CoursePlanner

*A **PLSMLABS** PROJECT*
FOR THE PROVOST TECHNOLOGY GROUP
AT THE UNIVERSITY OF TEXAS AT DALLAS

Team 4 - Matthew Cocco, Stephen Abraham, Logan Starkey, Parth Badhiwala

Contents

Executive Summary 3

Systems Proposal: CoursePlanner 4

Primary Requirements 6

Structural Models 7

Behavioral Models 7

- Use Case Diagram 7
- Use Case Descriptions 8

Dynamic Models 11

- Sequence Diagrams 11
  - Use Case: View Courses 11
  - Use Case: View Degree Plans 11
  - Use Case: Register/Add/Drop Courses 12
  - Use Case: Plan Semester Schedule 12
  - Use Case: Create Degree Plan 13

Design Documentation 14

- UI Mockup: Graph Layout 14
- UI Mockup: Desktop Browser 15
- UI Mockup: Mobile Browser 16
- Sample Controls 17
- Methods 18

Testing 24

- Method Testing 25

Project Management Documentation 27

- Work Breakdown Structure 27
- Meeting Minutes 30

Lessons Learned 32

- Parth Badhiwala 32
- Logen Starkey 33
- Stephen Abraham 34
- Matthew Cocco 35

Executive Summary

The current Add/Drop/Swap & schedule planning systems are not connected. The fragmentation of online catalogs, Coursebook, and Galaxy/Orion causes frustration and poor user experiences. This disorganization of critical systems makes long-term schedule planning without human advisor assistance difficult.

CoursePlanner is an application designed to reduce time-to-graduation and improve course scheduling & registration experiences for students and advisors at the University of Texas at Dallas.

This improvement to degree-, course-, and schedule-planning will primarily be accomplished through creation of a database of courses & catalogs, and the development of application logic to leverage open-source libraries and users' browsers to display graphs of course prerequisite chains and clarify the optimized orders in which to take courses at UT Dallas.

The interface for users will be a progressively-enhanced web application which runs on devices as small as mobile phones, and exposes more features as screen spaces allow (up to lap- and desktop screen sizes), to support as many users as possible.

As a user-friendly web application extending the online Catalog, Coursebook, and Galaxy/Orion systems, CoursePlanner will improve the experience of students selecting which courses to enroll in & help them plan semesters in advance based on directed acyclic graph models of degree plans and degree plan catalogs with information provided by previously-submitted student evaluations and historical schedule data. This, in turn, will reduce time spent by advisors that would otherwise be spent giving the same or similar advice and instruction

Systems Proposal: CoursePlanner

Sponsor & Stakeholders	Provost's Technology Group @ UTD UT Dallas Advising & UTD Student Body	**Start Date**	10 OCT 2016
Project Leader	Matthew Cocco	**End Date**	28 NOV 2016

Background

The current Add/Drop/Swap & schedule planning systems are not connected, and the fragmentation of online catalogs, Coursebook, and Galaxy/Orion causes frustration and poor user experiences. This disorganization of critical systems makes long-term schedule planning without human advisor assistance difficult.

Purpose

The purpose of this project is to improve academic career planning and reduce average time-to-graduation via intelligent, understandable computerized assistance; reduce advisor workload for basic student schedule planning; and improve student understanding of course pre/co-requisite chains.

Project Objectives

- Create developer-usable databases of courses and catalogs for 2015-16 academic year
- Develop, publish web app to suggest course schedules & display graphs of degree plans
    - Provide fuzzy-search for courses and directed graph rendering of degree plans
    - Provide substantial time savings to students and advisors exploring course graphs

Scope

This project's scope is to plan, design, and create a web application for use by students and advisors to better understand and visualize how degree plans and course pre/co-requisite chains actually work. Scope will be limited to the development of a single-page HTML5/CSS3 + JS webapp backed by a JSON flatfile database, using only open-source libraries and data scraped from the publicly-accessible UT Dallas Catalog and Coursebook webpages. Planning and design will be conducted in a waterfall-style process; development which follows (including updates to design and requirements, as they arise) will follow an agile methodology.

Within	● Database of UT Dallas AY15/16 Catalogs & Course Descriptions ● Static web application development (hosted website, flatfile database)
Out	● All other years of catalogs/course-descriptions

	● Dynamic application components ● Non-website servers/hosted systems

(Project Charter, continued)

Assumptions

- University is not already developing something to meet these needs
- University will continue to provide public access to valid catalog & coursebook information

Deliverables

- Progress & Issue Reports
- Web Application (Prototype, continuously iterated): CourseExplorer
- Web Application: CoursePlanner

Project Success Criteria

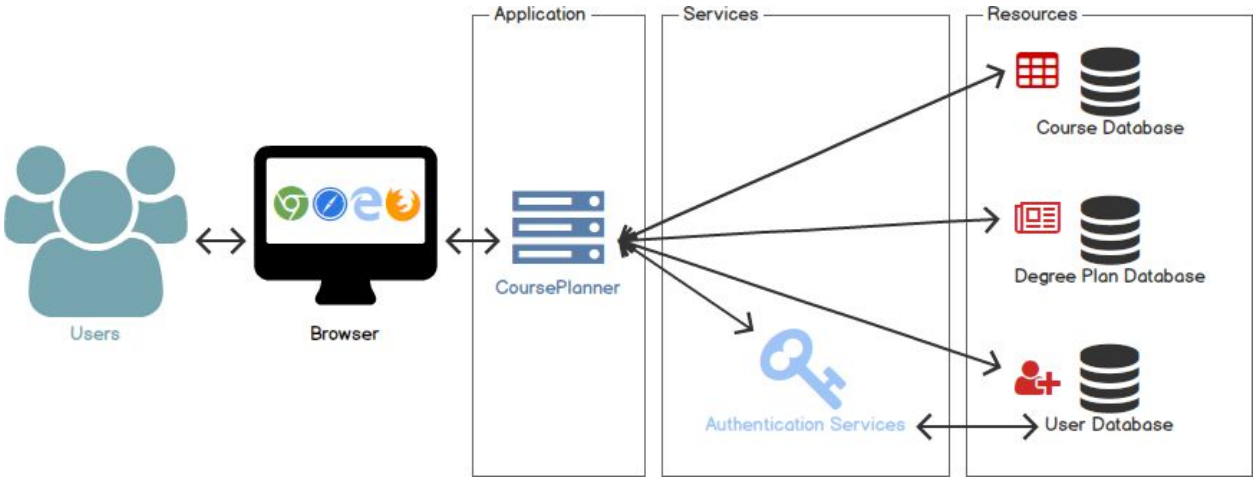
1. Application launched (published) on time & feature-complete
2. App used and then reviewed positively (averaging 4+ of 5 stars) by 40 of 50+ users

Primary Requirements

#	Requirement	Name	Reference Description
R-FT-001	Functional Technical	**Search Bar**	Has a search bar allowing for (based on user input) exact-match/fuzzy-search and selection of degree plans and courses.
R-FT-002	Functional Technical	**Graph Display**	Constructs and displays a directed acyclic multigraph of courses and their prerequisites
R-FT-003	Functional Technical	**Session Persistence**	Maintains user data across sessions, including which degree plan is selected and which courses have been marked as completed by the user.
R-FT-004	Functional Technical	**Graphs Printable**	Graph Display and supplementary information can be displayed in and/or exported to a print-friendly format.
R-FT-005	Functional Technical	**Display Course Information**	Course Information should be displayed on user request/interaction, including the description, pre/corequisites, and whether or not the user has completed the course.
R-BP-006	Behavioral Performance	**Loads Quickly**	Page should begin displaying information in less than five seconds after user connects. First load of application should take no longer than thirty seconds.
R-BP-007	Behavioral Performance	**Highly Available**	Application should be available to a minimum of 20,000 concurrent users, with graceful failover in case of overload.
R-BU-008	Behavioral Usability	**Multi-OS Support**	Application should support the most-used versions of the Android &iOS mobile operating systems as well as Chrome & Firefox web browsers (as described by developer statistics)

