ELEN4010 - Software Development III
Meeting Scheduling System
Post Mortem Analysis

Ari Croock (718005)
Kanaka Babshet (678851)
Alice Yang (597609)
Daniel Weinberg (547937)

May 19, 2016

1 Introduction

The purpose of this report is to analysis the implementation of the web application in its first, second, and third iteration implementations. The implementation of the application are divided between four developers. The testing results, release features, accuracy of time estimates and other considerations are discussed.

2 Iteration Implementation

2.1 First Iteration

The following high priority user stories were implemented:

1. View Schedule
2. Allow a lecturer to book a meeting on behalf of a student
3. Allow students to book a meeting with an academic

By implementing these user stories basic functionality was achieved. For simplicity, the academic schedule is divided into 30 minutes time slots. This means the length of each meeting must be a multiple of 30 minutes. Based on this assumption, a simple calendar interface was implemented.

A major defect was encountered during the first iteration. The bug was observed when a user attempted to book a second meeting with a particular academic after booking a previous meeting. The new meeting would overwrite the previous booking, meaning only one booking could exist at a time. This was fixed towards the end of the first iteration.

Unit tests were found to fail often during this iteration. This was caused by frequent changes in the database interfaces which were still being developed. These have stabilised by the second iteration.

2.2 Second Iteration

The application was extended with the following features:

1. The lecturer can create group meetings
2. Students can join group meetings that have been created by a lecturer
3. Lecturers can view and edit the list of attendees of group meetings

Since basic functionality was implemented in the first iteration, the following iterations contain fewer new features and added functionality.

These features required changes to the database schema; consequently the database was erased and recreated to accommodate for the additional required data.

Unit tests failed much less frequently compared to the first iteration, since the database design had mostly stabilised.

Since far fewer features were included in this iteration, the time to release and add additional value to the application was significantly reduced.

2.3 Third Iteration

Reporting functionality was added to the application. This helps to improve lecturer feedback by allowing lecturers to view all meetings in a specified period.

The second and third iterations were completed significantly quicker than the first, since most technical hurdles were overcome in the first iteration.

It was ensured that each iteration constituted a viable and usable product, and that value was added in each successive iteration.

Code reviews were found to be valuable since they helped detect coding style issues as well as subtle problems before the code could be merged and deployed.

3 Time Estimates

The time estimates of each feature is introduced by the use cases and user stories. The poker planning method was applied when estimating the time for each task. The poker planning requires the participation of all software developers, whereby the each member makes an assumption of the time allocation for each task privately without external influences. The documented time estimates are finalised when all software developers are gathered and the time estimations are agreed upon. Estimated as well as actual time spent on each task can be seen in Table 1.

Table 1: Table illustrating the Time Estimations and Actual Time Spent

Tasks	Time Estimate(hours)	Actual Time (hours)
Create Homepage	1	1
Create student html	1	2
Create Lecturers html	0.5	0.5
Setup database	1	1.5
Booking form script for lecturer	2	3.5
Create database interface for Meeting table	2	3
Create database interface for Meeting Students table	2	2
Make booking form script work for students	1.5	0.75
Calendar template	2	6
Add group functionality to booking form for lecturer	2	1.5
Connect meeting_students with meeting table interface	1	2
Calendar script to generate calendar	2	2.5
Meeting scheduling script to receive from booking form	2	2
Move monthly calendar into template so it can be reused	0.5	1
Link student monthly calendar to weekly calendar	0.5	0.5
Use monthly calendar in lecturer's page	0.5	0.5
Bug Fix: Booking second meeting without overwrites	1	0.75
Report Generator for lecturers	2	3

4 Developer Task Assignment

The actual time spent on tasks was usually fairly close to the estimate, with a few exceptions. The precision of the estimates can be attributed to the poker planning method, as well as the fact that time spent was measured with a resolution of only half an hour. Initial setup tasks (such as the database interfaces) took longer than expected, since they required additional unforeseen work. Ideally, these tasks should have been split into smaller subtasks, however this is difficult to do up front.

