Project Documentation

EZ-PLAN

1. Project Vision

It seemed to be the consensus that the current systems in place at UBCO are needlessly convoluted for navigating requirements and finding interesting or applicable courses. EZ-Plan was conceived as a tool that would make course registration easier, or even plan the student's schedule entirely, as well as provide a method for students to find specific courses that might be interesting to them.

2. Problem Statement

For students who are new to UBCO or to university in general (and for current students, to a lesser degree), course navigation is needlessly complex. It is easy to make mistakes when juggling course load requirements, pre-requisites, and other complicating factors, especially in the midst of course registration times or after failing or withdrawing from courses, when the pressure to reconsider future degree options is highest. A ccordingly, the EZ-Plan web application is intended to supplement existing Academic Advising programs, providing a direction so that UBCO students can register more easily and confidently, or a starting point from which to seek further input. It will recommend a course list for Computer Science students, considering factors such as existing plans for the major, current degree progress, and relevant interests (especially in order to find interesting electives outside one's major). Common pitfalls such as multi-part courses or courses with pre-requisites will be more immediately visible, as well. Additionally, as switching or choosing between degree programs is a daunting task, EZ-Plan also serves as a tool to compare possible degree programs, incorporating requirements that were already met and what courses one would have to take in order to catch up.

2.1 Comparison with Original Problem

In the original plan for the project, we did not foresee that the phrasing for degree requirements and pre-requisites was very inconsistent (the very problem we were trying to solve), meaning they could not be recorded automatically in a way that a program could easily make calculations from. Accordingly, we had to code some of these manually and reduced the size of the project from considering all possible degrees to only considering Computer Science degrees. As such, tools such as viewing eligibility for a course directly could not be completed, and the degree comparison feature is trivial.

2.2 Comparison with Original Deliverables

2.2.1 Deliverables Added

- We did not add any significant new functions to our plans for this project over the course of development.

2.2.2 Deliverables Removed

- Degree types other than Major in Computer Science, at this time, are not supported.
- Degree Comparison feature uses static data for other degree options rather than multiple user-scheduled degrees.
- Pre-requisites and other eligibility issues will, at best, be shown in tooltips (rather than used in calculations like giving an error if a course is taken without its pre-requisites).

---

3. Team Members

Arturo Padilla: **Developer**. I was assigned to work on the database design; my responsibility was creating the database DDL according to the database UML diagram

Ilerioluwa Oyedele: **Developer**. I was assigned the task of designing a database for storing all our site data. My deliverable was UML Diagram and relational schema.

Ashan Kevin Semasinghe: **Product Owner** and **Developer**. As the product owner, I write the user stories and help my team members with writing them. I facilitate discussion about the project as a whole, lead parts of the sprint planning meeting and help resolve conflicts with user stories or with the general direction of the project. As a developer, i am responsible for the visual design of the webpage, mainly focused on creating mockups and coding up the final website.

Eliana Wardle: **Scrum Master** and **Developer**. I facilitated the Agile Scrum process by directing the focus of scrum meetings and ensuring their timeliness, scheduling meeting times, and taking notes from each meeting. As a developer, I also provided input for sprint planning, scrums, and documentation, and completed tasks selected from the backlog. My tasks have primarily resolved around the crawler project, which initially retrieved course data automatically from the UBCO website and populated our database with it according to the fields in the database UML diagram. In Sprint 2, it was modified to support adding degree requirements as well (which had to be defined manually). Finally, I worked on some pages on the website that used this information (namely, suggesting courses and validating a schedule).

Lam Ng: **Developer.** As a developer, I was assigned the task of creating and designing a paper prototype mock up and HTML template for the website.

4. Development Process

4.1 Agile Scrum Process

Our team uses an Agile Scrum process. This entails (among other things) the creation and maintenance of a product backlog of all functions and tasks involved in the production of the project (each of which involves describing user stories, planning poker size estimation, acceptance criteria, and deliverables); holding brief, regular scrum meetings and feedback sessions; and alternating between periods of planning and themed sprints (containing tasks from the product backlog) intended to iterate on a potentially shippable product.

Some features of this process were relaxed: scrum meetings were sometimes done remotely where each member checked in on their own; when done in person, scrums were usually followed by longer planning sessions, in which issues brought up in the scrum were addressed and often fixed. Also, tasks were typically framed more around individual functions than added-value statements for users, as we found this method easier for determining what would be required for the tasks.

4.2 Development and Management Platforms

Our team uses JIRA as a management platform. We have set up our product backlog on a JIRA page and created our sprints off this, logging work, updating (if needed) and closing issues during an ongoing sprint.

We supplement JIRA with a Confluence space that we use to share resources such as mockups, deliverables and documentation of designs for relevant pieces, and as a platform for documenting our entire project development process.

Furthermore, we decided to use GitHub as the platform for sharing any code between ourselves--to pull code from and work on in our own branches before merging the code when everyone has completed their part of the task.

A Facebook message chain was often used to check in on progress or ask questions outside of our regular Scrum meetings.

4.3 Communication Methods

We meet for bi-weekly scrums every Monday after class and Thursday at lab start. We discuss what work we have been doing regarding our project and then move onto what our goals for the upcoming week is regarding either the current sprint or planning for the next one. We also discuss the current sprint in terms of how its moving along and if any issues need to be changed/looked at.

In addition to bi-weekly scrums, we conducted longer work sessions before the start of each sprint and before each demo presentation. These were typically from 2 to 3 hours in length and were focused on ensuring the product backlog and sprint backlog contained all the necessary detail and that developers had selected their tasks, or that the demo was scheduled and all materials were in a presentable format.

As an aside, we have created a messaging group to facilitate discussion on short notice or to bring up any issues we may come across between scrums or simply discuss items for the current or next sprint.

5. Glossary of Terms

Acceptance Criteria: Criteria/checklist to determine if a given deliverable meets specification. Must be checked off fully for a task to be deemed complete.

Confluence: Secondary management platform that focuses on storing and sharing documentation for tasks such as deliverables and designs as well as storing our project documentation.

Deliverable: List of items or item that needs to be delivered as a proof of task completion.

DDL: "Data Definition Language"; the statements that prescribe how data tables will be stored in the database.

GitHub: Code management platform used to share code between team members.

JIRA: Management platform used to store product backlog and organize sprints and task assignments.

Planning Poker: Method of estimating the size of a task (in hours, which are most intuitive for a team of our level of experience) in which each team member shows a numbered card of their time estimate, discusses each other's reasoning, and repeats until there is a consensus.

Scrum: Short, bi-weekly meeting that discusses work-log with current sprint as well as any issues that have come up with an item in the project.

User Story: Functional or non-functional description of a task related to the project.

6. Results

6.1 Requirement Gathering and Specifications

6.1.1 Product Backlog

Before Scope Change

Issue Type	Key	Summary	Priority	Status	Resolution	Due Date
Story	EZ-66	As a developer, I want the fields taken from the UBCO course website to populate the database, so that they can be queried from the EZ-Plan website.	Major	Open	Unresolved	
Story	EZ-65	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Fixed	
Story	EZ-64	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Fixed	
Story	EZ-63	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Fixed	
Story	EZ-62	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Fixed	
Story	EZ-61	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Validate	Done	
Story	EZ-50	As developers, we need the search algorithm (for finding relevant courses) to return results that are actually relevant, so the user does not need to look for them manually.	Major	Open	Unresolved	
Story	EZ-49	As developers, we want the rudimentary page template to have a proper layout and visual design, so the user can find things on the page more easily.	Minor	Open	Unresolved	
Story	EZ-48	As developers, we need to code the template for the site's pages so that it can display consistently.	Major	In Progress	Unresolved	
Story	EZ-47	As a user, I want to be notified when the database of available courses has been updated or changed.	Trivial	Open	Unresolved	
Story	EZ-46	As a developer, we want to confirm that our course database is complete and accurate to the one offered on the UBC website, at regular intervals (roughly every 3 months).	Trivial	Open	Unresolved	

Story	EZ-45	As a user, I'd like to be able to browse courses by subject area (brief summary--course number, title, credits), and be referred to the full description if wanted, or schedule it (or mark as interest) outright.	Major	Open	Unresolved	
Story	EZ-44	As developers, we need to remind the users that (since time information is not available) they should confirm in the UBC worklist or with academic advising that there may be conflicts.	Minor	Open	Unresolved	
Story	EZ-43	As a user still deciding between options, I want to see a side-by-side comparison of potential degree programs or paths or schedules, to help me decide which I would prefer to take.	Minor	Open	Unresolved	
Story	EZ-42	As a user, I want to be able to modify a schedule (or add specifications to the scheduler beforehand) in case an auto-generated/default one does not fit or does not include the courses I want.	Major	Open	Unresolved	
Story	EZ-41	As a first year student, I would like EZ-Plan to calculate a schedule for me so that i do not need to go look for courses through various subjects to figure out what i need to take.	Major	Open	Unresolved	
Story	EZ-40	As a user, I want to be able to see my eligibility for each course (if missing prereqs, blocked entirely, already completed) when viewing a course list.	Major	Open	Unresolved	
Story	EZ-39	As a developer, I want EZ-Plan to filter out or distinguish between courses that are mutually exclusive or otherwise not eligible to the major/user.	Major	Open	Unresolved	
Story	EZ-38	As a user, I want to be recommended electives to take, or even a major, based on my preferences/interests.	Minor	Open	Unresolved	
Story	EZ-37	As a user, I want to see the progress I've made in my degree (whenever viewing a breakdown--credit requirements and course requirements)	Minor	Open	Unresolved	
Story	EZ-36	As a user, I want to be able to get a more detailed visualization of what courses are required for my major, or what the more specific graduation requirements are (possibly including double major or a minor).	Major	Open	Unresolved	
Story	EZ-35	As a user, I want to see how many of each type of credits I need (each year) for my major.	Major	Open	Unresolved	
Story	EZ-34	As a user, I need EZ-Plan to be able to make changes based off sudden cancellations of courses, so the results given will be accurate.	Major	Open	Unresolved	
Story	EZ-33	As a user, I need to be able to update my info (in case I pass courses or change degrees or something like that), so that EZ-plan can incorporate those changes into the plan later.	Major	Open	Unresolved	
Story	EZ-31	As a developer, I need to parse the course info from plain HTML on the website into its individual fields, so they can later be added to the database.	Critical	Validate	Fixed	
Story	EZ-30	As a developer, I want to know the database design in order to have a clear direction in systems implementation.	Major	Closed	Done	
Story	EZ-29	As a developer, I want to have a paper markup in order to have a clear idea of direction in implementation.	Major	Closed	Fixed	
Story	EZ-28	As a developer, I need a way to be able to store all the information pertaining to a single user efficiently so that it may be accessed and updated quickly.	Major	Closed	Fixed	

After Scope Change (Final Backlog)

Issue Type	Key	Summary	Priority	Status	Resolution	Epic Link	Sprint	Due Date
Bug	EZ-76	As developers, we need the database schema to be consistent with the functions on the website, so we can finalize the queries we use.	Blocker	Closed	Done	Database	Sprint 3: Improvements	25-Mar-16
Story	EZ-75	As developers, we need to be able to easily retrieve the requirements for a given major, so that course lists can be computed based on them.	Major	Closed	Done	Database	Sprint 2: Major Functions	15-Mar-16

Story	EZ-74	As a user, I want to be able to compare my degree plan with another option, so that I can make a more informed decision.	Major	Closed	Fixed	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements	6-Apr-16
Story	EZ-73	As a developer, we need to make sure the website can display a breakdown of a given degree to the user	Major	Closed	Fixed	User interface and functionality	Sprint 2: Major Functions	
Story	EZ-72	As a user, I want to see and edit my current progress in my degree, so that it can be used to find what else I need to take.	Major	Closed	Fixed	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements	6-Apr-16
Story	EZ-71	As a user, I need to be able to update my info, so that EZ-plan can incorporate those changes into the plan later.	Major	Closed	Done	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements	6-Apr-16
Story	EZ-70	As a user, I want to be able to access a home page which shows me general info about my profile and degree	Major	Closed	Fixed	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements	6-Apr-16
Story	EZ-68	As a developer i need to have the users information stored in the database when they sign up to be able to use it later on.	Major	Closed	Fixed	Database	Sprint 2: Major Functions	15-Mar-16
Story	EZ-67	As a developer I need to have the database framework for the User Profile page in place so I can add all the other features.	Critical	Closed	Fixed	User interface and functionality	Sprint 2: Major Functions	15-Mar-16
Story	EZ-66	As a developer, I want the fields taken from the UBCO course website to populate the database, so that they can be queried from the EZ-Plan website.	Major	Closed	Done	Database	Sprint 1, Sprint 2: Major Functions	15-Mar-16
Story	EZ-65	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Fixed		Sprint 1	
Story	EZ-64	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Fixed		Sprint 1	
Story	EZ-63	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Fixed		Sprint 1	
Story	EZ-62	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Fixed		Sprint 1	
Story	EZ-61	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Major	Closed	Done		Sprint 1	
Story	EZ-50	As developers, we need the search algorithm (for finding relevant courses) to return results that are actually relevant, so the user does not need to look for them manually.	Major	Open	Unresolved	Course Selection		

Story	EZ-49	As developers, we want the rudimentary page template to have a consistent layout and visual design, as well as a site logo, so the user can recognize and find things on the site more easily.	Minor	Closed	Done	User interface and functionality	Sprint 3: Improvements	31-Mar-16
Story	EZ-48	As developers, we need to code the template for the site's pages so that it can display consistently.	Major	Closed	Fixed	User interface and functionality	Sprint 1, Sprint 2: Major Functions	15-Mar-16
Story	EZ-47	As a user, I want to be notified when the database of available courses has been updated or changed.	Trivial	Open	Unresolved	User interface and functionality		
Story	EZ-46	As a developer, we want to confirm that our course database is complete and accurate to the one offered on the UBC website, at regular intervals (roughly every 3 months).	Trivial	Open	Unresolved	Database		
Story	EZ-45	As a user, I'd like to be able to browse courses by subject area (brief summary--course number, title, credits), and be referred to the full description if wanted, or schedule it (or mark as interest) outright.	Major	Closed	Fixed	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements	6-Apr-16
Story	EZ-44	As a user, I want to see that the degree plan I have chosen will be valid for my degree type, so that I know it will be close to ready for registration.	Minor	Closed	Done	User interface and functionality	Sprint 3: Improvements	6-Apr-16
Story	EZ-43	As a user still deciding between options, I want to see a side-by-side comparison of potential degree programs or paths or schedules, to help me decide which I would prefer to take.	Minor	Closed	Duplicate	User interface and functionality	Sprint 2: Major Functions	15-Mar-16
Story	EZ-42	As a user, I want to be able to modify a schedule (or add specifications to the scheduler beforehand) in case an auto-generated/default one does not fit or does not include the courses I want.	Major	Open	Unresolved	User interface and functionality		
Story	EZ-41	As a first year student, I would like EZ-Plan to calculate a schedule for me so that i do not need to go look for courses through various subjects to figure out what i need to take.	Major	Open	Unresolved	Course Selection		
Story	EZ-40	As a user, I want to be able to see my eligibility for each course (if missing prereqs, blocked entirely, already completed) when viewing a course list.	Minor	Closed	Fixed	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements	6-Apr-16
Story	EZ-38	As a user, I want to be recommended a schedule for my degree, including electives to take based on my preferences/interests.	Minor	Closed	Done	Course Selection	Sprint 3: Improvements	6-Apr-16
Story	EZ-37	As a user, I want to see the progress I've made in my degree (whenever viewing a breakdown--credit requirements and course requirements)	Minor	Closed	Duplicate	User interface and functionality	Sprint 2: Major Functions	15-Mar-16
Story	EZ-36	As a user, I want to be able to get a more detailed visualization of what courses are required for my major, or what the more specific graduation requirements are.	Major	Closed	Duplicate	User interface and functionality	Sprint 2: Major Functions	15-Mar-16
Story	EZ-35	As a user, I want to see how many credits I need (each year) for my major.	Major	Closed	Won't Fix	User interface and functionality	Sprint 2: Major Functions	15-Mar-16
Story	EZ-34	As a user, I need EZ-Plan to be able to make changes based off sudden cancellations of courses, so the results given will be accurate.	Major	Open	Unresolved	Course Selection		

