Karnataka Law Society’s GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi – 590008 Karnataka, India.



*A Salesforce Lightning Course Project on,*

#### “1. Add Reports and Dashboards to a Travel Approval

#### 2. Communicate between Lightning Web Components-Explore different approaches to communicating between Lightning web components.”

*Submitted for the requirements of 6th-semester B.E in ISE For*

**“Introduction to Salesforce”**

Subject Code: 18IS653

*Submitted By:*

|  |  |
| --- | --- |
| **Name** | **USN** |
| John Nixon | 2GI19IS016 |
| Hrutuja Patnekar | 2GI19IS017 |

Under the guidance of

Internal guide: Prof. S K Madi

External guide: Apoorva and Sanjana, PwC.

Dept. of ISE, KLS GIT

Academic Year 2021-2022 (EVEN Semester)

Karnataka Law Society’s GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi – 590008 Karnataka, India.

#### Department of Information Science and Engineering



**Certificate**

This is to certify that the Project work titled “**1. Add Reports and Dashboards to a Travel Approval 2. Communicate between Lightning Web Components-Explore different approaches to communicating between Lightning web components.**” carried out by Students**: John Nixon, Hrutuja Patnekar** bearing USNs: **2GI19IS016, 2GI19IS017** is submitted in partial fulfillment of the requirements for 6th Semester B.E. in **INFORMATION SCIENCE AND ENGINEERING**, Visvesvaraya Technological University, Belagavi. It is certified that all corrections/ suggestions indicated have been incorporated in the report. The project report has been approved as it satisfies the academic requirements regarding research work prescribed for the said degree.

Date: 05-07-2022

Place: Belagavi

(Signature of Guide) (Signature of HOD ISE)

**Prof. S K Madi Dr. Kiran K Tangod**

#### Dept. of ISE, KLS GIT

KLS Gogte Institute of Technology, Belagavi

**Table of Contents** **Page No.**

1. **Problem Statement: 01** - 01

|  |  |
| --- | --- |
| 1. Objectives | - 01 |
| 2. Introduction of Salesforce and Assigned topic | - 02 |
| 3. Design –Schema Diagram/Use Case/ER Diagrams | - 03 |
| 4. Implementation-Steps/Procedure | - 04 |
| 5. Advantages/Disadvantages | - 11 |
| 6. Result/Screenshots | - 12 |
| 7. Summary | - 13 |
| 8. References | - 13 |
| II. **Problem Statement: 02** | - 14 |
| 1. Objectives | - 14 |
| 2. Introduction of Salesforce and Assigned topic | - 20 |
| 3. Design –Schema Diagram/Use Case/ER Diagrams | - 21 |
| 4. Implementation-Steps/Procedure | - 22 |
| 5. Advantages/Disadvantages | - 27 |
| 6. Result/Screenshots | - 28 |
| 7. Summary | - 29 |
| 8. References | - 29 |

# PROBLEM STATEMENT 01

Add Reports and Dashboards to a Travel Approval- Use reports and dashboards to analyze your travel approvals any sample data for source to destination and the passenger details.

**OBJECTIVES**

Through this project we will learn to-

* Load Data Using the Data Import Wizard.
* Create a Travel Requests by Department Report.
* Create a Travel Requests by Month Report.
* Create a Travel Approvals Dashboard.

## INTRODUCTION

## Salesforce is a cloud-based software company that provides its customers with a platform to develop their own applications without following the tough steps that they used to follow in the legacy system. The software or application once created can be uploaded onto the cloud allowing the end-users to view them.

## Salesforce is currently providing various software solutions and platforms for developers to create and distribute custom software/applications. Tech giants like Google, Twitter, Amazon, and Facebook are using Salesforce either in the form of SaaS or PaaS.

## How great would it be to get a report in your inbox every morning that shows the number of open travel requests waiting on your approval? Or access a report that provides all the key information of your travel approvals across your departments and their status? And see that information displayed as a graphical chart via a dashboard from your desktop or a mobile device? Good news because that is all possible.

## DESIGN –SCHEMA DIAGRAM

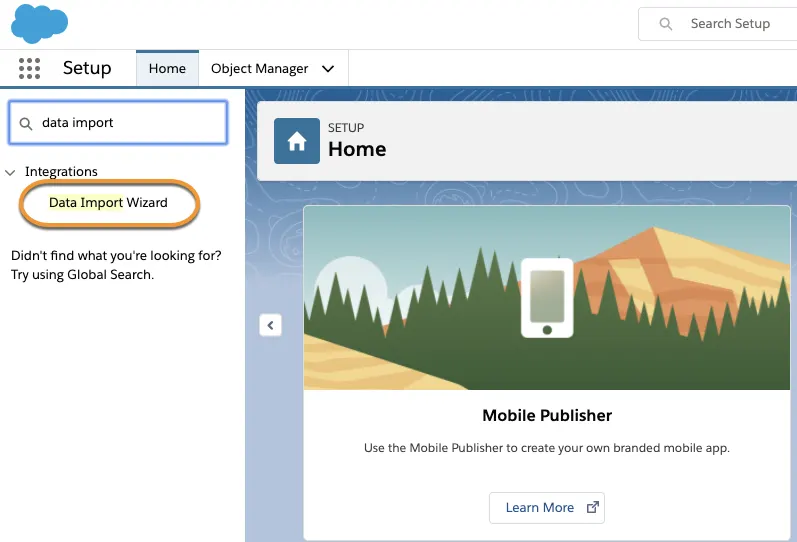
## 

**IMPLEMENTATION-STEPS/PROCEDURE**

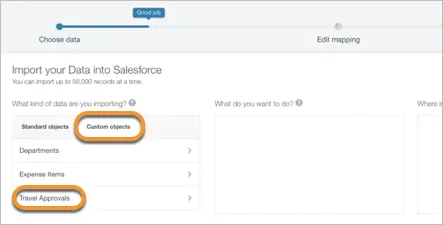
**Load Data Using the Data Import Wizard**

