Project Step 7 (Portfolio Assignment)

URL: http://flip1.engr.oregonstate.edu:9222/

1. **Names**: Brandon Schultz, Robert Collins
2. **Group:** 85

Project Title: Pharma-Mate: A Precursor to an Automated Pharmacy Experience

Brandon Schultz

Summary of Feedback:

The changes made throughout the course of creating the final project for CS 340 were immense, in the most unfortunate, yet realistic of ways. By this, I mean that in its original form we attempted to try and create a replacement for an entire industry, but in its final form it's a patchwork site with a database backend that only partially works. It does, look nice, however.

The initial, ambitious nature of our project is exemplified in the complexity of initial table names and attempted functionality of them. Originally, tables like Medications were titled things like "MedicationProviderInformation" and contained properties that are now distributed amongst three separate tables, (Medications, Side Effects, Diseases Treated). Complications like these were the main hurdles faced throughout the first half of the quarter. After getting thorough feedback on our schema and er diagram in weeks 4-5, those critiques stopped, and the project began to resemble its final form from a database perspective.

What become a giant challenge was the website itself. The HTML components were fine, and the initial plan was to just use JS and NodeJS with handlebars, however, to get experience with python, Brandon lead the way down a flask filled bottleneck of snakes before deciding to go back to using JS, NodeJS and handlebars. This was a huge time sink and is one of the reasons for the poor state of some components of the site/project.

The last course I took at OSU was CS290, a little over a year ago. I very much spent more time going through and re-learning web development than I did working with databases for this project. Time management was an issue a year ago, and in a way, it affected how this project turned out again, despite what I believe to genuinely be my best effort. The time used to try to learn flask/python before going back with false confidence to JS, Node and handlebars put us about a week behind from about week eight until now.

The project is what it is though. I have put in more time and effort for this assignment than any other at OSU, and although it isn't what I aimed for when beginning it, I did learn a lot and appreciated how this course was ran and taught.

Brandon Schultz

Overview:

Historically, employee efficiency has increased dramatically in response to the addition of new technologies. Computers have revolutionized the workforce of the last 50 years and will continue to do so as progress in fields like automation is made. Ideally, automation of tasks traditionally completed by employees results in more effective work being completed to a degree of quality not previously possible. This is not always the case, as made evident by the existence of self-checkout lanes at nearly all grocery stores. Eventually, we will be able to complete a purchase without requiring the assistance of an in-person employee and no longer will their presence be required, nor will their job need to exist.

Pharmacies and the employees that work in them appear vulnerable to a similar, automated model. In their current state, pharmacies exist primarily as a means of supplying some item(s) to a customer, much like a grocery store. Pharmacy Techs exist in a role like that of a cashier, however, pharmacists and the medicines they manage resemble a more complex system of tasks. Our project will explore the feasibility of mimicking the role and knowledge of a pharmacist by attempting to manage and link an individual customer to their prescribed medicines and associated documentation through the creation of a web app containing a database backend.

The project aims to resemble the functionality of a mobile app, in that one user would be associated with one “Customer” entity, and thus also be associated with a single “CustomerRecords” entity.

The “Medications” entity represents the initial interaction between an individual customer’s prescription and the worker tasked with determining if said prescriptions requirements can be met and consequentially filled, then provided to the customer/user. It is a branching point for the unique specifications regarding the customer/user’s order, as it’s PK value will be the name of one of ten medicines that is used by “DiseasesTreated”, “SideEffects”, and “Inventories” entities.

“DiseasesTreated” and “SideEffects” will consist of the most common ailments treated and potential side effects caused by one of the ten medicines depicted by the “medName” property value of the “Medications” entity. “Inventories” will use the “medID” property value of the “Medications” entity to determine if the users medicine is in stock or carried by the database and associate a medicine name to a medicine ID number. “Inventories” serves as the gross inventory of medicines that a pharmacy would contain and will contain information that associates a user/customer’s medicine to a medicine in the database.

