Software Requirements Specification

for

TTrack – Degree Tracker

SDM404 - Assessment 2 – Software Requirements Specification	
Project Name	TTrack – Degree Tracker
Group #	#1
Group Members Names	Hussain Jameel – **A00180177** Luis Guilherme de Barros Andrade Faria - **A00187785** Nomayer Hossain - **A00176827** Rosa Carolina Cortes Galvis – **A00193201** Victor Javier Dorantes Meneses – **A00179705**
Version	1.2.0
Date Created	18, July, 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) documents the functional and non-functional requirements for the **TTrack** system. TTrack is designed as a desktop application to help Torrens University students and academic staff track academic progress by matching student transcripts to prescribed curriculums. This document will serve as a reference for development, testing, and future maintenance.

1.2 Product Scope

TTrack automates academic progress tracking, eliminating the time-consuming manual process of comparing student transcripts against official degree requirements. It ensures accuracy, provides elective recommendations, and offers visual dashboards for data-driven decisions.

The system addresses key pain points in academic administration:

- Manual transcript reviews that typically take 30-45 minutes per student
- Error-prone subject matching due to code variations across systems
- Inconsistent progress tracking between academic advisors
- Limited visibility into elective options that satisfy degree requirements
- Lack of visual reporting tools for academic progress assessment

What TTrack offers:

- Upload and validation of transcript and curriculum data in Excel format
- Intelligent automated matching of completed subjects to curriculum requirements
- Visual dashboards for comprehensive academic progress monitoring
- Smart elective recommendations based on curriculum gaps
- Offline desktop operation with no internet dependency
- Exportable reports for record-keeping and academic planning
- Dark/light theme support for extended usability

2. Overall Description

2.1 Product Perspective

TTrack is a sophisticated standalone desktop application designed with a modular, event-driven architecture. Unlike web applications requiring online services, TTrack operates entirely offline, respecting data privacy and allowing institutions to avoid complex integrations with live academic systems.

TTrack fits into the broader academic ecosystem as follows:

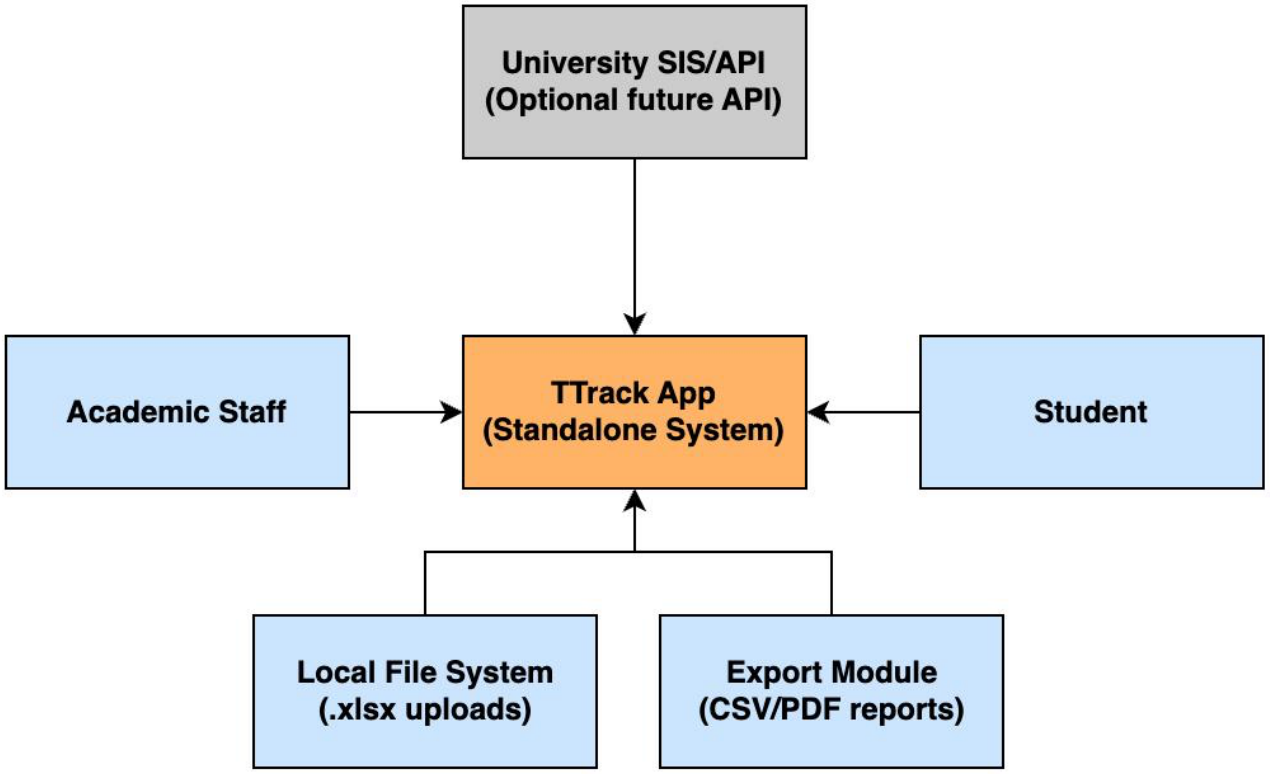


Figure 2.1: System Context Diagram showing TTrack's position in the university academic ecosystem

The application follows a layered architecture pattern with clear separation of concerns:

1. **Presentation Layer**: PyQt5-based user interface components
2. **Business Logic Layer**: Matching engine and data processing modules
3. **Data Layer**: Excel file parsing and data transformation capabilities

While the current version (v1.7.0) operates as a standalone application, future versions may integrate with university information systems as shown in the context diagram.

2.2 Product Functions

TTrack delivers the following core functions:

Data Management
- Upload and validate transcript files in Excel format (.xlsx)
- Upload and validate curriculum files in Excel format (.xlsx)
- Save and load analysis sessions for continued work
- Export results in multiple formats (PDF, CSV, Excel)

Academic Analysis

- Execute matching engine to compare transcript against curriculum requirements
- Identify subject status (Done, Missing, Invalid)
- Calculate completion percentages for degree progress
- Determine remaining credit point requirements

Visualization & Reporting
- Display results in sortable, filterable interactive tables
- Provide interactive dashboard charts for visual progress insights
- Generate comprehensive completion reports
- Offer visual progress tracking across semesters

Recommendations
- Suggest electives based on remaining requirements
- Identify prerequisite chains and optimal subject sequencing
- Flag potential issues with current course selection

The relationship between these functions is illustrated in the functional hierarchy diagram below:

