Project Title:
ATCE (Automated Transfer Credit Evaluation)

Team Members:
Tyler Dionne (tdionne2021@my.fit.edu), Kendall Kelly (kelly2021@my.fit.edu)

Project Advisor:
Sneha Sudhakaran, ssudhakaran@fit.edu

Client:
Sneha Sudhakaran, ssudhakaran@fit.edu

Dates of Meetings with the Client for Developing this Plan:
Once a week every other week

Goal and Motivation:

- The overall goal is to create a streamlined and user-friendly tool to help university students determine whether the credits they earned at a previous institution will be accepted at the Florida Institute of Technology. This tool will simplify the transfer credit evaluation process and allow students to obtain accurate transfer credit evaluation reports in a timely manner which will overall result in a smoother, efficient transfer between institutions.

- University students often face significant pain and challenges when transferring between institutions because of the lack of clarity and transparency when it comes to which credits will be accepted. The current systems in place are inefficient and force students to wait for manual evaluations. This tool will address the pain faced by these students by allowing students to upload their transcript and receive a report right away, eliminating the uncertainty and easing the decision making process when moving from one institution to another.

Approach (key features of the system):

- Two Tier User System: In our web-application there exists two user types; user and admin. The user can access the ATCE (Automated Transfer Credit Evaluator) tool where they can upload their transcript and select a catalog to receive a detailed report on the transfer credit evaluation. The user shall also be able to create a user-level account on our platform. The admin shall be able to access the source of the website, site traffic information and backend processing information. The key focus will be to ensure the user-level user may not access any restricted resources on our server along with any information that may allow for privilege escalation. The admin-level user will act as an example of an account with escalated privileges. This will involve enforcing IAM (identity and access management) best practices to ensure that the user and the admin remain separate with access to different resources.

- Two User Inputs: There exists two user inputs to our software. The first user input is a file containing the student's transcript. The ATCE tool currently has support for both .txt and .pdf files in a particular format. The transcript file shall contain information regarding courses taken previously, the credit amount for each course and in the future shall contain a brief course description. The second user input will be a selection from a dropdown menu of Florida Institute of Technology catalogs stored inside of our web application. This will allow the students to choose which degree program/catalog should be analyzed with respect to their transcript and provide a detailed report of which transfer credits will be accepted within the selected catalog.

- Detailed Evaluation Results: There shall be one output from our tool which will display a detailed analysis of which credits will be accepted. This is currently done via a table displayed underneath the analyze button on the ATCE page of our web application. In the future this table shall display additional information about which credits will be accepted, the course number, the number of credits accepted along with the equivalent course at the Florida Institute of Technology. This information shall be sufficient enough to communicate the results of the transfer credit evaluation to the user and clearly state which credits will be accepted.

- Full Stack Web Application: Using the Flask python framework such as Flask this web application shall have a full implementation of front-end and back-end functionality and act as a complete stand alone web application with database integration in the backend to store user information and Florida Institute of Technology catalogs.

Algorithms and tools for the key features:

- The languages html, css, javascript and Python will be used all together to create the full stack web application.

- For the front end, github pages was used to host the html and css pages for the website (Homepage, ATCE tool, About, Documentation).

- For full stack web application integration the Flask Python framework will be used to convert the website into a full stack web application with backend support.

- PDF.js is used for PDF file parsing support. A javascript script is embedded within the html for the web page to support the parsing of data from a pdf file using this library.

- Sqlite3 will be used inside of the Flask web application to include database support and allow for the storage of catalogs and possibly user login information as well in a database.

Novel features:

- The evaluation is automated rather than manual. Typically, the systems already in place require staff to manually type in a list of the transfer credits to compare to the FIT catalog. Our application will only require an upload of a file with the transfer credits rather than having to manually input the transfer credits.
- Students will be able to independently perform transfer credit evaluations simplifying the process of transferring from one institution to another and clarifying which credits will be accepted at the Florida Institute of Technology.

- We are ensuring the security and integrity of our application by creating a two tier system which will only allow admin accounts to have access to sensitive information about/within the system.

