Final Report

Christian Cameron, James White, Naissa Charles, Nyameye Adom, Joel Armstrong

Requirements

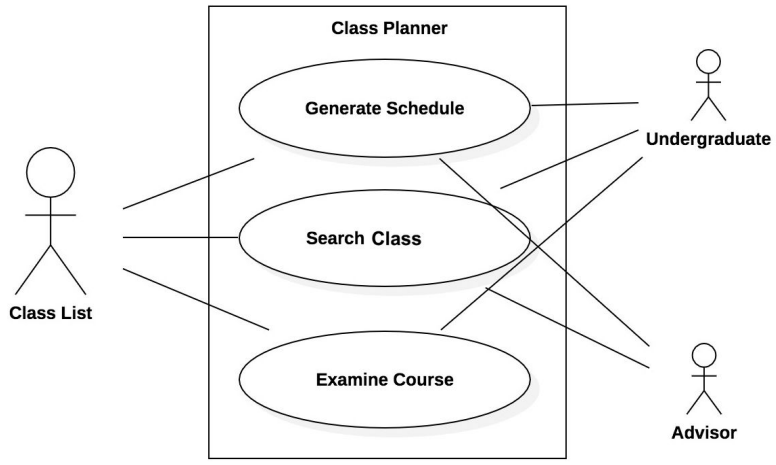
User Requirements/System Requirements

- Save a schedule by clicking a “Generate CSV” button.
    - Export a .csv file to represent a given schedule.
    - Each column in the csv file represents a semester.
- Examine each course in the schedule by clicking on a semester box to view class details like credits, description, prerequisites, and sections.
    - Create an intuitive UI to present the Details for each course.
    - Assemble a file storage system with the details of each course.
- Have a powerful search field which allows the user to find specific course offerings and details for those courses. The details include credits, description, prerequisites, and sections.
    - Search our full school year course map to find the desired course.
- Use the schedule generator to create multiple schedules to graduation that fit the user's interests, like specific majors/minors.
    - Have multiple major curriculum on hand to generate multiple schedules.

Non-functional/Domain Requirements

- Must quickly generate a class scheduling option upon changing/adding majors/minors.
- Check tags offered by each course and compare them to the Christ at The Core Curriculum Requirements.
- Must be flexible, accounting for students’ class preferences.

Use Cases



Use Case 1: Generate Schedule

Entity	**Boundary**	**Control**
Courses	Major and minor form	GenerateScheduleControl
	Course list (per semester)	

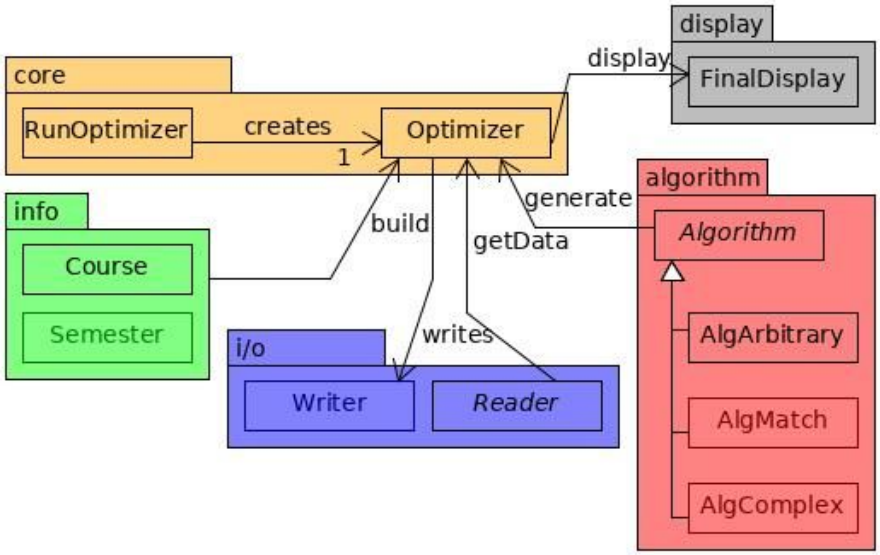
Use Case 2: Search Curriculum

Entity	**Boundary**	**Control**
Courses	Search button	SearchCourseControl
Requirement List	AddToRequirementListButton	

