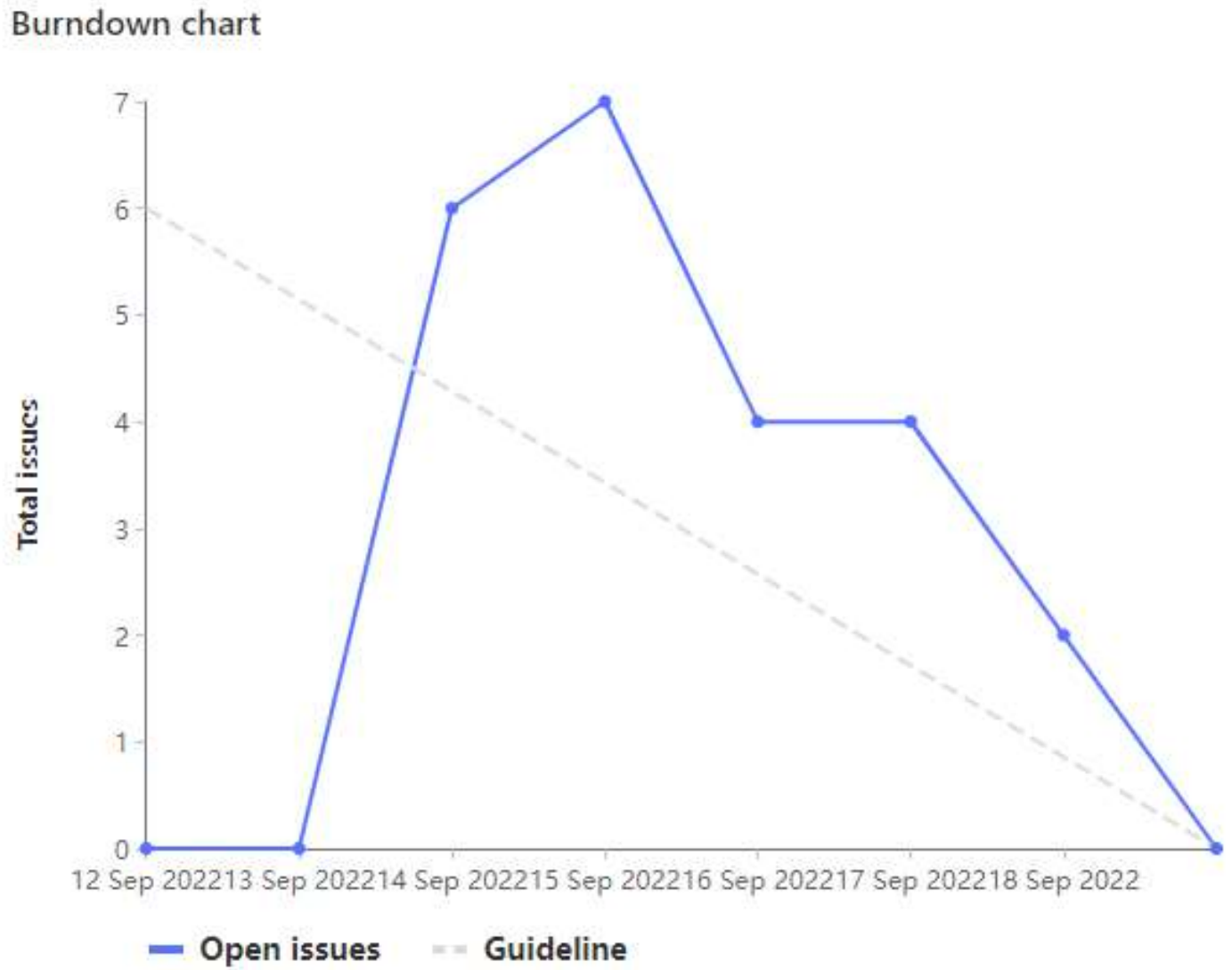
CIS*3760 SPRINT 1

Group 203

Burndown


Burndown chart
Total issues
7
6
5
4
3
2
1
0
12 Sep 2022
13 Sep 2022
14 Sep 2022
15 Sep 2022
16 Sep 2022
17 Sep 2022
18 Sep 2022
Open issues
Guideline

Outcomes

Overall, Sprint 1 was successful. We were able to add several additional features to our programs. We are happy with the outcome.

