

Requirement document - Application flow by Origin App

Flowchart

graph TD

A((Student visits application site)) → B[1 - Fills form]

B → C[2 - Form data saved to SharePoint List]

C → D[3 - Origin App fetches data]

D → D1[5 - Display data using Graph API]

D1 → E[5 - Origin App auto-assesses application and Admin manual check]

E → F{Assessment Result}

F → |All documents OK| G1[6 - Origin app check if All files OK AND LLN test compl

F → |Missing documents| G1

F → |Rejected| H((Send rejection email))

G1 → |Yes| Q1[7 - Local Drive app Send Student application detail to RTO via OLP]

G1 → |NO| Q2[7 - Local Drive app Send Student application detail to RTO via OLP]

Q1 → P1((RTO generates FULL offer by Admin))

Q2 → P2((RTO generates CONDITIONAL offer by Admin))

D → G[4.0 - Origin App sends POST to Local Drive app - same day]

G → J[4.1 - Local Drive app transfer list to CSV]

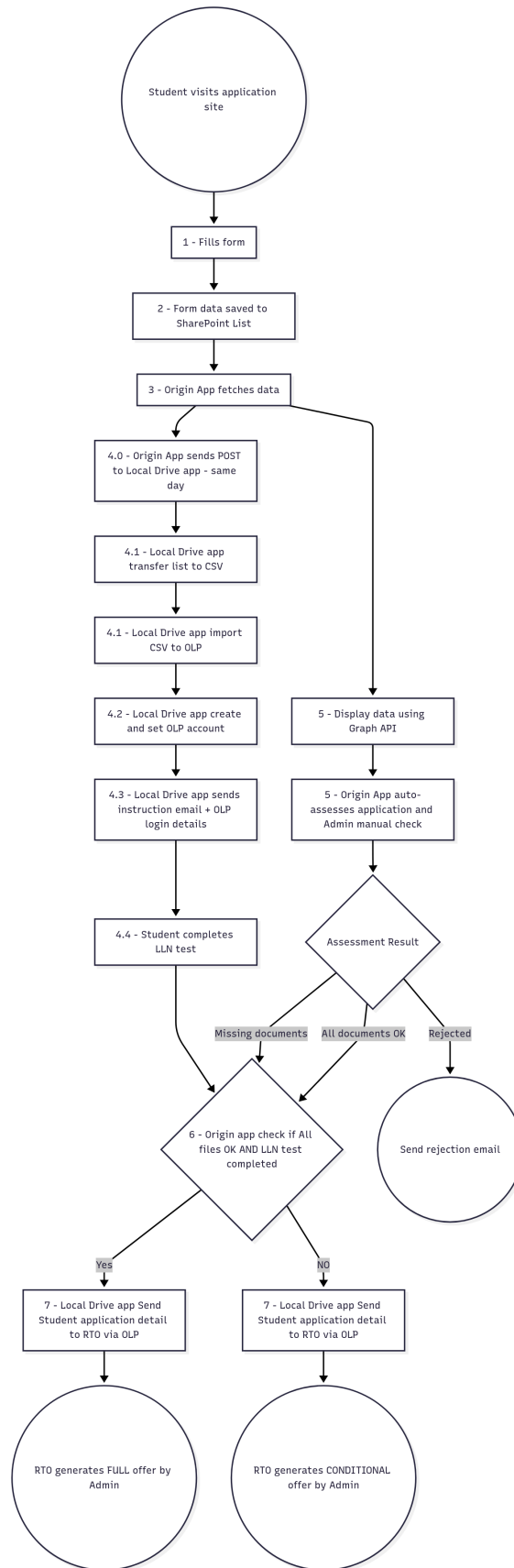
J → K[4.1 - Local Drive app import CSV to OLP]

K → L[4.2 - Local Drive app create and set OLP account]

L → M[4.3 - Local Drive app sends instruction email + OLP login details]

M → N[4.4 - Student completes LLN test]

N → G1



Technique flowchart:

graph TD

A[Student - Application website] → B[Web form - Azure]

B → C[SharePoint List]