Story	EZ-33	As a user, I need to be able to update my info (in case I pass courses or change degrees or something like that), so that EZ-plan can incorporate those changes into the plan later.	Major	Closed	Fixed	User interface and functionality	Sprint 2: Major Functions	15-Mar-16
Story	EZ-31	As a developer, I need to parse the course info from plain HTML on the website into its individual fields, so they can later be added to the database.	Critical	Closed	Fixed	Database	Sprint 1	
Story	EZ-30	As a developer, I want to know the database design in order to have a clear direction in systems implementation.	Major	Closed	Done	Database	Sprint 1	
Story	EZ-29	As a developer, I want to have a paper markup in order to have a clear idea of direction in implementation.	Major	Closed	Fixed	User interface and functionality	Sprint 1	
Story	EZ-28	As a developer, I need a way to be able to store all the information pertaining to a single user efficiently so that it may be accessed and updated quickly.	Major	Closed	Fixed	Database	Sprint 1	

6.1.2 Functional Requirement Specification

Sprint 1

Key	Summary	Issue Type	Status	Priority	Resolution	Resolved	Description	Epic Link	Sprint
EZ-48	As developers, we need to code the template for the site's pages so that it can display consistently.	Story	In Progress	Major	Unresolved		Deliverable: sample page with the menu, etc. elements of a basic template that all group members agree on look and feel of Refer to JIRA link for acceptance criteria.	User interface and functionality	Sprint 1, Sprint 2: Major Functions
EZ-29	As a developer, I want to have a paper markup in order to have a clear idea of direction in implementation.	Story	Closed	Major	Fixed	18/02/2016 15:38	Refer to picture on confluence (Prototype breakdown) for description and acceptance criteria Deliverable: paper prototype of user interfaces, integrated with a proper decision tree	User interface and functionality	Sprint 1

Sprint 2

Key	Summary	Issue Type	Status	Priority	Resolution	Resolved	Description	Epic Link	Sprint

EZ-33	As a user, I need to be able to update my info (in case I pass courses or change degrees or something like that), so that EZ-plan can incorporate those changes into the plan later.	Story	Closed	Major	Fixed	18/Mar/16 5:50 PM	Acceptance criteria: User must be able to make changes to their information if needed and see these changes have an immediate impact As a developer, ensure that changes are stored appropriately in database Deliverable: almost identical to registration page, except instead of creating new user, updates fields for an existing user	User interface and functionality	Sprint 2: Major Functions
EZ-35	As a user, I want to see how many of each type of credits I need (each year) for my major.	Story	Closed	Major	Won't Fix	18/Mar/16 5:25 PM	Acceptance criteria: User must be able to see the credit required for the degree they are interested in Deliverable: A functional display for the user	User interface and functionality	Sprint 2: Major Functions
EZ-37	As a user, I want to see the progress I've made in my degree (whenever viewing a breakdown--credit requirements and course requirements)	Story	Closed	Minor	Duplicate	18/Mar/16 5:52 PM	Acceptance criteria: Deliverable: page listing met requirements and credit types in a credit or course breakdown Acceptance criteria: Database must be able to handle changes and reflect these changes accurately on the website User must be able to see their progress with a chosen degree as they update	User interface and functionality	Sprint 2: Major Functions

EZ-40	As a user, I want to be able to see my eligibility for each course (if missing prereqs, blocked entirely, already completed) when viewing a course list.	Story	Open	Major	Unresolved	6/Apr/16 5:07 PM	Write a script to be used when courses displayed on any page (and give a description) based on the user's eligibility. Acceptance criteria: * Each of these courses, when hovered over, would display a tooltip showing the pre-requisites and co-requisites, OR link to the appropriate course page. (Refer to Balsamiq mockups for more specifics.) Deliverable: script for custom object that makes color-coded, tooltip/linked eligibility status, wherever a course shows up in the course list. Additional notes: Each page, when rendering courses, should eventually make use of this script.	User interface and functionality	Sprint 2: Major Functions
EZ-43	As a user still deciding between options, I want to see a side-by-side comparison of potential degree programs or paths or schedules, to help me decide which I would prefer to take.	Story	Closed	Minor	Duplicate	03/Mar/16 3:40 PM	Acceptance criteria: See balsamiq mockup. Must be similar Deliverables: multi-column/comparison view showing more than one degree plan ("compare with..." options)	User interface and functionality	Sprint 2: Major Functions
EZ-36	As a user, I want to be able to get a more detailed visualization of what courses are required for my major, or what the more specific graduation requirements are (possibly including double major or a minor).	Story	Closed	Major	Duplicate	18/Mar/16 5:54 PM	Acceptance criteria: User must be able to see a detailed course breakdown for a given degree Deliverable: page displaying specific courses and elective requirements according to a given major	User interface and functionality	Sprint 2: Major Functions
EZ-45	As a user, I'd like to be able to browse courses by subject area (brief summary--course number, title, credits), and be referred to the full description if wanted, or schedule it (or mark as interest) outright.	Story	In Progress	Major	Unresolved		Acceptance criteria: Course Browser page	User interface and functionality	Sprint 2: Major Functions

EZ-70	As a developer, we need to ensure that the user can see their credit requirements for a given degree on the website	Story	Open	Major	Unresolved		Deliverable: Web-page working with database, displaying the correct info regarding credit requirements for a given degree with their yearly breakdown Acceptance criteria: User must be able to see an accurate credit value for a given degree along with the accurate breakdown User must be able to swap between degrees and see the value change if expected to.	User interface and functionality	Sprint 2: Major Functions
EZ-73(sub-task of EZ-70)	As a developer, we need to make sure the website can display a breakdown of a given degree to the user	Sub-Task	Open	Major	Unresolved		deliverable and acceptance criteria same as main task. This task focused more on making sure the info is displayed (EZ-70)	User interface and functionality	Sprint 2: Major Functions
EZ-71	As a user, I need to be able to update my info, so that EZ-plan can incorporate those changes into the plan later.	Story	Open	Major	Unresolved		Acceptance criteria: • My Info page ◦ User can define list of up to 5 interests, max 20 chars each ◦ Login-related fields can be changed ◦ Connect to and update User table ◦ Error-checking to confirm that the e-mail is valid (in form) and unique (in database) ◦ Link to Edit Degree page for changing taken courses Deliverable: Setting up and creating fields for the edit page so that it is ready for data to be inserted and linking the pages together	User interface and functionality	Sprint 2: Major Functions
EZ-72	As a user, I want to see and edit my current progress in my degree, so that it can be used to find what else I need to take.	Story	Open	Major	Unresolved		Acceptance Criteria: Edit Degree page	User interface and functionality	Sprint 2: Major Functions

EZ-74	As a developer, the website must be able to compare degrees using a breakdown	Story	Open	Major	Unresolved		Description: A user should be able to add degrees they want to compare and then see a breakdown that displays information about each of the degrees they have chosen Acceptance Criteria: 1) User goes to comparison page, initially displaying a box with an add symbol 2) User clicks on the add button and selects from a variety of majors listed in their degrees to add to the comparison list 3) After the user has added as many degrees as they want to compare (Maybe max this out at 3?) they can hit compare to see a breakdown 4) Breakdown should show the user a side by side comparison of each degree, colour-coded to highlight their good points or bad 5) Breakdown will show each degree choice's total credits, the remaining credits the user must acquire for that degree, and list the course breakdown for the user	User interface and functionality	Sprint 2: Major Functions

Sprint 3

Key	Summary	Issue Type	Status	Priority	Resolution	Resolved	Description	Epic Link	Sprint

EZ-74	As a user, I want to be able to compare my degree plan with another option, so that I can make a more informed decision.	Story	Closed	Major	Fixed	6/Apr/16 4:25 PM	Description: A user should be able to add degrees they want to compare and then see a breakdown that displays information about each of the degrees they have chosen Acceptance Criteria: 1) User goes to comparison page, initially displaying 3 dropdown menus to select upto 3 different degrees from 2) After the user has added as many degrees as they want to compare (Maybe max this out at 3?) they can hit compare to see a breakdown 3) Breakdown should show the user a side by side comparison of each degree, listing all required courses and elective requirement as well as total credits required and how many credits the use has completed towards this degree if any	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements
EZ-72	As a user, I want to see and edit my current progress in my degree, so that it can be used to find what else I need to take.	Story	Closed	Major	Fixed	6/Apr/16 5:05 PM	Acceptance Criteria: Edit Degree page * Sections for completed courses and for current courses ** Drop down menu of all the courses in the degree ** Show currently registered ones, with links/options to drop them ** Connect to Course (query) and UserCourse (insert) * User can search for a course to add ** Two fields: dropdown of specific requirements for the degree, and plain text entry ** Database gets updated based off their additions *** UserID+course code is a composite key so it will not add duplicates if they try * Link to Validate page	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements

EZ-71	As a user, I need to be able to update my info, so that EZ-plan can incorporate those changes into the plan later.	Story	Closed	Major	Done	6/Apr/16 3:48 AM	Acceptance criteria: * My Info page ** User can define list of up to 5 interests, max 20 chars each ** Login-related fields can be changed ** Connect to and update User table ** Error-checking to confirm that the e-mail is valid (in form) and unique (in database) ** Link to Edit Degree page for changing taken courses ** Edit Name and degree standing Deliverable: Setting up and creating fields for the edit page so that it is ready for data to be inserted and linking the pages together	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements
EZ-70	As a user, I want to be able to access a home page which shows me general info about my profile and degree	Story	Closed	Major	Fixed	6/Apr/16 4:25 PM	Deliverable: Home page displaying the user's profile info as well as general degree overview Acceptance criteria: 1)Shows the user's name and degree choice on left, if logged in Connect to User in database (query) 2)"About…" (what does this site do) 3)On right: degree choice, credits taken so far; button link to Edit Degree page	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements
EZ-45	As a user, I'd like to be able to browse courses by subject area (brief summary--course number, title, credits), and be referred to the full description if wanted, or schedule it (or mark as interest) outright.	Story	Closed	Major	Fixed	6/Apr/16 2:38 PM	Acceptance criteria: Course Browser page that shows: * Course name * Course title * Course description * Credits * Pre-requisites and co-requisites * An option to add course to user's list of courses * Course title should be provided to page (and details shown) by GET method so it can be linked to from other pages	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements

EZ-44	As a user, I want to see that the degree plan I have chosen will be valid for my degree type, so that I know it will be close to ready for registration.	Story	Closed	Minor	Done	6/Apr/16 4:08 PM	Acceptance criteria: * Validate Degree page (check that all degree requirements are met in the user's plan) ** Linked to from Edit Degree page *** Script to check that, for each requirement, the right number of courses from it are present in the user plan or already-taken courses *** Connect to UserCourse, DegreeView, CourseRequirement *** May color already taken courses differently than planned ones ** Give confirmation in green if valid, or list unmet requirements (descriptions) in red if not * Static disclaimer (and popup and confirmation), as soon as the user schedules courses, that this is not an infallible system (should confirm in SSC or with academic advising in case of conflicts)	User interface and functionality	Sprint 3: Improvements

EZ-40	As a user, I want to be able to see my eligibility for each course (if missing prereqs, blocked entirely, already completed) when viewing a course list.	Story	Closed	Minor	Fixed	6/Apr/16 5:07 PM	Write a script to be used when courses displayed on any page (and give a description) based on the user's eligibility. Acceptance criteria: * Each of these courses, when hovered over, would display a tooltip showing the pre-requisites and co-requisites, and link to the appropriate course page. (Refer to Balsamiq mockups for more specifics.) Deliverable: script for custom object that makes color-coded, tooltip/linked eligibility status, wherever a course shows up in the course list. Additional notes: Each page, when rendering courses, should eventually make use of this script (make a Course object that calls display on itself)	User interface and functionality	Sprint 2: Major Functions, Sprint 3: Improvements

EZ-38	As a user, I want to be recommended a schedule for my degree, including electives to take based on my preferences/interests.	Story	Closed	Minor	Done	6/Apr/16 3:40 PM	Acceptance criteria: Suggested Schedule page * Based on selected degree type ** Connect to the DegreeView table for that user’s id, or if one doesn’t exist, generate a new one from requirements * List all the courses needed to be taken within the given degree ** Connect to CourseRequirement ** In first list, show only ones that don’t contain “elective” in the description (use “description” field rather than list of courses, for elective requirements) * List of electives based off the user’s interests ** Connect to User table (query) (interest field) and return courses based off those keywords (List 3 electives per keyword (potential 27 credits)) *** Connect to Courses and select the first 3 courses with the ‘keyword’ in their description, title, or course code *** If keyword wasn’t found, return an “error” message instead * Within both those lists: each course has a “remove” button to remove it from the schedule, and links to corresponding entry in course browser	Course Selection	Sprint 3: Improvements

6.1.3 Non-Functional Requirement Specification

6.1.3.1 Non-Functional User Stories

Sprint 1

Key	Summary	Issue Type	Status	Priority	Resolution	Resolved	Description	Epic Link	Sprint

EZ-30	As a developer, I want to know the database design in order to have a clear direction in systems implementation.	Story	Closed	Major	Done	17/02/2016 21:09	Deliverable: UML diagram of all relations and fields to be used in the database. This will be a document detailing the functional design of our database. This will have to be completed before implementation can be done. Acceptance criteria: - A readable document list of all the tables and attributes. - Documentation must contain: > A list of domain assumptions. > UML diagram. > An executive summary. - The design must be complete and implementable.	Database	Sprint 1
EZ-31	As a developer, I need to parse the course info from plain HTML on the website into its individual fields, so they can later be added to the database.	Story	Validate	Critical	Fixed	18/02/2016 2:19	Acceptance criteria: complete course list (according to UBCO's non-login-protected course browser website) has been parsed into fields as specified by the database design. Deliverable: text file displaying the records (each course, in tab-separated components) line-by-line	Database	Sprint 1, Sprint 2: Major Functions
EZ-61	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Story	Validate	Major	Done	18/02/2016 2:20	Acceptance criteia: complete the code academy javascript course up until and including unit 5 Deliverable: screenshot of completion		Sprint 1, Sprint 2: Major Functions
EZ-28	As a developer, I need a way to be able to store all the information pertaining to a single user efficiently so that it may be accessed and updated quickly.	Story	Closed	Major	Fixed	18/02/2016 12:10	Acceptance criteria: Database has been created in MySQL on the school server with temporary data in order to test the database, ensuring that it is working as expected before being populated with real data using the crawler. Deliverable: -A document containing a relational schema constructed from UML diagram written using SQL DDL	Database	Sprint 1

