Automated Check-In and Scheduling System for a Web-Based Testing Application

Ashwin V. Kumar
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/gradreports

Part of the Computer Sciences Commons

Recommended Citation
Kumar, Ashwin V., "Automated Check-In and Scheduling System for a Web-Based Testing Application" (2013). All Graduate Plan B and other Reports. 288.
https://digitalcommons.usu.edu/gradreports/288
AUTOMATED CHECK-IN AND SCHEDULING SYSTEM FOR A WEB-BASED TESTING APPLICATION

by

Ashwin V. Kumar

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Donald Cooley
Major Professor

Dr. Nicholas Flann
Committee Member

Dr. Daniel Watson
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah
June, 2013
Abstract

Automated Check-In and Scheduling System for a Web-Based Testing Application

by

Ashwin V. Kumar, Master of Science
Utah State University, June, 2013

Major Professor: Dr. Donald Cooley
Department: Computer Science

Ideally, a testing center and associated software should be an effective tool for learning assessment. It should serve as a link between faculty members who create tests and their students who are assessed by those tests.

Testing centers are generally limited in the number of computer systems available as compared to the number of students taking tests. Automated testing plays an important role in such situations. Automated testing outside the classroom offers students the flexibility of choosing a preferred time for taking tests. A completely automated system also contributes in reducing a portion of the workload by automatically grading some or all of the exams.

This report presents a feature called “Automated Check-In and Scheduling”. This feature uses magnetic card readers to provide an interface between a student and the testing system. This report also discusses a card swiper interface which is linked to a new scheduling system. Together, they provide for better utilization of a laboratory’s resources.

(51 pages)
Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>Acronyms</td>
<td>vii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Testing Centers</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Automated Testing</td>
<td>2</td>
</tr>
<tr>
<td>2 Problem Analysis</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Overview</td>
<td>4</td>
</tr>
<tr>
<td>2.2 User Roles</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1 Student</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2 Proctor</td>
<td>6</td>
</tr>
<tr>
<td>2.2.3 Instructor</td>
<td>6</td>
</tr>
<tr>
<td>2.2.4 Administrator</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Existing Scheduling Model</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Motivation</td>
<td>8</td>
</tr>
<tr>
<td>2.5 New Scheduling Model and Interface</td>
<td>10</td>
</tr>
<tr>
<td>2.5.1 The Card Swiper Interface</td>
<td>11</td>
</tr>
<tr>
<td>2.5.2 The New Scheduling Model</td>
<td>11</td>
</tr>
<tr>
<td>3 Design Analysis</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Overview</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Software Analysis</td>
<td>15</td>
</tr>
<tr>
<td>3.2.1 User Roles</td>
<td>16</td>
</tr>
<tr>
<td>3.2.2 The Card Swiper Interface</td>
<td>18</td>
</tr>
<tr>
<td>3.2.3 Automated Check-in and Scheduling System</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Hardware Analysis</td>
<td>25</td>
</tr>
<tr>
<td>3.3.1 The Magnetic Card Reader</td>
<td>25</td>
</tr>
<tr>
<td>4 Implementation</td>
<td>28</td>
</tr>
<tr>
<td>4.1 The Card Swiper Interface</td>
<td>28</td>
</tr>
<tr>
<td>4.2 Automated Check-in and Scheduling System</td>
<td>33</td>
</tr>
<tr>
<td>5 Testing</td>
<td>39</td>
</tr>
<tr>
<td>6 Future Work</td>
<td>42</td>
</tr>
<tr>
<td>7 Conclusion</td>
<td>43</td>
</tr>
</tbody>
</table>
References

44
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Instructor view of iNetTest’s test scheduling system.</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Portion of student’s page indicating him/her to make a reservation.</td>
<td>4</td>
</tr>
<tr>
<td>2.3</td>
<td>List of time slots opened for a student to make a reservation.</td>
<td>5</td>
</tr>
<tr>
<td>2.4</td>
<td>Test is unlocked when a student’s reservation time starts.</td>
<td>5</td>
</tr>
<tr>
<td>2.5</td>
<td>Proctor unlock.</td>
<td>9</td>
</tr>
<tr>
<td>2.6</td>
<td>Proctors interface to commit check-ins.</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>The Magnetic Card reader.</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>The Magnetic Card reader with A-Card being swiped.</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>The card swiper interface displaying only the student input section.</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>The card swiper interface displaying the lab-information section.</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>The card swiper interface displaying the check-in section.</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Interface indicating a successful check-in.</td>
<td>33</td>
</tr>
<tr>
<td>4.5</td>
<td>Interface indicating failure for check-in as no seats are currently vacant.</td>
<td>34</td>
</tr>
<tr>
<td>4.6</td>
<td>Simple design illustration of the automated check-in and scheduling system.</td>
<td>35</td>
</tr>
<tr>
<td>4.7</td>
<td>Workflow of Action 1 to make seats available and predict timestamps.</td>
<td>36</td>
</tr>
<tr>
<td>4.8</td>
<td>Workflow of Action 2 to process on-the-spot reservations and commit check-ins.</td>
<td>37</td>
</tr>
</tbody>
</table>
Acronyms

TST    Test Start Time
TET    Test End Time
FWT    Finished With Test
Chapter 1

Introduction.

1.1 Testing Centers.

Testing centers are controlled environments that house a finite number of computer systems. These computer systems are authorized by a server to administer tests. Students can come in during the working hours of the testing center and take the allotted test(s) for a particular course. Thus, testing centers resemble a dedicated environment provided within an institution for test-taking purposes. Associated with testing centers are personnel called “proctors” that aid in authenticating and authorizing a student to take a test. The proctor has the permission to unlock a requested test, only if a seat is available.

Over time, the Internet has become a large repository of data related to almost any type of question. In order to provide anti-cheating mechanisms, most computer systems in a testing center have limited access to the Internet and certain software tools. This helps in maintaining a healthy and honest testing environment while dissuading students from cheating. Furthermore, testing centers may also have restrictions on the material brought in by a student, and even include security cameras for monitoring of students taking a test. Proctors may also enforce some level of cheating prevention by seating students randomly.

One of the biggest benefits of using testing centers to administer tests is that it provides flexibility to a student in choosing a test time. With a testing center, students are not necessarily constrained to take a test on a specific day and time. Tests administered at a testing center are usually kept active or open for a number of days. Depending on each student’s unique schedule, he/she may opt for a time slot as per his/her convenience. With computer-based tests, the need for paper is completely eliminated. The student submits only an electronic copy of an attempt which is stored in the testing software’s database. At some institutions’ testing centers, specific personnel are paid for the purpose of managing
students taking paper-based tests and collecting their test sheets.

