Capstone C271

Data Structures I

Drew Jordan, Ramiz Hameed and Julian Solis

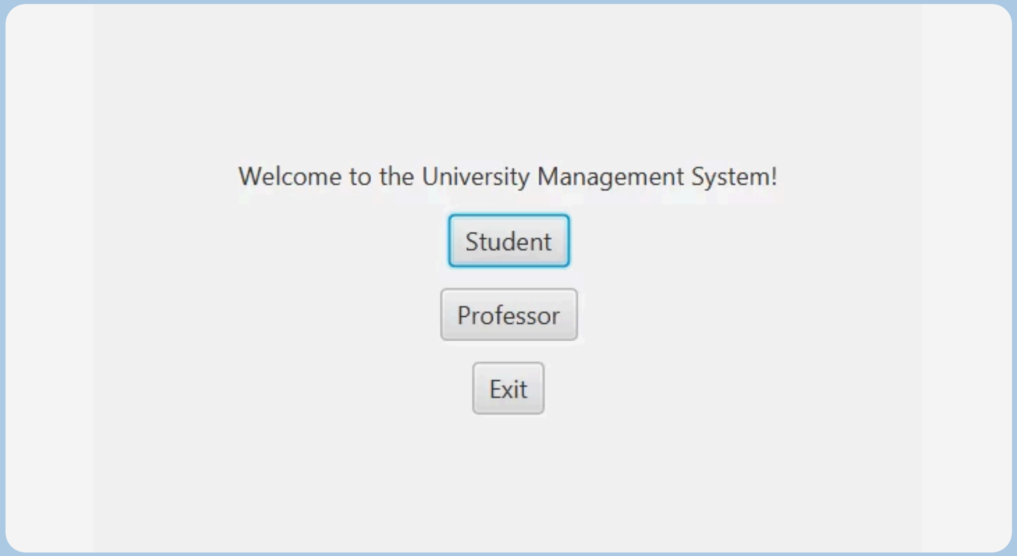
App introduction

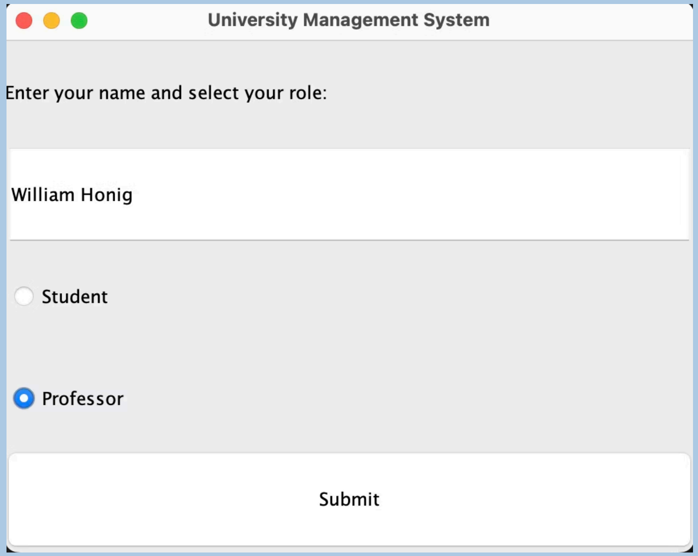
Enhanced Course Management and Semester Integration

Overview:

1. Semester-Based Course Enrollment: semester-specific course enrollments, allowing students to register or drop courses based on the designated semester
2. Professor Assignments and Tracking: Enable assigning professors to courses each semester with a user-friendly GUI. Provide functionality to generate and sort a list of all courses a named professor is teaching in a given semester.
3. Integrated Waitlist System: Ensure the waitlist feature accommodates the semester model, maintaining its functionality across semester changes for courses with high demand.

