**Test Plan**

**Office Hours Scheduling System**

**Andrew McCracken and James Gruss**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Issue** | **Description** | **Author** |
| 11/29/2016 | 1,0 | Initial Version | JG, AM |
| 12/15/2016 | 1.1 | Updates: Approach, deliverables | JG, AM |
|  |  |  |  |

# 

# Overview

## Outline

a) Test plan identifier;

b) Introduction;

c) Test items;

d) Features to be tested;

e) Features not to be tested;

f) Approach;

g) Item pass/fail criteria;

h) Test deliverables;

i) Risks and contingencies;

# 1 Test plan identifier

Test Plan Unique Identifier: OO-1.0

2 Introduction

This test plan has been created to test subsets of functionality of the Office Hours System - an office hour scheduling system created by students of Gannon University. Much of the functionality was created in the Spring of 2016, but some was created during the current semester - fall of 2016. This plan does not intend to cover every unit of the software, but intends to formulate the testing plan for those units that were developed or vastly improved during this current semester. Furthermore, the testing will include system acceptability testing, specifically as it relates to the functionality tested during unit testing.

### Functionality to be Unit Tested:

* Login Manager (LoginManager.cs)
  + Create New User
  + Check Password
* Email Sender (EmailSender.cs)

### Functionality tested (additionally) via System Acceptance testing:

* Welcome Page
* Calendar Page
* Unit test components (seen above)

### Project Document References:

* Project Proposal: “Project\_Proposal\_OfficeHours”
* Software Quality Assurance Plan: “Software\_Quality\_Assurance\_Plan\_OfficeHours”
* C# Coding Standards Document: https://msdn.microsoft.com/en-us/library/ff926074.aspx

Previously created tests for other aspects of the project can be found in the project folder under “test\_cases/old\_test\_cases”.

3 Test items

Login Manager (LoginManager.cs)

This item has been revised multiple times throughout the lifecycle of our project. One can find a whole version history of the item on our public GitHub page. For this iteration of the project (this class) the item was fully implemented. Before this class, some of the functionality was written, but it was not tested nor in use in the application. The functions within the component were rewritten and extended. Login Manager also interfaces with an MS SQL database which is on a Gannon ITS Server.

The functions are:

Create New User

Check Password

Email Sender (EmailSender.cs)

This item interfaces with a Gmail account which sends the office hours invitations on behalf of the system. The actual email workflow functionality is only being tested in the context of system acceptance testing.

#### Document References:

Requirements specification: Can be found in the “Requirements” folder, titled “VisionScopeBaseline.docx”

Only newly created and newly updated pieces of the software are to be unit tested, as mentioned, but other aspects with be tested during system acceptance testing.

4 Features to be tested

#### Functionality to be Unit Tested:

* Login Manager (LoginManager.cs)
  + Create New User
  + Check Password
* Email Sender (EmailSender.cs)

#### Functionality tested (additionally) via System Acceptance testing:

* Welcome Page (extensively)
* Registration Page (extensively)
* Calendar Page (minimally)
* Unit test components in system context (seen above)

5 Features not to be tested

During system acceptance testing, many aspects of the system will be touched. For unit testing, however, multiple components of the system will not be tested.

#### Items not to be tested:

* Cafe Data Interface
  + Responsible for the communication between the system and the Cafe database which contains the data on professors and office hours. The cafe data interface is used by the login manager, though, and is therefore, in some ways, tested.
* Calendar
  + The calendar page, the main user page of the system, is being tested extensively in system acceptance testing, but it is not being unit tested at this time. For example, deep-dives into selection functionality are not completed. This means generation of the correct dates and times for appointments will not be tested.
* Appointment Creation
  + The functionality of actually creating, storing, and displaying appointments is not tested
* Delete User
  + This function is within Login Manager but it has not been tested or implemented. It is not an essential item at this time.

6 Approach

Unit testing will be adopted to test specific methods in the newly created and updated classes. This will include testing of creating a new user and authenticated login functions within the LoginManager, and creation of an email and ICS file upon submission of an appointment request.

