University of Glasgow | School of Computing Science

Molecular Methods Web Application

Angelos Constantinides
George Popa
Iulia Popescu
Matt Frost
Paul Dalziel
Petar Lishov

Level 3 Project — July 26, 2017

Abstract

This project aims to produce a teaching and revision web application to be used within the University of Glasgow School of Life Sciences by students wishing to enhance and assess their knowledge of the Molecular Methods course. The application was developed using the Django Web Framework and deployed on the School of Life Sciences web server. This piece of software is comprised of two main sections, Labs and Revision, each containing both electronic materials (i.e. lab manual, videos) and digital tools (i.e. converters, restriction mapping exercise, quizzes) aimed at improving the understanding of the topics taught within the Molecular Methods course. In addition, the application also provides administrative functionality through an admin interface, to be used by the course coordinators to edit the existing resources, as well as adding new ones. Therefore, the web application will be able to incorporate changes made to the course content and structure.

The application is available from molecularmethods.clinmed.gla.ac.uk.

Education Use Consent

We hereby give our permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

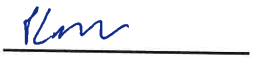
Name: Matthew Frost Signature: 

Name: Iulia Popescu Signature: 

Name: GEORGE POPA Signature: 

Name: Angelos Constantinides Signature: 

Name: Paul Dalziel Signature: 

Name: Peter Lishov Signature: 

1

Contents

1 Introduction **5**

1.1 Motivation 5

1.2 Background 5

1.2.1 Previous work 6

1.2.2 Existing Applications 7

1.3 Aims 7

1.4 Dissertation Outline 8

2 Requirements **9**

2.1 Initial Requirements 9

2.1.1 Feedback Forms 9

2.2 Requirements Gathering Process 10

2.3 Functional/Non Functional Requirements 11

2.3.1 Functional Requirements 11

2.3.2 Non-Functional Requirements 12

3 Design **13**

3.1 Goals and Considerations 13

3.2 Student Interface 13

3.2.1 Prototyping Initial Designs with Wire-frames 14

3.2.2 Key Features 15

3.3 Administrator Interface 18

3.4 Data Storage 18

4 Implementation **19**

4.1 Framework and Tool Choices 19

4.2 Working with Django 20

4.3 Front end 21

4.3.1 Animations 21

4.3.2 Asynchronous searching 22

4.3.3 Application Responsiveness 22

4.4 Middleware 23

4.5 Back end 27

4.5.1 Database Management System 27

4.5.2 Database design 27

4.5.3 Database population 28

4.6 Admin interface 28

4.7 Improving User Interface Performance 29

4.7.1 Lazy Loading 30

4.7.2 Content Delivery Network 30

4.8 Deployment 30

5 Evaluation **31**

5.1 Testing and Deployment 31

5.2 User Evaluation 32

5.2.1 Feedback Forms 32

5.2.2 Student focus group 35

6 Conclusion **36**

6.1 Summary 36

6.2 Project Management 36

6.3 Future Work 37

6.4 Reflection 37

Chapter 1

Introduction

1.1 Motivation

Molecular Methods is a Molecular Biology lecture and lab-based course given to all third year biology students in the School of Life Sciences at the University of Glasgow. The course was first developed several years ago for students undertaking a degree in Molecular Biology or Genetics. In more recent years, there was a recognition that the course provided knowledge and techniques which were considered essential across all disciplines in biology. Therefore, it was decided that all students aiming for biology degrees would be required to take this course during level 3.

The course is taken at various times each academic year by approximately 400 students undertaking a diverse mix of biology subjects such as: Molecular Biology, Pharmacology, Immunology, Biochemistry, and Sports Science. Many students find various concepts introduced in the course difficult, particularly when Molecular Biology is not their main degree subject.

The Molecular Methods App was initially developed by a previous Computing Science group in 2013-14. Further development was required based on student feedback and new functional requirements introduced by the project's clients, Dr. Pam Scott and Dr. Nicola Veitch from the School of Life Sciences.

1.2 Background

Molecular Methods aims to teach various basic molecular biology techniques and, although students will have studied the basics of this course previously, they have often forgotten this material. As a result, the course coordinators have noticed students often struggle with some of the concepts, and are looking for a tool to help them learn and revise the content within the course.

1.2.1 Previous work

As this project is built upon a previous year's project, it was necessary to evaluate the existing implementation in order to improve the understanding of the project tasked with. As a result, it was found that certain aspects of the application were already well implemented and could be re-used or adapted. A good example of this is the Restriction Mapping section from the previous application. Restriction mapping is the process within the field of Molecular Biology dealing with the obtaining of structural information on a piece of DNA using restriction enzymes. [21] A restriction map is a map detailing known restriction sites within a sequence of DNA. Given the results of these maps, information can be inferred from the DNA sequence. The standard problem involves getting the results of these splits and determining which section was cut by which enzyme. Within a Restriction mapping problem, the process of combining results and displaying a final solution was implemented in Python by last year's project group. The pre-existing code was deemed well written and highly functional, so, with prior permission, it was included in the current system.

In the evaluation of the already existing application, some other aspects were found to be poorly executed and needed to be improved upon. Once such example was the colour scheme and overall design. This included some issues with scaling when using the application on certain devices. An example of this can be seen in Figure 1.1 - when viewing the application on a desktop or laptop screen, the scaling issues are illustrated by the empty space at the bottom. It was decided a complete re-design was necessary, ensuring full scalability and incorporating the University of Glasgow official colours.

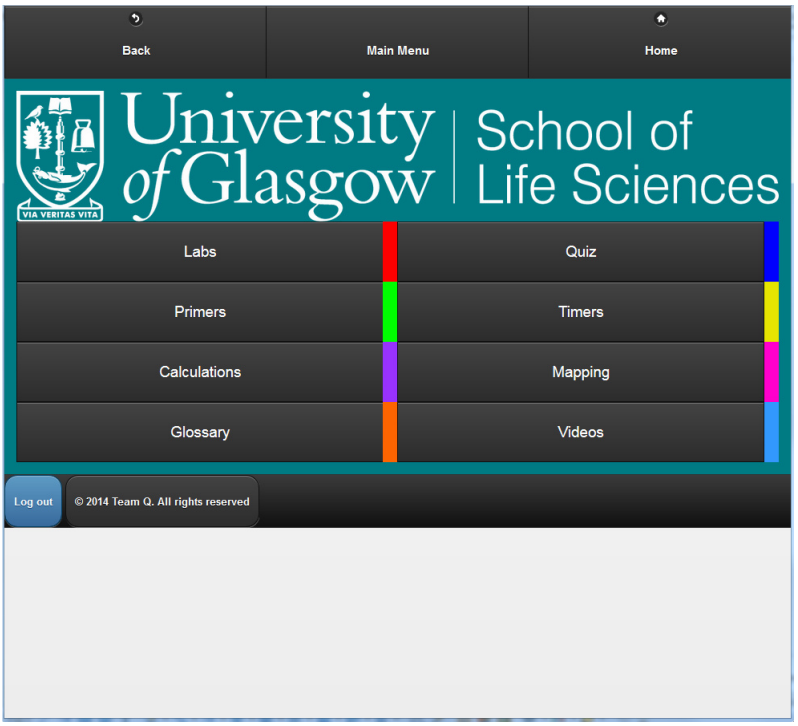


Figure 1.1: Screenshot of the home screen from previous year's application

1.2.2 Existing Applications

When engaging with the development of an online learning tool, it was decided the team should review other applications similar in scope. A good example of a learning tool for Molecular Biology is the Promega application [7]. Although Promega is a downloadable Mobile application, quite different from a web-based implementation, it is a useful illustration of a biology-based learning tool.

In this system, the user is able to view subject related videos and animations, as well as review various Molecular Biology based protocols and processes. There is also a small section dedicated to listing definitions of terms. Bearing Promega in mind, such features were included into the application's development.

In Figure 1.2, you can see that this application allows navigation via icons on the bottom of the screen and that each section is expandable.

Figure 1.2: Screenshots of the Promega mobile application

1.3 Aims

The team's aims included extending the functionality of the previous web application by adding a revision section, where students can test and enhance their knowledge through quizzes, videos, and tutorials. Another aim was to redesign the user interface such that the content of the application is split into two main sections: labs and revision. The course coordinators would be able to add or change the content through an admin interface and also monitor the amount of time students spent using the various functionalities of the tool.

During the first client meetings and focus group, the following aims were identified for the developed application:

- Organise the content of the website such that students can access the all aspects of the application on every page, removing the need of browsing backward and forward.
- Adapt the design to scale properly on multiple devices with different screen sizes.
- Add a revision section that will contain four different topics: PCR and Primer Design [3], Restriction Mapping and Data Calculations; each topic will include YouTube videos, quizzes, and examinable materials available in electronic format.
- Extend the calculations page, such that students can check their results with specific formulae and add converters for mass, volume, concentration, and dilution from stocks.
- The Labs section will be improved by adding six topics consisting of PCR, Litigation and Transformation, Blue/White Screening, Plasmid miniprep, DNA Sequencing, Real-time quantitative PCR, as well as including Intended Learning Outcomes for each laboratory.
- Add an admin interface, so course coordinators will be able to keep the content up to date.

1.4 Dissertation Outline

The dissertation outlines in detail the development process of the Molecular Methods App project. Below is an outline of the report:

Chapter 2 outlines the functional and non-functional requirements for the application and discusses the requirement gathering process.

Chapter 3 discusses various aspects of the design process for the application, including the initial design ideas and the final design.

Chapter 4 discusses the implementation of the project, the frameworks, and technologies used through-out the development of the application.

Chapter 5 discusses the how the application was tested and the evaluation process carried out.

Chapter 6 provides a summary of the project, team organisation and future work.

