Digital Justice project

IT aspects

Initial scope-setting

The technical project began with the creation of a set of loosely-defined goals for what students might achieve by the end of the project. These helped to set the initial scope for the project, as well as acting as a way to check that the project was still on-course during its lifetime.

It was agreed that students should:

1. Experience the challenges of exploiting technology to provide legal help
2. Learn about the opportunities that technology in the workplace can bring
3. Gain awareness of the challenges and issues involved
4. Learn to identify useful legal services that can be delivered through an app
5. Acquire some technical and project management skills
6. Design and build a useful app that enables self-help in an agreed area of law
7. Share real-world experience with other students

The technology aims of the project were to enable students to identify and analyse a legal problem, elucidate and design a valid, comprehensive solution to it, and then use technology to deliver this solution effectively.

The following set of guiding principles was used to ensure a consistent approach to delivering the project:

1. Scalability: the programme must ultimately be deliverable to the whole W360 cohort
2. Currency: information displayed by the app must be up-to-date
3. Independence: Content must be independent from presentation
4. Portability: Information must be portable between repositories
5. Quality: the app code should be of an industry-acceptable standard
6. Flexibility: the app must support changing requirements without significant impact
7. Cost: the app must be deliverable within a minimal budget

These principles helped to shape the choice of technology, to set some limits on the work that we would expect students to do, and to define the way that the project would be run.

It was decided to run the pilot within the Law faculty only, while keeping in mind that the project would be an opportunity to think about how it might be extended in future to create opportunities for interdisciplinary collaboration between Law students and others in Computing & IT, Design, etc.

Team structure and process

Students were organised into teams to facilitate management of the work they had to do. Having considered a number of different options and configurations it was decided that to group participants into teams in which the members worked together to contribute ideas and carry out separate pieces of work that were brought together to create a single app per team. It was thought that a key limiting factor would be the available time that the team manager (tutor) would have available to support students working on apps, so this model minimised the number of apps to be managed.

The members of each team followed a simple process that guided them through defining requirements, understanding how these might be met in principle, and then applying the available tools to create an app-based solution. This was a repeatable process so that the apps created evolved through a number of iterations across the time available. While the first iteration or two inevitably required more tutor guidance, the team developed an understanding of how the process worked and therefore become more self-sufficient over the course of the project.

Delivery methodology

The delivery process was loosely based along Agile principles. As well as giving experience of a real-life software development methodology, this approach ensured a project structure that was predictable but that also maximised the ability to handle new features and priorities that the students might discover during the project.

The project was divided into a number of *phases* corresponding to fixed-duration time boxes. Each team maintained a *backlog*, i.e. a list of the tasks that they would like to execute, such as *learning activities* in which they learned how to use the tools available, *research activities* where they source

information that they intend to incorporate into their app, or *development activities* such as the actual editing and creation of app content.

The *budget* available to each team was the number of hours that the team members have available.

The intended approach was that each phase would follow the same process:

1. The team would review their backlog, which would be empty in the first iteration:
    a. New items would be added to the backlog – these might be novel or may be changes to existing app features suggested by experience from the previous phase; "there are no bad ideas";
    b. Each item on the completed backlog would be assessed and prioritised and the backlog would be ordered (highest priority first);
    c. The budget would be apportioned across as many items on the backlog as the team agreed was realistic – these would form the scope for the forthcoming phase;
    d. Tasks would be allocated to individuals or to groups within the team and the scope for this phase locked down. Any follow-up input from non-attendees must be added to the backlog for consideration in the next phase.
2. The team would carry out the agreed tasks until the phase deadline;
3. Progress would be reported to the whole team on a regular basis;
4. At the end of the phase a progress report of updates from each team member would be assembled and used as input to the backlog review of the next phase.

The project followed this approach to a large extent, though not rigidly. In practice, it was difficult to assemble all members of all teams in real-time, and so it was necessary to complement the scheduled activities with asynchronous alternatives such recordings of online meetings held and online forums. In addition, the time was spent on discussing the kind of work the students wanted to do in the next phase, rather than defining sets of rigid requirements.

Meetings were held online in an Adobe Connect Room, always recorded, with access only to the team members. Follow-up (e.g. ideas to be considered in the next phase backlog review), team communication and progress reports were shared on a private online forum. Team members also organised communication between themselves via email and by setting up WhatsApp groups.

Students were free to explore ideas offered by the tutors and to come up with their own so long as these met the over-arching project goals, with the tutors advising or vetoing ideas put forward when appropriate. Some mandatory activities were also incorporated into the schedule, such as briefing sessions from the team manager at the start of each phase in which the high-level goals and success criteria for the phase were agreed, as well as learning sessions where new tools were introduced to the students.

