Michael Aylesbury

Based in Edinburgh • moaylesbury@gmail.com • 07483833014

ABOUT ME

I am a hardworking, passionate and empathetic person, with demonstrated experience as a Salesforce developer. I am confident in producing high quality Lightning Web Components and Apex code, understanding business logic, translating technical jargon and training agents in component use. In my own time I enjoy weightlifting, music production and cooking.

EDUCATION

The University of Edinburgh — *Computer Science (BSc Hons)*

2017 - 2022

North Berwick High School — *Advanced Higher (x4)*

2011- 2017

EMPLOYMENT HISTORY

Greengage — *Salesforce Developer*

September 2022 - Present

Write code according to specification, design user interfaces and review code accuracy and functionality. Analyse data to ensure quality of projects. Maintain the code base and unit tests, work closely with others in the development team to drive projects forward, communicate with stakeholders as needed and explain technical jargon in a clear way. Present projects to the business and provide training on how to use them. Manage projects through Wrike.

Bright Network — *Intern*

June 2021 - June 2021

Received seminars from Amazon, Google, Vodafone, EY, and other firms. Briefed by Google, implemented core YouTube functionalities as a command line application using Python. Took part in workplace skill sessions, software development upskilling and had daily networking opportunities.

AWARDS AND CERTIFICATIONS

Lightning Web Components Specialist — *Salesforce Trailhead Superbadges*

October 2022

Bright Internship Certificate of Completion — *Bright Network*

July 2021

JPMorgan Chase & Co. Empowering Minorities Award — *Hack the Burgh V*

March 2019

PROJECT HISTORY

Below are two examples of projects I have completed while at Greengage. The following Salesforce specific acronyms are used: Salesforce Object Query Language (SOQL), Salesforce Object Search Language (SOSL) and Lightning Web Component (LWC).

Reconciliation Lightning Web Component (Apex, JavaScript, HTML, SOQL, Approval Processes)

Problem: Agents needed to track bank account balances and payment transaction amounts across ledgers to ensure they match. If they did not match, they needed the ability to investigate and fix this.

Solution: I created a reconciliation object in Salesforce; a header record providing a summary of the child reconciliations for the day (one child for payment reconciliations, and one for account balances), as well as an Apex REST API endpoint. A HTTP PUT request is made to the endpoint from the backend nightly to create a reconciliation object for the day's transactions and balances. Next, I made a custom Reconciliation Viewer Lightning Web Component. Two lightning inputs allow for a date range to be selected. Using a SOQL query and applying pagination to the results, reconciliation records are displayed in a lightning datatable. Upon selecting a reconciliation record, child reconciliations are fetched from the backend REST API using a HTTP GET request. The children are also displayed in lightning datatables, presented below the parent datatable in a lightning tabset. All reconciliations have a row action to get further details, and child reconciliations have two more row actions, allowing agents to change payment/balance reconciliation status and escalate via HTTP PATCH callouts. I also put an approval process in place, allowing for reconciliation of header records after all children are escalated or reconciled. This LWC is displayed in two locations: on a lightning page allowing access to all header records, and also on each header record page, preselecting the current record.

Outcome: I ran a training session and demonstration for the Chief Technical Officer, Chief Operating Officer, Operations Manager and operations agents. Following this training session, daily reconciliations across the bank are now carried out using this tool.

Account Merge Lightning Web Component (Apex, JavaScript, HTML, SOQL, SOSL)

Problem: The ability to merge any two accounts of the same record type was required as a custom component.

Solution: I created a quick action LWC on the account record page, with an account selection stage and field-value selection stage. The account selection stage takes text (partial or full account name) as input, executes a wildcard SOSL query for accounts partially matching the input name and record type, and displays results with preview information (type, owner, created date). After selecting an account to merge with, account fields are dynamically retrieved from schema. Fields that are updateable, accessible and not calculated are processed. Iterating over each field, an Aura Enabled "comparison entry" class is populated with the field name, parent values, child values, and alternate values. Parent values are any values stored in the current field on the source account, and child values are the same but from the selected account. If the current field is of display type Phone or Email, all other field values of the same data type (that are not already stored in parent or child values) are fetched and stored as alternate values. Lookup field values are converted from record ids to names (fetched dynamically via the object's standard name field). Null values are displayed to the user as "*[Clear Field]*" and can be toggled visible if values both are null or matching. I created a "Greengage Datatable" component, extending lightning datatable. This was used for displaying the comparison entries, specifically for alternate values to display multiple radio buttons in a single cell. By default, all parent values are selected. Once the user has confirmed the selected field-value pairs, values are dynamically cast to the appropriate data type (via a display type to data type map), inserted into the parent account, and the merge DML operation is applied.

Solution: Agents with sufficient permissions can now merge any two accounts of matching record type across the organisation, with the ability to select from parent, child and alternate values.