We used this week to help bring everyone up to speed on Python

For Sprint 2, we plan to have everyone equally on development

A friction point this week was deciding why and when to meet to discuss new ideas/issues/blockers

Going forward, we have decided to run brief daily stand-up meetings. This will allow each member to voice ideas/issues/blockers

The team-lead will act as the scrum-master

Kyler Swanson

TASKS	Weights
Setting up a development environment with the latest stable release of python 3 (3.10.7) on local machine.	1
Introductory meeting	2
Sprint 1 planning	3
Add functionality to search by course code.	4
Add functionality to search by code name	3
Unit testing suite	3
Sprint 1 Catch-up Meeting	1
Create a README file with overall documentation.	2
Powerpoint presentation	2
Discord server setup	1
Total Weights	22 (11 hours worked)

Kyler Swanson

```python
class CourseSearchTests(unittest.TestCase):
    def setUp(self):
        self.coursesearch = CourseSearch('tests/data/example.json', 'tests/data/example_mapping.json')

    def test_load_courses(self):
        search = CourseSearch('tests/data/example.json', 'tests/data/example_mapping.json')

        self.assertEqual(search.courses, courses_fixture)
        self.assertEqual(search.course_mapping, mapping_fixture)

    def test_search_course_code(self):
        result = self.coursesearch.search_course('CIS3760')

        self.assertEqual(result, course_fixture)

    def test_search_course_code_invalid(self):
        result = self.coursesearch.search_course('CIS1337')

        self.assertEqual(result, None)

    def test_search_course_asterisks(self):
        result = self.coursesearch.search_course('CIS*3760')

        self.assertEqual(result, course_fixture)

    def test_search_course_name(self):
        result = self.coursesearch.search_course('Software Engineering')

        self.assertEqual(result, course_fixture)

    def test_search_course_name_invalid(self):
        result = self.coursesearch.search_course('Software Design')

        self.assertEqual(result, None)

if __name__ == "__main__":
    unittest.main()
```

