



AI Chef

Phase 1: Problem Understanding & Industry Analysis - AI Chef Project

Core Problem Definition

The **AI Chef** project addresses the critical challenge of **leftover ingredient utilization** and **personalized recipe generation** in modern households. With global food waste reaching **30-40% of all food produced**, there is a significant opportunity to leverage AI technology to minimize waste while creating personalized culinary experiences.

Primary Problem Areas:

- **Food Waste Crisis:** Approximately 17% of total global food production is wasted, with 11% occurring at household level.
- **Decision Fatigue:** Home cooks struggle to create meals from available leftover ingredients like meat, dal, vegetables, and pantry staples.
- **Dietary Restriction Management:** Growing need for AI-powered solutions that accommodate specific dietary needs (gluten-free, vegan, keto, diabetic-friendly).
- **Meal Planning Inefficiency:** Lack of personalized, intelligent systems that consider serving size, meal type, and ingredient availability.

Industry Analysis & Market Landscape

Market Size & Growth Trajectory

The AI-powered culinary technology sector demonstrates robust growth potential:

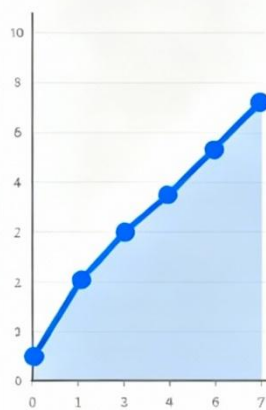
- **AI-Driven Meal Planning Apps:** Market valued at **USD 11.57 billion by 2034** with a **CAGR of 28.10** .
- **Recipe App Market:** Projected to reach **USD 1.8 billion by 2033** with a **CAGR of 17.5%**.
- **AI-Driven Recipe Creators:** Market size of **USD 3.5 billion in 2024**, expanding to **USD 10 billion by 2032** at **15% CAGR**.
- **Food Waste Management Apps:** Expected to grow from **USD 1.2 billion in 2024** to **USD 5.7 billion by 2034** at **16.8% CAGR**.

AI Culinary Technology Market Growth 2024-2034



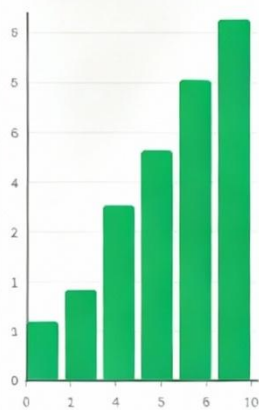
AI-Driven Meal Planning Apps:

\$11.57B by 2034
28.10% CAGR



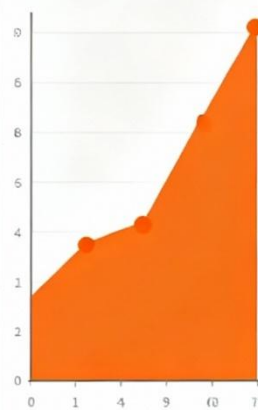
Recipe App Market:

\$1.8B by 2033
17.5% CAGR



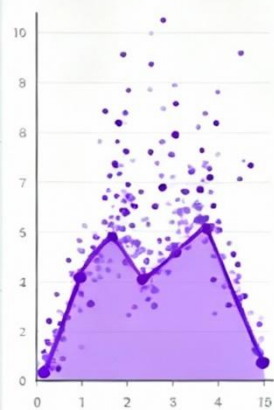
AI-Driven Recipe Recommenders:

\$10B by 2032
15% CAGR



Food Waste Management Apps:

\$5.7B by 2034
16.8% CAGR



Key Market Drivers

Technology Integration Trends:

- Integration with wearables and health apps for holistic nutrition management.
- IoT-based smart kitchen integration for seamless cooking experiences.
- Real-time food recognition using computer vision technology.
- Multi-lingual voice interfaces for accessibility across diverse user bases.

Consumer Behavior Shifts:

- **80% increase in user engagement** through personalized experiences.
- Growing demand for specialized dietary solutions (vegan, keto, gluten-free) .
- Rising health consciousness driving personalized nutrition demand.

Competitive Landscape Analysis

Existing Solutions

Leading AI Recipe Platforms:

- **FoodsGPT, DishGen, ChefGPT:** Focus on general recipe generation.
- **SuperCook:** Ingredient-based recipe suggestions.
- **RecipeRadar:** Search recipes by available ingredients.
- **Samsung Food Plus:** Vision AI recognizing 40,000 ingredients with cloud processing.

Unique Value Proposition for Salesforce AI Chef

Differentiating Features:

1. **Salesforce Agentforce Integration:** Leverage advanced prompt builder and Einstein Trust Layer for secure, personalized AI interactions.
2. **Enterprise-Grade Data Management:** Utilize Salesforce's robust CRM capabilities for user preference tracking and meal history analytics.
3. **Multi-Parameter Intelligence:** Advanced algorithm considering ingredients, dietary restrictions, serving size, and meal type simultaneously.
4. **Leftover-Centric Approach:** Specialized focus on transforming common leftovers (meat, dal, vegetables) into complete meals.
5. **Cultural Adaptation:** Built-in understanding of regional ingredients and cooking methods, particularly valuable for Indian cuisine integration.

Key Stakeholders & User Personas

Primary Stakeholders

End Users:

- **Health-Conscious Home Cooks:** Seeking personalized nutrition solutions.
- **Busy Professionals:** Requiring quick, efficient meal planning with available ingredients.
- **Dietary-Restricted Individuals:** Managing specific health conditions through AI-guided meal planning .
- **Sustainability-Focused Households:** Committed to reducing food waste through intelligent ingredient utilization.

Business Stakeholders:

- **Food Retailers:** Potential integration for inventory-based recipe suggestions.
- **Nutritionists & Dietitians:** Professional tools for client meal planning.
- **Hospitality Industry:** Restaurants managing dietary restrictions and menu optimization.

Business Model Opportunities

Revenue Streams:

- **Subscription Model:** Premium features for advanced personalization and unlimited recipe generation.
- **Integration Services:** B2B partnerships with grocery retailers and meal kit services.
- **Data Analytics:** Insights for food industry stakeholders on consumption patterns and waste reduction.
- **Corporate Wellness:** Employee health programs focusing on personalized nutrition.

Success Metrics & Project Goals

Quantifiable Objectives:

- **Food Waste Reduction:** Target 20-30% reduction in household ingredient waste based on industry benchmarks.
- **User Engagement:** Achieve 80% user retention through personalized experience.
- **Recipe Success Rate:** 85% user satisfaction with AI-generated recipes matching taste preferences.
- **Dietary Compliance:** 90% accuracy in accommodating specified dietary restrictions and serving sizes.

Technical Goals:

- Real-time ingredient recognition and processing.
- Sub-second recipe generation response times.
- Seamless integration with Salesforce ecosystem.
- Scalable architecture supporting 10,000+ concurrent users.

Phase 2: Org Setup & Configuration

Learning Objectives:

- Set up Salesforce Developer Edition effectively
- Configure healthcare-specific organizational settings
- Establish foundation for user management and security

1. Salesforce Org Setup:

Create Developer Edition account and configure company profile.

salesforce

Build enterprise-quality apps fast and get hands-on with Agentforce and Data Cloud.

Sign up for your Developer Edition.

- ✓ Build apps fast with drag-and-drop tools
- ✓ Go further with Apex code
- ✓ Build AI agents with Agentforce
- ✓ Harmonize your data with Data Cloud
- ✓ Ground Agentforce with structured and unstructured data
- ✓ Integrate with anything using APIs

Sign up for your Developer Edition

A free Salesforce Platform environment with Agentforce and Data Cloud

First name Last name

Job title Work email

Company Country/Region

Your org may be provisioned on or migrated to Hyperforce, Salesforce's public cloud infrastructure.

☐ I agree to the Main Services Agreement - Developer Services and Salesforce Program Agreement. I acknowledge, as described in the Developer Documentation: (1) the Developer Edition includes autonomous and other generative AI features; and (2) Salesforce may limit use of those features and the org, and may terminate any org that has been inactive for 45 days.

We value your privacy. To learn more, visit our Privacy Statement.

Airtel_Sonkusale Internet access

2. Company Profile Details

- **Company Name:** Tech Tales Healthcare Pvt Ltd
- **Legal Name:** Tech Tales Healthcare Private Limited
- **Address:** 2nd Floor, Crystal Plaza, MG Road, Bengaluru, Karnataka, India, 560001
- **Default Currency:** INR (Indian Rupee)
- **Time Zone:** (GMT+05:30) India Standard Time (Asia/Kolkata)
- **Phone:** +91-9876543210
- **Primary Contact:** Avinash Varikuti (Project Owner)
- **Locale:** English (India)
- **Division:** Healthcare Services

Company Information | Salesforce

orgfarm-009bd262ce-dev-ed.develop.lightning.force.com/lightning/setup/CompanyProfileInfo/page?address=%2F00DgL00000BuyG6%3FretURL%3D%252Fhome%26appLayout%3...

ACTION MOVIESMARATHI HISTORI...ONE SHOT IWTYCCCE ALUMINI AI IMAGE GENERAT...SQL PRACTICEAGENTIC AI AI TOOLSAll Bookmarks

Search Setup

SetupHomeObject Manager

Company

Company Settings

Business Hours

Calendar Settings

Public Calendars and Resources

Company Information

Data Protection and Privacy

Fiscal Year

Holidays

Language Settings

My Domain

Didn't find what you're looking for? Try using Global Search.

SETUP

Company Information

Company Information

Techi Tales Healthcare Pvt Ltd

The organization's profile is below.

User Licenses (10) | Permission Set Licenses (10) | Feature Licenses (11) | Usage-based Entitlements (10)

Organization Detail

Organization Name

Primary Contact

Division

Address

Fiscal Year Starts In

Activate Multiple Currencies

Enable Data Translation

Newsletter

Admin Newsletter

Hide Notices About System Maintenance

Hide Notices About System Downtime

Locale Formats

Techi Tales Healthcare Pvt Ltd

Avinash Varikuti

Healthcare Services

2nd Floor, Crystal Plaza, MG Road, Bengaluru 560001
Karnataka
India

January

☐

☐

☒

☒

☐

☐

ICU

Phone

Fax

Default Locale

Default Language

Default Time Zone

Currency Locale

Used Data Space

Used File Space

API Requests, Last 24 Hours

Streaming API Events, Last 24 Hours

Restricted Logins, Current Month

Salesforce.com Organization ID

Organization Edition

+91-9876543210

English (United States)

English

(GMT+05:30) India Standard Time (Asia/Kolkata)

Hindi (India) - INR

342 KB (7%) [View](#)

17 KB (0%) [View](#)

39 (15,000 max)

0 (10,000 max)

0 (0 max)

00DgL00000BuyG6

Developer Edition

Phase 4: Process Automation (Admin)

- Creating a new lightning app:

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

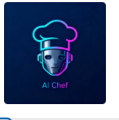
App Details

* App Name [?]
AI Chef

* Developer Name [?]
AI_Chef

Description [?]
Enter a description...

App Branding

Image [?]

Clear

Primary Color Hex Value [?]
#0070D2

Org Theme Options
☐ Use the app's image and color instead of the org's custom theme

Next

Flow Category	ID	Name	Description	Created	Version	Status
Lightning Bolt Solutions	11	Data Cloud	Audience360	Build a thorough and complete understanding of your customers.	9/17/2025, 1:55 AM	Lightning
	12	Data Manager	DataManager	Use Data Manager to view limits, monitor usage, and manage recipes.	9/17/2025, 1:55 AM	Lightning
Mobile Apps	13	Developer Edition	Developer Edition	Welcome to your Developer Edition Org.	9/17/2025, 4:16 AM	Lightning (Managed)

App Name: AI chef

Developer Name: AI_Chef

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.


