

A Modular, Reactive Approach to Digital Education

by

Ben Follington

School of Information Technology and Electrical Engineering,
The University of Queensland.

Submitted for the degree of
Bachelor of Engineering
in the field of Software Engineering

2015

74 Jubilee Terrace
Bardon, 4065
Tel. 0438 168 458
November 9, 2015

Prof Paul Strooper
Head of School
School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia, Q 4072

Dear Professor Strooper,

In accordance with the requirements of the degree of Bachelor of Engineering in the division of Software Engineering, I present the following thesis entitled "A Modular, Reactive Approach to Digital Education". This work was performed under the supervision of Dr. Jim Steel.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and has not been previously submitted for a degree at The University of Queensland or any other institution.

Yours sincerely,

A handwritten signature in black ink, appearing to be 'Ben Follington', with a long horizontal stroke extending to the right.

Ben Follington.

Contents

1	Introduction	1
1.1	Project Purpose	1
2	Related Work	3
2.1	Digital Education Theory	3
2.1.1	Connectivism	3
2.1.2	Learning Objects	4
2.1.3	Reusable Learning Objects	5
2.2	Recommendation Systems	5
2.2.1	Content Filtering Recommendation	6
2.2.2	Collaborative Filtering Recommendation	6
2.2.3	Hybrid Recommendation	7
2.3	Graph Theory	7
2.4	Existing Approaches to Digital Learning	8
2.4.1	University Approaches	8
2.4.2	Tutorials	9
2.4.3	Massively Open Online Courses	10
2.5	Existing Approaches to Modular Learning	11
2.5.1	Khan Academy	11
2.5.2	Treehouse	12
3	Project Design Overview	13
3.1	Key Features	15
3.2	System Overview	15

4	System Implementation	16
4.1	Technology Stack	17
4.1.1	Back-end	17
4.1.1.1	Database	17
4.1.1.2	Application	18
4.1.2	Front-end	19
4.1.2.1	React	20
4.1.2.2	Flux	21
4.1.2.3	SASS	22
4.1.2.4	Build System	23
4.2	Domain Model of System	24
4.2.1	Account	24
4.2.2	Project	25
4.2.2.1	Metadata	25
4.2.3	Topic	25
4.2.4	Learning Module	26
4.2.4.1	Comment	26
4.2.5	Content Block	26
4.2.6	Feedback Data	27
4.3	Interface	27
4.3.1	User Interface Flow and Functionality	28
4.3.1.1	Tasks	28
4.3.1.2	Intended User Interface Usage	28
4.3.1.3	Login Form	30
4.3.1.4	Registration Form	31
4.3.1.5	User Dashboard	32
4.3.1.6	Project Creation	33
4.3.1.7	Project Summary	35
4.3.1.8	Content Consumption	37
4.3.1.9	Content Editor	39

4.3.2	Design Patterns	41
4.3.2.1	Message Bubble	41
4.3.2.2	Input Fields	41
4.3.2.3	Topic Cloud	42
4.3.2.4	Inline Commenting	42
4.3.2.5	Avatar and User Identification	43
4.3.2.6	Content Types	44
4.3.2.7	Recommendations	45
4.3.2.8	Parameterisation Indicator	46
4.3.2.9	External Content	46
4.3.3	Implementation	47
4.3.3.1	Flux Implementation	49
4.3.3.2	Content Editor Implementation	52
4.4	Content writing	52
4.5	Recommendation System	53
4.5.1	Content Driven Aspects	54
4.5.2	Collaboration Driven Aspects	55
4.5.3	Combination and Interpretation	56
4.5.3.1	Algorithm	56
4.6	Parameterisation Functionality	57
5	Verification and Validation	61
5.1	Code Testing	61
5.2	User testing	62
5.2.1	Results and Observations	63
5.2.1.1	Onboarding Process	63
5.2.1.2	General Interface	65
5.2.1.3	Recommendations	66
5.2.1.4	Content	67

<i>CONTENTS</i>	vii
6 Conclusion, Perspectives and Future Work	68
6.1 Limitations	70
6.1.1 Caching of Results	70
6.1.2 Worker Queue	70
6.2 Future Research & Potential Improvements	70
6.2.1 Insight Into Student Behaviour	70
6.2.2 Collaborative Authoring	71
6.2.3 Deep Collaboration	71
6.2.4 Assessment and Feedback	71
6.2.5 Live Collaboration	72
6.2.6 Recommendation System	72
Appendices	74
A Recommendation Algorithm Psuedocode	75
B User Testing Question Sheet	78
C User Testing Information and Consent	80
Bibliography	84

Chapter 1

Introduction

In 2015, educational content is more widely available than ever before. This is in no small part due to the rapid growth and spread of online, digital education. Digital education employs a non-traditional educational paradigm, focused on self-directed and self-motivated learning. There are many approaches to digital education; from University created materials, to online schools, to simple tutorials.

Digital education facilitates new learning structures, wider availability and greater financial viability. It has been shown to be at least as effective as in-person learning [1, 2]. However, up until this point, most digital learning material has been created to cater to the widest audience, with limited consideration for the specific needs of individual students. The content from these materials is also typically difficult to reuse, leading to a large time investment to produce new or updated material.

With current education distribution models, it is now possible to build modular, reusable and intelligent content, and to tailor the learning experience to each individual student. Content such as this would also reduce the time investment required to produce educational content, allowing educators to produce a larger quantity, of higher quality material.

1.1 Project Purpose

This project aims to produce a platform for the creation and consumption of modular learning materials. This will enable rapid production of new lessons and intelligent lesson plans for students.

There have been many approaches to a modular format for educational content, but this is typically accompanied by the loss of structure. This requires students to formulate their own lesson plan, which in turn can cause both intentional and subconscious avoidance of difficult concepts, and can be daunting when the student is presented with a large pool of learning resources. Before a student is familiar with a field, they cannot make an informed decision on the concepts they wish to learn, let alone the order in which they wish to learn them.

This project will specifically target intermediate-level computer science students. Students in this field, at this experience level, typically wish to create their own projects but find themselves incapable of doing so. This is due to the large number of concepts required to create full software applications. Students in the field often do not know what concepts will be required to complete a problem before attempting to tackle it, and as such are often frustrated when moving into new territory.

The learning platform produced by this project will allow students to bring their own projects, specify which broad topics they feel are relevant to their project and have a dynamic, evolving lesson plan created for them as their project develops.

This will allow students to make progress on their project without having to plan their own learning simultaneously. The potential for students to begin creating full projects, before they become experts in the relevant concepts, furthers their personal education and produces higher quality software.

Chapter 2

Related Work

A wealth of previous relevant work exists in the field of digital learning. As educators attempt to adapt traditional teaching theory to the digital platform, new models emerge from the content they produce. Specifically, the work related to this project falls into the following broad categories:

- Digital Education Theory
- Recommendation Systems
- Graph Theory
- Existing Approaches to Digital Learning
- Existing Approaches to Modular Learning

2.1 Digital Education Theory

Since first being pursued as an option, digital learning has been a topic of extensive research. The adaptation of existing pedagogy theory is not, however, straightforward and there are many aspects to the process.