EZ-62	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Story	Closed	Major	Fixed	22/02/2016 12:12	Acceptance criteia: complete the code academy javascript course up until and including unit 5 Deliverable: screenshot of completion		Sprint 1
EZ-63	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Story	Closed	Major	Fixed	22/02/2016 12:12	Acceptance criteia: complete the code academy javascript course up until and including unit 5 Deliverable: screenshot of completion		Sprint 1
EZ-64	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Story	Closed	Major	Fixed	22/02/2016 12:12	Acceptance criteia: complete the code academy javascript course up until and including unit 5 Deliverable: screenshot of completion		Sprint 1
EZ-65	As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Story	Closed	Major	Fixed	22/02/2016 12:12	Acceptance criteia: complete the code academy javascript course up until and including unit 5 Deliverable: screenshot of completion		Sprint 1

Sprint 2

Key	Summary	Issue Type	Status	Priority	Resolution	Resolved	Description	Epic Link	Sprint
EZ-66	As a developer, I want the fields taken from the UBCO course website to populate the database, so that they can be queried from the EZ-Plan website.	Story	Closed	Major	Done	10/Mar/16 2:39 PM	Acceptance criteria: database has been fully populated with the fields parsed from the UBCO course website Deliverable: screen or text of output (from SQL query in software used to access database) of all fields related to UBCO courses, according to the database design	Database	Sprint 1, Sprint 2: Major Functions
EZ-67	As a developer I need to have the database framework for the User Profile page in place so I can add all the other features.	Story	Closed	Critical	Fixed	16/Mar/16 4:07 AM	This will be the framework for the DegreView (DV) page. Acceptance Criteria: • Framework for DV page. • Add basic intractability with the database and the DV page. • Done ASAP	User interface and functionality	Sprint 2: Major Functions

EZ-68	As a developer i need to have the users information stored in the database when they sign up to be able to use it later on.	Story	Closed	Major	Fixed	10/Mar/16 10:09 AM	Functionality of storing the users information in database. Acceptance criteria • User information is successfully stored in database.	Database	Sprint 2: Major Functions
EZ-75	As developers, we need to be able to easily retrieve the requirements for a given major, so that course lists can be computed based on them.	Story	Closed	Major	Done	15/Mar/16 8:29 PM	Add additional functions to the crawler to read and save degree program requirements (credits and courses). Acceptance criteria (REVISED 02/Mar/16): • Find, and leave a comment or work log on this issue describing, where the program requirements for each major would be found. • Define the list of tuples relevant to at least the B.Sc. Computer Science major and add them to the database. • Screenshot of tuples present in database	Database	Sprint 2: Major Functions

Sprint 3

Key	Summary	Issue Type	Status	Priority	Resolution	Resolved	Description	Epic Link	Description

EZ-76	As developers, we need the database schema to be consistent with the functions on the website, so we can finalize the queries we use.	Bug	Closed	Blocker	Done	26/Mar/16 12:25 PM	Acceptance Criteria: Produce, and post on Confluence, revised UML and DDL to reflect the following changes: * UML in general should use the variable types actually in the database (eg. VARCHAR(10), not String) * User table: ** User ID must auto-increment (as primary key) ** Email must be unique ** Convert Interest table to a field (comma-separated list--use string methods, explode and such) * Schema must include CourseRequirement and DegreeType tables ** (note, DDL for those two and the Course table are already complete in the Crawler project on Github) * Post the UML diagram on Confluence (edit existing page) and underneath the initial design in the Project Documentation Post the DDL on Confluence (edit existing page) Let the group know when finished so queries can be updated Perform the alter table (or drop and re-create) updates in Ileri's 304 database	Database	Sprint 3: Improvements

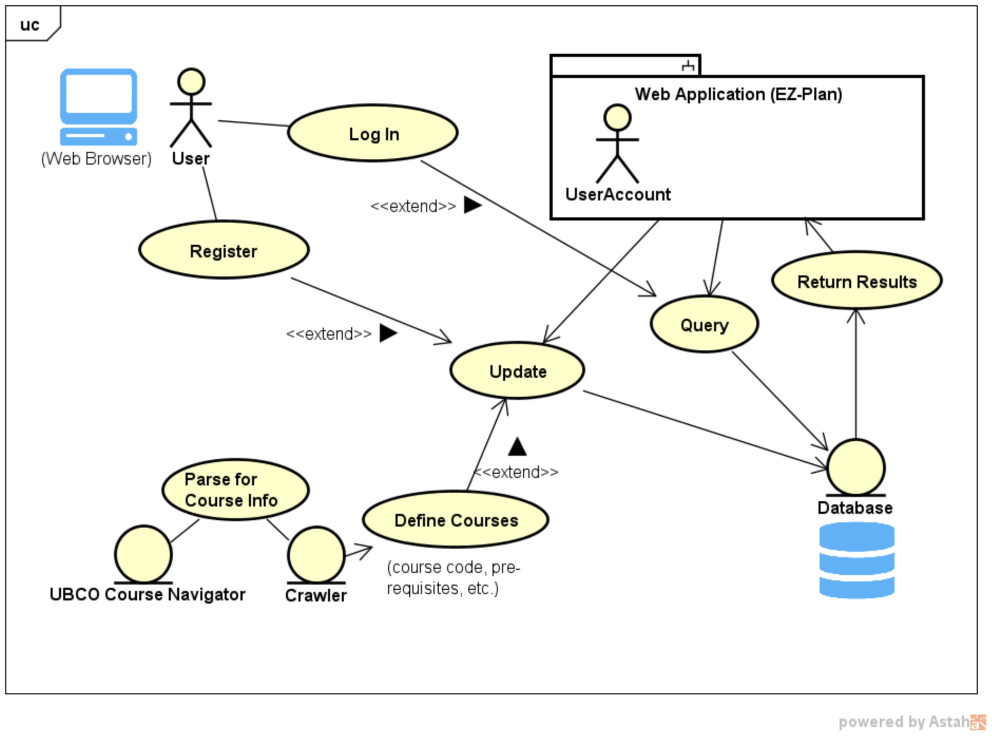
EZ-49	As developers, we want the rudimentary page template to have a consistent layout and visual design, as well as a site logo, so the user can recognize and find things on the site more easily.	Story	Closed	Minor	Done	6/Apr/16 4:45 PM	Acceptance criteria: * CSS style for all pages to use * Site logo * Header PHP file that contains site menu, logo, etc. * Footer PHP file that contains disclaimers etc. and closes content body ** Note to other group members: both header and footer should be included in other pages' PHP files	User interface and functionality	Sprint 3: Improvements

---

6.1.3.2 Architecture, Components, and Layers

Application Architecture

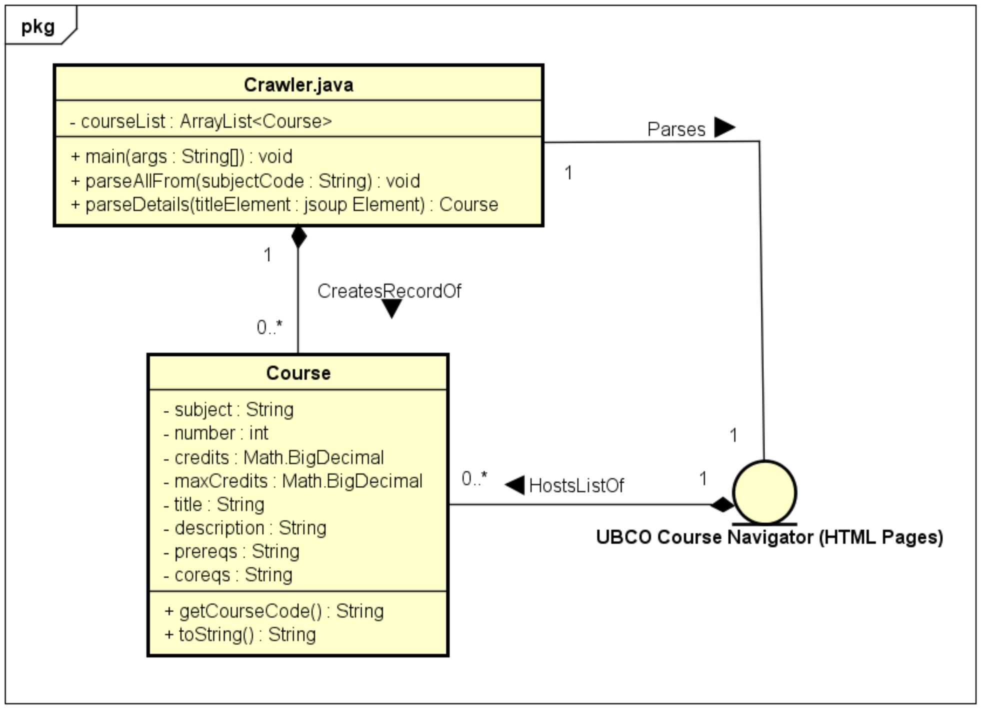
The high-level architecture for the system is fairly straightforward, The user connects to a web application, and interacts with it via a user account (with further detail in the user scenario sketches linked to in the Functionality section below), and in the course of various site functions, the application connects to the database for various queries and updates, which may return results that pages in the application can use for calculations.



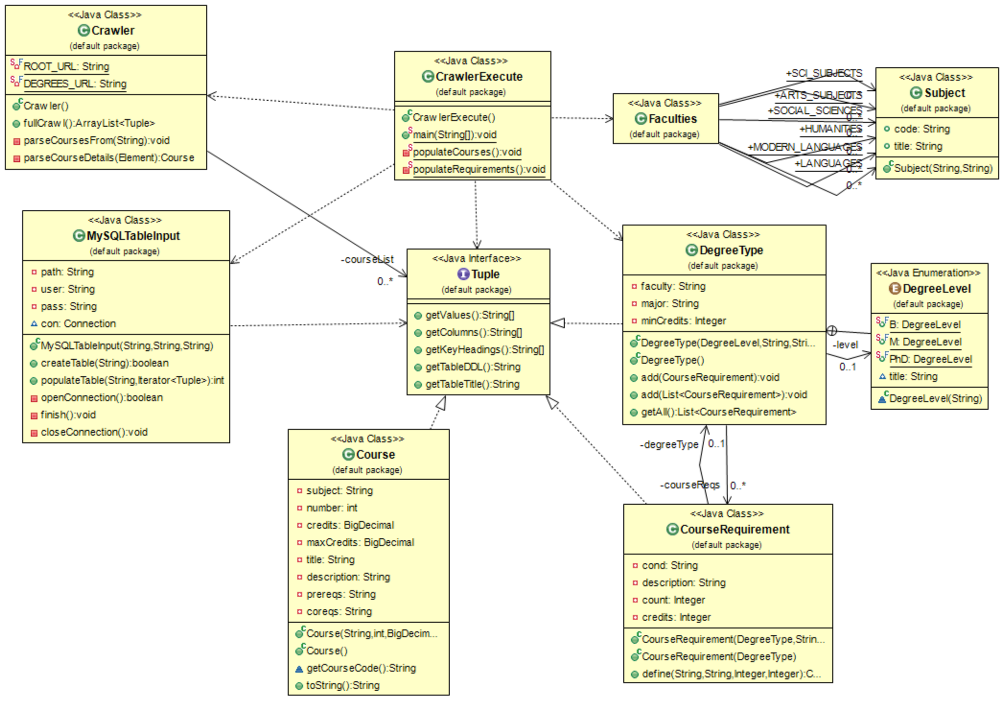
The web application (separate from the crawler) follows a ***component-based*** architectural style (reference), which emphasizes keeping each concern or task independent of others and can be thought of as "gluing together" separate components. Each page encapsulates its own functionality and uses the web browser and its functions (such as hyperlinks) as an interface for connecting them. The site's pages usually do not require use of other pages (they are, for the most part, atomic). Some code was reused between pages, but not in a manner that creates dependencies. A downside of our approach to this method, however, is that there was a high risk of inconsistency across the site, and object-oriented principles were not applied very much on the web side.

Crawler Architecture

The crawler itself is a simple Java program (making use of the external jsoup library) that, as of the end of Sprint 1, created course objects with which to populate the Course table in the database.


pkg
Crawler.java
- courseList : ArrayList<Course>
+ main(args : String[]) : void
+ parseAllFrom(subjectCode : String) : void
+ parseDetails(titleElement : jsoup Element) : Course
Parses
1
1
CreatesRecordOf
0..*
Course
- subject : String
- number : int
- credits : Math.BigDecimal
- maxCredits : Math.BigDecimal
- title : String
- description : String
- prereqs : String
- coreqs : String
+ getCourseCode() : String
+ toString() : String
0..*
HostsListOf
1
1
UBCO Course Navigator (HTML Pages)

This was expanded and cleaned up in Sprint 2 to allow degrees and requirements to be added, though they had to be input manually. The design was also modified to make better use of interfaces, for the purpose of adding entries to the database.



(Note, dependencies with unspecified type (dotted line with half-arrow) are aggregation *relationships despite the appearance of the diagram, as the aggregation label was not supported.)*

Functionality

The website contains most of the user-visible functionality of this project. At first, a series of sketches corresponding to a user "storyboard" were produced early in sprint planning, giving a rough outline of the functions available in the website.

Scans of these sketches are available in PDF format in the Confluence space's Site Flow - User Scenario file list.