Demo of application

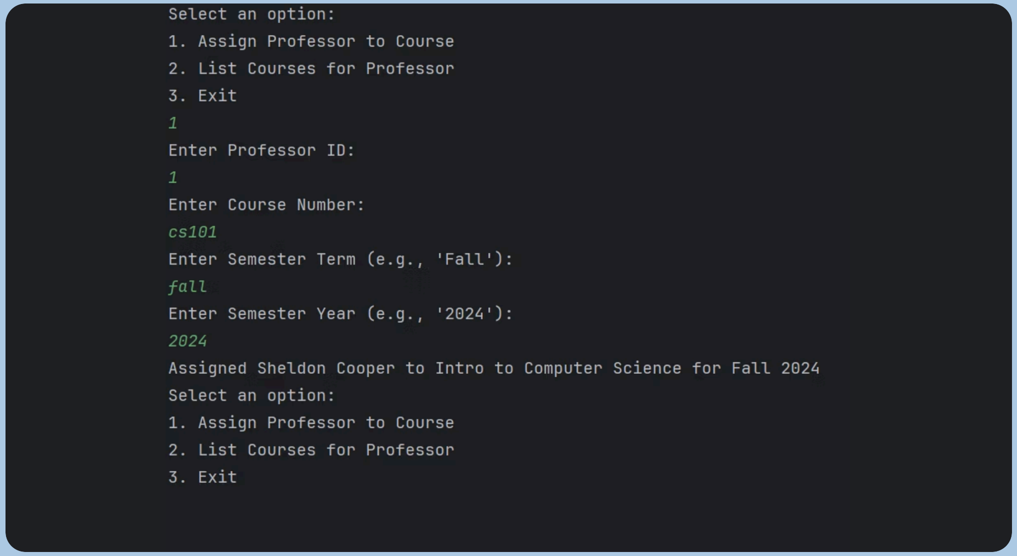




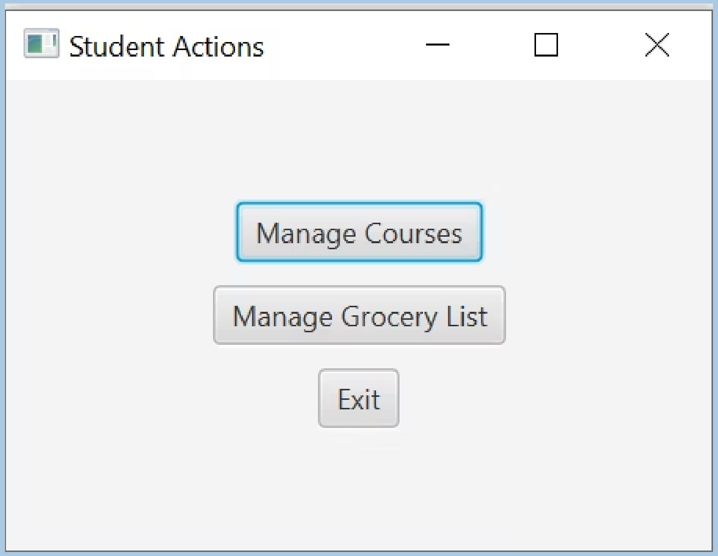
Enter your name and select your role:

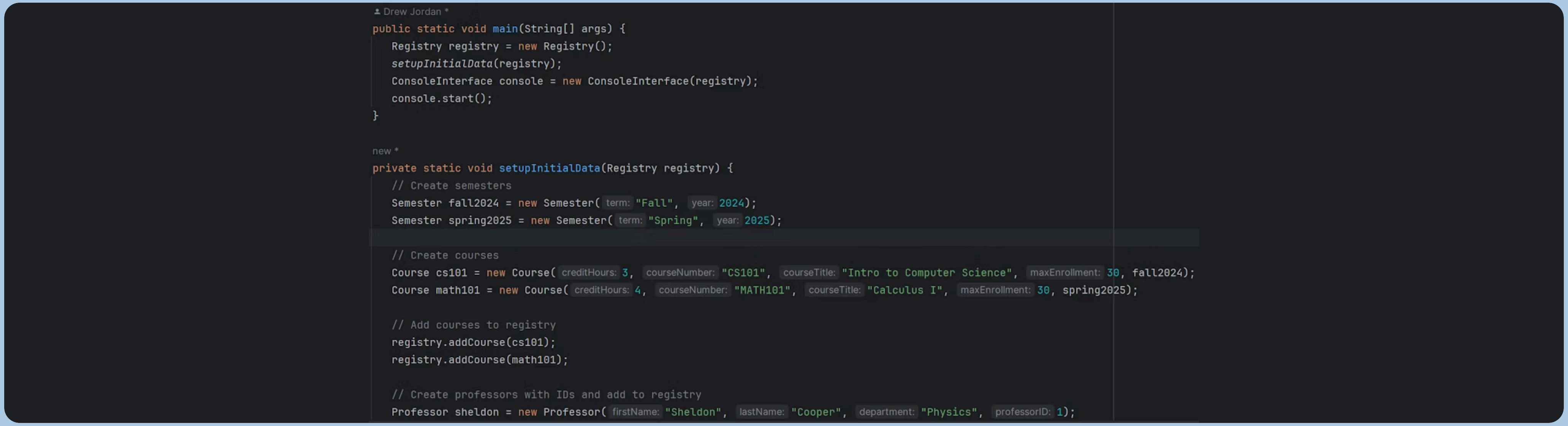
William Honig

Student [✓ Professor]



```
Select an option:
1. Assign Professor to Course
2. List Courses for Professor
3. Exit
1
Enter Professor ID:
1
Enter Course Number:
cs101
Enter Semester Term (e.g., 'Fall'):
fall
Enter Semester Year (e.g., '2024'):
2024
Assigned Sheldon Cooper to Intro to Computer Science for Fall 2024
Select an option:
1. Assign Professor to Course
2. List Courses for Professor
3. Exit
```

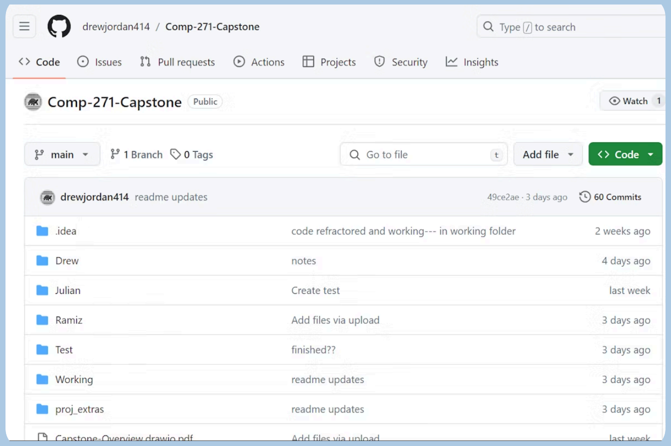




```
Drew Jordan *
public static void main(String[] args) {
    Registry registry = new Registry();
    setupInitialData(registry);
    ConsoleInterface console = new ConsoleInterface(registry);
    console.start();
}

new *
private static void setupInitialData(Registry registry) {
    // Create semesters
    Semester fall2024 = new Semester( term: "Fall",  year: 2024);
    Semester spring2025 = new Semester( term: "Spring",  year: 2025);

    // Create courses
    Course cs101 = new Course( creditHours: 3,  courseNumber: "CS101",  courseTitle: "Intro to Computer Science",  maxEnrollment: 30, fall2024);
    Course math101 = new Course( creditHours: 4,  courseNumber: "MATH101",  courseTitle: "Calculus I",  maxEnrollment: 30, spring2025);

    // Add courses to registry
    registry.addCourse(cs101);
    registry.addCourse(math101);

    // Create professors with IDs and add to registry
    Professor sheldon = new Professor( firstName: "Sheldon",  lastName: "Cooper",  department: "Physics",  professorID: 1);
```

Design approach and challenges

What did we learn to do more of or differently when making OOP apps in the future?



drewjordan414 / Comp-271-Capstone

Type / to search

Code · Issues · Pull requests · Actions · Projects · Security · Insights

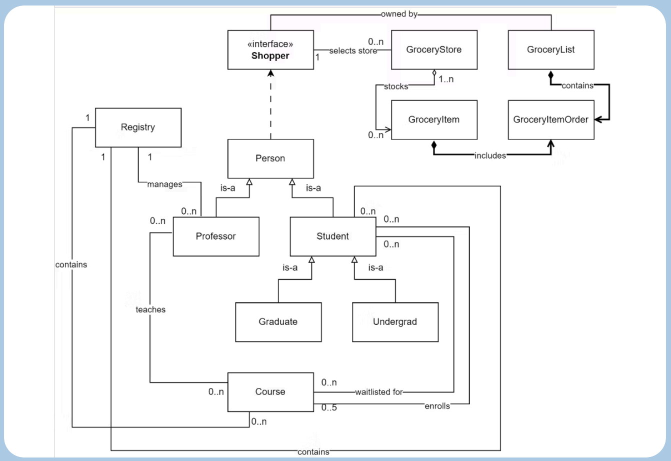
Comp-271-Capstone Public

Watch 1

main · 1 Branch · 0 Tags · Go to file · t · Add file · Code

drewjordan414 readme updates · 49ce2ae · 3 days ago · 60 Commits

.idea	code refractored and working--- in working folder	2 weeks ago
Drew	notes	4 days ago
Julian	Create test	last week
Ramiz	Add files via upload	3 days ago
Test	finished??	3 days ago
Working	readme updates	3 days ago
proj_extras	readme updates	3 days ago
Capstone-Overview.drawio.pdf	Add files via upload	last week



