Project Manual 2018–2019

Module Data & Information (201700279)
University of Twente

Klaas Sikkel
Luís Ferreira Pires
Maurice van Keulen
Djoerd Hiemstra
Faiza Bukhsh

14 April 2019

Contents

1 Introduction

In the module Data & Information you will learn and practise with the design, implementation and testing of complex layered software systems, by using de facto standard tools such as database management systems, CSS and scripting frameworks and web services. Data play an important role in this module, in terms of classical structured data stored in databases, semi-structured data and data retrieval from social media and other web resources.

In this module, a big project is performed according to the agile development principles by using Scrum, which is a modern and currently popular way of organising a (software) project. This means that (i) you will get experience with handling stakeholders (more specifically the client who has ordered the system), (ii) the teams manage themselves (there is no single manager who tells you what to do), (iii) the teams are multidisciplinary, and (iv) every two weeks a new or improved product is delivered.
Scrum is explained in an introductory guest lecture (don't miss it!).
For further reference there is a separate document *Scrum Concepts and Practices*, which describes how we use Scrum in this module.

The project teams will be allowed to choose from the realistic cases that have been proposed by our commercial partners (see Section 1.1) or define their own case for the project (see Section 1.2). The teams have to make their preferences known through Canvas on the first day of the project. Hopefully every team can do the project of their first choice. However, there is a maximum number of teams per project.
For each project, one of the teachers will act as product owner. If you choose one of the project for external clients, the product owner acts as an intermediate between the Scrum team and the client. But you will also have opportunities to talk to the real client and discuss the proposed system with them.

Deadline: 24 April 2019, 12:30: As a team, indicate your preferences for a module project on Canvas.

1.1 Projects for external clients

Several projects have been defined by clients inside and outside the University. All clients will give a pitch at the Module Introduction lecture. More detailed project descriptions will be available from Canvas.

1.2 Own case

Teams also have the option to define their own case for the project. Design and implement a web application for your sports club, your theatre group, your uncle's business, your church or any other (legal) organisation that you prefer. Such a project should involve developing a web application with nontrivial underlying logic and a database.
Proposals for such a project have to be approved by the module coordinator.

2 Work procedures, tools, and facilities

When working on a big project, it is impossible (or at least far too expensive) to write all the necessary programming code from scratch. Therefore it is common practice to make use of available software libraries, frameworks, applications and tools. The choice of work procedures and tools depends on the working environment and is often arbitrary. The Agile Manifesto proclaims to favour 'individuals and interactions over processes and tools' (see *Scrum Concepts and Practices*). However, many work procedures and tools for software development are known to be helpful. Furthermore, team work is only effective if all the team members agree to follow the same procedures and to use the same tools. All the team members have to use the same tools for a team to be productive, even if some individual team member has some other preferences, as illustrated by the words of Guido van Rossum, the driving force behind Python:

> You can make me use Eclipse but you can't make me like it.
>
> *Guido van Rossum (https://twitter.com/gvanrossum/status/424595747397332992)*

Big successful software companies often force their projects to follow standard procedures, so that new project members can be added much more easily to projects than if each project had its own special procedures. Actually, all software projects share some similarities: source code and unit tests can be found at some location, software is produced according to more or less standard steps (e.g., compile, test, package, verify, deploy), documentation should comply with a certain pre-defined structure, etc. In short, without standard procedures and specific tools it takes a lot of time to understand a project and contribute to it. Also, without these agreements it would take the module staff much more time to assess the project results.

Therefore each project team has to use the following tools and libraries:

- Trello (for planning the work).
- Eclipse (for software development).
- Git and GitLab (for cooperation and version management).
- Maven (for defining software dependencies and facilitating software building).
- Java and Java Servlets (for implementing web applications).
- JavaScript, HTML5 and CSS (for implementing web application front-ends).
- Jersey (for implementing RESTful web services in Java).
- JUnit (for unit tests).
- PostgreSQL (database management system).
- Apache Tomcat (application server and web container).

For drawing UML diagrams, you can use Visual Paradigm or another system of your own choice.

