Requirements Engineering Report

Date: 10/29/2018

Version:

[You can add an abstract or other key statement here. An abstract is typically a short summary of the document content.]

\*\*\*Team Name

UNCW Schedule Generator

Team Members

Chase Robison

Mark Thompson

Dakota Perryman

Contents

[1. Introduction 1](#_Toc529141895)

[1.1. Project Objective 1](#_Toc529141896)

[1.2. Project Scope 1](#_Toc529141897)

[1.3. Success Criteria 1](#_Toc529141898)

[2. Project Plan 2](#_Toc529141899)

[2.1. Work Breakdown Structure (WBS) 2](#_Toc529141900)

[2.2. Project Resources 2](#_Toc529141901)

[2.3. Responsibility Matrix 3](#_Toc529141902)

[2.4. Pert Chart 3](#_Toc529141903)

[2.5. Gantt chart 4](#_Toc529141904)

[2.6. Cost Estimation 5](#_Toc529141905)

[2.6.1. Function Point Estimation 5](#_Toc529141906)

[2.6.2 Lines of Code Estimation 7](#_Toc529141907)

[2.6.3 Cost Estimates 7](#_Toc529141908)

[2.7 Risk Plan 7](#_Toc529141909)

[2.8 Project Monitoring and Control Mechanisms 7](#_Toc529141910)

[3 Requirements/Analysis Models 8](#_Toc529141911)

[3.1 Major Software Functions 8](#_Toc529141912)

[3.2 Use Case Diagrams 8](#_Toc529141913)

[3.3 Use Case Descriptions (Fully Dressed) 13](#_Toc529141914)

[3.4 Activity Diagrams 14](#_Toc529141915)

[3.5 Sequence Diagrams 14](#_Toc529141916)

[3.6 Requirements Class Models 15](#_Toc529141917)

[3.7 Prototype Description 16](#_Toc529141918)

[3.8 Data Dictionary 16](#_Toc529141919)

[3.9 Limitations and Constraints 16](#_Toc529141920)

[3.10 Non-functional Requirements 17](#_Toc529141921)

[4 Problems Encountered 18](#_Toc529141922)

[5 Bibliography 19](#_Toc529141923)

# Introduction

## Project Objective

To develop a website that provides a user-friendly atmosphere to UNCW students for aiding in the process of making efficient class schedules. This project will take in a number of course subjects with associating course numbers and then generate a schedule based on those constraints and others specified criteria by the user. The result will be a course schedule displayed to the user with no time conflicts, if possible.

