHouseholdTotalCosts();

}

/// <summary>

/// List each provider with all services he or she is qualified to render.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetProvidersAndServices()

{

return DataLayer.ReportDL.GetProvidersAndServices();

}

/// <summary>

/// List each service available with all providers who are qualified to offer this service.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetServicesAndProviders()

{

return DataLayer.ReportDL.GetServicesAndProviders();

}

/// <summary>

/// List all future appointments by name of patient, appointment date and time, estimated length

/// of service, and contact home phone number.Dates and times should be in calendar order

/// for each patient.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetFutureAppointmentsByPatient()

{

return DataLayer.ReportDL.GetFutureAppointmentsByPatient();

}

/// <summary>

/// For a given date, list all services provided by each provider in alphabetical order by name of

/// the provider.Show the service ID, service description and cost of service.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetAllServicesProvided()

{

return DataLayer.ReportDL.GetAllServicesProvided();

}

/// <summary>

/// For a given date, list the total amount of services each provider rendered.Show in

/// alphabetical order by the provider’s name.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetTotalServicesForProviders(DateTime serviceDate)

{

return DataLayer.ReportDL.GetTotalServicesForProviders(serviceDate);

}

}

}

#### RoleBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class RoleBL

{

public Role Get(int id)

{

return DataLayer.RoleDL.Get(id);

}

public Role Get(string name)

{

return DataLayer.RoleDL.Get(name);

}

public List<Role> GetRoles() {

return DataLayer.RoleDL.GetRoles();

}

}

}

#### RoomBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class RoomBL

{

public List<Room> GetRoomsByClinicId(int clinicId)

{

return DataLayer.RoomDL.GetRoomsByClinicId(clinicId);

}

}

}

#### ServiceBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class ServiceBL

{

public List<Service> Get()

{

return DataLayer.ServiceDL.Get();

}

public List<Service> GetEligibleProviderServicesForUserId(int userId)

{

return DataLayer.ServiceDL.GetEligibleProviderServicesForUserId(userId);

}

public List<Service> GetProviderServicesForUserId(int userId)

{

return DataLayer.ServiceDL.GetProviderServicesForUserId(userId);

}

public void UpdateProviderServicesForUserId(int userId, List<Service> services)

{

var existingServices = GetProviderServicesForUserId(userId);

var newServices = services.Where(x => !existingServices.Any(existing => existing.Id == x.Id)).ToList();

var deletedServices = existingServices.Where(existing => !services.Any(x => x.Id == existing.Id)).ToList();

var provider = DataLayer.ProviderDL.GetProviderByUserId(userId);

newServices.ForEach(x => DataLayer.ServiceDL.AddToProvider(x, provider));

deletedServices.ForEach(x => DataLayer.ServiceDL.DeleteFromProvider(x, provider));

}

}

}

#### UserBL.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BL

{

public class UserBL

{

public User Get(string userId, string password)

{

return DataLayer.UserDL.Get(userId, password);

}

public void Create(User user)

{

DataLayer.UserDL.Create(user);

}

public List<User> GetAdmins()

{

return DataLayer.UserDL.GetAdmins();

}

public List<User> GetClients()

{

return DataLayer.UserDL.GetClients();

}

public List<User> GetStaff()

{

return DataLayer.UserDL.GetStaff();

}

/// <summary>

/// Household should already exist if creating user this way

/// </summary>

/// <param name="householdId"></param>

/// <param name="user"></param>

public void Create(int householdId, User user)

{

var person = DataLayer.PersonDL.GetPayerByHouseholdId(householdId);

if (!person.IsEmpty())

{

var newUser = (User)((Entity)person).Copy();

newUser.Username = user.Username;

newUser.Password = user.Password;

DataLayer.UserDL.Create(user);

}

}

public void Update(User user)

{

if (user.EntityId == null)

{

BusinessLayer.EntityBL.Create(user);

}

else

{

BusinessLayer.EntityBL.Update(user);

}

if (user.Id.HasValue)

{

DataLayer.UserDL.Update(user);

}

else

{

DataLayer.UserDL.Create(user);

}

}

public void Delete(User user)

{

DataLayer.UserDL.Delete(user);

}

}

}

### Business Objects

#### Constants.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public static class Constants

{

public static string Admin = "Administrator";

public static string Office = "Office";

public static string User = "User";

}

}

#### Appointment.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Appointment : BusinessBase

{

public Person Person { get; set; }

public Clinic Clinic { get; set; }

public List<AppointmentService> AppointmentServices { get; set; }

}

}

#### AppointmentService.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class AppointmentService : BusinessBase

{

public Provider Provider { get; set; }

public Service Service { get; set; }

public Room Room { get; set; }

public bool IsQualified

{

get

{

return Provider.Services.Any(x => x.Id == Service.Id);

}

}

public decimal Cost { get; set; }

public DateTime StartTime { get; set; }

public string StartTimeString

{

get

{

return StartTime.ToString("dd-MMM-yyyy HH:mm").ToUpper();

}

}

}

}

BusinessBase.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public abstract class BusinessBase

{

public int? Id { get; set; }

// public abstract bool IsValid();

}

}

#### Clinic.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Clinic : Entity

{

public string Name {

get

{

return this.Name1;

}

set

{

this.Name1 = value;

}

}

public List<Provider> Providers { get; set; }

public List<Service> Services { get; set; }

public List<Room> Rooms { get; set; }

}

}

#### Entity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Entity : BusinessBase

{

public int? EntityId { get; set; }

protected string Name1 { get; set; }

protected string Name2 { get; set; }

protected string Name3 { get; set; }

public string Address1 { get; set; }

public string Address2 { get; set; }

public string City { get; set; }

public string State { get; set; }

public string Zip { get; set; }

public string Phone1 { get; set; }

public string Phone2 { get; set; }

public string Phone3 { get; set; }

public void SetName(string name1, string name2, string name3)

{

Name1 = name1;

Name2 = name2;

Name3 = name3;

}

public string GetName1()

{

return Name1;

}

public string GetName2()

{

return Name2;

}

public string GetName3()

{

return Name3;

}

public Entity Copy()

{

return new Entity

{

EntityId = this.EntityId,

Name1 = this.Name1,

Name2 = this.Name2,

Name3 = this.Name3,

Address1 = this.Address1,

Address2 = this.Address2,

City = this.City,

State = this.State,

Zip = this.Zip,

Phone1 = this.Phone1,

Phone2 = this.Phone2,

Phone3 = this.Phone3

};

}

public void CopyTo(Entity destination)

{

destination.EntityId = this.EntityId;

destination.Name1 = this.Name1;

destination.Name2 = this.Name2;

destination.Name3 = this.Name3;

destination.Address1 = this.Address1;

destination.Address2 = this.Address2;

destination.City = this.City;

destination.State = this.State;

destination.Zip = this.Zip;

destination.Phone1 = this.Phone1;

destination.Phone2 = this.Phone2;

destination.Phone3 = this.Phone3;

}

public bool IsEmpty() {

return !EntityId.HasValue &&

string.IsNullOrWhiteSpace(Name1) &&

string.IsNullOrWhiteSpace(Name2) &&

string.IsNullOrWhiteSpace(Name3) &&

string.IsNullOrWhiteSpace(Address1) &&

string.IsNullOrWhiteSpace(Address2) &&

string.IsNullOrWhiteSpace(City) &&

string.IsNullOrWhiteSpace(State) &&

string.IsNullOrWhiteSpace(Zip) &&

string.IsNullOrWhiteSpace(Phone1) &&

string.IsNullOrWhiteSpace(Phone2) &&

string.IsNullOrWhiteSpace(Phone3);

}

}

}

#### ErrorMessage.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class ErrorMessage

{

public ErrorMessage(string field, string message)

{

Field = field;

Message = message;

}

public string Field { get; set; }

public string Message { get; set; }

}

}

#### Field.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Field

{

public Field(string name, string value) {

Name = name;

Value = value;

}

public string Name { get; set; }

public string Value { get; set; }

}

}

#### Household.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Household : BusinessBase

{

public Person Payer

{

get

{

return People != null ? People.SingleOrDefault(x => x.IsPayer == true) : null;

}

}

public List<Person> People { get; set; }

public string InsuranceName { get; set; }

public string PolicyNumber { get; set; }

public string GroupNumber { get; set; }

}

}

#### Person.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Person : Entity

{

public string FirstName

{

get

{

return this.Name1;

}

set

{

this.Name1 = value;

}

}

public string MiddleName

{

get

{

return this.Name2;

}

set

{

this.Name2 = value;

}

}

public string LastName

{

get

{

return this.Name3;

}

set

{

this.Name3 = value;

}

}

public int HouseholdId { get; set; }

public bool IsPayer { get; set; }

public DateTime DateOfBirth { get; set; }

public string DateOfBirthString

{

get

{

return DateOfBirth.ToString("dd-MMM-yyyy").ToUpper();

}

}

public Relationship Relationship { get; set; }

public List<Appointment> Appointments { get; set; }

}

}

#### Provider.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Provider : Entity

{

public string FirstName

{

get

{

return this.Name1;

}

set

{

this.Name1 = value;

}

}

public string MiddleName

{

get

{

return this.Name2;

}

set

{

this.Name2 = value;

}

}

public string LastName

{

get

{

return this.Name3;

}

set

{

this.Name3 = value;

}

}

public List<Service> Services { get; set; }

}

}

#### Relationship.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Relationship : BusinessBase

{

public string Name { get; set; }

public int ParentId { get; set; }

}

}

#### Role.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Role : BusinessBase

{

public string Name { get; set; }

}

}

#### Room.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Room : BusinessBase

{

public string Name { get; set; }

}

}

#### Service.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class Service : BusinessBase

{

public string Name { get; set; }

public decimal Cost { get; set; }

public int Minutes { get; set; }

}

}

#### User.cs

using Clinic.BO;

using Clinic.DL;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.BO

{

public class User : Entity

{

private Role \_role;

public string FirstName

{

get

{

return this.Name1;

}

set

{

this.Name1 = value;

}

}

public string MiddleName

{

get

{

return this.Name2;

}

set

{

this.Name2 = value;

}

}

public string LastName

{

get

{

return this.Name3;

}

set

{

this.Name3 = value;

}

}

public string Username { get; set; }

public string Password { get; set; }

public Role Role

{

get

{

if (\_role == null)

{

\_role = DataLayer.RoleDL.Get(Constants.User);

}

return \_role;

}

set

{

\_role = value;

}

}

}

}

### Communication Layer

#### AppointmentController.asmx

using Clinic.BL;

using Clinic.BO;

using Clinic.Utilities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Script.Services;

using System.Web.Services;

namespace Clinic.Controllers

{

[ScriptService]

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

public class AppointmentController : System.Web.Services.WebService

{

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetAppointmentsForClinic()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<IdContext>(Context);

Nullable<DateTime> date = null;

if (!string.IsNullOrWhiteSpace(obj.ServiceDate))

{

date = Convert.ToDateTime(obj.ServiceDate);

}

var result = BusinessLayer.AppointmentBL.GetAppointmentsForClinic(obj.Id, date);

json = JsonParser.ToJson(result);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetAppointmentsForUser()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<IdContext>(Context);

Nullable<DateTime> date = null;

if (!string.IsNullOrWhiteSpace(obj.ServiceDate))

{

date = Convert.ToDateTime(obj.ServiceDate);

}

var result = BusinessLayer.AppointmentBL.GetAppointmentsForUser(obj.Id, date);

json = JsonParser.ToJson(result);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetAvailableAppointments()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<GetAvailabilityContext>(Context);

var result = BusinessLayer.AppointmentBL.GetAvailableAppointments(obj.ClinicId, obj.ServiceId, obj.ServiceDate);

json = JsonParser.ToJson(result);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Create()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<Appointment>(Context);

BusinessLayer.AppointmentBL.Create(obj);

json = JsonParser.ToJson(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Delete()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<Appointment>(Context);

BusinessLayer.AppointmentBL.Delete(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

public class GetAvailabilityContext {

public int ClinicId { get; set; }

public int ServiceId { get; set; }

public DateTime ServiceDate { get; set; }

}

public class IdContext

{

public int Id { get; set; }

public string ServiceDate { get; set; }

}

}

}

#### ClinicController.asmx

using Clinic.BL;

using Clinic.Utilities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Script.Services;

using System.Web.Services;

namespace Clinic.Controllers

{

[ScriptService]

