SKEDGE

Smarter course scheduling for our University of Rochester

Dan Hassin

Supervised by
Professor Philip Guo

Department of Computer Science
University of Rochester
Rochester, New York

April 12, 2016

submitted in partial fulfillment of
the requirements for the degree
honors bachelor of science

Contents

List of Tables

List of Figures

Abstract

In this paper I present Skedge, a web application for students to comfortably and effectively engage with the University's course catalog. Skedge matches and surpasses the capabilities of the existing University tool for this purpose, "Course Description / Course Schedule" (CDCS) and presents its information in a more visually pleasing way. As a result, Skedge boasts strong user-retention rates, long session durations, and high student adoption despite having virtually no advertisement. Through collected usage data, I demonstrate that a) Skedge's differences from and additions to CDCS are usable and have real need, b) the two major use-cases associated with course browsing—direct search and exploratory search—are effectively accommodated by Skedge, and c) Skedge's search mechanism is user-friendly and self-teaches to users over time.

Chapter 1

Introduction

This paper will begin by

1.1 Space of course explorers and schedulers

1.2 Overview of CDCS

1.3 Overview of Skedge

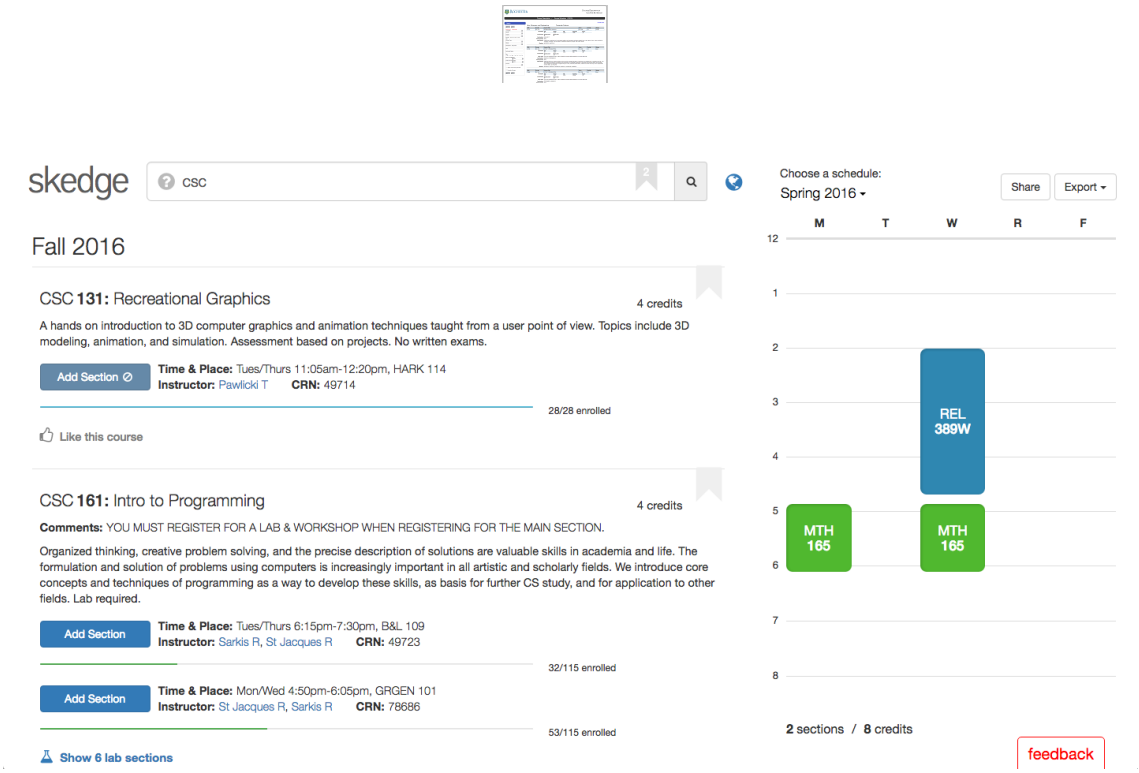

Figure 1.1: CDCS (top) and Skedge (bottom) for the search query `csc`.

Chapter 2

Design as a reaction to CDCS

From its very inception, Skedge's functionality and visual design were driven by the shortcomings of CDCS. It was built *bottom-up*, not *top-down*, as every aspect of the application was either made as a reaction to a particular grievance in CDCS or as the natural evolution of an existing feature. Skedge is thus rooted in *usability* derived from real need, not mere conjecture along the question "what could students want?". Its success with students, shown in Chapter 4, demonstrates that this usability extends beyond my own standard and can fulfill the various discovered use-cases of students in general.

In this chapter, I will

2.1 Modernity

CDCS is an old system.