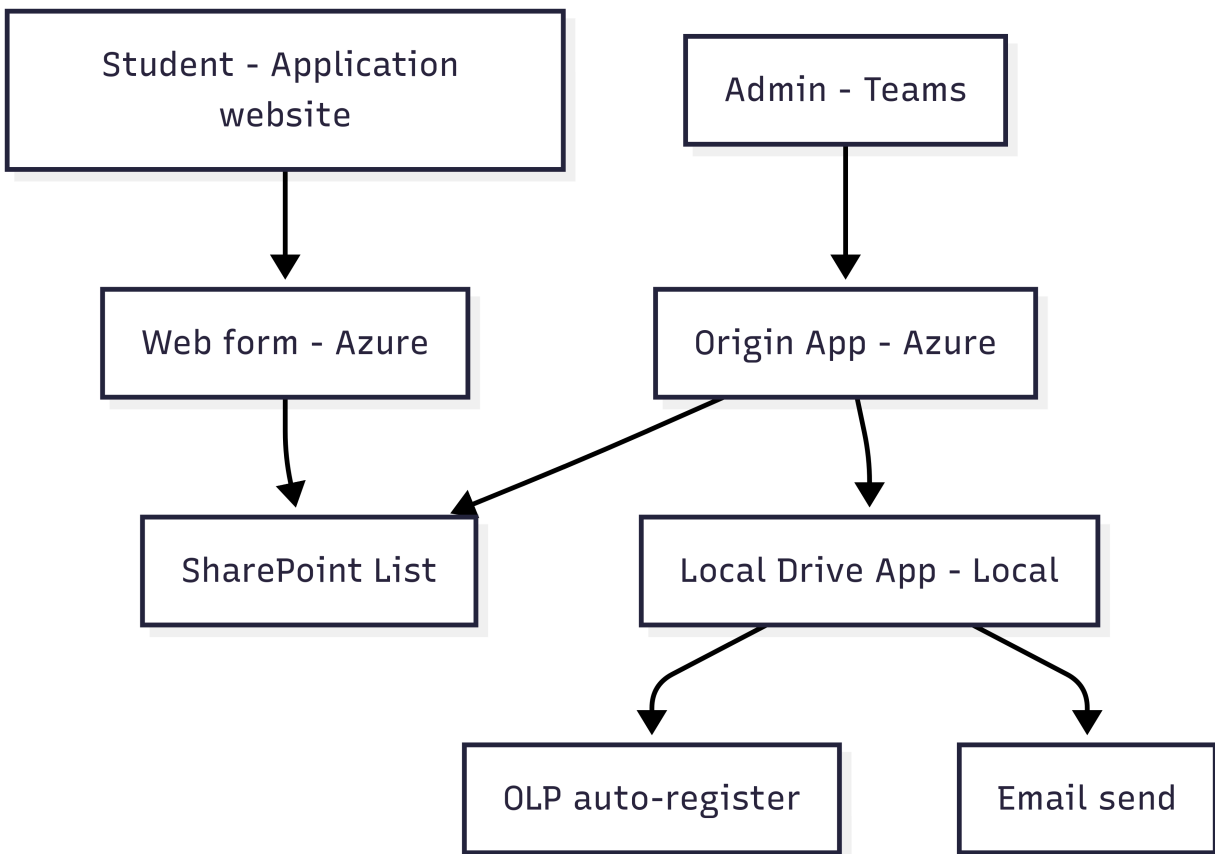
D[Admin - Teams] → E[Origin App - Azure]

E → C

E → F[Local Drive App - Local]

F → G[OLP auto-register]

F → H[Email send]



This project contains:

- **Web form**
- **Teams App (Origin App)**

- **Microsoft Graph / SharePoint**
- **Local Drive App**
- **Ammointe OLP auto-register**

Agent Flow:

same implementation pattern as student application, but use different API to register in RTO if required

graph TD

A((Agent fill Agent application form)) → B[1 - Get Application ID]

B → C[2 - Form data saved to SharePoint List]

C → D[3 - Origin App fetches data]

D → D1[4 - Display agent list using Graph API, student can by filter by Application ID]

D1 → E[5 - Admin assess]