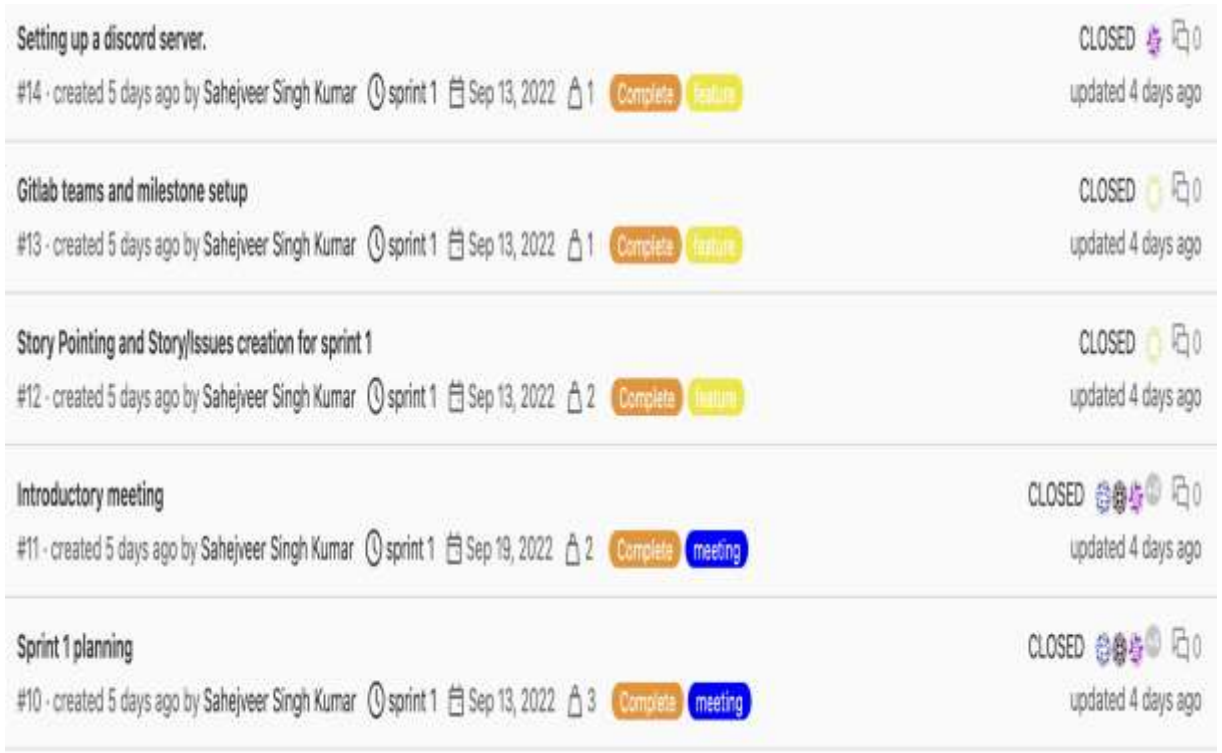
```python
class HTMLParserTests(unittest.TestCase):
    def test_parse_courses(self):
        parser = HTMLCourseParser()

        with open('tests/data/courses.html', 'r') as f:
            for line in f:
                parser.feed(line)

        parser.close()
        courses = parser.get_course_dict()

        file = open('tests/data/results.json')
        courses_fixture = json.load(file)
        file.close()
        self.assertEqual(courses, courses_fixture)

    def test_parse_mapping(self):
        parser = HTMLCourseParser()

        with open('tests/data/courses.html', 'r') as f:
            for line in f:
                parser.feed(line)

        parser.close()
        courses = parser.get_course_mapping()

        file = open('tests/data/course_mapping.json')
        mapping_fixture = json.load(file)
        file.close()
        self.assertEqual(courses, mapping_fixture)

if __name__ == "__main__":
    unittest.main()
```