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

public class ClinicController : System.Web.Services.WebService

{

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetAll()

{

string json = string.Empty;

try

{

var clinics = BusinessLayer.ClinicBL.GetClinics();

json = JsonParser.ToJson(clinics);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetAllServices()

{

string json = string.Empty;

try

{

var services = BusinessLayer.ServiceBL.Get();

json = JsonParser.ToJson(services);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetEligibleProviders()

{

string json = string.Empty;

try

{

var clinicId = JsonParser.FromJson<IdContext>(Context).Id;

var users = BusinessLayer.ClinicBL.GetEligibleProviders(clinicId.HasValue ? clinicId.Value : -1);

json = JsonParser.ToJson(users);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetRooms()

{

string json = string.Empty;

try

{

var clinicId = JsonParser.FromJson<IdContext>(Context).Id;

var roles = BusinessLayer.RoomBL.GetRoomsByClinicId(clinicId.Value);

json = JsonParser.ToJson(roles);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Update()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<Clinic.BO.Clinic>(Context);

BusinessLayer.ClinicBL.Update(obj);

json = JsonParser.ToJson(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Delete()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<IdContext>(Context);

BusinessLayer.ClinicBL.Delete(obj.Id.Value);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

}

public class IdContext

{

public int? Id { get; set; }

}

}

#### GlobalSettingsController.asmx

using Clinic.BL;

using Clinic.BO;

using Clinic.Utilities;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Script.Serialization;

using System.Web.Script.Services;

using System.Web.Services;

namespace Clinic.Controllers

{

[ScriptService]

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

public class GlobalSettingsController : System.Web.Services.WebService

{

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetSettings()

{

string json = string.Empty;

try

{

json = GlobalSettings.GetJson();

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

}

}

#### HouseholdController.asmx

using Clinic.BL;

using Clinic.BO;

using Clinic.Utilities;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Script.Serialization;

using System.Web.Script.Services;

using System.Web.Services;

namespace Clinic.Controllers

{

[ScriptService]

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

public class HouseholdController : System.Web.Services.WebService

{

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetByUserId()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<UserContext>(Context);

var household = BusinessLayer.HouseholdBL.GetByUserId(obj.UserId);

json = JsonParser.ToJson(household);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetByPayerName()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<PayerContext>(Context);

var households = BusinessLayer.HouseholdBL.GetByPayerName(obj.FirstName, obj.MiddleName, obj.LastName);

json = JsonParser.ToJson(households);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Get()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<HouseholdContext>(Context);

var household = BusinessLayer.HouseholdBL.Get(obj.Id);

json = JsonParser.ToJson(household);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Update()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<Household>(Context);

BusinessLayer.HouseholdBL.Update(obj);

json = JsonParser.ToJson(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetRelationships()

{

string json = string.Empty;

try

{

var obj = BusinessLayer.RelationshipBL.GetRelationships();

json = JsonParser.ToJson(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

public class UserContext

{

public int UserId { get; set; }

}

public class HouseholdContext

{

public int Id { get; set; }

}

public class PayerContext

{

public string FirstName { get; set; }

public string MiddleName { get; set; }

public string LastName { get; set; }

}

}

}

#### ReportController.asmx

using Clinic.BL;

using Clinic.Utilities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Script.Services;

using System.Web.Services;

namespace Clinic.Controllers

{

[ScriptService]

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

public class ReportController : System.Web.Services.WebService

{

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GenerateAllHouseholdReport()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GenerateAllHouseholdReport();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetHouseholdAndInsurance()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GetHouseholdAndInsurance();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetAllPatientsAndInsurance()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GetAllPatientsAndInsurance();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetAllBilling()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GetAllBilling();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetHouseholdTotalCosts()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GetHouseholdTotalCosts();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetProvidersAndServices()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GetProvidersAndServices();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetServicesAndProviders()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GetServicesAndProviders();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetFutureAppointmentsByPatient()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GetFutureAppointmentsByPatient();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetAllServicesProvided()

{

string json = string.Empty;

try

{

var response = BusinessLayer.ReportBL.GetAllServicesProvided();

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetTotalServicesForProviders()

{

string json = string.Empty;

try

{

var serviceDate = JsonParser.FromJson<DateContext>(Context).Date;

var response = BusinessLayer.ReportBL.GetTotalServicesForProviders(serviceDate);

json = JsonParser.ToJson(response);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

public class DateContext

{

public DateTime Date { get; set; }

}

}

}

#### UserController.asmx

using Clinic.BL;

using Clinic.BO;

using Clinic.Utilities;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Script.Serialization;

using System.Web.Script.Services;

using System.Web.Services;

namespace Clinic.Controllers

{

[ScriptService]

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

public class UserController1 : System.Web.Services.WebService

{

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Login()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<UserDataContext>(Context);

var user = BusinessLayer.UserBL.Get(obj.username, obj.password);

if (user.Id == null)

{

throw new Exception("Login not valid");

}

CurSession.User = user;

json = JsonParser.ToJson(user);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Register()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<RegistrationContext>(Context);

var user = new User() {

Username = obj.username,

Password = obj.password,

Role = obj.role

};

if (GlobalSettings.AdminExists)

{

if (obj.householdId.HasValue)

{

BusinessLayer.UserBL.Create(obj.householdId.Value, user);

}

else

{

BusinessLayer.UserBL.Create(user);

}

}

else

{

user.Role = BusinessLayer.RoleBL.Get(Constants.Admin);

BusinessLayer.UserBL.Create(user);

}

if (CurSession.User == null)

{

CurSession.User = user;

}

GlobalSettings.CheckForAdmin();

json = JsonParser.ToJson(user);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public void Logout()

{

CurSession.User = null;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Update()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<User>(Context);

BusinessLayer.UserBL.Update(obj);

if (CurSession.User.Id == obj.Id)

{

CurSession.User = obj;

}

json = JsonParser.ToJson(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object Remove()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<User>(Context);

if (obj.Id == CurSession.User.Id)

{

throw new Exception("You can not delete yourself");

}

BusinessLayer.UserBL.Update(obj);

CurSession.User = obj;

json = JsonParser.ToJson(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetClients()

{

string json = string.Empty;

try

{

var obj = BusinessLayer.UserBL.GetClients();

json = JsonParser.ToJson(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetStaff()

{

string json = string.Empty;

try

{

var obj = BusinessLayer.UserBL.GetStaff();

json = JsonParser.ToJson(obj);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetRoles()

{

string json = string.Empty;

try

{

var roles = BusinessLayer.RoleBL.GetRoles();

json = JsonParser.ToJson(roles);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetEligibleQualifications()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<UserContext>(Context);

var roles = BusinessLayer.ServiceBL.GetEligibleProviderServicesForUserId(obj.UserId.Value);

json = JsonParser.ToJson(roles);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object GetQualifications()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<UserContext>(Context);

var roles = BusinessLayer.ServiceBL.GetProviderServicesForUserId(obj.UserId.Value);

json = JsonParser.ToJson(roles);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

[ScriptMethod(ResponseFormat = ResponseFormat.Json)]

[WebMethod(EnableSession = true)]

public object UpdateQualifications()

{

string json = string.Empty;

try

{

var obj = JsonParser.FromJson<UserQualificationsContext>(Context);

BusinessLayer.ServiceBL.UpdateProviderServicesForUserId(obj.UserId, obj.Services);

}

catch (Exception ex)

{

json = JsonParser.ExceptionToJson(ex);

}

return json;

}

public class UserContext

{

public int? UserId { get; set; }

}

public class RegistrationContext : UserDataContext

{

public int? householdId { get; set; }

public Role role { get; set; }

}

public class UserDataContext

{

public string username { get; set; }

public string password { get; set; }

}

public class UserQualificationsContext

{

public int UserId { get; set; }

public List<Service> Services { get; set; }

}

}

}

### Data Layer

#### AppointmentDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class AppointmentDL : DlBase

{

private void Populate(Object obj, OracleDataReader reader)

{

var target = (Appointment)obj;

target.Id = Convert.ToInt32(reader["appointment\_id"]);

target.Clinic = DataLayer.ClinicDL.Get(Convert.ToInt32(reader["clinic\_id"]));

target.Person = DataLayer.PersonDL.Get(Convert.ToInt32(reader["household\_person\_id"]));

target.AppointmentServices = GetAppointmentServices(target.Id.Value).OrderBy(x => x.StartTime).ToList();

}

private void PopulateService(Object obj, OracleDataReader reader)

{

var target = (AppointmentService)obj;

target.Id = Convert.ToInt32(reader["appointment\_service\_id"]);

target.Provider = DataLayer.ProviderDL.Get(Convert.ToInt32(reader["provider\_id"]));

target.Service = DataLayer.ServiceDL.Get(Convert.ToInt32(reader["service\_id"]));

target.Room = DataLayer.RoomDL.Get(Convert.ToInt32(reader["service\_id"]));

target.StartTime = DateTime.Parse(reader["time"].ToString());

}

public void Create(Appointment appointment)

{

int id = GetNextVal(Sequences.Appointment);

string sql = string.Format(@"

INSERT INTO APPOINTMENT

(APPOINTMENT\_ID, CLINIC\_ID, HOUSEHOLD\_PERSON\_ID)

VALUES

({0},{1},{2})

",

id,

appointment.Clinic.Id,

appointment.Person.Id);

ExecuteQuery(sql);

appointment.Id = id;

foreach(var appointmentService in appointment.AppointmentServices)

{

AddService(appointment, appointmentService);

}

}

public void Delete(Appointment appointment)

{

appointment.AppointmentServices.ForEach(x => DeleteService(x));

string sql = string.Format(@"

DELETE FROM APPOINTMENT

WHERE APPOINTMENT\_ID = {0}

",

appointment.Id);

ExecuteQuery(sql);

}

public void AddService(Appointment appointment, AppointmentService appointmentService)

{

int id = GetNextVal(Sequences.AppointmentService);

string sql = string.Format(@"

INSERT INTO APPOINTMENT\_SERVICE

(APPOINTMENT\_SERVICE\_ID,

APPOINTMENT\_ID,

PROVIDER\_ID,

SERVICE\_ID,

ROOM\_ID,

COST,

TIME)

VALUES

({0},{1},{2},{3},{4},{5},TO\_DATE('{6}', 'dd/mon/yyyy HH24:MI'))

",

id,

appointment.Id,

appointmentService.Provider.Id,

appointmentService.Service.Id,

appointmentService.Room.Id,

appointmentService.Cost,

appointmentService.StartTimeString);

ExecuteQuery(sql);

appointmentService.Id = id;

}

public void DeleteService(AppointmentService appointmentService)

{

string sql = string.Format(@"

DELETE FROM APPOINTMENT\_SERVICE

WHERE APPOINTMENT\_SERVICE\_ID = {0}

",

appointmentService.Id);

ExecuteQuery(sql);

}

public List<Appointment> GetAppointmentsForUser(int userId, DateTime? date)

{

var obj = new List<Appointment>();

var dateFilter = !date.HasValue ? ""

: "AND TRUNC(aps.TIME) = TO\_DATE('" + date.Value.ToString("dd-MMM-yyyy").ToUpper() + "')";

string sql = string.Format(@"

SELECT a.\*

FROM APPOINTMENT\_SERVICE aps

JOIN APPOINTMENT a ON aps.APPOINTMENT\_ID = a.APPOINTMENT\_ID

JOIN HOUSEHOLD\_PERSON hp ON a.HOUSEHOLD\_PERSON\_ID = hp.HOUSEHOLD\_PERSON\_ID

JOIN USERS u ON hp.ENTITY\_ID = u.ENTITY\_ID

WHERE u.USER\_ID = {0}

{1}

",

userId,

dateFilter);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Appointment> GetAppointmentsForClinic(int clinicId, DateTime? date)

{

var obj = new List<Appointment>();

var dateFilter = !date.HasValue ? ""

: "AND TRUNC(aps.TIME) = TO\_DATE('" + date.Value.ToString("dd-MMM-yyyy").ToUpper() + "')";

string sql = string.Format(@"