Brandon Schultz

Database Outline, in Words:

Entity #1:

- **Customers**: Keeps track of various customer values and associated medications. A customer has a unique ID, a first and last name, and medication prescribed to them. SELECT, DELETE and UPDATE functions are planned to be used to change CustomerMedication and to remove customers from DB if medicine fails them. IdNum is used as PK to implement, and link Customer and CustomerOrder.
    - **Properties**:
        - **PK = *customerID**: int, auto_increment, unique, not NULL. Tables ID number property.
        - **fname**: Customers first name. Uses varchar, not NULL.
        - **lname**: Uses varchar, not NULL. Customers last name.
    - **Relationships**: M:1 to **CustomerRecords**

Entity #2:

- **CustomerRecords:** Information linking customers idNum to Medication entity. Contains information regarding individual, customers prescription.
    - **Properties**:
        - **PK = *recordID:** int, auto_increment, unique, not NULL Tables idNum property.
        - **FK = customersID:** int, auto_increment, unique, not NULL. FK connecting customerID property to CustomerRecords.
        - **FK2 = medID**: int, auto_increment, unique, not NULL. FK connecting to composite, Medication's entity.
        - **prescriptionCount**: varchar(255), not NULL. Number of medicines prescribed to patient, unique to prescribed medicine of customer.
        - **prescriptionDosage:** varchar(255), not NULL. Customers prescribed medicine's specific dosing criteria.
    - **Relationships**: 1:M to **Customers, Medications**

Entity #3:

- **Medications**: Prescribing information for medicines in inventory.
    - **Properties**:
        - **PK = *medID**: int, auto_increment, unique, not NULL. CustomerRecords, Inventories, DiseasesTreated, and SideEffects all are linked to Medications table via this property.
        - **medName:** Uses varchar(255), not NULL. Name of medicine used as composite to form M:M between CustomerRecords and DiseasesTreated, SideEffects and inventories entities.
    - **Relationships**:
        - M:1 to **CustomerRecords, DiseasesTreated, SideEffects, Inventories**

Entity #4:

- **DiseasesTreated:** Diseases treated by medicine.
    - **Properties:**
        - **PK = *diseaseID:** int, auto_increment, unique, not NULL. Tables ID num.

        - **FK= medID: :** int, not NULL. . ID of medicine used to associate multiple diseases that can be treated using a single medicine.
        - **disease**: varchar(255), not NULL. The most common illness treated by medication is returned.
    - **Relationships**: 1:M to **Medications**

Entity #5:

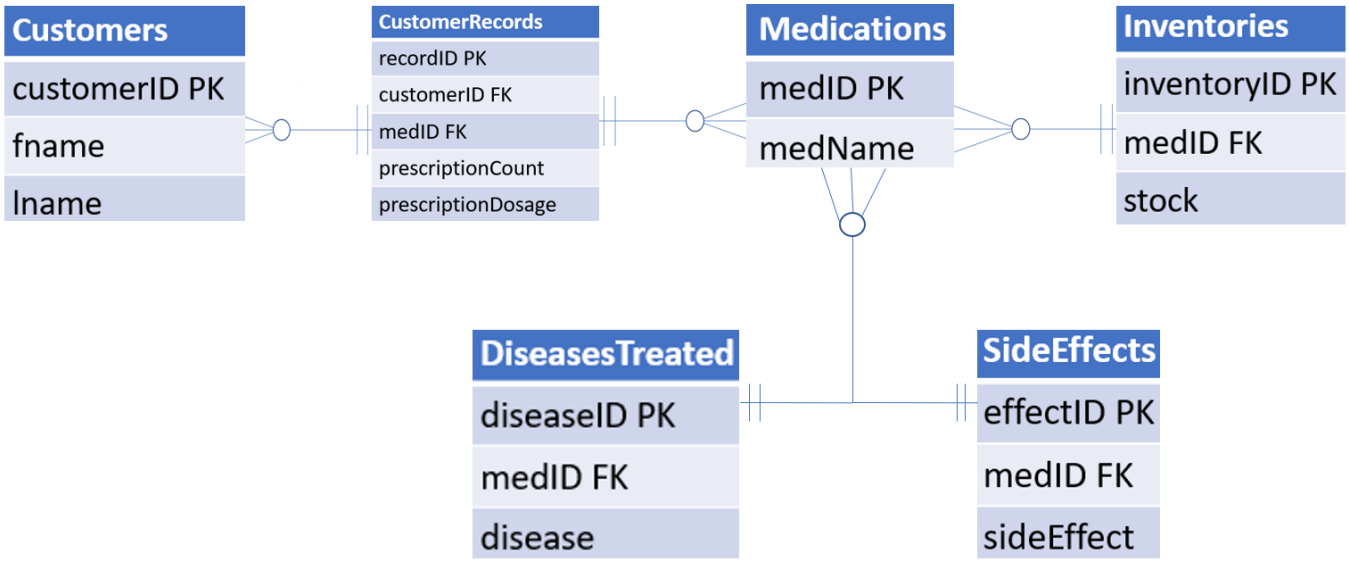
- **SideEffects:** Side effects of prescribed medicine
    - **Properties:**
        - **PK = * effectID:** int, auto_increment, unique, not NULL. Tables ID num.
        - **FK = medID:** int, not NULL. . ID of medicine used to associate multiple side effects to a single medicine.
        - **sideEffect:** varchar(255), NOT NULL**.** Most common side effect of medication is returned via a string of chars.
    - **Relationships**: : 1:M to **Medications**

