Recourse Limited Testing Center Scheduling for a Web-Based Testing Application

Adam J. Graham
Utah State University

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

Part of the Computer Sciences Commons

Recommended Citation
Graham, Adam J., "Recourse Limited Testing Center Scheduling for a Web-Based Testing Application" (2012). All Graduate Plan B and other Reports. 133.
https://digitalcommons.usu.edu/gradreports/133

This Report is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Plan B and other Reports by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
RECOURECE LIMITED TESTING CENTER SCHEDULING
FOR A WEB-BASED TESTING APPLICATION

by

Adam J. Graham

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. Stephen Allan
Committee Member

Dr. Curtis Dyreson
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah
2012
ABSTRACT

Resource Limited Testing Center Scheduling

For a Web-Based Testing Application

by

Adam J. Graham, Master of Science

Utah State University, 2012

Testing centers are a useful tool to help instructors deliver computer-based tests, but computers resources are expensive and therefore limited. This paper describes a method by which testing center(s) may use iNetTest, a web-based computer aided testing system, to house and administer exams. The algorithm discussed in this paper makes it possible for instructors to schedule tests for a given time frame while ensuring that enough computer resources will be available to all of the students. The algorithm prevents the testing center from getting overwhelmed with students while attempting to maximize the usage of the valuable computer resources.
CONTENTS

Page

ABSTRACT .......................................................................................................................... iii

LIST OF TABLES .............................................................................................................. vi

LIST OF FIGURES ........................................................................................................... vii

CHAPTER

I. INTRODUCTION ........................................................................................................... 1

   Testing Centers ........................................................................................................... 1

   Automated Testing ................................................................................................. 3

II. PROBLEM ANALYSIS ............................................................................................... 5

   Roles ......................................................................................................................... 5

   Problem Description ............................................................................................. 6

III. DESIGN ................................................................................................................... 10

   Integration With iNetTest .................................................................................... 10

   Student Role ......................................................................................................... 11

   Proctor Role ......................................................................................................... 14

   Instructor Role .................................................................................................... 15

   Administrator Role ............................................................................................ 19

   Scheduling .......................................................................................................... 20

IV. IMPLEMENTATION .................................................................................................. 23

   Database ............................................................................................................. 23

   Interface .............................................................................................................. 27
Scheduling Algorithm.......................................................................................... 32

Results.................................................................................................................. 39

V. FURTHER WORK.................................................................................................. 40

Data Use................................................................................................................. 40

Adjustments........................................................................................................... 41

Future Testing....................................................................................................... 41

VI. CONCLUSION.................................................................................................. 44

REFERENCES......................................................................................................... 45
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Example Optimal and Suboptimal Scheduling of Exams</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Schedule Values by Step</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>Sample Test Distribution</td>
<td>36</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dialog Interface for Scheduling a Test</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>West Virginia University Psychology Experiment Results</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>ER Diagram of the iNetTest Database Additions</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>Student &quot;To Do&quot; List</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>Student View for Reserving a Time</td>
<td>28</td>
</tr>
<tr>
<td>6</td>
<td>List of Currently Scheduled Tests</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>Proctor Lab View</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>Student Reservations during a Selected Time</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>Edit Schedule View</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>Pseudo Code for Scheduling a Test</td>
<td>35</td>
</tr>
<tr>
<td>11</td>
<td>Pseudo Code for Distributing Unknown Reservations</td>
<td>37</td>
</tr>
<tr>
<td>12</td>
<td>Scenario Output</td>
<td>39</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

Testing Centers

Testing centers are a useful tool in institutes of higher education. They allow for a controlled environment where students may take a test on their own schedule while being prevented from cheating through proper proctoring. This is advantageous particularly for large enrollment classes. For example, materials entering a testing center can be monitored and controlled more easily and effectively than in an ordinary classroom where all students enter the room and simply take an available seat. While one can never control all improper activities, their likelihood can be reduced significantly in a testing center. Furthermore, those students who would only choose to cheat if the situation presented itself are less likely to do so in such an environment.

The use of a testing center reduces the need for paid teaching assistants. In many cases the number of teaching assistants is increased for the sole purpose of aiding in the administration of the exams for the class. Instead of employing several students in all of the large courses, it is more cost effective to employ fewer teaching assistants and have the exam administered in the testing center where proctors are used for several classes.

In a testing center, students can chose to take a test at a particular time. Instead of taking the test during a single class period, they are allowed the freedom to take the test when it is most convenient or least inconvenient for their schedule. Furthermore, testing center proctors can arrange the students in the room such that those taking the same test are not next to one another, decreasing opportunities to cheat.
At Brigham Young University (BYU), they house the largest testing center in the United States and Canada [1]. This facility has a total capacity for 740 students. They administer 700,000 tests per year giving that test taking time back to the instructors and providing greater flexibility for university staff and students. Nonetheless, the testing center at BYU fails to fully address an important challenge, scheduling reservations for tests. Students simply arrive at the center and wait in line until there is an available seat. When the testing center is overwhelmed with too many students, the center reserves the right to turn students away. This method of scheduling could be especially difficult for tests using the 40 computer stations in the center. Because of the associated costs for hardware, furnishings, and space, it is unlikely that a university could ever afford enough computer stations to handle any number students. Consequently a testing center generally has a subset of the “ideal” number of computer stations. Delivering a computer aided test for a large enrollment class means that some sort of scheduling must be done so that all students do not arrive at the testing center at the same time, e.g. the last hour before the testing period ends. Thus, there is a requirement for a software tool to assist in scheduling a testing lab.