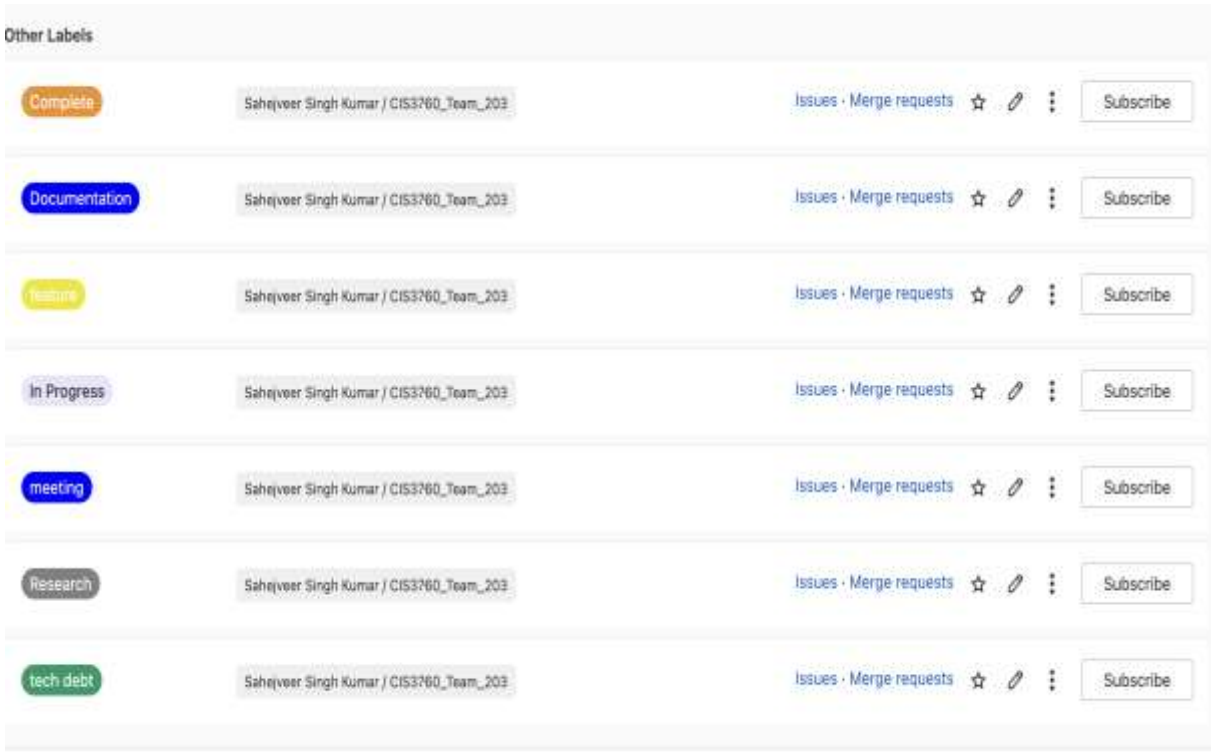
Sahejveer Singh Kumar

TASKS	Weights
Setting up a development environment with the latest stable release of python 3 (3.10.7) on local machine.	1
Setting up the team on Gitlab and creating a milestone/iteration for the sprint run.	1
Introductory meeting	2
Sprint 1 planning	3
Weighting the issues and Adding plus Assigning issues on Gitlab for sprint 1.	2
Add functionality to search by course code.	4
Add functionality to search by code name	3
Sprint 1 Catch-up Meeting	1
Assigning labels and closing issues plus gitlab maintenance.	1
Create a README file with overall documentation.	2
Powerpoint presentation	3
Total Weights	23 (11.5 hours worked this week)

Sahejveer Singh Kumar

Gitlab Issues and Labels





Sahejveer Singh Kumar

Search Code Functionality

```
if coursekey not in self.json_dict:
    self.json_dict[coursekey] = []

self.json_dict[coursekey].append(self.section_dict.copy()) # section dictio

self.course_mappings[self.section_dict['courseName'].upper()] = coursekey

self.section_dict = {
    'meeting': []
}

self.current_data = ''
```

```
exit_cond = 0
while exit_cond == 0:
    course = search.search_course(search_term)

    if course:
        course_code = course[0]['department'] + course[0]['courseCode']
        print("Course: " + course_code + "\n")

        # print each section
        for i in range(len(course)):
            section = course[i]

            print("Section: ", section['section'])

            # print each key we want to display from SECTION_INFO
            for key, title in SECTION_INFO.items():
                # ensure that the key exists in the course dictionary
                if key in section and section[key]:
                    print("" + title + ": ", section[key])
```

Tristan Kerec

TASKS	Weights
Setting up a development environment with the latest stable release of python 3 (3.10.7) on local machine.	1
Research and create documentation on Visual Basic for Applications (VBA) for Office.	4
Introductory meeting	2
Sprint 1 planning	3
Add functionality to search by course code.	4
Add functionality to search by code name	3
Sprint 1 Catch-up Meeting	1
Create a README file with overall documentation.	2
Powerpoint presentation	2
Total Weights	22 (11 hours worked)

Microsoft Visual Basic for Applications for Office

(Worked on by Tristan, Andrew)

- Visual Basic for Applications for Office (VBA) is a programming language that allows the user to write scripts that extend, customize, and expand several features of Microsoft Office applications.
- VBA is an event-driven programming language, which means that executions within the program are determined by events
- The applications which utilize VBA are the following:
  - Access, Excel, Office for Mac, Outlook, PowerPoint, Project, Publisher, Visio, Word