App Details

* App Name [?]
AI CHEF

* Developer Name [?]
AI_CHEF

Description [?]
Gives the recipe of dish


App Branding

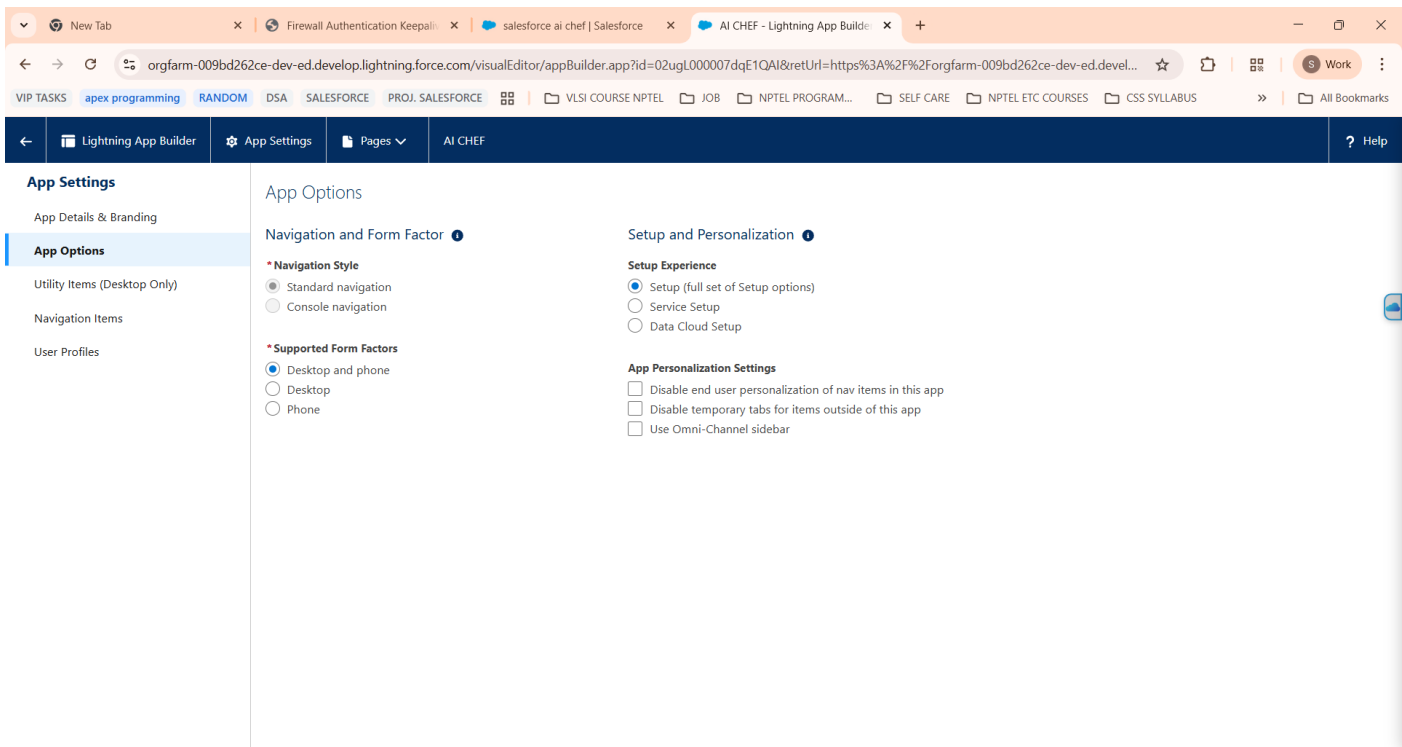
Image [?]

Clear

Primary Color Hex Value [?]
#847A80

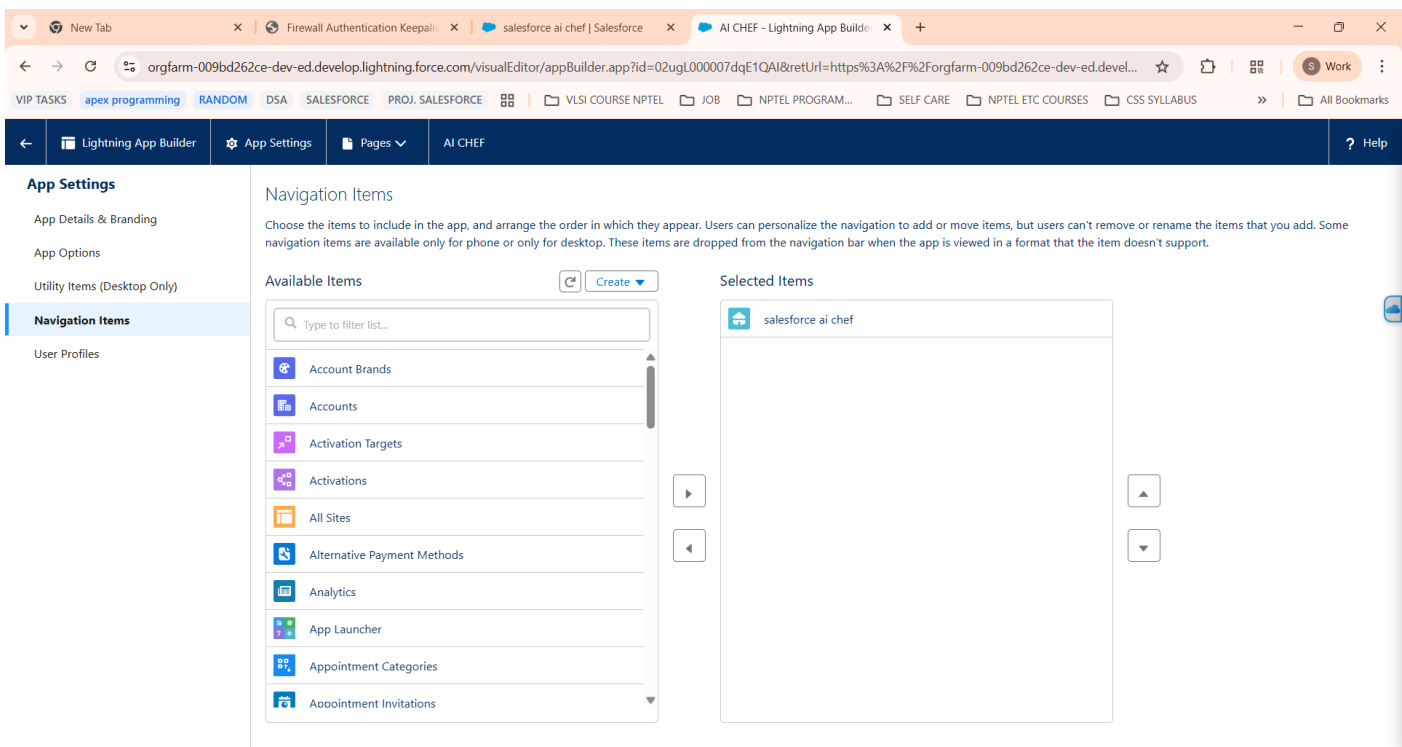
Org Theme Options
☐ Use the app's image and color instead of the org's custom theme

