BOOKMYSPACE

Project report submitted to

Thanthai Periyar Government Arts and Science College (Autonomous),

Tiruchirappalli–620023

(Affiliated to Bharathidasan University, Tiruchirappalli– 24)

In partial fulfillment of the requirements for the award of the degree of

**MASTER OF SCIENCE**

**IN**

**COMPUTER SCIENCE**

# By

# M.SEDHURAMAN

# (Reg. No: 21PCS11)

Under the guidance of

**MS. R. ROSLIN MARY, M.Sc., PGDOR.,**

Associate Professor of Computer Science



**PG & RESEARCH DEPARTMENT OF COMPUTER SCIENCE THANTHAI PERIYAR GOVERNMENT ARTS AND SCIENCE COLLEGE (Autonomous), TIRUCHIRAPPALLI – 620 023**

**APRIL–2023**

**PG & RESEARCH DEPARTMENT OF COMPUTER SCIENCE**

**THANTHAI PERIYAR GOVERNMENT ARTS AND SCIENCE COLLEGE (Autonomous),**

**TIRUCHIRAPPALLI–620023**



CERTIFICATE

This is to certify that the project work entitled **“BOOKMYSPACE”,** submitted in partial fulfillment of the requirements for the award of the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE** to Thanthai Periyar Government Arts and Science College (Autonomous), Tiruchirappalli –23 affiliated to Bharathidasan University is a bonafide record of the work done by **M.SEDHURAMAN (Reg.No:21PCS11)** during the even semester of the academic year 2022-2023.

Signature of the Signature of the Guide

Head of the Department

Viva-voice examination for this project work held on

InternalExaminer ExternalExaminer

**ACKNOWLEDGEMENT**

I would like to express my sincere thanks to our Honorable Principal **Dr. P.S. VIJAYALAKSHMI, M.Com., M.B.A., M.Phil., Ph.D.,** who grant me permission to do this project for three months.

I wish to express my sincere thanks to [**Mr. T. KANNADASAN**](http://www.periyarevrcollege.ac.in/dept_cs_staff.php), **M.C.A.,** Headofthe Department of Computer Science for his valuable support and helping me in various way throughout this project duration.

My heartfelt thanks to my guide **MS. R. ROSLIN MARY, M.Sc., PGDOR.,** Associate Professor, Department of Computer Science. She encourages me more and more.

I also thank all the staff of the department and programmer **Mr. T. SARAVANAN**  for their support and help during the period my project. They are the best because they brought out the best in students.

**DECLARATION**

I **M.SEDHURAMAN (Reg.No:21PCS11),** declare

that the project entitled **“BOOKMYSPACE”** submitted to the PG & Research Department Of Computer Science**,** Thanthai Periyar Government Arts and Science College (Autonomous), Tiruchirappalli – 23 affiliated to Bharathidasan University, Tiruchirappalli – 24 in partial fulfilment of the requirements for the award of the degree of Master of Science in Computer Science is the original project work done by me and it has not been submitted anywhere else for any degree.

Place: Tiruchirappalli – 23 Name: M.SEDHURAMAN

Date : Reg. No : 21PCS11

**BOOKMYSPACE**

**ABSTRACT**

BookMySpace is a fictitious Space reservation company. This application helps manage space reservations and find the right theme for their customers on the salesforce platform. The application provides an easy-to-use interface that allows users to browse through various types of venues, view detailed information about each space, and book their preferred space for their event. The application features a search functionality that allows users to filter their search results by location, capacity, amenities, and price range. BookMySpace agents can use the Reservation Manager to browse the list of leads and contacts that need reservations. The Reservation Manager is a Lightning page built with custom Lightning web components and flows. To make a reservation, select a Lead or Contact card and create a draft reservation. Make sure City, State/Province, Name, and Email pre-populated. Click the Draft Reservation button and fill in the following reservation details, Where do they want the space, When do they need the space, how many peoples, How long do they want the space. Click the Next button. A new reservation record is created. Design a space for reservations. Choose the Space Designer menu From the Open Reservations list, click one of the Open Reservation items. From Filter By Space Type, select a space. Click Add and Go To Reservation. The reservation record and the reservation moved from the draft stage to the proposal ready stage.

**CONTENT**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **CHAPTER NAME** | **PAGE NO** |
| 1. | **INTRODUCTION** |  |
| 2. | **SYSTEM STUDY**  2.1 Existing System  2.2 Proposed System |  |
| 3. | **SYSTEM REQUIREMENTS**  3.1 Packages Selected  3.2 Hardware Specification  3.3 Software Specification  3.4 Software Description  3.5 Data Flow Diagram |  |
| 4. | **SYSTEM DESIGN**  4.1 Architectural Design  4.2 I/O Form design  4.3 Database Design  4.4 E-R Diagram |  |
| 5. | **TESTING**  5.1 UnitTesting  5.2 IntegrationTesting  5.3 Data Management  5.4 UserInterface Testing  5.5 Workflow Testing  5.6 Security Testing |  |
| 6. | **CONCLUSION** |  |
|  | **ANNEXURES**  Annexures - a - Source Code  Annexures - b - Output |  |
|  | **BIBLIOGRAPHY** |  |

**CHAPTER - 1**

**INTRODUCTION**

BookMySpace is a web and Salesforce application that helps event organizers and venue owners find, book, and manage event spaces. The platform offers an easy-to-use interface that allows users to search for event spaces based on location, capacity, amenities, and price range. The application offers real-time availability of spaces and allows users to make reservations directly from the platform. BookMySpace web and Salesforce application aim to simplify the process of finding and booking event spaces, making it a one-stop solution for event organizers and venue owners alike. The salesforce application is built using modern web technologies such as Apex for the backend, and HTML, CSS and javascript for the frontend, ensuring a smooth and intuitive user experience. The Salesforce application is a customer relationship management (CRM) platform that integrates with the BookMySpace web application to provide a seamless experience for event organizers and venue owners. BookMySpace application is designed to meet the needs of event organizers and venue owners, providing them with the tools they need to find, book, and manage event spaces effectively. The platform is scalable, reliable, and easy to use, making it a popular choice for users around the world.

**CHAPTER – 2**

**SYSTEM STUDY**

**2.1 EXISTING SYSTEM**

The existing system has many flaws which make it inefficient to carry on with it. Space reservation work of the referred company is done manually. It becomes difficult maintaining all details of spaces, customers and the services. The execution of the event sometimes delays due to some unmanageable obstruction in planning. As far as quality is concerned it is ok and not satisfying but not as good when handled using a computerized system.

**Disadvantages:**

* It is time consuming as the system is handled manually.
* Assurance of data security is not given.
* In long run, it is difficult to maintain records.
* Manpower requirement is huge.

**2.2 PROPOSED SYSTEM**

BookMySpace is a powerful software application used to manage and book meeting rooms, bookmyspace, and other facilities in shared office spaces. When implemented in the Salesforce platform, it provides a robust and user-friendly solution for managing facilities and bookings the proposed system for BookMySpace in Salesforce aims to provide a more efficient, user-friendly, and customizable solution for facility management and bookings. By incorporating these enhancements, users will be able to streamline their operations and better manage their facilities and bookings.

**Advantages:**

* Easy to create and manage the events
* Improved Security and Improved User Interface
* Reduces The Workload and Less Manpower

**CHAPTER - 3**

**SYSTEM REQUIREMENTS**

**3.1 PACKAGES SELECTED**

* Front End : LWC(HTML, CSS, JavaScript,XML)
* Back End : Apex

**3.2 HARDWARE SPECIFICATION**

* Processor : AMD PRO A4-4350B R4, 5 COMPUTE CORES 2C+3G 2.50 GHz
* Ram : 4 GB
* Hard Disk : 415GB
* Compact Disk : DVD
* Keyboard : Standard Keyboard

**3.3 SOFTWARE SPECIFICATION**

* Operating System : Windows 10 Pro Education
* Front End : LWC (HTML, CSS, JavaScript,XML)
* Back End : Apex
* Tool : Visual Studio Code
* Server : Salesforce
* Database : Salesforce Object Query Language (SOQL)

**3.4 SOFTWARE DESCRIPTION**

**LWC : ( HTML , CSS , JavaScript )**

Salesforce Lightning Web Components (LWC) is a modern programming model for building web applications on the Salesforce platform. It is based on web standards like HTML, CSS, and JavaScript, and allows developers to create reusable components that can be used in multiple applications.

LWC provides a lightweight, fast, and efficient way to create user interfaces that are optimized for performance.

Salesforce LWC is built on top of the Lightning Component framework and provides an easy-to-use development experience for building custom applications and extending the functionality of existing Salesforce applications.

LWC supports both server-side and client-side rendering, and provides built-in support for data binding, event handling, and other common UI features.

**HTML :**

HTML, or Hypertext Markup Language, is a markup language used to create and structure content for web pages. It provides a way for web developers to define the structure and content of a web page, including headings, paragraphs, images, links, and other elements.

HTML uses a set of tags or markup to define the various elements of a web page, such as the <html> tag to define the overall structure of the page, the <head> tag to specify metadata about the page, and the <body> tag to define the content of the page.

HTML also allows web developers to add styling to their pages using CSS, or Cascading Style Sheets, which provides a way to define the look and feel of a web page, such as fonts, colours and layout.

**CSS :**

CSS or Cascading Style Sheets is a style sheet language used to define the visual presentation of web pages written in HTML or XHTML.

CSS allows web developers to separate the content of a web page from its presentation, making it easier to maintain and update the design of a site.

CSS works by associating style rules with HTML elements. These rules define the properties and values that determine the appearance of the element, such as its font, color, size, and position.

CSS allows developers to apply styles to individual elements, groups of elements, or the entire page.

**JAVASCRIPT :**

JavaScript is a high-level, dynamic, interpreted programming language that is used to add interactivity to web pages.

JavaScript is a client-side language that runs directly in the web browser, allowing developers to create dynamic web applications that respond to user interactions.

JavaScript code is embedded directly into HTML web pages using the <script> tag, or included as an external file using the <script src="..."> tag.

JavaScript supports object-oriented programming concepts, including classes, inheritance, and encapsulation.

JavaScript supports functional programming concepts, including first-class functions, closures and higher-order functions.

**XML :**

XML stands for Extensible Markup Language, and it is a markup language that is used to structure and store data in a hierarchical format.

XML is a text-based format that is designed to be human-readable and machine-readable.

XML documents are made up of elements, which are defined by start and end tags, and can contain other elements or text.

XML documents can be validated against a set of rules using a Document Type Definition (DTD) or an XML Schema.

XML is a flexible language, and can be customized to meet specific needs using namespaces and attributes.

**APEX :**

Apex is a programming language that is specifically designed for developing applications on the Salesforce platform. It is a strongly-typed, object-oriented language that allows developers to write custom business logic, including triggers, classes, and controllers. Apex syntax is similar to Java and C#, and it is compiled and executed on the Salesforce server.

**Some key features of Apex include:**

Integration with the Salesforce platform: Apex has built-in support for accessing and manipulating Salesforce data and metadata.

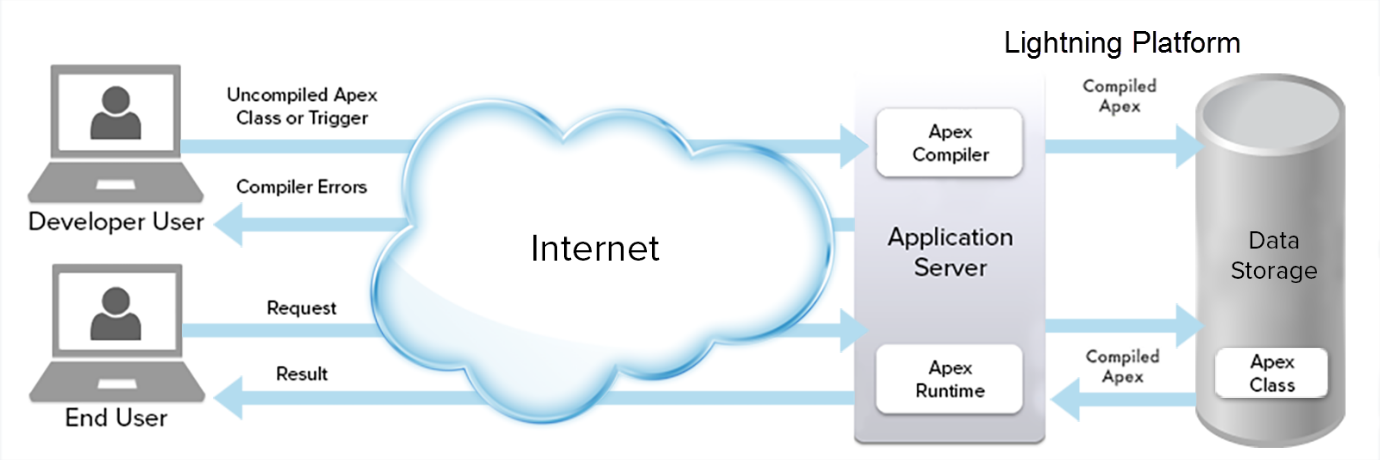
Object-oriented programming: Apex supports object-oriented programming concepts like classes, interfaces, and inheritance.

Strongly-typed: Variables in Apex must be explicitly declared and their data type cannot be changed.

Governor limits: Apex enforces governor limits to ensure that code does not monopolize server resources.

Testing framework: Apex has a built-in testing framework that allows developers to write unit tests to ensure code quality and prevent regression bugs.

Overall, Apex is a powerful language that allows developers to build custom business logic on the Salesforce platform.



**TOOL :( Visual Studio Code)**

Visual Studio Code is a free, open-source, lightweight, and powerful code editor developed by Microsoft. It is designed to provide developers with a seamless and efficient development experience. It is available for Windows, Linux, and macOS operating systems.

Visual Studio Code supports a wide range of programming languages such as C++, C#, Python, JavaScript, TypeScript, and many more. It offers features such as IntelliSense, syntax highlighting, code completion, debugging, version control, and much more.

Visual Studio Code is a popular and powerful code editor that provides developers with a rich set of features, enabling them to develop and debug their code quickly and efficiently.

**SERVER : ( Salesforce )**

Salesforce is a cloud-based customer relationship management (CRM) platform that provides a suite of tools and services for businesses to manage their sales, marketing, customer service, and analytics operations.

The platform enables businesses to store and manage customer information, automate sales processes, track customer interactions, and analyze customer behavior to improve business operations.

Salesforce is a cloud-based platform, which means that it does not have a traditional server architecture. Instead, the platform is hosted on Salesforce's own servers, which are located in data centers around the world. These servers are managed and maintained by Salesforce's team of engineers and technicians, who ensure that the platform is running smoothly and securely at all times.

Salesforce's cloud-based architecture provides several advantages over traditional server-based systems. For example, it allows businesses to access the platform from anywhere with an internet connection, and it provides automatic updates and maintenance without the need for businesses to manage their own servers. Additionally, Salesforce's cloud-based approach allows businesses to scale their usage up or down as needed, without having to make significant investments in hardware or infrastructure.

**DATABASE : ( SOQL)**

Salesforce SOQL (Salesforce Object Query Language) is a query language used to retrieve data from Salesforce databases.