2.1.1 GET requests vs. AJAX

- Can use back button - Can send a link to a course or search

2.1.2 Built-in scheduler vs. browser extension

- Better UX - Data is centralized

2.1.3 Mobile

- Important nowadays - Mobile traffic stat or smth

2.1.4 Public API

- Important nowadays, extends student possibility - JSON - Brief demo of API

2.2 Usability

2.2.1 Data quality

- Courses don't shout - Typos in comments - 12-hour time

2.2.2 Section display

- Grouped course sections - Embedded labs (A/B too), workshops, & recitations

2.2.3 Course reference

- Clickable/hoverable course links, professor searches

2.2.4 Multiple schedule support

- Old CDCS+betterCDCS system can't keep track of this, have conflicts when adding stuff

2.2.5 Exporting to GCal, .ics, image

- Mobile sync support - Security: BetterCDCS export gcal is currently broken and sends netID in PLAINTEXT over http(!!!)

2.2.6 Search

Most important usability concern is finding courses.

2.3 Search

Use cases, natural language.

2.3.1 Course selection criteria

Narrowed it down to three criteria. Keep in mind that *none* of the things listed below are supported by CDCS, and they are all supported by Skedge.

Requirements

- Finding crosslists - Clusters

Browsing

- "New" courses - "Autofit" search - Random - Sorts

Friends

- "What are my friends taking?" ("what are you taking this semester" = probably most common smalltalk phrase uttered on campus) - "What do my friends recommend?" - "have you taken this class, and if so, what did you think of it?"

2.3.2 Natural language search

See figure.

Advantages

- 15 fields reduced to 1

  vs form entry: - Faster - More intuitive - More easily extendable

Disadvantages

Having to know the DSL, grammar ambiguities (can be solved with a 'did you mean')

2.3.3 Multipurpose

Used by other links (instructors, course references) around the site

2.3.4 Added features

- CRN (!) - Crosslist - Class size

2.4 Social

2.4.1 The issue

Static image vs. live site

- Edits don't update - Referencing courses

Finding common courses

- requires your friends to share their schedules on FB publicly and you to see their post
    - is schedule-first, not search-first - typically only occurs for the current semester

2.4.2 Skedge Social

Friends' course enrollments

Mini-feed

Friends' course likes

Likes & enrollments embedded in results

Personal schedule synchronization

Privacy

Notifications

Chapter 3

Technical overview

3.1 Back-end

Skedge's infrastructure is built

nginx, *unicorn*, *Ruby on Rails*, *PostgreSQL*, *React.js*, *Ahoy*, and Google Analytics.

3.2 Front-end

3.3 Analytics

Chapter 4

Data analytics

Hypotheses:

1. Skedge's differences from and additions to CDCS are usable and have real need

2. Skedge's navigations-per-add and other metrics demonstrate effectiveness of the use cases a) direct searching, and b) course browsing

3. Skedge's DSL is user-friendly; users learn more advanced search types over time by using it

4.1 Usage

4.1.1 General

Since November 3rd 2015 (137 days) 3,768 unique users 4,500 schedules Average 90 sessions/day Average 4.92 pages/session Average 5:31 minutes/session 28% of sessions are from new users

MOBILE RESULT

4.1.2 Search

Empty searches

Can learn from these Some funny ones

4.1.3 Course blocks

40% of sessions have at least one block-click Average of 4.94 block-clicks per session

4.1.4 Social

90 users have linked Skedge to Facebook Since March 1st, 4,000+ visits (200 visits/day) 60% of visits to /social were returning visitors 90 overlays onto friends' schedules 10 clicks to Facebook profiles :(- get stats from the fb dashboard

4.1.5 Conclusion

Success! Considering skedge is OPTIONAL. + course blocks (obv usecase, can't click) + exports (not supported by thing) + mobile

4.2 Navigations-per-add

4.2.1 Definitions

A navigation is defined as a search, or a click on an instructor's name, or a click on a crosslisted or prerequisite course link

The navigations-per-add, bookmark measure is the number of navigations a user took (within one session) until a course was added, bookmarked

4.2.2 Trends

4.2.3 Breaking them apart

behavioral patterns Direct search for specific course Discovery, browsing, exploring

Direct searches

Browse

4.2.4 Conclusion

Effective++

4.3 Users' search types over time

4.3.1 Definitions

Points for search by (omits number and dept.):

description credits crosslisted CRN instructor title year term 'random' upper-level writing

"CSC" 0 "MTH 165" 0 "taught by hema" 1 (2 searches) "random mur 1-2 credits" 2 (1 search)

4.3.2 Trends

First increase (60.5Median: 2 searches Average: 4.23 searches (Starting at 1 counts as an increase value of 0)

Second increase (7.9Median: 8 searches Average: 17.52 searches

4.3.3 Conclusion

DSL++

Chapter 5

Looking forward

5.1 Features

5.2 Analytics

Chapter 6

Conclusions

6.1 Proposal to the University

6.2 Resources

Source code

The source code for Skedge is available online under an open source license:

`https://github.com/RocHack/skedge`.

Live site

The site can be found at: `http://skedgeur.com`.

Bibliography

[1] Takis Konstantopoulos *Introductory lecture notes on Markov Chains and Random Walks.* Uppsala University,

`http://www2.math.uu.se/~takis/L/McRw/mcrw.pdf`

Appendix