Entity #6:

- **Inventories**: Keeps track of medicine inventory available for dispensing to patients.
    - **Properties**:
        - **PK = *inventoryID**: int, auto_increment, unique, not NULL. ID num that distinguishes each medication in inventory.
        - **FK = medID:** varchar, not NULL. ID of medication stocked.
        - **stock:** varchar(255), NOT NULL**.**. Value of medicine held in stock.
    - **Relationships**: : 1:M to **Medications**

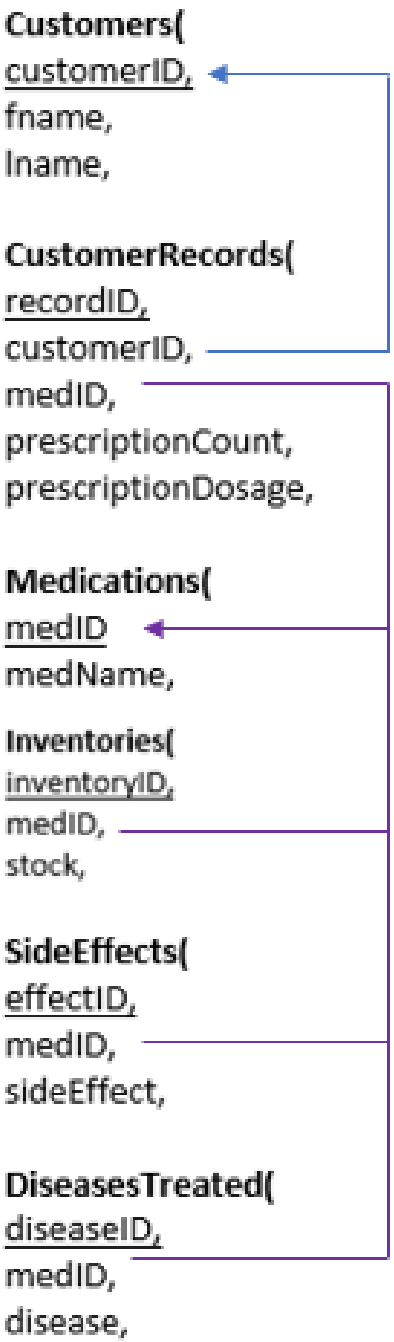
Brandon Schultz

Entity-Relationship Diagram:



