**1. Testing Report**

**1.1 Test Coverage**

**1.1.1. Tested Items**

**Tested Units**

With regards to item testing, all requirements were grouped into appropriate units/classes based on similarity and interdependence of function (E.g.: Login and Logout). Each requirement is, in actuality, a use case previously outlined in Deliverable 1 and improved upon in Deliverable 2. In addition, a unit was created consisting of a new feature/use case: “Create Student Account”. This unit consists of the use case “Create Student Account”, which was not included in previous deliverables due to its relative newness. More information on this use case can be seen in section 1.3.

All requirements below are considered to be part of the core functionality of The Force, and thus are given a higher priority in comparison to something such as admin access and features (refer to 1.1.2 for further information). All units, and the core features/requirements they represent, can be seen in the table below. The test cases associated with each unit and requirement is numbered, and can be seen in further detail under section 1.2.2.

|  |  |  |
| --- | --- | --- |
| Unit/Classes | Related Requirements | Test Case(s) |
| Login | UC01 | 1.1 – 1.2 |
| Logout | UC02 | 2.1 – 2.2 |
| Create Student Account | UC16 | 16.1 – 16.8 |
| Edit Student Information | UC03 | 3.1 – 3.3 |
| Edit Courses | UC05, UC06, UC07 | 5.1 – 5.8, 6.1 – 6.4, 7.1 – 7.3 |
| Schedule Generation | UC08, UC11 | 8.1 – 8.3, 11.1 |

Login

The login procedure is incredibly important with regards to The Force as a whole. In order to access the scheduler, student information, or administrative access, a proper email and password combination is required. The login will only be successful if the combination of this email and password previously existed in the database (associated to a single user). Should this combination not exist, access to any further functions would be denied and the email and password fields would be cleared. Login is a unit that would be used several times, incredibly often, by every user of The Force. As such it requires extensive testing, as it contains the requirement that allows all other features to be accessed effectively.

Logout

Logging out is needed in order to allow a student or administrator to exit their account once they’ve completed their actions. Logging out ensures safety and security for the user’s account should they be sharing the device they used The Force on with others. Without logging in and logging out, all functionalities provided by The Force become irrelevant, thus it is a very high priority. In addition, all users will heavily use the logout feature, just as they use the login feature. As such, the logout unit must be tested in order to ensure the safety of all users, otherwise it runs the risk of massive security breaches across multiple users.

Create Student Account

In the case that a student wishes to create a schedule, it is required they have an account in place. For this reason, the creation of student accounts is another incredibly important requirement for The Force. Anytime a student without an account already in place would like to use The Force, they would need to make an account. This unit will be tested by following the instructions provided upon requesting to create an account, and then confirming whether or not a new entry was created in the databases with all the appropriate information. In the case that an account is made properly, the student will be redirected to the main page with their current schedule and courses.

Edit Student Information

Based on the student currently using The Force, the appropriate account information will be displayed on the Accounts page. From this page, the student will be given the option to change their password should they like. This will be tested through a mock-account, and the new password can be checked through accessing the database after the change, or simply attempting to re-enter the application after logout using the new password. This feature is more so a novelty for the student/administrator using it, but it is important due to the comfort it can provide a user as well as the added security provided thanks to the option to change one’s password.

Edit Courses

Course addition is an incredibly vital part of the entire schedule generation process. Without the option to manually add courses, students would be left without any true control over the courses they take. Every student using The Force needs to add courses, and thus it is considered extremely important with regards to the application. Courses added would need to ensure no conflicting time slots, as well as ensuring that the course is not allowed should the necessary co- and prerequisites not be present. In addition to choosing a course, students are expected to choose an appropriate section as well, which will also be tested for time slot conflicts before being added to their schedule. Should these two options misbehave, students will be unable to add courses to their schedule, nor edit the sections they are in, thus invalidating the entire schedule generation process, and the application as a result.

