

PLAN GESTIUNE LABORATOARE STUDENTI - LAB 7-9

Paul Tal

LISTA FUNCȚIONALITĂȚI

F1. Adaugă student în sistem **F2.** Șterge student după ID **F3.** Modifică datele unui student **F4.** Căutare student după nume, ID sau grupă **F5.** Listare toți studenții **F6.** Adaugă problemă de laborator **F7.** Șterge problemă după ID laborator **F8.** Modifică datele unei probleme **F9.** Căutare problemă după ID **F10.** Listare toate problemele **F11.** Creare teme (atribuire problemă la student) **F12.** Notare temă **F13.** Statistici și rapoarte

PLANUL DE ITERAȚII

Iterația	Funcționalități
I1	F1-F10 + CLI de bază + Help
I2	F11-F12 + Gestionare teme
I3	F13 + Statistici și rapoarte

MODELAREA ITERAȚIEI 1

F1 - Adăugare student

Utilizator	Program	Descriere
Start aplicație	CLI Prompt	Afișare prompt principal
Tastează "add_student John 917"	"Student added successfully: ID 1, Name: John, Group: 917"	Confirmare adăugare
Tastează "list_students"	Tabel cu studenți	Afișare listă completă

F4 - Căutare student

Utilizator	Program	Descriere
Lista: [John-917, Jane-918, Bob-917]	Context pentru căutare	Bază de date existentă
Tastează "search_student name John"	"STUDENTS MATCHING NAME: 'John'"	Căutare după nume
	"ID: 1, Name: John, Group: 917"	Rezultat găsit
Tastează "search_student group 917"	Lista cu 2 studenți din grupa 917	Căutare după grupă

F6 - Adăugare problemă

Utilizator	Program	Descriere
Tastează "add_problem 7_1 'Sort array' 2024-12-15"	"Problem added successfully: Lab 7, Problem 1"	Confirmare adăugare
Tastează "list_problems"	Tabel cu probleme	Afișare listă completă

F9 - Căutare problemă

Utilizator	Program	Descriere
------------	---------	-----------

Lista: [7_1-Sort, 7_2-Search, 8_1-Matrix]	Context pentru căutare	Bază de probleme
Tastează "search_problem 7_1"	"PROBLEM MATCHING ID: '7_1'"	Căutare după ID
	"Lab: 7, Problem: 1, Description: Sort array"	Rezultat găsit

TASK-URI ITERAȚIA 1

T1. Clasa Student

- Constructor cu ID, nume, grupă
- Metode: get_id(), get_name(), set_name(), get_group(), set_group()

T2. Clasa Problem

- Constructor cu lab_number, problem_number, description, deadline
- Metode getter și setter pentru toate proprietățile

T3. StudentRepository

- add_student(), remove_student(), modify_student()
- get_all_students(), search_students()

T4. ProblemRepository

- add_problem(), remove_problem(), modify_problem()
- list_problems(), search_problems_by_id()

T5. StudentService

- add_student(), list_students(), remove_student()
- Logica de business pentru validări

T6. CLI Interface

- Comenzi pentru toate operațiile CRUD
- Help system și error handling

T7. Integrare și testare

- Test runner automat pentru toate modulele

MODELAREA ITERAȚIEI 2

F11 - Creare temă

Utilizator	Program	Descriere
Studenti: [1-John, 2-Jane], Probleme: [7_1, 7_2]	Context pentru atribuire	Entități existente
Tastează "create_assignment 1 7_1"	"Assignment created: Student John assigned to Problem 7_1"	Confirmare atribuire
Tastează "list_assignments"	Tabel cu toate temele	Verificare atribuirii

F12 - Notare temă

Utilizator	Program	Descriere
Tema existentă: Assignment ID 1	Context pentru notare	Temă de evaluat

Tastează “grade_assignment 1 9”	“Assignment graded: Student John - Problem 7_1 - Grade: 9”	Confirmare notă
Tastează “list_assignments”	Afișare note actualizate	Verificare evaluări

TASK-URI ITERAȚIA 2

T8. Clasa Assignment

- Constructor cu student_id, problem_id, grade (opțional)
- Metode pentru gestionarea atribuirilor și notelor

T9. AssignmentRepository

- create_assignment(), grade_assignment()
- list_assignments(), search_assignments()

T10. AssignmentService

- Logica de business pentru atribuirii
- Validări pentru studenți și probleme existente

