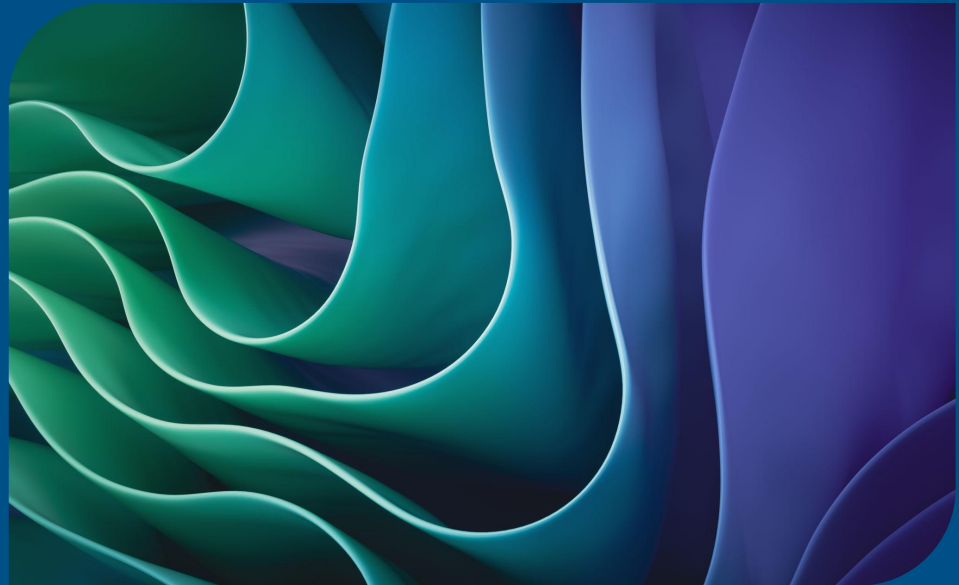


FIT Automated Transfer Credit Evaluator

Group Member(s): Tyler Dionne,
Kendall Kelly, Braden Corkum

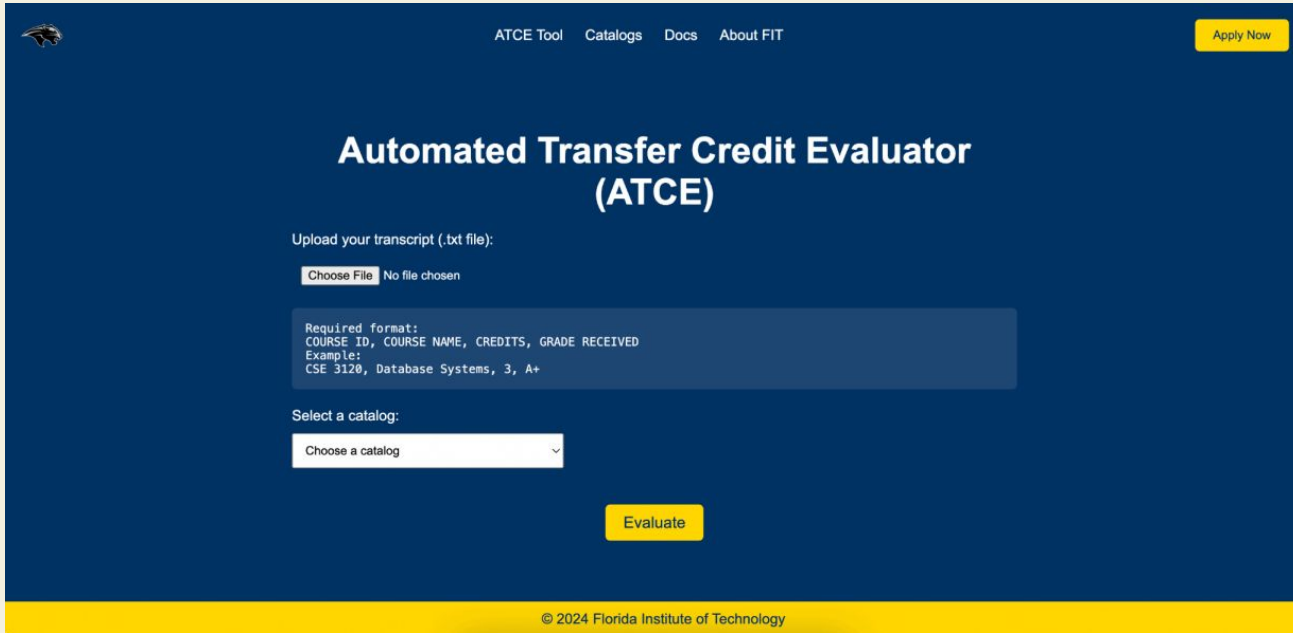
Advisor: Dr. Sneha Sudhakaran



Current Progress

ATCE Tool UI Improvements

Previous Design



The screenshot shows the previous design of the ATCE tool interface. It features a dark blue background with a yellow header bar at the top. The header bar contains a small bird logo on the left, navigation links for 'ATCE Tool', 'Catalogs', 'Docs', and 'About FIT' in the center, and a yellow 'Apply Now' button on the right. The main content area is centered and contains the title 'Automated Transfer Credit Evaluator (ATCE)' in white. Below the title is a section for uploading a transcript, with a 'Choose File' button and the text 'No file chosen'. A light blue box provides instructions on the required format: 'Required format: COURSE ID, COURSE NAME, CREDITS, GRADE RECEIVED' and an example: 'Example: CSE 3120, Database Systems, 3, A+'. Below this is a 'Select a catalog:' label and a dropdown menu with the text 'Choose a catalog'. A yellow 'Evaluate' button is positioned at the bottom center of the main content area. The footer is a yellow bar with the text '© 2024 Florida Institute of Technology'.

ATCE Tool Catalogs Docs About FIT [Apply Now](#)

Automated Transfer Credit Evaluator (ATCE)

Upload your transcript (.txt file):

[Choose File](#) No file chosen

Required format:
COURSE ID, COURSE NAME, CREDITS, GRADE RECEIVED
Example:
CSE 3120, Database Systems, 3, A+

Select a catalog:

