**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **29/07/2020** | **Name:** | **Nayanashree k s** |
| **Course:** | **Coursera** | **USN:** | **4AL16EC042** |
| **Topic:** | Industrial IoT on Google Cloud Platform | **Semester & Section:** | **8th A** |
| **Github Repository:** | **Nayana\_onine** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
|  |
| **REPORT** IoT Devices are Connected to the Cloud   The general structure of an IoT network includes devices to interact with the environment; a gateway to gather the data and communicate with the cloud; and the cloud to store, process, and analyze the data.    **Devices**  Devices interact with the environment by gathering data or performing an action upon the environment. They can be connected versions of common objects you might already know, or new and purpose-built devices. They can be devices that you own personally and carry with you (for example, a phone or fitness tracker) or keep in your home (Google Home).  Devices can also be embedded in factory equipment or as part of the infrastructure of the city where you live. Each of them is able to convert valuable information from the physical world into digital data that provides increased visibility into how your users interact with your products, services, or applications.    **Gateway**  A gateway ensures that devices are securely connected to the cloud. A gateway can be a cell phone, a personal assistant, or a microprocessor platform. It controls messaging between the device and the cloud, and in some cases, does real-time analytics or machine learning, which is called edge computing.  A gateway device might be used even when the participating devices are capable of communicating without one. In this scenario, the gateway adds value because it provides processing of the data across multiple devices before it is sent to the cloud. In that case, the direct inputs would be other devices, not individual sensors    **Cloud**  The cloud handles computation and data storage, real-time and batch analytics, and machine learning and visualization. In an IoT network, devices are often added, removed, or modified. The cloud connected to the device must be capable of scaling quickly to meet these demands. Communication protocols and data pipelines must be able to handle rapidly changing message traffic and storage needs. You need an efficient, scalable, affordable way to both manage those devices and handle all that information and make it work for you. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **29/07/2020** | **Name:** | **Nayanashree** | |
| **Course:** | **Salesforce** | **USN:** | **4AL16EC042** | |
| **Topic:** | **Salesforce platform basics** | **Semester& Section:** | **8th A** | |
| **AFTERNOON SESSION DETAILS**    **s2.PNG** | | | |
| **REPORT**  **Overview of Objects**  DreamHouse is a realty company that provides a way for customers to shop for homes and contact real estate agents online. DreamHouse brokers use some of Salesforce’s standard functionality, like contacts and leads, to track home buyers.  But when it comes to selling houses, there are a lot more things they want to track. For example, Salesforce doesn’t include a standard way to track properties. How is DreamHouse supposed to know which homes they have for sale or how much each home costs?  Luckily, their Salesforce admin, D’Angelo, knows that the Salesforce platform offers a solution. We’ll work with D’Angelo to see what he’s building.  Let’s start with the **data model**. A data model is more or less what it sounds like. It’s a way to model what database tables look like in a way that makes sense to humans.  If you’re not familiar with databases, think about storing data in a spreadsheet. For example, D’Angelo can use a spreadsheet to track all DreamHouse’s properties. Columns can store the address, cost, and other important attributes. Rows can store this information for each property that DreamHouse is selling. Database tables are set up in a similar way.  A spreadsheet that stores property information.  But looking at data in tables isn’t ideal for humans. That’s where the data model comes in.  In Salesforce, we think about database tables as **objects**, we think about columns as **fields**, and rows as **records**. So instead of an account spreadsheet or table, we have an Account object with fields and a bunch of identically structured records.  A property record with the same information as the table.  When we talk about the data model, we’re talking about the collection of objects and fields in an app. Let’s learn more about objects and fields so you can start building your own data model.  **Get to Know Objects**  Salesforce supports several different types of objects. There are standard objects, custom objects, external objects, platform events, and BigObjects. In this module, we focus on the two most common types of objects: standard and custom.  **Standard objects** are objects that are included with Salesforce. Common business objects like Account, Contact, Lead, and Opportunity are all standard objects.  **Custom objects** are objects that you create to store information that’s specific to your company or industry. For DreamHouse, D’Angelo wants to build a custom Property object that stores information about the homes his company is selling.  Objects are containers for your information, but they also give you special functionality. For example, when you create a custom object, the platform automatically builds things like the page layout for the user interface.  **Create a Custom Object**  Let’s work alongside D’Angelo to see how he builds the Property object. We need this object later, so don’t skip these steps!   1. Scroll to the bottom of this page. 2. Click the arrow next to Launch and select **Create a Trailhead Playground**. Don’t skip this step! You need to use a fresh and clean Trailhead Playground for this module. 3. Once your playground is created (it takes a minute!), press **Launch**. 4. Click the gear icon The setup gear. at the top of the page and launch setup. 5. Click the **Object Manager** tab. 6. Click **Create** | **Custom Object** in the top-right corner. 7. For Label, enter Property. Notice that the Object Name and Record Name fields auto-fill. 8. For Plural Label, enter Properties. 9. Check the box for **Launch New Custom Tab Wizard after saving this custom object**. 10. Leave the rest of the values as default and click **Save**. 11. On the New Custom Object Tab page, click the Tab Style field and select a style you like. The style sets the icon to display in the UI for the object. 12. Click **Next**, **Next**, and **Save**. | | | |