Chapter 2

Requirements

This section details the requirements gathering process for the project. Due to the fact that the system was requested by external clients, the requirements gathering carries great importance, as the team needed to ensure all of the clients‘ needs are met.

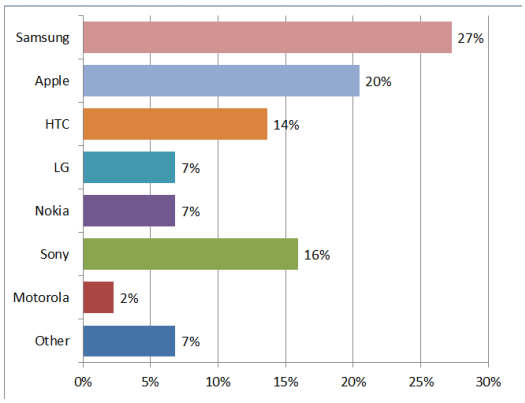
2.1 Initial Requirements

A list of initial requirements was drafted based on the initial project specification and discussed among the team, even before the first client meeting. As several pieces of information were missing, assumptions had to be made with respect to the design. These allowed the team to get a better understanding of what the project was about and provided insight into how it should be approached. They were also discussed with the supervisor to ensure the original project description was followed. During this period, the team started identifying the tools required for implementation and began familiarising with them.

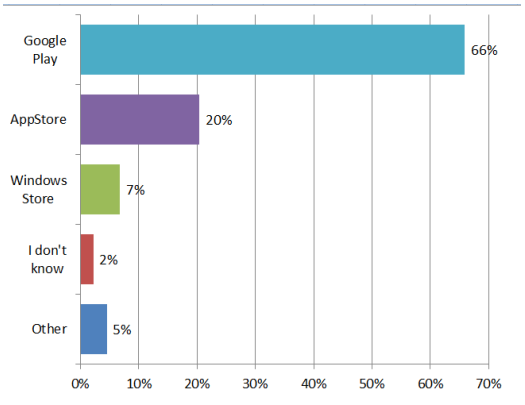
2.1.1 Feedback Forms

The requirements gathering process was started by analysing the data collected by the School of Life Sciences regarding the existing application. Research into how students were accessing the system provided interesting results: both laptop and mobile access were fairly represented, with a slight prevalence toward the mobile side. Desktop and tablet use was limited. The purpose of the use was also recorded, showing that the majority of students preferred to utilize the system to assist them in revision, rather than using the application during laboratory sessions. Additional data collected from the feedback forms indicated which parts of the application were under heavy use and which parts were found to be unsatisfactory.

A short online questionnaire was distributed in order to determine the dissemination of mobile phone brands among students. Based on the spread identified, as shown in Figure 2.1, the team and the clients came to an agreement that implementing several mobile applications would not be viable for the scope of the project, with respect to the temporal constraints imposed.



(a) Which make of tablet / mobile phone do you mainly use for internet browsing?



(b) Where do you go to download your applications?

Figure 2.1: Results of the mobile platform usage survey

2.2 Requirements Gathering Process

The team was involved in weekly meetings with the clients; specifically lecturers from the School of Life Sciences. Dr Scott and Dr Veitch clarified the initial assumptions and provided additional requirements. These included, but were not limited to:

- optimise the existing system's performance;
- fix several scaling issues on desktop platforms;
- change the functionality of some of the features;
- provide functionality for an administrative system;
- integrate statistics to observe website access.

A one-hour focus group with six students who had used the existing application was held on the 22nd of October 2014 to gather their opinion. With prior agreement, this focus group was recorded, in order for the team to confirm their findings. It was discovered that a more robust system was desired; multiple improvements to the interface to enhance user experience. Students also had a positive attitude towards the idea of quizzes, as they believed it would greatly enhance their understanding of the course. The features that the students found the most important were given a higher priority, as those were the most relevant in their lab and revision work. Scaling was also identified as a problem for desktop and laptop users. Mobile phones, however, had issues with loading times. Students also noticed some inconsistencies in the colour scheme of the current system.

The subsequent client meetings were carried out as a part of the iterative development process. Each change was presented and discussed with the clients, altered, if necessary, then integrated into the system. Client feedback was paramount, as it allowed us to make adjustments to the content.

When the desktop version was completed and hosted on the School of Life Sciences server, a second one-hour focus group was organised with several lecturers and demonstrators of the Molecular

Methods Course, this time, on 4th of February 2015. This was to ensure the correct functionality of the website and to finalise the design for the mobile and tablet version. The key point for the portable version identified in this meeting was usability: user-friendliness and performance.

The requirements for the administrative part of the project were clear from the start. There was the need for a system where lecturers and demonstrators could log in and modify the content being displayed on the website. This included editing, adding, and deleting both questions and glossary terms, and altering the lab and revision material. A method to track users was also requested, the staff being particularly interested in how the learning tool is used, what pages are being browsed at what time, and what platform it is being accessed from.

2.3 Functional/Non Functional Requirements

During the requirements gathering process, the following functional and non-functional requirements were inferred.

2.3.1 Functional Requirements

The requirements relating to Students accessing the application:

Register: students must be able to create an user account and use it to login.

View lab material: students must be able to navigate through the labs, as this will provide a better understanding of what is required within a given lab session.

View revision material: students must be able to view the revision material, as this will assist them in their revision.

Answer quiz questions: students must be able to browse through and answer the questions in a quiz. This will allow them to test their knowledge.

View Restriction Mapping Exercise: students must be able to view the problem statement and, eventually, the answer, as this will allow them to practice their restriction mapping skills.

View videos: students must be able to view the video tutorials present in the application, as this will provide assistance in the labs.

Use calculation tools: students must be able to use the calculators and converters, as this will allow them to be more efficient in their work.

Browse glossary terms: students must be able to browse the glossary entries, as this will allow for straightforward retrieval of the definition of any unfamiliar term.

Use timer: students must be able to employ the timer in their lab work, as this will facilitate their timed experiments.

The requirements relating to Teaching staff acting as site administrators:

Modify content of lab and revision pages: staff should be able to change the content being displayed on any of the lab pages, as this allows for convenient updates, should any of the course material be changed.

Modify, add, or delete quiz questions: staff should be able to update the question pool, as this facilitates consistency between the Moodle quizzes and those on the system.

Edit Restriction Mapping Exercise: staff should be able to modify the Restriction Mapping Exercise, as this allows them to easily alter its difficulty.

Add or remove videos: staff should be able to add or remove videos as desired, as this allows for the application to be kept abreast of new or additional content.

Modify, add, or delete glossary terms: staff should be able to update or add new glossary definitions, as this allows for modification of glossary entries, should any of them be redefined.

Monitor student usage: staff should be able to visualise statistical information about the website, as this allows for observation of student usage patterns.

2.3.2 Non-Functional Requirements

The non-functional requirements are as follows:

Educational. The main purpose of the application is to assist students undertaking the Molecular Methods course in their studies. As such, the application should serve as a learning tool and facilitate study.

Intuitive. It is paramount for the application to be straightforward to use and not pose any difficulties, no matter how inexperienced users may be.

Cross-Platform. As the application will be accessed from a multitude of platforms and operating systems, it needs to function well on all major platforms. Based on device screen resolution, the user interface must also dynamically rescale.

Performance The application should minimise the response times for user requests to display pages, play videos, process quizzes, perform calculations and conversions.

Scalable. As the course is undertaken by more than 400 students, it is expected that the application should be able to handle multiple concurrent user requests with no disruption to their experience.

Chapter 3

Design

The following section covers the design process of the application. The design of the application was of great importance, due to the fact that our clients were very strict in terms of design decisions and the application was going to be used as a learning tool by the biology students.

3.1 Goals and Considerations

Below the main goals and considerations of the project are listed:

Manageability. The teaching staff should be able to manage the content of the web application at any time.

Performance Scalability. Multiple students will access the application at the same time, during their lab sessions or revisions. A major objective is to design the application in such a way so that it can respond to multiple user requests as fast as possible.

Reliability and Security. Even though there is no sensitive information apart from the users' email (needed for registration), all user data is safely stored in the database and unauthorized access is not allowed. The application should be reliable, as well and respond promptly to any user request.

Clean, simple, easy to use interface. Since there are both experienced and inexperience users, the application should provide an easy to use interface for all.

Improve the students study efforts during revision and lab sessions. The application should enhance students' understanding of biology terms that are used in the labs and, moreover, help them when revising at home.

3.2 Student Interface

This section details some of the key aspects of the final design and a brief discussion of some of the factors that influenced the design decisions.

3.2.1 Prototyping Initial Designs with Wire-frames

Once initial design goals were established with the clients, the team began prototyping potential designs with the use of wire-frames. Computer generated wire-frames offered and an effective method to compare potential student user interface designs and allowed for experimentation with various design configurations, as show in Figure 3.1. This method of design prototyping was particularly suited for this project, as it facilitated a very high level of client engagement. Clients were able to provide feedback and review early design ideas when being presented several design prototypes. The clients were then able to easily visualise different interpretations of the requirements, allowing the initial design process to be very fluid, giving them the opportunity to select and combine desired elements from the wire-frame prototypes.

As a result of the close collaboration with the clients during this phase, the team was able to further refine the design requirements and rapidly progress to the implementation phase of the project.