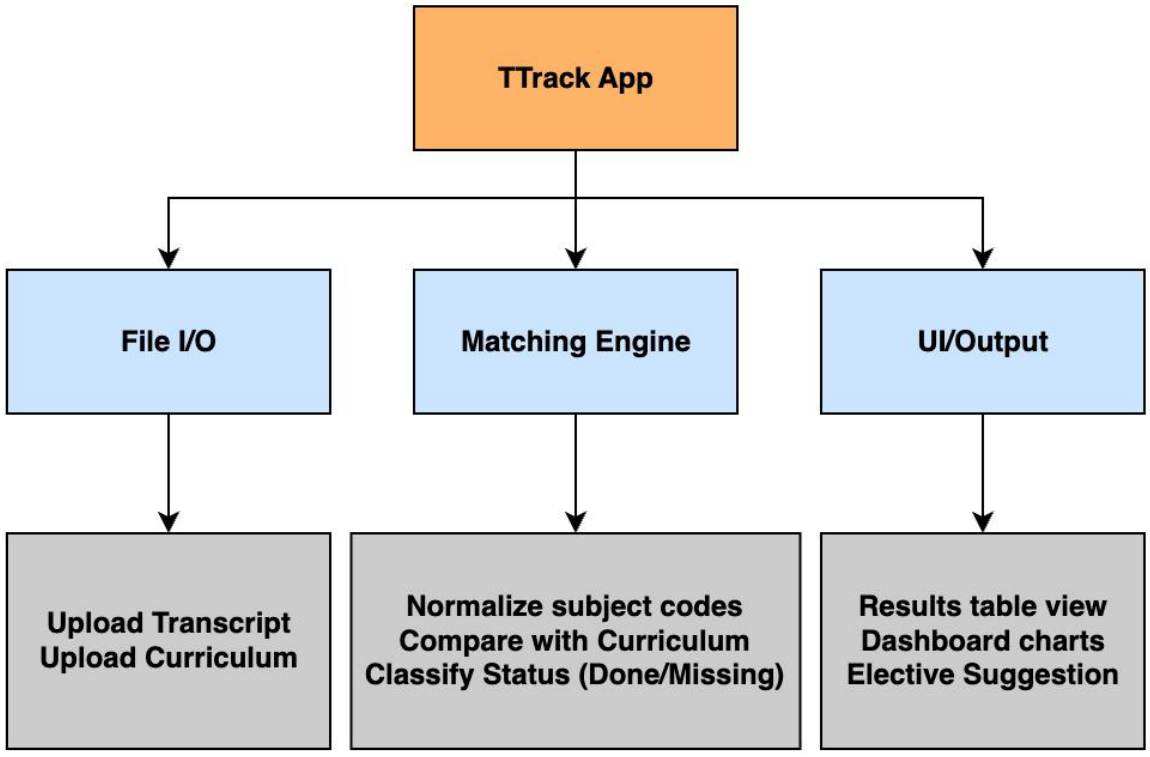


Figure 2.2: Functional Hierarchy Diagram showing the organization of TTrack features

2.3 Stakeholders

Stakeholder	Role	Interest
Student	End user	Wants clear visibility into academic progress
Academic Advisor	End user	Needs to analyze and advise students accurately
Program Director	Manager	Oversees curriculum compliance and student outcomes
University IT	Technical	Ensures software compatibility and support

2.4 Operating Environment

- Supported OS: Windows, macOS, Linux
- Python 3.8+ environment
- Excel files (.xlsx) required for inputs
- 8 GB RAM recommended for large datasets

2.5 Design and Implementation Constraints

- The application must run offline.
- UI must support dark/light themes.
- Processing must handle large Excel files efficiently.
- Must conform to Agile development processes.
- Developed using PyQt5 or ElectronJS frontend.

2.6 Assumptions

- Users have valid transcript and curriculum files in Excel format.
- Subject codes are mostly consistent between transcript and curriculum.
- End-users may not be technically advanced, requiring simple interfaces.

2.7 Dependencies

- pandas for data processing
- openpyxl for Excel I/O
- matplotlib for charts
- PyInstaller for desktop builds

2.8 Business Context and Benefits

TTrack directly supports the objectives of Torrens University by:

- Reducing time spent manually checking degree progress.
- Minimizing errors in student course planning.
- Increasing student satisfaction by offering transparent progress tracking.
- Providing academic staff with reliable data for advising and compliance.

The project aligns with institutional goals of improving operational efficiency and delivering modern, technology-driven solutions to students and staff.

3. External Interface Requirements

3.1 User Interfaces

- Home Screen → System logo, navigation buttons.
- Input Tab → Upload transcript and curriculum files, preview tables.
- Results Tab → Display matched subjects with status.
- Dashboard Tab → Charts for progress analysis.
- Settings Tab → Dark/light mode toggle, preferences.

3.2 Software Interfaces

This section outlines the key software components, tools, libraries, and APIs that TTrack depends on, along with their communication patterns, data exchange mechanisms, and integration characteristics.

- Libraries and Tools:
    - pandas 1.3+
    - openpyxl 3.0+
    - matplotlib
    - OS-level file dialogs for file selection.

3.3 Communications Interfaces

- None required. TTrack is an offline application in version 1.0.