App Launcher Preview

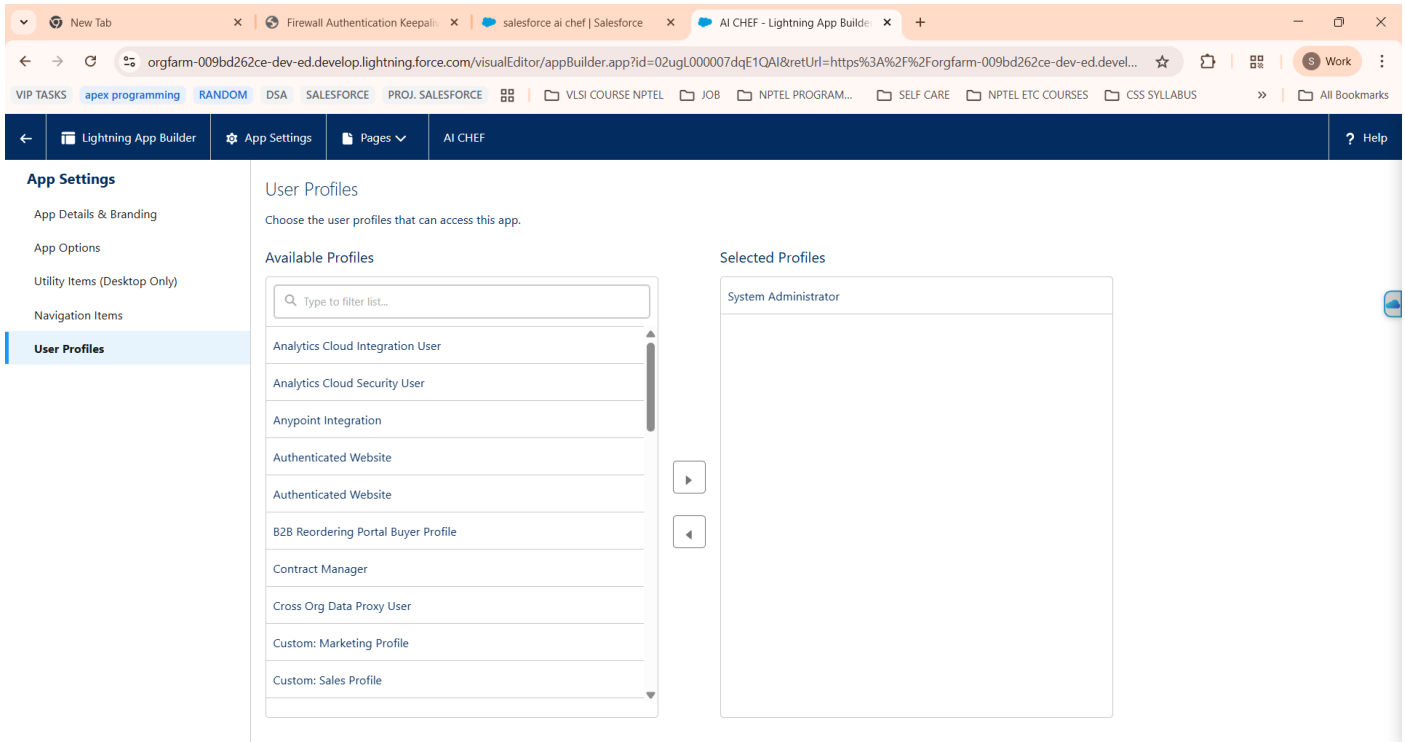
 AI CHEF
Gives the recipe of dish



• Adding a page in app

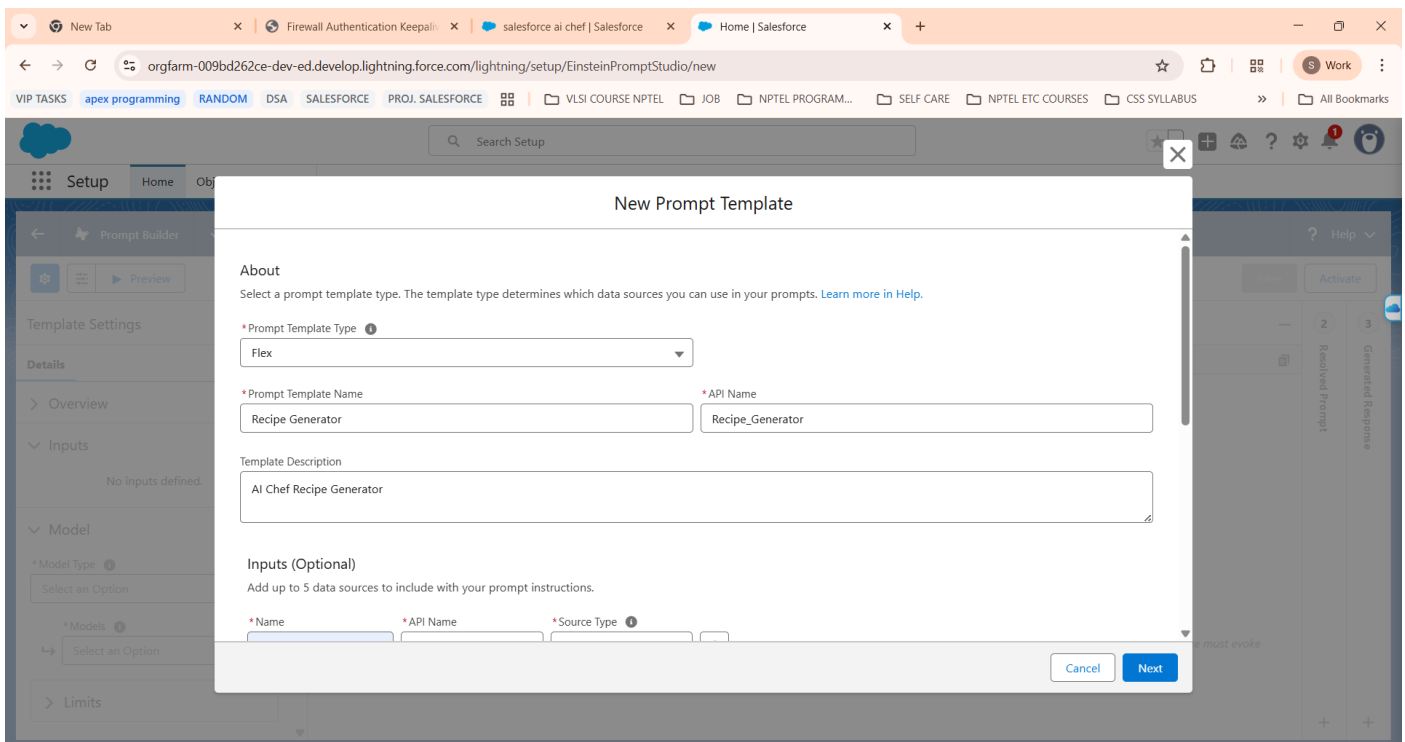


- Giving permission to user profile

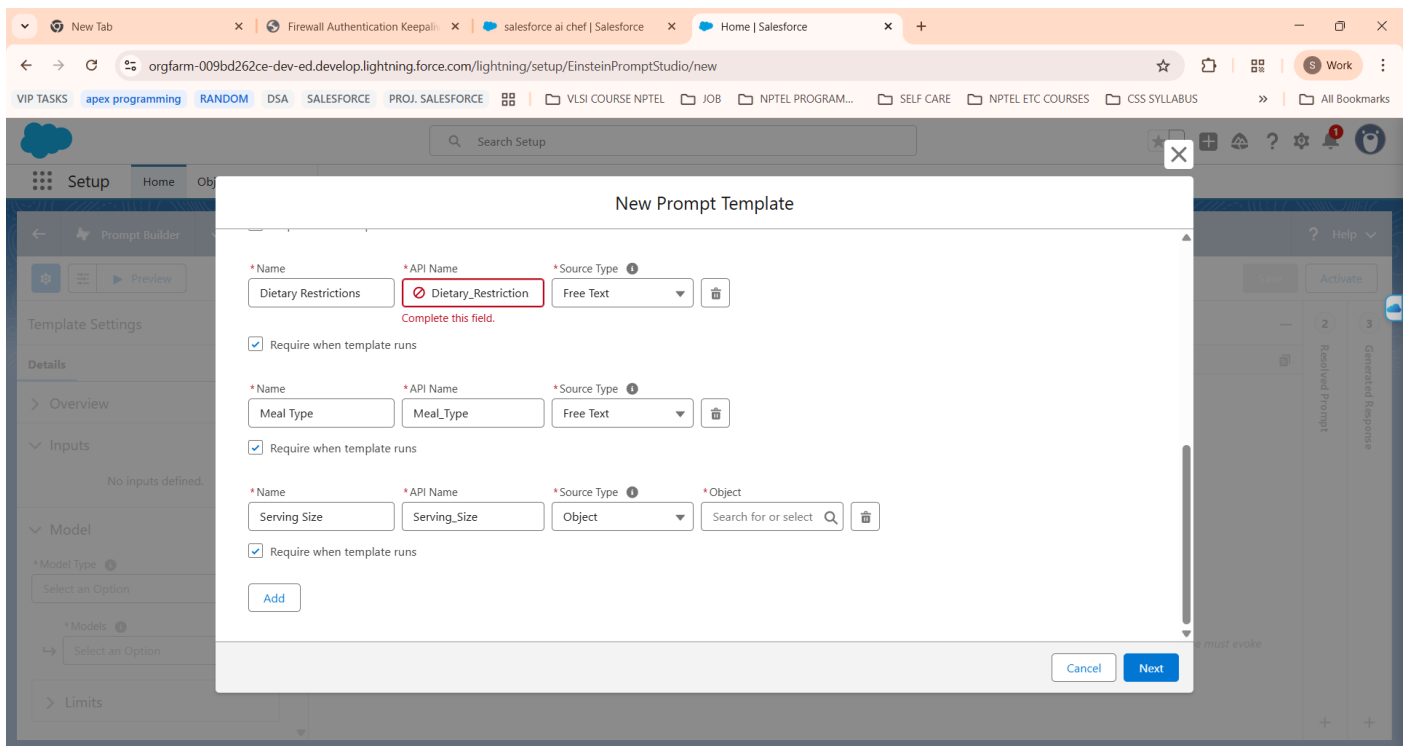
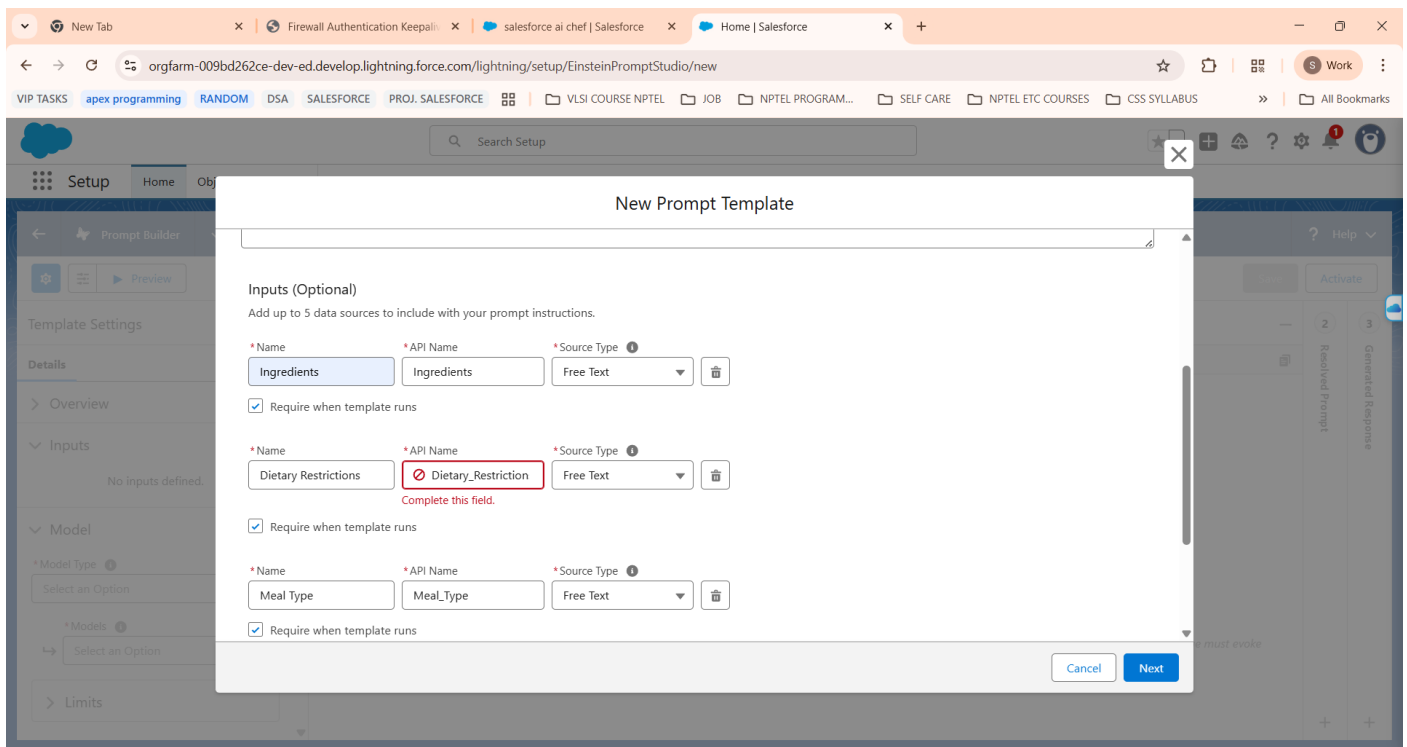


- Prompt Builder

- Creating a prompt Template



- **Prompt Template Name:** Recipe Generator
- **API Name:** Recipe_Generator
- **Inputs:** Ingredients, Dietary Restrictions, Meal Type, Serving Size



➤ Writing Prompt

Prompt:

You are a professional chef specialising in creative cooking. Generate 3 different recipes based on these parameters:

Ingredients available: {!\$Input:Ingredients}

Dietary restrictions: {!\$Input:Dietary_Restrictions}

Meal type: {!\$Input:Meal_Type}

Servings: {!\$Input:Serving_Size}

Return your response in the following JSON format for easy HTML rendering:

```
{
  "recipes":[
    {
      "recipe_number":"1",
      "servings":"1",
      "recipe_name":"Name of recipe 1",
      "description":"Breif 1-2 sentence description of the dish",
      "prep_time":"Time in minutes",
      "cook_time":"Time in minutes",
      "total_time":"Time in minutes",
      "difficulty":"Easy/Medium/Hard",
      "ingredients":[
        {
          "amount":"quantity",
          "unit":"measuring unit",
          "name":"ingredient name"
        },
        {
          "amount":"quantity",
```

```

        "unit":"measuring unit",
        "name":"ingredient name"
    }
],
"instructions":[
    "step1 instruction",
    "step2 instruction"
],
"tips":"Optional cooking or serving tip",
"tags":["tag1", "tag2"]
}
]
}

```

The screenshot displays the Salesforce Einstein Prompt Studio interface for a 'Recipe Generator' prompt. The interface is divided into several sections:

- Header:** Includes navigation tabs for 'Setup', 'Home', and 'Object Manager'. The main header shows 'Prompt Builder', 'Recipe Generator', and 'Version 1 (Active)'.
- Template Settings:** A sidebar on the left with sections for 'Details', 'Overview', 'Inputs', 'Model', and 'Response'.
- Prompt Configuration:** The main area shows a prompt titled '1 Prompt'. It includes an 'Insert Resource' button and a text area with the following content:

You are a professional chef specialising in creative cooking. Generate 3 different recipes based on these parameters:
 Ingredients available: [Input:Ingredients](#)
 Dietary restrictions: [Input:Dietary_Restrictions](#)
 Meal type: [Input:Meal_Type](#)
 Servings: [Input:Serving_Size](#)

Return your response in the following JSON format for easy HTML rendering:

```
{
  "recipes":[
    {
      "recipe_number":"1",
      "servings":"1",
      "recipe_name":"Name of recipe 1",
      "description":"Brief 1-2 sentence description of the dish",
      "prep_time":"Time in minutes",
      "cook_time":"Time in minutes",
      "total_time":"Time in minutes",
      "difficulty":"Easy/Medium/Hard",
      "ingredients":[

```
- Generated Response:** A sidebar on the right showing the 'Resolved Prompt' and the 'Generated Response'.
- Buttons:** At the top right, there are buttons for 'Save As', 'Save', 'Deactivate', and 'Delete Version'.