1.2 Automated Testing.

With a testing center, greater efficiency is achieved. Time, money, and effort expended by an institution for assessment purposes are also significantly reduced. Paper-based tests need to be graded by an individual, typically a grader or the professor. Grading is generally a cumbersome and time consuming process, involving giving appropriate credit to answers. Computer systems are capable of processing data at higher speeds with less erroneous results, if the underlying design is accurate. Automation in the form of auto-grading substantially helps reduce time, effort and therefore money. Automation may restrict students from malpractices and enforce stronger security validations during the taking of a test.

iNetTest is an automated computer-based testing system that was developed by Utah State University’s Computer Science Department and is housed in the TARLab (ESLC 131), an on-campus automated testing center.

Instructors using iNetTest can choose from multiple options such as creating, editing, updating, deleting, or scheduling a test for either an individual or a group of students. They can choose from ten different types of pre-defined template questions (e.g. fill in the blank, matching, programming, etc.) while creating tests. Automatic grading works in the background for each of these question types except the essay type question, and for certain forms, the programming question. This helps in ensuring a quick and efficient on-the-spot grading. With instructor permission, students may choose to see their grades as soon as they finish taking a test.

Other attributes such as a time limit, expiration date, an IP range, etc. may also be set for a test, to provide security. iNetTest has a range of over 20 different permission levels to choose from in order to restrict access to sensitive data. The ability to create groups and share access to these groups between instructors is also possible.

Miscellaneous features such as sending e-mail notifications, text message notifications, cheating logs, and on-the-spot test unlocks help in making iNetTest ready for commercial
iNetTest is a Java-based system that utilizes current J2EE technologies running within a JBoss Application Server 6. Older but stable technologies such as struts are used to interact with the business logic between a client and server. A postgresql database holds all relevant data accessed and managed by iNetTest and is interfaced using Enterprise Java Beans. The front end or user interface utilizes the most recent jQuery tools to provide a dynamic web environment.
Chapter 2

Problem Analysis

2.1 Overview.

Although iNetTest was well equipped to handle all its demands from students and instructors, because of limited resources, it was necessary to develop a scheduling capability to allow for efficient use of computer workstations for test takers. Such a reservation system was recently developed and put to use on iNetTest [1]. This reservation system required an instructor to schedule a test with a valid start and end date. Only tests created by an instructor or tests to which an instructor was granted access were allowed to be scheduled by him/her. After the instructor had scheduled a test (see Fig. 2.1), the students were required to log-in to their respective iNetTest accounts and make a reservation.

![Fig. 2.1: Instructor view of iNetTest’s test scheduling system.](image)

Only authorized scheduled tests would be highlighted on a student’s log-in page, requesting him/her to make a reservation (see Fig. 2.2).

![Fig. 2.2: Portion of student’s page indicating him/her to make a reservation.](image)
To make a reservation, a student was asked to choose from a list of days and available time slots. After the student made a reservation for a test, the system would reserve a seat until the student appeared during his/her reservation time to take it (see Fig. 2.3).

Fig. 2.3: List of time slots opened for a student to make a reservation.

On the day and time of a student’s reservation, a proctor would first authenticate the identity of the student via an ID-card and then assign the student to a computer to take the test. The test was automatically unlocked after the reservation time began and the “take test” button on the student’s page was made active (see Fig. 2.4).

Fig. 2.4: Test is unlocked when a student’s reservation time starts.

2.2 User Roles.

To better understand the existing scheduling model, the needs of all users interacting with the model must first be defined [1]. Four such users with their respective roles are:
2.2.1 Student.

Students are users with lowest privileges and permissions. Their interactions are only limited to the student interface. Being test takers, they can take tests, make reservations and cancel or reschedule their reservations.

2.2.2 Proctor.

Proctors are personnel appointed by the testing center, who have privileges to administer tests. Their duties include authenticating student identities, admitting students and supervising the testing center. They have limited privileges; they can neither manipulate or schedule a test, nor change a reservation. They can however, override the scheduling system by unlocking a test. A student without a prior registration can thereby take a test in case of a vacant seat.

2.2.3 Instructor.

Instructors are persons responsible to create, edit, delete or schedule a test for a student. They author tests and have higher privileges and permissions than both students and proctors.

2.2.4 Administrator.

Administrators are synonymous to super users. They have full privileges and permissions to iNetTest. They also have access to adding, deleting, and editing information of students, proctors and instructors in the database. They can also access test and lab information, as well as perform most over rides to make changes within iNetTest.

2.3 Existing Scheduling Model.

The existing scheduling model laid a simple and straightforward workflow. The instructor created and scheduled a test within a range of valid dates. All students included in the instructor’s group had an automatically enabled option to reserve a seat for the corresponding test after logging into their accounts. If email reminders had been enabled by the
instructor, the student would get a reminder mail providing test time and date information, on registering for the test.

To stratify time intervals during the working hours of a lab for which a student reserves a seat, a decision to have a granularity of 30 minutes was chosen in the existing scheduling system. Thus, the student could choose time instants round the clock separated at 30 minutes to reserve a test. i.e., 12:00, 12:30, 1:00 and so on. The scheduling algorithm ensured that no student was allowed to make a reservation, if a full portion of the time frame was not available during the time instant that he/she wished to make a reservation. This was also the case when the lab was completely reserved for the day. One of the challenging aspects the scheduling system dealt with was the element of uncertainty. That is, if the student came in late for a test, the system would check to see if the matching reservation could be incremented by factors of 30 minutes until the student got his/her full time, or would give him/her an option of rescheduling the test. If the student still wished to take the test, he/she would have to take the test in the remaining time. A seat could only be guaranteed, if a reservation had been made by the student. The scheduling system allowed for multiple tests to be scheduled at the same lab, and also dealt with conflicts of variable test times.

With the existing system, it was seen that an optimal scheduling for a lab relied entirely on the promptness and assurance of a student to reserve a test once it was scheduled. The lab faced a problem of uneven distribution of reservations, which meant that certain peak hours of the day were entirely filled while other hours were relatively vacant. Suboptimal distributions of reserved tests also led to waste of time segments. For example, if a 60 minute test was reserved by student “A” between 12:00pm to 1:30pm and student B made a reservation from 2:00pm to 3:30pm, the time segment between 1:30pm to 2:00pm was unused. The granularity for making a reservation on the system was placed at 30 minutes. While making a reservation, the system added 10 minutes to the test duration. These 10 minutes were provided as a buffer time for each student to collect their belongings after they finished with a test. Thus a 60 minute test was calculated at 70 minutes while making the
reservation, and since the granularity was placed at 30 minutes, the 70 minute test made a 90 minute reservation. However, a 50 minute test made a 60 minute reservation.

It was also noticed that a set of students always made last minute reservations, and were hence limited to inconvenient time slots. (example: early morning or late night). Furthermore, the loss in time segments compounded when students made reservations for various kinds of tests having completely different reservation lengths (example: test 1 = 30 minute duration, test 2 = 60 minute duration, etc).

