

Small note for grading, peer reviews, etc.: the major changes from the draft proposal are:

- *Removed many wireframes for different pieces of profile UI since the profile is less important to the core functionality; replaced the remainder with Balsamiq wireframes (previously Appendix A contained scans of hand-drawn draft UI)*
- *Added a few comments about course metadata where relevant, and included the metadata format as Appendix B (by no means mandatory reading!)*
- *Added an abstract – I saw this in a number of the exemplary proposals and I felt it would add to this proposal as well*

Abstract

Advising in the OMSCS program currently relies on two unofficial pathways: the Unofficial Course Review Spreadsheet and individual inquiries on the Google+ group. This proposal outlines a web application that will make the advising process *scalable* and *repeatable* by organizing course reviews and suggesting courses to students. This application may also serve as a framework for later study of factors affecting student success in and affinity for particular courses.

Motivation

Because of its scale and its internet-mediated nature, the OMSCS program presents a number of unique challenges in the field of educational technology.

Thus far, it has successfully addressed many of these through two patterns of adaptation. One is a **shift from centralized operations to peer support structures**. For example, the program now provides many of its own TAs in classes like Knowledge-Based Artificial Intelligence (KBAI), Computational Complexity and Algorithms (CCA), and even Educational Technology (EdTech) itself – this is possible because a class of two to four hundred students is virtually guaranteed to have at least 10-20 very good ones who are very interested in the subject and can be brought on to TA the next semester. The other is a **shift from following traditions to making decisions based on data gathered from students**. For example, Dr. David Joyner’s controlled experiments on grader efficacy (performed in conjunction with the summer offering of KBAI) attempt to isolate factors governing the quality and speed of feedback, so as to develop grading “best practices” which could be applied more generally in the program.

That said, there are some pieces of the program that have not finished this transformation (and thus lag far behind in scalability). One of these is the practice

of academic advising as it applies to the program. Dr. Charles Isbell, a Senior Associate Dean and one of the professors most heavily involved in the program thus far, has cited this as a key risk. It's simply not feasible to support each and every student with a professional advisor (the way a more expensive "executive Master's" program might) or a faculty advisor (the way a smaller research-oriented or PhD feeder program might) – nor will it ever be.

Advising in OMSCS appears to have begun the transition towards peer-reliant, high-data solutions. Many OMSCS students currently read the unofficial course review spreadsheet and ask their peers or the Google+ group for advice on which courses they ought to take next. To a casual observer these solutions might seem to leverage the program's size fairly well, and indeed they are significantly more feasible than a traditional advising structure in the context of OMSCS. However, because a student is still limited to manually sifting through posts or reviews, the process becomes less efficient, not more, as the program scales up; this is costly to both advisors and advisees. Furthermore, the quality of advice a student receives varies quite a bit depending on when that student asks, who happens to read and respond, which details of academic background are included in the inquiry, and so forth.

With the recent departure of Mimi Haley, the program is back down to one academic advisor, with an advisor-advisee ratio of between 1:2000 and 1:3000 – the highest in the program's history. It's now more critical than ever to offer OMSCS students the planning information they need in an approachable, scalable and consistent format.

Vision

This project draws on two distinct traditions.

From the field of *academic advising* we learn that to be effective we must gather data on each individual student (including their goals and background / transcript info), keep abreast of overall student feedback on the available courses, offer suggestions for courses that match their preferences and allow them to complete their degree, and give them access to all the information they need to evaluate those suggestions or adapt the plan for themselves.

From the field of *recommendation systems* we learn that we must supplement the factors we know to be important (a student's specialization, for instance) with other factors we can infer as we go. We also inherit a massive repertoire of techniques that we can apply to make sense of the data we obtain.

In this first phase we will not have time to make great use of the latter (more on that under "Scope"). However, it's important to bear it in mind – the design of the system in this phase (particularly its data collection) is intended to keep all those options open for later building.

