OnTarget

A Course Scheduling Tool for College Students and Advisors

Member(s): Belinda Copus,Team #24 (document created 2/17/2017)

1. Introduction

College students face many challenges transitioning to university life, including planning what courses to take each semester. Students are often unaware of the courses that need to be taken, what prerequisites are needed, and when these courses will be offered. It has been stated that academic advising is a vital link in retention (Drake, 2011). A logical conclusion could be made in that a student without a well- defined plan of study is less likely to complete a degree. Long range course offering schedules often are not available for students to access and create long range plans of study, independently, nor with any confidence that the courses they choose will be offered in the semester they believe it should be taken. Also, it is increasingly likely that a university will offer the same course on more than one campus and students often wish to take classes on more than one campus. Students must rely on a faculty or academic advisor to assist them in planning their coursework, but few students actually regularly meet with their advisor. Advisors also find the task of drafting a plan of study for the student to be quite tedious and time consuming. For an advisor to create an optimal plan that accounts for the many variables involved, is nearly impossible. Often, students have unique situations and constraints, adding even more complexity to the task. A software application that automates this effort is needed by both students and advisors.

2. Project Goal and Objectives (revised)

2.1 Overall Goal

OnTarget is an application that will facilitate the task of creating a student's college course plan. There is an additional side benefits to the departmental administrators, in that, a sense of future course demand can be determined by this application.

2.2 Significance/Uniqueness

There has been great effort made in creating automated advising tools, similar to this proposed system. There have been systems which are prescriptive, in that the student is provided a plan of study with little input. Others have created web-based decision support tools for academic advising is based on promoting engagement between the advisor and student (Feghali, Zbib, & Hallal, 2011). The system proposed is not necessarily unique in capabilities, but will be offered free and openly for those who would like to use the application. This system is superior to many readily available systems, in that it is able to consider distant future course offerings, whereas most currently marketed systems only consider a scheduling from a one-semester look ahead.

2.3 Objectives

This system is vital to saving time and providing a more accurate plan of study than what a person is able to create. It is the objective to develop and deliver this system for use by Fall 2017. While this proposed system is intended to be used by Computer Science students and faculty at a small university, it is possible that the application could be extended and adapted to other schools and degree programs.

2.4 System Features

The system will utilize the following data:

- Course catalog major requirements and general education requirements,
- Student's completed coursework and major/concentration,
- Course ranking per course in terms of time requirement outside of class,

- Course offering 5-year rolling plan, and
- Pre-requisite mapping.

This list comprises **the Foundational Data** used by the system.

A student will be able to generate a complete plan of study that leads to graduation. The student will be able to indicate which campus they prefer to take courses, the number of hours they wish to undertake during a semester, or whether they would like to take classes during the summer. General education requirements will be scheduled in addition to the requirements specified by the major. The student will also be able to specify load balance, in that the student can choose to balance course load between demanding courses and less demanding course. The plan will be stored so that it can be accessed by the student and/or advisor at some later time and by different hardware platforms.

The system will notify the advisor and student if a change is made in the 5-year rolling plan or in a pre-requisite requirement that will impact the current plan. Additionally, if the student is unable to obtain a sufficient grade in a course, the student and advisor will be notified and the plan can be automatically revised. If a student does not follow a plan for a particular semester, the system will automatically adjust the plan to reflect the shortfall.

The system will also track the number of students expected to take a particular course during a given semester to assist in allocating resources.

This system will web-based and be able to be utilized on multiple platforms, i.e. desktop, Android, and iOS platforms.

2.4.1 Specific Features

There are three primary types of users(actors) of this application. These include:

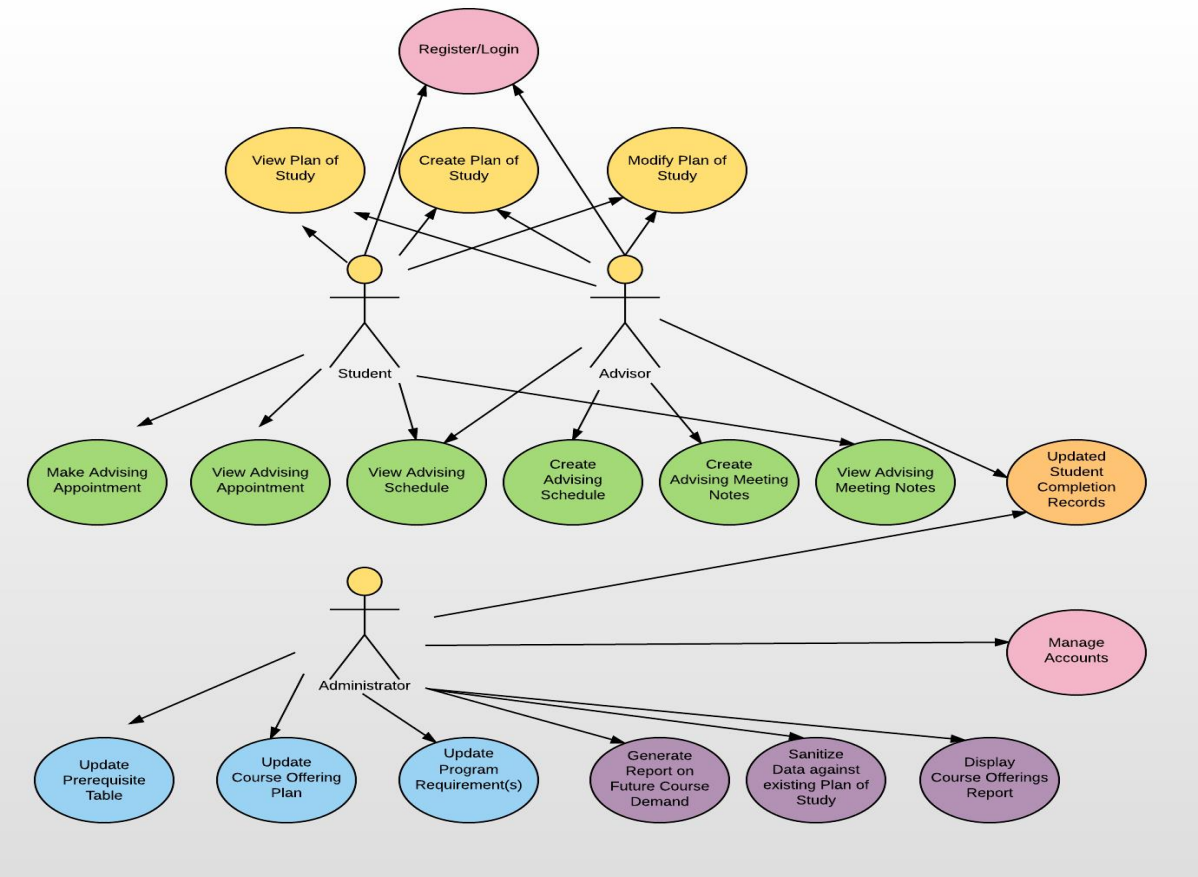
1) **Administrator** who are responsible for entering and maintaining the data used as input to the application,
2) **Advisor** who will review and modify student plans as needed, and
3) **Student** who will create a plan of study using the application.

Specific features of the application include:

1. The Administrator will be able to populate/modify data used as input (foundational data) to application, including:
   a) long range course schedule table,
   b) prerequisite lists, program requirements,
   c) initial student records entry,
   d) storage and maintenance of student plans created by the application.
2. The Administrator will also manage the accounts of users who have access to the application.
3. The Administrator will run sanity checks to verify that foundational data and created plans are still congruent.
4. The Administrator will be able to run a report showing the course demand for future semesters.
5. The Advisor will be able to:
   a) View a plan for a specific student
   b) Modify a plan for a specific student
   c) Schedule advising appointments
   d) Be notified if changes to foundational data impact any created student plans.

6. The Student will be able to specify information regarding campus preference, number of hours per semester and generate a plan of study.
7. The Student will be notified if the generated plan needs to be modified for reasons, such as, course offering schedule has changed, prerequisite course has changed or was not obtained.
8. The Student with visual impairment will have the option of having their created plan spoken to them, removing the need for the student to visually review their plan.

Use Case Diagram for Plan of Study System



Use Case Descriptions

Manage Accounts

General case for managing accounts. Cf. Add Account, Delete Account, Edit Account.

Update Course Offering Plan

Goal: Modify/Update course offerings to reflect the departmental schedule of course offerings. This schedule is a 5-year rolling plan that tracks course offerings over three semesters during an a academic year, i.e. Fall, Spring, Summer.

Preconditions: Changes to the departmental schedule has been made.

Successful End Condition: Course offering additions and deletions are reflected.

Failed End Condition: none.

Primary Actor: Administrator

Secondary Actor: none.