2.4 Motivation.

The existing scheduling system brought about a significant change by alleviating most of iNetTest’s scheduling problems. TARLab, the testing center running on iNetTest was now capable of handling and addressing the needs demanded by an automatic testing center. However, after using the scheduling system, it became clear that a new set of strategies needed to be implemented to resolve certain problems unaddressed by the old scheduling system. These were:

• Time Efficiency.
  As explained earlier, while scheduling a 60 minute test, a reservation of 90 minutes was made by the system, leading to a static wastage of 30 minutes.

• Inconvenience in registration.
  There was no provision for students to make on-the-spot reservations. Students always had to make a prior reservation for taking a test. The soonest a student could make a reservation for a test was at the next 30th minute interval.

• Additional responsibility on the proctor.
  The first version of iNetTest was without a scheduling system. It required a student to walk in during the working hours of the testing center, and if seats were available, the proctor could authorize him and manually unlock the test on a computer. The proctor performed this unlock on the student’s computer by entering his/her credentials (user id, password). This required the proctor to physically move to the student’s computer
and unlock it for the student. To ensure that no test continued beyond the testing center’s working hours, the proctor needed to know the duration of a test he/she was about to unlock for the student (see Fig. 2.5).

With the addition of the scheduling system, a test was automatically unlocked for students as soon as their reservation time started. This eliminated the need for the proctor to move around performing unlocks. The scheduling system enforced stringent measures to ensure that no reservation time exceeded the testing center’s working hours. A reservation to a student was denied if the system detected a time conflict. So, a proctor never needed to know the duration of a test that a student was about to take. This was a great improvement, but the system did not address the possibility of students arriving at the testing center without a reservation, but wishing to take a test. This meant that the proctor had to utilize the old method of performing unlocks while overriding the scheduling system. Ideally, this problem would not have arisen if students always made prior reservations. However, with the scheduling system on iNetTest running at TARLab for a year, it was observed that there were a large number of students who never made prior reservations and relied on the proctors to unlock tests.

Fig. 2.5: Proctor unlock.
• Mandatory reservations and speed of check-in.

As noted, the system did not automate the process of accommodating students without prior reservations. Moreover, for every student who came in during his/her reserved time, the system required the proctor to open up a dialog box for that time and monitor the list of students who had made reservations at that time. The proctors had to checkmark the corresponding student’s name from the list and press a button to check-in the student (See Fig. 2.6). This meant additional work for a proctor while also decreasing the speed of a check-in for a student.

![Fig. 2.6: Proctors interface to commit check-ins.](image)

The existing scheduling system did bring about a greater efficiency to iNetTest as a module in whole, but the opportunity to improve the system continued to exist. This report explains how a few discrepancies were eliminated and a new interface for taking tests was provided. Detailed analysis is provided in the next section.

### 2.5 New Scheduling Model and Interface.

The new scheduling model proposed an entirely new interface. Here, all check-ins for students under single or multiple instructors having single or multiple tests were made accessible to a proctor. The new model relieved a proctor from unlocks, created on-the-spot reservations for students without prior reservations, and increased the testing center’s time efficiency.
2.5.1 The Card Swiper Interface.

I. Since test takers are generally registered USU students, they each have a unique identity card (with a unique A-Number). Throughout iNetTest, these A-Numbers were used to uniquely identify students and retrieve their related information. Because the A-Number plays a key role in accessing a student’s information it was used for performing further tasks such as check-ins, making reservations, etc. To take full advantage of a physical ID-card and use it as a key to access information, the new interface was designed to accept data directly from an ID-card.

II. A magnetic card swipe reader was used to accept data from an ID-card which was then parsed to find the actual A-Number and perform automatic check-ins for students. The new card swiper interface was designed to directly access iNetTest’s database for relevant student and test information upon the entry of a valid A-Number. The new interface retrieved a students information after swiping the ID-card. All relevant upcoming test details were displayed on the proctors screen so he/she could assign a student to a test station and unlock the desired test. In case the card reader generated an error in reading a student ID-card, or if the student did not have an ID-card present with him/her, the interface allowed the proctor to perform a keyboard entry of an A-Number.

2.5.2 The New Scheduling Model.

The original design of the scheduling system required all students taking a test to have prior reservations. A significant challenge for the new scheduling system was to accommodate students without a reservation. A central requirement of the automated check-in and scheduling system was to provide iNetTest’s existing scheduling system with these additional features.

I. In order to accommodate students with no reservation and to make on-the-spot reservations with check-ins, all student modalities were considered. These modalities were as follows:
[1.] Student is scheduled for only one test and has made a reservation.

[2.] Student is scheduled for multiple tests and has made multiple matching reservations.

[3.] Student is scheduled for multiple tests and has made reservations for some but not all tests.

[4.] Student is scheduled for either single or multiple tests and has not made reservation(s).

[5.] Student has arrived to take a test, but their instructor has not entered the test in the scheduling system (no information).

II. Given a student with only one test and a corresponding reservation (case 1), the interface assumes that he/she should be checked-in and commits a check-in. If the student arrived early for the test, the interface prompts with a message indicating when the reservation was set and the wait period before his/her reservation starts.

III. Given a student with multiple scheduled tests and matching reservations (case 2), the interface allows the student to choose a test that he/she wishes to be checked-in for. The interface also enumerates all valid details such as test name, duration of the test, test reservation status, calculator options, etc. This enables the proctor to confirm with the student the test that he/she wishes to take and thereby remind him/her of the test details before selecting an option.

IV. Given a student with one or more tests without reservations (case 3 and 4), the interface again enumerates all test details. Upon attempting to take a test with no prior reservation, the system makes an on-the-spot reservation for the student by checking future available time slots to accommodate the entire test. The system also checks to see if there are vacant seats available within the testing center at the current time. If there are time conflicts, the system prompts a message notifying the cause. If not, the system makes an on-the-spot reservation, creates an entry in the reservation
V. If a student has no tests scheduled at the moment by any instructors, the interface detects this and prompts with a message stating the same. The proctor may then inform the student that he/she has no tests scheduled and hence cannot take a test.

VI. As discussed earlier, reservations in the old scheduling system were set to a granularity of 30 minutes for a test. If the duration for a test was 60 minutes, the existing scheduling system made a 90 minute reservation. In order to prevent this wastage of time and provide an exact time slot for a student taking a test, a method to clock-in the precise start and end times of a test was required. Two cases are possible when a student makes a reservation and starts a test:

[1.] If a reservation is made at 3:00pm for a 60 minute test, the seat is reserved until 4:30pm. So an entry with a start time set at 3:00pm and end time set at 4:30pm is made in the reservation table. However, when the student actually logs in and hits the “take test” button, the start time of the corresponding reservation must be set to the actual start time which is the system’s current time (say 3:06pm). The end time of the test is computed using the actual start time. Thus the end time of the reservation must be changed to 4:06pm from 4:30pm. This is because that would be the last time instant at which the student can take the test. It is not needed to keep the student’s reservation active after 4:06pm. Here, the student gets an exact reservation length equal to his/her test duration and is not allowed to occupy the seat after the end time.

