



# Module IV

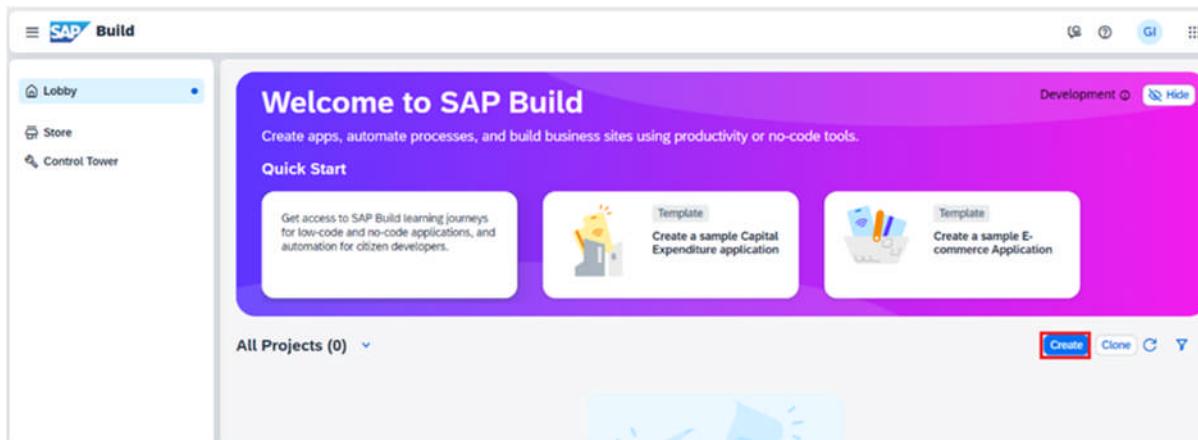
## Foundations of SAP ABAP and Hand Book

### Code with Joule

## Lab:-2 Create a Full-Stack SAP Fiori Application with Joule in SAP Build Code

### 1 Create a New Project Using SAP Build Code

1. Navigate to the SAP Build lobby.
2. Click **Create** to start the creation process



3. Click the **Build an Application** tile.

SAP Build

## What would you like to do?

The image shows three cards side-by-side, each with a title, a brief description, and a 'Learn more' button. The first card, 'Build an Application', has a red border around its icon and title.

Icon	Title	Description	Action
	Build an Application	Use productivity tools to create and extend business applications and services.	<a href="#">Learn more</a>
	Build an Automated Process	Use SAP Build Process Automation to automate processes and tasks with drag-and-drop simplicity.	<a href="#">Learn more</a>
	Build a Business Site	Create and customize business sites with an easy no-code experience.	<a href="#">Learn more</a>

[Cancel](#)

- 4.Click the **SAP Build Code** tile to develop your project in SAP Business Application Studio, the SAP Build Code development environment, leveraging the capabilities of the services included in SAP Build Code.

[Build an Application](#)

## How do you want to build your app?

The image shows two cards side-by-side, each with a title, a brief description, and a 'Learn more' button. The second card, 'SAP Build Code', has a red border around its icon and title.

Icon	Title	Description	Action
	SAP Build Apps	Use visual tools to create web and mobile applications, including cloud-based backends for data and business logic.	<a href="#">Learn more</a>
	SAP Build Code	Use graphical and code editors to develop, test, and deploy SAP business applications and extensions such as SAP Fiori, Mobile, and CAP.	<a href="#">Learn more</a>

[Cancel](#)

- 5.Click the **Full-Stack Application** tile.

SAP Build Code

## Select the development configuration for your scenario



### SAP Fiori Application

Create SAP Fiori freestyle or SAP Fiori elements applications.

[Learn More](#)



### Full-Stack Application

Easily develop, extend, and deploy a full-stack application with a mobile or desktop UI.

[Learn More](#)

[Back](#)

[Additional Application Types ⓘ](#)

[Cancel](#)

6. Enter a name for your project.
7. Select the dev space where you want the project to reside.
8. Click Create.

Create a Full-Stack project

## Give your project a name

Project Name:\*

My\_New\_Project

Description:

Select Dev Space: ⓘ

You can run up to two dev spaces at a time. For more options, go to the [Dev Space Manager](#)

[Back](#)

[Create](#)

[Cancel](#)

9. You can see the project being created in the Project table of the lobby.

The creation of the project may take a few moments.

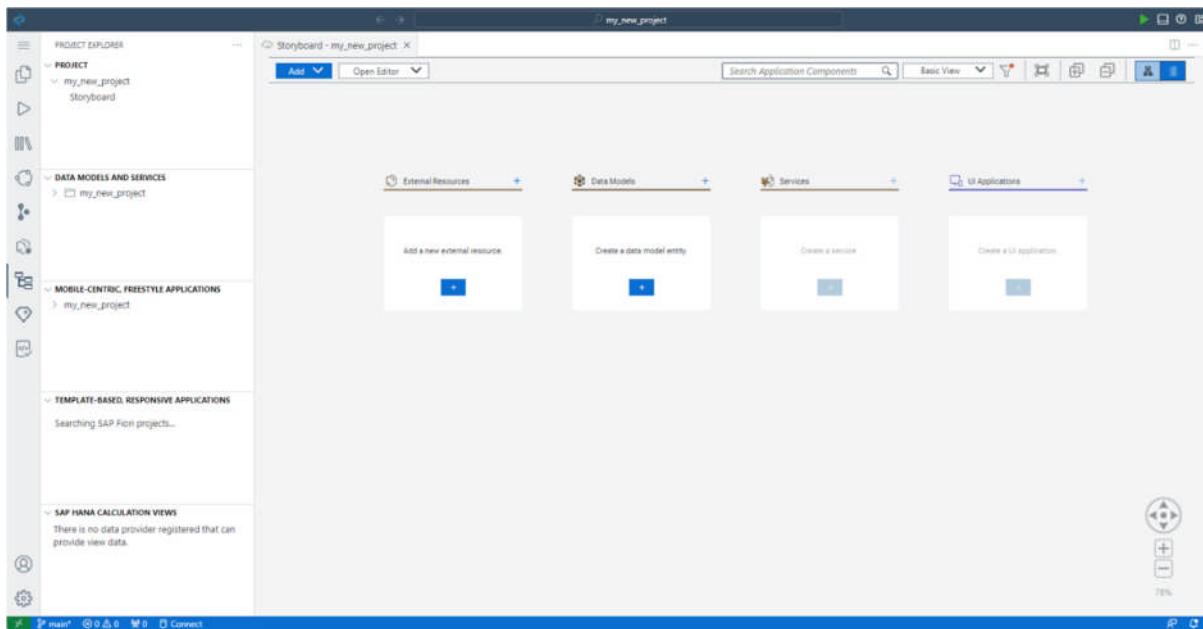
The screenshot shows the SAP Build interface. At the top, there's a purple header bar with the SAP Build logo and navigation links like 'Lobby', 'Store', and 'Control Tower'. Below the header is a 'Welcome to SAP Build' banner with the subtext 'Create apps, automate processes, and build business sites using productivity or no-code tools.' A 'Quick Start' section features three cards: 'Access our SAP Build Learning Journeys', 'Create a sample Capital Expenditure application', and 'Create a sample E-commerce Application'. The main area is titled 'All Projects (1)' and contains a table with one row. The table columns are 'Name', 'Versions', 'Type', 'Last Accessed', 'Members', and 'Options'. The single project listed is 'My\_New\_Project' (with a small blue icon). The 'Type' column indicates it's a 'Build Code Full-Stack Application'. The 'Last Accessed' column shows 'Dec 4, 12:46 pm'. The 'Members' column lists 'Me'. The 'Options' column has a 'Creating...' button. A search bar at the top right says 'Search by Project name and description'.

Name	Versions	Type	Last Accessed	Members	Options
My_New_Project		Build Code Full-Stack Application	Dec 4, 12:46 pm	Me	<span>Creating...</span>

10. After you see a message stating that the project has been created successfully, click the project to open it.

This screenshot is identical to the one above, showing the SAP Build interface with a single project named 'My\_New\_Project'. However, a new message box at the bottom center states 'Project My\_New\_Project has been created successfully.' This message is enclosed in a light gray box with a thin border.