E → F{Assessment Result}

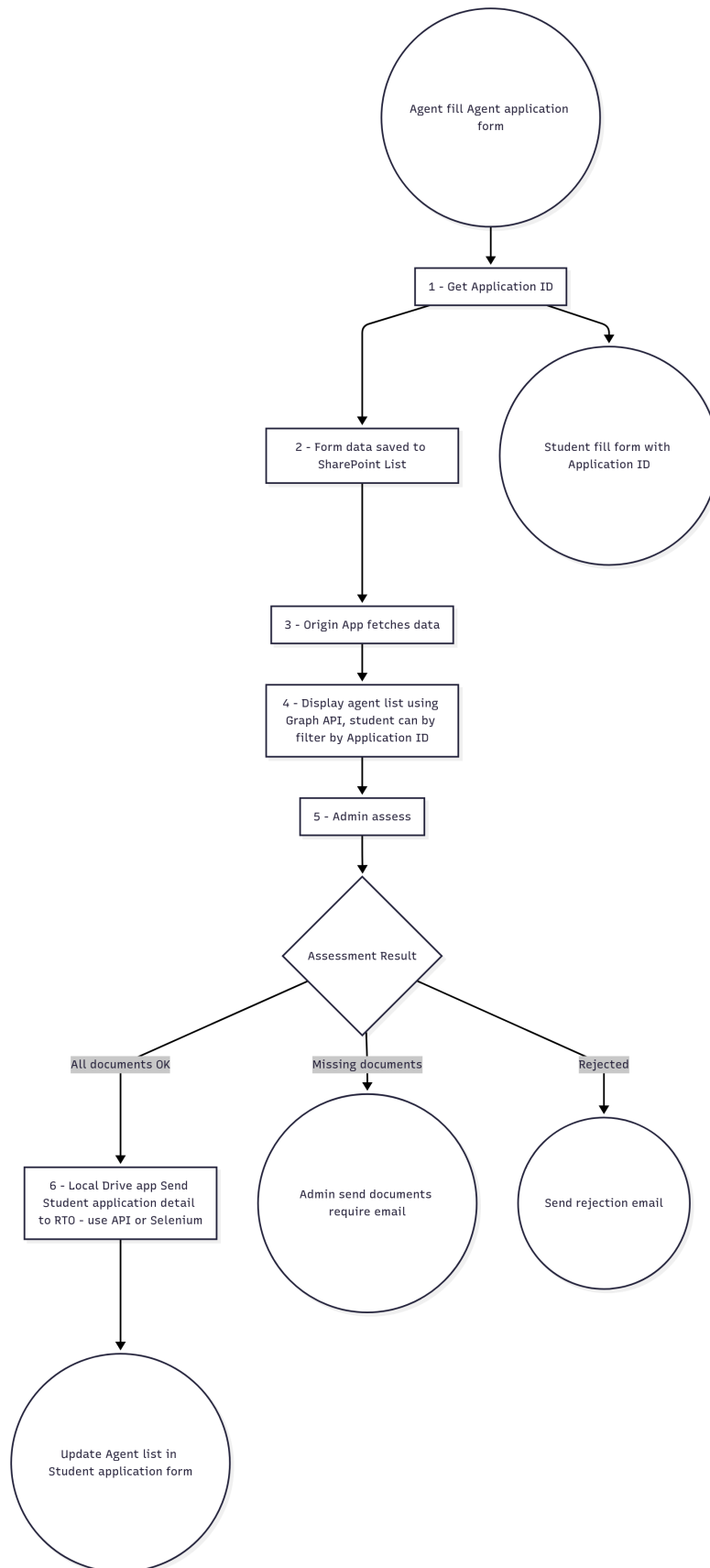
F → |All documents OK| G1[6 - Local Drive app Send Student application detail to RTO]

F → |Missing documents| G2((Admin send documents require email))

F → |Rejected| H((Send rejection email))

B → I((Student fill form with Application ID))

G1 → J((Update Agent list in Student application form))



Full Process Architecture and Implementation Methods for Each Step (Technology + Tools)