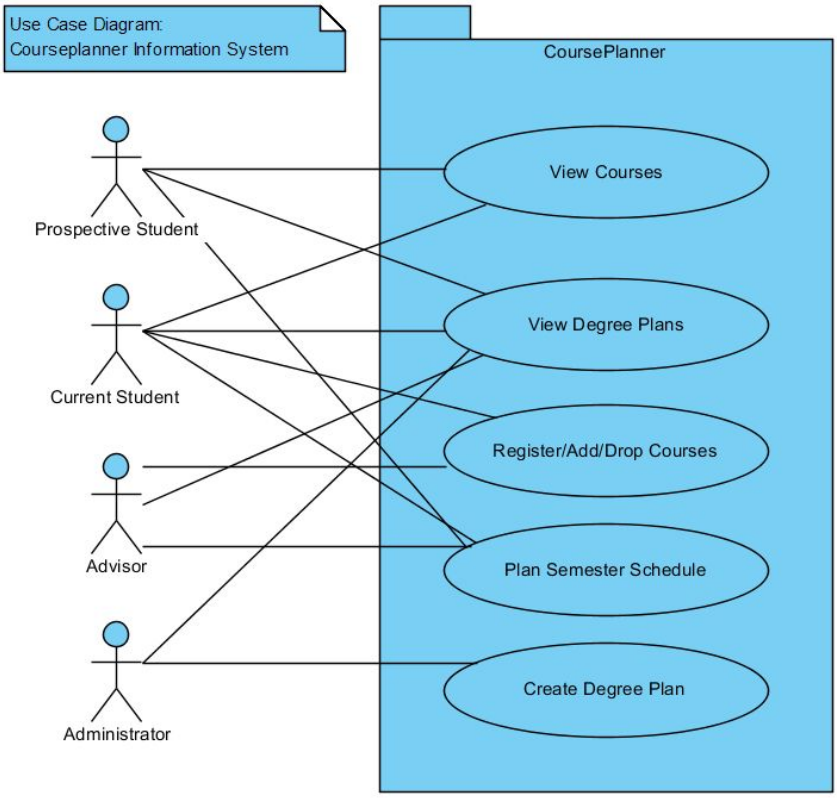
Structural Models

Architectural Diagram



Behavioral Models

Use Case Diagram



Use Case Descriptions

Use Case Name:	View Courses
Primary Actor:	Prospective Student / Current Student
Brief Description:	User views courses in a degree plan in graph view
Stakeholders:	Prospective Student – wants to explore courses in degree plan Current Student – wants to view/mark courses in degree plan
Trigger:	User has a degree plan selected
Normal flow of events:	1. Graph View loads 2. Graph View displays linked courses to user
Subflows:	
Alternate/Exception flow:	1a. Graph View loading fails; retries loading & transmits problem report; if failure continues, throws user-visible error.

Use Case Name:	View Degree Plans
Primary Actor:	Prospective Student / Current Student / Advisor / Administrator
Brief Description:	A user utilizes the Find Degree Plans feature
Stakeholders:	Prospective Student – wants to explore degree plans Current Student – wants to select current degree plan Advisor – wants to view specific degree plan during advising Administrator – wants to review extant degree plans
Trigger:	User loads application & does not have a degree plan selected OR User signals they want to see other degree plans
Normal flow of events:	1. User loads application 2. User accesses search features 3. User searches the various degree plans sorted by school/department/major
Alternate/Exception flow:	

Use Case Name:	Register/Add/Drop Courses
Primary Actor:	Current Student / Advisor
Brief Description:	User adds/drops courses selected in graph view
Stakeholders:	Current Student – wants to add/drop selected courses Advisor – wants to add/drop courses selected for student during/after advising session
Trigger:	User has course(s) selected and signals they want to add/drop
Normal flow of events:	1. User selects courses 2. User clicks “add/drop” button 3. User verifies correct courses selected 4. User confirms enrollment changes 5. User is notified of status enrollment changes
Subflows:	3a. User is shown selected courses 3b. User selects/deselects courses as necessary, resubmits via step 2
Alternate/Exception flow:	2a. User not authenticated; must log in / create new account

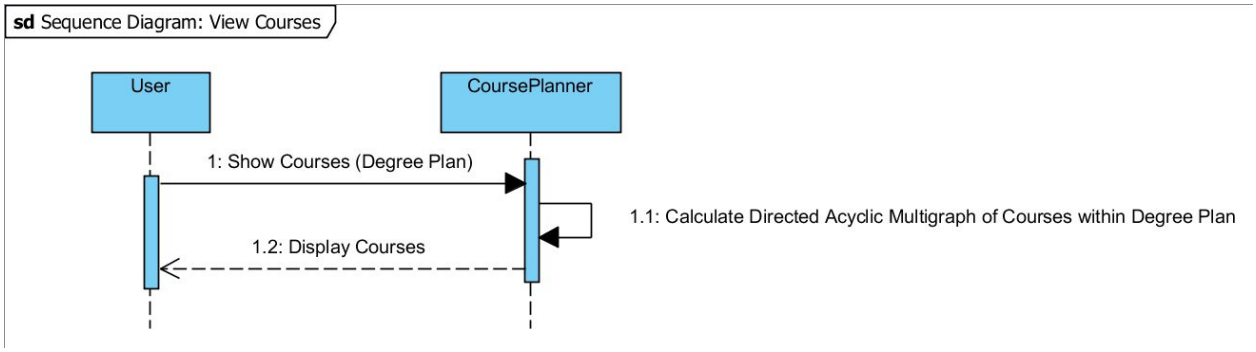
Use Case Name:	Plan Semester Schedule
Primary Actor:	Prospective Student / Current Student / Advisor
Brief Description:	User plans a semester’s schedule based on available
Stakeholders:	Prospective Student – wants to explore hypothetical semester schedule(s) Current Student – wants to add/drop selected courses Advisor – wants to add/drop courses selected for student during/after advising session
Trigger:	User with degree plan selected signals they want to plan a semester schedule based on available course
Normal flow of events:	1. User clicks “plan semester schedule” button 2. User is shown list of courses whose prerequisites are met and are offered for the desired semester 3. User selects courses 4. User confirms selection
Alternate/Exception flow:	1a. User not authenticated; must log in/create new account. 3a. Courses invalid; error returned and selection undone.

Use Case Name:	Create Degree Plan
Primary Actor:	Administrator
Brief Description:	User creates new or modify existing degree plan
Stakeholders:	Administrator – wants to create meaningfully improved plans
Trigger:	Authenticated Administrator loads “Create Degree Plan” (sub)application
Normal flow of events:	1. User signs in to application 2. User selects template for new degree plan 3. User selects courses to add to / remove from plan 4. User saves new degree plan
Subflows:	2a. Application displays most popular/recent degree plans and fuzzy-search bar, as well as statistics on displayed plans 2b. User selects degree plan from display or search 3a. Application displays fuzzy-search bar for adding new courses, automatically suggests adding default prerequisites 3b. User adds/removes pre/corequisite relationships to courses
Alternate/Exception flow:	1a. User is not authenticated; is asked to sign in or create new account.