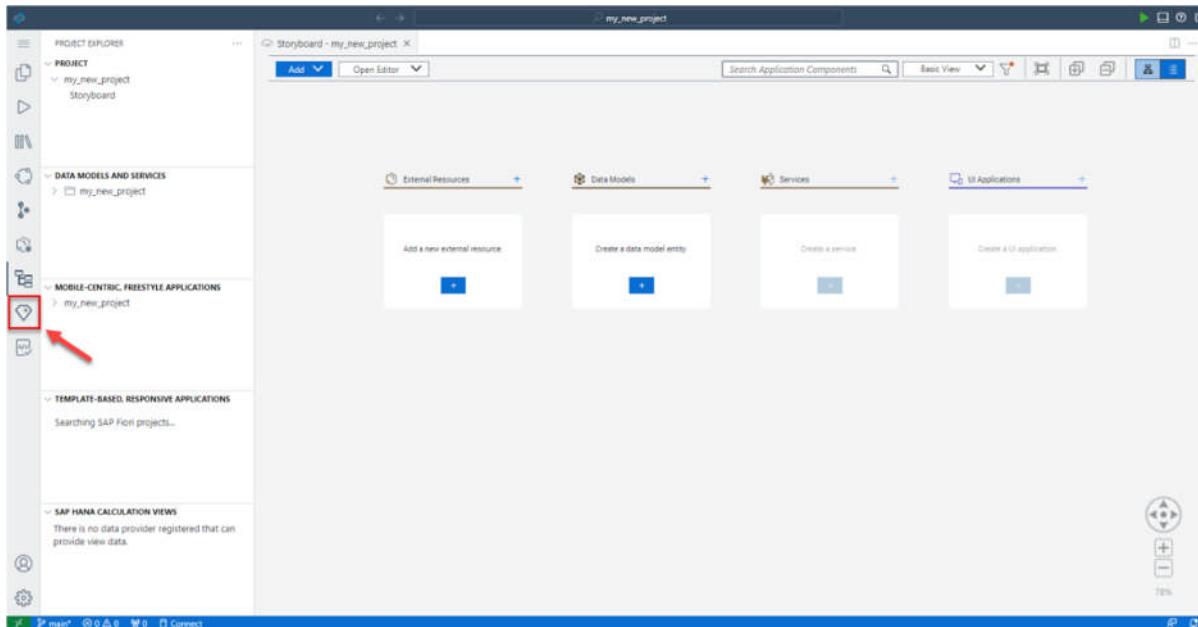
The project opens in SAP Business Application Studio, the SAP Build Code development environment.



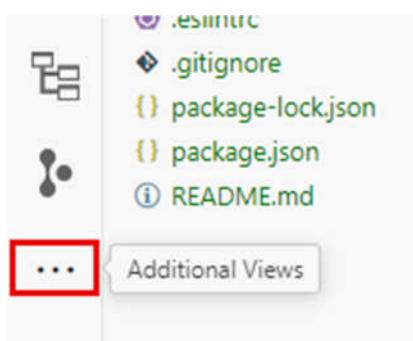
## 2 Create Data Entities with Joule

In SAP Business Application Studio, the SAP Build Code development environment, open the digital assistant, Joule, from the activity bar.

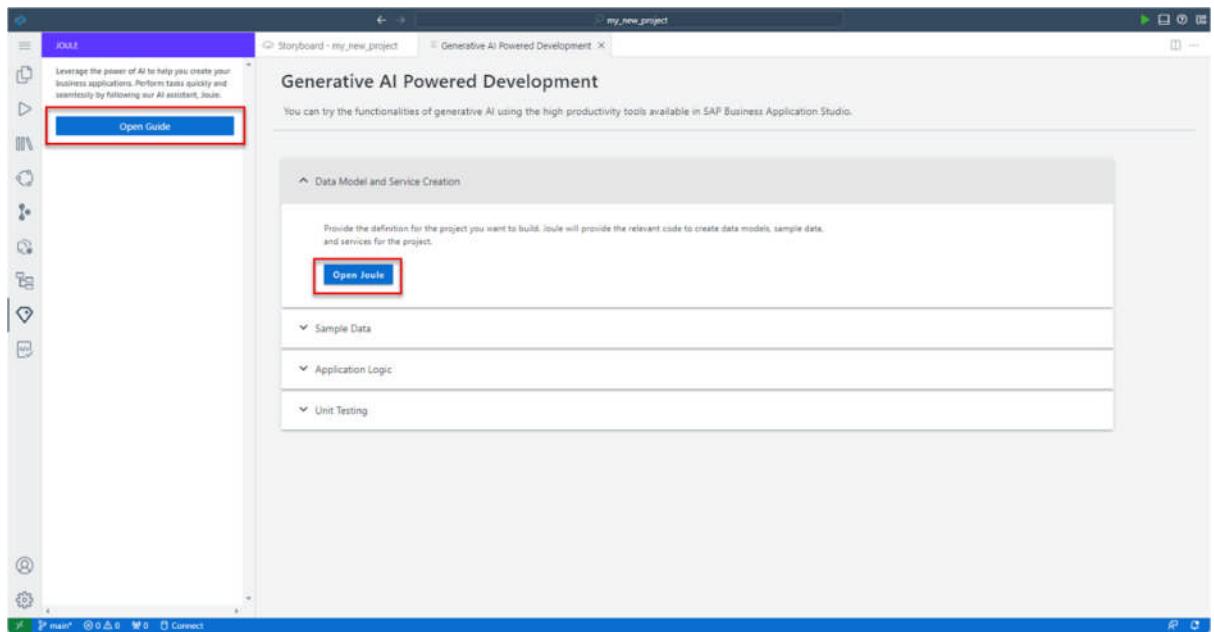
1. In SAP Business Application Studio, the SAP Build Code development environment, open the digital assistant, Joule, from the activity bar.



Note: If you do not see the icon, click Additional Views and select **Joule** from the list.



2. Click **Open Guide**.
3. Expand the **Data Model and Service Creation** section, and click **Open Joule**.



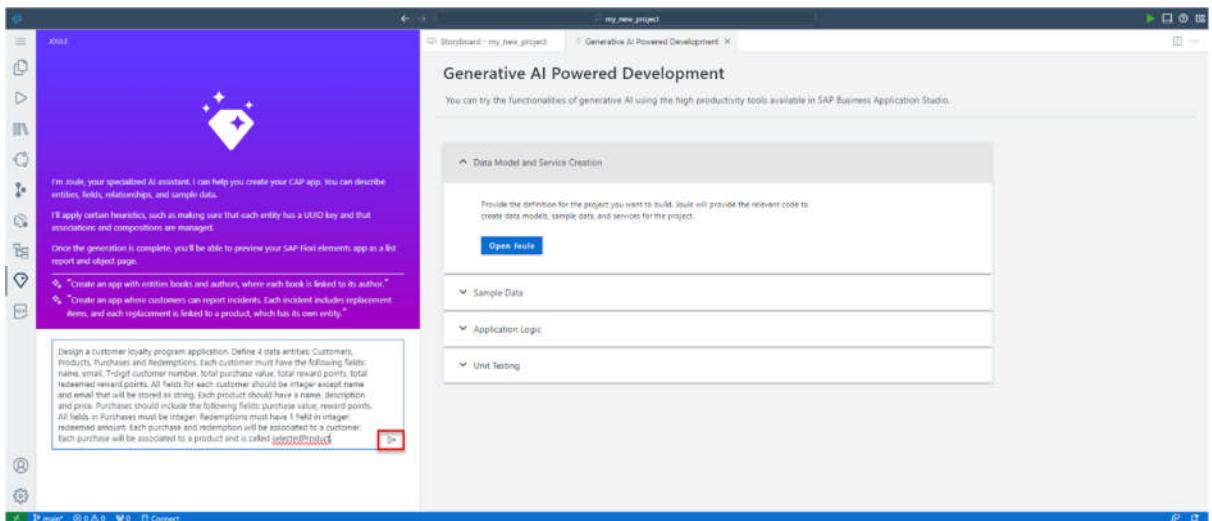
Copy the prompt below.