This report describes the scheduling tool developed to meet this requirement. The tool allows instructors to schedule a date(s) for a test and be assured that there are sufficient seats over the time span given for all students to complete the test. It allows students to choose an available time to take a test (first come first served) and be assured that a station will be available at the desired time.
Automated Testing

Automated testing software tools help to reduce the amount of time an instructor must spend in creating and especially in grading class assessments such as tests. The grading process for a test can be monotonous as well as very time consuming. Computers are excellent at performing such tasks, but may pose security issues. For example, with a computer, students have access to the Internet and the wealth of information available through powerful search engines. Another security concern would be protecting the content of the exam so that it is only available to authorized test takers.

The iNetTest system is a web-based computer aided testing system. INetTest uses current J2EE technologies running within a JBoss Application Server 6. It uses struts to manage the business logic between the client and the server. The database interface is implemented using Enterprise Java Beans to access a postgresql database. The user interface utilizes the most recent jQuery tools to provide a dynamic web environment. Instructors using iNetTest can house the creation, editing, administration, and grading of a test over the Internet. The software allows the instructor to specify the time span for a test and the IP range for the test taking computers, restricting when and where an exam may be taken. The iNetTest system also includes software to monitor any web browsing that is done during the test. Because it is a web-based application, iNetTest can be accessed from any web-enabled computer but restricts access to certain test information to authorized users only. INetTest has a complex and flexible security environment allowing for various levels and degrees of access.
INetTest allows for ten different types of questions. The various question types allow the instructor significant flexibility and accuracy in assessing student learning. Nearly all question types are automatically graded with only essay questions to be manually graded by the instructor. Automatic grading of questions can be overridden by the instructor as circumstances require. For example, if a question is inherently flawed or the correct answer is mislabeled, the instructor may choose to re-grade it. These capabilities combined with ease of test creation, delivery, and taking make iNetTest a powerful tool for instructors to use in delivering and administering exams to students. INetTest was designed to make the testing process as straightforward as possible for both the instructor and the student. Considering all of the easy to use features implemented in iNetTest, it is an excellent candidate for a starting point in the development of software to manage a computer aided testing center.
CHAPTER II
PROBLEM ANALYSIS

Even with all of its assessment capabilities, it would be problematic to use iNetTest in a testing center with limited resources, i.e. computer stations. In essence, the system lacks the element of time management for scheduling tests. In order to allow for test scheduling, the needs of four different users must be met; namely, student, proctor, instructor, and administrator. The sections that follow discuss these needs and the problems that a time management system must address.

Roles

As noted, the needs of four different users must be addressed by a test scheduler:

- **Student Role**- these are the individuals who are being tested. They will need to be able to reserve a time in the testing center to take a test.
- **Proctor Role**- these are the employees (proctors) of the testing center(s) that admit students to the center and proctor the exams.
- **Instructor Role**- These are the test authors and owners. They will use iNetTest to create, deliver, and schedule examinations in the testing center(s).
- **Administrator Role**- Administrators are similar to super users in that they have permissions to add instructors, students, and/or proctors to the database and make changes to tests, etc.
Interactions between these individuals present challenging problems in coordinating the demands of scheduling the testing center.

**Problem Description**

The challenges that the time management or scheduling software has to solve relate to the scheduling of limited resources with elements of uncertainty. The workflow for scheduling a test in the testing center is as follows:

1. The instructor creates the test and allows for it to be taken during a specified time frame for the students.
2. The student makes a reservation at a testing center during the time frame that is allowed by the professor.
3. The student arrives at the appointed time, is admitted into the testing center, and takes the exam.

Deciding how to handle the uncertainty between the times that the instructor makes the test available to be taken and when the students reserve a time to take a test is one of the main challenges that the software must handle.

It is not feasible to require that the instructor assign specific test times for each of his/her students. He or she is most likely not aware of students’ individual schedules. If the class size is larger than the testing center size or other tests are already scheduled at that time, then it is also not feasible to require all students to take the test during the regular lecture hour. The only option is to allow students to schedule their own time to take a test limiting the number signing up to the number of seats available.
Allowing students the freedom to reserve a time in the testing center is the cause of most of the uncertainty that the software must manage. It would be infeasible to allow students to schedule tests with a granularity as small as a minute. Thus, test schedules were set on the half-hour. The optimal arrangement for scheduling the students would be one in which all of the seats in the testing center are filled for all times during open lab hours. One cannot assume that students would reserve times in such a way. In the worst case, depending on the length of the test, students could fragment the available time resources to take up 3X more time than the optimal arrangement. Suboptimal arrangements can yield small, useless segments of time for available testing stations. Table 1 shows how times can become fragmented for two exams 1 hour and 1.5 hours in length. Furthermore, not only will there be fluctuations in demand during different times of the day, there will be peak hours that are filled while at other times, the testing center will be relatively unused. These fluctuations are likely to intensify during the times when it is common to offer midterm exams.

Table 1: Example Optimal and Suboptimal Scheduling of Exams

<table>
<thead>
<tr>
<th>Time</th>
<th>Optimal Seat 1</th>
<th>Optimal Seat 2</th>
<th>Suboptimal Seat 1</th>
<th>Suboptimal Seat 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:30</td>
<td>Test A</td>
<td>Test A</td>
<td>Test A</td>
<td>Test B</td>
</tr>
<tr>
<td>2:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:30</td>
<td>Test A</td>
<td>Test B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:00</td>
<td>Test A</td>
<td></td>
<td></td>
<td>Test A</td>
</tr>
<tr>
<td>3:30</td>
<td>Test A</td>
<td></td>
<td>Test A</td>
<td></td>
</tr>
<tr>
<td>4:00</td>
<td>Test A</td>
<td>Test B</td>
<td></td>
<td>Test B</td>
</tr>
<tr>
<td>4:30</td>
<td>Test A</td>
<td></td>
<td>Test A</td>
<td></td>
</tr>
<tr>
<td>5:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:30</td>
<td>Test A</td>
<td>Test A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To compound the complexity, the exact times that will be booked are not set until the students actually reserve a time. In the case that two instructors wish to assign the same or overlapping time frames for their tests, it is a challenge to determine if there are enough resources to accommodate the two exams. As students schedule an exam, the available seats at various times changes but becomes more known. If the second instructor does not schedule their overlapping exam until all of the students for the first exam have made their reservations, the scheduling process is more or less straightforward. However, by that time, it could be too late to schedule another exam.