➤ Model Selection

▼ Model

* Model Type ⓘ

Standard ▼

* Models ⓘ

↪ OpenAI GPT 4 Omni Mini ▼

> Limits

> File Exclusion Priority

➤ Resolved Prompt:

The screenshot shows the Salesforce Einstein Prompt Studio interface. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. The main content area is titled 'Recipe Generator' and 'Version 1 (Active)'. On the left, the 'Preview Settings' sidebar shows the following inputs:

- * Ingredients: meat, daal
- * Dietary Restrictions: none
- * Meal Type: lunch
- * Serving Size: 2

The main area displays the 'Resolved Prompt' (tab 2) with the following text:

You are a professional chef specialising in creative cooking. Generate 3 different recipes based on these parameters:
Ingredients available: meat, daal
Dietary restrictions: none
Meal type: lunch
Servings: 2

Return your response in the following JSON format for easy HTML rendering:

```
{
  "recipes": [
    {
      "recipe_number": "1",
      "servings": "1",
      "recipe_name": "Name of recipe 1",
      "description": "Brief 1-2 sentence description of the dish",
      "prep_time": "Time in minutes",
      "cook_time": "Time in minutes",
      "total_time": "Time in minutes",
      "difficulty": "Easy/Medium/Hard",
      "ingredients": [
```

On the right, the 'Generated Response' (tab 3) is visible but empty.

➤ Generated Response:

The screenshot displays the Salesforce Einstein Prompt Studio interface. The browser address bar shows the URL: `orgfarm-009bd262ce-dev-ed.develop.lightning.force.com/lightning/setup/EinsteinPromptStudio/0hfgL000006RZRFQ44/edit?c__versionId=3vNgL000000TCqbUAG`. The interface includes a top navigation bar with tabs for 'Setup', 'Home', and 'Object Manager'. Below this, the 'Recipe Generator' tab is active, showing 'Version 1 (Active)'. The main workspace is divided into three panels: 'Preview Settings', 'Resolved Prompt', and 'Generated Response'. The 'Preview Settings' panel on the left contains input fields for 'Ingredients' (meat, daal), 'Dietary Restrictions' (none), 'Meal Type' (lunch), and 'Serving Size' (2). The 'Generated Response' panel on the right displays a JSON object representing a recipe.

```
1 {
2   "recipes": [
3     {
4       "recipe_number": "1",
5       "servings": "2",
6       "recipe_name": "Spiced Meat and Daal Curry",
7       "description": "A hearty curry combining tender meat with flavorful daal, perfect for a satisfying lunch.",
8       "prep_time": "15",
9       "cook_time": "30",
10      "total_time": "45",
11      "difficulty": "Medium",
12      "ingredients": [
13        {
14          "amount": "300",
15          "unit": "grams",
16          "name": "meat (chicken or lamb)"
17        },
18        {
19          "amount": "200",
20          "unit": "grams",
21          "name": "daal (lentils)"
22        }
23      ]
24    }
25  ]
26 }
```

Phase 5: Apex Programming (Developer)

- **Creating a Apex Class:**

Apex class name: AI Recipe Controller

Writing Program in LWC

```
public with sharing class AIRecipeController {
    @AuraEnabled
    public static String generateAIRecipes(String ingredients,
String dietaryRestrictions, String mealType, String servings) {
        try{
            //1. First will create a Map to hold the input
Values

            Map<String, ConnectApi.WrappedValue> inputParams =
new Map<String, ConnectApi.WrappedValue>();

            ConnectApi.WrappedValue ingredientsValue = new
ConnectApi.WrappedValue();
            ingredientsValue.value = ingredients;
            ConnectApi.WrappedValue dietaryRestrictionsValue =
new ConnectApi.WrappedValue();
            dietaryRestrictionsValue.value =
dietaryRestrictions;
            ConnectApi.WrappedValue mealTypeValue = new
ConnectApi.WrappedValue();
            mealTypeValue.value = mealType;
            ConnectApi.WrappedValue servingsValue = new
ConnectApi.WrappedValue();
            servingsValue.value = servings;

            inputParams.put('Input:Ingredients',
ingredientsValue);
            inputParams.put('Input:Dietary_Restrictions',
dietaryRestrictionsValue);
            inputParams.put('Input:Meal_Type', mealTypeValue);
            inputParams.put('Input:Serving_Size', servingsValue);
```



```

        //2. Will create the configuration which is required
for invoking the prompt template
        ConnectApi.EinsteinPromptTemplateGenerationsInput
executeTemplateInput = new
ConnectApi.EinsteinPromptTemplateGenerationsInput();
        executeTemplateInput.additionalConfig = new
ConnectApi.EinsteinLlmAdditionalConfigInput();
        executeTemplateInput.additionalConfig.applicationName
= 'PromptBuilderPreview';
        executeTemplateInput.isPreview = false;
        executeTemplateInput.inputParams = inputParams;

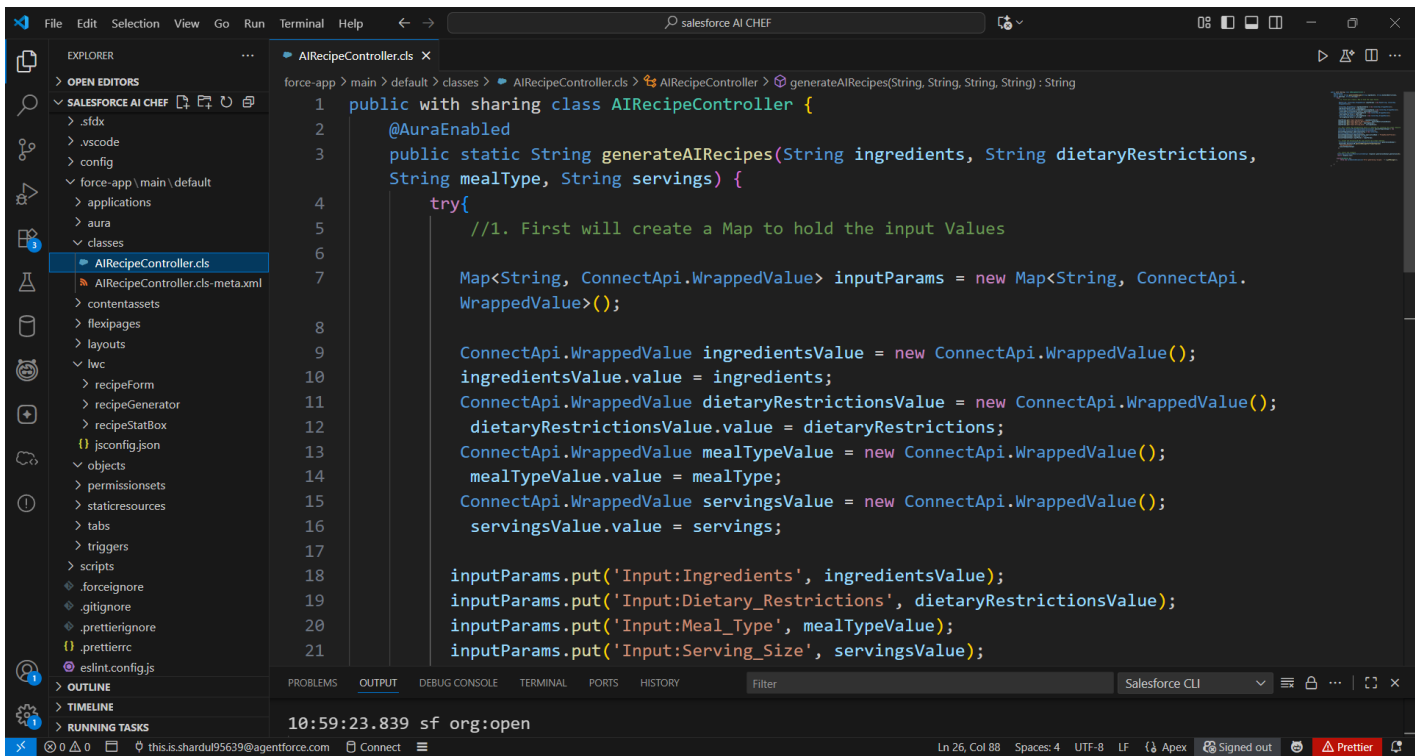
        //3. Invoke the EinsteinLLM APi and execute the
prompt template
        ConnectApi.EinsteinPromptTemplateGenerationsRepresen
tation generationsOutput =
ConnectApi.EinsteinLLM.generateMessagesForPromptTemplate(
        'Recipe_Generator',
        executeTemplateInput
        );

        //4. return the response
        ConnectApi.EinsteinLLMGenerationItemOutput response=
generationsOutput.generations[0];
        return response.text;

    }catch(Exception e){
        throw new AuraHandledException('Error generating
recipes: ' + e.getMessage());
    }

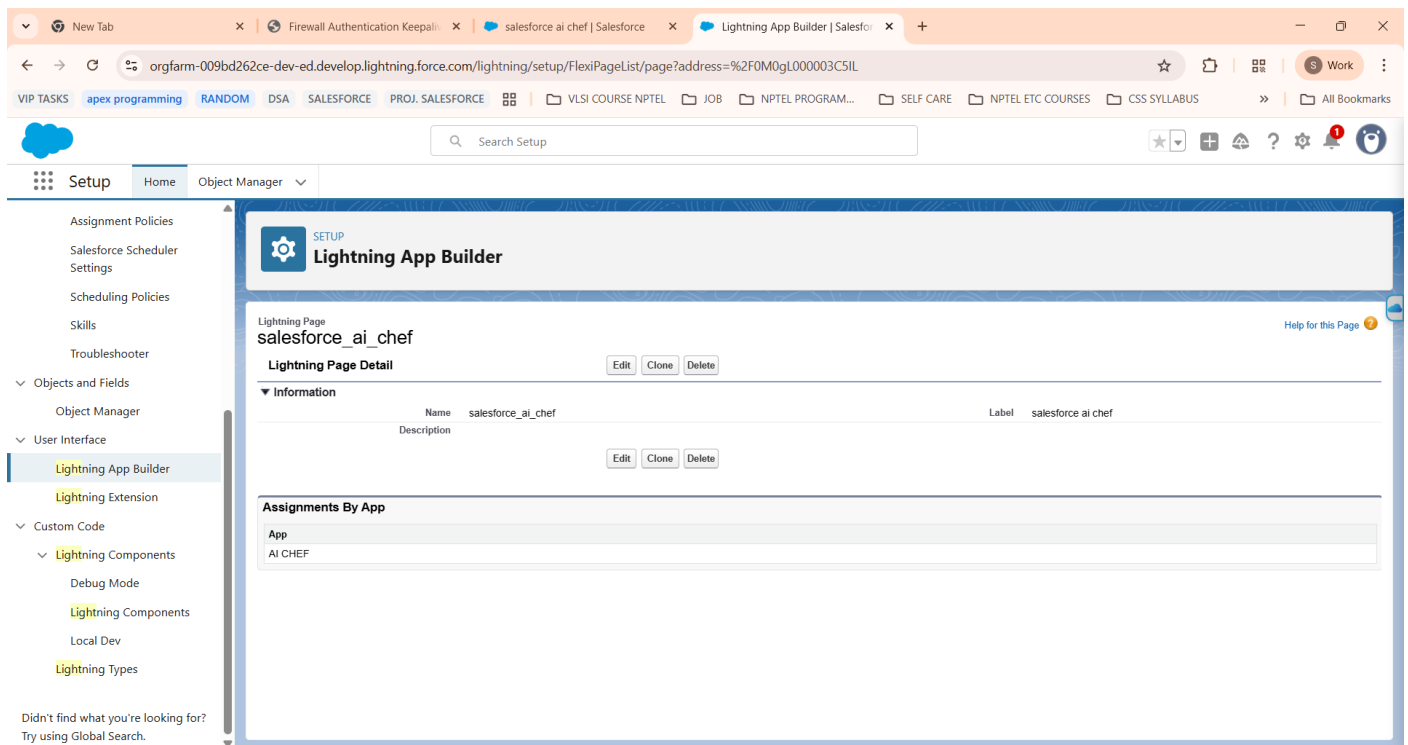
}
}

