SEM5640: Developing Advanced Internet-Based Applications

Individual Report 2015-2016

Go Aber! - Fitness Tracking Application

Connor Luke Goddard
(clg11@aber.ac.uk)

January 5, 2016

1 Introduction

As part of the formal assessment for the SEM5640: Developing Advanced Internet-Based Applications module, the class of 2015 were tasked with providing the design, implementation and testing strategy for a new web-based fitness tracking application, designed to promote healthy living and regular exercise within pre-existing community structures including, but not limited to, academic institutions, hobby societies and sports clubs.

The supplied customer specification [1] set out requirements for a system capable of recording and managing fitness data obtained through both manual and 3rd-party data sources. In addition to allowing participants to manually enter fitness data (e.g. running steps or swimming strokes), it was specified that the system should also allow fitness data to be retrieved automatically from APIs provided by two popular activity-tracking services, namely; *Fitbit* [2] and *Jawbone* [3].

With an emphasis on encouraging increased exercise through promotion of friendly competition, the system was also expected to provide a mechanism through which rival communities could engage in challenges between one another, with all participants jointly contributing towards the total score recorded for their respective community. From a technical perspective, the customer stipulated that the final system was to be capable of supporting both the JavaEE and ASP.NET (Microsoft) platforms. As such it would become necessary, upon finalisation of the architectural and database designs, to implement two functionally-equivalent versions of the same application.

This report provides a personal evaluation into the overall successes and failures of the project, focussing discussion onto the various technical, organisational and interpersonal aspects that - the author believes - are likely to have contributed towards its final state.

2 Personal Contributions & Evaluation

2.1 User Membership & Security

In terms of the technical contributions made by the author, one of the most significant relates to the implementation of mechanisms for supporting user membership and the enforcement of appropriate access restrictions.

Taking guidance from the project brief, the work focussed on three key functional areas: the ability for participants to create, access and manage their own account, details and associated data; the ability for a user to join an existing collection of users belonging to a larger community; and the provision of controlled access to the system and its functionality based upon the authorisation level granted to a given user.

Considering the sensitive nature of the data expected to be processed by the system, it became necessary to introduce mechanisms for preventing the disclosure of such data to users for whom it did not directly concern. In conjunction with the rest of the development team, one of the first tasks undertaken by the author saw the design of a relational database architecture that would support the attribution of fitness entries back to an individual participant by means of an enforced one-to-many relationship.

In an attempt to maximise the available time for developers to begin prototyping and implementation, the decision was taken to produce this initial database design immediately following the analysis of customer requirements. This resulted in design discussions taking place *before* the author undertook research into what - if any - existing membership and authorisation support was offered by either of the target platforms. Whilst no significant problems or delays were met as a consequence, it is the case that less time could have been spent in needing to perform subsequent design changes, had all of the originally identified research been completed prior to beginning any up-front design.

Following completion of the necessary research, it was concluded that much of the required functionality could be accomplished, at least in part, through use of the standard authentication and authorisation frameworks included as part of the ASP.NET and JavaEE platforms. Whilst it would have been possible to implement a bespoke authentication and authorisation solution, this neither represented an efficient use of the available time, nor an approach that would adhere well to the spirit of what these kind of web platforms aimed to promote.

Based purely on the author's prior (yet albeit limited) experience in developing web applications using ASP technology, it was decided that development would begin first with the ASP.NET version. After investigating a number of possible frameworks for assisting in the provision of site membership facilities, the author settled on the recently-updated Microsoft Identity Framework [4].

Almost immediately after starting work on the application, the author noted a detailed level of integration between the Identity Framework and the other ASP.NET frameworks also in use at the time. With the team choosing to exploit the scaffolding feature provided by the ASP.NET MVC framework, the author later found - to their surprise - that alongside the collections of model, controller and view classes, the resultant 'barebones' MVC application also sported a full-functioned user registration and authentication system, complete with a comprehensive utilities library and supporting backend database.

Having evaluated the suitability of this pre-provided user membership system against the needs of the project, the decision was made to shift focus toward supporting a vanilla-style approach to user membership, rather than continuing with original plans to utilise organisational login as the primary mechanism for user identification. Acting in-part to exploit the auto-generated membership facilities in ASP.NET, the decision was also influenced by what at the time appeared to be, a lack of clear documentation relating to the implementation of organisational authentication for JavaEE (e.g. via LDAP Realms [5]). Although the team always intended to provide support for organisational identity providers at a later stage, due to eventual time constraints this was unfortunately never accomplished.