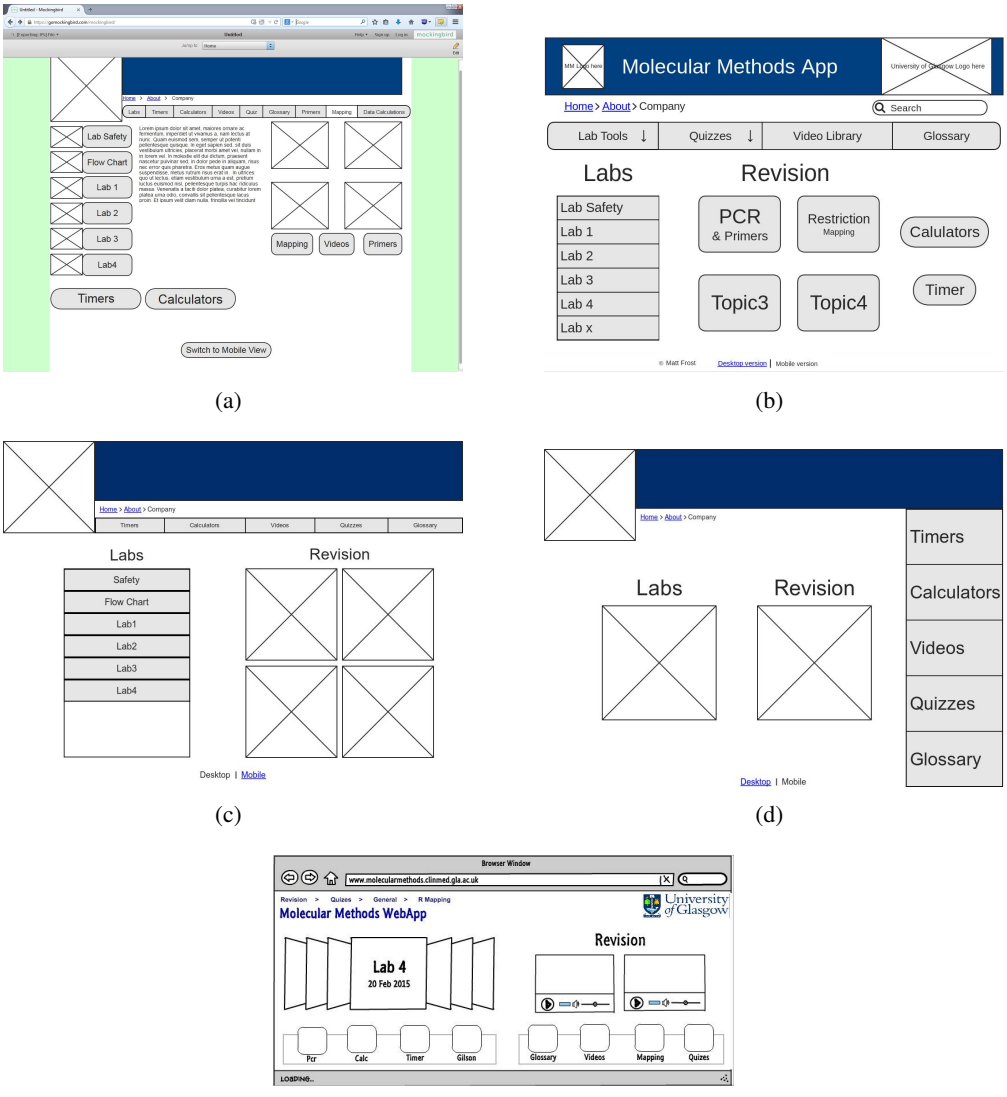


Figure 3.1: Early wire-frame prototypes of the student interface

3.2.2 Key Features

One of the most difficult parts of the design was the student interface of the application. The main problem that had to be considered was that a component such as an icon, for example, might make perfect sense for a computer scientist, but an inexperienced user might find it meaningless or confusing. The general aim was to keep every component of the website simple and intuitive, so that both experienced and inexperienced users would feel comfortable when using it.

Since the project is a learning tool for the School of Life Sciences, the colours of the department were integrated into the application to ensure that students would feel more familiar with the design. Furthermore, according to the first focus group results, students raised complaints about the previous application's design and interface. The system they used did not have the look and feel of an application. This was due to the fact that the previous application had some performance and design issues. The new system was designed with focus on usability and user-friendliness.

Another important aspect of the application, was to take into consideration the distinction between the two different sections: Lab and Revision. This was based on requirements from our clients, since the students were using the application both at home, when revising, and in the lab, during experiments. The sections most used by the students in the previous implementation were kept and improved, while unnecessary ones were removed or re-factored. Extra features were added, based on additional student feedback.

Below is a brief description of some of the key features that are included in the student interface of the application.

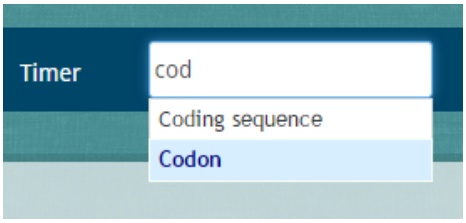
Navigation Bar

One of the primary features is the navigation bar, visible and located at the top of every page. In addition, it should include links to every section of the website.

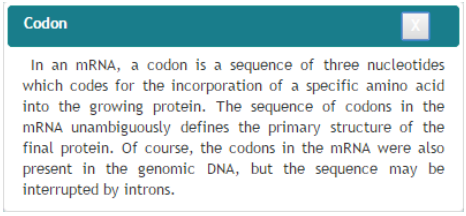
One of the first considerations was whether to include drop-down menus for sections that have subsections; or instead, links to subsequent pages that would provide the various options. Since the goal was to make the user's experience as smooth as possible, the final design included every page link on the navigation bar. This, and the use of drop-down menus, facilitates rapid access to the whole website.

Features that were required by our customers, such as the timer, posed design challenges. The timer in it's inception, was a pop-up, in order to allow the user to perform concurrent activities. Eventually, it was integrated into the navigation bar. To create a seamless experience, the timer is clicked, it appears as a drop-down, allowing unobstructed access to the rest of the page. An auto-complete search box was added to allow browsing the glossary from any page. When accessed, a drop-down menu appears with the results as shown in Figure3.2(a), and by clicking on one results, a small windows is popped-up with the definition of the term, as shown in Figure3.2(b).

In Figure 3.3 the desktop version of the navigation bar is presented. As stated above, it includes links for every section in the application. Furthermore, a glossary search box is integrated into the navigation bar. On the other hand, Figure 3.4 displays the mobile version of the navigation bar. As the screen resolution scales down, the desktop version of the navigation bar is hidden, and the



(a) Auto-complete search results



(b) Search pop-up window

Figure 3.2: Searching the glossary

mobile version of the navigation bar appears. This is, in fact, another drop-down menu which is similar to the original bar in terms of contents, while scaling properly on a mobile device.

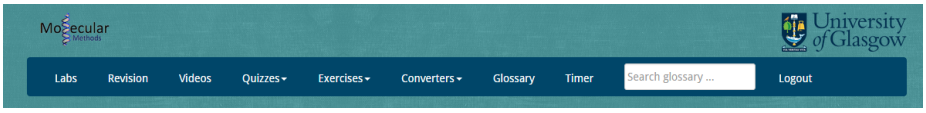


Figure 3.3: Main Navigation Bar (Desktop)

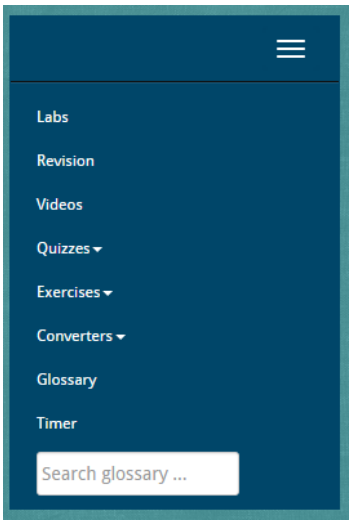


Figure 3.4: Main Navigation Bar (Mobile)

Lab Section

The lab section lists all the different labs that the students have to undertake during the Molecular Methods course,

The first design attempts of the of the Lab section included only links to the different labs. According to the requirements gathered and the continuous design improvement, a briefing about the course and a link to the lab manual was included in order to make it easily accessible for students. Furthermore, a welcome video and a link to a safety video were included as well.

The links to each individual lab at the bottom of the page were represented as plain text at the beginning and, later on, as it is shown in Figure 3.5, replaced by both icons and text. Each individual

link directs to another page where a list of tasks and intended learning outcomes of the specified lab are included, along with links to relevant documents.

A miniature navigation bar exists within the labs to facilitate navigation to different labs without the need of going back to the main lab section. The desktop lab section includes the same content as the mobile view; the latter scaling down, in order to fit on a mobile screen.

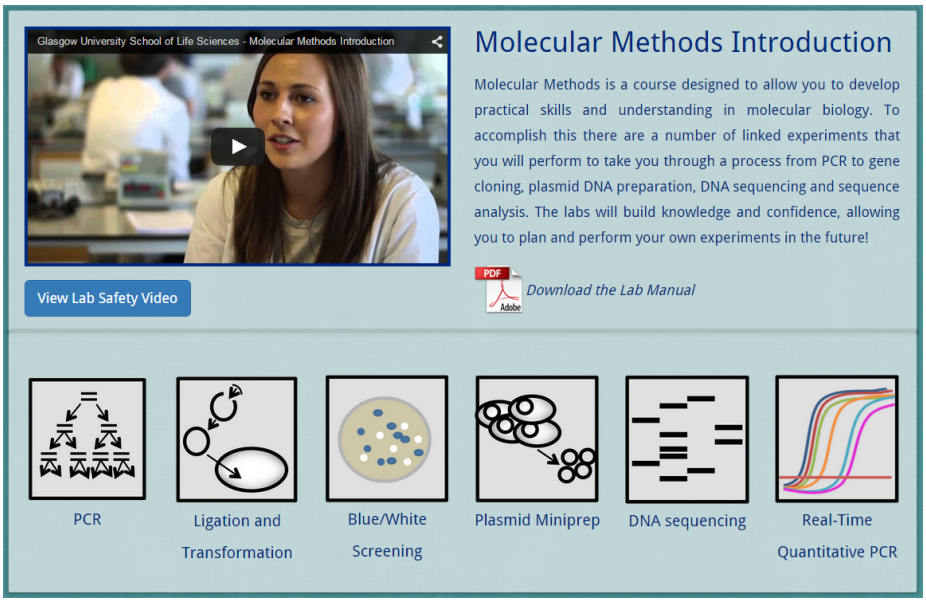


Figure 3.5: Lab Section

Revision Section

Initial design ideas for the revision section of the application included a topic oriented division of the section into four categories. Underneath each category, the relevant material was included. That is, a link to videos, a link to the category's quiz, and, if available, a link to the category's exercise.

