Product Backlog

ID	Feature Description (User Story)	Story Points	Priority Level
US 1	**Automatic Assignment and Announcement Retrieval/Update:** As a student/student aid, I want to have my complete list of assignments and announcements updated when the application is opened with an internet connection so that I do not miss any important updates.	Extra-Large (13)	1st
US 2	**Planner Organization and Filtering:** As a student/student aid, I want to be able to organize my planner as I wish, whether that be by date and class, or by level of difficulty so that I can better keep on top of my classes and course work.	Large (8)	2nd
US 3	**Offline Access:** As a student/student aid, I want to be able to access my assignments while not connected to the internet, so that I can view assignments at any time/place.	Small (3)	3rd
P-Set up	**First-Run Orientation and Token Setup:** Implement the first-run orientation, including guiding the user to obtain and enter the API token, token validation, and persistence choice.	N/A	High

Team Roles and Ownership

Team Role	Focus Area	Key Components Owned
Member 1: Kenneth	**Application Flow & Logic (GUI Controller)**	`GuiController` (Flow, Navigation, State Management), `Auth Guard`.
Member 2: Daniel	**Data Access Layer (Services/API)**	`Services/API` (Connectivity, Fetching, Refresh Policy), `ApiClient`.
Member 3: Quinten	**User Interface (Displays)**	`Displays` (Main Page, Orientation, Settings Views), UI elements and layout.
Member 4: Amanda	**Data Structure & Persistence (Models/Storage)**	`Models` (Assignment, Announcement), `Storage` (local files: `.profile`, `.assignments`, etc.), and `Ordering Rules` implementation.

Click here for link to SE 370 Project Outline

Sprint Backlog Planning

Sprint 1: Foundation and Authentication

Sprint Goal: Establish the core application architecture, successfully guide the user through the initial setup, and authenticate against the Cougar Courses API to confirm connectivity.

Duration: 2 to 4 Weeks

Design/Flexible Plan: Focus on implementing the `GuiController` logic for startup and the `Orientation` view. The `Services/API` layer will be introduced to handle connectivity checks and token validation.

Task ID	Task Description	Role
1.1	**Implement Core Architecture:** Set up the fundamental classes: `GuiController`, `Displays`, `Models`, and `Services/API`.	Shared
1.2	**Implement `start()` logic:** Decide whether to display orientation or the main week view based on whether it is the first run.	Member 1 (Flow) **Kenny**
1.3	**Implement `Orientation (First-Run) View`:** Include token instructions, token entry, token persistence choice, and start-of-week choice (Sunday or Monday).	Member 3 (UI) ***QUINTEN***
1.4	**Implement `.profile` Storage:** Set up the file structure (`.profile`) to persist orientation status, week starting day, and the user's token (if persistence is chosen).	Member 4 (Data/Storage) **Amanda**
1.5	**Implement `checkConnection()`:** Create the API service function to confirm internet connectivity and perform a lightweight API ping using the entered token.	Daniel
1.6	**Implement Token Validation:** Ensure the `displayOrientation()` flow validates the token and informs the user of success/failure.	Member 1 (Flow) **Kenny**
1.7	**Implement Basic Models:** Define the structure for `Assignment` and `Announcement` (date, time, course, name).	Member 4 (Data/Storage) **Amanda**

Sprint 2: Core Data Retrieval and Display (Completing US 1)

Sprint Goal: Successfully retrieve assignment and announcement data from the API, store it locally, and display the information in a centralized, default organized view on the Main Page.

Duration: 2 to 4 Weeks

Design/Flexible Plan: This sprint targets the heavy lifting of US 1 (Extra-Large, 13 points). Focus on completing the `Services / API (Data Access Layer)` and the `Main Page` display.

Task ID	Task Description	Role
2.1	**Implement `getCourses()`:** Fetch all courses, sort them, and persist them to the `.courses` storage file.	Daniel
2A	**Implement .courses Storage Utilities:** Implement persistence methods to save/load course mapping data to the local .courses CSV file (name, course_id)**.**	Member 4 (Data/Storage) **Amanda**
2.2	**Implement `getAssignments()`:** Fetch assignments, sort them (initially by date/time), and persist them to the `.assignments` storage file.	Daniel
2B	**Implement .assignments Storage Utilities:** Implement persistence methods to save/load assignment data to the local .assignments CSV file (course, name, date, time).	Member 4 (Data/Storage) **Amanda**
2.3	**Implement `getAnnouncements()`:** Fetch announcements, sort them, and persist them to the `.announcements` storage file.	Daniel
2C	**Implement .announcements Storage Utilities:** Implement persistence methods to save/load announcement data to the local .announcements CSV file (course, name, date, time).	Member 4 (Data/Storage) **Amanda**
2.4	**Implement Main Page (Week View):** Display assignments and announcements for the current week context.	Member 3 (UI) ***QUINTEN***
2.5	**Implement:** Create main page controller.	Member 1 (Flow) **Kenny**

2.5.1.	**Implement Basic Navigation:** Add buttons and logic for `displayNextWeek()` and `displayPrevWeek()`.	**Kenny**
2.6	**Implement Refresh Policy:** Set up the automatic hourly refresh of data when online.	Daniel

Sprint 3: Organization, Settings, and Offline Capability (Focus on US 2 & US 3)

Sprint Goal: Implement user customization by allowing organizational preferences (US 2), and ensure the cached data can be reliably viewed when offline (US 3).

Duration: 2 to 4 Weeks

Design/Flexible Plan: This sprint focuses on the remaining User Stories (US 2: Large, 8 points; US 3: Small, 3 points). The `Settings View` is the central component, controlling organization and token management.

Task ID	Task Description	Role
3.1	**Implement `Settings View`:** Create the UI to view/change preferences, including start-of-week and ordering options.	Member 3 (UI) ***QUINTEN***
3.2	**Implement Ordering Rule 1 (Course then Date):** Apply sorting logic: group by course (A→Z), then sort by date ascending, then by time ascending.	Member 4 (Data/Storage) **Amanda**
3.3	**Implement Ordering Rule 2 (Difficulty):** Apply sorting logic: sort by difficulty descending (5→1, hardest first), then by date and time.	Member 4 (Data/Storage) **Amanda**
3.4	**Allow User-Set Difficulty:** Implement the ability for the user to set the difficulty attribute (1=easiest, 5=hardest) on an assignment.	Member 4 (Data/Storage) **Amanda**

3.5	**Implement Offline Viewing (US 3):** Ensure that if the application is offline, it successfully loads and displays the cached data stored in the local `.assignments` and `.announcements` files.	Member 4 (Data/Storage) **Amanda**
3.6	**Implement Offline Messaging:** Display a banner to the user indicating they are offline and showing cached data.	Member 3 (UI) ***QUINTEN***
3.7	**Implement Token Management in Settings:** Allow the user to save or delete a saved token.	Member 1 (Flow) **Kenny**