3.4 GUI / Screen Prototypes

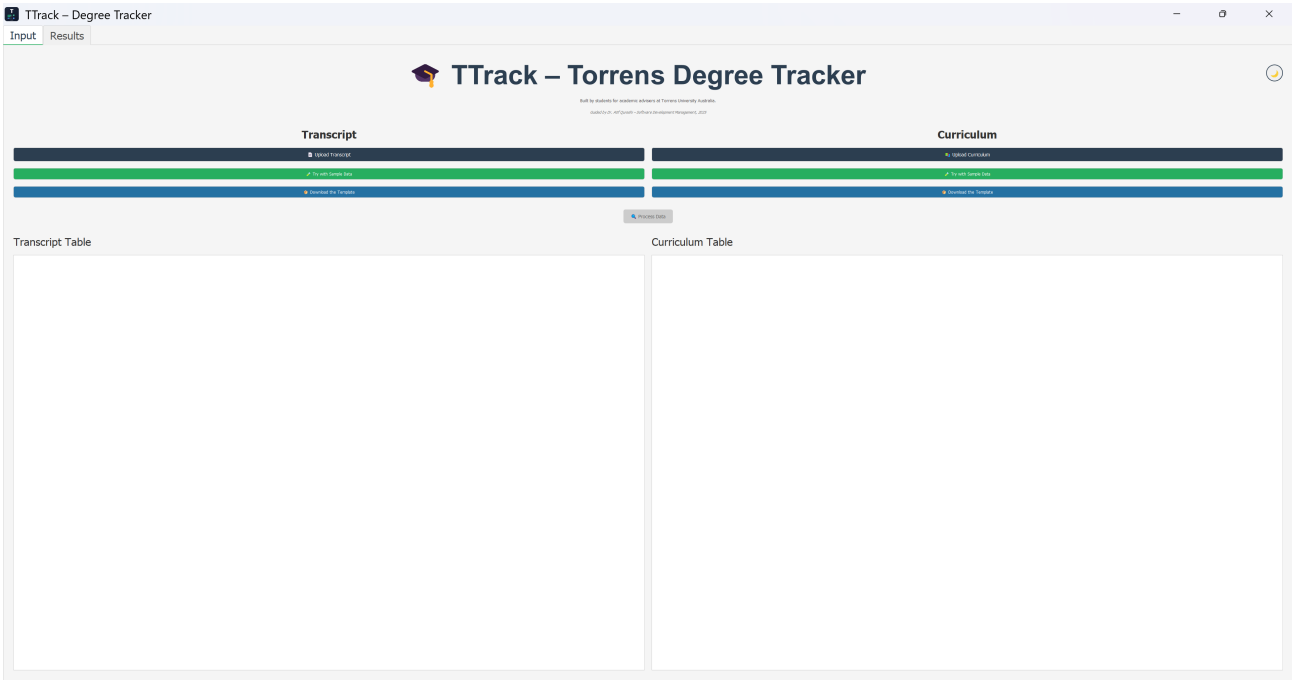
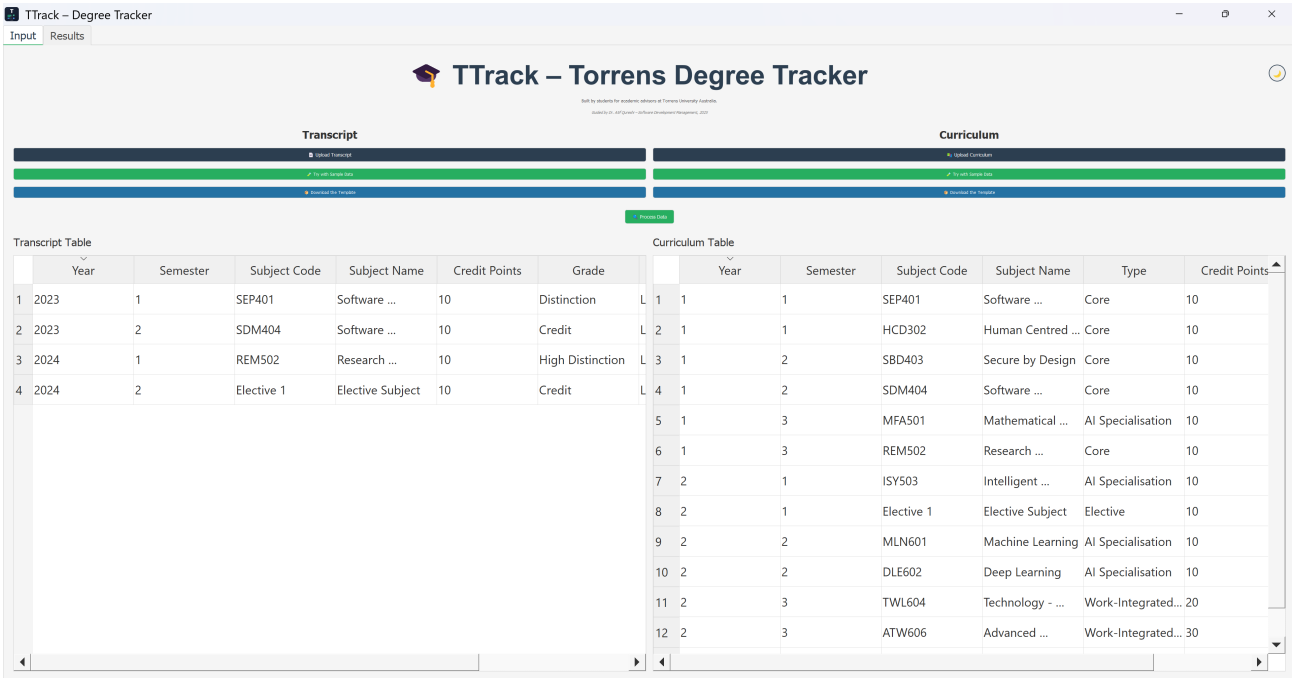


Figure 3.4.1: Landing/Home Page



	Year	Semester	Subject Code	Subject Name	Credit Points	Grade
1	2023	1	SEP401	Software ...	10	Distinction
2	2023	2	SDM404	Software ...	10	Credit
3	2024	1	REM502	Research ...	10	High Distinction
4	2024	2	Elective 1	Elective Subject	10	Credit

	Year	Semester	Subject Code	Subject Name	Type	Credit Points
1	1	1	SEP401	Software ...	Core	10
2	1	1	HCD302	Human Centred ...	Core	10
3	1	2	SBD403	Secure by Design	Core	10
4	1	2	SDM404	Software ...	Core	10
5	1	3	MFA501	Mathematical ...	AI Specialisation	10
6	1	3	REM502	Research ...	Core	10
7	2	1	ISY503	Intelligent ...	AI Specialisation	10
8	2	1	Elective 1	Elective Subject	Elective	10
9	2	2	MLN601	Machine Learning	AI Specialisation	10
10	2	2	DLE602	Deep Learning	AI Specialisation	10
11	2	3	TWL604	Technology - ...	Work-Integrated...	20
12	2	3	ATW606	Advanced ...	Work-Integrated...	30