SOQL is similar to SQL, but it is specific to the Salesforce platform and is used to query data from Salesforce objects such as accounts, contacts, leads, and opportunities.

SOQL queries can be executed using the Salesforce user interface, the Salesforce API, or through other Salesforce development tools.

SOQL queries are executed against the Salesforce database, and the results can be returned as a list of records or as a single record.SOQL provides a number of features to filter, group, and sort data based on specific criteria. Users can filter data using WHERE clauses, specify fields to be returned in the query, and sort the results by one or more fields.

**3.5 DATA FLOW DIAGRAM**

User

Finish Reservation

Contact List

Lead List

Market List

Space List

Reservation List

Choose Space

Create Reservation

Contact

Lead

Market

Space

Reservation

Space Designer

ReservationManager

BookMySpace

App Launcher

Salesforce

Login

SignUp

**Reservation Manager Diagram :**

agents

reservation manager

lead contact

draft reservation

create reservation

finish

**Space Designer Diagram :**

agents

space designer

open reservation

filter by space and types

Choose Space

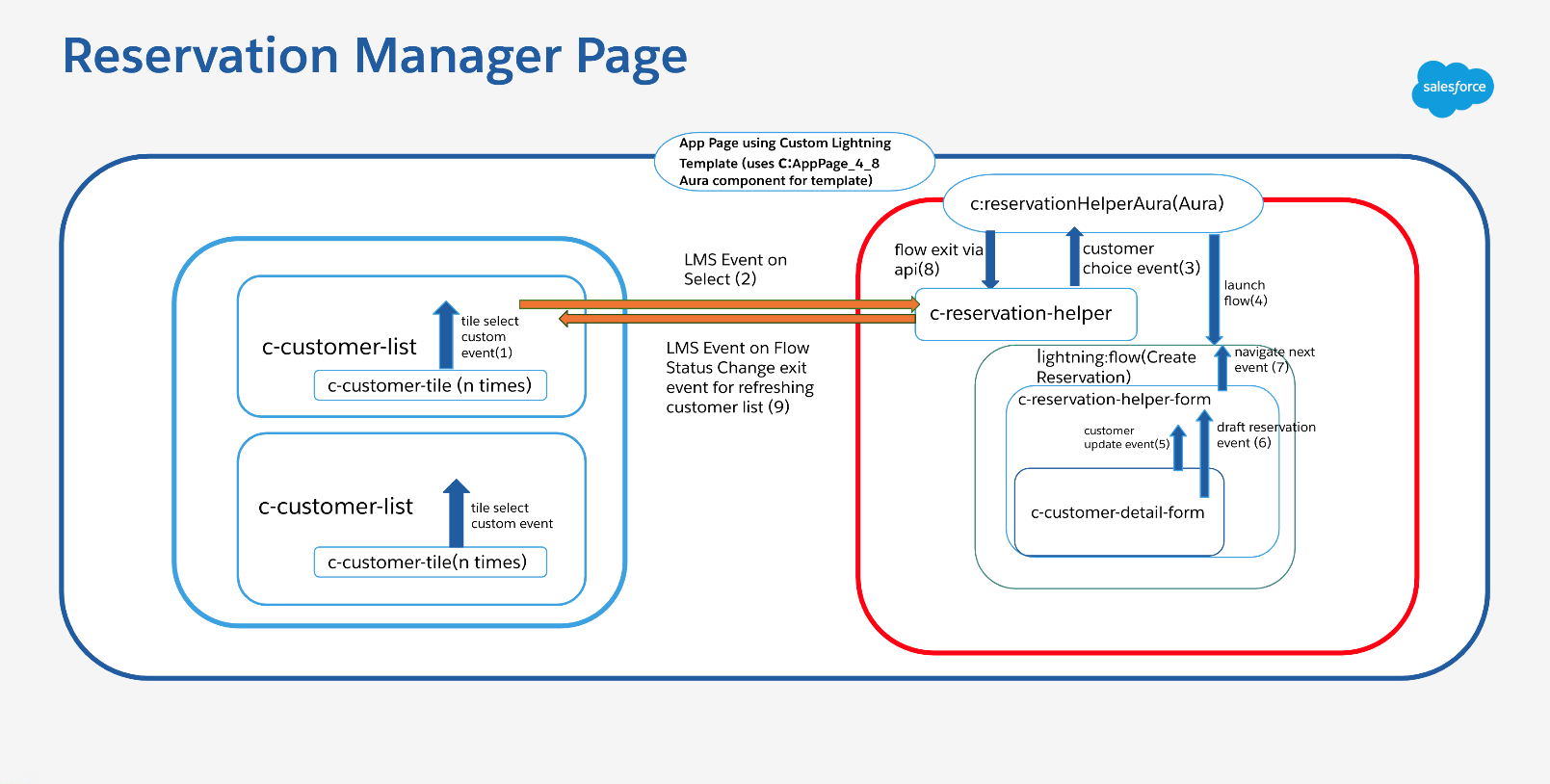
Finish Reservation

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 ARCHITECTURAL DESIGN**

**Reservation Manager Architecture :**

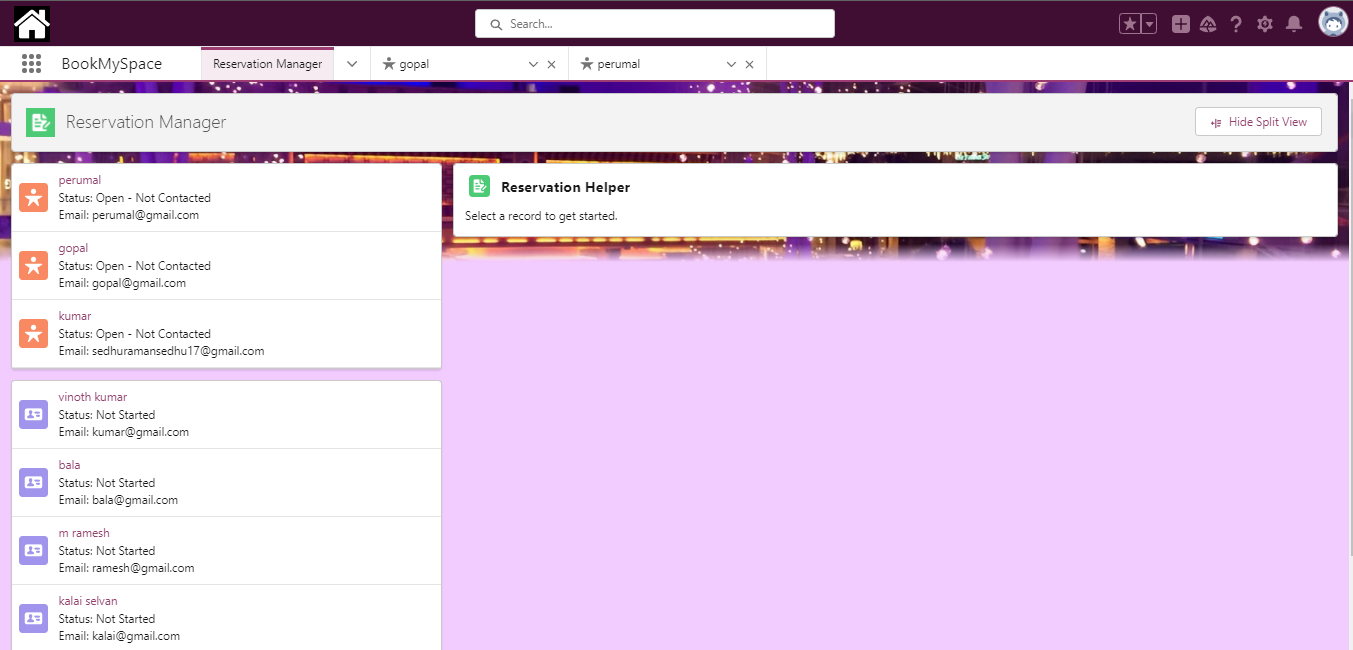


**Space Designer Architecture :**

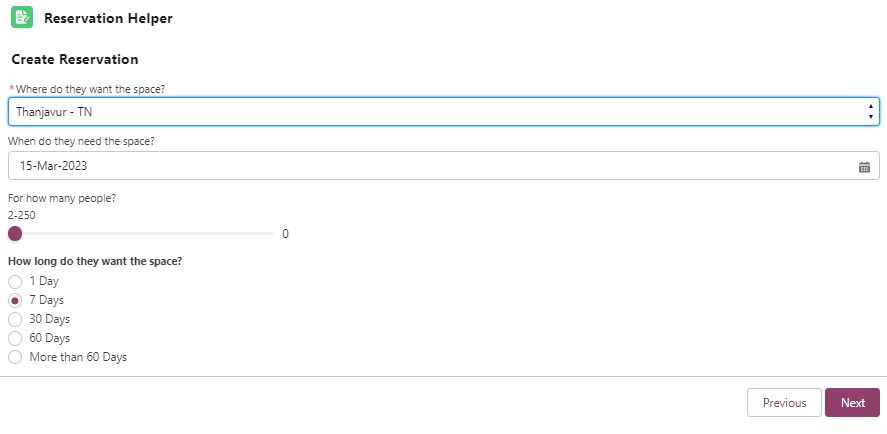
****

**4.2 I/O FORM**

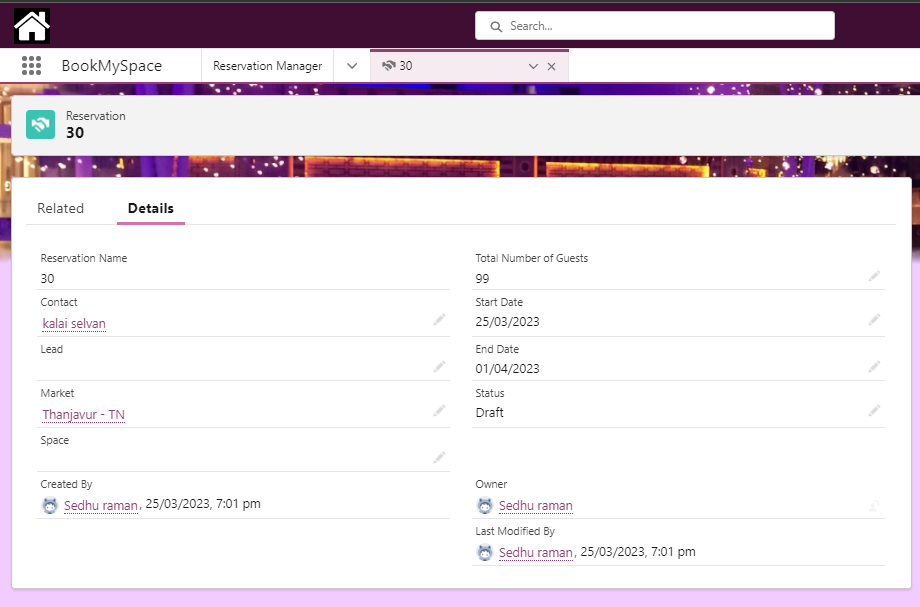
**Reservation Manager Page:**

****

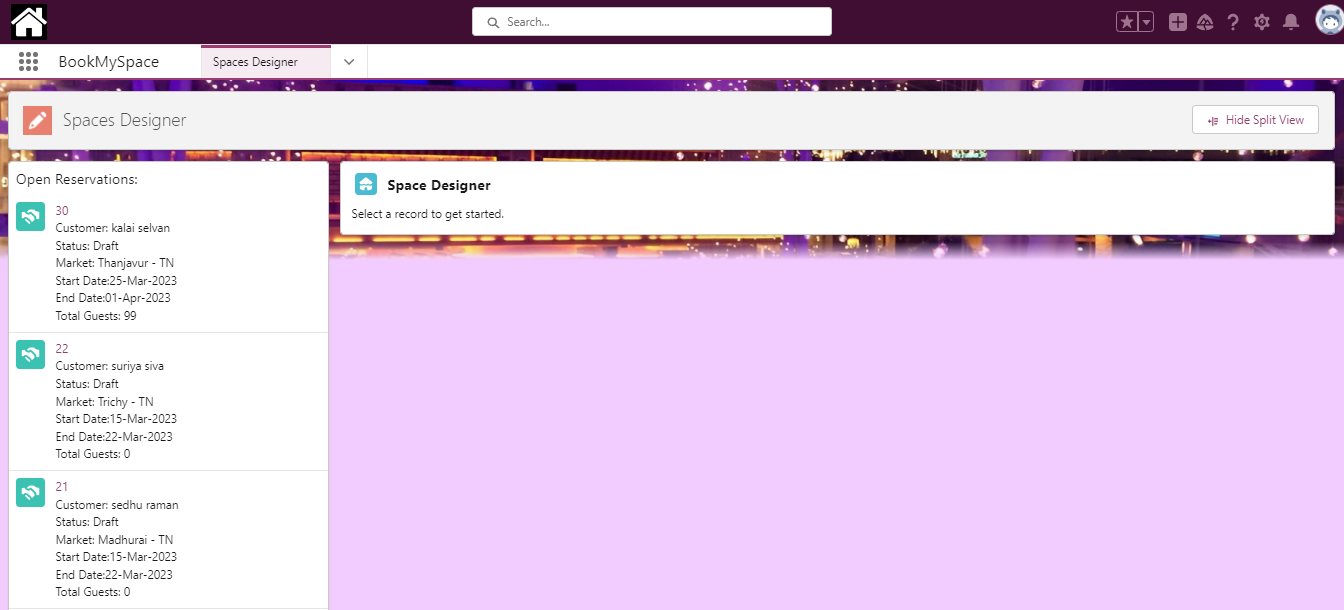
**Reservation Manager Form :**

****

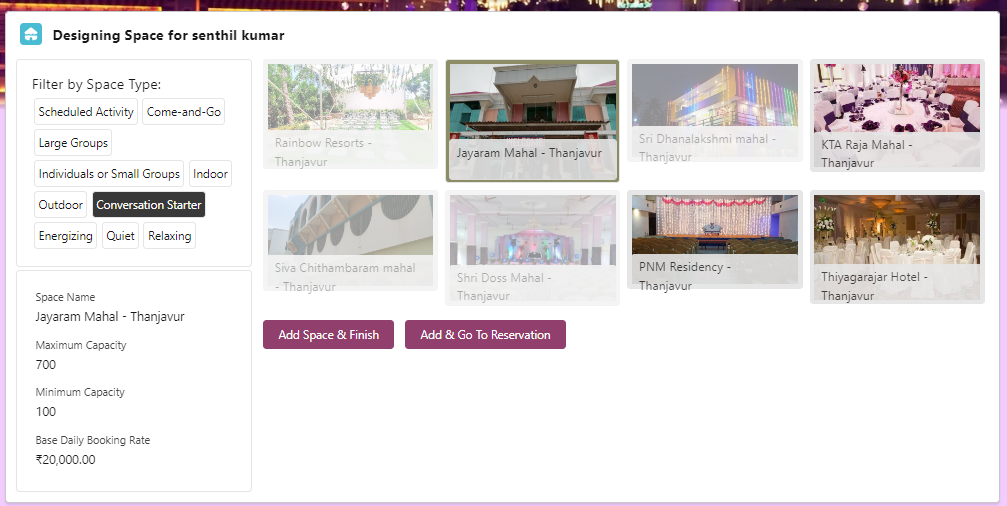
**Reservation Manager Output :**

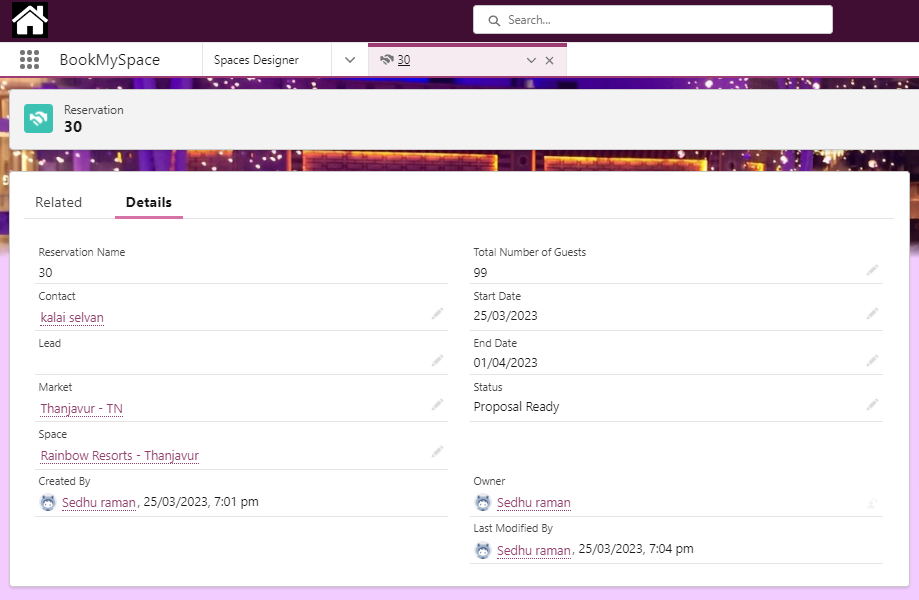
****

**Space DesignerPage :**

****

**Space Designer Form:**

****

****

**Space Designer Output :**

**4.3 DATA BASE DESIGN**

**Objects :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **LABEL** | **API NAME** | **TYPE** |
| 1 | Market | Market\_\_c | Custom Object |
| 2 | Space | Space\_\_c | Custom Object |
| 3 | Reservation | Reservation\_\_c | Custom Object |
| 4 | Lead | Lead | Standard Object |
| 5 | Contact | Contact | Standard Object |

**FIELDS AND RELATIONSHIP:**

**Market Fields :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **FIELD LABEL** | **FIELD NAME** | **DATA TYPE** | **SIZE** |
| 1 | city | City\_\_c | Text | 50 |
| 2 | country | Country\_\_c | Picklist | - |
| 3 | Created by | CreatedById | Lookup(user) | - |
| 4 | Last Modified By | LastModifiedById | Lookup(user) | - |
| 5 | Market Name | Name | Text | 80 |
| 6 | Owner | OwnerId | Lookup(user,Group) |  |
| 7 | Predicted Booking Rate | Predicted\_booking\_Rate\_\_c | Number | (5,2) |
| 8 | State | State\_\_c | Picklist | - |
| 9 | Total Booking Rate | Total\_Booking\_Rate\_\_c | Roll-up Summary (SUM space) | - |

**Space Fields :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **FIELD LABEL** | **FIELD NAME** | **DATE TYPE** | **SIZE** |
| 1 | Base Daily Booking Rate | Base\_Daily\_Booking\_Rate\_\_c | Currency | (10,2) |
| 2 | Category | Category\_\_c | Picklist | - |
| 3 | Created By | CreatedById | Lookup(user) | - |
| 4 | Last Modified By | LastModifiedById | Lookup(user) | - |
| 5 | Market | Market\_\_c | Lookup(user) | - |
| 6 | Maximum capacity | Maximum\_Capacity\_\_c | Number | (18,0) |
| 7 | Minimum Capacity | Minimum\_Capacity\_\_c | Number | (18,0) |
| 8 | Picture URL | Picture\_URL\_\_c | URL | 255 |
| 9 | Predicted Booking Rate | Predicted\_Booking\_Rate\_\_c | Currency | (16,2) |
| 10 | Predicted Demand | Predicted\_Demand\_\_c | Text | 80 |
| 11 | Space Name | Name | Text | 80 |
| 12 | Type | Type\_\_c | Picklist(Multi-Select) | - |

**Reservation fields:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **FIELD LABEL** | **FIELD NAME** | **DATATYPE** | **SIZE** |
| 1 | Account | Account\_\_c | Lookup(Account) | - |
| 2 | [Contact](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNCl/view) | Contact\_\_c | Lookup(Contact) | - |
| 3 | [Created By](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/CreatedBy/view) | CreatedById | Lookup(User) | - |
| 4 | [End Date](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNCq/view) | End\_Date\_\_c | Date | - |
| 5 | [End Time](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaND0/view) | End\_Time\_\_c | Time | - |
| 6 | [Last Modified By](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/LastModifiedBy/view) | LastModifiedById | Lookup(User) | - |
| 7 | [Lead](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNDA/view) | Lead\_\_c | Lookup(Lead) | - |
| 8 | [Market](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNDK/view) | Market\_\_c | Lookup(Market) | - |
| 9 | [Owner](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/Owner/view) | OwnerId | Lookup(User,Group) | - |
| 10 | [Reservation Name](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/Name/view) | Name | Auto Number |  |
| 11 | [Space](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNDP/view) | Space\_\_c | Lookup(Space) | - |
| 12 | [Start Date](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNDZ/view) | Start\_Date\_\_c | Date | - |
| 13 | [Start Time](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNDe/view) | Start\_Time\_\_c | Time | - |
| 14 | [Status](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNDj/view) | Status\_\_c | Picklist | - |
| 15 | [Total Number of Guests](https://wise-fox-noicrv-dev-ed.trailblaze.lightning.force.com/lightning/setup/ObjectManager/01I2w000003G8T3/FieldsAndRelationships/00N2w00000SaNHw/view) | Total\_Number\_of\_Guests\_\_c | Number | (18,0) |

**Contact Fields :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **FIELD LABEL** | **FIELD NAME** | **DATA TYPE** | **SIZE** |
| 1 | Email | Email | Email | - |
| 2 | Name | Name | Name | - |
| 3 | Mailing Address | Mailing Address | Address |  |
| 4 | Mobile | MobilePhone | Phone | - |
| 5 | Reservation Status | Reservation\_Status\_\_c | Picklist | - |
| 6 | Title | Title | Text | 30 |

**Lead Fields :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **FIELD LABEL** | **FIELD NAME** | **DATA TYPE** | **SIZE** |
| 1 | Address | Address | Address | - |
| 2 | Comapny | Company | Text | 255 |
| 3 | Email | Email | Email | - |
| 4 | Name | Name | Name | - |
| 5 | Industry | Industry | Picklist | - |
| 6 | Mobile | MobilePhone | Phone | - |
| 7 | Title | Title | Text | 128 |
| 8 | Lead Status | Status | Picklist | - |

**4.4 E-R DIAGRAM**

ER Diagram stands for Entity Relationship Diagram, that display the relationship of entity sets stored in a database.

emaill

city

emaill

city

name

state

state

name

contact

lead

agent

Create

reservation

design

Space designer

Reservation

manager

Space name

Minimum capacity

Booking rate

Maximum capacity

filter

day

indoor

outdoor

realxing

date

No of

peoples

quiet

city

**CHAPTER-5**

**TESTING**

**5.1 UNIT TESTING**

Unit testing in Salesforce BookMySpace involves testing individual components of the code to ensure that they function as expected. Unit testing in Salesforce BookMySpace should focus on ensuring that each component of the code works as expected, and that all required functionality is covered by the tests. By testing each component thoroughly, developers can identify and resolve any issues or defects, and ensure that the code is of high quality.