Brandon Schultz

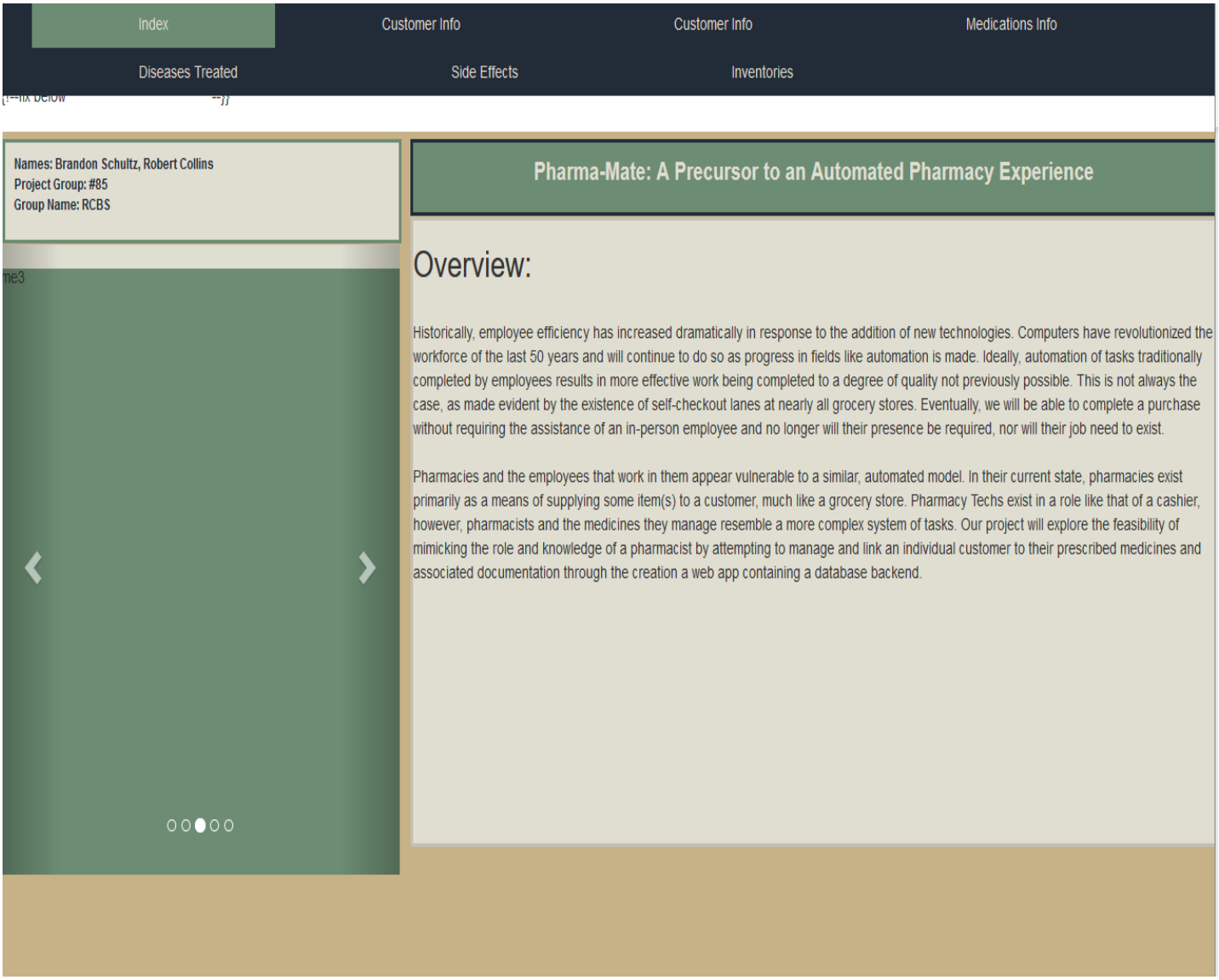
Schema:



Brandon Schultz

Screen captures:

Index:



Customers: Includes CREATE, UPDATE, READ, DELETE,

Index | Customer Info | Customer Record Info | Medications Info | Diseases Treated | Side Effects | Inventories

Customer's Details:

-This page holds Information concerning prescribing information for medicines in inventory.
-Acts as composite to form M:M between CustomerRecords and DiseasesTreated, SideEffects and inventories entities.
-Will be used as branching point between medicine prescribed to patient and medcine held by inventory.

