Project: Mini Vault CLI (Node.js + OOP)

Project Story

You are building **Mini Vault**, a small **command-line application** used by a company to manage **private notes**.

The application runs entirely in the **terminal** and allows users to:

- Register
- Login
- Manage their own private notes

Data Storage Rules (IMPORTANT)

All data must be stored **locally in arrays**.

Users Data

A file that exports a **users array**.

Each user object must include:

- `id` (string)
- `username` (string, unique)
- `password` (string) ← **plain text for now**
- `createdAt` (ISO date string)

Example structure (conceptual):

- `users = [`
- `  { id, username, password, createdAt }`

- `]`

Notes Data

A file that exports a **notes array**.

Each note object must include:

- `id` (string)
- `ownerUsername` (string)
- `text` (string)
- `createdAt` (ISO date string)

Each user can only access **their own notes**.

OOP Responsibilities

Models (Entities)

- **User**
    - Represents a system user
    - Holds user data only
- **Note**
    - Represents a private note
    - Holds note data only

Repositories (Data Layer)

Repositories are the **only place allowed to access the arrays**.

UserRepository

- `findByUsername(username)`
- `exists(username)`
- `add(user)`

NoteRepository

- `add(note)`
- `listByOwner(username)`
- `deleteById(ownerUsername, noteId)`

Services (Business Logic)

AuthService

- `register(username, password)`
    - validates input
    - ensures unique username
    - creates a User object
    - stores it using UserRepository
- `login(username, password)`
    - verifies credentials
    - returns a logged-in user object

VaultService

- `addNote(ownerUsername, text)`
- `listNotes(ownerUsername)`
- `deleteNote(ownerUsername, noteId)`

UI Layer (CLI)

CliMenu

- Displays menus
- Reads user input (`readline-sync`)
- Prints messages

Must NOT:

- access data arrays
- contain business logic

Application Controller

App

- Controls the app flow
- Manages session state:
    - `currentUser = null | { username }`
- Switches between guest and logged-in menus

Required Menus

Guest Menu

1. Register
2. Login
3. Exit

Logged-in Menu

1. Add note
2. List notes
3. Delete note
4. Logout

Validation Rules

- Username:
    - required
    - minimum 3 characters
    - must be unique
- Password:
    - required
    - minimum 6 characters

- Note text:
    - required
    - maximum 120 characters

Security (Core vs Bonus)

Core (Required)

- When a user registers, the system must automatically create **TWO** examples notes for that user**.** You want it to happen from the **User class constructor**, by calling a function that creates a note with the same Notes structure for these examples.
- Password is stored as **plain text**
- Logic must be written so hashing can be added later
- No tokens, no sessions, no JWT

BONUS – Security Upgrade

Students may replace:

- `password`

with:

- `passwordHash`

And use **bcrypt** to:

- hash passwords on registration

- compare passwords on login

Bonus points are awarded only if:

- bcrypt is implemented correctly
- plain passwords are never stored

Optional Bonus Ideas

- Change password feature
- Login attempt limit
- Search notes by keyword