

Building Cloud Native Applications with Ballerina and Choreo

Training Objective:

In this training, you will:

1. Build an HTTP service(API) using [Ballerina](#).
2. Use [Choreo](#) to deploy the cloud-native application.
 - a. Deploy a Ballerina service.
 - b. Deploy a web application.

High-Level Steps:

1. Develop the HTTP service using the Ballerina programming language.
2. Push the code to your GitHub account.
3. Deploy the cloud-native application on Choreo.
 - a. Deploy the Ballerina HTTP service
 - b. Deploy the ReactJS web application

Prerequisites:

1. A GitHub Account.
2. Git installed in your workstation.
3. A recent version of Google Chrome, Mozilla Firefox.
4. [Ballerina v 2201.8.4](#) installed in your workstation.
5. Microsoft Visual Studio (VSCode) and [WSO2 Ballerina plugin](#)
6. [PostMan](#) and [curl](#) (or any HTTP client) installed in your workstation.
7. [Choreo](#) Account

Business Scenario:

Build a reservation system for a Luxury Hotel.

Solution

Build a web application that enables users to book rooms. This application will offer the following features:

Room Search:

- Users can search for rooms by specifying check-in and check-out dates, with an option to filter by number of guests.
- Search results will showcase a list of room types (eg: single, double etc).
- Each room listing will feature a "Reserve" button for easy booking.

Room Reservation:

- To reserve a room, users need to input personal information: full name, contact number, and email.
- The form validates and activates the "Reserve" button once all fields are properly filled. Upon reservation, a unique reference number is displayed for the user to copy.

List My Reservations:

- Users can look up their reservations once they are logged in.
- Each entry in the list will provide options to either update the reservation details or cancel the reservation.

Update Reservations:

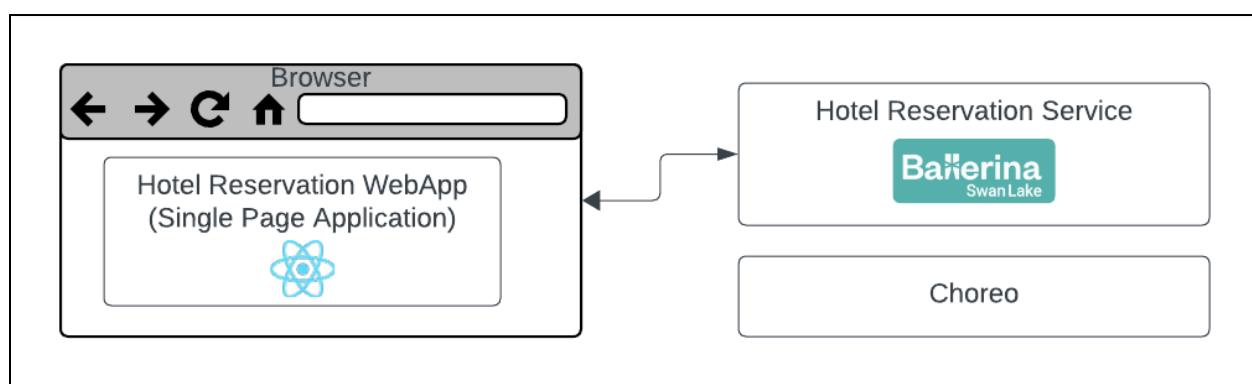
- Users have the flexibility to modify any part of their reservation.

Cancel Reservations:

- Users can cancel their reservations at any time, with a straightforward option to cancel their booking.

Design

The Hotel Reservation System will include a Hotel Reservation Service developed using Ballerina, and a Hotel Reservation Web Application developed using ReactJS.



Detailed Steps:

Developing the Hotel Reservation Service using Ballerina

The Hotel Reservation Service consists of a Ballerina HTTP service with 5 resources. A dedicated Ballerina project for the hotel reservation service has been created, complete with the required types and utility functions. During this session, we will focus on creating a Ballerina service and complete the implementation of the resources.

Base URL

<http://localhost:9090/reservations>

Resources

Resource	Path	Action	Query Param	Path Param	Request	Response
Get All available room types	/roomTypes/	GET	string checkinDate string checkoutDate int guestCapacity			[{ "id": 0, "name": "Single", "guestCapacity": 1, "price": 80 }, { "id": 0, "name": "Double", "guestCapacity": 2, "price": 100 }]

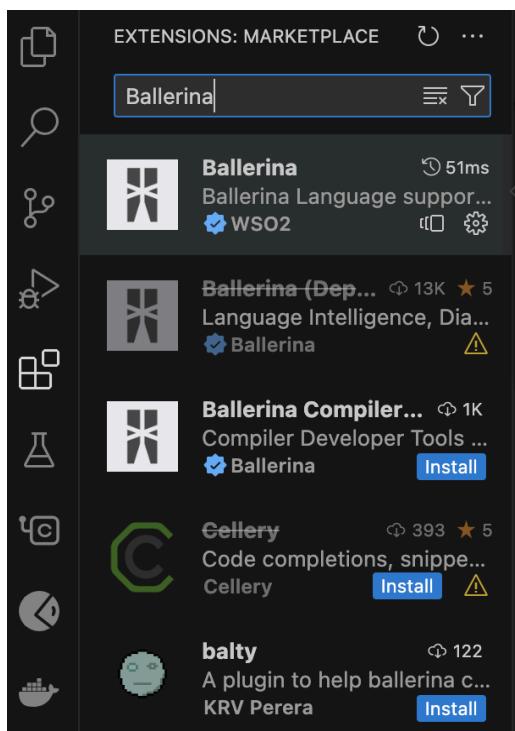
]
Create a new reservation		POST			{ "checkinDate": "2024-02-19T14:00:00Z", "checkoutDate": "2024-02-20T10:00:00Z", "rate": 100, "user": { "id": "123", "name": "waruna", "email": "waruna@someemail.com" }, "mobileNumber": "987", "roomType": "Family" }	{ "id": "1", "checkinDate": "2024-02-19T14:00:00Z", "checkoutDate": "2024-02-20T10:00:00Z", "user": { "id": "123", "name": "waruna", "email": "waruna@someemail.com" }, "mobileNumber": "987", "room": { "number": 201, "type": { "id": 0, "name": "Double", "guestCapacity": 2, "price": 100 } } }
Update an existing reservation		PUT		reservation_id	{ "checkinDate": "2024-02-20T14:00:00Z", "checkoutDate": "2024-02-21T10:00:00Z" }	{ "id": "1", "checkinDate": "2024-02-19T14:00:00Z", "checkoutDate": "2024-02-21T10:00:00Z", "user": { "id": "123", "name": "waruna", "email": "waruna@someemail.com" }, "mobileNumber": "987", "room": { "number": 201, "type": { "id": 0, "name": "Double", "guestCapacity": 2, "price": 100 } } }
Remove a reservation		DELETE		reservation_id		

Get all reservations for a given user id	/users/	GET		userID	[{ "checkinDate": "2024-02-19T14:00:00Z", "checkoutDate": "2024-02-20T10:00:00Z", "rate": 120, "user": { "id": "123", "name": "waruna", "email": "waruna@someemail.com" }, "mobileNumber": "987", "roomType": "Family" }, { "checkinDate": "2024-02-23T14:00:00Z", "checkoutDate": "2024-02-24T10:00:00Z", "rate": 100, "user": { "id": "123", "name": "waruna", "email": "waruna@someemail.com" }, "mobileNumber": "987", "roomType": "Double" }]
--	---------	-----	--	--------	---

Table 1: Reservation Service Table

Step 1: Create data types

1. Download and install Ballerina from the [download page](#).
2. Install [Ballerina Extension](#) from Visual Studio Code



- Fork the GitHub Repo - <https://github.com/ballerina-guides/hotel-reservation-demo>. Make sure you **untick** the option “Copy the main branch only”.