Add Customers to DB Table:

Add to Entity Table:

{{!-- in router file else if(req.body.formType == "update"){ --}} {{!-- change this { --}} {{!-- Used to update idnum column of various tables in project.---}} {{!-- fixxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx--------}} {{!-- fixxxxxxxxxxxxxxxxxxxxxxxxx {{!-- additional input might be needed--------}} {{!-- fixxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx--------}}

Customers Entity Properties:	Customers Property Values:
Customer ID #	
Customers First Name:	
Customers Last Name:	

Add Customers

Remove From Entity Table:

{{#each Customers}} {{/each}}

Select	Table ID #	Customers ID #	Customers First Name:	Customers Last Name:
☐	{{customerID}}	{{customerID}}	{{fname}}	{{lname}}

Clear Selection | Remove Selection | Check to Enable Remove: ☐

View/Search Customers Entity Table:

View Entity Table:

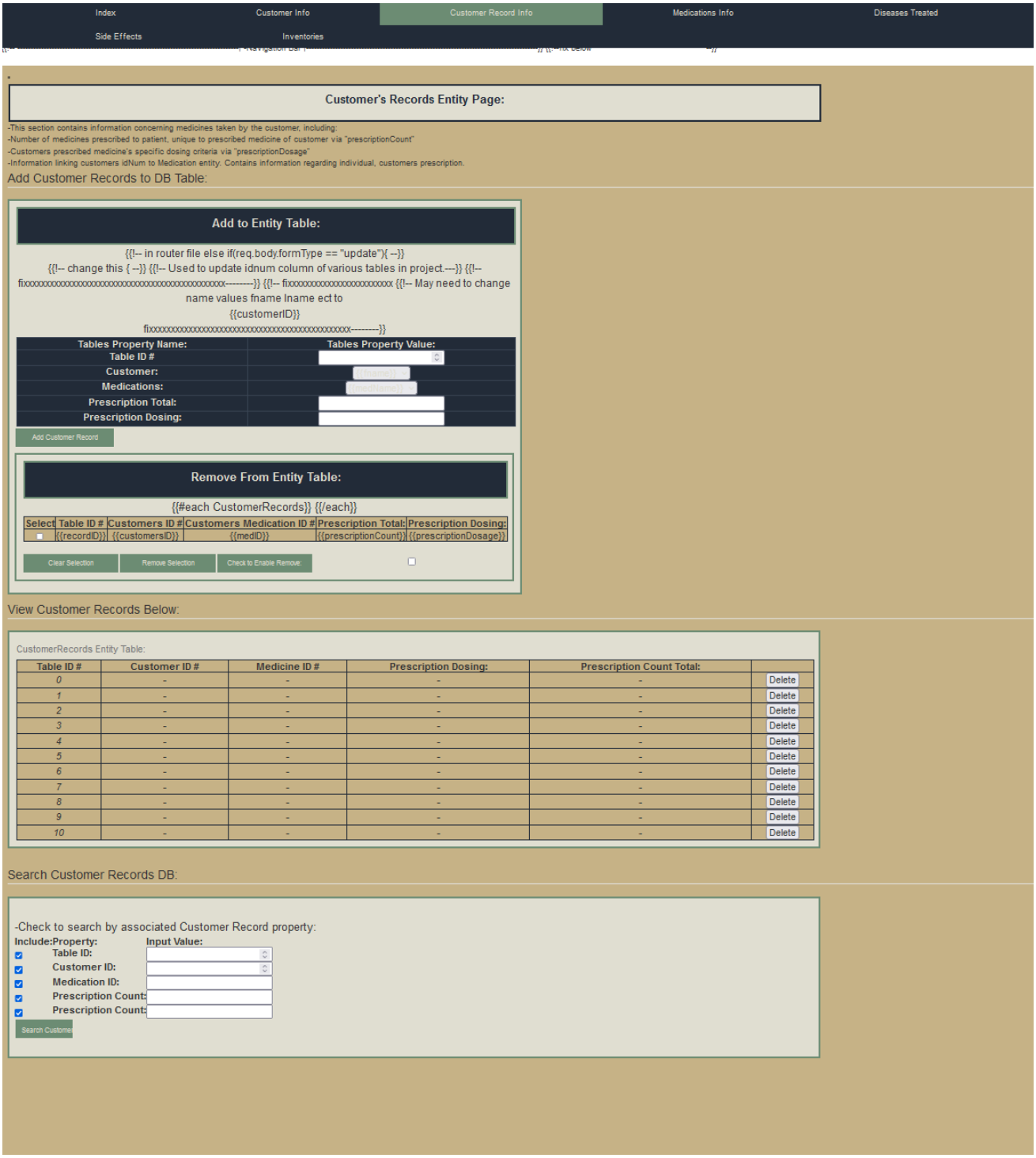
Customers Table ID #	Customers First Name:	Customers Last Name:	Remove Row
0	-	-	Delete
1	-	-	Delete
2	-	-	Delete
3	-	-	Delete
4	-	-	Delete
5	-	-	Delete
6	-	-	Delete
7	-	-	Delete
8	-	-	Delete
9	-	-	Delete
10	-	-	Delete