**5.2 INTEGRATION TESTING**

This involves testing the integration between BookMySpace and Salesforce, ensuring that data is exchanged correctly, that errors are detected and resolved, and that the integration is reliable and secure. Testing should verify that data is synced correctly between the two systems, and that any updates made in one system are reflected accurately in the other.

**5.3 DATA MANAGEMENT TESTING**

This involves testing the management of data between BookMySpace and Salesforce, ensuring that data is collected accurately, stored securely, and can be accessed easily. Testing should verify that data is synced correctly between the two systems, and that any conflicts or duplicates are resolved.

**5.4 USER INTERFACE TESTING**

This involves testing the user interface for BookMySpace within the Salesforce platform, ensuring that it is intuitive, easy to use, and consistent with the rest of the Salesforce interface. Testing should verify that all buttons, links, and menus are functional, and that users can navigate between the two systems easily.

**5.5 WORKFLOW TESTING**

This involves testing the workflow for BookMySpace within the Salesforce platform, ensuring that users can create and manage workflows for booking and managing meeting rooms and event spaces. Testing should verify that workflows are created correctly, that they function as intended, and that any issues or errors are resolved promptly.

**5.6 SECURITY TESTING**

This involves testing the security of the integration between BookMySpace and Salesforce, ensuring that data is stored securely, and that access is restricted to authorized users only. Testing should verify that data is encrypted during transmission, and that user permissions are set up correctly to prevent unauthorized access.

Overall, testing BookMySpace in Salesforce should focus on ensuring that the integration between the two systems is reliable, secure, and meets the needs of the users. By testing the integration thoroughly, developers can identify and resolve any issues or defects, and ensure that the integration is of high quality.

**CHAPTER-6**

**CONCLUSION**

BookMySpace is a Salesforce application that provides a complete solution for find and reserve the venues. With BookMySpace, users can easily find event venues, price, types, and much more. The application is designed to be user-friendly, intuitive, and customizable to meet the unique needs of each organization. BookMySpace leverages the power of the Salesforce platform to provide a seamless experience for users. The application is built with Visualforce and Apex, and it integrates with a wide range of Salesforce tools and services, including Salesforce Mobile App, Salesforce Communities. BookMySpace is a powerful and versatile application that can help organizations of all sizes streamline their booking the correct event venues. With its robust set of features, user-friendly interface, and deep integration with Salesforce, BookMySpace is a top choice for companies booking to their event venues.

**ANNEXURES**

**ANNEXURES –A**

**SOURCE CODE**

**ReservationManagerController.cls**

public class ReservationManagerController {

//Note: This is a sample, so we're using 'LIMIT' to simulate proper set/pagination controls

@auraEnabled(cacheable=true)

public static List<CustomerServices.Customer>getCustomerList(

String sObjectType

) {System.debug(sObjectType);

List<CustomerServices.Customer> customers = new List<CustomerServices.Customer>();

Customer\_Fields\_\_mdt c = [

SELECT

Customer\_Name\_\_r.QualifiedAPIName,

Customer\_Email\_\_r.QualifiedAPIName,

Customer\_City\_\_r.QualifiedAPIName,

Customer\_State\_\_r.QualifiedAPIName,

Customer\_Status\_\_r.QualifiedAPIName,

Customer\_Draft\_Status\_Values\_\_c

FROM Customer\_Fields\_\_mdt

WHERE Sobject\_Type\_\_r.QualifiedAPIName = :sObjectType

WITH SECURITY\_ENFORCED

LIMIT 1

];

List<String>draftStatus = new List<String>();

draftStatus = c.Customer\_Draft\_Status\_Values\_\_c?.split(',');

String query = 'SELECT ';

query += c.Customer\_Name\_\_r.QualifiedAPIName + ', ';

query += c.Customer\_Email\_\_r.QualifiedAPIName + ', ';

query += c.Customer\_City\_\_r.QualifiedAPIName + ', ';

query += c.Customer\_State\_\_r.QualifiedAPIName + ', ';

query += c.Customer\_Status\_\_r.QualifiedAPIName + ', ';

query += 'Id';

query += ' FROM ' + sObjectType;

if (draftStatus.size() > 0) {

query +=

' WHERE ' +

c.Customer\_Status\_\_r.QualifiedAPIName +

' IN: draftStatus';

}

query += ' WITH SECURITY\_ENFORCED ORDER BY CreatedDate DESC LIMIT 6';

for (Sobjectsobj :Database.query(query)) {

CustomerServices.Customer customer = new CustomerServices.Customer(

(String) sobj.get(c.Customer\_Name\_\_r.QualifiedAPIName),

(String) sobj.get(c.Customer\_Email\_\_r.QualifiedAPIName),

(String) sobj.get(c.Customer\_City\_\_r.QualifiedAPIName),

(String) sobj.get(c.Customer\_State\_\_r.QualifiedAPIName),

(String) sobj.get(c.Customer\_Status\_\_r.QualifiedAPIName),

(String) sobj.get('Id')

);

customers.add(customer);

}

return customers;

}

@auraEnabled(cacheable=true)

public static List<Reservation\_\_c>getOpenReservations() {

return [

SELECT

Id,

Name,

Contact\_\_c,

Contact\_\_r.Name,

Lead\_\_c,

Lead\_\_r.Name,

Market\_\_c,

Market\_\_r.Name,

Status\_\_c,

Start\_Date\_\_c,

End\_Date\_\_c,

Total\_Number\_of\_Guests\_\_c

FROM Reservation\_\_c

WHERE Status\_\_c IN ('Not Started', 'Draft')

WITH SECURITY\_ENFORCED

LIMIT 10];}

}

**ReservationManagerController.cls-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<ApexClass

xmlns="urn:metadata.tooling.soap.sforce.com"

fqn="reservationManagerController">

<apiVersion>55.0</apiVersion>

<status>Active</status>

</ApexClass>

**LWC**

**customerList.html**

<template>

<div class="slds-card">

<template if:true={errorMsg}>

<c-error-panel

friendly-message={msgForUser}

errors={errorMsg}

></c-error-panel>

</template>

<ul class="slds-list\_verticalslds-has-dividers\_bottom-space">

<template for:each={customers} for:item="record">

<li class="slds-item" key={record.Id}>

<c-customer-tile

customer={record}

object={sobject}

oncustomerselect={publishSelect}

></c-customer-tile>

</li>

</template>

</ul>

</div>

</template>

**customerList.js**

import { LightningElement, api, wire } from 'lwc';

import { refreshApex } from '@salesforce/apex';

import getCustomerList from '@salesforce/apex/reservationManagerController.getCustomerList';

import TILE\_SELECTION\_MC from '@salesforce/messageChannel/Tile\_Selection\_\_c';

import FLOW\_STATUS\_CHANGE\_MC from '@salesforce/messageChannel/Flow\_Status\_Change\_\_c';

import {

subscribe,

APPLICATION\_SCOPE,

MessageContext,

publish

} from 'lightning/messageService';

export default class CustomerList extends LightningElement {

@apisobject;

customers = [];

errorMsg;

msgForUser;

wiredRecords;

@wire(MessageContext)

messageContext;

subscribeToMessageChannel() {

subscribe(

this.messageContext,

FLOW\_STATUS\_CHANGE\_MC,

(message) =>this.handleMessage(message),

{ scope: APPLICATION\_SCOPE }

);