MVP Version (Core Functions) - 7-9 Weeks

Phase	Tasks	Estimated Time
Week 1-2	Infrastructure Setup	2 days
	Microsoft 365 environment configuration <i>(Completed)</i>	
	SharePoint List design and creation <i>(Completed-List items still need to confirm with Tiffany)</i>	
	Development environment setup <i>(Completed)</i>	
Week 1-2	Application Form Development	4 days
	React-based form UI development <i>(Completed)</i>	
	Integration with Microsoft Graph API <i>(Completed)</i>	
	Deployment	
Week 3-5	Origin App Development	10 days
	Teams custom app framework	
	Application list display UI	
	Review functionality and status update	
	Teams deployment and testing	
Week 6-9	Local Drive App Development	12 days
	Backend development with Flask/FastAPI	
	CSV generation and processing	
	Selenium automation scripts	
	Local deployment with Docker and ngrok	
Week 10-11	Integration Testing and Optimization	5 days
	End-to-end process testing	
	Error handling and logging	
	Email notification feature	
	Documentation and deployment guide	

✓ 1. Students Submit Applications on the Application Website

📌 Technology

- Does **not** rely on Teams SDK — built with pure **React/HTML**, deployed to Azure

✓ Implementation Approach:

- **Shared Data Storage**

Both applications write to the **same SharePoint List**

Use the same **Microsoft Graph API**

Different **App Registrations** but access the **same resources**

Application 1: Public Form App (for students)

Name: Student Application Form

Deployment: Azure

Access: Public HTTPS link

Purpose: For students to submit application forms

Technology: Pure React/HTML using Microsoft Graph API

Application 2: Origin App (for administrators)

Name: Origin App

Deployment: Custom Teams app

Access: Internal within Teams, requires login

Purpose: Admins view application list, perform reviews, and initiate follow-up processes

Technology: Teams SDK + React, using Microsoft Graph API

- Data is transmitted to the SharePoint List via **Microsoft Graph API** (see Step 3)
- **Data Flow:**

graph TD

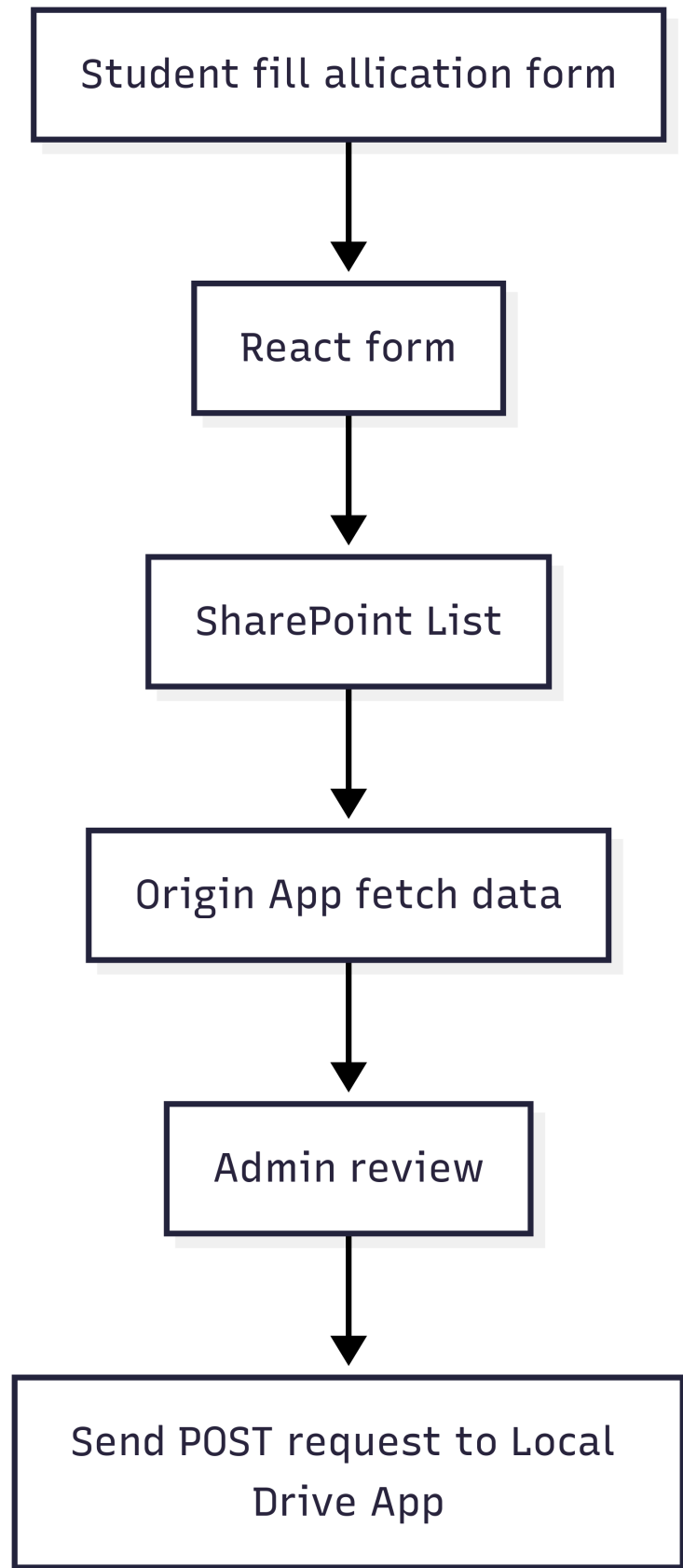
A[Student fill allocation form] → B[React form]