Technical Challenges:

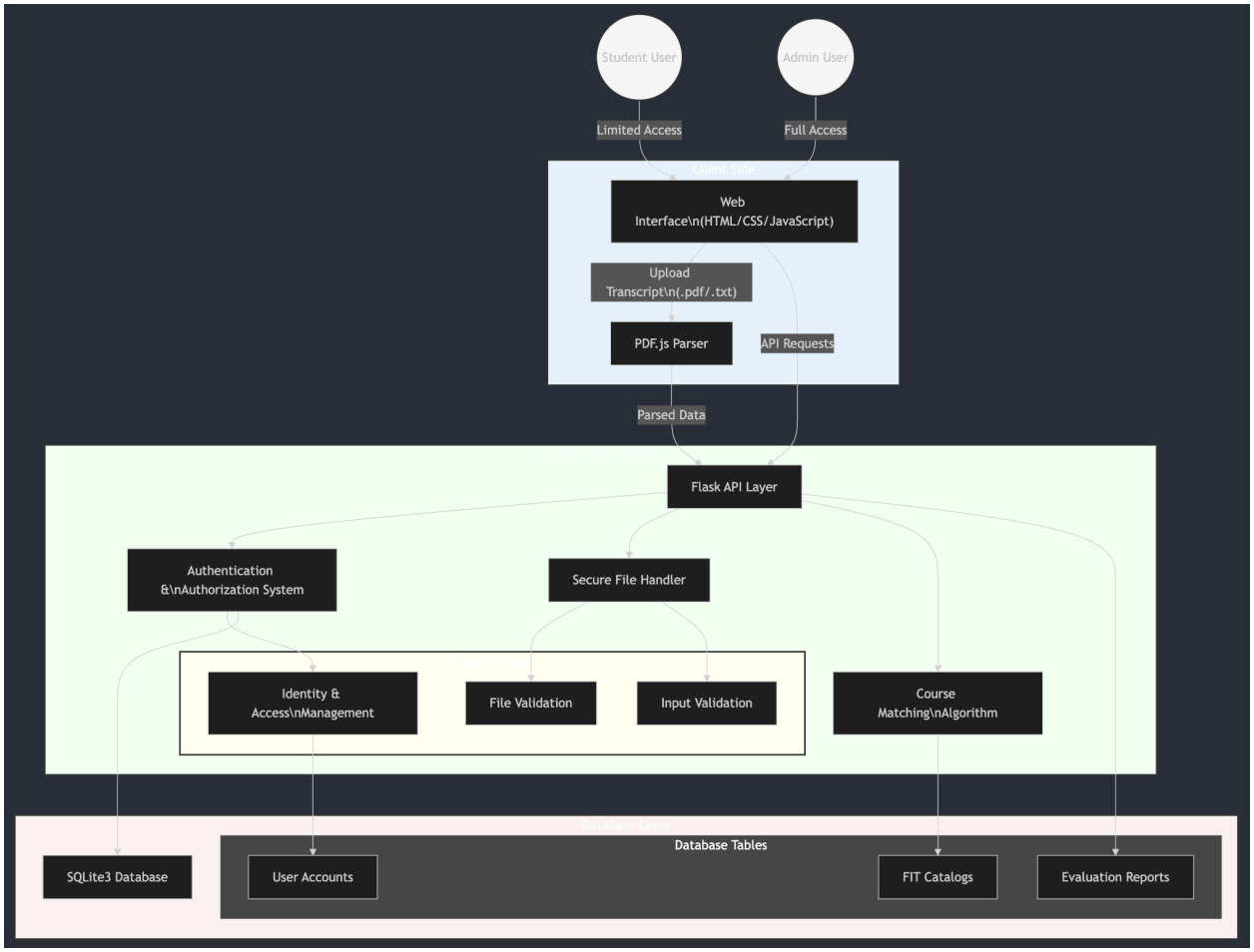
- We will need to create an algorithm to accurately match courses from other universities to courses at FIT. This algorithm will need to be able to properly evaluate course equivalencies. This can be done via course descriptions at the Florida Institute of Technology which can be used with the students' transcripts which shall also contain a course description for each course to match equivalent courses and determine which credits shall be accepted based on similarity in course description.

- We will need to implement the two tier system in order to ensure security in our web application. The users should not be allowed any unauthorized access to the source code of the website, or data analytics on the usage of the tool. The user shall only be able to create a user level account, upload a transcript, select a catalog, and perform an evaluation. Any other data and or resources on the server and website shall only be accessed by an admin user.

- We shall ensure that our full stack web application can handle basic web application vulnerabilities. This will be done via the built in protections offered by the Flask framework as well as some penetration testing of the website once complete.

- We shall implement secure file handling (upload, processing, and storage) in order to prevent any possible security vulnerabilities. For example, an attacker could upload an executable file that could result in some executable or malicious script that could be used to manipulate our tool or access sensitive information. Therefore the user input where they upload a transcript shall be secured to only allow for certain file extensions and this should be validated further than the file extension. For example it should not just check if the file ends with a .pdf or .txt because an attacker could disguise an executable file as a .pdf.