Figure 3.4.2: Screen after uploading Inputs

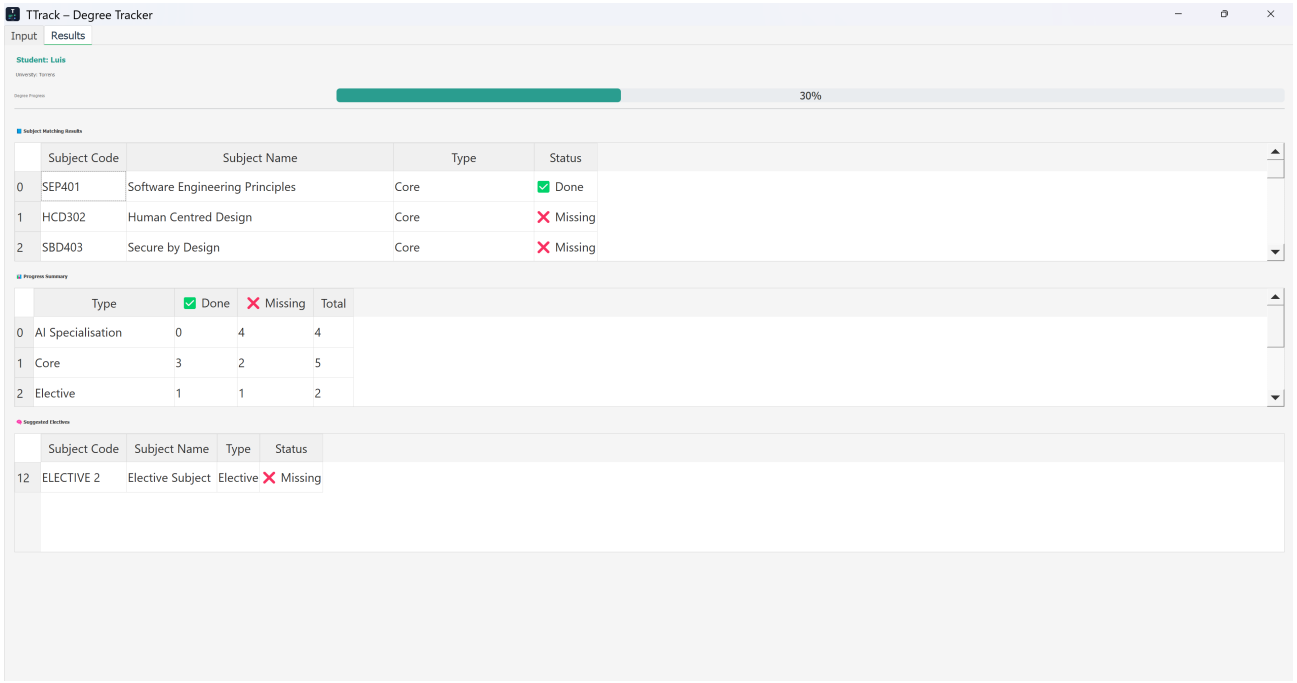


Figure 3.4.3: Results Page

4. System Features

4.1 Upload Transcript File

4.1.1 Description and Priority

This function will allow users to upload a transcript file in .xlsx format for analysis and interpretation. Priority is set High because it is one of the main entry points for processing and generating results.

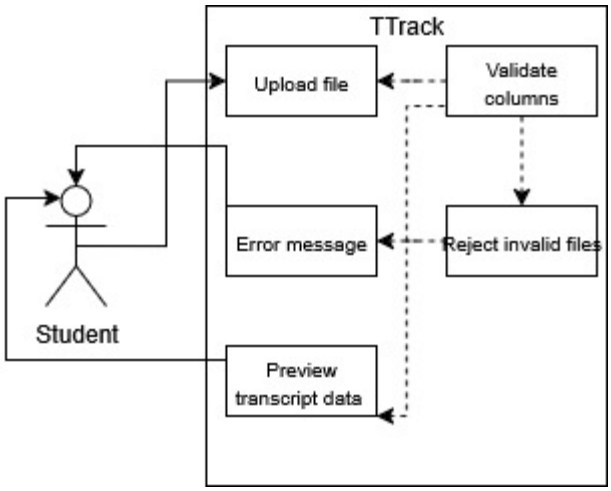
4.1.2 Stimulus/Response Sequences

- User clicks “Upload Transcript.”
- File dialog appears.
- User selects file.
- System validates structure and loads data.
- System shows preview of input data.

4.1.3 Functional Requirements

- REQ-01: The system shall allow users to upload a transcript file in .xlsx format.

- REQ-02: The system shall validate required columns such as Subject Code, Subject Name, Grade, and Completion Status.
- REQ-03: The system shall reject files with invalid or incomplete structure and display appropriate error messages.
- REQ-04: The system shall preview the uploaded transcript data in a structured table format.



4.1.4 Acceptance Criteria:

- Users can upload valid transcript files successfully.
- The system provides clear feedback for incorrect file formats or missing data.
- Uploaded data is displayed accurately in the preview table.

4.2 Upload Curriculum File

4.2.1 Description and Priority

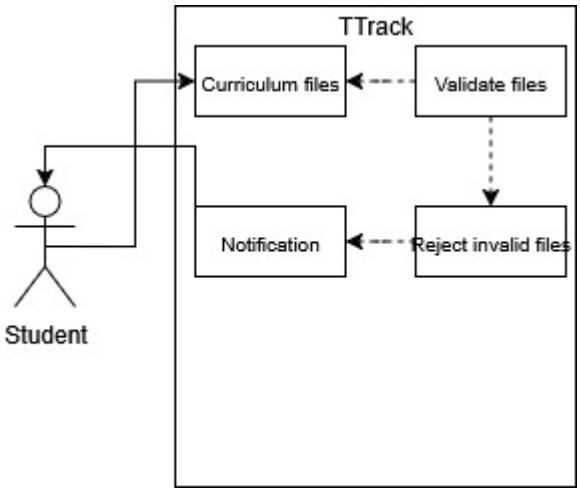
This function will allow users to upload a curriculum file in .xlsx format for analysis and interpretation. The priority is also set as High because it is the main entry point for processing, crossing against transcripts and generating insights.

4.2.2 Stimulus/Response Sequences

- User clicks “Upload Curriculum.”
- File dialog appears.
- User selects a file.
- System validates structure and loads data.

4.2.3 Functional Requirements

- REQ-05: The system shall accept curriculum files in .xlsx format.
- REQ-06: The system shall validate that curriculum files contain subject codes, names, categories (core/specialisation/elective), and credit points.
- REQ-07: The system shall notify users of missing or malformed fields in uploaded curriculum files.



4.2.4 Acceptance Criteria:

- The curriculum file is parsed without errors.
- Data is stored in memory for further matching.
- Invalid files are not processed, and appropriate messages are shown.

4.3 Run Matching Engine

4.3.1 Description and Priority

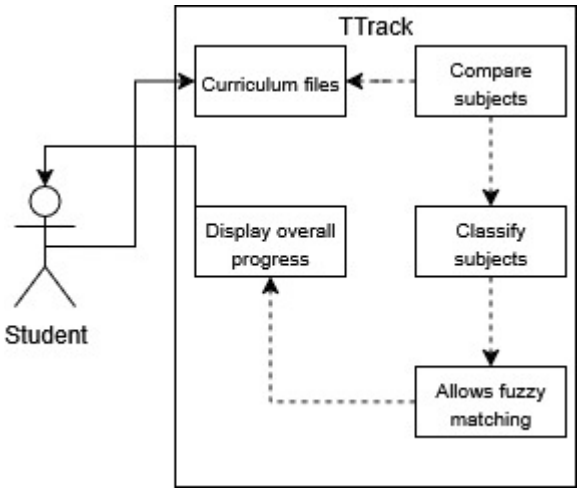
This feature is the core functionality that compares the uploaded transcript with the curriculum to determine academic progress. Priority: High

4.3.2 Stimulus/Response Sequences

- User clicks “Process Data.”
- System matches subjects.
- Output table displays results.

4.3.3 Functional Requirements

- REQ-08: The system shall compare subjects in transcript with curriculum requirements using subject codes.
- REQ-09: The system shall classify subjects into categories such as “Done”, “Missing”, or “Invalid”.
- REQ-10: The system shall support fuzzy matching to handle minor inconsistencies in subject codes.
- REQ-11: The system shall calculate and display overall progress in percentage and credit points.



4.3.4 Acceptance Criteria:

- Matching results are generated within 5 seconds for files up to 10,000 records.
- Completed, missing, and invalid subjects are distinctly marked.
- Completion status is accurate and easy to understand.