```



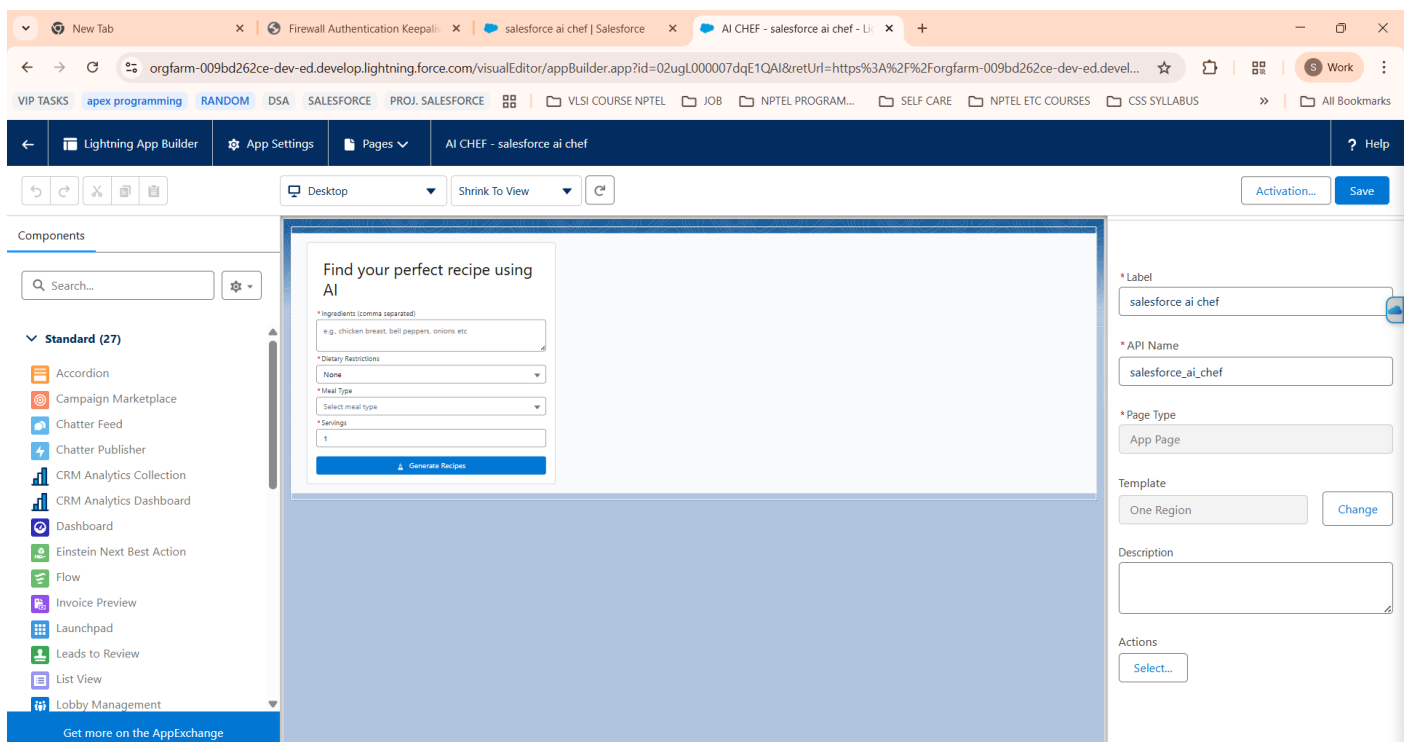
Phase 6: User Interface Development

- Lightning App Builder



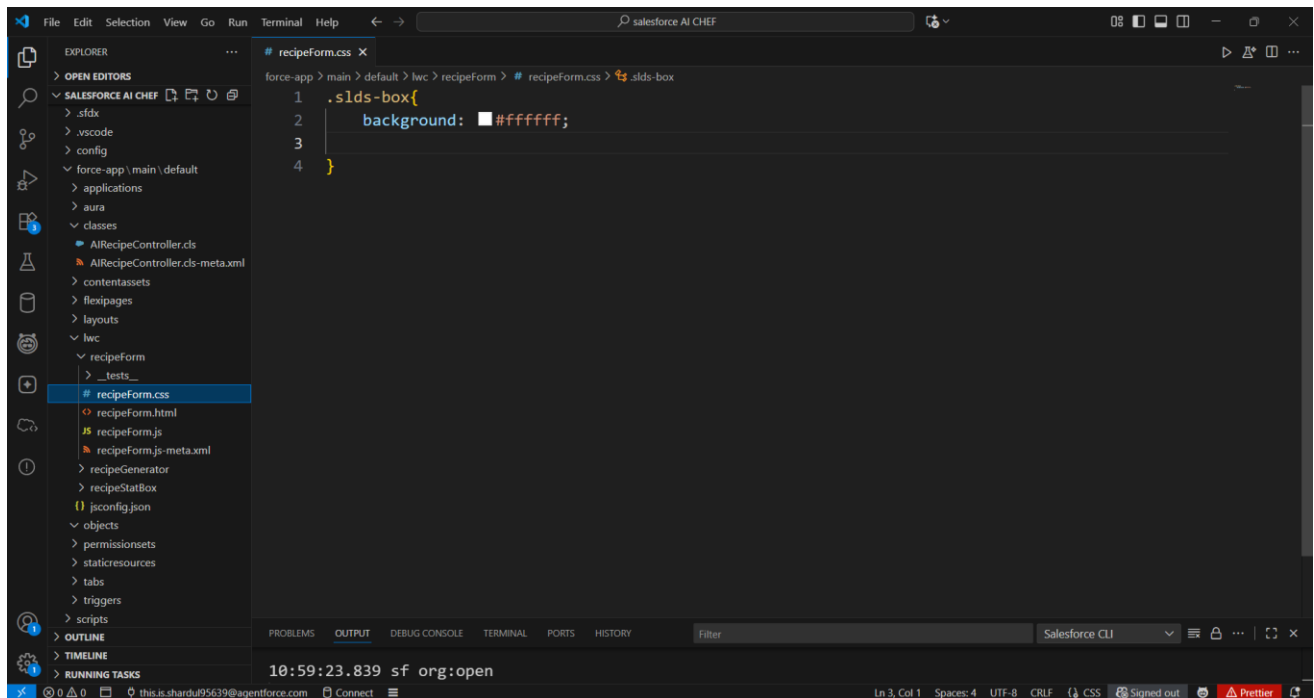
Name: salesforce ai chef

Assignments By App: AI CHEF



- Codes of Custom LWC
 - Recipe Form
- css code of recipe Form

```
.slds-box{  
  background: #ffffff;  
}
```



HTML Code for recipe form

```
<template>  
  <form class="slds-box slds-p-around_medium">  
    <h1 class="slds-text-heading_large slds-p-around_small">Find your perfect recipe using AI</h1>  
    <lightning-textarea  
      label="Ingredients (comma separated)"  
      placeholder="e.g., chicken breast, bell peppers,  
onions etc"  
      required  
      value={formData.ingredients}  
      name="ingredients"  
      onchange={handleChange}></lightning-textarea>
```

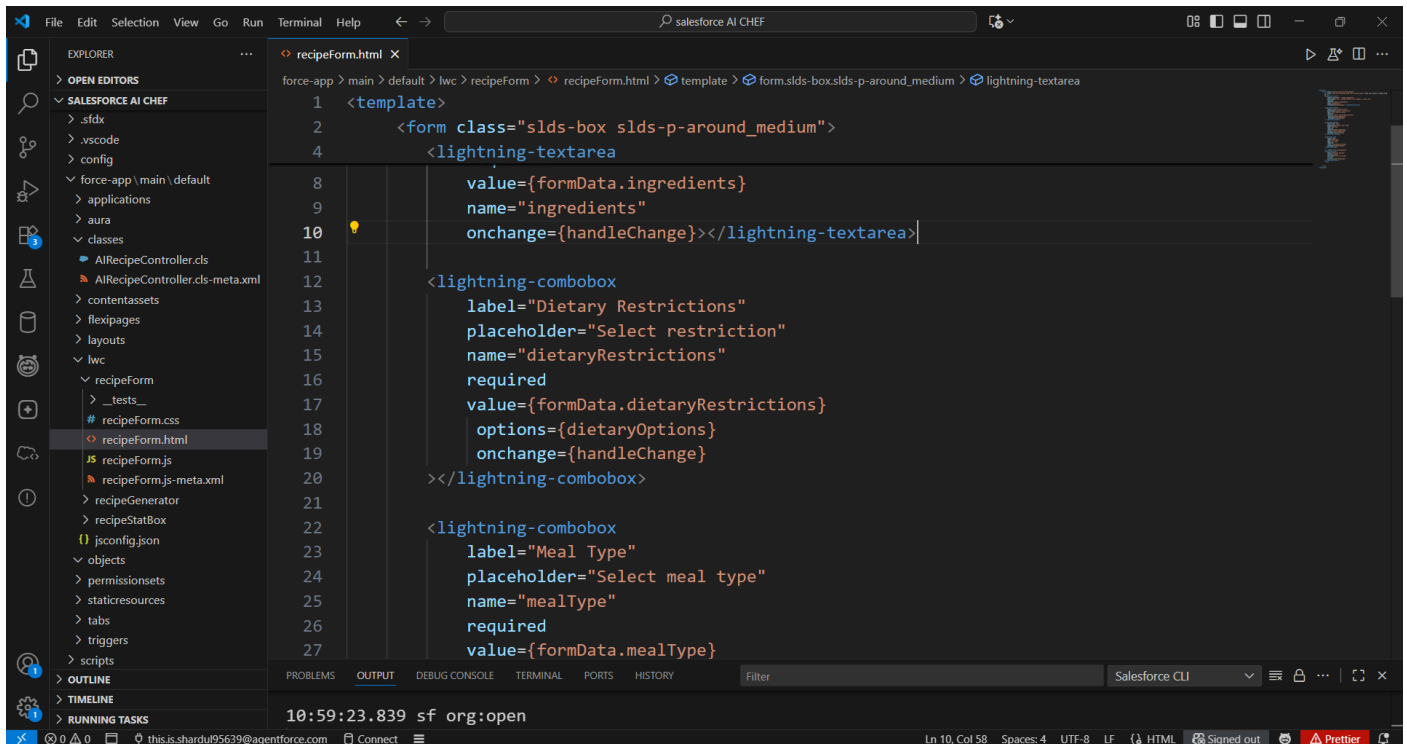
```
<lightning-combobox
  label="Dietary Restrictions"
  placeholder="Select restriction"
  name="dietaryRestrictions"
  required
  value={formData.dietaryRestrictions}
  options={dietaryOptions}
  onChange={handleChange}
></lightning-combobox>
```

```
<lightning-combobox
  label="Meal Type"
  placeholder="Select meal type"
  name="mealType"
  required
  value={formData.mealType}
  options={mealTypeOptions}
  onChange={handleChange}
></lightning-combobox>
```

```
<lightning-input
  type="number"
  label="Servings"
  name="servings"
  required
  value={formData.servings}
  onChange={handleChange}>
</lightning-input>
```