Deleting a course is of an equal importance. Should a student wish to remove a course they had entered in their schedule during the registration period (possibly due to a new, more important course conflicting with the time), or during the semester for some other reason, it is imperative that the option function perfectly. This option will once again be used by all students, and can be tested by simply removing a course from a schedule and ensuring that the necessary updates take place

Schedule Generation

Schedule generation is, in fact, the primary purpose of The Force, and as such may very well be the most important unit present (aside from login/logout). Once courses have been added or removed according to the student’s wishes, their schedule will be generated and saved. This can be tested by simply observing whether the schedule is still present for a given semester upon refresh of the main page. Seeing as it is the entire purpose of the application, extra precautions must be taken.

Assuming no courses were manually added/removed, or if a student doesn’t care for the courses they have already added, a schedule can also be auto generated based on the courses said student needs to take next in their sequence. Though this may not result in a student’s ideal schedule, the option to edit sections and remove courses will still be present after the generated schedule has been accepted. Auto generation of a schedule is far more complicated than manual generation, as a student will normally know what courses they need to take next and can physically see and avoid time conflicts. As this is not the case for auto generation, extra precautions are placed and need to be tested (e.g.: Overlapping timeslots, registering for previously completed courses, registering for courses without the pre- or co-requisites required). This can be tested through observation of the generated schedules, and addressing any issues that may arise. Both methods of schedule generation, should they not be implemented properly, could jeopardize the entire purpose of the application, and thus are of the utmost importance.

Once a schedule has been generated, a student may give his or her approval and save it to the semester they are currently creating it for. Once saved, the schedule, as well as the courses they are currently enrolled in, will be visible upon login and on the main student page. All students will have access to this saved schedule, and without properly implementing the save feature, a student would never be able to store their schedule for later use (effectively removing the need for one). Though the program would function without this feature, it would serve no real purpose, as students need to be able to see the classes and sections they are currently, and will be, enrolled in.

**Tested Requirements**

The requirements displayed above were tested for one of 2 reasons:

1. They are considered integral to the development of web application project as a whole, and thus were tested in order to gain an understanding of the progress being made as a whole (e.g.: Login, add course).
2. They were lesser features, which were ready for testing (e.g.: View / Edit Profile).

Any explanation for the necessity of testing of these requirements can be found below, as well as an indication of their level of importance in comparison to the other requirements being tested

* Login (Use Case 1)
  + This requirement is considered to be of a **high importance** level, as should the login feature fail, further access to the application would be restricted. Not only would further testing be impossible, but any use of the program would be as well.
* Logout (Use Case 2)
  + This requirement is considered to be of a **high importance** level, as without the option to logout, other unique users would not be able to utilize the application from a shared device. In addition, overall security for each user would diminish due to the fact that their accounts would, in a sense, always be available to anyone who managed to get the device said individual used.
* View / Edit Information (Use Case 3)
  + This requirement is considered to be of a **high** **importance** level, as the ability to view ones account information can be important (e.g.: Seeing which courses have already been taken), as well as editing something such as a password in the case of security risks. In addition, it is important to be able to view any errors in a student’s personal or academic information, thus allowing the proper help to be notified.
* Add Course (Use Case 5)
  + This requirement is considered to be of a **high importance**, as it serves as the backbone for manual schedule generation. Without the ability to manually add a course to ones schedule, students would be forced into only auto-generating their schedule without the ability to then add one which they would prefer (as manual generation simply would not exist).
* Change Course Section (Use Case 6)
  + This requirement is considered to be of a **high importance** level, as without it a student would be restricted to the section either provided to them by the auto-generator, or which they themselves chose while initially creating their schedule. Should a situation arise where a student wanted to switch their section, perhaps into a better-timed one, or in order to avoid a conflict with another course, this feature is required.
* Remove Course (Use Case 7)
  + This requirement is considered to be of a **high importance** level, as without it students would be unable to remove course automatically assigned to them, or which they lost interest in. This feature is required when editing one’s schedule, as without it new courses could never be added (assuming the student has reached a “maximum” number of courses, or they’ve decided to no longer take a particular course).
* Generate Schedule (Use Case 8)
  + This requirement is considered to be of a **high importance** level, as it represents the entirety of the auto-generated schedule feature. Without this feature, students would be restricted to only creating schedules manually, even though they may be indifferent to the outcome of their schedule; they may simply wish to follow their sequence (and perhaps make changes after auto-generation takes place).
* View Saved Schedule (Use Case 11)
  + This requirement is considered to be of a **medium importance** level, as it doesn’t represent a core functionality of the application, nor does it provide additional user security. Even so, the entire purpose of creating a schedule, aside from enrolling in classes, is to be given the option to refer to said schedule at a later date (perhaps to check course times, or locations). Thus though it is not a main pillar of The Force, it is relevant nonetheless.
* Request New Account (Use Case 16)
  + This requirement is considered to be of a **high importance** level, as every student is required to make an account before access to The Force’s features are provided. This is equally as important as the ability to login, as it serves as the precursor to it. Without the ability to create an account, a student would never be able to eventually login and begin creating their schedule.