SELECT a.\*

FROM APPOINTMENT\_SERVICE aps

JOIN APPOINTMENT a ON aps.APPOINTMENT\_ID = a.APPOINTMENT\_ID

WHERE a.CLINIC\_ID = {0}

{1}

",

clinicId,

dateFilter);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<AppointmentService> GetExistingAppointmentServices(int clinicId, DateTime serviceDate)

{

var obj = new List<AppointmentService>();

string sql = string.Format(@"

SELECT aps.\*

FROM APPOINTMENT\_SERVICE aps

JOIN APPOINTMENT a ON aps.APPOINTMENT\_ID = a.APPOINTMENT\_ID

WHERE a.CLINIC\_ID = {0}

AND TRUNC(aps.TIME) = TO\_DATE('{1}')

",

clinicId,

serviceDate.ToString("dd-MMM-yyyy").ToUpper());

ExecuteReader(sql, obj, PopulateService);

return obj;

}

public List<AppointmentService> GetAppointmentServices(int appointmentId)

{

var obj = new List<AppointmentService>();

string sql = string.Format(@"

SELECT \*

FROM APPOINTMENT\_SERVICE

WHERE APPOINTMENT\_ID = {0}

",

appointmentId);

ExecuteReader(sql, obj, PopulateService);

return obj;

}

}

}

#### ClinicDL.cs

using Clinic.BL;

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class ClinicDL : DlBase

{

public void Populate(Object obj, OracleDataReader reader)

{

var target = (Clinic.BO.Clinic)obj;

int entityId = Int32.Parse(reader["entity\_id"].ToString());

DataLayer.EntityDL.Get(entityId).CopyTo(target);

target.Id = Int32.Parse(reader["clinic\_id"].ToString());

target.Providers = DataLayer.ProviderDL.GetProvidersByClinicId(target.Id.Value);

target.Rooms = DataLayer.RoomDL.GetRoomsByClinicId(target.Id.Value);

target.Services = DataLayer.ServiceDL.GetServicesByClinicId(target.Id.Value);

}

public Clinic.BO.Clinic Get(int id)

{

var obj = new Clinic.BO.Clinic();

string sql = string.Format(@"

SELECT \* FROM CLINIC

WHERE CLINIC\_ID = {0}

", id);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Clinic.BO.Clinic> GetClinics()

{

var obj = new List<Clinic.BO.Clinic>();

string sql = string.Format(@"

SELECT \* FROM CLINIC

");

ExecuteReader(sql, obj, Populate);

return obj;

}

public void Create(Clinic.BO.Clinic clinic)

{

int id = GetNextVal(Sequences.Clinic);

string sql = string.Format(@"

INSERT INTO CLINIC

(CLINIC\_ID, ENTITY\_ID)

VALUES

({0},{1})

",

id,

clinic.EntityId);

ExecuteQuery(sql);

clinic.Id = id;

}

public void Update(Clinic.BO.Clinic clinic)

{

// DO Nothing

// Name (on entity) and all others should be update in their respective layers

}

public void Delete(int clinicId)

{

string sql = string.Format(@"

DELETE CLINIC

WHERE CLINIC\_ID = {0}

", clinicId);

ExecuteQuery(sql);

}

}

}

#### DataLayer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public static class DataLayer

{

public static UserDL UserDL = new UserDL();

public static EntityDL EntityDL = new EntityDL();

public static RoleDL RoleDL = new RoleDL();

public static PersonDL PersonDL = new PersonDL();

public static HouseholdDL HouseholdDL = new HouseholdDL();

public static RelationshipDL RelationshipDL = new RelationshipDL();

public static ProviderDL ProviderDL = new ProviderDL();

public static ServiceDL ServiceDL = new ServiceDL();

public static RoomDL RoomDL = new RoomDL();

public static ClinicDL ClinicDL = new ClinicDL();

public static AppointmentDL AppointmentDL = new AppointmentDL();

public static ReportDL ReportDL = new ReportDL();

}

}

#### DlBase.cs

using Oracle.DataAccess.Client;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class DlBase

{

protected string \_connectionString = "Data Source=terminus;User ID=AFRANZEN;Password=P0ppyCh3w";

public DlBase() {

}

public OracleCommand GetCommand(string sqlQuery) {

OracleConnection con = new OracleConnection(\_connectionString);

OracleCommand cmd = new OracleCommand(sqlQuery, con);

con.Open();

return cmd;

}

public void CloseConnection(OracleCommand cmd)

{

cmd.Connection.Close();

}

protected void ExecuteQuery(string sqlQuery)

{

var cmd = GetCommand(sqlQuery);

cmd.ExecuteNonQuery();

CloseConnection(cmd);

}

private void PopulateList(Object obj, OracleDataReader reader, PopulateData populateData)

{

Type type = obj.GetType().GetGenericArguments()[0];

var instance = Activator.CreateInstance(type);

populateData(instance, reader);

((IList)obj).Add(instance);

}

public delegate void PopulateData(Object data, OracleDataReader reader);

/// <summary>

///

/// </summary>

/// <param name="sqlQuery"></param>

/// <param name="data">A single object of T or a List<T> where T is the datatype in PopulateData</param>

/// <param name="populateData"></param>

protected void ExecuteReader(string sqlQuery, Object data, PopulateData populateData)

{

var cmd = GetCommand(sqlQuery);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

if (data.GetType().IsGenericType && data is IEnumerable)

{

PopulateList(data, reader, populateData);

}

else

{

populateData(data, reader);

}

}

CloseConnection(cmd);

}

protected Object ExecuteScalar(string sqlQuery)

{

var cmd = GetCommand(sqlQuery);

var obj = cmd.ExecuteScalar();

CloseConnection(cmd);

return obj;

}

protected int GetNextVal(string sequenceName)

{

string sql = "SELECT " + sequenceName + ".nextval FROM dual";

return Int16.Parse(ExecuteScalar(sql).ToString());

}

}

}

#### EntityDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class EntityDL : DlBase

{

public void Populate(Object obj, OracleDataReader reader)

{

var target = (Entity)obj;

target.EntityId = Convert.ToInt32(reader["entity\_id"]);

target.Address1 = reader["address1"].ToString();

target.Address2 = reader["address2"].ToString();

target.City = reader["city"].ToString();

target.State = reader["state"].ToString();

target.Zip = reader["zip"].ToString();

target.Phone1 = reader["phone1"].ToString();

target.Phone2 = reader["phone2"].ToString();

target.Phone3 = reader["phone3"].ToString();

target.SetName(reader["name1"].ToString(), reader["name2"].ToString(), reader["name3"].ToString());

}

public Entity Get(int id)

{

var obj = new User();

string sql = string.Format(@"

SELECT \* FROM ENTITY

WHERE ENTITY\_ID = {0}

", id);

ExecuteReader(sql, obj, Populate);

return obj;

}

public void Create(Entity entity)

{

int id = GetNextVal(Sequences.Entity);

string sql = string.Format(@"

INSERT INTO ENTITY

(ENTITY\_ID, NAME1, NAME2, NAME3, ADDRESS1, ADDRESS2, CITY, STATE, ZIP, PHONE1, PHONE2, PHONE3)

VALUES

({0},'{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}')

",

id,

entity.GetName1(),

entity.GetName2(),

entity.GetName3(),

entity.Address1,

entity.Address2,

entity.City,

entity.State,

entity.Zip,

entity.Phone1,

entity.Phone2,

entity.Phone3);

ExecuteQuery(sql);

entity.EntityId = id;

}

public void Update(Entity entity)

{

string sql = string.Format(@"

UPDATE ENTITY

SET NAME1 = '{1}',

NAME2 = '{2}',

NAME3 = '{3}',

ADDRESS1 = '{4}',

ADDRESS2 = '{5}',

CITY = '{6}',

STATE = '{7}',

ZIP = '{8}',

PHONE1 = '{9}',

PHONE2 = '{10}',

PHONE3 = '{11}'

WHERE ENTITY\_ID = {0}

",

entity.EntityId,

entity.GetName1(),

entity.GetName2(),

entity.GetName3(),

entity.Address1,

entity.Address2,

entity.City,

entity.State,

entity.Zip,

entity.Phone1,

entity.Phone2,

entity.Phone3);

ExecuteQuery(sql);

}

}

}

#### HouseholdDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class HouseholdDL : DlBase

{

private void Populate(Object obj, OracleDataReader reader)

{

var target = (Household)obj;

target.Id = Convert.ToInt32(reader["household\_id"]);

target.GroupNumber = reader["group\_number"].ToString();

target.InsuranceName = reader["insurance\_name"].ToString();

target.PolicyNumber = reader["policy\_number"].ToString();

target.People = DataLayer.PersonDL.GetByHouseholdId(target.Id.Value);

}

public Household Get(int id)

{

var obj = new Household();

string sql = string.Format(@"

SELECT \*

FROM HOUSEHOLD\_PERSON

WHERE HOUSEHOLD\_PERSON\_ID = {0}

", id);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Household> GetAll()

{

var obj = new List<Household>();

string sql = string.Format(@"

SELECT \*

FROM HOUSEHOLD\_PERSON

");

ExecuteReader(sql, obj, Populate);

return obj;

}

public Household GetByUserId(int userId)

{

var obj = new Household();

string sql = string.Format(@"

SELECT \*

FROM USERS u

JOIN HOUSEHOLD\_PERSON hp ON u.entity\_id = hp.entity\_id

JOIN HOUSEHOLD h ON hp.household\_id = h.household\_id

WHERE u.user\_id = {0}

", userId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Household> GetByPayerName(string firstName, string middleName, string lastName)

{

var obj = new List<Household>();

var where = string.Empty;

if (!string.IsNullOrEmpty(firstName) && !string.IsNullOrEmpty(middleName) && !string.IsNullOrEmpty(lastName))

{

where = string.Format(@"

UPPER(e.NAME1) LIKE UPPER('{0}%') AND

UPPER(e.NAME2) LIKE UPPER('{1}%') AND

UPPER(e.NAME3) LIKE UPPER('{2}%')

", firstName, middleName, lastName);

}

else if (!string.IsNullOrEmpty(firstName) && !string.IsNullOrEmpty(lastName))

{

where = string.Format(@"

UPPER(e.NAME1) LIKE UPPER('{0}%') AND

UPPER(e.NAME3) LIKE UPPER('{1}%')

", firstName, lastName);

}

else if (!string.IsNullOrEmpty(lastName))

{

where = string.Format(@"

UPPER(e.NAME3) LIKE UPPER('{0}%')

", lastName);

}

else

{

return obj;

}

string sql = string.Format(@"

SELECT DISTINCT h.\*

FROM HOUSEHOLD h

JOIN HOUSEHOLD\_PERSON hp ON h.household\_id = hp.household\_id

JOIN ENTITY e ON hp.ENTITY\_ID = e.ENTITY\_ID

WHERE {0}"

, where);

ExecuteReader(sql, obj, Populate);

return obj;

}

public void Create(Household household)

{

//TODO InjectionValidator(household.);

//TODO InjectionValidator(lastName);

string sql;

int id = GetNextVal(Sequences.Household);

sql = string.Format(@"

INSERT INTO HOUSEHOLD

(HOUSEHOLD\_ID, INSURANCE\_NAME, POLICY\_NUMBER, GROUP\_NUMBER)

VALUES

({0},'{1}','{2}','{3}')

",

id,

household.InsuranceName,

household.PolicyNumber,

household.GroupNumber);

ExecuteQuery(sql);

household.Id = id;

}

public void Update(Household household)

{

//TODO InjectionValidator(firstName);

//TODO InjectionValidator(lastName);

string sql = string.Format(@"

UPDATE HOUSEHOLD

SET INSURANCE\_NAME = '{1}',

POLICY\_NUMBER = '{2}',

GROUP\_NUMBER = '{3}'

WHERE HOUSEHOLD\_ID = {0}

",

household.Id,

household.InsuranceName,

household.PolicyNumber,

household.GroupNumber);

ExecuteQuery(sql);

}

}

}

#### PersonDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class PersonDL : DlBase

{

private void Populate(Object obj, OracleDataReader reader)

