MAX FIT EVENT MANAGEMENT WITH SALESFORCE

AKSHAY DAGWAR

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd

Part of the Data Storage Systems Commons, and the Other Computer Engineering Commons

Recommended Citation
DAGWAR, AKSHAY, "MAX FIT EVENT MANAGEMENT WITH SALESFORCE" (2023). Electronic Theses, Projects, and Dissertations. 1601.
https://scholarworks.lib.csusb.edu/etd/1601

This Project is brought to you for free and open access by the Office of Graduate Studies at CSUSB ScholarWorks. It has been accepted for inclusion in Electronic Theses, Projects, and Dissertations by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
MAX FIT EVENT MANAGEMENT WITH SALESFORCE

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Akshay Pandurang Dagwar
January 2023
MAX FIT EVENT MANAGEMENT WITH SALESFORCE

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Akshay Pandurang Dagwar
January 2023

Approved by:

Dr. Yan Zhang, Advisor, Computer Science and Engineering
Dr. Jennifer Jin, Committee Member
Dr. Amir Ghasemkhani, Committee Member
ABSTRACT

MAX FIT Gym is looking for an event management software program to help manage activities very efficiently, along with attendees and environmental statistics. The event management program is developed and deployed using the Salesforce platform. MAX FIT can efficiently create, edit, and remove events and send email alerts to clients. This task operated on opportunities captured under MAX FIT, including all clients, and prepared information in the Salesforce cloud. This also includes product inventory with various varieties of protein products, and business owners can also add more products to their inventory. In the event management program, the event addresses within the USA are verified through third-party service providers. MAX FIT Event Management application provides a platform to manage clients information and selling MAX FIT’s protein products to the clients, and all the data related to the application can be stored on the cloud. The protein product inventory is specially built for the MAX FIT events, and clients can place an order online after the event. A developer has developed the whole process in the application, from creating the event to the placement of product orders.
ACKNOWLEDGEMENTS

With immense pleasure, I would like to present this project report on “MAX Fit Event Management with Salesforce.” It has been an enriching experience for us to undergo such a project of Deep Learning and Machine Learning, which would not have been possible without the goodwill and support of the people around us.

As a California State University – San Bernardino student, I express deep gratitude towards Dr. Yan Zhang, my project advisor, and committee members Dr. Jennifer Jin and Dr. Amir Ghasemkhani for their support during the whole study session and development. Because of them, we were prompted to do hard work and adopt new technologies.

We are also thankful to our university for providing such self-development subjects in our curriculum, which help us explore different technologies.
TABLE OF CONTENTS

ABSTRACT.................................................................................................................................................. iii

ACKNOWLEDGEMENTS......................................................................................................................... iv

LIST OF FIGURES ........................................................................................................................................ vii

CHAPTER ONE: INTRODUCTION ........................................................................................................ 1
  Background ............................................................................................................................................. 1
  Purpose .................................................................................................................................................. 2

CHAPTER TWO: TECHNOLOGIES ........................................................................................................ 4
  Hardware Requirement ....................................................................................................................... 5
  Software Development Requirement ................................................................................................ 5
    Apex (Programming Language) ........................................................................................................ 5
    JavaScript (Programming Language) ................................................................................................ 7
    Custom Objects, Custom Fields (Salesforce) .................................................................................... 9
    Lightning Aura Components (UI) ..................................................................................................... 11

CHAPTER THREE: USER INTERFACE .................................................................................................. 14
  Lightning Design Systems .................................................................................................................. 14
  Badges with Icons ............................................................................................................................ 14
  What are Styling Hooks ...................................................................................................................... 15
Application UI ............................................................................................................. 16

CHAPTER FOUR: SYSTEM DESIGN .............................................................................. 22

Design Pattern ............................................................................................................. 22

CHAPTER FIVE: EXPLORING DATABASE ..................................................................... 28

SOQL Database Creation ............................................................................................ 28

REST API Integration ................................................................................................ 29

CHAPTER SIX: TESTING .............................................................................................. 31

Salesforce Testing ........................................................................................................ 31