2.1.1 Connectivism

Connectivism is a framework to aid in understanding the learning process [3, 4]. It is centred around the idea that true learning occurs when a student makes a connection between two ideas they have been introduced to previously [4]. The principles of connectivism state that learning is a process of connecting nodes in a graph, joining

existing ideas together to forge new ones. According to connectivism, knowledge is best gained from a diverse set of opinions and sources and the ability to combine multiple concepts is key to learning. These connections must be maintained and reinforced to cement knowledge. [3]

The core principles of connectivism, when applied to online learning, are [2, 3]:

1. **Autonomy:** learners are given control of what to learn, and when to learn it
2. **Diversity:** peers must be from a sufficiently wide network to avoid all students arriving at the same conclusions
3. **Openness:** students must feel included no matter the level of commitment they have to the course
4. **Connectedness & Interactivity:** without the ability to connect to other students, and interact with them, none of the above principles are possible

Connectivism provides a backbone for any online learning, ensuring the students can extract maximum benefit from the educational content. However, much of the online material available today from Universities and online learning platforms does not embrace all of these principles. Specifically, there is a noticeable lack of community built around University online offerings. There is also a tendency to lean too heavily on the principle of autonomy, leaving students without direction when presented with learning resources [4].

2.1.2 Learning Objects

Learning objects are a concept designed to solve a long-standing problem with educational content: can a piece of educational material be shared between courses, or institutions? If so, in what context can it be shared? If every university creates their own course on a topic, countless hours are spent reproducing the same content. Conversely, shared content reduces time spent adapting and duplicating content [5]. Learning objects are an approach to creating educational content by drawing inspiration from computer science [6]. In computer science, reuse and modularity are two of the core goals of any program architecture. Specifically, object-oriented programming is highly analogous to the structured learning materials proposed by the concept of learning objects [6]. Each learning object shares the same prototype, however they differ in a range of defined attributes.

Traditionally, courses are seen as the atomic unit of educational content. However, learning objects allow us to break courses down into individual lessons that can be composed to create a course [5, 6]. Each learning object should address only one learning objective and be able to stand alone, to facilitate reordering and restructuring of courses.

Learning objects present content authors with a greater challenge than traditional educational materials, as there is no guarantee that a given piece of content will appear before or after any other content while being presented to the student. The concept of prerequisites can be applied to learning objects to help alleviate this difficulty; however, this decreases the number of possible configurations for the learning objects.

2.1.3 Reusable Learning Objects

The idea of a reusable learning object takes the primary qualities of a learning object, and combines them with the concept of complete reusability. Effectively, this means a reusable learning object must be completely self-contained with no external references, further increasing the production effort [7]. However, the concept opens up many possibilities for sharing learning objects. A truly reusable learning object can be used year after year, by many different courses, and be updated by simply changing a single source.

Furthermore, as put forward by Downes, reusable learning objects could be organised into open repositories [6, 8], allowing anyone to draw from open pools of material in order to put together new courses. This is conceptually very similar to the current open-source landscape in software engineering, where many modern applications frameworks are developed from the composition of several other projects. Using this approach, entire courses targeting varied audiences can be created without authoring any new content. Under this system, it is the role of a course coordinator to compose the reusable learning objects into a coherent package.

2.2 Recommendation Systems

As the Internet has grown in size and scope many services and platforms have shifted their focus from a searchable database to user-focused suggestions [9]. This promotes a model of discovery rather than search, which many believe to be the future of the Internet [10]. As the amount of information available to users has grown,

consumption has become more challenging. Rather than leaving the content selection to the user, recommendation systems recommend relevant content or products to users. In practice, this consists of media such as films (Netflix¹), music (Spotify²) or reading material (Amazon³) [11]. These systems are equally applicable to learning content as there is a large pool of content to draw from, each with a depth of supplementary metadata.

The software systems designed to perform this task as known as Recommendation Systems. These can be broadly categorised into three varieties.

2.2.1 Content Filtering Recommendation

Content-filtered recommendation systems focus on the actual items that are recommended to users. It is based on the premise that if a user has previously indicated that they have a preference or dislike for an item, their preferences can be generalised to similar items [11]. The primary challenge with a content filtered system is determining how similar two items are.

Typical implementations of content filtering make use of tagging content, derived metadata, difference algorithms and string matching [11]. The overall goal of any content-filtered system is to derive a score for how relevant any single item is to any other single item. This has excellent applicability to learning objects in a learning system, as each of these follows the same overall structure.

2.2.2 Collaborative Filtering Recommendation

Collaborative filtering is an alternative approach to recommendation that focuses on user behaviour [11]. This consists of using past user behaviour, of an individual as well as their peers, to predict future user preferences.

One of the main benefits of a collaborative filtering approach is that the recommendation system is not tightly coupled to the type of content being recommended to users [11]. Often, collaborative filtering can produce more accurate and personalised results over content filtering, but can be more challenging to implement [9].

¹<http://netflix.com>

²<http://spotify.com>

³<http://amazon.com>

2.2.3 Hybrid Recommendation

By far the most popular approach to recommendation systems however is a hybrid approach [9]. This combines both content and collaborative filtered measures into a unified system. This can be performed as two distinct operations which are later merged or a combined process.

These are thought to produce the most accurate recommendations due to the large pool of information they draw from. One of the foremost implementations of a hybrid recommendation system is the Youtube⁴ video recommendation service. Youtube combines both content and collaborative filtered methods to compute both the set of potential recommendations and their ordering [12].

2.3 Graph Theory

Recommendation-based modular learning is accomplished by taking individual lessons and “joining” them together with other related modules. This effectively forms a series of nodes in a network, connected with other nodes that share similar concepts. This allows the system to be modeled using a graph of the learning network available to a student.

In computer science and discrete mathematics, a graph is a concept used to model any set of objects that have relationships with one another. A graph is defined as a set of vertices (V) and a set of edges (E). Vertices are connected together by the edges to show relationships between the various points. Graphs can either be undirected or directed, to represent any relationship between vertices or to indicate a hierarchy. [13]

Edges of a graph can also be weighted or unweighted, where the weight is defined a numerical rating assigned to the edge. The weight of an edge may be used to represent the strength of the relationship between two vertices [13]. This can be used to model the strength of a recommendation from one learning module to another.

As users follow a string of recommendations, they are traversing the adjacent edges of a graph and as such their individual path can be modelled using a sub-graph.

⁴<http://youtube.com>

2.4 Existing Approaches to Digital Learning

Digital learning is a broadly defined term, referring to any platform, body or material pertaining to education that is delivered digitally [14]. There are many varied approaches to the issue, especially regarding software education.

2.4.1 University Approaches

Universities have thoroughly embraced digital education with many institutions worldwide offering content online. Allowing students to enroll online allows Universities to reach far more students than otherwise possible, which aligns closely with the core goals of any educational body. Typically when a University offers an online course a digital adaptation of an existing course is created and offered either in tandem with the on-campus version or entirely separately. There are two common approaches to the authoring of these courses:

The first is to simply upload lecture materials (recordings, slides) and tutorial materials, providing limited guidance to online students [15, 2]. This relies on students managing their own learning and time effectively, which can attract many students but often results in a low completion rate for a given course.

This approach offers little advantage over the traditional model, and is often less effective than on-campus education. A course relying on tutor-student connections, depends heavily on how accessible the tutor is to the student. Hence, the concept of courses taught both online and on-campus simultaneously was developed. This however provides little support for students wishing to work solely online, as tutors remain difficult to reach.

The second is to upload a network of smaller lessons, along with a work-plan for students to tackle at their own pace [15, 2]. This also relies on a student's proactivity and resourcefulness, but gives them clear direction and makes online learners first-class citizens. This is often accompanied by online discussion between students, via an online forum or email. This has been referred to as an "uncourse" [2], discarding much of the existing structure that universities employ.