**Embed in Lightning Experience and Beyond**

1. Click and select Setup.
2. In the Quick Find search box, enter Data Import
3. Select Data Import Wizard.



1. Select Launch Wizard!
2. Select the Custom Objects tab and select the Travel Approval object.

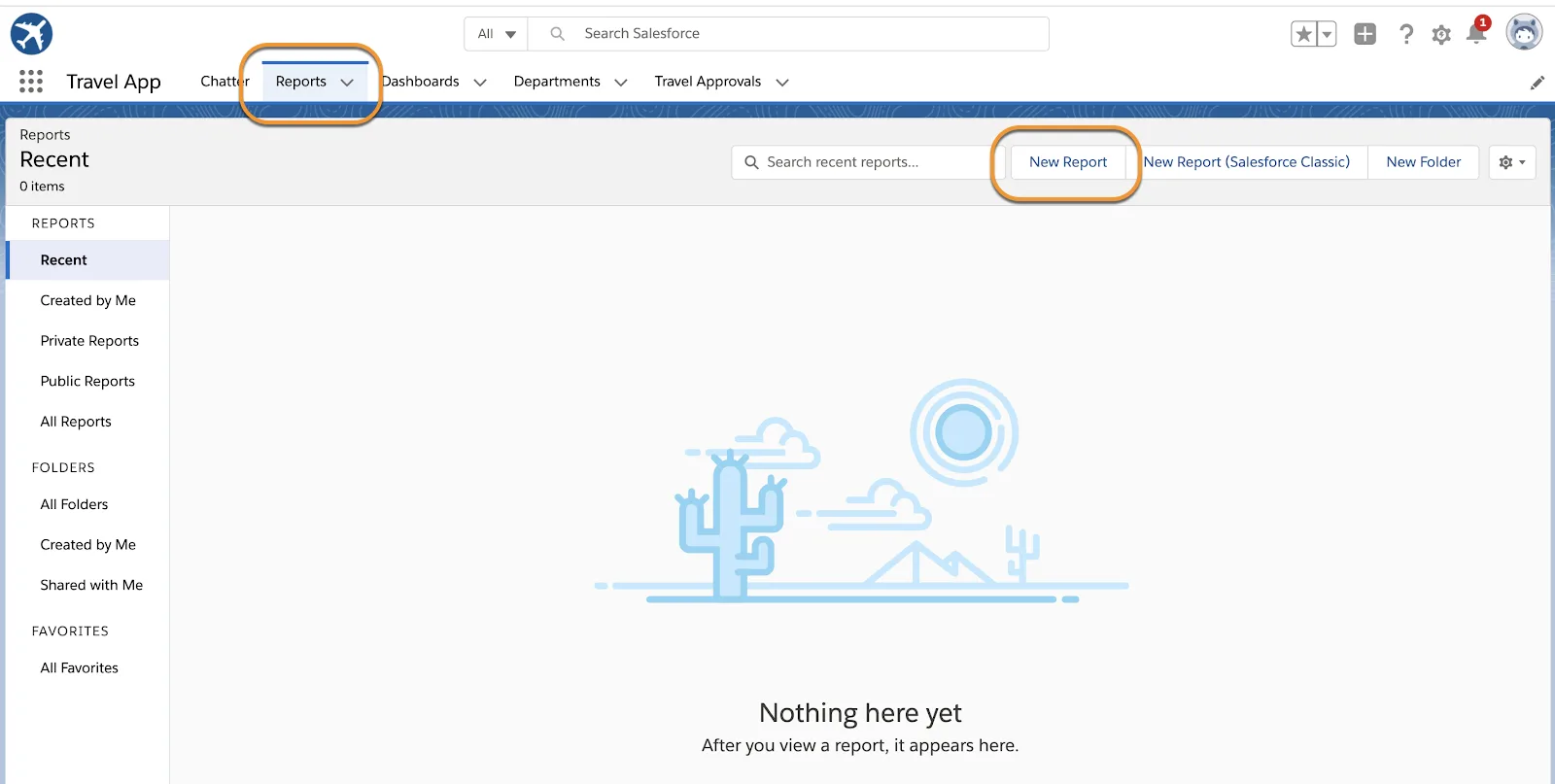


1. Select Add new records.
2. On the Add new record screen, enter the following values.
3. Drag the TravelApprovals.csv file (from the folder you downloaded in the Build a Data Model for a Travel Approval App project) to the Drag CSV file here to upload section.
4. Select Next.
5. Ensure the names in the CSV file are the same as your object (these are automatically mapped).
6. Click Next.
7. Click Start Import.
8. Click OK.

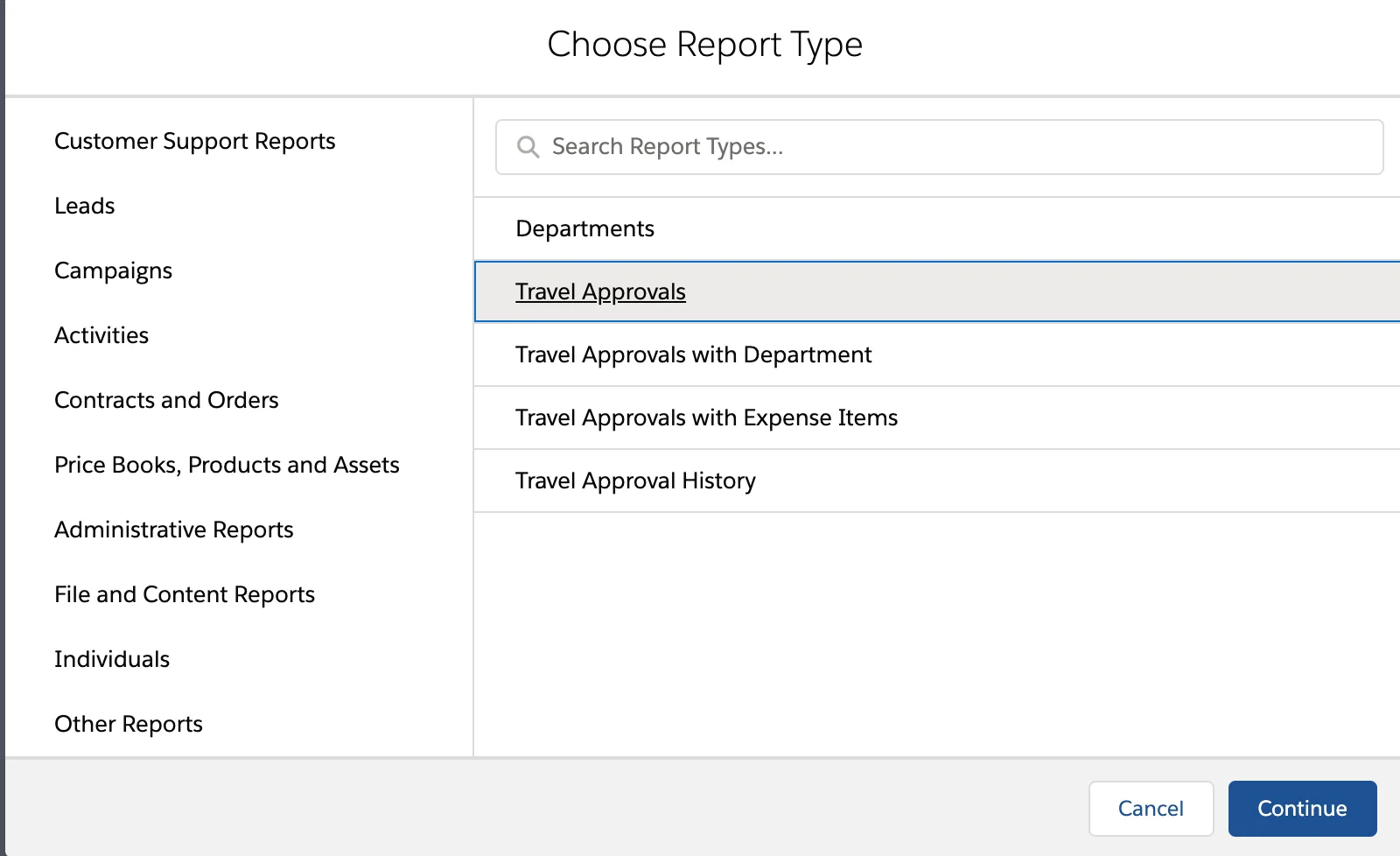
**Create a Travel Requests by Department Report Select the Report Type**