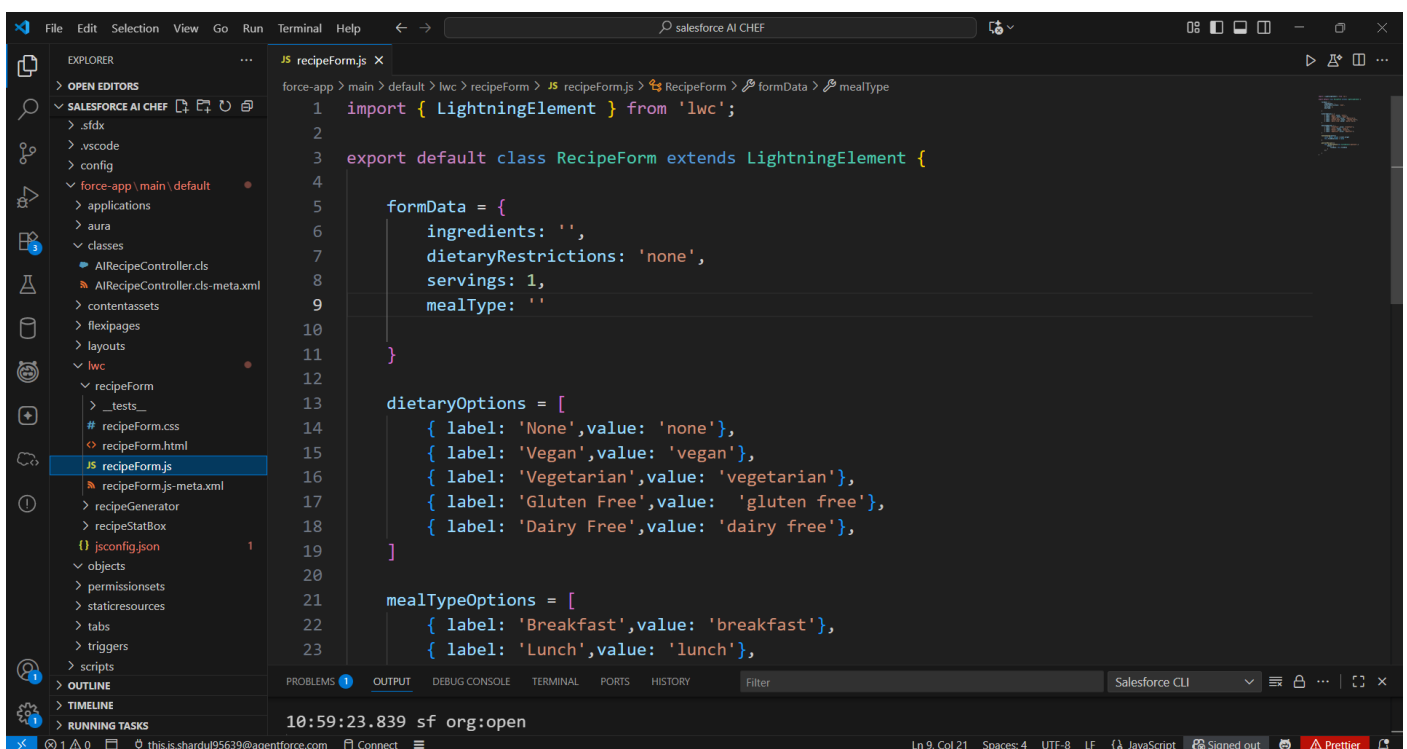
```
<div class="slds-m-top_medium">
<lightning-button
  label="Generate Recipes"
  variant="brand"
  icon-name="utility:recipe"
  stretch
  onclick={generateRecipes}
></lightning-button>
</div>
```

```
</form>
</template>
```



```
1 <template>
2   <form class="slds-box slds-p-around_medium">
4     <lightning-textarea
8       value={formData.ingredients}
9       name="ingredients"
10      onchange={handleChange}></lightning-textarea>
11
12     <lightning-combobox
13       label="Dietary Restrictions"
14       placeholder="Select restriction"
15       name="dietaryRestrictions"
16       required
17       value={formData.dietaryRestrictions}
18       options={dietaryOptions}
19       onchange={handleChange}>
20   </lightning-combobox>
21
22   <lightning-combobox
23     label="Meal Type"
24     placeholder="Select meal type"
25     name="mealType"
26     required
27     value={formData.mealType}
```

JS code for recipe form



```
1 import { LightningElement } from 'lwc';
2
3 export default class RecipeForm extends LightningElement {
4
5   formData = {
6     ingredients: '',
7     dietaryRestrictions: 'none',
8     servings: 1,
9     mealType: ''
10  }
11
12  dietaryOptions = [
13    { label: 'None', value: 'none'},
14    { label: 'Vegan', value: 'vegan'},
15    { label: 'Vegetarian', value: 'vegetarian'},
16    { label: 'Gluten Free', value: 'gluten free'},
17    { label: 'Dairy Free', value: 'dairy free'},
18  ]
19
20  mealTypeOptions = [
21    { label: 'Breakfast', value: 'breakfast'},
22    { label: 'Lunch', value: 'lunch'},
23  ]
24 }
```

```
import { LightningElement } from 'lwc';

export default class RecipeForm extends LightningElement {

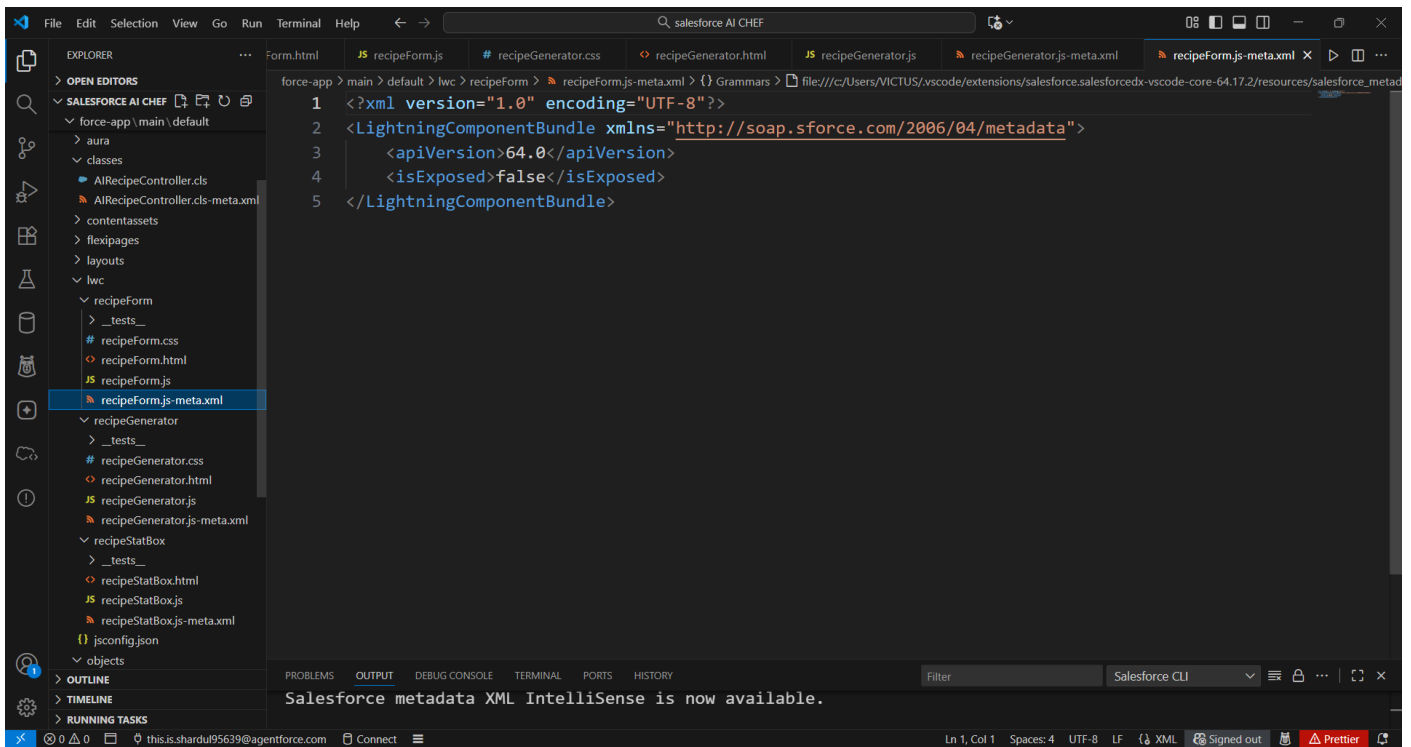
  formData = {
    ingredients: '',
    dietaryRestrictions: 'none',
    servings: 1,
    mealType: ''
  }

  dietaryOptions = [
    { label: 'None', value: 'none' },
    { label: 'Vegan', value: 'vegan' },
    { label: 'Vegetarian', value: 'vegetarian' },
    { label: 'Gluten Free', value: 'gluten free' },
    { label: 'Dairy Free', value: 'dairy free' },
  ]

  mealTypeOptions = [
    { label: 'Breakfast', value: 'breakfast' },
    { label: 'Lunch', value: 'lunch' },
    { label: 'Dinner', value: 'dinner' },
    { label: 'Appetizer', value: 'appetizer' },
  ]

  handleChange(event) {
    const { name, value } = event.target
    this.formData[name] = value
  }

  generateRecipes() {
    this.dispatchEvent(new CustomEvent('generate', {
      detail: {
        formData: this.formData
      }
    })))
  }
}
```