B → C[SharePoint List]

C → D[Origin App fetch data]

D → E[Admin review]

E → F[Send POST request to Local Drive App]



Example structure of a shared SharePoint List (List items still need to confirm with Tiffany)

```
json
{
  "Title": "Student Applications",
  "Fields": [
    "StudentName",
    "Email",
    "Course",
    "Documents",
    "Status",
    "CreatedBy",
    "ReviewedBy",
    "ReviewDate"
  ]
}
```

✅ 2. Submit Form → Origin App Receives Data

📌 Implementation / Technologies:

- The front-end form (built with pure React) submits data to the back-end
- The back-end calls **Microsoft Graph API** to write data into the **SharePoint List**
- When an admin opens the **Origin App**, it calls **Graph API** to read from the **SharePoint List**
- **Origin App** displays the application list for review

graph TD

A[Student fill application form] → B[React from submit]

B → C[Backend API]

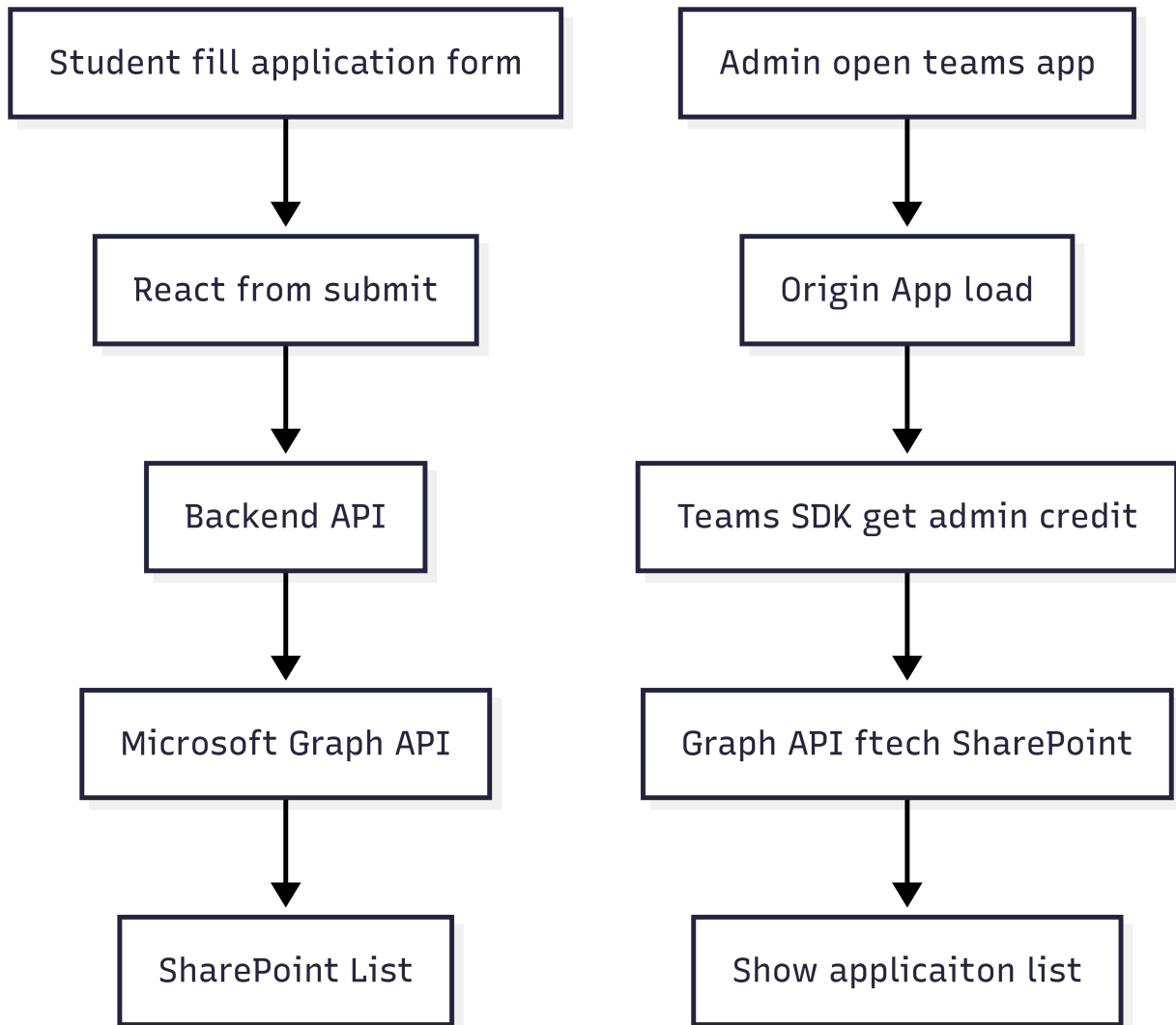
C → D[Microsoft Graph API]

D → E[SharePoint List]

F[Admin open teams app] → G[Origin App load]

G → H[Teams SDK get admin credit]

H → I[Graph API fetch SharePoint]
I → J[Show application list]



✓ 3. Parallel Processing Logic After Origin App Fetches Data

Once the **Origin App** retrieves data from the **SharePoint List** via **Microsoft Graph API**, it immediately triggers two parallel processes:

🔄 ① 4.0: Origin App Initiates POST Request to Local Drive App (Auto Registration)

📌 **Implementation Approach:**