> My goals were different and I wanted to make an opinionated piece of software and I preferred the notion of convention over configuration. I wanted a project's infrastructure to look the same and work the same so I continued to pursue my own model for a project (...) I wanted to save people time by being able to find things in the same place.
>
> *Jason van Zyl* (http://maven.apache.org/background/history-of-maven.html)

Some tools, like Maven, support rigid standard procedures that should only be ignored if there are very strong reasons to apply dissimilar procedures. This practice is often termed 'convention over configuration'. Many software frameworks force developers to follow such rigid procedures. Here, a framework is considered to be a reusable software component, which can be configured so that it can be used for some specific purpose. The most general the framework, the more configuration is necessary, and therefore the less time is saved and the less useful is the framework for a specific project. The project teams are advised to make additional agreements concerning procedures and tools, such as, for example, about the framework to be used for HTML5, CSS and JavaScript.

Frameworks like Spring are not allowed for this module. Computer Science students should have a thorough understanding of what Spring actually does 'under the hood'. Therefore, in this module you are requested to program this yourself, so that you know what goes on there if you use such frameworks in future.

For all project teams a project room has been allocated for all the project sessions during the module.

During most project sessions (except for the first few weeks) teaching assistants will be available for some hours.

3 Project management

A separate document, *Scrum Concepts and Practices*, describe the general ideas about Scrum project management. This section describes specifics of what you have to do, and when.

3.1 Project teams

Make sure you become a member of a project team consisting of 5-6 people (see information on Canvas). *Please note*: when all teams have been assigned to projects, team will be renumbered. Therefore, you should wait until Fri 26 April to install the Gitlab environment as described below.

3.2 Git and GitLab

Each member of the team should be able to log in to https://git.snt.utwente.nl/ by using their student number. Ask one of the Scrum mentors for your team number and create a group in GitLab for your team. For example, if your team is number 42, then your group in GitLab should be called di42, and the URL for your team is https://git.snt.utwente.nl/di19-42. Create a project group and add all team members to this project, with Master permissions.
Add also both Scrum mentors to this project, with *reporter* permissions (not observer).
After that add a `README.md` file to this project.
A suggested structure for your GitLab repository is shown in Figure 1.

Important: Make sure that the HEAD always compiles to a working product, branches are created for each new feature and unit tests are ready (and committed) before new functionality is implemented. Useful information on Git can be found at http://git-scm.com/book/en/Getting-Started

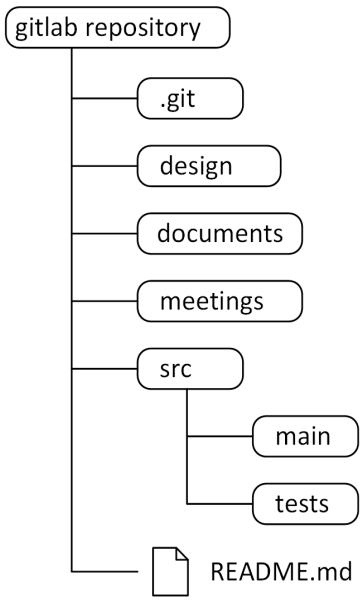

Figure 1: Gitlab structure

Deadline: Fri 26 April 2019. All students have access to a project in GitLab defined in the scope of their team (GitLab group).

3.3 Sprints and global milestones

In order to make the whole project manageable, the duration of the sprints has been optimized regarding the irregularities in the schedule (there are a number of free days for different reasons). Some global milestones have also been defined.

Sprint 1 (including Preparation, Requirements): 24 April – 15 May

Before you can start with a real sprint, a lot of preparation has to be done: getting the team started, getting to know the client, gathering requirements, coming up with a first product idea. Consequently, it is impossible to be make a planning on the first day. Probably you will need the first two weeks (with not that much project sessions) for getting yourselves organized, gathering requirements, defining a product idea, and a first attempt at making a product backlog. The first project session in week 3 (8 May) should be a good moment to take stock of what you have and make a planning for the remainder of Sprint 1.

At the sprint review meeting on 15 May you are asked to present the essential product idea, the requirements for this product and a mock-up of what the product could look like. You are encouraged – but for this sprint it is not mandatory – to build an initial core of what the system could do and give a demo at the print review.

Sprint 2: 16 May – 3 June

A working system with core functionality is the intended result of this sprint. A high-level design of the application (requirements and database models) should also be delivered at the end of this sprint.
At the sprint review meeting you are required to demonstrate a working system (*potentially shippable product increment*).

Sprint 3: 5 June – 18 June

The next potentially shippable product increment is realized and demonstrated at the sprint review meeting.