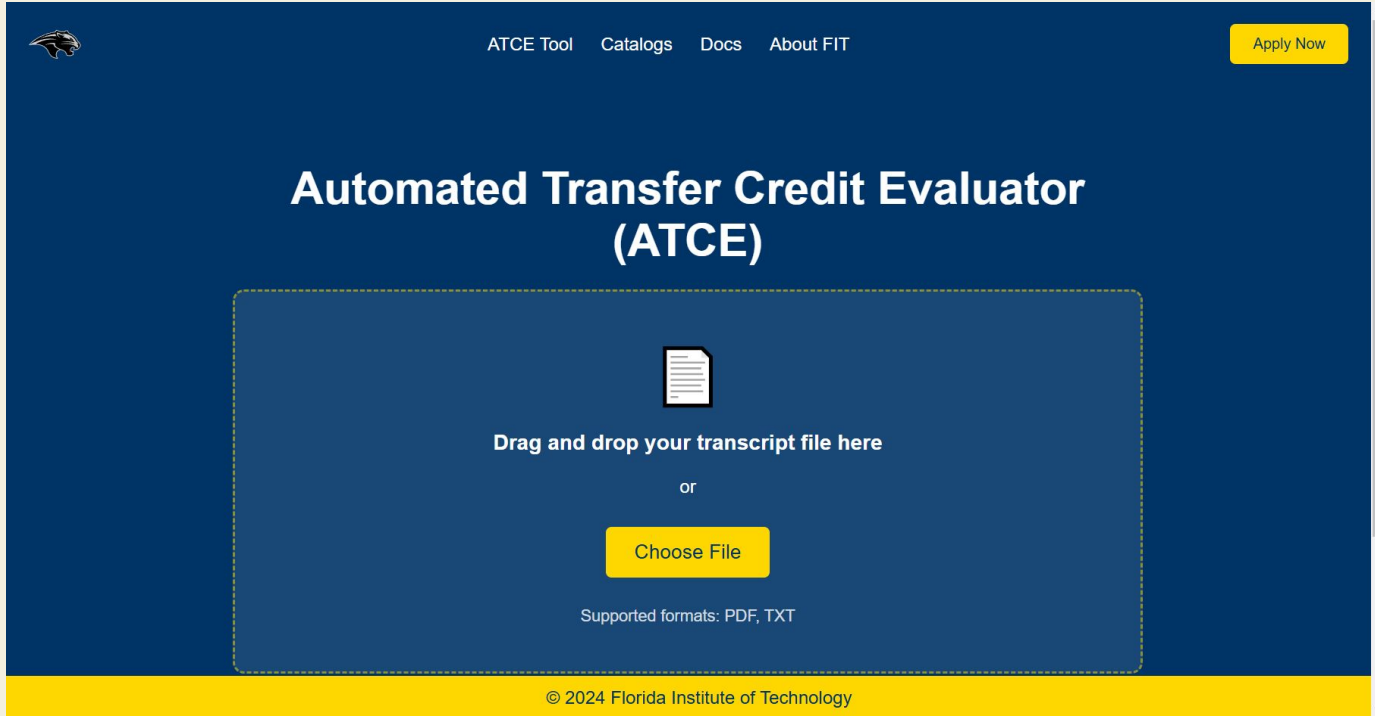
[Choose a catalog](#) ▾


[Evaluate](#)

© 2024 Florida Institute of Technology


ATCE Tool UI Improvements

New Design



 [ATCE Tool](#) [Catalogs](#) [Docs](#) [About FIT](#) [Apply Now](#)

Automated Transfer Credit Evaluator (ATCE)



Drag and drop your transcript file here

or

[Choose File](#)

Supported formats: PDF, TXT

© 2024 Florida Institute of Technology

ATCE Tool UI Improvements

Previous Design:

- Visually un-appealing to the user
- Does not quite look like a tool
- Not convenient and or easy to use
- Outdated UI design

Improvements:

- Lighter colors in the backdrop and icons
- Looks more like a tool that you would find online in 2024
- Drag & Drop functionality provides the option to choose whether to select a file traditionally or drag and drop
- Drag and Drop features are popular among users who may want to upload various files at one time
- Convenient file uploading and extended support for pdf files the tool
- Front increase in user friendliness, practicality and usability

PDF Support

Scripts were integrated directly into the front end html for the atce tool.
Thus far the tool can read in .txt files and .pdf files in a specific format.

The tool utilizes the PDF.js library to handle PDF file processing. This library is initialized at the beginning of the script:

```
pdfjsLib.GlobalWorkerOptions.workerSrc =  
'https://cdnjs.cloudflare.com/ajax/libs/pdf.js/3.11.174/pdf.worker.m  
in.js';
```

PDF Support

When the user uploads the pdf, the processPDF function is called and deals with extracting text content from the file.

The function extracts the text by:

- Converting the file to an ArrayBuffer
- Using the PDF.js library to load the document and then iterate through each page of the pdf and extract the text content
- Updating a progress bar to the user
- Splitting the text from the .pdf file into lines and passing them to validateAndProcessContent

```
async function processPDF(file) {
  try {
