Evaluation Report

Physics Lab Booking Project

Kartik Gupta
Puravin Sivaganam
Christopher Ballouz
Julio Deak Harada
Mohammad Heydari
Albert Christopher Hyde

Client: Dr Xue (Sue) Yang

Gentlemen, we are going to relentlessly chase perfection, knowing full well we will not catch it, because nothing is perfect. But we are going to relentlessly chase it, because in the process we will catch excellence. I am not remotely interested in just being good.

-Vince Lombardi

Table of Contents

1 INTRODUCTION

1.1 ABSTRACT

This technical document is an evaluation plan for Sprint 1 for the project: Web-based Booking System for Senior Physics Laboratory. The purpose of this evaluation plan is to reflect commitment to the Extreme Programming (XP) methodology. This document will present our initial testing plan. The plan includes unit testing, acceptance testing and usability testing. It will also detail the client's interaction with our team with the involvement they have had in the process of defining appropriate tests.

1.2 EXECUTIVE SUMMARY

The project our group has been assigned is, "Web-based Booking System for Senior Physics Laboratory". The purpose of this project is to carry out an evaluation on booking system to ascertain requirements and help resolve the bugs to make it a viable product.

To verify that the existing system functions as expected, we conduct 3 levels of software testing, unit tests, integration tests and acceptance tests. First, we test certain functions and areas with unit testing. This method allows checking whether the function is returning the proper output with given set of inputs. Normally the set of inputs are declared before going into implementation so the code won't be biased to be operating in specific way. After the unit testing, we move to integration test to expose faults in the interaction between integrated units. Finally, acceptance test takes place determine if the requirements of the function is met.

Since the system is coded with Ruby on Rails, we will use RSpec as our testing framework. RSpec is testing tool for the Ruby programming language. RSpec supports Behaviour-driven development, which enables to write the tests for rails models clean and human readable.

2 CORE

2.1 USER STORIES

The 6 user stories we have planned:

1. As a student, I want to book experiments based on my credit point requirements.
2. As a student, I can only make one experiment session per day
3. As a student, I want to be shown when days are fully booked
4. As a student, I want to be shown a confirmation when making a booking
5. As a student, I want to be able to cancel a booking a certain number (X) days before the booked date
6. As an admin, I want to be able to add students with their personal details

2.2 TEST FRAMEWORK

RSpec is testing tool for the Ruby programming language. Born under the banner of Behaviour-Driven Development, it is designed to make Test-Driven Development a productive and enjoyable experience with features like:

- a rich command line program (the rspec command)
- textual descriptions of examples and groups (rspec-core)
- flexible and customizable reporting
- extensible expectation language (rspec-expectations)
- built-in mocking/stubbing framework (rspec-mocks)

We use Rspec with extra gems, like *Capybara,* which allows us to simulate a user's interaction with the sample application using a natural English-like syntax, together with *Selenium*, one of Capybara's dependencies.

To write unit tests, the entire group gets together and has a brain storming session. At this point tests are written in simple English. They are later translated to Rspec before coding can begin. Rspec makes it easier to translate tests written in simple English to Ruby. An example of a Rspec test is:

```
describe MovieList do
        context "when first created" do
                it "is empty" do
                        movie_list = MovieList.new
                        movie_list.should be_empty
```

```
        end
    end
end
```

The it() method creates an example of the behavior of a MovieList, with the context being that the MovieList was just created. The expression movie_list.should be_empty is self-explanatory.

Running this code in a shell with the rspec command yields the following specification:

MovieList when first created is empty

Add some more contexts and examples, and the resulting output looks even more like a specification for a MovieList object.

MovieList when first created is empty

MovieList with 1 item is not empty includes that item

This example is taken from the *RSpec Book, Behaviour Driven Development with RSpec, Cucumber and Friends* by David Chelimsky (et. all), Oreilly Publishing, 2014

2.3 ACCEPTANCE TESTS

Acceptance tests are created from user stories which are selected during each iteration of planning. They are then given and shown to the customer/client to ensure if the user story has been correctly implemented. Each test represents a result from the system in which the customer/client is responsible for verifying the correctedness of the acceptance test.

These user stories where then transformed into acceptance tests. With these acceptance tests, our Client Liaison Puravin Sivaganam contacted and emailed our client Dr. Sue about whether or not she wanted these tests to be implemented into the system or whether they were on track to what she hoped to be in the system. The full table is shown in the appendix(2.1.2).