**Design a customer loyalty program application.**  
**Define 4 data entities: Customers, Products, Purchases and Redemptions.**  
**Each customer must have the following fields: name, email, 7-digit customer number, total purchase value, total reward points, total redeemed reward points.**  
**All fields for each customer should be integer except name and email that will be stored as string.**  
**Each product should have a name, description and price.**  
**Purchases should include the following fields: purchase value, reward points.**  
**All fields in Purchases must be integer.**  
**Redemptions must have 1 field in integer: redeemed amount.**

4. Paste the code in the text field, and click the arrow

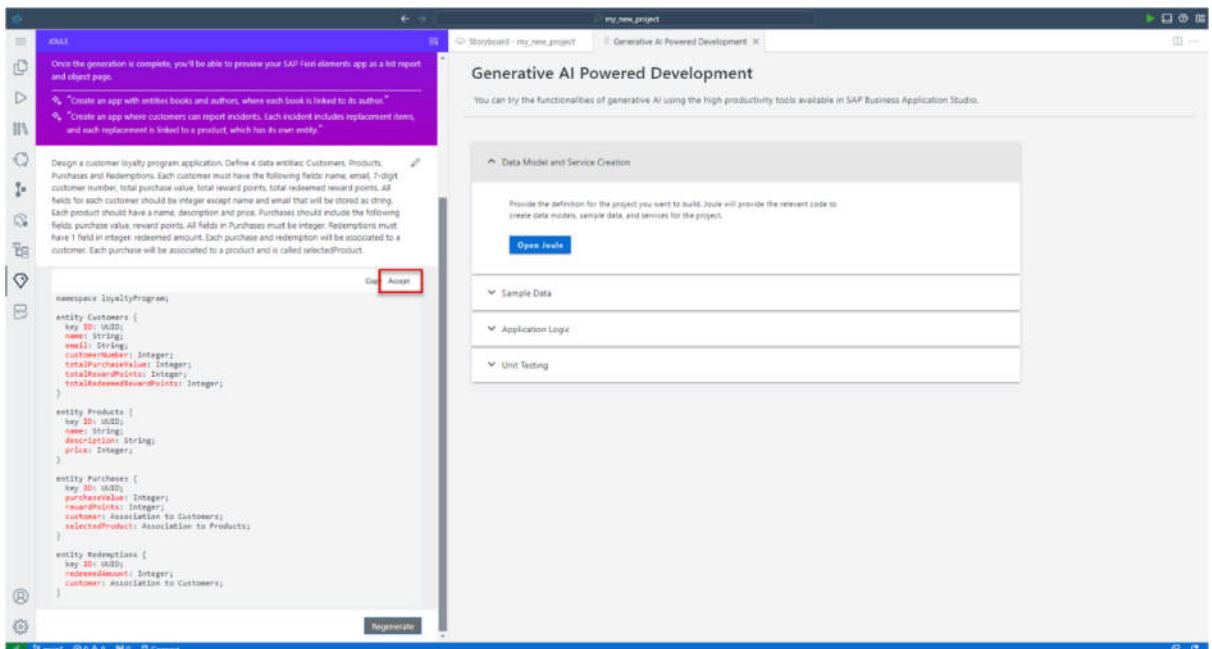


to send the prompt to Joule.



The code is generated and is displayed below your prompt.

## 6. Accept the code.



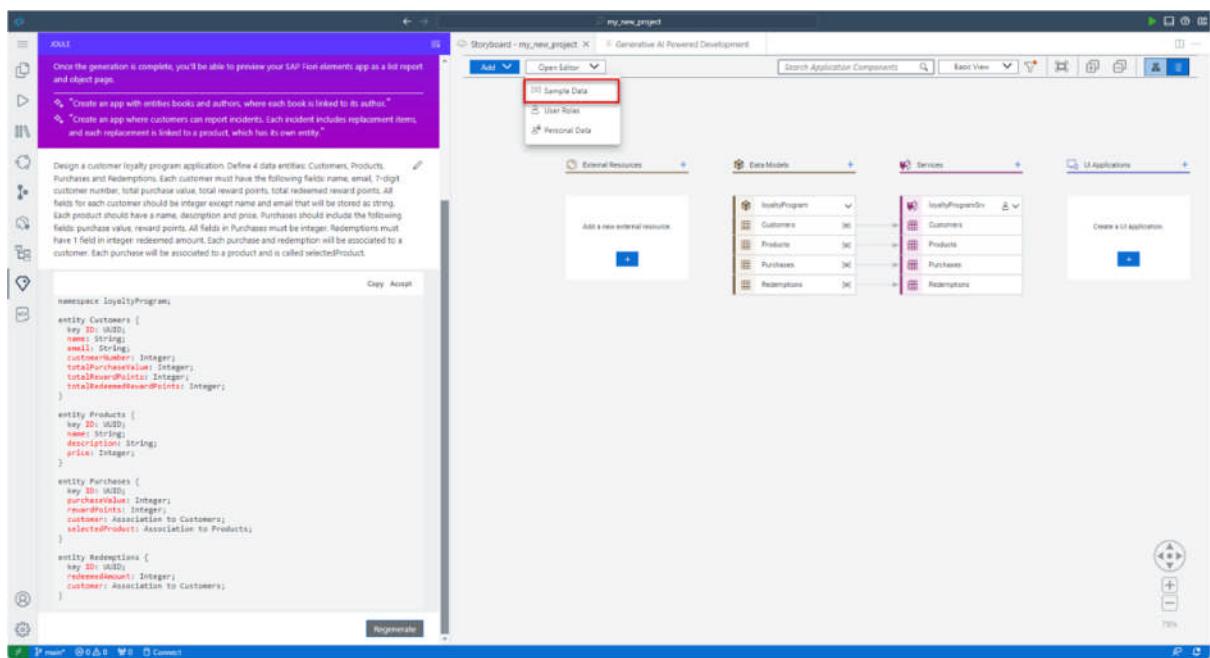
Depending on the server, it may take a few moments for Joule to create the data models and services for you.

Once you accept the code, you will see the update on the right side in the Storyboard tab.

### 3 Enhance the Sample Data Using Joule

Joule created the CAP data model and the OData service. In addition, Joule created some sample data by default. We will now ask Joule to provide additional sample data.

1. Open the Sample Data editor in the Storyboard by selecting **Open Editor -> Sample Data**.



2. In the Sample Data Editor, select the **Customers** data entity, and add 5 more rows. Click **Add**.

The screenshot shows the Joule interface with the 'Sample Data' tab selected. On the left, there's a sidebar with a tree view of entities: Customers, Products, Purchases, and Redemptions. Below the sidebar is a code editor window showing the Entity-Relationship (ER) diagram and the corresponding Java code for the LoyaltyProgram namespace.

**Customer Entity:**

```
entity Customer {
    key ID: UUID;
    name: String;
    email: String;
    customerNumber: Integer;
    totalPurchaseValue: Integer;
    totalRewardPoints: Integer;
    totalRedeemedRewardPoints: Integer;
}
```

**Product Entity:**

```
entity Product {
    key ID: UUID;
    name: String;
    description: String;
    price: Integer;
}
```

**Purchase Entity:**

```
entity Purchase {
    key ID: UUID;
    date: Date;
    quantity: Integer;
    product: Product;
    customer: Customer;
}
```

**Redemption Entity:**

```
entity Redemption {
    key ID: UUID;
    date: Date;
    points: Integer;
    purchase: Purchase;
}
```

**Customer Data Table:**

Customer Number	Name	Email	Total Purchase Value	Total Reward Points	Total Redeemed Reward Points
1	John Doe	john.doe@example.com	1234567	1000	100
2	Jane Smith	jane.smith@example.com	7054321	1000	100
3	Mike Johnson	mike.johnson@example.com	9876543	2000	200
4	Emily Davis	emily.davis@example.com	5432107	1100	110
5	Sarah Wilkes	sarah.wilkes@example.com	8765432	3000	300
6	David Wilson	daavid.wilson@example.com	7891234	9187	8462
7	James Green	james.green@example.com	2345678	6139	5618
8	Anna Blue	anna.blue@example.com	3456789	5029	5250
9	Robert Black	robert.black@example.com	6543210	9477	121
10	Samuel White	samuel.white@example.com	7890123	5619	1008

- Click **Enhance**. This will reopen Joule to modify the sample data.

The Joule interface is shown again, but the 'Enhance' button is now highlighted with a red box. The table below shows the enhanced data for two customers.