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks](#).

Required fields are marked with an asterisk (*).

Owner * <input type="button" value="Choose an owner"/>	Repository name * <input type="text" value="hotel-reservation-demo"/>
--	---

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Copy the `main` branch only
 Contribute back to ballerina-guides/hotel-reservation-demo by adding your own branch. [Learn more](#).

Create fork

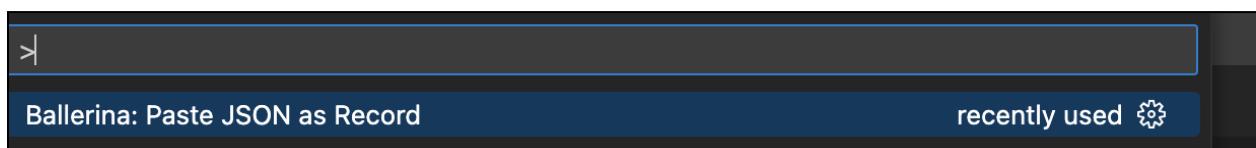
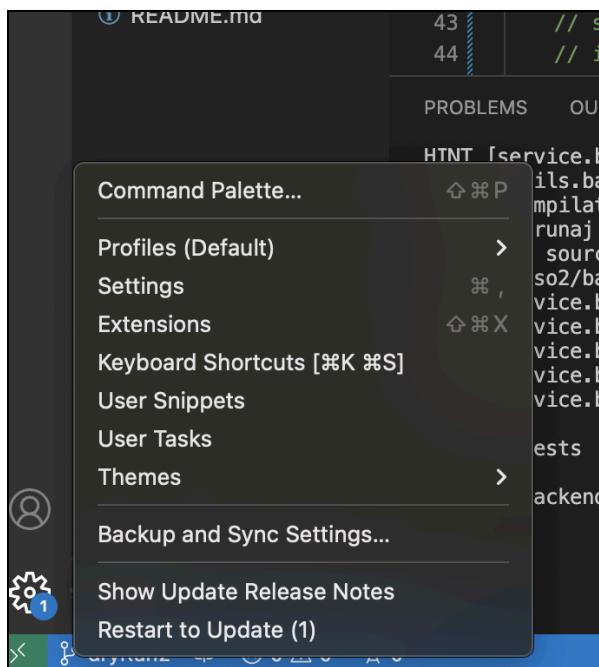
- Clone the GitHub Repo https://github.com/<your_username>/ballerina-guides/hotel-reservation-demo using the following command.

```
Unset
git clone
https://github.com/<your_username>/ballerina-guides/hotel-reserva
tion-demo
```

- Open the hotel-reservation-demo directory using Visual Studio Code.
- Go to the service directory and navigate to the types.bal file.
- We need to generate the following two record types:
ReservationRequest
- To generate the **ReservationRequest** type, copy the following JSON.

```
Unset
{
    "checkinDate": "2024-02-19T14:00:00Z",
    "checkoutDate": "2024-02-20T10:00:00Z",
    "rate": 100,
    "user": {
        "id": "123",
        "name": "waruna",
        "email": "waruna@someemail.com",
        "mobileNumber": "987"
    },
    "roomType": "Family"
}
```

Use the **Ballerina: Paste JSON as Record** option from the Command Palette.



9. Rename the record as Reservation Request.

```
Document this
public type ReservationRequest record {
    string checkinDate;
    string checkoutDate;
    string roomType;
    User user;
    decimal rate;
};
```

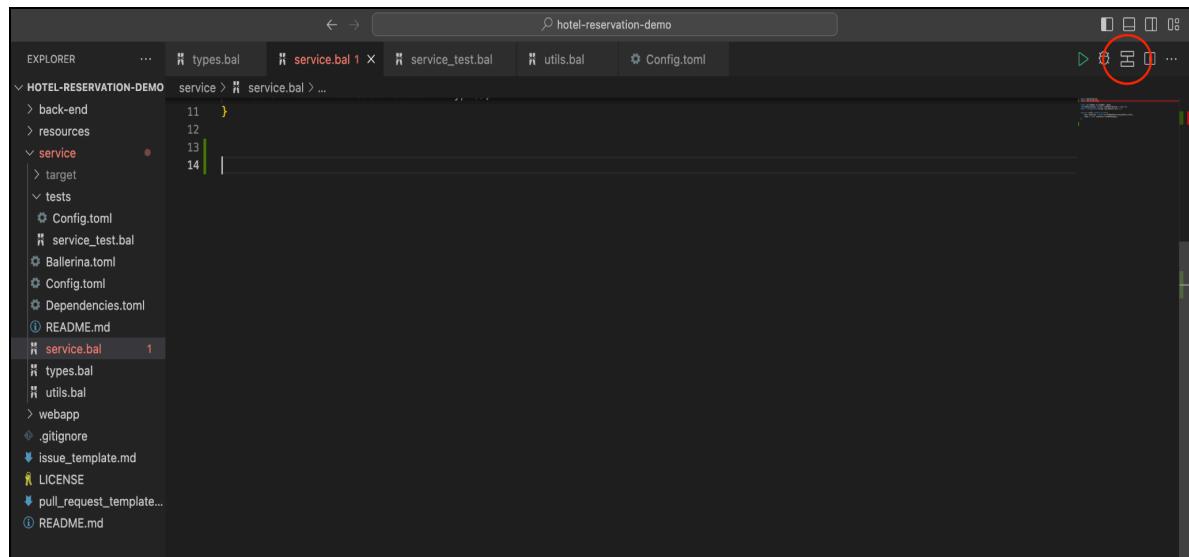
Note: *User record is already added to types.bal since it is required for the **Reservation** record type. You can remove the duplicate **User** record.*

Step 2: Create the HTTP service component

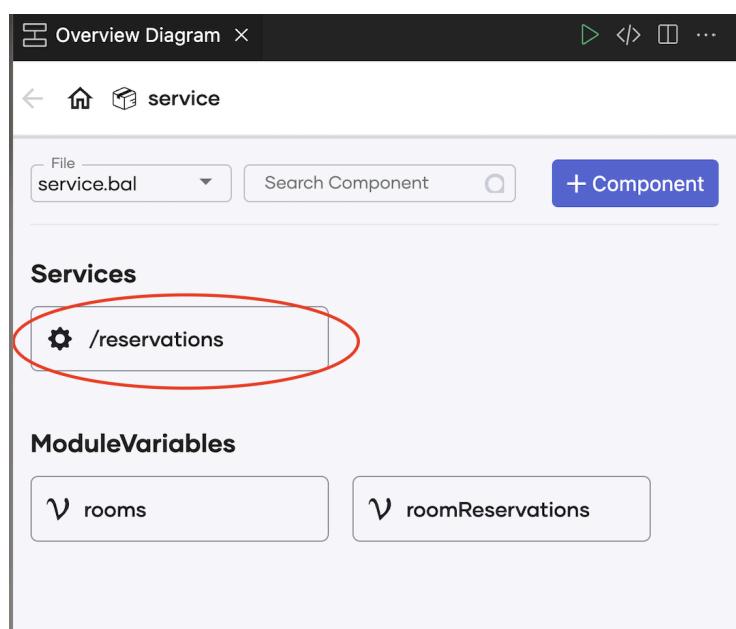
1. Add an HTTP service component to implement the hotel reservation API. Refer Table 1 for the required values for each resource.
 - a. Get available room types
 - b. Create a reservation
 - c. Update the reservation
 - d. Get user reservations
 - e. Delete the reservation

You can use the VScode UI to add the HTTP services and resources.