{

var target = (Person)obj;

int entityId;

if (Int32.TryParse(reader["entity\_id"].ToString(), out entityId))

{

DataLayer.EntityDL.Get(entityId).CopyTo(target);

}

target.Id = Convert.ToInt32(reader["household\_person\_id"]);

target.HouseholdId = Convert.ToInt32(reader["household\_id"]);

target.Appointments = new List<Appointment>();

target.IsPayer = reader["is\_payer"].ToString() == "Y" ? true : false;

target.DateOfBirth = DateTime.Parse(reader["dob"].ToString());

var relationshipId = Convert.ToInt32(reader["relationship\_id"]);

target.Relationship = DataLayer.RelationshipDL.GetRelationships().First(x => x.Id == relationshipId);

}

public Person Get(int id)

{

var obj = new Person();

string sql = string.Format(@"

SELECT \*

FROM HOUSEHOLD\_PERSON

WHERE HOUSEHOLD\_PERSON\_ID = {0}

", id);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Person> GetByHouseholdId(int householdId)

{

var obj = new List<Person>();

string sql = string.Format(@"

SELECT \*

FROM HOUSEHOLD\_PERSON

WHERE HOUSEHOLD\_ID = {0}

", householdId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public Person GetPayerByHouseholdId(int householdId)

{

var obj = new Person();

string sql = string.Format(@"

SELECT \*

FROM HOUSEHOLD\_PERSON

WHERE HOUSEHOLD\_ID = {0}

AND IS\_PAYER = 'Y'

", householdId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public Person GetByUserId(int userId)

{

var obj = new Person();

string sql = string.Format(@"

SELECT \*

FROM HOUSEHOLD\_PERSON

", userId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public void Create(int householdId, Person person)

{

//TODO InjectionValidator(person.);

//TODO InjectionValidator(lastName);

string sql;

int id = GetNextVal(Sequences.Person);

sql = string.Format(@"

INSERT INTO HOUSEHOLD\_PERSON

(HOUSEHOLD\_PERSON\_ID, RELATIONSHIP\_ID, HOUSEHOLD\_ID, ENTITY\_ID, IS\_PAYER, DOB)

VALUES

({0},{1},{2},{3},'{4}',TO\_DATE('{5}'))

",

id,

person.Relationship.Id,

householdId,

person.EntityId,

person.IsPayer ? 'Y' : 'N',

person.DateOfBirthString);

ExecuteQuery(sql);

person.Id = id;

}

public void Update(int householdId, Person person)

{

//TODO InjectionValidator(firstName);

//TODO InjectionValidator(lastName);

string sql = string.Format(@"

UPDATE HOUSEHOLD\_PERSON

SET RELATIONSHIP\_ID = {1},

HOUSEHOLD\_ID = {2},

ENTITY\_ID = {3},

IS\_PAYER = '{4}',

DOB = '{5}'

WHERE HOUSEHOLD\_PERSON\_ID = {0}

",

person.Id,

person.Relationship.Id,

householdId,

person.EntityId,

person.IsPayer ? 'Y' : 'N',

person.DateOfBirthString);

ExecuteQuery(sql);

}

}

}

#### ProviderDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class ProviderDL : DlBase

{

public void Populate(Object obj, OracleDataReader reader)

{

var target = (Provider)obj;

int entityId = Int32.Parse(reader["entity\_id"].ToString());

DataLayer.EntityDL.Get(entityId).CopyTo(target);

target.Id = Int32.Parse(reader["provider\_id"].ToString());

target.Services = DataLayer.ServiceDL.GetServicesByProviderId(target.Id.Value);

}

public Provider Get(int id)

{

var obj = new Provider();

string sql = string.Format(@"

SELECT \* FROM PROVIDER

WHERE PROVIDER\_ID = {0}

", id);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Provider> GetAll()

{

var obj = new List<Provider>();

string sql = string.Format(@"

SELECT \* FROM PROVIDER

");

ExecuteReader(sql, obj, Populate);

return obj;

}

private bool GetIsProvider(int providerId, int clinicId, int entityId)

{

string sql = string.Format(@"

SELECT ENTITY\_ID

FROM PROVIDER

WHERE ENTITY\_ID = {0}

AND CLINIC\_ID = {1}

AND PROVIDER\_ID = {2}

", entityId, clinicId, providerId);

var result = ExecuteScalar(sql);

return string.IsNullOrEmpty(result.ToString());

}

public List<Provider> GetProvidersByClinicId(int clinicId)

{

var obj = new List<Provider>();

string sql = string.Format(@"

SELECT \* FROM PROVIDER

WHERE CLINIC\_ID = {0}

", clinicId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public Provider GetProviderByUserId(int userId)

{

var obj = new Provider();

string sql = string.Format(@"

SELECT p.\*

FROM PROVIDER p

JOIN USERS u ON p.ENTITY\_ID = u.ENTITY\_ID

WHERE u.USER\_ID = {0}

", userId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public void Create(int clinicId, Provider provider)

{

string sql;

int id = GetNextVal(Sequences.Provider);

sql = string.Format(@"

INSERT INTO PROVIDER

(PROVIDER\_ID, CLINIC\_ID, ENTITY\_ID)

VALUES

({0},{1},{2})

",

id,

clinicId,

provider.EntityId);

ExecuteQuery(sql);

provider.Id = id;

}

public void Delete(int clinicId, Provider provider)

{

provider.Services.ForEach(x => DeleteProviderQualification(provider, x));

string sql = string.Format(@"

DELETE FROM PROVIDER

WHERE PROVIDER\_ID = {0}

AND CLINIC\_ID = {1}

AND ENTITY\_ID = {2}

",

provider.Id,

clinicId,

provider.EntityId);

ExecuteQuery(sql);

}

public void Update(int clinicId, Provider provider)

{

var existingServices = DataLayer.ServiceDL.GetServicesByProviderId(provider.Id.Value);

var newServices = provider.Services.Where(x => !x.Id.HasValue).ToList();

var updatedServices = provider.Services.Where(x => x.Id.HasValue).ToList();

var removedServices = existingServices.Where(existing => !updatedServices.Any(cur => cur.Id.Value == existing.Id.Value)).ToList();

newServices.ForEach(x => InsertProviderQualification(provider, x));

removedServices.ForEach(x => DeleteProviderQualification(provider, x));

}

private void InsertProviderQualification(Provider provider, Service qualification)

{

int id = GetNextVal(Sequences.ProviderQualification);

string sql = string.Format(@"

INSERT INTO PROVIDER\_QUALIFICATION

(PROVIDER\_QUALIFICATION\_ID, PROVIDER\_ID, SERVICE\_ID)

VALUES

({0},{1},{2})

",

id,

provider.Id,

qualification.Id);

ExecuteQuery(sql);

provider.Services.Add(qualification);

}

private void DeleteProviderQualification(Provider provider, Service qualification)

{

string sql = string.Format(@"

DELETE FROM PROVIDER\_QUALIFICATION

WHERE PROVIDER\_ID = {0}

AND SERVICE\_ID = {1}

",

provider.Id,

qualification.Id);

ExecuteQuery(sql);

provider.Services.RemoveAll(x => x.Id == qualification.Id);

}

}

}

#### RelationshipDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class RelationshipDL : DlBase

{

private List<Relationship> \_relationships;

private List<Relationship> Relationships {

get

{

if(\_relationships == null)

{

\_relationships = new List<Relationship>();

string sql = string.Format(@"

SELECT \*

FROM RELATIONSHIP");

this.ExecuteReader(sql, \_relationships, Populate);

}

return \_relationships;

}

}

private void Populate(Object obj, OracleDataReader reader)

{

var target = (Relationship)obj;

target.Id = Convert.ToInt32(reader["relationship\_id"]);

target.Name = reader["name"].ToString();

}

public Relationship Get(int id)

{

return Relationships.First(x => x.Id == id);

}

public Relationship Get(string name)

{

return Relationships.First(x => x.Name == name);

}

public List<Relationship> GetRelationships()

{

return Relationships;

}

}

}

#### ReportDL.cs

using Clinic.BO;

using Clinic.DL;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Dynamic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class ReportDL : DlBase

{

/// <summary>

/// List all households with the household ID, name, address, and home phone along with the

/// patient ID, name and relationship for each patient.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GenerateAllHouseholdReport()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

h.HOUSEHOLD\_ID as ACCOUNT\_NUMBER,

r.NAME as RELATIONSHIP,

hp.HOUSEHOLD\_PERSON\_ID as PATIENT\_ID,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as NAME,

e.ADDRESS1,

e.ADDRESS2,

e.CITY,

e.STATE,

e.ZIP,

e.PHONE1,

e.PHONE2,

e.PHONE3

FROM HOUSEHOLD h

JOIN HOUSEHOLD\_PERSON hp ON h.HOUSEHOLD\_ID = hp.HOUSEHOLD\_ID

JOIN ENTITY e ON hp.ENTITY\_ID = e.ENTITY\_ID

JOIN RELATIONSHIP r ON hp.RELATIONSHIP\_ID = r.RELATIONSHIP\_ID

ORDER BY h.HOUSEHOLD\_ID

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Account\_Number = reader["account\_number"].ToString();

newObj.Relationship = reader["relationship"].ToString();

newObj.Patient\_ID = reader["patient\_id"].ToString();

newObj.Name = reader["name"].ToString();

newObj.Address = reader["address1"].ToString();

newObj.City = reader["city"].ToString();

newObj.State = reader["state"].ToString();

newObj.Zip = reader["zip"].ToString();

newObj.Phone = reader["phone1"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// List the insurance coverage for all households by household ID, household name, insurance

/// company ID and company name.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetHouseholdAndInsurance()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

h.HOUSEHOLD\_ID as ACCOUNT\_NUMBER,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as NAME,

h.INSURANCE\_NAME

FROM HOUSEHOLD h

JOIN HOUSEHOLD\_PERSON hp ON h.HOUSEHOLD\_ID = hp.HOUSEHOLD\_ID

JOIN ENTITY e ON hp.ENTITY\_ID = e.ENTITY\_ID

WHERE hp.IS\_PAYER = 'Y'

ORDER BY e.NAME1, e.NAME2, e.NAME3

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Account\_Number = reader["account\_number"].ToString();

newObj.Name = reader["name"].ToString();

newObj.Insurance\_Name = reader["insurance\_name"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// List all patients in alphabetical order by patient ID, name, and date of birth along with the

/// name of the insurance company and policy number.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetAllPatientsAndInsurance()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

hp.HOUSEHOLD\_PERSON\_ID as PATIENT\_ID,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as NAME,

hp.DOB,

h.INSURANCE\_NAME,

h.POLICY\_NUMBER,

h.GROUP\_NUMBER

FROM HOUSEHOLD h

JOIN HOUSEHOLD\_PERSON hp ON h.HOUSEHOLD\_ID = hp.HOUSEHOLD\_ID

JOIN ENTITY e ON hp.ENTITY\_ID = e.ENTITY\_ID

JOIN RELATIONSHIP r ON hp.RELATIONSHIP\_ID = r.RELATIONSHIP\_ID

ORDER BY h.HOUSEHOLD\_ID

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Patient\_ID = reader["patient\_id"].ToString();

newObj.Name = reader["name"].ToString();

newObj.Date\_Of\_Birth = reader["dob"].ToString();

newObj.Insurance\_Name = reader["insurance\_name"].ToString();

newObj.Policy\_Number = reader["policy\_number"].ToString();

newObj.Group\_Number = reader["group\_number"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// Show itemized billings for all households with the household ID, household name, patient ID,

/// patient name, service received, and the cost of the service.Show the output in alphabetical

/// order by household name, patient name and billing date.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetAllBilling()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

h.HOUSEHOLD\_ID as ACCOUNT\_NUMBER,

payer.NAME as HOUSEHOLD\_NAME,

hp.HOUSEHOLD\_PERSON\_ID as PATIENT\_ID,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as PATIENT\_NAME,

s.NAME as SERVICE,

asp.COST,

asp.TIME as SERVICE\_DATE

FROM HOUSEHOLD h

JOIN HOUSEHOLD\_PERSON hp ON h.HOUSEHOLD\_ID = hp.HOUSEHOLD\_ID

JOIN ENTITY e ON hp.ENTITY\_ID = e.ENTITY\_ID

JOIN (

SELECT

h.HOUSEHOLD\_ID,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as NAME

FROM HOUSEHOLD h

JOIN HOUSEHOLD\_PERSON hp ON h.HOUSEHOLD\_ID = hp.HOUSEHOLD\_ID

JOIN ENTITY e ON hp.ENTITY\_ID = e.ENTITY\_ID

WHERE hp.IS\_PAYER = 'Y'