    const arrayBuffer = await file.arrayBuffer();
    const pdf = await pdfjsLib.getDocument({ data: arrayBuffer
  }).promise;
    let extractedText = '';
    progress.style.width = '30%';

    for (let i = 1; i <= pdf.numPages; i++) {
      const page = await pdf.getPage(i);
      const textContent = await page.getTextContent();
      const pageText = textContent.items.map(item =>
item.str).join(' ');
      extractedText += pageText + '\n';
      progress.style.width = `${30 + (70 * i /
pdf.numPages)}%`;
    }

    const lines = extractedText.split('\n');
    validateAndProcessContent(lines);
  } catch (error) {
    formatError.textContent = 'Error processing PDF: ' +
error.message;
    formatError.style.display = 'block';
    progressBar.style.display = 'none';
  }
}
```

PDF Support

After the text is extracted and sent to the `validateAndProcessContent` function it makes sure that the content follows the desired format and if so it is validated.

The function checks each line to make sure it contains four comma-separated values (course ID, course name, credits, and grade) and it verifies that the credits are numeric and grade follows the expected format (A+). If the content passes the checks it makes an array of processed course objects and the `displayResults` function is called to present the data to the user/console.

```
function validateAndProcessContent(lines) {
  let isValid = true;
  const processedLines = [];

  for (let line of lines) {
    line = line.trim();
    if (line === '') continue;
    const parts = line.split(',').map(part => part.trim());
    if (parts.length !== 4) {
      isValid = false;
      break;
    }
    if (isNaN(parts[2])) {
      isValid = false;
      break;
    }
    if (!parts[3].match(/^[A-F][+-]?$/)) {
      isValid = false;
      break;
    }
    processedLines.push({
      courseID: parts[0],
      courseName: parts[1],
      credits: parseFloat(parts[2]),
      grade: parts[3]
    });
  }

  // ... (code for handling validation results)
}
```