The goal for this system is to allow for maximum utilization of the testing center(s) while allowing for maximum utility to the students. Even if there are more seats in the given time span than there are students, it is only by requiring that students pre-register that all can be guaranteed a seat. What we are dealing with in this case is students’ propensity to wait to the last minute to take an exam. The longer a student waits, the more likely they will have to take their test at the start of the time frame rather than at the end. When all of the students have designated a time to take an exam, there are still uncertainties that are likely to occur. For instance, some students may show up late to the exam. It would be helpful to provide students with reminders to avoid this, but some students cannot be helped with punctuality. The system should be flexible enough to allow for accommodation of such situations, i.e. to take the test late as long as a seat is available through that new test time. Allowing students to take the test late and for the full length of time can’t always be done with limited resources, but there are likely to be occasions when the time following the late student’s reservation is still open. If that is not the case,
and the student is still in their reserved (but late) time frame. Then the student should be
warned that they have less than the assigned time to take the test, and given the choice to
start the exam or find another time to take it.

The system should also be sufficiently robust to handle events such as changes in
testing center open hours. It is possible for a crisis to arise such as a crashed server,
forcing the lab to close unexpectedly. Closures are undesirable, but can happen and it is
important to plan for such worst case scenarios.

The uncertainties that can occur make this a challenging tool to develop. The
scope of this project was to develop a scheduling system that represented a first solution
to test center scheduling, but which was sufficiently configurable and extensible to allow
the system to be further developed if a better solution became evident.
CHAPTER III
DESIGN

The proposed testing center scheduler software will be an addition to the already functioning iNetTest system. This will allow for the testing center scheduler to use the already existing features of iNetTest to help organize the tests, classes, and various roles that the scheduler must consider. This chapter discusses the integration of the scheduler with the current iNetTest system, the specific designs of the four roles in the scheduler system, and an overview of the algorithm used to solve the problem of efficiently utilizing testing center resources.

Integration With iNetTest

The iNetTest system already has several of the components that will be needed for scheduling. First, there are already various roles that can be formed in the system. Students to be considered in the testing center will directly correlate to the student user type in the iNetTest system. Proctors, instructors, and administrators will all require administrator privileges in the system, but the privileges of each will be different. Proctor and instructor permissions are already set in the system and will only require appropriate additional permissions for scheduling a test. A new set of permissions will need to be defined for testing center administrators that will allow them to view, create and modify lab information. Features for creating permissions for different roles are already a part of the current iNetTest system.
Classes in the iNetTest system are referred to as groups. The groups and tests that already exist in the iNetTest system can be used for the testing center scheduler. Knowing the class size will help the system calculate the seating demands for testing centers. The current iNetTest software has all that is needed for delivering tests except the ability for students to schedule a specific test. We refer to this as a scheduled test. It will be necessary for the instructor to use a test created in iNetTest to be able to schedule a test for a lab. The current scheduling software leaves other types of tests to be used such as a Canvas test or a paper and pencil test are left to future work.

As previously discussed, the iNetTest system uses locks and unlocks to govern the time and place of a student taking a test. These unlocks will have an important function in the reservation process. To date, a test unlock could only be performed by an administrator. With this system, a student indirectly creates an unlock when they reserve a time to take a test in a specific testing center.

**Student Role**

The proposed system will be sensitive to the needs of students and thus help them through the scheduling process as much as possible. The following is a design description of the reservation process from the perspective of a student.

When an instructor schedules a test, the student will be notified by email and instructed that they should schedule a time and lab in which to take the test. Also, an entry will appear in the “To Do” list on the student’s page of iNetTest that will allow them to schedule the exam. Upon clicking the “To Do”, the software will provide the student with a list of available testing center labs. The student may traverse a cascaded
list to select the lab, the day, and then the time at which to make a reservation. The possible times in the list will account for the time limit of the test and will not display times that are not available for the entire duration of the test. After selecting a time, the student will be sent an email for their personal record detailing the reservation and the materials that the instructor allows during the test. The allowable materials list for a test is something which the instructor will have entered at the time they create or edit the test.

After a student has reserved a time and location to take a test, the system will create an unlock for the student at that time in the specified lab. The unlock will be limited to the IP range of the systems in the lab that the student selected and it’s duration will be the same as the time scheduled by the student’s reservation. If a student wishes to reschedule their test time, they may do so as long as the request meets the following three criteria:

1. The instructor will allow it.
2. The time that the instructor allotted for the test has not expired.
3. There are still open seats available during that allotted time.

If these criteria do not hold, then it is up to the student to work out the situation with the instructor. Upon rescheduling, the previous unlock is removed and a new unlock is created for the new time and/or lab.

It is likely that at some point a student will forget what time they signed up for a test. Students might even forget about the test entirely. To address this scenario, the student will be able to look up the scheduled time from the student side of iNetTest.
Also, when the student makes a reservation, they will have the option of receiving reminder emails and/or texts. They may also select a day-before and/or an hour-before reminder. The reminder will include the date, time, and location of the reservation.

When the time comes to take a test, the student will go to the lab, present his or her student id card, and sit in a proctor assigned seat. The time allotted for a test will account for a transition time of at least ten minutes. Tests can only be scheduled on the half-hour. Thus, a 50-minute test scheduled for 8:00 would occupy the time from 8:00-9:00 allowing for a 10 minute transition. A sixty minute test scheduled for 8:00 would occupy the time from 8:00-9:30 allowing for a 30 minute transition time.