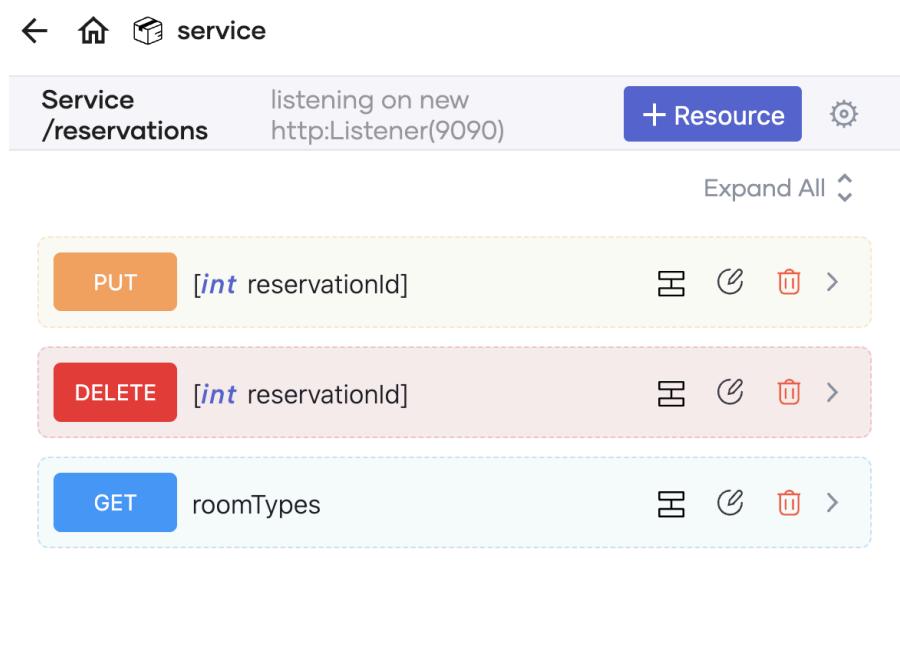
2. Click on the **Show Diagram** icon.



3. Click on /reservations service under **Services**.



4. Click on **+Resource**.



5. Configure the resource as below.

Configure Resource XHTTP Method* Resource Path Optional

POST

.

[+ Add Path Param](#)

Parameters

[+ Add Parameter](#)

PAYLOAD

NewReservationRequest payload

[Advanced Parameters](#) [Show](#)

Http Method : POST

Path : .

Then add payload and select NewReservationRequest.

In Response section create a new error response as below.

Responses

Select Code*

Type Optional

404 - Not Found

string

 Define a name record for the return type

NewReservationError

[Cancel](#)[Save](#)

Add more response types as below.

HTTP Method* Resource Path Optional

POST . [+ Add Path Param](#)

Parameters

[+ Add Parameter](#)

PAYLOAD	NewReservationRequest payload	
---------	-------------------------------	--

[Advanced Parameters](#) [Show](#)

Responses

201	Reservation	
404	NewReservationError	
500	error	

[+ Add Response](#)

[Cancel](#) [Save](#)

It should generate code as below.

```
resource      function      post      .(NewReservationRequest      payload)      returns
Reservation|NewReservationError|error {
}
```

6. Add the other resources similar to the steps above.

Refer to the following table for the types required for each resource.

Resource	Path	Action	Path Param	Request Payload Type	Response Type
Get all reservations for a given user id	/users/	GET	userID		Reservation[]



Configure Resource

X

HTTP Method*

Resource Path Optional

GET

users/[string userId]

+ Add Path Param

Parameters

+ Add Parameter

Advanced Parameters [Show](#)

Responses

200

Reservation[]



+ Add Response

Cancel

Save

10. Completed two resources will be as follows.

```
resource function post .(NewReservationRequest payload) returns Reservation|NewReservationError|error {
```

```
}
```

```
resource function get users/[string userId]() returns Reservation[] { }
```

11. Complete the body of the resource with the application logic. You can use functions in utils.bal to complete the steps.

12. Run the Tests and verify that all resources are working fine. You can find the ballerina test code in tests/service_test.bal

Note: For Windows, we need to use “..\resources\rooms.json” in Config.toml file

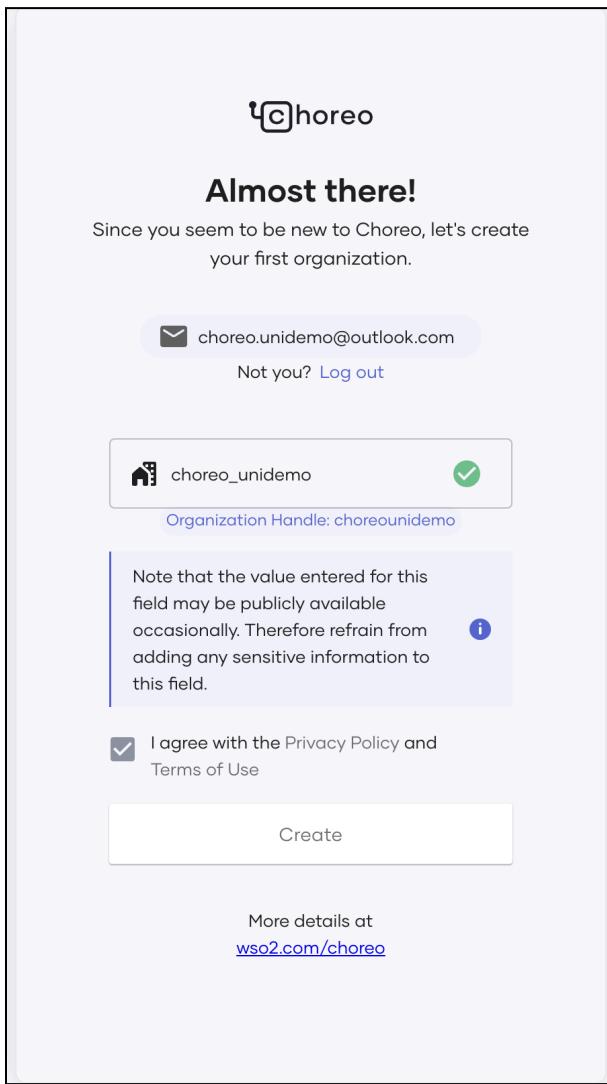
Unset

bal test

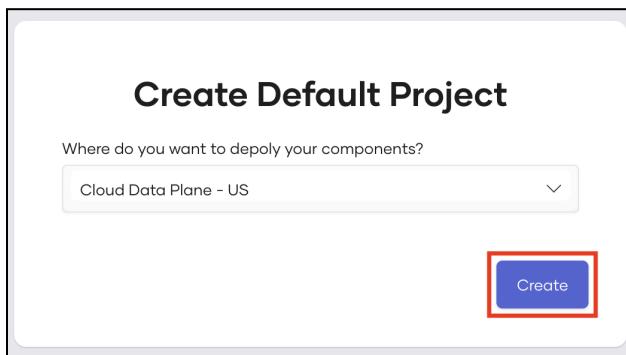
Deploying the Hotel Reservation App using Choreo

Step 1: Sign Up and Login to Choreo

1. Sign Up to Choreo by visiting the <https://choreo.dev/> URL.
2. Once you log in to Choreo for the first time, you will be asked to provide an organization handle name. Provide a handle name and click **Create**.