Update Prerequisite Table

Goal: Modify/Update prerequisite list to reflect changes in course prerequisites. Prerequisite changes impact all courses, regardless of academic catalog.

Preconditions: Changes to the prerequisite has been made by the department.

Successful End Condition: Prerequisite additions and deletions are reflected.

Failed End Condition: none.

Primary Actor: Administrator

Secondary Actor: none.

Create Program Requirements

Goal: Every academic year program requirements are adjusted. A new program entry will be made every academic year for the programs used by this system. This system will be tracking students under four academic catalogs and six programs under each catalog.

Preconditions: A new course catalog has been approved for the next academic year.

Successful End Condition: Course offering additions and deletions are reflected.

Failed End Condition: none.

Primary Actor: Administrator

Secondary Actor: none.

Update Course Offerings

Goal: Every academic year new courses are added. A new course entry will be made every academic year for the programs used by this system.

Preconditions: A new course catalog has been approved for the next academic year.

Successful End Condition: Course offering additions and deletions are reflected.

Failed End Condition: none.

Primary Actor: Administrator

Secondary Actor: none.

Update Program Requirements

Goal: Modify/Update degree requirements to reflect the departmental schedule of course offerings. This schedule is a 5-year rolling plan that tracks course offerings over three semesters during an a academic year, i.e. Fall, Spring, Summer.

Preconditions: Changes to an existing program has been made.

Successful End Condition: Program additions and deletions are reflected.

Failed End Condition: none.

Primary Actor: Administrator

Secondary Actor: none.

Sanitize Foundational Data Against Existing Plan of Studies

Goal: As changes are made in the foundational data – course offering schedule, program requirements, prerequisites, and student course completion, current Plan of Study could be impacted. There is a need to compare any changes to the plans and notify student and advisor of possible changes and/or issues.

Preconditions: Changes to foundational data has been made.

Successful End Condition: A list of students and courses are created for each potential issue discovered, if any.

Failed End Condition: none.

Primary Actor: Administrator

Secondary Actor: Student and Advisor.

Generate Report on Future Course Demand

Goal: Generate report displaying the number of students intending to take a course, by course and semester.

Preconditions: none.

Successful End Condition: Report is created.

Failed End Condition: none.

Primary Actor: Administrator

Secondary Actor: none.

Display Course Offerings Report

Goal: Generate a report that will display when a course is planned to be offered, by course and semester. The report should also be able to display the courses offered in a particular semester, or over the entire 5-year period.

Preconditions: Request must contain the course prefix to display a single course offering. A semester will be provided to display the courses in a single semester, and an option will be selected to display the entire 5-year period.

Successful End Condition: Report is created.

Failed End Condition: none.

Primary Actor: Administrator

Secondary Actor: none. Although providing this functionality to the Advisor and Student should be considered for a future revision.

Register/Login

Goal: Allow a new user to register an account. Allow an existing user to log into their account. A user can be a student, advisor, or administrator.

Preconditions: To register the user must already be included in the list of students. The student will provide a 700# student id number and password. The advisor will provide a 700# identification number and a password. To login, the student, advisor, or administrator will provide their 700# for a login name and their password for verification.

Successful End Condition: For registration – account is created. For login – user is allowed access to the system or access is denied.

Failed End Condition: none.

Primary Actor: Administrator, Advisor, and Student

Secondary Actor: none.

Create a Plan of Study

Goal: A plan of study is generated for the student based on the foundational data.

Preconditions: Student must be included in the list of students.

Successful End Condition: Report for a plan of study is displayed to the user and saved to the system.

Failed End Condition: none.

Primary Actor: Advisor and Student

Secondary Actor: none.

View Plan of Study

Goal: A plan of study is displayed based on a student 700#.

Preconditions: Student must have a plan of study previously created.

Successful End Condition: Report for a plan of study is displayed to the user. If no plan for the student exists, the user is notified of this condition.

Failed End Condition: none.

Primary Actor: Advisor and Student

Secondary Actor: none.

Modify Plan of Study

Goal: A course in a plan of study is modified based on a student 700#. This will allow courses to be manually moved to a different semester or to allow for the occasional exception to a course requirement. The user will identify the course to remove or move, the originating semester and destination semester, and a course number that will replace, if needed.

Preconditions: Student must have a plan of study previously created.

Successful End Condition: Report for a plan of study is displayed to the user. If no plan for the student exists, the user is notified of this condition. Updated plan is saved to the system.