Downes [2] and Siemens [4] have been vocal in support of the latter of these two methodologies. They argue that successful digital education relies on connectivism, and that the current University approach typically fails to address the connectivist principles.

2.4.2 Tutorials

Online tutorials have been a popular form of learning since near the beginning of the Internet. Originating as a way of sharing knowledge that was otherwise unavailable, tutorials facilitated technical education before educational institutions had embraced online learning. Often, creators of software tools and frameworks are expected to produce tutorial material, so that prospective users can easily begin using their tools. However, if the first-party documentation is lacking, it is typical for third parties to create tutorials as an unofficial form of documentation.

Most third-party tutorials are created by well-meaning individuals attempting to explain challenging concepts or approaches to a problem. This typically follows after the individual has solved the problem through their own effort.

Software tutorials tend to follow a similar structure, beginning with some background as to why the author has created the content and often an example of what the final product will be as can be seen in the work of Case and Whittaker [16, 17]. This is followed by a model of the product, explaining the relevant objects and how they interact. This usually also includes a code foundation the user is expected to copy down or translate for their project. Any relevant algorithms or mathematical approaches are outlined next.

High quality tutorials attempt to teach without large amounts of code; however, many tutorials cannot convey the concept correctly without providing the entire source code. This can be overwhelming for students, and can often lead to broken implementations with limited understanding of the content.

It can be difficult for students to determine whether they are prepared to tackle the material in a tutorial, as there is often no structure or relation between the range of information available. These tutorials are typically developed in isolation and as such do not directly follow on from to lead to other tutorials.

When comparing the tutorial approach to the connectivist principles, there are many shortcomings. Tutorials provide no diversity in their approach or in their opinion, as they are created by a single author, and typically incorporate limited discussion.

2.4.3 Massively Open Online Courses

A relatively new phenomenon for digital education are Massively Open Online Courses, or MOOCs. A MOOC is typically characterised by the combination of social networking, expertise in the field of study and a collection of freely accessible resources [18]. The main factor that differentiates MOOCs from other offerings, is the massive number of students, who self-manage their education experience. Many MOOCs and MOOC platforms have arisen to cater to the huge audience seeking education. These range from general platforms such as edX⁵ and Coursera⁶, to specialised “schools” such as Codecademy⁷.

MOOCs typically place limited expectations on students in terms of time commitment, participation or prerequisites. They embrace connectivism in almost all aspects, focusing on generating a feeling of comradeship and community.

MOOCs have been shown to be a highly effective learning paradigm, with extensive evidence in support of them [1, 2]. They have been shown to be especially effective for part-time students, who would otherwise be unable to dedicate enough time to understand the content. By making use of new approaches to learning that the traditional classroom cannot provide, MOOCs target a far wider audience than previously thought possible for educational content.

Many MOOCs make use of learning objects, in that their lessons are self-contained. For example, learning on edX is organised into packages such as “Introduction to Computer Science”, “Programming with C#” and “Introduction to Psychology” [2]. While these lessons may be conceptually related, they are developed separately. These lessons hold the potential to become true reusable learning objects, though they are not organised on any higher level. These modular pieces are expected to be discovered individually, and combined at the will of the student.

⁵<https://www.edx.org/>

⁶<https://www.coursera.org/>

⁷<https://www.codecademy.com/>

Despite their success and many benefits, MOOCs suffer from the same issues that plague online education: content duplication and lack of modularity. On edX alone, “Introduction to Computer Science”, “Introduction to Computer Science and Programming Using Python” and “Introduction to Computer Programming, Part 1” [19] cover the same content in a number of different ways. The student must manually plan their own learning paths through this content, which can be overwhelming if students are presented with hundreds of courses.

2.5 Existing Approaches to Modular Learning

There are some online “schools” or “academies” that have moved beyond the basic MOOC model, with novel approaches to content organisation and presentation. These approaches not only make use of connectivist principles and learning objects, but integrate the learning objects to form whole “learning tracks”. These tracks provide guidance for students by directing them to learning objects to explore after they feel they have sufficiently covered their current topic.

2.5.1 Khan Academy

Khan Academy⁸ is perhaps the most widespread online learning platform in the world, with over 10 million past and present students [20]. Khan Academy encompasses many fields, from Mathematics, Physics and Chemistry, to Biology, Economics and Computer Science. Content is organised into a many-tiered structure, from fields, to topics, to modules to individual lessons.

Despite the wealth of knowledge Khan Academy exposes to the world, there is limited content reuse and even some content duplication between similar fields, such as Chemistry and Organic Chemistry. The unique aspect of Khan Academy’s approach is the very fine granularity of content. Students with time constraints on their learning can still make definite progress due to this granularity. This is in stark contrast to most other platforms, which generally expect at least an hour of contact time to tackle a lesson.

⁸<https://www.khanacademy.org/>

2.5.2 Treehouse

Treehouse⁹ is a specialised learning platform targeted at web development. They make use of learning objects, referring to them as “courses” [21]. However, these courses are also organised into “tracks”. Tracks consist of many courses, and courses may belong to many tracks, resulting in a true reusable learning object architecture.

Treehouse also makes use of all of the connectivist principles through an inclusive, friendly and massive user base. With more than 120,000 students [21] around the world and a wealth of content, discussion tools and interactivity, Treehouse is an excellent model for an online school.

⁹<https://teamtreehouse.com/>

Chapter 3

Project Design Overview

Despite the success and innovation present in online learning, there are still many obstacles facing both students and educators. One such obstacle is the manual organisation of learning content, which is a process that can be performed by the student (edX¹, Coursera²) or by the teacher (Treehouse³, Khan Academy⁴). Manual content organisation is open to significant bias and potentially limits the effectiveness of a learning system if the content is organised by one single individual.

This project aims to overcome this hurdle by combining established methods of online learning with intelligent and automatic organisation of learning objects. By modelling the relationships between learning modules a graph of recommended learning pathways can be created. A possible learning graph that could be produced using this system is shown in Figure 3.1. The traversals of this graph represent the possible “learning tracks” a student can take through the content. This provides the student with a possible path to follow, without firmly deciding on one in advance.

To provide initial direction to the student’s learning, a learning track corresponds to a project the student is developing. This allows the student to pursue several learning tracks at once, and to specify different topics of interest for each of these tracks. This system will facilitate a more flexible approach to learning for each individual student. In turn this could allow more effective and engaging learning experiences.