**1.1.2. Untested Items of Interest**

Two main units were not tested, but remain relevant nonetheless: Admin Access/Features, and the ability to export a saved schedule. Both units were left untested due to time constraints leading to neither feature being implemented yet, however both are important with regards to the end goal of the application: Proving an easy to use experience, and a distinction between different types of users and their unique features.

Admin Access / Features

Admins would be a separate type of user from a student: An admin would be given the ability to not only modify a section’s capacity, but also view and edit any student’s current course list. Upon selecting a given course from their administrative list, admins would be given the option to change said course’s (as well its sections’) size capacity. This could be tested through trial (i.e.: Black-box testing) and by inputting various values in the section’s capacity is observing not only whether a change takes place, but also whether the value entered is a legal one (i.e.: No negative values). This is important to test to due to fact that courses and sections will natural change capacity from time to time, thus having the ability to manually do so is imperative.

Additionally, admins are given the ability to change any student’s current schedule. This is required as an administrator must be able to move students between courses and sections should the student come to them for help, or if they’re required to move certain individual due to classroom size limitations. This is done by adding a given student to a queue; a queue that may only contain one individual at a time. Once added, the administrator will be given the ability to view said students schedule, course list, and account information. From there, the process of adding and removing courses, as well as changing sections, is identical to the process used by the student. Once the necessary changes have been made, the student will be removed from the queue, allowing the admin to modify a different student. This can be tested, similarly to the previous step, with black box testing. By providing certain inputs to the administrative features, the results can be observed and the necessary modifications can be made. All administrative features are important due to the differentiation it provides between user types, and the unique role it plays in comparison to a normal student. Should they not be tested properly, students may be given administrative access, or administrators may be given none at all, thus ruining the differentiation created.

Export Saved Schedule

This feature is relevant due to the convenience it provides the students using The Force. By allowing their schedules to be exported, students are given the option to easily access their schedule without needing to go through the application (as well as print said schedule, providing them with a physical copy). As it is another requirement, this too would be tested through a black box, and should a properly exported file/image be created on request, the feature will be deemed successful. It is important to test due to the fact that, should it fail to work properly, user satisfaction would decrease as a result, and ease of access for the student’s schedule would be reduced. As the entire purpose of The Force is to create student schedules (primarily), it stands that once created, these schedules should be made easily accessible to those who made them.

**1.2. Test Cases**

**1.2.1. Unit Testing**

**1.2.2. Requirements Testing**

The test cases for all tested requirements presented in section 1.1.1. can be found below. These requirements are based upon the use cases highlighted in deliverable 1 and 2. Each requirement includes numbered test cases, a description of each test case, an example of possible input which could be provided (as not every situation can be addressed for most requirements, thus partition testing is applied and test values are used to represent larger groups of examples) for said test case, the expected output, and whether each output was actually achieved. It is important to note that all test cases with failures are considered to be top priorities, and thus are primary focuses with regards to the end product of the application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC01 | Login |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 1.1 | User inputs valid email and password combination | Email = “test@case.com”  Password = “password” | User is redirected to main page | PASS |
| 1.2 | User inputs an invalid email and password combination | Email = “test@case.com”  Password = “password1” | Empty email and password fields. User is not redirected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC02 | Logout |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 2.1 | User is not logged in and attempts to log out | User attempts to press Logout button | User is on login page, and thus no logout feature is provided | PASS |
| 2.2 | User is logged in and attempts to logout | User presses Logout button in navigation bar | User is redirected to login page, without access to any other feature | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC03 | View / Edit Information |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 3.1 | Student wishes to view their account | “Account” button is pressed | User is redirected from schedule/main page to Account page | PASS |
| 3.2 | User enters a new password and confirms their request | newPass = “password1”  confirm = “password1” | Message appears confirming user’s actions | FAIL |
| 3.3 | Administrator views a student’s profile | Student number is provided.  Example: “12345678” | Administrator is redirected to student’s account page | FAIL |

