Project Specification: Credit for Prior Learning (CPL) Automation System

Project Title

Automated Credit for Prior Learning Evaluation System

Project Sponsor

- Luke Bozzetto, Dean
- Skyline Higher Education Australia (SHEA)

Project Overview

Skyline Higher Education Australia receives student transcripts from a wide range of domestic and international institutions. Academic staff must manually review these transcripts and determine whether students are eligible for Credit for Prior Learning (CPL) toward units in SHEA's Bachelor of Information Technology (BIT) and Master of Information Technology (MIT).

This process is currently manual, time-consuming, and inconsistent.

The goal of this project is to develop a software system that automates and supports CPL assessment by:

- Extracting units from student transcripts (PDF)
- Retrieving unit information from external institutions
- Comparing those units to SHEA's units
- Suggesting appropriate CPL outcomes
- Providing a clear, defensible rationale for decisions

This system will assist academic staff — not replace academic judgement.

Core Objectives

The system should:

1. Accept student transcript PDFs
2. Extract unit codes, names, grades, and institution
3. Retrieve detailed information about those units from the internet
4. Compare them against SHEA units

5. Suggest possible CPL mappings
6. Provide a confidence score and explanation
7. Allow academic staff to review, approve, or reject suggestions

Users

Primary users:

- Dean
- Course Coordinators
- Academic staff assessing CPL
- Admissions staff

Secondary users (future):

- Students (view status)
- Admin staff

Functional Requirements

1. Transcript Upload and Processing

The system must allow users to:

- Upload transcript PDFs
- View uploaded transcripts
- Automatically extract:
    - Institution name
    - Unit code
    - Unit name
    - Grade
    - Year/semester (if available)

Example input:

Transcript PDF containing:

- COSC101 – Introduction to Programming – Credit
- COSC202 – Data Structures – Distinction

Implementation suggestions:

- Python
- pdfplumber or PyMuPDF
- OCR fallback using Tesseract if needed

Output example:

```
[
 {
  "institution": "University of Technology Sydney",
  "unit_code": "COSC101",
  "unit_name": "Introduction to Programming",
  "grade": "Credit"
 }
]
```

If possible, also get information on what AQF level.

2. External

Unit Information Retrieval

The system should retrieve detailed unit information from external sources.

Sources may include:

- University websites
- Unit handbook pages
- Course catalogs
- Public web pages

Information to retrieve:

- Unit description
- Learning outcomes
- Topics covered
- AQF level (if available)
- Credit points

Example:

Input:

COSC101 – Introduction to Programming – UTS

System retrieves:

- Description
- Learning outcomes
- Topics
- Credit points

Implementation suggestions:

- Web scraping (BeautifulSoup, Playwright)
- API access where available
- Caching results locally

3. SHEA Unit Database

The system must store SHEA unit information, including:

- Unit code
- Unit name
- Description
- Learning outcomes
- AQF level
- Course (BIT or MIT)

Example:

```
{
 "unit_code": "ITOP502",
 "unit_name": "Programming Foundations",
 "aqf_level": 7,
 "learning_outcomes": [...]
}
```

This can be stored in:

- SQLite database
- JSON files (initial version)

4. Matching and CPL Suggestion Engine

The system must compare external units with SHEA units and suggest possible CPL.

Comparison criteria:

- Unit name similarity
- Description similarity
- Learning outcome similarity
- AQF level compatibility
- Credit level compatibility

Output example:

External Unit	Suggested SHEA Unit	Confidence
COSC101	ITOP502	92%
COSC202	ITDS503	88%

5. AI-Assisted Similarity (Recommended)

Use embeddings or LLM comparison to evaluate similarity.

Suggested approach:

- Convert unit descriptions to embeddings
- Calculate similarity scores

Technologies:

- OpenAI embeddings API or sentence-transformers
- cosine similarity

6. User Interface

The system should include a simple web interface.

Recommended framework:

- Streamlit (recommended for rapid development)

Core interface features:

Page 1: Upload Transcript

- Upload PDF
- View extracted units

Page 2: CPL Suggestions

- Display suggested matches
- Show confidence score
- Show reasoning

Page 3: Review and Approval

- Approve CPL
- Reject CPL
- Override suggestion

7. Explanation and Transparency

Each suggestion must include reasoning.

Example:

“Suggested match based on 91% similarity in unit description and overlapping learning outcomes including variables, control structures, and functions.”

This is critical for:

- Academic defensibility
- TEQSA compliance
- Audit trail

8. Export Functionality

The system should allow exporting results to:

- PDF report
- Excel
- CSV

Example export:

Student: John Smith

Institution: UTS

External Unit	SHEA Unit	Recommendation
COSC101	ITOP502	Grant CPL

Non-Functional Requirements

The system should be:

- Easy to use
- Reliable
- Fast
- Transparent
- Secure (student data)

Suggested Technology Stack

You may use whatever tech stack you wish, but the following are suggestions for across the pipeline:

Backend:

- Python

Frontend:

- Streamlit

Database:

- SQLite

Libraries:

- pdfplumber
- BeautifulSoup
- Playwright
- sentence-transformers or OpenAI embeddings

Optional:

- Docker

Suggested Architecture

Transcript PDF
↓
PDF Parser
↓
Extracted Units
↓
Web Scraper
↓
Unit Information Database
↓
Similarity Engine
↓
CPL Suggestions
↓
Web Interface
↓
Export / Approval

Stretch Goals (Optional Advanced Features)

These are optional but highly valuable.

AI-powered learning outcome comparison

Use LLM to compare learning outcomes directly.

Institutional unit database

Cache scraped units locally to avoid repeated scraping.

Confidence calibration

Provide confidence bands:

- High confidence (>85%)
- Medium confidence (60–85%)
- Low confidence (<60%)

Historical decisions database

Learn from previous CPL decisions.

Bulk transcript processing

Upload multiple transcripts at once.

Deliverables

Intern team must deliver:

1. Working application
2. Source code
3. Documentation
4. Installation instructions
5. Demo

Success Criteria

The system is successful if:

- Transcript units are correctly extracted
- External unit information is retrieved
- Relevant CPL suggestions are generated
- Staff can review and approve suggestions easily

- The system saves significant time

Real-World Impact

This system will:

- Save academic staff significant time
- Improve consistency in CPL decisions
- Improve student experience
- Support scalable admissions growth