Next, we create a report to show the number of travel approval requests by department.

1. Via the App Launcher, navigate to the Travel App and click the Reports tab.



1. Click New Report.
2. In the Choose Report Type screen, open the Other Reports folder and select Travel Approvals.

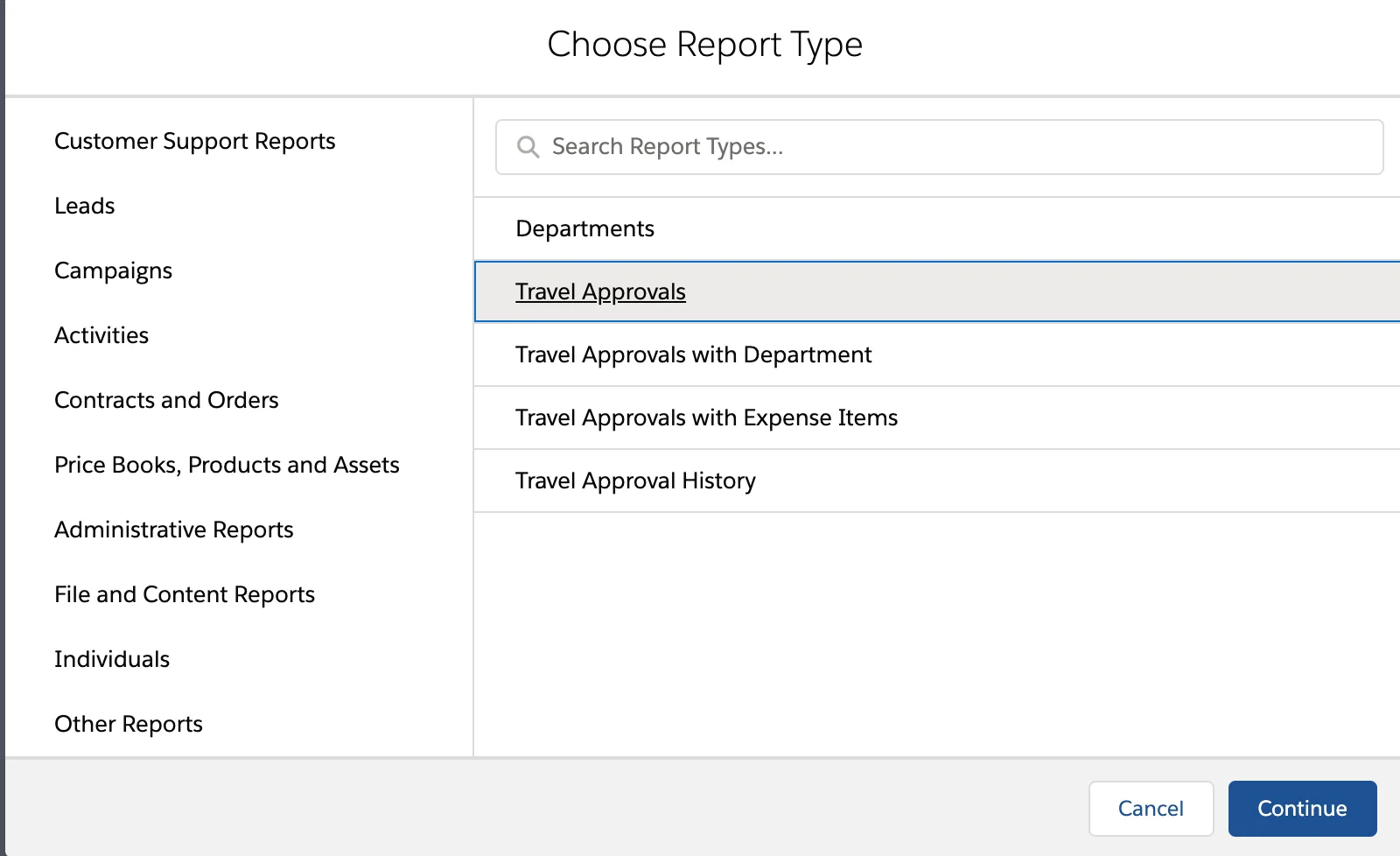


1. Click Continue.
2. The report will now be grouped by Department.
3. Click Save and set the following parameters.
4. Click Save.
5. Click Run.

**Create a Travel Requests by Month Report**

Let’s create another report for our dashboard.

1. From the Reports Tab, click New Report.
2. From the Choose Report Type screen, open the Other Reports folder and select Travel Approvals.