3. To create the Default Project, choose the dataplane where you wish to deploy your components and click **Create**.



4. You will now be directed to the overview of the **Default Project**.

Step 2: Create a new Project

1. Click on the **Organization** card in the top menu.



2. Click the **Create Project** card.

3. Add the details shown below and click **Create**.

Field Name	Field Value
Name	Luxury Hotels
Description	Luxury Hotels reservation system.
Project Type	Multi Repository

The screenshot shows the 'Create Project' wizard. On the left, there's a vertical sidebar with four steps: STEP 01 (Provide Basic Details), STEP 02 (Select Git Provider), STEP 03 (Connect Repository), and STEP 04 (Import Component Code). The first step is highlighted with a blue circle. The main area is titled 'Create Project' and has a sub-section 'Provide Basic Details'. It contains fields for 'Name' (set to 'Luxury Hotels') and 'Description' (set to 'This is a project on a hotel reservation system.'). Below these fields are two buttons: 'Mono Repository' (described as a single repository containing all code and assets) and 'Multi Repository' (described as multiple repositories for specific components). At the bottom are 'Back' and 'Create' buttons.

4. You will be directed to the overview of the project.

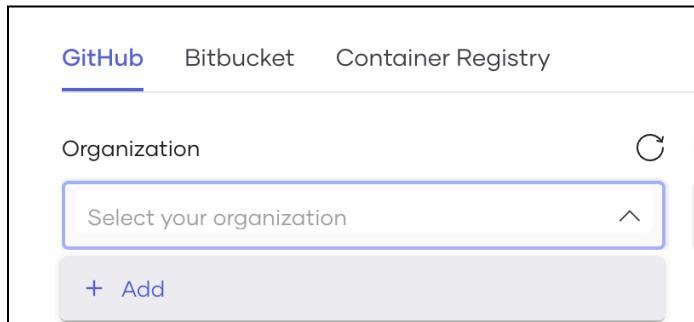
Step 3: Create and Deploy a Service for the Hotel Reservation Ballerina App

1. Select the **Service** card.
2. Provide a name and a description for the Service.

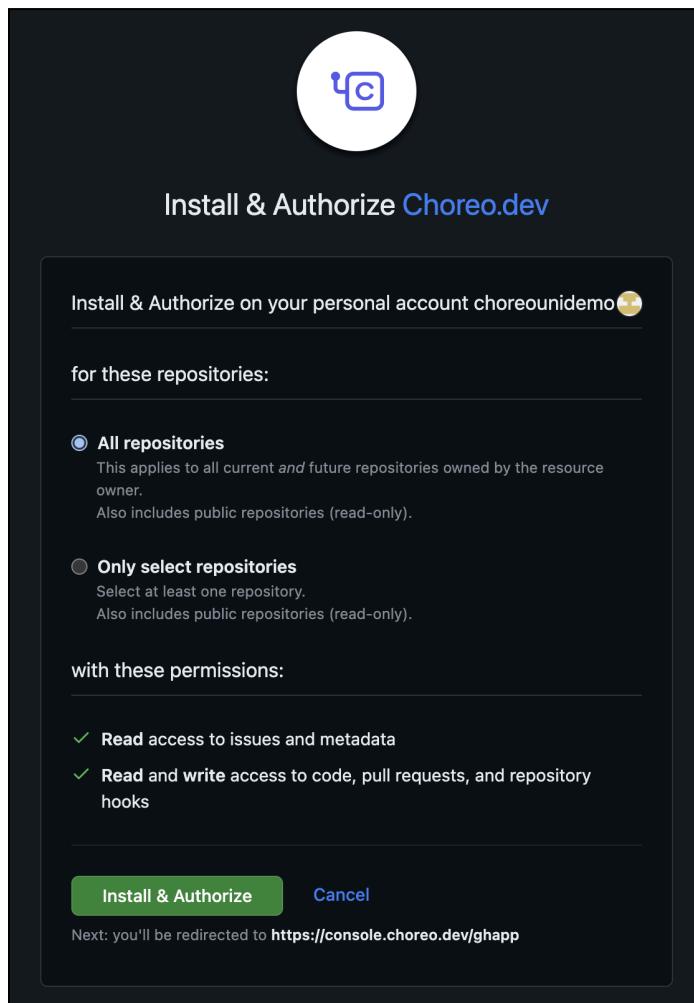
Field Name	Field Value
Name	Hotel Reservation Service
Description	Service for the hotel reservation application.

3. Click on the **Authorize With GitHub** and Click **Authorize Choreo.dev**.

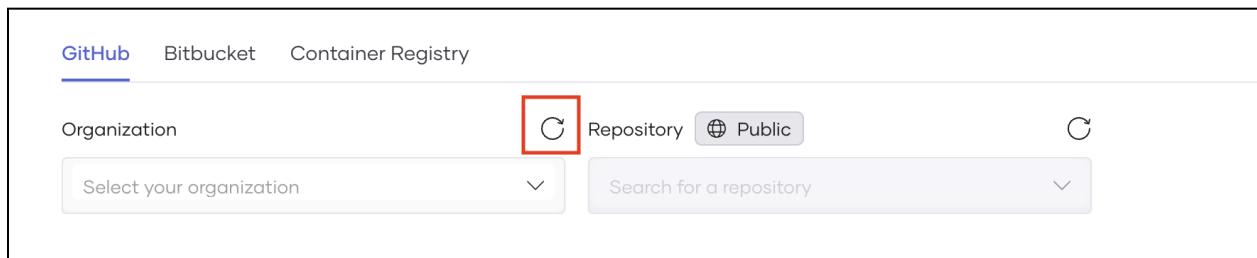
4. Expand the Select your organization section and click **Add**.