¹<https://www.edx.org/>

²<https://www.coursera.org/>

³<https://teamtreehouse.com/>

⁴<https://www.khanacademy.org/>

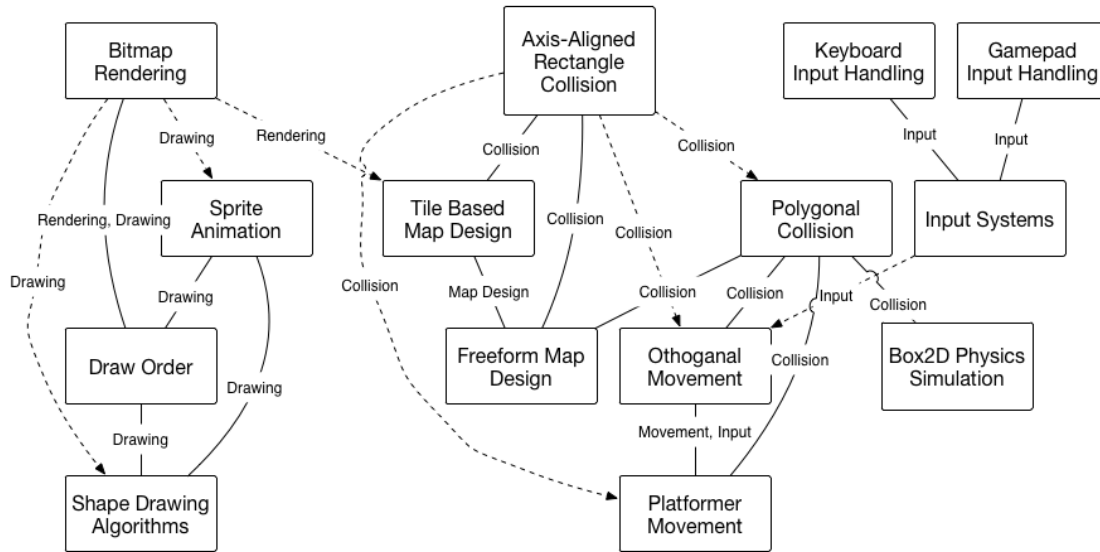


Figure 3.1: A possible learning graph for video game development, Nodes are connected by the general concepts they relate to. Prerequisites are denoted using a dashed line. Note: for readability not all possible connections are shown.

As modules lack strong relation to one another and have no explicit order of completion, it can become difficult to illustrate the connections between concepts. To maintain the cognitive flow and increase the relevance of content to a user, the content of learning modules is parameterised. This allows authors to specify sections of content to include or exclude based on a user's progress and background in the system. This can be used to highlight connections to related content, reassure users that they know enough to complete a module and to abridge or expand content based on a user's current understanding. Using this system content can adapt itself to draw connections to other concepts users have previous covered, and reinforce both ideas through the connectivist principles.

The production of content in this system does not require the course planning typically associated with producing a MOOC. This allows authors to focus on the explanation of individual concepts rather than higher level planning. Existing learning material can either be recreated in the system or simply embedded into a learning module. This content embedding allows existing online tutorials to be included in the recommendation system and allows users to discover this material naturally.

Using this embedding, any existing series of tutorials can be imported into the system rapidly and receive the benefits associated with tagging, feedback and recommendation provided by the platform.

3.1 Key Features

Given these requirements, and the core connectivist principles, the key features for the system can be defined as follows:

1. Dynamic construction of “learning tracks” by assembling learning modules automatically
2. Parameterisation of the content of learning modules to increase relevance
3. Fine granularity of learning modules, allowing flexible “learning tracks”
4. Discussion with peers of individual sections of the content
5. Flexible authoring tools to create for and import content to the system
6. The ability to include external content from the Internet

3.2 System Overview

Students begin using the system by creating a project (see Section 4.2.2) that corresponds to what they are currently working on. This can represent either a real project they are working through, or a field of study they wish to learn more about. This project stores all information about the student’s use of the system. When creating a new project users enter information about their desired areas of learning and their current experience. This information is then used by the system to create an initial set of recommended learning materials.

From this point users can begin consuming educational material on the platform. Users work through learning objects (hereafter referred to as “learning modules”) at their own pace. These are composed of text, images, code and mathematical content as well as embedded interactive content. Each of these content blocks can be parameterised using the metadata contained in the student’s current project. Additionally, each of these content blocks can be discussed with other users to further a student’s understanding. At the conclusion of a learning module users are presented with a set of 4 recommendations for what to learn next. This process continues, with recommendations being refined over time as more data is collected.

Users are also free to search for content they wish to learn and shift the focus of their project to another set of topics. The system is able to respond to this and change its recommendations rapidly to adapt. This platform gives users greater freedom, more personal content and lower cognitive overhead than existing learning systems.

Chapter 4

System Implementation

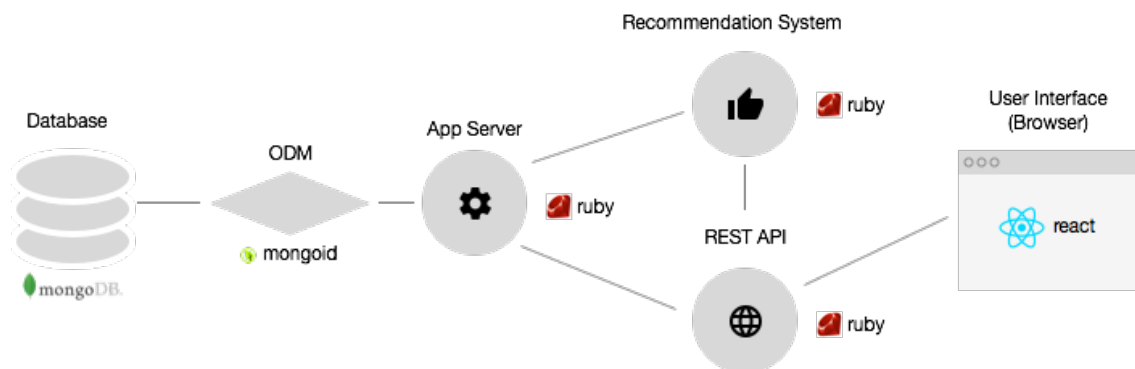


Figure 4.1: System Architecture Overview

The platform itself is implemented as a web application whose architecture is shown in Figure 4.1. The application server, recommendation system and REST API are written in Ruby¹ using the Padrino² web framework. MongoDB³ is used for persistence and is accessed via the Object Data Mapper (ODM) Mongoid⁴. The user interface is displayed to the end user in their browser and is constructed using the React⁵ and Flux⁶ Javascript frameworks. Each of these technology choices is discussed in the following sections.

¹<https://www.ruby-lang.org/en/>

²<http://www.padrinorb.com/>

³<https://www.mongodb.org/>

⁴<https://github.com/mongodb/mongoid>

⁵<https://facebook.github.io/react/>

⁶<https://facebook.github.io/flux/>

4.1 Technology Stack

To create the full application a variety of technologies were combined. The remainder of this section details each choice, split between back-end technologies (server-side) and front-end (client-side).

4.1.1 Back-end

There are two major technology choices to make for the server-side code: which database to use and what application framework to use.

4.1.1.1 Database

To persist user accounts, data and the content itself the application requires a database. The databases that are used in web application development are either SQL or NoSQL based. SQL databases are typically relational and have a strict schema that must be enforced. Conversely, NoSQL databases are either “key-value stores”, “document stores” or graph databases and do not have a schema defined. MySQL⁷, PostgreSQL⁸ and SQLite⁹ are the most common relational database systems, whereas MongoDB¹⁰, CouchDB¹¹ and Redis¹² are common examples of NoSQL databases. NoSQL databases are often faster, more easily scaled and simpler to work with [22].

MongoDB (or Mongo) was selected as the persistence layer for the system. Mongo provides schemaless storage of data, along with built-in support for sharding and replication of the database [23]. This provides a fast development environment coupled with a highly scalable database.

The BSON data format used to store documents in Mongo is easily translated to JSON for communication with Javascript. This allows for simple interplay between the server and client sides of the system.