In order to satisfy client needs, the revision section had to be modified. The final design was divided into three main categories, instead of four; these were: PCR and Primers, Restriction Mapping, and Laboratory calculations. Underneath each category, the link to the video was replaced by the actual embedded video. If there were more than one video associated with a specific category, a video carousel was used instead.

In addition, links to each category's exercise and quiz were presented beneath the carousel. Initially, the links were represented only as icons, but, due to clarity considerations, text was also added to the final design. While, from a design perspective, both the labs and the revision section are similar, the latter is differently structured in order to provide a better learning experience. The mobile view of the revision section includes the same content as the desktop view, but, similarly, it scales down in order to fit on a mobile screen.

Quiz Section

The quiz section was required by the clients in order to help students when revising. A bank of questions with four different categories was imported from the School of Life Sciences Moodle page. Based on the quiz category selected, the appropriate questions are loaded into the page.

The initial plans for the quiz page included having a single question per page, with two buttons to allow navigation through the other questions and a check button at the bottom of each page in order to validate the selected answer. According to later requirements, this was changed to display on one page 15 randomly-selected questions. The page includes a button at the bottom, which the user can click and get redirected to another page, where feedback for the questions answered will be displayed.

3.3 Administrator Interface

The admin interface is an essential part of the application, supposed to encompass all the functions that the course coordinators need to use to maintain the website. Due to the fact that the clients were from a non-technical background, the admin panel had to be designed with simplicity as a priority.

Navigation through the panel was taken into account in order to allow the clients to rapidly move through different sections. For it to support ease of use, a simple theme was needed. Meaningful feedback and information in every section and with every action taken was mandatory in order to make the inexperienced administrators familiarise themselves with the web application.

There are two essential features the clients were interested in: easy manipulation of data on the website, which would be accomplished through the implementation of database management into the interface, and the ability to collect statistical data on the usage patterns, visualised through a chart.

3.4 Data Storage

For storing information on the website it was decided to use a database, as it allows controlled, concurrent access and modification of data from multiple locations. Along with a Database Management System, it would allow implementation of simple ways for the course coordinators to update and add information on the website so that it can be maintained without external intervention. This would drastically increase the longevity and adaptability of the system.

Chapter 4

Implementation

This chapter covers the process of implementing the design, how the appropriate tools and frameworks were decided upon and how those technologies were used to fulfil the application's design requirements. Firstly, different approaches to web development were evaluated. Then, the team decided on a set of tools to use in the implementation. Lastly, the project was divided into Front end, Middleware and Back end components, implemented concurrently. The chapter focuses on some of the key challenges faced during this process.

4.1 Framework and Tool Choices

Based on the the results of the requirements gathering process, the team decided that a web application would be the most appropriate way to satisfy the clients. There are two main ways to develop such an application.

The first one is to design and develop everything anew. That is, for every page of a web application there exists a single file that the client computer will request in order to fetch the specified page from the server. This approach offers a great deal of flexibility, but results in longer development times and a large amount of repeated boiler plate code. An alternative is to use a web application framework, which automates much of the basic functions to allow developers to focus on building features unique to their application. This results in a shorter development time frame, as implementing all the basic components from scratch is not required. Such frameworks can be considered as templates or structures which the developer can use and extend.

A web framework sounded like the most preferable choice for our project, considering the time constrains implied by the project deadline. There are numerous frameworks available and each has their own benefits and drawbacks. Two of the most well known such frameworks are Ruby on Rails [10] and Django [13].

For the sake of compatibility with the previous year's Molecular Methods project code, due to the team's broad experience with Python (the programming language used by Django) and because of the famous and easy to follow tutorial (Tango with Django[2]), Django became the web framework of choice.

4.2 Working with Django

Django is free, open source and considered to be a modern Web development framework. It offers high-level abstractions of common Web development patterns, simplifies many frequent programming tasks, and encourages rapid development along with clean design. Django reduces much of the repetition in Web development, allowing the developer to concentrate on the functional aspects of projects. In addition to saving valuable development time, Django is also considered secure, as it avoids common security mistakes such as SQL injections [16] and cross site scripting [4]. Django can also be flexibly scaled according to developer needs in terms of traffic demands. Extras, such as user authentication, RSS feeds, and administrator panel are included in Django. These extras are robust, secure, and can be easily integrated into the code. Due to the narrow scope of the project, it was decided that the system should be implemented as a single Django application.

Django uses the Model-Template-View (MTV) model as shown in Figure 4.1. It is split into three main components and these are referred to as: *models*, *views*, and *templates*. It is common for programmers more familiar with a Model-View-Controller (MVC) model to get confused with this design. A developer can clearly identify the similarities between the two if they consider the following: Django's views are controllers and Django's templates are views in the MVC model.

These components communicate in the following manner; when a web browser sends a request to the web application, the URL dispatcher, as shown in Figure 4.1, process the request and responds with the appropriate view, according to the web-browser's request. If the specified view uses database information, then these will be retrieved through Django's model components, and passed from the view component to the appropriate template and, finally, back to the browser.

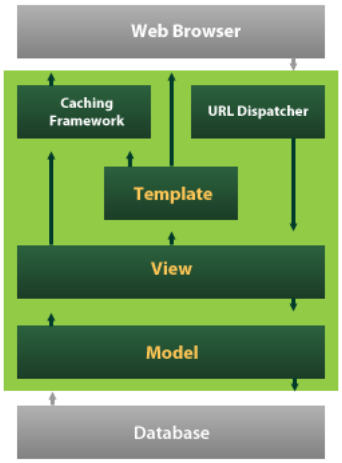


Figure 4.1: Django's MTV Model [8]

The rest of this Section presents additional details on each individual component of the MTV model.

Models. A Django model is a python object which contains the essential fields and behaviours of the data being stored. Each of these models contains attributes which represent the relation fields and properties. In general, each model maps to a single database table. The Django object relational mapping functions and database encapsulation allow the developer to write simple python commands in python objects to query a database, instead of writing raw database queries. In that way, Django ensures that it sanitises any parameters that the user can input and hence makes the

web application secure against SQL injections [13].

Views. A view is a python function that takes a web request and, according to the function implementation, returns a response, such as the content of a web page, an image, or an error message. Usually a template and a context dictionary are rendered into the response. From inside a view function the developer can query the database and retrieve the data needed. In addition, the data can be passed from view to template through a dictionary data structure. The data is then passed to the client which requested the specific view. The code of these views can be located anywhere in the project as long as it is included on python's path.

Templates. A Django template is a text string that is used to separate the application logic from the presentational aspects of the application. Templates also fulfil another design goal by reducing code repetition. Most websites have a repetitive layout and structure, which Django templates can automatically generate for the developer. Templates are typically used for producing HTML, but are capable of generating any text-based format. Additionally, Django provides the developer with a template mini-language which allows the template to contain variables, which are replaced with values as soon as the template is evaluated. Template also contain basic logic constructs called template tags, which are used in a similar way to programming constructs such as if statements and for loops. These tend to be useful in cases where there would be a large amount of repeating code, such as displaying all the terms of a glossary on a page. When objects need to be loaded from a database and displayed multiple times, instead of writing the same piece of code to display each of the objects multiple times, a for loop statement can be used, with the objects being assigned as variables. This allows the the template file to look cleaner and to be easily understandable.

4.3 Front end

The implementation of the front end component involved the development of the user interface for the web application. The user interface comprises the visual elements that the customers see and interacts with directly. The front end implementation required both programming (knowing which tools to choose and how to use them) and design skills (understanding how to arrange elements in the interface and choosing fonts and colour schemes that are visually appealing).

During the development of the user interface, the team found some of the tools and techniques used to implement the clients' requirements challenging. Firstly, a robust and friendly user interface had to be created. When opened, this should display the relevant information in an easy to read and persistent format. Secondly, due to the variety of devices with different screen sizes and resolutions, the user interface had to be dynamically scalable and adaptable for each type of device, including mobile phones, tablets, laptops or desktop computers. Thirdly, the user interface for the web application was designed and implemented to offer full functionality on laptops and desktops, but to hide some elements when opened on a mobile platform due to the smaller screen size.

4.3.1 Animations

One of the initial requirements of this project was to produce a piece of software that would look and feel like an application. This was achieved using JavaScript [11], an event-based programming language, that is used to transform a static HTML page into a dynamic application. An interesting

challenge in this respect was represented by the task of implementing the timer functionality. The requirements for the timer were for it to be resettable, hideable, and not to interfere with user's actions when browsing the pages of the application. The last requirement was the most difficult to implement because simple events, such as mouse-click or mouse-over, can interfere with the timer's counting function. This problem was solved by having two separate functions: one for handling events and the other for counting. This approach was feasible due to the Document Object Model (DOM) [11] used by JavaScript libraries, which allows setting and listening to different types of events while performing other operations such as counting.

4.3.2 Asynchronous searching

An important functionality of the web application is represented by the glossary search function. This performs a glossary search in order to retrieve the term and its definition from the database, then display them on the screen. In the initial implementation, the team encountered problems when sending requests to the server and returning the results list to be displayed. This problem was solved by sending asynchronous requests that retrieve the data and then render it on screen, on the same page, using dialog boxes. The asynchronous requests technique, shown in Figure 4.2, is usually known as AJAX (Asynchronous JavaScript and XML) and is part of the JQuery library, which provides simple methods for client-side scripting of HTML pages. These methods were used to look for and retrieve glossary terms in the Search function.