5. Keep the default selected option of **All repositories**. Then click **Install & Authorize**.



6. Click the refresh icon next to Organization.



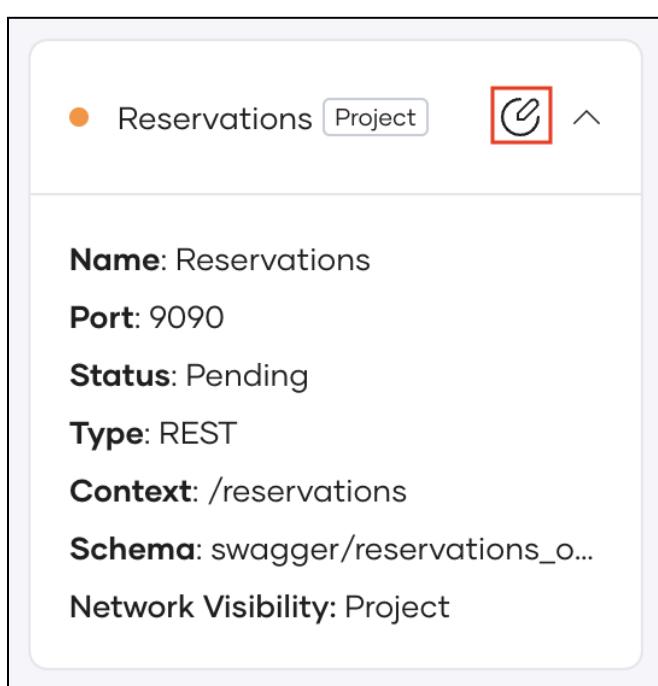
7. Select the **hotel-reservation-demo** repository and select the branch **complete**.



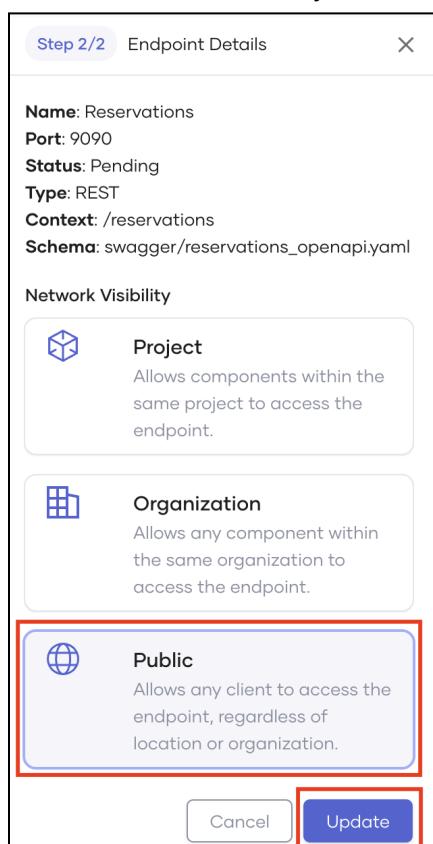
8. Select **Ballerina** as the **Buildpack** and set the Ballerina Project Directory as below. Then select **Create**.

Field Name	Field Value
BuildPack	Ballerina
Ballerina Project Directory	/service

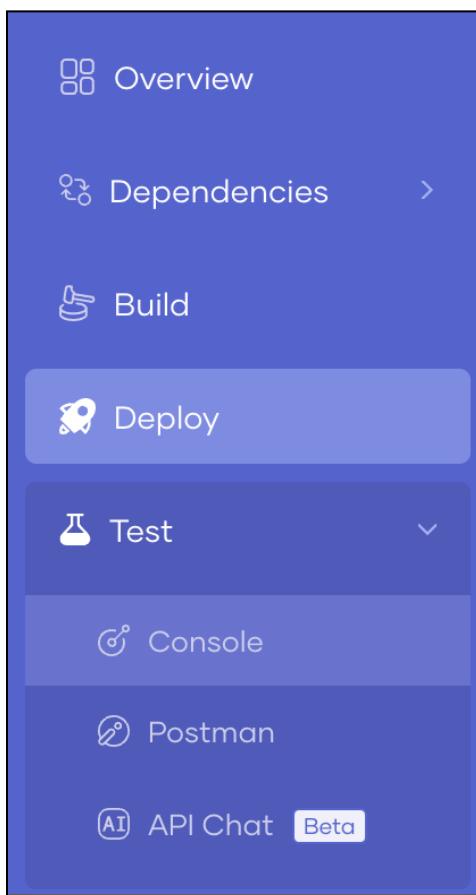
9. Click **Build** on the left navigation menu. Click the **Build** button on the Build page. This will open the **Commits** panel on the right. Select the most recent commit and click the **Build** button in the commits panel.
10. Click **Deploy** on the left navigation menu. In the Deploy page, click the **Configure & Deploy**.
11. Click the **Edit** button.



12. Set the Network Visibility as **Public** and click **Update**.



13. Click **Deploy** to deploy the service.
14. Once the service is deployed successfully, click **Promote** to promote the build to the Production environment.
15. Select **Use Development Configurations** and click **Next**. This will load the provided development configs in the room_details_file. Click **Next** and click the **Promote** button in the right side panel.
16. Click and expand **Test** on the left navigation menu and click **Console**.



17. Expand the **POST/** section and click **Try it out**. In the **Request body** section, add the following details:

Field Name	Field Value
Request body	{ "checkinDate": "2024-02-19T14:00:00Z", "checkoutDate": "2024-02-20T10:00:00Z", "rate": 100, "user": { "id": "123", "name": "testuser", "email": "testuser@someemail.com", "mobileNumber": "911234567821" }, "roomType": "Family" }

18. Click **Execute**.

19. Upon a successful POST request, you will receive a response as shown below.

Server response

CODE	DETAILS
201	<p>Response body</p> <pre>{ "id": 1, "room": { "number": 303, "type": { "id": 2, "name": "Family", "guestCapacity": 4, "price": 200 } }, "checkinDate": "2024-02-19T14:00:00Z", "checkoutDate": "2024-02-20T10:00:00Z", "user": { "id": "123", "name": "testuser", "email": "testuser@someemail.com", "mobileNumber": "911234567821" } }</pre> <p>Download</p> <p>Response headers</p> <pre>content-length: 207 content-type: application/json</pre>

20. Expand the **GET /roomTypes** section and click **Try it out**.

21. Add the following details in each parameter and click **Execute**.

Field Name	Field Value
checkinDate	2024-02-19T14:00:00Z
checkoutDate	2024-02-20T10:00:00Z
guestCapacity	2

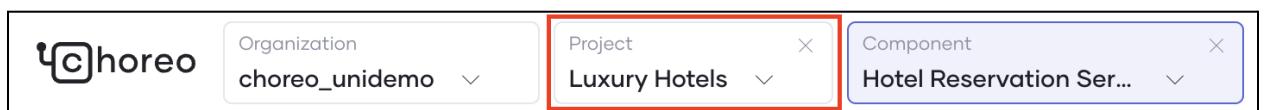
22. Upon a successful GET request, you will receive a response as shown below.

Server response

CODE	DETAILS
200	<p>Response body</p> <pre>[{ "id": 1, "name": "Double", "guestCapacity": 2, "price": 120 }, { "id": 3, "name": "Suite", "guestCapacity": 4, "price": 300 }, { "id": 2, "name": "Family", "guestCapacity": 4, "price": 200 }]</pre> <p>Download</p> <p>Response headers</p> <pre>content-length: 118 content-type: application/json</pre>

Step 4 - Deploy the Hotel Reservation Web Application

- From the top main menu, select the **Project** tab.