}

handleMessage(message) {

if (

message.flowName === 'createReservation' &&

message.status === 'FINISHED' &&

message.state

) {

if (message.state.sobjecttype === this.sobject) {

refreshApex(this.wiredRecords);}}}

connectedCallback() {

this.subscribeToMessageChannel();

}

@wire(getCustomerList, { sObjectType: '$sobject' })

wiredCustomerData(value) {

this.wiredRecords = value;

if (value.error) {

this.errorMsg = value.error;

this.msgForUser = 'There was an issue loading customers.';

} else if (value.data) {

this.customers = value.data;

}

}

publishSelect(event) {

const payload = { tileType: 'customer', properties: event.detail };

publish(this.messageContext, TILE\_SELECTION\_MC, payload);

}

}

**customerList.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>true</isExposed>

<targets>

<target>lightning\_\_AppPage</target>

<target>lightning\_\_HomePage</target>

</targets>

<targetConfigs>

<targetConfig targets="lightning\_\_AppPage, lightning\_\_HomePage">

<property name="sobject" datasource="Lead,Contact" type="String" />

</targetConfig>

</targetConfigs>

</LightningComponentBundle>

**APEX**

**CustomerServices.cls**

public inherited sharing class CustomerServices {

@auraEnabled(cacheable=true)

public static Customer getCustomerFields(String objectType) {

Customer customer;

for (Customer\_Fields\_\_mdtc : [

SELECT

Customer\_Name\_\_r.QualifiedAPIName,

Customer\_Email\_\_r.QualifiedAPIName,

Customer\_City\_\_r.QualifiedAPIName,

Customer\_State\_\_r.QualifiedAPIName,

Customer\_Status\_\_r.QualifiedAPIName

FROM Customer\_Fields\_\_mdt

WHERE Sobject\_Type\_\_r.QualifiedAPIName = :objectType

WITH SECURITY\_ENFORCED

]) {

customer = new Customer(

c.Customer\_Name\_\_r.QualifiedAPIName,

c.Customer\_Email\_\_r.QualifiedAPIName,

c.Customer\_City\_\_r.QualifiedAPIName,

c.Customer\_State\_\_r.QualifiedAPIName,

);

}

return customer;

}

public class Customer {

//attributes that we associate with customers

@auraEnabled

public String email { get; set; }

@auraEnabled

public String name { get; set; }

@auraEnabled

public String city { get; set; }

@auraEnabled

public String state { get; set; }

@auraEnabled

public String status { get; set; }

@auraEnabled

public String Id { get; set; }

//putting them together, we get our wrapper object:

public Customer(

String name,

String email,

String city,

String state,

String status,

String custId

) {

this.name = name;

if (String.isNotEmpty(email))

this.email = email;

if (String.isNotEmpty(city))

this.city = city;

if (String.isNotEmpty(state))

this.state = state;

if (String.isNotEmpty(status))

this.status = status;

if (String.isNotEmpty(custId))

this.Id = custId;

}

}

}

**customerServices.cls-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<ApexClass

xmlns="urn:metadata.tooling.soap.sforce.com"

fqn="customerServicesTest">

<apiVersion>55.0</apiVersion>

<status>Active</status>

</ApexClass>

**Apex**

**TestDataFactory.cls**

@isTest

public class TestDataFactory {

public static List<Account>makeAccounts(Integer acctNum) {

//make some Accounts

List<Account>testAccts = new List<Account>();

for (Integer x = 0; x <acctNum; x++) {

Account a = new Account();

a.Name = 'Test Account' + x;

testAccts.add(a);

}

return testAccts;

}

public static List<Contact>makeContacts(Integer contactNum) {

List<Contact>testPeeps = new List<Contact>();

for (Integer x = 0; x <contactNum; x++) {

Contact c = new Contact();

c.FirstName = 'Testing' + x;

c.LastName = 'Contact' + x;

c.MailingCity = 'Test City' + x;

c.MailingStreet = 'Test City Ave' + x;

c.MailingState = Math.mod(x, 2) == 0

? null

: (Math.mod(x, 3) == 0 ? 'NY' : 'OR');

c.MailingPostalCode = Math.mod(x, 2) == 0

? '1011 AJ'

: (Math.mod(x, 3) == 0 ? '10006' : '97230');

c.MailingCountry = Math.mod(x, 2) == 0

? 'Amsterdam'

: 'United States';

c.Reservation\_Status\_\_c = 'Not Started';

testPeeps.add(c);

}

return testPeeps;

}

public static List<Lead>makeLeads(Integer leadNum) {

List<Lead>testPeeps = new List<Lead>();

for (Integer x = 0; x <leadNum; x++) {

Lead l = new Lead();

l.FirstName = 'Test' + x;

l.LastName = 'Lead' + x;

l.Company = 'Test Company' + x;

l.City = 'Test City' + x;

l.Street = 'Test Street' + x;

l.State = Math.mod(x, 2) == 0

? null

: (Math.mod(x, 3) == 0 ? 'NY' : 'OR');

l.Country = Math.mod(x, 2) == 0 ? 'Amsterdam' : 'United States';

l.PostalCode = Math.mod(x, 2) == 0

? '1011 AZ'

: (Math.mod(x, 3) == 0 ? '10006' : '97230');

testPeeps.add(l);

}

return testPeeps;

}

public static List<Market\_\_c>makeMarkets(Integer marketsNum) {

List<Market\_\_c> markets = new List<Market\_\_c>();

for (Integer x = 0; x <marketsNum; x++) {

Market\_\_c m = new Market\_\_c();

m.Name = 'Test Market' + x;

m.City\_\_c = 'Test City' + x;

m.State\_\_c = Math.mod(x, 2) == 0

? null

: (Math.mod(x, 3) == 0 ? 'NY' : 'OR');

m.Country\_\_c = Math.mod(x, 2) == 0 ? 'Amsterdam' : 'United States';

markets.add(m);

}

return markets;

}

public static List<Space\_\_c>makeSpaces(Integer spacesNum) {

List<Space\_\_c> spaces = new List<Space\_\_c>();

for (Integer x = 0; x <spacesNum; x++) {

Space\_\_c s = new Space\_\_c();

s.Name = 'Test Space - ' + x;

s.Base\_Daily\_Booking\_Rate\_\_c = Decimal.valueOf((x + 10) \* 1000);

s.Category\_\_c = Math.mod(x, 2) == 0

? 'Cooking Classroom'

: (Math.mod(x, 3) == 0 ? 'Café' : 'Exploration Lab');

s.Maximum\_Capacity\_\_c = (x + 1) \* 10;

s.Minimum\_Capacity\_\_c = x + 1;

s.Picture\_URL\_\_c = 'https://pics.net.com/' + x;

s.Type\_\_c = Math.mod(x, 2) == 0

? 'Conversation Starter;LargeGroups;Outdoor;Energizing;Scheduled Activity'

: (Math.mod(x, 3) == 0

? 'Individuals or Small Groups;Come-and-Go;Indoor;Relaxing;Quiet'

: 'Scheduled Activity;Come-and-Go;Indoor;Outdoor;LargeGroups;Relaxing;Conversation Starter');

spaces.add(s);

}

return spaces;

}

public static List<Reservation\_\_c>makeReservations(Integer resvNumber) {

List<Reservation\_\_c>resrvs = new List<Reservation\_\_c>();

for (Integer x = 0; x <resvNumber; x++) {

Reservation\_\_c r = new Reservation\_\_c();

r.Start\_Date\_\_c = Date.today();

r.End\_Date\_\_c = Date.today().addDays(x);

r.Total\_Number\_of\_Guests\_\_c = x + 10;

resrvs.add(r);

}

return resrvs;

}

}

**TestDataFactory.cls-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<ApexClassxmlns="urn:metadata.tooling.soap.sforce.com" fqn="testDataFactory">

<apiVersion>55.0</apiVersion>

<status>Active</status>

</ApexClass>

**LWC**

**customerDetailForm.html**

<template>

<template if:true={errorMsg}>

<c-error-panel

friendly-message={msgForUser}

errors={errorMsg}

></c-error-panel>

</template>

<div if:true={detailfields}>

<lightning-layout

multiple-rows="true"

class="slds-var-m-horizontal\_small">

<lightning-layout-item size="12">

<h2 class="slds-text-heading\_smallslds-var-p-bottom\_small">

Customer Info:

</h2>

<lightning-record-form

class="slds-var-m-vertical\_small"

record-id={recordid}

object-api-name={sobjecttype}

fields={detailfields}

mode="view"

columns="2"

onsuccess={handleSavedRecord}

></lightning-record-form>

</lightning-layout-item>

</lightning-layout>

<div class="slds-clearfixslds-var-p-top\_large">

<lightning-button

variant="brand"

label="Draft Reservation"

title="Draft"

onclick={handleDraft}

class="slds-var-m-left\_x-small"

></lightning-button>

</div>

</div>

<br />

</template>

**customerDetailForm.js**

import { LightningElement, api } from 'lwc';

import getCustomerFields from '@salesforce/apex/customerServices.getCustomerFields';

const STATE\_FIELD\_LABEL = 'State';

export default class CustomerDetailForm extends LightningElement {

@apisobjecttype;

@apirecordid;

detailfields;

errorMsg;

msgForUser;

connectedCallback() {

this.getDetailFields();

}getDetailFields() {

if (this.sobjecttype) {

getCustomerFields({ objectType: this.sobjecttype })

.then((result) => {

this.detailfields = Object.values(result);})

.catch((error) => {

this.errorMsg = error;

this.msgForUser =

'There was an issue loading customer data.';});}}

handleSavedRecord(event) {

let stateVal;

for (let value in event.detail.fields) {

if (value.includes(STATE\_FIELD\_LABEL)) {

stateVal = event.detail.fields[value].value;}}

constsaveEvent = new CustomEvent('customerupdate', {

detail: stateVal});

this.dispatchEvent(saveEvent);}

handleDraft() {

constdraftevt = new CustomEvent('draftreservation');

this.dispatchEvent(draftevt);}}

**customerDetailForm.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>false</isExposed>

</LightningComponentBundle>

**LWC**

**customerTile.html**

<template>

<div class="pointer" onclick={handleClick}>

<lightning-layout vertical-align="center">

<lightning-layout-item>

<lightning-icon

icon-name={icon}

alternative-text={alttext}

></lightning-icon>

</lightning-layout-item>

<lightning-layout-item padding="horizontal-small">

<a href={navRef} alt={alttext}>{customer.name}</a>

<p>Status: {customer.status}</p>

<p>Email: {customer.email}</p>

</lightning-layout-item>

</lightning-layout>

</div></template>

**customerTile.js**

import { LightningElement, api } from 'lwc';

import { NavigationMixin } from 'lightning/navigation';

export default class CustomerTile extends NavigationMixin(LightningElement) {

@api customer;

@api object;

navRef;

get icon() {

return 'standard:' + this.object.toLowerCase();}

get alttext() {

return (

'Navigate to ' +

this.object +

' record detail for ' +

this.customer.name);

}connectedCallback() {

this.customerRecordRef = {

type: 'standard\_\_recordPage',

attributes: {

recordId: this.customer.Id,

actionName: 'view'}};

this[NavigationMixin.GenerateUrl](this.customerRecordRef).then(

(url) => (this.navRef = url));}

handleClick() {

constclickevt = new CustomEvent('customerselect', {

detail: {

customerId: this.customer.Id,

sobjectType: this.object,

state: this.customer.state

}});

this.dispatchEvent(clickevt);}}

**customerTile.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>false</isExposed>

</LightningComponentBundle>

**AURA**

**reservationHelperAura.cmp**

<aura:component implements="flexipage:availableForAllPageTypes" access="global">

<lightning:navigationItemAPIaura:id="navigationItemAPI" />

<aura:attribute name="recordId" type="String" />

<aura:attribute name="error" type="String" />

<aura:attribute name="sobjectType" type="String" />

<aura:attribute name="state" type="String" />

<aura:attribute name="tabId" type="String" />

<c:reservationHelper

aura:id="reservHelperLWC"

oncustomerchoice="{!c.handleSelect}"

></c:reservationHelper>

<lightning:cardiconName="custom:custom18" title="Reservation Helper">

<aura:ifisTrue="{!v.recordId}">

<lightning:flow

aura:id="flowCmp"

onstatuschange="{!c.handleStatusChange}"/>

<aura:set attribute="else">

<div class="slds-card">

<h1 class="slds-var-p-left\_small">

Select a record to get started.</h1>

</div></aura:set></aura:if>

{!v.error}</lightning:card></aura:component>

**reservationHelperAura.cmp-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<AuraDefinitionBundle

xmlns="urn:metadata.tooling.soap.sforce.com"

fqn="reservationHelperAura">

<apiVersion>55.0</apiVersion>

<description>A Lightning Component Bundle</description>

</AuraDefinitionBundle>

**reservationHelperAuraController.js**

({

handleSelect: function (component, event, helper) {

var recId = event.getParam('customerId');

var objType = event.getParam('sobjectType');

var state = event.getParam('state');

component.set('v.recordId', recId);

component.set('v.sobjectType', objType);

component.set('v.state', state);

helper.launchFlow(component, event, helper);},

handleStatusChange: function (component, event, helper) {

if (event.getParam('status') === 'FINISHED') {

helper.refreshTab(component);}}

});

**reservationHelperAuraHelper.js**

({

launchFlow: function (component) {

var flow = component.find('flowCmp');

var rec = component.get('v.recordId');

var sobj = component.get('v.sobjectType');

var state = component.get('v.state');

var inputVariables = [

{ name: 'varRecordId', type: 'String', value: rec },

{ name: 'varSobjectType', type: 'String', value: sobj },

{ name: 'varState', type: 'String', value: state }];

if (flow) {

flow.startFlow('createReservation', inputVariables);}},

refreshTab: function (component) {

var lwcHelper = component.find('reservHelperLWC');

var sobj = component.get('v.sobjectType');

lwcHelper.handleFlowExit({ detail: sobj });

var navItemAPI = component.find('navigationItemAPI');

navItemAPI.refreshNavigationItem().catch(function (error) {

// eslint-disable-next-line no-console

console.error(error);

});

}});

**LWC**

**reservationHelper.js**

import { LightningElement, api, wire } from 'lwc';

import { ShowToastEvent } from 'lightning/platformShowToastEvent';

import TILE\_SELECTION\_MC from '@salesforce/messageChannel/Tile\_Selection\_\_c';

import FLOW\_STATUS\_CHANGE\_MC from '@salesforce/messageChannel/Flow\_Status\_Change\_\_c';

import {

subscribe,

unsubscribe,

MessageContext,

publish

} from 'lightning/messageService';

export default class ReservationHelper extends LightningElement {

/\*   Component coordinates event comms between Aura-based parent component

\*   and LWC-based siblings. Uses LMS as replacement for ltng:selectSObject event.