Design (system architecture diagram):



Evaluation

- Speed - Currently the system processes the transcript data within 5 seconds.
- Accuracy - The system we currently have set up can properly parse lines in the proper format from txt and pdf files. However, it can only parse the first line of a page so if there is more than one line per page it skips every line after the first one.
- Reliability - The system uptime is currently 99%. It handles all of the types of files that it states it can handle. As previously stated, we still need to fix some issues with pdf parsing, but aside from this it parses files properly in the correct format. The drag and drop feature works properly as well.
- The system is very user friendly. The instructions are clear and the functionality is very simple.

Progress Summary

Module/feature	Completion	Todo
Website front end html & css webpages hosted with	100%	N/A

github pages		
Specified Format Txt file parsing/processing	100%	N/A
Specified Format Pdf file parsing/processing	100%	N/A
Full Stack Standalone Web Application Conversion Using Flask	0%	Use Flask Python web application framework to convert the current version of our website (only frontend hosted with github pages using html and css) to a full stack web application with backend functionality using sqlite3.
User and Admin Account Creation + Logins	0%	Use the backend functionality created with Flask to allow the creation of User and Admin accounts and storage of user information in the backend using sqlite3.
Built in Flask Common Web Application Security Mechanisms	0%	Use Flask's built- in security protection mechanisms in our web application once developed. This can be done via lines of Python code inside of the app.py file once the frontend has been integrated.
Secure File Upload/Processing	0%	Be sure to validate the extensions of files and ensure that they are genuine. For example we currently accept .pdf and .txt files and we want to make sure that an attacker cannot change the extension of an executable file (.exe, .bin) to a .pdf and cause damage to our

		server.
Basic Web Application Pentesting	0%	Attempt basic pentesting on our web application once it has been finished to ensure that our application is secure and safe from common web application vulnerabilities.

Milestone 4 (Feb 24) Itemized Tasks:

- Use Flask Python web application framework to convert the current version of our website (only frontend hosted with github pages using html and css) to a full stack web application with backend functionality using sqlite3.

Milestone 5 (Mar 26) Itemized Tasks:

- Use the backend functionality created with Flask to allow the creation of User and Admin accounts and storage of user information in the backend using sqlite3.

Milestone 6 (Apr 21) Itemized Tasks:

- Use Flask's built- in security protection mechanisms in our web application once developed. This can be done via lines of Python code inside of the app.py file once the frontend has been integrated.
- Be sure to validate the extensions of files and ensure that they are genuine. For example we currently accept .pdf and .txt files and we want to make sure that an attacker cannot change the extension of an executable file (.exe, .bin) to a .pdf and cause damage to our server.
- Attempt basic pentesting on our web application once it has been finished to ensure that our application is secure and safe from common web application vulnerabilities.

Task Matrix For Milestone 4

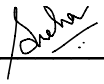
Task	Tyler	Kendall
Initial set up of flask environment	0%	100%
Frontend migration	100%	0%
Implement, test, and demo the database	50%	50%
Implement, test, and demo the file upload system	0%	100%

Implement, test, and demo API development	100%	0%

Description of Each Planned Task for Milestone 4

1. Initial Flask Setup - Set up the virtual python environment and install flask and all other required components. Then, create the application structure and configure the development server.
2. Frontend Migration - We will need to transfer our HTML templates into the Flask template system. Then we'll need to update how we handle CSS and Javascript files and make sure all of the file paths still work correctly. We will do this by converting our pages to work with Flask and test that everything is still displayed properly.
3. Integrating the Database - Design the SQLite database schema and create the initial migration scripts as well as set up the database connection. We will then need to create database models for the FIT course catalogs, transfer credit evaluations, and session management. Implement operations such as create, read, update, and delete.
4. File Upload System - We need to create a system that can handle file uploads securely, focusing mainly on txt and pdf files. To do this we will need to build the upload functionality, store files on the server safely, and ensure we can properly process the files. We then need to validate the files to ensure they're not malicious.
5. API Development - We need to build the core API endpoints for uploading transcripts, selecting catalogs, and getting the results for the evaluations. These endpoints need to be reliable and need to be able to handle errors correctly while providing the appropriate feedback to users when they're triggered.

Approval From Faculty Advisor:

- “I have discussed with the team and approved this project plan. I will evaluate the progress and assign a grade for each of the three milestones”
- Signature:  Date: 01/23/2025