If the student is so late that the time to take the test is more than the time allocated, the time limit for the test will be adjusted to end at the same time that the reservation ends. The student will be informed of this fact and given the option to reschedule or still take the test. Also, a proctor may assign extra time (enough for the full test time) before the student begins a test if there is sufficient room in the lab schedule and if the student accepts the offer of extra time. For example, suppose a student has a 50 minute test and arrives 20 minutes late. The time allocated for the test in this case would be 1 hour. 10 minutes are designated for transition time and so the student would only have 40 minutes to take the exam. If the testing center has available seats after the student’s original reservation, the proctor may assign the student to stay for another half hour segment in order to get the full 50 minute test time.
**Proctor Role**

The role of a proctor is defined by the administration of the testing center; however, the system defines a general procedure intended for all proctors. The expected interaction of the proctor with the student is as follows:

- The proctor will direct the lab procedures and will verify identification of the student.
- When a student comes in to take a test, the student must use his or her student id card to check in.
- Available to the proctor will be a list of students that have reserved that time at the testing center.
- Clicking on a student entry will reveal a list of the materials that the student is permitted to have for the test.
- The proctor will assign the student to a numbered computer station.
- In the unlikely event of a system failure or if the student unintentionally exits the test, the proctor will have permission to resume the test.
- If a student arrives late for an exam, the system will check to see if there is extra time past the original reservation to extend the test and allow the student the full amount of time for the test.
- If there is not extra time, the proctor will receive a message that the test time will be reduced.
It is possible that upon entry to the lab a student will not identify themselves to the proctor. To prevent this from occurring, the system will block all reservations, i.e. unlocks, until the student has been checked in. This will ensure that the proctor has the opportunity to verify the identity of all individuals taking a test.

Proctors will also be there to assist students that may have arrived at the wrong time to take the test. They will have access to a comprehensive list of reservations for the lab in a searchable table that will allow them to locate a student’s reservation quickly given either a name or a date. They will also have access to the current reservations in the lab and if time and space are available allow the student to take the test at that time.

**Instructor Role**

It is important to ensure that the process for the instructor is as simple and seamless as possible. If an instructor chooses to use the system, then the students that wish to complete the course have to use the system as well. If the instructor is not satisfied with the functionality of the system, then it is likely that the system will not get used. The following is the design of the system from the instructor’s perspective.

For an instructor to be able to schedule an exam, currently the system requires the existence of an iNetTest test. In this initial set-up of the scheduler, the ability to schedule Canvas tests or written tests is not included. In the same window that the test may be edited, there is a section of the layout for scheduling the test. The test can only be scheduled if it has a time limit and if it is a locked test. If this is not the case, the system will inform the instructor what must be done to the test so that it can be scheduled. Once these requirements are met, the instructor is allowed to schedule the test.
Upon clicking the schedule button the instructor is given a dialog defining all of the options for scheduling the exam. First and foremost in the dialog is the option for selecting the time frame that the test may be taken, i.e. days and times for taking a test. Next are the options for allowing a student to reschedule (as described for the student role), if needed, and the options for the instructor to receive an email when a student schedules a test. An instructor might wish to know which students have scheduled the test but if the class size is large, these emails could be overwhelming. The option is therefore defaulted to false.

There might be cases that arise in which a student requires an exception to the scheduled time. To account for these exceptions there will remain an option to open a test for the entire class, or for a single student. This would be the case for a student registered through the Disability Resource Center and thus to be given additional time to take a test.

Next in the dialog is a collapsible list of testing center locations. The list is broken down into regions and locations. Within a region such as a main campus, there might be several locations that provide testing center times. There also might be various distance sites in other regions (Regional Campuses and Distance Education facilities) that support testing centers. The instructor will need to set up separate groups, based on the number of students registered at each (RCDE) site and schedule the test separately for these groups.

The last section in the scheduling dialog requires input of a list of instructions or materials that the instructor wishes to permit the students to use for the test. Such a list
might include index or note cards, a specific calculator, a book, etc. The interface allows for any number of materials and is set up in a list format for the benefit of the proctor who will see it in a bulleted form. The list will be used to ensure that the test is properly proctored. This list will also be e-mailed to students to inform them of what will be allowed for the exam. Figure 1 shows the design of the dialog interface for scheduling tests.

![Dialog Interface for Scheduling a Test](image)

**Figure 1: Dialog Interface for Scheduling a Test**

After submitting the options for opening the test, the system will process the request to verify whether or not the testing center can house the test during the specified time. If there are not enough resources in the selected testing center labs and at the times requested, the system will return a message informing the instructor. More on the scheduling processing can be found in the scheduling section that follows.
If the instructor wishes to view a list of the students that have scheduled the test, it will be available under the list of unlocks that have been created for that test. Further specifics on the reservation including location can then be found on the student’s information page in the iNetTest system.

When difficulties arise with getting all of the students in the class scheduled for the exam, the instructor will have an option to extend the test duration. The instructor can find the scheduled exam in the same location that he or she opened the test and extend the exam for a specified number of days. This will open more time for the students to take the test. This situation might occur if all students wait until the last few hours of the scheduled test to make a reservation. If this does not work, the standard features of the iNetTest system can allow the instructor to give the test on his or her own terms outside of the testing centers as was previously common in the system.

Since the instructor might need to reschedule a test, the system must allow this to happen. When a test is rescheduled, iNetTest will check to see if any of the students have already reserved a time. If so, the software will warn the instructor how many students will be affected by the rescheduling. As a result of the complexities of scheduling, the system must remove all current reservations for the rescheduled test. If the instructor proceeds to reschedule, then the affected students will receive an email notifying them of the changed time and that they must sign up for a new time. The unlock and the reservation belonging to the scheduled students will also be removed.
Administrator Role

The testing center administrator is the governing official over the procedures in the testing center. The administrator will have a high level of access to the system and will be able to manipulate the various parts of the testing center system.