- After fetching the data, the **Origin App** immediately sends a **POST request** to the locally deployed **Local Drive App** using `fetch`.
- The request body contains student information, course applied for, and other relevant fields.
- Upon receiving the data, the **Local Drive App** executes the following process:
 - Writes data to a **CSV file**
 - Automatically logs in to the **OLP system** using **Selenium** and uploads the CSV file
 - Creates an **OLP account** and records the result
 - Sends a notification email containing account information and **LLN test** link
 - Waits for the student to complete the **LLN test**

Implementation example:

```
js
CopyEdit
fetch("https://your-ngrok-url.com/trigger", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Authorization": "Bearer your_token_here"
  },
  body: JSON.stringify({
    first_name: "John",
    last_name: "Doe",
    email: "johndoe@email.com",
    course: "Cert III in Carpentry"
  })
})
```

Notes:

- This process is independent and does not directly rely on subsequent review outcomes.

- Idempotent design should be considered to prevent duplicate registration or import.
-

② 5: Display Application Data and Perform Automated Review

Objective:

- Origin App frontend uses Graph API to load the application list.
- The automated review logic includes:
 - Checking whether all required fields are completed.
 - Verifying whether all required documents are uploaded.
 - Validating file naming format and type.
 - Review results are categorized as: All OK / Missing / Rejected.