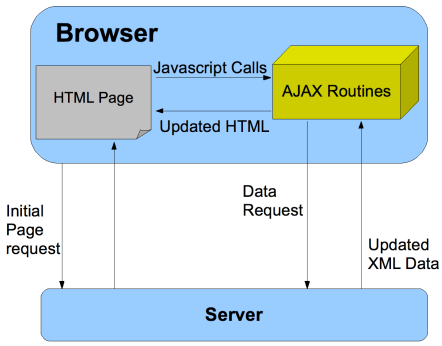


Figure 4.2: AJAX Flow Diagram [22]

4.3.3 Application Responsiveness

Making the application available on mobile, tablet and desktops computers posed a significant challenge. The different screen dimensions of the devices would require individually tailored files to display the same content effectively on each of the given devices. Producing two versions of the application was under initial consideration. One version would be optimised to support mobile devices and a second version would be created for desktop computers. The application would detect what type of device was being used and show the optimised version for that device. However, developing multiple versions of every page on the site would be very time consuming. In order to overcome this challenge and streamline the development process, it was decided to use a responsive web design. This approach involves laying-out and coding the website so it provides an optimal viewing experience across a range of devices, ensuring the user's ease of reading and navigation with a minimum of resizing, panning, and scrolling. By adopting this approach, the time required

for testing and development was reduced significantly and the need for device specific pages was eliminated.

Although there are a number of frameworks available to assist in responsive web design, the Twitter Bootstrap Framework [20], commonly referred to as Bootstrap, is widely used and has a large number of supporting resources easily available, making it an obvious choice for the project. The Bootstrap framework is a free, open-source collection of tools used for creating responsive websites and web applications. By using a selection of HTML and CSS-based design templates, Bootstrap allows the easy generation of interface components such as: forms, buttons, and navigation elements. Bootstrap also offers a number of JavaScript extensions to implement dynamic components, such as the carousel used in the videos and revision sections. In addition, Bootstrap simplifies the testing process by guaranteeing compatibility with the latest versions of the Google Chrome, Firefox, Internet Explorer, Opera, and Safari browsers.

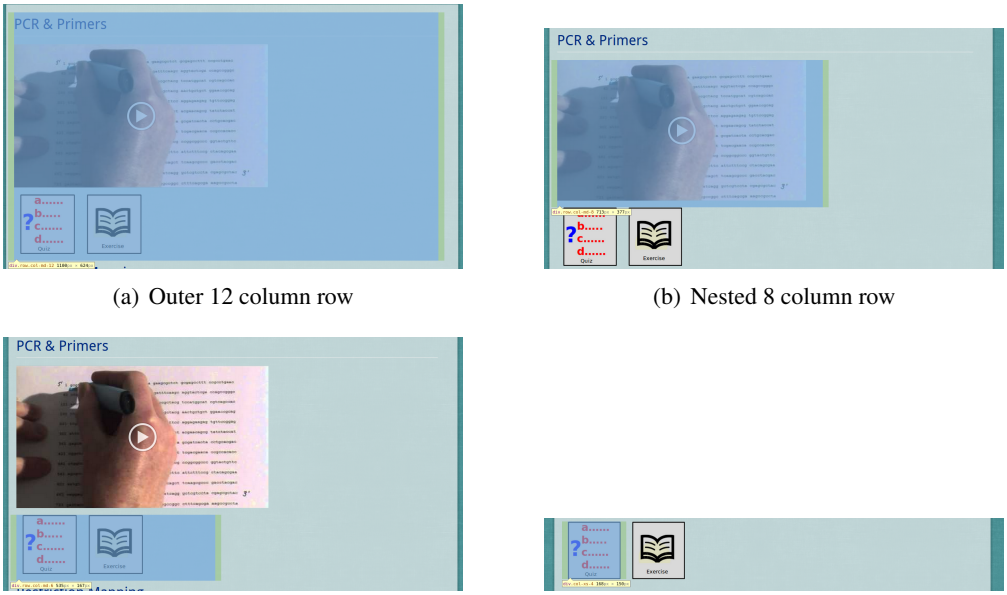
The responsive features in Bootstrap are derived from a grid system which divides the visible screen area into 12 equal columns and an unlimited number of rows. The column sizes, representing a percentage of the visible screen area, can be declared as any width between 1 and 12 and combined in any variation, provided they sum to 12 or less. Columns can be set to different sizes depending on the size of screen they are being viewed on, allowing for greater flexibility.

The capabilities of Bootstrap were utilised in the application extensively, most notably for the Navigation Bar and the Lab Section. For the Revision Section, it was necessary to override Bootstrap's default behaviour of vertically stacking all screen elements when scaling, as maintaining a consistent screen layout across all devices was desired. This was achieved by the using Bootstrap's grid system to create nested rows, enabling control over how the page scaled and ensuring a consistent layout for each of the sections on the page.

Each of the topic sections on the revision page was placed within a 12 column spanning row, as shown in the highlighted section of Figure 4.3(a), as this uses the full available width of the screen. This containing row is subdivided with an 8 column row, highlighted in Figure 4.3(b) to hold the carousel containing YouTube video elements and, below that, a 6 column row which holds the icons linking to quizzes and exercises, as shown in Figure 4.3(c). Each of the icons is then assigned a 4 column width, see Figure 4.3(d). These varying column sizes act to constrain the maximum size of the responsive elements, such as icons and images, which would otherwise automatically resize to use all of the screen width available. A side effect of this solution was that, when viewed on a mobile device, the Youtube elements were too large for the containing carousel. Since Bootstrap is able to hide or display any element based on the size of the user screen, properly scaled YouTube elements for mobile device screen sizes were created, which are hidden from desktop, laptop, and tablet users. This allows the revision section to maintain a consistent layout when viewed from a mobile, tablet, laptop, or desktop.

4.4 Middleware

The term middleware refers to the connection between the back-end (the database) and the front-end (Django views, or simply, the pages that the client will be able to see). While some parts of the system were straightforward to implement, others required complex querying of the database and manipulation of data through python functions. This was done to extract and pass the required information to the views.



(a) Outer 12 column row

(b) Nested 8 column row

(c) Nested 6 column row

(d) Nested 4 column row

Figure 4.3: Nested row column structure

In order to avoid repeating code for the navigation bar which should be located at the top of every page, a base template was implemented using Django's template language, as shown in Figure 4.4. This is extended by every other template of the application.

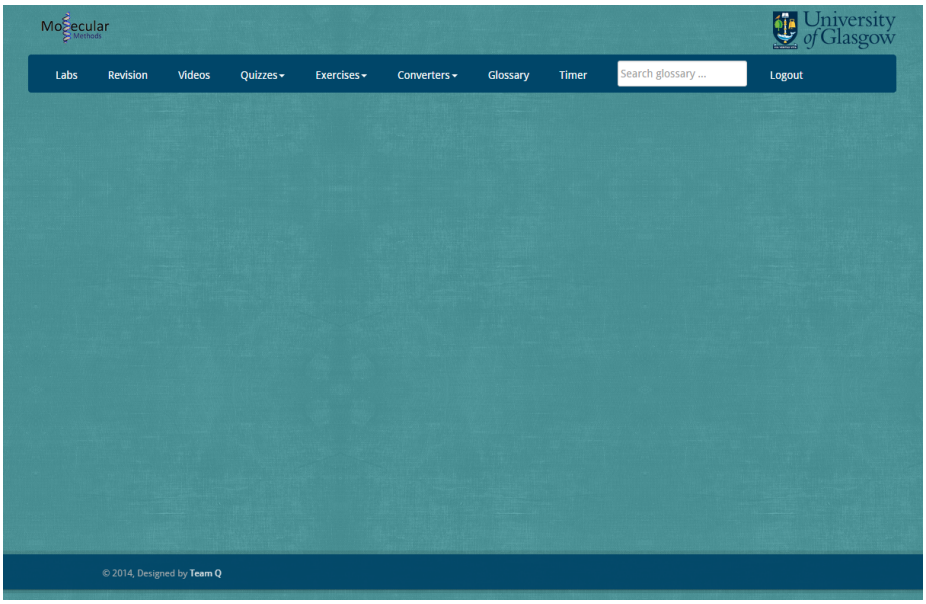


Figure 4.4: Base Template

Sections such as login and registration, required extensive editing. The default Django login and registration forms were used, since only a simple user authentication mechanism was required by the clients. One of the problems encountered was that the default Django forms could not be customized at the required level. Default templates for both registration and login features were

adapted, using a method that wrapped all the elements contained within a form in a paragraph, as shown in Figure 4.5(a). As such, the need of listing each one individually was removed. After some research on Django forms, it was found out that, in order to be able to customize each individual element of the form, the elements had to be listed one by one as shown in Figure 4.5(b).

```
<form id="user_form" method="post" action="{% url 'desktop:register'

            {% csrf_token %}
            <!-- Display each form. The as_p method wraps each eleme
                 (<p>) element. This ensures each element appears on
                 making everything look neater. -->
            {{ user_form.as_p }}

            <!-- Provide a button to click to submit the form. -->
            <input type="submit" name="submit" value="Register" />
</form>
```

(a) Elements wrapped and rendered

```
<form id="user_form" method="post" action="{% url 'desktop:register' %}" enctype="multi
    {% csrf_token %}
    <br>
    <p><input class="input-large form-control" type="text" style="color: #00008b"
              name="username" value="" id="username" placeholder="Username"></p>

    <p><input class="input-large form-control" type="email" style="color: #00008b"
              name="email" value="" id="email" placeholder="Email"></p>

    <p><input class="input-large form-control" type="password" style="color: #00008b
              name="password" value="" id="password" placeholder="Password"></p>

    <!-- Provide a button to click to submit the form. -->
    <input type="submit" name="submit" value="Register" class="form-control btn btn-pri
</form>
```

(b) Elements rendered individually

Figure 4.5: Registration Page Form

The quiz section of the web application was complex in terms of data manipulation. It listed multiple choice questions that have one correct answer. According to the requirements gathered, (see Functional Requirements Section), a bank of quizzes of 4 different categories was needed. Based on the category selection of the user, 15 random questions of the respective topic were to be shown on screen. Listing the appropriate questions according to the category chosen by the user was possible using iterative loops.