## Project ScoPe

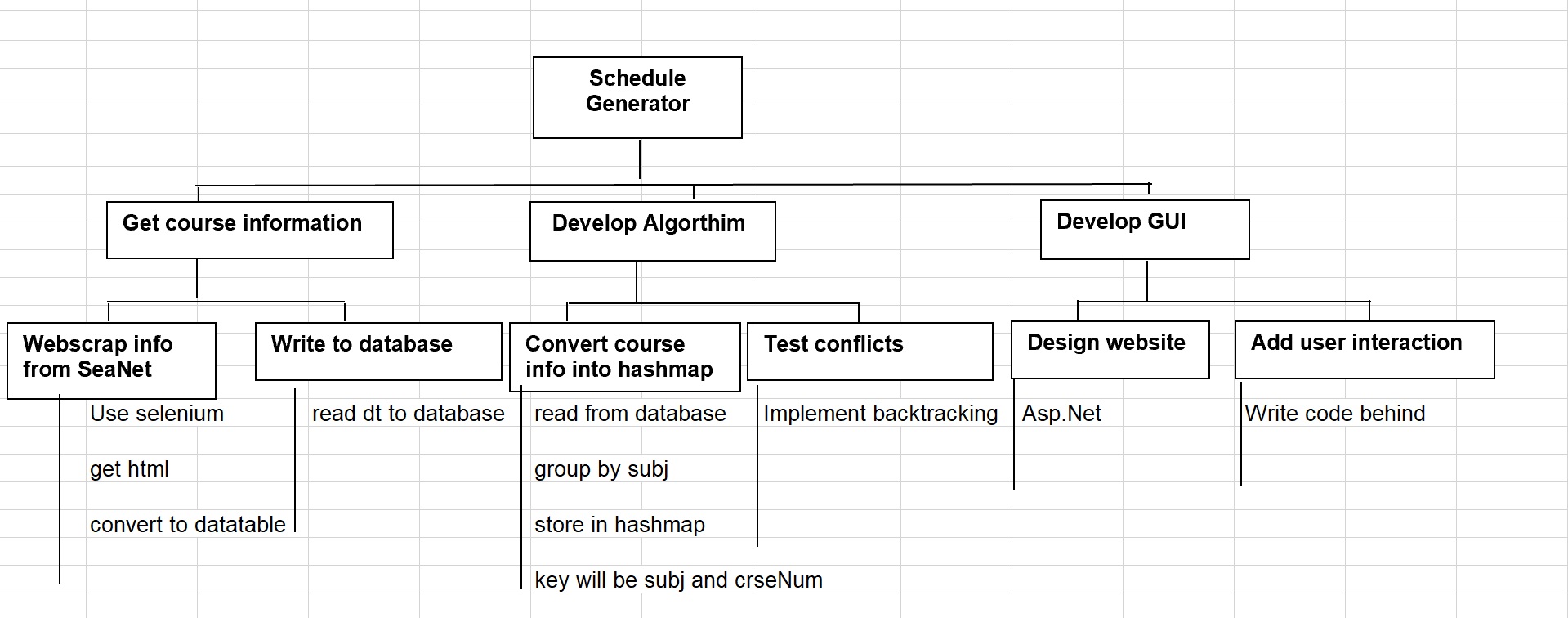
Taking the time to construct numerous schedule permutations is a mental burden that puts unnecessary stress on a college student, when they are still working through their previous semesters course load. The process of generating a schedule that has no conflicts and fits all of the students’ needs is a very time-consuming process. The current process consists of writing classes down on paper and manually checking conflicts by comparing them against other classes on SeaNet. With this tool, we hope to reduce time spent finding the perfect schedule by taking this burden off the students and implementing this problem into our own unique algorithm. Once our algorithm finds an optimal schedule, if one exists, the user will be presented with all their desired courses in a structured weekly format.

## Success Criteria

* Use standalone application to gather information from SeaNet
* Update database with SeaNet information
* Have a functional website
* Take user input for desired courses
* Return a schedule with no time conflicts based on input
* Allow user to make multiple searches