) payer ON h.HOUSEHOLD\_ID = payer.HOUSEHOLD\_ID

JOIN APPOINTMENT a ON hp.HOUSEHOLD\_PERSON\_ID = a.HOUSEHOLD\_PERSON\_ID

JOIN APPOINTMENT\_SERVICE asp ON a.APPOINTMENT\_ID = asp.APPOINTMENT\_ID

JOIN SERVICE s ON asp.SERVICE\_ID = s.SERVICE\_ID

ORDER BY payer.NAME, e.NAME1, e.NAME2, e.NAME3, asp.TIME

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Account\_Number = reader["account\_number"].ToString();

newObj.Household\_Name = reader["household\_name"].ToString();

newObj.Patient\_ID = reader["patient\_id"].ToString();

newObj.Patient\_Name = reader["patient\_name"].ToString();

newObj.Service = reader["service"].ToString();

newObj.Cost = "$" + reader["cost"].ToString();

newObj.Start\_Time = reader["service\_date"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// List the total cost of all services received for each household.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetHouseholdTotalCosts()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

payer.NAME as HOUSEHOLD\_NAME,

SUM(asp.COST) as TOTAL

FROM HOUSEHOLD h

JOIN HOUSEHOLD\_PERSON hp ON h.HOUSEHOLD\_ID = hp.HOUSEHOLD\_ID

JOIN APPOINTMENT a ON hp.HOUSEHOLD\_PERSON\_ID = a.HOUSEHOLD\_PERSON\_ID

JOIN APPOINTMENT\_SERVICE asp ON a.APPOINTMENT\_ID = asp.APPOINTMENT\_ID

JOIN (

SELECT

hp.HOUSEHOLD\_ID,

SUM(aps.COST) as TOTAL

FROM HOUSEHOLD\_PERSON hp

JOIN APPOINTMENT a ON hp.HOUSEHOLD\_PERSON\_ID = a.HOUSEHOLD\_PERSON\_ID

JOIN APPOINTMENT\_SERVICE aps ON a.APPOINTMENT\_ID = aps.APPOINTMENT\_ID

GROUP BY hp.HOUSEHOLD\_ID

) totals ON h.HOUSEHOLD\_ID = totals.HOUSEHOLD\_ID

JOIN (

SELECT

h.HOUSEHOLD\_ID,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as NAME

FROM HOUSEHOLD h

JOIN HOUSEHOLD\_PERSON hp ON h.HOUSEHOLD\_ID = hp.HOUSEHOLD\_ID

JOIN ENTITY e ON hp.ENTITY\_ID = e.ENTITY\_ID

WHERE hp.IS\_PAYER = 'Y'

) payer ON h.HOUSEHOLD\_ID = payer.HOUSEHOLD\_ID

GROUP BY payer.NAME

ORDER BY payer.NAME

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Household\_Name = reader["household\_name"].ToString();

newObj.Total = "$" + reader["total"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// List each provider with all services he or she is qualified to render.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetProvidersAndServices()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as PROVIDER,

s.NAME as QUALIFIED\_SERVICE

FROM PROVIDER p

JOIN PROVIDER\_QUALIFICATION pq ON p.PROVIDER\_ID = pq.PROVIDER\_ID

JOIN SERVICE s ON pq.SERVICE\_ID = s.SERVICE\_ID

JOIN ENTITY e ON p.ENTITY\_ID = e.ENTITY\_ID

ORDER BY e.NAME1, e.NAME2, e.NAME3, s.NAME

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Provider = reader["provider"].ToString();

newObj.Qualified\_Service = reader["qualified\_service"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// List each service available with all providers who are qualified to offer this service.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetServicesAndProviders()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

s.NAME as SERVICE,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as QUALIFIED\_PROVIDER

FROM SERVICE s

JOIN PROVIDER\_QUALIFICATION pq ON s.SERVICE\_ID = pq.SERVICE\_ID

JOIN PROVIDER p ON pq.PROVIDER\_ID = p.PROVIDER\_ID

JOIN ENTITY e ON p.ENTITY\_ID = e.ENTITY\_ID

ORDER BY s.NAME, e.NAME1, e.NAME2, e.NAME3

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Service = reader["service"].ToString();

newObj.Qualified\_Provider = reader["qualified\_provider"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// List all future appointments by name of patient, appointment date and time, estimated length

/// of service, and contact home phone number.Dates and times should be in calendar order

/// for each patient.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetFutureAppointmentsByPatient()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as PATIENT,

asp.TIME as APPOINTMENT\_DATE,

s.MINUTES,

e.PHONE1 as PHONE

FROM APPOINTMENT\_SERVICE asp

JOIN APPOINTMENT a ON asp.APPOINTMENT\_ID = a.APPOINTMENT\_ID

JOIN HOUSEHOLD\_PERSON hp ON a.HOUSEHOLD\_PERSON\_ID = hp.HOUSEHOLD\_PERSON\_ID

JOIN ENTITY e ON hp.ENTITY\_ID = e.ENTITY\_ID

JOIN SERVICE s ON asp.SERVICE\_ID = s.SERVICE\_ID

WHERE asp.TIME > SYSDATE

ORDER BY e.NAME1, e.NAME2, e.NAME3, asp.TIME

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Patient = reader["patient"].ToString();

newObj.Appointment\_Date = reader["appointment\_date"].ToString();

newObj.Minutes = reader["minutes"].ToString();

newObj.Phone = reader["phone"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// For a given date, list all services provided by each provider in alphabetical order by name of

/// the provider.Show the service ID, service description and cost of service.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetAllServicesProvided()

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

asp.TIME as SERVICE\_DATE,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as PROVIDER,

s.SERVICE\_ID,

s.NAME as SERVICE\_NAME,

s.COST

FROM PROVIDER p

JOIN APPOINTMENT\_SERVICE asp ON p.PROVIDER\_ID = asp.PROVIDER\_ID

JOIN SERVICE s ON asp.SERVICE\_ID = s.SERVICE\_ID

JOIN ENTITY e ON p.ENTITY\_ID = e.ENTITY\_ID

ORDER BY asp.TIME, e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Service\_Date = reader["service\_date"].ToString();

newObj.Provider = reader["provider"].ToString();

newObj.Service\_ID = reader["service\_id"].ToString();

newObj.Service\_Name = reader["service\_name"].ToString();

newObj.Cost = "$" + reader["Cost"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

/// <summary>

/// For a given date, list the total amount of services each provider rendered.Show in

/// alphabetical order by the provider’s name.

/// </summary>

/// <returns></returns>

public List<ExpandoObject> GetTotalServicesForProviders(DateTime serviceDate)

{

var list = new List<System.Dynamic.ExpandoObject>();

string sql = string.Format(@"

SELECT

asp.TIME as SERVICE\_DATE,

e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3 as PROVIDER,

SUM(s.COST) as TOTAL

FROM PROVIDER p

JOIN APPOINTMENT\_SERVICE asp ON p.PROVIDER\_ID = asp.PROVIDER\_ID

JOIN SERVICE s ON asp.SERVICE\_ID = s.SERVICE\_ID

JOIN ENTITY e ON p.ENTITY\_ID = e.ENTITY\_ID

GROUP BY asp.TIME, e.NAME1 || ' ' || e.NAME2 || ' ' || e.NAME3

ORDER BY SERVICE\_DATE, PROVIDER

");

var cmd = GetCommand(sql);

var reader = cmd.ExecuteReader();

while (reader.Read())

{

dynamic newObj = new ExpandoObject();

newObj.Service\_Date = reader["service\_date"].ToString();

newObj.Provider = reader["provider"].ToString();

newObj.Total = "$" + reader["total"].ToString();

list.Add(newObj);

}

CloseConnection(cmd);

return list;

}

}

}

#### RoleDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class RoleDL : DlBase

{

private List<Role> \_roles = new List<Role>();

private List<Role> Roles {

get

{

if(\_roles.Count == 0)

{

\_roles = new List<Role>();

string sql = string.Format(@"

SELECT \*

FROM ROLES");

this.ExecuteReader(sql, \_roles, Populate);

}

return \_roles;

}

}

private void Populate(Object obj, OracleDataReader reader)

{

var target = (Role)obj;

target.Id = Convert.ToInt32(reader["role\_id"]);

target.Name = reader["role\_name"].ToString();

}

public Role Get(int id)

{

var roles = Roles;

return roles.First(x => x.Id == id);

}

public Role Get(string name)

{

return Roles.First(x => x.Name == name);

}

public List<Role> GetRoles()

{

return Roles;

}

}

}

#### RoomDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class RoomDL : DlBase

{

public void Populate(Object obj, OracleDataReader reader)

{

var target = (Room)obj;

target.Id = Int32.Parse(reader["room\_id"].ToString());

target.Name = reader["name"].ToString();

}

public Room Get(int id)

{

var obj = new Room();

string sql = string.Format(@"

SELECT \* FROM ROOM

WHERE ROOM\_ID = {0}

", id);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Room> GetRoomsByClinicId(int clinicId)

{

var obj = new List<Room>();

string sql = string.Format(@"

SELECT \*

FROM ROOM

WHERE CLINIC\_ID = {0}

", clinicId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public void Create(int clinicId, Room room)

{

int id = GetNextVal(Sequences.Room);

string sql = string.Format(@"

INSERT INTO ROOM

(ROOM\_ID, CLINIC\_ID, NAME)

VALUES

({0},{1},'{2}')

",

id,

clinicId,

room.Name);

ExecuteQuery(sql);

room.Id = id;

}

public void Update(Room room)

{

string sql = string.Format(@"

UPDATE ROOM

SET NAME = '{1}'

WHERE ROOM\_ID = {0}

",

room.Id,

room.Name);

ExecuteQuery(sql);

}

public void Delete(Room room)

{

string sql = string.Format(@"

DELETE FROM ROOM

WHERE ROOM\_ID = {0}

",

room.Id);

ExecuteQuery(sql);

}

}

}

#### Sequences.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public static class Sequences

{

public static string Appointment = "APPOINTMENT\_S";

public static string AppointmentService = "APPOINTMENT\_SERVICE\_S";

public static string Clinic = "CLINIC\_S";

public static string Entity = "ENTITY\_S";

public static string Person = "HOUSEHOLD\_PERSON\_S";

public static string Household = "HOUSEHOLD\_S";

public static string ProviderQualification = "PROVIDER\_QUALIFICATION\_S";

public static string Provider = "PROVIDER\_S";

public static string Room = "ROOM\_S";

public static string Service = "SERVICE\_S";

public static string ServiceClinic = "SERVICE\_CLINIC\_S";

public static string User = "USERS\_S";

}

}

#### ServiceDL.cs

using Clinic.BO;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class ServiceDL : DlBase

{

public void Populate(Object obj, OracleDataReader reader)

{

var target = (Service)obj;

target.Id = Int32.Parse(reader["service\_id"].ToString());

target.Name = reader["name"].ToString();

target.Cost = Decimal.Parse(reader["cost"].ToString());

target.Minutes = Int32.Parse(reader["minutes"].ToString());

}

public Service Get(string name)

{

var obj = new Service();

string sql = string.Format(@"

SELECT \*

FROM SERVICE

WHERE NAME = '{0}'

", name);

ExecuteReader(sql, obj, Populate);

return obj;

}

public Service Get(int id)

{

var obj = new Service();

string sql = string.Format(@"

SELECT \*

FROM SERVICE

WHERE SERVICE\_ID = {0}

", id);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Service> Get()

{

var obj = new List<Service>();

string sql = string.Format(@"

SELECT \*

FROM SERVICE

");

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Service> GetServicesByClinicId(int clinicId)

{

var obj = new List<Service>();

string sql = string.Format(@"