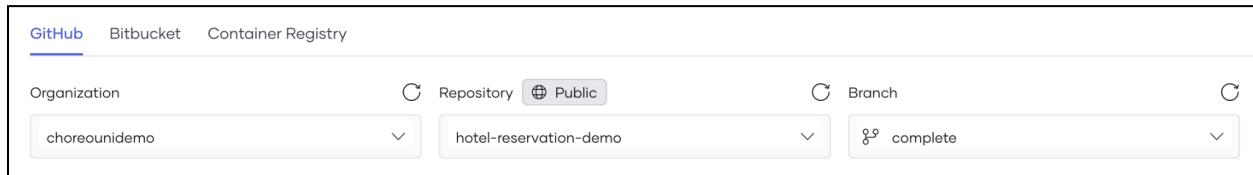


- Click **Create**.

3. Select the **Web Application** card.
4. Provide a name and a description for the Web Application.

Field Name	Field Value
Name	Hotel Reservation Web Application
Description	Web Application for the hotel reservation application.

5. Click on the **Authorize With GitHub**.
6. Select the **hotel-reservation-demo** repository and select the branch **complete**.



7. Select **React** as the **Buildpack** and set the following details. Then click **Create**.

Field Name	Field Value
BuildPack	React
Project Directory	/webapp
Build Command	npm run build
Build Path	/build
Node Version	20.11.0

8. Click and expand **Dependencies** on the left navigation menu and click the **Connections** tab.
9. Click **Create**.

10. Select the **Hotel Reservation** Service that was created in step 3..

The screenshot shows the 'Create Connection' wizard in the choreo interface. The left sidebar has icons for various services. The main area shows three steps:

- STEP 01 Select a Service**: Shows a list of services, with 'Hotel Reservation' selected.
- STEP 02 General Details**
- STEP 3 Connection Configuration**

11. Provide a name and a description for the connection.

Field Name	Field Value
Name	Hotel Reservation System Connection
Description	Connection between the hotel reservation service and the web app.

12. Click **Next**. Then click **Finish**.

13. Copy and save the **Service URL** for later use.

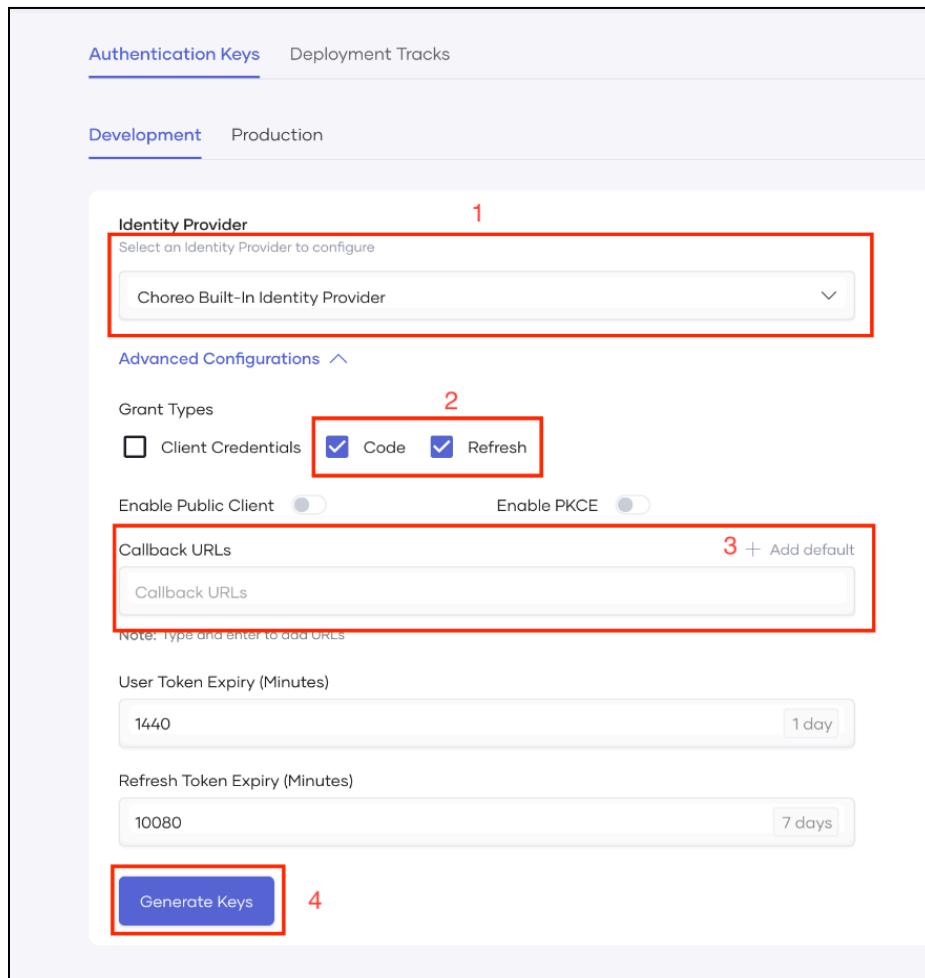
The screenshot shows the 'Hotel Reservation System Connection' details page. It includes:

- Connection description: 'Connection between the hotel reservation service and the web app.'
- Connecting to: 'Hotel Reservation Service-Reservations'
- Connection Schema: 'Default OAuth Connection - Public'
- Service URL: '/choreo-apis/owkb/hotel-reservation-service/reservativ' (with a copy icon)

14. Click **Build** on the left navigation menu. Click the **Build** button in the Build page. This will open the **Commits** panel on the right. Select the top most commit and click the **Build** button on the commits panel.

15. Once the build is completed, Click **Settings** on the left navigation menu.

16. Under the **Development** tab, select Choreo Built-In Identity Provider as the Identity Provider. Expand the **Advanced Configuration** section. Select the **Code** and **Refresh** grant types. Click **Add Default** to add a default callback url. Then click **Generate Keys**.



17. Click **Deploy** on the left navigation menu. Click the **Configure & Deploy** in the Deploy page.

In the right side panel, provide config.json as the Configuration File Name and provide the following content:

```
Unset
window.configs = {
    apiUrl: '<Service URL>',
};
```

Replace <Service URL> with the value that you copied when creating a connection to the Service in Step 13.

Ex:

```
Unset
window.configs = {
    apiUrl:
        '/choreo-apis/owkb/hotel-reservation-service/reservations-365/v1.
        0',
};
```