Technical Implementation notes:

- It is recommended to encapsulate the automated review logic in a backend function (e.g., Azure Function) to support logging, scalability, and reusability.
 - The review result will be written back to the Status field in the SharePoint List.
-

After Passing the Review, Trigger Information Delivery to the RTO (Process Synchronization Point)

- When an administrator clicks the "Approve" button in the Origin App, an additional request is sent to the Local Drive App to deliver the student information to the RTO for issuing the offer.
 - This step strictly depends on the successful result of the automated review.
-

4.1 Local Drive App (Locally Deployed) Receives Request and Executes Registration Logic

This process starts as soon as the Origin App fetches data and runs in parallel with the review process.

Technology (To Be Developed):

- Python + Flask / FastAPI to receive POST requests.
- Write data to a CSV file (for OLP import).

- Automatically run a registration script (using Selenium to control a browser to upload the CSV and complete the registration).

```
@app.route("/trigger", methods=["POST"])
def trigger_olp():
    data = request.json
    append_to_csv(data)
    subprocess.run(["python", "register_olp.py"])
    return {"status": "OLP created"}
```

✓ 4.2 Use Selenium Script to Log in to OLP and Register

Technology:

- Use Chrome + Selenium (headless mode)
- Simulate CSV upload and complete the registration
- Locate elements (By.ID / XPATH) to upload the file and click the submit button
- Registration steps:
 1. Visit Import Page to upload the CSV
 2. Based on the `course` option in the checklist, add a course on the webpage (C3, C4, or Diploma)
 3. Click the submit button

✓ Logging and error handling mechanisms are recommended.

✓ 4.3 Send Email Notification After Account Creation

Technology:

- Extract account information from the Selenium operation
- Use Python's `smtplib` or `email` module
- Send emails using an HTML template, including username/password, OLP login link, and LLN test instructions

✓ 4.4 Students Take the LLN Test

- Guidance for students can be included in the email

✓ 6. Determine Whether LLN Test and Offer Conditions Are Met

📌 Completion Path:

- Students complete the LLN assessment via the test link in the email
- The LLN test consists of 4 main parts and a total of 8 tasks

<input type="checkbox"/>	Learning Competency Assessment				
<input type="checkbox"/>	Part 1: Online Learning Platform Introduction				
<input type="checkbox"/>	Instructions to Learner - Online Learning Platform	14 Mar 2025	14 Mar 2025	Task Completed	100%
<input type="checkbox"/>	Part 2: Learning Barrier and Support Needs				
<input type="checkbox"/>	Suitability Assessment	14 Mar 2025	18 Mar 2025	Task Completed (Tiffany Lian)	100%
<input type="checkbox"/>	Learning Barriers	16 Mar 2025	18 Mar 2025	Task Completed (Tiffany Lian)	100%
<input type="checkbox"/>	Skill and Experience Summary - CPC30220	16 Mar 2025	18 Mar 2025	Task Completed (Tiffany Lian)	100%
<input type="checkbox"/>	Part 3: Language, Literacy and Numeracy Test				
<input type="checkbox"/>	Origin Institute LLN Introduction	16 Mar 2025	16 Mar 2025	Task Completed	100%
<input type="checkbox"/>	Origin Institute Literacy and Numeracy Test	16 Mar 2025	18 Mar 2025	Task Completed (Tiffany Lian)	100%
<input type="checkbox"/>	Candidate Confirmation	16 Mar 2025	18 Mar 2025	Task Completed (Tiffany Lian)	100%
<input type="checkbox"/>	Part 4: Construction Site Visit Term and Conditions				
<input type="checkbox"/>	Construction Site Work Placement Agreement and Release	16 Mar 2025	18 Mar 2025	Task Completed (Tiffany Lian)	100%

- After the LLN test is completed, the Origin App uses the Ammonite API to retrieve the course list and completion status.