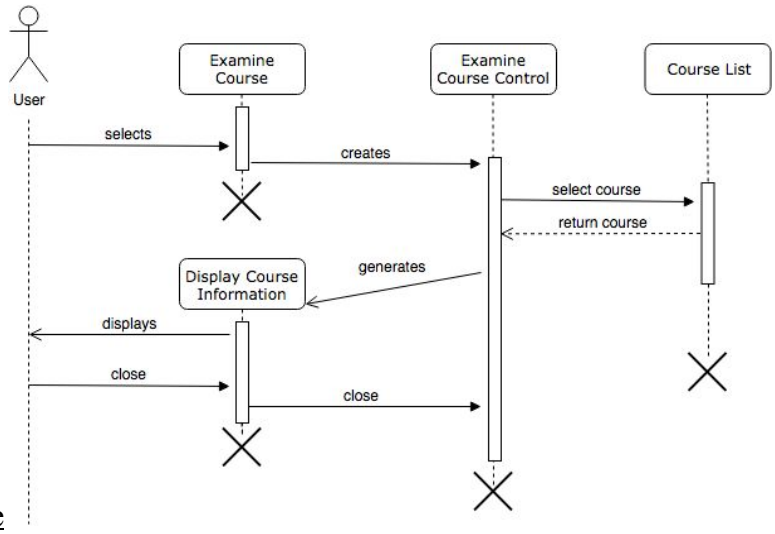
Use Case 3: Examine Course

Entity	**Boundary**	**Control**
Course (time, prof, description)	Advance search button	ExamineCourseControl
	DisplayCourseInfo	