2.4 CUSTOMER INTERACTIONS

The communication methods that we use to interact with our customer/client are emails and face-to-face meetings. The emails are extensive and could be viewed via our bit-bucket site under 'Client Communication'. The face-to-face meetings however are brief and straight to the point.

Our client, Dr. Xue (Sue) Yang, works with the Scientific Computational Office in the School of Physics here at Sydney University. Dr. Xue is looking on improving the current physics booking system that the School of Physics uses for booking experiments.

The project began two years ago when Dr. Xue worked with a group of COMP3615 students to develop the booking system. The current system was developed using Ruby on Rails and our client has requested us to maintain the current programming standard.

Dr. Sue has given us a document form of all the priority list of requirements that we deconstructed into a tabular form (2.1.3), including experiment data which we can use for testing the web based scheduler (2.1.4).

Below are the client meeting notes that were made from our meeting with the client during Week 2 and Week 4 respectively.

Week 2

Tuesday 5/8/2014 1:00 PM to 2:00 PM

```
Every team member present and Client
```

Meeting Notes

- client has the source code , which she shown to us. Experiment are set by the staff in the beginning of each semester
- client gave us the description of the system they need and they expect us to deliver that
- website is not under use by any users at the moments. They use another website
- admin, tutor, students are the roles.
- Log on page are the same for everybody
- some problems with database schema: there are some non required tables, more refinements on the database..probably design new database from scratch the document is based on previous project
- there is no usage for booking status table, for example
- one of the field in the table when it gets updated, it doesn't get reflected on the system...semester_start
- they want to deploy on Linux server
- they don't want students to book first few weeks of the semester. they want to spread the load of student booking across the semester. Like if a student has 5 experiments they need to be spread across the semester
- each experiments need 2 sessions
- lab opening days and running week across semester are defined by the admin. Admin makes changes at the beginning of the semester, and one during the semester.
- currently, all experiments are available on all days of the week, they want to have the experiments to be available in different weeks.
- for example, experiment1 is available on Monday, and Tuesday , and experiment2 are available on next 2 days.
- Experiments can be available on certain weeks too.
- currently it's been developed using Rails 4, ruby 1.9.3
- client gave us the source code, project specification and description and user document.

Week 3

No meeting scheduled

Week 4

Tuesday 19/8/2014 2:00 PM to 3:00PM

```
Puravin and Client
```

Meeting Minutes

1. Discussed Progress of the Team
2. Discussed the issues related to the system
3. Clarified certain requirements that required further understanding of the client's expecations
4. Client suggested issues in the database
5. Client suggested to email a copy of her schema to the team
6. Client showed a refiend document with the requirements and detailed explanations
7. Suggested to client that it would be useful to have the requirements listed with their respective priority
8. Client agreed to email the document
9. Cleared up the expectations of the team and explained how the tasks will be carried out
10. Proposed to the client the option of being part of the bitbucket page
11. Agreed to add the client to the bitbucket page

Meeting Notes

- Booking Experiment: Time (defined to be 2pm to 6pm) of day is not required, just selecting the day only
- Booking Experiment Bug: Admin selects the tool to Book an Experiment for a student
  - Calender does not provide the checkbox necessary to select the days to be booked
  - Problem selecting availability of experiments
- Booking Availability: Add functionality to allow Admin to select the days for each experiment (Flexible Dates)
  - Experiments should be listed on the calendars page, next to the calender itself
    - Admin is able to select the experiment then choose the specific days that it will be available on the calender
- Database:
  - Uneccessary attribute -> details_id (table: users)
  - Uneccessary tables -> booking_stats, experiments_availability

Wednesday 13/8/2014 11:17 AM

```
Client to Puravin
```

Hi Puravin and others,

Good to hear from you. Ok, we won't meet this week. However I would like to stress these points for you to do before the first demos in week 5,

1. Testing every function available in the system I gave you. Identify bugs (I know there are a few) . fixing these bugs will also be part of the project you are doing.
2. review the database schema in the system. Propose changes which would make it more effective and which are required for the new functionality, which was highlighted in read in the document, system specifications.

I would like you to show me the work as mentioned above before the first demos. With these work done, you will have sufficient contents to say at the first demonstration.

The attached file is mhtml document.