Search Entity Table:

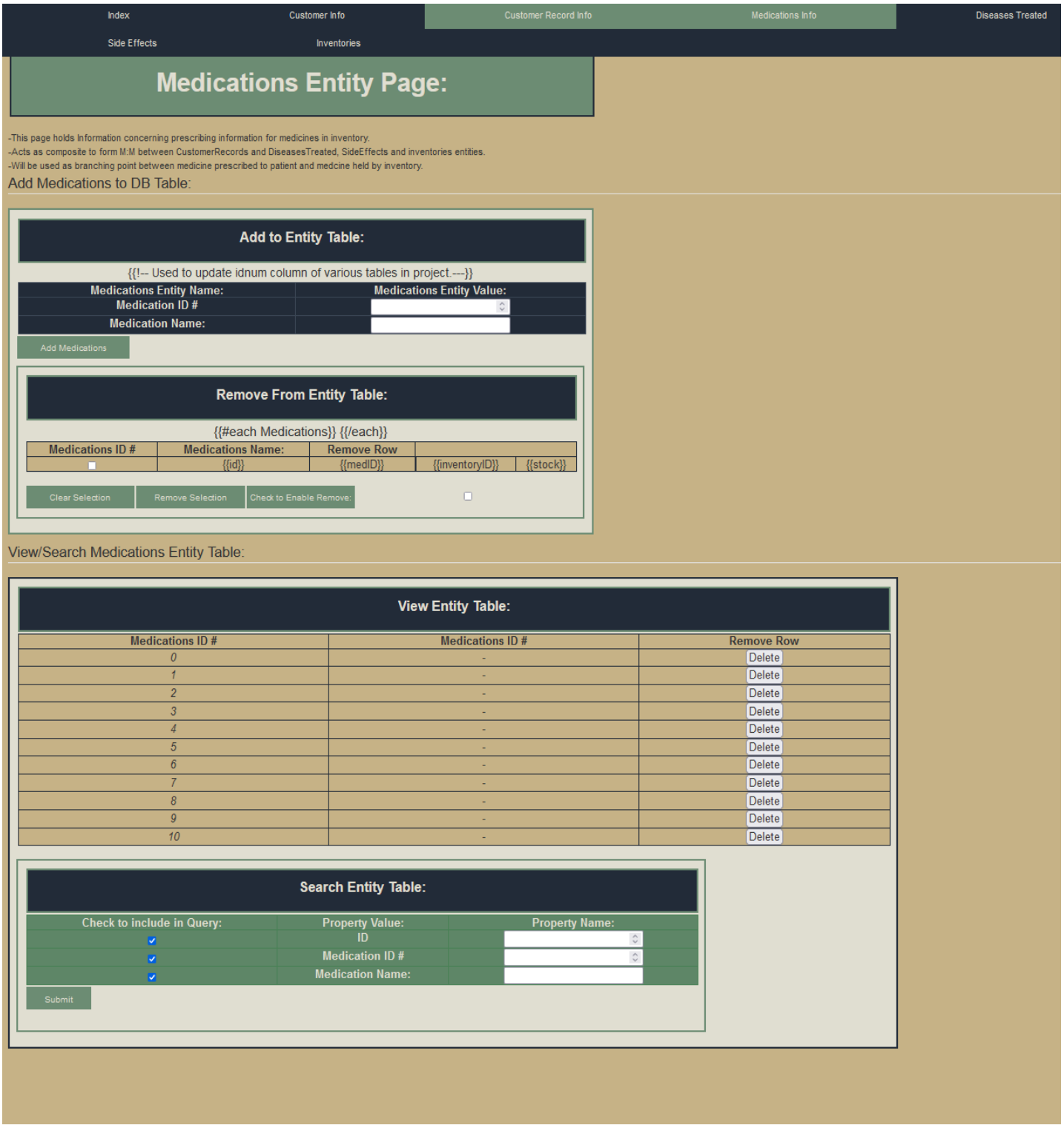
Check to include in Query:	Property Value:	Property Name:
☑	ID	
☑	Customer ID #	
☑	First Name:	
☑	Last Name:	

