CAMPUS CONNECT SCHEDULING

Easiest way to schedule an advising appointment

Abstract

Part 3 of CampusConnect homework.

Giries Hattar

Contents:

Overview

The campus connect feature to be developed is a Advising Appointment Reservation System within the academic Progress tab. The current features within the Academic Progress tab include advisors, degree programs, apply for graduation, apply for certification. With the scheduling feature, students would be able to view their advisors available dates and times in order to efficiently schedule an appointment to meet with them to discuss any school related questions and/or concerns.

Update

This shows document updates

02/19	Initial draft
02/20	Updated domain
02/21	Updated use case
02/22	Updated sequence diagram
02/23	Added class diagram
02/24	Updated class diagram and added object diagram
02/25	Added a state machine diagram
03/12	Added a activity diagram and design class diagram
03/13	Updated class diagram after doing design level class diagram, created a design level sequence diagram, and created a package diagram
03/14	Added deployment diagram

Key Users

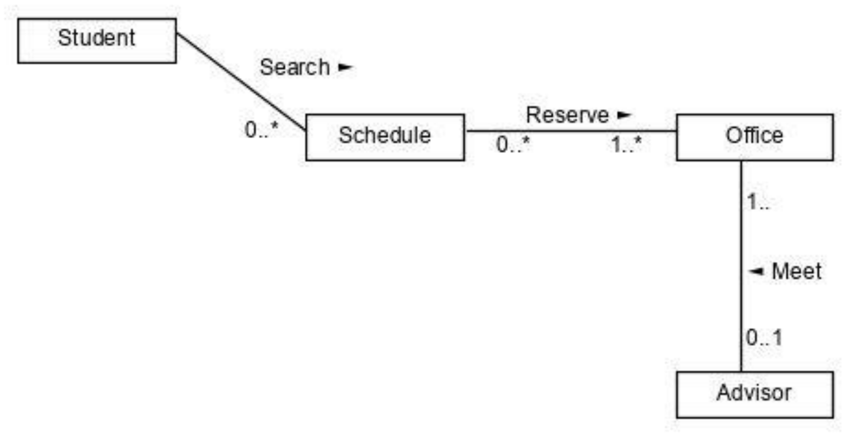
- ❖ Student
- ❖ Advisor

Features & Functions

- ❖ Student
  - ➢ Browse available dates and times
  - ➢ schedule appointment
  - ➢ Update/cancel current appointment
- ❖ Advisor
  - ➢ Set/sync availability
  - ➢ Update/Cancel appointments

1. Domain

This defines the domain of the system that is being built.



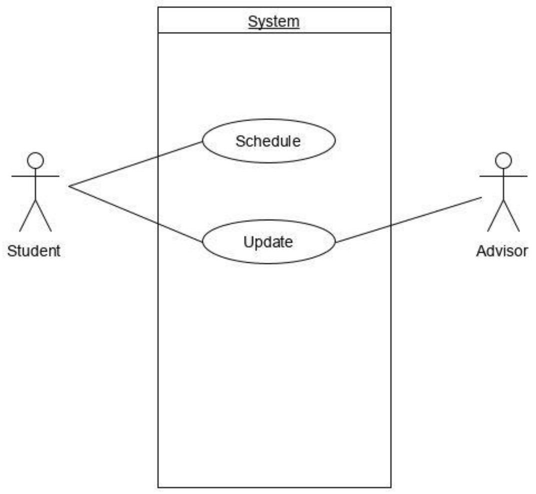
Glossary

Student	Student can be Undergraduate, Graduate, or anyone who is taking courses
Advisor	One who gives advice in a particular field

Analysis

This defines what problems we are trying to solve

Use Case Diagram:



Use Case:

Schedule

Actor

Student

Description

The objective of this use case is to provide a User the ability to select and confirm an advising appointment

Preconditions

The following activities must take place before the use case can be started.

1. User has access to the internet and can access the system
2. Requires an active campus connect profile.

Main Steps

The following is the relative priority of implementing the functionality required to allow this use case to be executed.

1. User searches for available date and time
2. User selects the desired available date and time.
3. System validates the date and time.
4. System generates confirmation.

Alternative Courses

The following are other usage scenarios that can take place within this use case independently from the initial schedule case.