Mackenzie Kean

TASKS	Weights
Setting up a development environment with the latest stable release of python 3 (3.10.7) on local machine.	1
Introductory meeting	2
Sprint 1 planning	3
Writing the script to Parse HTML File to JSON	4
Documentation of Parsing script and KT to search development team.	3
Sprint 1 Catch-up Meeting	1
Creating second JSON output on parser file and minor bug fixes	2
Create a README file with overall documentation.	2
Powerpoint presentation	2
Total Weights	20 (10 hours worked)

Mackenzie Kean

```
self.section_dict = {
    'meeting': []
}

self.meeting_dict = {}

# flags and other variables
self.current_data = ''
self.meeting_info_count = 0

self.json_dict = {}
self.course_mappings = {}
```

```
def handle_data(self, data):
    if self.current_data == 'term':
        self.section_dict['term'] = data
        self.current_data = ''

    elif self.current_data == 'status':
        self.section_dict['status'] = data
        self.current_data = ''

    elif self.current_data == 'section_title':
        tokens = data.replace('*', '(').replace(')', '(')
        tokens = tokens.split('(')

        self.section_dict['department'] = tokens[0]
        self.section_dict['courseCode'] = tokens[1]
        self.section_dict['section'] = tokens[2]
        self.section_dict['num'] = tokens[3]
        self.section_dict['courseName'] = tokens[4].strip()
        self.current_data = ''
```

```
data_type_identifiers = {
    'term' : 'WSS_COURSE_SECTIONS_',
    'status' : 'LIST_VAR1_',
    'location' : 'SEC_LOCATION_',
    'meeting' : 'meet',
    'prof' : 'SEC_FACULTY_INFO_',
    'available_capacity' : 'LIST_VAR5_',
    'credits' : 'SEC_MIN_CRED_',
    'section_title' : 'SEC_SHORT_TITLE',
    'level' : 'SEC_ACAD_LEVEL_'
}
```

Alireza Sharif

TASKS	Weights
Setting up a development environment with the latest stable release of python 3 (3.10.7) on local machine.	1
Bringing up to speed meeting	2
Writing the script to parse HTML file to JSON	4
Add functionality to search by course code.	4
Add functionality to search by code name	3
Sprint 1 Catch-up Meeting	1
Creating second JSON output on parser file and minor bug fixes	2
Create a README file with overall documentation.	2
Powerpoint presentation	2
Total Weights	21 (10.5 hours worked this week)

Andrew Heft

Task	Weight
Setting up a development environment with the latest stable release of python 3 (3.10.7) on local machine.	1
Introductory meeting	2
Sprint 1 planning	3
HTML Documentation	3
Visual Basic for Applications for Office Research	4
Sprint 1 Catch-up Meeting	1
Create a README file with overall documentation.	2
PowerPoint creation, presentation	2
Total	18