Each question had numerous possible answers beneath it; the correct answer would be selected by clicking the radio button next to it. Radio buttons were chosen in order to prevent the user from selecting more than one option for each question. However, this caused an issue due to the quiz page being implemented as a single form. The user was able to select only one radio button from the all the answers on the whole quiz page. To resolve this, the radio buttons associated with a specific question were labelled with a unique name and grouped dynamically.

Since questions are dynamically loaded, radio buttons could not be statically named and assigned. A 'for' loop counter variable was used to group the radio buttons as shown in the assignment of the '`name`' attribute in Figure 4.6. This variable was also used to identify each radio button of a specific question, as shown by the assignment to the '`id`' attribute in Figure 4.6.

```
{% for answer1 in answers %}
    {% if answer1.question == question1 %}
        <input type="radio" name="ans{{ forloop.parentloop.counter }}" id="ans-{{forloop.counter}}"
        <br><br>
    {% endif %}
{% endfor %}
```

Figure 4.6: Part of Quiz Template

In order to validate the questions' answers, a button was included at the bottom of the page. As previously explained, this button redirects to another URL, processed through the URL dispatcher, in order for the application to respond with the appropriate view and template. The view-python function was complex in terms of implementation, since it required a reverse-engineering mechanism to capture all the radio button group names. An integer variable was used in a 'while' loop statement in order to capture all the radio buttons, including the selected ones. Afterwards, the answer selected was checked. This was done via the inclusion of a boolean type field for each answer of each question.

Finally, the questions, including all the possible answers, are listed on the page. A score counter was added to the bottom of the page, allowing the user to check how many questions were answered correctly.

Extensive error checking was required of the quiz section in order to display the appropriate messages: when there are no questions about a specific section in the database, when the user has not answered any questions, and when questions are incorrectly added into the database. Moreover, since images were required both in questions and answers, Django's template language was used to allow the rendering of HTML tags.

Sections like the glossary were implemented in a simpler way. The glossary retrieves terms from the database and appends them to dictionary data structures according to the first letter of each term. We used 27 structures in total: one for each letter of the alphabet, and another, for terms that do not start with a letter. After being passed to the template, as shown in Figure 4.7, each dictionary is checked if it is empty. If that is not the case, all the terms that start with that letter are listed; otherwise, the section is skipped. This allows for a clearer implementation of the glossary section.

```
{% if termsA %}
    <div id="a"><a href="#top"><font size="4">A</font></a></div>
        {% for item in termsA %}
            <br>
            <font color="darkblue" size ="4.5"> {{ item.title }} </
            <br>
        {% endfor %}
    <hr>
{% endif %}
```

Figure 4.7: Part of Glossary template

4.5 Back end

The back end deals mainly with the storage, structuring and management of data. The design of this project requires an implementation and population of a flexible database through a database management system.

4.5.1 Database Management System

Database creation and access is usually accomplished through a database management system (DBMS). Django provides a choice between a couple of different popular systems - SQLite [12], MySQL [5], Oracle [6] and PostgreSQL [9]. Each comes with its benefits and flaws, but the final choice of a DBMS for this project was SQLite - an embedded database management system.

Its main advantages are listed below:

- it is file-based and the whole database is represented by a single file, making it very easy to move between servers and share between team members;
- it is lightweight, yet implements almost all basic SQL functions that are needed. For this project no SQL queries were written, but the implementation allows for that to be added, should it be necessary.

SQLite's main disadvantages are:

- it has no user management, which can be a security issue if any sensitive information is stored within;
- it is not configurable, so performance cannot be optimized;
- it allows for only one write to the database at a time, thus it is not a good solution for applications requiring high write volumes.

4.5.2 Database design

The database was designed to accommodate all information on the website that the clients specified may undergo changes. This allows for immediate updating of the website content with every change to the database without the need for any further actions.

The labs section would always consist of six labs, but their structure could change. Thus, a database table with tuples for each of the six labs was created, which stores all the text that is displayed on the website.

Another important aspect of the database design was that of the quizzes. Multiple tables were created to store each question and its corresponding multiple choice answers, along with a table for storing results for each user's latest answer to each question. The results table exists, but the

functionality to populate it was not implemented in the middleware due to time constraints and was left as a plan for the future.

The restriction mapping exercise from last year's Molecular Methods website was ported to the project, and uses a database to store each example exercise. For that purpose, a table in the database was dedicated to representing a complete copy of last year's model.

A glossary table containing an entry for each glossary term and its description was also created, as it was required to allow the course coordinators to modify term descriptions and names, as well as add new ones. The implementation of a glossary search function was made easier by the use of a database, as it allowed for querying of the contents of the table.

4.5.3 Database population

Initial data to populate the website with was provided by exporting data from the Moodle database, the course's lab manual, and other files given by the clients. The process of importing them into the database included the need for attentive parsing of the given files with scripts and testing the results manually inside the website.

4.6 Admin interface

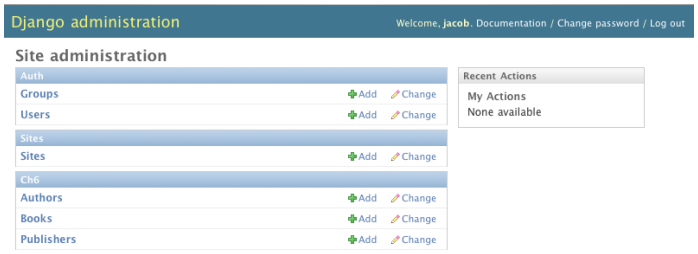
Django already comes with a default admin interface which can modify information from the database. However the project's requirements included some additional functionality. Clients needed to be able to also import and export data, edit fields that support HTML without the need to be proficient in writing it, and see statistics about the website, all through the same interface. The easiest way to achieve these tasks was to research a large amount of plug-ins created for the Django framework. Below is a short list of the main ones implemented in this project:

Yawd-admin extends the default admin website for Django and adds support for the following features:

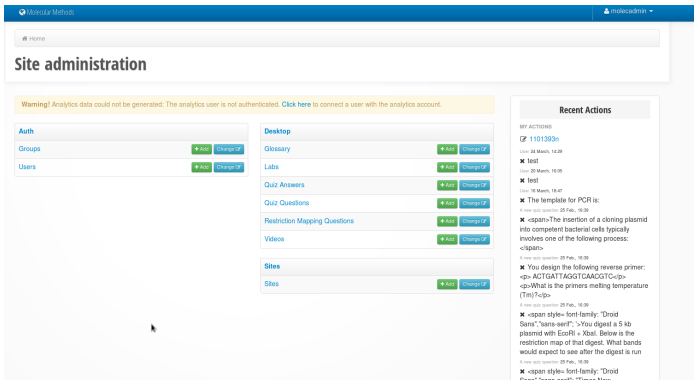
- clean, responsive Bootstrap interface (comparison shown in Figure 4.8);
- integration with Google Analytics for displaying statistics on the admin home page;
- refurbished admin widgets.

Django-import-export is a Django application for importing and exporting data. Its features are:

- support for many famous formats including Excel, CSV, JSON etc;
- admin integration for importing and exporting;
- filtering of data to be exported;
- preview of data to be imported.



(a) default admin



(b) Yawd-admin

Figure 4.8: Comparison of Yawd-admin and Django default admin views

Google-api-python-client is the Python client library for Google's discovery based APIs. It co-operates with **oauth2client** - a client library for accessing resources protected by Oauth 2.0 - to allow the admin website to authenticate Google Analytics in order to analyze and provide statistics to yawd-admin, later to be displayed on the admin home page.

Django-redactor-editor integrates the Redactor text editor [14] into any text input field that is specified in the admin configuration file. Rich text on the Molecular Methods website is defined with HTML. This plug-in allows text formatted with HTML to be displayed as rich text and edited in the same way as some popular document editors, like Microsoft Word.

4.7 Improving User Interface Performance

The performance of the application was an important aspect of the Non-Functional Requirements. This section details some of the challenges faced in ensuring the implementation met this requirement.

4.7.1 Lazy Loading

Due to the design requirements of the project, it was necessary to display several embedded YouTube videos on the revision page and on the video index page. The first challenge raised by this requirement was displaying these embedded video elements in an efficient way that would be easy for users to navigate and would scale effectively.

However, the successful implementation of the video carousel lead to significant delays during the loading of the revision and videos pages. This problem was solved by implementing a 'Lazy loading' design pattern, which is commonly used to defer initialisation of an object until the point at which it is needed.

Instead of embedding the full embedded YouTube video player, the carousel element only displays the thumbnail preview of a YouTube video and overlays a play icon, so the image resembles a video player. When the play icon is clicked, the thumbnail is replaced with the embedded video player, resulting in the extra player-specific resources being loaded only when the user has decided to play the embedded video and not otherwise.