Failed End Condition: none.

Primary Actor: Advisor and Student

Secondary Actor: none.

Make Advising Appointment

Goal: Students need to attend advising as often as every semester. Appointments can be made through this interface that will update the calendars of individuals involved.

Preconditions: Student must be listed in the list of students and be registered.

Successful End Condition: Appointment is scheduled and appears on the Student and Advisor calendar. The Advisor may choose to schedule the student, and if the Student is not a registered user, message appears to the Advisor.

Failed End Condition: none.

Primary Actor: Advisor and Student

Secondary Actor: none.

View Advising Appointment

Goal: Students need to attend advising as often as every semester. Scheduled Appointments can be viewed through this interface.

Preconditions: Student must be listed in the list of students and be registered.

Successful End Condition: Appointment schedule is displayed to the Advisor or Student. The Advisor may choose to schedule of a single student, and if the Student is not a registered user, message appears to the Advisor.

Failed End Condition: none.

Primary Actor: Advisor and Student

Secondary Actor: none.

Create Advising Schedule

Goal: Students need to attend advising as often as every semester. The Advisor must create a list of times in which the student can select to attend advising.

Preconditions: Student must be listed in the list of students and be registered. The Advisor must be a registered user of the system.

Successful End Condition: Appointment schedule is created and available to the Student and Advisor.

Failed End Condition: none.

Primary Actor: Advisor and Student

Secondary Actor: none.

View Advising Schedule

Goal: Students need to attend advising as often as every semester. Scheduled Appointments can be viewed through this interface. Advisors can view their entire schedule.

Preconditions: Student must be listed in the list of students and be registered. Advisor must also be a registered user of the system.

Successful End Condition: Appointment schedule is displayed to the Advisor. The Advisor may choose to view a schedule for a day, week, or month time period.

Failed End Condition: none.

Primary Actor: Advisor and Student

Secondary Actor: none.

Add Advising Meeting Notes

Goal: Advisors will need to maintain notes from advising meeting with a student.

Preconditions: Student must be listed in the list of students and be registered. Advisor must also be a registered user of the system.

Successful End Condition: Advising notes are stored to the system and associated with a plan of study.

Failed End Condition: none.

Primary Actor: Advisor

Secondary Actor: none.

View Advising Meeting Notes

Goal: Students and Advisors will need to refer to notes from an advising meeting.

Preconditions: Student must be listed in the list of students and be registered. Advisor must also be a registered user of the system.

Successful End Condition: Advising notes are displayed to the user.

Failed End Condition: none.

Primary Actor: Advisor and Student

Secondary Actor: none.

Update Student Completion Records

Goal: Student course completion data is vital to this system. At this time, it must be manually updated at the end of a semester. Courses that are successfully completed by the student are added to the Students completed coursework list.

Preconditions: Student must be listed in the list of students and be registered. Advisor must also be a registered user of the system.