CHAPTER SEVEN: FUTURE ADVANCEMENTS ............................................................. 32

APPENDIX A: CODE OF CRUCIAL PART .................................................................... 33

REFERENCES ................................................................................................................. 42
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New Salesforce Environment</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Creating Custom Fields Data Type</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Declare Custom Object, Custom Fields</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Lightning Aura Component Code (UI)</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Creating Any Card With Any Icons</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>CSS Button Example In Lightning Aura Component</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Attendee List View</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Event List View</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>Event Organizers List View</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>Protein Explorer Products</td>
<td>19</td>
</tr>
<tr>
<td>11</td>
<td>Protein Explorer Shopping Cart</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>Protein Explorer Order Detail</td>
<td>21</td>
</tr>
<tr>
<td>13</td>
<td>After Creating The Objects/Fields App View</td>
<td>23</td>
</tr>
<tr>
<td>14</td>
<td>Salesforce Entity Relationship Diagram</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>Object Permissions</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>Snapshot of Event Details Component</td>
<td>27</td>
</tr>
<tr>
<td>17</td>
<td>Example of SOQL Query on Contact Object</td>
<td>28</td>
</tr>
<tr>
<td>18</td>
<td>Example of Creating SOQL Queries</td>
<td>29</td>
</tr>
<tr>
<td>19</td>
<td>Snapshot of REST API Callout Class</td>
<td>30</td>
</tr>
</tbody>
</table>
CHAPTER ONE:
INTRODUCTION

Background

Salesforce is a customer success platform designed to help businesses sell, serve, market, analyze, and connect with their customers. Salesforce provides everything they need to run a business from anywhere. Salesforce manages connections with prospects and customers, engages and interacts with employees and partners, and securely stores business data in the cloud with off-the-shelf products and features. It's hard to get a complete picture of business prospects when data is stored in spreadsheets, hidden in emails and text messages, or pinned to bulletin boards. And a business can access data anytime, anywhere with the Salesforce platform.

MAX FIT Gym is searching for event management software that can assist them in properly managing their events and attendance, and location information. For example, MAX FIT is organizing a Yoga event for the clients, clients are registered for the in-person event, and the business owner is getting clients data. After the event, the organizer can suggest clients purchase their protein products. As a developer, I have developed and configured the custom objects and deployed the code using the Salesforce Platform. This project handles the MAX FIT Gym events, which comprises all of the clients data in the
Salesforce cloud with structured data. The features of product inventories are that clients can place an order online and have coupon code functionality when purchasing the product.

Purpose

The MAX FIT project’s first objective is to create, identify events, and determine appropriate control measures. Multiple updates or changes may be made to a service or configuration item. In the MAX FIT event, business owners can create events for their clients. Some of these configurations can be critical, while some configurations may be minor without impacting other aspects of the IT service. After creating the events, if the owner of the event wants to cancel the event, then the owner can remove or delete it directly from the event object, and all the corresponding configurations will get deleted automatically. Categorizing these events and defining appropriate control actions for these various events is the goal of the event management process. Attendees that have registered for an event will receive an email alert regarding this event.

The second objective of the MAX FIT Event Management process means that events can be programmed in ways such as client information is transferred. For instance, the client can register for one or more events that are undergoing in MAX FIT. These actions indicate changes to configuration items or services that are classified as events. The event management process is intended to program this type of flow and management
configurations and information in an IT Service Provider. There is a product inventory of MAX FIT Gym, which can be suggested by the owner of the event to the clients. And after the event, clients can also buy the gym protein products provided by MAX FIT Gym.
The MAX FIT project is deployed in the Salesforce environment. Salesforce is a cloud technology that produces cloud-based software designed to help businesses find more leads, close more deals, and delight their customers with excellent service. It helps teams work better together. The business may use a single Customer 360 app or a combination of many. By improving team communications and productivity, enterprises drive tremendous success.

Figure 1 shows the new Salesforce environment for the MAX Fit project and what the new salesforce environment looks like. It is the developer org only for the MAX Fit project in which I have implemented all the required functionality for the project.

The Salesforce Developer Edition Org is a special Salesforce edition that provides a full-featured Salesforce environment for developing and testing existing or new features and testing your own custom applications. It's free and can be stored forever. In the developer org, salesforce developers can build the elements related to their projects and requirements. In general, developer orgs allow running tests that developers don't want to run in a production or sandbox environment.
Hardware Requirement

- Memory: 4 GB (minimum)
- Graphics Card: NVIDIA GeForce GTX 970/ Apple M1 Chip
- CPU: Intel Core i5 or above, Apple M series
- OS: Windows, Mac OS

Software Development Requirement

**Apex (Programming Language)**

Apex is a strongly typed, object-oriented programming language that
enables builders to execute float and transaction manipulation statements along with Lightning Platform API calls on Lightning Platform servers. Apex allows developers to access the Salesforce platform's back case database and client-server interface to build third-party Software as a Service (SaaS) applications. Apex consists of APIs (Application Programming Interfaces) that Salesforce authors can use to access personal records on the platform.

Software as a Service (SaaS) is a means of delivering Software as a Service over the Internet. Instead of installing and updating software, simply access it over the Internet, freeing from complicated Software and device maintenance. SaaS programs are also known as web-based software, on-demand software, and hosted software. SaaS programs, whatever their names, are hosted on the SaaS provider's servers. Application access is managed by providers, including security, availability, and performance.