⁷<https://www.mysql.com/>

⁸<http://www.postgresql.org/>

⁹<https://www.sqlite.org/>

¹⁰<https://www.mongodb.org/>

¹¹<http://couchdb.apache.org/>

¹²<http://redis.io/>

To interface with the main web application, the Mongoid¹³ ODM was used. Mongoid is the recommended ODM by MongoDB. Mongoid allows declarative definition of models and allows an optional “schema” to be defined on top of Mongo. Mongoid’s flexible relationships, validation and querying are designed to facilitate rapid development and iteration.

```
class Account
  include Mongoid::Document

  # Fields
  field :email,           :type => String
  field :username,       :type => String

  # Validations
  validates_presence_of :email, :username

  has_many :projects
  belongs_to :current_project
end
```

Listing 4.1: An example model declaration using Mongoid. This is a simplified version of the account model used in the system.

Listing 4.1 shows a heavily simplified model definition using Mongoid syntax. Models are defined using standard Ruby syntax and Mongoid helper methods.

4.1.1.2 Application

The server-side application was developed using Ruby. This choice was made due to Ruby’s expressive, declarative nature and its extensive use in the web application field [24]. Declarative definition of application functionality typically reduces errors in code and increases legibility to developers [25]. This is especially relevant for the development of a Representational State Transfer (REST) Application Programming Interface (API), as much of the code written simply describes the available endpoints.

¹³<https://github.com/mongodb/mongoid>

There are many available options for web application development in Ruby; the most common of these are Ruby on Rails¹⁴ and Sinatra¹⁵. However, the web application was created using the Padrino web framework. Padrino builds on Sinatra, and retains the lightweight and performant request handling of Sinatra while adding more complex web application features [26]. This includes permissions, sessions and a rendering system [26]. Padrino enables the clear syntactical definition of API routes as shown in Listing 4.2.

```
Doublejump::App.controllers "/api" do

  get :account do

    if !current_account.nil?
      send_json current_account.to_hash
    else
      send_json({success: false, error: "Not logged in"})
    end

  end

end
```

Listing 4.2: Sample controller declaration showing the concise and readable syntax Sinatra provides.

This reduces the cognitive overhead required to develop web applications and instead shifts focus to the actual application logic.

4.1.2 Front-end

The front-end (or client-side) application is a single page application (SPA) written using modern Javascript features. All Javascript is written using the ES2015¹⁶ standards, this is detailed in Section 4.1.2.4.

¹⁴<http://rubyonrails.org/>

¹⁵<http://www.sinatrarb.com/>

¹⁶<http://www.ecma-international.org/ecma-262/6.0/>

The front-end application is built using a unidirectional dataflow design pattern known as Flux¹⁷. This was developed by Facebook to provide a functional programming inspired approach to user interface programming. The “view” layer of this architecture is the React¹⁸ framework, also created by Facebook. For the visual style of the application Syntactically Awesome Style Sheets (SASS)¹⁹ is used.

Each of these technologies’ merits and basic principles are detailed below.

4.1.2.1 React

There are many challenges associated with building rich user interfaces on the web. Foremost among these is the lack of enforceable consistency in appearance and lack of insight into the rendering process. React²⁰ was developed by Facebook to alleviate these issues. React provides component-based user interface development, with a functional programming inspired rendering pipeline.

React components define a render method which maps the properties passed to a component, and the current internal state of the component, to an HTML-like intermediate representation of the Document Object Model (DOM) known as JSX²¹. An example React component definition is shown in 4.3.