Advising

The initial goal of the project is to provide advising services (specifically academic advising – career or other advising is outside the scope of the project), and to provide them in such a way that their quality improves rather than degrading as the program grows larger. This breaks down into four major components:¹

- User profile & history: allows the student to input (at a minimum) their previously-completed courses, desired specialization, timetable for graduation, and study hours per week. Eventually, may serve as the basis for social features (sharing one's class schedule with others to plan study groups).
- Planner: considers the user's profile and suggests appropriate courses and enrollment dates; allows the user to save their plan and tweak it later, or start over from scratch if they want to make a major change (e.g., taking a term off or switching specializations).
- Course browser: allows the user to access additional course information that may help in evaluating how appropriate the planner's suggestions are for their unique interests and circumstances.
- Review system: allows the user to give feedback on courses they've completed, to benefit both the system (which will use some of this data, e.g. effort estimates, in recommendations) as well as other students (who will look at their predecessors' opinions when deciding whether to take a course).

Platform for research

The OMSCS program is one of the largest course-based CS MS programs in the country, and the average class offers more than 200 seats per term. Moreover, the current group of OMSCS students has shown a high degree of “civic-mindedness”, a willingness to go to some lengths to enhance the program experience for others. These factors combine to present a unique opportunity for data collection.

I have already identified some features that are likely predictors for how much a student will like a course and/or how much value they will derive from it. In the long run, though, these give us merely a starting point; application of machine learning techniques could yield a far richer set of factors to consider for each student. For example, no official prerequisites are currently established for any course in the OMSCS program; does this make sense, or are there “unofficial” prerequisites one needs in order to do well in certain courses? Can we infer that if you do well in one course requiring Java, you'll do well in another course requiring Java?

¹ Note: I took this breakdown from my own miniproposal. That in turn largely recycled the breakdown from my project readme at <https://github.com/Arkaaito/omscs-advisor>.

Conversely, can our data offer value to members of the OMSCS community beyond the student body? For instance, can we hope to advise professors on what factors make a course well-loved, or how many students are likely to enroll in their course for the upcoming term (and thus how many TAs they need)?

Social features

In recent years, advisors have begun to experiment with online communities of advisees. These communities are often built on course management platforms like Moodle and facilitate one-to-one or one-to-many communication. However, they rarely encourage interactions between peers. In addition, those rare communities that *do* encourage peer interactions are still largely formed around artifacts provided by the advisor (e.g., an advising syllabus) as opposed to material created by the students.

Because of the youth and size of the OMSCS program, its social landscape is still very much in flux. We've already outlined the reasons for pursuing a peer-driven overhaul of "traditional" advising functions; might the *advising community* benefit from a similar overhaul? Even very rudimentary social features, such as the ability to add "friends" and later see which friends are planning to take which classes when, might help students form smaller, more manageable study groups within larger classes.

Not only could these features benefit students directly; they would also enhance the advising and research functionality mentioned above. If a large number of students opted into the social elements, we could begin to approach such research questions as "How much does a student's connectedness change their likelihood of dropping out?" and "Do students prefer taking classes with 'high-profile' peers?" We could also consider the presence of known study buddies as a factor influencing a student's success in a course, and invite them to adjust their schedule accordingly.

Future collaboration

In the long term, the vision for omscs-advisor must evolve to more closely fit the needs of the OMSCS community. The clearest way to get there is by eventually open-sourcing the project or at least inviting code contributions from other students.

In order to attract and sustain a significant development community, I've made the decision to build the entire app on the Django web MVC framework. Django is a popular framework; it was one of the winners of my informal (and admittedly quite flawed) poll in the OMSCS Google+ group. In addition, because Django applications are written in Python, OMSCS students who participate in a Django project will be building useful skills that can be applied in many of the program's courses.

This presents some challenges of its own, as I have never previously written a web application with Django! However, I believe my background in using other web

frameworks (and recent work in Python) has prepared me to pick it up relatively quickly and with a minimum of fuss.

In the near term, the advisor will require periodic metadata updates to stay up to date as new courses are released, instructors changes, and other program data becomes available. For this reason, the metadata format has been standardized and published on GitHub so that other students (not just the developer) can submit corrections and updates.

The current plan is to accept these metadata submissions as pull requests through GitHub itself. (The reasoning is somewhat similar to the reasoning for using Django given above; GitHub is a popular tool, widely used in industry and in the OMSCS program, so students who are not already used to this workflow will probably benefit from the introduction.)

