Cougar Planner Project Outline

0) Files and formats

`.profile` (key=value; defaults shown; keep keys exactly as written)

Parameter	Default values	Description/Values
`authToken`	`empty`	Token or empty
`orientationCompleted`	`false`	
`storeData`	`true`	"true" to write CSVs locally; "false" to never write local data
`storeToken`	`true`	"true": token stored in .profile; "false": user re-enters at launch
`weekStart`	`monday`	Monday or Sunday
`defaultView`	`week`	Week or Day
`openTo`	`current_week`	Current week or last Viewed week
`defaultSort.mode`	`date_time`	YYYY-MM-DD
`defaultSort.difficultyOrder`	`ascending`	Descending means things due much later are shown first instead
`showRefreshCountdown`	`false`	True shows a timer somewhere for when the app will refresh again
`lastViewedWeekStart`		Date for the last viewed week if they chose this option

`lastViewedDay`		Same as above but for day

Local data (CSV; one file per entity, lowercase snake_case headers)

`courses.csv`

- `course_id`
- `course_name`

`assignments.csv`

- `assignment_id`
- `course_id`
- `assignment_name`
- `due_date` (YYYY-MM-DD or empty)
- `due_time` (HH:MM or empty)
- `difficulty` (1|2|3|4|5 or empty)

`announcements.csv`

- `announcement_id`
- `course_id`
- `title`
- `posted_at` (YYYY-MM-DD HH:MM)
- `body` (optional; may be empty)

---

1) Orientation (first run)

- **Intro:** Explain a Canvas token is required and how to get it.
- **Token entry:** Single plain-text input.
- **Required warning (group can finalize exact text):**
  "Your Canvas token is stored in plain text in a local file. Anyone with access to this device could use it. You are responsible for keeping it safe."

- **Choices (show short descriptions when choosing an option for each setting):**
    - Week start: Sunday | Monday
    - Open to: Current week | Last viewed week
    - Default view: Week | Day
    - Default sort: Date→Time or Difficulty (Asc/Desc)
    - Store token: Yes (keep in `.profile`) | No (user re-enters at launch)
    - Store data (offline): Yes (`storeData=true`) | No (`storeData=false`)
    - Show refresh countdown: Yes | No
- Write `.profile` (put `authToken` on the **first line**) and set `orientationCompleted=true`.

---

2) Launch decision tree (every run)

1. Read `.profile`. If file missing or `orientationCompleted!=true` → go to Orientation.
2. If `storeToken=true` and `authToken` is empty → **Token-Missing** screen:
   Message: token not found; enter a token or proceed offline (if any data exists).
   Actions: Enter Token, Continue Offline.
3. If `storeToken=false` → **Enter Token** screen
   Actions: Enter Token, Continue Offline.
4. Otherwise → **Main app**:
    - If `openTo=current_week`: compute current week bounds (system timezone) and open.
    - If `openTo=last_viewed`: open stored week/day if present; else fallback to current week.

---

3) Main app behavior

3.1 Navigation

- Week/Day toggle; Prev / Today / Next.
- Prev/Next jumps exactly one week (or one day).
- Today jumps to the week/day containing “now”.
- Two tabs: **Announcements** and **Assignments**.

3.2 Announcements

- Week view: show those posted within the selected week.
- Day view: show those posted on the selected day.

3.3 Assignments

- Show assignments for the selected period (Week/Day).
- Inline difficulty editor (1–5). If `storeData=true`, save/update to `assignments.csv`.

---

4) Sorting rules (simplified and consistent)

4.1 Date→Time

Order by:

1. `due_date` ascending (empty dates go **last**)
2. `due_time` ascending within the same date (empty times go **last**)

3. `course_name` A→Z
4. `assignment_name` A→Z

4.2 Difficulty (Asc/Desc) → Date/Time

Order by:

1. `difficulty` (Asc or Desc per `defaultSort.difficultyOrder`; empty difficulty goes **last**)
2. `due_date` ascending (empty dates last)
3. `due_time` ascending (empty times last)
4. `course_name` A→Z
5. `assignment_name` A→Z

---

5) Refresh & “changes since last open”

- A background refresh runs every **15 minutes** while the app is open.
  (No manual refresh; no refresh immediately on open.)
- After a successful refresh, compute **new items count** compared to the last snapshot.
  “New” = items whose IDs were not present in the prior snapshot.
- If count > 0, show a **changes summary** (name/appearance is a [Group decision]).
  If count == 0, show nothing.
- Update the snapshot.