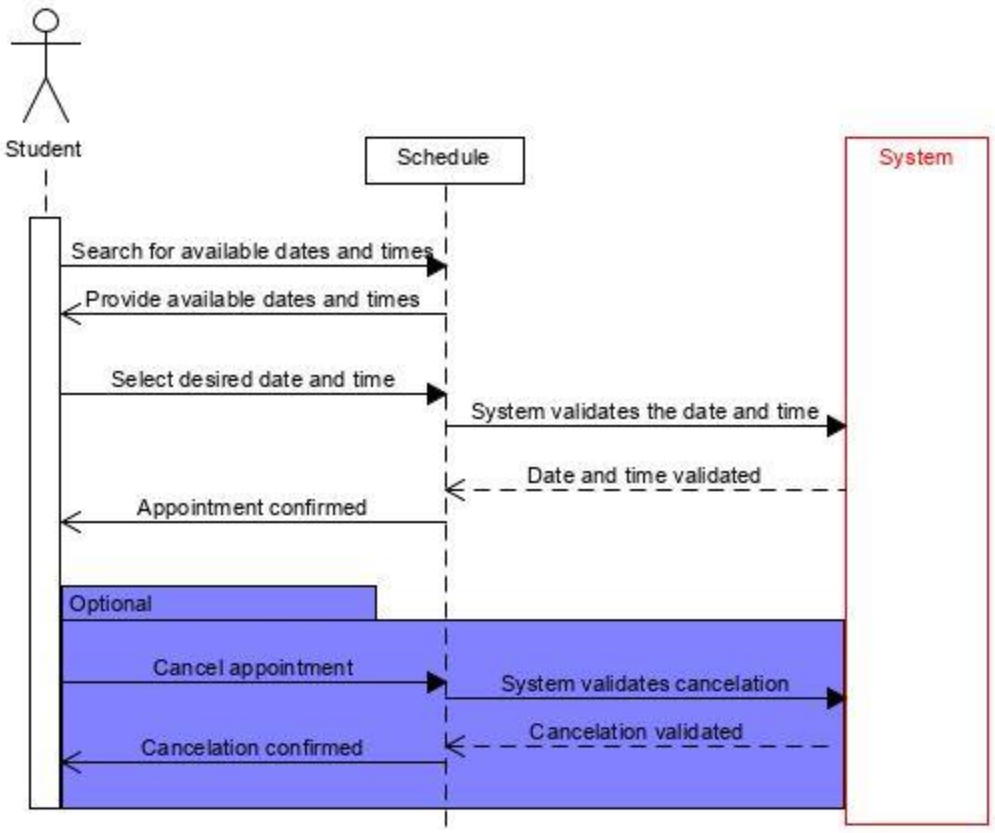
1. User would like to update their current appointment.

Postconditions:

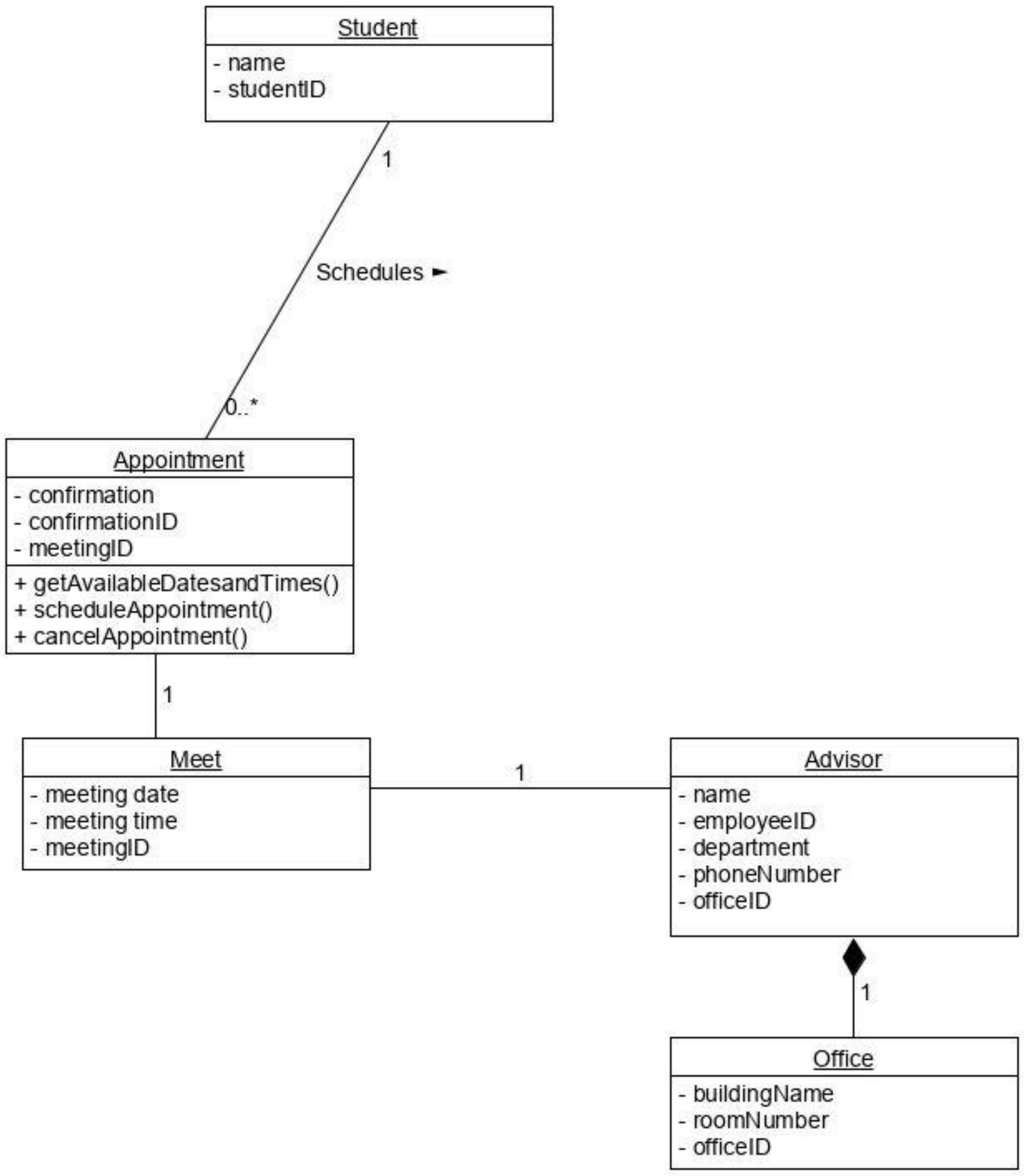
Describes the state of the system at the conclusion of the use case execution.

1. User will be able to view current appointments
2. Option to update or cancel an existing appointment

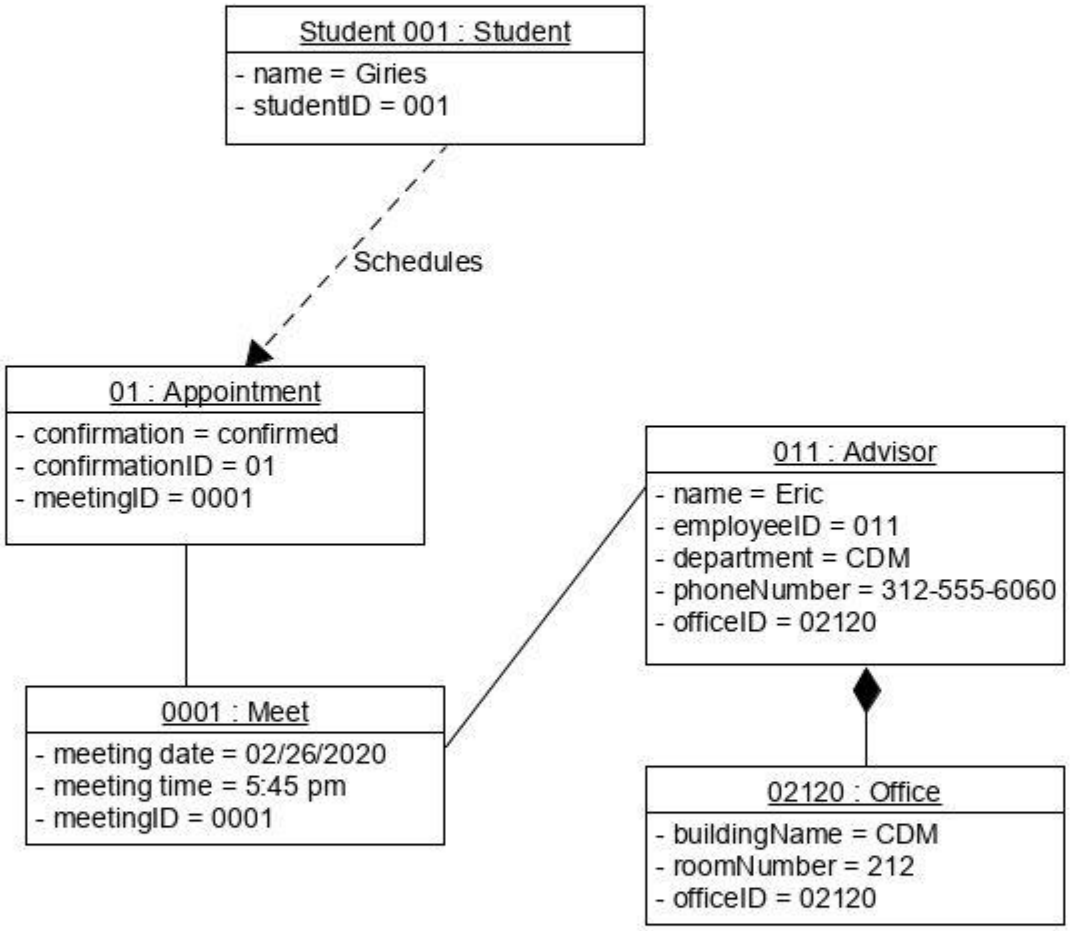
Sequence Diagram:


Student
Schedule
System
Search for available dates and times
Provide available dates and times
Select desired date and time
System validates the date and time
Date and time validated
Appointment confirmed
Optional
Cancel appointment
System validates cancelation
Cancelation validated
Cancelation confirmed

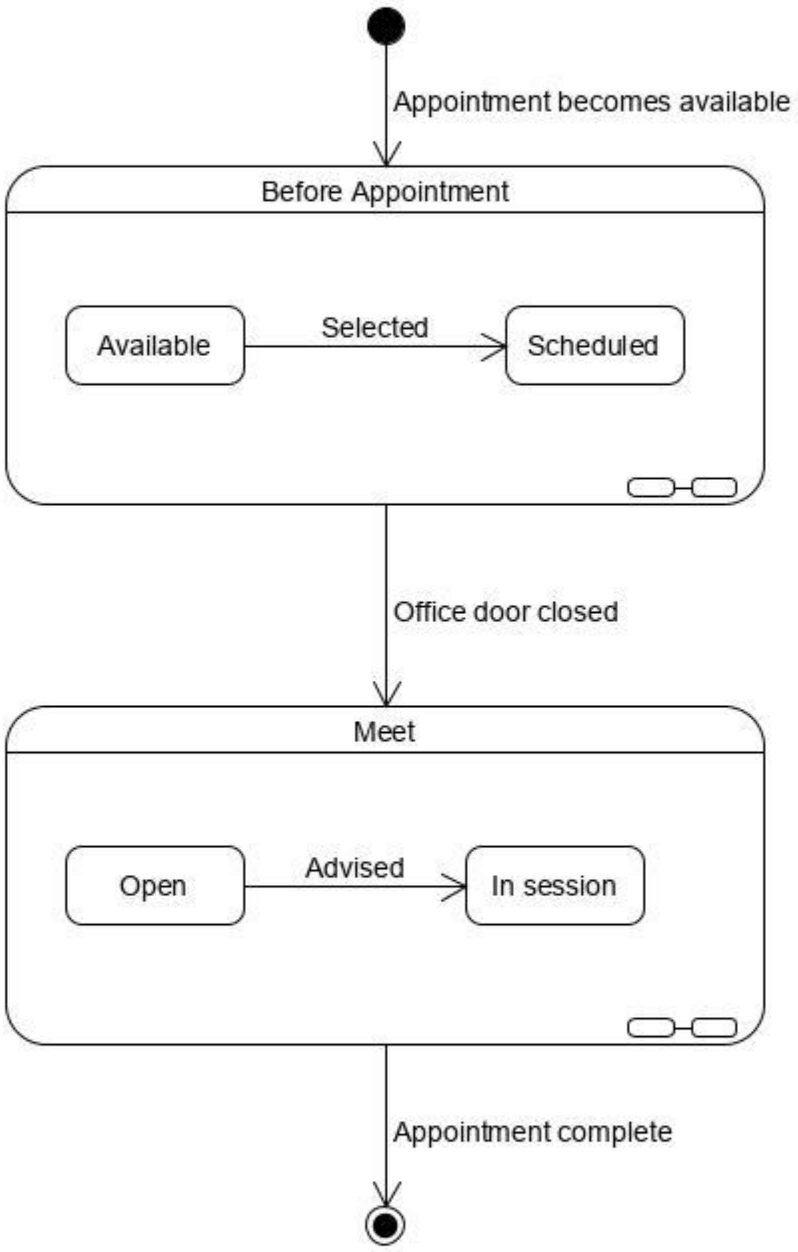
Class Diagram:



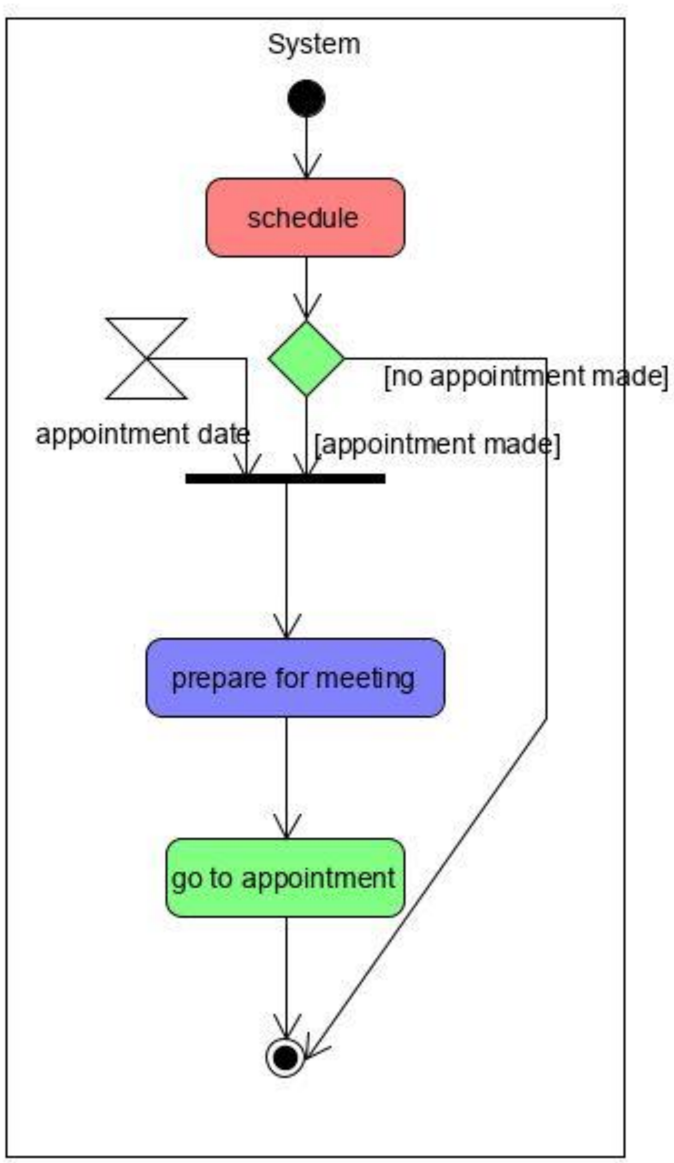
Object Diagram:



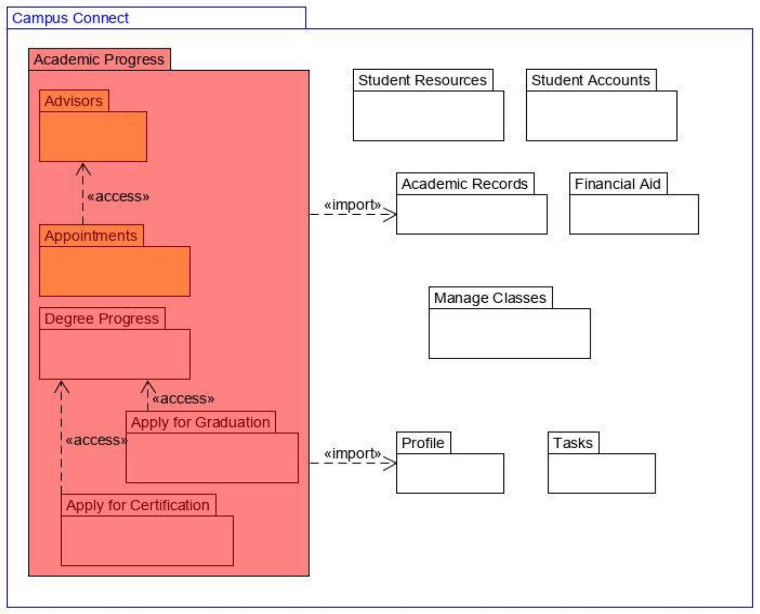
State Machine Diagram:



Activity Diagram:



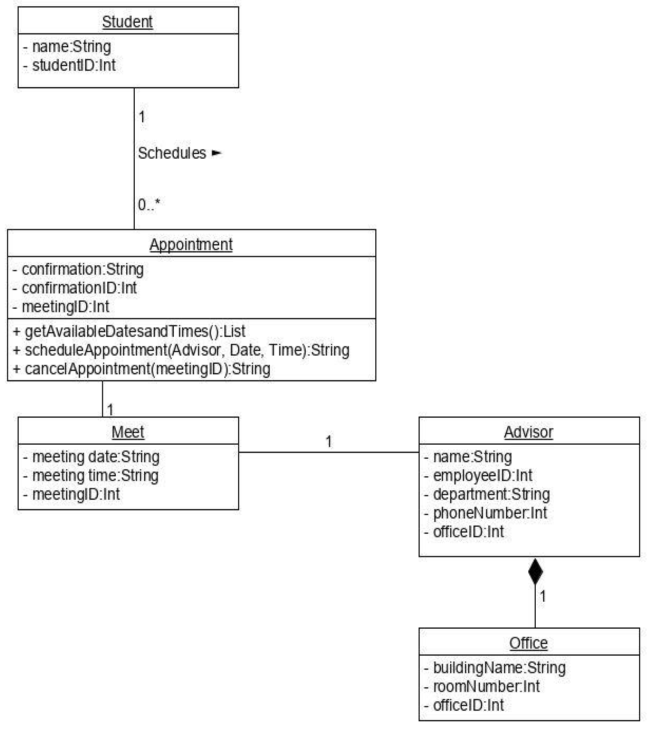
Package Diagram:



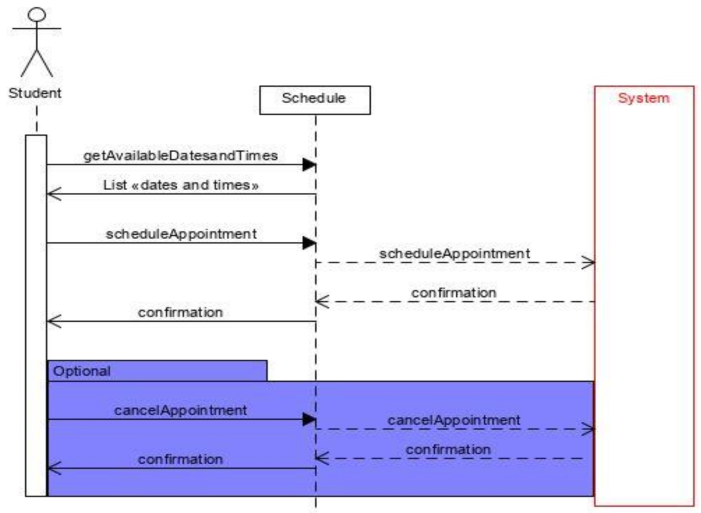
Design:

This defines insights on the implementation.

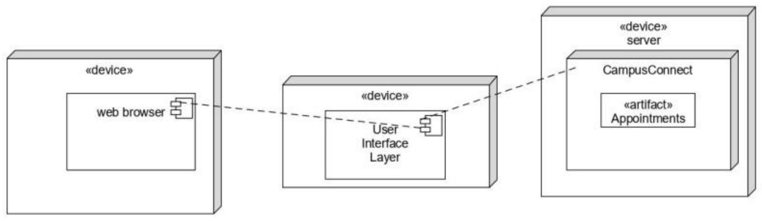
Design Class Diagram:



Design Sequence Diagram:



Design Deployment Diagram:



Non-Functional Requirements:

Non-Functional Requirement	Name at least 2 non-functional
Functional	System should have a UI that integrates to many different form factors such as a desktop, laptop, tablet and smartphone.
Usability	System should allow advisors to sync their availability from their existing Google, Yahoo, etc.. calendars.
Reliability	
Performance	
Supportability	

Lessons learned:

Date:	Lesson:
02/21	Always save a copy of the diagrams being created instead of recreating them over and over.
02/22	Learned how to apply arrows in umlet faster
02/23	Learned more about class diagrams and how to make them easier to understand
02/24	Learned more about object diagrams and in the process thought of new things to add to my class diagram
02/25	Learned how to make a state machine diagram
03/12	Learned how to make a activity diagram and design class diagram
03/13	Learned how to make a design level class diagram, after which I had to update my class diagram. I also learned how to make a package diagram.
03/14	Learned how to make a deployment diagram