**Note that when adding a course (UC05), the case of have “co-requisites” completed but not prerequisites does not exist, as all courses in software engineering that require a prerequisite also allows the course to be completed concurrently. As such, this situation is not relevant.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC05 | Add Course |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 5.1 | Student adds a course they already have in their schedule | Student selects “COMP 232”, while COMP 232 is already in their schedule | No changes should occur | PASS |
| 5.2 | User adds a course they do not have the prerequisites, nor co-requisites, for | Student adds “SOEN 287” before completing “COMP 248” | Inform user of error (lack of pre/co-requisite) and do not add course to schedule | FAIL |
| 5.3 | User adds a course they have completed the necessary prerequisites for, but not co-requisites | Student has not yet added “SOEN 343”, but adds “SOEN 345” to their schedule and generates | Inform user of error (lack of co-requisite) and do not add course to schedule | FAIL |
| 5.4 | User adds a course they have previously completed (no need to retake the course) | Previously completed “COMP 248”, add to schedule once again | Inform user of error (course completed) and do not add to schedule | FAIL |
| 5.5 | User adds a course which conflicts with a course already in the schedule | “COMP 248” times overlap with “SOEN 228”, but the student adds “COMP 248” to their schedule anyways | Inform user of error (time conflict between courses) and do not add course to schedule | FAIL |
| 5.6 | User adds a course which is currently full (either a course or section) | “COMP 248” is already full, but is added to the schedule anyways | Inform user of error (course/section is full) and do not add course to schedule | FAIL |
| 5.7 | User adds a course with all the necessary prerequisites completed, no time conflicts, and space in the class remaining | “COMP 248” is added to the schedule, and all conditions are satisfied | Course and section is added to the student’s current schedule | PASS |
| 5.8 | Student adds a course after indicating preferred personal constraints (days off, between two times, etc…), assuming no other conflicts | Student selects “COMP 248” after indicating they would like no courses between 8:00 AM and 10:00 AM | Course and section would be added to the student’s current schedule | FAIL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC06 | Change Course Section |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 6.1 | Student selects a course or section from within their schedule | Student clicks on “COMP 248” from within their schedule | Popup window appears containing alternate sections and their times | FAIL |
| 6.2 | Student selects a new section from the popup, but this conflicts with another course in the schedule | Student selects section “AA” for “COMP 248”, but this conflicts with “COMP 232” | Inform user of error (time conflict) and section change does not occur | FAIL |
| 6.3 | Student selects a new section from popup, but this section is full | Student selects section “AA” for “COMP 248”, but section “AA” is full | Inform user of error (section size full) and section change does not occur | FAIL |
| 6.4 | Student selects a new section from popup, and no issues are associated with this section | Student selects section “AA” for “COMP 248” | Section change occurs. Both course list and the student’s schedule are updated | FAIL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC07 | Remove Course |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 7.1 | Student tries to removr course when no courses are saved | Student presses “Remove” button | No “Remove Course” button is present | PASS |
| 7.2 | Student chooses to drop a course, and confirms their decision | Student finds the course they wish to remove in their schedule, clicks it, and selects “Remove Course” | The course is removed from the course list, as well as from the saved schedule | FAIL |
| 7.3 | Student chooses to drop a course, and cancels their decision | Student finds the course they wish to remove in their schedule, clicks it, and cancels their decision | The course will remain in the course list and schedule | FAIL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC08 | Generate Schedule |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 8.1 | Student auto generates a schedule for the current semester | Student presses “Auto Generate Schedule” button | A popup will appear containing an auto generate schedule, based on the next courses in said student’s sequence | FAIL |
| 8.2 | Student dismisses the option created automatically | “Cancel” button is pressed on popup window | Schedule will not be saved and user will remain on main page | FAIL |
| 8.3 | Student confirms the schedule displayed | “Accept” button is pressed on popup window | Schedule will be saved and will now be featured on the main page, next to a course list | FAIL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC11 | View Saved Schedule |  |  |  |
| Test Case | **Description** | **Input provided** | **Expected Output** | **Result** |
| 11.1 | Student views their schedule | Student signs in, makes an account, or presses “Schedule” from the “Accounts” page. | Student is redirected to the main / schedule page and their current saved schedule is displayed. | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UC16 | Request New Account |  |  |  |
| Test Case | **Description** | **Input(s) provided** | **Expected Output** | **Result** |
| 16.1 | Create Account option is chosen | User presses “Create Account” button | Popup page with all required fields appears | PASS |
| 16.2 | User leaves any of the 4 fields blank (First name, Last name, Email, Password) | Any blank field will result in the same expected output.  First = “Test”  Last = “Case”  Email = “”  Password = “password” | Popup window remains open. Field with error is highlighted in red in order to notify user. | PASS |
| 16.3 | Email does not follow the format “xxx@yyy.zzz” | 1. @test.com 2. testcase.com 3. test@case | Popup window remains open. Email field is highlighted in red. Message appears below field notifying user of invalid entry | FAIL |
| 16.4 | Email follows the format “xxx@yyy.zzz” | Email = “test@case.com” | No error message displayed. Assuming other fields are valid on submission, registration is completed | PASS |
| 16.5 | Password is less than 6 characters | Password = “passw” | Popup window remains open. Password field is highlighted in red. Message appears below field notifying user of invalid entry | PASS |
| 16.6 | Password length is equal to or greater than 6 characters | Password = “password” | No error message displayed. Assuming other fields are valid on submission, registration is completed | PASS |
| 16.7 | Previously completed courses are marked (checkboxes) | Checkbox indicating “COMP232” is checked | Assuming successful registration, marked courses can be seen in “Accounts” page | FAIL |
| 16.8 | All fields are filled with valid entries | First = “test”  Last = “case”  Email = “test@case.com”  Password = “password” | Upon submission, user is redirected to main schedule page. | PASS |