Customer Number	Total Purchase Value	Total Reward Points	Total Redeemed Reward Points
34567	500	50	10
54321	1000	100	20

Copy the prompt below:

Enhance my sample data with meaningful data. Any phone numbers must be 10 digits long.  
 All customer numbers must be 7 digits long and one customer must use the customer number 1200547.  
 No fields may be empty.  
 Total purchase value must be smaller than 10000 not rounded.  
 Both total reward points and total redeemed reward points must not be rounded, must not be identical. and must always sum to one-tenth of the total purchase value for each customer.

Paste the prompt in the text field, and click the arrow (



) to send the prompt to Joule.

The screenshot shows the Joule AI assistant interface. On the left is a vertical toolbar with icons for file operations, navigation, and other tools. The main area has a purple header with the Joule logo (a diamond with stars) and a purple footer. The central content area contains the following text:

I'm Joule, your specialized AI assistant. I can help you enhance sample data.  
So, whenever you need help with enhancing sample data, just ask, and I'll be here to make your experience even better.  
Caution: Adding too much sample data can lead to exceeding your AI quota and can cause performance issues.  
Though I strive for perfection, I might not get everything right all the time.

---

❖ "Change 'books' entity to non-fiction relevant values."  
❖ "Make sure the 'criticality' value in the range of [1(Negative), 2(Critical), 3(Positive)]."

In the purple footer box, there is a text input field containing a detailed instruction about enhancing sample data, followed by a red-bordered send button (a white triangle inside a red square).

On the right side of the interface, there is a sidebar titled "Sample Data" with a search bar and a list of entities and their associated loyalty programs:

- Customers: loyaltyProgram
- Products: loyaltyProgram
- Purchases: loyaltyProgram
- Redemptions: loyaltyProgram

The code is generated and is displayed below your prompt.

4. Accept the code. This will add the customer names, email addresses, and purchases.

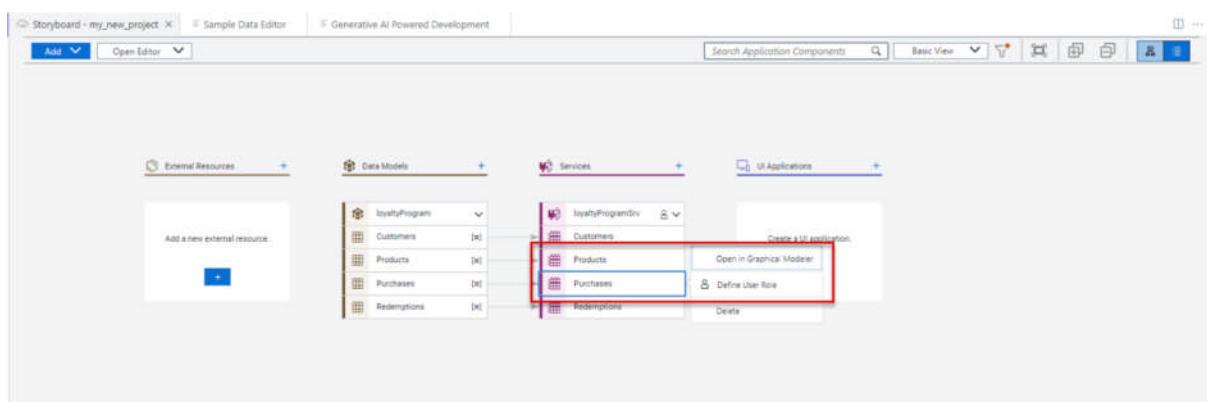
The screenshot shows the Joule AI interface. On the left is a sidebar with various icons. The main area has a purple header with the text "JOULE" and a warning message: "Caution: Adding too much sample data can lead to exceeding your AI quota and can cause performance issues." Below this is a section with two bullet points: "Change 'books' entity to non-fiction relevant values." and "Make sure the 'criticality' value in the range of [1(Negative), 2(Critical), 3(Positive)]." A large text block below these points describes requirements for phone numbers, customer numbers, and purchase values. Further down, it says "Based on your requirements, here is the updated sample data:" followed by a table of sample data. At the bottom of the table, there are "Copy" and "Accept" buttons, with "Accept" being highlighted with a red box. The right side of the interface shows a "Storyboard - my\_new\_project" panel with sections for "Sample Data", "Customers", "Products", "Purchases", and "Redemptions", each containing the word "loyaltyProgram".

ID	name	email	customerNumber	totalPurchaseValue	totalRewardPoints
9142c93b-8ac5-43bf-82cc-a9db2cecb308	John Doe	johndoe@email.com		1200547	10000
c7fe7251-dca8-4a93-b8aa-aaa28ec4f518	Jane Smith	janesmith@email.com		1200547	10000
a46e1f6e-9637-467a-b94e-dff5ee53566f	Mike Johnson	mikejohnson@email.com		1200547	10000
0b145f66-2a5e-4eb5-bbaf-969d6f0a9be5	Emily Davis	emilydavis@email.com		1200547	10000
b4ec84b2-de8d-43af-b541-1ff5470f892b	Sarah Wilson	sarahwilson@email.com		1200547	10000
9852c001-299c-457a-ad0e-f9ae03790903	Robert Brown	robertbrown@email.com		1200547	10000
aaa11975-b644-4e92-b008-f75cdf8227d9	Laura Green	lauragreen@email.com		1200547	10000
3d6f906a-abaa-4cb0-93e2-0abd6acf1068	James White	jameswhite@email.com		1200547	10000
e50f19ba-2006-40a8-b5ea-350b3d8825f9	Anna Black	annablack@email.com		1200547	10000
546099b2-6748-4477-8339-2d8d51ace0ca	David Gray	davidgray@email.com		1200547	10000

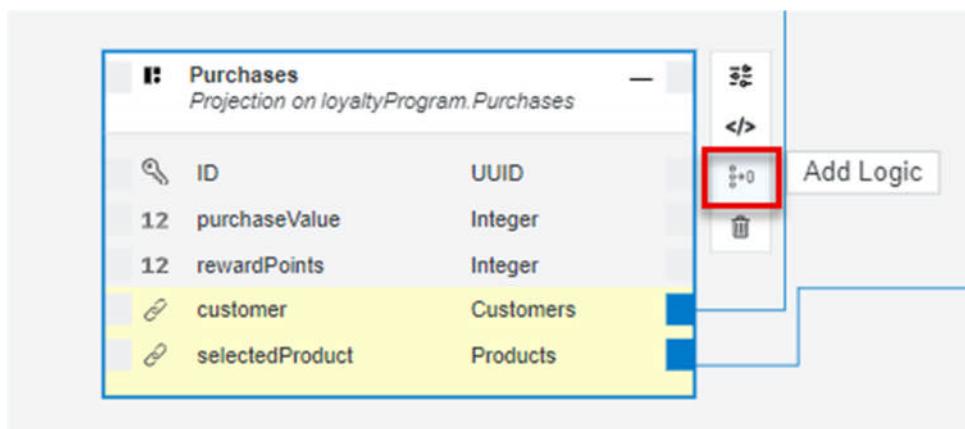
## 4 Create Application Logic with Joule

We already have created the data model, service, and sample data with Joule. Now we want to create some logic for our service. We would like to calculate the bonus points automatically when a customer makes a purchase. Additionally, we want to provide logic for customers to redeem these bonus points.

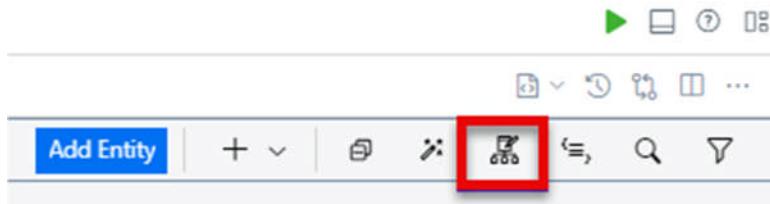
1. In the Storyboard, click on the **Purchases** entity under **Services**, and select **Open in Graphical Modeler**.