# Project Plan

## Work Breakdown Structure (WBS)



## Project Resources

* Visual Studio
* Selenium
* HTMLAgilityPack
* MySQL
* SeaNet
* GitHub

## Responsibility Matrix

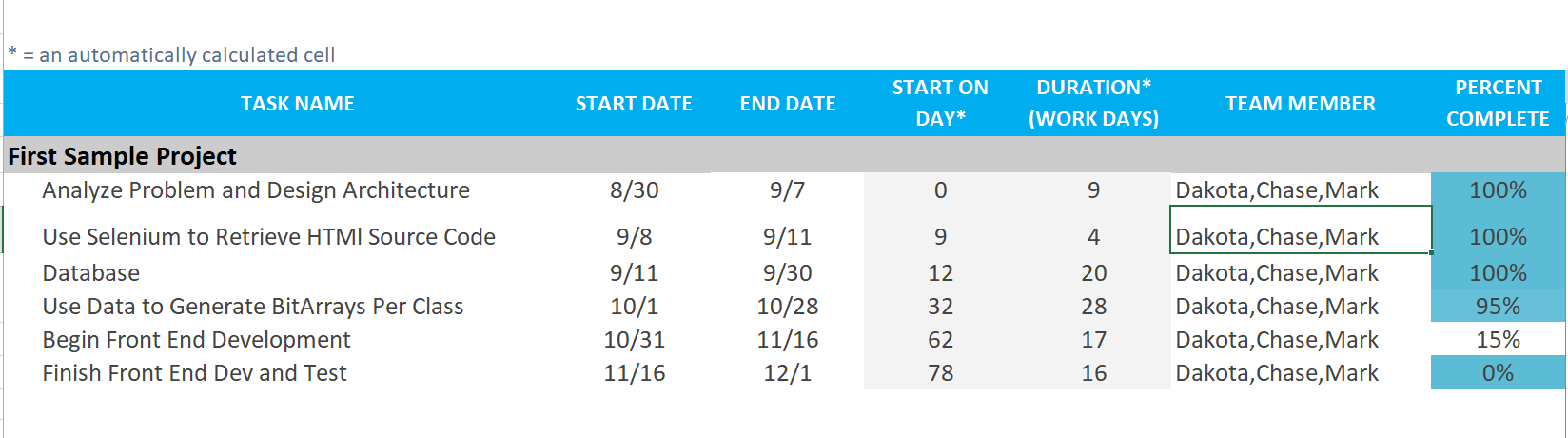
|  |  |  |  |
| --- | --- | --- | --- |
|  | Chase | Mark | Dakota |
| Database | I | I | R |
| Time Conflict Algorithm | I | R | C |
| WebPage | R | I | C |
| StandAlone App | R | C | C |

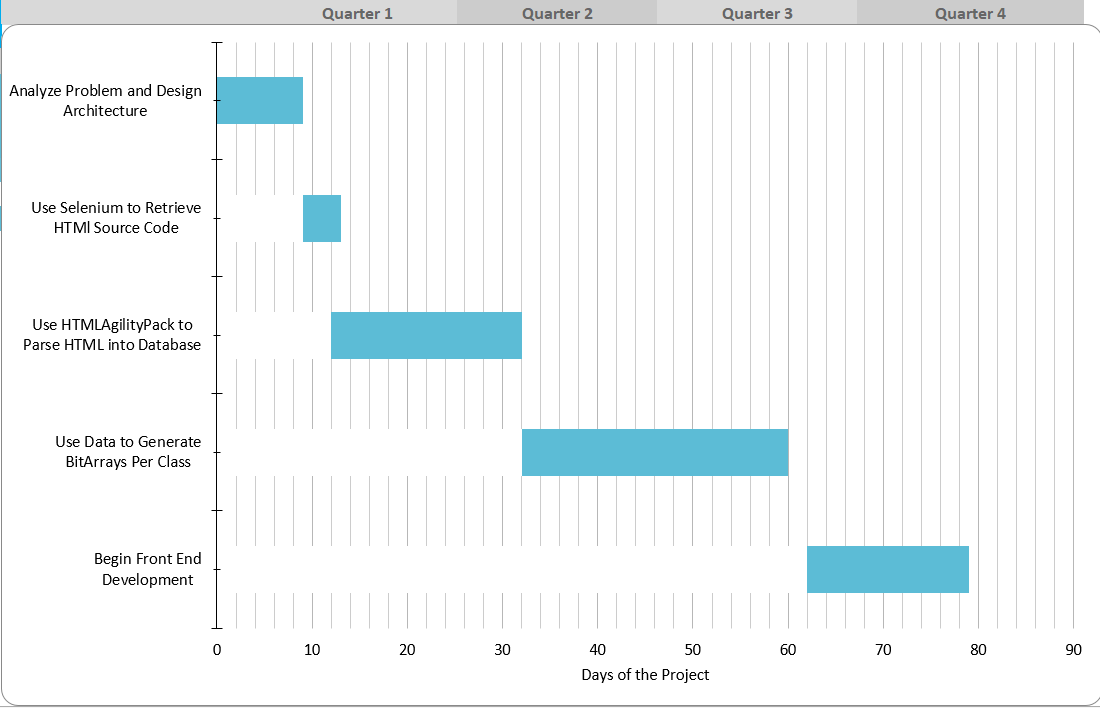
R- Responsible A-Accountable C-Consulted I-Informed