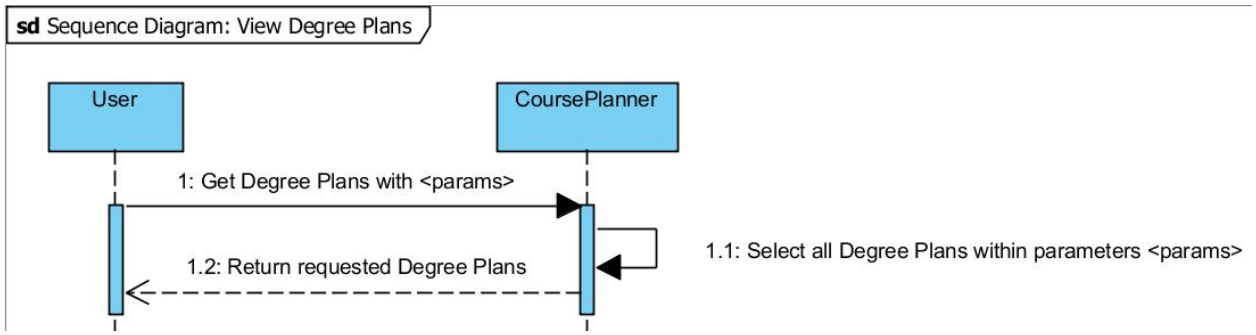
Dynamic Models

Sequence Diagrams

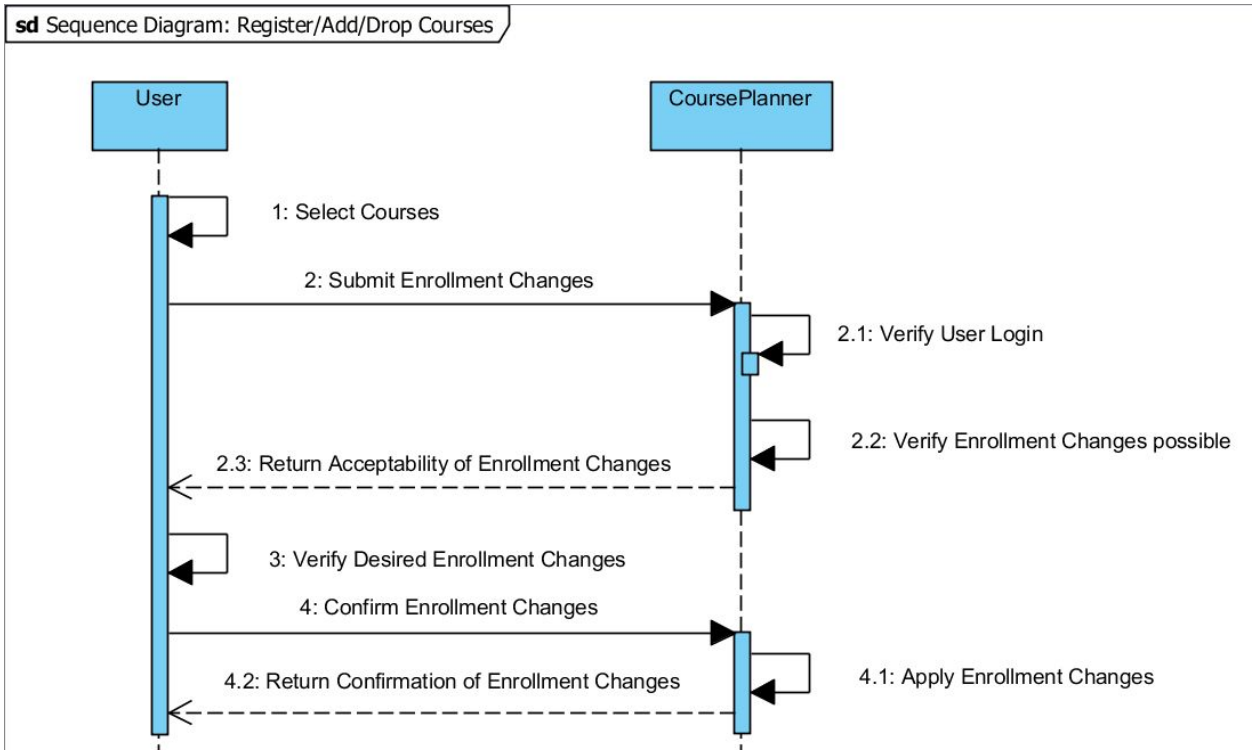
Use Case: View Courses



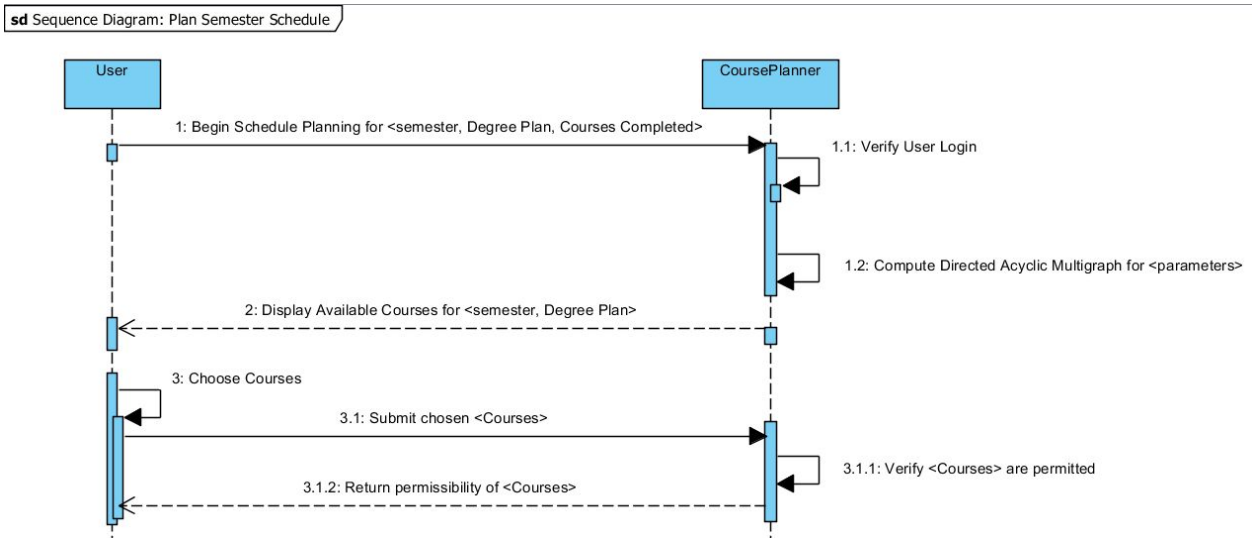
Use Case: View Degree Plans



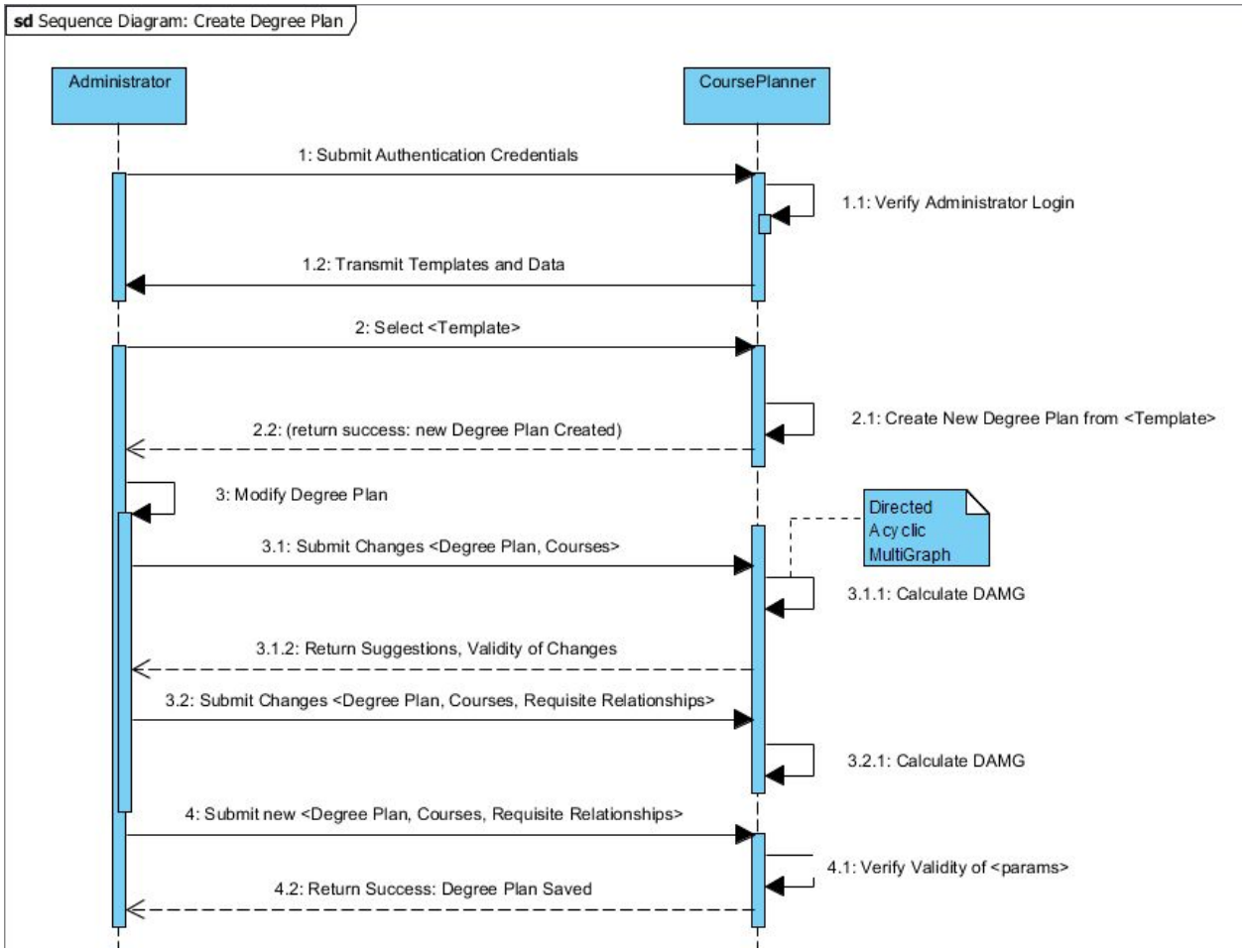
Use Case: Register/Add/Drop Courses



Use Case: Plan Semester Schedule

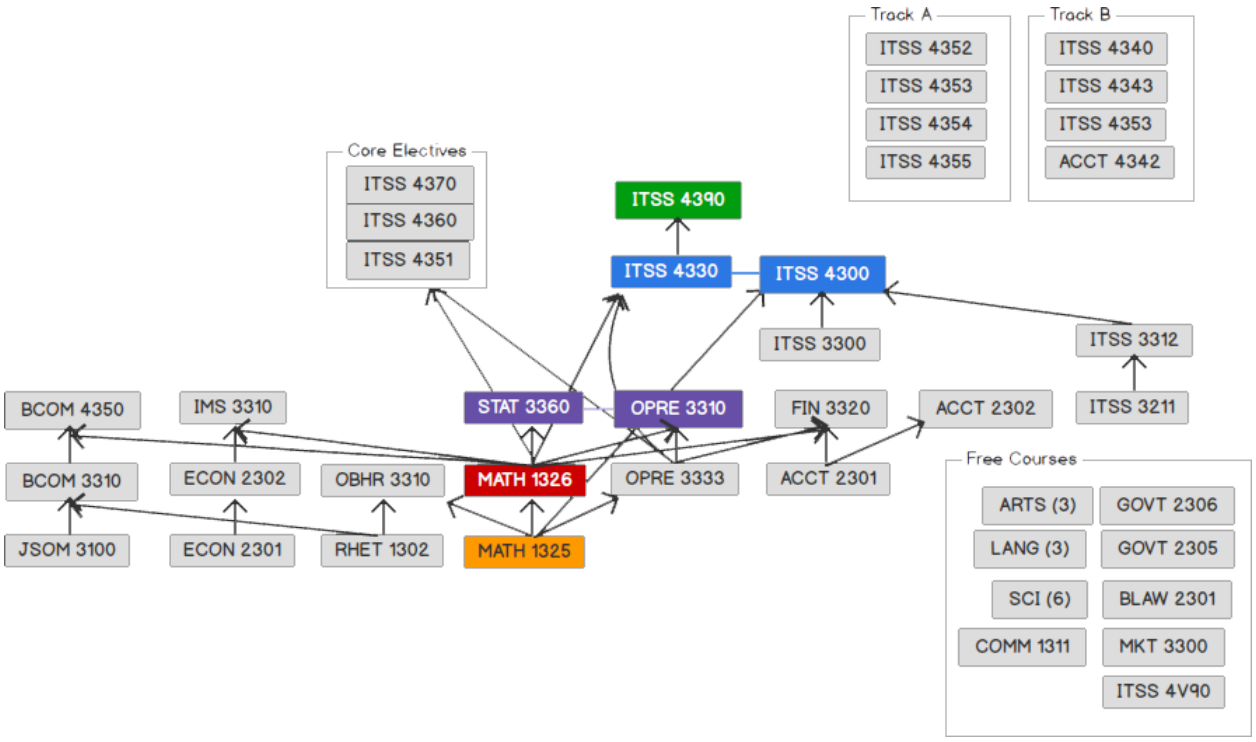


Use Case: Create Degree Plan

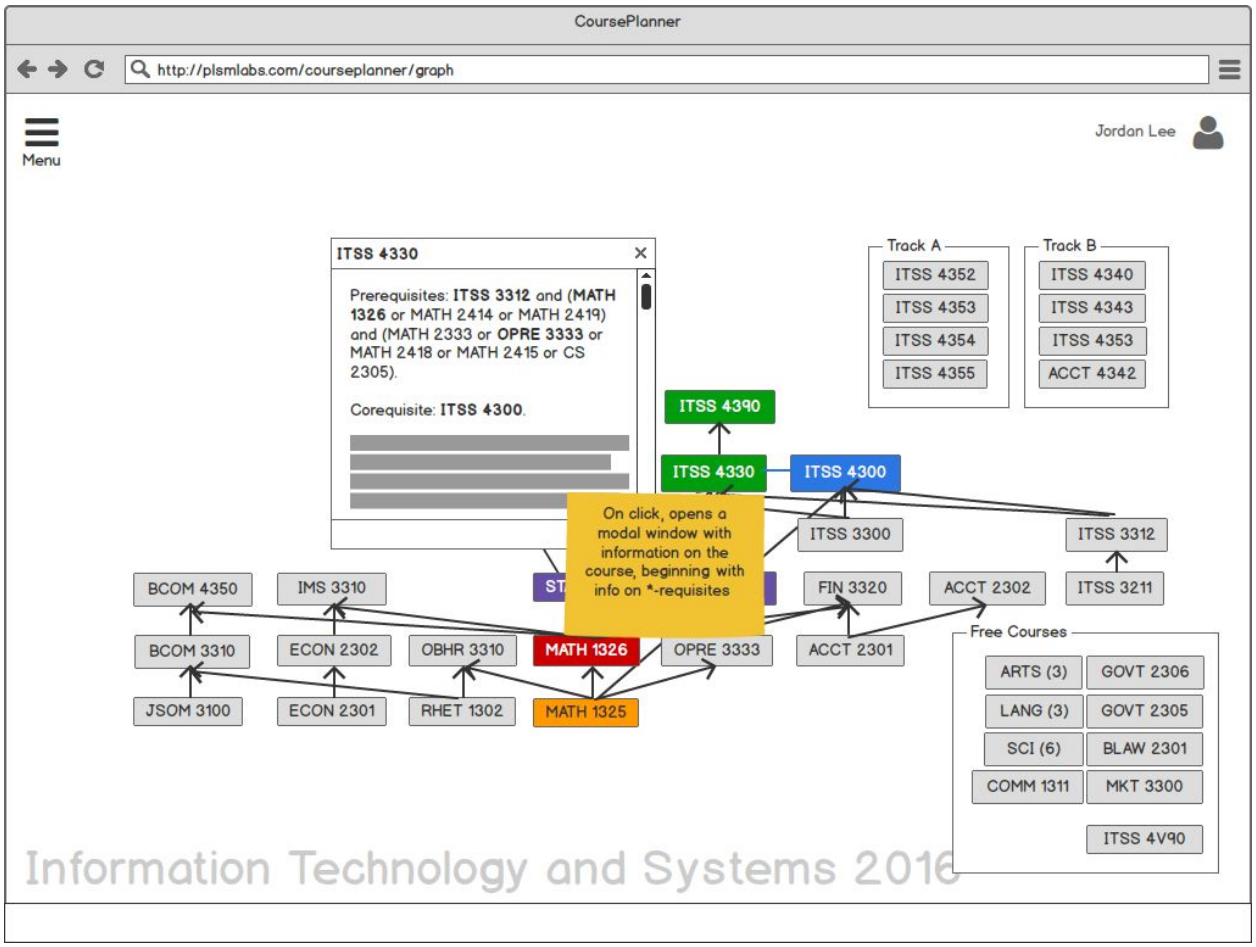


Design Documentation

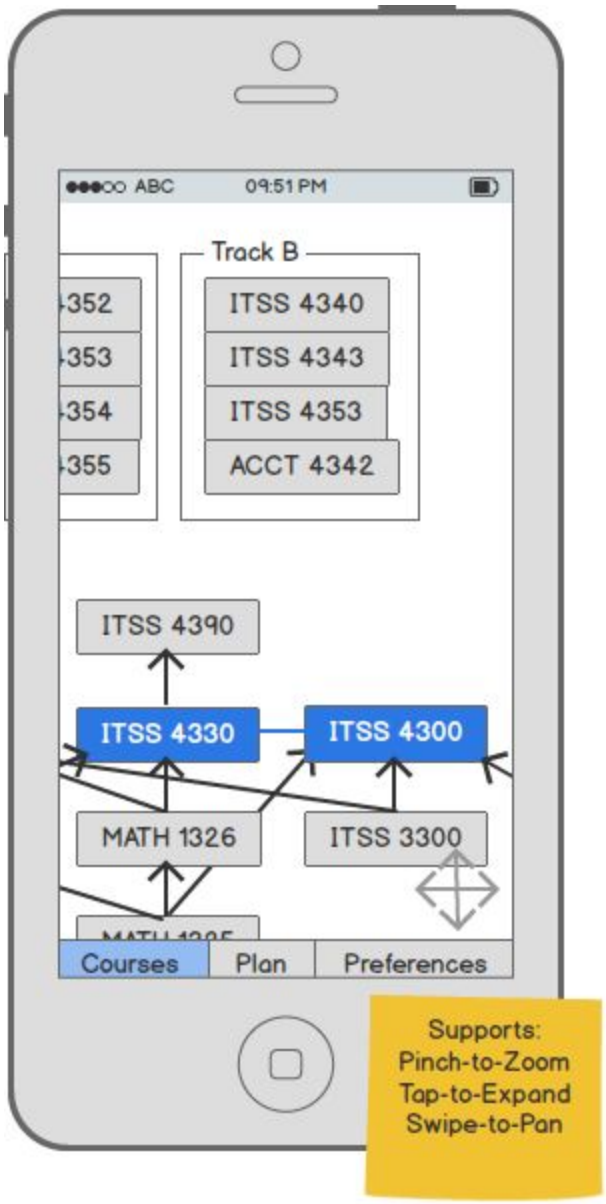
UI Mockup: Graph Layout



UI Mockup: Desktop Browser



UI Mockup: Mobile Browser



Sample Controls

1. **User Interface Control**: **Search Bar** will only accept input in the format AB[CD] 1234 (class shortcodes)
2. **User Interface Control:** Admin can override prerequisites for student schedule.