Whilst a large proportion of functionality relating to user membership and authentication came provided 'out of the box', a number of additional enhancements were required in order for the system to make best use of the features provided by the Identity Framework. One key example from this work involved the implementation of a new service class providing a collection of common utility and data functions relating to the retrieval of user information and management of the current session (e.g. login/logout, retrieval of user's email address etc.). By abstracting away much of the technical detail associated with the acquisition and management of the current user context, other developers were able to quickly access the necessary user data within their own development features (e.g. management of activity data, group assignment etc.) without having to worry about obtaining a direct reference to the current session, or interacting with Identity Framework directly.

At the centre of JavaEE's support for user authentication, lay the notion of *realms* - a collection of *"users and groups identified as valid users... controlled by the same authentication policy."* [5]. For this project, the author made exclusive use of the JDBC Realm, which allowed for the retrieval of user credentials from a database as opposed to a file, or via digest authentication [6].
While the Identity Framework followed pre-defined *convention* by way of auto-generating the necessary database tables, the JDBC Realm relied instead upon *manual configuration* to specify the database, tables and fields destined to hold the necessary credential information. This had the benefit of allowing complete control over the schema definition used to store the password information, and became particularly useful in situations where the author wished to insert new fields into the table (e.g. when wanting to define additional details for a user), or to make modifications

to existing table fields (e.g. updating the length limit of the user's email address).

Unfortunately, this heavy reliability upon manual configuration was to become a double-edged sword, after considerable time was lost in trying to identify the necessary password encryption settings within the JDBC Realm configuration. At the source of these issues, lay a lack of clarity within the vendor documentation between settings relating to the encryption of passwords within the database, and encryption during the exchange between server and client (digest authentication).

After suffering from great confusion over the correct values to use for the two different settings, the author eventually turned away from the official documentation, and instead followed advice from other affected individuals in order to finally establish the correct configuration. Whilst a significant amount of time was sacrificed in solving what came to be a relatively minor configuration issue, it most be noted that the circumstances leading up to this situation were both unexpected and challenging, given that the documentation itself transpired to be the cause.

2.2 Testing

Owing primarily to a failure by the team to sufficiently plan and enforce an appropriate testing strategy, the author found themselves left with only enough time to accomplish what, at best, could be described as a basic level of unit testing.

Whilst a lack of proper process structure was considered to be the main reason for this poor performance, the author knew of additional factors, such as overcoming technical difficulties and fixing discovered bugs, that also became attributed to the lack of time left available for testing activities. It was true to say that the author was certainly disappointed with the quality of testing that had been achieved. In spite of this, their work did make a considerable contribution to the overall level of testing that was achieved by the team. In addition to writing unit tests for their own components, the author invested considerable time in establishing support for mocking of real objects from within the ASP.NET testing suite.

As the author had no prior experience of working with object mocking, it took some time to fully appreciate the need to fake not only the properties of an object, but also its behaviour. Making use of a third-party mocking library (Moq [7]), the author was able to generate entire model entities on-the-fly for each individual test. This not only made it easy to customise a model with values specific to an individual test, but also ensured that testing data remained unaffected with respect to changes occurring in other tests.

Whilst the mocking framework provided many useful features that consequently saved the author considerable time in implementing their mock tests, the support it demonstrated toward the core range of LINQ extension functions was found to be severely lacking. As a consequence, the author was required to dedicate considerable time to researching how it may be possible to emulate these highly-used data functions within their unit tests.

To prevent other developers from being impacted by the same issue in the future, the author chose to abstract their findings out to a collection of generic utility functions that subsequently any test employing the same mocking framework could successfully utilise.

2.3 Email Support

With the team wishing to capitalise on having up-front access to the list of customer requirements, efforts were made to undertake advance research into unfamiliar areas that had the potential to pose the greatest challenges in pursuit of implementing the final system.

As part of their allocated workload, the author was tasked with investigating potential solutions for the provision of automated email generation and management of user mailing lists within both *Go Aber!* applications. Drawing on the author's previous experience within the field of email marketing, the author was able to recommend a third-party service (SendGrid [8]) that was known to be capable of providing the required facilities.