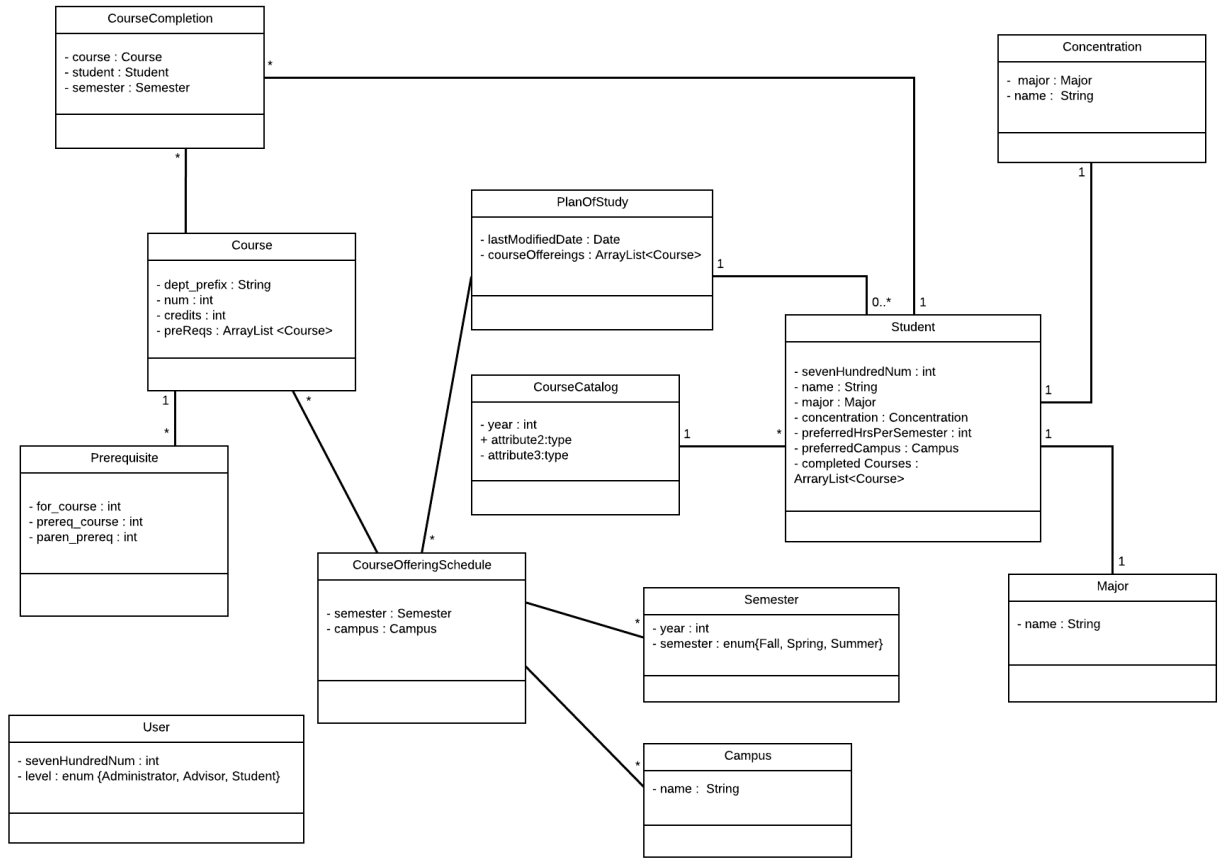
Successful End Condition: Data is updated appropriately.

Failed End Condition: none.

Primary Actor: Advisor

Secondary Actor: none.