Timescales

The app project was planned to start in mid-December, finishing by early April. The project scope was set such that it would require around 2 man-weeks (80 hours at 100% effort) to complete, on the assumption that as an optional “spare time” project, students would not spend more than 1 hour each per week on the project, so this effort would be met by a team of five over a duration of 16 weeks (5 x 16 hours = 80 hours).

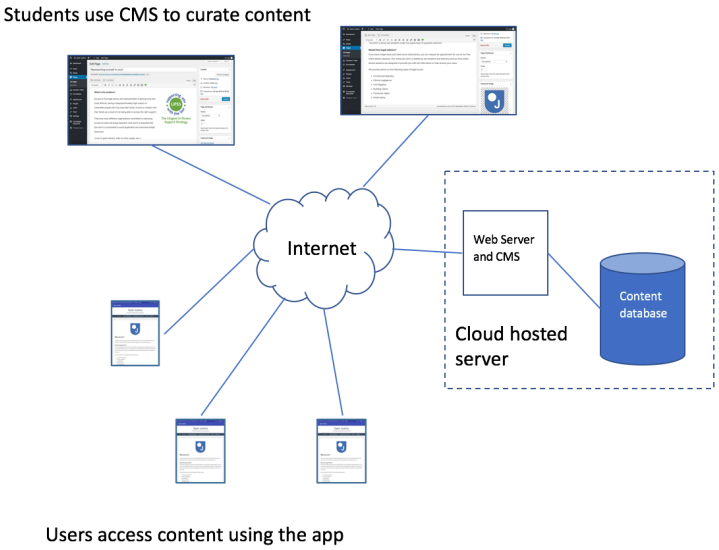
Clearly the three factors of scope, effort and duration could have been flexed in many ways - students could have been asked to spend 2 hours per week to deliver twice the scope, the scope could have been halved to reduce the duration to eight weeks etc.

The schedule for the project followed four phases of four weeks each. Although each phase followed the same framework outlined earlier, each had a particular natural emphasis on "Familiarisation", "Implementation", "Refinement" and "Reflection" respectively.

Technical Architecture

In line with the guiding principles of Scalability, Quality and Flexibility, a Client-Server technical architecture was chosen. Supported clients included standard web browsers, as well as a custom-built lightweight mobile app. The server component was an industry-standard three-tier back-end comprising a web server, a component for designing logical flows (using decision trees) and a relational database to hold the pages presented and the data that defined the decision trees.

The app was deployed on Android devices for the pilot project to avoid the additional approval and licencing requirements required when apps are to be distributed through the iTunes app store to Apple iOS devices. This would be possible to do at any point in future, and as a proof-of concept in this pilot the app was published in Android and iOS formats through an emulation platform called gonative.io.



Key components

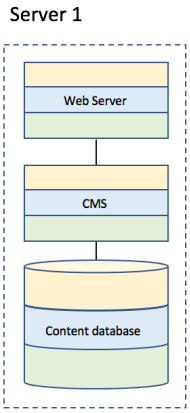
App
Software installed on a user device that displays content provided by Open Justice.

Web server
Handles requests from the app for content, which it retrieves from the content database.

Content Management System (CMS)
Used to manage and design the content that is displayed by the app. Accessed from any web browser; individual accounts with appropriate privileges were created for different editors. Students curated the app content using this CMS: the skills required are commonly available and are easy to acquire – no technical knowledge is needed.

Content database
Stores all of the local content that can be published to the app for display. Not directly visible to the editors or non-technical users.

Hardware



The project used a "shared-everything" design, comprising a single server with one instance of the web server hosting a separate virtual instance for each student team, and a single instance of the CMS and database that each contain a separate area for each team.

This approach simplifies administration, making it easy to set up new teams, sites and users and is also the most cost-effective as it maximises storage and processing capability by sharing these across all teams. It can scale both vertically (increase the server size) or horizontally (add more servers to the virtual cluster).

To meet the flexibility and scalability principles, it must be possible to both grow the environment to support as many students who choose to participate, which could be anywhere between five and five hundred individuals, but also to shrink it so that only the required resources are being paid for. This requirement was met by using an elastically-scalable virtual server platform hosted on the Amazon Web Services (AWS) cloud.

The server for the pilot project used 512MB memory, 1 CPU, 20GB storage, 1TB outbound data transfer, a standard specification for many cloud-provisioned servers such as the entry-level AWS Lightsail[1] product. Usage observed during the project suggested that this server would support up to fifty non-concurrent students on the non-commercial licencing basis described at an average monthly cost for hardware and software of under £10.00.

Software

Open source software products were chosen for all of the components required. This helped to minimise the costs of the project, as non-commercial licences were used, and carried the additional benefits that students might already have had experience using some of them, or would gain experience that would be useful in future.