[2.] Student makes an on-the-spot reservation and starts with a test, but finishes it earlier than the calculated test end time (say 3:30pm). Although the test end time is till 4:06pm, there is again no need to keep this reservation active beyond 3:30pm. So, a method is created to detect when students hit the “finish with test” button. The student is no longer allowed to occupy his/her seat after he/she finishes with a test.
VII. In order to increase the testing center’s time efficiency by meeting the preceding two requirements, a mechanism to continuously update the lab information was also created. This information included the total number of seats available, the number of seats occupied, and the authorized IP range for test access. A time label accurately indicating the time at which the next seat in a lab would be available is also be displayed on the interface. If a student tries to check-in when there are no vacancies in the lab, this information is useful in informing the student when a seat is most likely to be available for taking a test.

The uncertainties of developing such a system on a previously laid architecture made it challenging and at the same time extremely educational. The new scheduling system takes advantage of the previous scheduling system’s features and attempts to make it more efficient by adding new features to it.
Chapter 3
Design Analysis

3.1 Overview.

The new scheduling model adds an additional layer to the existing scheduling model. In order to provide for on-the-spot reservations and check-in with the use of ID-Cards, a new interface was designed. This interface manages most of the work through software and leaves less work for the proctor. The new scheduling system may be split into two components:

1. The card swiper interface: The front-end responsible for interfacing iNetTest’s new scheduling model to a magnetic card swipe reader.

   II. Automated check-in and scheduling model: The back-end that provides features such as automatic unlocks and on-the-spot reservations while increasing the testing center’s time efficiency.

3.2 Software Analysis.

Interactions with the new scheduling system can be understood in detail if considered from the user’s point-of-view. Since the new scheduling system was designed to accommodate more dynamic decisions, it presents many sub-cases corresponding to many options and possibilities. Before describing how the card swiper interface and the automated check-in and scheduling systems work, interactions between users and the system are described briefly with the help of certain roles.
3.2.1 User Roles.

User roles can be classified into student roles, proctor role, instructor role and administrator role. These roles are described below.

**Student Role.**

The previous scheduling system required the student to make a reservation to be assured of a seat. An automatic unlock was also created for the reserved test. The new scheduling system accommodates students without reservations. Hence the student has two choices:

1. Student wishes to make a reservation.

   When a test has been scheduled by an instructor for a student, the student is notified via an e-mail to make a reservation for the test. The student logs in to iNetTest and looks at the “To Do” section for any notifications. For each test scheduled for a student, the “To Do” section highlights the test name along with the option to reserve it. On clicking the “Reserve button”, the student is presented with a list of days and time slots for making a reservation. Once a suitable choice is confirmed, the reservation is made and another e-mail indicating the date and time of the reservation is sent to the student.

   After the student has reserved a seat, the reservation remains active until he/she has finished taking the test or the reservation expires. A student can cancel or reschedule a reservation, if the option is enabled by the instructor while scheduling the test.

   On the day and time of the test, the student needs to present an ID-card to the proctor for checking-in. The proctor would swipe the ID-Card on the magnetic card reader and depending on what time the student arrived, the interface decides if the student may be checked in or gives a reason for failure. Upon check-in, the student automatically has his/her matching test unlocked and can immediately start taking the test.

2. Student does not wish to make a reservation.

   In this case, the student fails to make a reservation after receiving an e-mail from an
instructor about a scheduled test. The student attempts to find an available seat at
the time he/she arrives at the testing center. As in case (1), the student enters the
testing center with an ID-Card and the proctor will swipe the card to determine if
there is an available seat. If the result is successful, the student is allotted a seat and
the test is automatically unlocked. If there is a failure, i.e. no seats are available;
the system notifies the proctor of the next available time a seat is likely to be vacant.
The student is informed of this time, so that he/she may come to the testing center
at that time. There is no accommodation for development of a waiting list. However,
if the student wished, they could go to another lab and log into iNetTest, and make
a reservation.

Proctor Role.

In the new scheduling system, the proctor is not required to unlock a student’s test by
logging in to the student’s computer. He/she is also not required to keep a count of the
number of seats available in the lab. The system takes care of this and updates the proctor
of the lab information every 8 seconds. The proctor is only required to validate a student’s
identity prior to a check-in.

The proctor also has the option to follow the old no-reservation checking method of
admitting a student by checking a hard-copy list of students for a given test to perform
check-ins. As before, he/she may also perform unlocks directly on the student’s computer.
Whether a student comes in with or without a reservation, the proctor is always presented
with information regarding a student’s reservation status from the system.

Instructor Role.

The role of an instructor is to create and schedule tests. Generally, an instructor
schedules a test for all the students in a group (class). An instructor may choose to create
a new test or use a previously created test. He/she may also request access to tests owned
by other instructors to include questions, sections or perhaps the entire test. Once an
instructor has finalized their question set and assigned weights (values) for the questions,
he/she may proceed to schedule it.

On clicking the “schedule” button the instructor gets a list of options. These include selecting the time frame for scheduling a test, options for allowing students to reschedule the test, and options for an instructor to be notified via e-mail when a student schedules a test.

Next, the instructor specifies a location and region where the test would be administered (ex. TARLab). He/she may also choose to create a text field to indicate the list of materials a student is allowed to bring in to the lab (ex. calculator, index cards, periodic table printouts etc.). Once these options are submitted, iNetTest determines if there are sufficient available resources (seats) to administer the test in the specified lab during the selected times. If the scheduling is successful, the instructor is informed of this and the students are asked to schedule the test(s) at the specified day(s)/time. If there are any errors in scheduling a test the system informs the instructor of the reason for failure.

**Administrator Role.**

The role of an administrator is similar to a super user. They may assume any or all privileges in iNetTest to create, edit, delete, or reschedule tests, and make database changes. They hold control over the most sensitive information in iNetTest. Administrators are required to perform these operations on iNetTest when something goes wrong or the iNetTest interface restricts any of the above users from legally changing information. Hence, administrators are only sought after when a valid reason to make changes to information within iNetTest is made unavailable on the interface, but is crucially important.