---

6) Data fetch policy (clear & predictable)

6.1 Initial and routine fetching

- **On first open after Orientation:** fetch **current week only**.
- **On navigation to a week:** if we haven't fetched that week before in this session, fetch it **once** (and cache to CSV if `storeData=true`).
- **On each 15-minute refresh:** fetch and sync **current week + future weeks only**

6.2 When an assignment's due date moves

- Because we always fetch current/future on the 15-minute loop, a moved assignment into current/future will appear with its new `due_at`.
- We **upsert by ID** in `assignments.csv` (update fields; **preserve difficulty**).
- We **do not** auto re-fetch old/past weeks.

---

7) Networking & errors (user-facing behavior)

- **Invalid/missing token (401/403 or blank):** show a banner with an action to Enter Token. Pause further requests until fixed.
- **No internet:** show a banner.
  - If `storeData=true`: show saved CSV data.
  - If `storeData=false`: lists are empty with an explanatory message.
- **Rate limiting (429):** wait 60 seconds, retry once quietly; otherwise let the next 15-minute refresh recover.
- **Timeouts/5xx:** treat like "No internet" for the user; recover on the next cycle.
- **Security warning placement (required, wording [Group decision]):**
  Next to token entry in Orientation, and next to token input when **Change Token** is used in Settings.

---

8) Settings (mirrors Orientation; no extras)

- Week start: Sunday | Monday
- Open to: Current week | Last viewed
- Default view: Week | Day
- Default sort: Date→Time or Difficulty (Asc/Desc)
- Store token: Yes/No (token never displayed)
- Store data: Yes/No
- Show refresh countdown: Yes/No
- **Change Token:** plain text input; overwrite `.profile` `authToken` on save
- **Delete Token:** clear `authToken` in `.profile` (confirm first)

---

9) Merge logic

- On any fetch, for each returned item **(by ID)**:
    - **If ID exists** in CSV: update its fields with latest from API.
    - **If ID does not exist:** append a new row.
    - **Never** duplicate rows.
- `difficulty` is local; **do not overwrite** it during updates (only change if the user edits it in the UI).

---

10) Task ordering for implementation (top-down, matches user flow)

1. **File I/O utilities:** read/write `.profile`; read/write CSVs (append, update by ID).
2. **Orientation screens:** including Review & Confirm + token warning placement.
3. **Launch controller:** implement the decision tree.
4. **Canvas Service:** minimal fetch for a given week range; parse → DTOs → CSV upsert.
5. **Main view:** Week/Day toggle; Prev/Today/Next; render Announcements + Assignments.
6. **Sorting:** implement the simplified Date→Time and Difficulty modes exactly as specified.
7. **Difficulty editor:** inline stars; write to CSV (if `storeData=true`).
8. **Auto-refresh timer:** 15-minute loop; compute “new items count”; show summary only if > 0 ([Group decision] for appearance).
9. **Past navigation fetch:** on navigating to any week not yet fetched this session, fetch it once (covers past weeks **on demand**).
10. **Settings:** mirror Orientation choices; Change/Delete Token (plain text); no token display.

File tree for visualization

```
cougar-planner/
├── pom.xml
├── README.md
├── data/                          # runtime data (gitignored)
│   ├── .profile                   # created by Orientation
│   ├── courses.csv                # optional if storeData=true
│   ├── assignments.csv
│   ├── announcements.csv
│   └── snapshots.json             # last seen IDs for "changes since last open"
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── com/cougarplanner/
│   │   │       ├── App.java                  # JavaFX launcher
│   │   │       ├── config/
│   │   │       │   ├── Profile.java
│   │   │       │   ├── ProfileReader.java
│   │   │       │   ├── ProfileWriter.java
│   │   │       │   └── SettingsValidator.java
│   │   │       ├── io/
│   │   │       │   ├── CsvPaths.java
│   │   │       │   ├── CsvReader.java
│   │   │       │   ├── CsvWriter.java
│   │   │       │   ├── CoursesRepository.java
│   │   │       │   ├── AssignmentsRepository.java
│   │   │       │   └── AnnouncementsRepository.java
│   │   │       ├── model/
│   │   │       │   ├── Course.java
│   │   │       │   ├── Assignment.java
│   │   │       │   ├── Announcement.java
│   │   │       ├── service/
│   │   │       │   ├── CanvasService.java
│   │   │       │   ├── MergeService.java
│   │   │       │   ├── API.java
│   │   │       │   └── RefreshScheduler.java
│   │   │       ├── controller/
│   │   │       │   ├── OrientationController.java
│   │   │       │   ├── MainController.java
│   │   │       ├── view/
│   │   │       │   ├── MainView.fxml
│   │   │       │   ├── AnnouncementsView.fxml
│   │   │       │   ├── AssignmentsView.fxml
│   │   │       │   ├── OrientationView.fxml
│   │   │       │   │
│   │   │       │   ├── SettingsView.fxml
│   │   │       │   ├── Banners.fxml
│   │   │       │   └── CountdownView.fxml
│   │   │       └── util/
│   │   │           ├── DateTimeUtil.java
│   │   │           ├── WeekUtil.java
│   │   │           └── Sorting.java
│   │   └── resources/
│   │       ├── styles.css
│   │       └── icons/...
│   └── test/
│       └── java/
│           └── com/cougarplanner/
│               ├── config/ (ProfileReader/Writer tests)
│               ├── io/ (Repositories upsert tests)
│               ├── service/ (MergeService/SnapshotService tests)
│               └── util/ (Sorting/WeekUtil tests)
│
```