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
  }

  render() {
    return (
      <div>
        <h1>{props.name}</h1>
      </div>
    );
  }
}
```

Listing 4.3: A sample component declaration using React and JSX.

¹⁷<https://facebook.github.io/flux/>

¹⁸<https://facebook.github.io/react/>

¹⁹<http://sass-lang.com/>

²⁰<https://facebook.github.io/react/>

²¹<https://facebook.github.io/jsx/>

Whenever the state or properties of a component change, the render method is invoked and the changes to the DOM are applied. Rather than completely replacing the existing elements, React checks for differences and changes the existing DOM. This has massive performance benefits as modification of the DOM is one of the most costly operations for the browser to perform [27].

The benefits of these design patterns and architecture choices, as claimed by Facebook, include [28]:

- Predictable rendering
- Ensured synchronisation of data and user interface
- Simple testing and debugging
- Performant rendering of complex interfaces
- Promotion of good coding practice through decoupling and composition

4.1.2.2 Flux

Flux²² is a design pattern pioneered by Facebook as an alternative to the Model-View-Controller (MVC) architecture. It promotes unidirectional data flow, immutability and other functional programming concepts. There are three major components to Flux:

Actions

Actions represent all possible updates to an application's current state. This can include both user input to the application interface and the result of any data requests dispatched to a server.

Stores

Stores contain both the application data and logic for processing new data. This consists of more than repositories of objects in use and can extend to anything related to user interface state. The goal of stores is to place all data that may be read by more than one "section" of the application into a common location. This helps to overcome to difficulties of cascading changes and tightly coupled data.

²²<https://facebook.github.io/flux/>

Stores subscribe to the stream of Actions that are dispatched and use these to update the data within the store. While there is no specific rule given, it is recommended to create one store per type of data.

Views

Views are provided by React in this platform. In Flux, the views translate the data in stores to a user interface representation.

Unidirectional Data Flow

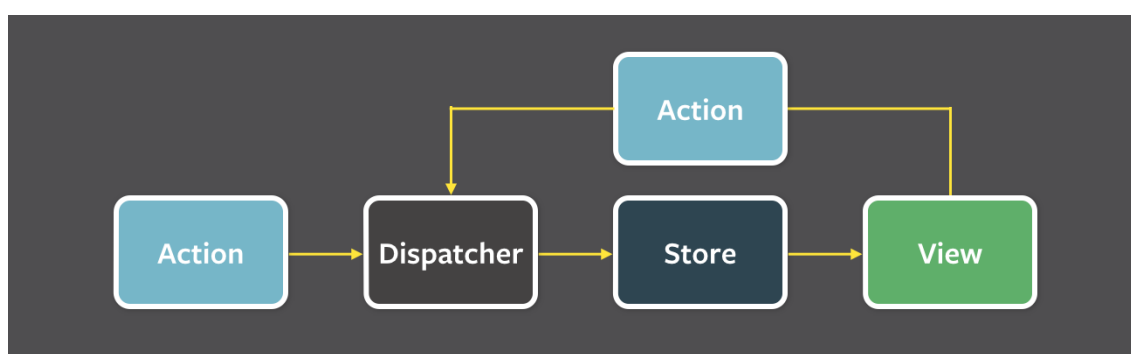


Figure 4.2: Unidirectional Data Flow as explained by Facebook[29]

The overall process of Flux can be seen in Figure 4.2. The benefits of Flux according to Facebook include [29]:

- Reduction in tight coupling and edge cases
- Obvious separation of concerns
- Simple API interaction
- Predictable application state
- Simple testing

4.1.2.3 SASS

Syntactically Awesome Style Sheets²³ (SASS) is a preprocessor for Cascading Style Sheets (CSS). It extends CSS with features such as variables, functions, mixins and plugins. Specifically, SASS is used in this project as it complements the component-driven approach used in React. Styles for components can extend and compose existing styles, reducing repetition when defining similar components.

²³<http://sass-lang.com/>

4.1.2.4 Build System

All front-end code in the system is written using ECMA Script 2015 (ES2015), a modern standard for Javascript. This ES2015 code also contains UI components defined using JSX. ES2015 introduces many new features to Javascript, many of which have arisen from the community. These include a formal class syntax, decorators, promises and “arrow” (`=>`) closure definition. While these features do not change what is possible in the browser, they greatly simplify development and increase readability of the code. However, this code does not run natively in the browser without preprocessing. Several projects have attempted to create a transpiler from ES2015 to ECMA Script 5 (the previous, well supported standard). These include Tracuer²⁴ and Babel²⁵, of which Babel is more commonly used.

Babel To build a deployable copy of the front-end system this ES2015 code is be converted, or transpiled, to ES5. To accomplish this transpilation the Babel project is used. Babel is capable of reading ES2015 and JSX code and emitting equivalent, though less readable, ES5.

Browserify When delivering Javascript to the client in the browser, the number of HTTP requests made are of great importance. A large number of requests can result in an extended page load time, frustrating users.

This is often circumvented by bundling all Javascripts modules and libraries into a single file. Browserify and Webpack are some of the many tools that can perform this task. For this project, Browserify is used as it is more widespread. Browserify parses Javascript, extracts the dependencies for every module and bundles these all together into one file. By using closures these modules retain their default scoping and normal operation.

SASS Similarly, before delivering CSS to the browser it must be converted from SASS into regular CSS and bundled into a single package. This is accomplished using the official tools developed by the SASS team.

²⁴<https://github.com/google/traceur-compiler>

²⁵<https://babeljs.io/>

Sourcemaps & Minification Due to the obfuscation caused by the “compilation” of both the Javascript and CSS, debugging can become problematic. To combat this sourcemaps are generated from both the SASS compiler and Browserify. Sourcemaps can be loaded by browser debugging tools to map errors in the final code to the original sourcecode.

This allows developers to enjoy the benefits of these technologies while mitigating the drawbacks for debugging.

Gulp Gulp is a task runner that can be used to invoke the Browserify and SASS build processes. It also includes functionality to “watch” the filesystem and triggering a build as files change. This allows the latest code to be available in-browser as soon as a save has completed.

4.2 Domain Model of System

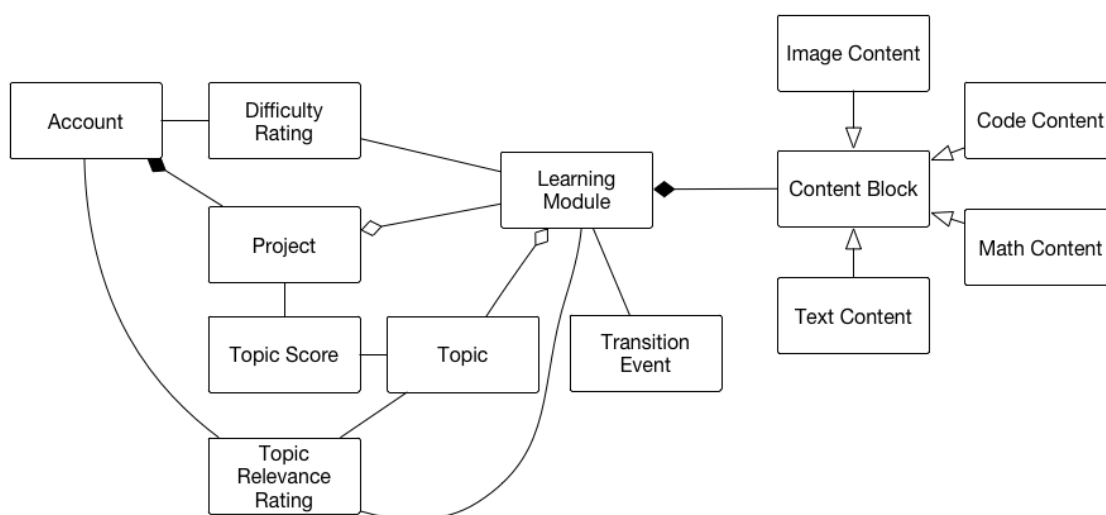


Figure 4.3: Domain Model of the System with all acting Entities

The domain model of the system (see Figure 4.3) includes user related data, learning content and all information collected from users during their use of the system. Each entity, its purpose and certain specific areas of interest are detailed below.

4.2.1 Account

All users of the system require an account to access or author content. Accounts simply record the user’s name, email, username and password.

Passwords are stored in an encrypted form through use of bcrypt²⁶. Accounts are authenticated by encrypting the password supplied by users and comparing the hashes. This is a standard method of authentication and provides a second layer of protection should the database be compromised.