3. **Flow Logic**: if a user tries to authenticate and fails, redirect them to login screen with an error message explaining how they failed (and include a link to reset their password) so that they are properly authenticated
4. **Flow Logic**: If an unauthenticated user tries to access an admin page, they will be redirected to the homepage
5. **Business Rule**: a degree plan must be selected for a user to start adding courses (otherwise, we don't know what courses they need to complete).
6. **Business Rule**: If a course is selected but the prerequisite(s) has not been met, a notification will inform the user that something is amiss.
7. **Business Rule:** If a class that was just selected is the same time of another class that has already been selected, an alert will inform the user about trying to be in two places at once.

Methods

Method Name: **searchForDegreePlan**	Class Name **searchBar**
Description of Responsibilities Implement the necessary behavior to search through extant Degree Plans for the user to choose from. Can occur on KeyUp or when input not changed for arbitrarily small amount of time	
Arguments Received - searchString	Data Type string
Return Value: results	Data Type List of integer DegreePlanIDs

Message & Example
 	
`searchForDegreePlan(searchString) : List[Int]`
 	
`searchForDegreePlan("Information Te")`
`// Information Technology and Systems`
 	
`searchForDegreePlan("ITSS 201")`
`// ITSS 2016`	

Algorithm Specification **If searchString not null then** *// something has been entered* **Find searchString in degreePlanNames** **add matches to results list** **Return list of results**	