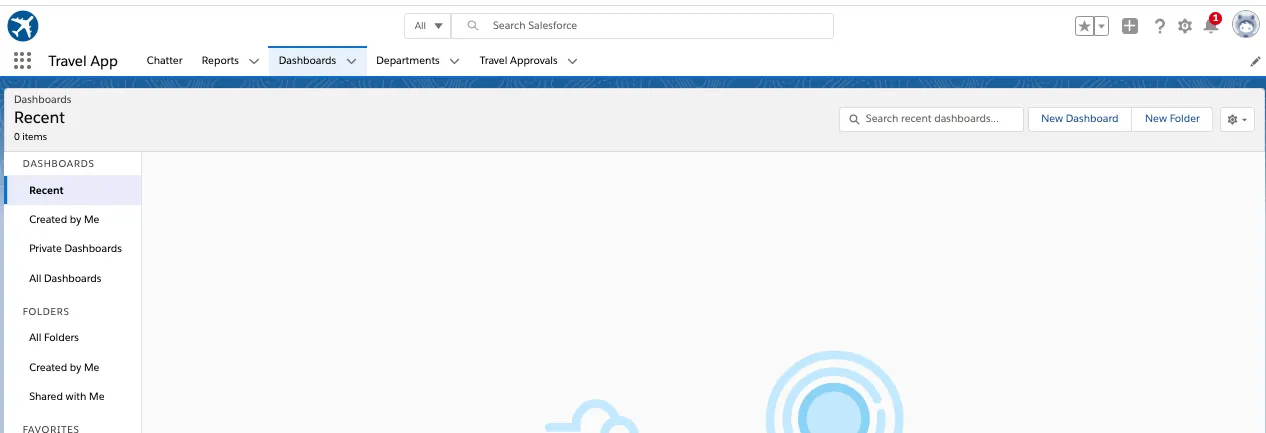


1. Click Continue.
2. The report is now grouped by Calendar Month and then Out-of-State flag.
3. Click Save and set the following parameters for the report.
4. Click Save.
5. Click Run.

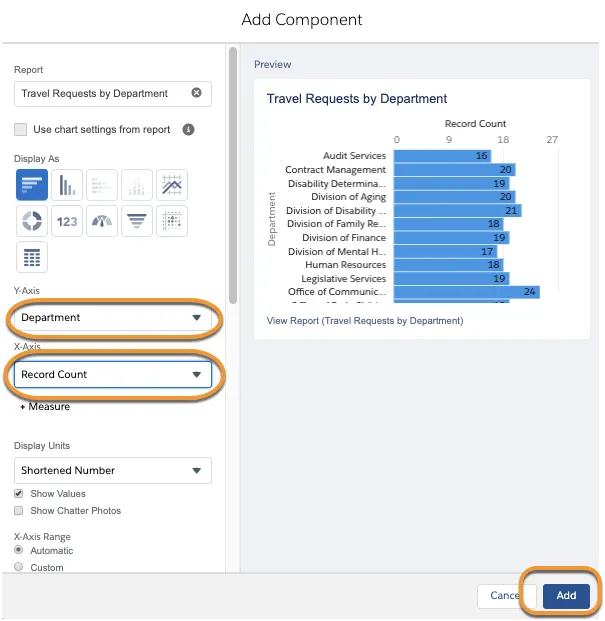
**Create a Travel Approvals Dashboard Create a Dashboard**

You can take your analysis to the next level by placing your reports on a Salesforce dashboard for quick and easy viewing. A dashboard provides an interactive visual display of key metrics and trends. Multiple dashboard components can be shown together on a single dashboard layout, creating rich visual displays of multiple reports that have a common theme.

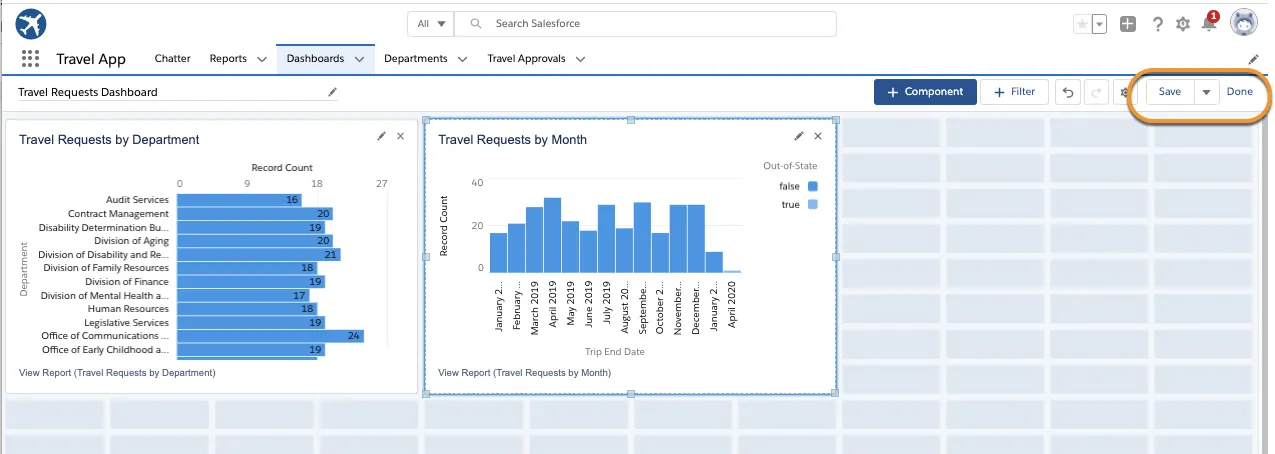
1. Click the Dashboards tab.
2. Click New Dashboard.



1. Enter the following values.
2. Click Create. You can now add reports to your dashboard and move them on to different sections of the dashboard. You can also stretch the components across the grid to have the exact layout of components you need for your dashboard.
3. Click + Component.
4. Click the Travel Request by Department report and click Select. You are presented with options to configure this component for the dashboard.
5. Make sure the Y-Axis is set to Department and the X-Axis is set to Record Count.



1. Click Add.
2. Click + Component to add our other report.
3. Select the Travel Requests by Month report and click Select.
4. Let’s make this a vertical, stacked bar chart that sorts data from oldest to newest by adjusting the following: [2x2 table]
5. Display As | Stacked Vertical Bar Chart
6. Sort By | Trip End Date
7. Click Add.
8. Drag the chart on the bottom, and position it to the right of the first chart we added.



Notice that you have a very flexible grid structure. Play around with stretching your dashboard components to various shapes on the grid.

1. Click Save and then the Done. Your dashboard will look like the following.

For real-world scenarios, you can add other reports to the dashboard and show data and key performance indicators (KPIs) from multiple objects from your database

## ADVANTAGES

## It keeps your team on track. An in-depth sales report might not always be what your team needs to stay on pace to meet its goals. The top-level information that a Travel Approval dashboard presents can be enough to tell your salespeople what needs to get done now and what can wait, saving the time and effort of analyzing detailed reports.