The administrator will have the ability to execute any of the tasks the instructor or proctor can perform with additional capabilities. The administrator will have access to the schedules for all regions and labs in the system. The administrator will have the ability to add, edit, and remove both regions and labs from the database. They will also be able to modify the privileges associated with all roles for testing center personnel, i.e. Admin, Instructor, Proctor, and Student.

As an administrator, an individual will be able to assign the resources that are available in the lab. They will also be able to create and edit lab hours. If the administrator deletes a lab time that is scheduled by a student, the administrator will be given a confirmation warning with the number of students affected. As with an instructor changing a scheduled test time, completing the deletion will result in sending emails to the affected students informing them of the changes in lab hours.

An administrator can also remove a student reservation or a scheduled test. Searchable tables will give a comprehensive list of the reservations and scheduled tests for a lab, even ones that have already past. The administrator can choose to delete any or all items as circumstance may require. Corresponding emails are sent to the affected individuals.
Scheduling

The greatest complexity in the testing center software is in the scheduling procedures. Since the testing center locations will have a limited number of seats for students, the system needs to prevent multiple instructors from allotting the same time before the schedule actually starts filling up with student reservations. This can become a problem when an administrator edits lab times and resource counts as well. Actions that will affect the number of time slots available for student scheduling will inform administrators, instructors, and students appropriately.

Throughout the semester, testing frequency will vary. As such, there will be times in which multiple instructors will want to allot the same time frame or overlapping time frames for tests. Peak hours will especially be challenging for students to be able to schedule. It is not likely that every seat will be booked during every hour, but it is likely that during peak hours, the testing center will fill to capacity. The system will require procedures to ensure that all students will have a chance to reserve a test time. The system will provide a modifiable “uncertainty” value that will be multiplied against the test times that do not have a reservation. This will be used to determine if there are sufficient remaining seats to allow for another instructor to allocate an overlapping time for test taking. For simplicity, this threshold will be the same over all of the labs. Over time, and with more empirical data, the uncertainty value can be improved in order to maximize the use of the lab(s) and avoid situations in which a student cannot schedule a test. While a more sophisticated means of determining this parameter might be needed, the current system will simply provide this value as a constant, the value 3. Along with
the restriction that all students must make a reservation before the scheduled test, it can be proven that the value 3 will guarantee that all of the students will have a time span available to schedule in the worst case scenario. Given that a test requires $n$ time segments, the worst case scheduling would happen if a student schedules a reservation so that there are $n - 1$ time segments before and after the test. Such a scheduling would effectively occupy $n + 2(n - 1)$ segments of time. This can be reduced to $3n - 2$ and it is easy to infer that $3n - 2 < 3n$ always holds true. Therefore with an uncertainty value of 3, we guarantee that every student will have at least one time frame available to them.

The proposed algorithm assumes that students will reserve tests in an even distribution. This assumption is not made without supporting data. At West Virginia University, a psychology experiment was conducted analyzing the behavior of students regarding quizzes. [2] The instructor would open a quiz for an entire week and it was up to the students to decide when, during that week, they would take the quiz. The data from the experiment demonstrated that the closer to the due line of the quiz, the greater the frequency of students taking the test. It was far from a uniform distribution; however, this same experiment was performed on five different sections of the same class. The results from each class were the same but the due dates of the quizzes for each class were on separate days of the week. When the classes were considered as a whole, the resulting distribution turned out to be uniform. (See Figure 2) In this case, the assumption is not that individual tests will be uniformly distributed, but when considering the entire population of students, there will be an overall uniform distribution.
Figure 2: West Virginia University Psychology Experiment Results
CHAPTER IV
IMPLEMENTATION

This chapter focuses on the most complex portions of the integration of the testing center scheduling software into the iNetTest software. The discussion of the implementation will be split into three sections. The first section is a comprehensive description of how the iNetTest database is enhanced to store the information needed for the testing center. The second section discusses the implementation of the interface. Lastly, there is a section that includes a detailed report on the business logic and algorithms implemented to link the front and back ends of the application.

Database

Only minimal database changes to iNetTest were required to accommodate the test scheduling software. In total, seven tables were added in order to provide for the testing center scheduling functionality. The purpose of each table is described in this section. Figure 3 contains an entity relationship (ER) diagram of the testing center tables.¹

The most complex table in the new functionality is the allotment table. This table contains the time frame that the instructor allows for a test and records various options for the test. The record in the table is related to a test in the system. While the test time limit is stored in the test table, the reservation length is recorded here in the allotment table.Normally it is good practice to avoid derived traits. The purpose of storing the reservation length, derivable from the test time limit, is for computational

¹ Diagram was generated using DbVisualizer available at http://www.dbvis.com/
speed. This duplication of data avoids a duplication of work in a computationally heavy scheduling algorithm. An allotment record also specifies the id of the instructor who scheduled the test and relates it to the class or student for which it was opened. An allotment is tied to a region which shares the primary key. The reason the allotment record was implemented in this form was to be able to account for distance sites in the future as well as have a common allotment id for all the regions related to the allotment.

The lab table contains the information needed to represent the testing center. The name of the lab, the description, the number of testing stations, the region to which it belongs, and the IP range of the lab computers are all specified in this table. The number
of stations is important for the resource count of the lab. The IP range is necessary in order to restrict each reservation’s unlock. The lab time table, i.e. hours available, is also associated with the lab. This is a simple table that contains a timestamp for each time period or half hour block that a lab is open for scheduling. The lab id is paired with the timestamp.

