Hi!

We are Majesco-ClaimVantage, we develop a suite of leading Salesforce App solutions for the insurance market.

We really hope you can join the work in our projects. The following topics are recurrently used on our Salesforce development, so we will be asking about them during call:

- Apex code: Apex Class, Apex Triggers, Apex Test classes.
- Object Oriented Concepts: Inheritance, Interfaces.
- VisualForce: VisualForce Pages, Custom controllers.
- LWC Development, LWC communication, Calling Server-Side Classes.
- git and GitHub: branching, pull requests, commit, push.
- SFDX: creation of Scratch orgs, push/pull code.

Before we can setup the interview call, there are 2 exercises we need you to do, and send us.
The approaches taken to these exercises will be discussed in your interview.

Prep work

The initial Salesforce project will be in the GitHub repository https://github.com/claimvantage/cv-candidates-sf-experience.git

Please clone this Salesforce project into your local machine. The definition includes some metadata that you will need for the exercises, namely the custom object and custom fields that are references in the exercises below.

Use your favourite IDE to implement the exercises, deploying the code to a scratch org to test the results. Once you are done, do pull the changes and send us the complete project.

Best of luck, and if you have any questions, please do not hesitate to reach out.

Exercise 1 – Apex

Note: the custom fields Account.Budject__c, Contact.Budject__c, Contact.Use_Marketing_Budget__c, and Contact.Use_Sales_Budget__c, are defined in the original Salesforce project that you cloned from the GitHub repository.

Write a trigger on Account to perform the logic described below whenever “Budget” changes. Note that the “Budget” field may sometimes be empty.

The Account “Budget” is to be divided into 2 portions (60% for Sales and 40% for Marketing). The Sales portion is to be equally distributed by all Contacts of that Account that have “Use Sales Budget” checked. The Marketing portion is to be equally distributed by all Contacts of that Account that have “Use Marketing Budget” checked.

A Contact that has both “Use Sales Budget” and “Use Marketing Budget” receives a contribution from both Sales and Marketing portions.

Do take into consideration that you are dealing with currencies, to the second decimal place (i.e. you cannot split currencies past the cent - e.g. €2.05 divided by 2 needs to be split into €1.03 + €1.02, and not into €1.025 + €1.025).

Also, divisions need to be as fair as possible, that means a maximum of one cent different between them any parts - e.g. €9.02 divided by 3 should be €3.01 + €3.01 + €3.00, and not €3.02 + €3.00 + €3.00.

For this exercise, you do not need to also write Contact triggers to handle child Contacts being added or deleted from the Account.

Do supply all the unit tests class for this trigger that you find to be necessary.

Example:

An Account has a Budget of €100.00 and the following child Contacts:

Contact	**Use Marketing Budget**	**Use Sales Budget**
Contact 1	True	False
Contact 2	True	True
Contact 3	False	True
Contact 4	False	True
Contact 5	False	False

Result:

Account Budget = €100.00

Marketing Budget = 40% of €100.00 = €40.00

Sales Budget = 60% of €100.00 = €60.00

Users that use Marketing Budget = 2

Users that use Sales Budget = 3

Share of the Marketing Budget = €40.00 / 2 = €20.00

Share of the Sales Budget = €60.00 / 3 = €20.00

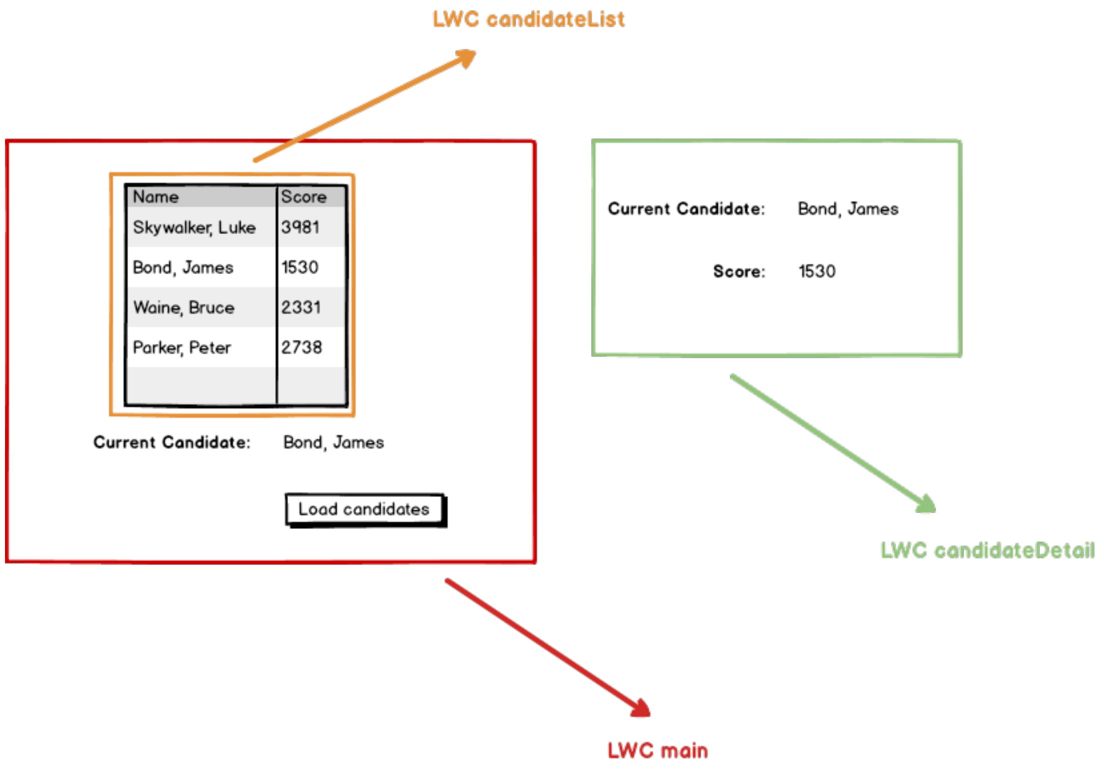
Distribution of Budget:

Contact	**Budget**	**Explanation**
Contact 1	€20.00	1 share of the Marketing Budget
Contact 2	€40.00	1 share of the Marketing Budget + 1 share of the Sales Budget
Contact 3	€20.00	1 share of the Sales Budget
Contact 4	€20.00	1 share of the Sales Budget
Contact 5	€0.00	No shares from either the Sales Budget or from the Marketing Budget

Exercise 2 – Lightning Web Components

Note: the custom object Candidate__c is defined in the original Salesforce project that you cloned from the GitHub repository.

On this exercise you will create 3 Lightning Web Components (LWCs as per picture below)



The *candidateList* LWC displays the the Candidate__c records the user has access to.

The *main* LWC has a *candidateList* LWC. The *candidateList* is originally empty, and only gets populated from the database when the button 'Load candidates' in the *main* LWC is pushed.

As you click in the different lines in the *candidateList*, the 'Current Candidate' label in the *main* LWC gets updated with the name of the line that was selected.

In addition to that, clicking on a line in the *candidateList* will also update the *candidateDetail* LWC with the data from the selected record.

Final notes for the interview.

During the interview you will be asked to explain the exercises that you sent.

Please have the exercises at hand and ready to share the screen.
Among other things, you could be asked to do:

- Walk through the code in your IDE;
- Perform git commands in the terminal, like making a pull request into GitHub;
- Make small changes in the IDE, and push them to a scratch org.

We are looking forward to hearing from you!

References

For the topics we are going to cover, here are some relevant links with information. This is obviously not and exhaustive list, but a starting point should you like to refresh or consolidate your knowledge on some topics, in order to better prepare for the exercises and interview.

Topic	**Links**
Apex code	https://trailhead.salesforce.com/en/content/learn/trails/build-apex-coding-skills https://trailhead.salesforce.com/content/learn/modules/apex_triggers/apex_triggers_bulk https://trailhead.salesforce.com/content/learn/modules/apex_testing https://developer.salesforce.com/docs/atlas.en-us.224.0.apexcode.meta/apexcode/apex_debug_test_deploy.htm
Object Oriented Programming	https://www.salesforcehut.com/2020/09/oops-in-salesforce-with-real-world.html https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes.htm https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_extending.htm https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_casting.htm https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_interfaces.htm
VisualForce	https://trailhead.salesforce.com/content/learn/modules/visualforce_fundamentals https://trailhead.salesforce.com/content/learn/projects/quickstart-visualforce
Lightning Web Components	https://trailhead.salesforce.com/en/content/learn/modules/lightning-web-components-basics https://trailhead.salesforce.com/content/learn/modules/lightning-web-components-and-salesforce-data?trail_id=build-lightning-web-components https://trailhead.salesforce.com/content/learn/projects/lwc-build-flexible-apps?trail_id=build-lightning-web-components https://trailhead.salesforce.com/content/learn/projects/communicate-between-lightning-web-components?trail_id=build-lightning-web-components
git and GitHub	https://trailhead.salesforce.com/en/content/learn/trails/move-to-a-continuous-integration-development
SFDX	https://trailhead.salesforce.com/content/learn/projects/quick-start-salesforce-dx https://trailhead.salesforce.com/content/learn/projects/quickstart-vscode-salesforce https://trailhead.salesforce.com/en/content/learn/modules/sfdx_app_dev