I have worked on following Apex classes to work on the implementation of business logic, starting from creating, editing, and deleting the events till placing the protein products order into the database from the MAX FIT application,

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Apex Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>AttendeeEventsService.cls</td>
</tr>
<tr>
<td>2.</td>
<td>CustomSearchController.cls</td>
</tr>
<tr>
<td>3.</td>
<td>DeleteEventBatch.cls</td>
</tr>
</tbody>
</table>
JavaScript (Programming Language)

JavaScript is an object-oriented programming language used to create interactive websites (clickable buttons, popup menus, etc.).

JavaScript (regularly shortened to JS) is a lightweight, interpreted, object-orientated language with fine functions and is fine called the scripting language for Web pages; however, it is used in lots of non-browser environments as well. Developers can add JavaScript to programmatically control objects within their context. JavaScript has a standard object library that includes classes such as Array, Date, and Math and a basic set of
language components that include operators, control structures, and statements. When a developer is creating any Lightning Aura Component at that time, there are two files, controller and helper in these two files, the developer can write their logic using JavaScript. The developer worked on the following JavaScript classes,

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>JavaScript Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>ProteinExplorerController.js</td>
</tr>
<tr>
<td>2.</td>
<td>ProteinListController.js</td>
</tr>
<tr>
<td>3.</td>
<td>CartInfoController.js</td>
</tr>
<tr>
<td>4.</td>
<td>CartItemController.js</td>
</tr>
<tr>
<td>5.</td>
<td>CartDetailController.js</td>
</tr>
<tr>
<td>6.</td>
<td>OrderDetailController.js</td>
</tr>
<tr>
<td>7.</td>
<td>CreateProteinOrderController.js</td>
</tr>
<tr>
<td>8.</td>
<td>EventDetailsService.js</td>
</tr>
<tr>
<td>9.</td>
<td>EventManager.js</td>
</tr>
<tr>
<td>10.</td>
<td>HeaderComponentController.js</td>
</tr>
<tr>
<td>11.</td>
<td>EventManagerController.js</td>
</tr>
</tbody>
</table>
Custom Objects, Custom Fields (Salesforce)

To develop any project in Salesforce, it is essential to create custom objects and custom fields because the developer has to store the data in the created custom objects. If there is any automation process in the business scenario developer has to code and write a trigger on that object. For example, in MAX FIT, there are some automation processes like sending email alerts to the registered clients for this automation developer has written a code on the particular custom object known as “Event_Attendees__c.”

Standard objects are objects that come with Salesforce. Common business objects such as Account, Contact, Lead, and Opportunity are all standard objects. Custom objects are objects that the developer creates to store information specific to the business or industry. For example, in MAX FIT developer has created an “Event” object to store all the information related to events. In Chapter Four, I have given all the custom objects that are used in the MAX FIT event. Custom objects are containers of information, but they also provide special functionality. For example, when a developer creates a custom object, the platform automatically creates things like the page layout for the MAX FIT user interface.

Custom fields are used to assign titles, excerpts, or non-content values to articles or pages. Figure 3 is the process to create custom fields data type, click Edit to modify the custom field, add custom help text, or change the data type. Custom Fields is a metadata category that allows administrators to describe fully the types of assets they store in their digital
library. Figure 2 shows how to declare the custom objects “Location__c” and their custom fields “Verified__c”, “Street__c”, “City__c”, “State__c,” and “Postal_Code__c” while writing the Apex code.

Figure 2: Creating Custom Fields Data Type
Aura Additives are devices for self-contained, reusable applications. They represent reusable sections of the user interface and can vary in granularity from a single line of text to an entire app. The framework consists of several commercially available additives. For example, an additive containing Lightning Design System patterns can be used within the Lightning namespace. These additives are also called base flush additives. Additives can be merged and configured to create new additives in the app. Components are rendered to present HTML DOM elements
within the browser. Figure 4 shows an example of a Lightning Aura Component. Components can contain HTML, CSS, JavaScript, or other web-friendly code, in addition to various additives. This allows developers to create apps with state-of-the-art user interfaces.

Lightning Aura Component is one of the latest User Interface (UI) frameworks provided by Salesforce in which developers can design the application or any software, and the database will be stored in the cloud. I have developed the MAX FIT application by using the Lightning Aura Component framework and worked on following Lightning Aura Components to create this MAX FIT application,
<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Lightning Aura Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>AttendeeEvents.cmp</td>
</tr>
<tr>
<td>2.</td>
<td>ModalPopup.cmp</td>
</tr>
<tr>
<td>3.</td>
<td>ProductList.cmp</td>
</tr>
<tr>
<td>4.</td>
<td>HeaderComponent.cmp</td>
</tr>
<tr>
<td>5.</td>
<td>ProteinSearch.cmp</td>
</tr>
<tr>
<td>6.</td>
<td>SelfRegister.cmp</td>
</tr>
<tr>
<td>7.</td>
<td>CartDetailController.cmp</td>
</tr>
<tr>
<td>8.</td>
<td>ProteinExplorer.cmp</td>
</tr>
<tr>
<td>9.</td>
<td>OrderDetailController.cmp</td>
</tr>
<tr>
<td>10.</td>
<td>CartItem.cmp</td>
</tr>
<tr>
<td>11.</td>
<td>CartInfo.cmp</td>
</tr>
</tbody>
</table>
CHAPTER THREE:

USER INTERFACE

Lightning Design Systems

The Lightning Design System displays the underlying styles and additives of Salesforce merchandise. These styles and additives offer appearance and sense whilst designing packages and merchandise in the Salesforce ecosystem. The Salesforce Lightning Design System is ready to apply to any Lightning apps and components. This now regularly guards against Lightning components running within Lightning Experience and the Salesforce S1 Cell App. It's not strictly necessary to provide static, valuable resources for Lightning components working in these environments.

Badges with Icons

Developers can add an icon to the badge. For visual clarity, utility icons are the preferred icon type in badges. Figure 5 shows an example of how to add the badge icon to any button or card. Developer can also add icons to badges without text, but be sure to include help text in this case.
Figure 5: Creating any card with any Icons

What are Styling Hooks

Styling hooks use CSS custom houses which make it smooth to customize difficult styling and specific a logo, mainly even when working with internet components and shadow DOM.

Figure 6 shows an example of the logo button and the use of styling hooks. These buttons are already defined by Lightning Design Systems.

Figure 6: CSS button example in Lightning Aura Component
Application UI

Few snapshots of the User Interface have been attached here:

Figure 7 references the Attendee object, a custom object that contains a list view of all attendees attending a MAX FIT event. In this custom object, there are some fields that give the information of an attendee who is attending the event.

In this attendee list view, attendees can be added by clicking the New button and then giving all the valid details of the attendee for the particular event. The owner of the event can also be changed from this view.

![Attendee List View](image)

Figure 7: Attendee List View
Figure 8 is related to the Event object which is a custom object that has a list view of all the events organized by MAX FIT Gym. There can be several events like yoga events, Zumba events, etc. All the data related to the event can be seen in this event list view, including whether the event is live or not, the organizer of the event, and other information related to events.

Once the event goes live, clients can register for the event. Otherwise, the clients cannot register for the event.

Figure 8: Event List View
Figure 9 is related to the Event Organizer object, which is a custom object that has details of the organizer. The organizer can directly register any client for his particular event, but the organizer cannot register a client for any other events. The Layout and the design of the Event Organizers are developed with the Lightning Aura Component framework.

![Event Organizers List View](image)

Figure 9: Event Organizers List View
Figure 10 is related to Protein Explorer, which has all the product inventories of MAX FIT Gym from where clients can place an order for the products directly. In this view, clients can see the details of the protein products and can add them to the cart. Clients can purchase multiple products at a time and can search for a particular product in the search bar. By clicking the View Details button, the details of the product can be seen, and the clients can also place an order from the detail pages.

Figure 10: Protein Explorer Products
Figure 11 is related to the Protein Explorer Shopping Cart, which has all the products that a client is purchasing. Here clients can find the names of products, the price of the product, the quantity of the products that they are trying to purchase, and also the delete option in case clients do not want to purchase any product from the selected list. The product inventories of MAX FIT Gym also show the total price of the products and the functionality of coupons.

![Protein Explorer Shopping Cart](image)

Figure 11: Protein Explorer Shopping Cart
Figure 12 is related to the Protein Explorer Order Details which has all the information related to the order which has been placed by clients. It shows the order number, most importantly shipping address and the amount. It also shows the information related to the discount amount after using the coupon code. This is a very convenient way to purchase products by using the Salesforce platform.

Figure 12: Protein Explorer Order Detail
CHAPTER FOUR:
SYSTEM DESIGN

Design Pattern

MAX FIT is looking for an event management software that can help them to manage their events very effectively along with the attends & location information. In this project, we have created custom objects (Events, Event organizers, Attendees, Event Attendees, Speakers, Locations) and custom fields related to the project requirement using the Salesforce platform. Figure 13 shows the view of developer org after creating custom objects and custom fields in Salesforce. Here is the complete information on the project.