SELECT s.\*

FROM SERVICE s

JOIN SERVICE\_CLINIC sc ON s.SERVICE\_ID = sc.SERVICE\_ID

WHERE sc.CLINIC\_ID = {0}

", clinicId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Service> GetServicesByProviderId(int providerId)

{

var obj = new List<Service>();

string sql = string.Format(@"

SELECT s.\*

FROM SERVICE s

JOIN PROVIDER\_QUALIFICATION pq ON s.SERVICE\_ID = pq.SERVICE\_ID

WHERE pq.PROVIDER\_ID = {0}

", providerId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<Service> GetEligibleProviderServicesForUserId(int userId)

{

var services = new List<Service>();

var clinics = new List<Clinic.BO.Clinic>();

string sql = string.Format(@"

SELECT c.\*

FROM USERS u

JOIN ENTITY e ON u.ENTITY\_ID = e.ENTITY\_ID

JOIN PROVIDER p ON e.ENTITY\_ID = p.ENTITY\_ID

JOIN CLINIC c ON p.CLINIC\_ID = p.CLINIC\_ID

WHERE u.USER\_ID = {0}

", userId);

ExecuteReader(sql, clinics, DataLayer.ClinicDL.Populate);

foreach(var clinic in clinics)

{

foreach(var service in clinic.Services)

{

if (!services.Any(x => x.Id == service.Id))

{

services.Add(service);

}

}

}

return services;

}

public List<Service> GetProviderServicesForUserId(int userId)

{

var obj = new List<Service>();

string sql = string.Format(@"

SELECT s.\*

FROM SERVICE s

JOIN PROVIDER\_QUALIFICATION pq ON s.SERVICE\_ID = pq.SERVICE\_ID

JOIN PROVIDER p ON pq.PROVIDER\_ID = p.PROVIDER\_ID

JOIN ENTITY e ON p.ENTITY\_ID = e.ENTITY\_ID

JOIN USERS u ON e.ENTITY\_ID = u.ENTITY\_ID

WHERE u.USER\_ID = {0}

", userId);

ExecuteReader(sql, obj, Populate);

return obj;

}

public void Create(Service service)

{

string sql;

int id = GetNextVal(Sequences.Service);

sql = string.Format(@"

INSERT INTO SERVICE

(SERVICE\_ID, NAME, COST, MINUTES)

VALUES

({0},'{1}',{2},{3})

",

id,

service.Name,

service.Cost,

service.Minutes);

ExecuteQuery(sql);

service.Id = id;

}

public void AddToProvider(Service service, Provider provider)

{

string sql;

int id = GetNextVal(Sequences.ProviderQualification);

sql = string.Format(@"

INSERT INTO PROVIDER\_QUALIFICATION

(PROVIDER\_QUALIFICATION\_ID, PROVIDER\_ID, SERVICE\_ID)

VALUES

({0},{1},{2})

",

id,

provider.Id,

service.Id);

ExecuteQuery(sql);

}

public void AddToClinic(Service service, Clinic.BO.Clinic clinic)

{

var existingService = Get(service.Name);

if (existingService.Id == null)

{

Create(service);

} else

{

service = existingService;

}

string sql;

int id = GetNextVal(Sequences.ServiceClinic);

sql = string.Format(@"

INSERT INTO SERVICE\_CLINIC

(SERVICE\_CLINIC\_ID, CLINIC\_ID, SERVICE\_ID)

VALUES

({0},{1},{2})

",

id,

clinic.Id,

service.Id);

ExecuteQuery(sql);

}

public void DeleteFromProvider(Service service, Provider provider)

{

string sql = string.Format(@"

DELETE FROM PROVIDER\_QUALIFICATION

WHERE SERVICE\_ID = {0}

AND PROVIDER\_ID = {1}

",

service.Id,

provider.Id);

ExecuteQuery(sql);

}

public void DeleteFromClinic(Service service, Clinic.BO.Clinic clinic)

{

string sql = string.Format(@"

DELETE FROM SERVICE\_CLINIC

WHERE SERVICE\_ID = {0}

AND CLINIC\_ID = {1}

",

service.Id,

clinic.Id);

ExecuteQuery(sql);

}

/// <summary>

/// Deletes the service for all clinics and providers

/// </summary>

/// <param name="service"></param>

public void Delete(Service service)

{

var clinics = GetClinicsUsingService(service.Name);

var providers = GetProvidersUsingService(service.Name);

clinics.ForEach(x => DeleteFromClinic(service, x));

providers.ForEach(x => DeleteFromProvider(service, x));

string sql = string.Format(@"

DELETE FROM SERVICE

WHERE SERVICE\_ID = {0}",

service.Id);

ExecuteQuery(sql);

}

/// <summary>

/// Modifies the service for all clinics and providers

/// </summary>

/// <param name="service"></param>

public void Update(Service service)

{

string sql = string.Format(@"

UPDATE SERVICE

SET NAME = '{1}',

COST = '{2}',

MINUTES = {3}

WHERE SERVICE\_ID = {0}

",

service.Id,

service.Name,

service.Cost,

service.Minutes);

ExecuteQuery(sql);

}

private List<Clinic.BO.Clinic> GetClinicsUsingService(string serviceName)

{

var obj = new List<Clinic.BO.Clinic>();

string sql = string.Format(@"

SELECT c.\*

FROM CLINIC c

JOIN SERVICE\_CLINIC sc ON c.CLINIC\_ID = sc.CLINIC\_ID

JOIN SERVICE s ON sc.SERVICE\_ID = s.SERVICE\_ID

WHERE s.NAME = '{0}'

", serviceName);

ExecuteReader(sql, obj, DataLayer.ClinicDL.Populate);

return obj;

}

private List<Provider> GetProvidersUsingService(string serviceName)

{

var obj = new List<Provider>();

string sql = string.Format(@"

SELECT p.\*

FROM PROVIDER p

JOIN PROVIDER\_QUALIFICATION pq ON p.PROVIDER\_ID = pq.PROVIDER\_ID

JOIN SERVICE s ON pq.SERVICE\_ID = s.SERVICE\_ID

WHERE s.NAME = '{0}'

", serviceName);

ExecuteReader(sql, obj, DataLayer.ProviderDL.Populate);

return obj;

}

}

}

#### UserDL.cs

using Clinic.BO;

using Clinic.DL;

using Oracle.DataAccess.Client;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.DL

{

public class UserDL : DlBase

{

private string dummyPass = "dummy";

private void Populate(Object obj, OracleDataReader reader)

{

var target = (User)obj;

int entityId;

if (Int32.TryParse(reader["entity\_id"].ToString(), out entityId))

{

DataLayer.EntityDL.Get(entityId).CopyTo(target);

}

target.Id = Convert.ToInt32(reader["user\_id"]);

target.Username = reader["username"].ToString();

target.Password = dummyPass;

target.Role = DataLayer.RoleDL.Get(Convert.ToInt32(reader["role\_id"]));

}

public List<User> GetAdmins()

{

var obj = new List<User>();

string sql = string.Format(@"

SELECT \*

FROM USERS u

JOIN ROLES r ON u.ROLE\_ID = r.ROLE\_ID

WHERE ROLE\_NAME = '{0}'

", Constants.Admin);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<User> GetClients()

{

var obj = new List<User>();

string sql = string.Format(@"

SELECT \*

FROM USERS u

JOIN ROLES r ON u.ROLE\_ID = r.ROLE\_ID

WHERE ROLE\_NAME = '{0}'

", Constants.User);

ExecuteReader(sql, obj, Populate);

return obj;

}

public List<User> GetStaff()

{

var obj = new List<User>();

string sql = string.Format(@"

SELECT \*

FROM USERS u

JOIN ROLES r ON u.ROLE\_ID = r.ROLE\_ID

WHERE ROLE\_NAME IN ('{0}','{1}')

", Constants.Admin, Constants.Office);

ExecuteReader(sql, obj, Populate);

return obj;

}

public User Get(string userName, string password)

{

var hashedPass = password.GetHashCode();

var obj = new User();

string sql = string.Format(@"

SELECT \*

FROM USERS

WHERE USERNAME = '{0}' AND PASSWORD = '{1}'

", userName, hashedPass.ToString());

ExecuteReader(sql, obj, Populate);

return obj;

}

public void Create(User user)

{

var id = GetNextVal(Sequences.User);

var hashedPass = user.Password.GetHashCode();

string sql = string.Empty;

if (user.EntityId.HasValue)

{

sql = string.Format(@"

INSERT INTO USERS

(USER\_ID, USERNAME, PASSWORD, ROLE\_ID, ENTITY\_ID)

VALUES

({0},'{1}','{2}',{3},{4})

", id, user.Username, hashedPass, user.Role.Id, user.EntityId);

}

else

{

sql = string.Format(@"

INSERT INTO USERS

(USER\_ID, USERNAME, PASSWORD, ROLE\_ID)

VALUES

({0},'{1}','{2}',{3})

", id, user.Username, hashedPass.ToString(), user.Role.Id);

}

ExecuteQuery(sql);

}

public void Update(User user)

{

string sql = string.Empty;

if (!string.IsNullOrWhiteSpace(user.Password) && user.Password != dummyPass)

{

var hashedPass = user.Password.GetHashCode();

sql = string.Format(@"

UPDATE USERS

SET USERNAME = '{1}',

PASSWORD = '{2}',

ROLE\_ID = {3},

ENTITY\_ID = {4}

WHERE USER\_ID = {0}

",

user.Id,

user.Username,

hashedPass,

user.Role.Id,

user.EntityId);

}

else

{

sql = string.Format(@"

UPDATE USERS

SET USERNAME = '{1}',

ROLE\_ID = {2},

ENTITY\_ID = {3}

WHERE USER\_ID = {0}

",

user.Id,

user.Username,

user.Role.Id,

user.EntityId == null ? "NULL" : user.EntityId.ToString());

}

ExecuteQuery(sql);

}

public void Delete(User user)

{

var sql = string.Format(@"

DELETE FROM USERS

WHERE USER\_ID = {0}

", user.Id);

ExecuteQuery(sql);

}

}

}

CurSession.cs

using Clinic.BO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.Utilities

{

public static class CurSession

{

public static User User

{

get

{

var user = HttpContext.Current.Session["User"];

return user != null ? (BO.User)user : null;

}

set

{

HttpContext.Current.Session["User"] = value;

}

}

}

}

### Utility Classes

#### CustomException.cs

using Clinic.BO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.Utilities

{

public class CustomException : Exception

{

public CustomException() {

}

public CustomException(List<ErrorMessage> errorMessages)

{

ErrorMessages = errorMessages;

}

public List<ErrorMessage> ErrorMessages { get; set; }

}

}

#### GlobalSettings.cs

using Clinic.BL;

using Clinic.BO;

using Clinic.DL;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.Utilities

{

[JsonObject(MemberSerialization.OptIn)]

public static class GlobalSettings

{

private static Nullable<bool> \_adminExists;

[JsonProperty]

public static bool AdminExists

{

get

{

if (!\_adminExists.HasValue)

{

CheckForAdmin();

}

return \_adminExists.Value;

}

}

[JsonProperty]

public static User User

{

get

{

return CurSession.User;

}

}

public static void CheckForAdmin() {

var admins = BusinessLayer.UserBL.GetAdmins();

\_adminExists = admins.Count > 0;

}

public static string GetJson() {

dynamic obj = new System.Dynamic.ExpandoObject();

obj.AdminExists = AdminExists;

obj.User = User;

return JsonConvert.SerializeObject(obj);

}

}

}

#### InjectionValidator.cs

using Clinic.BO;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Clinic.Utilities

{

public static class InjectionValidator

{

/// <summary>

/// Throws SqlInjectionException if not valid

/// </summary>

/// <param name="fields"></param>

public static void Validate(List<Field> fields)

{

var errorMessages = new List<ErrorMessage>();

foreach (var field in fields)

{

var protectedWords = "";

if (field.Value.ToLower().Contains("insert"))

{

protectedWords += "INSERT, ";

}

if (field.Value.ToLower().Contains("drop table"))

{

protectedWords += "DROP, ";

}

if (field.Value.ToLower().Contains("select"))

{

protectedWords += "SELECT, ";