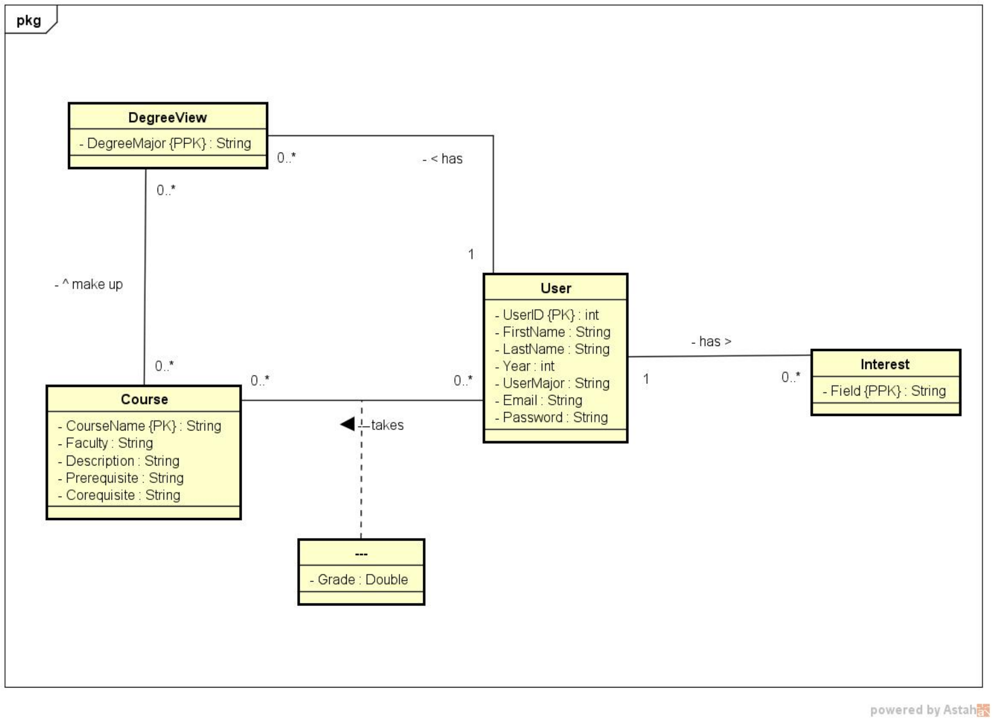
By our third sprint, we defined the major functions of the site and what links there would be to the database in implementation:

User Login/Registration	**Edit Info**	**Course Browser**	**Home Page**	**Compare Degrees**	**Suggested Sched**
Fields for user to enter: • email • password • name (registration) • major and year level (registration)	Fields for user to update: • name • password • major and year level • "interests" tags	Allows finding coursesbased on name or description Search results show: • course code • course title • credits offered	Displays: • user's name, major, year level • brief explanation of what the site is and how it works • overview of progress (credits taken and needed) • links to edit info or schedule	Selection of three majors to compare Overview of: • degree name • required credits • current eligible credits • specific required courses	Displays the plan a progress for a user • currently planr courses • required cours • electives, base interests • save or delete as appropriate
Tables Used: • User (query) **Pages Involved:** • index.php • login.php • registerpage.php • registerpage2.php • logout.php	**Tables Used:** • User (query, update) **Pages Involved:** • my_info_edit.php	**Tables Used:** • Course (query) **Pages Involved:** • course_browser.php • searchresults.php	**Tables Used:** • User (query) • UserCourse (query) • DegreeType (query) **Pages Involved:** • home_page.php	**Tables Used:** • DegreeType (query) • CourseRequirement (query) • UserCourse (query) • Course (query) **Pages Involved:** • compare_page.php • none.php • test1.php, test2.php (placeholders)	**Tables Used:** • User (query) • DegreeType (c • DegreeView (c update) • Course (query • CourseRequire (query) **Pages Involved:** • suggested_scl

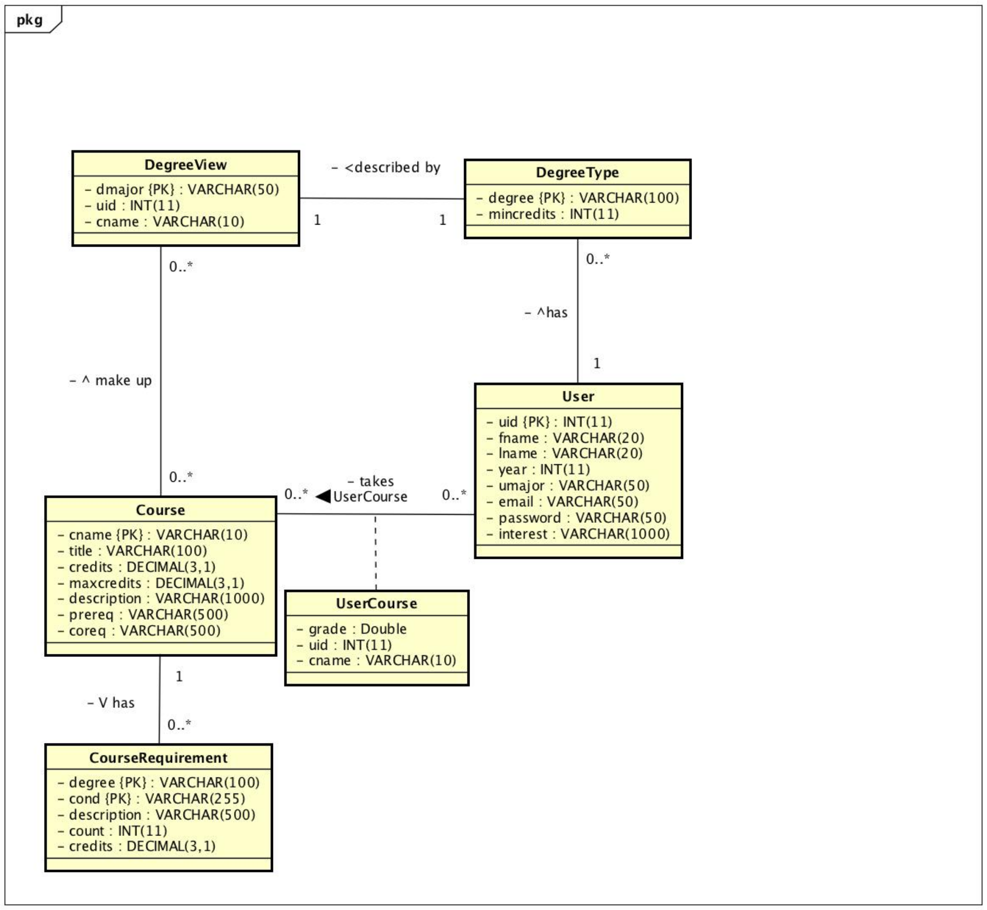
(Additionally, the site uses completely custom CSS styles, logo, menu bar/header, and footer with copyright information. A connection script, since it is required on every page, is also used by each of the above components.)

Database Architecture

The database is the second major component of the architecture, and represents various data objects the user will need to interact with. As of the end of Sprint 1, the database design accounts for user profiles, degree views, and course information as provided by a crawler (with detail provided in the Crawler class diagram above).



This design was revised during Sprint 2 and used for the remainder of development.



6.1.3.3 Development Environments

Eclipse: Java IDE used to create the web crawler.

NetBeans: HTML/CSS/JS/PHP IDE used to create web page documents.

Astah: UML diagram editor.

6.1.3.4 Builds

As this is a web application, builds are done manually by copying the page documents into a host machine, which acts as a local server. Procedures for a more persistent build have not yet been decided.

6.1.3.5 Installation

As EZ-Plan is implemented as a web site, in theory (once it is hosted) there is no installation needed, and it can be accessed directly from any device with a web browser.

6.1.3.6 Maintenance

Source code is located in the following GitHub organization and its repositories:

https://github.com/COSC310-EZ-Plan

Plans for maintenance (after the project term) have yet to be decided.

6.2 Agile Development Planning

6.2.1 Sprint Planning Summary

Sprint 0

Goals

- Planning our project development process
- Defining all the user stories for our product backlog
- Assigning tasks to team members for sprint 1 and deciding on time estimates

Our Sprint 0 was essentially planning out our project development process. We created our product backlog and partitioned them into their designated sprints as well as discussed time estimates for the items in sprint 1 and who was to be assigned which task. We also began creating an outline for what elements need to be on the website which can be found here. We had also created a use case for what a user would do if they were interacting with the website which can be found here.

User Stories and Burndown chart

As we were still unfamiliar with the Agile process this early on, we had not planned to track Sprint 0 as a list of user stories on JIRA. Instead, we ended up creating our entire product backlog, which was displayed earlier in this document. Accordingly, as a result of having no user stories for Sprint 0, we did not have a burndown chart; many of the Sprint 0 tasks were completed in lab sessions and an additional 3-hour meeting.

Sprint 1

Theme/Goals

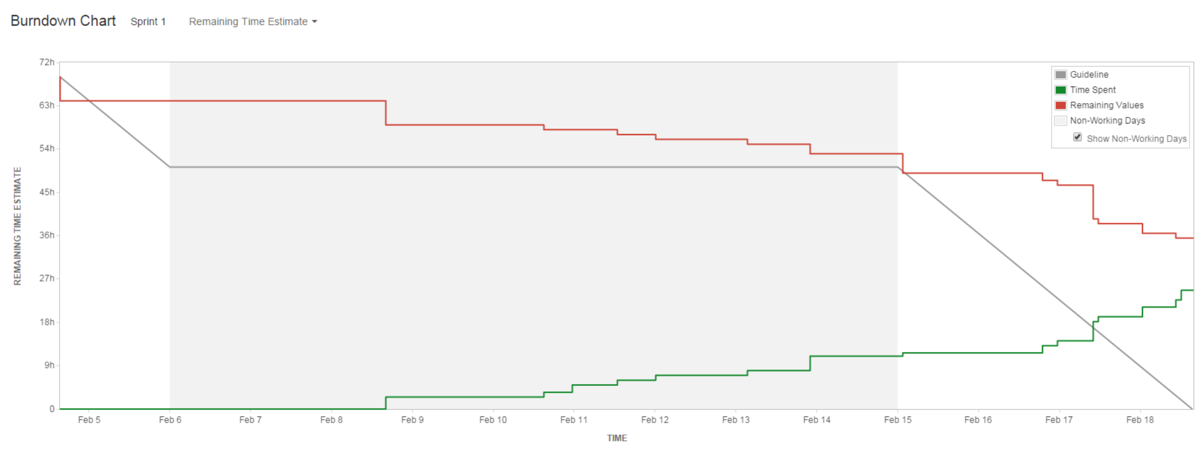
- From user perspective: Proof-of-concept and preliminary design (and feasible), and placeholders for intended features (show mock-ups)
- From developer perspective: Understanding what the user's functional requirements would be, creating designs for the database and site that would meet these goals in further implementation
- Template is intended to be proof-of concept; sprint 1 generally covered non-functional requirements and matches the theme of showing the "skeleton" and preliminary design of the project

User Stories

Summary	Assignee	Estimated (hrs)	Spent (hrs,mins)	Remaining (hrs,mins)	Status
As a developer, I want to know the database design in order to have a clear direction in systems implementation	Ileri Oyedele	4	5	0	Completed

As a developer, I want to know the database design in order to have a clear direction in systems implementation. (Sub-task)	Arturo Padilla	4	0	4	Completed (off main)
As a developer, I need a way to be able to store all the information pertaining to a single user efficiently so that it may be accessed and updated quickly.	Arturo Padilla	2	4,30	0	completed
As a developer, I need a way to be able to store all the information pertaining to a single user efficiently so that it may be accessed and updated quickly. (Sub-task)	Ileri Oyedele	2	0	2	Completed (off main)
As a developer, I need to parse the course info from plain HTML on the website into its individual fields, so they can later be added to the database.	Eliana Wardle	8	5	3	Completed
As a developer, I want the fields taken from the UBCO course website to populate the database, so that they can be queried from the EZ-Plan website.	Arturo Padilla	3	0	3	Incomplete/moved to later sprint
As a developer, I want to have a paper markup in order to have a clear idea of direction in implementation.	Kevin Semasinghe	4	1	3	Completed
As a developer, I want to have a paper markup in order to have a clear idea of direction in implementation. (sub-task)	Lam Ng	4	0,40	3,20	Completed
As developers, we need to code the template for the site's pages so that it can display consistently.	Lam Ng	4	0	0	incomplete
As developers, we need to code the template for the site's pages so that it can display consistently. (sub-task)	Kevin Semasinghe	4	3,30	0,30	completed
As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Eliana Wardle	5	4	1	completed
As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Kevin Semasinghe	5	1	4	incomplete
As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Ileri Oyedele	5	0	5	incomplete
As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Arturo Padilla	5	0	5	incomplete
As a developer, I need to learn how to code using Javascript so that I can create components for the website (such as coding database queries) and understand those of teammates.	Lam Ng	5	0	5	incomplete
Total time	N/A	64	28,40	35,20	N/A

Burndown Chart



The total estimate ended up being too high by itself, as we assumed it would be better to overestimate "just in case" rather than run into trouble and have to increase the estimate. However, especially for paper prototypes, putting enough time into the planning before the beginning of the sprint made the task itself trivial. In the case of the web crawler (for which the estimate was already too high due to a typo not being noticed before the sprint began), finding a third-party library designed for parsing HTML made the task much easier, as well.

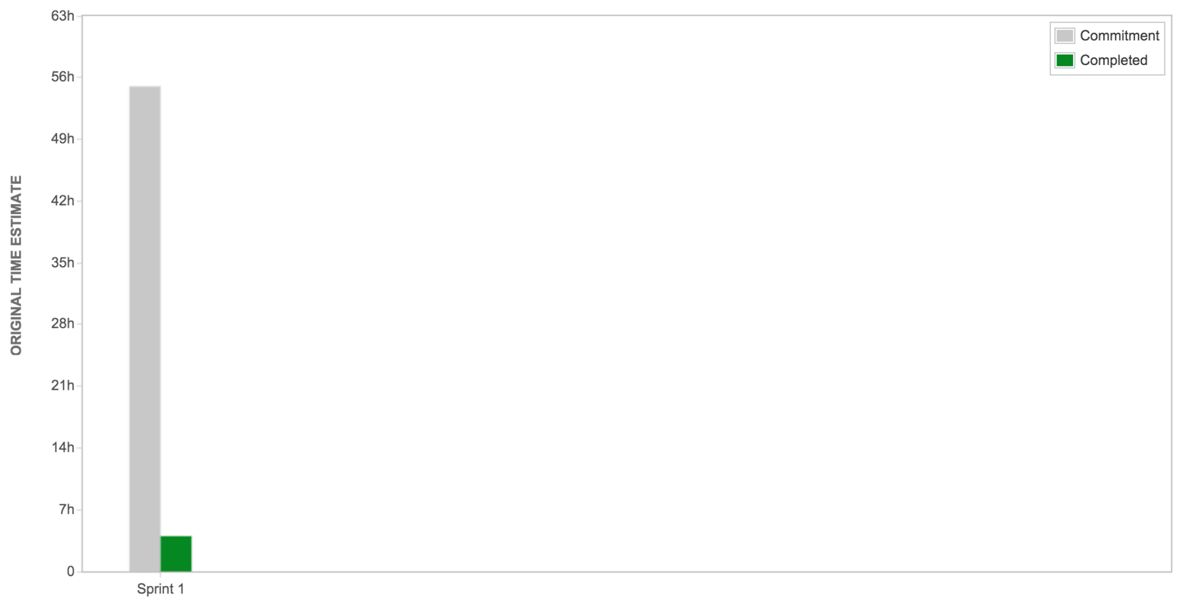
Aside from that, most of the unused time was due to confusion surrounding the Javascript tutorials, which most of us did not complete, as after one member went through it and described its subject matter, we doubted whether it would provide any new or useful information to the developers. Since we decided to code the crawler using Java, the need for JavaScript became less important except for functionalities within our website; we felt that coming from a Java background, the major difference would be syntax, most of which was very similar to Java (which we were all familiar with), leaving the specifics of database querying, which are not suited for JavaScript. Accordingly, we would attempt to avoid tasks such as this in the future, or at least more closely examine the contents before committing to a tutorial.

Another task that was not completed was populating the database; this was primarily because we were not sure which server location to put the information on.

The timeline is skewed towards the end due to lack of commitment: reading break was a large distraction, essentially leaving 3 working days to complete most of the tasks, and leaving others unfinished; for example, the site templates still needed to be completed by the end of this sprint.

(Refer to this sprint's Retrospectives and Conclusions sections for more details.)

Velocity and Hours



The total estimated time requirement for Sprint 1 was 55 hours, which seems erroneous, as it does not agree with the numbers we have down for our sprint 1 based off the table. According to the diagram, only 4 hours were spent, mainly because we did not close tasks as they were finished (discussed further in the Conclusions section for this sprint). We used hours as fine detail rather than story points mainly because of our unfamiliarity with the Agile Process and how story points work; hours were a more familiar measure to us, and it seemed reasonable and logical to estimate task size based on them. These estimates were determined by a planning poker process (described in the glossary for this document) for each task. The actual velocity for Sprint 1 is 2.84 hours/day calculated from our user story breakdown for sprint 1. Time estimation is an issue that needs to be improved and is addressed in the Retrospectives section.

Hours Spent:

	Start (Estimate)	End (Actual)
Velocity Chart	55h	4h
Backlog (Work Logged)	64h	24h 40m

Risk Tracking

Risk tracking was not available for this sprint. During the planning stages, we were not aware that this needed to be determined for each task, and only had completed fields fields available in JIRA and those requested of us in person.