This service fitted well to the technical needs of the project, providing pre-built client libraries for both ASP.NET and JavaEE, with additional access to a fully-featured RESTful web API. As a result of SendGrid's tiered-approach to pricing, the author also concluded that the application was eligible to make use of the service without any financial cost to the project. This proved highly beneficial when proposing use of the service firstly to the rest of the team, and secondly to the customer.

Although this work resulted in an agreeable solution being identified, due to unfortunate time constraints its findings were unable to be included within the final implementation.

2.4 Product Owner

Having elected to adopt the SCRUM methodology [9] as the framework for managing the project development process, a proportion of team members were required to undertake additional responsibilities set out within the methodology guidelines.

Being the only member of the team to hold previous industry experience of working within SCRUM-based projects, the author was elected to become the dedicated product owner in addition to their original role as a developer. Taking sole responsibility for the management of the product backlog, the author played a significant part in driving key decisions relating to the prioritisation of work and appropriate allocation of tasks to developers.
In addition to managing the prioritisation of work items, the role also involved heading all communication between the project team and the customer representative. This would include representing the views of the customer within team meetings, and if required, approaching the customer directly with any questions, or inviting them to express their opinion on matters of particular importance (e.g. obtaining the opinion of the customer in light of the need to make crucial priority changes).

3 Group Evaluation

Having investigated the relative strengths and weaknesses across a range of potential agile approaches, the team eventually came to choose the SCRUM framework as their adopted development methodology. Key to this decision, lay a desire to maximise the team's flexibility toward sudden - and often late - changes, made as consequence of customer indecision, or following the discovery of technical constraints and/or implementation issues.

In selecting SCRUM [9] as the preferred development methodology, the team had identified particular characteristics seen to be beneficial in pursuit of supporting an adequate level of flexibility. These characteristics included an ability to appreciate that unexpected changes can arise during the development process - consequently providing a mechanism for handling these types of changes appropriately; and the provision of a structured set of development guidelines, upon which a team could adapt their own working practices to meet the specific needs of the individual project.

With respect to the choice of development methodology, the author believes that the adoption of SCRUM represented the right decision for the project, believing that the methodology played a crucial part in enabling the team to make significant progress by way of accomplishing the stipulated requirements, despite facing some difficult and time-consuming issues during the process.

It is true to state that throughout duration of the project, the types of late changes that had been originally predicted by the team did in fact prove to become a reality. Rather than being caused by customer-led alterations to the requirements, many of these late changes were in truth the result of unforeseen technical issues stemming from the team's lack of prior experience in working with the specified web technologies (ASP.NET & JavaEE). As a positive consequence of adopting an agile-based methodology, the team were in a position whereby it was possible to re-schedule upcoming work in response to delays caused by either the discovery of technical issues, or - noted unfortunately on a few occasions - the act of developers making unexpected alterations to core system components.

This highlighted one of the main weaknesses typically associated with use of the SCRUM approach, where under an ethos of allowing the team total control over the management of their own workload, there lay an increased risk of the team suffering set-backs as a result of poor decision-making on the part of individual developers. To counteract this, SCRUM necessitates a high level of communication both internally within the project team, and externally with the customer. In this respect, the team showed a high level of compliance, partaking in all 'official' project meetings (including those relating to the planning and subsequent review of sprints), and making sure to actively engage in on-going discussions either in person, or via digital channels (i.e. email and dedicated group chat room).

With all members of the team expected to undertake a developer role, it became necessary to break-down large bodies of technical work (e.g. customer requirements) into collections of smaller tasks that could be delegated out to individual developers for them to complete. This process provided two key benefits.

Firstly, it provided a regular opportunity for the team to discuss - together - the work that was upcoming over the next few project iterations. Within this discussion, developers would seek to refine and clarify their understanding of the related requirements (referring back to the customer if required), and also share knowledge and ideas of how to solve any technical aspects that were predicted to be particularly challenging.

Secondly, by viewing work as a collection of small individual activities, the team were able to look beyond the 'obvious' aims, solutions and potential pitfalls for a task, and instead gain access to a much 'finer' level of detail that would otherwise have most likely been missed. This led to the team having the capability to precisely deploy particular developer skill-sets onto tasks where they were deemed likely to provide the greatest benefit.