Standard Salesforce objects represent objects that the Salesforce platform has already created for customers. Salesforce objects require minimal configuration and provide functionality appropriate for a wide range of businesses. These objects provide users with complete information about the stored details of specific records and their relevance in a project.
Figure 13: After Creating the Objects/Fields App View

Custom objects involved in MAX FIT application -

- Location (Address Book) Object. Below are the fields:

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>Text</td>
</tr>
<tr>
<td>Country</td>
<td>PickList</td>
</tr>
<tr>
<td>Verified</td>
<td>Checkbox</td>
</tr>
<tr>
<td>Street</td>
<td>Text</td>
</tr>
<tr>
<td>Postal Code</td>
<td>Text</td>
</tr>
<tr>
<td>State</td>
<td>Text</td>
</tr>
<tr>
<td>Landmark</td>
<td>Text</td>
</tr>
</tbody>
</table>

- Event Organizer (Object)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Standard field</td>
</tr>
<tr>
<td>Alternative Phone</td>
<td>Phone</td>
</tr>
<tr>
<td>Email</td>
<td>Email</td>
</tr>
<tr>
<td>Phone</td>
<td>Phone</td>
</tr>
</tbody>
</table>
### Address

<table>
<thead>
<tr>
<th>Address</th>
<th>Lookup - Location</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Alternative Email</th>
<th>Email</th>
</tr>
</thead>
</table>

### Event (Object)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>Standard Auto Number</td>
</tr>
<tr>
<td>Name</td>
<td>Text</td>
</tr>
<tr>
<td>Status</td>
<td>Picklist</td>
</tr>
<tr>
<td>Organizer</td>
<td>Lookup - Event Organizer</td>
</tr>
<tr>
<td>Max Seats</td>
<td>Number</td>
</tr>
<tr>
<td># People Attending</td>
<td>Rollup Summary Field</td>
</tr>
</tbody>
</table>

### Attendees (Object)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Standard</td>
</tr>
<tr>
<td>Phone</td>
<td>Phone</td>
</tr>
<tr>
<td>Email</td>
<td>Email</td>
</tr>
<tr>
<td>Address</td>
<td>Lookup - Location</td>
</tr>
<tr>
<td>Company Name</td>
<td>Text</td>
</tr>
</tbody>
</table>
• Speaker (Object)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Standard Name Field</td>
</tr>
<tr>
<td>Email</td>
<td>Email</td>
</tr>
<tr>
<td>Phone</td>
<td>Phone</td>
</tr>
<tr>
<td>Event</td>
<td>M-D Event</td>
</tr>
<tr>
<td>Attendee</td>
<td>M-D Attendee</td>
</tr>
</tbody>
</table>

Figure 14 shows the entity relationship diagram between all the objects that are created for the MAX FIT Event. It shows all the relationships between all the required objects for this MAX FIT project.

![Salesforce Entity Relationship Diagram](image-url)
Figure 15 shows what permission we can give to the object which is created for the MAX FIT Event. Figure 14 shows CRED - it denotes Create, Read, Edit, and Delete permissions that we are giving to any custom object while creating application functionality or configuration.

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Profile</th>
<th>Event Manager</th>
<th>Speaker</th>
<th>Attendee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>CRED</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Event - Organizer</td>
<td>CRE</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Speaker</td>
<td>CRE</td>
<td>CRED</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Attendee</td>
<td>R</td>
<td>X</td>
<td>CRE</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>CRED</td>
<td>R</td>
<td>RCE</td>
<td></td>
</tr>
<tr>
<td>Event - Speaker</td>
<td>CRED</td>
<td>RCE</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Event - Attendees</td>
<td>CRED</td>
<td>X</td>
<td>RC</td>
<td></td>
</tr>
</tbody>
</table>

Figure 15: Object Permissions
In the Event Detail Component Figure 16 shows that recordId of custom object “Event__c” is included. Likewise, the developer has done the same for all the other custom objects in the MAX FIT application. These custom objects also have default values. In MAX FIT developers can use Lightning Navigation to open and create a record wizard for the same.

```javascript
import { NavigationMixin } from 'lightning/navigation';
import { encodeDefaultFieldValues } from 'lightning/pageReferenceUtils', const
defaultValueS = encodeDefaultFieldValues({
   Event__c: this.recordId
 });
this[NavigationMixin.Navigate]({
   type: 'standard__objectPage',
   attributes: {
      objectApiName: 'Eventspeakers__c', actionName: 'new'
   },
   state: {
      defaultFieldValues: defaultValueS
   }
 });
```

Figure 16: Snapshot of Event Details Component
CHAPTER FIVE:
EXPLORING DATABASE

SOQL Database Creation

The specific information in MAX FIT organization's Salesforce data use the Salesforce Object Query Language (SOQL). SOQL is similar to the widely used Structured Query Language (SQL) SELECT statement but is designed specifically for Salesforce data. SOQL lets developers create simple yet powerful query strings in multiple environments. Developers should avoid SOQL Queries or DML statements inside FOR Loops to avoid Salesforce governor limits.