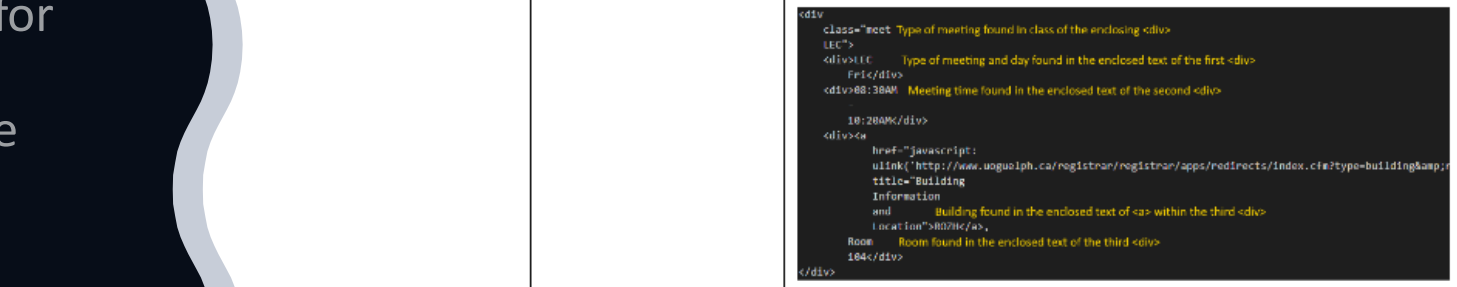
Robert Stegmann

Task	Weight
Setting up a development environment with the latest stable release of python 3 (3.10.7) on local machine.	1
Introductory meeting	2
Sprint 1 planning	3
HTML Documentation	3
Documentation of Parsing script and KT to search development team	3
Sprint 1 Catch-up Meeting	1
Create a README file with overall documentation.	2
PowerPoint creation, presentation	2
Add the ability to call coursesearch with an html file	3
Total Weighting	20 (10 hours)

HTML Documentation

- Along with analyzing which information needed to be extracted from the HTML file, I create a summarized table for where to find all the information in the HTML file

The information can be found in the tag of each <td>

Term	Found within the value of <label>
Status	Found within the value of <label>
Section Name and Title	Found in the enclosed text of <a> in the format [Department]*[Course Number]*[Section Number] ([Unknown Number]) [Course Name]
Location	Found in the enclosed text of <p> and the value of <label>
Meeting Information	A string containing all the information is found in the value of <label> Parsed information can be found within subsequent after label. The amount of varies. 
Faculty	Found within the value of <label>
Capacity	Found within the value of <label>. It is in the format “[Registered Students] / [Total Capacity]“ To obtain the value of each separately, parsing will need to be done.
Credits	Found within the value of <label>
SEC_CUES	This is an invisible cell with no discernible purpose or information
Academic Level	Found within the value of <label>

Documentation of Parsing script and KT to search development team.

The original documentation for the JSON formatting

```
JSON format for a course section

{"term":string,"open":bool,"department":string,"courseCode":integer,"section":string,"num":string,"courseName":string,"location":string,"meeting":["{meetingJSON}

term: The term the course is offered in, e.g. "Fall 2022"
open: Whether the course is open (true) or closed (false)
department: The department the course belongs to e.g. "CIS"
courseCode: The course code e.g. 3760
section: The section number e.g. "0102"
num: A number of unknown importance. Included just in case, e.g. "7264"
courseName: Name of the course e.g. "Software Engineering"
location: The campus the course is on, e.g. "Guelph"
meeting: An array of meeting JSONs as a string. See below
faculty: The instructor for the course e.g. "G. Klotz"
registeredStudents: Number of students registered for a course, e.g. 0
capacity: How many students can register for a course e.g. 32
credits: How many credits the course is worth e.g. 0.75
academicLevel: Course level e.g. "Undergraduate"

--------------------------------------------------------------------------------

JSON format for meeting times

{"type":string,"day":string,"startTime":string,"endTime":string,"building":string,"room":string}

type: Whether the meeting is a lecture, lab, seminar or exam e.g. "LEC"
day: Day(s) of the week the meeting is held e.g. "Tues, Thur"
startTime: Starting time for the meeting e.g "08:30AM"
endTime: Ending time for the meeting e.g "09:50AM"
building: Building in which the meeting is held e.g. "RICH"
room: Room where the meeting is held e.g. "2529"

--------------------------------------------------------------------------------

Example of the JSON format for course sections

{"term":"Fall 2022","open":false,"department":"CIS","courseCode":3760,"section":"0102","num":"7264","courseName":"Software Engineering","location":"Guelph","meet

Example of JSON format for meetings

{"type":"LEC","day":"Tues, Thur","startTime":"08:30AM","endTime":"09:50AM","building":"RICH","room":"2529"}

{"type":"LAB","day":"Mon","startTime":"01:30PM","endTime":"04:20PM","building":"THRN","room":"2420"}
```