4.4 Display Results Table

4.4.1 Description and Priority

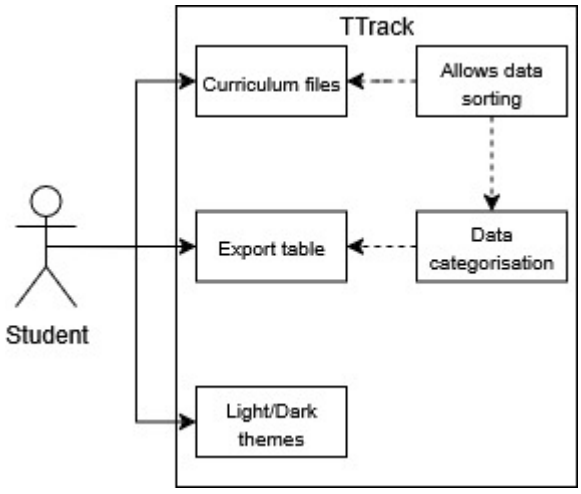
After the matching engine runs, the system displays the results in an interactive table format.
Priority: High

4.4.2 Stimulus/Response Sequences

- User clicks “View Results.”
- System generates and displays a table of matched data.
- Users can sort, reorder, and filter table columns.
- Users may switch between light/dark themes.
- Users may choose to export the visible data.

4.4.3 Functional Requirements

- REQ-12: The system shall display results in a sortable, reorderable, and filterable table.
- REQ-13: The table shall include columns for Subject Code, Subject Name, Category, Status, and Credit Points.
- REQ-14: The system shall allow switching between dark and light themes.
- REQ-15: The system shall allow users to export the table in PDF, CSV, or Excel format.



4.4.4 Acceptance Criteria:

- Results are displayed clearly in a structured table.
- Users can sort, reorder, and filter data for better insights.
- Export functionality works as expected.

4.5 Generate Elective Recommendations

4.5.1 Description and Priority

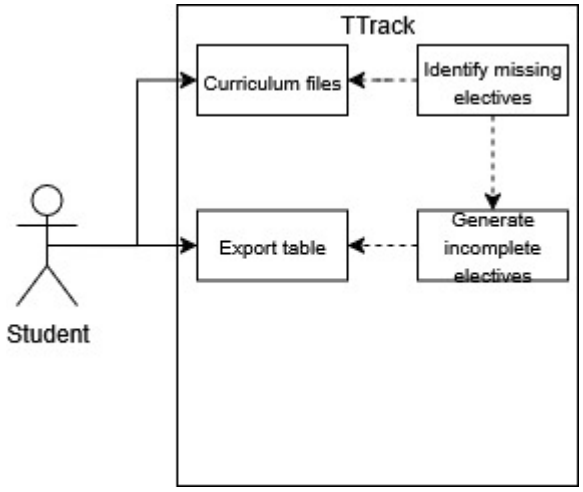
Based on the unmatched elective requirements, the system suggests potential subjects to complete the remaining degree criteria. Priority: Medium

4.5.2 Stimulus/Response Sequences

- User clicks “Generate Recommendations.”
- System scans curriculum for elective subjects not yet completed.
- System displays a list of suggested electives.
- User can view details such as subject code, name, and credit points.
- User may export this list.

4.5.3 Functional Requirements

- REQ-16: The system shall identify missing electives based on curriculum data.
- REQ-17: The system shall generate and display a list of recommended electives not yet completed.
- REQ-18: The system shall allow exporting recommended electives.



4.5.4 Acceptance Criteria:

- Elective suggestions are based on actual curriculum gaps.
- Recommendations are shown in a separate table with relevant details.
- Exported lists are consistent with on-screen data.

4.6 Dashboard Analytics

4.6.1 Description and Priority

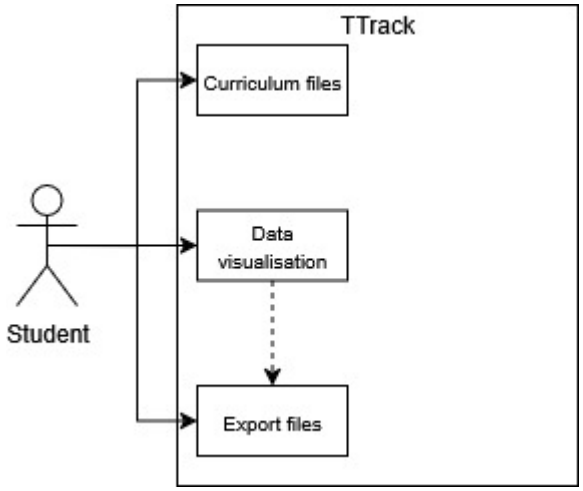
Provides a visual overview of academic progress through charts and graphs. Priority: Medium

4.6.2 Stimulus/Response Sequences

- User clicks on the “Dashboard” tab.
- System processes aggregated progress data.
- System displays pie chart showing completed vs. missing subjects.
- System displays bar chart visualizing elective completion status.
- User can toggle views or export visualizations.

4.6.3 Functional Requirements

- REQ-19: The system shall display a pie chart showing the percentage of subjects completed vs. pending.
- REQ-20: The system shall display a bar chart visualising completion of elective subjects.
- REQ-21: The dashboard shall update dynamically based on the data uploaded.
- REQ-22: The system shall allow exporting charts as image files.



4.6.4 Acceptance Criteria:

- Dashboard visuals update instantly after data is uploaded.
- The charts are interactive and easy to interpret.
- Exported images match what is shown on screen.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system shall load transcript or curriculum files containing up to 10,000 rows in under 5 seconds.
- The user interface shall respond to user actions within 1 second for typical operations.

5.2 Usability Requirements

- The system shall require no more than 10 minutes of training for basic usage.
- The interface shall follow standard design principles for desktop applications.

5.3 Security Requirements

- TTrack shall process all data locally, ensuring no data leaves the user's machine.
- No internet connection shall be required for core functionality.

5.4 Software Quality Attributes

- The codebase shall follow modular design principles (Stephens, 2015).
- TTrack shall be compatible with Windows and macOS.

5.5 Business Rules

- Only subjects listed in the official curriculum file shall count towards completion.
- Recommendations for electives shall be based strictly on the program's prescribed elective pool.
- Data privacy shall be ensured by keeping all student data local to the machine and never transmitting it over the internet.

6. References

- Cobb, C. G. (2015). The Project Manager's Guide to Mastering Agile. Wiley.
- Stephens, R. (2015). Beginning Software Engineering. Wrox.
- Ewusi-Mensah, K. (2003). Software Development Failures. MIT Press.

- Schwaber, K. & Sutherland, J. (2020). The Scrum Guide.
- Torrens University. (2025). SDM404 Assessment 2 Brief.

Appendix A: Glossary

Term	Definition
TTrack	The academic progress intelligence system.
Curriculum	Official list of subjects required for a degree.
Transcript	Official record of student's completed subjects.
pandas	Python library for data manipulation.
openpyxl	Python library for Excel file processing.
PyQt5	Python library for desktop GUI applications.
Agile	Software development methodology emphasizing iterative progress.
Scrum	Agile framework used for iterative development.
UI	User Interface.
Matching Engine	Logic that compares transcript to curriculum.
Fuzzy matching	Technique used to identify strings that are similar but not exactly the same.

Appendix B: Analysis Models

Use cases:

#	Name	Description
UC-01	Upload Transcript	User uploads a transcript file.
UC-02	Upload Curriculum	User uploads curriculum file.
UC-03	Run Matching	System matches transcript with curriculum.