4.7.2 Content Delivery Network

A content delivery network or content distribution network (CDN) is a large distributed system of servers deployed in multiple data centres across the Internet. The CDN is used to serve content to users, ensuring high rates of availability and performance [19]. The rationale in using CDN hosted versions of Bootstrap and JQuery is that the CDN will provide an increased amount parallelism available to the web browser, allowing increased performance [17], coupled with a greatly increased chance that there will be a cache-hit (where the browser has a locally cached copy from visiting another site linked to the CDN) on initial access, eliminating the need for downloading the CDN-hosted content completely. The use of CDN-hosted content also helps reduce the amount of bandwidth used by the university server and ensures that the total amount downloaded by the user will be minimised, as CDN providers pre-compress.

The potential drawback of employing a CDN is that, if the service is unavailable, the site will not function correctly. However, this is mitigated by using a local fall-back script which loads locally hosted versions, should the CDN fail.

4.8 Deployment

The deployment of the web application was achieved with the help of the system administrator of the School of Life Sciences. Certain issues with communication occurred, as the main method of interaction was through email. Otherwise, the deployment process went relatively smooth with the help of a professional; the experience gained from deploying to a web hosting platform was useful testing purposes.

Chapter 5

Evaluation

This chapter discusses the methods used for testing and evaluating the web application. Feedback was collected from a large number of users, since the application was available to all level 3 Biology students. This allowed the team to collect a significant amount of feedback and consider future improvements.

5.1 Testing and Deployment

Throughout development of the application, testing was carried out on a regular basis by all members of the team. This was possible as the team worked individually on their own computer systems. Every time a new function was introduced or changes were made, tests were performed on a local server to ensure the website's stability. This also ensured different operating systems, browsers, and machines worked flawlessly with the website. Additionally, the weekly meetings with the clients allowed them to see new features soon after their completion, and ensure that these were implemented according to specifications.

The Labs, Revision, Videos, and Glossary sections were relatively straightforward to implement, and although some had to connect to the database to draw the information they needed to display, the main issues with these sections were related to the way the content was layout. Thus, no bugs were found in those parts of the website. The Quizzes were a more complex feature, which required server computation. The testing of this functionality therefore had to be more thorough, as problems could arise in very specific scenarios. For example, the questions were imported through a script, which used files exported from the Moodle database of the Molecular Methods course. During the testing, it was found that Moodle does not export certain questions properly, and a some had multiple correct answers. Additionally, a number of questions had images associated with them; which were essential to their understanding. Since there was no automated way of adding these images, we were required to add them individually by hand.

The Exercises, Converters, Timer, Search Bar, and Admin Interface were implemented by adapting code from on-line sources, as well as from last year's project. This resulted in few bugs code-wise, despite some problems with adapting the features to the current project. A good example of problems that arose with the implementation of these functions was code that, at first glance, looked

perfect for integration, actually required time-consuming adaptations in order to fit the working version of the current system.

Towards the completion of the implementation, it became apparent that testing locally does not fully capture all possible problems or errors that might occur; for example, concurrent requests from multiple users. Therefore, in order to test if the application was sufficiently robust in preparation for a live release, deployment on a remote server environment was necessary. This was done using a web hosting service, known as PythonAnywhere [18]. This service allows developers to build and host a web application or any other code directly from a browser without having to install software or manage a server.

Before the application was released, the clients set up a focus group on 4th February 2015 with additional teaching staff from the Molecular Methods course. In this session, the application was reviewed and tested. Some issues arose as a result of this session, which led to minor changes in the design and functionality. No problems in terms of performance or loading times in a remote server environment were identified. In this respect, this testing session was considered a success.

With testing complete, the team was ready to deploy the application for use by the students of the Molecular Methods course. Although deploying a web application using PythonAnywhere proved rapid and simple, it is only feasible on a temporary basis, i.e for testing purposes. Therefore, a permanent and more reliable server environment was required. This was provided by the Biology department's technology support team's server. From here, the application was deployed and made available for access by the students, which allowed for the user evaluation to commence.

5.2 User Evaluation

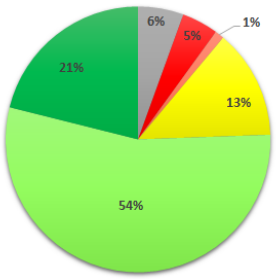
In order to test the usability of the Molecular Methods application and to see how it was used by the students, Dr Scott and Dr Veitch were requested to invite students to trial the application during the Labs and the times they were revising for the course. A focus group was gathered on the 18th March 2015 to collect feedback and identify any remaining issues. It was at this feedback session that the questionnaires were handed out. The students were not able to answer questions in terms of the usefulness of the application for revision, as many were just starting to use it for that purpose.

5.2.1 Feedback Forms

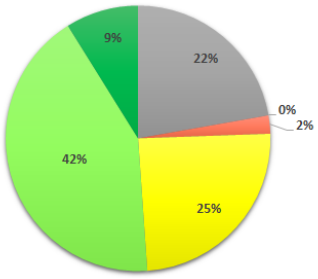
The design of the questionnaire was taken into careful consideration, as it was important to gather as much information as possible from as many students as possible. The questions were designed in such a way to prevent confusion among students and the number of questions was not excessive. For this purpose the System Usability Scale was used, which provides a standardized way to evaluate an application, even from a small sample size. The questionnaire is available in Appendix A.

In total, feedback from 90 students was collected and below is a summary of the extracted data from the questionnaires.

Figure 5.1(a) presents the results concerning whether the students agreed that the website was easy to use. The majority of students agreed and, when asked in person, did not have any questions



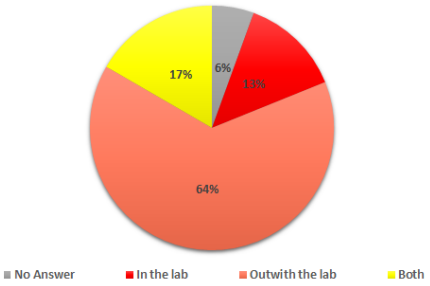
(a) The application was easy to use



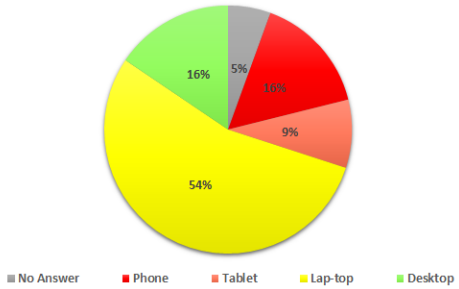
(b) Did you like the design of the application

Figure 5.1: Results of questions related to project aims

or problems with navigation and usability. Figure 5.1(b) shows that the decision to prioritize the application's design was not overlooked and students found no problems with understanding the styling. Most students found the application appealing from a visual perspective and very few (2%) did not like the design of application.



(a) When have you accessed digital resources



(b) How did you access the application

Figure 5.2: Results of questions related to project aims

Figure 5.2(a) shows where the students used the application. It demonstrates that more than 80% of students use the application outside of lab hours, but only around 30% used it inside the lab. This is a possible concern for the usefulness of the functions implemented to aid with lab exercises, as it can be seen that the students mostly used the application for revision purposes.

Figure 5.2(b) concerns the devices that the students used to access the website. The results make it clear that this application is predominantly used on a desktop or laptop machine (nearly 70% of the sample). This statistic correlates with Figure 5.2(a) and may be either a reason for, or a consequence of, the lack of use of the application during lab hours. It also shows that emphasis on designing a desktop application has been rewarded and students find it a lot more convenient for desktop environments than the application from last year.

Figure 5.3 concerns the specific usefulness of each separate function implemented. It can be concluded that the lack of use of lab-related features, such as the timer, the calculations, and the lab links, can be attributed to the fact that students utilised the application as a revision tool. All the other functions were relatively useful, with a general appeal towards videos and quizzes.

Figure 5.4(a) demonstrates that a large majority students related to the material presented in the application and found that it enhanced their learning. Only a very small under (2%) disagreed with

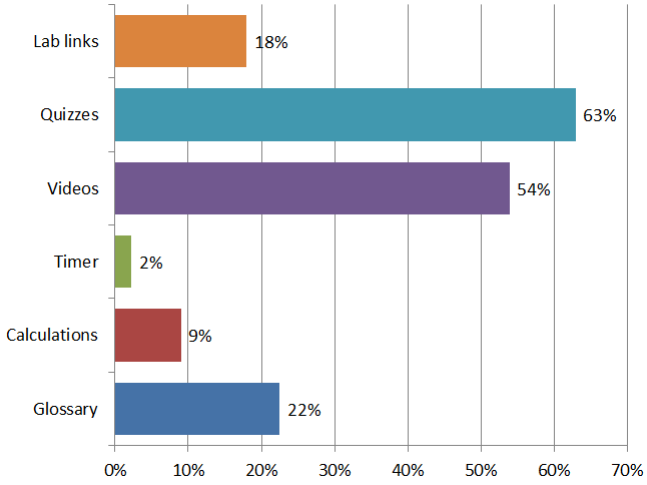
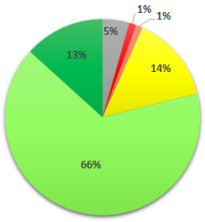
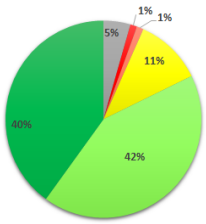


Figure 5.3: Which parts of the application were useful



(a) The app enhanced my understanding of Molecular Methods



(b) I would like similar digital resources available for other aspects of my degree

Figure 5.4: results to some questions related to project aims

the application enhancing their understanding of Molecular Methods. This is the most significant outcome of the evaluation as it demonstrates that the application has achieved the main goal of the project.

Figure 5.4(b) show that their is a clear interest in having similar applications for other courses. This gives an idea for a further development of this application: make it more flexible in terms of the content stored and the structure of the website as to allow it to completely adapt to a different course. Collaboration with more future customers would be necessary for the purpose. While an ambitious idea, it is certainly one of great interest.