**1.2.3. Stress Testing**

**1.2.4. Security Testing**

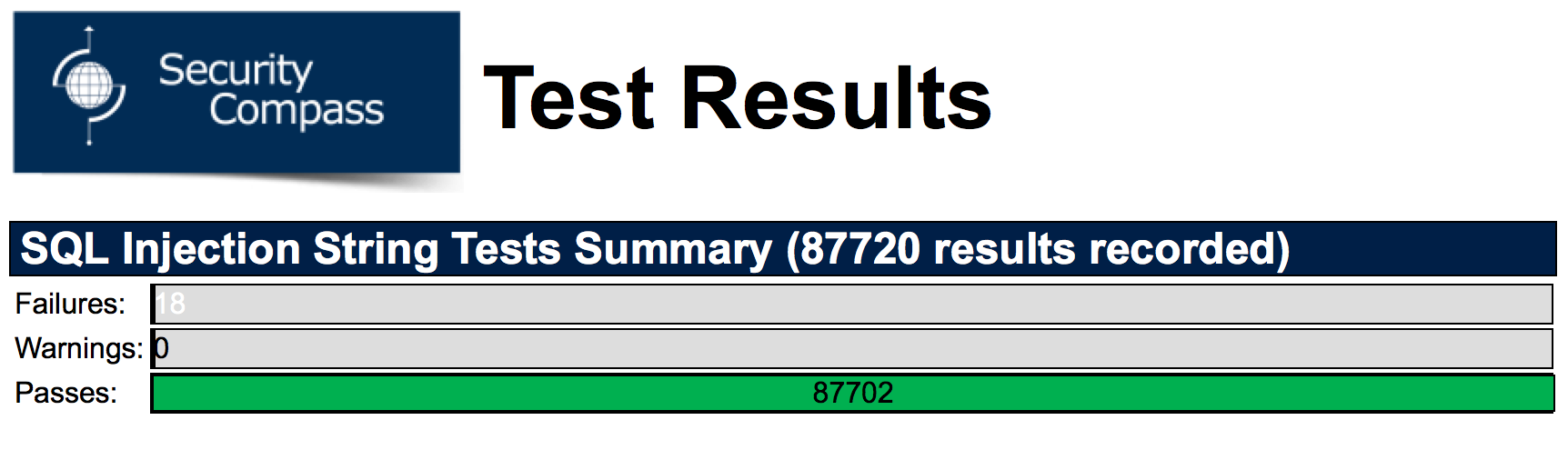
**SQL Injection**

SQL injection is a technique by which malicious SQL statements are inserted into an entry field in order to trick the system into giving more information than would be normally available. By exploiting wording used in SQL statements, security vulnerabilities can be exploited and incredible damage can be done to an application, thus it is imperative that this threat be tested against.

In order to test for these vulnerabilities, a FireFox Extenstion called SQL Inject-Me was used. By submitting HTML forms with substituted values (i.e.: Strings representing what would be used during an SQL Injection attack), the tool allows for the inspection of said vulnerabilities. Once escape strings have been sent to the database through the form fields, the extension then looks for database error messages that have been outputted into the