Method Name: **selectDegreePlan**	Class Name **SearchBar**
Description of Responsibilities Implement the necessary behavior to select one degree plan from the list of results and set it as the user's degree plan; specifies list of courses required by degree plan and adds them to state. Calls DrawGraph() with list [courseIDs].	
Arguments Received - degreePlanID	Data Type integer
Return Value: Tuple : Success error_msg	Data Type Boolean String (empty if Success is True)
Message & Example **selectDegreePlan(degreePlanID)** **selectDegreePlan(20)** *// twenty-first degree plan ID implemented, zero-indexed;*	
Algorithm Specification **onClick(degreePlanID):** **Set user.degreePlanSelected to degreePlanID** **Find degreePlanID in database** **add information to App.state** **With degreePlan list of courseIDs:** **Get courseInfo for each course** **add information to App.state** **With App.state:** **DrawGraph(list [courseIDs])** **If no errors ? Return True : (False, error details)**	

Method Name: **searchForCourse**	Class Name: **searchBar**
Description of Responsibilities Implement the necessary behavior to search through extant Courses for the user to choose from.	
Arguments Received - searchString	Data Type String
Return Value: results	Data Type List of integer courseIDs
Message & Example **searchForCourse(searchString) : List[Int]** **searchForCourse("ITSS 43")** // ITSS 4330 **searchForCourse("Systems An")** // Systems Analysis and Design	

Algorithm Specification
 	
If searchString not null AND searchString not "" then
// string is not null or empty
// something has been entered
Find searchString in courseIDs and add matches to results list
Find searchString in courseNames and add matches to results list
Return list of results	

Method Name: **getCourseInfo**	Class Name **Course**
Description of Responsibilities Implement the necessary behavior to display information about a course with the provided courseID	
Arguments Received - courseID	Data Type Integer
Return Value: results	Data Type Array: [courseID, courseName, courseDescription, [[Prerequisites], [Corequisites]]]
Message & Example **getCourseInfo(courseID)** **getCourseInfo(1714)** *// get course info of 1715th course added, zero-indexed*	
Algorithm Specification **If courseID not null AND courseID >= 0** **Find Course with courseID in database** **Return Array with** **course ID, Name, Description,** **and optional array of arrays of** **pre- & corequisite courses**	

Method Name: **shareDegreePlan**	Class Name: **App**
Description of Responsibilities Implements the behavior necessary to export an easily-shareable description of the state of the degree plan, including: - Degree Plan - Completion Status of Courses	
Arguments Received - `selectedDegreePlanID` - `coursesCompleted`	Data Type Integer List of Integers (courseIDs)
Return Value: `appStateHash`	Data Type String

Message & Example

```
shareDegreePlan(selectedDegreePlanID,
                coursesCompleted[courseID, ...])
shareDegreePlan(7, [3,4,5,8,15,21,22,23,27,29,104,151])
```

Algorithm Specification

```
// create a string to store value to be hashed
Create new String stateHashSource

// add the degreePlanID
stateHashSource += degreePlanID.toString() + ";"

// add all the completed courses' courseIDs
For courseID in coursesCompleted
      stateHashSource += courseID.toString() + ";"

// hash the values, reduce to A-Z,0-9 string
stateHash = SHA256(stateHashSource).toBase36()
// probably deterministic

// when URIs or other sharing methods are used,
// hash will be referenced to source to restore state
If stateHash not in Database Table: App State
      Add stateHash, stateHashSource to DBT: App State
```

Method Name: **meetsCourseReqs**	Class Name: **Course**
Description of Responsibilities Implement the necessary behavior to see if the prerequisites have been met for the class that has been selected. Returns a boolean TRUE if there are no prerequisites or if the prerequisites have been met.	
Arguments Received - courseID	Data Type Integer
Return Value: success error_msg	Data Type Boolean String

Message & Example

```
meetsCourseReqs(int courseID)
meetsCourseReqs(4130)
// four thousand thirty-first course added, zero-indexed;
```

Algorithm Specification

```
If courseID >= 0 AND courseID in Database
      If Course with courseID has no Prerequisites
            AND has no Corequisites
                  Return True, ""
      If Course with courseID has one or more
            Prerequisites or Corequisites
                  Set flag_OK to True
                  Set error_msg to ""
                  For Prerequisite in Prerequisites
                        If Prerequisite is marked not Completed
                              Set flag_OK to False
                              Append Prerequisite to error_msg
                              Break
                  For Corequisite in Corequisites
                        If Corequisite is marked not Completed
                              Set flag_OK to False
                              Append Corequisite to error_msg
                              Break
                  Return flag_OK, ""
Return False, "Bad courseID"
```

Testing

Delivering thoroughly-tested software from development environment to production is a core tenet of PLSM LABS software development strategy.

Testing is integrated into the design phase. As we outline the user stories and expected behavior, we develop user stories and corresponding use cases. Based on these expectations, we develop testing code alongside our feature code such that feature code satisfies the cases in our testing code. Following design, we begin development operations, which includes further automated testing. Our automated testing is conducted on staging platforms, configured to mirror the production/deployment environment. After vetting for regressions and unexpected behavior in staging, versions are packaged and shipped to production. In addition to the stable production environments, developers can also provide experimental features to a secondary beta-testing channel, for use and exploration by diverse groups of users who opt-in to usage monitoring and provide feedback on beta versions of an application.

Test-Driven Design is where our process begins. By enumerating the expected behaviors of our solution, and mapping out the expected responses to a variety of input parameters, we can test for expected results; and in developing code that provides these expected results, we aim to produce code that behaves as desired & expected.

Continuous Integration is the second facet of our testing strategy. By automating testing and requiring that (by default) testing take place from development to staging to production, we give ourselves as many opportunities as possible to discover (and correct) unexpected behavior. From automated testing of (edge) cases in the development build process to the simulation of realistic user behavior in staging to regular audits of behavior in production, PLSM LABS aims to efficiently cover 100% of the codebase in testing.

Beta-testing and proactively seeking user feedback is one of our most important strategies to deliver and improve the value expected from our products. By placing features in front of actual users and following along as they use and explore our products, we can determine if (and how) we've created positively delightful user experiences. By making it as easy & as seamless as possible for our users to provide feedback, we open dialogues and are constantly looking for ways to improve our product and feature offerings.

Method Testing

	`searchForDegreePlan(string searchString)`	
Step	**Description**	**Expected Result**
1	Open a browser	Browser opens
2	Navigate to CoursePlanner	CoursePlanner loads
3	Click Search Bar	Search Bar displays cursor for entry
4	Begin typing search term	Search Bar displays loading animated icon, displays results from returned list of matches

	`searchForCourse(string searchString)`	
Step	**Description**	**Expected Result**
1	Open a browser	Browser opens
2	Navigate to Courseplanner	Courseplanner loads
3	Click Search Bar	Search Bar displays cursor for entry
4	Begin typing search term	Search Bar displays loading animated icon, displays results from returned list of matches

	`getCourseInfo(int courseID)`	
Step	**Description**	**Expected Result**
1	Open a browser	Browser opens
2	Navigate to Courseplanner	Courseplanner loads, displays graph
3	Click a Course in Graph	Course clicked changes color, displays “more information” icon
4	Click “more information” icon on Course selected in Graph	The information related to the selected course is shown to the user within a modal window

	`meetsCourseReqs(int courseID)`	
Step	**Description**	**Expected Result**
1	Open a browser	Browser opens
2	Navigate to Courseplanner	Courseplanner loads, displays graph
3	Click a Course in Graph which has not yet been marked completed and whose pre-/corequisites are completed	Course clicked changes color, displays “mark completed” toggle
4	Click “mark completed” toggle	Course successfully marked completed, with accompanying style changes
5	Click a Course in Graph which has not yet been marked completed and whose pre-/corequisites are incomplete	Course clicked changes color, displays “mark completed” toggle
6	Click “mark completed” toggle	Course marked completed with warning, with accompanying style changes and warning displayed