## Pert Chart

## 

## Gantt chart





## Cost Estimation

### Function Point Estimation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Weighting Factor | | | Count |
| Simple | Average | Complex |
| Inputs | Input subject and course numbers or CRNs |  | 4 |  | 8 |
| Input unavailability for times and days |  | 4 |  |
| Outputs | Display the weekly class time schedule |  | 5 |  | 8 |
| Display information for online classes | 3 |  |  |
|  | View course information by CRN |  | 4 |  | 4 |
| Files | Schedule generation |  | 10 |  | 17 |
| Schedule conflict testing | 7 |  |  |
| Interfaces | Application server to database |  |  | 10 | 20 |
| User to application server |  |  | 10 |
| **Total UFP** |  |  |  |  | **58** |

|  |  |  |
| --- | --- | --- |
| Number | Complexity Weighting Factor | Value |
| 1 | Backup and recovery | 1 |
| 2 | Data communications | 4 |
| 3 | Distributed processing | 3 |
| 4 | Performance critical | 1 |
| 5 | Existing operating environment | 1 |
| 6 | On-line data entry | 4 |
| 7 | Input transaction over multiple screens | 0 |
| 8 | Master files updated online | 1 |
| 9 | Information domain values complex | 1 |
| 10 | Internal processing complex | 3 |
| 11 | Code designed for reuse | 2 |
| 12 | Conversion/installation in design | 1 |
| 13 | Multiple installations | 1 |
| 14 | Application designed for change | 1 |
|  | **Total complexity adjustment value** | **24** |

FP = 58 \* (0.65 + (0.01 \* 24))