Implementation

Scope

All of the foregoing is nice, but because of the constrained duration of this course, it's necessary to pick and choose somewhat. The absolute core functionality of the advisor is the ability to recommend courses, so that's where we'll start. (Where possible, I have tailored the UI design to allow for future extension in the other two areas.)

Furthermore, building an intelligent recommender system is itself an open-ended project. In order to avoid going too far down the rabbit hole, I've limited myself to a planner that considers the user's objective, explicitly stated preferences. More advanced course ranking techniques, such as collaborative filtering, would be of limited utility with a small initial set of users. With the current user interface and data collection, these techniques can still be added once the user base (and thus the set of available data) has grown.

The trimmed version of the project includes only 12 webpages. For a slightly more detailed picture of this planned interface, see the requirements draft at <https://github.com/Arkaaito/omscs-advisor/blob/master/requirements.md> (excluding the items listed under 2.3, "Future Items"). The requirements document should also illuminate the proposed course metadata format, statistics, and other technical details. Some current wireframes are included in *Appendix A*; however, they are not final (nor are they very legible).

Schedule

The following is a proposed schedule for this phase of the omscs-advisor project.

Date	Theme	Deliverable(s)
10/3	Setup, ramp-up	Register omscs-advisor.com domain

		Set up EC2 instance running LAMPython + Django Complete Django tutorial Create landing page, login page, and account creation page Stuff code into omcs- advisor repository Update license terms Write ToC / privacy policy
10/10	Metadata	Encode metadata for specializations Encode metadata for courses
10/17	Planner	Implement core planner UI Allow user to set planner restrictions/preferences Draft & send e-mail to key supporters inviting them to a pre-alpha test
10/24	Profile	Implement ability to save preferences Implement ability to save plan
10/31	Profile, cont'd	Add ability to configure privacy settings Add ability to view others' profiles (if not private) Write alpha user recruitment post & publish it to Piazza
11/7	Course browsing	Add course index (browse & search) page Add course detail page Film project video trailer
11/14	Course reviews	Add course review / "add to course history" interface Account for already-completed courses in user plan Add per-course statistics on course thumbnails and

		course details Write beta user recruitment post & publish to Google+ groups
11/21	Course statistics	Finish course review importer (from UCRSS), populate legacy reviews Adjust course recommendation algorithm to prioritize courses with higher rating
11/28	Review & retrospective	Final paper/presentation
12/3*	Preparation for future work	Solicit beta user feedback Plan future research directions

Risk mitigation

The project does not depend extensively on external integrations. While use of omcs-adv is limited to students at Georgia Tech, integrating with IAP appeared too involved for the duration of this project – therefore, identity verification will be performed purely through e-mail address verification (i.e., students will be required to sign up with a Georgia Tech e-mail address). This avoids complex data use restrictions and scary FERPA requirements; it does require that the students manually enter any course history data they might wish the advisor to consider, but even this is not such a great burden since they should have fewer than 10 courses in total (far fewer than the ~50 for an undergraduate).

However, the project does suffer from another source of risk:

No project plan survives first contact with the user.

