Campus Course & Records Manager (CCRM) — Project Deliverables & Template

This document is a ready-to-use project README + `requirements.md` + code skeleton and supporting artifacts for the CCRM Java SE assignment .

Source: Project brief

This template follows the project brief (Campus Course & Records Manager). The original brief is attached and referenced in the repo. filecite turn0file0

What's included in this template

1. `README.md` — detailed README ready for your repo submission (how to run, JDK used, screenshots checklist, mapping table linking syllabus topics to files, notes for assertions and running tests).
2. `requirements.md` — any special steps needed to run the project locally (build & run commands, environment variables, CSV sample format, sample data location).
3. `USAGE.md` — examples of CLI commands and sample workflows to demonstrate the minimum demo flow.
4. `skeleton/` — starter Java package structure and key class skeletons (one-file-per-class is suggested; actual code should be added to these files):
    - `edu.ccrm.cli.Main.java` (runnable main class)
    - `edu.ccrm.domain.Person.java` (abstract)
    - `edu.ccrm.domain.Student.java`
    - `edu.ccrm.domain.Instructor.java`
    - `edu.ccrm.domain.Course.java` (with `Course.Builder`)
    - `edu.ccrm.domain.Enrollment.java`
    - `edu.ccrm.domain.enums.Semester.java`
    - `edu.ccrm.domain.enums.Grade.java`
    - `edu.ccrm.service.*` interfaces and simple implementations
    - `edu.ccrm.io.ImportExportService.java` and `BackupService.java`
    - `edu.ccrm.util.FileUtils.java` (recursive directory utilities)
    - `edu.ccrm.config.AppConfig.java` (Singleton)
5. `test-data/` — two minimal CSV files (`students.csv`, `courses.csv`) and a short README describing format.
6. `CHECKLIST.md` — submission checklist (README + screenshots + optional video link + sample run evidence).

How to use this template

1. Copy the files into a new Git repository (or fork if you prefer). Keep `README.md` and `USAGE.md` updated with your screenshots and exact JDK version.
2. Implement the detailed logic required by the brief using the given packages.
3. Run locally and capture screenshots for the `screenshots/` folder (JDK install, eclipse project, CLI run showing menus, export/backup folder tree).
4. Push to GitHub and place the repo link in your submission portal.

Quick run instructions (to paste in README `Quick Start`)

1. JDK: Java 17 (LTS) or Java 11 is acceptable — specify the exact version you used.
2. Build: `javac -d out $(find src -name "*.java")`
3. Run: `java -cp out edu.ccrm.cli.Main`

(If using an IDE, include the `src` folder as a Java project and run `edu.ccrm.cli.Main`.)

`requirements.md` (special runtime notes)

- JDK required: Java 17 (or the version you used) — include exact download link in README.
- No external libraries required — pure Java SE.
- Optional: If you add logging or JSON libs (e.g., Gson), add a `lib/` folder and update the `README.md` with classpath instructions.
- How to enable assertions: run with `java -ea -cp out edu.ccrm.cli.Main`.

Sample CSV formats (place in `test-data/`)

`students.csv` (header line required):

```
id,regNo,fullName,email,status,createdAt
S1001,REG2025/001,Arun Kumar,arun@example.com,ACTIVE,2025-09-01
S1002,REG2025/002,Sana Rao,sana@example.com,ACTIVE,2025-09-02
```

`courses.csv`:

```
code,title,credits,instructor,semester,department
CS101,Intro to Programming,3,Dr. Mehta,SPRING,Computer Science
MA201,Calculus II,4,Dr. Ali,FALL,Mathematics
```

Suggested minimal `Main` workflow (for demo)

- AppConfig singleton loads `data` folder path.

- Show menu with options: Manage Students, Manage Courses, Manage Enrollment/Grades, Import/Export, Backup & Show Size, Reports, Exit.
- Keep sample commands in `USAGE.md` that run through the minimum demo flow required by the brief.

Mapping table (Syllabus topic → File/Place in repo)

Create a table in `README.md` that maps each required topic to a file. Example entries:

- Streams & NIO.2 → `edu.ccrm.io.ImportExportService` and `edu.ccrm.io.BackupService`
- Enums & Grade mapping → `edu.ccrm.domain.enums.Grade`
- Singleton → `edu.ccrm.config.AppConfig`
- Builder → `edu.ccrm.domain.Course.Builder`
- Nested classes → documented inside `Course` (static nested) and `Transcript` (inner)
- Custom exceptions → `edu.ccrm.exception.MaxCreditLimitExceededException`

CHECKLIST for submission

- ☐ GitHub repo link included
- ☐ `README.md` with: run instructions, Java timeline, Java SE/ME/EE comparison, JDK/JRE/JVM description
- ☐ `requirements.md` present and accurate
- ☐ `USAGE.md` showing the minimum demo flow
- ☐ `test-data/` CSVs included
- ☐ `screenshots/` folder with required screenshots
- ☐ Optional demo video link in README
- ☐ Ensure plagiarism/AI notes in acknowledgements if you used references

---

Academic Integrity note

Follow the project's academic integrity notes: ensure the submitted code is your work and cite any references you used in `README.md` acknowledgements. The automated evaluator may check for AI content — keep a log of edits and design rationale in the README to show authorship.

Next steps I can do for you (pick one, or ask for another):

- Generate the full skeleton Java source files (one-per-class) ready to paste into `src/`.

- Produce the full `README.md` text with the Java evolution bullets, Java ME vs SE vs EE comparison, and JDK/JRE/JVM explanation prefilled (so you can paste directly to repo).
- Create the `students.csv` and `courses.csv` sample files and place them in `test-data/` (I can output them as downloadable files).

If you want any of the above, tell me which and I will generate them next.

End of template.

Name- Yash

Reg. No.- 24BCE10442