## Unit Testing

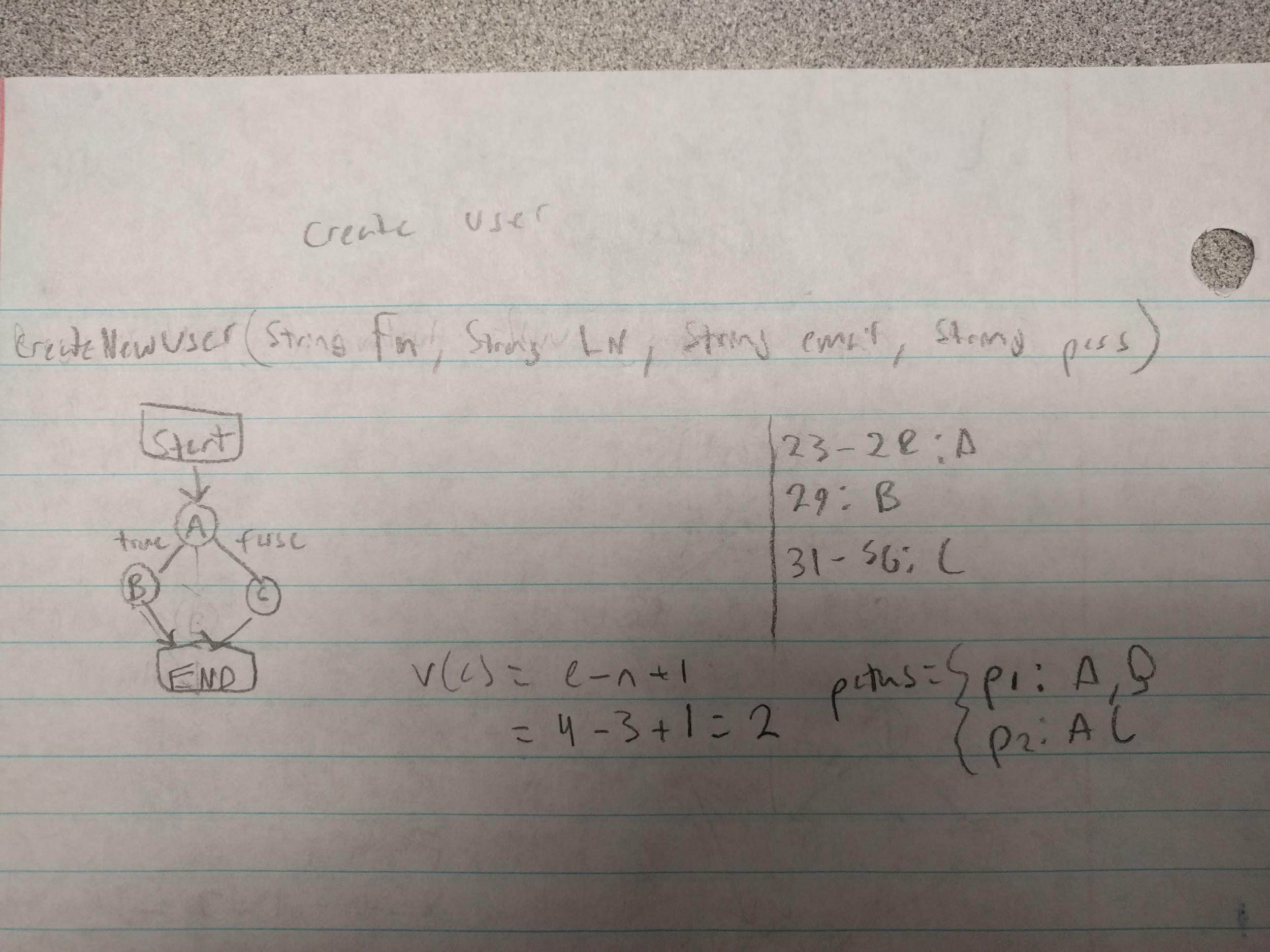
### Login Manager

This unit is one class with three (3) functions. The two (2) that will be tested are: “createUser” and “checkPassword”. Both functions are important to the functioning of the system. The following approach will be taken to ensure test adequacy: Basis Path Testing, Condition and Decision, and Black Box Boundary Value Analysis, Adhoc testing.

These test cases will ensure that every decision and condition is tested, and thus every statement will be tested. Further, the black box style tests boundary values and special cases that are possibilities for the system to encounter, using the requirements document, usability guidelines, and security guidelines as success criteria.

