**Design decisions for the assignment:**

1. **ORG Selection:** ORG B is my primary salesforce org and ORG A is my secondary salesforce ORG. Code has been written to get Data from ORG A and show it in ORG B.
2. **Authentication:**

**Named Credential approach** has been chosen for authentication as it provides a mechanism to authenticate without providing user name/password and the code will be very efficient with Named credential approach.

**Other Alternative possible design decisions:** We could also authenticate via various OAUTH techniques (for ex : user name, password flow). It is also possible to do SOAP Login authentication to get session Id.

1. **Rest Service**:

Rest Service accepts the query string and sObject Names separated by semi colon) as parameters so this service can query any sObject and return records.

Needed a wrapper class to deliver the response in order to send the error message in case of failures with rest service.

If no SObject Record Names is passed, Rest Service queries all sObject Records and returns the results.

Both the query string, sObject Record Name string can be customized to use it anywhere.

1. **Pagination:**

StandardSet Controller approach is taken for doing pagination as it has lot of built in methods to support pagination and easy to manage. Record size of 10 per page is chosen.

**Other Alternative possible design decisions:**

Pagination can also be achieved via complete custom code to assign records per page, enabled/disable buttons etc.

Salesforce also has OFFSET concept which is often used to do pagination but it has limitations with number of records it can handle though.

1. **UI/UX Strategy:**

When we design a page, we need to make sure it is mobile ready and works great in all devices including lightning experience. We have number of ways to build a code that is mobile ready. For ex, We could use frame work like BootStrap, Google Material Design and build the page with respective grids.

We could also achieve the same with lightning components and then calling the lightning component dynamically in visualforce page.

We can also replace with angular JS framework where we can move the UI to html templates and JS can do lot of back lifting.

For the purpose of simplicity, I have not implemented any of such techniques but just used simple visual force page. Visualforce page tags won’t be device friendly.

1. **Create Account from Visualforce Page:**

There are numerous ways to insert an account from a visualforce page. We could build a section to accept input fields and then do a DML. We could build code to select record type and then fields can be read via field sets for record types etc.

Out of all these, it is always easiest to redirect to a standard creation page so we always follow same process for creation. I have followed this approach instead of building a form in a visual force page.

1. **Future method strategy:**

Future method has been designed to capture all account names per trigger invocation in a string and do a call OUT. This is to make sure we do one call out per trigger invocation so we respect a max of 50 future calls per invocation.

1. **Error Handling strategy**:

In general, it is best practice to show user friendly error messages to the front-end screen. There will be scenarios where we should implement different error handling techniques. Ex Future methods. Since the error handling in future techniques is somewhat time consuming and complex, its’ not considered in the design. Where ever possible, the error handling is done to throw error messages to the front end.

**Other Alternative possible design decisions:**

* Capture error messages per record ID, record Name. Send an email to system admin in a csv format.
* Create a custom log object which contains various fields to store various details about error message. Ex: Location, class name, method name, full error message, stack trace etc. Also, create a re-usable apex method to insert data into this custom log object. A workflow rule on this can notify the system admin.