The allotment_lab table describes the “many to many” relationship between an allotment and a lab. This relationship includes one attribute, the allotment percent. This value is used to estimate how many of the students that have not yet reserved a time will go to the associated lab assuming an even distribution based on the number of testing stations in the lab. For example, if a test is set up for 10 students who can schedule a reservation in a testing center A with 40 seats and a testing center B with 60 seats, the calculations for distributing the students will assume 4 \((40/100*10)\) students got to lab A and 6 students go to lab B. After 5 of the students reserve a time, the scheduling will assume that 2 students will go to lab A and 3 students will go to lab B. This calculation is done independent of the lab choices of the scheduled students. In other words, if all of the reservations were made in only one of the labs, the distribution of the remaining students will stay the same.

The reservation table stores the data for the reservations made by students. The table records the date, length, and place of the reservation along with a reference to the student to whom the reservation belongs. It also stores the reminder options that the student selected upon creating the reservation. When the student is admitted to a testing center and assigned a seat, this information is stored along with the reservation details.
Any reservation in the system must correlate to a test allotted by an instructor. A student can only have one reservation for a test. For this reason, the reservation primary key includes the allotment and user ids. Finally, the unlock that is created when the reservation is made is recorded here so that if the reservation is canceled, the corresponding unlock can be removed as well.

The final two tables in the system are simple ones. The region table stores a name and description of the region. Each region may contain one or more labs. Allotted time for a test is also associated with a region to handle classes with distance education students. The test materials table simply stores the instructors’ allowed list of materials for a test. This is associated with each opened test time with any materials specified.
The interface needed to provide for the testing center capability includes four components. Each component of the interface corresponds to the personnel roles involved with the testing center; namely, student, instructor, proctor, and administrator. The interfaces for each are described in this section.

The additions to the student interface are minimal. After the student receives an email informing him or her that a test has been opened for scheduling reservations, the student will receive a highlighted button under the “To Do” list as in Figure 4. Clicking the button will take them to the screen as shown in Figure 5. For each day for a given
lab, a list of test times is shown from which the student can make their selection. If there are no open test times in a lab, the interface will inform the student of this fact.

![Student View for Reserving a Time](image)

**Figure 5: Student View for Reserving a Time**

The interface that the instructor will see when scheduling a test is part of the test information window. There is a list of recent openings of the test under the scheduled test section of the window. The list displays those times that have been opened within the previous week or are due to open any time in the future. This list gives the instructor the icons to both extend the time for the test or to remove the scheduled test entirely (See
Figure 6). At the bottom of the scheduled test section is a button to schedule the tests for students. This will open the dialog that is detailed in the design section for the instructor.

![Image](image_url)

**Figure 6: List of Currently Scheduled Tests**

The proctor interface and the administrator interface are the same with the exception that the administrator has access to edit the lab information (See Figure 7).

The main feature of the lab page is the lab schedule. The lab schedule section has two modes, view and edit. From the view mode, the table cells display the lab’s schedule for the selected week. There is a date picker at the bottom allowing the user to easily select any desired date. The times that are shown can be adjusted to any widow of time in a 24 hour clock. A block of cells is blank if the lab is closed during that block of time. If the lab is open, a fraction is displayed showing how many of the lab seats in that time are still available. The fraction is displayed as a link so that proctors can select that time block to view the students that have reserved seats in the lab (See Figure 8). In edit mode, the
administrator is still given this fraction, but there is an additional icon to allow the administrator to remove that open lab hour (See Figure 9). The administrator can also add time by selecting empty table cells. There is an option on the scheduling tool to repeat the added time until a selected date. This allows the administrator to quickly set up the lab schedule for an entire semester. Anomalies that occur in the semester such as holidays are left to be removed from the added times individually. There is not a repeat option for the removal or deletion of times to prevent accidental deletions.

The other features that are included on the lab page for the proctor and administrator include the general lab information and comprehensive lists of reservations.
and scheduled tests. The lists of reservations and scheduled tests have a searchable table tool that is used throughout the iNetTest system. These tables allow not only searches, but an administrator may select individual reservations or scheduled tests to delete. This feature is not expected to be used frequently, but it allows the flexibility to the administrator if the occasion requires such action.

Figure 8: Student Reservations during a Selected Time
Figure 9: Edit Schedule View

Scheduling Algorithm

The scheduling algorithm is a key feature of the testing center scheduling system. The effectiveness of the algorithm significantly impacts the usefulness of the system. One can expect the algorithm to be improved after further research and collection of data. Such data will not be available until the software is added to the iNetTest system and actual scheduling is begun.
In order to schedule a test in the system, the algorithm must be passed the allotment options given by the instructor and the database access object. First the options must be validated. False is returned if the validation fails. Invalid options might include an end date that is before the start date, dates that have already passed, or a missing testing center selection. If the allotment is valid, the algorithm must check the availability of each lab selected by the instructor.

The methods that govern test scheduling will be mainly based on the assumption of a uniform distribution of students across labs and across blocks of time. For example, consider two labs A with 30 seats and B with 60 seats. When an instructor with 90 students wishes to schedule an exam in these two labs, the system will assume that 30 of the students will go to lab A and 60 will go to lab B. After dividing the students in the class, iNetTest will execute the following procedure for both labs:

1. A list of all of the available times the lab is open in the interval for the new test will be created.
2. Any student reservations currently assigned to the lab in the interval will remove availability (seats) from the list.
3. For each test associated with the lab, students who have not scheduled a time for the test will be distributed uniformly over the time frame according to the number of students in the class multiplied by the uncertainty constant.
4. The number of students to be scheduled is multiplied by the uncertainty value and the resulting tests are scheduled in the remaining time slots assuming an even distribution.
5. If all of the tests are successfully scheduled, then the new test is finally considered.