Retrospectives

What we did well	Lessons Learned
• Made sure our backlog was ready before the start of sprint (as far as we were aware at the time). • Had small scrum meetings consistently. • Took initiative in asking teammates for feedback and warning each other of complications.	• Do planning meeting earlier and more often, to avoid the burnout of one long meeting. • Stick to the I-N-V-E-S-T guideline for making user stories. • Gather requirements from users rather than from developers. • Do risk tracking. • Make more honest estimates that reflect reality rather than try to overestimate. • Set due dates on issues. • Be more committed (and work more consistently throughout the sprint). • Ensure everybody is working at any given time. • Validate and finalize issues. • Make sure everyone is aware of their responsibilities and any changes to them.

Scrum Meeting Notes (Samples)

Scrum + Work Meeting - Thursday, Feb. 4, 2016

Location: SCI 234 (lab room)

Duration: ~2 hours

Present: Arturo, Eliana, Ileri, Kevin, Lam

- Changed "official" Thursday scrum time to start of lab, rather than half an hour before, as Lam and Arturo can't arrive earlier
- Reviewed edits/recommendations made by James in order to have sprint 1 backlog items in the proper form (including specific acceptance criteria)
  - Titles should be names of user stories; description should be the details, and deliverable with acceptance criteria
    - which components are involved, what format to use, how to access, include exceptional cases... step-by-step/"unit test"
    - cannot determine that "it's done" until these are met
  - When a component involves multiple developers working on it, should split the story into copies (and specify which member does what)
- Confirmed who selected which issues to tackle, and each member added acceptance criteria to them
- Essentially, "themes" of our sprints are:
  - Sprint 1: Groundwork and Setup
  - Sprint 2: Critical Functions
  - Sprint 3: Functional Improvements
- Began sprint 1

Scrum - Monday, Feb. 15, 2016

Location: COSC 310 lecture hall

Present: Arturo, Eliana, Kevin, Lam

- Ileri was not able to make it to class, but during reading break, had already let everyone know (via Facebook message) that a preliminary design for the database structure was ready for review, and several group members commented on it
- Lam had done some work at the end of reading break towards prototyping the website pages, and Kevin finished his; they will confer to look it over among themselves a bit more before publishing it
- Arturo had looked over the database design proposed by Ileri and also made some comments
- Eliana had gone through the Javascript tutorial assigned to the group and (also via Facebook message during reading week) recommended doing just the later ones as they are more relevant; commented on Ileri's database design; did some preliminary background research for the crawler (things like processing webpage data); decided that, as the crawler only populates the database and need not be part of the website, may as well make it a Java executable instead for a shallower learning curve
- For Thursday, intend to have the sprint work complete so as to be able to start working on the team evaluation and other release-related materials during lab (and start working on individual assessments when sprint work is done)

Conclusions

Overall this sprint did not go as well as it could have, but this was to be expected given that it was our first official sprint and that it was during our reading break. We managed to get a majority of our tasks done, most importantly the critical ones that were required in order for us to move onto items in later sprints. We have a solid database design and a good idea of what the website should be like, although we need to extend this to asking students from other faculties for their recommendations and opinions.

Planning, and performance in general, could be streamlined by being more self-motivated and doing additional planning on our own time rather than limiting it to (relatively infrequent) group meetings. Doing work nearer the beginning of the sprint would also help to avoid leaving things undone and ensure that problems are detected early.

In a general sense, our time estimation was abysmal for this sprint with several tasks being done well before the estimated time and some taking longer. We overestimated the time required due to our own tendencies to procrastinate; this seemed like good forethought, but in retrospect it gave us a mindset of having more time than was actually available.

Furthermore, we need to do a better job writing out user stories. We must remember to set due dates for them when we start the sprint, make them as independent from each other as possible (so that a blocker with late completion does not impede other teammates' ability to work). We also need to start start validating and closing issues once they are done, both for the sake of communication and for accurate reports (such as the velocity chart). This entails meeting the acceptance criteria so that we have a clear indication of what is complete and what is not, and creating them to be representative of the requirements, so that everyone is on the same page and can avoid confusion regarding what needs to be done. As we now have some experience with using JIRA as a project management tool, we should be able to plan future sprints better in advance and avoid pitfalls such as stories that are too large in scope to estimate, co-dependent tasks, and vague acceptance criteria.

Sprint 2

Theme/Goals

- From user perspective: Performs the most basic advertised functions (in isolation; features such as registration or viewing courses).
- From developer perspective: Implement pages and include real data so that the site's basic functions can be performed.
- Generally moved more towards "front-end" work, as well as the logic and data to make the underlying functions work

User Stories

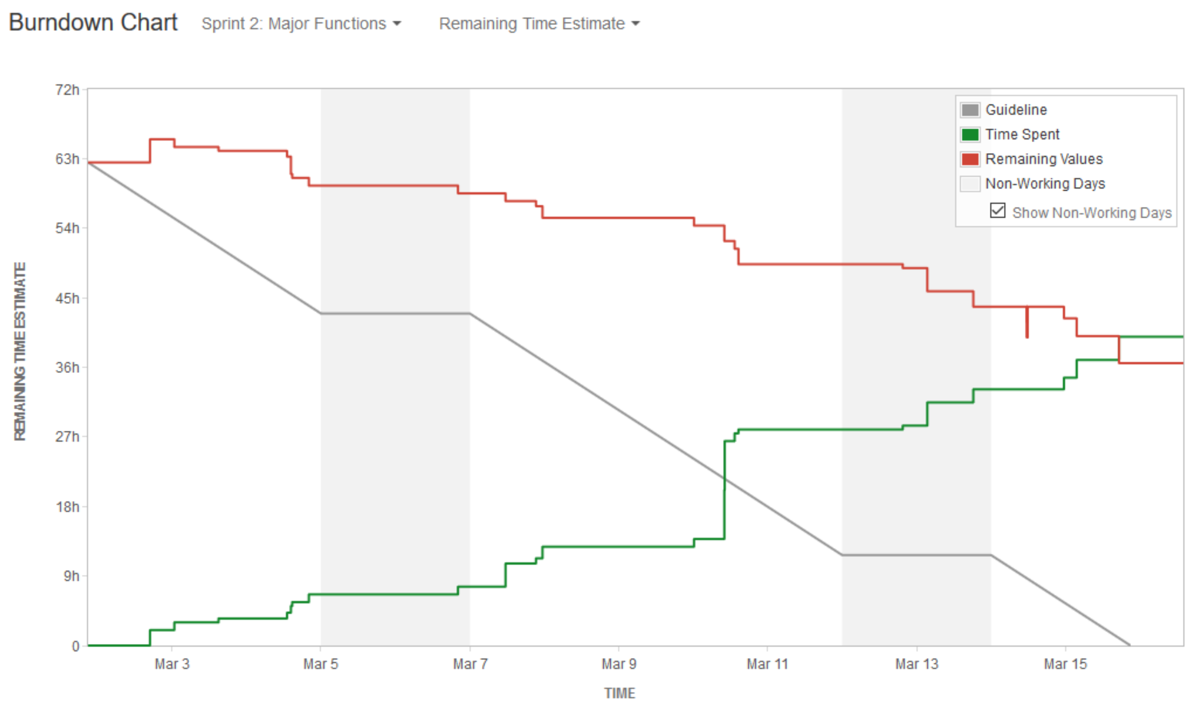
Summary	Assignee	Estimated (hrs)	Spent (hrs,mins)	Remaining (hrs,mins)	Status
As developers, we need to be able to easily retrieve the requirements for a given major, so that course lists can be computed based on them.	Eliana Wardle	3	5	7,30	Closed
As a developer, the website must be able to compare degrees using a breakdown	Kevin Semasinghe	6	2,10	3,50	Open (moved to sprint 3)
As a developer, we need to make sure the website can display a breakdown of a given degree to the user(Sub-Task)	Kevin Semasinghe	2	0	2	Open (moved to sprint 3)
As a user, I want to see and edit my current progress in my degree, so that it can be used to find what else I need to take.	Ileri Oyedele	5	0	5	Open (moved to sprint 3)
As a user, I need to be able to update my info, so that EZ-plan can incorporate those changes into the plan later.	Lam Ng	3	2,42	0,18	Open (moved to sprint 3)
As a developer, we need to ensure that the user can see their credit requirements for a given degree on the website	Kevin Semasinghe	5	1,30	3,30	Open (moved to sprint 3)
As a developer i need to have the users information stored in the database when they sign up to be able to use it later on.	Ileri Oyedele	2	12,40	0	Closed
As a developer I need to have the database framework for the User Profile page in place so I can add all the other features.	Ileri Oyedele	3	3	0	Closed
As a developer, I want the fields taken from the UBCO course website to populate the database, so that they can be queried from the EZ-Plan website.	Eliana Wardle	3	3,30	0	Closed
EZ-48 As developers, we need to code the template for the site's pages so that it can display consistently.	Lam Ng	4	2,37	0	Closed
As a user, I'd like to be able to browse courses by subject area (brief summary--course number, title, credits), and be referred to the full description if wanted, or schedule it (or mark as interest) outright.	Arturo Padilla	3	2	1	In Progress (moved to sprint 3)
As a user still deciding between options, I want to see a side-by-side comparison of potential degree programs or paths or schedules, to help me decide which I would prefer to take.	Lam Ng	4	0	4	Closed(Duplicate)
As a user, I want to be able to see my eligibility for each course (if missing prereqs, blocked entirely, already completed) when viewing a course list.	Ileri Oyedele	2	0	2	Open (moved to sprint 3)

As a user, I want to see the progress I've made in my degree (whenever viewing a breakdown--credit requirements and course requirements)	Arturo Padilla	4	0	4	Closed(Duplicate)
As a user, I want to be able to get a more detailed visualization of what courses are required for my major, or what the more specific graduation requirements are.	Arturo Padilla	4	0	4	Closed(Duplicate)
As a user, I want to see how many credits I need (each year) for my major.	Lam Ng	4	2,18	1,42	Closed(Won't fix)
As a user, I need to be able to update my info (in case I pass courses or change degrees or something like that), so that EZ-plan can incorporate those changes into the plan later.	Ileri Oyedele	2	0	2	Closed
Total time*	N/A	64 (59(+4))**	43,27	35,32	N/A

*Estimates (+/- a few mins)

**One task in the backlog was a subtask of another task from an earlier sprint; as a result, 5 hours was added to the estimate even though the initial task was completed.

Burndown Chart



Our initial time estimate was slightly lower than it needed to be as seen in the initial bump a few days in. This is due to the fact that we did not have enough time to estimate the time for each task in our backlog using the planning poker method. As a result, each person assigned a 'guesstimate' time to their task and this caused some initial updating as some tasks took longer than originally anticipated.

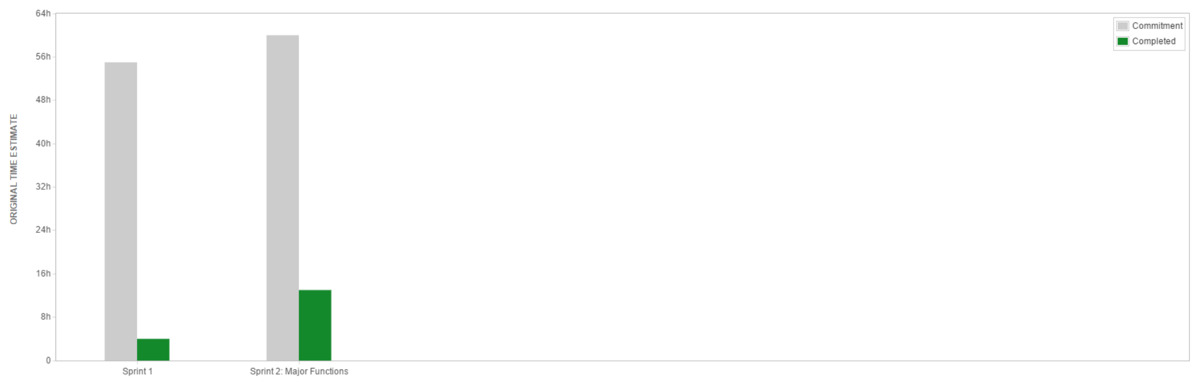
Apart from this, we did display some consistent work log compared to the last sprint. Although, since we reduced the scope of our project fairly early, once again our original time estimates were no longer valid as we would need to do less work for most of our tasks. This is why the few flatlines occur as we stopped working on one task and moved to the other. Due to the change in scope, these tasks were completed in steps which can be seen clearly in our burn-down. Further attributing to the flat-lines is the issue we had of not being able to query from our database effectively. This stopped a lot of progress and made us create dummy variables which do not take as much time to do as querying actual data would.

Due to the major issue of not being able to query from our database, a lot of tasks could not be completed in accordance with their acceptance criteria and we hit a point where no more considerable work could be done for any of our tasks. We focused more on trying to fix the issues since nearly all of our work was stopped or moved to more trivial changes such as updating templates or getting side function and features to work with some of the pages.

Overall, our lack of commitment in the planning phase created many problems going into this sprint that affected our ability to log work consistently.

Velocity and Hours

Velocity Chart



Once again, the shape of the velocity chart was not accurate to the amount of work completed: in this sprint, complications within tasks arose that made them take much longer to complete, and some were also left in the validation stage as the sprint concluded. Most of the discrepancy was due to underestimating time, so the number of tasks done was much lower than we had anticipated, although it was improved over our first sprint.

Hours were used as a time estimate again, being a familiar measure to us and consistent with our measurement in the first sprint. Some of these estimates were determined by a planning poker process (described in the glossary for this document), but as we did not have a long enough meeting to complete it for the entire backlog, many estimates were simply guessed by the people who would work on them and had to be adjusted as time went on. Team commitment was a major cause of this failure, and we intend to improve it for our final sprint.

Hours Spent:

	Start (Estimate)	**End (Actual)**
Velocity Chart	63h	13h
Backlog (Work Logged)	63h	43h 27m

Risk Tracking

Risk tracking was not performed in this sprint.