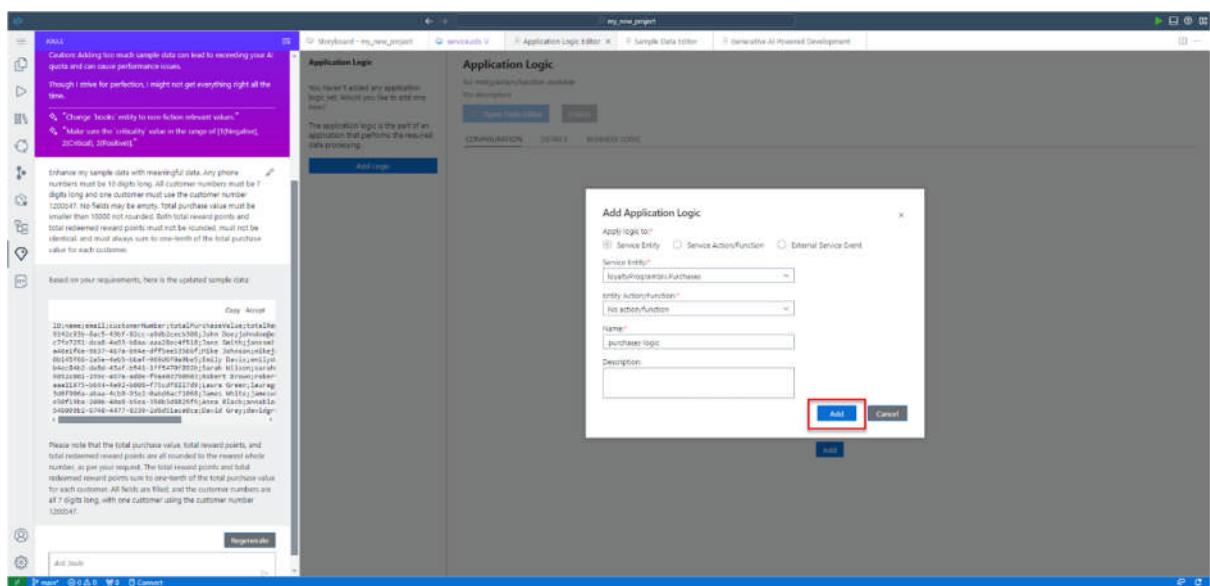
2. Select the **Purchases** entity by clicking on the title. Then, click **Add Logic**.



If you do not see the entity, click the Show All icon.



3. In the **Add Application Logic** dialog, leave the default values, and click **Add**.



The Application Logic Editor opens.

4. In the **Standard Event** section, select **Create**. That means that this logic will be automatically executed if an OData create operation is requested.

The screenshot shows the Joule application logic editor interface. On the left, there's a sidebar titled 'Application Logic' with a list of services: 'loyaltyProgramSrv' and 'Purchases'. Under 'Purchases', there's a card for 'purchases-logic' with the description 'No description' and the event 'On | CREATE'. On the right, the main panel is titled 'purchases-logic' and shows the configuration for the 'Purchases' service. It includes sections for 'Phase' (with options 'Before', 'On', and 'After'), 'Standard Event' (with checkboxes for Create, Read, Update, and Delete, where 'Create' is checked and highlighted with a red box), and 'Business Logic'. At the top, there are tabs for 'Storyboard - my\_new\_project', 'service.cds U', 'Application Logic Editor X', 'Sample Data Editor', and 'Generative AI Powered Development'.

5. Click **Open Code Editor**, and select **Application Logic**. This will open Joule again to allow us to send a prompt to Joule to create the logic for us.

This screenshot is similar to the previous one but focuses on the 'Open Code Editor' button and the 'Application Logic' tab. The 'Open Code Editor' button is highlighted with a red box. Below it, the 'Application Logic' tab is also highlighted with a red box. The rest of the interface is identical to the first screenshot, showing the 'purchases-logic' configuration page with its various settings and event checkboxes.

Copy the prompt below:

Reward points of each purchase will be the one tenth of the purchase value.  
Each purchase value will be added to the total purchase value of the related customer.  
Each reward point will be added to the total reward points of the related customer.

Paste the prompt in the text field, and click the arrow (



) to send the prompt to Joule.

The screenshot shows the Joule AI assistant interface. On the left, there's a sidebar with icons for file operations like Open, Save, and Delete. The main area has a purple background with a diamond icon and text: "I'm Joule, your specialized AI assistant. I can help you generate application logic code." Below this, it says "Whether you're creating new logic or refining an existing one, I'm here to help." and "Though I strive for perfection, I might not get everything right all the time." At the bottom of this section, there are two bullet points: "Change the 'criticality' value to 1 when the 'impact' value is bigger than 100000, else change to 2." and "Throw an error when the associated status is not 'ACTIVE.'". A blue box highlights a text input field containing the logic rule: "Reward points of each purchase will be the one tenth of the purchase value. Each purchase value will be added to the total purchase value of the related customer. Each reward point will be added to the total reward points of the related customer." To the right of this text is a red-bordered arrow button labeled "▶". The top right of the screen shows tabs for "Storyboard - my\_new\_project", "service.cds", "Application Logic Editor", "purchases-logic.js", and "Sample Data Editor". The "purchases-logic.js" tab is currently active, displaying the following code:

```
1 /**
2 * 
3 * @On(event = { "CREATE" }, entity = "loyaltyProgramSrv.Purchases")
4 * @param {Object} request - User information, tenant-specific CDS model, headers and query parameters
5 */
6 module.exports = async function(request) {
7     // Your code here
8 }
```

So Joule created code that implements the following logic:

- Check if the customer exists
- Calculate the rewardPoints from the purchase value

- Update the total purchase value and the total reward points in the customers entity

## 6. Accept the code.

The screenshot shows a software interface with a purple header bar containing icons for file operations like Open, Save, and Print. The main area has a dark blue sidebar on the left with various icons. A central window titled "JOULE" displays a message from a character named Joule, followed by two bullet points of advice. Below this is a detailed description of a logic function's purpose. A modal window titled "Storyboard - n" is open, showing a snippet of code with line numbers 1 through 8. The "Accept" button in this modal is highlighted with a red box. The code itself is a Node.js-style module export:

```
1  /**
2   * @On(event = { "CREATE" }, entity = "loyaltyProgramSrv")
3   * @param {Object} request - User information, tenant-s
4   */
5  module.exports = async function(request) {
6      // Calculate reward points based on purchase value
7      request.data.rewardPoints = Math.floor(request.data
8
9      // Update the total purchase value and total reward
10     const customer = await SELECT.one('loyaltyProgramSrv')
11         .where({ ID: request.data.customer_ID });
12
13     if (customer) {
14         const updatedCustomer = {
15             ID: customer.ID,
16             totalPurchaseValue: customer.totalPurchaseV
17             totalRewardPoints: customer.totalRewardPoin
18         };
19
20         await UPDATE('loyaltyProgramSrv.Customers')
21             .set(updatedCustomer)
22             .where({ ID: updatedCustomer.ID });
23     }
24 }
```

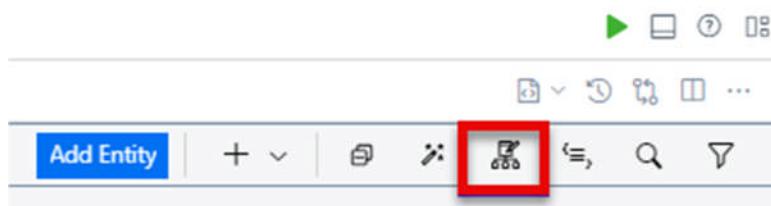
Below the code, a note explains the function's purpose: calculating reward points and updating customer totals. The bottom of the interface features a footer with a person icon and the text "This function first calculates the reward points based on the purchase value. Then it retrieves the related customer and updates their total purchase value and total reward points."

**Note:** Joule typically generates different code each time for the same prompt. If yours is different to what you can see here, that's fine as long as it does the same job. If there are no obvious errors, just keep working on the exercise. If you aren't sure, you can ask Joule to try again by clicking **Regenerate**.

7. Go back to the **service.cds** tab.
8. Select the **Redemptions** entity by clicking on the title. Then, click **Add Logic**.

The screenshot shows the Application Logic Editor interface. At the top, there are tabs for Storyboard - my\_new\_project, service.cds U, Application Logic Editor, purchases-logic.js U, Sample Data Editor, and Generative AI Powered De. Below the tabs, the 'loyaltyProgramSrv' project is selected. In the center, three entities are listed: Redemptions, Customers, and Purchases. The Redemptions entity is currently selected, as indicated by a blue border around its title bar. To the right of the entities, there is a toolbar with various icons. One icon, specifically the 'Add Logic' button (represented by a plus sign inside a box), is highlighted with a red box. This indicates that the user should click this button to add logic to the selected entity.