Class Diagram for Plan of Study System



Schema for Plan of Study System Database

```
drop database if exists ontarget;
create database ontarget;
use ontarget;


create table Course (
  course_id int unsigned not null auto_increment,
  dept_prefix varchar(6) not null,
  course_num smallint not null,
  course_name varchar(50) not null,
  credit_hours smallint unsigned not null,
  primary key (course_id),
```

```
  index(course_id),
  index(dept_prefix, course_num)
);

create table Prerequisite (
  prerequisite_id int unsigned not null auto_increment,
  for_course int unsigned default null,    #fk Course.course_id
  prereq_course int unsigned default null, #fk Course.course_id
  parent_prereq int unsigned default null, #fk Prerequisite.prerequisite_id
  primary key (prerequisite_id),
  index(prerequisite_id),
  index(for_course)
);

create table CourseCompletion (
  completion_id int unsigned not null auto_increment,
  course int unsigned not null,   #fk
  student int unsigned not null,  #fk
  semester int unsigned not null, #fk
  primary key (completion_id),
  index(completion_id),
  index(student)
);

create table Major (
  major_id int unsigned not null auto_increment,
  name varchar(50) not null,
  primary key (major_id),
  index(major_id)
);

create table Concentration (
  concentration_id int unsigned not null auto_increment,
  name varchar(50) not null,
  major int unsigned not null, #fk
  primary key (concentration_id),
  index(concentration_id)
);

create table Campus (
  campus_id int unsigned not null auto_increment,
  name varchar(50) not null,
  primary key (campus_id),
  index(campus_id)
```

```
);

create table PlanOfStudy (
  pos_id int unsigned not null auto_increment,
  dateLastModified datetime default null,
  student int unsigned not null, #fk
  primary key (pos_id),
  index(pos_id),
  index(student)
);

create table Student (
  student_id int unsigned not null,
  l_name varchar(50) not null,
  f_name varchar(50) not null,
  major int unsigned not null, #fk
  concentration int unsigned not null, #fk
  preferred_load smallint default null,
  preferred_campus int unsigned default null, #fk
  primary key (student_id),
  index(student_id),
  index(l_name),
  index(f_name),
  index(l_name, f_name)
);

create table Semester (
  semester_id int unsigned not null auto_increment,
  academic_year smallint not null,
  semester_code smallint not null,
  primary key (semester_id),
  index(semester_id)
);

create table CourseOffering (
  offering_id int unsigned not null auto_increment,
  course int unsigned not null, #fk
  semester int unsigned not null, #fk
  campus int unsigned not null, #fk
  primary key (offering_id),
  index(offering_id),
  index(semester),
  index(campus)
);
```

```
create table CourseCatalog (
  catalog_id int unsigned not null auto_increment,
  catalog_year smallint not null,
  concentration int unsigned not null, #fk
  primary key (catalog_id),
  index(catalog_id),
  index(concentration)
);

create table CourseGroup (
  group_id int unsigned not null auto_increment,
  name varchar(50) not null,
  catalog_id int unsigned not null, #fk CourseCatalog.catalog_id
  parent_id int unsigned default null, #fk CourseGroup.group_id
  minimum_hours smallint unsigned not null,
  primary key (group_id),
  index(group_id),
  index(catalog_id),
  index(parent_id)
);

create table CourseMapCourseGroup (
  id int unsigned not null auto_increment,
  course_id int unsigned not null, #fk Course.course_id
  course_group_id int unsigned not null, #fk CourseGroup.group_id
  primary key (id),
  index (id),
  index (course_id),
  index (course_group_id)
);

create table DegreeRequirement (
  requirement_id int unsigned not null auto_increment,
  catalog_id int unsigned not null, #fk CourseCatalog.catalog_id
  concentration_id int unsigned not null, #fk Concentration.concentration_id
  course_group_id int unsigned not null, #fk CourseGroup.group_id
  primary key (requirement_id),
  index (requirement_id),
  index (catalog_id),
  index (concentration_id),
  index (course_group_id)
);
```

```
alter table Prerequisite
  add foreign key(for_course)
  references Course(course_id);

alter table Prerequisite
  add foreign key(prereq_course)
  references Course(course_id);

alter table Prerequisite
  add foreign key(parent_prereq)
  references Prerequisite(prerequisite_id);

alter table CourseCompletion
  add foreign key(course)
  references Course(course_id);

alter table CourseCompletion
  add foreign key(student)
  references Student(student_id);

alter table CourseCompletion
  add foreign key(semester)
  references Semester(semester_id);

alter table Concentration
  add foreign key(major)
  references Major(major_id);

alter table PlanOfStudy
  add foreign key(student)
  references Student(student_id);

alter table Student
  add foreign key(major)
  references Major(major_id);

alter table Student
  add foreign key(concentration)
  references Concentration(concentration_id);

alter table Student
  add foreign key(preferred_campus)
  references Campus(campus_id);
```

```
alter table CourseOffering
  add foreign key(course)
  references Course(course_id);

alter table CourseOffering
  add foreign key(semester)
  references Semester(semester_id);

alter table CourseOffering
  add foreign key(campus)
  references Campus(campus_id);

alter table CourseCatalog
  add foreign key(concentration)
  references Concentration(concentration_id);

alter table CourseGroup
  add foreign key(catalog_id)
  references CourseCatalog(catalog_id);

alter table CourseGroup
  add foreign key(parent_id)
  references CourseGroup(group_id);

alter table DegreeRequirement
  add foreign key(catalog_id)
  references CourseCatalog(catalog_id);

alter table DegreeRequirement
  add foreign key(concentration_id)
  references Concentration(concentration_id);

alter table DegreeRequirement
  add foreign key(course_group_id)
  references CourseGroup(group_id);

alter table CourseMapCourseGroup
  add foreign key(course_group_id)
  references CourseGroup(group_id);

alter table CourseMapCourseGroup
  add foreign key(course_id)
  references Course(course_id);
```

3. Project Plan

3.1 Schedule

Increment 1 (Due on 2/17/2017)

1. Draft Use cases
2. Draft Scenarios/Use Case Stories
3. Develop schema for the databases: course offering schedule, prerequisite list, student completed coursework, programs of study, student plans, courses
4. Design User Interface for entering data into the database
5. Implement User Interface for entering data into the database
6. Populate databases (course offering schedule, prerequisite list, student completed coursework, programs of study, and courses) with data using user interface
7. Test item 5 from this increment.

Increment 2 (Due on 3/10/2017)

1. Develop Student Page that allows student to specify campus and desired hours and button to create a plan.
2. Develop Student Page that allows student to retrieve their plan.
3. Develop Advisor Page that allows advisor to view a student's plan of study.
4. Develop Advisor Page that allows advisor to modify a student's plan of study.
5. Create Launch Page
6. Create Login Page
7. Create Registration Page
8. Test each item in increment 1 & 2
9. Update use cases and scenarios, if needed.