Sprint 4: 19 June – 28 June

At the end of sprint 4 you have a working application that satisfies the criteria mentioned in Appendix B. It is this version that will be assessed for technical (programming) quality, which determines a substantial part of the project grade.

Consolidation: 1 July – 5 July

The last week is intended for consolidation. This could consist of:

- improving technical defects (could be requested by the module staff), and/or
- adding last features or improving existing ones, and
- preparing a convincing presentation for the client.

The final presentation will be important for the overall assessment of the product you delivered. This accounts for another part of the project grade.

3.4 Sprint backlog in Trello / planning meetings

In this module we use Trello, which is (free) project management application. All students are a requested

to create a free account on http://trello.com.
Trello is a suitable tool for creating and maintaining a sprint backlog.
Each group should create a project in Trello, add tasks to this project as user stories and create a sprint backlog. Make sure each task has an estimated effort (see *Scrum Concepts and Practices*) Allow your Scrum mentors to become members of your Trello project, so that they can follow the progress of your project team. Add the URL of the Trello project board (for example, https://trello.com/b/0tcjABX8/foobar, if your team is called 'foobar') to the `README.md` file in the GitLab repository of your project.

For each sprint, you are requested to make a sprint planning, resulting in a sprint backlog. Making a planning involves effort estimation for all (potential) items on the sprint backlog.

> Suggested dates for planning meetings: 8 May, 16 May, 5 June, 19 June, 1 July
>
> Deadline: 8 May 2019. All students have a Trello account and can access the project board. The project backlog is in the Trello project board.

3.5 Daily Standup meetings

Every project session (apart from Mondays in weeks 7-11), one of your mentors will visit the team between 9:00 and 10:00 for the Daily Standup. You are always present, unless illness or calamity strikes, as you don't want to be disloyal to the team.

3.6 Sprint review meetings

Sprint reviews are one of the cornerstones of Scrum. Sprint review meetings take place on 15 May, 3 June, 18 June, and 28 June. You can upload the materials used for the presentation (e.g., slides) to the Gitlab repository of your project, in a director called `meetings`.

3.7 Sprint retrospectives

As part of the Project Skills component of the module, you should send a summary of your sprint retrospective to your skills teacher. In this way, the skills teacher can see that you internalized concepts from the Skills tutorials. For teams that do a good job on this, the final assignment for Project Skills will be waived.
You can also store notes and summaries of your retrospectives in the `meetings` directory in the GitLab repository of your project.

> Suggested dates for sprint retrospectives: 8 May, 16 May, 5 June, 19 June, 1 July

3.8 Teamwork

Each team member actively participates in the project. Team members distribute the work evenly among themselves and can try out different roles in the course of the project. In particular, every team member should have the chance to act as a scrum master for a certain period of time.

Red/Green card procedure

A team can give a red card to one of its members when the student endangers the project by contributing insufficiently, repeatedly not showing up, or not keeping promises. A red card can only be given if all other team members agree to this. Handing out a red card should be immediately reported to one of the mentors. A red card can be withdrawn if the team agrees that the insufficient contribution has been

compensated (e.g. by extra efforts). This should also be reported to the mentors. The consequence of a red card can be the subtraction of a point from the grade of the student who has received the red card. In extreme cases, the module staff may decide in consultation with the team to exclude a student from the project. Students who think their red card is unfair or unjustified should contact their mentors or the module coordinator.

At the end of the project a team can hand out at most one green card to a team member as a reward for an extraordinary effort or an important contribution. The team should explain to one of the mentors why they consider a green card to be appropriate. A green card may add up to one point to the grade of a student (to be agreed with by the supervisor).

4 Design

This section describes some mandatory steps in the design of your system.

4.1 Choose a case/project

Each team should write a document of 300–500 words that describes the case chosen by the team. This document may contain text fragments from the project description on Canvas. Convert the document to PDF format, call it `project.pdf` and upload it to the GitLab repository of your project in a directory called `design`.

> Deadline: 10 May 2019. Document `project.pdf` with project description is copied to the design directory of your project in the GitLab repository.

4.2 Mock-up

Select a CSS framework (see http://en.wikipedia.org/wiki/CSS_frameworks) and learn how to use it. Build a mock-up of your application by using the CSS framework of your choice. This mock-up should consist of one or more HTML pages that use the stylesheets available in the CSS framework. Generate an archive file `mockup.tar.gz`, `mockup.zip` or `mockup.war` that contains the files of your mock-up, and copy this file to the `design` directory of the GitLab repository of the project. Alternatively you can create your mock-up with the help of a drawing tool, such as Sketch or Paint, or with pen and paper. In this case you should create a `mockup.pdf` file by saving the drawing file as PDF or scanning your drawing.