**3.2.2 The Card Swiper Interface.**

The card swiper interface is explained in greater detail in this section. In order for a student to check into iNetTest, the previous scheduling model required the proctor to view a list of reservation entries made by students and commit check-ins. This meant that when a student entered the testing center to check-in, the proctor had to verify the student’s identity and check to see if he/she existed on the list of reservations, and then, check him or her in.
The new interface uses a student’s ID-Card to facilitate and speed check-in. The typical format for any USU registered students ID-card/A-Number is a character “A” followed by 8 digits (ex. A00000000). This number is assigned uniquely to each USU registered student and remains during their tenure at USU. The ID-Card is issued to a student when he/she enrolls at USU as a student or faculty member. Since the A-Number is unique, it is also encrypted on the magnetic strip at the back of the ID-Card. This allows every student to gain quick access into USU’s recreational facilities such as gymnasiums, racquet ball courts, swimming pools and other places such as libraries, labs, etc. by swiping their ID-cards.

iNetTest provides computer based tests to be housed in a testing center within USU’s campus called the TARLab. The TARLab houses approximately 35 computers that are recognized by iNetTest as authorized testing machines. To make check-ins to the TARLab as straightforward as for other USU facilities, a magnetic card reader and swiper interface were needed to be integrated into the existing iNetTest scheduling system. The interface required the following capabilities:

Terms:

- **Simple check-in**: Procedure to validate a student’s identity and perform a check-in if he/she has made a reservation and arrived at the testing center during the reservation time.

- **Complex check-in**: Procedure to validate a student’s identity and attempt to perform an on-the-spot reservation if the student had made no prior reservation for a test, and then perform a check-in.

The criteria required for the interface are listed below.

- The interface must only be made available to users with permission levels from a proctor and higher. A brief text label instructing how to interact with the interface and an area where dynamic interactions take place between the proctor and system must be clearly delineated.
• The interface should be designed to always be ready to accept inputs from the magnetic card reader when it is started.

• As the interface would be used only by a proctor to commit check-ins, the information displayed on it must be restricted to only limited and valid test details and student information. This becomes crucial as the interface queries iNetTest with a student’s A-Number. iNetTest’s database holds all records for a student, some of which are confidential and do not pertain to a proctor’s role.

• The interface must also provide for creating on-the-spot reservations for students. For this, it is necessary for a proctor to know if he/she was directly committing a simple check-in or complex check-in.

• A simple editable text-box displaying the student’s A-Number upon a card swipe should also be present. In case of any system parsing errors or a damage to the magnetic strip on the card, the proctor should also be able to manually enter the A-Number to proceed with regular interactions on the interface.

• As data from a magnetic card reader is in the form of serial keystrokes, the interface must include a method that checks the text box for a possible A-Number format-match to directly trigger a check-in procedure. If this is not done, the proctor may be required to hit a button after every card swipe entry to trigger the check-in event.

• A separate portion of the interface must be dedicated to detailing the testing center’s information. Details such as lab-name, total computers available in a lab, total computers vacant and an IP-range of all authorized computers within the testing center must be updated periodically and displayed in this portion.

• Details enumerating the students A-Number as well as the first and last name must always be displayed on the interface upon a successful swipe. Given a set of tests a student is scheduled to take, the interface must also enlist details such as test-name, test-duration, test-status and calculator options for each test.
• The interface is designed to reduce load on a proctor and yield dynamic results for a card swipe. Quick textual responses from the system should be accompanied by consistent graphical icons representing a success or failure.

• In the possibility of a system error, a “reset” button to reset the interface is also required to be present at all times. To quit the interface a “cancel” button should also be present.

3.2.3 Automated Check-in and Scheduling System.

From the previous discussion, we can summarize the following points.

• The old scheduling system required every student to make reservations before they could take a test.

• Reservations could only be made on the half-hour, since the granularity to make reservations on the calendar was set to 30 minutes. As discussed earlier, when a reservation is made for a test, the reservation length is set to the duration of a test plus a constant of 10 minutes. This additional 10 minutes is included as a buffer time for a student to collect their belongings before they leave a station after taking a test. So, a reservation for a 60 minute test would increase to 70 minutes, and since the granularity was set to 30 minutes, the reservation time would increment to 90 minutes. However, a 50 minute test would be set to a reservation length of 60 minutes.

• If a student “A” starts taking a 60 minute test exactly when the reservation started, he/she would finish with the test at the most in 60 minutes. But since a 60 minute test makes a 90 minute reservation, 30 minutes are entirely wasted.

• If a student “B” were to finish a 60 minute test within the first 10 minutes, the scheduler will not make a seat available for the remaining 80 minutes. If another student “C” arriving at the testing center wants to take a test on the vacant computer left by student “B”, student “C” would have to request the proctor to manually unlock
a test. Furthermore, neither does student “C” or the proctor has knowledge about the exact time left on student “B”s computer.

This meant that there was no mechanism in the old system to fetch the time instant at which a student started and ended a test. There were no checks performed on the number of students who finished their tests earlier than their reservation time. There was also no method that deleted a student’s record from the reservation table after a test was finished. When a reservation is deleted immediately after a student finishes a test, the chances for another student to reserve a test are higher. Lastly, there was no automatic procedure to legally make an on-the-spot reservation for a student wanting to take a test.

After a year of housing tests at TARLab with the scheduling system, a trend was observed that very few students actually made prior reservations. Most students relied on the proctor to unlock tests for them if a seat was available. TARLab faced an unequal distribution of test takers.

Providing legal on-the-spot reservations.

After verifying a student’s A-Number, the system gets a list of all scheduled tests for a particular student. It stores the “test_id” of all the scheduled tests. The system then looks into the reservation table, and checks via the received A-number for any entries. If the system finds any entries, it only extracts the “test_id” column from the set of entries in the reservation table and stores it. It is now possible to determine the number of tests a student has reserved and not reserved. The system compares the two lists of “test_id” and if a match is found, the test is declared as a reserved test. If no matches are found the test is declared unreserved for a student and a new list of “test_id” containing only unreserved tests is created. With this list of unreserved “test_id”, the system proceeds to make on-the-spot reservations.

The old scheduling system defined a granularity of 30 minutes to make a reservation. Reservations could only be made at the 30th minute round the clock. i.e. 12:00pm, 12:30pm, 1:00pm etc. Furthermore, a reservation could only be made at a future timestamp. So, if a student arrived at the testing center at 12:15pm, a reservation could only be made at
12:30pm and not 12:00pm. So an on-the-spot reservation is made at the next 30th minute interval. i.e. 12:30pm.

The need for providing an on-the-spot reservation is so a student can be checked-in immediately after a reservation is made and have the test unlocked. In the old scheduling system, a check-in was only possible if a student’s reservation had started. However, since the reservation was made for a future timestamp, the student would have to wait until the reservation started. So, the reservation time for a student is now brought back to the previous 30th minute interval. i.e: 12:00 pm. Now as the system’s current time is 12:15 pm and the reservation had started at 12:00 pm, a student can be checked-in.