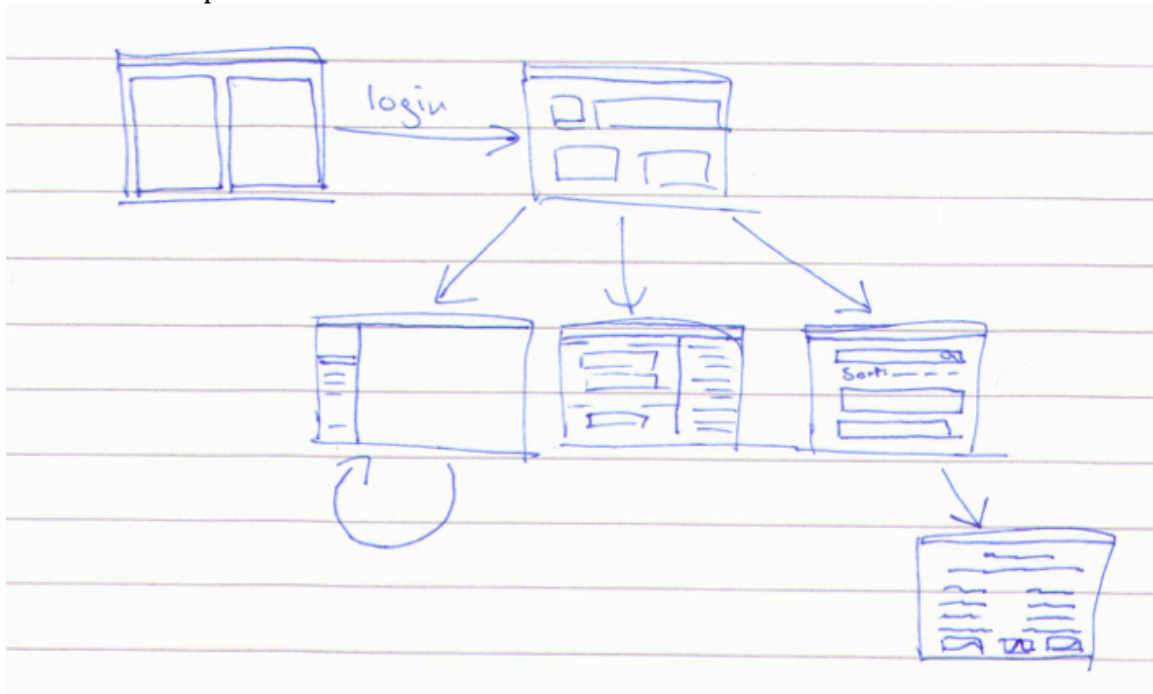
As the project is somewhat over-planned, this could hit it particularly hard. The plan calls for mitigating this risk through the use of staged rollouts:

- By 10/17, a hand-picked set of users will be solicited to give feedback on just the planner UI. (I identified these students during the Spring 2015 registration period, the Fall 2015 registration period, and the first few weeks of this class. They represent a small group very interested in the subject of course selection.)
- By 10/31, any users enrolled in Educational Technology will be solicited to try the initial version of the webapp.
- By 11/14, all users in the program will be invited to create accounts and give feedback.

In the event that beta users express dissatisfaction with the app, the work currently planned for the week of 11/21 and 11/28 can be swapped out for more in-demand features. No such accommodation is currently in the schedule to allow for major changes suggested by pre-alpha and alpha users. This may prove to be a major weakness of the plan.

Appendix A: Draft wireframes

Overall sitemap:



Dashboard / logged in homepage:

Home

← → × ↗

http://www.omscs-advisor.com/


Q

Home

Profile

Courses

Logout



Student

Location: Seattle, Washington, USA

Joined: 2015/10/01

Specialization: Machine Learning
(Interactive Intelligence, Systems)

Graduating: Summer 2016

This term you are enrolled in [CS 6460/EdTech](#) and [CS 8803/RL](#).
[Changes?](#)

Next term (Spring 2016) we project [23 courses](#). Your plan has you in [CS6505/CCA](#). [Replan?](#)

Please review CS7637/KBAI!
Oct. 1, 2015

Blog post: Advisor v1.1 released!
Sept. 22, 2015

Blog post: Quiz time! Which prof are you?
Sept. 22, 2015

Add course [to history] / review course modal:

CS6300/SDP

Summer 2015

☐ Withdrawn

Help omscs-advisor learn about you.

Difficulty: ★★☆☆☆

Effort max:

20

Value: ★★☆☆☆

Average:

2

Grade:

A

Recording your grade is optional. Your individual grades or GPA will never be shared under ANY circumstances.





Help your classmates learn about this course.

☐ Leave this review anonymously


Done

Planner UI:

Planner



http://www.omscs-advisor.com/




Home

Profile

Courses

Logout



Add new course

Spring 2016

*

CS6505

(CCA)

Computational Complexity and Algorithms

Charles Brubaker presenting I Chris

Rating

Popularity

Workload



★★★★★

3rd

25-30

10 hours/week, 20 hours/week peak, max

25



Summer 2016

Specializations:

Primary

 Machine Learning

 Systems

Graduation:

Fall 2016



Constraints:

Max hours:

25



/week

Group projects:

☒ Yes






☐ No

Plan

Save

Course index / search page:

Courses



Home

Profile

Courses

Logout

Courses offered in

Fall 2016

Q search

Sort by: [number](#) [name](#) [popularity](#)

CS7637 (KBAI)	Knowledge-Based Artificial Intelligence: Cognitive Systems Ashok Goel presenting David Joyner instructing	Rating ★★★★★	Popularity 1st	Workload 10-25
CS7641 (ML)	Machine Learning Charles Isbell presenting Charles Isbell and Michael Littman instructing	Rating ★★★★★	Popularity 2nd	Workload 10-40

Course detail page:

CS 7637 (KBAI) - Knowledge-Based Artificial Intelligence: Cognitive Systems

← → ✕ 🏠 🔍


CS 7637 (KBAI)

Knowledge-Based Artificial Intelligence: Cognitive Systems


Ashok Goel presenting
Established: Fall 2014

David Joyner instructing
[Readiness test](#)
[Syllabus](#)


At a glance




Quality



Difficulty



Workload



Alumnus

I thought this course was awesome! Here, let me tell you about it. First, we talked about the meaning of intelligence. Then...

Rating

★★★★★

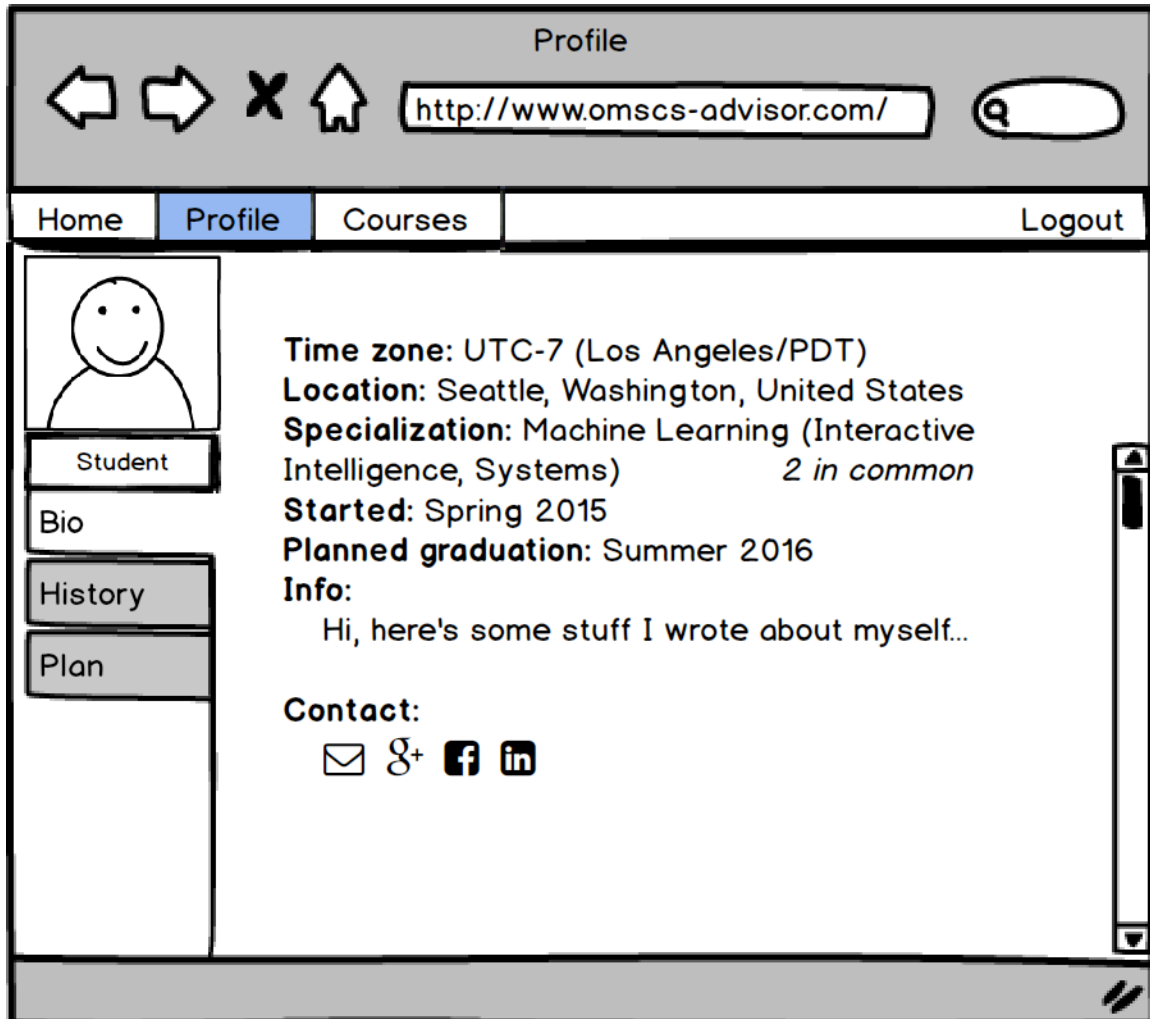
Difficulty

★★★★☆☆

Workload

10-25

Profile (others'):



Appendix B: Draft metadata specification

*(Excerpted from the requirements document on GitHub. Included for convenience in envisioning the scope of metadata work. **Not required reading!**)*

The course metadata will be defined in the JSON file described in 2.1.3.1. The specialization metadata will be defined in the JSON file described in 2.1.3.2.

Generally, the following points are true of both sets of metadata:

1. Data will never be erased from these files, only added.
2. All data will have an "EffectiveFrom" date.
3. The only change which can be made to an existing record in these files is the addition of an 'effective until' date. All other changes will be expressed with the addition of new records.
4. The effect of adding an "EffectiveUntil" date to an existing record is to make the corresponding entity unavailable after the EffectiveUntil date. For courses, this means that it will cease to be offered after the named term. For specializations, this means that it will not be listed as an option for students whose first term in the program is *after* the named term.
5. The effect of adding a new record with the same key as an old record is to update the contents of the corresponding entity from the effective date on. So, for instance, adding a new record for CS 6300 with "Instructor" => "Spencer Rugaber" and "EffectiveFrom" => "Fall16" in the course metadata file will cause course offerings prior to Fall Term, 2016 to retain the instructor Alessandro Orso, while offerings during or after Fall Term, 2016 will reflect Spencer Rugaber as the instructor.