If you do not see the entity, click the Show All icon.



9. In the **Add Application Logic** dialog, leave the default values, and click **Add**.

The screenshot shows two overlapping dialogs. The background dialog is titled "Application Logic" and contains a message about adding application logic. The foreground dialog is titled "Add Application Logic" and has the following fields:

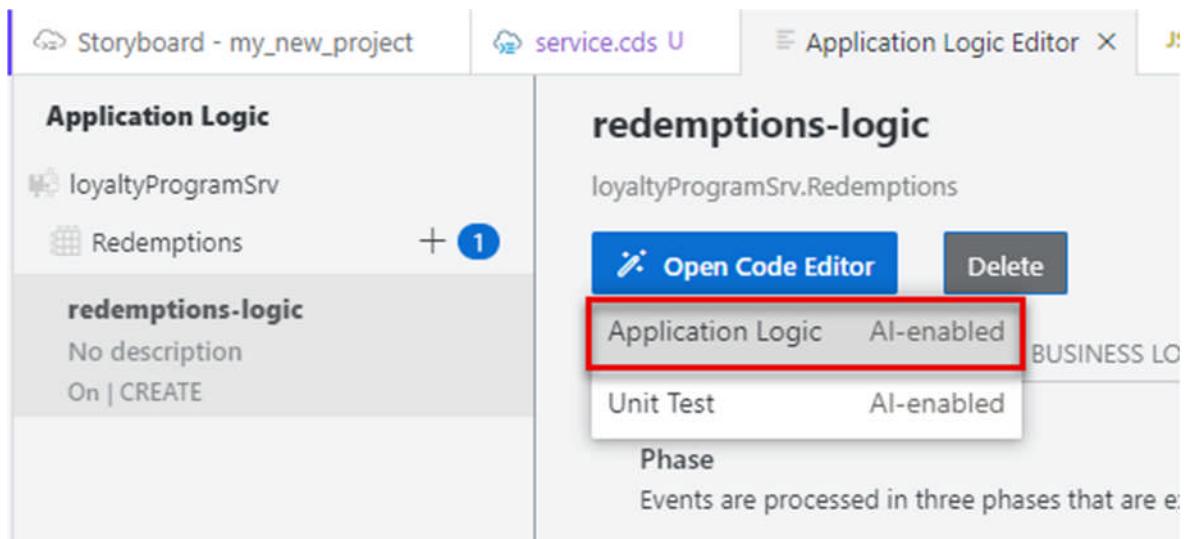
- Apply logic to:  
 Service Entity    Service Action/Function    External Service Event
- Service Entity:  
loyaltyProgramSrv.Redemptions
- Entity Action/Function:  
No action/function
- Name:  
redemptions-logic
- Description:  
[empty text area]

A red box highlights the "Add" button at the bottom right of the "Add Application Logic" dialog.

10. In the **Standard Event** section, select **Create**.

The screenshot shows the "Application Logic Editor" interface. On the left, there's a sidebar with "Storyboard - my\_new\_project" and tabs for "service.cds U", "Application Logic Editor X", "JS purchases-logic.js U", and "Sample Data Editor". The main area shows an "Application Logic" section with a "Redemptions" entity and a "redemptions-logic" configuration. The "redemptions-logic" configuration has a "Phase" section and a "Standard Event" section. The "Standard Event" section includes checkboxes for "Create", "Read", "Update", and "Delete", with "Create" being checked. A red box highlights the "Create" checkbox.

11. Click **Open Code Editor**, and select **Application Logic**. This will open Joule again to allow us to send a prompt to Joule to create the logic for us.



Copy the prompt below::

Deduct the redemption amount from the customer's total reward points and add that to their total redeemed points.

Paste the prompt in the text field, and click the arrow (



) to send the prompt to Joule.

The Joule AI interface is shown in a browser window. On the left, a sidebar contains icons for file operations, navigation, and settings. The main area is titled "JOULE" and features a large diamond icon with three smaller stars above it. Below the icon, text reads: "I'm Joule, your specialized AI assistant. I can help you generate application logic code. Whether you're creating new logic or refining an existing one, I'm here to help. Though I strive for perfection, I might not get everything right all the time." A horizontal line separates this from a list of two items:

- ❖ "Change the 'criticality' value to 1 when the 'impact' value is bigger than 100000, else change to 2."
- ❖ "Throw an error when the associated status is not 'ACTIVE'."

At the bottom, a box contains the instruction: "Deduct the redemption amount from the customer's total reward points and add that to their total redeemed points." To the right of this text is a red-bordered "▶" button.

On the right side of the screen, a code editor window titled "Storyboard - my\_new\_project" displays the following code:

```
1  /**
2   *
3   * @On(event = { "CR
4   * @param {Object} r
5   */
6  module.exports = asy
7  ... // Your code here
8 }
```

16.

The screenshot shows the edunet Storyboard interface. On the left, there's a sidebar with various icons. The main area has a purple header titled "JOULE" with the sub-header "Though I strive for perfection, I might not get everything right all the time." Below this are two bullet points:

- ❖ "Change the 'criticality' value to 1 when the 'impact' value is bigger than 100000, else change to 2."
- ❖ "Throw an error when the associated status is not 'ACTIVE'."

Below these points is a task: "Deduct the redemption amount from the customer's total reward points and add that to their total redeemed points." Underneath this task, it says "Here is the custom logic function that you can use:" followed by a code editor window. The code is a Node.js module export for an "CREATE" event on the "loyaltyProgramSrv" entity:

```
/** * @On(event = { "CREATE" }, entity = "loyaltyProgramSrv") * @param {Object} req - User information, tenant-specific */module.exports = async function(req) {  const { redeemedAmount, customer_ID } = req.data;  const tx = cds.transaction(req);

  // Fetch the customer's current reward points and redeemable points
  const [{ totalRewardPoints, totalRedeemedRewardPoints }] = await tx.run(`    SELECT.from('loyaltyProgramSrv.Customers')      .where({ ID: customer_ID })      .columns('totalRewardPoints', 'totalRedeemedRewardPoints')`);

  // Check if the customer has enough reward points to redeem
  if (totalRewardPoints < redeemedAmount) {
    req.error(400, 'Insufficient reward points');
    return;
  }

  // Deduct the redeemed amount from the customer's total reward points
  // and add it to their total redeemed points
  await tx.run(`    UPDATE('loyaltyProgramSrv.Customers')
      .set({
          totalRewardPoints: totalRewardPoints - redeemedAmount,
          totalRedeemedRewardPoints: totalRedeemedRewardPoints + redeemedAmount
      })
      .where({ ID: customer_ID })`);}
```

At the bottom of the code editor, there are "Copy" and "Accept" buttons. The "Accept" button is highlighted with a red box. Below the code editor, there's a note: "This function will be triggered when a new redemption is created. It first fetches the customer's current reward points and redeemable points. Then it checks if the customer has enough reward points to deduct the amount specified in the request. If not, it returns an error. Otherwise, it deducts the amount from the customer's total reward points and adds it to their total redeemed points." A small user icon is next to this note.

Have a closer look at the generated code. It even includes some checks to see if the customer has enough points to redeem.