The website deployed on the server was realised by using the combination of an operating system (Ubuntu 16.04.4 LTS Linux), web server (Apache 2.4.18), content management system (Wordpress version 4.9.8) and database (MySQL Ver 14.14, Distrib 5.7.23). Other equivalent compatible options could have been chosen to fit the standard offering/support capability of the provider had this been required.

Rather than creating many servers or many instances of the CMS on a single server, a *multi-site approach* was used whereby multiple websites are created within a single instance of the CMS. Each team was allocated a virtual server that could be accessed using the corresponding subdomain, e.g. team1.ojapp.org.uk, team2.ojapp.org.uk etc. This enabled complete flexibility in terms of the underlying configuration, which could be expanded and re-organised without affecting either the student teams or the end-users of the apps.

The default features within the CMS were customised so that students had access to editorial settings (create menus, pages, decision trees, forms) but not to administration functions (add users, change site settings).

Additional components ("plug-ins") were provided so that students could develop more complex functionality within their apps than would be possible by using simple webpages and navigation features such as menus and links. These included a component for constructing and displaying Decision Trees (Decision Trees[2] 1.1) and others for creating Forms (Formidable Forms[3] 3.04.02, FormidablePRO2PDF[4] 2.90) that end-users could complete to submit information to the team.

Additional components could be added using this approach as new ideas develop. For example, the forms capability might be extended to create "wizards" that prompt for information that is used to auto-populate and generate official forms, which otherwise are complex for end-users to navigate and complete.

[1] https://aws.amazon.com/lightsail/pricing/

[2] http://sidecar.tv/

[3] https://formidableforms.com/

[4] https://www.formidablepro2pdf.com/

It was decided that students should keep a number of design considerations in mind when creating their solutions, though these were not formally assessed. The key requirements were that the apps must be *accessible* to users with additional needs and *usable* by all: clear and easy to navigate.

To support these design goals, the base sites were configured to be responsive (i.e. the content displayed automatically adjusts to match the physical size and orientation of the device in use) and were customised using a "minimalist" Wordpress theme (Bhari 1.0.4.9) with sidebars disabled so that the maximum amount of screen real-estate is available for displaying app content.

Considerations for future iterations

Additional focus on legal design

The initial pilot was concerned with analysing a legal problem, finding a solution and using prescribed tools to make this solution available. Some indirect introduction was also given to other aspects of the software development lifecycle, including project planning, requirements management and testing.

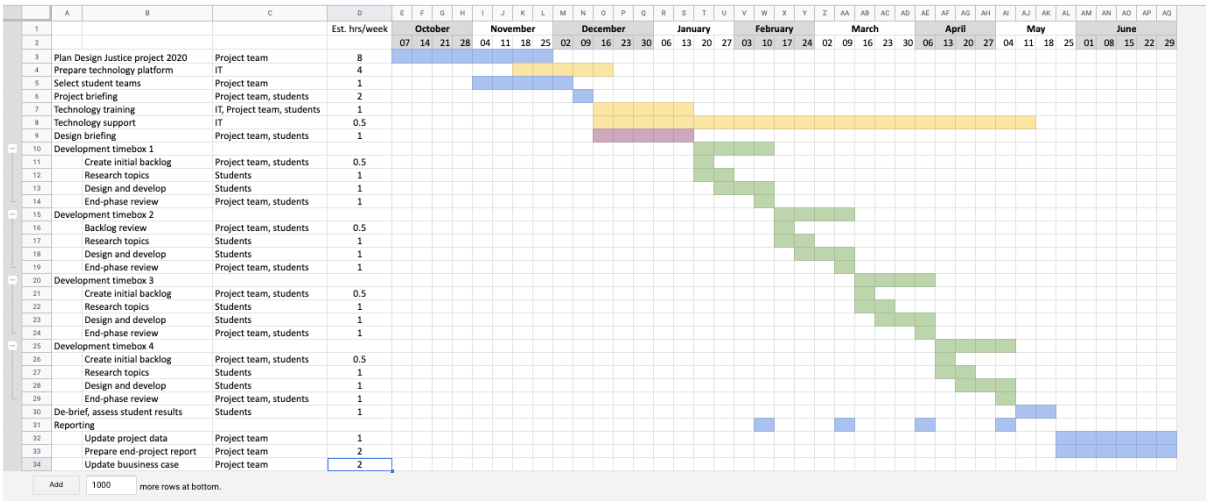
Future pilots should extend these topics to include legal design – both from the perspective of the content (clarity, how easy the information is to understand) and the presentation (attractiveness, usability, accessibility, user interface/user experience). They should also integrate the design and IT aspects to show how these work together with the legal content to form a holistic solution.