Project Management Documentation

Work Breakdown Structure

For extended definitions, see appendix item: *Work Breakdown Structure Dictionary*

	Predecessor	Effort	Estimated Start Date	Estimated End Date	Member(s) Assigned
1 - Analysis		**[7 days]**	**10/10/2016**	**10/21/2016**	
1.a - Collect Requirements		(4 days)	10/10/2016	10/15/2016	Stephen
1.a.1 - Conduct Interviews		3 days	10/10/2016	10/13/2016	Stephen, Matthew
1.a.2 - Summarize Req'ts		1 day	10/14/2016	10/15/2016	Stephen
1.b - Create Class Diagrams		1 day	10/17/2016	10/18/2016	Cocco
1.c - Process Model	1.b	1 day	10/19/2016	10/20/2016	Logen
1.d - Object Behavior Model	1.b, 1.c	1 day	10/21/2016	10/21/2016	Parth
2 - Design	**1**	**[9 days]**	**10/24/2016**	**11/1/2016**	
2.a - User Interface Design	1	(4 days)	10/24/2016	10/26/2016	Cocco
2.a.1 - Wireframes	1.a.2, 1.d	1 day	10/24/2016	10/25/2016	Cocco
2.a.2 - High Quality mockups	2.a.1	3 days	10/25/2016	10/28/2016	Cocco
2.b - Software Design	1, 2.a	2 days	10/27/2016	10/29/2016	Stephen
2.c - Controls	1, 2.a	2 days	10/27/2016	10/28/2016	Parth
2.d - Test Cases	1, 2.a-c	1 day	10/31/2016	11/1/2016	Cocco
3 - Implementation (Prototype)	**1, 2**	**[14 days]**	**11/2/2016**	**11/10/2016**	**DevOps**
3.a - Repo & Workplace Initialization		2 hours	11/2/2016	11/2/2016	
3.b - Interface Implementation	2.a.2	2 days	11/3/2016	11/7/2016	
3.c - Database Collection		3 days	11/3/2016	11/7/2016	
3.d - Application Logic Development	1, 2	3 days	11/7/2016	11/10/2016	
4 - Implementation (Product)	**3**	**[17 days]**	**11/11/2016**	**11/28/2016**	**DevOps**
4.a - Collect feedback on Prototype	3	17 days	11/11/2016	11/28/2016	
4.b - Bugfixes to Prototype	3	17 days	11/11/2016	11/28/2016	
4.c - Expanded App Logic Development	3.d	17 days	11/11/2016	11/28/2016	

Work Breakdown Structure Dictionary

1 - **Analysis**

1.a - Collect Requirements - For CoursePlanner, we need to discern what specific feature and accessibility needs should be met to create a product useful to its users.

1.a.1 - Conduct Interviews - a minimum of twelve individuals should be located (from different majors and years, including at least 4 freshmen), and interviewed about what their class registration process is like, as well as how receptive they would be to using technology to help better-plan their academic career (especially regarding what features they would need and want).

1.a.2 - Summarize Req'ts - After conducting interviews, condense and summarize the features requested and needs expressed, from most-desired to won't-implement.

1.b - Create Class Diagrams - Based on the requirements from [1.a.2], construct models for classes and describe them in UML Diagram format.

1.c - Process Model - Utilizing the class diagrams from [1.b], construct and compile use case diagrams and supplemental documentation for the use cases.

1.d - Object Behavior Model - Based on the requirements collected in [1.a.2] and diagrams constructed in [1.b-c], create a sequence diagram for the major use cases.

2 - **Design**

2.a - User Interface Design - Based on the requirements and needs collected in [1.a], extract the relevant information hierarchies and (using the process and behavior information gleaned from [1.b-d]) lay out interfaces accordingly.

2.b - Software Design - Using our team's expertise in languages such as HTML and CSS, we can program how our website will be used and how the application will be executed.

2.c - Controls Design - We will also be able to choose how the user can interact with our application.

2.d - Test Cases - Before it is launched, we need to create tests in order to fix bugs and other errors in the application.

Work Breakdown Structure Dictionary (continued)

3 - **Implementation (Prototype)**

3.a - Repo & Workplace Initialization - Setup of shared git repositories across devices.

3.b - Interface Implementation - Based on mockups, create and publish web frontend.

3.c - Database Collection - Scrape information from UTD resources into JSON database.

3.d - Application Logic Development - Write javascript logic to yield CourseExplorer behavior.

4 - **Implementation (Product)**

4.a - Collect feedback on Prototype - Including interviewees who expressed interest in the technology during requirements collection, ask potential users to offer their thoughts on the prototype and plans for the final product.

4.b - Bugfixes to Prototype - correct issues as they arise during testing (and from feedback).

4.c - Expanded App Logic Development - Extend javascript logic to yield -Planner behavior.

Meeting Minutes

04 Sep 2016 @ 7:11PM

- Business: App Idea Selection:
    - Attendance Tracking
    - Course/Schedule Planning Assistant
    - Grocery Store Mapper
- Selected: Course Planner
- TODO: Write Proposal (Cocco volunteers to write it, to be posted for member review)
- Deadline: #MS1 (Memo) due by 06 SEP 2016

15 Sep 2016 (group members met separately

- Business: discuss idea in further detail
    - Sketches, pros, cons, features of app discussed
    - WBS extra credit opportunity discussed (assigned: Parth Badhiwala)
    - Milestone 2 outline created, sections assigned
    - Deadline: WBS, due by 25 SEP 2016
    - Deadline: #MS2 (Project Charter & WBS) due 03 OCT 2016

14 Oct 2016 @ 4PM

- Business: Idea Clarification (Use Case Stories & UI Sketches)
- #MS3 Prep Discussion (due date is 9 NOV 2016)
    - Requirements doc done by end of week
    - Everyone submits 2 each (functional, non-functional)
- Standup after class on Monday
- TODO:
    - #MS3 Scratch Document -- Cocco
    - Requirements Document -- All Members
    - Use Case Diagram -- TBA (Cocco)
    - Class Diagram -- TBA (Badhiwala)
    - Sequence Diagrams -- TBA (Cocco)
- Deadlines:
    - 17 OCT 2016 -- 2+2 Requirements in #MS3 Scratch Doc (All Members)
    - 9 NOV 2016 -- #MS3 Due (Cocco)

19 Oct 2016 @ 11AM

- Requirements -- all group members submit two functional & non-functional requirement suggestions (brainstorming)
- User Stories -- Cocco shared these in the 14 Oct meeting, they're done
- Use Case Diagram -- first drafts assigned to Abraham, due by 2 Nov 2016 (collab w/ Starkey on UC Descriptions)
- Use Case Descriptions -- first drafts assigned to Starkey, due by 2 Nov 2016 (collab w/ Abraham on UC Diagram)
- Class Diagram -- first draft assigned to Badhiwala, TBD but tentatively due by 5 Nov 2016
- Sequence Diagram -- first draft assigned to Cocco, TBD but tentatively due by 7 Nov 2016

02 Nov 2016 -- Electronic meeting via group chat (status check for #MS3 tasks)

- On track to deliver before deadline, group members are swamped with exams and work in other classes.
- Revisions & Final Document -- Cocco will handle this & upload to Discussion Board
- Minutes -- Cocco will type these up and submit them with #MS3

08 Nov 2016

- Completed rough draft Use case and class diagram that was assigned to Parth and Stephen.