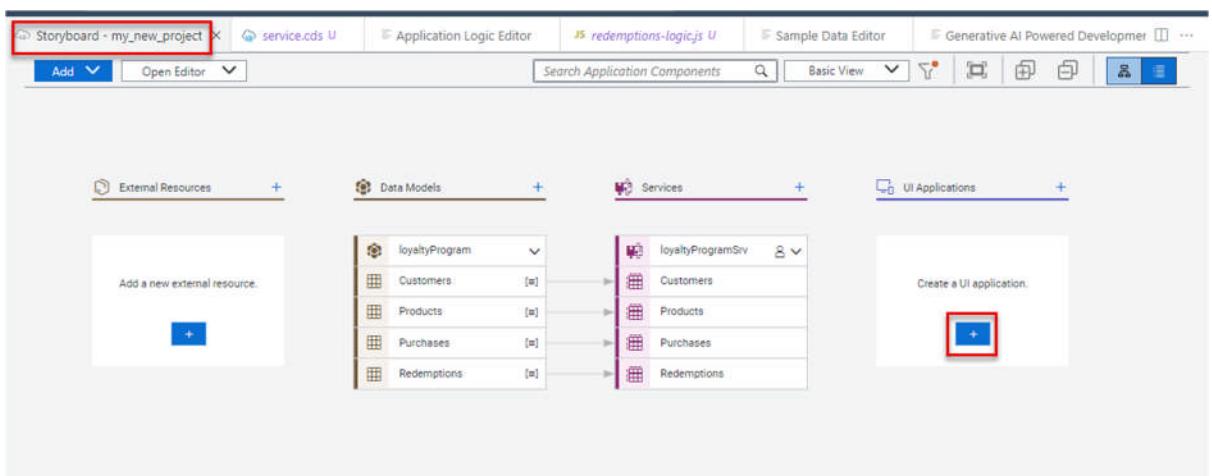
The screenshot shows a software interface with a dark header bar. In the top right corner, there is a logo for "edunet foundation". Below the header, there is a tab bar with several tabs: "Storyboard - my\_new\_project", "service.cds U", "Application Logic Editor", "JS redemptions-logic.js U X", and "Sample Data I". The main area of the interface is a code editor displaying a CDS logic file. The file contains JavaScript-like code for handling a "CREATE" event on the "Redemptions" entity. It includes a database query to fetch the customer's current reward points and redeemed points, followed by a conditional check to ensure the customer has enough reward points to redeem. If the condition fails, it returns an error. Otherwise, it deducts the redeemed amount from the total reward points and adds it to the total redeemed points. The code is numbered from 1 to 33. A red rectangular box highlights the conditional check block (lines 17-21).

```
1  /**
2   * 
3   * @On(event = { "CREATE" }, entity = "loyaltyProgramSrv.Redemptions")
4   * @param {Object} req - User information, tenant-specific CDS model, headers and query parameters
5   */
6  module.exports = async function(req) {
7      const { redeemedAmount, customer_ID } = req.data;
8      const tx = cds.transaction(req);
9
10     // Fetch the customer's current reward points and redeemed points
11     const [{ totalRewardPoints, totalRedeemedRewardPoints }] = await tx.run(
12         SELECT.from('loyaltyProgramSrv.Customers')
13             .where({ ID: customer_ID })
14             .columns('totalRewardPoints', 'totalRedeemedRewardPoints')
15     );
16
17     // Check if the customer has enough reward points to redeem
18     if (totalRewardPoints < redeemedAmount) {
19         req.error(400, 'Insufficient reward points');
20         return;
21     }
22
23     // Deduct the redeemed amount from the customer's total reward points
24     // and add it to their total redeemed points
25     await tx.run(
26         UPDATE('loyaltyProgramSrv.Customers')
27             .set({
28                 totalRewardPoints: totalRewardPoints - redeemedAmount,
29                 totalRedeemedRewardPoints: totalRedeemedRewardPoints + redeemedAmount
30             })
31     );
32 };
33 
```

## 5 Add UI to the Application

To display and test the content we created for the customer loyalty program, we need to create an SAP Fiori elements UI.

1. Go to back to the Storyboard and add a UI application.



2. We will start with the user interface for the **Purchases** data entity. Set the **Display name** to **Purchases** and the **Description** to **Manage Purchases**, and then click **Next**.

The screenshot shows the SAP UI5 Application Generator interface. At the top, there are tabs for 'Storyboard - my\_new\_project', 'New UI Application X', 'service.cds U', 'Application Logic Editor', and 'JS redemptions-logic.js U'. The main window title is 'Create New UI Application'. On the left, there's a sidebar with radio buttons for 'UI Application Details', 'UI Application Type', 'UI Application Template', and 'Data Objects'. The 'UI Application Details' section is selected and highlighted with a red box. It contains three input fields: 'Display name' with 'Purchases', 'Application name' with 'Purchases', and 'Description' with 'Manage Purchases'. Below this section, the other options in the sidebar are visible.

3. We are using the browser, so we will select **Template-Based Responsive Application** as the UI Application type, and click **Next**.

Storyboard - my\_new\_project    New UI Application X    service.cds U    Application Logic Editor    JS redemptions-logic.js U    Sample

## Create New UI Application

**UI Application Details**

**Display Name:** Purchases  
**Application Name:** Purchases  
**Description:** Manage Purchases

**UI Application Type**

**What Kind Of Application Do You Want To Create?**: Template-Based, Responsive Application

**UI Application Type**  
 **UI Application Template**  
 **Data Objects**

**UI Application Type**

What kind of application do you want to create? \*

**Template-Based, Responsive Application**

Create a browser-based application with standard yet extensible floorplans, runs on desktop and mobile. It is derived from your data structures and metadata, automatically applies the latest SAP Fiori design.

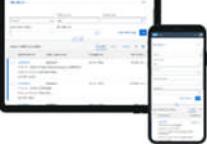
[More Information](#)



**Mobile-Centric, Freestanding Application**

Create an application to run natively on mobile (Android & iOS) with device specific features such as biometric authentication and barcode scanning. It also runs on desktop browsers.

[More Information](#)



[Back](#) [Next >](#)

4. Select **List Report Page** as the UI application template, and click **Next**.

The screenshot shows the SAP Fiori application setup interface. The top navigation bar includes tabs for Storyboard - my\_new\_project, New UI Application (selected), service.cds, Application Logic Editor, redemptions-logic.js, Sample Data Editor, and other system-related icons.

The main content area is titled "SAP Fiori application" and displays the "UI Application Template" configuration. On the left, under "UI Application Details", the "Display Name" is set to "Purchases", "Application Name" is "Purchases", and "Description" is "Manage Purchases". Under "UI Application Type", the "What Kind Of Application Do You Want To Create?" section indicates "Template-Based, Responsive Application".

The "UI Application Template" section lists three options:

- List Report Page** (selected, highlighted with a red box): "Create an SAP Fiori elements application containing a list report and an object page." It includes a "More Information" link and a small screenshot of the application interface.
- Form Entry Object Page**: "Create an SAP Fiori elements application containing an object page optimized for data entry." It includes a "More Information" link and a small screenshot.
- Custom Page**: "Create an SAP Fiori elements application containing a custom page based on the flexible programming model." It includes a "More Information" link and a small screenshot.

At the bottom, there are "Back" and "Next >" buttons. The "Next >" button is highlighted with a red box.

5. Select **Purchases** as the **Main entity**, and click **Finish**. The page will be created now.

SAP Fiori application ⓘ

**UI Application Details**

**Display Name:** Purchases

**Application Name:** Purchases

**Description:** Manage Purchases

**Data Objects**

Main entity \*

Purchases

Automatically add table columns to the list page and a section to the object page if none already exists?

Yes  No

ⓘ Basic value helps will also be created.