}

if (field.Value.ToLower().Contains("delete"))

{

protectedWords += "DELETE, ";

}

if (field.Value.ToLower().Contains("update"))

{

protectedWords += "UPDATE, ";

}

if (field.Value.ToLower().Contains("create"))

{

protectedWords += "CREATE, ";

}

if (protectedWords.Length > 0)

{

errorMessages.Add(new ErrorMessage(field.Name, "Prohibited keywords: " + protectedWords));

}

}

if (errorMessages.Count > 0)

{

throw new CustomException(errorMessages);

}

}

}

}

#### JsonParser.cs

using Clinic.BO;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Script.Serialization;

namespace Clinic.Utilities

{

public static class JsonParser

{

public static T FromJson<T>(HttpContext context)

{

var jsonString = String.Empty;

var stream = context.Request.InputStream;

stream.Position = 0;

using (var inputStream = new StreamReader(stream))

{

jsonString = inputStream.ReadToEnd();

}

return new JavaScriptSerializer().Deserialize<T>(jsonString);

}

public static string ToJson(Object obj)

{

return new JavaScriptSerializer().Serialize(obj);

}

public static string ExceptionToJson (Exception ex)

{

List<ErrorMessage> errorMessages;

if (ex.GetType() == typeof(CustomException))

{

errorMessages = ((CustomException)ex).ErrorMessages;

}

else

{

errorMessages = new List<ErrorMessage>();

errorMessages.Add(new ErrorMessage("", ex.Message));

}

return "{\"errorMessages\":" + new JavaScriptSerializer().Serialize(errorMessages) + "}";

}

}

}

### SQL Create/Drop

#### CREATE.sql

CREATE TABLE ENTITY

(

ENTITY\_ID NUMBER NOT NULL,

NAME1 VARCHAR2(50) NOT NULL,

NAME2 VARCHAR2(50),

NAME3 VARCHAR2(50),

ADDRESS1 VARCHAR2(50) NOT NULL,

ADDRESS2 VARCHAR2(50),

CITY VARCHAR2(50) NOT NULL,

STATE VARCHAR2(2) NOT NULL,

ZIP VARCHAR2(10) NOT NULL,

PHONE1 VARCHAR2(15) NOT NULL,

PHONE2 VARCHAR2(15),

PHONE3 VARCHAR2(15)

)

/

ALTER TABLE ENTITY ADD CONSTRAINT ENTITY\_PK PRIMARY KEY ( ENTITY\_ID )

/

CREATE SEQUENCE ENTITY\_S START WITH 1 ORDER

/

CREATE TABLE ROLES

(

ROLE\_ID NUMBER NOT NULL,

ROLE\_NAME VARCHAR2(50) NOT NULL

)

/

ALTER TABLE ROLES ADD CONSTRAINT ROLE\_PK PRIMARY KEY ( ROLE\_ID )

/

ALTER TABLE ROLES ADD CONSTRAINT ROLE\_UC1 UNIQUE (ROLE\_NAME)

/

INSERT INTO ROLES (ROLE\_ID, ROLE\_NAME) VALUES (1, 'Administrator');

INSERT INTO ROLES (ROLE\_ID, ROLE\_NAME) VALUES (2, 'Office');

INSERT INTO ROLES (ROLE\_ID, ROLE\_NAME) VALUES (3, 'User');

CREATE TABLE USERS

(

USER\_ID NUMBER NOT NULL,

ROLE\_ID NUMBER NOT NULL,

ENTITY\_ID NUMBER NULL,

USERNAME VARCHAR2(50) NOT NULL,

PASSWORD VARCHAR2(500) NOT NULL,

IS\_ACTIVE CHAR(1) DEFAULT 'Y'

)

/

ALTER TABLE USERS ADD CONSTRAINT USERS\_PK PRIMARY KEY ( USER\_ID )

/

ALTER TABLE USERS ADD CONSTRAINT USERS\_FK1 FOREIGN KEY ( ENTITY\_ID ) REFERENCES ENTITY ( ENTITY\_ID ) ON DELETE CASCADE

/

ALTER TABLE USERS ADD CONSTRAINT USERS\_FK2 FOREIGN KEY ( ROLE\_ID ) REFERENCES ROLES ( ROLE\_ID ) ON DELETE CASCADE

/

ALTER TABLE USERS ADD CONSTRAINT USERS\_UC1 UNIQUE (USERNAME)

/

CREATE SEQUENCE USERS\_S START WITH 1 ORDER

/

CREATE TABLE HOUSEHOLD

(

HOUSEHOLD\_ID NUMBER NOT NULL,

INSURANCE\_NAME VARCHAR2(50) NULL,

POLICY\_NUMBER VARCHAR2(50) NULL,

GROUP\_NUMBER VARCHAR2(50) NULL

)

/

ALTER TABLE HOUSEHOLD ADD CONSTRAINT HOUSEHOLD\_PK PRIMARY KEY ( HOUSEHOLD\_ID )

/

CREATE SEQUENCE HOUSEHOLD\_S START WITH 1 ORDER;

COMMIT;

ALTER SEQUENCE HOUSEHOLD\_S INCREMENT BY 10000;

SELECT HOUSEHOLD\_S.NEXTVAL FROM dual;

/

ALTER SEQUENCE HOUSEHOLD\_S INCREMENT BY 1;

/

CREATE TABLE RELATIONSHIP

(

RELATIONSHIP\_ID NUMBER NOT NULL,

NAME VARCHAR2(50) NOT NULL

)

/

ALTER TABLE RELATIONSHIP ADD CONSTRAINT RELATIONSHIP\_PK PRIMARY KEY ( RELATIONSHIP\_ID )

/

ALTER TABLE RELATIONSHIP ADD CONSTRAINT RELATIONSHIP\_UC1 UNIQUE (NAME)

/

INSERT INTO RELATIONSHIP (RELATIONSHIP\_ID, NAME) VALUES (1, 'Primary');

INSERT INTO RELATIONSHIP (RELATIONSHIP\_ID, NAME) VALUES (2, 'Spouse');

INSERT INTO RELATIONSHIP (RELATIONSHIP\_ID, NAME) VALUES (3, 'Partner');

INSERT INTO RELATIONSHIP (RELATIONSHIP\_ID, NAME) VALUES (4, 'Child');

INSERT INTO RELATIONSHIP (RELATIONSHIP\_ID, NAME) VALUES (5, 'Dependent Child');

INSERT INTO RELATIONSHIP (RELATIONSHIP\_ID, NAME) VALUES (6, 'Dependent Adult');

CREATE TABLE HOUSEHOLD\_PERSON

(

HOUSEHOLD\_PERSON\_ID NUMBER NOT NULL,

RELATIONSHIP\_ID NUMBER NOT NULL,

HOUSEHOLD\_ID NUMBER NOT NULL,

ENTITY\_ID NUMBER NOT NULL,

IS\_PAYER CHAR (1) NOT NULL,

DOB DATE

)

/

ALTER TABLE HOUSEHOLD\_PERSON ADD CONSTRAINT HOUSEHOLD\_PERSON\_PK PRIMARY KEY ( HOUSEHOLD\_PERSON\_ID )

/

ALTER TABLE HOUSEHOLD\_PERSON ADD CONSTRAINT HOUSEHOLD\_PERSON\_FK1 FOREIGN KEY ( HOUSEHOLD\_ID ) REFERENCES HOUSEHOLD ( HOUSEHOLD\_ID ) ON DELETE CASCADE

/

ALTER TABLE HOUSEHOLD\_PERSON ADD CONSTRAINT HOUSEHOLD\_PERSON\_FK2 FOREIGN KEY ( ENTITY\_ID ) REFERENCES ENTITY ( ENTITY\_ID ) ON DELETE CASCADE

/

ALTER TABLE HOUSEHOLD\_PERSON ADD CONSTRAINT HOUSEHOLD\_PERSON\_FK3 FOREIGN KEY ( RELATIONSHIP\_ID ) REFERENCES RELATIONSHIP ( RELATIONSHIP\_ID ) ON DELETE CASCADE

/

ALTER TABLE HOUSEHOLD\_PERSON ADD CONSTRAINT HOUSEHOLD\_PERSON\_UC1 UNIQUE (HOUSEHOLD\_ID, ENTITY\_ID)

/

CREATE SEQUENCE HOUSEHOLD\_PERSON\_S START WITH 1 ORDER

/

CREATE TABLE CLINIC

(

CLINIC\_ID NUMBER NOT NULL,

ENTITY\_ID NUMBER NOT NULL

)

/

ALTER TABLE CLINIC ADD CONSTRAINT CLINIC\_PK PRIMARY KEY ( CLINIC\_ID )

/

ALTER TABLE CLINIC ADD CONSTRAINT CLINIC\_FK1 FOREIGN KEY ( ENTITY\_ID ) REFERENCES ENTITY ( ENTITY\_ID ) ON DELETE CASCADE

/

CREATE SEQUENCE CLINIC\_S START WITH 1 ORDER

/

CREATE TABLE ROOM

(

ROOM\_ID NUMBER NOT NULL,

CLINIC\_ID NUMBER NOT NULL,

NAME VARCHAR2(50) NOT NULL

)

/

ALTER TABLE ROOM ADD CONSTRAINT ROOM\_PK PRIMARY KEY ( ROOM\_ID )

/

ALTER TABLE ROOM ADD CONSTRAINT ROOM\_FK1 FOREIGN KEY ( CLINIC\_ID ) REFERENCES CLINIC ( CLINIC\_ID ) ON DELETE CASCADE

/

CREATE SEQUENCE ROOM\_S START WITH 1 ORDER

/

CREATE TABLE SERVICE

(

SERVICE\_ID NUMBER NOT NULL,

NAME VARCHAR2(50) NOT NULL,

COST NUMBER (6,2) NOT NULL,

MINUTES NUMBER NOT NULL

)

/

ALTER TABLE SERVICE ADD CONSTRAINT SERVICE\_PK PRIMARY KEY ( SERVICE\_ID )

/

ALTER TABLE SERVICE ADD CONSTRAINT SERVICE\_UC1 UNIQUE (NAME)

/

CREATE SEQUENCE SERVICE\_S START WITH 1 ORDER

/

CREATE TABLE SERVICE\_CLINIC

(

SERVICE\_CLINIC\_ID NUMBER NOT NULL,

CLINIC\_ID NUMBER NOT NULL,

SERVICE\_ID NUMBER NOT NULL

)

/

ALTER TABLE SERVICE\_CLINIC ADD CONSTRAINT SERVICE\_CLINIC\_PK PRIMARY KEY ( SERVICE\_CLINIC\_ID )

/

ALTER TABLE SERVICE\_CLINIC ADD CONSTRAINT SERVICE\_CLINIC\_FK1 FOREIGN KEY ( CLINIC\_ID ) REFERENCES CLINIC ( CLINIC\_ID ) ON DELETE CASCADE

/

ALTER TABLE SERVICE\_CLINIC ADD CONSTRAINT SERVICE\_CLINIC\_FK2 FOREIGN KEY ( SERVICE\_ID ) REFERENCES SERVICE ( SERVICE\_ID ) ON DELETE CASCADE

/

ALTER TABLE SERVICE\_CLINIC ADD CONSTRAINT SERVICE\_CLINIC\_UC1 UNIQUE ( CLINIC\_ID, SERVICE\_ID )

/

CREATE SEQUENCE SERVICE\_CLINIC\_S START WITH 1 ORDER

/

CREATE TABLE PROVIDER

(

PROVIDER\_ID NUMBER NOT NULL,

CLINIC\_ID NUMBER NOT NULL,

ENTITY\_ID NUMBER NOT NULL

)

/

ALTER TABLE PROVIDER ADD CONSTRAINT PROVIDER\_PK PRIMARY KEY ( PROVIDER\_ID )

/

ALTER TABLE PROVIDER ADD CONSTRAINT PROVIDER\_FK1 FOREIGN KEY ( CLINIC\_ID ) REFERENCES CLINIC ( CLINIC\_ID ) ON DELETE CASCADE

/

ALTER TABLE PROVIDER ADD CONSTRAINT PROVIDER\_FK2 FOREIGN KEY ( ENTITY\_ID ) REFERENCES ENTITY ( ENTITY\_ID ) ON DELETE CASCADE