rendered HTML of the page. Despite all this, the tool does not compromise the security of the application, as it simply looks for entry points to attack through (no actual attacks are done).

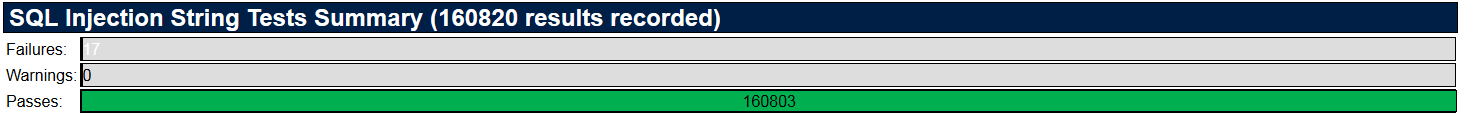
The SQL Injection tool was run on 3 of The Force’s pages: It’s login page (including both logging in and creating an account), the main / schedule page, and the user account page. Below are the results. As the results (when errors were found) and number of form fields can be quite extensive at times, only the overall results are shown. To mitigate this, explanations for failures are provided when they occur.

1. Login Page

****

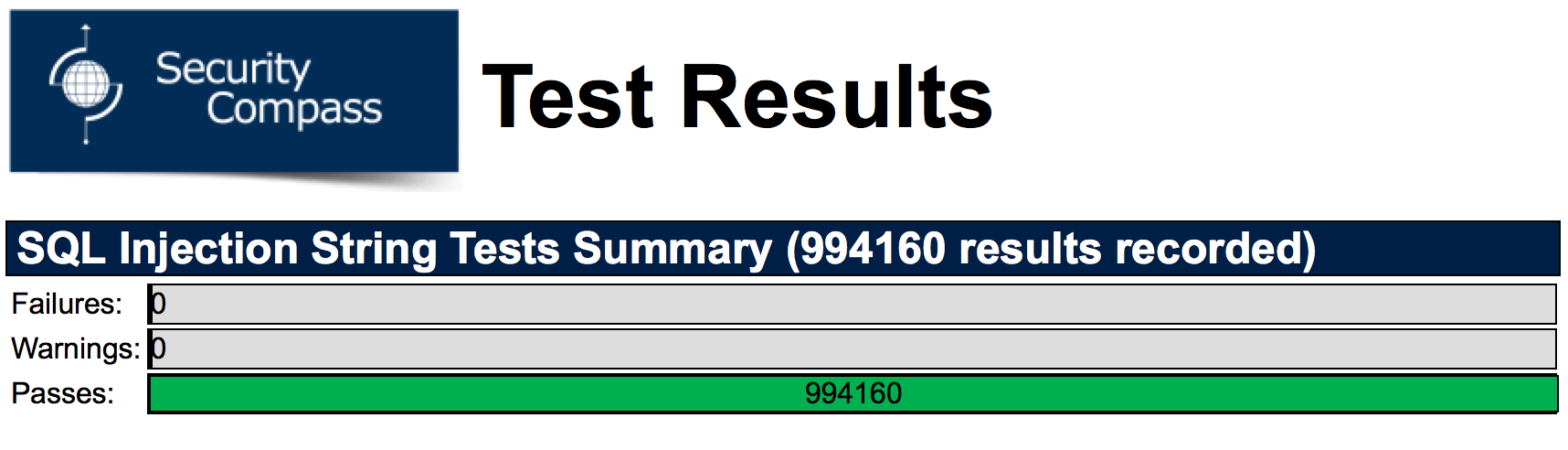
The only (18) failures experienced on the login page were associated with the email and password fields, thus extra precaution will be taken with respect to the final project to ensure that security is thoroughly provided for these two fields.