The uncertainty constant and methods used to schedule tests are separate modules from the main algorithm so that future modifications to the use of the uncertainty constant can more easily be made.

<table>
<thead>
<tr>
<th>Time</th>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
<th>Step 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:30</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2:00</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2:30</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3:00</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3:30</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4:00</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4:30</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5:00</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5:30</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6:00</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6:30</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7:00</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7:30</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>8:00</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8:30</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

To help clarify the steps of the scheduling algorithm, consider the scheduling of a testing center with 6 seats as given in Table 2. In step 1 (column 1 of the table), a list of 6 seats during each time segment is created. In step 2 of the table, a set of current reservations has been accounted for. Suppose that for a previously scheduled 50 minute test there are 5 students who have not yet made a reservation. In step 3 with an
uncertainty factor of 3, the algorithm would distribute $5 \times 3 = 15$ tests, one test in each one hour time segment. Now suppose that an instructor with 5 students in his class wishes to schedule this time frame for a 30 minute test. Considering the buffer time previously mentioned, this test will actually schedule for an hour in the system. Since the students do not yet have a reservation, the uncertainty value is used in step 4 to determine if the instructor can schedule the test which would require 15 one hour segments in the schedule. After distributing 10 tests, there are no more available hour segments of time. (See Step 4 in Table 2) Step 5 would then reject the instructor’s request informing the instructor that there is not enough space (time slots) to schedule the exam.

A restriction that is used for step 3 of the process is that tests in the past are irrelevant to the calculation. It is necessary, however, to consider all of the current and future scheduled tests in the system because a new addition has the potential to have a cascading effect on all of the overlapping tests.

```
function SCHEDULE_TEST(allotment, dao) returns true if the test can be scheduled
    inputs: allotment: the scheduled test options
            dao: the database access object
    canSchedule ← CHECK_OPTIONS(opts)
    if not canSchedule return false
    for each lab in allotment
        start ← dao.getSoonestAllotmentStart(lab, allotment)
        end ← dao.getLastestAllotmentEnd(lab, allotment)
        times ← dao.getAvailableLabTimes(lab, start, end)
        for each dbAllotment in dao.getAllotments()
            DISTRIBUTED_ALLOTMENTS(times, start, end, dbAllotment)
        n ← DISTRIBUTED_ALLOTMENTS(times, start, end, allotment)
        if n > 0 return false
    dao.create(allotment)
    return true
```

Figure 10: Pseudo Code for Scheduling a Test
The process of distributing the test times evenly is one that must be considered carefully. Suppose that 7 one hour tests must be distributed in the lab with available times given in column 1 of Table 3. In these available times there are 10 spots highlighted in which a one hour test can be placed. The uncertainty constant of 3 is multiplied by the number of tests to give 21 one hour segments. Integer division is used to determine that we will assign 2 tests to each available time with 1 left over. The left over test time is recorded as overflow. When distributing the 2 tests per available time, note that after assigning a single test to the 1:30 time, 1:30 and 2:00 are no longer available. This causes 3 test times to not be scheduled and they are added to the overflow values. After the initial distribution is completed, there are 5 test times left in the overflow that are distributed randomly over the remaining open times highlighted in column 2 using a uniform distribution. After attempting to distribute all of the tests in the

<table>
<thead>
<tr>
<th>Time</th>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:30</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2:00</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2:30</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3:00</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3:30</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4:00</td>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4:30</td>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5:00</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5:30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6:00</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6:30</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7:00</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7:30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8:00</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8:30</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
example, there is 1 test time remaining signifying that there is not enough space to schedule the test. Figure 11 gives the detailed pseudo code of this distribution process.

```
function DISTIBUTE_ALLOTMENTS(times, start, end, allotment)
returns:  the number of students that were not able to be scheduled
inputs:  times: the available times for a lab
       start: the time and date of the start of the times list
       end: the time and date of the start of the times list
       allotment: the allotment to distribute into the times list

n ← allotment.getNumberOfStudents()

n ← CEILING(n * GET_UNCERTAINTY(n))
c ← CEILING((allotment.getTestTime() + BUFFER) / INCREMENT_LENGTH
openList ← empty

for each t in times that is during allotment
    if t is available add t to openList
if openList is empty return n

seatsPerincr ← n / openList.length
overflow ← n % openList.length
if seatsPerincr > 0
    for each t in openList
        count ← 0
        available ← IS_SEAT_AVAILABLE(times, c, t)
        while available and count < seatsPerincr
            available ← MARK_RESERVED(times, c, t)
            Increment count
            if not available
                overflow ← overflow + seatsPerincr – count
                remove t from openList
        while overflow > 0 and openList is not empty
            random ← uniformly distributed random number < openList.length
            if IS_SEAT_AVAILABLE(times, c, openList[random])
                available ← MARK_RESERVED(times, c, openList[random])
            if not available
                remove openList[random] from openList
            decrement overflow
            else remove openList[random] from openList

return overflow
```

Figure 11: Pseudo Code for Distributing Unknown Reservations
The complexity of the algorithm is polynomial. There are several cascaded loops in the algorithm that account for the complexity. First the outermost loop in the algorithm traverses the various labs $\lambda$ that were selected by the instructor. The next loop goes through all of the future allotments $\alpha$ for that lab. Each allotment is distributed into the open times $\tau$ of the lab during that allotment. The value for $\tau$ is determined by examining the entire time $T$ that future allotments exist. There are a couple of scenarios that can occur next in the calculation. Either the number of students $n$ is greater than the number of open time slots, or the number of open time slots is greater than the number of students. First consider the situation where $n \geq \tau$. For this case, the number of students must be distributed into the times slots evenly at $\left\lfloor \frac{n}{\tau} \right\rfloor$ per time slot. Then to distribute a test into a time slot we must cycle through the number of time slots $c$ that the test covers. Then, overflow can be distributed which in worst case would be on the order of $\tau - 1$.