Documentation found in README.md reflecting the latest format

JSON Formating

JSON format for a course section

{ "meeting": [], "term": "Fall 2022", "status": "Closed", "department": "CIS", "courseCode": "3760", "section": "0101 ", "num": "7263", "courseName": "Software Engineering", "location": "Guelph", "faculty": "G. Klotz", "available": "0", "capacity": "32", "credits": "0.75", "academicLevel": "Undergraduate" }

meeting: An array of meeting JSONs as a string. See below term: The term the course is offered in, e.g. "Fall 2022" status: Whether the course is open or closed, e.g. "Closed" department: The department the course belongs to e.g. "CIS" courseCode: The course code e.g. "3760" section: The section number e.g. "0102" num: A number of unknown importance. Included just in case, e.g. "7264" courseName: Name of the course e.g. "Software Engineering" location: The campus the course is on, e.g. "Guelph" faculty: The instructor for the course e.g. "G. Klotz" registeredStudents: Number of students registered for a course, e.g. "0" capacity: How many students can register for a course e.g. "32" credits: How many credits the course is worth e.g. "0.75" academicLevel: Course level e.g. "Undergraduate"

JSON format for meeting times

{ "meeting_type": "LEC", "meeting_day": "Tues,Thur", "start_time": "08:30AM", "end_time": "09:50AM", "building": "RICH", "room": "Room 2529" }

meeting_type: Whether the meeting is a lecture, lab, seminar or exam e.g. "LEC" meeting_day: Day(s) of the week the meeting is held e.g. "Tues,Thur" startTime: Starting time for the meeting e.g "08:30AM" endTime: Ending time for the meeting e.g "09:50AM" building: Building in which the meeting is held e.g. "RICH" room: Room where the meeting is held e.g. "Room 2529"