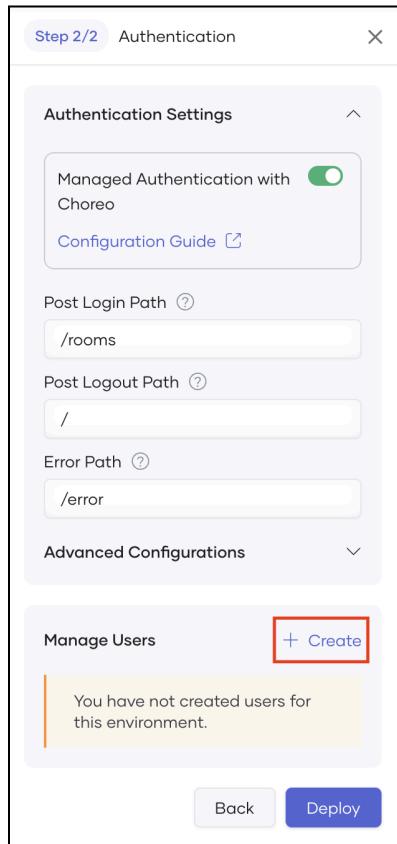


18. Click **Next**.

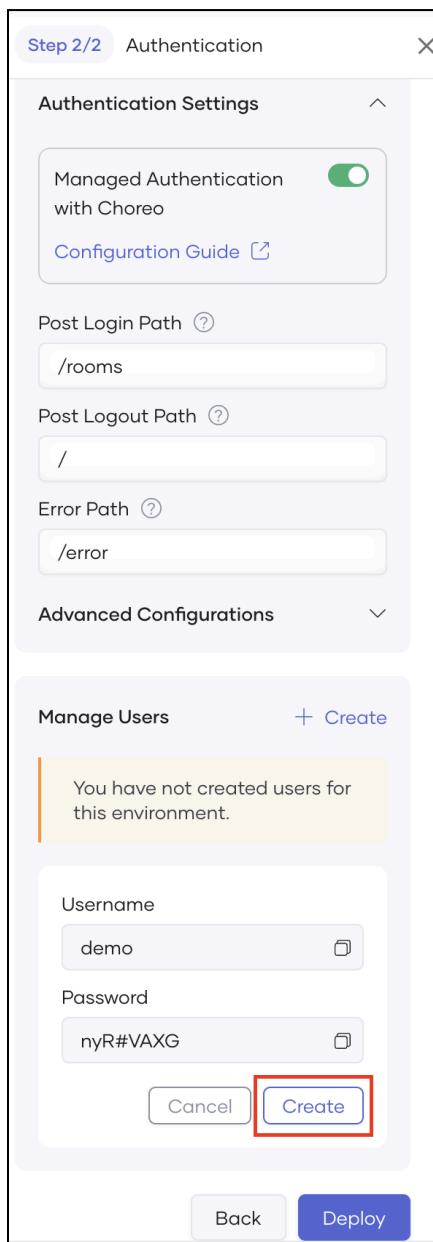
19. Provide following values for each parameter in the Authentication Settings window.

Field Name	Field Value
Post Login Path	/rooms
Post Logout Path	/
Error Path	/error

20. Click **Create** to expand the demo user creation window.



21. Click **Create** to create the user.



22. Once the user is created, save the username and the password for later use.
23. Click **Deploy** in the right side panel to deploy the web application.
24. Once the web application is deployed successfully, click on the **Web App URL** to access the web app.

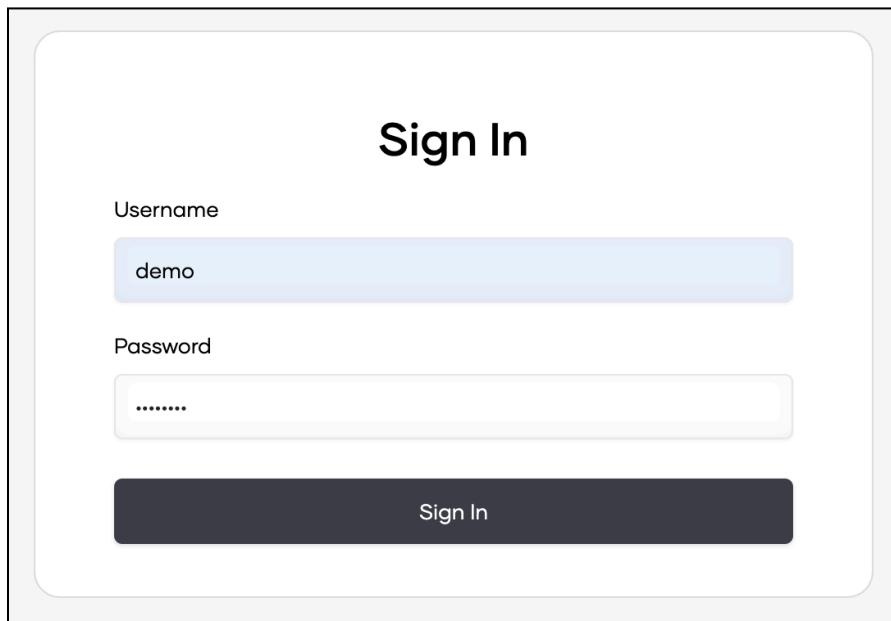
The screenshot shows the WSO2 Choreo deployment interface. The top navigation bar includes 'Organization: choreo_unidemo', 'Project: Luxury Hotels', and 'Component: Hotel Reservation We...'. The left sidebar has various icons for deployment, monitoring, and configuration. The main area is divided into 'Set up' and 'Development' sections. In the 'Development' section, there is a 'Deployment Status' card showing 'Active' and a 'Deployment History' section. Below these are 'Image' and 'Web App URL' sections. The 'Web App URL' section contains the URL <https://b4f1c834-1f76-47c7-8261-...>, which is highlighted with a red box. To the right, there is a 'Production' section labeled 'Not yet Deployed'.

25. You will be greeted with the landing page shown below:

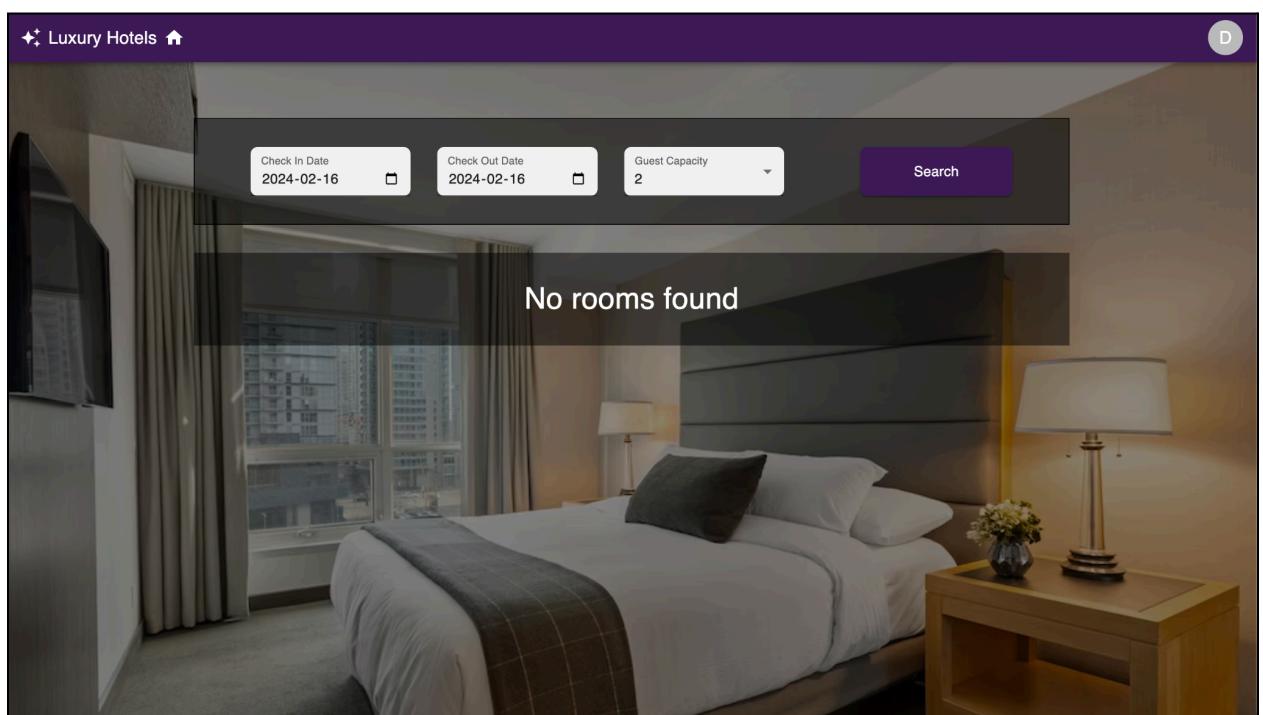
The screenshot shows the 'Luxury Hotels' landing page. The background features a photograph of a modern hotel room with a large window overlooking a city skyline. The text 'Reserve a room online' is displayed in a large, bold, white font. Below it, the text 'Enjoy convenience' is shown in a smaller, white font. At the bottom center, there is a purple button with the text 'Get Started...'. The top of the page has a purple header bar with the text 'Luxury Hotels' and a home icon.