Whenever a reservation is made, a corresponding unlock for a test is also created at the reservation time. When the reservation begins, the test is automatically unlocked for a student. So, since the reservation time was brought back to the previous 30th minute interval, the corresponding unlock in the unlock table was also set to the previous 30th minute interval. Now after a student is checked-in, the matching test is automatically unlocked.

**Calculating the exact time period for a test.**

The previous scheduling model did not have a method to calculate the precise time period used by a student to complete a test. There was also no method that detected if a student had completed a test. To calculate the exact time period used by a student to complete a test, two additional timestamp columns are included in the reservation table. When the student starts taking a test, the start time of the test is stored in one column labeled “test start time”. This start time is the system’s current time. A precise end time is also calculated by adding the test duration to the start timestamp. This end timestamp is stored in the other column labeled “test end time”. When a reservation is made, the value of the start time column is set to null and the value of the end time is set to the reservation length. After a student logs-in and starts taking a test, the precise start time and end times are calculated and stored in the matching entry of the reservation table. This information is useful for making seats available the future. The test end time is an exact value for which
a student can occupy a seat in the testing center for taking a test.

To indicate if a student has finished with a test, a Boolean column labeled finished with test is also added to the existing reservation table. This column is set to false by default when an on-the-spot reservation is made for a student. When a student taking a test decides to finish with a test, the matching entry for the student in the reservation table is referred, and the finished with test column is set to true. This column in the reservation table helps to determine all the students who completed their test before the calculated test end time.

**Releasing vacant seats.**

With a testing center’s lab-id, the system looks up the lab table and obtains the total number of seats available for a lab. It compares this count with the total count of entries in the reservation table which represents the number of seats currently occupied. If the count of entries in the reservation table is less than the total count of seats in the lab, it concludes that a seat is vacant. If the count of entries in the reservation table (seats currently occupied) is equal to the total count of seats in a lab, it concludes that a seat is unavailable.

The current system uses the reservation table to store entries of reservations that are ongoing. That is, it only keeps entries of test takers who are currently taking a test at the testing center and deletes entries when they are finished. i.e. if the TET column entry in the reservation table shows a time in the past, the entry in the reservation table is deleted. The TET values are compared with the system’s current time before they are deleted. The TET value for every student in the reservation table is the exact time at which the student’s test would time out. So a student can no longer occupy a seat in the testing center if a test is timed-out, the system assumes that he/she has completed the test.

Reservations for students who complete the test before the TET are also deleted from the reservation table. The FWT flag is a Boolean column that is set to false when a reservation is made. When a student finishes with a test, the FWT column for the student’s entry in the reservation table is set to true. The system looks into the FWT column in the
reservation table and deletes all entries that are set to true.

This helps the system to release unused seats. The procedure to delete entries from
the reservation table is called periodically at 8 seconds. So, every 8 seconds the system
attempts to release unused seats. This procedure starts when the card swiper interface is
launched and exits when the card swiper interface is closed. Every 8 seconds, if the system
releases a seat, it informs the proctor via the card swiper interface of a seat availability. It
also allows for on-the-spot reservations. If the testing center is at capacity and the system
is unable to release seats, it informs the proctor of the next time at which a seat can be
made available. It does this by storing a list of all TETs and displays the smallest TET in
a date-time format.

Even if seats are unavailable and the proctor gets a timestamp of the next seat avail-
ability, a seat can potentially become available before the predicted time. This occurs if a
student finishes their test before the allotted TET.

3.3 Hardware Analysis.

The hardware used in the automated scheduling and check-in system is described in
this section.

3.3.1 The Magnetic Card Reader.

In order to read information from a student’s ID-Card, a magnetic card reader “4000M
series by Scan Technology, Inc” is used. Magnetic card readers are capable of reading the
information stored in magnetic stripe cards. This information is usually in an encrypted
form and is stored by modifying the magnetism of tiny iron-based particles on the band of
a magnetic stripe card [2]. The output obtained from the magnetic card reader is in the
form of simple key strokes, like symbols from a keyboard. To extract relevant information
from the output of the magnetic stripe card, the information has to be first decrypted and
then parsed.

Data on a magnetic card reader is stored in the form of “tracks” [3]. Current day
magnetic stripe cards have up to three tracks and different information can be stored on
each of these tracks. This stored data can be read in numerous ways, such as serially reading either all or a few of the tracks, or through another method called field parsing. Field parsing is used in the system so that the A-Number can be directly extracted from the card reader and unwanted data can be omitted. Field parsing helps in creating pre-ambles and post-ambles to the information, and it also makes provisions for setting offsets for reading the data.

The 4000M series card reader supports interfacing with a computer via a keyboard wedge, USB, USB-HID, or USB-Serial, and up to 3 different card formats. In this application, the proctor’s computer is interfaced with the magnetic card reader via a USB.

The magnetic card reader is also capable of reading information from other sources such as credit or debit cards and driver’s licenses. Scan technology provides a Windows based software and driver to program the unit in order to extract only relevant information. Since the magnetic card reader for the card swiper interface is only used to extract a student’s A-number, it is programmed accordingly. Magnetic card readers have low power consumption, compact size, and are portable (see Fig. 3.1 and Fig. 3.2).

Fig. 3.1: The Magnetic Card reader.
Fig. 3.2: The Magnetic Card reader with A-Card being swiped.
4.1 The Card Swiper Interface.

The card swiper interface is the front end of the automated check-in and scheduling system. It is interfaced with the magnetic card reader 4000M series. It is also the sole medium through which all the information is displayed to commit check-ins. The interface is programmed to receive data from the magnetic card reader every time a card is swiped. It uses the latest JQuery-UI technologies to render sharp appropriate graphics. The interface is designed using numerous JQuery effects and transitions to facilitate easier understanding for the proctor while presenting information and accepting inputs. The card swiper interface is divided to 3 parts, 2 of which render dynamic results and one which stays static.

The first section is called the student input section and illustrates how to interact with the interface. A textbox is provided to display a students A-Number when a card is swiped. If a student is unable to present an ID-Card on entering the testing center, or if the magnetic card reader renders an error, the proctor can directly input an A-Number in the textbox via their keyboard. To submit the A-Number to the automated check-in and scheduling system, there is no need to hit the enter key or click any button. This is because the interface uses a smart regular expression matching technique to look for a possible A-Number format match. An A-Number has a total of 9 characters, of which the first character is an A, followed by 8 digits. When an A-Number match is found, the interface directly sends the information to the automated check-in and scheduling system. The regular expression matching technique helps to reduce the overall time required for a proctor to perform a student check-in (see Fig. 4.1).