UC-04	View Results	User views matching results.
UC-05	Generate Recommendations	User sees elective suggestions.
UC-06	Export Report	User exports progress report.
UC-07	Switch light and dark themes	The user can toggle between light and dark themes in the settings to personalize the interface.

Class Diagrams

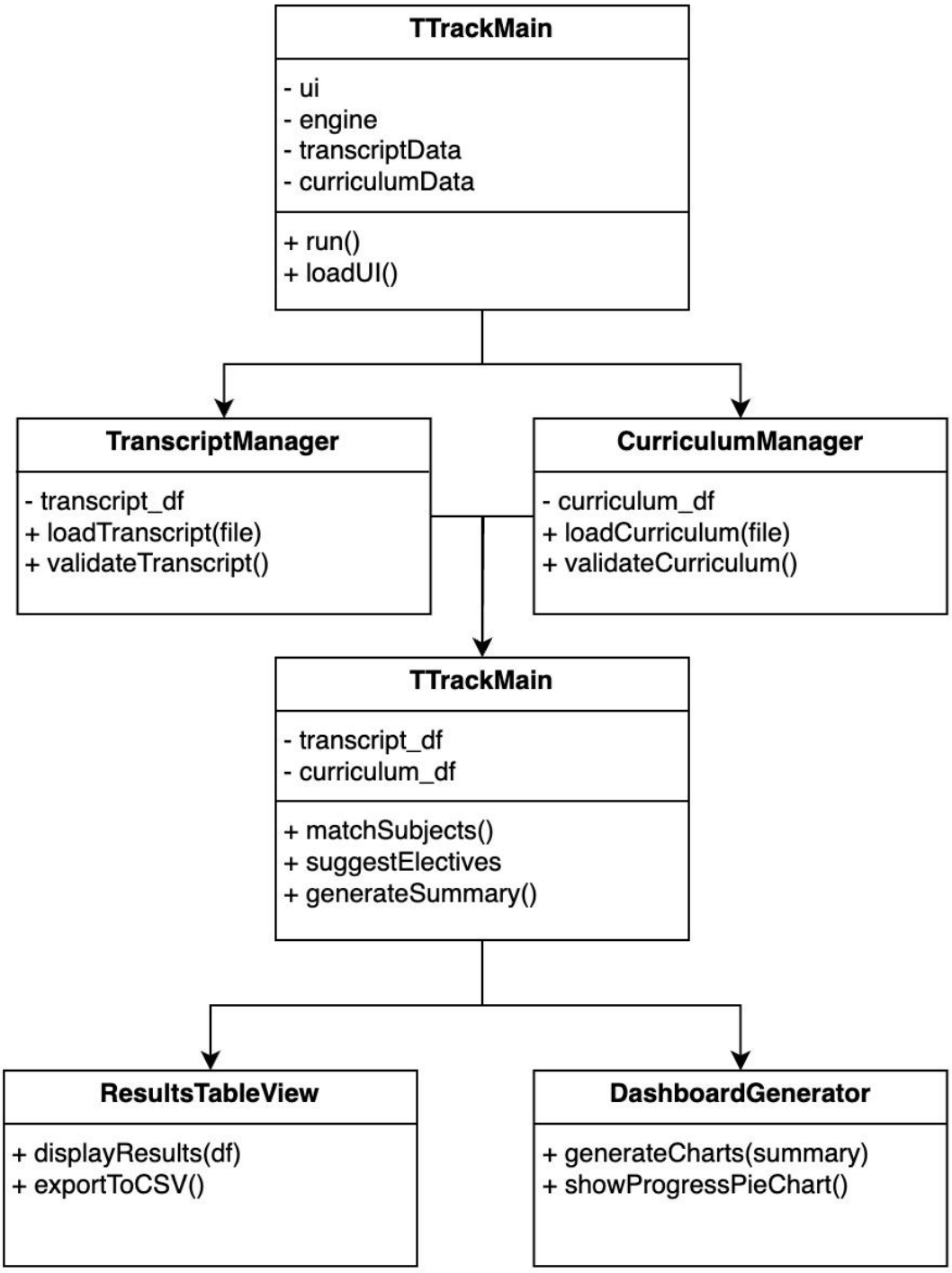


Figure A.3: Class Diagram of TTrack System

Data Flow Diagram

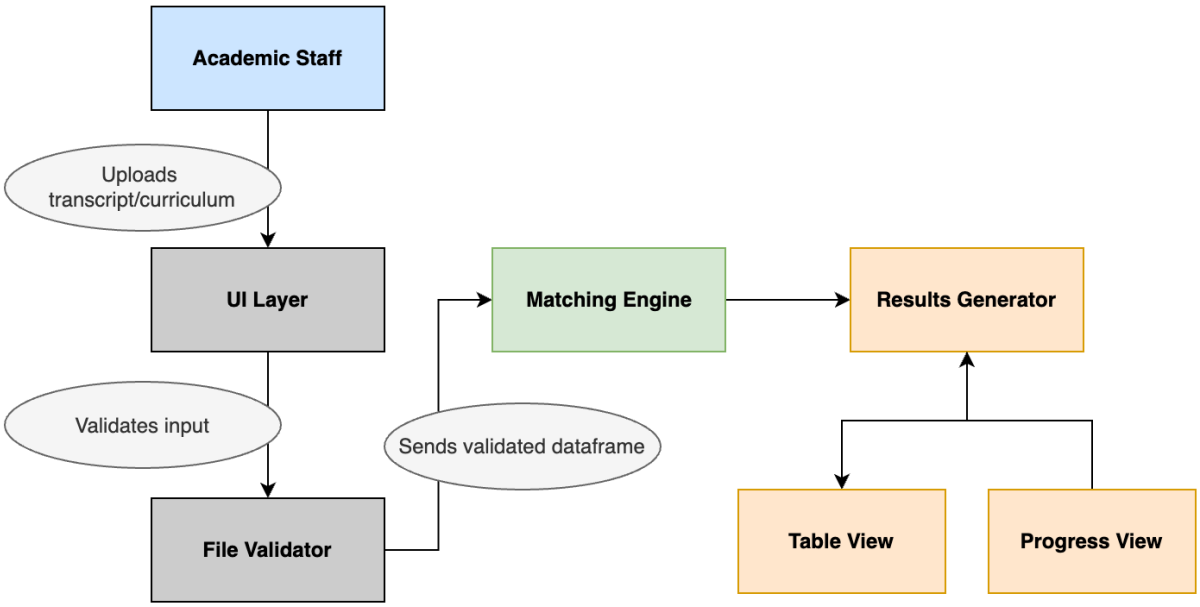


Figure A.4: Data Flow of TTrack System

State Diagram

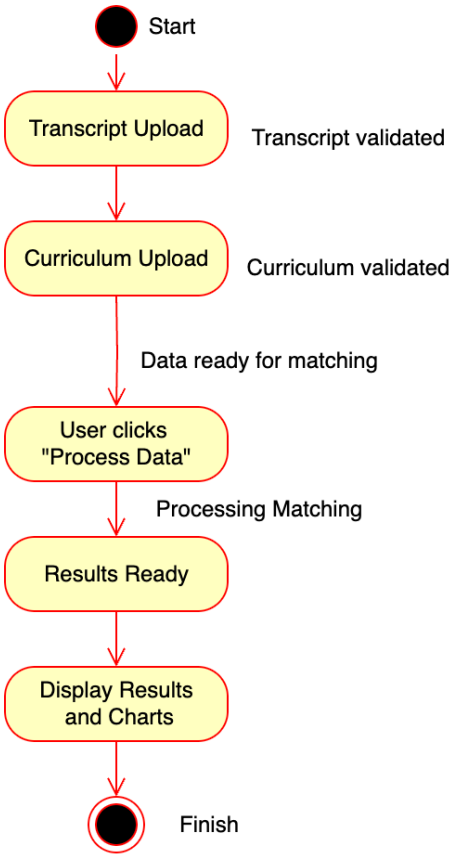


Figure A.5: State Diagram of TTrack System

Sequence Diagram

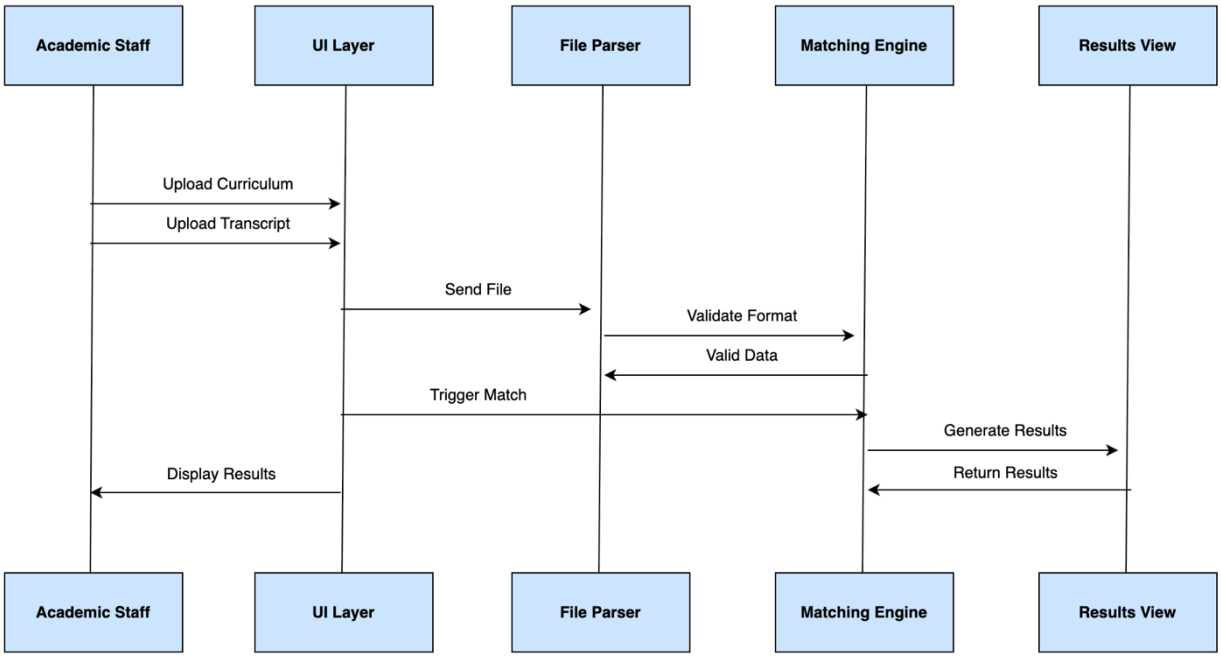


Figure A.6: Sequence Diagram of TTrack System

Appendix C: Product Backlog

User Stories and Acceptance Criteria – INVEST format:

ID	Priority	User Story (INVEST Format)	Acceptance Criteria	Est. Effort (Story Points)	Sprint	Notes / Dependencies
1	High	As a user, I want to upload my transcript file so that TTrack can analyze my completed subjects.	- User can select an Excel file- System validates format - Error shown for invalid files	5	1	Needs UI input tab
2	High	As a user, I want to upload my curriculum file so that TTrack knows degree requirements for matching.	- User can select an Excel file - System validates curriculum structure - Successful import message displayed	5	1	Depends on file input implementation
3	High	As a user, I want TTrack to compare my transcript with the curriculum so I can see which subjects	- Matching logic processes input - Status shown: ✅ Done, ❌ Missing	13	3	Requires uploaded transcript and curriculum

		are completed or missing.	- Handles inconsistent subject codes			
4	High	As a user, I want to view my matched subjects in a table so that I can easily see my progress.	- Results table displays: Subject Code, Name, Type, Status - Table sortable by columns	8	3	Depends on matching engine