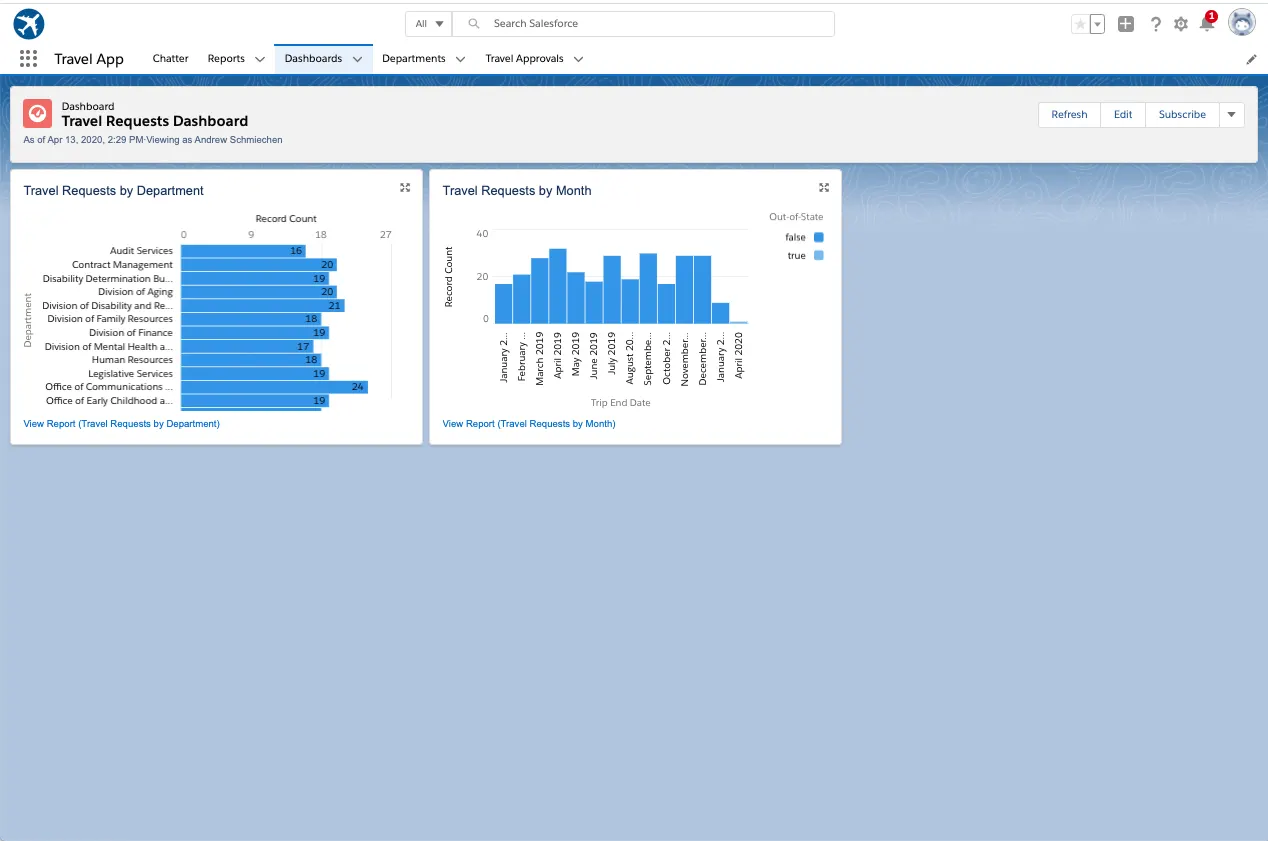
## It streamlines your team’s work. Analyzing reports isn’t the only sales activity that takes time – so does generating those reports. With Travel dashboards, your team can rapidly determine which tasks should be done first without clicking through your dashboard to pull up reports. This function can save tremendous amounts of time.

## It results in stronger sales processes overall. Since Travel dashboards bring your sales team’s work into the present tense, they result in better overall sales performance. A current picture of your sales pipeline and your prospects’ progress along it is far more meaningful to your team than data from an old report, so Travel dashboards are especially powerful in guiding your sales team.

## DISADVANTAGES

1. The integration part is time consuming.
2. There can be some restrictions on the embedding of the dashboards.
3. Users/business teams should have Salesforce Developer accounts and authorize appropriate permissions to set up and work with Travel Analytics Dashboard.

## RESULT/SCREENSHOTS

****

**SUMMARY**

Through this project we learned to –

* Embed a dashboard in a Lightning Experience home page.
* Embed a dashboard in a Lightning Experience account page.
* Use a filter to show only data that’s relevant for a particular record.

**REFERENCES**

* https://www.businessnewsdaily.com/16069-crm-dashboard.html
* https://trailhead.salesforce.com/content/learn/projects/embed-an-einstein-analytics-dashboard-in-lightning-experience
* https://trailhead.salesforce.com/content/learn/projects/embed-an-einstein-analytics-dashboard-in-lightning-experience/embed-an-analytics-dashboard-in-a-home-page
* https://trailhead.salesforce.com/content/learn/projects/embed-an-einstein-analytics-dashboard-in-lightning-experience/filter-an-embedded-dashboard

# PROBLEM STATEMENT 02

Communicate between Lightning Web Components-Explore different approaches to communicating between Lightning web components.

**OBJECTIVES**

Through this project we will learn to-

* Communicate from Child to Parent.
* Communicate from Parent to Child.
* Communicate Between Unrelated Components.

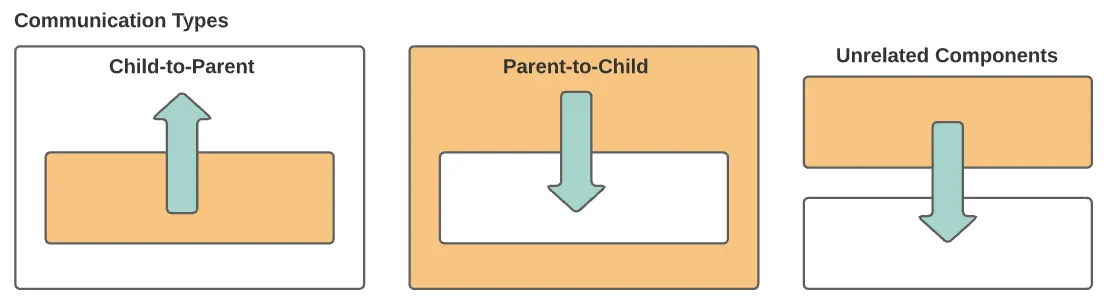
## INTRODUCTION

Salesforce is a cloud-based software company that provides its customers with a platform to develop their own applications without following the tough steps that they used to follow in the legacy system. The software or application once created can be uploaded onto the cloud allowing the end-users to view them.