Figure 17 shows writing SOQL Queries in Salesforce Extensions for Visual Studio Code for Contact object. Figure 18 shows developers use SOQL and Salesforce Object Query Language (SOQL) Queries in the Apex to retrieve the data from the database by using the “Select” or “Find” keyword. In SOQL, developers can use the “Select” keyword and in SOSL developers can use the “Find” keyword. Developers can write SOQL queries in the Apex programming language, but SOSL queries cannot be written in Apex.

```
1 Contact c = new Contact(Account = [SELECT Name FROM Account
2 WHERE NumberOfEmployees > 10 LIMIT 1]);
3 c.FirstName = 'James';
4 c.LastName = 'Yorke';
```

Figure 17: Example of SOQL Query on Contact Object
The REST API is one of several web interfaces developers can use to access Salesforce data outside of the Salesforce UI. The use of the REST API tool is to create, edit and view data in Salesforce by making HTTP requests to Salesforce endpoints. A major advantage of REST APIs is that developers don't need many tools to access the data. It's easier to use than the SOAP API but still offers more functionality. Figure 19 is an example of REST API classes that show the declaration and implementation of REST API.

Figure 18: Example of Creating SOQL Queries
In MAX FIT event management program, the event addresses within USA are verified through third party service provider. Developers use REST API to perform this functionality in the Salesforce platform.

Figure 19: Snapshot of REST API Callout Class
CHAPTER SIX:
TESTING

Salesforce Testing

In the testing of this application, the developer has created some test data which work correctly to send the email alerts, product list, order placement functionality, and coupon features. The UI testing works well as per the required functions and compatibility. Risk management and timely delivery of high-quality Salesforce releases require testing and test automation. A sophisticated strategy is needed to get it right, and context is essential in deciding which option is best. Accept that test automation (and quality) is a continuous process because Salesforce is a very dynamic platform where change is driven by both Salesforce's platform expansion and modifications to meet business objectives.

The developer has sent some test emails to see if this REST API integration works. All the features regarding adding events and adding products to the cart are tested. The success message also works correctly as per the requirement.
Here are a few of the future advancements that can be made in the application:

- Using Lightning Web Component, and developers can develop this application in a more efficient way.
- Incorporating a consumer complaint section where people can get help from the authority quickly.
- Adding a social network to the app where people can connect with each other.
- Adding a shipment feature to the application like UPS or FedEx by buying their subscription.
- Providing a platform for the event managers in the city to publish and advertise their events and also sell their products.
- Developers or business owners can also use the community cloud to create this application.
APPENDIX A:

CODE OF CRUCIAL PART
Apex Class: CartDetailsAuraServices

public class CartDetailsAuraServices {

    private static List<Coupon__c> get_coupon(String Name){
        List<Coupon__c> coupon = [Select Id, Price__c From Coupon__c Where Name =: Name];
        return coupon;
    }

    @AuraEnabled
    public static Decimal checkCoupon(String Name, String CartId){
        List<Coupon__c> couponList =  get_coupon(Name);
        if(couponList !=null & couponList.size() > 0){
            Cart__c cart = new Cart__c(Id = cartId, Coupon__c =couponList[0].Id );
            update cart;
            return couponList[0].Price__c;
        }else{
            return null;
        }
    }

    @AuraEnabled
    public static String createCartItems(List<String> proteinList, String cartId){
        System.debug('#### beerList '+ proteinList);
        List<Cart_Item__c> cartItemList = new List<Cart_Item__c>();
        List<Cart_Item__c> cartItemToUpdate = new List<Cart_Item__c>();
        Map<Id, Cart_Item__c> beerQntyMap = getCartItems(cartId);
        For(String protein : proteinList){
            if(beerQntyMap != null & beerQntyMap.containsKey(protein)){
                Cart_Item__c it = beerQntyMap.get(protein);
                Cart_Item__c item = new Cart_Item__c(
                    Item_Quantity__c = it.Item_Quantity__c+1,
                    Id = it.Id
                );
                cartItemToUpdate.add(item);
            }else{
                Integer Qty = 1;
                Cart_Item__c item = new Cart_Item__c(
                    Cart__c = cartId,
                    Protein__c = protein,
                    Item_Quantity__c = Qty
                );
                cartItemList.add(item);
            }
        }
        insert cartItemList;
        if(cartItemToUpdate != null & cartItemToUpdate.size() > 0)
            update cartItemToUpdate;
        return cartId;
    }
}
@AuraEnabled
public static String getCartId(List<String> proteinList){
List<Cart__c> cartList = getCart();
if(cartList != null && cartList.size() > 0){
createCartItems(proteinList, cartList[0].Id);
    system.debug('cartlist1:'+cartList[0].Id);
    return cartList[0].Id;
}
else{
    Cart__c cart = new Cart__c(Coupon__c =
    getCoupon('Default')[0].Id, Cart_Id__c=String.valueOf(Math.random()),
    Cart_Status__c='Open', Is_Active__c=true, User__c =
    UserInfo.getUserId());
    insert cart;
    createCartItems(proteinList, cart.Id);
    return  cart.Id;
}
}

private static List<Cart__c> getCart(){
List<Cart__c> cartList = Select Id, Name From Cart__c Where User__c=:UserInfo.getUserId() AND Is_Active__c = true;
    system.debug('cartlist:'+cartList);
    return cartList;
}