In addition, the team was able to collect several valuable user comments from the questionnaires. Some of them proved helpful, as they confirmed the results shown above in the charts. Below, we list some of the key comments collected.

As shown in Figure 5.3, the students found the quizzes and videos sections the most helpful. According to their comments, this is due to the fact that a video can be followed along while carrying the task to be learned. Some students specifically stated that they understand the concept of restriction mapping more by watching the videos. In addition, students affirmed that they used the quizzes, since the shuffling technique that was implemented in that section allowed for self-assessment at any time. For the lab links section, students found it was helpful, as they could browse and prepare for their following lab session.

Another interesting fact that was concluded from the questionnaire results was that the students were looking for similar digital resources to become available for other courses of their degree, as shown in Figure 5.4(b). According to student comments, this can be attributed to the clear and intuitive layout of the application; an effective and entertaining method of learning.

Comments concerning future improvements and additional functionality of the application were also captured. A few students stated that they would prefer to use the application while off-line and have a downloadable mobile application rather than a web-based implementation. This option was carefully considered, although according to the requirements gathered from the clients (see the functional requirements in Section 2.3.1), it was decided that it would not be feasible. Another important point that was stated by students, was to give appropriate feedback on quiz answers and explain why a certain answer is the correct one. Again, it was discussed with the clients, but due to the size of the question pool, it proved difficult for the clients to provide such specific feedback; therefore, the feature was discarded.

5.2.2 Student focus group

The focus group mentioned at the beginning of this chapter provided valuable information, as it allowed a more personal interaction with the users. The main feedback received from this concerned the functionality and design.

Most of the features the students used were deemed satisfactory and they did not find any fault with respect to the implemented features. The timer was mentioned as being out of place and likely to not be used. When asked about ideas for new features, outside the ones collected from the questionnaires, two interesting ideas were given: the addition of a “frequently asked question” section and a link to the course book that can already be found in the library.

Design-wise, most students enjoyed the colour scheme and overall look and feel of the web application. No problems with slow loading times or lag were reported. Some students added that the website felt similar to an actual application. When asked about what they disliked, some mentioned that on both mobile phones and desktops, the fonts looked too large, which was especially bothersome for the quizzes. Ideas of improvements on the navigation bar were also given, for example, making the bar always stay visible on the screen and adding text on the mobile view, guiding users that the navigation bar collapsed to a drop-down menu (currently it is just an image which may be vague in terms of its functionality).

It can be deduced that the project has succeeded in being useful to the students and would not need much further improvement. Interesting correlations between the results arose, and, as these may point to possible future goals, they were deemed out of scope for the time being.

Chapter 6

Conclusion

6.1 Summary

The main aim of this project was to build a teaching and revision tool for the Molecular Methods students, which can be used both in the laboratory hours and at home to test and improve the knowledge taught within this course. The feedback gathered in testing and evaluation sessions, held with the clients and the students, showed that the team successfully met the requirements for this project.

The web application offers full functionality to anyone (from Molecular Methods students to people who want to learn more about Molecular Biology) who creates an account and logs into the system. Users are able to watch videos, solve quiz questions, test their restriction mapping knowledge, search glossary terms, and read through the course materials and lab manual, both available in electronic format.

The Molecular Methods course coordinators are able to view and modify the content within the application through the administrative interface. This includes adding, deleting, and modifying quiz questions, glossary terms, and restriction mapping exercises, as well as managing the existing users.

6.2 Project Management

The team collaborated in gathering the requirements necessary, designing, and prototyping the components required for the project. In order to maximize individual efficiency, members were assigned to work on different aspects of the system.

Each sub-team took responsibility for a specific area of implementation:

Front End: Iulia Popescu, Paul Dalziel
Middleware: Angelos Constantinides, Matthew Frost
Administration interface and Back End: George Popa, Peter Lishov

To facilitate communication between team members, the team used Asana[1] as a project management tool. Tasks were divided into atomically discernible entities that could be allocated to individual members in the software. Each member of the team could view, edit, and assign new tasks, allowing bugs and other issues to be dealt with efficiently. Asana also provided an overview feature, facilitating the visualisation of the progress towards the completion of each task, as well as the total progress of the project.

Source control was of cardinal importance in managing integration between multiple features developed concurrently. Multiple existing solutions were analysed; however, given the multi-branching functionality present in the Git project, the version control system was deemed appropriate for the needs of the team. Furthermore, GitHub,[15] one of the more popular hosting platforms, offers free access to its services for every member of the team.

The infrastructure on which the project is based upon is hosted on the Master branch of the repository, with sub-teams individually developing enhancements in their respective branches. An acceptance process was established such that, when individual features were completed and deemed satisfactory, they would be integrated with the root of the project.

6.3 Future Work

After processing the responses from the feedback forms, handed in on the 18th of March 2015, the team decided that the following enhancements should be made:

- enable Google Analytics plug-in in the admin interface, so that the course coordinators can gather statistics with respect to website's traffic;
- extend the web application, so that it can be used as a learning tool for other courses taught within the School of Life Sciences. The impetus behind this was a result of the positive feedback received from the questionnaire, in regards to similar digital resources, as seen in Figure 5.4(b);
- add an About section, including an online feedback form and a brief description of the development team;
- create an FAQ section. Here the students could find necessary support, or review any issues encountered when browsing the website;
- attach more informative video thumbnails to each video.

6.4 Reflection

The team found this project highly rewarding, as the web application will help Molecular Methods students during their studies at the University of Glasgow. Valuable experience was gathered through weekly client meetings, where members of the team were able to put in practice the concepts taught in the Professional Software Development 3 course. During the past six months, the team learned to work with new technologies, varying from version control systems (i.e. Github)

to web frameworks (i.e. Django). In addition, the pair programming technique and agile practices brought a significant contribution in meeting the project's deadlines and clients' requirements.

Were the team to start this project again, extensive research, regarding the advantages and disadvantages of the different versions of technologies used (i.e. Bootstrap and Django), would have been completed prior to the implementation process.

Bibliography

[1] Asana. Asana. www.asana.com/.

[2] L. Azzopardi and D. Maxwell. *Tango with Django: a beginners guide to web development with Django 1.7*. GitHub, 2014.

[3] PREMIER Biosoft. PCR primer design guidelines. www.premierbiosoft.com.

[4] Web Application Security Consortium. Cross site scripting. projects.webappsec.org, February 2011.

[5] Oracle Corporation. MySQL. www.mysql.com/.

[6] Oracle Corporation. Oracle database. www.oracle.com/us/products/database/overview/.

[7] Promega Corporation. Promega App. www.promega.co.uk/resources/mobile-apps/.

[8] Jeff Croft. Django architecture. DjangoArchitecture-JeffCroft.png.

[9] PostgreSQL Global Development Group. postgresql.org/.

[10] Michael Hartl. *Ruby on Rails Tutorial: Learn Web Development with Rails*. Addison-Wesley, 2012.

[11] Marijn Haverbeke. *Eloquent JavaScript*. GitHub, 2014.

[12] Dwayne Richard Hipp. SQLite. sqlite.org/.

[13] A. Holovaty and J. Kaplan-Moss. *The Django Book*. GitHub, 2009.

[14] Imperavi. Redactor Text Editor. imperavi.com/redactor/.

[15] GitHub Inc. Github. www.github.com/.

[16] Wikimedia Foundation Inc. Sql injection. wikipedia/SQLInjection.

[17] Christopher Grant Jones, Rose Liu, Leo Meyerovich, Krste Asanovic, and Rastislav Bodik. Parallelizing the web browser. In *Proceedings of the Workshop on Hot Topics in Parallelism*, 2009.

[18] PythonAnywhere LLP. www.pythonanywhere.com/.

[19] Erik Nygren, Ramesh K Sitaraman, and Jennifer Sun. The Akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[20] M Otto and J. Thornton. Bootstrap. github.com/twbs/bootstrap.

[21] Donald Slish. Restriction Mapping. faculty.plattsburgh.edu/donald.slish/Restmap.html.

[22] Erik Wilde, Dilan Mahendran, and Brad Andrews. Anatomy of an advanced web application. dret.net/lectures/web-spring11/webapps-advanced.

Appendix A - Evaluation Questionnaire

7045

Molecular Methods Questionnaire 2014 DGC

We are looking for feedback on the digital resources we have developed for the Molecular Methods course. Can you please respond to the following:

What is your degree subject?

The app was easy to use

○ strongly disagree ○ disagree ○ neutral ○ agree ○ strongly agree

The app enhanced my understanding of Molecular Methods

○ strongly disagree ○ disagree ○ neutral ○ agree ○ strongly agree

I would like similar digital resources available for other aspects of my degree

○ strongly disagree ○ disagree ○ neutral ○ agree ○ strongly agree

Give further details

When have you accessed the digital resources

○ in the lab ○ outwith the lab ○ both

How did you access the app

○ phone ○ tablet ○ lap-top ○ desktop

Which parts of the app were useful; circle all that apply

Lab links Quizzes Videos Timer Calculations Converter Glossary Search facility

The videos helped me understand aspects of Molecular Methods

○ strongly disagree ○ disagree ○ neutral ○ agree ○ strongly agree

Which further resources or videos would you have found beneficial?

What do you feel worked well with the app?

4688070452

Continue overleaf

5907070457

Registering on the app was an easy process

○ strongly disagree ○ disagree ○ neutral ○ agree ○ strongly agree

Did you like the design of the app

○ strongly disagree ○ disagree ○ neutral ○ agree ○ strongly agree

Any further feedback

Other students would learn to use this app very quickly

○ strongly disagree ○ disagree ○ neutral ○ agree ○ strongly agree

I was confident using the app

○ strongly disagree ○ disagree ○ neutral ○ agree ○ strongly agree

Any further comments