Salesforce is currently providing various software solutions and platforms for developers to create and distribute custom software/applications. Tech giants like Google, Twitter, Amazon, and Facebook are using Salesforce either in the form of SaaS or PaaS.

When multiple Lightning web components compose an app, we often want those components to share information. How we communicate from one component to another depends on whether and how the components are related. A component inside another component creates a parent-child relationship. How a parent communicates with a child is different from how a child communicates with its parent. And those are both different from how unrelated components (components in separate DOM subtrees) communicate with one another.

## DESIGN –SCHEMA DIAGRAM/USE CASE/ER DIAGRAMS



**IMPLEMENTATION-STEPS/PROCEDURE**

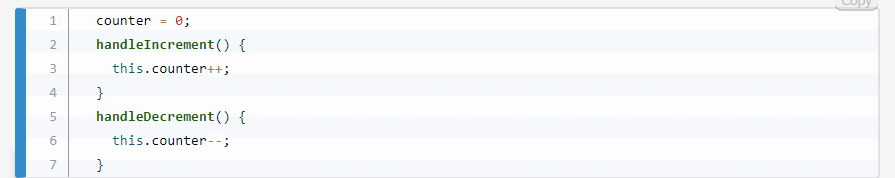
**Communicate from Child to Parent**

For this project, you need to create a new Trailhead Playground. Scroll to the bottom of this page, click the playground name, then click Create a Trailhead Playground. It typically takes 3–4 minutes to create a new Trailhead Playground.

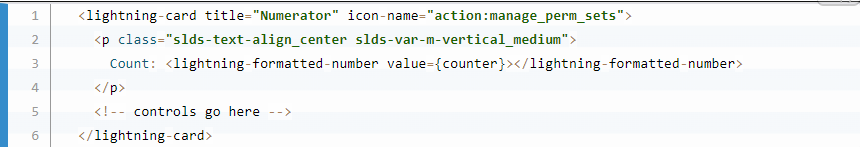
Note: Yes, we really mean a brand-new Trailhead playground! If you use an existing org or playground, you can run into problems completing the challenges.

Let's get started. Go to your Trailhead Playground. (If it’s not already open, scroll to the bottom of this page and click Launch.) If you see a tab in your org labeled Get Your Login Credentials, great! Skip ahead to step 1. Otherwise, from the App Launcher, find and open Playground Starter and follow the steps

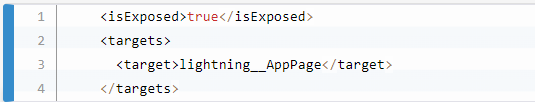
1. Create a three-region Lightning app page named Event Comms:
2. In your org (your Trailhead Playground), open Setup.
3. In the Quick Find box, enter Lightning App Builder and then select Lightning App Builder.
4. Click New.
5. With App Page selected, click Next.
6. For Label, enter Event Comms and then click Next.
7. Select Three Regions and then click Finish.
8. Click Save, Activate, Save, and Finish.
9. Click <- to exit Lightning App Builder.
10. Create the parent Lightning web component named numerator:
11. In Visual Studio Code, under force-app/main/default, right-click the lwc folder and select SFDX: Create Lightning Web Component.
12. Enter numerator for the name of the new component.
13. Press Enter and then press Enter again to accept the default force-app/main/default/lwc.
14. Code the numerator component files:
15. In numerator.js, paste this code inside the Numerator class



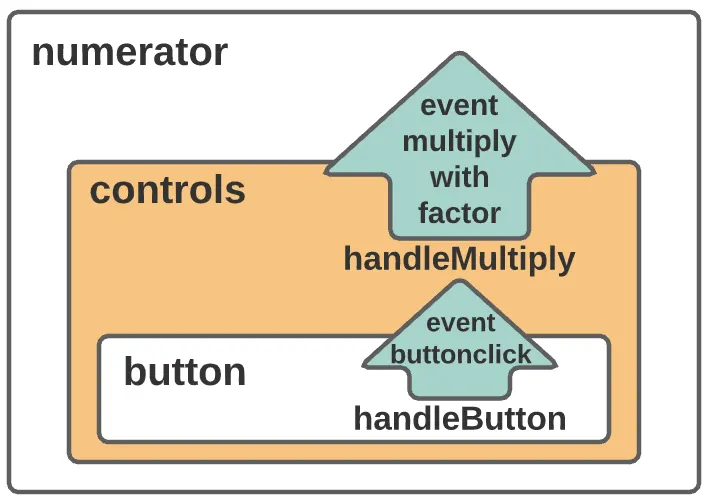
1. Save the file.
2. Open numerator.html and add this code between the template tags



1. Save the file.
2. To make the numerator component available in Lightning app pages, open numerator.js-meta.xml and replace the isExposed tag with these lines:



1. Save the file.
2. Verify communications:
3. To see the changes in Salesforce, deploy the lwc folder and then refresh the Event Comms app page.
4. Click Add and see the count increase.
5. Click X 2 and notice that nothing happens. Why? By default, a custom event doesn’t bubble up past the component that dispatched it.
6. Save the file and deploy it again.
7. Refresh the Event Comms app page.
8. Now click Add to set the count to 1.
9. Click X 2 to see the count multiplied by 2.
10. Click X 6 to see the count multiplied by 6.
11. Click X 0 to reset the count to zero.
12. Try the following clicks: Add, X 2, X 5, X 2, Add, X 2. Or: Add, X 3, X 2, X 2, Subtract, X 2, Subtract, X 2. Or try your own combination.



**Communicate from Parent to Child**

To enable communication from a parent component to a child component, the child exposes a property or function to make it public. Then the parent can update the child’s public property or call the child’s public function.

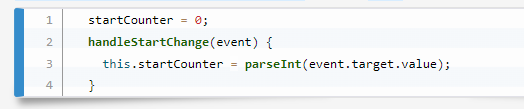
Additionally, if you want to add some functionality, update the public property to a getter and setter on the child component.

Let’s start with a simple public property update. Another business unit found the Numerator you built. They want to use it and add to it. Their first request is to be able to set the starting number of the counter. We don’t want to make any changes that will affect the original business use case, so let’s wrap the numerator component in another component, which will hold the new functionality

**Update a Public Property**

The @api decorator in the child component exposes a property, making it public, so that the parent component can update it.