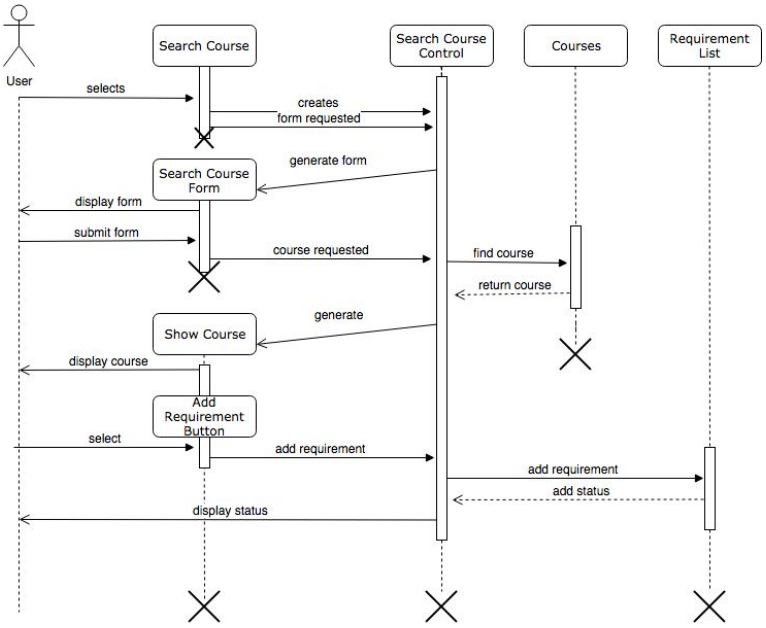
Class Diagram



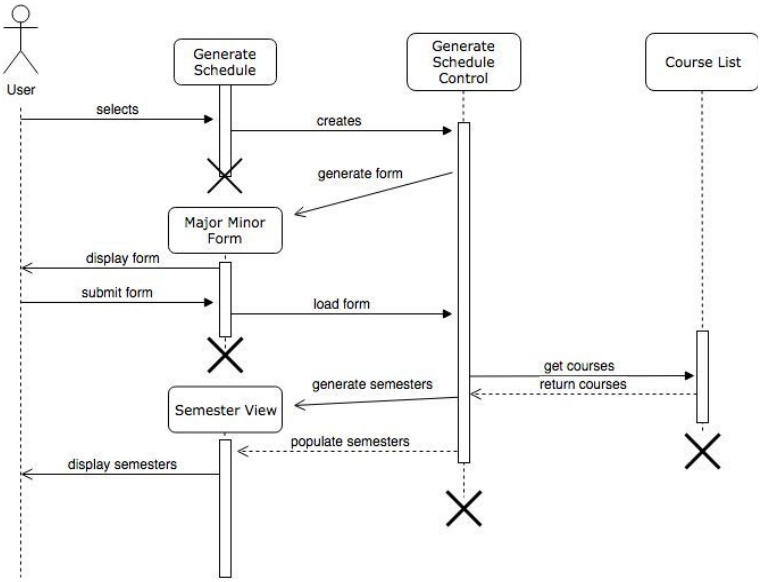
Sequence Diagrams



Examine Course

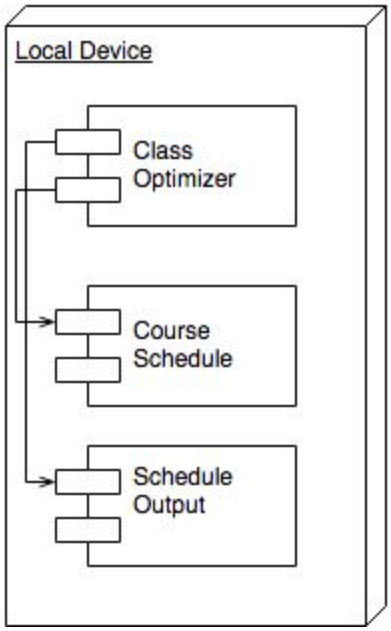


Search Course



Generate Schedule

Deployment Diagram



Written specifications

Generate Schedule: is the main service provided - it generates schedule plans that fulfill all of the user's majors/minors/interests.

1. A reader class will interpret a csv and create "course" objects.
2. An algorithm will generate a course schedule based on user constraints using the Course objects.
3. A writer object will put the course schedule in a format that can be saved for later and displayed on the screen.

Search Class: This use case allows a user to enter search criteria and find courses in Wheaton's curriculum that match those criteria.

1. An algorithm will retrieve the course object and display it in a meaningful way
2. Those courses will be displayed in a list for the user to interact with, like adding the course to the semester schedule

Examine Course: This use case allows a user to get more information about a course they selected.

1. Details about each course will be stored in a CourseDetails object.
2. A writer object will take the information from a CourseDetails object and send it to the display.

Subsystems

Five subsystems

- Core role: the main repository, responsible for managing connections and requests from the other subsystems.
- Info role: The course information, responsible for holding the course/lab/section hierarchy.

- I/O role: responsible for reading various files and writing the generated course schedule.
- Algorithm role: multiple algorithms, responsible for generating the course schedule.
- Display role: the graphical user interface.

Core subsystem interface

```
// Responsible for parsing the class schedule csv and mediating all interactions between subsystems.
// provides: buildSemesters, createSemester, write, generate

interface Core {
        Semester [] buildSemesters()
        Semester createSemester(String csvFile)
        Semester [] write()
        Semester [] generate()
} // Core
```

Info subsystem interface

```
// Info defines the semester, course, and class hierarchy.
// requires: ClassType
// provides: addSection, addLab, addCourse

interface Info {
        void addSection(ClassType t)
        void addLab(ClassType t)
        void addCourse(Course c)
} // Info
```

I/O subsystem interface

```
// Responsible for reading and writing different files
// provides: read, makeStream, moveLine, writeSchedule
interface I/O {
        Map<String,String> read()
        Scanner makeStream(String fileName)
        void writeSchedule(Semester[] schedule)
} // I/O
```

Algorithm subsystem interface

```
// Defines algorithms for generating the course schedule
// requires: algorithm interface
// provides: build, distribute

interface Algorithm{
        Semester [] build(Dparam d)
        void distribute(Semester [] s)
} //Algorithm
```

Display subsystem interface

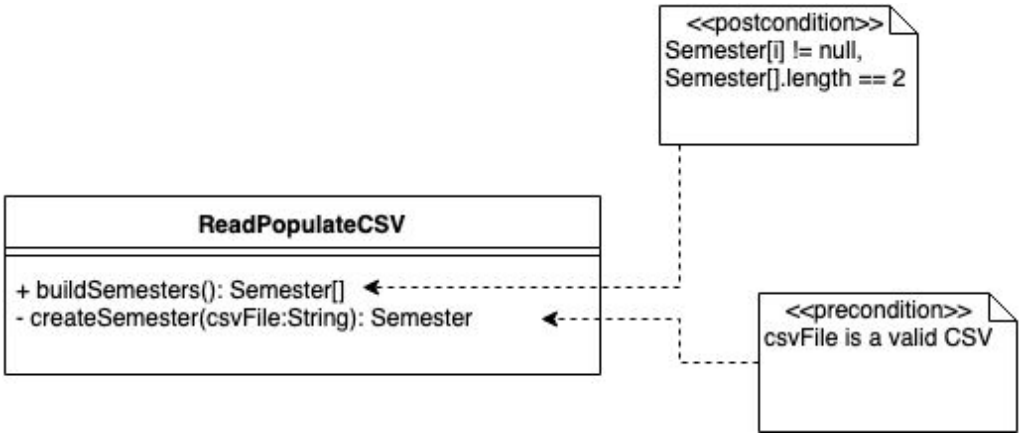
```
// Class that generates and manipulates GUI
// provides: displaySchedule, parseSemester,update,
// getParameters

interface Display {
        Void displaySchedule(semester[] s)
        String parseSemester(semester s)
        Void update(semester[] s)
} //Display
```