✓ Actual Workflow:

📄 The list of student information you already have (e.g., in SharePoint):

```
{
  "students": [
    {
      "name": "Jane Smith",
      "email": "jane.smith@example.com",
      "course": "Carpentry"
    },
    {
```

```
"name": "Leo Wong",
"email": "leo.wong@example.com",
"course": "Carpentry"
}
]
```

Operation Logic (For Each Student) :

1. Retrieve **email** and **course**
2. Call **/v2/usercourses** :

```
{
  "email": "jane.smith@example.com",
  "units": true}
```

3. Extract **user_course_id** from the response

- Filter by **course_full_name** equal to **"Carpentry"**

```
{
  "data": [
    {
      "id": 54321, // user_course_id
      "course_full_name": "Carpentry",
      "course_short_name": "WCC",
      "status": "Enrolled",
      "date_started": "2025-05-28",
      "user_id": 9876,
      "units": [
        {
          "unit_code": "Carpentry-UNIT",
          "unit_name": "Carpentry Unit",
          "status": 20, // 20 = Competency Achieved
          "date_started": "2025-05-28",
          "date_completed": "2025-06-01"
        }
      ]
    }
  ]
}
```



```

    ]
  }
],
"message": "Found 1 course for user"
}

```

4. Call `/v2/usercoursetasks?user_course_id=...`

GET `/v2/usercoursetasks?user_course_id=xxxxx`

Each task in the response includes:

- `task_id`
- `task_description`
- `is_complete`
- `date_completed`

5. Check whether the three LLN-related tasks are completed

✓ Example Logic (Pseudocode):

```

const LLNTaskIDs = [3504, 3499, 3500, 11864, 3493, 3492, 3497, 11865];

function isLLNCompleted(userCourseTasks) {
  return LLNTaskIDs.every(id => {
    const task = userCourseTasks.find(t => t.task_id === id);
    return task?.is_complete === 1;
  });
}

```

✓ If all 8 tasks are completed, the LLN Assessment is considered complete and meets one of the conditions for issuing an offer.

- If any task is incomplete, the system should mark:

"LLNCompleted": false

- This evaluation result will be a key part of the logic determining whether an offer is sent.
- The Origin App will check this status during the final review process, along with the "All OK" document status, to determine the offer type.

```
{
  "StudentName": "Jane Smith",
  "Email": "jane.smith@example.com",
  "Course": "White Card Course",
  "Documents": {
    "PassportUploaded": true,
    "AcademicTranscriptUploaded": true,
    "VisaUploaded": true,
    "EnglishProofUploaded": true,
    "GTStatementUploaded": false,
    "AllDocumentsOK": false
  },
  "LLNCompleted": true,
  "OLPAccountCreated": true,
  "OfferStatus": "Conditional",
  "ReviewedBy": "admin@origin.edu.au"
}
```

✓ 7. Origin App Sends Request to Local Drive App, Which Pushes Student Information to the RTO System; Admin Manually Issues Offer




After receiving the request from the Origin App, the Local Drive App completes student registration and account creation. It then uses a **Selenium automation script to log in to the OLP system**, navigates to the corresponding student profile page, and **simulates clicking the "Send to RTO" button**, thereby pushing the student information to the RTO management system.

📌 Issuance Timing:

- When the Origin App confirms that documents are complete **and** the LLN test is finished, it displays a "Completed" status on the frontend and sends a POST request to the Local Drive App — this triggers the student information to be pushed to the RTO.

- Admin then issues either a FULL or CONDITIONAL offer based on the list status displayed in the Origin App frontend.

Technology:

-  **Locator Method:** Search for the student in the OLP system using email or name to enter the student profile page.
-  **Automation Actions:** Use Selenium's `find_element` method to accurately click the "Send to RTO" button.
-  **Scalability:** If the OLP system exposes this function via API in the future, the Selenium automation

Process Enhancements

Feature	Recommendation
Security	Include API Key authentication in requests from Origin App to Local Drive App
Logging	Log all operations (success/failure) into SharePoint Log List or local log files
Error Handling	Local Drive App should return status on registration failure to notify Origin App
Duplicate Check	Check if the email already exists before proceeding with registration

Summary of Technical Components

Module	Technology
Form Hosting	Azure
Data Storage	SharePoint List
Data Access/Writing	Microsoft Graph API
Origin App UI	React + MS Teams Toolkit
Automated Registration	Python + Flask + Selenium
Notifications	SMTP Email or Power Automate