Retrospectives

What we did well	**Lessons Learned**
• Discovered problems early in the sprint, giving us time to adjust and create a new plan (even limiting the project to a more realistic scope). • Balanced teamwork and individual work; tasks were less dependent on each other but we still kept in communication while working. • Learned what we needed as we went (in terms of implementation). • Regardless of issues of scale towards the end of the sprint, there was a more consistent work log compared to last sprint	• Take more time to research and understand the problem before working on it (limit assumptions). • Make the effort to produce good time estimates and acceptance criteria (schedule made it difficult to meet for planning poker, etc.) • Keep having scrum meetings consistently, and keep them concise. • Be self-motivated; commit to the project (even in the middle of 4th-year course load). • Work on “high-level” aspects more (get above the coding to see how the pages actually interconnect) • User stories still need more work. A lot of duplicate or non-issue user stories were found after the sprint • Need to work on planning better to avoid potential issues coming up midway through the sprint that halt progress. • Meetings need to have a set end time as to not waste time going over things that do not matter as much or to prevent us from dwelling on a topic for too long • Time estimates need to be within AGILE guidelines

Scrum Meeting Notes (Samples)

Scrum Meeting - Wednesday, March 9, 2016

Location: Library lobby

Present: Eliana, Ileri, Lam

- Arturo and Kevin were not present at class today or on Monday, and were not able to attend the scrum; work was logged on JIRA, though
- Eliana has been working on the course crawler and has succeeded in getting some of the courses into a database
  - Tried to do so in a way extensible to requirements, but hasn't had a chance to code the latter yet
  - Noted some necessary changes to database design (posted on Confluence on the Database DDL).
- Ileri had a lot to do this week for other courses which has taken up a lot of time, but otherwise has not run into problems with the tasks themselves
  - May be able to make the changes to database UML to account for degree/program requirements and course data
- Lam has been working on the site page implementations
  - Since the degree requirements, eg. number of credits, aren't yet available, will need to use dummy data for now; hopefully by Thursday after lab, a script that runs a query and returns the needed values can be made

Scrum + Work Meeting - Thursday, March 3, 2016

Location: SCI 234

Present: Arturo, Eliana, Ileri, Kevin, Lam

Time: ~2 hours

- Kevin:
  - has not started working yet (jobs: functionalities such as viewing degree breakdowns, including querying database, and comparison feature on website
  - No problems or questions so far in that regard
- Lam:
  - Working on templates (finished registration, next working on course schedule suggestion view) from previous sprint (spent an hour, about one-third done)
  - No problems or questions in that regard
- Eliana:
  - Working on getting degree requirements into database-usable form
  - Ran into issues in that they're not in consistent places and the format is all over the place (some in nice tables, most are plain text)
  - Might be easier to hard-code it manually but that's still a lot of work
  - Database also needs to be able to support it (next task right now would be to at least get the course list on the database)
- Ileri:
  - Was going to do work on implementing registration page logic but it hadn't been done yet
  - Suggested we come up with standards (how we program the pages) so we're consistent with each other
    - Lam, so far, has been doing backend in PHP (as suggested from the start)
    - Getting most of the required knowledge from COSC 360 anyways
  - Suggested that index page should be registration, or redirect to degree breakdown if already logged in (cookie)
    - Kevin suggests we use dummy values in queries for now until the data is in the database
- Arturo:
  - Has been reviewing/relearning database basics and going over how to construct PHP pages

- Overall, too early in the sprint to comment on most possible issues
- Spent some time going over needed changes to the database design, possible solutions to the problem of inconsistent requirements
  - Results of discussion available on Confluence (Degree Requirements Specification and Usage)
  - (Made note of another step for Eliana, in the existing task for populating database with values, to set the faculty (eg. "Arts") for each course based on course code, as that was not available on the page directly)
- As far as project scope, we may need to restrict functionality to only one or a few degree paths and might not be able to "suggest" a degree (so might limit it to check that a certain course list suggested by user fulfills requirements)
  - Our "one in mind" for the demo is "just B.Sc. in Computer Science"

Conclusions

Overall, our sprint was on track initially, however all of the issues that were brought up throughout the sprint ended up forcing us to look back at our initial goals for this project and rework them in a way that we would be able to produce a still meaningful product in the end. As a whole, this sprint was not successful as it did achieve what we had initially planned as per the theme of the sprint; to produce a working product.

Lack of commitment with meetings and our inability to use our meeting time appropriately led to the failure of this sprint and we need to make improvements moving on if we want to produce a working product. We need to have set guidelines for our meetings and take time to produces better time estimates rather than having the individuals working on the task assign them. This is not in accordance with AGILE guidelines and we have already failed to focus on working software for the past two sprints. Despite some of the improvements we've made to be more consistent with our work log, the issues that come up through miscommunication and lack of planning undermine this greatly.

Compared to our previous sprint, we did do a better job creating acceptance criteria, setting due dates and closing issues once they met all the acceptance criteria. We also tried to limit the dependencies between tasks and keeping sub-tasks assigned to the same user the main task is assigned to. Although we tried our best to limit dependencies, a few did arise mainly due to the database needing to be updated before data could be displayed on the web-pages however unlike the last sprint, a fair amount of work could be done for each task before the dependency became an issue.

Sprint 3

Theme/Goals

- Theme: Communication/Launch Preparation
- From a user perspective, See a working product; the website performs all the advertised functions (can go all the way from registration through degree validation)
- From a developer perspective, Finalize and link together the site’s structure, scripts, and design

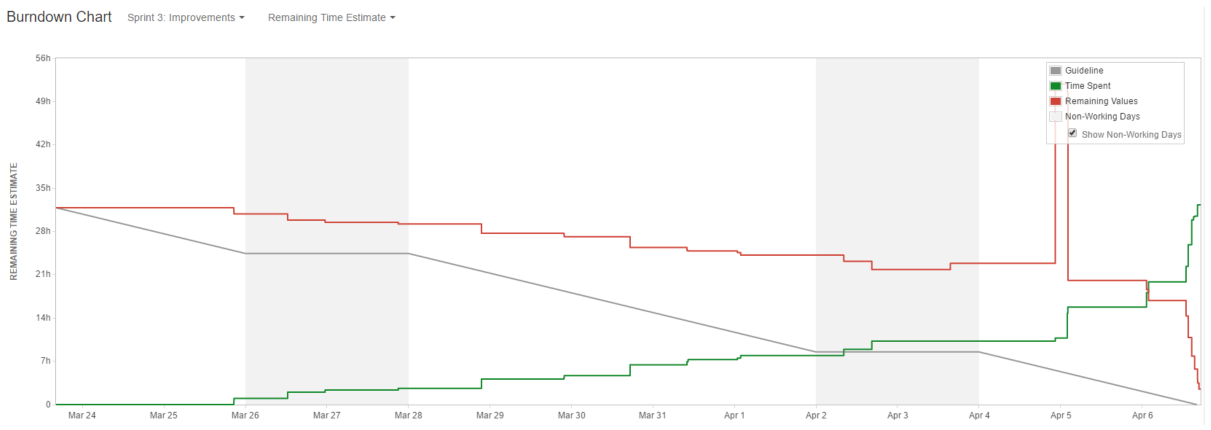
Generally:

- Rework tasks so that we can deliver a working product as intended

User Stories

Summary	Assignee	Estimated (hrs)	Spent (hrs,mins)	Remaining (hrs,mins)	Status
As developers, we need the database schema to be consistent with the functions on the website, so we can finalize the queries we use.	Arturo Padilla	1,30	1	0,30	Closed
As a user, I want to be able to compare my degree plan with another option, so that I can make a more informed decision.	Kevin Semasinghe	6	5,55	0	Closed
As a user, I want to see and edit my current progress in my degree, so that it can be used to find what else I need to take.	Ileri Oyedele	5	6,30	0	Closed
As a user, I need to be able to update my info, so that EZ-plan can incorporate those changes into the plan later.	Lam Ng	3	5,32	0	Closed
As a user, I want to be able to access a home page which shows me general info about my profile and degree	Kevin Semasinghe	5	4,40	0	Closed
As developers, we want the rudimentary page template to have a consistent layout and visual design, as well as a site logo, so the user can recognize and find things on the site more easily.	Lam Ng	4	2,19	0	Closed
As a user, I'd like to be able to browse courses by subject area (brief summary--course number, title, credits), and be referred to the full description if wanted, or schedule it (or mark as interest) outright.	Arturo Padilla	3	5	0	Closed
As a user, I want to see that the degree plan I have chosen will be valid for my degree type, so that I know it will be close to ready for registration.	Eliana Wardle	2,30	2,30	0	Closed
As a user, I want to be able to see my eligibility for each course (if missing prereqs, blocked entirely, already completed) when viewing a course list.	Ileri Oyedele	2	0	2	Closed
As a user, I want to be recommended a schedule for my degree, including electives to take based on my preferences/interests.	Eliana Wardle	4,30	7,15	0	Closed
Total Time	N/A	36,30	40,41	2,30	N/A

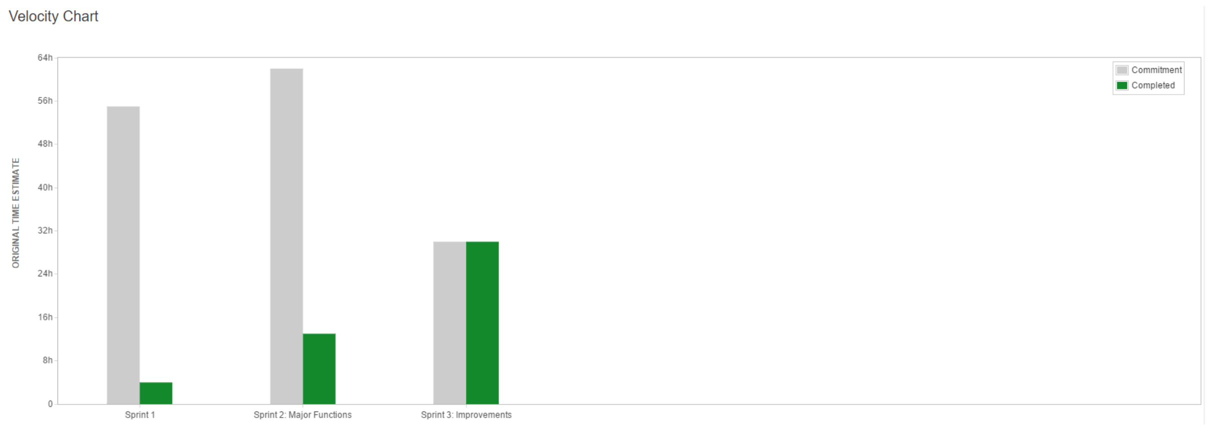
Burndown Chart



Work was logged fairly consistently for this sprint. There was a hold on work done for the first few days, as we were all waiting on EZ-76 (Blocker story) to be completed, so that our queries would be consistent with the database structure. After its completion, we did log work consistently, although once again, due to external reasons unrelated with the sprint, we did not have as much time as hoped and there was a lapse in work done. We do have a jump in time estimated towards the end due to one of us erroneously logging 30 hours of work instead of 30 minutes; to counteract the jump in time spent, we reduced the time estimate to its correct amount, but this did not change the curve retroactively.

We did complete a fairly large portion of the remaining work in the last few days, but we did manage to complete all but one task by the end of it. Overall, compared to previous sprints, we did not have large jumps throughout the sprint (except towards the end) meaning we had logged work much more consistently compared to our previous sprints, and managed to put in enough hours to finish more of the tasks that we had intended.

Velocity and Hours



Before our sprint began, we took the extra time to set it up appropriately, define more clearly what needed to be done for each task, and work on setting up proper time estimates; because of this, there was less ambiguity than in prior sprints, and we managed to complete nearly all our tasks. As a result, we actually had a reasonable velocity for this sprint, as seen in the chart. We managed a velocity of 3.183 hours/day for this sprint, which was enough to get all our important tasks done. It is much closer to an "ideal" 5 hours/day; however, given that this sprint was much smaller in comparison to our previous sprints, our velocity was more than enough to get things done. According to our backlog hours however, we had a velocity of 4.068 hours/day; the discrepancy is likely due to a few tasks logging more hours than their initial time estimates. Additionally, some auxiliary tasks which were in the product backlog were not assigned to this sprint and were not completed over the course of this project.

Hours Spent:

	Start (Estimate)	**End (Actual)**
Velocity Chart	31.83h	31.83h
Backlog (Work Logged)	36.5h	40.68h

Risk Tracking

Risk tracking was not performed in this sprint.

Retrospectives

What we did well	Lessons Learned
• Set realistic goals and achieved them • Logged work consistently, closed issues properly • Communication was good, everyone knew what they had to do and how to do it • No major issues compared to last sprint, planning was done well and issues were brought up well ahead of time • Time estimates were much better compared to previous sprints • User stories were set up and assigned much better than previous sprints with acceptance criteria and due dates set • Important user stories were completed on time	• Time management is harder than it seems (balance different academic commitments) • Overall commitment was better but was still lacking when it came to scrum meetings • Finally managed to get a better understanding of GitHub workflow

Scrum Meeting Notes (Samples)

Scrum + Planning Meeting - Thursday, March 31, 2016

Location: SCI 234

Present: Kevin, Arturo, Ileri, Eliana, Lam

- Kevin:
  - did listDeg.php page; currently just prints "working" to see that it can be referenced;
  - added some AJAX functions but not sure if they work yet (still in learning stages)
  - updated template for homepage, query statements set up, but need to test it (make sure database can be accessed) (can do as soon as he has SQuirreL or other access software set up)
  - Concerns: need the session variables to include User.uid (because it's used in almost every single other table) - single line of code change in index.php
- Eliana:
  - 2/3 of the way through displaying part of Suggested Schedule page
  - Dummy buttons (need to write scripts, possibly using AJAX) to add/delete them from the user's plan (or add all button), and needs to finish suggested electives section
  - Concerns: need the registration page to dropdown degree rather than manually specify (otherwise case-sensitive, spelling, etc. might make it not work)
    - Also, suggested schedule (when a requirement can use different options) should provide a dropdown (use that instead of "next option" button)
    - And use checkbox buttons instead of "save" ones, and a "save selected" at the bottom of the table, for required courses and suggested electives (and oppositely for existing plans; "delete all" option)
- Ileri:
  - Started making edit degree page; had specifications from acceptance criteria but wasn't sure what it was supposed to look like
  - Hasn't figured out how to determine what courses the user requires for the degree
    - Found in CourseRequirement table; script already available that finds courses a user needs
  - Adding interests could be done based on words found in course titles and descriptions (either process on the page, or have some kind of external file that has the resultant "words")
    - Five textfields, basically, with autocomplete
- Arturo:
  - Finished the DDL/UML task
  - Did some work in course browser–querying and such, displaying the actual information rather than hard-coding
  - Suggestion: use GET method in course browser to show a course page; eg. "searchresults.php" would take variable course name by GET instead of POST so it can be referenced in <a href="searchresults.php?cname=COSC 111"> (although space character might be a complication)
- Lam:
  - Made a footer and header (trying to get footer to stick at the bottom of the screen)
    - takes some rather odd settings (Kevin gave a potential solution)
  - Started on CSS
  - Still needs to upload to Github, but after that we can apply those all to our own pages
  - Header menu: should use "my degree" rather than just "my schedule", have a sub-menu with options "compare degrees (Kevin's), schedule plan (Eliana's), degree progress (Ileri's)"–essentially we need to make sure there's some way to get to every page
  - Logout should invalidate session/cookie, and direct to index.php (login page)

James announced that as a group we will need an additional "summary" report (cover results of project as a whole)

Will have take-home final as well (due back, via Connect, by the listed exam day)

Scrum - Monday, April 4, 2016

Location: SCI 234

Present: Arturo, Ileri, Eliana

- End of sprint, according to JIRA, is midday Wednesday (4:11pm)
  - Try to get everything done by the end of Tuesday and then sort anything else out after class on Wednesday
  - For the rest of Wednesday we can focus more on getting the presentation ready
- Ileri did manage to reuse a fair amount of the suggested degree page (for queries, functions, etc.), and updated a bit of the UI and index/register pages
  - One problem with header discovered: menu dropdowns are underneath the page contents/div (and changing z-depth doesn't make a difference)
- Arturo: no new problems; started by changing the course page to get method, and still working on tasks in general
- Eliana was unable to get work done this weekend but already has a good idea of what to do (from previous scrum meeting notes)
  - Planning to finish both pages by end of the night
- Lam checked in via the Facebook message chain
- Kevin worked on the home page; testing all the query statements to make sure they were working. Page is essentially complete just needs to have css done. Compare page still needs ajax scripts or would need to take a different approach.