1. Expose a public property in the child component (numerator):
2. In Visual Studio Code, open numerator.js and apply the @api decorator to the counter property
3. Import the api decorator from the lwc module.
4. Save the file
5. Create and code a new parent component named augmentor:
6. Create a Lightning web component named augmentor.
7. In augmentor.js, paste this code inside the Augmentor class



1. Save the file.
2. Open augmentor.html and add this code between the template tags

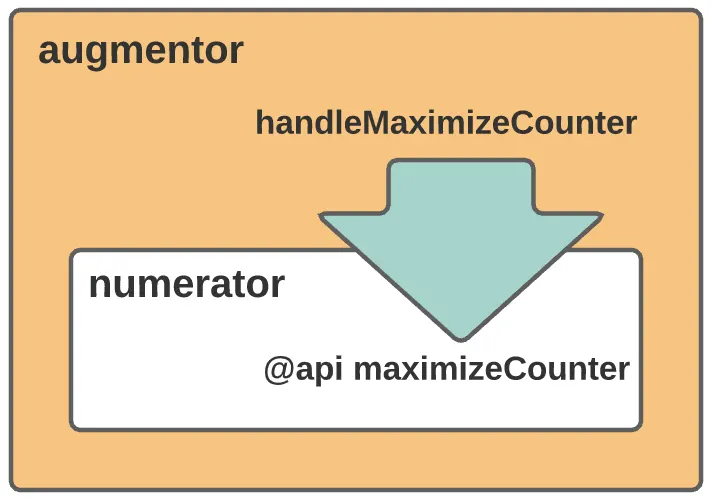


1. Save the file.
2. Update augmentor.js-meta.xml to make the augmentor component available on Lightning app pages.
3. Save the file.
4. Add the new component (augmentor) to the Event Comms app page:
5. Deploy the lwc folder and then refresh the Event Comms app page.
6. Open the Event Comms app page for editing.
7. Drag the augmentor component to the center region of the page.
8. Click Save and then exit Lightning App Builder.
9. Verify communications:
10. To see the changes in Salesforce, refresh the Event Comms app page.
11. Enter a number in the Set Starting Counter field.
12. The count updates to what you entered.
13. Click one of the multiply buttons.
14. Notice that the counter updates but the Set Starting Counter stays the same.
15. Change values in the original numerator component.
16. It continues to work as expected.

**Call a Public Function**

The second request from the business is to bump the count by one million. They don’t want the Set Starting Count to change. This means we can’t just update the startCounter property. We also don’t have the current count in the augmentor component to add to. We’ll call a public function on the child to do the update for us.

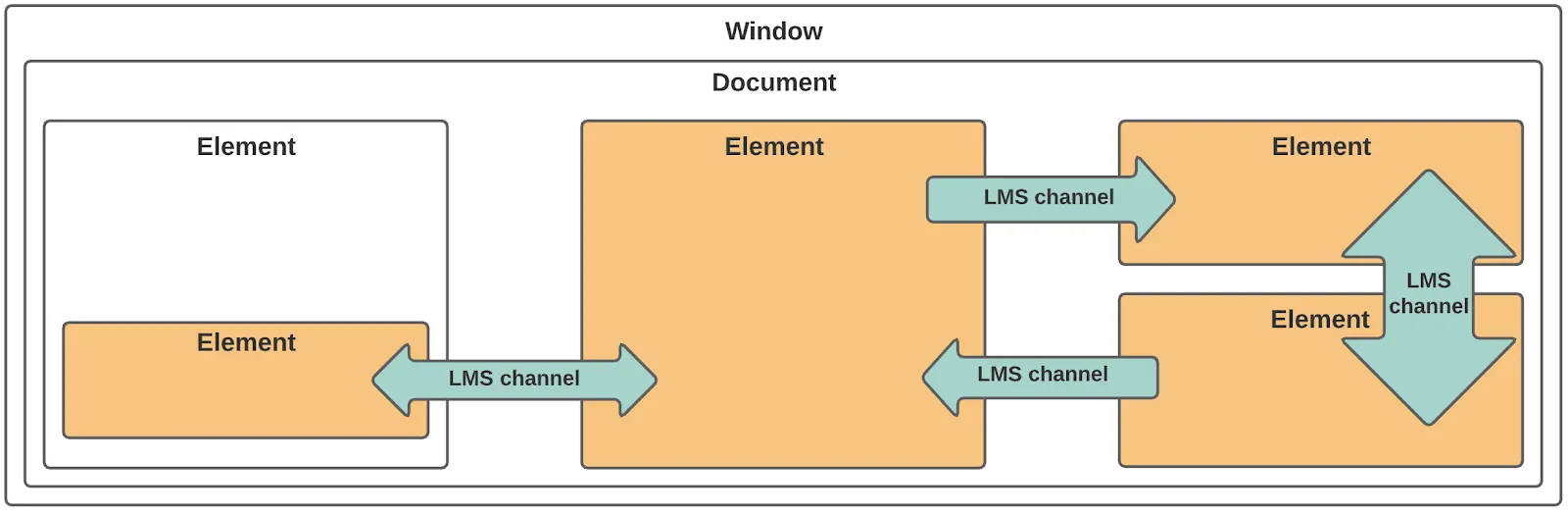
1. Create a public function in the child component (numerator):
2. Open numerator.js and add this maximizeCounter function after the handleMultiply function:
3. Save the file.
4. In the parent component (augmentor), add a button and its handler:
5. Open augmentor.js and add this handleMaximizeCounter function after the handleStartChange function
6. his function finds the c-numerator tag in augmentor.html and calls the public maximizeCounter function.
7. Save the file.
8. Open augmentor.html and add this lightning-button after the Set Starting Counter lightning-input
9. Save the file.
10. Verify communications:
11. To see the changes in Salesforce, deploy the lwc folder and then refresh the Event Comms app page.
12. Click Add 1m To Counter.
13. The count is increased by one million.
14. In the parent (augmentor), the new button triggers the handleMaximizeCounter handler, which calls the child component (numerator) and triggers its public maximizeCounter function.



**Communicate Between Unrelated Components**

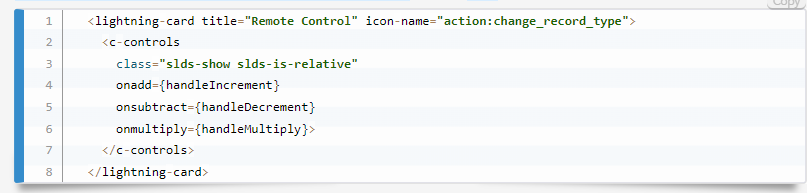
To communicate across the subtrees in the DOM (and under some circumstances between different browser windows logged into the same org), use Lightning message service (LMS). LMS is a publish and subscribe service that facilitates communication between Lightning web components, Aura components, and Visualforce pages.