FP = 58 \* 0.89

FP = 51.62

### 2.6.2. Lines of Code Estimation

* SLOC = 51.62 (Adjusted Function Points) \* 54 (LF for C#)
* SLOC = 2787

### 2.6.3. Cost Estimates

PM = 2.4 \* (KDSI) 1.05

KDSI = 2.79 KLOC

PM = 2.4 \* (2.79) 1.05 = 7.05

TDEV = 2.5 \* (7.05) 0.38 = 5.25

## Risk Plan

* None

## Project Monitoring and Control Mechanisms

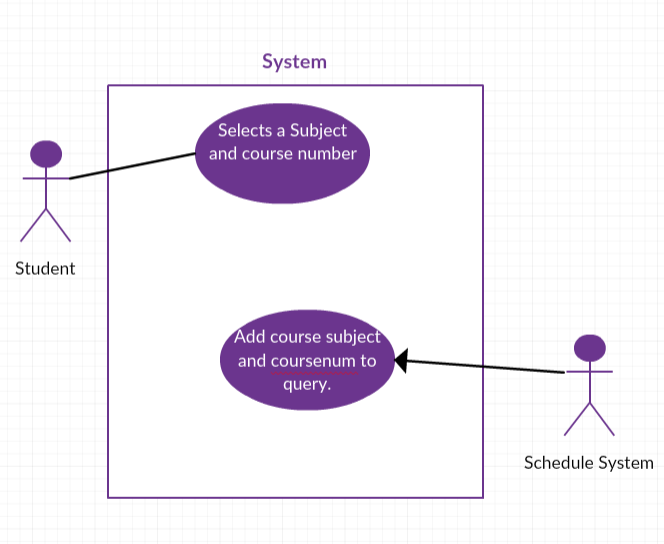
* None

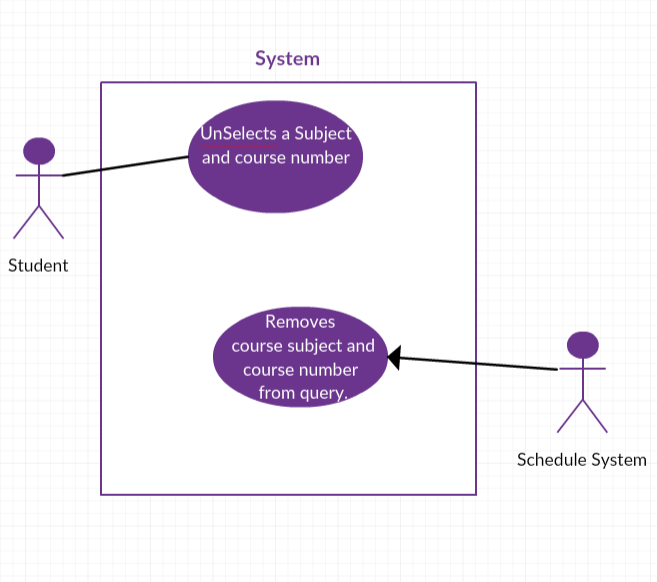
# Requirements/Analysis Models

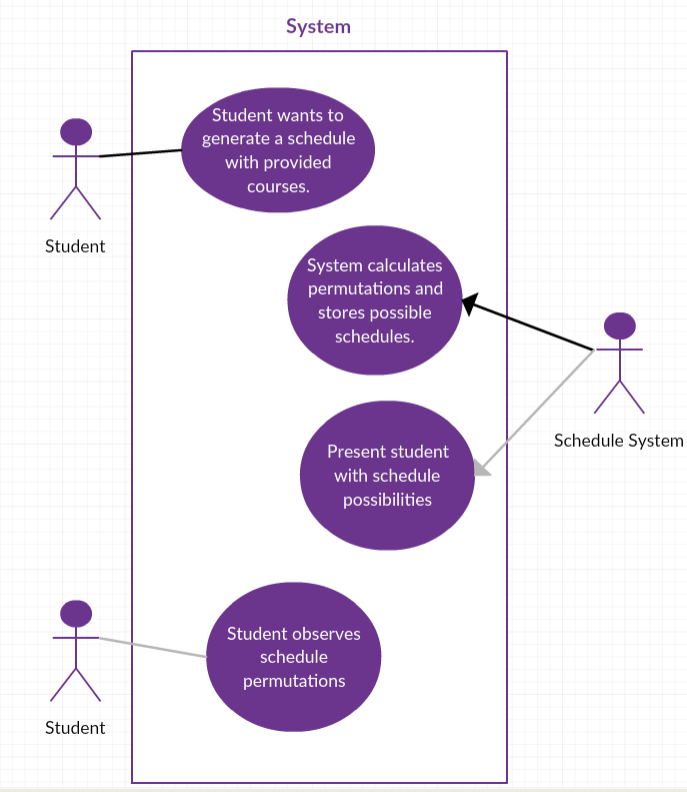
## Major Software Functions

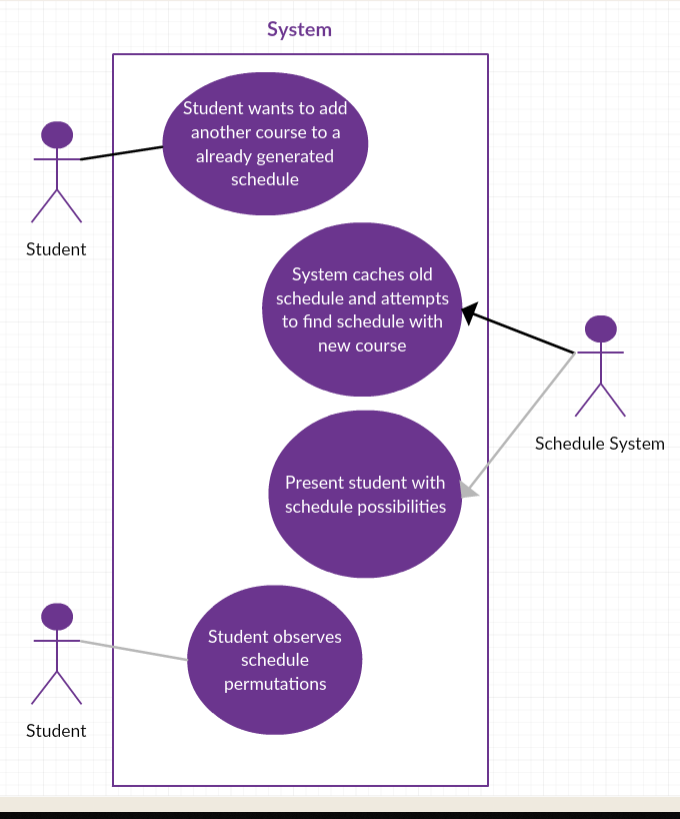
* Allow a student to select multiple courses.
* Generate schedule permutations for a student with no conflicts.

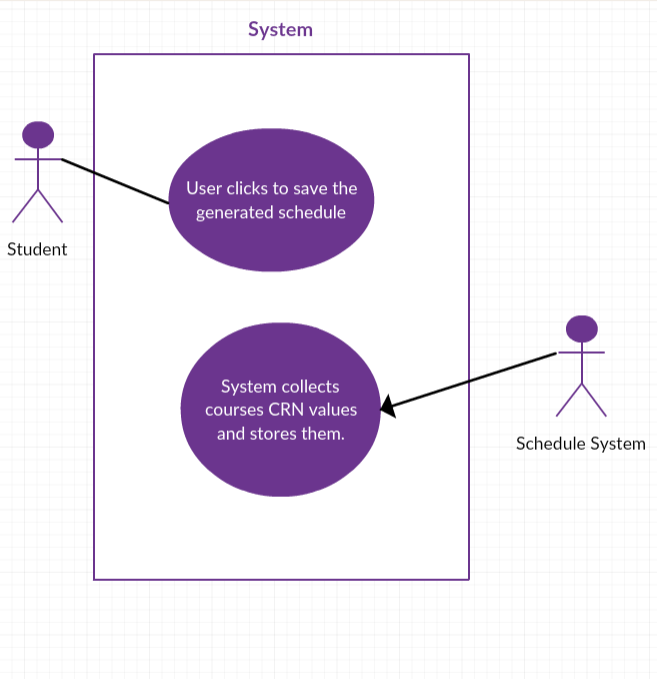
## Use Case Diagrams

  
(Add a class use case)

  
(Remove a class use case)

  
(Generate schedule use case)

  
(Add a class to a generated schedule use case)

  
(Save a schedule use case)

## Use Case Descriptions (Fully Dressed)

1. **Use Case:** Add a Class  
   **Actors:** Student and System  
   **Type:** Primary and Essential  
   **Description:** The student wants to add a course to be factored into a schedule generation process.   
   **Cross Ref:** None  
   **Use-Cases:** None
2. **Use Case:** Remove a Class  
   **Actors:** Student and System  
   **Type:** Primary and Essential  
   **Description:** The student wants to remove a course from the schedule generation process.   
   **Cross Ref:** None  
   **Use-Cases:** None
3. **Use Case:** Generate a Schedule  
   **Actors:** Student and System  
   **Type:** Primary and Essential  
   **Description:** The student wants to generate a schedule with the courses they provided.  
   **Cross Ref:** None  
   **Use-Cases:** None
4. **Use Case:** Add a Course to Generated Schedule  
   **Actors:** Student and System  
   **Type:** Primary and Essential  
   **Description:** The student wants to add a course to an already generated schedule.  
   **Cross Ref:** Add a Class  
   **Use-Cases:** Add a Class
5. **Use Case:** Save a Generated Schedule to User Account  
   **Actors:** Student and System  
   **Type:** Primary and Essential  
   **Description:** The student wants to save one of the generated schedule permutations presented to them.   
   **Cross Ref:** None  
   **Use-Cases:** None