Regards,

Dr. Xue (Sue) Yang Scientific Computational Officer School of Physics, University of Sydney Ph: 02 9351 6081 Email: xue.yang@sydney.edu.au

In the email above, Dr Sue highlights the need for us to review the system thoroughly and extensively to test and identify any bugs that the system currently has. With this in hand we as a group were able to send back a list of all the new bugs which we found as well as the features that she wanted to be implemented to the system.

In the email below, Dr Sue attached the acceptance tests and was pleased with the development we were making with a few tips and instructions.

Hi Puravin and others,

Thanks. Glad to see you are getting on track.

I suggest that you take this semester as an example for experiment calendar setup (see the attachment: calendar.docx) before doing the tasks.
Current system uses the calendar that all experiments are available on Mondays, Tuesdays and Fridays except semester break. Lab days open for experiments will be blue as you can see.

Each student's bookings must be evenly across the semester (rules should be defined). For example, student A is not allowed to book all his sessions in weeks of first half semester.

I also suggest that you add all experiments (there are over 30 experiments) I gave you in the database before doing the tasks.

I strongly suggest that if anything about the system/tasks is unclear to you, please come to my office and I would explain it to you face to face with system demonstration.

Regards,

Sue

2.5 GROUP PROCESSES

The major roles within the scope of the project are:

1. Manager and Tracker
2. Integration Tester
3. Usability Tester
4. Client Liaison
5. eXtreme Programming Expert
6. Mercurial Expert
7. Bitbucket Expert
8. Programmers

Keeping within the boundaries of the Agile framework we will follow the reverse pyramid structure with the Programmers and Testers at the top and the Manager at the bottom representing a style of management where the manager supports the team. This way the team is able to choose tasks freely knowing that the manager will deal with any roadblocks.

The programming methodology being followed is Extreme Programming, which has a great emphasis on testing. The group processes have also been set up to reflect the same. Before, any programming begins; the group gets together and thinks about **unit tests** for each of the user stories. This session of brain storming helps identify any cases that might be missed by an individual. It also provides an estimate of how complex a user story is from start to finish which helps during **The Planning Game.**

During the Planning Game, Issues are created for each user story on Bitbucket. Each team member is free to choose the user story they wish to work on in that sprint. They then assign, on a scale of 0 to 2, a level of complexity, priority and difficulty, to the user story. The unit tests are first converted to **Rspec** and then the user stories are then developed through **Test Driven Development**. Alongside development, the programmer also comes up with **acceptance tests**, which he/she thinks satisfies the need of the client. Alongside, the **Client Liaison** these acceptance tests are passed to the client for approval.

After passing all the unit tests, the **Integration Tester** checks the compatibility of the user story developed with existing code. A method of **Continuous Integration** is followed where the code before the development of the new user story and after development is checked.

This is then passed to the **Usability Tester** to check the view and single out any anomalies that may arise from the use of the functionality. Both the "happy case" and extreme case is checked for how the user may navigate through the system. Use cases diagrams are created and need to be approved by the client if they differ from the acceptance test.

Finally, before resolving the issue on Bitbucket the acceptance test criteria is checked and if it is met, only then the issue is marked as resolved.

2.5.1 The Good and The Bad

This section identifies the strengths and weaknesses of the group from the standpoint of XP and the Big 5 terms. These characteristics have been developed based on the observations from the first sprint.

Manager and Tracker – Being responsible for supporting the group, keeping tasks on time and being agile are all part of the successes of the manger. Meeting agendas are made on time and are uploaded to the Bitbucket wiki page for everyone to see beforehand.

Trusting other group members to complete the task on their own is where the tracker and manager can improve.

Integration Tester - Responsible for ensuring various modules (user stories) work well together after being developed. This comes after unit testing but before acceptance testing. The Integration Tester is also responsible for developing the rspec framework that we will be using to do TDD as well as to train fellow team mates for the same.

Areas for improvement include, mutual performance monitoring, so that he can adjust along with the demand and supply of the work being completed. Sometimes the tester will have a lot to test in one sprint and maybe in one he will have less. He also needs to set the testing standard and set up continuous testing to counter this.

Usability Tester – Responsible for evaluating product by testing the User Interface. He is also involved in the prototyping phase in case new view is being created. A survey after a user story has finished and is ready to be marked resolved a survey should be done of the new functionality.

The usability tester needs to be adaptable, have back up behaviour in case of quick changes and needs to have continuous testing.

Client Liaison – is adaptive, on time with client communication and up to date with the happenings of the group.