Useful information can be found at http://en.wikipedia.org/wiki/Mockup#Software_Engineering (for mock-ups in general), http://www.bohemiancoding.com/sketch/ (for Sketch), http://en.wikipedia.org/wiki/Pen (for pen) and http://en.wikipedia.org/wiki/Paper (for paper) ☺

The terms 'HTML5', 'Responsive' and 'mobile-first' will make you sound 'cool' during the first sprint presentation, but make sure you know what they mean before using them.

> Deadline: 15 May 2019. A Mock-up file is uploaded to the `design` directory of your project in the Gitlab repository

4.3 UML diagrams

Using Visual Paradigm or any other UML tool of your preference, draw a UML use case diagram and a UML class diagram for the web application that you are going to develop. Export your UML diagrams to a standard picture format (`.jpg` or `.pgn`) and copy the files to the `design` folder of the GitLab repository of your project.

> Deadline: 3 June 2019. Picture files of the UML use case and class diagrams are uploaded to the `design` directory of your project in the GitLab repository.

Please note that UML class diagrams are used somewhat differently at different levels of abstraction. At the programming level, class diagrams contain details like methods, constructors, abstract classes, etc. In this module, a class diagram is a data model, to be refined into a database schema.

4.4 SQL Database design

Create the file `schema.sql`, which should contain the database design that has been derived from the UML class diagram of Section 4.3 . Make sure that the primary keys and the foreign keys are properly defined in this schema. Upload this schema to `design` directory of the GitLab repository of the project.

> Deadline: 3 June 2019. A file `schema.sql` containing the database definitions is uploaded to the `design` directory of your project in the GitLab repository.

5 Programming

This section discusses the criteria concerning the programming work that will be used to assess the project results. The use of several tools, libraries and components is mandatory. You will probably only be able to fully understand these criteria after following the lectures and doing the exercises that cover these tools, libraries and components.

5.1 Compilation to a WAR file with Maven

The team must follow the Maven standard directory structure for Java projects (see http://maven.apache.org/), as explained in the first laboratory session. Make sure your web application compiles 'out-of-the-box' by declaring all the project dependencies in the `pom.xml` file located in the root of the Maven project directory. The build process (compilation) should generate a `.war` file that can be deployed in the Apache Tomcat Application Server (see http://tomcat.apache.org/). Copy the Java code and the Maven `pom.xml` file to the GitLab repository of the project.

5.2 JUnit Unit tests

The code (Java classes) should be tested with Junit (see http://en.wikipedia.org/wiki/Junit) unit tests. Make sure that the unit tests that use the database work on an empty PostgreSQL database, so that the unit tests can add and remove data from the database accordingly. The JUnit annotations `@BeforeClass` and `@AfterClass` can be quite useful and are recommended.

Tools for 'code coverage', such as, e.g., EclEmma (http://www.eclemma.org/) can be also useful.

5.3 Test-driven development [recommended]

When applying test-driven development the unit tests are developed and committed to the GitLab repository before the code to be tested is implemented (see http://en.wikipedia.org/wiki/Test-driven_development). The unit tests play the role of the requirements for the code, by describing what the code is supposed to do (or how the code is supposed to behave). Use a new Git branch for code that fails the tests, and make sure the JUnit tests always succeed in the HEAD. Never check in code that does not compile.

5.4 Java Servlets

In this module we use Java Servlets to implement web applications (see http://en.wikipedia.org/wiki/Java_Servlet). The code produced by the groups should therefore make correct use of Java Servlets. The Apache Tomcat Application Server (http://tomcat.apache.org/) is the reference implementation of the Java Servlet and JSP standards (APIs), and that is one of the reasons why we use Tomcat in this module.

5.5 RESTful web services and XML

In this module we also introduce the students to RESTful web services (http://en.wikipedia.org/wiki/RESTful). The application should therefore make use of RESTful services and XML serialisation, possibly using JSON (http://json.org/). Jersey (https://jersey.java.net/) is the standard implementation of the JAX-RS standard, which defines how RESTful services can be implemented using Java. This is one of the main reasons why we use Jersey in this module.