\*   TO DO: Replace Aura parent component when support for LWC flow screens available.\*/

flowStarted = false;

subscription = null;

@wire(MessageContext)

messageContext;

subscribeToMessageChannel() {

this.subscription = subscribe(

this.messageContext,

TILE\_SELECTION\_MC,

(message) =>this.handleCustomerSelect(message));

}

unsubscribeToMessageChannel() {

unsubscribe(this.subscription);

this.subscription = null;

}

connectedCallback() {

this.subscribeToMessageChannel();

}

disconnectedCallback() {

this.unsubscribeToMessageChannel();

}

handleCustomerSelect(message) {

if (message.tileType === 'customer') {

if (this.flowStarted) {

consttoastEvt = new ShowToastEvent({

title: 'Flow interview already in progress',

message:

'Finish the flow interview in progress before selecting another customer.',

variant: 'error'

});

this.dispatchEvent(toastEvt);

} else {

this.flowStarted = true;

constchoiceEvt = new CustomEvent('customerchoice', {

detail: message.properties});

this.dispatchEvent(choiceEvt);}}}

@api

handleFlowExit(event) {

const payload = {

flowName: 'createReservation',

status: 'FINISHED',

state: { sobjecttype: event.detail }};

publish(this.messageContext, FLOW\_STATUS\_CHANGE\_MC, payload);}

}

**reservationHelper.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>false</isExposed>

</LightningComponentBundle>

**LWC**

**reservationHelperForm.html**

<template>

<c-customer-detail-form

recordid={customerid}

sobjecttype={objecttype}

oncustomerupdate={handleCustomerUpdate}

ondraftreservation={handleDraftReservation}

></c-customer-detail-form>

</template>

**reservatioHelperForm.js**

import { LightningElement, api } from 'lwc';

import { FlowNavigationNextEvent } from 'lightning/flowSupport';

export default class reservationHelperForm extends LightningElement {

//Flow input variables

@apicustomerid;

@apiobjecttype;

currentstate;

@api

get state() {

return this.currentstate;}

set state(value) {

this.currentstate = value;

}

//Flow output variables

\_endDays = 7;

\_numberOfPeople = 20;

@api

get startDate() {

return this.\_startDate;

}

@api

get endDays() {

return this.\_endDays;

}

@api

get numberOfPeople() {

return this.\_numberOfPeople;

}

@api

get requestedMarket() {

return this.\_requestedMarket;

}

customerFields = [];

get detailsLoaded() {

return this.customerFields.length> 0 ? true : false;

}

handleDetails(event) {

this.customerFields = event.detail;

}

handleCustomerUpdate(event) {

if (this.currentstate !== event.detail) {

this.currentstate = event.detail;

}

}

handleDraftReservation() {

constnextNavigationEvent = new FlowNavigationNextEvent();

this.dispatchEvent(nextNavigationEvent);

}

}

**reservationHelperForm.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>true</isExposed>

<targets>

<target>lightning\_\_FlowScreen</target>

</targets>

<targetConfigs>

<targetConfig targets="lightning\_\_FlowScreen">

<property

name="customerid"

type="String"

label="Record Id"

role="inputOnly"/>

<property

name="objecttype"

type="String"

label="Sobject Type"

role="inputOnly"/>

<property

name="state"

type="String"

label="State"

role="inputOnly"/>

<property

name="endDays"

type="Integer"

label="Reservation length"

role="outputOnly"/>

<property

name="startDate"

type="Date"

label="Start date"

role="outputOnly"/>

<property

name="numberOfPeople"

type="Integer"

label="No. of guests"

role="outputOnly"/>

<property

name="requestedMarket"

type="String"

label="Market Id"

role="outputOnly"/>

</targetConfig>

</targetConfigs>

</LightningComponentBundle>

**LWC**

**reservationList.html**

<template>

<div class="slds-card">

<template if:true={errorMsg}>

<c-error-panel

friendly-message={msgForUser}

errors={errorMsg}

></c-error-panel>

</template>

<template if:false={errorMsg}>

<p class="slds-text-heading\_smallslds-var-m-around\_x-small">

Open Reservations:</p>

<template if:true={noRecords}>

<p class="slds-var-m-left\_small">No open reservations found.</p>

</template>

<template if:false={noRecords}>

<ul class="slds-list\_verticalslds-has-dividers\_bottom-space">

<template for:each={reservations} for:item="record">

<li class="slds-item" key={record.record.Id}>

<c-reservation-tile

reservation={record}

onreservationselect={handleSelectEvent}

></c-reservation-tile>

</li></template></ul></template>

</template></div></template>

**reservationList.js**

import { LightningElement, wire } from 'lwc';

import { refreshApex } from '@salesforce/apex';

import getOpenReservations from '@salesforce/apex/reservationManagerController.getOpenReservations';

import TILE\_SELECTION\_MC from '@salesforce/messageChannel/Tile\_Selection\_\_c';

import FLOW\_STATUS\_CHANGE\_MC from '@salesforce/messageChannel/Flow\_Status\_Change\_\_c';

import {

subscribe,

unsubscribe,

APPLICATION\_SCOPE,

MessageContext,

publish

} from 'lightning/messageService';

export default class ReservationList extends LightningElement {

wiredResult;

reservationSelected = false;

selectedRecId;

\_records = [];

reservations = [];

errorMsg;

msgForUser;

noRecords = false;

subscription = null;

@wire(MessageContext)

messageContext;

subscribeToMessageChannel() {

this.subscription = subscribe(

this.messageContext,

FLOW\_STATUS\_CHANGE\_MC,

(message) =>this.handleMessage(message),

{ scope: APPLICATION\_SCOPE });

}

unsubscribeToMessageChannel() {

unsubscribe(this.subscription);

this.subscription = null;

}

handleMessage(message) {

if (

message.flowName === 'spaceDesigner' &&

message.status === 'FINISHED') {

refreshApex(this.wiredResult);

}

}

connectedCallback() {

this.subscribeToMessageChannel();

}

disconnectedCallback() {

this.unsubscribeToMessageChannel();

}

@wire(getOpenReservations)

wiredReservations(value) {

this.wiredResult = value;

if (value.error) {

this.errorMsg = value.error;

this.msgForUser = 'There was an issue loading reservations.';

} else if (value.data) {

if (value.data.length) {

this.\_records = [...value.data];

this.handleMute();

if (this.noRecords) this.noRecords = false;

} else {

this.noRecords = true;}}}

handleMute() {

if (!this.reservationSelected) {

let muted = '';

this.reservations = this.\_records.map((record) => {

if (this.selectedRecId) {

this.reservationSelected = true;

muted = record.Id === this.selectedRecId ? false : true;

} else {

muted = false;

}

return { record, muted };});}}

handleSelectEvent(event) {

this.selectedRecId = event.detail.reservationId;

const payload = { tileType: 'reservation', properties: event.detail };

publish(this.messageContext, TILE\_SELECTION\_MC, payload);

this.handleMute();

}

}

**reservationList.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>true</isExposed>

<targets>

<target>lightning\_\_AppPage</target>

<target>lightning\_\_HomePage</target>

</targets>

</LightningComponentBundle>

**LWC**

**reservationTile.html**

<template>

<div onclick={handleTileClick} class={cssClass}>

<lightning-layout vertical-align="start">

<lightning-layout-item>

<lightning-icon

icon-name="custom:custom14"

alternative-text="Reservation icon"

></lightning-icon>

</lightning-layout-item>

<lightning-layout-item padding="horizontal-small">

<a href={navRef}>{reservation.record.Name}</a>

<p if:true={reservation.record.Contact\_\_c}>

Customer: {reservation.record.Contact\_\_r.Name}

</p><p if:true={reservation.record.Lead\_\_c}>

Customer: {reservation.record.Lead\_\_r.Name}</p>

<p>Status: {reservation.record.Status\_\_c}</p>

<p if:true={reservation.record.Market\_\_c}>

Market: {reservation.record.Market\_\_r.Name}</p><p>

Start Date:

<lightning-formatted-date-time

value={reservation.record.Start\_Date\_\_c}

></lightning-formatted-date-time></p><p>

End Date:

<lightning-formatted-date-time

value={reservation.record.End\_Date\_\_c}

></lightning-formatted-date-time></p><p>

Total Guests: {reservation.record.Total\_Number\_of\_Guests\_\_c}</p>

</lightning-layout-item>

</lightning-layout></div></template>

**reservationTile.js**

import { LightningElement, api } from 'lwc';

import { NavigationMixin } from 'lightning/navigation';

export default class ReservationTile extends NavigationMixin(LightningElement) {

@api reservation;

navRef;

get cssClass() {

return this.reservation.muted ? 'mute pointer' : 'pointer';}

connectedCallback() {

this.reservationRecordRef = {

type: 'standard\_\_recordPage',

attributes: {

recordId: this.reservation.record.Id,

actionName: 'view'}};

this[NavigationMixin.GenerateUrl](this.reservationRecordRef).then(

(url) => (this.navRef = url));}

handleTileClick() {

constclickevt = new CustomEvent('reservationselect', {

detail: {

reservationId: this.reservation.record.Id,

marketId: this.reservation.record.Market\_\_c,

customerName: this.reservation.record.Contact\_\_c

? this.reservation.record.Contact\_\_r.Name

: this.reservation.record.Lead\_\_r.Name}});

this.dispatchEvent(clickevt);}

}

**reservationTile.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>false</isExposed>

</LightningComponentBundle>

**APEX**

**MarketServices.cls**

public inherited sharing class MarketServices {

@auraEnabled(cacheable=true)

public static List<Space\_\_c>getRelatedSpaces(Id recordId) {

return [

SELECT

Id,

Base\_Daily\_Booking\_Rate\_\_c,

Name,

Market\_\_c,

Maximum\_Capacity\_\_c,

Minimum\_Capacity\_\_c,

Picture\_URL\_\_c,

Category\_\_c,

Type\_\_c

FROM Space\_\_c

WHERE Market\_\_c= :recordId

WITH SECURITY\_ENFORCED];}

}

**MarketServices.cls-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<ApexClassxmlns="urn:metadata.tooling.soap.sforce.com" fqn="marketServices">

<apiVersion>55.0</apiVersion>

<status>Active</status>

</ApexClass>

**AURA**

**spaceDesignerAura.cmp**

<aura:component implements="flexipage:availableForAllPageTypes" access="global">

<lightning:navigationItemAPIaura:id="navigationItemAPI" />

<aura:attribute name="reservation" type="String" />

<aura:attribute name="customer" type="String" />

<c:spaceDesigner

aura:id="spaceDesignerLWC"

onreservchoice="{!c.handleChoice}"

></c:spaceDesigner>

<aura:ifisTrue="{!v.reservation}">

<lightning:card

iconName="custom:custom50"

title="{! 'Designing Space for '+ v.customer}">

<lightning:flow

aura:id="flowCmp"

onstatuschange="{!c.handleReset}"/>

</lightning:card>

<aura:set attribute="else">

<lightning:cardiconName="custom:custom50" title="Space Designer">

<h1 class="slds-text-body\_regularslds-var-p-left\_small">

Select a record to get started.

</h1>

</lightning:card>

</aura:set></aura:if></aura:component>

**spaceDesignerAura.cmp-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<AuraDefinitionBundle

xmlns="urn:metadata.tooling.soap.sforce.com"

fqn="spaceDesignerAura">

<apiVersion>55.0</apiVersion>

<description>A Lightning Component Bundle</description>

</AuraDefinitionBundle>

**spaceDesignerAuraController.js**

({

handleChoice: function (component, event, helper) {

var recId = event.getParam('reservationId');

var market = event.getParam('marketId');

var customer = event.getParam('customerName');

component.set('v.reservation', recId);

component.set('v.customer', customer);

helper.startFlow(component, recId, market);},

handleReset: function (component, event) {

if (event.getParam('status') === 'FINISHED') {

var spaceDesignerLWC = component.find('spaceDesignerLWC');

spaceDesignerLWC.handleFlowExit();

var navigationItemAPI = component.find('navigationItemAPI');

navigationItemAPI.refreshNavigationItem().catch(function (error) {

// eslint-disable-next-line no-console

console.error(error);

});}}})

**spaceDesignerAuraHelper.js**

({

startFlow: function (component, recId, market) {

var flow = component.find('flowCmp');

var inputVariables = [

{ name: 'varReservId', type: 'String', value: recId },

{ name: 'varMarketId', type: 'String', value: market }];

if (flow) {

flow.startFlow('spaceDesigner', inputVariables);}}

});

**LWC**

**spaceDesinger.js**

import { LightningElement, api, wire } from 'lwc';

import { ShowToastEvent } from 'lightning/platformShowToastEvent';

import TILE\_SELECTION\_MC from '@salesforce/messageChannel/Tile\_Selection\_\_c';

import FLOW\_STATUS\_CHANGE\_MC from '@salesforce/messageChannel/Flow\_Status\_Change\_\_c';

import {

subscribe,

unsubscribe,

MessageContext,

publish

} from 'lightning/messageService';

export default class SpaceDesigner extends LightningElement {

/\*   Component coordinates event comms between Aura-based parent component

\*   and LWC-based siblings. Uses LMS in place of custom Aura application event.

\*   TO DO: Replace Aura parent component when support for LWC flow screens available.\*/

flowStarted = false;

subscription = null;

@wire(MessageContext)

messageContext;

subscribeToMessageChannel() {

this.subscription = subscribe(

this.messageContext,

TILE\_SELECTION\_MC,

(message) =>this.handleReservationSelect(message));

}

unsubscribeToMessageChannel() {

unsubscribe(this.subscription);

this.subscription = null;

}

connectedCallback() {

this.subscribeToMessageChannel();

}

disconnectedCallback() {

this.unsubscribeToMessageChannel();

}

handleReservationSelect(message) {

if (message.tileType === 'reservation') {

if (this.flowStarted) {

consttoastEvt = new ShowToastEvent({

title: 'Flow interview already in progress',

message:

'Finish the flow interview in progress before selecting another reservation.',

variant: 'error'});

this.dispatchEvent(toastEvt);

} else {

this.flowStarted = true;

constchoiceEvt = new CustomEvent('reservchoice', {

detail: message.properties});

this.dispatchEvent(choiceEvt);}}}

@api

handleFlowExit() {

const payload = { flowName: 'spaceDesigner', status: 'FINISHED' };

publish(this.messageContext, FLOW\_STATUS\_CHANGE\_MC, payload);}

}

**spaceDesigner.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>false</isExposed>

</LightningComponentBundle>

**LWC**

**spaceDesignForm.html**

<template>

<div class="slds-card">

<template if:true={errorMsg}>

<c-error-panel

friendly-message={msgForUser}

errors={errorMsg}

></c-error-panel>

</template>

<template if:false={errorMsg}>

<lightning-layout>

<lightning-layout-item size="3">

<div

class="slds-box slds-var-m-top\_xx-small slds-var-m-right\_x-small">

<h1

class="slds-text-heading\_smallslds-var-p-bottom\_xx-small">

Filter by Space Type:

</h1>

<c-pill-list

pills={pillvalues}

onfilterschange={handleFilters}

></c-pill-list>

</div><div

class="slds-box slds-var-m-top\_xx-small slds-var-m-right\_x-small">

<template if:true={selectedtile}>

<lightning-record-view-form

record-id={selectedtile}

object-api-name="Space\_\_c">

<div class="slds-grid slds-grid\_vertical">

<div class="slds-col">

<lightning-output-field

field-name="Name"