5	Medium	As a user, I want TTrack to suggest electives I haven't taken so I can plan my next enrolment.	- Electives missing from transcript shown - Displayed separately from core subjects - Links to curriculum reference	8	4	Requires complete matching data
6	Medium	As a user, I want a dashboard with charts so I can visualize my academic progress.	- Chart shows:% completed by type - Responsive to uploaded data - Supports dark/light themes	8	4	Depends on data aggregation logic
7	Medium	As a user, I want TTrack to support dark mode so I can use it comfortably in different environments.	- System auto-detects OS theme - User can manually toggle themes - UI changes instantly	5	2	None
8	Low	As a user, I want to export my progress report so I can keep a copy for my records.	- Export to PDF or CSV - Matches visual output in UI - Includes summary stats	8	5	Depends on UI and matching output
9	Low	As a user, I want error messages to be clear and user-friendly so I know how to fix issues.	- All error dialogs are descriptive - Suggestions provided for common issues	3	2	Ongoing across sprints
10	Medium	As a user, I want TTrack to save my theme and settings so they persist next time I open the app.	- Preferences saved between sessions - Stored securely in local config	3	2	Related to UI theme work

Appendix D: Test Cases

Use cases:

Test ID	Test Description	Input	Expected Result
TC-01	Upload valid transcript	Valid .xlsx file	System loads file and displays preview.
TC-02	Upload invalid transcript	Missing columns	System displays error message.

TC-03	Run matching engine	Valid files uploaded	System displays matched results.
TC-04	Export report	Click export button	File saved as PDF or CSV.
TC-05	View dashboard	Data available	Charts displayed correctly.
TC-06	Switch to dark mode	The user activates dark mode in the settings	All screens update with dark color scheme
TC-O7	Attempt to upload an unsupported file type	Upload .jpg .pdf file instead of .csv	System displays error: Invalid file format. Only CSV files are allowed.