5.6 RESTXQ [recommended]

In the SEMI-part of this module, you will also learn to work with RESTXQ, which is a standard for developing RESTful web services using XQuery only (XQuery is a standard query language for XML data). We recommend that you try to use this technology for one or more of your web services to experience its strengths and weaknesses as compared to developing them in Java.

5.7 PostgreSQL and JDBC

In this module, we use PostgreSQL as database management system (DBMS). In order to decouple the Java code from the specific DBMS chosen in this module, the code should interact with PostgreSQL through JDBC (see http://en.wikipedia.org/wiki/Java_Database_Connectivity). It may be desirable to develop certain functionality using stored procedures.

5.8 Framework for HTML, CSS and JavaScript

The code should make use of a CSS framework to implement the user interface (also called the 'front-end'). The more complete the mock-up built in the initial design (see Section 4.2), the easier it will be to reuse it in the implementation of the front-end.

5.9 Concurrency

The web application correctly uses database transactions where they are required by the application. You may use lower isolation levels to improve performance, but be sure to use it correctly.

6 Project finalisation and delivery

This section discusses criteria for the final product and the project delivery.

6.1 Working system

The final product (the web application) should be deployed and run in the central Tomcat Application Server (AS) at http://datainfo.ewi.utwente.nl. In due course, you will be given a user id and a password to sign in to this server. Add the URL of your application to `README.md` in the GitLab project of your team. For example, the URL should be http://datainfo.ewi.utwente.nl/di10_app if your application is called `di10_app`.

Since all users of the central Tomcat AS can manage all applications, you have to make sure your application has a unique name. Therefore we use the convention that the application name starts with `diXX_`, where XX is the team number. Furthermore, you should only deploy the application to the central Tomcat AS after the application has been thoroughly tested using Eclipse and your local Tomcat installation, since this central Tomcat AS is shared by all groups and it takes a while before it can be restarted in case you manage to make it crash. Making the central Tomcat AS crash will not make you popular with your fellow students.

Some ambitious groups may assign a separate domain name for their applications. In this case, you should inform the domain name and the application name to the Scrum mentors, so that the central server can be properly configured. A separate domain name does not result in a higher mark, so this is only 'for fun'.

> Deadline: 29 June 2019. A working system (all updated artefacts and running web application).

This version will be extracted from your Gitlab repository and be used to assess the technical (programming) quality of your work. It should be a complete working system, i.e., a potentially shippable product increment, but you can still add or improve things in the last week, see Section 3.3.

> Deadline: 5 July 2019. Final presentation.

6.2 Application testing

In addition to testing the Java classes using JUnit as mentioned in Section 5.2 , you also have to test your application systematically as a black box. Web application can be tested by (Java) applications that simulate browsers, or by using browser plugins. For example, Selenium (http://docs.seleniumhq.org/) is a popular environment that offers many tools for facilitating the systematic testing of web applications, including an environment to define test suites and a Firefox Add-On for automatically performing Selenium test cases. Your web application should be systematically tested using one of these advanced tools.

6.3 Security analysis

The team has to analyse the security aspects of the system, and make sure the application is properly protected against SQL-injection and cross-site scripting, amongst others. Each team has to write a security analysis document of 300-500 words and upload it to the GitLab repository of the project.

More information will be given during the Security lectures.

Deadline: 29 June 2019. A security analysis is uploaded to the `design` directory in the Gitlab respository.

7 Afterword

If you have gone through all the chapters and arrived at this point, congratulations! This means that either you are ready to start your project or even you have finished the project itself, because who nowadays reads a manual from cover to cover? In case you have finished the project, you have probably noticed that some parts of the project were easy, but some others were difficult or even frustrating. The installation of some software packages, for example, may have costed many hours of your sleep. The use of some software libraries force you to think like the programmer of these libraries and may require some time and effort.