1. Main Page



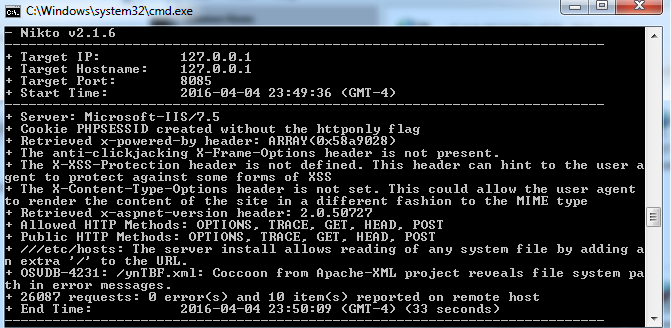
On the main page, the extension detected 17 failures. However, these failed test cases were all encountered on the same field: The second “unnamed field” located at the bottom of the main page (at the time of this testing). This extra field is not relevant to the main functionality of the scheduler, and will be fixed to avoid security vulnerabilities once all revisions have been completed. Removing this field is expected to remove the failed cases with it.

1. Account Page



No failures or warnings were detected on the accounts page, thus it can safely be considered secure from SQL Injection attacks.

**Nikto**

Nikto is an open source web server scanner, used to perform comprehensive tests against web servers from multiple items (e.g.: potentially dangerous files/programs, outdated server versions, and version specific server issues). As web server security is an ever-increasing issue, namely since Web Server vulnerabilities can be easily taken advantage of, it is imperative that the server used for The Force be tested for security. The Force is hosted on an IIS (Internet Information Services), which is a Microsoft-made extensible web server, and the results of running Nikto on our server can be found below.

The security tool failed to find any particularly dangerous errors, and thus the server security can be considered to be quite high.

**NMap**

NMap (or Network Mapper) is a security scanner used to discover hosts and services on a computer network. In addition, NMap can be used as a port scanner, to determine whether network ports are open, closed, filtered (i.e.: NMap can’t determine if the port is opened, making it frustrating for attackers), etc.… The main result of this scan can be seen below:



As can be seen above, several ports are currently open. While this makes accessibility higher (i.e.: For the team), open ports are simply paths to be attacked through. Since possible attackers wish to exploit open ports, it would be ideal to close (or at the very least filter) these ports once work on the project has completed. In doing so, the legitimate users (the team) will not be reprimanded, but possible attackers will no longer be able to utilize these open ports to target the network (as ports present an endpoint of communication in an operating system).

**1.3. New Use Case**

|  |  |
| --- | --- |
| ID: | UC16 |
| Name: | Request new Account |
| Importance (/5): | 5/5 |
| Difficulty (/5): | 3/5 |
| Risk Assessment: | Low |
| Actors: | Student |
| Goals: | * To allow students the ability to create a new account for the system |
| Summary: | By using this feature, the student will be able to create a new account by providing their first name, last name, password and email to the system. |
| Preconditions: | User does not have an existing account, and is on the login page. |
| Basic Flow: | 1. User selects “Create Account” 2. System prompts user to enter profile information 3. User enters profile information 4. System verifies user email and password combination to be valid 5. System redirects user to main page of website |
| Post Conditions: | Success: System redirects User to home page, account information is saved  Failure: User remains on current page, system displays error |