6. The format for effective and end dates is . A year code may be two digits or four digits (so "Fall16" and "Fall2016" are both valid representations of Fall Term, 2016). Term names should be case-insensitive (so "Fall16" and "FALL16" are both valid representations). Effective and end dates are inclusive.

2.1.3.1 Course metadata format

The course metadata will be contained in a file named course-metadata.js.

This file will contain one (1) array named courseMetadata. Each entry in the array will be an object.

These fields are used to describe the set of course offerings to which a given metadata object should be applied:

- Id: unique identifier for the course. All metadata objects with the same ID are considered to apply to the same course, and represent different values for its properties at different points in time.
- Summer: (boolean) if true, the attributes in this bundle apply only to summer offerings of the course. If false, they apply only to spring and fall terms. Note that in the absence of at least one metadata entry for a given course id that sets Summer=true, the course is assumed *not* to be available in summer terms.

- **EffectiveFrom:** if the first entry for a course, the course was not available / is not available prior to this date. If a subsequent entry, the changes in this entry go into effect on this date.
- **EffectiveUntil:** the course will be unavailable after this date.

The following fields will be recognized and applied to all terms in the given range:

- **Number:** (string) official course number.
- **Title:** (string) official course title.
- **Summary:** (string) course summary from Udacity.
- **Instructors:** (array of strings) official instructor name, last name first; e.g., "Orso, Alessandro".
- **Syllabus:** (nullable string) link to the syllabus.
- **Readiness:** (nullable string) link to the readiness quiz or set of bullet points.
- **UdacityRequired:** (boolean)
- **PiazzaRequired:** (boolean)
- **Specialization:** (string) the specialization ID of the specialization which should be displayed for the course in parts of the UI where only the "primary" specialization should be displayed. Note that this attribute is cosmetic; the actual data about what specializations a course fits into (and how) is encoded in the specialization metadata.