Increment 3 (Due on 4/7/2017)

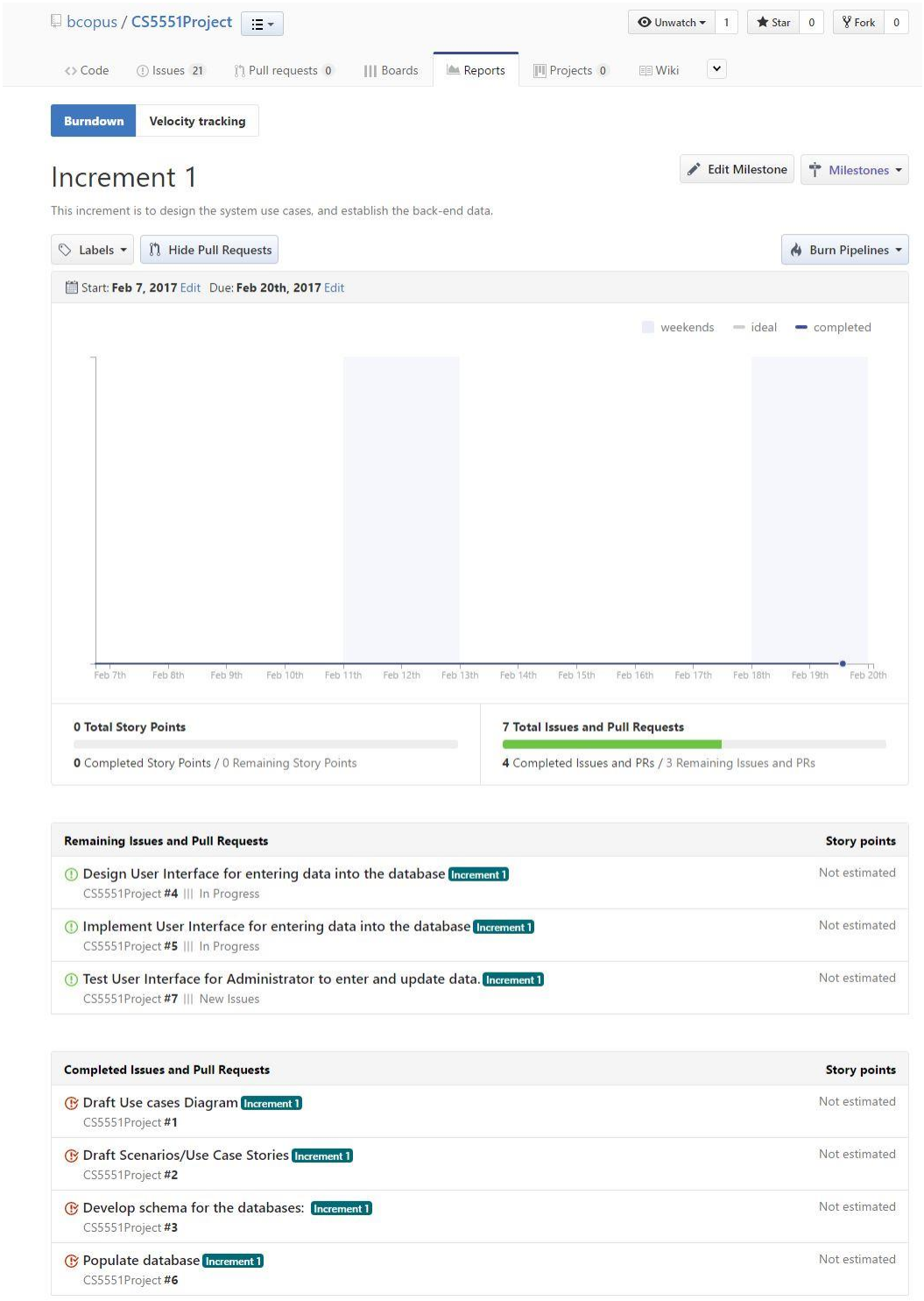
1. Add functionality to administrator check student plans if a change has been made in foundational data. This will also include notifying the advisor and student regarding potential issue.
2. Add functionality to student page and advisor page that allows for manual modification of a created plan.
3. Add functionality that will speak a student plan to the student.
4. Test each item in increment 1-3
5. Update use cases and scenarios, if needed.