@AuraEnabled
public static Map<Id, Cart_Item__c> getCartItems(String CartId){
List<Cart_Item__c> existingItemList = Select Id, Name, Item_Quantity__c,Total_Amount__c, Protein__c, Protein__r.Name From Cart_Item__c Where Cart__c =: cartId;
    Map<Id, Cart_Item__c> proteinQntyMap = new Map<Id, Cart_Item__c>();
    For(Cart_Item__c item : existingItemList){
        if(!proteinQntyMap.containsKey(item.Protein__c)){
            proteinQntyMap.put(item.Protein__c, item);
        }
    }
    return proteinQntyMap;
}

@AuraEnabled
public static void deleteItem(String CartItemId){
    Database.delete(CartItemId);
}

@AuraEnabled(cacheable=true)
public static String fetchAddressBook()
{
    List<Address_Book__c> add = [SELECT Id, Name, City__c, State__c,
                              Postal_Code__c, Country__c,
                              Street__c, User__c FROM Address_Book__c
                              WHERE User__c=:userinfo.getUserId() ];
    return JSON.serialize(add);
}

@AuraEnabled
public static String createOrder(String AddressId, String CartId, String
UserId,Decimal SubTotal) //
{
    system.debug('Starting');
    Order__c ord = new Order__c();
    ord.Shipping_Address__c = AddressId;
    ord.Cart__c = CartId;
    ord.User__c = UserId;
    ord.SubTotal__c = SubTotal;
    Cart__c car = new Cart__c();
    car.Cart_Status__c = 'Closed';
    car.Is_Active__c = false;
    car.Id = CartId;
    insert ord;
    update car;
    system.debug('Ending');
    return ord.Id;
}


ProteinExplorer.cmp

<aura:component
  implements="force:appHostable, flexipage:availableForAllPageTypes"
  controller="ProteinSearchAuraService">

    <aura:handler name="ProteinEvent" event="c:SearchEvent"
    action="{!c.handleCompEvent}" />
    <aura:handler name="init" value="{!this}" action="{!c.handleCompEvent}" />
    <aura:handler name="addToCart" event="c:AddToCart"
    action="{!c.updateCart}" />
    <aura:attribute name="beerList" type="Object" />
    <div>
      <c:HeaderComponent aura:id="headerComp" TagLine="ProteinExplorer"
        message="Search and Order Protein in a very easy manner!" />
    </div>
    <c:ProteinSearch />
    </div>
  </aura:component>
ProteinListComponent.cmp

<aura:component>
    <aura:attribute name="recordList" type="Object"/>
    <aura:attribute name="beerId" type="String"/>
    <lightning:overlayLibrary aura:id="overlayLib"/>
    <aura:attribute name="beervalue" type="string"/>
    <aura:handler name="init" value="{!this}" action="{!c.doInit}" />
    <aura:registerEvent name="addToCart" type="c:AddToCart" />
    <div class="slds-grid slds-wrap">
        <aura:iteration items="{!v.recordList}" var="item" indexVar="index">
            <div class="slds-col slds-size_1-of-4 slds-p-around_small">
                <lightning:card title="{!item.Name}"
                    footer="{!item.Brewery_Name__c}">
                    iconName="custom:custom7">
                        <p class="slds-p-horizontal_small">
                            <div class="slds-gutters">
                                <div class="slds-col slds-size_1-of-3">
                                    <img src="{!item.Image__c}" alt="{!item.Name}" />
                                </div>
                                <div class="slds-col slds-size_2-of-3">
                                    Name       :  {!item.Name} <br/>
                                    Id         :  {!item.Id__c} <br/>
                                    Price      :  {!item.Price__c}
                                </div>
                            </div>
                        </p>
                        <aura:set attribute="actions">
                            <lightning:button name="{!item.Id}" label="View Details"
                                value="{!item.Name}"
                                variant="brand" onclick="{!c.showInfo}" />
                        </aura:set>
                        <aura:set attribute="footer">
                            <lightning:button name="{!item.Id}" label="Add To Cart"
                                value="{!index}"
                                variant="brand" onclick="{!c.addToCart}" />
                        </aura:set>
                    </lightning:card>
                </div>
            </div>
        </aura:iteration>
    </div>
</aura:component>