In case of a short notice the client liaison needs to be able to act quick. Hence needs to have back up behaviour and avoid falling into the vicious cycle of closed loop communication.

Experts – Great with technical jargon and can simplify it for group members. He sets procedures for the rest of the group to follow and make the most of the technologies and tools we are allowed to use such as Bitbucket and mercurial. The experts need to be adaptable and roles need to be assigned to different people (which is not the case right now). If the roles are diversified this will create a system of checks and balances between the team as well as between the tracker and manager.

3 APPENDIX

3.1 UNIT TESTS

3.1.1 Confirmation Dialogue Spec

```ruby
require File.expand_path("../spec_helper", __FILE__)

# As a student, I want to be shown a confirmation when making a booking

describe "confirm dialog" do

  before :all do
    @student = Student.create(:sid => 123456789, :first_name => "Frodo", :last_name => "Baggins", :cp => 4, :comments => "The ring is mine.", :email => "frodo@baggins.com")
    @experiment = Experiment.new(:name => "physics 1", :exp_num => 15, :num_sessions => 2, :weight => 2)
    # Booking.makeBooking(@student, @experiment.exp_num, Date.today, "123456789")
  end

  it "has javascript confirm" do

    assign(:experiment, @experiment)
    assign(:student, @student)

    render

    rendered.should include('<input name="commit" type="submit" onclick="return confirm(\'Booking will be made\nAre you sure?\')" value="Book Experiment" />')

  end
end
```

3.1.2 Cancel Booking Spec

```ruby
require File.expand_path("../spec_helper", __FILE__)

# As a student, I want to be able to cancel a booking a certain number(X) days before the booked date

describe Booking do

  before :all do
    @student = Student.create(:sid => 123456789, :first_name => "Frodo", :last_name => "Baggins", :cp => 4, :comments => "The ring is mine.", :email => "frodo@baggins.com")
    @experiment = Experiment.new(:name => "physics 1", :exp_num => 15, :num_sessions => 2, :weight => 2)
  end

  it "has cancel button disabled" do

    Booking.makeBooking(@student, @experiment.exp_num, Date.today, "123456789")

    assign(:experiment, @experiment)
    assign(:student, @student)

    render

    rendered.should include('<input disabled="disabled" name="commit" type="submit" value="Delete Bookings" />')

  end

  it "has cancel button enabled" do

    Booking.makeBooking(@student, @experiment.exp_num, (Date.today) += 7, "123456789")

    assign(:experiment, @experiment)
    assign(:student, @student)

    render

    rendered.should include('<input name="commit" type="submit" value="Delete Bookings" />')

  end


end
```

3.1.3 Admin Add Student Spec

```ruby
require File.expand_path("../spec_helper", __FILE__)

# As an admin, I want to be able to add students with their personal details

describe Student do

  it "won't create without sid given" do
    @student = Student.new
    @student.save.should be_false
  end

  it "should not create without a unique SID" do
    @student = Student.create(:sid => 123456789, :first_name => "Frodo", :last_name => "Baggins", :cp => 4, :comments => "The ring is mine.", :email => "frodo@baggins.com")
    @student_dup = Student.create(:sid => 123456789, :first_name => "Frodo", :last_name => "Baggins", :cp => 4, :comments => "The ring is mine.", :email => "frodo1@baggins.com")
    @student_dup.save.should be_false
  end

  it "should not create without a unique email" do
    @student = Student.create(:sid => 123456789, :first_name => "Frodo", :last_name => "Baggins", :cp => 4, :comments => "The ring is mine.", :email => "frodo@baggins.com")
    @student_dup = Student.create(:sid => 133456789, :first_name => "Frodo", :last_name => "Baggins", :cp => 4, :comments => "The ring is mine.", :email => "frodo@baggins.com")
    @student_dup.save.should be_false
  end

  it "should not create with an SID under 9 numbers" do
    @student = Student.create(:sid => 12345678, :first_name => "Frodo", :last_name => "Baggins", :cp => 4, :comments => "The ring is mine.", :email => "frodo@baggins.com")
    @student.save.should be_false
  end

  it "should create with valid parameters" do
    @student = Student.create(:sid => 123456789, :first_name => "Frodo", :last_name => "Baggins", :cp => 4, :comments => "The ring is mine.", :email => "frodo@baggins.com")
    @student.save.should be_true
  end

  it "should not create with invalid amount of credit points" do
    student = Student.create(:sid => 123456789, :first_name => "Frodo", :last_name => "Baggins", :cp => -2, :comments => "The ring is mine.", :email => "frodo@baggins.com")
    @student.save.should be_false
  end
end
```