></lightning-output-field>

</div>

<div class="slds-col">

<lightning-output-field

field-name="Maximum\_Capacity\_\_c"

></lightning-output-field>

<lightning-output-field

field-name="Minimum\_Capacity\_\_c"

></lightning-output-field>

</div>

<div class="slds-col">

<lightning-output-field

field-name="Daily\_Booking\_Rate\_\_c"

></lightning-output-field>

</div>

</div>

</lightning-record-view-form>

</template>

<template if:false={selectedtile}>

<p>Select a space to see details.</p>

</template>

</div>

</lightning-layout-item>

<lightning-layout-item size="9">

<c-image-gallery

items={items}

onitemselect={handleItemSelect}

></c-image-gallery>

<template if:true={selectedtile}>

<div class="slds-var-p-vertical\_small">

<div>

<lightning-button

class="slds-var-p-left\_xx-small slds-var-p-right\_small"

variant="brand"

label="Add Space &amp; Finish"

onclick={handleSimpleAdd}

></lightning-button>

<lightning-button

variant="brand"

label="Add &amp; Go To Reservation"

onclick={handleAddWithNav}

></lightning-button>

</div>

</div>

</template>

<template if:false={selectedtile}>

<div class="slds-var-p-vertical\_small">

<lightning-button

disabled

class="slds-var-p-left\_xx-small slds-var-p-right\_small"

variant="neutral"

label="Add Space &amp; Finish"

></lightning-button>

<lightning-button

disabled

variant="neutral"

label="Add &amp; Go To Reservation"

></lightning-button>

</div>

</template>

</lightning-layout-item>

</lightning-layout>

</template>

</div>

</template>

**spaceDesignForm.js**

import { LightningElement, api, wire } from 'lwc';

import getRelatedSpaces from '@salesforce/apex/marketServices.getRelatedSpaces';

import { FlowNavigationNextEvent } from 'lightning/flowSupport';

export default class SpaceDesignForm extends LightningElement {

pillvalues = [

'Scheduled Activity',

'Come-and-Go',

'Large Groups',

'Individuals or Small Groups',

'Indoor',

'Outdoor',

'Conversation Starter',

'Energizing',

'Quiet',

'Relaxing'];

items = [];

errorMsg;

msgForUser;

\_records = [];

\_filters = [];

\_selectedtile;

//Flow Input Variables

@api market;

//Flow Output Variables

\_popTabOnFinish = false;

@api

get selectedtile() {

return this.\_selectedtile;

}

@api

get popTabOnFinish() {

return this.\_popTabOnFinish;

}

@wire(getRelatedSpaces, { recordId: '$market' })

wiredSpaces({ error, data }) {

if (error) {

this.errorMsg = error;

this.msgForUser = 'There was an issue loading related market data.';

this.\_records = [];

} else if (data) {

this.errorMsg = undefined;

this.msgForUser = undefined;

this.\_records = data;

this.filterItems();}

}

filterItems() {

this.items = this.\_records.map((record) => {

const types = record.Type\_\_c.split(';');

const muted = this.\_filters.some(

(filter) => !types.includes(filter));

return { record, muted };});

}

handleFilters(event) {

this.\_filters = [...event.detail.filters];

this.filterItems();

}

handleItemSelect(event) {

this.\_selectedtile = event.detail.recordId;

}

handleSimpleAdd() {

constnextNavigationEvent = new FlowNavigationNextEvent();

this.dispatchEvent(nextNavigationEvent);

}

handleAddWithNav() {

this.\_popTabOnFinish = true;

constnextNavigationEvent = new FlowNavigationNextEvent();

this.dispatchEvent(nextNavigationEvent);}}

**spaceDesignForm.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>true</isExposed>

<targets>

<target>lightning\_\_FlowScreen</target>

</targets>

<targetConfigs>

<targetConfig targets="lightning\_\_FlowScreen">

<property

name="market"

type="String"

label="Market Record Id"

role="inputOnly"/>

<property

name="popTabOnFinish"

type="Boolean"

label="Go to record?"

role="outputOnly"/>

<propertyname="selectedtile"

type="String"

label="Space Record Id"

role="outputOnly"/>

</targetConfig>

</targetConfigs></LightningComponentBundle>

**LWC**

**relatedSpaces.html**

<template>

<template if:true={records}>

<c-image-gallery

items={records}

onitemselect={handleItemSelect}

></c-image-gallery>

</template>

<template if:true={noRecords}>

<p>No related spaces found.</p>

</template>

<template if:true={errorMsg}>

<c-error-panel

friendly-message={msgForUser}

errors={errorMsg}

></c-error-panel>

</template>

</template>

**relatedSpaces.js**

import { LightningElement, api, wire } from 'lwc';

import getRelatedSpaces from '@salesforce/apex/marketServices.getRelatedSpaces';

import { NavigationMixin } from 'lightning/navigation';

export default class RelatedSpaces extends NavigationMixin(LightningElement) {

@apirecordId;

records;

errorMsg;

msgForUser;

noRecords = false;

@wire(getRelatedSpaces, { recordId: '$recordId' })

wiredSpaces({ error, data }) {

if (error) {

this.errorMsg = error;

this.msgForUser = 'There was an issue loading related market data.';

} else if (data) {

if (data.length) {

this.records = data.map((record) => {

return { record, muted: false };});

this.noRecords = false;

} else {

this.noRecords = true;}}}

handleItemSelect(event) {

event.stopPropagation();

if (event.detail.recordId) {

this[NavigationMixin.Navigate]({

type: 'standard\_\_recordPage',

attributes: {

recordId: event.detail.recordId,

objectApiName: 'Space\_\_c',

actionName: 'view'}});

}}}

**realtedSpaces.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>55.0</apiVersion>

<isExposed>true</isExposed>

<targets>

<target>lightning\_\_RecordPage</target>

</targets>

</LightningComponentBundle>

**LWC**

**imageGallery.html**

<template>

<lightning-layout multiple-rows="true">

<template for:each={items} for:item="item">

<div

key={item.record.Id}

class="slds-size\_1-of-4 slds-var-p-around\_xx-small">

<c-image-tile

record={item.record}

muted={item.muted}

selected={item.selected}

onclick={handleClick}

></c-image-tile>

</div>

</template>

</lightning-layout>

</template>

**imageGallery.js**

import { LightningElement, api } from 'lwc';

export default class ImageGallery extends LightningElement {

\_items = [];

@api

set items(values) {

if (Array.isArray(values)) {

this.\_items = values.map((value) => {

return {

record: value.record,

muted: value.muted,

selected: false};

});

} else {

this.\_items = [];}

}

get items() {

return this.\_items;

}

handleClick(event) {

const id = event.target.record.Id;

this.\_items = this.\_items.map((item) => {

if (item.record.Id === id) {

return Object.assign({}, item, { selected: !item.selected });

} else if (item.selected) {

return Object.assign({}, item, { selected: false });

}

return item;

});

this.dispatchEvent(

new CustomEvent('itemselect', {

detail: { recordId: id }}));}}

**imageGallery.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundle

xmlns="http://soap.sforce.com/2006/04/metadata"

fqn="imageGallery">

<apiVersion>55.0</apiVersion>

<isExposed>false</isExposed>

</LightningComponentBundle>

**LWC**

**imageTile.html**

<template>

<div class={cssClass}>

<imgsrc={record.Picture\_URL\_\_c} alt={record.Name} />

<div class="slds-is-absolute slds-box overlay">

<h2 class="slds-text-align\_left text">{record.Name}</h2>

</div>

</div>

</template>

**imageTile.css**

.es-tile {

position: relative;

background-color: #e6e6e6;}

.es-tile:hover {

opacity: 0.75;}

.selected {

background-color: #8c8c67;}

.muted {

opacity: 0.45;}

.overlay {

top: 65%;

left: 1.4%;

right: 1.3%;

bottom: 1.9%;

background-color: #e6e6e6;

opacity: 0.9;}

.text {

margin-top: -0.85rem;

margin-left: -0.5rem;

font-size: 0.85rem;

color: #393939;}

**imageTile.js**

import { LightningElement, api } from 'lwc';

export default class ImageTile extends LightningElement {

@api record;

@api muted;

@api selected;

get cssClass() {

return (

'slds-box slds-box\_xx-small es-tile' +

(this.muted ? ' muted' : '') +

(this.selected ? ' selected' : ''));

}

**imageTile.js-meta.xml**

<?xml version="1.0" encoding="UTF-8" ?>

<LightningComponentBundle

xmlns="http://soap.sforce.com/2006/04/metadata"