Submit

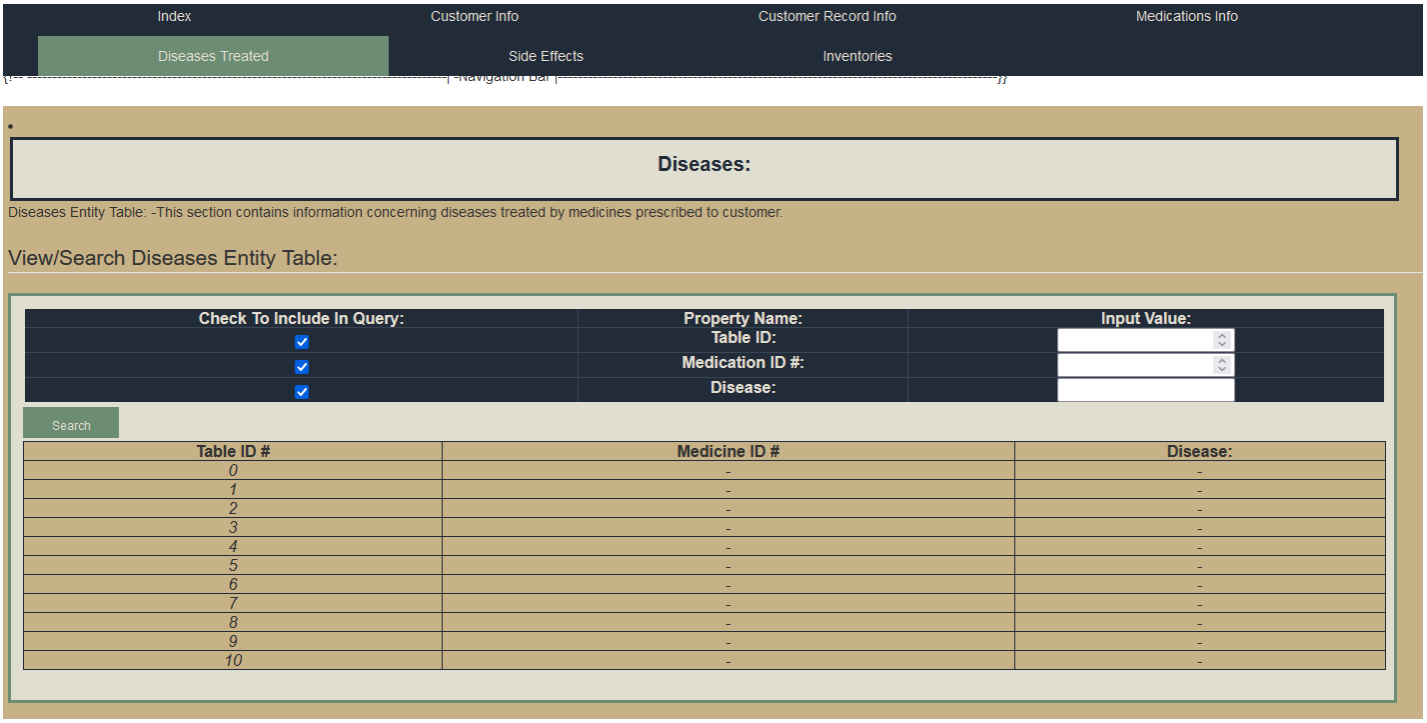
CustomerRecords: Includes CREATE, UPDATE, READ, DELETE,