4.2.2 Project

A project in the system encapsulates a student's learning in a particular field. All metadata and topic biases are scoped within an individual project and do not influence the system as a whole. Projects also contain the set of learning modules completed by a user so far.

4.2.2.1 Metadata

Metadata is contained within a project and may consist of the topic biases a project has, which modules have been completed in the project and general key-value storage. Any arbitrary key-value data can be stored in the project via the REST API. This can be accessed by content when rendered to make use of user input.

4.2.3 Topic

Topics are used to tag and categorise content within the platform so it can be interpreted and suggested by the recommendation system. Each learning module has a set of topics it pertains to, as does each project. For example, a lesson explaining how to use pointers in C may have the topics: Memory Management and Programming Basics. Topics are a key component of recommendations in the system. As outlined in the Recommendation System (section 4.5), they are the most significant factor in recommendation.

²⁶<http://bcrypt.sourceforge.net/>

4.2.4 Learning Module

Learning modules represent a single unit of learning on the platform. These are the actual entities that are suggested to students via the recommendation system. Learning Modules are categorised by tagging them with the key topics they relate to. Each learning module is built using a series of Content Blocks, which are outlined in Section 4.2.5.

The learning module provides a uniform interface for dealing with content in the platform and allows the system to function without any explicit knowledge of the content held within a module.

4.2.4.1 Comment

Comments are embedded within Learning Modules using Mongo's embedding functionality. Comments can therefore only be queried within a given learning module. This keeps data encapsulated and simplifies data transfer. Additionally, in a large scale MongoDB system embedding has positive performance implications [30]. When querying for the comments relating to a Content Block, no filtering of results needs to occur as the comments are automatically retrieved when accessing the Content Block.

4.2.5 Content Block

Content Blocks are used to build the Learning Modules in the system. A content block takes one of four forms; each of these has a distinct appearance when rendered to users and a slightly different set of fields.

1. Markdown
 - Stores a markdown string which can be filtered and parameterised in real time (see 4.6)
2. Code
 - Stores source code and the language it is written in
3. Image
 - Stores a URL to access the image
4. Math
 - Stores source LaTeX markup to generate math

4.2.6 Feedback Data

To increase the accuracy and personalisation of learning content several types of feedback data can be stored in response to user actions. The usage of all data types in the recommendation system is explored in full in Section 4.5.

Difficulty Rating

Users can rate the difficulty of particular Learning Modules after completing them. This is used to determine how early a module should be recommended to a user.

Topic Score

For each topic a user has encountered in the system an experience score is recorded. This represents how much exposure a user has to a topic. The higher this value the more likely it is that a Topic will be recommended to a user.

Topic Relevance Rating

Users can indicate which Topic they feel is most relevant to a Learning Module after completing it. This is used to scale Topic Scores using relevance calculations.

Transition Event

When a user transitions between two Learning Modules the system records this path and will, over time, suggest the most common paths even if they conflict with the expected set of recommendations.

4.3 Interface

Learning platforms depend heavily on their user interfaces [31]. No matter how relevant content is, or how easy it is to comprehend, the interface itself can prevent users from ever learning. It follows that the user interface and user experience design is core to success of this project.

4.3.1 User Interface Flow and Functionality

The user interface of the system has been designed in response to the tasks that users complete on the platform. Within this section each view in the application is detailed in terms of which tasks it enables how it enables them.

4.3.1.1 Tasks

For a user to make use of the platform a set of tasks the platform much facilitate was devised:

1. Register Account
2. Log Into Account
3. Create a Project
4. View all Projects
5. Receive Recommendations
6. Search for Learning Modules
7. Undertake Learning Module
8. Discuss Learning Material
9. Resume Project
10. Enter Feedback
11. View the Reason for a Recommendation
12. View Previous Completed Modules
13. Create New Content
14. Test Parameterised Content

4.3.1.2 Intended User Interface Usage

Generally there are a few paths users can take through the user interface on the platform. These are defined by the sequence of views that users visit on these paths. Each view, its purpose and its functionality are detailed in the follow sections.

A: New User Joining Platform

Users register for the platform and are directed to their dashboard to introduce them to the concept, from here they are immediately directed to create a new project and then begin learning within that project, see Figure 4.4.

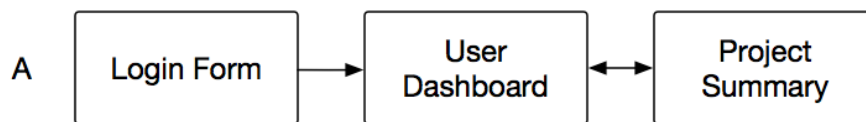


Figure 4.4: UI flow for a new user joining the platform

B: Existing User Resuming Work

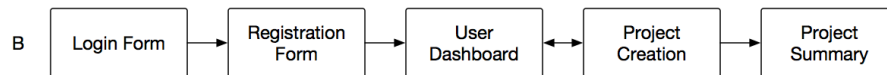


Figure 4.5: UI flow for a user resuming their work

After logging in a user is taken to their dashboard where they can resume either the last project they undertook, create a new project or resume any older projects, see Figure 4.5.

C: Browsing Content and Receiving Recommendations

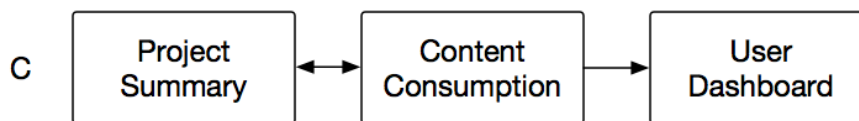


Figure 4.6: UI flow for a browsing content and receiving recommendations

Users receive their recommendations on the Project Summary page and can begin content consumption. Users can return to the dashboard or project summary from any learning module, see Figure 4.6.

D: Editing Existing Content

If a user is an administrator they are permitted to edit and update existing learning modules, see Figure 4.7.

E: Create New Content

If a user is an administrator they are permitted to create a new learning module from their user dashboard, see Figure 4.8.

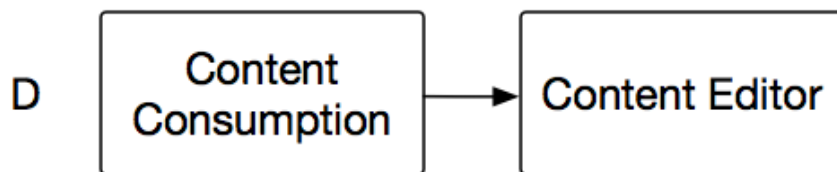


Figure 4.7: UI flow for a user editing existing content

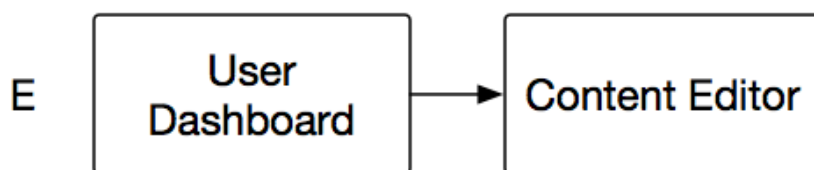


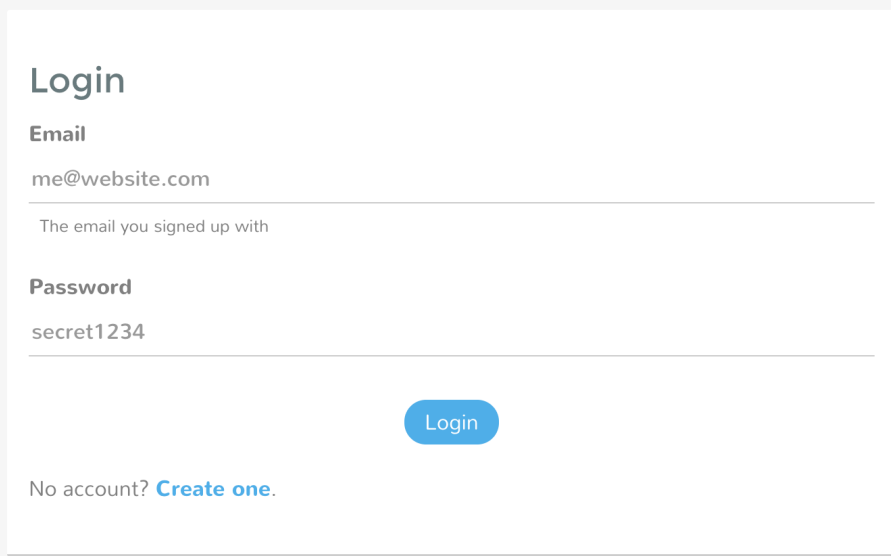
Figure 4.8: UI flow for a user creating new content

4.3.1.3 Login Form