Appendix E: Requirements Traceability Matrix

Requirement ID	Description	Related Use Case	Priority
REQ-01	Upload Transcript	UC-01	High
REQ-02	Validate Transcript	UC-01	High
REQ-04	Upload Curriculum	UC-02	High
REQ-06	Run Matching Engine	UC-03	High
REQ-09	Display Results Table	UC-04	High
REQ-12	Generate Elective Recommendations	UC-05	Medium
REQ-11	Support light and dark themes	UC-07	Low
REQ-16	Export Report	UC-06	Medium

7. Conclusion

TTrack represents a significant advancement in automating degree tracking for Torrens University. Its offline capabilities, user-friendly interface, and robust data processing engine enable both students and academic staff to save time and make informed decisions.

Future iterations of TTrack may include cloud integration, real-time analytics, and expanded predictive features to further enhance academic planning and support institutional goals.

8. Individual Contribution Log

Student Name	Contribution
Luis Faria	**Tasks Performed:** As a full-stack developer, I led the architecture design and core implementation of TTrack. My responsibilities included setting up the file parsing logic using pandas and openpyxl, configuring the PyQt5 UI for local use and orchestrating the subject matching engine logic to compare completed subjects with curriculum requirements. **Challenges:** The biggest challenge was mapping course codes accurately between transcript and curriculum sheets, especially considering inconsistent formatting in real-world transcripts. I also faced complexity when trying to keep the UI simple while still delivering rich information, such as course status and suggestions. **Lessons Learned:** This project helped me apply software engineering principles in a real-life context. I deepened my skills in Agile project management, version control (via GitHub) and GUI frameworks. Working in sprints helped us deliver incrementally and handle scope adjustments without losing momentum. I also learned how critical it is to clarify data assumptions early, especially when working with unstructured inputs like transcripts. And finally, collaborating in a multidisciplinary group sharpened my communication and coordination skills, which I consider key in project's delivery.
Nomayer Hossain	**Tasks Performed:** For the group project titled TTrack – Degree Tracker, I played a key role in the development of the Functional Requirements section within the Software Requirements Specification (SRS) document. My contributions included: • Writing detailed functional requirements for each system feature: Upload Transcript File, Upload Curriculum File, Run Matching Engine, Display Results Table, Generate Elective Recommendations, and Dashboard Analytics. • Structuring these requirements with consistent use of REQ IDs, stimulus/response sequences, and acceptance criteria. • Ensuring alignment of the functional requirements with the system use cases, product backlog, and UML diagrams. • Reviewing and integrating feedback from teammates to refine and finalize the functional specification content. In addition, I supported the team in preparing Appendix B (Use Cases) by reviewing UC-01 to UC-07 to ensure they accurately reflected the intended software behavior described in the functional section. **Challenges:** One major challenge I faced was ensuring consistency across all artefacts; such as, matching requirements with use cases, product backlog items, and UML diagrams. Since multiple team members were working on different parts of the document, there were initial inconsistencies in terminology, requirement IDs, and feature descriptions.

	Another difficulty was refining the scope and granularity of functional requirements. Striking the right balance between overly generic and overly technical descriptions was a challenge, especially when working with evolving user stories. To resolve the issue of consistency, we implemented a shared requirement-tracking matrix, where we mapped each functional requirement to its respective use case, user story, and test case. This helped us eliminate duplication and ensured traceability. To refine the structure and wording of the functional requirements, I referred to IEEE SRS standards. This reference helped me structure my section professionally and in a way that was easy to understand and assess. We collaborated via MS Teams to synchronise our understanding of requirement definitions and UML integration. This collaborative effort helped ensure our diagrams and textual descriptions were logically aligned. **Lessons Learned:** This project has been a valuable learning experience in collaborative documentation and software requirements engineering. I developed stronger skills in: • Writing clear, testable functional requirements. • Applying traceability principles to connect user needs with design artefacts. • Collaborating effectively within an Agile team setting. Moreover, I gained confidence in using industry-standard formats (like the IEEE SRS template) and in critically reviewing both my work and that of my peers.
Hussain Jameel	**Tasks Performed:** As part of the team working on the TTrack Degree Tracker project, I contributed to reviewing the Software Requirements Specification (SRS) document and ensuring consistency across different sections. I participated in discussions with team members, stayed updated on the project's progress, and supported the overall organization and formatting of the report. I also assisted in minor content structuring and provided input on how the information could be presented more clearly to align with assessment expectations. **Challenges:** One of the main challenges was understanding the technical depth of the content contributed by different members, especially when it involved system features, data matching logic, and architectural decisions. Another challenge was maintaining consistency across various parts of the document as inputs came from multiple people

	with different writing styles. Coordinating schedules for review and feedback was also occasionally difficult due to time constraints. **Lessons Learned:** This project gave me practical exposure to how a professional SRS is structured and the importance of collaborative work in software documentation. I learned how each team member's contribution builds the foundation of a clear and complete requirements document. Additionally, I understood the value of reviewing and refining written work to ensure clarity and consistency.
Rosa Galvis	**Tasks Performed:** My main responsibility remained focused on the User Interface Design, but my participation expanded to include test case creation, backlog development, and collaborative improvements to the documentation and risk management. I proposed and documented **two additional test cases**: • TC-06: Switch to dark mode • TC-07: Attempt to upload unsupported file format I ensured these were aligned with real user scenarios and included expected results for verification. **Backlog Creation:** I created the **product backlog** in a Kanban-style layout using **Notion**, organizing user stories across status categories (To Do, In Progress, Done). Each card includes priority, estimation, and descriptions. **Use Case Traceability:** I assisted in connecting **test cases** to their respective **use cases** for Appendix E (Requirements Traceability Matrix). I verified that the system features (like uploading transcripts, toggling themes) were well-mapped to user needs. **Challenges:** One of the main challenges was effectively integrating into a rapidly moving project. At times, the speed of technical development outpaced my design, requiring me to adapt quickly and find spaces where my contribution as a UI Designer could be more meaningful. Another challenge was understanding some technical aspects of the system, such as integrations or engine logic, which led me to focus on the areas of user experience and documentation, where I could contribute confidently and effectively. **Lessons Learned:**

	This assessment helped me better understand the **importance of diverse roles** in software development projects. While not every team member codes, everyone plays a critical part. I learned how: • **UX/UI design** shapes development priorities and impacts user satisfaction. • **Traceability and testing** help validate whether the design and implementation align with user needs. • **Collaborative tools** like Notion can make backlog management more efficient, especially when the team is decentralized.
Victor Dorantes	**Tasks Performed:** As the database manager, I set up the data schema and implemented Excel parsing using pandas and openpyxl. Tested the app through the development phases to ensure user experience is been considered as an integral part of the software. **Challenges:** Encountered issues with normalizing inconsistent Excel data and ensuring efficient data processing. Faced issues while testing cells containing non-ASCII characters. Found opportunity to learn technologies that allow to connect to SQL server directly within python. TTracker works as a stand-alone app, meaning there is no backend support. Therefore, a technology like Supabase becomes necessary; through this SaaS, we could access a self-hosted SQL database. **Lessons Learned:** Learned the importance of robust data validation and schema design in handling real-world data. Learned to use PyQT to deploy python projects as apps. Learned to integrate openpyxl along pandas library to ensure we take advantage of Excel features.