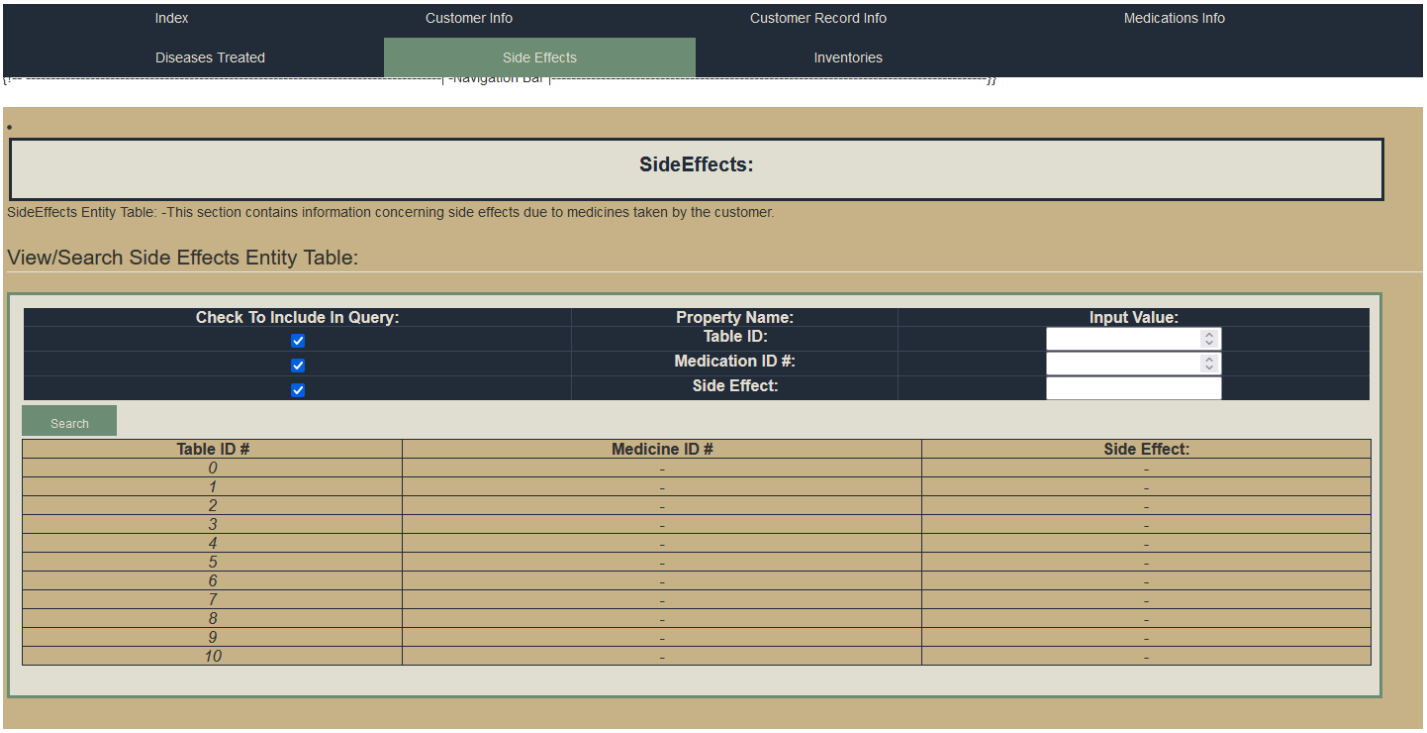
Medications: Includes CREATE, UPDATE, READ, DELETE,



DiseasesTreated: Read



SideEffects: Read



Inventories: Read