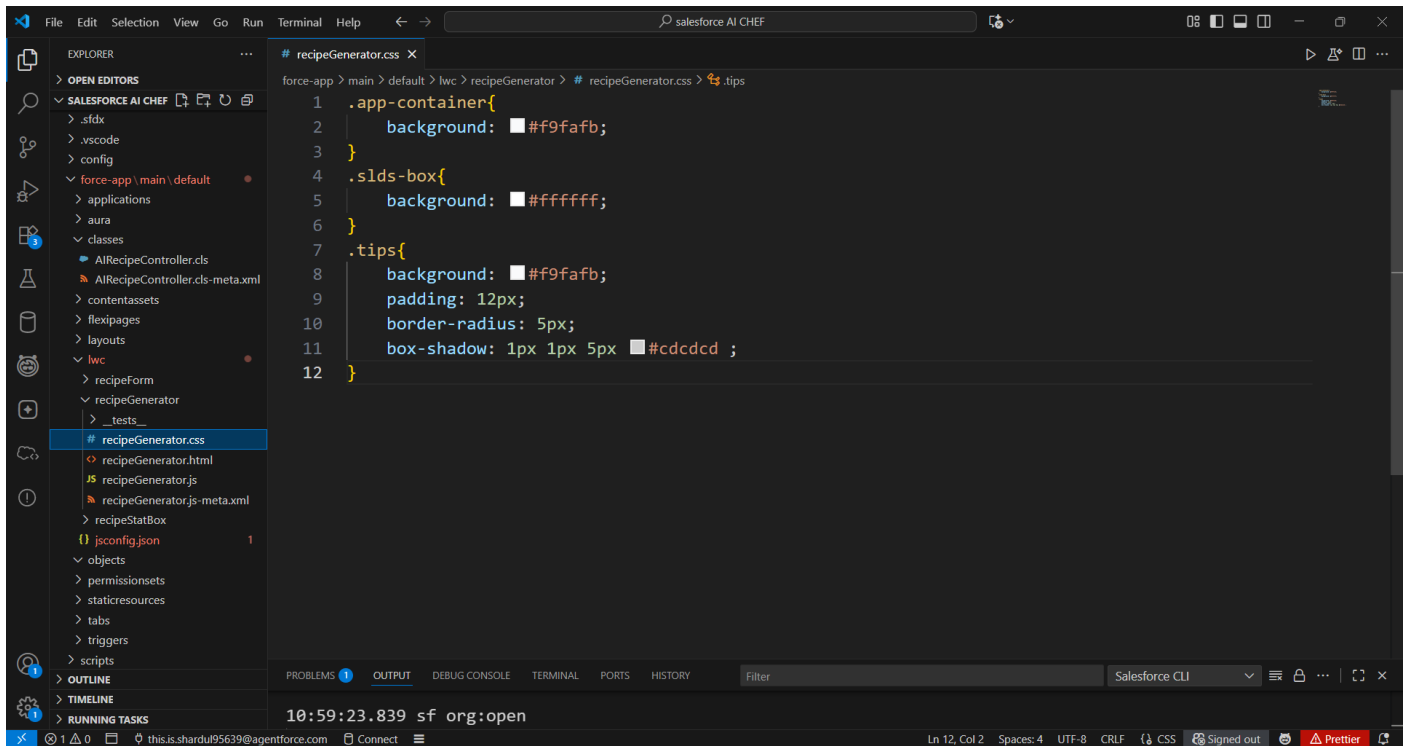
➤ Recipe Generator

CSS for Recipe Generator

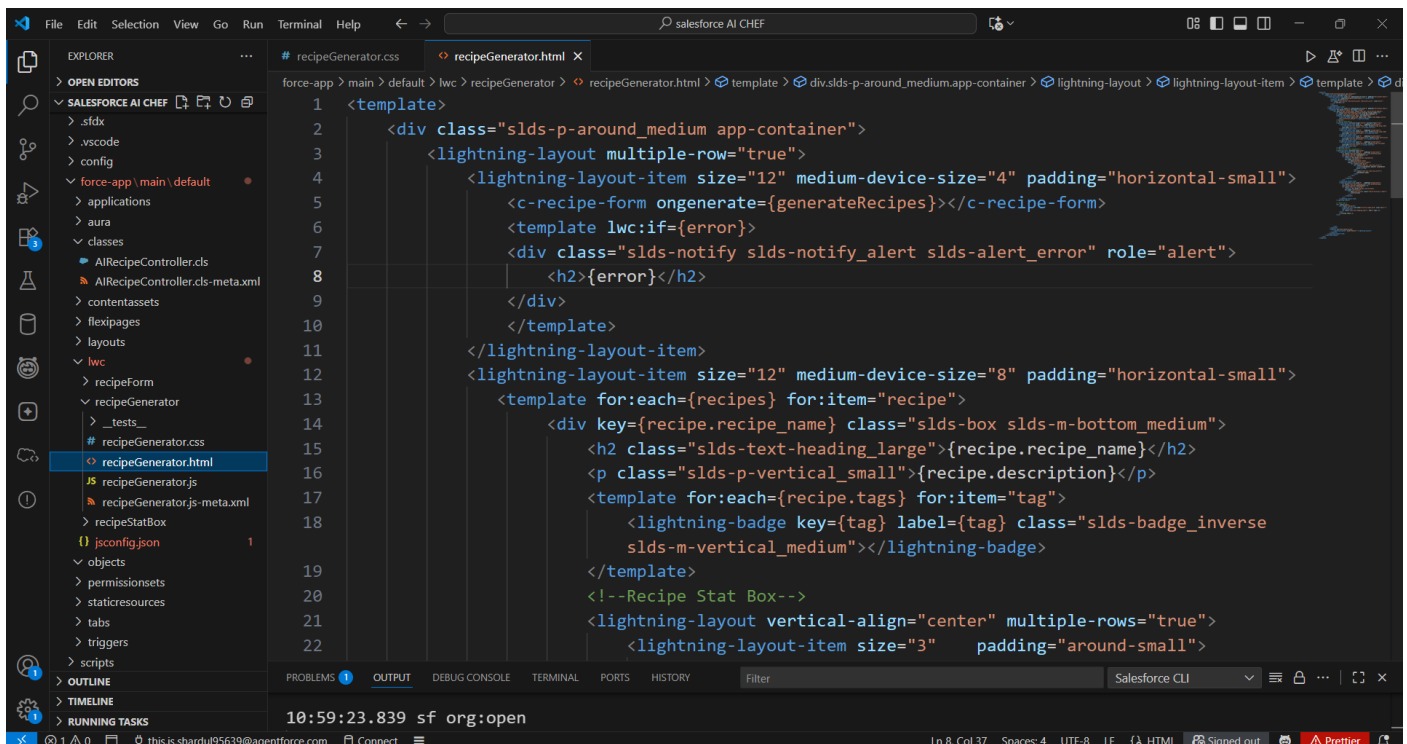
```
.app-container{
  background: #f9fafb;
}

.slds-box{
  background: #ffffff;
}

.tips{
  background: #f9fafb;
  padding: 12px;
  border-radius: 5px;
  box-shadow: 1px 1px 5px #cdcdcd ;
}
```

HTML Code For Recipe Generator



```
<template>
  <div class="slds-p-around_medium app-container">
    <lightning-layout multiple-row="true">
```

```

        <lightning-layout-item size="12" medium-device-
size="4" padding="horizontal-small">
            <c-recipe-form
ongenerate={generateRecipes}></c-recipe-form>
            <template lwc:if={error}>
                <div class="slds-notify slds-notify_alert slds-
alert_error" role="alert">
                    <h2>{error}</h2>
                </div>
            </template>
        </lightning-layout-item>
        <lightning-layout-item size="12" medium-device-
size="8" padding="horizontal-small">
            <template for:each={recipes} for:item="recipe">
                <div key={recipe.recipe_name} class="slds-
box slds-m-bottom_medium">
                    <h2 class="slds-text-
heading_large">{recipe.recipe_name}</h2>
                    <p class="slds-p-
vertical_small">{recipe.description}</p>
                    <template for:each={recipe.tags}
for:item="tag">
                        <lightning-badge key={tag}
label={tag} class="slds-badge_inverse slds-m-
vertical_medium"></lightning-badge>
                    </template>
                    <!--Recipe Stat Box-->
                    <lightning-layout vertical-
align="center" multiple-rows="true">
                        <lightning-layout-item
size="3" padding="around-small">
                            <c-recipe-stat-box title="Prep
Time" value={recipe.prep_time} icon-name="utility:clock"></c-
recipe-stat-box>
                        </lightning-layout-item>
                        <lightning-layout-item
size="3" padding="around-small">

```

```

                <c-recipe-stat-box title="Cook
Time" value={recipe.cook_time} icon-
name="utility:real_time"></c-recipe-stat-box>
            </lightning-layout-item>
            <lightning-layout-item
size="3"    padding="around-small">
                <c-recipe-stat-box
title="Difficulty" value={recipe.difficulty} icon-
name="utility:data_model"></c-recipe-stat-box>
            </lightning-layout-item>
            <lightning-layout-item
size="3"    padding="around-small">
                <c-recipe-stat-box
title="Servings" value={recipe.servings} icon-
name="utility:user"></c-recipe-stat-box>
            </lightning-layout-item>
        </lightning-layout>
        <!--Instructions and ingredients-->
        <lightning-layout multiple-rows="true">
            <lightning-layout-item
size="5"    padding="around-small">
                <h3 class="slds-text-
heading_small slds-p-bottom_small slds-
border_bottom">Ingredients</h3>
                <ul class="slds-list_dotted">
                    <template
for:each={recipe.ingredients} for:item="ingredient">
                        <li
key={ingredient.name} class="slds-p-vertical_x-small">
                            <div class="slds-
grid">
                                <div
class="slds-col slds-size_2-of-4">
                                    <strong>{in
gredient.amount} {ingredient.unit}</strong>
                                </div>
                                <div
class="slds-col slds-size_2-of-4">

```

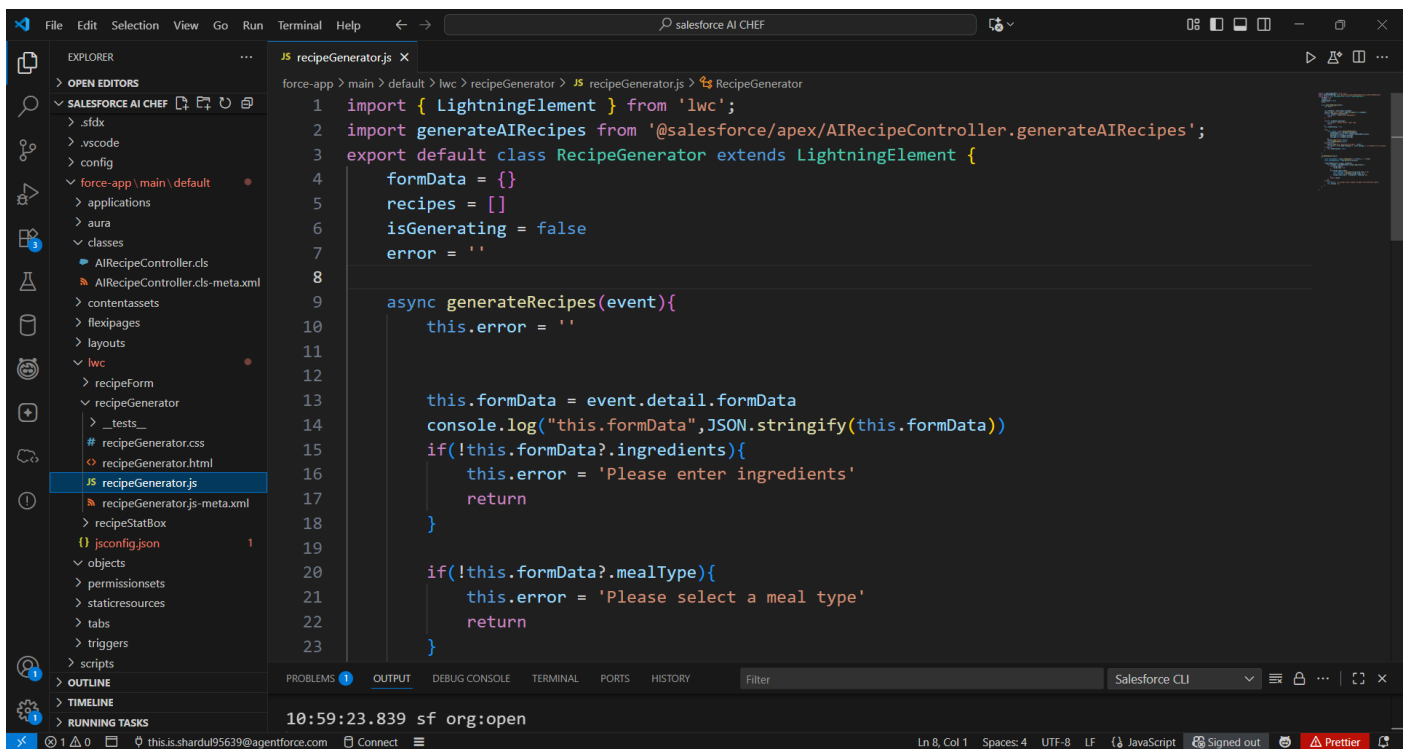


```

        </div>
    </template>
    <template lwc:if={isGenerating}>
        <lightning-spinner
size="medium"></lightning-spinner>
    </template>
</lightning-layout-item>
</lightning-layout>
</div>
</template>

```

JS Code for Recipe Generator



The screenshot shows the VS Code editor interface with the 'recipeGenerator.js' file open. The Explorer sidebar on the left shows the project structure, including the 'lwc' directory and the 'recipeGenerator' component. The main editor area displays the following JavaScript code:

```

1  import { LightningElement } from 'lwc';
2  import generateAIRecipes from '@salesforce/apex/AIRecipeController.generateAIRecipes';
3  export default class RecipeGenerator extends LightningElement {
4      formData = {}
5      recipes = []
6      isGenerating = false
7      error = ''
8
9
10     async generateRecipes(event){
11         this.error = ''
12
13         this.formData = event.detail.formData
14         console.log("this.formData", JSON.stringify(this.formData))
15         if(!this.formData?.ingredients){
16             this.error = 'Please enter ingredients'
17             return
18         }
19
20         if(!this.formData?.mealType){
21             this.error = 'Please select a meal type'
22             return
23         }
24     }
25 }

```

The status bar at the bottom indicates the file is at line 8, column 1, with 4 spaces, in UTF-8 encoding, using the JavaScript language. The user is signed out and the Prettier extension is active.

```

import { LightningElement } from 'lwc';
import generateAIRecipes from
 '@salesforce/apex/AIRecipeController.generateAIRecipes';
export default class RecipeGenerator extends LightningElement {
    formData = {}
    recipes = []

```

```

isGenerating = false
error = ''

async generateRecipes(event){
  this.error = ''

  this.formData = event.detail.formData
  console.log("this.formData",JSON.stringify(this.formData))

  if(!this.formData?.ingredients){
    this.error = 'Please enter ingredients'
    return
  }

  if(!this.formData?.mealType){
    this.error = 'Please select a meal type'
    return
  }
  this.isGenerating = true

  try{
    const result = await generateAIRecipes({
      ingredients:this.formData.ingredients,
      dietaryRestrictions:this.formData.dietaryRestrictions,
      mealType:this.formData.mealType,
      servings:this.formData.servings
    })
    console.log("result",result)
    this.formatResponse(result)
  }catch(error){
    console.log("Error generating recipes: ",error)
    this.error = error.body?.message || error.message
    || 'an unknown error occured';
  } finally{
    this.isGenerating = false
  }
}