Whilst definite efforts were made by developers to sufficiently test their own work, unfortunately in many cases, these tests were implemented far later into the project than had originally been anticipated, and as a consequence, a lack of remaining time prevented a number of these tests from demonstrating what the author would recognise as an adequate level of code coverage. In their own view, the author attributes these issues to a team decision - taken at the start of the project - to delegate responsibility for organising their own testing out to individual developers.

There were occasions towards the end of the project where developers may have felt driven to neglect some of their testing responsibilities, in order to instead focus on fixing important errors or completing necessary features. This of course represented a less-than-ideal situation, and leaves the author feeling that a more structured testing approach such as Test Driven-Development [10] or Behaviour-Driven Development [11], may have represented a better overall choice.

Given more time, the author would also have wished to introduce stress testing to become part of the overall testing strategy. With the project focussing exclusively on the development of web applications, it appeared to be very sensible - if not in fact essential - to have means for verifying application stability under the types of exaggerated conditions that such a system could feasibly be exposed to (e.g. peak user demand, poor network connectivity etc.).

Despite the lack of sufficient unit testing, the team did manage to implement a comprehensive suite of acceptance tests that focussed on assuring that the implemented applications adhered correctly to the customer's original requirements.

In collaboration with these tests, the team also made a concerted effort to integrate manual acceptance testing into the review process for pull requests submitted to the version control system (Visual Studio Online [12]). By expecting other developers to undertake testing at the point of code integration, it was hoped that they may be able to identify additional issues that the original author may have missed through their own individual testing, thereby reducing the risk of diluting overall system quality by way of introducing poor-quality 'smelly' code.

4 Conclusion

In considering all aspects of the technical implementation, project management and completed deliverables, it is the author's view that the team have shown themselves to be fully capable of implementing a non-trivial web application, not just in one, but two separate - and often contrasting - web technologies.

As with any development project, there were of course issues to overcome and areas in need of improvement. In this particular case, it proved to be time that represented the team's greatest adversary, and testing - representing their weakest performance.

With this said, the team demonstrated great cooperation and support for one another, incorporating the the appropriate organisational processes set out by their choice of development methodology. Narrowly missing the completion of all stipulated features, the author remains confident - given just a little more time - that the project could have resulted in the delivery of two full *Go Aber!* applications, standing ready for deployment within a real-life community setting.

References

[1] Neil Taylor and Nigel Hardy. *SEM5640 Group Project Go! Aber Requirements Specification.* Aberystwyth University, October 2015.

[2] *Fitbit - Official Site for Activity Trackers & More.* `https://www.fitbit.com`, 2016.

[3] *Jawbone - UP Fitness Trackers.* `https://jawbone.com`, 2016.

[4] Pranav Rastogi, Rick Anderson, Tom Dykstra, and Jon Galloway. *Introduction to ASP.NET Identity Framework.* Microsoft Developer Network, October 2013.

[5] Eric Jendrock, Ricardo Cervera-Navarro, Ian Evans, Devika Gollapudi, Kim Haase, William Markito, and Chinmayee Srivathsa. *Working with Realms, Users, Groups, and Roles - The JavaEE 6 Tutorial.* Oracle.

[6] Eric Jendrock, Ricardo Cervera-Navarro, Ian Evans, Devika Gollapudi, Kim Haase, William Markito, and Chinmayee Srivathsa. *Digest Authentication - The JavaEE 6 Tutorial.* Oracle.

[7] *Moq: An enjoyable mocking library.* `https://www.nuget.org/packages/Moq`, October 2015.

[8] *SendGrid - Delivering your transactional and marketing email through one reliable platform.* `https://sendgrid.com`, 2016.

[9] Ken Schwaber and Jeff Sutherland. *The Scrum Guide*, 2001.

[10] Martin Fowler. *Test Driven Development*. `http://martinfowler.com/bliki/TestDrivenDevelopment.html`, March 2005.

[11] Dan North and Chris Matts. *Introducing BDD*. `http://dannorth.net/introducing-bdd/`, March 2006.

[12] Ed Blankenship. *Visual Studio 2013 : Introducing Visual Studio Online*. `https://msdn.microsoft.com/en-us/magazine/dn519923.aspx`, January 2014.