Design Approach:

While difficult to understand what exactly the final outcome would be, we all understood where to begin. Looking at each other's code the first time it wasn't too hard to understand some relationships between classes. However we made the mistake of not starting with an Overview UML, which in hindsight might've saved us a little more time. We were quick to realize that an Overview UML was necessary before we start touching the code. Once the UML was made the everyone in the group was on the same page and knew exactly how to make the right connections.

Significance of repository :

Use of GitHub streamlined our integration and modification process and enhanced our team collaboration and code management. Helped easy edits that everyone could view right away.

Challenges:

Understanding the associations between classes and how they were going to interact with one another, given the polarity of Registering for a course and going to the Grocery Store.

Code Review

Over the span of three TSPI inspection meetings, our team exhibited a remarkable progression. Collectively, we began with a total of 42 major and 26 minor defects, and through rigorous code reviews and refinements, we reduced this number to 16 major and 8 minor defects by the final meeting. Our overall codebase grew more robust, from 900 LOC (Lines of Code) to 1,160 LOC

TSPi Inspection Reports

TSPi Inspection Meeting one	Defects (Major | Minor)	Preparation data(Size | Time)
Drew Jordan	12 | 7	320 LOC | 15hrs
Ramiz Hameed	14 | 9	300 LOC | 12hrs
Julian Solis	16 | 10	280 LOC | 11hrs
TSPi Inspection Meeting Two	**Defects (Major | Minor)**	**Preparation data(Size | Time)**
Drew Jordan	7 | 5	350 LOC | 14hrs
Ramiz Hameed	9 | 6	340 LOC | 13hrs
Julian Solis	10 | 7	320 LOC | 13hrs
TSPi Inspection Final Meeting	**Defects (Major | Minor)**	**Preparation data(Size | Time)**
Drew Jordan	4 | 2	420 LOC | 15hrs
Ramiz Hameed	5 | 3	380 LOC | 14hrs
Julian Solis	7 | 5	360 LOC | 13hrs

Meetings

Drew:

- **Meeting #1.**
  - Need to separate GroceryList from Person class.
  - Organized project materials in GitHub for improved version control.
- **Meeting #2:**
  - Coding for extension class.
  - Semester class development completed on April 11th.

Julian:

- **Meeting #1.**
  - Refine Course class to remove redundancies.
  - Postponed Course class modifications until Person class updates finalized.
  - Proposed creating Main class for UI testing & integration with other classes.
- **Meeting #2:**
  - Preparing presentation.
  - Code compiled successfully, discussed extension, presentation logistics, and UML design.

Ramiz:

- **Meeting #1.**
  - Identified the need to craft subclasses for Professors, Undergrads, and Graduates.
  - Uploaded prior work on the Person class to GitHub for centralized access.
- **Meeting #2:**
  - Engaged in the development of both detailed and overview UML diagrams.
  - Noted the completion of the extension by Drew Jordan on April 12th, with the remainder of tasks being fine-tuning and clarifications of OOP principles.
  - Acknowledged the nearly finished state of the overview UML; planning to resolve outstanding confusions in an upcoming weekend meeting.
- **Meeting #3:**
  - Set a definitive deadline for the completion of the overview UML for April 13th, to ensure clarity on class interactions (Achievement documented on GitHub).

Conclusion

Integration Insights:

We learned that fusing three unique coding approaches demands more than expertise—it calls for robust version control systems like Git. Mastering such tools can turn a challenging code merge into a smooth and efficient process.

Interface Insight :

Developing an intuitive interface to combine complex functions—like academic scheduling with grocery list management—taught us the value of a solid initial design framework, such as UML diagrams, to guide the development process.

Structure Insight :

Our experience confirmed the vital role of strategic data structuring. It's the key to producing code that's not just functional but also readable and efficient, resulting in enhanced system performance.

Thank You for you time.