3.1.4 Book an Experiment

```ruby
require File.expand_path("../../test_helper", __FILE__)

class ExperimentTest < ActiveSupport::TestCase
    test "won't create without name given" do
        experiment = Experiment.new
        assert !experiment.save, "saved experiment without critical information"
    end

    test "won't create if exp_num is not a number" do
        experiment = Experiment.new(:name => "physics 1", :exp_num => "aaa")
        assert !experiment.save, "created experiment with invalid experiment number"
    end

    test "should create with valid name and number" do
        experiment = Experiment.new(:name => "physics 1", :exp_num => 15, :num_sessions => 2, :weight => 2)
        assert experiment.save, "failed to create a valid experiment"
    end
end
```

3.2 ACCEPTANCE TESTS

These acceptance tests have been approved by the client and has feedback attached.

User Task	Acceptance Test	Client Feedback (confirm accuracy of tests)
As a student, I want to book experiments, based on my credit point requirements	Given I am a student, When I have credit points of 2 Then 3 experiments will be available for me to book. When I have credit points of 4 Then 5 experiments will be available for me to book. When I have credit points of 6 Then 8 experiments will be available for me to book. When I have credit points of 8 Then 10 experiments will be available for me to book.	When booking, I choose one from the list of experiments. With 2 cps, I can only book 6 sessions; for example, 2 sessions for one experiment or 4 sessions for one experiment as required.
As a student, I can only make one experiment session per day	Given I am a student, When I book an experiment on any particular day that the lab is open Then the day becomes occupied and I cannot book another session for an experiment on that day	ok
As a student, I want to be shown when days are fully booked	Given I am a student, When I proceed to make a booking Then the fully booked days appears occupied (white cell) so I cannot select them.	ok
As a student, I want to be shown a confirmation when making a booking	Given I am a student, When I have selected the day(s) to make a booking for an experiment Then a confirmation box pops up with the options OK and Cancel. When OK is selected the booking will be processed. Otherwise booking will be cancelled.	after OK selected, the program should check the table and make final sure the day is not booked by other students
As a student, I want to be able to cancel a booking a certain number(X) days before the booked date	Given I am a student, When I select a booking made and proceed to cancel that booking and I am X days before the experiment's booked date Then I can cancel the booking. Otherwise it will not let me to select an option to cancel the booking.	ok
As an admin, I want to be able to add students with their personal details	Given I am an admin, When I select to create a student, there will be fields to input their first name, last name, SID (username), email, number of credit points, phone number, comments,	This may be a small bug. Although there is an error message, a new student has actually been added. admin doesn't need confirmation

	and a password and once I select to create a student Then there will be a confirmation to notify that the student's details have been accepted.	because he can edit/change the student's details when needed. Could you fix the bug in import student list as well?

3.3 CLIENT'S PRIORITY LIST FOR REQUIREMENTS

The list below mentions Admin functions, Tutor functions and Student functions and is the basis for our User Stories.

3.3.1 Admin functions

Requirement	Priority	Additional Description
Setup dates for available experiments in current semester.	1st	Different experiments may have different calendars (different days). In current system, all experiments use the same calendar View -- list of experiments and a calendar. Select experiments and days to set up calendar for specific experiments
Fix the bug in adding new student	1st	
Fix the bug in import student list	1st	
Admin is able to make/delete any bookings at any time and send emails to affected students for notification of the changes.	2nd	The same booking rules (see students section below) apply. The emails sent out will have a backup in the system and be viewed as required. It displays what experiments have been booked and done so far for each student.
View the booking summary	3rd	Bookings Summary also provides information about total free sessions, total required sessions, and total booked sessions
Fix the bug in downloading student's report	3rd	
In Experiments, venue should be included in the table and displayed in the relevant pages, such as booking summary of student's login.	4th	

3.3.2 Tutors

Requirement	Priority	Additional Description
Fix the bug in Download student's report	3rd	Similar to admin
View bookings summary	4th	Similar to admin