Use LMS for communication between unrelated components unless you control both components and a common parent. LMS is powerful, effective, and easy to use, but don’t let that tempt you to use it when it’s not necessary. Firing DOM events is much more efficient. When you need to communicate between components with a parent that you can’t control, such as two App Builder slots, Lightning message service is the perfect choice.

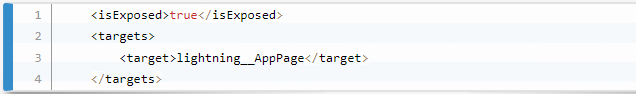


**Create a Lightning Message Channel**

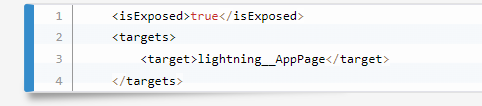
1. In Visual Studio Code, in the default folder, create a folder named messageChannels.
2. In the messageChannels folder, create a file named Count\_Updated.messageChannel-meta.xml.
3. In Count\_Updated.messageChannel-meta.xml, paste this code:
4. Save and deploy the file.
5. Create a Lightning web component named remoteControl.
6. Replace the contents of remoteControl.js with this code:
7. We import publish and MessageContext from the Lightning message service. We also import the channel we just created (Count\_Updated\_\_c). The data payload is sent with the publish function.
8. Save the file.
9. Open remoteControl.html and add this code between the template tags



1. Notice that we’re re-using the controls component.
2. Save the file.
3. Update remoteControl.js-meta.xml to make the remoteControl component available on Lightning app pages:



1. Create a Lightning web component named counts.
2. Replace the contents of counts.js with this code
3. @wire(MessageContext) ensures that unsubscribe runs during the component destruction lifecycle.
4. Save the file.
5. Open counts.html and add this code between the template tags:
6. Save the file.
7. Update counts.js-meta.xml to make the counts component available in Lightning app pages:



1. Save the file.
2. Add the New Components to the Event Comms App
3. Deploy the lwc folder and then refresh the Event Comms app page.
4. Open the Event Comms page for editing.
5. Drag the remoteControl component to the right region of the page.
6. Drag the counts component to the right region, just under the remoteControl component.
7. Click Save and then exit Lightning App Builder.

**Verify communications**

Click the buttons in Remote Control to change the count in the counts component.

You can use Lightning message service to communicate with Aura components and Visualforce pages as well. It’s the cleanest way to keep different types of components in sync.

Now you have a good working knowledge of how to communicate between Lightning web components. You worked through child-to-parent, parent-to-child, and unrelated component scenarios. Be sure to check out the resources to learn more about communicating with Lightning web components.

## ADVANTAGES

* Tracks internal requests for headcount as they move through the approval process
* Analyzes candidate demographics to identify new-hire recruiting patterns
* Monitors the number of applicants for each open position and the status of each of those applicants
* This application will remove the manual creating multiple sheets which will consume lot of time and hard work and also automate the recruitment process and make it easy and simple.
* Admin can access the list of candidate and also can select the candidates who fulfill the requirement of the organization.
* This application will help us to keep the details of the candidates in each step of selection process, the list of candidate who got selected in each step of recruitment process.

## DISADVANTAGES

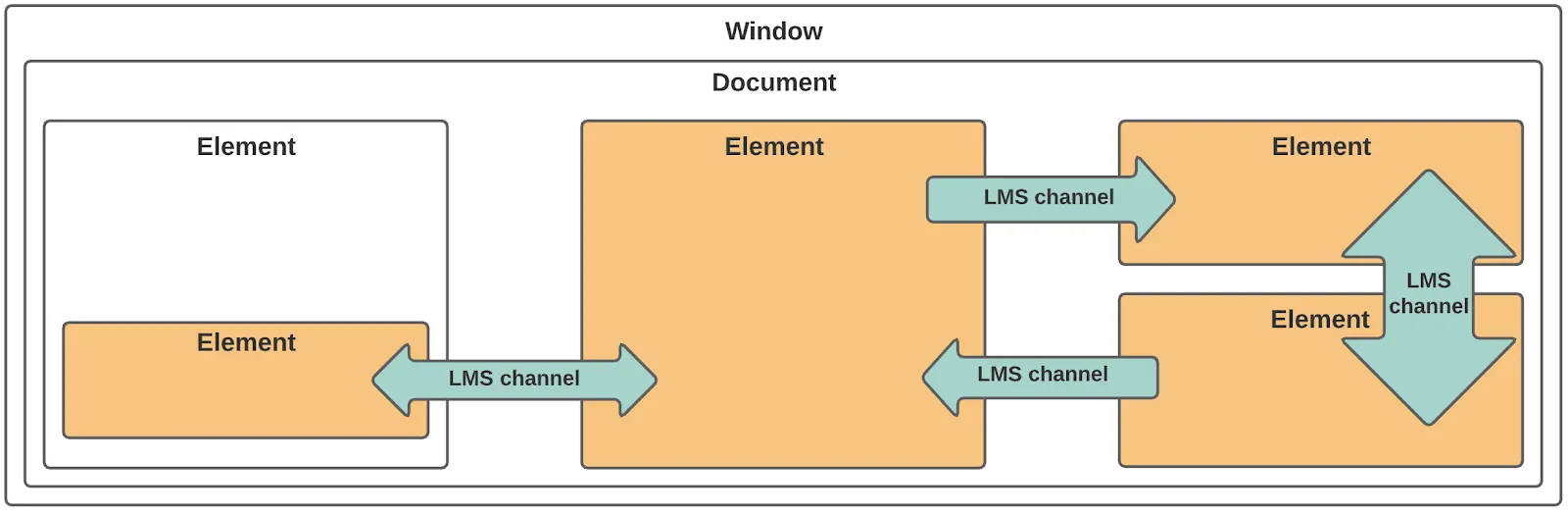
## The recruitment application project has certain limitations such as-

## In this application the list of selected students in each round of interviews is still done manually

## HR selects the list and creates the emails manually.

## Another limitation is that there must be an active and stable internet connection and a browser application for the users to make use of this application.

## RESULT/SCREENSHOTS



**SUMMARY**

Through this project we learned to –

We have tackled both child-to-parent and parent-to-child communication between Lightning web components. In the next step, you use the Lightning message service to communicate between components that don’t have a parent-child relationship.

## REFERENCES

* [*https://trailhead.salesforce.com/en/content/learn/projects/automate-business-processes-recruiting-app?trail\_id=build-platform-apps-in-lightning-experience*](https://trailhead.salesforce.com/en/content/learn/projects/automate-business-processes-recruiting-app?trail_id=build-platform-apps-in-lightning-experience)