Increment 4 (Due on 4/26/2017)

1. Develop report function for the administrator that displays demand for an upcoming course
2. Draft deployment plan
3. Final testing
4. Update use cases and scenarios, if needed.

3.2 Project Timelines, Members, Task Responsibility

3.3 Burndown Chart



4. First Increment Report

4.1 Existing Services/REST API

No services were used in this increment. API service of text to voice will be used in Increment 3.

4.2 Detail Design of Features

Please refer to section 2.4

4.3 Testing

N/A in this increment

4.4 Implementation (using Android/Angular.js/Bootstrap)

N/A in this increment

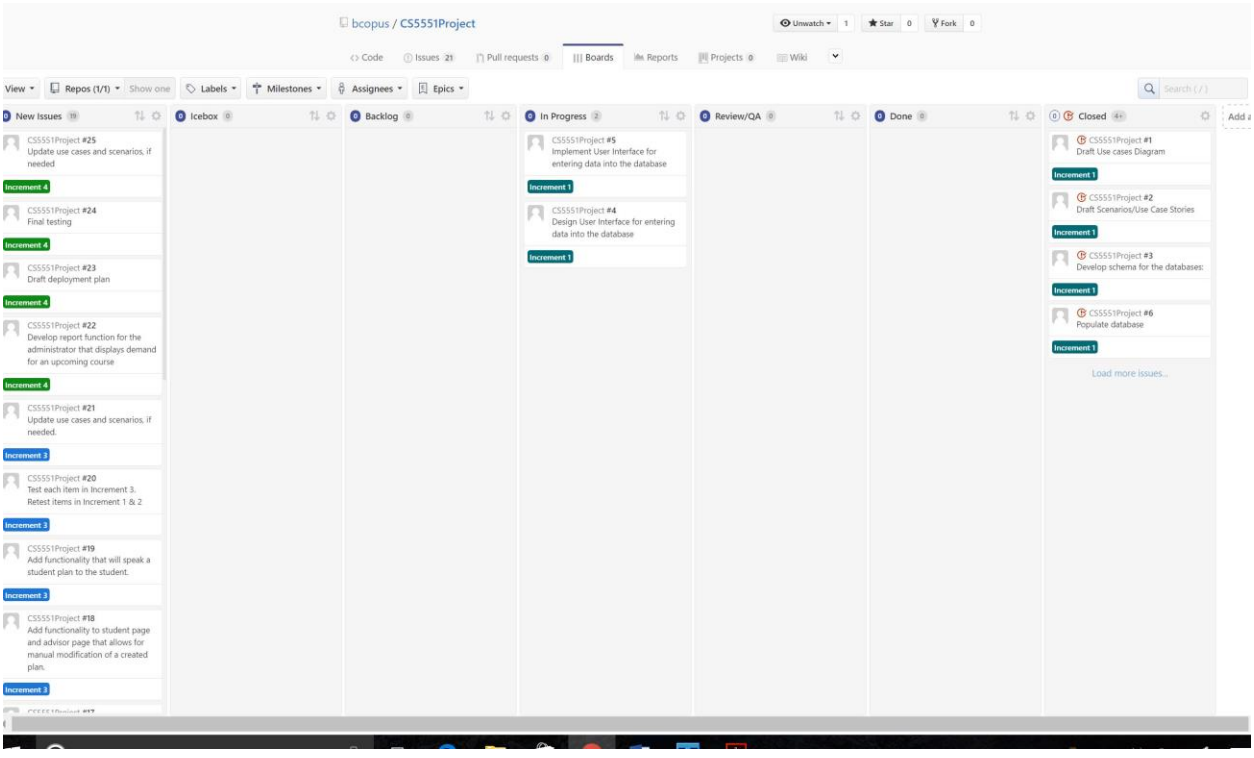
4.5 Deployment

N/A in this increment

4.6 Project Management

Please see section on Burndown Chart.

Increments and Issues entered into ZenHub.



4.7 Bibliography

Drake, J. (2011). The role of academic advising in student retention and persistence. About Campus. 16(3), 8-12.

Feghali, T., Zbib, I., & Hallal, S. (2011). A web-based decision support tool for academic advising. Educational Technology & Society. 14(1), 82-94.

Colleague Student Planning http://www.ellucian.com/Software/Colleague-Student-Planning/

Smart Planner http://www.tbginc.com/smart-planner/