<!-- {!v.recordList} -->
{!v.recordid}
<aura:iteration items="{!v.recordList}" var="item" indexVar="index">
    <div class="slds-col slds-size_1-of-4 slds-p-around_small">
        <lightning:card title="{!item.Name}"
            footer="{!item.Brewery_Name__c}"
            iconName="custom:custom7">
            <p class="slds-p-horizontal_small">
                <div class="slds-gutters">
                    <div class="slds-col slds-size_1-of-3">
                        <img src="{!item.Image__c}" alt="{!item.Name}" />
                    </div>
                    <div class="slds-col slds-size_2-of-3">
                        Name       :  {!item.Name} <br/>
                        Id         :  {!item.Id__c} <br/>
                        Price      :  {!item.Price__c}
                    </div>
                </div>
            </p>
            <aura:set attribute="actions">
                <lightning:button name="{!item.Id}" label="View Details"
                    value="{!item.Name}"
                    variant="brand" onclick="{!c.showInfo}" />
            </aura:set>
            <aura:set attribute="footer">
                <lightning:button name="{!item.Id}" label="Add To Cart"
                    value="{!index}"
                    variant="brand" onclick="{!c.addToCart}" />
            </aura:set>
        </lightning:card>
    </div>
</aura:iteration>
<!-- <aura:if isTrue="{!v.beervalue !==null}"-->
CartInfoController.js

```javascript
({
    goToCart : function(component, event, helper) {
        var action = component.get('c.getCartId');
        // debugger;
        action.setParams({
            'proteinList' : component.get('v.beerNameList')
        });

        action.setCallback(this, function(response){
            var state = response.getState();
            //debugger;
            if(state === 'SUCCESS' || state === 'DRAFT'){
                var pageReference = component.find("navigation");
                var pageReferenceNav = {
                    "type": "standard__component",
                    "attributes": {
                        "componentName": "c__CartDetail"
                    },
                    "state": {
                        "c__cartId": response.getReturnValue()
                    }
                };
                pageReference.navigate(pageReferenceNav, true);
                debugger;
            }else if(state === 'INCOMPLETE'){
                console.log('User is offline System does not support offline');
            }else if(state === 'ERROR'){
                var errors = response.getError();
                if(errors || errors[0].pageMessage){
                    console.log(' page Error ', errors[0].pageMessage);
                }
                if(errors || errors[0].duplicateResults){
                    console.log(' duplicate Error ', errors[0].duplicateResults);
                }
            }else{
            }
        });
    }
});
```
LocationTriggerHandler.apxc

public class LocationTriggerHandler {

    @future(callout=true)
    public static void verifyAddress( String recordId ){
        Location__c loc = [Select Id, Name, Verified__c, Street__c, City__c, Postal_Code__c, State__c From Location__c Where Id=: recordId];
        String baseURL = 'https://us-street.api.smartystreets.com/street-address?auth-id=046d2eba-bb34-290d-4a98-c2793c970d8d&auth-token=g3LSvoAJ7wTDjTiOrG9j';
        baseURL+= '&street='+EncodingUtil.urlEncode(loc.Street__c, 'UTF-8')
            + '&city='+EncodingUtil.urlEncode(loc.City__c, 'UTF-8')
            + '&state='+EncodingUtil.urlEncode(loc.State__c, 'UTF-8')
            + '&zipcode='+EncodingUtil.urlEncode(loc.Postal_Code__c, 'UTF-8')
            + '&match=invalid&candidates=10';

        HttpRequest httpReq = new HttpRequest();
        httpReq.setMethod('GET');
        httpReq.setEndpoint(baseURL);
        //httpReq.setHeader('Content-Type', 'application/json');

        Http http = new Http();

        HttpResponse httpRes = new HttpResponse();

        try{
            httpRes = http.send(httpReq);
        }catch(Exception e){
            System.debug(e.getMessage());
        }
    }
}
System.debug(' ResponseBody ' + httpRes.getBody());
if (httpRes.getStatusCode() == 200 && httpRes.getStatus() == 'OK') {
    String responseBody = httpRes.getBody();
    if (!String.isBlank(responseBody) && responseBody.length() > 2) {
        loc.Verified__c = true;
    } else {
        loc.Verified__c = false;
    }
    update loc;
} else {
    TransactionLogHandler.doHandleExceptionWithError(httpRes.getBody(), 'LocationTriggerHandler');
}
} catch (System.CalloutException ex) {
    System.debug(' Exception Executed ' + ex.getStackTraceString());
    TransactionLogHandler.doHandleException(ex, 'LocationTriggerHandler');
}
REFERENCES

[1] Lightning Developer Guide -


[2] Libraries -

https://developer.salesforce.com/docs/component-library/overview/components


[4] Apex Developer Guide -


[5] Salesforce Developer Org -

https://developer.salesforce.com/signup