The following fields will be recognized and applied to only summer terms (if `summer=true`) or non-summer terms (otherwise):

- **ProctoredExams:** (nonnegative integer)
- **TimedExams:** (nonnegative integer)
- **OpenBookExams:** (nonnegative integer)
- **GroupProjects:** (nonnegative integer)
- **CodingAssignments:** (nonnegative integer)
- **WrittenAssignments:** (nonnegative integer)

Some fields appear above without further explanation; all are used directly with no further processing, and as such also appear in section 2.1.3.3 (where more details are given).

2.1.3.2 Specialization metadata format

The specialization metadata will be contained in a file named `specialization-metadata.js`.

This file will contain one (1) array named `specializationMetadata`. Each entry in the array will be a nested object. Each entry will require a `specializationName` field and a `specializationId` field. The following additional fields will be recognized: `oneOf`; `twoOf`; `threeOf`; `allOf`.

The value of each field is an array of either course identifiers or other nested objects (again with field names `oneOf`, `twoOf`, `threeOf` or `allOf`).

For example, an entry of the form

```
[{"specializationName": "Interactive Intelligence", "specializationId": "II",
  "allOf": [ [ "oneOf": [ "CS6300", "CS6505" ], [ "twoOf": [ "CS7637", "CS7641" ] ],
  "oneOf": [ [ "twoOf": [ "CS6440", "CS6460", "CS7634" ], [ "twoOf": [ "CS6795",
    "CS7610", "CS8803CC" ] ] ] ] ] ]
```


describes the Interactive Intelligence specification (albeit with some as-yet unavailable courses removed).

2.1.3.3 Defined course attributes

2.1.3.3.1 Id

A unique identifier for the course. This is usually the same as the official course number but for 8803 (Topics) courses, may include a unique suffix (e.g., CS8803AIR, CS8803RL, ...).

2.1.3.3.2 Number

The official course number, e.g., CS6300.

2.1.3.3.3 Title

The official course title, e.g., Software Development Process.

2.1.3.3.4 Summary

The one-sentence summary taken from Udacity, if available.

2.1.3.3.5 Instructors

The official instructor(s) for the course, e.g., Alessandro Orso.

2.1.3.3.6 First offered

The term in which the course was first offered. If the first offered term is in the past, terms prior to that one must not be listed as options when the user is adding this course to their history. If the first offered term is in the future, the course must not be added to any user's plan in a term prior to that one.

2.1.3.3.7 Available in summer

Whether or not the course is available in summer. If it is not, the course must not be added to any user's plan in a summer term.

2.1.3.3.8 Syllabus link

A link to the official syllabus (if available) on [the main OMSCS site](#).

2.1.3.3.9 Readiness link

A link to any official readiness survey or description of the course's recommended prerequisites.

2.1.3.3.10 Timed proctored exams

The number of exams proctored through ProctorU.

2.1.3.3.11 Timed non-proctored exams

The number of exams with time limits enforced through T-Square but no ProctorU requirement.

2.1.3.3.12 Untimed exams

The number of untimed, open book exams, also known as "take-home exams".

2.1.3.3.13 Group projects

The number of assignments, of any kind (coding or non-coding), that must be done with members of an instructor-assigned group.

2.1.3.3.14 Coding assignments

The number of coding projects that must be done alone.

2.1.3.3.15 Non-coding assignments

The number of non-coding projects (e.g., papers and problem sets) that must be done alone.

2.1.3.3.16 Udacity required?

Whether or not the course requires completing all Udacity quizzes / watching all Udacity videos while logged into your GATech account.

2.1.3.3.17 Primary specialization

The specialization to which this course may be applied. If this course is applicable to multiple specializations (most of them), one of them will be selected subjectively.

Note that this is used to determine how a course is displayed in certain parts of the UI, but it does *not* affect how the planner selects courses.

2.1.3.3.18 All specializations

All specializations to which this course may be applied.

2.1.3.4 Calculated attributes

2.1.3.4.1 Popularity

The popularity of a course is defined by a series of datapoints regarding how quickly it filled in previous registration periods. Two base popularity statistics and a prediction are available. The popularity statistics are based on data from a previous offering of the course. The prediction is calculated based on these statistics and other course metadata.