Data Storage

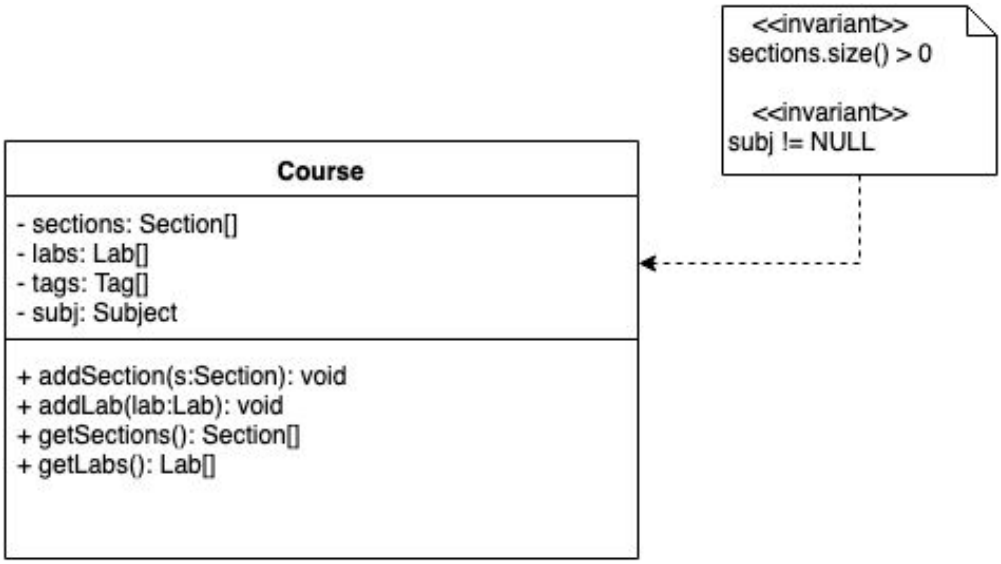
- input/
  - course_details/
    - prereqs.txt (list of prerequisites for every course)
    - descriptions.txt (list of descriptions for every course)
  - programs/
    - csci-major.pr (the computer science major requirements)
    - econ-major.pr (the econ major requirements)
    - math-major.pr (the math major requirements)
    - gen-ed.pr (the general education requirements)
  - schedules/
    - fall-2018.csv (the fall 2018 course schedule)
    - spring-2019.csv (the spring 2019 course schedule)