High-level packages

- `config` – `.profile` loading/writing/validation
- `io` – CSV paths, readers/writers, repositories (upsert by ID)
- `model` – domain classes (Course, Assignment, Announcement) + enums
- `service` – Canvas fetch, merging, snapshots, refresh scheduler
- `controller` – Orientation/Launch/Main/Settings flows
- `view` – JavaFX UI (FXML or code-only panes)
- `util` – time, week ranges, sorting comparators, small helpers

Key classes (what each is responsible for)

config/

- `Profile` – in-memory representation of `.profile` (all fields as strings; helpers to get parsed booleans/dates).
- `ProfileReader` – read `.profile`, trim whitespace, fill defaults.
- `ProfileWriter` – write `.profile` atomically with `authToken` on first line.

- `SettingsValidator` – ensure values ∈ allowed sets; return warnings + sanitized copy.

io/

- `CsvPaths` – centralizes file locations (`data/courses.csv`, etc.).
  `CsvReader` – generic CSV → list of maps (handles empty fields).
- `CsvWriter` – generic writer (append/overwrite, atomic write).
- `CoursesRepository` – read/all, upsert by `course_id`.
- `AssignmentsRepository` – read/all, query by week/day, **upsert by `assignment_id`** (preserve `difficulty`).
- `AnnouncementsRepository` – read/all, query by week/day, upsert by `announcement_id`.

model/

- `Course` – `course_id`, `course_name`.
- `Assignment` – `assignment_id`, `course_id`, `assignment_name`, `due_date`, `due_time`, `difficulty`.
- `Announcement` – `announcement_id`, `course_id`, `title`, `posted_at`, `body`.
- `SortMode` – `DATE_TIME`, `DIFFICULTY`.
- `DifficultyOrder` – `ASCENDING`, `DESCENDING`.
- `WeekStart` – `SUNDAY`, `MONDAY`.
- `DefaultView` – `WEEK`, `DAY`.
- `OpenTo` – `CURRENT_WEEK`, `LAST_VIEWED`.

service/

- `CanvasService` – fetch current/selected week; map JSON→`model` objects.
- `MergeService` – **idempotent upsert by ID**; never duplicates; preserve assignment `difficulty`.
- `RefreshScheduler` – 15-minute loop; pause/resume on app focus; triggers Canvas fetch + merge + snapshot compare.

controller/

- `OrientationController` – token entry + security warning; settings UI; Review & Confirm → writes `.profile`.
- `MainController` – parent for tabs; wires data providers for current Day/Week; owns navigation state.

view/ (names are suggestions; you can use FXML or code)

- `MainView.fxml` / `MainView.java` – shell with tabs (Announcements, Assignments), toolbar (Week/Day, Prev/Today/Next).
  `AnnouncementsView.fxml` – table/list with posted_at.
- `AssignmentsView.fxml` – table/list with due_date/time, inline difficulty stars.
- `OrientationView.fxml` – token + choices + Review page.
- `SettingsView.fxml` – same choices + Change/Delete Token.
- `Banners.fxml` – reusable error/info banners (token missing, offline, rate limit).
- `CountdownView.fxml` – optional refresh countdown widget.

util/

- `DateTimeUtil` – parse/format `YYYY-MM-DD`, `HH:MM`; "empty goes last" helpers.
- `WeekUtil` – compute week range from `WeekStart` + a `LocalDate`.
- `Sorting` – two comparators: Date→Time; Difficulty→Date/Time (exact spec).
- `Either` / small result types – optional; neat error handling without exceptions.
- `Csv` small helpers – quoting/escaping if needed.