3.3.3 Students

Requirement	Priority	Additional Description
Make online bookings for the experiments, based on their required CPs.	1st	
Each student can only do single experiment session at a time	1st	In booking calendar for each experiment, the days should be white, which are already booked by others for the same experiment and which are

		already booked by this student for other experiments
Student booking should be made evenly across the semester.	1st	Students are not allowed to over-book sessions (eg. With 4cps, students cannot book over 10 sessions).
There should be a confirmation window to make sure they want to book on these days before further action.	1st	
Students should be allowed to cancel the booking they made, but only eg. more than X (2/3/4?) days before the booked dates.	1st	
Admin is able to define parameters X. or not allowed to cancel the booking (cancellation must be done via admin).	1st	
There is a bug in make_booking of current system.	1st	
In 'Bookings Summary', display course requirements for the student	4th	CPs, total number of experiments required, number of sessions already booked, and number of sessions to be booked
In 'My Marks', display paperwork to be examined	4th	number of logbooks, report, poster, talk and assignment

3.4 EXPERIMENT DATA

This data was given to us by the client to do initial testing with. This can be useful for integration testing to see what the system's response should be in reaction to a user input.

id	exp_num	name	num_sessions	special	comments	created_at	updated_at	extended_name	available
1	1	Photonic Amplifier	4	0		2010-06-22 11:11:17	2012-08-02 00:08:19		1
2	2	Data acquisition with computers	2	0	From 2011 Semester 1, this experiment combines wha...	2010-06-22 11:11:17	2011-02-28 10:04:19		1
3	3	Fourier Analysis	2	0		2010-06-22 11:11:17	2010-07-20 07:54:53		0
4	5	Noise	2	0		2010-06-22 11:11:17	2012-04-18 23:00:33	Noise: Setup 1 (of 2)	1
5	5	Noise	2	0		2010-06-22 11:11:17	2010-08-01 23:37:35	Noise: Setup 2 (of 2)	1
6	6	Transmission Lines	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
7	7	Wave Propagation	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
8	8	Microwaves	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
9	10	RF Breakdown	2	0		2010-06-22 11:11:17	2010-08-01 23:37:53	RF Breakdown: Setup 1 (of 2)	1
10	10	RF Breakdown	2	0		2010-06-22 11:11:17	2010-08-01 23:38:15	RF Breakdown: Setup 2 (of 2)	1
11	12	Langmuir Probes	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
12	13	X-ray Diffraction	2	0		2010-06-22 11:11:17	2010-07-15 07:32:29		1
13	14	Nuclear Magnetic Resonance	4	0		2010-06-22 11:11:17	2011-02-28 10:05:00		1
14	15	Electron Spin Resonance	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
15	16	Optical Pumping	4	0		2010-06-22 11:11:17	2011-02-28 10:05:18		1
16	17	X-rays	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
17	18	Mossbauer Effect	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
18	19	Cosmic Radiation	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
19	20	Beta Decay	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
20	21	Alpha Particles	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
21	22	Gamma Radiation	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
22	23	Nuclear Lifetimes	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
23	24	Michelson Interferometer	2	0		2010-06-22 11:11:17	2010-08-01 23:38:39	Michelson Interferometer: Setup 1 (of 2)	1
24	24	Michelson Interferometer	2	0		2010-06-22 11:11:17	2010-08-01 23:38:54	Michelson Interferometer: Setup 2 (of 2)	1
25	25	Holography	2	0		2010-06-22 11:11:17	2010-07-20 07:55:42		0
26	26	Reflection	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
27	27	Fourier Optics	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
28	28	Interference Spectroscopy	2	0	*NULL*	2010-06-22 11:11:17	2010-06-22 11:11:17	*NULL*	1
29	29	Galactic Dynamics	2	0		2010-06-22 11:11:17	2012-02-28 01:08:09		0
30	30	Thin-film Deposition	2	0		2010-06-22 11:11:17	2012-07-24 02:17:29		1
35	31	Scanning Electron Microscope	4	1	SEM is not booked through this system. It is avail...	2010-07-20 07:57:30	2012-03-26 04:52:35		1
36	32	Electronics	0	1	Added by Peter Dobrohotoff 25-Nov-2010 for the Ele...	2010-11-25 01:47:40	2010-11-25 06:18:28		1
37	33	Atom Probe	2	1	New (2012 S1) externally provided experiment, not ...	2012-03-06 05:05:07	2012-03-26 04:54:11		1
38	34	Noise - Double	4	1	This is a double length version of the Noise exper...	2012-04-06 07:15:04	2012-04-06 07:15:04	Noise (double length)	1
39	35	Quantised Conductance	2	0		2012-07-24 02:17:15	2012-07-24 02:17:15		0