Conclusions

Overall, this sprint went very smoothly: due to proper sprint planning, we did not have any major issues this sprint. We worked well as a team and communicated well when we had any issues with any of our tasks or with how the sprint was going as a whole. Once again, however, some of us did not attend scrums as consistently as we should have, so the meetings were not as productive as they could have been. However, we still had some scrum meetings and otherwise kept in communication through Facebook, and were all informed as to what each member was working on. Also, in this sprint, we focused more on following Agile principles, such as setting up better user stories, closer to INVEST guidelines, taking extra time to ensure acceptance criteria was appropriate for each task, estimating time as a group and adjusting as needed, and setting due dates where appropriate. All these factors led to a sprint we would actually be able to complete. We took more time in advance to think of any problems we might face, and made sure there wasn't anything that could stop us from getting meaningful work done for this sprint, and ended up producing a fully-functional project, despite the loss of some features we had in mind before the project began.

Project Retrospective

What we did well	Lessons Learned
• Had a good initial plan for project and its scope • A lot of documentation • Worked well as a team when it came to solving issues related to the project as a whole	• Needed better understanding of scope before starting • Never made Agile work for us, was always trying to catch up to its principles (had mini-water-scrum-falls for the most part) • Shouldn't use scope as sole basis for sprint planning (always had time issues as a result) • Creating meaningful user stories with proper time estimates is important for clarity and organization • Work log was not consistent in the first 2 sprints, even sprint 3 left a lot of work towards the end

Looking back at our project, we made a lot of mistakes when it came to abiding by Agile standards, and in general, we failed when it came to utilizing our time appropriately. We did make small improvements from sprint to sprint, though, and finally got "on track" in our final sprint. Initially, we had a fairly clear idea of what we wanted to do for our project, but rather than following Agile standards, we had more of a "Big Design Up-Front" mentality. As a result, we had relatively large backlogs for Sprints 1 and 2 when compared to 3, with most of the tasks being delayed, since we did not account for how little time we would have (and tried to control using scope instead, as in the traditional "iron triangle"). For this reason, more often than not, we often fell back to a Waterfall model rather than sticking to Agile. Our planning was sub-par as well: especially early-on, we did not do time estimates well, and we had ambiguous user stories and acceptance criteria. These caused issues during the sprint, such as team members being unsure of what exactly they needed to include in their work. This was fixed by Sprint 3, but it did cause many issues in Sprints 1 and 2 (as discussed in their retrospectives).

Aside from planning, it took us some time to put a committed work load towards the project. Granted, all of us had other projects and courses to worry about as well, but we did not commit enough time each sprint to get enough work done, due in part to large backlogs and time estimates for each sprint. It did take us a while to find a good middle ground where each of us could commit the expected amount of hours per sprint. This improvement can somewhat be seen through our velocities and sprint burndowns, highlighted most obviously in sprint 3.

Overall, we've learned that more difficult than it seems to commit to a task, and easier than expected to fall back into different processes (such as "mini-waterfalls"). That said, once we got the hang of Agile, things did become a lot smoother. Furthermore, we came to understand the importance of well-written user stories and acceptance criteria, as they cleared a lot of confusion on what everyone needed to do, and they set clear goals for what needed to be done for each task. Additionally, getting planning done correctly and learning to spend time efficiently makes a massive difference when it comes to creating a feasible backlog for a sprint and ensuring that progress-hindering problems can be avoided. Our team did communicate fairly well, so many of the issues we faced were not as bad as they could have been, and we worked together to solve these issues relatively quickly. Although, in general, it is better to be proactive than reactive, it took extra time and commitment that we did not put forth until our last sprint.

---

7. References

7.1 Extensions and Third-Party Libraries

- jsoup Java HTML Parser (version 1.8.3)
    - http://jsoup.org/

7.2 Tutorials and Code Referenced

- Codecademy JavaScript tutorial
    - https://www.codecademy.com/learn/javascript

7.3 Other

- Icons DB (additional icons used in UML diagrams)
    - http://www.iconsdb.com/

Appendix A: Paper Prototypes and Design Outlines

Outline of Database

Degree Requirements Specification and Usage

Site outline/Navigation

Required User Info

Web Site Design Outline

Database DDL

Appendix B: Course List

Refer to File List item: Crawler-sample-output.txt

Tab-separated values (line-separated records) as parsed by crawler.

Appendix C: User Manual

C.0 Introduction

EZ-Plan is a web site that aims to facilitate decisions related to course registration for Computer Science students at UBCO by providing a concise view of courses available, recording current progress, finding courses that might be in your field of interest, and allowing you to check requirements, in one application.

C.1 Installation

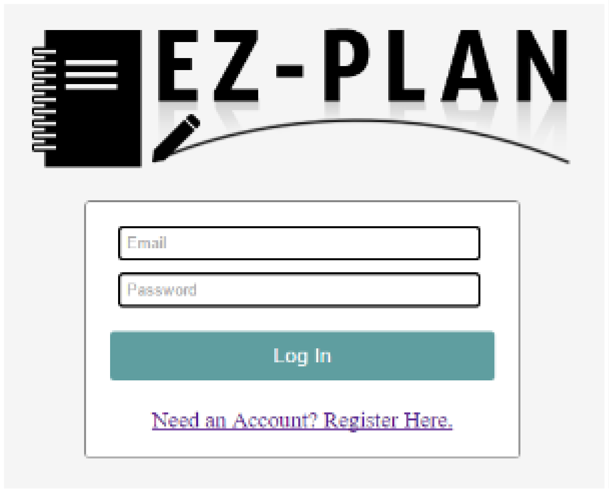
A dedicated web server is not available for this project; however, it can be hosted from a user's web browser or local network by configuring PHP and Apache on the local machine.

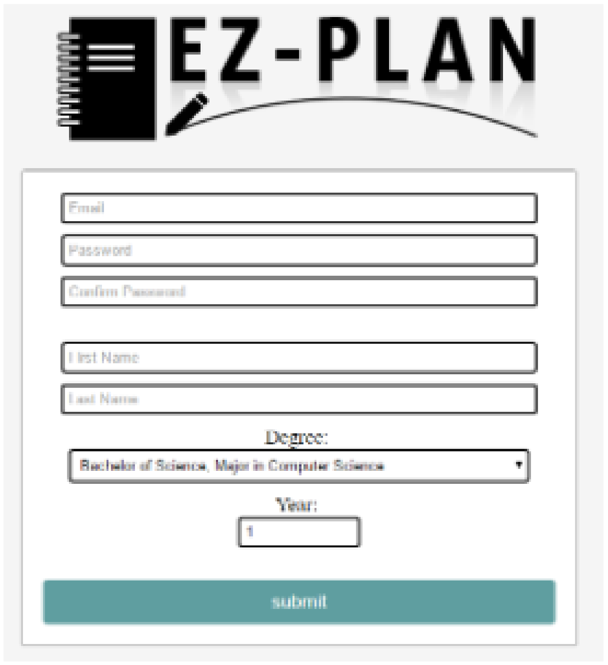
Tutorials for these purposes are available in numerous locations online, such as the following from Sitepoint for Apache and PHP on Windows machines.

Additionally, due to the permissions of the database, you must be on the UBC network or VPN in order to access the site. This can be done by accessing the site from on-campus or by installing a VPN; instructions on the latter are available on the UBC IT Services website.

C.2 Login and Registration

The initial page for this site, `(host location)/pages/index.php` by default, is the login page, also containing a link to the registration page:

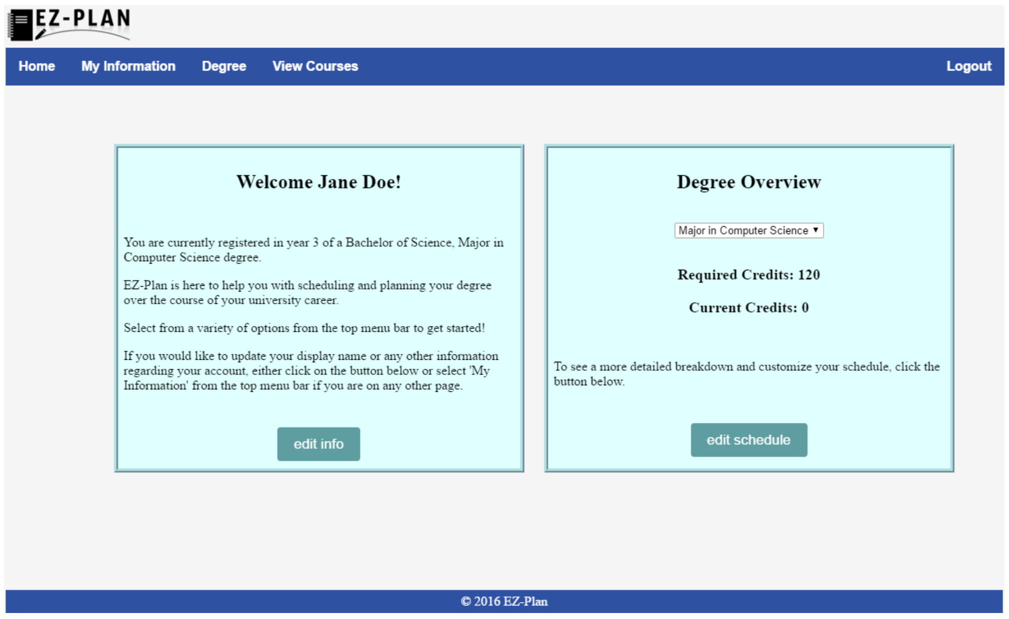




If this is your first time using the site from any location, you will need to create a new account on the registration page, and you will be logged in.

Otherwise, enter your information in the login page.

Afterwards, you will be redirected to the site's home page, which contains a description of the site's purpose, a navigation menu bar, and additional links to the same pages:



Welcome Jane Doe!

You are currently registered in year 3 of a Bachelor of Science, Major in Computer Science degree.

EZ-Plan is here to help you with scheduling and planning your degree over the course of your university career.

Select from a variety of options from the top menu bar to get started!

If you would like to update your display name or any other information regarding your account, either click on the button below or select 'My Information' from the top menu bar if you are on any other page.

edit info

Degree Overview

Major in Computer Science

Required Credits: 120

Current Credits: 0

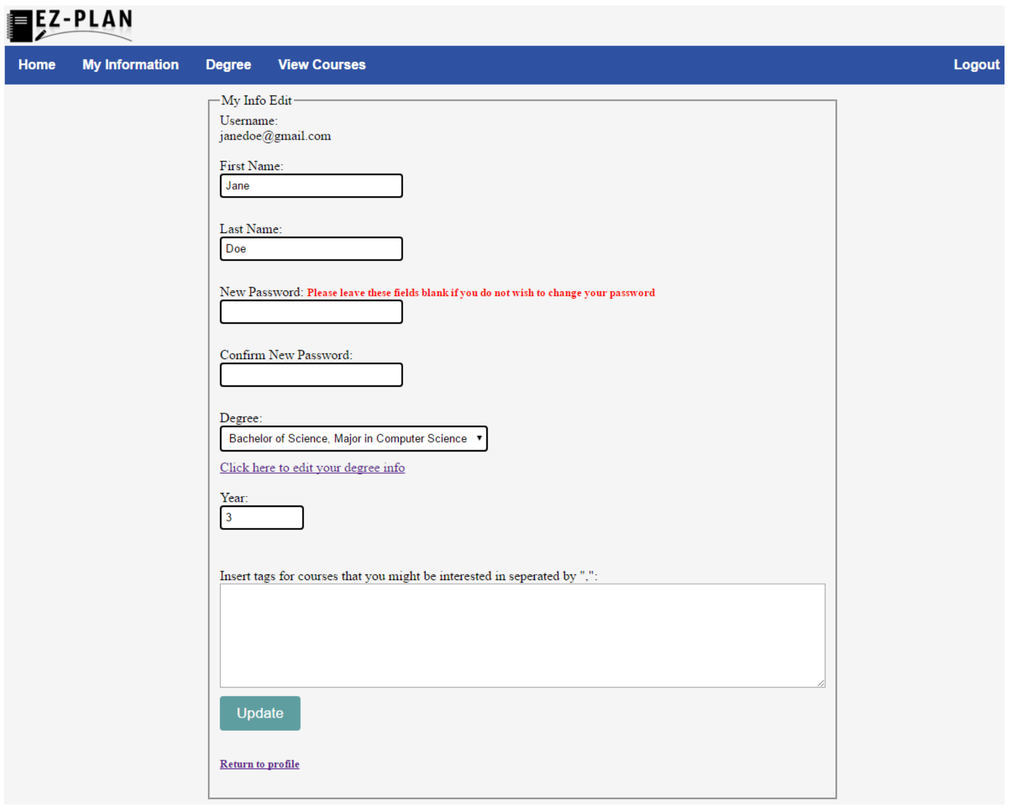
To see a more detailed breakdown and customize your schedule, click the button below.

edit schedule

© 2016 EZ-Plan

C.3 Editing Profile and Information

Information related to your account, including a list of keywords describing your interests, can be defined in the Edit Info page, linked to as "My Information" on the menu bar or "edit info" on the left column of the home page. Enter the fields you wish to change, and click "Update" to save these changes.



My Info Edit

Username:
janedoe@gmail.com

First Name:
Jane

Last Name:
Doe

New Password: Please leave these fields blank if you do not wish to change your password

Confirm New Password:

Degree:
Bachelor of Science, Major in Computer Science

Click here to edit your degree info

Year:
3

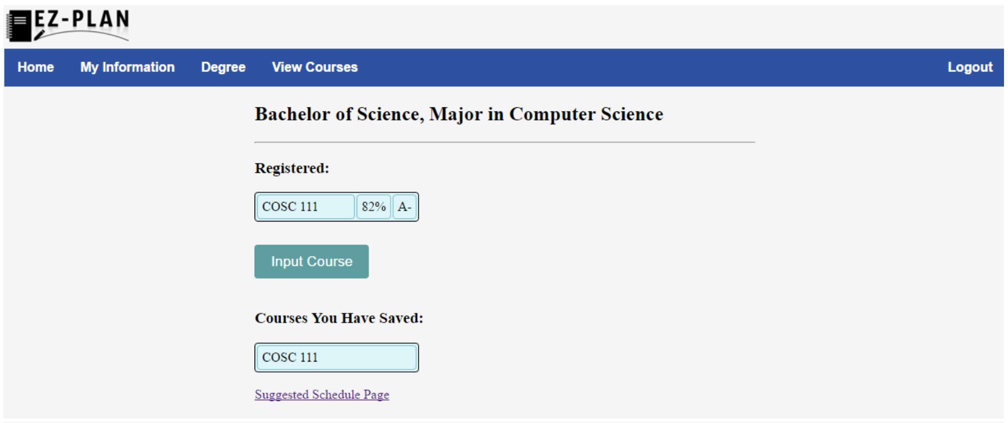
Insert tags for courses that you might be interested in seperated by ",":

Update

Return to profile

Your current schedule (that is, the courses you have already been taking at UBC) can be edited and added to on the Degree Breakdown page, accessed via the "Degree Info" link within the "Degree" dropdown on the menu bar.

Press the "Input Course" button to specify a course code and the grade you received in that course. Planned, or saved, courses can be defined in the Suggested Schedule page (described in Section **C.5**).