T11. CLI pentru Assignments

- Comenzi create_assignment și grade_assignment
- Integrare cu sistemul existent

T12. Testare completă iterația 2

- Teste pentru Assignment sistem
- Integrare cu test runner

MODELAREA ITERAȚIEI 3

F13 - Statistici studenți

Comandă	Output
stats_students	Lista studenți cu note medii
	“John (Group 917): Average 8.5, Assignments: 3”
	“Jane (Group 918): Average 9.2, Assignments: 2”
stats_problems	Lista probleme cu rata de rezolvare
	“Problem 7_1: Completed by 85% students, Average grade: 8.3”

F13 - Rapoarte

Comandă	Output
report_group 917	Raport pentru grupa 917
	“Total students: 15, Average grade: 8.1”
	“Best student: John (9.2), Lowest: Bob (6.5)”
export_grades csv	Export note în format CSV
	“File exported: grades_2024.csv”

TESTE ITERAȚIA 1

Student Model

Input	Output așteptat
Student(1, "John", 917)	get_name() = "John"
student.set_name("Johnny")	get_name() = "Johnny"
student.set_group(918)	get_group() = 918

Student Repository

Operație	Rezultat așteptat
add_student("John", 917)	Student cu ID auto-generat
search_students("john", "name")	Lista cu studenți care conțin "john"
search_students("917", "group")	Lista studenți din grupa 917
remove_student(1)	Student eliminat, ValueError dacă nu există

Problem Model și Repository

Operație	Rezultat așteptat
Problem(7, 1, "Sort", date(2024,12,15))	Obiect problemă valid
add_problem(7, 1, "Sort", deadline)	Problemă adăugată în repository
search_problems_by_id("7_1")	Lista cu problema găsită
search_problems_by_id("invalid")	Listă goală

Student Service

Operație	Rezultat așteptat
add_student("John", 917)	Student adăugat prin service layer
list_students()	Lista tuturor studenților
remove_student(999)	ValueError cu mesaj descriptiv

ARHITECTURA

```
main.py
├─ test_runner.py
├─ app.py
├─ ui/
│   └─ cli.py (comenzi CRUD + help)
├─ student/
│   ├── model.py (Student class)
│   ├── repository.py (StudentRepository)
│   └─ service.py (StudentService)
└─ problem/
    ├── model.py (Problem class)
    └─ repository.py (ProblemRepository)
```

COMENZILE CLI IMPLEMENTATE

Operații Student

Comandă	Descriere
add_student <name> <group>	Adaugă student nou
remove_student <id>	Șterge student după ID
list_students	Afișează toți studenții
search_student <type> <term>	Căută după name/id/group

Operații Problem

Comandă	Descriere
add_problem <lab_problem> <desc> <deadline>	Adaugă problemă nouă
remove_problem <lab_problem>	Șterge problemă după ID
list_problems	Afișează toate problemele
search_problem <lab_problem_id>	Căută problemă după ID

Comenzi Utilitate

Comandă	Descriere
help	Afișează lista comenzilor
clear	Șterge ecranul
exit/quit	Închide aplicația

FORMATE INPUT

Format Student

Exemplu	Descriere
add_student “John Doe” 917	Nume cu spații între ghilimele
search_student name John	Căutare parțială case-insensitive
search_student group 917	Căutare exactă după grupă

Format Problem

Exemplu	Descriere
add_problem 7_1 “Sort array” 2024-12-15	Lab_problem în format X_Y
search_problem 7_1	ID problemă în format X_Y
remove_problem 8_2	Ștergere după ID compus

VALIDĂRI ȘI ERROR HANDLING

Validări Student

Validare	Mesaj Error
ID inexistent la remove	“Student with id X not found”
Grupă invalidă	ValueError pentru input non-numeric

Nume gol	Acceptat - nu este validat
----------	----------------------------

Validări Problem

Validare	Mesaj Error
Format ID invalid	"lab_problem must be in format X_Y"
Deadline invalid	ValueError pentru format dată
Problem inexistentă la remove	"Problem X.Y not found"

TESTE COVERAGE

Modul	Teste	Coverage
student.model	Constructor, getters, setters	100%
student.repository	CRUD, search, error handling	100%
student.service	Business logic, error propagation	100%
problem.model	Constructor, getters, setters, dates	100%
problem.repository	CRUD, search, composite ID	100%

Rulare teste: `python3 test_runner.py`