The second section is called lab-information section on the card swiper interface and includes a container for displaying the testing center’s information. The card swiper interface
Fig. 4.1: The card swiper interface displaying only the student input section.

receives the testing center’s information from the automated check-in and scheduling system when it is launched. The information presented to the proctor includes the lab-name, total available seats, total vacant seats, the authorized IP-range etc. This information changes when a student finishes a test or times-out from a test and makes the matching seat vacant. In order to display seat availability to the proctor, the information is updated every 8 seconds through an AJAX call from the automated check-in and scheduling system. The time limit of 8 seconds is chosen to reduce the number of times the system references the iNetTest database while also providing for real time seat availability. The proctor can also extract information on the current availability of seats through the interface. If a seat is available, a message indicating the seat availability is displayed. If a seat is unavailable, a message indicating the next time instant at which a seat is likely to be available is displayed. Apart from text messages, an added feature used is color coding to display the
availability of seats. The colors red and green are used to highlight messages. Red indicates no seats available, while green indicates that seats are available. Such a color coding technique makes it easier for the proctor to perform check-ins as the need to read messages becomes optional. To further aid the proctor, a progress bar indicating the percentage of lab occupancy is also created using the JQuery-UI. Information regarding the percentage of seats currently occupied can be obtained from the progress bar (see Fig. 4.2).

Fig. 4.2: The card swiper interface displaying the lab-information section.

The third section is called the check-in section and displays detailed information pertaining to all tests scheduled for a student. The proctor uses this information to check in a student. After an A-Number is processed, if a student has any test(s) scheduled at the testing center, the automated check-in and scheduling system displays this information in the check-in section of the card swiper interface. If a student does not have any tests scheduled, no data is presented to the proctor, forbidding him from committing a check-in.
Furthermore, only if a seat is available at the current time at the testing center, can the proctor receive information from the automated check-in and scheduling system to commit check-ins.

After an A-Number is processed, a label with a student’s first and last name is displayed. Other useful information such as the number of scheduled tests and a reservation status is also displayed within the same label. For every test that is scheduled for a student, the matching information required for a check-in is presented to the proctor through a JQuery-UI widget called an "accordion".

Each tab on the accordion is set to automatically open and display relevant test information on a mouse-hover event. Each tab heading contains a test name and the student’s reservation status for that particular test. The reservation status shows the keyword “reserved” if the student has made a prior reservation for the test, or shows the keyword “unreserved” if the student has not made any reservations. With this basic information presented to a proctor as tab headings, the proctor does not need to hover over every tab to access the test details. He/she can obtain the test name directly from the student and proceed to check-in by hovering the cursor on the matching tab.

On opening a tab in the accordion, a list of test details is displayed to the proctor for committing a check-in. This list contains:

- Test name
- Test duration
- Test status
- Calculator options

Each tab also contains a button labeled “Attempt check-in”. A check-in for a student can be attempted by clicking this button. In the accordion, only one tab can be opened at a time. Hovering the mouse over another tab causes the previous tab to close. This helps in displaying only relevant information to a proctor during a check-in.
Since a magnetic card reader reads data only when a magnetic stripe card is swiped across it, a JQuery-UI effect is used when data is displayed on the interface. The data is presented from left to right as a slide effect (see Fig. 4.3).

Fig. 4.3: The card swiper interface displaying the check-in section.

In each stage of a check-in, colored icons are used for graphical representation of a success or failure, along with text messages and JQuery-UI effects. Successful check-in of a student is displayed with a “Tick” icon. Errors or failure to check-in a student is displayed with a “Cross” icon. An “Exclamation” icon is displayed when more information is needed from the proctor. Without having to read the text message, just by looking at the icon displayed, the proctor can obtain the information that he/she needs. When the interface is idle, an animated-GIF of a swiping card in motion is kept active prompting the proctor to perform card-swipes.

Messages are displayed on the proctors screen for 5 seconds, after which they timeout,
requiring the proctor to retry a check-in or try another student’s check-in (see Fig. 4.4 and Fig. 4.5).

![Interface indicating a successful check-in.](image)

**Fig. 4.4:** Interface indicating a successful check-in.

Since magnetic card readers are also capable of reading debit and credit cards and drivers-licenses, only valid A-Number formats get positive results. Errors on the screen accompanied with a “Cross” icon are displayed if a card other than a valid USU ID-Card is swiped. At any stage, the proctor may choose to reset the interface if there are any HTML errors or scripting errors. It is recommended to use Mozilla-Firefox as a browser to log-in. The reset button clears all the HTML content on the interface and restarts the interface while keeping all lab-information intact.

### 4.2 Automated Check-in and Scheduling System.

The automated check-in and scheduling system is the backend for the card swiper
Fig. 4.5: Interface indicating failure for check-in as no seats are currently vacant.

interface. The connection between the card swiper interface and the automated check-in and scheduling system is made using a struts framework. The card swiper interface invokes two AJAX calls to the scheduling system, requesting information. These calls are handled by an action class in the scheduling system and a response is sent back to the interface. The scheduling system also responds to other AJAX calls that are made when a student logs into iNetTest. A postresql database is used to store all records and the system is connected to the database via enterprise java beans (Ejb).

The first call to the scheduling system is automatically made when the interface is launched. A javascript “setinterval” function is used to periodically make this call every 8 seconds. This call is responsible to update the testing center’s information on the card swiper interface. Additionally, it also attempts at making seats available to students at 8 seconds. If seats are currently unavailable, it predicts a time instant at which the next
seat is likely to be available. The “clearInterval” function is called when the card swiper interface is closed.

The second call from the interface is made when a valid student A-Number is received. This call is responsible to provide on-the-spot reservation for students and commit check-ins. The system binds all the test information for a particular student and sends it back as a response to the interface (see Fig. 4.6).

Fig. 4.6: Simple design illustration of the automated check-in and scheduling system.

Action 1: Periodic call at 8 seconds to free seats, predict seat availability and update lab information. To make seats available in the testing center, this action looks into the reservation table to delete all entries that have a “finished with test” column set to true. This is because, the “finished with test” column is set to false when a reservation is made and is set to true when a student finishes taking a test. The scheduling system also compares the system’s current time stamp with all the values in the column “test end time” in the reservation table. It deletes all entries in the reservation table that have a “test end time” value lesser than the current timestamp. This is done since the “test end time” is an exact value for each student taking a test. A student’s test cannot be active after the timer runs
out (see Fig. 4.7).

Fig. 4.7: Workflow of Action 1 to make seats available and predict timestamps.

Action 2: Attempting on-the-spot reservations and check-in when a card is swiped

To provide for on-the-spot reservations requires two AJAX calls to the scheduling system. The first call, named ‘get test information’ is made after a valid A-Number is processed to get a student’s test information. As a student can have single or multiple tests scheduled, some which may be reserved or not reserved, the system binds all necessary information for each scheduled test and sends a response to the card swiper interface.