fqn="imageTile">

<apiVersion>55.0</apiVersion>

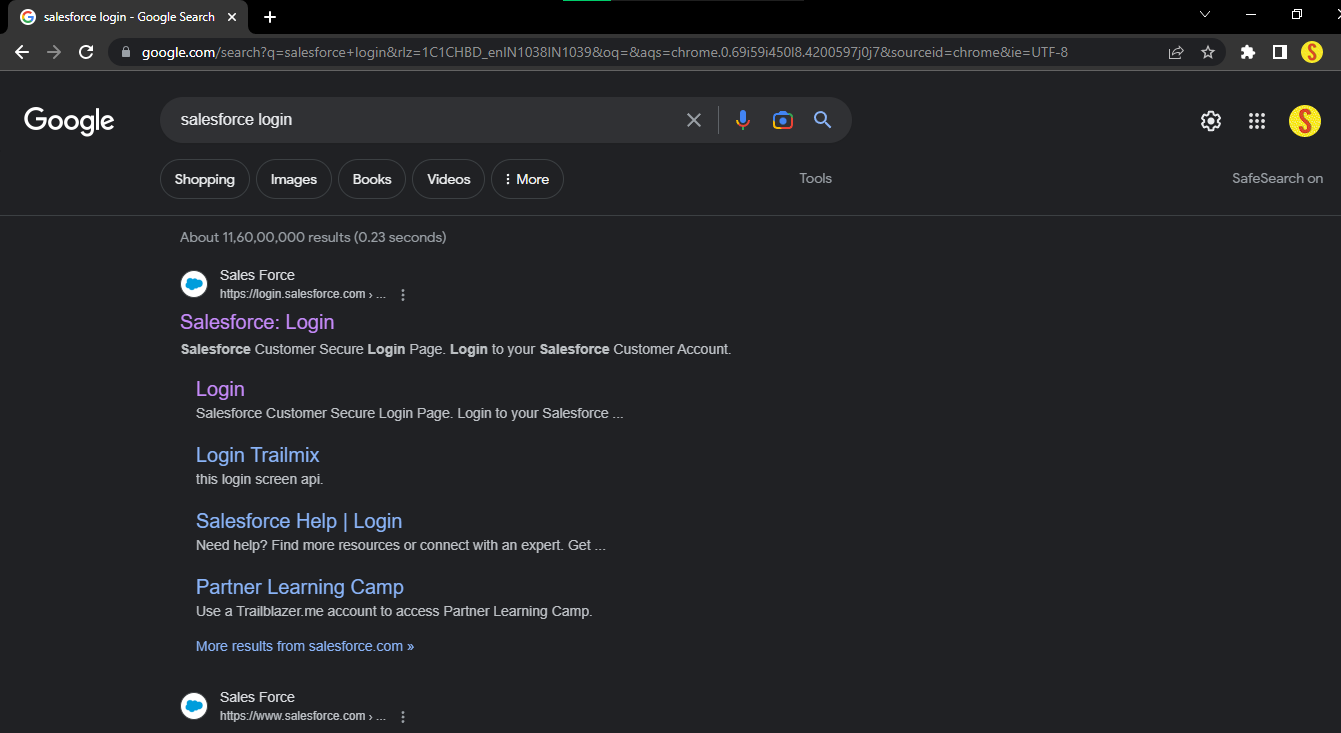
<isExposed>false</isExposed>

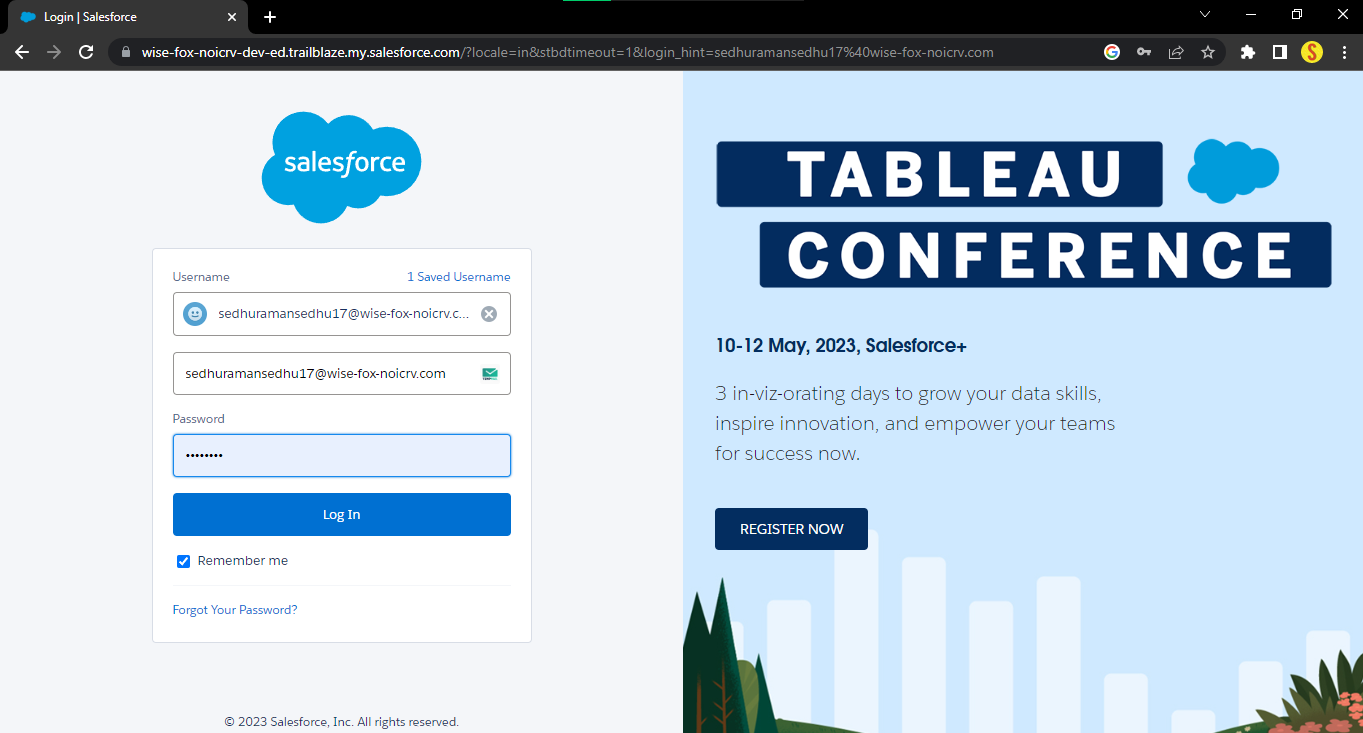
</LightningComponentBundle>

**ANNUEXURES-B**

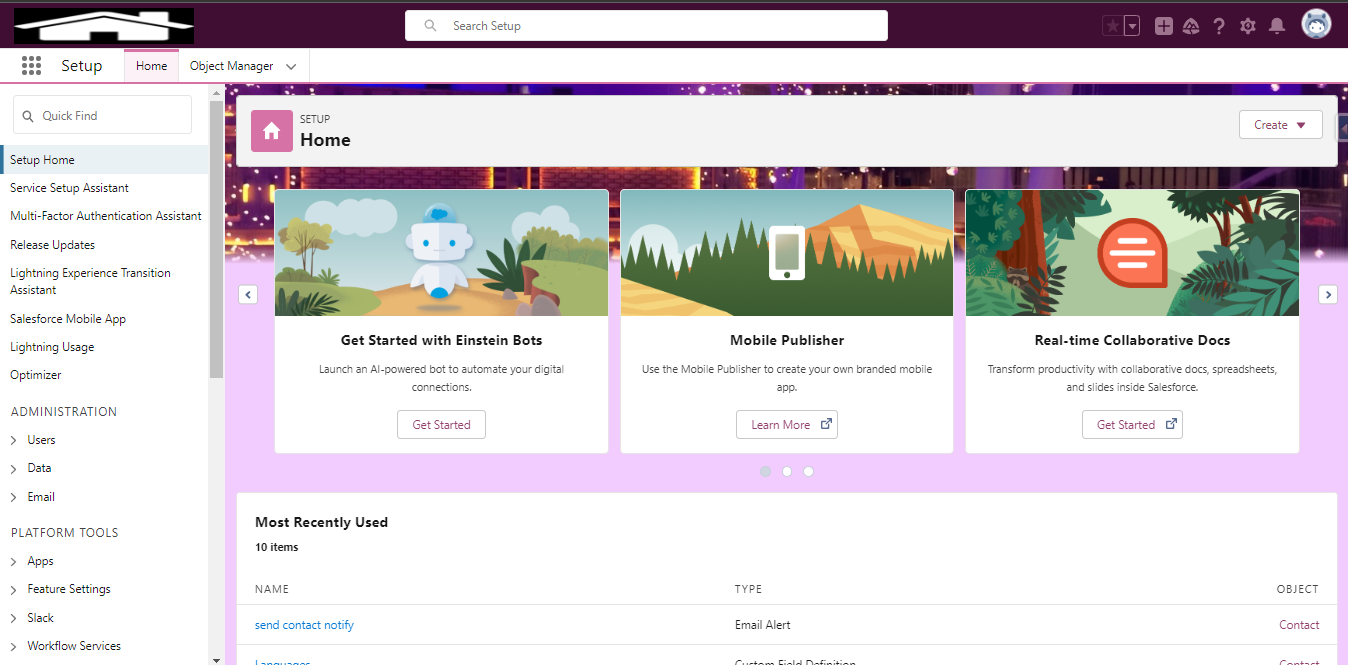
**OUTPUT**

**Salesforce Login**

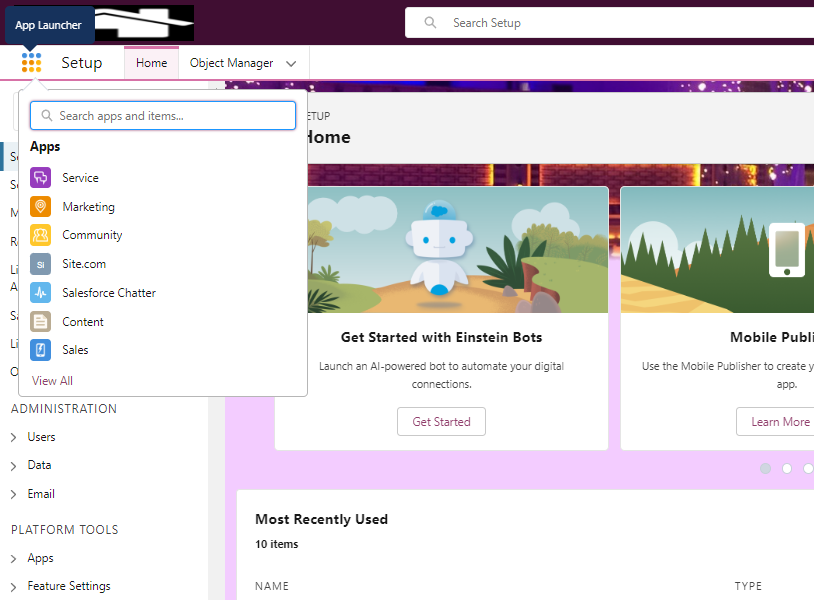




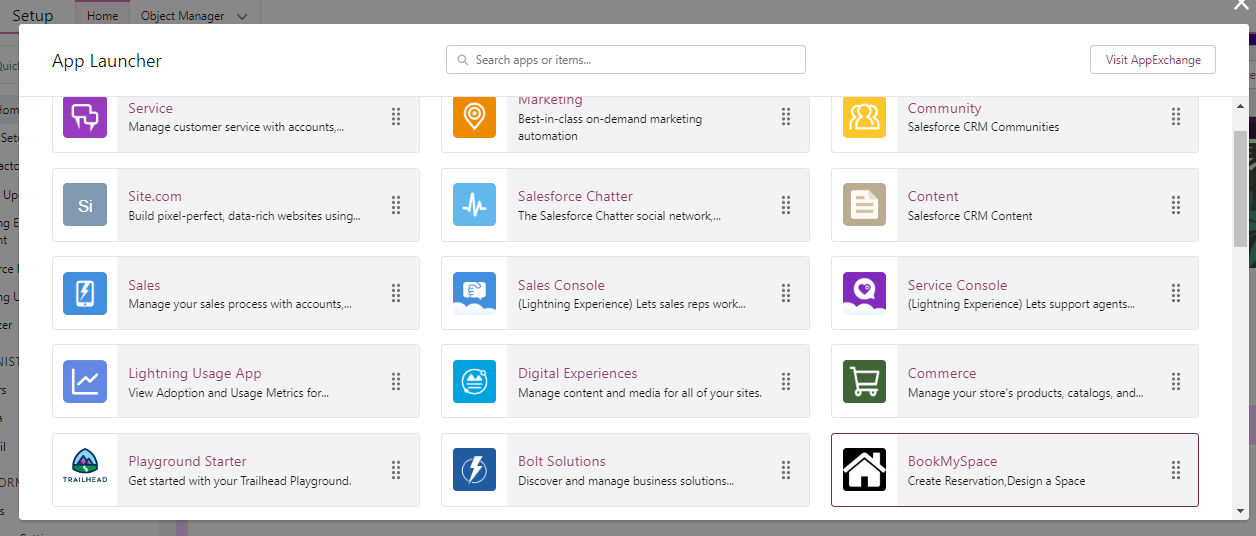
**Salesforce Home**



**Salesforce App Launcher**



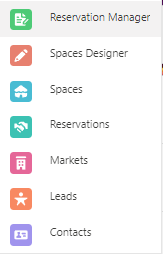
**Search BookMySpace App**



**BookMySpace Home Page**



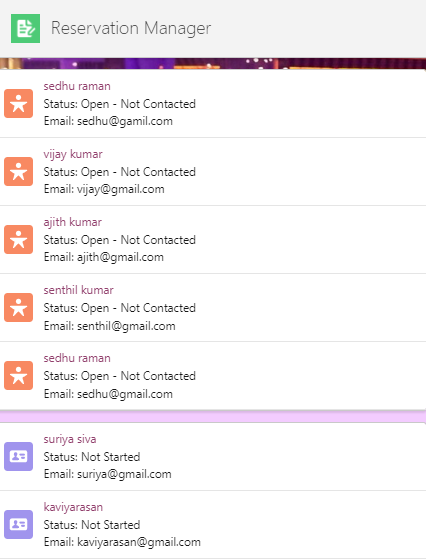
**BookMySpaceApp Tabs**



**Reservation Manager Tab**



**Customer List**



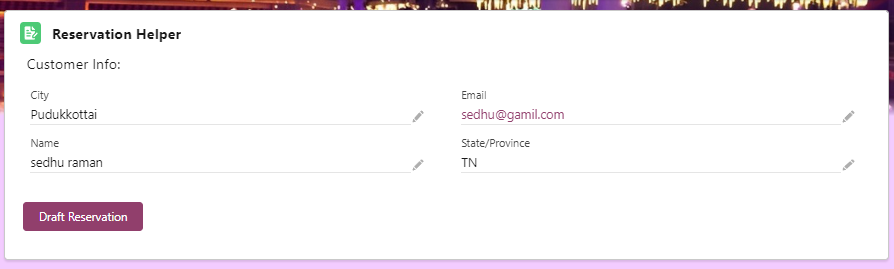
**OUTPUT FOR :customerList ( LWC )**

**Reservation Helper**



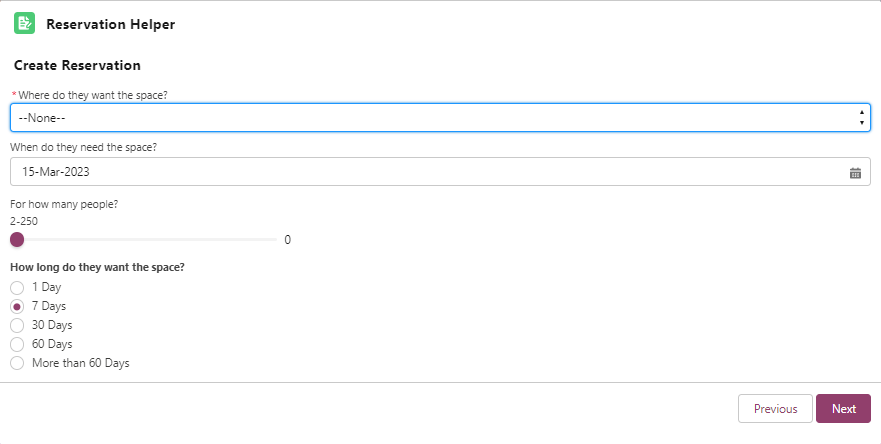
**OUTPUT FOR :reservationHelper (Aura)**