1. The previous course offering considered will be chosen based on the term currently under evaluation ("target term"), with preference given to terms in the following order.
 - i. If the target term is summer and the course has previously been taught in summer, the most recent summer offering of this course is preferred. If the target term is fall or spring, the most recent non-summer offering of this course is preferred.
 - ii. If the course has not previously been offered in a corresponding term, the most recent offering of the course is preferred.
 - iii. If the course has never before been offered, the popularity data for the *most popular* course offering the professor has taught previously is preferred.
 - iv. If the professor has never before offered an OMS course, the *average* popularity data across the *first offering* of all courses associated with the same specialization is preferred.
 - v. If the course has no associated specialization, the *average* popularity data across all courses offered for the first time in the *current term* (or most recent term for which data is available) is preferred.
2. The popularity index reflects which time ticket window was in effect when the course first reached cap. The goal of calculating this value is to enable the system to estimate the likelihood that a student with a given number of credits entering a term will be able to enroll in the course.
 - i. The popularity index is a decimal number in the range (30, -infinity).
 - ii. The base value for popularity index is based on when the course reached cap. It is a linear interpolation between the last time ticket for which it was open and the first for which it was closed. For example, if a course offering reached cap 50% of the way between the time ticket start for students with 6 earned credits and that for students with 3 earned credits, its popularity index was 4.5.

- iii. If the cap was reached but then raised, only the date/time at which the *final* cap was reached is relevant.
 - iv. If the final cap was never reached, the number of remaining slots is translated into a negative popularity index as follows: the number of remaining slots is divided by the number of students who signed up during the last time ticket (students with 0 credit hours) of registration and multiplied by -3. If no students signed up during the last time ticket, the popularity index should be displayed as 'N/A' and treated as negative infinity in all calculations for which it is used.
3. The popularity ranking reflects how quickly the course closed compared to other courses offered that term. The goal of calculating this value is to give the student a subjective idea of how popular one course is relative to another.
- i. The popularity index is an ordinal ranking (1, 2, etc., corresponding to 1st, 2nd, and so on).
 - ii. The first course offering to reach its registration cap in a term is #1. The second course offering to reach its registration cap is #2, and so on.
 - iii. Course offerings which do not reach their registration caps are ranked below the course offerings which did in order of the number of students enrolled in each at close of registration.
 - iv. For a particular course, if the cap was reached but then raised, only the date/time at which the *original* cap was reached is relevant.

2.1.3.4.2 Grade and withdrawal split

What proportion of the class receives an 'A', according to critique.gatech.edu.

- 1. The previous course offering considered will be chosen with the following order of preferences.
 - i. If grades are available from at least one OMSCS section of the same class taught by the same instructor, the most recent OMSCS section will be considered.
 - ii. If grades are not available from any OMSCS sections, the totals across all sections of the same class taught by the same instructor will be considered.
- 2. The grade split will be calculated as $(A \text{ count}) / (A \text{ count} + B \text{ count} + C \text{ count} + D \text{ count} + F \text{ count})$. Withdrawals will not be considered.
- 3. The withdrawal split will be calculated as $(A \text{ count} + B \text{ count} + C \text{ count} + D \text{ count} + F \text{ count}) / (A \text{ count} + B \text{ count} + C \text{ count} + D \text{ count} + F \text{ count} + W \text{ count})$.

2.1.3.4.3 Workload

The median estimated weekly workload (from course reviews for previous offerings), along with the 85% mark and the 15% mark (i.e., 85% of students thought it took less work than this; 15% of students thought it took less work than this).

- 1. The previous course offering whose reviews are considered will be chosen with the following order of preferences.

- i. If at least 20 reviews are available from the most recent offering of this course, only reviews from that offering will be considered.
- ii. If fewer than 20 reviews are available from the most recent offering of this course, the 3 most recent offerings of the course will be considered.

2.1.3.4.4 Difficulty

The average estimated difficulty (from course reviews for previous offerings).

The previous course offering whose reviews are considered will be chosen with the same preference rules as for 2.1.3.4.3: Workload.

2.1.3.4.5 Value

The average estimated value (from course reviews for previous offerings).

The previous course offering whose reviews are considered will be chosen with the same preference rules as for 2.1.3.4.3: Workload.