**UI Application Type**

**What Kind Of Application Do You Want To Create?:** Template-Based, Responsive Application

**UI Application Template**

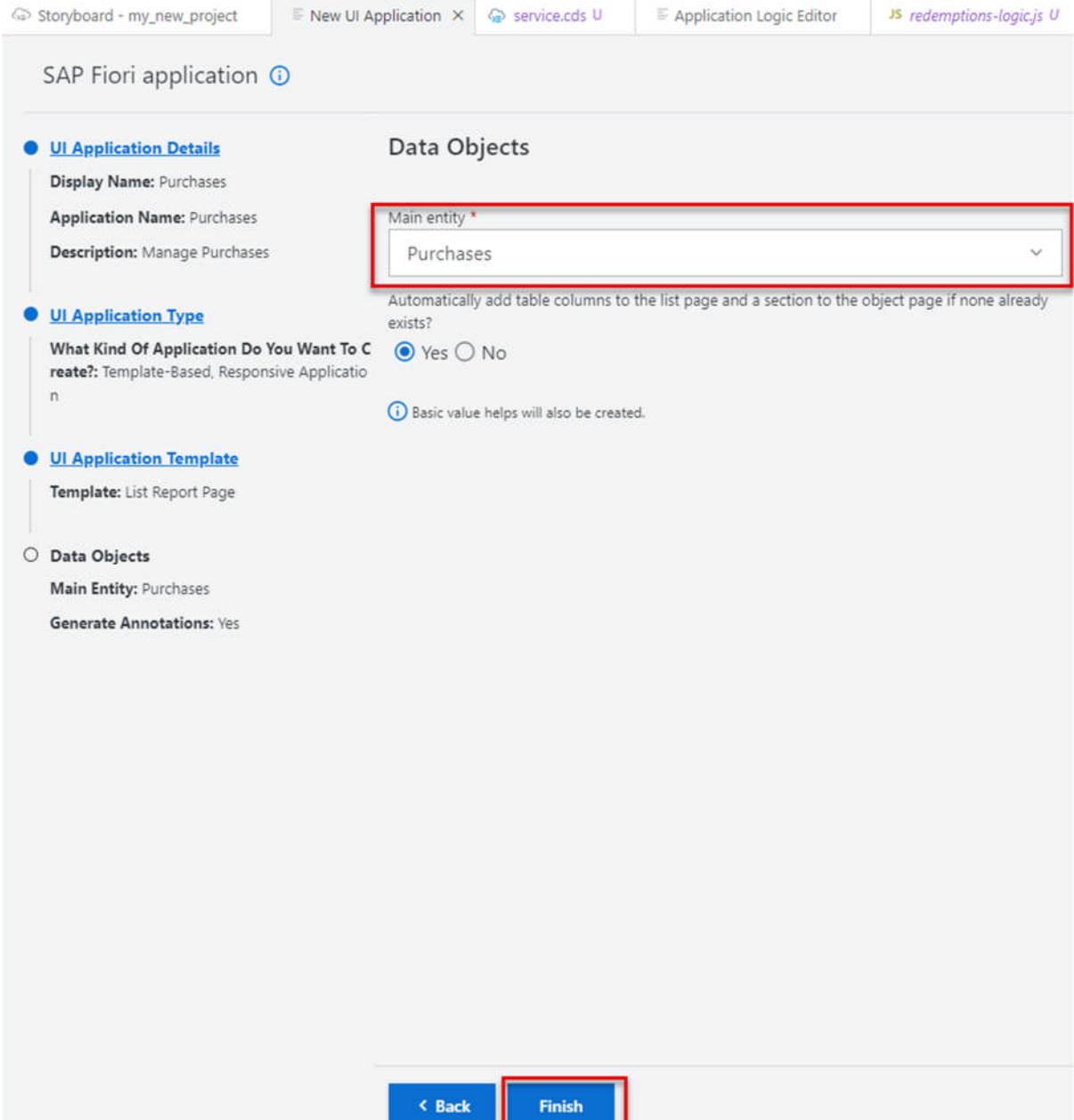
**Template:** List Report Page

**Data Objects**

**Main Entity:** Purchases

**Generate Annotations:** Yes

< Back **Finish**



It might take a few moments for the UI to be created because the dependencies need to be installed.

6. Repeat steps 2 through 5 to create additional UI apps for the **Customers** and the **Redemptions** entities.

#### Customer:

- o Display name: **Customers**

- Description: **Manage Customers**
- UI Application type: **Template-Based Responsive Application**
- UI Application Template: **List Report Page**
- Main Entity: **Customers**

### Redemptions:

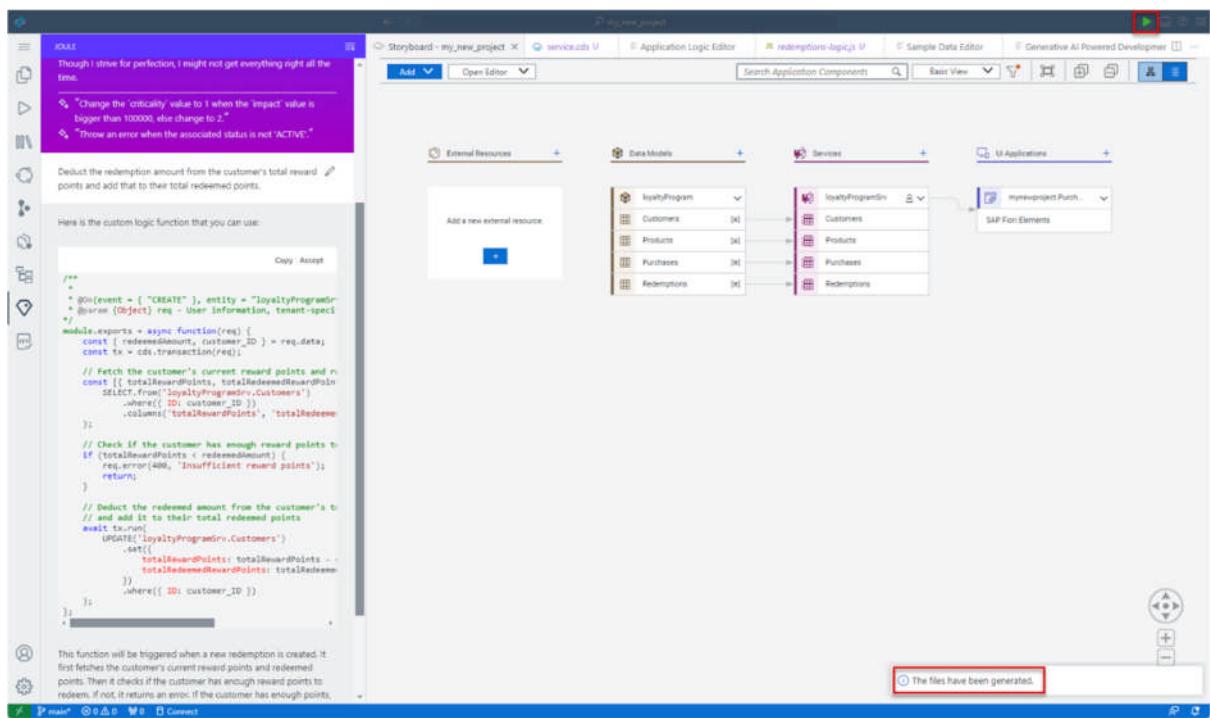
- Display name: **Redemptions**
- Description: **Manage Redemptions**
- UI Application type: **Template-Based Responsive Application**
- UI Application Template: **List Report Page**
- Main Entity: **Redemptions**

And that's it! You've created an application.

7. To preview your application, once the files have been generated, go to the upper-right corner, and click



(Run and Debug).



The application's preview is displayed.

SAP Application Development Project Preview

# my\_project

1.0.0

**Web Applications (3)**

The user will have the following applications. Click to explore them live.

**Services (1)**

**my\_projectService**

4 tables are included in this service. View [service details](#) and [metadata](#) for more info.

Customers	Products

Purchases	Redemptions

**My Home**

**Customers**  
[Manage Customers](#)

**Purchases**  
[Manage Purchases](#)

**Redemptions**  
[Manage Redemptions](#)

## 8. Click Go.

**Standard**

Editing Status:

Search	Q	All	Go	Adapt Filters (1)
--------	---	-----	----	-------------------

**Customers**

Name	Email	Customer Num...	Total Purchase V...	Total Reward Po...	Total Redeemed Re...
To start, set the relevant filters and choose "Go".					

The customer information is displayed.

Name	Email	Customer Num...	Total Purchase V...	Total Reward Po...	Total Redeemed Re...
Emma Johnson	emmajohnson@example.com	2,345,678	6,000	400	200
Sophia Williams	sophiawilliams@example.com	4,567,890	10,000	800	200
John Doe	johndoe@example.com	1,234,567	5,000	400	100
Jane Smith	janesmith@example.com	7,654,321	8,000	600	200
Michael Brown	michaelbrown@example.com	1,200,547	7,000	500	200
Lucas Davis	lucasdavis@example.com	5,678,901	9,500	750	200
Oliver Taylor	olivertaylor@example.com	3,456,789	9,000	700	200

- From the dropdown list at the top of the page, select **Home** to go back and preview the other applications.

Name	Email	Customer Num...	Total Purchase V...	Total Reward Po...	Total Redeemed Re...
Emma Johnson	emmajohnson@example.com	2,345,678	6,000	400	200
Sophia Williams	sophiawilliams@example.com	4,567,890	10,000	800	200
John Doe	johndoe@example.com	1,234,567	5,000	400	100
Jane Smith	janesmith@example.com	7,654,321	8,000	600	200
Michael Brown	michaelbrown@example.com	1,200,547	7,000	500	200
Lucas Davis	lucasdavis@example.com	5,678,901	9,500	750	200
Oliver Taylor	olivertaylor@example.com	3,456,789	9,000	700	200