**Customer Detail Form**

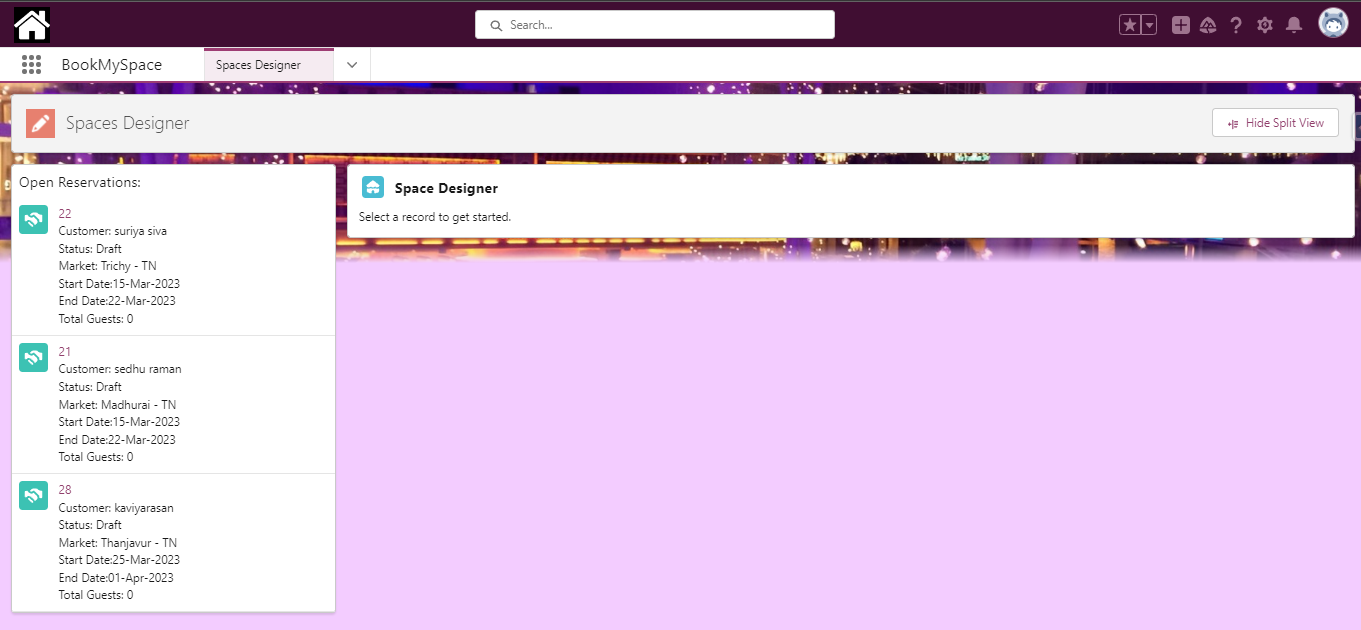


**OUTPUT FOR :customerDetailForm (LWC)**

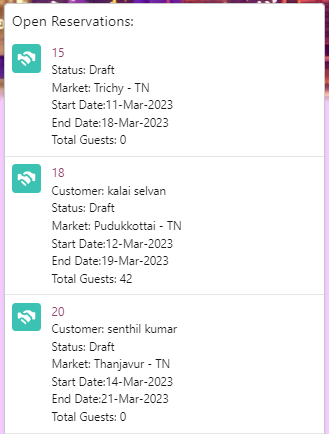
**Reservation Helper Form**

**OUTPUT FOR :reservationHelperForm ( LWC )**

**Space Designer Tab**



**Reservation List**

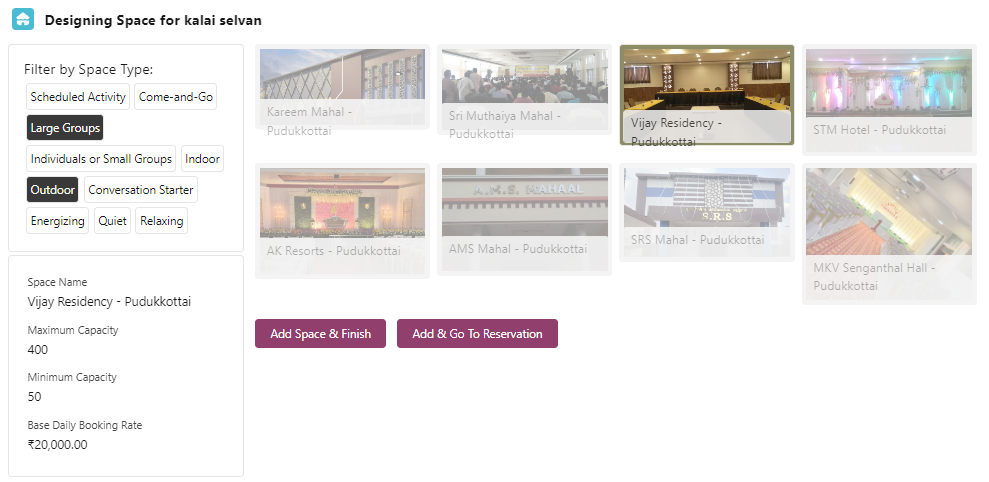


**OUTPUT FOR :reservationList ( LWC )**

**Spage Designer**

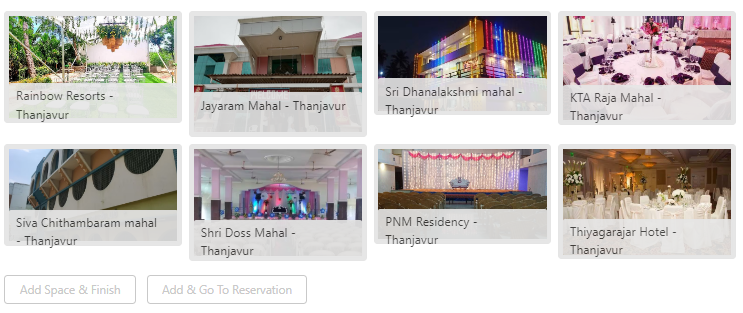


**OUTPUT FOR :spaceDesingerAura (Aura)**

**Space Design Form**

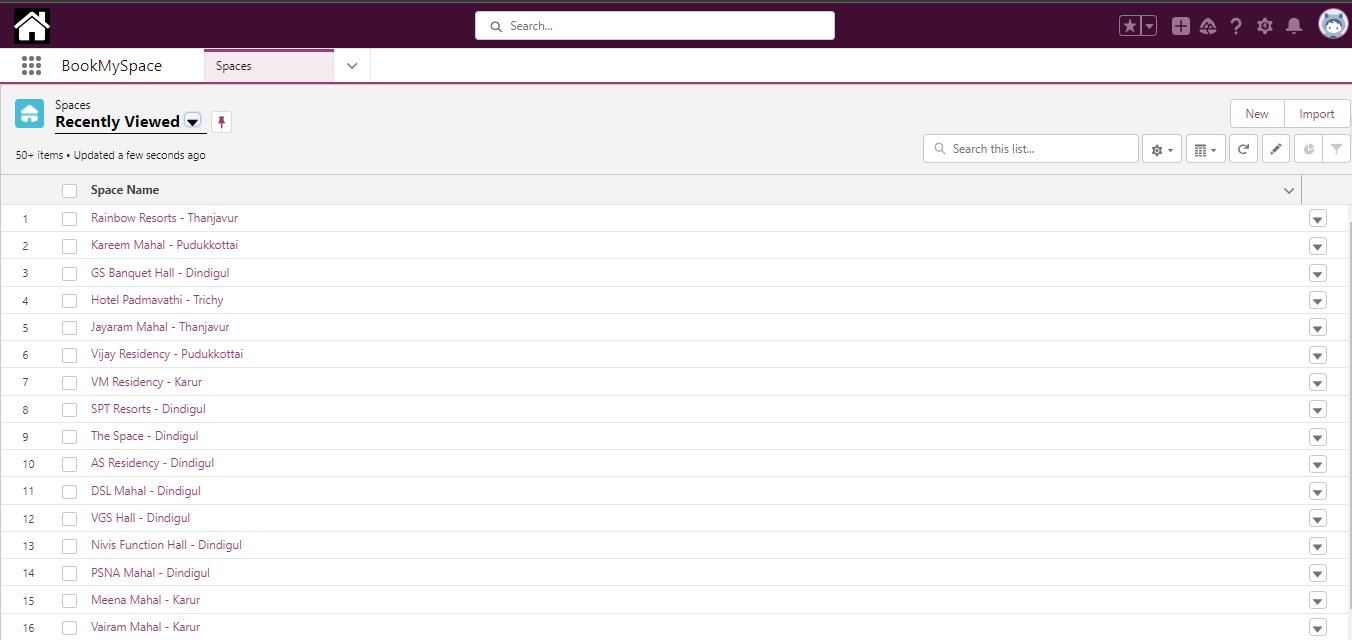
**OUTPUT FOR :spaceDesignForm (LWC)**

**Image Gallery page**

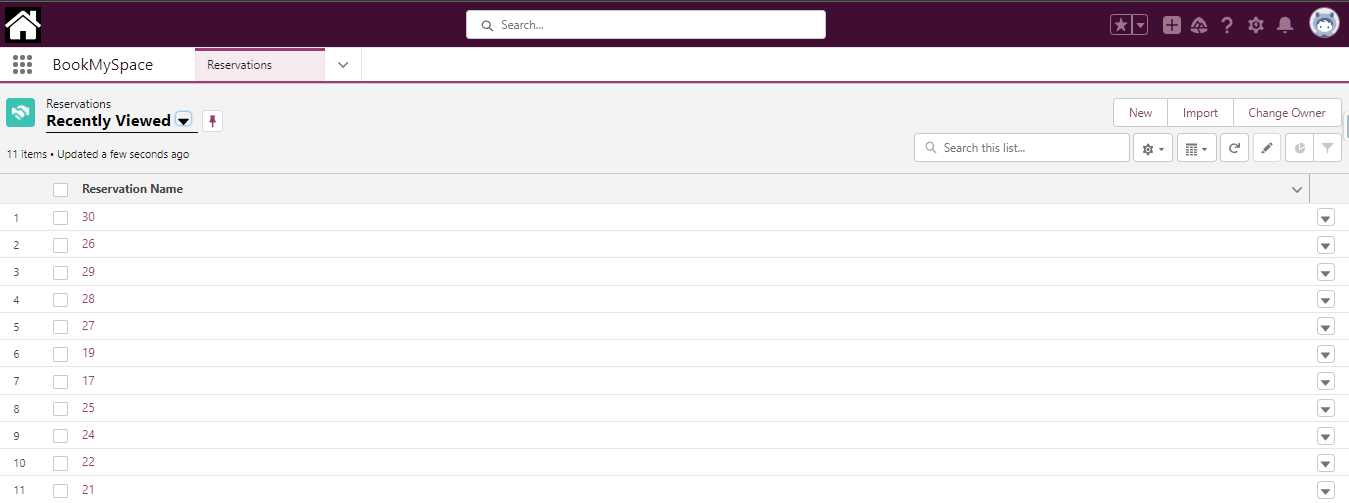


**OUTPUT FOR :imageGallery ( LWC )**

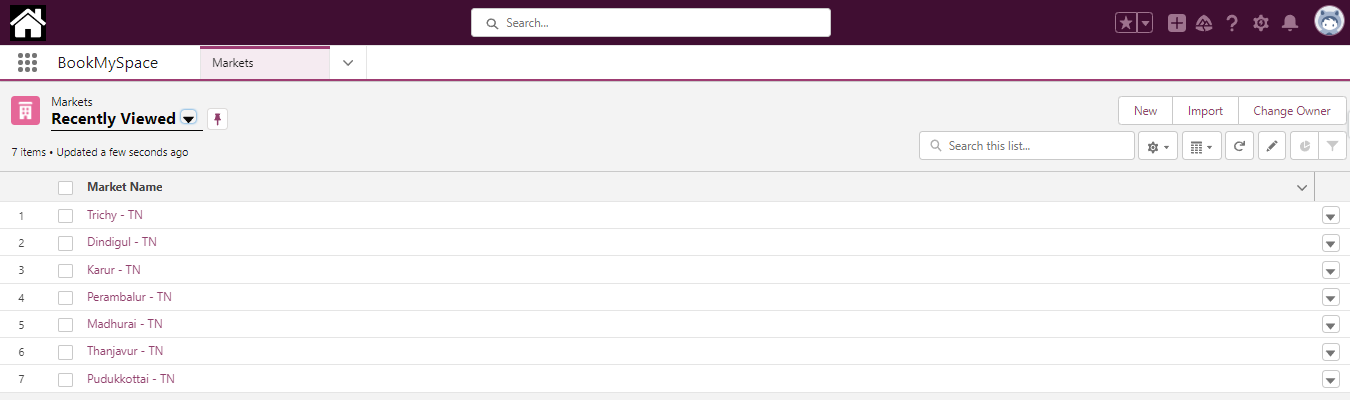
**Spaces Tab**



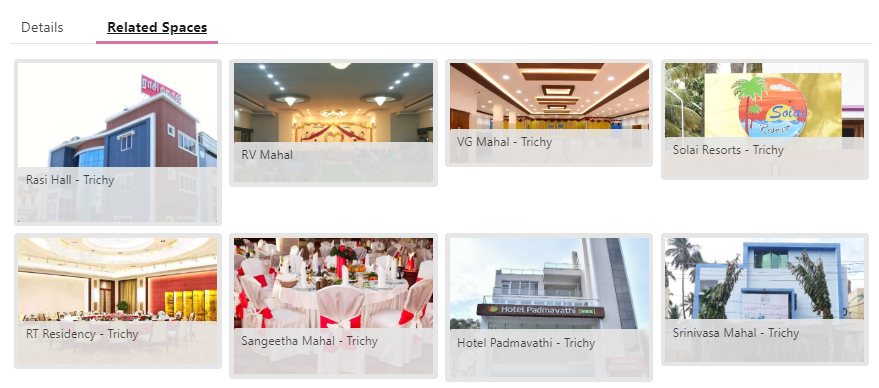
**Reservation Tab**



**Markets Tab**

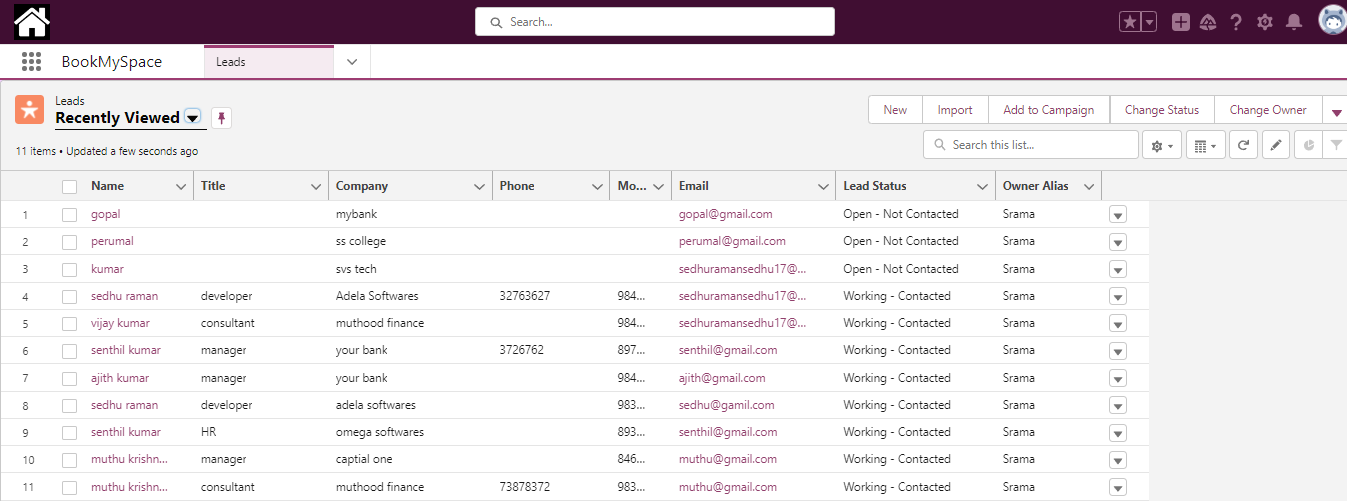


**Related Spaces (Market Record page)**

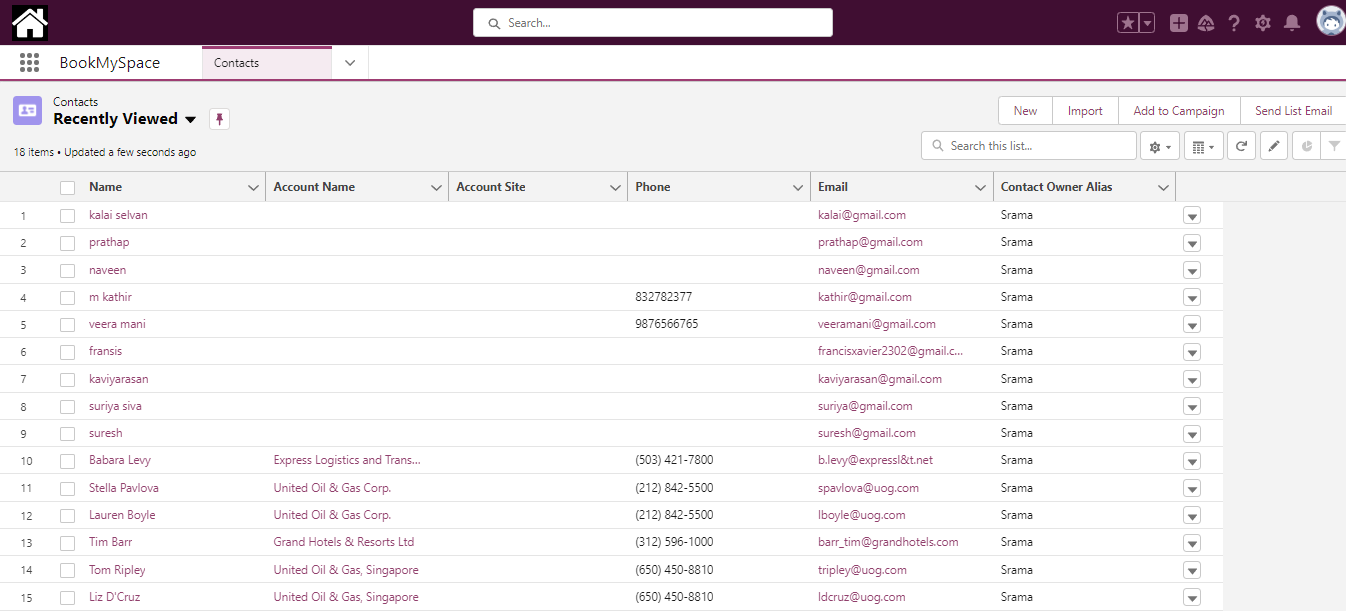


**OUTPUT FOR :realtedSpaces ( LWC )**

**Lead tab**



**Contact Tab**



**BIBLIOGRAPHY**

**BOOK REFERENCES:**

* Kao, Liz, and Jon Paz. Salesforce for Dummies. Wiley, 2020.
* Goodey, Paul. Salesforce CRM - The Definitive Admin Handbook. Packt Publishing, 2017.
* Shrivastava, Mohith. Salesforce Essentials for Administrators. Packt Publishing, 2018.
* Taber, David. Salesforce.com Secrets of Success: Best Practices for Growth and Profitability. Prentice Hall, 2009.
* Gupta, Rakesh. Mastering Salesforce CRM Administration. Packt Publishing, 2018.
* Fawcett, Andrew. Salesforce Lightning Platform Enterprise Architecture: Architect and deliver enterprise-level solutions. Packt Publishing, 2018.

**WEBSITE REFERENCES:**

* [**https://trailhead.salesforce.com/**](https://trailhead.salesforce.com/)
* [**https://developer.salesforce.com/**](https://developer.salesforce.com/)
* [**https://www.w3schools.com/**](https://www.w3schools.com/)
* [**https://www.tutorialspoint.com/**](https://www.tutorialspoint.com/)
* [**https://www.udemy.com/**](https://www.udemy.com/)
* [**https://www.salesforce.com/**](https://www.salesforce.com/)