26. Click the **Get Started...** button.

27. This will direct you to the login page. Provide the demo username and password (created in the Step 23) and click Sign In.

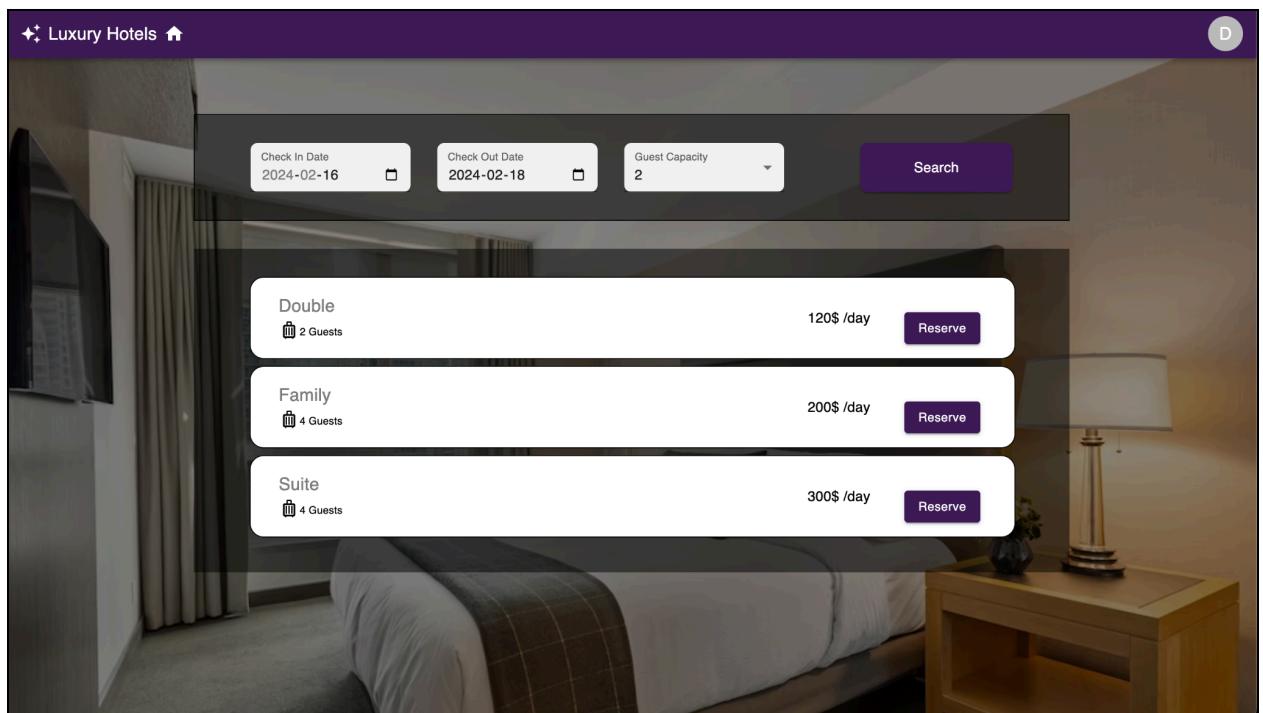


28. You will be logged into the Hotel Reservation web application.



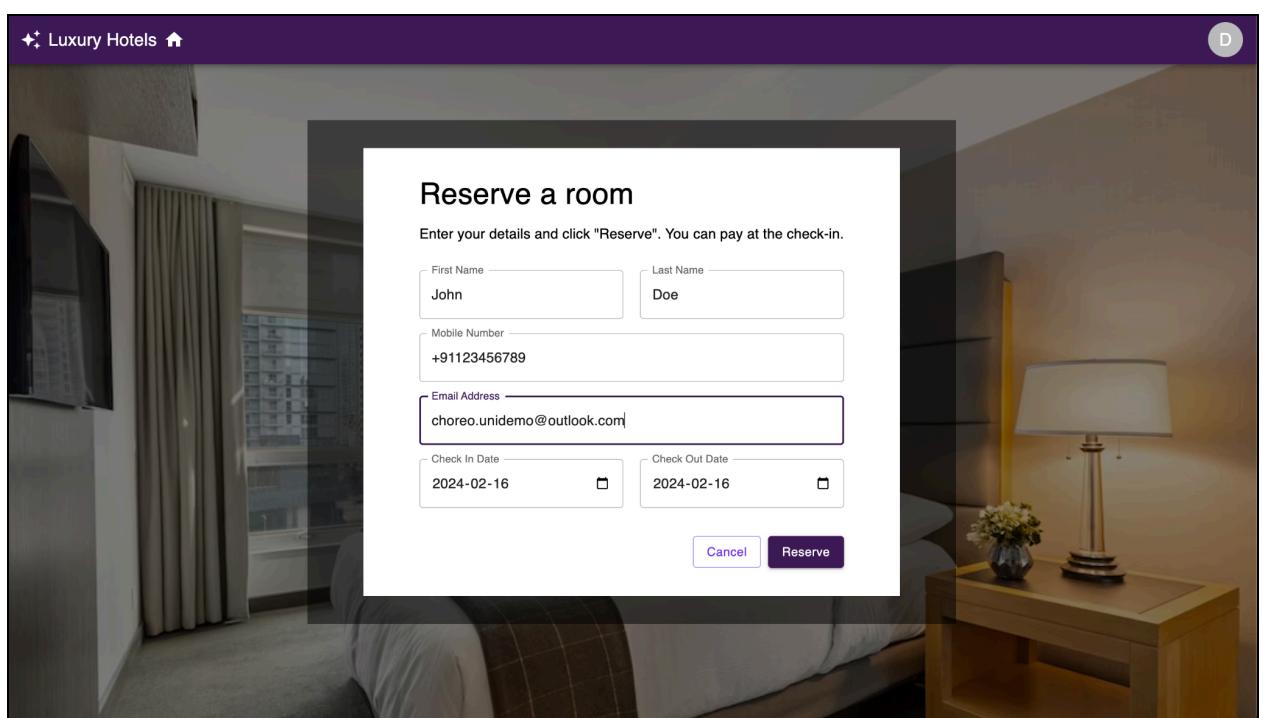
29. Set a check in date, check out date and select the no of guests. Then click **Search**.

30. You will be able to get the available rooms for the selected date range and guest capacity.



31. Click **Reserve**.

32. Provide required details as below and click Reserve.



33. Your reservation will be created and you will receive an email confirmation.