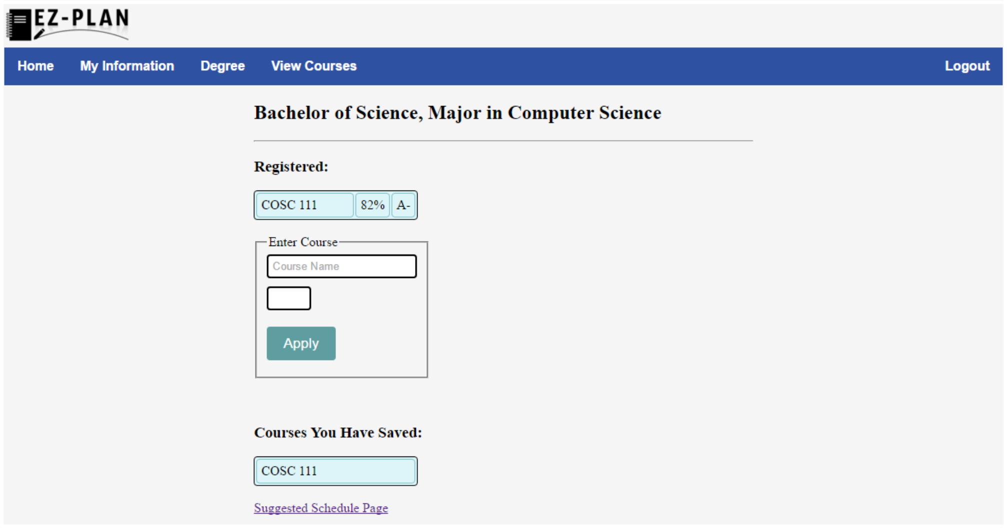
Bachelor of Science, Major in Computer Science

Registered:

COSC 111	82%	A-

Courses You Have Saved:

COSC 111



Bachelor of Science, Major in Computer Science

Registered:

COSC 111	82%	A-

Courses You Have Saved:

COSC 111

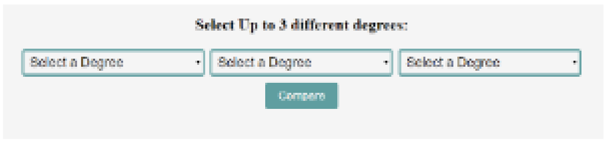
C.4 Viewing Requirements

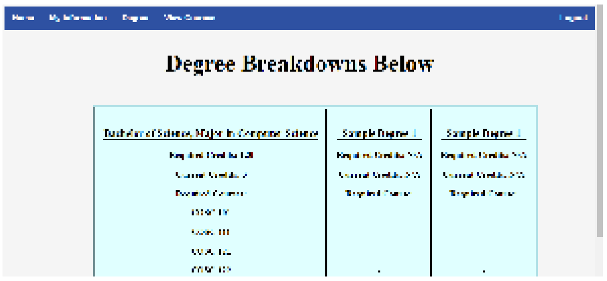
Requirements can be viewed either on the Compare page, or the Suggested Schedule page.

C.4.1 Compare Degrees

The Compare page can be accessed via the "Compare Degrees" link within the "Degree" dropdown on the menu bar.

Select three degree types from the dropdowns and click the Compare button in order to see a straightforward list of requirements for each degree type:





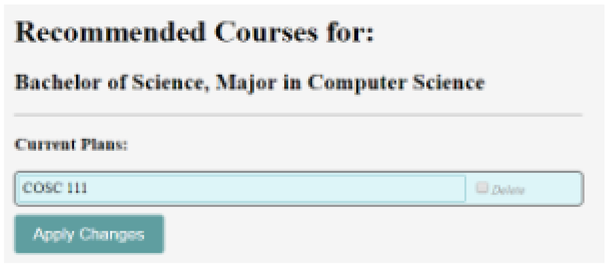
Degree Breakdowns Below

Bachelor of Science, Major in Computer Science	Sample Degree 1	Sample Degree 1
Required Credits: 120	Required Credits: N/A	Required Credits: N/A
Current Credits: 3	Current Credits: N/A	Current Credits: N/A
Required Courses:	Required Courses:	Required Courses:
COSC 111		
COSC 111		
COSC 121		
COSC 1[illegible]		

C.4.2 Suggested Schedule

The Suggested Schedule page can be accessed via the "Schedule Plan" link within the "Degree" dropdown on the menu bar, or the "edit schedule" link on the right column of the site home page.

The first table lists the courses you have saved to your schedule and provides options to delete (unsave) them.



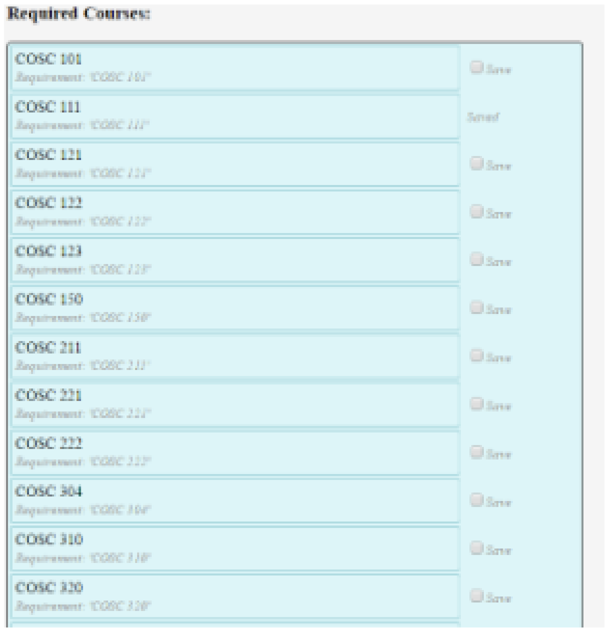
Recommended Courses for:

Bachelor of Science, Major in Computer Science

Current Plans:

COSC 111	Delete

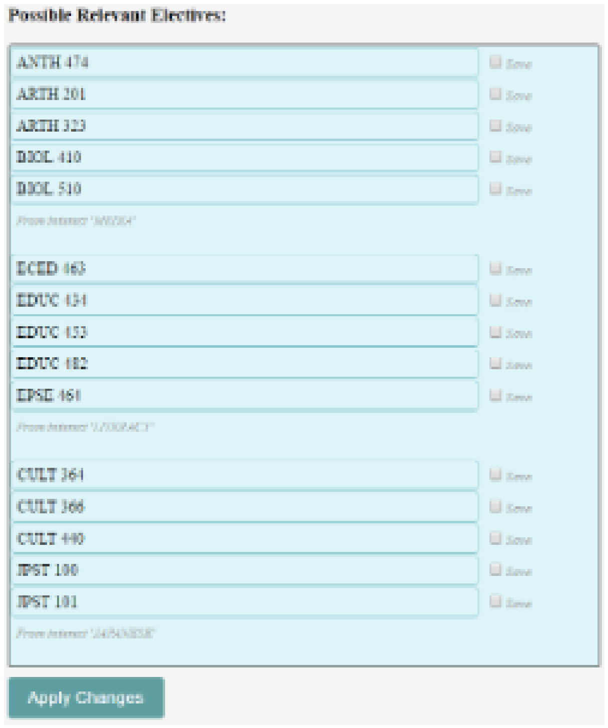
Below that is a second table, which lists the requirements and some suggestions for courses that will fill the requirements, as well as options to save each course into your plan where applicable.



Required Courses:

COSC 101 Requirement: 'COSC 101'	Save
COSC 111 Requirement: 'COSC 111'	Saved
COSC 121 Requirement: 'COSC 121'	Save
COSC 122 Requirement: 'COSC 122'	Save
COSC 123 Requirement: 'COSC 123'	Save
COSC 150 Requirement: 'COSC 150'	Save
COSC 211 Requirement: 'COSC 211'	Save
COSC 221 Requirement: 'COSC 221'	Save
COSC 222 Requirement: 'COSC 222'	Save
COSC 304 Requirement: 'COSC 304'	Save
COSC 310 Requirement: 'COSC 310'	Save
COSC 320 Requirement: 'COSC 320'	Save

The third table makes suggestions for elective courses that might be relevant to the interests you defined in your user profile on the Edit Info page (up to 5 per interest), and any of the courses may also be saved to your plan.



Possible Relevant Electives:

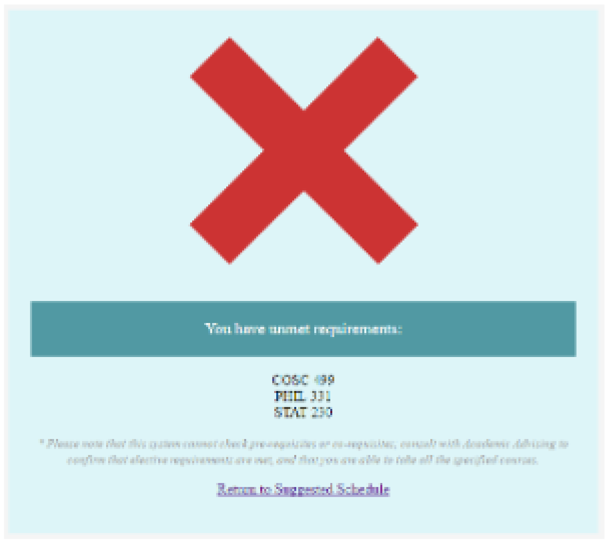
ANTH 474	Save
ARTH 201	Save
ARTH 323	Save
BIOL 410	Save
BIOL 510	Save
From Interest 'MEDIA'	
ECED 463	Save
EDUC 434	Save
EDUC 453	Save
EDUC 482	Save
EPSE 461	Save
From Interest 'LITERACY'	
CULT 364	Save
CULT 366	Save
CULT 440	Save
JPST 100	Save
JPST 101	Save
From Interest 'JAPANESE'	

C.5 Checking Validity

The validity of your scheduled plan can be determined on the Validate Degree page, linked to underneath the third table in the Suggested Schedule page.

Once it has finished loading, it will give a confirmation if the saved schedule is valid, or otherwise, a notice that the schedule does not satisfy certain requirements, and a list of the ones that are not fulfilled.

Please note that, at this time, EZ-Plan does not take factors such as course pre-requisites into consideration, and you should consult with UBCO's Academic Advising office before making significant degree-related decisions.

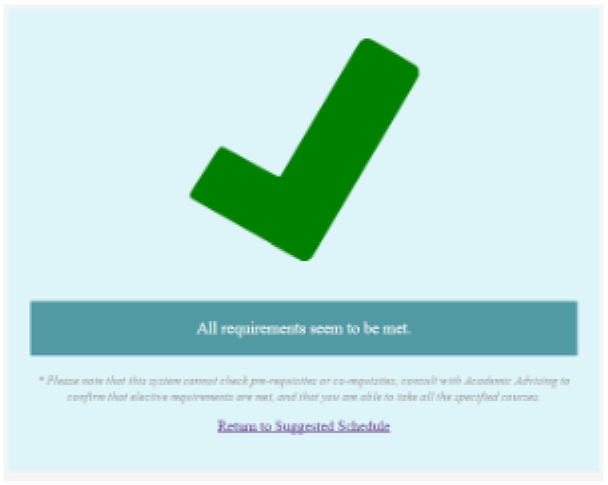


You have unmet requirements:

COSC 499
PHIL 331
STAT 230

** Please note that this system cannot check pre-requisites or co-requisites; consult with Academic Advising to confirm that elective requirements are met, and that you are able to take all the specified courses.*

Return to Suggested Schedule



All requirements seem to be met.

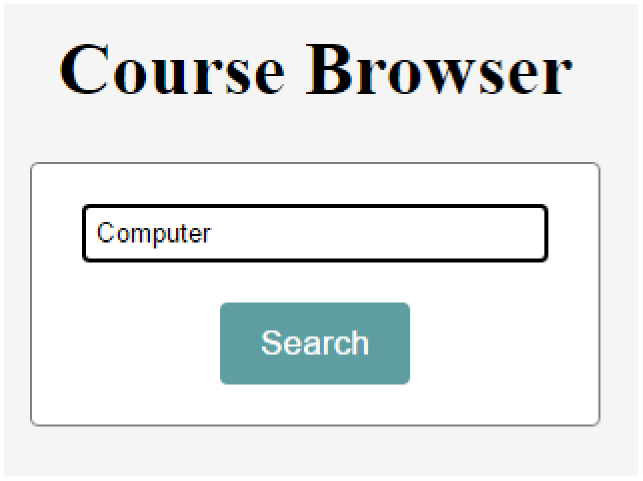
** Please note that this system cannot check pre-requisites or co-requisites; consult with Academic Advising to confirm that elective requirements are met, and that you are able to take all the specified courses.*

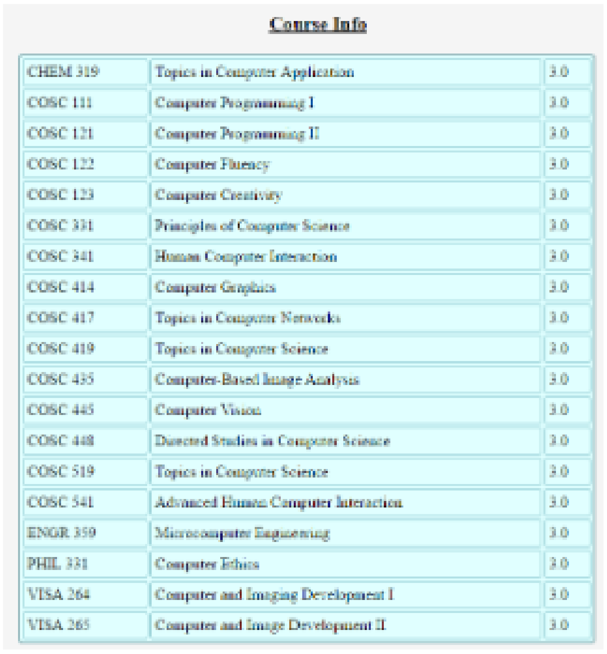
Return to Suggested Schedule

C.6 Viewing Course Information

Courses can be searched and basic information shown by using the Course Browser feature, available using the "View Courses" link on the menu bar.

Enter a query to find courses that match the description, as well as see information such as the number of credits offered and the course title.





Course Info

CHEM 319	Topics in Computer Application	3.0
COSC 111	Computer Programming I	3.0
COSC 121	Computer Programming II	3.0
COSC 122	Computer Fluency	3.0
COSC 123	Computer Creativity	3.0
COSC 331	Principles of Computer Science	3.0
COSC 341	Human Computer Interaction	3.0
COSC 414	Computer Graphics	3.0
COSC 417	Topics in Computer Networks	3.0
COSC 419	Topics in Computer Science	3.0
COSC 435	Computer-Based Image Analysis	3.0
COSC 445	Computer Vision	3.0
COSC 448	Directed Studies in Computer Science	3.0
COSC 519	Topics in Computer Science	3.0
COSC 541	Advanced Human Computer Interaction	3.0
ENGR 359	Microcomputer Engineering	3.0
PHIL 331	Computer Ethics	3.0
VISA 264	Computer and Imaging Development I	3.0
VISA 265	Computer and Image Development II	3.0

Appendix D. Presentation Slides

- Sprint 1:
    - Sprint 1 Presentation.pdf
    - Web: https://docs.google.com/presentation/d/1jxmp8_ek9G_sf1Zhkl2CM-A0w1lC8Xotn738N67bJXw/edit?usp=sharing
- Sprint 2:
    - Sprint 1 Presentation.pdf
    - Web: https://docs.google.com/presentation/d/17QfkrjTMaxcyG_V1N3BAQBEPcQjY96ifhpR_wjK3DqM/edit?usp=sharing
- Sprint 3:
    - Sprint 2 Presentation.pdf
    - Web: https://docs.google.com/presentation/d/1xp7Bf4MShytfX9YdcjqplYIXwLt4SeY-49jl47GzlOI/edit?usp=sharing