09 Nov 2016

- After class, met up and finished touch ups on the diagrams after being revised by all group members.

23 Nov 2016 -- Electronic meeting via group chat (poster production for Monday)

- Cocco producing PPTX & PDF deliverables for UTD printer
- Starkey to bring files for printing & hold poster until Monday

28 Nov 2016 -- 10:45AM

- Brief in-person meeting to view printed poster and discuss mounting options for Wednesday, to be taken care of Tuesday.
- #MS4 Due tonight (Cocco)
    - All members to submit at least 1 method, description & test case (50% done)
    - All members to submit ~400 word “Lessons Learned”
    - Compilation of previous documents (Cocco)
    - Previous #Milestone Content Updates (Team)
    - UI Graphics (Cocco, done)
    - Testing Strategy (Cocco)

Lessons Learned

Parth Badhiwala

Being a part of a team where everyone contributes something different is a team that I thoroughly enjoyed being a part of. Helping design CoursePlanner with my peers has taught me how to manage a project as well as how to work with others that offer a unique way of thinking.

This project allowed us to create and complete a dream that all of us have shared and a bond with members that think differently, yet share an equal amount of passion for this project. There are many things you can do with CoursePlanner and putting together a team with different minds allowed us to create this project. Aside from the material Dr. Owens taught us in class, I learned something new from my group members anytime we got together.

Working with my group members allowed me to learn about myself and what I can offer in a team setting. I tend to understand material better when I'm working alongside my peers and it helps that I was able to utilize the information taught to us by Dr. Owens in this project.

One of the many things that I learned was that even the smallest meetings with your group members makes a huge difference. We all have other commitments and different schedules, so outside of communication through GroupMe, we had meetings after class where we discussed things that needed to be completed or issues that should be resolved so that they won't happen in the next milestone.

I learned that it takes time to understand how other group members work and that everyone works differently. I learned that it isn't just splitting work amongst team members that get the project done, it's putting an equal amount of effort in every aspect of the project is what creates a successful project. An equal amount of effort was put in for anything that was assigned to each group member, which is what allowed us to complete each milestone.

I also learned that it is a good idea to have more than what is required so that you can look back and put together the most relevant and creative information. Working alongside Stephen, Matthew, and Logen has showed me what it takes to be a part of a successful team and how to be an effective team member.

I want to thank all my group members and Dr. Owens for helping me learn about myself and allowing me to use the material I learned so that I can be effective in the real world after college.

Logen Starkey

While working on the CoursePlanner project this semester there are multiple lessons and principles I will walk away with. Starting with communication and planning, they are the foundation to almost any project anywhere.

In the beginning our communication as a group was lackluster and somewhat thrown together, however after the first milestone we had a group meeting with different engaging discussions involving use case diagrams and descriptions that would help the group get on the same page early on.

Traditional meetings generally weren't available to our group because of different schedules and priorities that each of us face in our daily lives. However we were able to overcome these obstacles with the use of Google Drive and GroupMe, which enabled us to communicate daily and get different documents and diagrams completed in order to meet deadlines for milestones. The biggest take away for me is the effectiveness and use of different diagrams involved in the CoursePlanner project.

Although creating effective diagrams and descriptions of systems are time consuming tasks, if managed properly, the project as a whole will go much smoother, as it paints a vivid picture of how the system works and what it will accomplish. The use of effective visuals and descriptions can be applied to future work, because they truly helped me understand how our system would work and what it would do. With good documentation and diagrams, people working on the project are almost able to understand how a system works based on those documents alone.

Putting emphasis into the quality of these diagrams may be time consuming early, but will help bring cohesion and fluidity to the group. Ultimately we were able to meet all of our expectations and exceed a few as a group, with most credit being due to our project manager Matthew who kept us on track during the semester and would outline and delegate different tasks on the milestones to be completed.

Thank you Dr. Owens for providing us with the tools and information through our class lectures that would be utilized during the course of the semester to bring all of the parts in CoursePlanner together.

Stephen Abraham

I have had an enjoyable experience with my group members. Matthew, Logen, and Parth definitely have helped me and shaped me to become an efficient project member. Working on the CoursePlanner project let me understand that it takes time and effort to get a project done in a timely manner.

One of the things that worked well for us was communicating through GroupMe and Google Docs. We used GroupMe almost every day to communicate deadline, assign duties, scheduling group meetings, etc. We used Google Docs to write the milestones and create documents to help aid in our research and development. One thing that did not work for us was scheduling meetings properly.

Every time we scheduled a meeting, at the last minute, someone would say that they could not make it and give an excuse that may or may not have been true. So far, everyone has been able to do their part and contribute to the group. There have been times that I would have to keep asking the group members if they have contributed their part and they would eventually send in their information.

However, each member did an excellent job in the quality of their work and that greatly strengthened the group. In Google Docs, there is a feature call Versioning that helped us undo and redo changes we made in our documents, and that helped us a ton. We were able to get each milestone done a day before it is due, and that was something that we wanted to achieve. One of our goals as a group was making sure that we got all our work done early so that we can focus on other classes that we were taking. Another goal that we worked towards as a group was always providing feedback and giving encouragement.

Matthew is our Project Leader, so if we had any question whatsoever, he would answer it in a timely manner and also encouraged us on a job well done. Logen and Parth also did the same as well. We were able to attain all our goals throughout the semester, and we also developed a close friendship with each other.

This group has become my best friends, and I am glad that I was able to join them and work as a team. Thank you so much Dr. Owens for giving us the information to be able to delegate tasks needed in order to complete our milestones.

Matthew Cocco

Communication is key. (more) early, (more) often, and (more) clearly

Over the course of this semester, running the group project has taught me several things - the most important of which is that communication should be timely, regular, and clear. When any of these qualities is lacking, performance of the group as a whole suffers. From setting meeting times to preparing materials for meetings to running the meetings, it's important to have goals in mind and to plan such that the goals will be met. It is also of great importance to confirm in the affirmative that group members understand what one intends to communicate, not just what was said or written.

Delegating work is harder than it sounds. It's more than just tasking.

More than just sending someone a "please do _____" message, delegation is like teeing up a ball so a team member can knock it out of the park; it is setting them up for success while spreading the workload across the team. Effective delegation makes clear the expectations, responsibilities, and desired results. To make the most of a team, constructive feedback about delegated tasks (delivered in a timely manner) is critical.

Perfect is the enemy of Good. Trying to design perfection consumes more time than approximating the desired result and iterating on it.

Rapid Prototyping beats *Incredibly Deliberate + Thorough Design* to market (but Rapid Prototyping can incorporate elements of a deliberate + thorough design process). By creating rough drafts and sharing them for group critique and collaborative polishing, our team turns out content that is good faster than a single person would publish content that is perfect.

Running a lean project and working in an agile team necessitates useful & efficient meetings, as well as effective asynchronous communication outside of meetings.

This means preparing for the meetings in advance, spending time to schedule intelligently, and oftentimes communicating the *why* before the *what*. Preparing materials before meetings and sharing those materials in advance dramatically improves the outcome of such meetings.

Communicating an idea from one mind through a medium (or media) to another mind (accurately and intact) is quite possibly one of the most difficult exercises in exchanging information.

Investing in communicating ideas early and ensuring everyone has the same goals in mind aligns the team's interests; by getting buy-in from the team and getting everyone on the same page, the team avoids the friction and effort lost when members are working in different directions.