Interface Specification

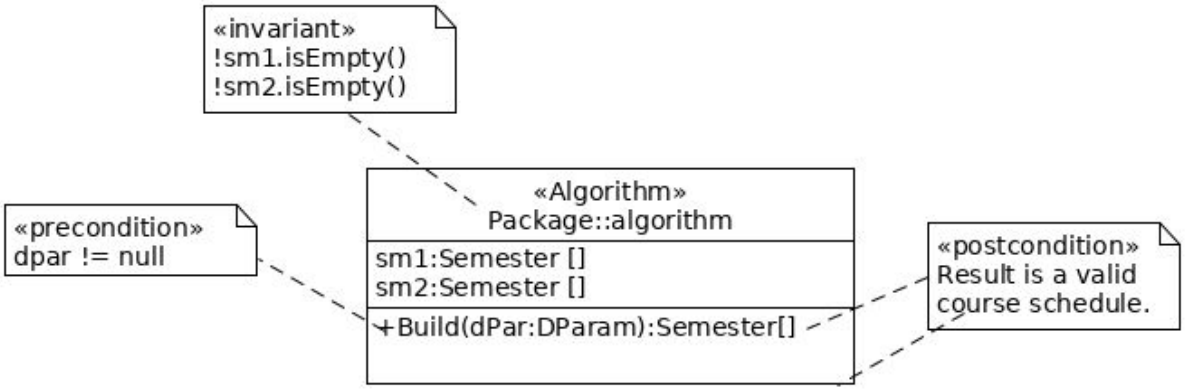
ReadPopulateCSV



Course



Algorithm

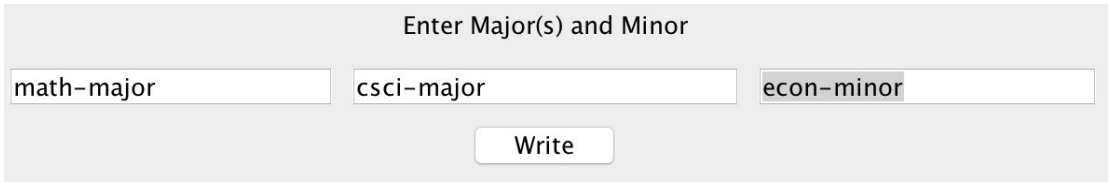


Subsystem Functionality

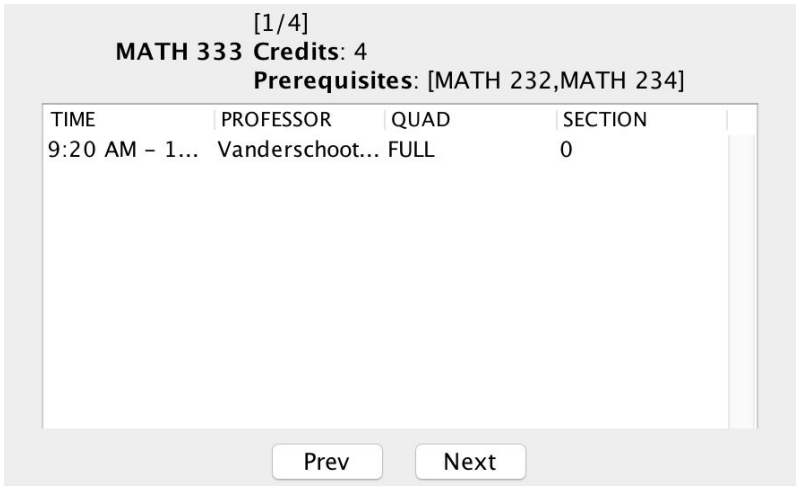
Five subsystems- core, info, I/O, algorithm, and display.

Semester 1	Semester 2	Semester 3	Semester 4
ECON 211 MATH 234 CSCI 245 MATH 131	ENGW 103 MATH 231 ECON 212 MATH 494 CSCI 351	PHIL 101 ECON 375 MATH 232 BEC 321	PHYS 231 MATH 331 BIOL 201 ART 101 CSCI 243
Semester 5	**Semester 6**	**Semester 7**	**Semester 8**
MATH 301 PHYS 221 MUCS 101 CSCI 235 MATH 351	MATH 333 COMM 101 ECON 345 ECON 325	MATH 245	

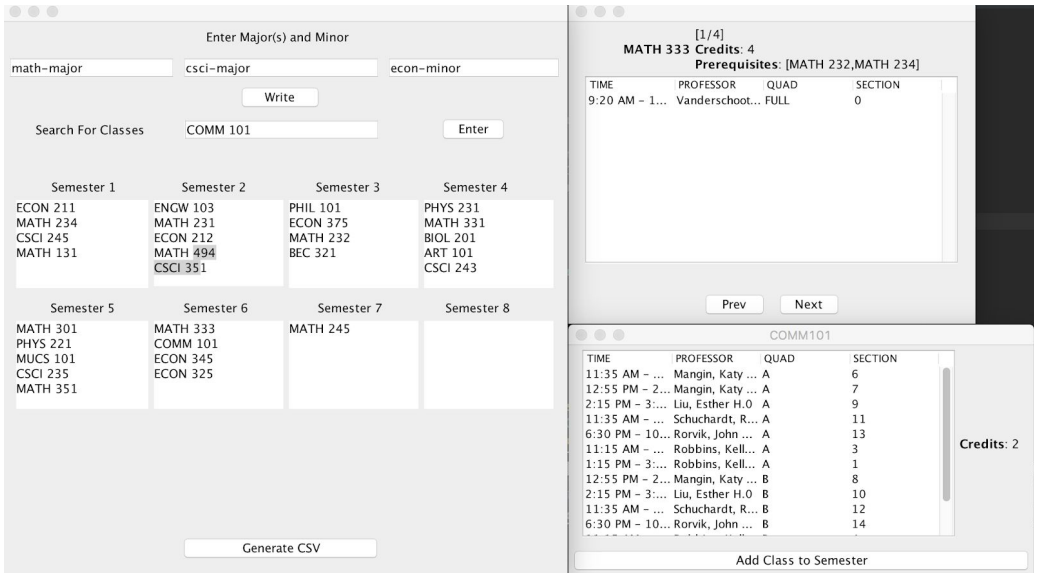
Core and algorithm



I/O



Info



Display