Table 2: Table showing the task assignment to each developer and it priority level

	Task	Priority Level
Developer 1	Setup database	High
	Booking form script for lecturer	High
	Create database interface for Meeting table	High
	Create database interface for Meeting students table	High
	Connect meeting_students with meetings table interface	High
	Calendar script to generate calendar	High
	Link student monthly calendar to weekly calendar	High
	Bug Fixes	High
Developer 2	Create homepage	High
	Meeting scheduling script to receive from booking form	High
Developer 3	Create student html	High
	Add group functionality to booking form for lecturer	Moderate
	Make booking form script work for students	High
Developer 4	Calendar template	High
	Move monthly calendar into template so it can be reused	Moderate
	Implement monthly calendar in lecturer's page	High
	Use monthly calendar in lecturer's page	High
	Report Generator for lecturers	Moderate

5 Individual Retrospectives

5.1 Developer 1

A major fundamental issue with the project development was encountered during the first iteration. Despite the use of user stories and use cases, the system was being developed in terms of *horizontal slices* instead of the preferred vertical slices. This meant that functionality was not being incrementally added from the user's perspective. Consequently, the first iteration took longer than expected and a large amount of functionality was released at once. This made gathering incremental feedback on the application's progress difficult.

During the second iteration several problems were encountered, involving links between various pages of the site. This was fixed once the final design of the system had stabilized.

Testing the features of the second iteration presented a challenge as not all html element tags are given a id or name, which makes it difficult to reference during testing.

Due to the tight schedule, simplicity of the user interface was favoured. Consequently, the sites visual appearance is lacking in some aspects.

5.2 Developer 2

Upon project initialisation, the first dilemma faced, was the integration of Python and HTML. The group therefore decided to incorporate Jinja2 for increased usability before further implementations.

Saving the meeting details to the database was attempted before implementing the complete functionality of the meetings. This resulted in technical debt requiring multiple reviews of prior code.

It was difficult to write successful tests that thoroughly test the functionality of a website. While user tests may seem ideal, a more concrete method of testing was required. Selenium, a web browser automating program, was therefore then implemented. While Selenium should allow a developer to record website surfing, these generated tests were never written correctly. All tests were therefore manually written - a process that was slightly tedious. Additionally, due to the unsystematic development of the project, tests were often rewritten in order to account for the minor design changes.

5.3 Developer 3

One of the initial decisions that was initially made was to use *mako* as a convenient way of integrating python with HTML. This was decided against when it was discovered that significantly more documentation existed for utilizing *jinja2* as a better alternative. The first integration presented several issues, primarily

with initial set-up, particularly due to network issues. A design decision that complicated matters was the idea of creating a weekly booking calendar. This became a problem when creating the student html page as it was difficult to come up with a method of choosing the week to display. Eventually a modified version of a date-picker was opted for.

In terms of the booking form for students when a group booking exists, an initial design choice was to attempt to grey out the subject to avoid changes. This was done by disabling the input which became an issue when submitting the form as it was ignored.

No problems were experienced when enabling a group booking option for the lecturer to select. A slight difficulty was experienced when trying to obtain information regarding previous inputs.

Testing was relatively straight forward through the use of Selenium automated testing for the front end. This was used to check usability of the website by testing inputs, submissions and links. Selenium does contain a recorder, however any test written through the use of the recorder was unnecessarily long and failed when run. As a result, manually written tests was preferred where all tests passed. The only time that testing presented issues was when elements had not been assigned an ID or a name. This made it slightly more complicated to reference and the element path (xPath) was required.

5.4 Developer 4

During the first iteration, the problems encountered were mainly introduced in integrating python with html as well as determining the type of information the page will receive from the database. The integration required accessing given dictionaries from python and displaying the correct information. The problems regarding the display of the data were easily resolved as the developer's experience increased overtime. Although, with regards to how the data is received from the database, it was time consuming as the database was not fully developed, due to the initial *horizontal slice* implementation.

The calendar template was easily tested as it only needed to ensure that the template opened correctly and the correct information was displayed. Although further unit tests could have been performed to ensure that every aspect of the code is functional.

In the third iteration, the developer's skills were improved, and hence the implementation of the additional feature was simple. Thus significantly reducing time taken to implement the feature.

6 Further Improvements

A continuous integration (CI) server could have been used to ensure that tests are run regularly, in order to prevent subtle errors in the application. This was not performed due to time constraints.

Not all features described in the system requirements specification were implemented in the first three iterations. Further iterations and additional development time would be required in order to implement these features and improve the user experience.

Authentication was not implemented as it was decided that this would not add significant value to a minimum viable product, although it should be implemented in future iterations.

No front-end visual styling was implemented as this is not required for a minimum viable product. The interface is still usable regardless of its lacking visual appearance.