After all test details are displayed on the proctor’s screen, a test is selected by clicking the “attempt check-in” button. The second call named ‘attempt check-in’ is made to
determine if the student requires an on-the-spot reservation and commits a check-in. If the reservation and check-in are successful, a response is sent back to the interface. If the on-the-spot reservation is unsuccessful, the process is rolled back and an error is sent back as a response (see Fig. 4.8).

Fig. 4.8: Workflow of Action 2 to process on-the-spot reservations and commit check-ins.

Lastly, when a student logs into an iNetTest account after checking-in and hits the “Start with test” button, an AJAX call to calculate and set the exact TST and TET is made. After the student finishes a test and hits the “Grade and Exit” button, the FWT column is set to true.
Thus, with providing for on-the-spot reservations for students and constantly making seats available at a testing center, the automated check-in and scheduling system along with the card swiper interface increases the efficiency of a lab by allowing students to immediately take tests.
Chapter 5

Testing

Testing on the automated check-in and scheduling system was conducted with the following setup.

- 10 dummy students with A-numbers ranging from A00000001 to A00000010 were created. Each dummy student's first name had a prefix “tester” and a suffix of the A-number in words. They each had a last name “swiper”. For example, the student with an A-Number “A00000001” had a first name of “testerOne” and last name of “swiper”. The student with an A-Number of “A00000002” had a first name of “testerTwo” and a last name of “swiper” and so on.

- Two dummy tests named “GradTest” and “UndergradTest” were created. The duration for “GradTest” was set to 60 minutes and the duration for “UndergradTest” was set to 30 minutes.

- A lab with the name of “SwiperLab” was created where the two tests were scheduled. “SwiperLab” was limited to only accommodate 5 students at any given time. So the total seats available for testing at SwiperLab were 5.

- The 10 dummy students were scheduled at “SwiperLab” to take “GradTest” and “UndergradTest”.

- The lab was set to be open between 9:00am to 3:00pm. So a total of 6 hours was available on each of the 5 machines at the testing center. Thus SwiperLab was capable of accommodating 30 hours testing in a day.

- As the students had to take both GradTest and UndergradTest, they each required 1.5 hours to complete both their tests at SwiperLab. So all the 10 students required a total of 15 hours at the testing center to complete with their tests.
Testing was conducted for the following three cases.

1. All students made reservations for both tests before taking them.

2. Some students made reservations while others did not.

3. No student made a reservation.

Two distribution patterns of students arriving at a testing center are also considered.

1. Regular distribution: During the working hours of the lab, there is at least one student taking a test in the lab.

2. Irregular distribution: The lab is relatively vacant during a few hours of the day, and is flooded with students at a few peak hours.

When all students make a reservation for both tests, each student makes a reservation for a 60 minute test and a 30 minute test. As discussed earlier, the reservation length for a 60 minute test was set at 90 minutes and the reservation length of a 30 minute test was set at 60 minutes. Since each student makes a reservation for their own respective tests, a total of $90 + 60 = 150$ minutes (2.5 hours) is reserved for each student. As there are 10 such students, a total of 25 hours are reserved on the computers at the testing center. Once a student starts with a test, the automated check-in and scheduling system tries to make seats available for other students. However, since all students have made prior reservations there is no use in making other seats available. The lab can accommodate all the students only during a regular distribution (see definition above) of students through its working hours.

When no students make reservations, the total number of hours reserved on the computers at the testing center is reduced. This is because, after making an on-the-spot reservation and when the student begins the test, the student is allotted only the amount of time required to complete their test. As soon as a student is timed-out from a test or finishes with a test, the seat is made free. Thus assuming a student makes an on-the-spot reservation at 12:15 pm, and starts with a 30 minute test at 12:25 pm, the student’s seat is made free at
12:55 pm and another student can make an on-the-spot reservation for 1:00 pm. In an ideal case, if each student was to start taking a test exactly at the time of making an on-the-spot reservation, a total of 60 + 30 minutes = 90 minutes (1.5 hours) is reserved for each student. As there are 10 such students, a total of 15 hours are reserved on the computers at the testing center. The system in this case can accommodate students at both regular distributions and irregular distributions.

In the case when only some students make reservations, the time efficiency of the testing center reduces as compared to when no students make reservations. Assume the testing center has seats available for say the next 60 minutes, and at the end of that time the testing center has students coming in with prior reservations. Now if student “SwiperOne” (A000000001) wants to take “GradTest” (60 minutes) by attempting an on-the-spot reservation, the system would generate an error. This is because the system always makes a reservation at the next 30th time interval on the clock, and since the system detects the prior reservations made by other students, it cannot accommodate “SwiperOne” to “GradTest”. Thus, although there are 60 minutes available, because of prior reservations, the system cannot process on-the-spot reservations.

TARLab never faced a situation where all students made reservations; most students relied on proctors to unlock tests without making reservations. The testing chapter proves that even in cases where no students make reservations in TARLab, the scheduling system would not only ensure a valid seat but also increase the testing centers overall efficiency.
Chapter 6

Future Work

The Automated check-in and scheduling feature provides an enhancement to increase time efficiency. It decreases the time for a student to check-in and allows for more efficient use of the available workstations by maximizing seat availability to students. The proctor’s screen now predicts the time at which a next seat is likely to be available. Students need not come to a testing center if seats are currently unavailable for testing. If details such as the next seat availability, total seats currently occupied, the testing centers working hours, etc. were made publically available to all USU students, each student who did not have a reservation could either more efficiently make a reservation, or know when a seat is likely to come available.

USU has an existing mobile app implemented for both iOS and Android enabled smart phones. With this app, a student has direct access to all upcoming events at USU, newsletters and articles, maps, USU radio etc. The app also allows students to log-in to their banner accounts. If a separate bulletin board containing TARLab’s information was created, students could simply check app for seat availability.

Another enhancement could be a waiting list. The current scheduling system does not provide a waiting list for students. When the testing center is at capacity, a waiting list could hold for each student, a name and phone number associated with a wait timestamp. When a seat is made available, the student could be immediately informed, via a text message of the seat availability.
Chapter 7

Conclusion

The architecture of the old scheduling system in iNetTest enables the automated check-in and scheduling system to seamlessly integrate with it. iNetTest can now manage computer based tests housed at a testing center much efficiently. The card swiper interface provides a quick technique to fetch a student’s information, while presenting all details for a check-in within one container. It reduces the workload on a proctor and also increases the speed of a check-in for a student. The automated check-in and scheduling system provides for on-the-spot reservations for students while also making seats available to them. Time efficiency for a testing center is highly improved as seats can potentially be made available every 8 seconds. The testing center can now accommodate a larger set of students at a given day for testing. Since the entire reservation process is automated, students can feel free to take a test without prior reservations.
References