## Activity Diagrams

A picture containing screenshot

Description generated with very high confidence

## Sequence Diagrams

A picture containing screenshot

Description generated with very high confidence

## Requirements Class Models

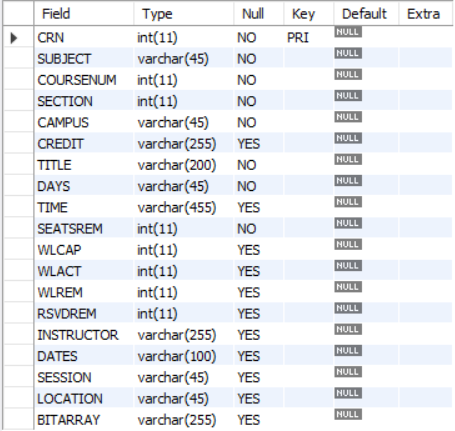
**Use Case: A student should be able to customize the courses they want in their schedule.**  
**Actor: Student**Our schedule generation application for students would be pointless without the ability of students to add courses desired courses they will want in their custom schedule. To enable this feature, our application must support input for a Subject (CSC) and Course Number (101). We must also be able to support CRN lookup for courses a student would like to add.   
  
The application also needs to support a student that may have selected the wrong course and wishes to remove it from consideration of schedule generation.

**Use Case: A student should be able to click a button and have a schedule presented.  
  
Actor: Student and System**Citing our success criteria, unless the student is presented with a schedule with no conflicts, then our application will not be successful. When the student clicks a button to generate the schedule, the application must be able to return in a reasonable amount of time one or more possible solutions to their schedule recommendations.   
  
  
**Use Case: A student should be able to save a schedule they like.  
  
Actor: Student and System**If our application is able to process their requests of certain courses and return schedules without conflicts, our user should have access to save the schedule permutation and be able to access it later in the session. This will be accomplished by enabling the user to create a database user and saving the course CRN values to the userID. This way, when the user logs in next session, they can pull up their schedule and ensure that this is the schedule permutation that they want for their next semester.

## Prototype Description

* None

## Data Dictionary

  
 User ID varchar(100) YES NULL  
 Password varchar(100) YES NULL

## Limitations and Constraints

* The patience of a user:
  + A student only wants to wait a few seconds for a schedule generation. In this case, we can only return a few options of the number of permutations. If this does not happen, a user may be waiting minutes for a generation.
* Student must be on a network connection
* Student must only search for under graduate courses at UNCW:
  + We are limiting the number of courses to only under graduate because of the deadlines of the course.
* A student must only be able to select a limited number of courses.

## Non-functional Requirements

* Accessibility
  + Enable any UNCW student on campus or around the world to access our website and create a schedule with ease.
* Documentation
  + Create quality documentation, incase our project is continued after the course ends to ensure quality over the lifecycle of the software.
* Usability
  + A website must be usable on the front end in order for it to be relevant in today’s atmosphere.
* Adaptability
  + Quality software must be able to adapt with the times and continue to be relevant over multiple semesters.
* Scalability
  + With good coding conventions, this system is easily scalable to other universities.

# Problems Encountered

* Parsing HTML source code.
* Dealing with courses that have certain days or times online.
* Converting courses times and days into bitarrays for quick calculations.

# Bibliography

References:

Selenium. Browser Automation. Retrieved August 21, 2018, from https://www.seleniumhq.org/

Software Management Solutions. FUNCTION POINT ANALYSIS: Counting Function Points Reference Guide. Retrieved November 4, 2018, from <https://people.eecs.ku.edu/~hossein/811/Papers/fpa-ref.pdf>