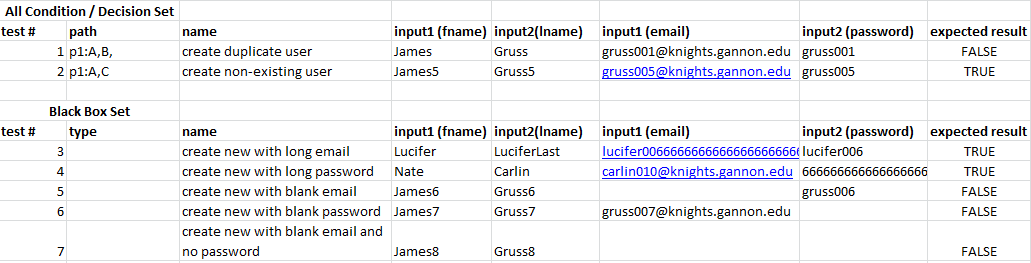
##### Create New User

Create New User Control Flow Diagram:



*Figure 6.1*

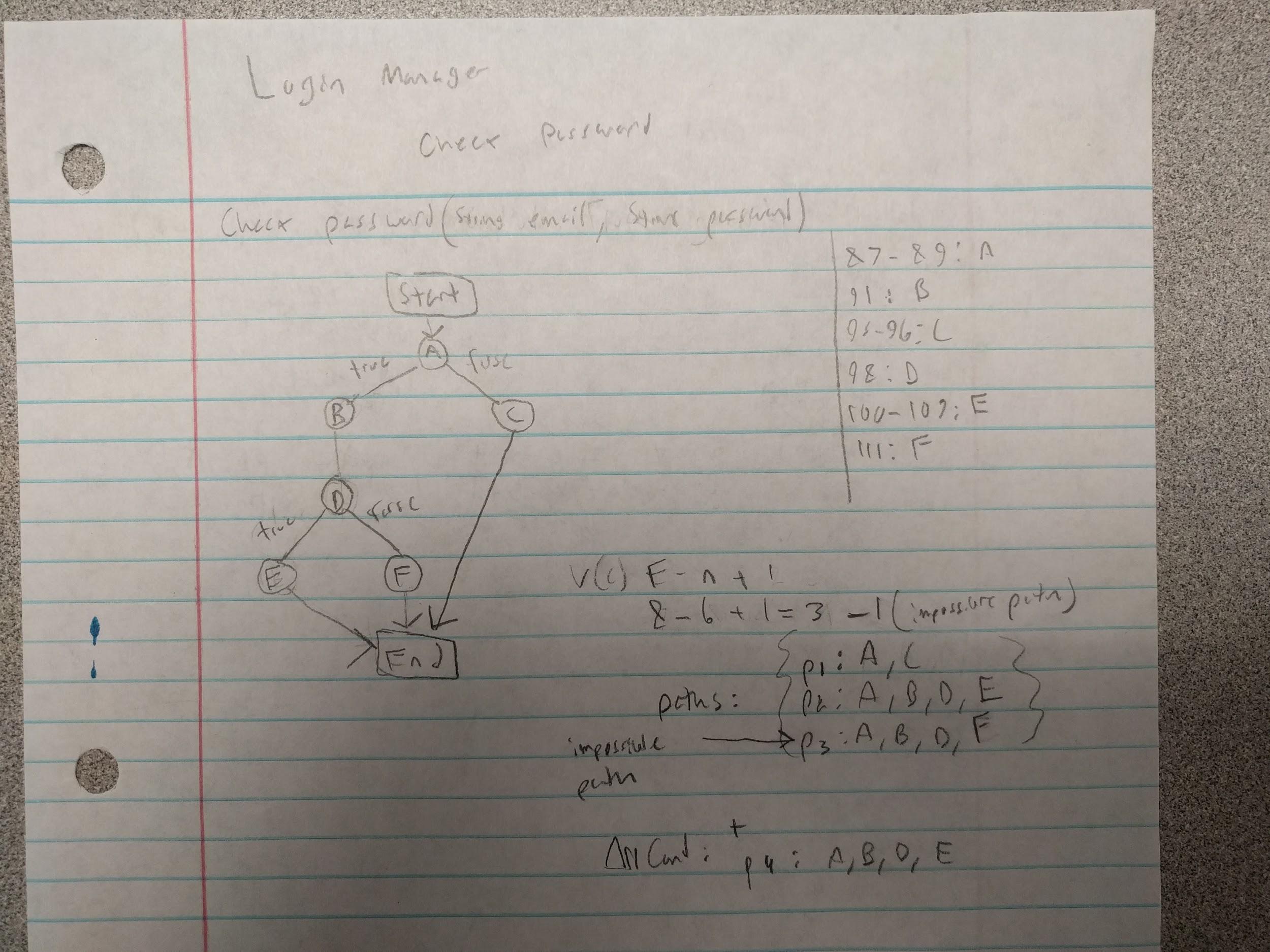
Create New User Test Cases:



*Figure 6.2*

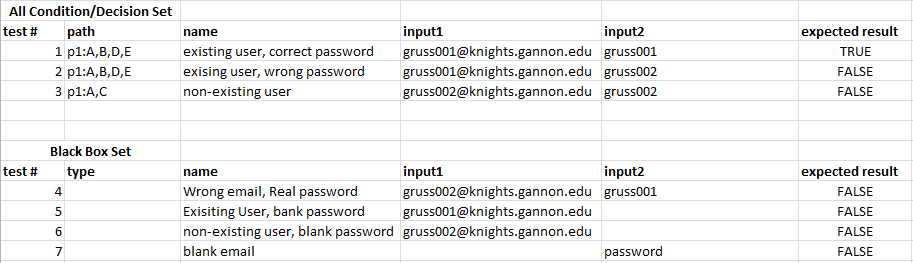
##### Check Password

Check Password Control Flow Graph



*Figure 6.3*

Check Password



*Figure 6.4*

##### Tools

Visual Studio Unit Testing utilities will be used for creating and running unit tests for Login Manager

### 

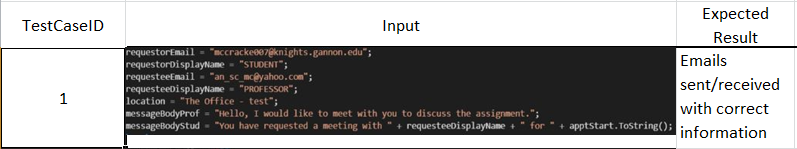
### 

### Email Sender

A standalone C# program will be created to unit test the functions of EmailSender. The functions being tested are sendEmailInvite(), setICSEmailValues(), and SendEmailWithIcsAttachment().

There are seven variables being passed to the email sender. Each of the arguments being passed to the email sender vary in type and accepted format. It will be imperative to create test cases that cover all of the possibilities of values that they could potentially hold because they are what create the email and ICS invitation. The system is expected to handle all of these possible values and have an appropriate response/action.

Below is an example test case for the EmailSender:



*Figure 6.5*

### System Acceptance Testing

System acceptance testing will be adopted to test the registration, login, and create appointment features. These three main features comprise the majority and the system functionality reflect the system requirements. This will test usability as well as checking that the underlying functionality meets the requirements when it is brought together. The estimated time to complete testing is two (2) hours for each of these features. Tests will be created to check adherence to the requirements specification and commonly followed usability practices.

Three methods are to be used to test the system:

1. Selenium - to record automated tests for system acceptance testing (Firefox Add-on)
2. Manual Testing

Selenium test case suites used for these tests can be opened and run in Firefox. They can be found in the functions’ respective folders.

#### Registration: System Test Case - Requirements Mapping:

|  |  |
| --- | --- |
| Requirement (Found in Vision and Scope doc) | Test Cases that test the requirement |
| **BR-1.1**: Due to Unavailable LDAP system, Students must create an account with an email and password before using the system. They must use their Gannon email. | Test cases: 1, 2, 3, 4, 5 |
| System interaction and boundary analysis | Test cases: 6, 7, 8, 9, 10, 11 |

#### Make Appointment: System Test Case - Requirements Mapping:

|  |  |
| --- | --- |
| Requirement (Found in Vision and Scope doc) | Test Cases that test the requirement |
| Allow students to select department | Test case 1 |
| Allow students to select professor | Test case 2 |
| Make appointment with any professor in CAFE | Test case 3 |
| Message body empty (generic will be created) | Test case 4 |
| Message body long | Test case 5 |
| Message body custom | Test case 6 |

7 Item pass/fail criteria

The criteria to be used to determine if a test case is passing or failing relies on the requirements specification and commonly followed usability and security practices that we think apply to our system.

9 Test deliverables

#### Test Deliverables

* Test plan: This document is the main test plan for this project at this time
* Test design specifications:
  + Please refer to figures 6.1, 6.2, 6.3, and 6.4
* Test case specifications
  + Please refer to documents on “/test\_cases/stqa\_NEW/

#### Test Tools:

* code/OfficeHours/OfficeHoursTest/... - Visual Studio Unit Testing
  + This project contains the Unit Testing for the Login Manager
* test\_cases/sendApptEmail/... - Visual Studio Program
  + This project holds the tester program for unit testing of the Send Email functions

10 Risks and contingencies

A risk to our test plan is the possibility that our Gannon ITS server goes down. At this point, our code base will still be existing, but our database

delivery of test items might require increased night shift scheduling to meet the delivery date).