```

```

    }

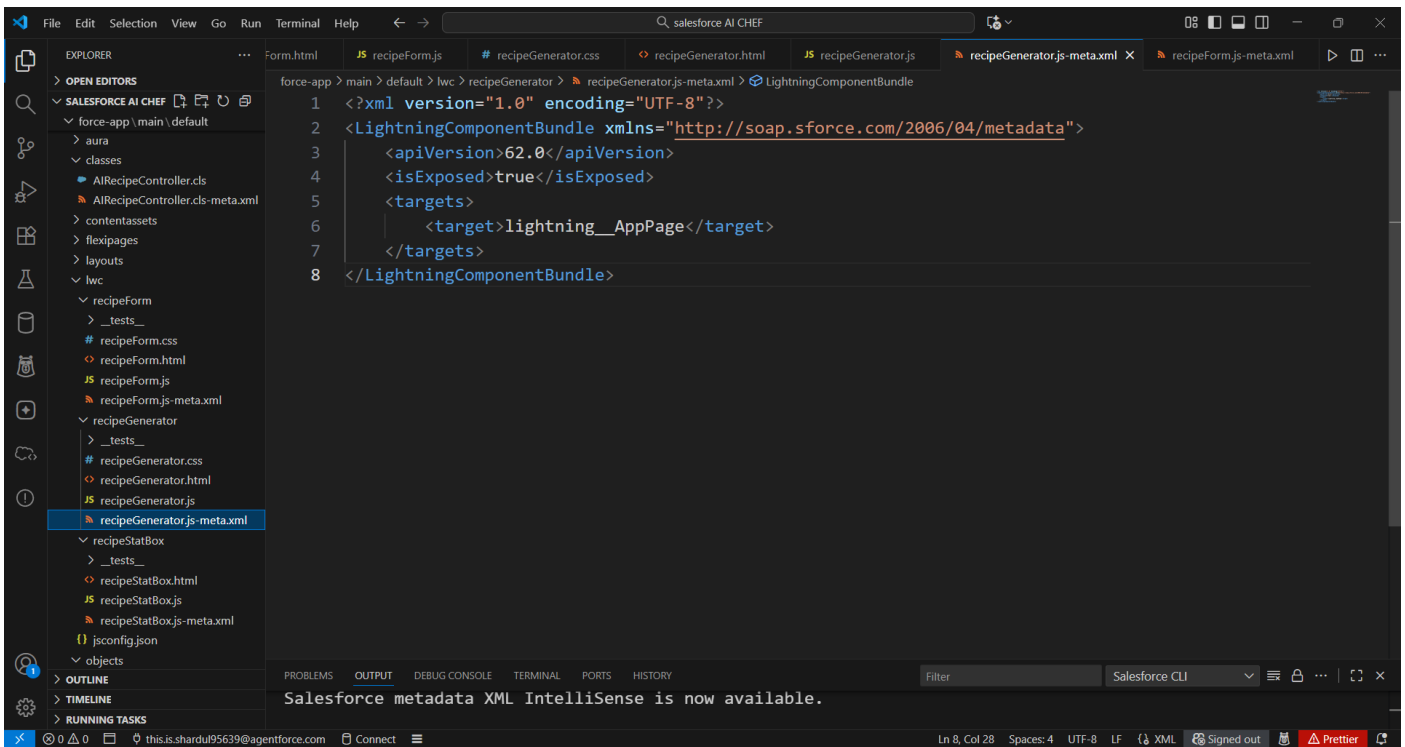
    formatResponse(result)
    {
        const correctJson = result.replaceAll(/\n\u00A0]/g,
        '').trim()
        const parsedResponse = JSON.parse(correctJson)

        if(parsedResponse?.recipes?.length>0){
            this.recipes = parsedResponse.recipes.map(recipe=>{
                if(!recipe.tags){
                    recipe.tags = []
                }
                if(!recipe.total_time){
                    const prepTime = parseInt(recipe.prep_time)
                    || 0;

                    const cookTime = parseInt(recipe.cook) ||
                    0;

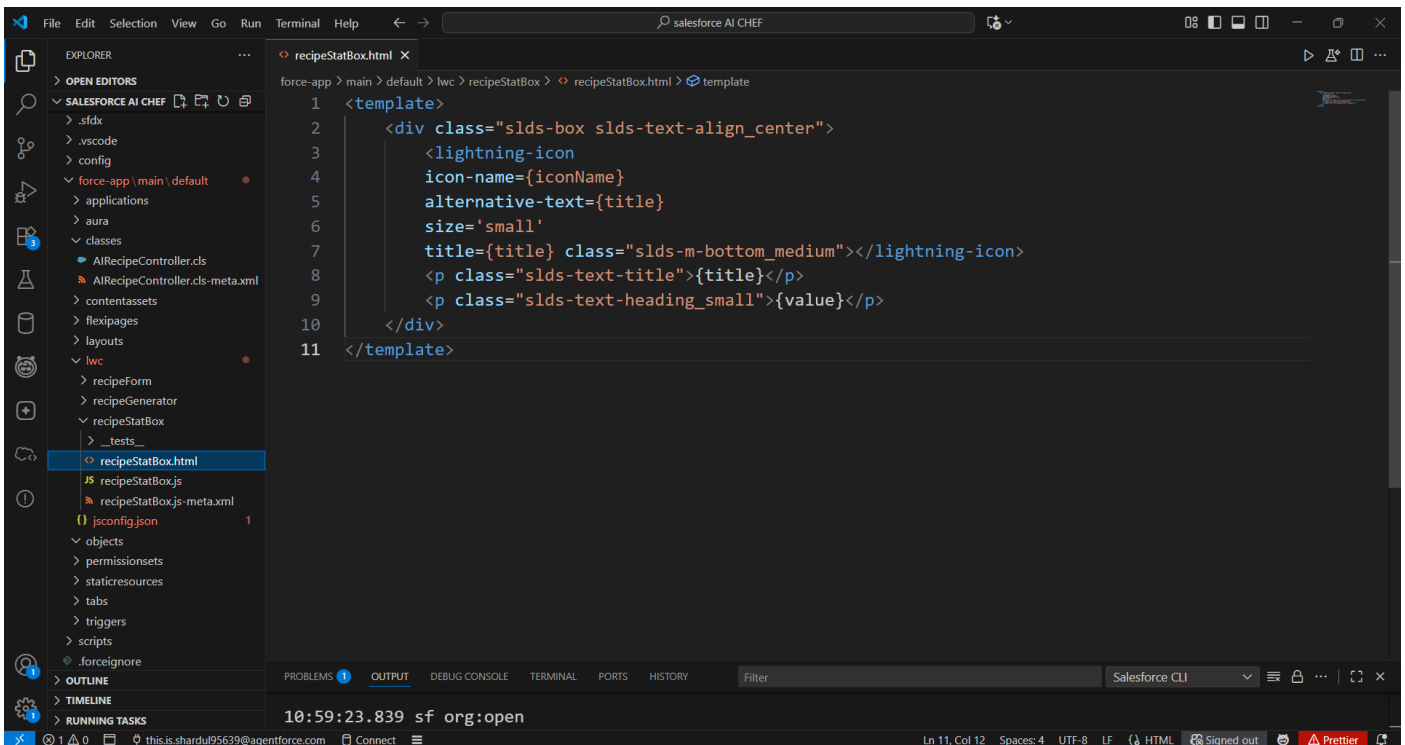
                    recipe.total_time = `${prepTime + cookTime}
                    min`
                }
                return recipe
            })
        } else {
            this.error = 'no recipes found. please try again
            with different inputs.'
            this.recipes = []
        }
    }
}

```

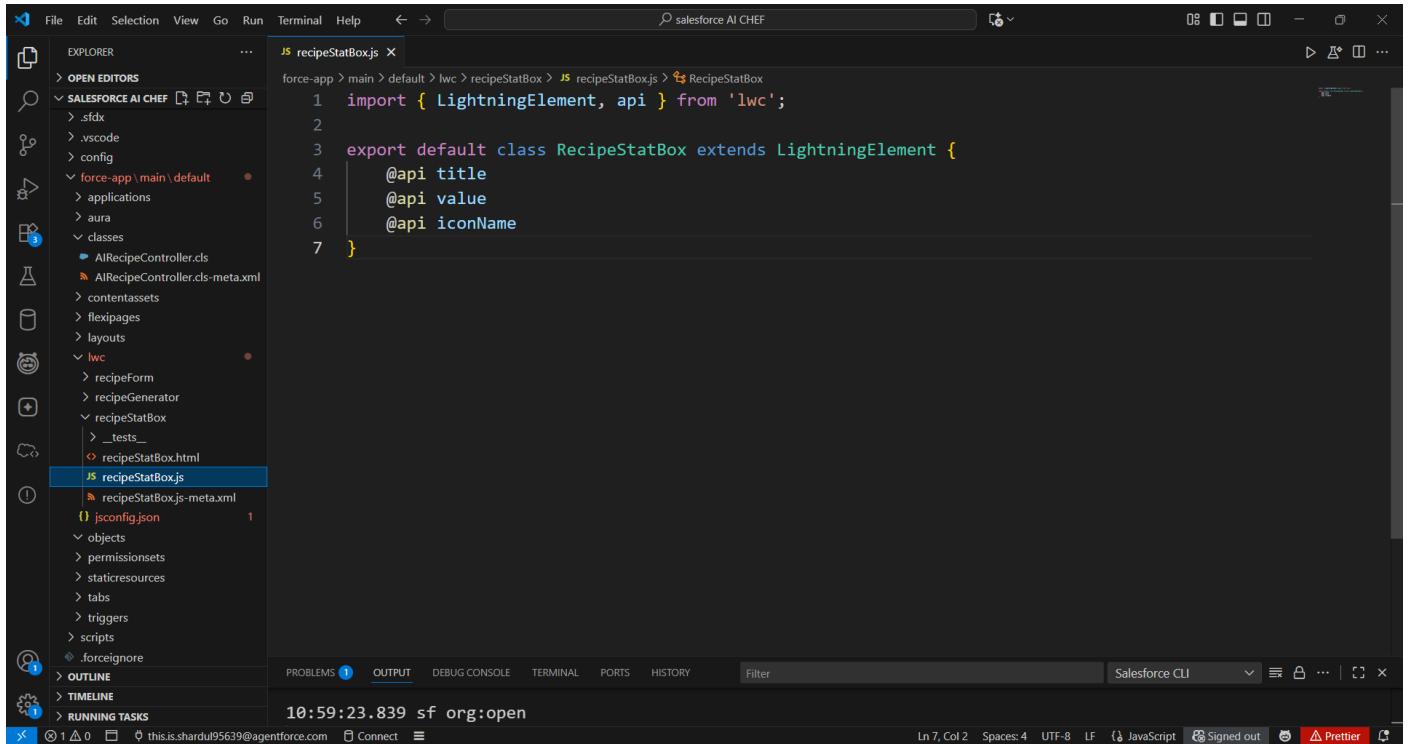


- Recipe Stat Box

➤ Html code for Recipe Stat Box



JS Code For Recipe Stat Box



The screenshot shows the Visual Studio Code editor interface with a Salesforce project. The Explorer sidebar on the left displays the project structure, including the 'recipeStatBox.js' file which is currently selected. The main editor area shows the following JavaScript code:

```
force-app > main > default > lwc > recipeStatBox > JS recipeStatBox.js > RecipeStatBox
1  import { LightningElement, api } from 'lwc';
2
3  export default class RecipeStatBox extends LightningElement {
4      @api title
5      @api value
6      @api iconName
7  }
```

The bottom status bar indicates the current position is Line 7, Column 2, with 4 spaces, UTF-8 encoding, and LF line endings. It also shows the file is a JavaScript file and the user is signed out. The Prettier extension is active.