Michael Aylesbury
Certified Full Stack Software Developer

Edinburgh, Scotland · moaylesbury@gmail.com · 07483833014 · LinkedIn · Trailhead

REFERENCES (AVAILABLE ON REQUEST)

Mark Pickering — *Chief Technical Officer, Greengage*

Michael Dalton — *Solutions Architect & Development Lead, Xenogenix*

ABOUT ME

Holding **Salesforce Platform Developer I** and **.Net Full Stack Foundation** certifications, I have a wide range of software development knowledge and can confidently research, design and implement full stack solutions. With consulting experience, I am able to effectively communicate with stakeholders, maintaining relationships and providing long-term support. I am personally passionate about weightlifting, music production and cooking.

TECHNOLOGIES

Salesforce, Apex, HTML, Javascript, CSS, Java, Python, C++, C#, REST APIs, SOAP APIs, VSCode, Async Programming, Azure, Web Components, YAML, SQL, DevOps, HTTP, JSON, Git, NetSuite, OAuth, GitHub, Postman, Integrations, Testing, UAT, Springboot, .NET

EDUCATION

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

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

EMPLOYMENT HISTORY

Xenogenix— *Full Stack Software Developer & Consultant*

April 2024 - September 2024 **(6 months)**

- Carried out requirements gathering workshops with clients
- Designed solutions and wrote solution design documents according to requirements
- Implemented solutions, carried out testing and maintained client communication
- Managed projects through Trello
- Logged and maintained timesheets
- Implemented GitHub organisation and upskilled other developers on its use
- Implemented coding standards documentation and put together a GitHub repository containing a wiki, reusable Apex classes and VSCode snippets

Greengage — *Full Stack Software Developer*

September 2022 - April 2024 **(1 year, 8 months)**

- Maintained and extended an internal Salesforce org in a Banking as a Service (BaaS) context
- Carried out deployments and testing, from scratch org to sandbox to production
- Submitted code and user interfaces for review and acted on feedback accordingly
- Maintained and extended Azure integration
- Communicated with stakeholders to gather project requirements
- Presented projects to the business and carried out training and upskilling sessions
- Created an Experience Cloud business-to-consumer (B2C) registration portal from scratch
- Implemented reconciliation tool, transaction viewer, and fee viewer, replacing Excel spreadsheets and significantly improving daily operational efficiency for internal agents

PROJECT HISTORY

Two significant projects I have completed are outlined below: one for Xenogenix, where I led a Salesforce-NetSuite integration, and another for Greengage, where I developed a reconciliation viewer Lightning Web Component.

Salesforce-NetSuite Integration (Batchable and Schedulable Apex, Triggers, SOQL, NetSuite, OAuth)

Task: Integrate NetSuite with Salesforce using the Apex Batchable and Schedulable interface, replacing an existing third-party integration tool in use that is inefficient.

Solution: First I set up relevant NetSuite configurations; an integration record for Salesforce, an access token, an RSA certificate and relevant user permissions. The token and certificate keys were stored in custom metadata in Salesforce. A method was put in place to fetch this metadata and add an OAuth header to each callout.

Next, I created endpoint mapping records, containing endpoint URL, HTTP method, expected response code, and sample JSON bodies for request and response. A NetSuite Id custom field was added to the account, product and user standard objects.

I then updated the opportunity trigger to queue an Apex batch for newly 'Closed Won' records. This batch creates customers in NetSuite from account records associated with the opportunity, populates the NetSuite Id field, and subsequently creates a customer-subsidiary relationship for each custom Salesforce subsidiary object record.

Two schedulable classes were created. The schedulable classes queue a batch for the user and product objects respectively. The Salesforce user object corresponds to the NetSuite employee object, and product to service sale item. Each batch queries the Salesforce objects using SOQL, and the corresponding NetSuite objects using the SuiteQL endpoint. Matching by email for users, and product code for products, Salesforce records are matched with corresponding NetSuite objects, and the NetSuite Id field on the Salesforce records is updated. Any errors are tracked and logged into a custom log record in Salesforce. Governor limits are respected by batch size and DML operations.

Outcome: The client found no errors during acceptance testing and replaced the third-party tool with the provided solution, completing an £18,000 project. Data is now efficiently synchronised between Salesforce and NetSuite.

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

Task: Allow finance agents to track bank account balances and payment transaction amounts across ledgers to ensure amounts match across systems. If they do not match, provide the functionality to investigate.

Solution: I created a reconciliation custom 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, replacing the previously used Excel spreadsheets, improving accuracy and efficiency. I proceeded to implement two similar viewers, a transaction viewer and a fee collection viewer. All three viewers are used daily by finance and operations agents.