This yields the complexity to be $O\left(\lambda \alpha \left( T + \tau \left\lfloor \frac{n}{\tau} \right\rfloor c + (\tau - 1)c \right) \right)$. Assuming that $\tau \left\lfloor \frac{n}{\tau} \right\rfloor \approx n$ and the fact that $n \geq \tau$ the complexity can be reduced to $O\left(\lambda \alpha (T + nc) \right)$. Based on this second scenario, it is evident that it is only necessary to distribute $n$ tests randomly at a uniform distribution and therefore resulting in the same expression. In practice the number of labs in a region and the number of time slots for a test are likely to both be small, thus the complexity is more accurately to be represented as $O\left(\alpha (T + n) \right)$. This calculation for both methods makes logical sense because in the end, the algorithm is simply distributing $n$ tests in a restricted timespan within $T$ time periods. Thus, the
proposed algorithm can be executed in polynomial time with a complexity similar to a second order polynomial.

**Results**

In order to verify the functionality of the allotment calculations, the example given in Table 3 was used as a test case. The reservations that would yield the available times given in column 1 were setup. Two tests were then scheduled, one class with 5 students and another with 7. The class with 5 students scheduled without any problems, as was expected. When attempting to schedule the second class of 7, the test was rejected. As expected, the 16 of the 21 projected reservations could not be scheduled. Figure 12 shows the output when attempting to schedule the test.

![Figure 12](image-url)

Figure 12: Scenario Output
CHAPTER V

FURTHER WORK

As previously mentioned, with empirical data, improvements can likely be made to the algorithm. Unfortunately, there is currently no such empirical data to show the applicability of the proposed method. There are also adjustments that might be made to more accurately determine if an instructor may schedule an exam. This section describes how future data can be used, possible adjustments to the algorithm, and the design for future testing of the system.

Data Use

Once the scheduling algorithm is placed in use, a careful analysis of the resulting data could assist in several ways. The data might reveal that the evenly distributed assumptions are not appropriate in practice. In such a case, it might be advantageous to distribute the students in a polynomial or exponential distribution over the allowed time frame. With this data one could certainly determine the distribution which best fits. Unfortunately, it might also be the case that tests given at different times in the semester invoke a different distribution of student test scheduling. The data could also be used to determine how many tests are being extended. A survey of student test takers could also be used to determine how well the system is meeting student needs. These data might also assist in refining the uncertainty constant and reveal that a constant is not sufficient.
Some heuristic that considers the class size and the size of the time frame might be developed to better handle the uncertainty.

**Adjustments**

There might be several small adjustments to the algorithm that could improve its performance. One adjustment might be that students will prefer a particular lab. If a popular lab is absolutely full and an unknown lab empty then the algorithm will deny an allotment including both labs, though it may be the case that the empty lab can house the entire class. Also, an adjustment might be made to take the overflow of one lab and place it in another.

Another adjustment that could be made is to consider the time of day. It is reasonable that a student is more likely to reserve an exam time in the late afternoon than in the early morning hours when the testing center first opens. This factor is neglected in the current procedure. It is reasonable to assume that if the peak hours are booked, a student is likely to make an early morning sacrifice in order to prevent failure in a course.

The algorithm always uses all of the future tests. This isn’t necessary because not all tests will have overlapping time frames. Considering only those tests that overlap with new tests to schedule might result in better performance; however, calculating which tests overlap might add unnecessary complexity to the algorithm.

**Future Testing**

After data is collected, there must be a way to test the effectiveness of the algorithm. Here we discuss two tests that can be performed to assist in refining the
scheduling procedures. The first test is designed to determine how many scheduled tests can be handled by the system. The second test is designed to refine the uncertainty value.

As the number of scheduled tests for a testing center rises, so does the computation time. It is vital to know how many scheduled tests the system can handle to ensure that the software can be used at a large scale. The procedure for this test is as follows:

1. Create a testing center that is always open for several years
2. Run a script that will randomly schedule tests of various lengths during the time frame.
3. Run the script until the response time reaches a threshold or until several requests in a row are returned that deny the scheduled time.

The parameters of the test that can be adjusted include the number of years the testing center is open, the number of seats in the testing center, the number of students in the class, the length of the test, the response time out threshold, and the scheduled test failure threshold.

The next test will be used to refine the uncertainty value. As proven previously the value 3 will guarantee that the students will have an available time to schedule the test. This guarantee comes at the cost of lower utilization of the computer resources at the testing center. It is desirable to experiment with other values or even replace the value with a function in order to better utilize the testing center resources. The test will be a simulation that will model two behaviors of the students. The first being the rates at
which the students reserve a time after the instructor has scheduled the test. The second
is the distribution of the students over the time frame of the scheduled tests. Using the
data, we can simulate the behavior of the students and experiment with various
uncertainty constants and functions. The simulation can record how many students do
not have an available time for a reservation. The results will be used to define a better
policy for dealing with the uncertainty.
CHAPTER VI

CONCLUSION

The iNetTest system with its various features makes it easy to integrate a new feature for managing the scheduling of a computer-based testing center. The testing center management software can be used to help coordinate the opening, reserving, and administration of tests between instructors, students, and test proctors. The scheduling algorithm presents some uncertainty in terms of its accuracy. Much of this uncertainty could be reduced with empirical data; however, in order to collect such data, the lab must go into use with a relatively large number of users. The proposed scheduling algorithm presents a first step solution that can provide a starting point until further data can be gathered. The algorithm itself has been written so that significant changes to its logic and parameterization are relatively easy to make. The scheduling software also provides for a seamless integration with the iNetTest system and a useful, easy to use tool for students, proctors, instructors, and administrators.
REFERENCES