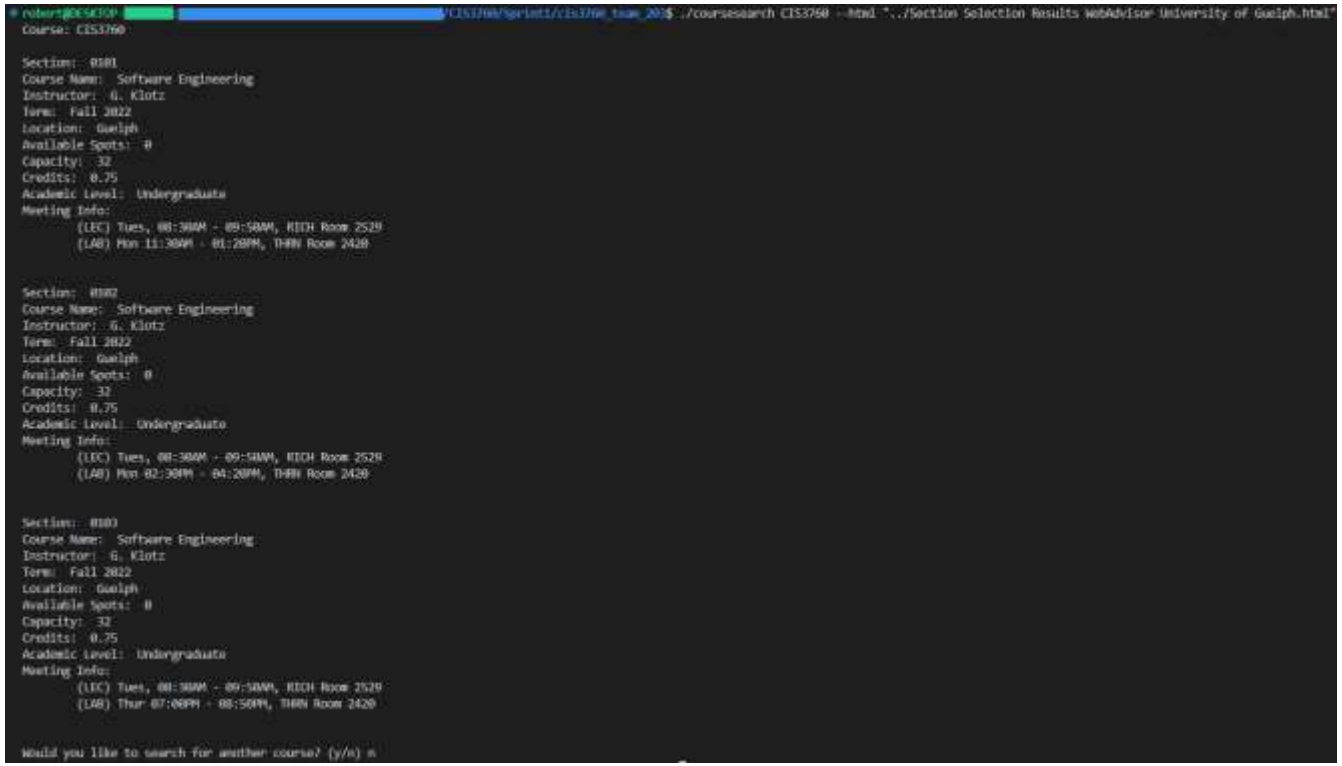
Example of JSON

{ "meeting": [{ "meeting_type": "LEC", "meeting_day": "Tues,Thur", "start_time": "08:30AM", "end_time": "09:50AM", "building": "RICH", "room": "Room 2529" }, { "meeting_type": "LAB", "meeting_day": "Mon", "start_time": "11:30AM", "end_time": "01:20PM", "building": "THRN", "room": "Room 2420" }], "term": "Fall 2022", "status": "Closed", "department": "CIS", "courseCode": "3760", "section": "0101 ", "num": "7263", "courseName": "Software Engineering", "location": "Guelph", "faculty": "G. Klotz", "available": "0", "capacity": "32", "credits": "0.75", "academicLevel": "Undergraduate" }

Add the ability to call coursesearch with an HTML file

Added the ability for coursesearch to be ran with a HTML file instead of a JSON file.

```
elif len(sys.argv) == 4 and sys.argv[2] == '--html':
    # run the parser to get json from html
    try:
        subprocess.run(["./htmlparser", sys.argv[3]], check=True)
    except:
        # parser failed
        sys.exit(1)
```



Create a README file with overall documentation

Created the README file including details on the usage of the programs

Usage

htmlparser

`htmlparser <html_file>`

`<html_file>` is the name of the `.html` file to be parsed.

`htmlparser` will create two `.json` files in a directory called `data`: `results.json` and `course_mapping.json`.

NOTE: any additional formatting applied to the HTML file to be parsed has the potential to cause unexpected behaviour and break the parser. Ensure the HTML file is as it was downloaded from the web without unnessecarry newline characters.

Notes

`results.json` may be renamed and used by `coursesearch` by using the `--in` tag.

DO NOT rename or move `course_mapping.json`. See the notes section for the coursesearch for more details

coursesearch

`coursesearch <course_code|course_name> [--in input_file.json|--html input_file.html] coursesearch [-h|--help]`

`<course_code|course_name>` is the course code or the name of the course. `input_file.json` is the name of the `.json` file to be used. `input_file.html` is the name of the `.html` file to be used.

Using the `-h` or `--help` tag will display a help message.

After the initial search, one may make additional searches or exit the program.

Notes

By default, `coursesearch` will use `/data/results.json` and `/data/course_mapping.json`. Using the `--in` tag will allow you to specify a file instead of `/data/results.json`

`coursesearch` **requires** there to be `/data/course_mapping.json`. **DO NOT RENAME OR MOVE THIS FILE.** Using `htmlparser` or a correct usage of the `--html` tag will recreate this file.

Usage of `--html` tag **requires** the `htmlparser` file to be in the same directory as `coursesearch`.

Usage of the `--html` is equivalent of running `htmlparser` prior to `coursesearch`. As a result, the initial search will take longer than normal.

Demo