> … somewhat unique to programming is that it consists of near constant failure. Unlike learning other skills where one can expect to be reasonably competent after sufficient practice, programming largely consists of constantly failing, trying some things, failing some more, and trying more things until it works. One of the biggest differences between experienced and novice programmers is that experienced programmers know more things to try.
>
> *Alicia Liu, Overcoming the Imposter Syndrome (https://medium.com/tech-talk/bdae04e46ec5)*

7.1 The myth of the 'real programmer'

In case you are worried about your own programming skills, or if you are not sure whether you are a 'real programmer', or if you think that the lecturers of this module have left you to your fate, read Alicia Liu's blog post mentioned above. A good programmer knows a couple of tricks that are worth mentioning:

1. Be patient and systematic, and try one solution at a time.
2. Read the … manual (http://en.wikipedia.org/wiki/RTFM).
3. GIYF: Google is your friend! Or Stackoverflow, or Duckduckgo (if you are paranoid about your privacy). In this manual we often use Wikipedia as an entry point for information on some subjects.
4. Explain your big problems to the other members of the team, and if they are not around, to your rubber duck (see http://en.wikipedia.org/wiki/Rubber_duck_debugging).

7.2 Module evaluation

This module is being offered for the sixth time this academic year. We collected all the feedback we received in the previous years and we tried to process this feedback in order to make the module more challenging and stimulating. We believe that feedback is the key to improvement, thus we encourage you to participate in the module evaluation.

Evaluation and monitoring will take place in different forms. Short online questionnaires will be available in weeks 4 and 8 in which the students will be asked to give their feedback about the module. A panel of half a dozen students will also have some feedback meetings with the module staff. If there is anything that requires immediate action, feel free to contact the module coordinator directly.

Appendix A Deadlines

> I love deadlines. I like the whooshing sound they make as they fly by.
> *Douglas Adams*

For your convenience, the deadlines set for the activities or deliverables described in this document are summarised in chronological order in Table 1.

Table 1 Deadline of project deliverables / activities.

Deliverable/ Activity	Deadline
All students have registered for participation in a team	Tue 23 April 2019, 19:00
Each team has indicated their preferences for the module project	Wed 24 April 2019, 12:30
All students have access to a project in Gitlab, defined in the scope of their team	Fri 26 April 2019
All students have a Trello account and can access the project board.	Wed 8 May 2019
A document `project.pdf` with project description is present in the `design` directory of your project in the Gitlab repository.	Fri 10 May 2019
A mock-up of the web interface is present in the `design` directory of your project in the Gitlab repository	Wed 15 May 2019
Sprint review 1	Wed 15 May 2019
Use case diagram(s) and class diagram (as `.jpg` or `.png`) is present in the `design` directory of your project in the Gitlab repository	Mon 3 June 2019
A file `schema.sql` containing the database definitions is present in the `design` directory of your project in the Gitlab repository	Mon 3 June 2019
Sprint review 2	Mon 3 June 2019
Sprint review 3	Tue 18 June 2019
A security analysis is present in the `design` directory of your project in the Gitlab repository	Fri 29 June 2019
Your project in the Gitlab repository contains a working system (*potentially shippable product increment*) for assessment by the module staff	Fri 29 June 2019, 23:59
Sprint review 4	Fri 29 June 2019
Final presentation to the client	Fri 5 July 2019

Appendix B Assessment criteria

> I never teach my pupils. I only attempt to provide the conditions in which they can learn.
> *Albert Einstein* (http://www.einsteinssecret.net/about-the-secret/)

In order to assess the project for the purpose of grading the module team will use the following criteria, which have been grouped in 4 main categories:

Project management

1.1 Correct use has been made of Git and Git branching.
1.2 The project team has used Scrum as described in the module's guidelines.
1.3 The deadlines for the deliverables described in Appendix A have been met.
1.3 Four presentations with demos have been given during the sprint review meetings.
1.5 Team work: all group members have contributed to Git, Trello, presentations, etc.
1.6 The team has communicated appropriately with the product owner / the client.

Design

2.1 Adequate UML use case diagrams have been defined for the system.
2.2 Adequate UML class diagrams have been defined for the system.
2.3 An adequate SQL Database design has been defined for the system.
2.4 A Representative mock-up for the application has been built with a CSS framework.

Programming

3.1 The code has been compiled using Maven to a Java Web Archive (WAR file).
3.2 The code has been adequately tested with unit tests.
3.3 Adequate use has been made of RESTful services with XML or JSON, implemented using Jersey.
3.4 Adequate use has been made of JDBC and stored procedures.
3.5 A framework for HTML, CSS and JavaScript has been used.
3.6 Concurrency: adequate use has been made of database transactions.

Project finalisation and delivery

4.1 Working system: the application can be deployed and runs on the central Tomcat server.
4.2 The functionality of the application systematically is tested with (automated) test cases.
4.3 Security issues have been properly analysed.
4.4 An appropriate final presentation with demo has been given to the client.