PDF Support

The function creates a table row for each course, inserts the extracted information, and calculates the total credits. The results are then displayed in a table directly below the atce tool.

```
function displayResults(processedLines) {
  const resultsContainer =
document.getElementById('results-container');
  const resultsBody = document.getElementById('results-body');
  const totalCreditsCell = document.getElementById('total-credits');

  resultsBody.innerHTML = '';

  const totalCredits = processedLines.reduce((sum, course) => sum +
course.credits, 0);

  processedLines.forEach(course => {
    const row = document.createElement('tr');
    row.innerHTML = `
      <td>${course.courseID}</td>
      <td>${course.courseName}</td>
      <td>${course.credits}</td>
      <td>${course.grade}</td>
    `;
    resultsBody.appendChild(row);
  });

  totalCreditsCell.textContent = totalCredits.toFixed(1);
  resultsContainer.style.display = 'block';
}
```

Automated Transfer Credit Evaluator (ATCE)



Drag and drop your transcript file here

or

Choose File

Selected file: test1.pdf

Expected format in document:
COURSE ID, COURSE NAME, CREDITS, GRADE RECEIVED
Example:
CSE 3120, Database Systems, 3, A+

Transcript Analysis Results

Course ID	Course Name	Credits	Grade
CSE 3120	Database Systems	3	A+
Total Credits:		3.0	

Testing

test-1.pdf

CSE 3120, Database Systems, 3, A+

Automated Transfer Credit Evaluator (ATCE)

Drag and drop your transcript file here
or
Choose File
Selected file: test-1.pdf

Expected format in document:
COURSE ID, COURSE NAME, CREDITS, GRADE RECEIVED
Example:
CSE 3120, Database Systems, 3, A+

Transcript Analysis Results

Course ID	Course Name	Credits	Grade
CSE 3120	Database Systems	3	A+
Total Credits:		3.0	

Default Test Case

test-8.pdf

CSE 1002 , Intro To Programming, 2 , G

Automated Transfer Credit Evaluator (ATCE)

Drag and drop your transcript file here
or
Choose File
Selected file: test-8.pdf

Invalid format. Please ensure each line follows: COURSE ID, COURSE NAME, CREDITS, GRADE RECEIVED

Expected format in document:
COURSE ID, COURSE NAME, CREDITS, GRADE RECEIVED
Example:
CSE 3120, Database Systems, 3, A+

We see here that the script picks up on any letter outside of the range A+ to F- and hence when we have G as a grade we have invalid format.

Testing (Continued)

test-4.pdf

CSE 1002 , Intro To Programming, , B+

Automated Transfer Credit Evaluator (ATCE)

Drag and drop your transcript file here
or
Choose File
Selected file: test-4.pdf

Expected format in document:
COURSE ID, COURSE NAME, CREDITS, GRADE RECEIVED
Examples:
CSE 3120, Database Systems, 3, A+

Transcript Analysis Results

Course ID	Course Name	Credits	Grade
CSE 1002	Intro To Programming	NaN	B+
Total Credits:		NaN	

If no credits are specified we see there is no error printed but instead the NaN (NotaNumber) value is placed in the credits column. This test case help us to find a bug and understand what additional checks need to be done.

Testing (Continued)

By developing tests for the new implementation we can use edge cases and fuzzing techniques to find any unexpected functionality of our program. These test cases help us to understand bugs and additional checks that need to be done in order to validate the format correctly in each scenario.

Developing test cases helps the team to find bugs in our html script and understand additional checks that need to be done to validate format.

Future Plans (Milestone 4)

- FERPA Compliance and Transcript Generation
 - Find legal solutions for obtaining transcripts
 - Research methods to generate realistic synthetic transcripts
- Data Complexity and Extension
 - Recommend and incrementally modify data complexity
 - Expand tool's data handling capabilities
- Database Implementation
 - Store FIT catalogs in database
 - Ensure data querying in standard format
- Backend Development
 - Implement SQLite DB for catalogs and logins
 - Use Flask for backend functionality
 - Develop web app with Docker
 - Conduct backend testing
- PDF Parsing Improvements
 - Enable multi-line PDF parsing
 - Develop more realistic transcript parsing schema

Questions?