/

CREATE SEQUENCE PROVIDER\_S START WITH 1 ORDER

/

CREATE TABLE PROVIDER\_QUALIFICATION

(

PROVIDER\_QUALIFICATION\_ID NUMBER NOT NULL,

PROVIDER\_ID NUMBER NOT NULL,

SERVICE\_ID NUMBER NOT NULL

)

/

ALTER TABLE PROVIDER\_QUALIFICATION ADD CONSTRAINT PROVIDER\_QUALIFICATION\_PK PRIMARY KEY ( PROVIDER\_QUALIFICATION\_ID )

/

ALTER TABLE PROVIDER\_QUALIFICATION ADD CONSTRAINT PROVIDER\_QUALIFICATION\_FK1 FOREIGN KEY ( PROVIDER\_ID ) REFERENCES PROVIDER ( PROVIDER\_ID ) ON DELETE CASCADE

/

ALTER TABLE PROVIDER\_QUALIFICATION ADD CONSTRAINT PROVIDER\_QUALIFICATION\_FK2 FOREIGN KEY ( SERVICE\_ID ) REFERENCES SERVICE ( SERVICE\_ID ) ON DELETE CASCADE

/

ALTER TABLE PROVIDER\_QUALIFICATION ADD CONSTRAINT PROVIDER\_QUALIFICATION\_UC1 UNIQUE (PROVIDER\_ID, SERVICE\_ID)

/

CREATE SEQUENCE PROVIDER\_QUALIFICATION\_S START WITH 1 ORDER

/

CREATE TABLE APPOINTMENT

(

APPOINTMENT\_ID NUMBER NOT NULL,

CLINIC\_ID NUMBER NOT NULL,

HOUSEHOLD\_PERSON\_ID NUMBER NOT NULL

)

/

ALTER TABLE APPOINTMENT ADD CONSTRAINT APPOINTMENT\_PK PRIMARY KEY ( APPOINTMENT\_ID )

/

ALTER TABLE APPOINTMENT ADD CONSTRAINT APPOINTMENT\_FK1 FOREIGN KEY ( HOUSEHOLD\_PERSON\_ID ) REFERENCES HOUSEHOLD\_PERSON ( HOUSEHOLD\_PERSON\_ID ) ON DELETE CASCADE

/

CREATE SEQUENCE APPOINTMENT\_S START WITH 1 ORDER

/

CREATE TABLE APPOINTMENT\_SERVICE

(

APPOINTMENT\_SERVICE\_ID NUMBER NOT NULL,

APPOINTMENT\_ID NUMBER NOT NULL,

PROVIDER\_ID NUMBER NOT NULL,

SERVICE\_ID NUMBER NOT NULL,

ROOM\_ID NUMBER NOT NULL,

COST NUMBER (6,2) NOT NULL,

TIME DATE NOT NULL,

MINUTES NUMBER NOT NULL

)

/

ALTER TABLE APPOINTMENT\_SERVICE ADD CONSTRAINT APPOINTMENT\_SERVICE\_PK PRIMARY KEY ( APPOINTMENT\_SERVICE\_ID )

/

ALTER TABLE APPOINTMENT\_SERVICE ADD CONSTRAINT APPOINTMENT\_SERVICE\_FK1 FOREIGN KEY ( APPOINTMENT\_ID ) REFERENCES APPOINTMENT ( APPOINTMENT\_ID ) ON DELETE CASCADE

/

ALTER TABLE APPOINTMENT\_SERVICE ADD CONSTRAINT APPOINTMENT\_SERVICE\_FK2 FOREIGN KEY ( SERVICE\_ID ) REFERENCES SERVICE ( SERVICE\_ID ) ON DELETE CASCADE

/

ALTER TABLE APPOINTMENT\_SERVICE ADD CONSTRAINT APPOINTMENT\_SERVICE\_FK3 FOREIGN KEY ( ROOM\_ID ) REFERENCES ROOM ( ROOM\_ID ) ON DELETE CASCADE

/

ALTER TABLE APPOINTMENT\_SERVICE ADD CONSTRAINT APPOINTMENT\_SERVICE\_FK4 FOREIGN KEY ( PROVIDER\_ID ) REFERENCES PROVIDER ( PROVIDER\_ID ) ON DELETE CASCADE

/

CREATE SEQUENCE APPOINTMENT\_SERVICE\_S START WITH 1 ORDER

/

COMMIT;

#### DROP.sql

--select 'drop table ', table\_name, 'cascade constraints;' from user\_tables;

--select 'drop sequence ', sequence\_name, ';' from user\_sequences;

drop table ENTITY cascade constraints;

drop table ROLES cascade constraints;

drop table USERS cascade constraints;

drop table HOUSEHOLD cascade constraints;

drop table RELATIONSHIP cascade constraints;

drop table HOUSEHOLD\_PERSON cascade constraints;

drop table CLINIC cascade constraints;

drop table ROOM cascade constraints;

drop table SERVICE cascade constraints;

drop table SERVICE\_CLINIC cascade constraints;

drop table PROVIDER cascade constraints;

drop table PROVIDER\_QUALIFICATION cascade constraints;

drop table APPOINTMENT cascade constraints;

drop table APPOINTMENT\_SERVICE cascade constraints;

drop table PAYMENT cascade constraints;

drop sequence APPOINTMENT\_S ;

drop sequence APPOINTMENT\_SERVICE\_S ;

drop sequence CLINIC\_S ;

drop sequence ENTITY\_S ;

drop sequence HOUSEHOLD\_PERSON\_S ;

drop sequence HOUSEHOLD\_S ;

drop sequence PAYMENT\_S ;

drop sequence PROVIDER\_QUALIFICATION\_S ;

drop sequence PROVIDER\_S ;

drop sequence ROOM\_S ;

drop sequence SERVICE\_CLINIC\_S ;

drop sequence SERVICE\_S ;

drop sequence USERS\_S ;

# Project Management Section

## Weekly Status Reports / Activities & Time

### End of Week 1

* Last Week (7.5 hours)
  + Studied requirements (1 hour)
  + ER Diagram 1st Draft (1 hour)
  + Considering possible UI design (2 hours)
  + Considering back end design (0.5 hours)
  + Research Entity Framework (3 hour)
* This Week
  + Finalize database design
  + Finalize front-end and back-end design
  + Research Entity Framework
  + Write requirements
  + Finalize Users/Roles
* Challenges
  + Had some slip-ups when analyzing the requirements

### End of Week 2

* Last Week (10 hours)
  + Finalized database design (3 hours)
  + Research Entity Framework (4 hours)
    - Throwing out Entity Framework – Non-functional for .NET 4.0
  + Wrote requirements based upon initial requirements doc (1.5 hours)
  + Finalized Users/Roles (0.5 hours)
  + Finalized front-end framework (0.5 hours)
    - Single Page Application, Angular JS, Bootstrap-UI, Angular-UI, LESS, AJAX-JSON
  + Finalize back-end framework (0.5 hours)
    - Microsoft Web API
    - Layered approach (Controller / Business Layer / Data Layer)
    - Data Layer will utilize inline SQL
  + Linked app code to Github for source control and project management
* This Week
  + Begin implementing features
    - Establish UI Framework and tooling
    - Establish back end framework layers and achieve Oracle connection
    - Landing Page
    - Logo
    - Login / Security
    - Navigation
    - Encryption / Decryption of passwords and Tax ID (Social Security Number)
* Challenges
  + Entity Framework
    - Struggled a lot trying to get it to work despite many tutorials working successfully
    - Research shows that Oracle and Entity Framework don’t play nicely in .NET 4.0

### End of Week 3

* Last Week (9.5 hours)
  + Finalized database design (2.5 hours)
  + Research Entity Framework (4 hours)
  + Throwing out Entity Framework – Non-functional for .NET 4.0
  + Wrote requirements based upon initial requirements doc (1.5 hours)
  + Finalized Users/Roles (0.5 hours)
  + Finalized front-end framework (0.5 hours)
  + Single Page Application, Angular JS, Bootstrap-UI, Angular-UI, LESS, AJAX-JSON
  + Finalize back-end framework (0.5 hours)
  + Microsoft Web API
  + Layered approach (Controller / Business Layer / Data Layer)
  + Data Layer will utilize inline SQL
* This Week
  + Begin implementing features
  + Establish UI Framework and tooling
  + Establish back end framework layers and achieve Oracle connection
  + Landing Page
  + Logo
  + Login / Security
  + Navigation
  + Encryption / Decryption of passwords and Tax ID (Social Security Number)
* Challenges
  + Spent a lot of time struggling with Entity Framework
  + Couldn’t get it working despite many videos/tutorials showing it working
  + Research shows that Oracle doesn’t play nicely with .NET 4.0 and Entity Framework

**End of Week 4**

* Last Week (24.8 hours)
  + Implementing features
  + Established UI Framework and tooling (12 hours)
  + Establish back end framework layers and achieve Oracle connection (6 hours)
  + Landing Page (not logged in) (1 hours)
  + Logo (.15 hours)
  + Login / Security (pending) (3 hours)
  + Navigation (0.5 hours)
  + Encryption/Hashing of sensitive data (0.15 hours)
  + Establish/Test database hooks (2 hours)
* This Week
  + Login (part 2)
  + Full database integration
  + Landing page (logged in)
  + Admin – Manage Users
  + Admin – Create Account
  + Start work on User profile management and dependent management
* Challenges
  + Challenges with .NET 4.0 and MVC framework compatibility
  + Seems that MVC and Web API prefer .NET 4.5
  + Settled on using ASMX services
  + Still obeying Web API standards and project layouts
  + Transmitting AJAX calls independent of .NET framework
  + Establishing Angular JS front end

**End of Week 5**

* Last Week (20 hours)
  + Update DDL to reflect new requirements (3 hours)
  + Update code to reflect new DDL
  + Login (part 2) (10 hours)
  + Full database integration
  + Establish initial permissions for logged in users
  + Register (4 hours)
  + Landing page (logged in) (2 hours)
  + Update Documentation (1 hour)
  + Admin – Manage Users (pending…)
  + Admin – Create Account (pending…)
  + Start work on User profile management and dependent management (pending…)
* This Week
  + Admin – Manage Users
  + Admin – Create Account
  + Start work on User profile management and dependent management
* Challenges
  + Some difficulties getting C# to generate dynamic JSON
  + Time!

**End of Week 6**

* Last Week (27.5 hours)
  + Develop side navigation (1.5 hours)
  + Admin – Manage Users (staff or clients (10 hours)
  + Create/Update/Delete User
  + Admin – Profile Edit (staff or client) - INCOMPLETE (4 hours)
  + Profile Management (incomplete) (8 hours)
  + Dependent management (pending…)
  + SQL injection and general exception bubbling (4 hours)
  + All errors bubbled up and “toasted” to user or thrown out
* This Week
  + Admin – Profile Edit (staff or client)
  + Manage Clinics
  + Manage Qualifications
  + Manage Appointments
* Challenges
  + Had to tweak how JSON is generated:
  + from browser to server
  + from server to browser

**End of Week 7**

* Last Week (24 hours)
  + Admin – Profile Edit (staff or client) (12 hours)
  + Manage Clinics/Services/Rooms (12 hours)
  + Providers (complete…)
  + Services (complete…)
  + Rooms (pending…)
  + Manage Qualifications (pending…)
  + Manage Appointments (pending…)
  + Dependent management (pending…)
* This Week
  + Delete capability (all areas)
  + Create/Edit already in place
  + Confirmation dialogs
  + Manage Clinics
  + Rooms
  + Manage Qualifications
  + Manage Appointments
  + Reports
  + Documentation Updates
  + Will we finish on time?
  + ANSWER: “IFFY”
* Challenges
  + Time

**End of Week 8**

* Last Week
  + Completed all major features with exception of billing (52 hours)

## Total Time Spent

175.3 hours = 22 8-hour days = 4.38 work weeks

WOW!!!!

# Project Management

## Difficulties Encountered

Please refer to “Challenges” section in each weekly report

## Technical Reports Used To Solve Difficulties

99% of solutions to problems came from http://www.stackoverflow.com

## Other Technical Information

NONE