A potential syllabus for the next iteration might look like this:

- Legal Design
    - What is Legal Design ?
    - Components of a legal technology solution
    - Technical architectures, legal process analysis
    - Using technology to improve productivity in the legal profession
    - Using technology to improve access to justice in the legal profession
- Design Thinking
    - The basics of user experience: usability, visual design, navigation and menu design, and accessibility
    - How to design for efficiency, persuasion, satisfaction, clarity
    - Problem analysis and solution design
    - Personas in design-thinking
- Software development methodologies
    - Waterfall vs. lean/agile/iterative
    - Methodology in practice - the Digital Justice project framework
    - Generating and managing requirements
    - Planning: time-boxing and prioritisation
- Bringing design and development together
    - Integrating human-centred design and agile development
    - Making the business case for user-centred design
    - Team-working within user-centred design
- The Digital Justice technology platform
    - Information Architecture - what is it ? Principles and application
    - Using the Digital Justice platform to create innovative solutions

- Assessing product quality in each iteration

This could be developed across the academic year as follows:



		Est. hrs/week
Plan Design Justice project 2020	Project team	8
Prepare technology platform	IT	4
Select student teams	Project team	1
Project briefing	Project team, students	2
Technology training	IT, Project team, students	1
Technology support	IT	0.5
Design briefing	Project team, students	1
Development timebox 1		
Create initial backlog	Project team, students	0.5
Research topics	Students	1
Design and develop	Students	1
End-phase review	Project team, students	1
Development timebox 2		
Backlog review	Project team, students	0.5
Research topics	Students	1
Design and develop	Students	1
End-phase review	Project team, students	1
Development timebox 3		
Create initial backlog	Project team, students	0.5
Research topics	Students	1
Design and develop	Students	1
End-phase review	Project team, students	1
Development timebox 4		
Create initial backlog	Project team, students	0.5
Research topics	Students	1
Design and develop	Students	1
End-phase review	Project team, students	1
De-brief, assess student results	Students	1
Reporting		
Update project data	Project team	1
Prepare end-project report	Project team	2
Update busuiness case	Project team	2

Add 1000 more rows at bottom.

Extend project reporting

As a practical project, the emphasis should be on the output – i.e. the content and quality of the apps produced. However, it seems that there is an opportunity to gather data on a number of areas that should not be missed. These include how effectively the project delivery process works, how the students experience taking part in the project, how much of a challenge (intellectually and practically) they found it etc. etc.

Governance/Editorial control/QA

Because legally sensitive advice may be presented through these apps, it is likely that some editorial control or approval process over the student-generated content will be required. This implies an additional staging environment where content can be reviewed by legal professionals before being published into the "live" apps. Further work would be required to design the process and platform to support this.

Public or private app ? Continuous development or a new app each year ?

Consideration should be given as to whether the output of the Digital Justice project is to be released to the public as a service or just used as a learning exercise. This is a critical decision that carries many implications. If the former, then the second question is resolved – the output needs to be treated as a product. This means that a level of rigidity should be introduced – for example it will be necessary to create some form of product strategy, to introduce design standards, to introduce higher levels of professional verification and scrutiny etc.

It is more likely that a public app or website created through over the course of multiple academic years will be more useful in terms of the number of features, the depth of information, the breadth of topics covered or other measures. This places constraints on the topics that students might work on, and as time goes on this will become ever-more restrictive.

The suggested project plan above could accommodate either of these approaches.

App distribution and management:

Again, the issue of whether the output is to be made available to the public raises a number of questions regarding the technology that is used. It makes it more difficult to avoid building an app for the Apple iOS operating system, which brings additional effort to run the process of building a second app framework, submitting it to the Apple, Inc. approval process and managing any technical maintenance releases that they demand in future.

Currency of information

If a public app is built, the data in/on it must be kept up to date. A solution in which the content is static (i.e. that the information is part of the client) will require new versions to be built and published when information changes. This may have liability implications if legal advice that is out-of-date is followed by a user, without the necessary warnings.

An alternative is to continue to use the pilot approach, where the information is maintained centrally instead of on the user's device, and is downloaded in real time when the app is used. This has the downside of not being available offline.

Support services required

If the app is made public, it is not possible to predict the number of concurrent users that it might attract. If this was large (even temporarily) then a centralised solution would have to be designed so that it scaled to accommodate the demand. Clearly this is not an issue if the data is held on the users own device.

In addition to technical designs that scale appropriately, the provider of the underlying technology would need to provide support services including but not limited to the following:

1. **Capacity management**: monitoring, measuring and ensuring that the platform is appropriately sized for the demand placed on it;
2. **Change management**: installation of upgrades, software patches and fixes while maintaining agreed levels of availability; backups and restores as agreed;
3. **Incident management**: specifically provision of a helpdesk to ensure that users can report problems, be made aware of any ongoing incidents and informed when these are likely to be resolved;
4. **Access management**: creation and deletion of new team sites (webserver and DNS administration) and user accounts (content management system administration).

These would be expected to be documented in an appropriate service agreement, along with the corresponding service levels agreed.