Tasks: 2

The login form (see Figure 4.9) is straightforward and implements the standard set of two fields. The user’s email is used for authentication as they are free to change their username. Each field has a brief description of its purpose in addition to the label.

If a user does not have an account they can choose to create a new account using the “Create One” link below the submission button. If an authentication error occurs the user is immediately informed without clearing the form fields. After login succeeds users are directed to their dashboard.



The image shows a login form interface. At the top, the word "Login" is displayed in a large, bold, dark font. Below it, the label "Email" is followed by the placeholder text "me@website.com". A horizontal line separates the email field from the password field. Below the line, the text "The email you signed up with" is displayed in a smaller font. The label "Password" is followed by the placeholder text "secret1234". Another horizontal line separates the password field from the login button. The button is a rounded rectangle with a blue background and the word "Login" in white text. Below the button, the text "No account? [Create one.](#)" is displayed, where "Create one." is a blue link.

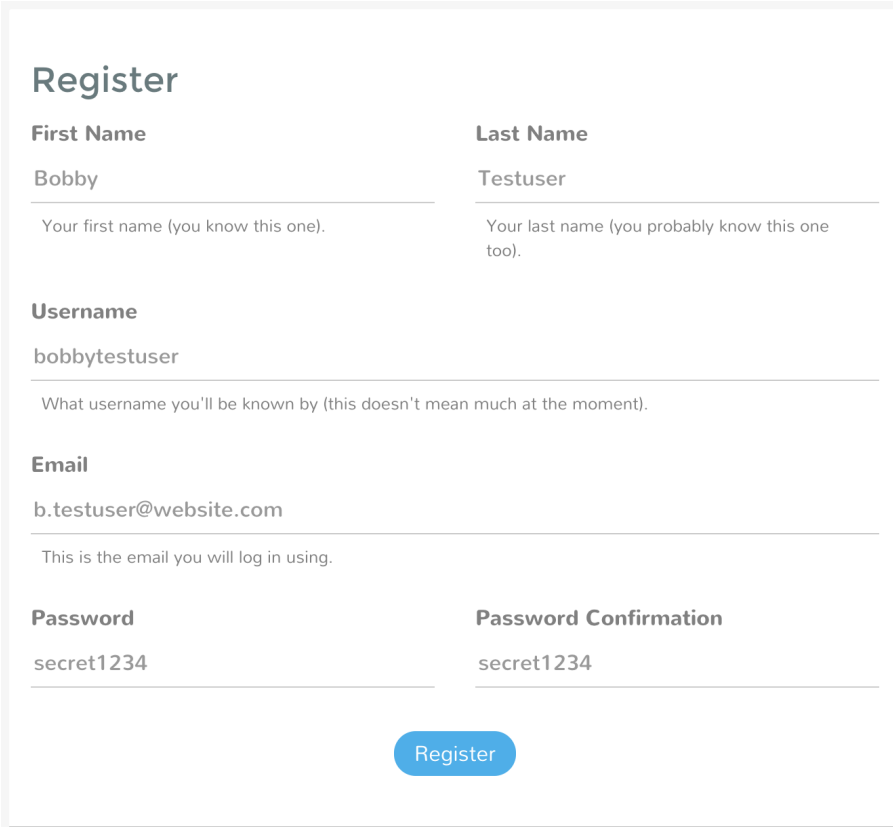
Figure 4.9: Login form interface

4.3.1.4 Registration Form

Tasks: 1

User registration (see Figure 4.10) consists of typical information entry including the user's name, their preferred username, their email and their password. The form makes use of the extended description area for each input field (see Section 4.3.2.2) to explain the purpose of each field. Placeholder values are also used to indicate the type of input expected from the user.

HTML5 validation is used on the form to catch user errors, such as invalid email addresses, before actually submitting to the server. Any error messages during registration are displayed above the submission button so users can immediately see them and take action. After registration succeeds users are directed to their dashboard.



The registration form is titled "Register" and is contained within a light gray border. It features several input fields with labels and placeholder text:

- First Name:** Labeled "First Name" with the value "Bobby". Below the input is the placeholder text "Your first name (you know this one)."
- Last Name:** Labeled "Last Name" with the value "Testuser". Below the input is the placeholder text "Your last name (you probably know this one too)."
- Username:** Labeled "Username" with the value "bobbytestuser". Below the input is the placeholder text "What username you'll be known by (this doesn't mean much at the moment)."
- Email:** Labeled "Email" with the value "b.testuser@website.com". Below the input is the placeholder text "This is the email you will log in using."
- Password:** Labeled "Password" with the value "secret1234".
- Password Confirmation:** Labeled "Password Confirmation" with the value "secret1234".

At the bottom center of the form is a blue rounded rectangular button with the text "Register".

Figure 4.10: Registration form interface

4.3.1.5 User Dashboard

Tasks: 3, 4, 9

The user dashboard (see Figure 4.11) gives a listing of all the projects a user has created as well as highlighting the most recent active project. This is intended to speed up the process of resuming where a user left off when returning to the system.

The dashboard allows users to manage their projects (see Section 4.2.2) and gives them an overview of every action they have undertaken so far on the platform. The grid-like display for projects was chosen over a simple list to increase scannability and decrease the effort required to locate an existing project.

The dashboard also facilitates creation of new projects via the action button displayed in Figure 4.11.

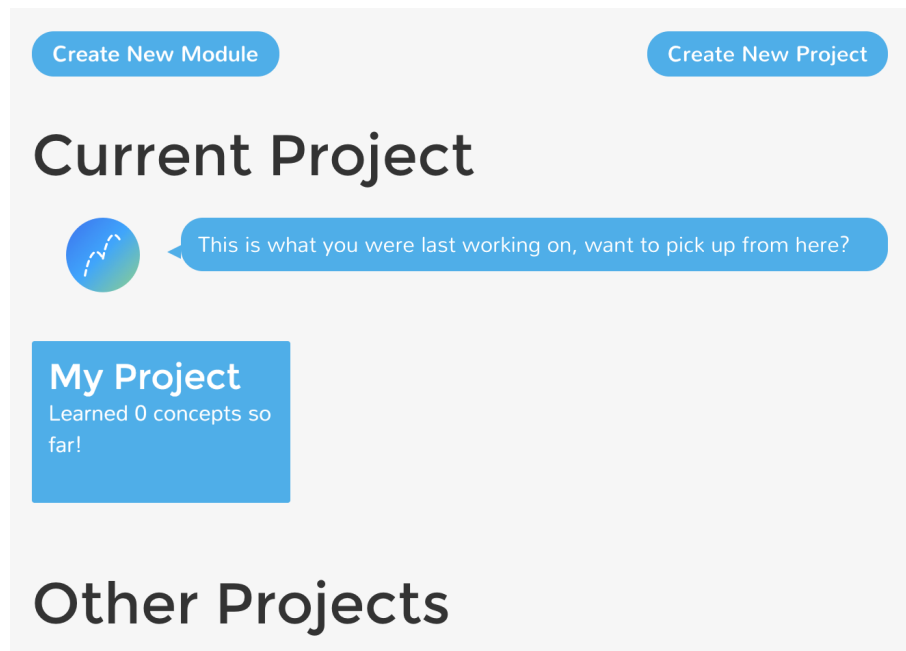


Figure 4.11: User dashboard display, the current project is the project the user was last active in

4.3.1.6 Project Creation

Tasks: 3

Before creating a project users are given a brief introduction to what a project is and how they should treat them (see Figure 4.12). Users can then go on to name their project, specify the areas they are interested in learning about and their prior experience.

Start a New Project



On doublejump, when you're learning about something we'll track your progress using something called a "Project". A project can refer to something you're working on and all the learning related to that (like "My Website"), or just a path of learning you're going down (like "Web Development").

Within your project we'll suggest things we think are relevant to your learning and attempt to understand your goals.

Figure 4.12: Introductory message bubble for creating a new project

Both the areas of interest and areas of prior experience are presented using the Topic Cloud (see Section 4.3.2.3 and Figure 4.13) interface. Users are required to nominate exactly 3 areas of interest to ensure that the system has enough information for the first round of recommendations. This limit is also imposed to prevent users from selecting too wide of a range of topics and producing meaningless recommendations.

Project Name

Give this project a name, it could be for something you're working on or just a description of what you want to learn.

Before we get started, we need to find out some information about this project.

Select 3 Topics You Would Like To Learn About

These are the topics you think this project should teach you or you would like to learn.

Collision Detection Geometry 2D Rendering Python Programming Basics
User Input Pixels File Handling String Manipulation Web Development HTML
CSS Javascript Page Layout Game Movement Programming Arrays
Asynchronous Programming Image Editing Memory Management

Select Up To 3 Topics You Are Already Comfortable With

These are topics have some experience with already.

Collision Detection Geometry 2D Rendering Python Programming Basics
User Input Pixels File Handling String Manipulation Web Development HTML
CSS Javascript Page Layout Game Movement Programming Arrays
Asynchronous Programming Image Editing Memory Management

Start Project

Figure 4.13: Project name input field and tag clouds displayed during project creation

Similarly, users can select up to three areas of prior experience but may choose to nominate as few as they wish. This is due to the fact that a student may literally have no areas of experience, or may feel their experience is irrelevant to the project they are undertaking.

4.3.1.7 Project Summary

Tasks: 5, 6, 11, 12

The project summary page shows users their current set of recommendations (see Figure 4.14). Reasons for these are available on hover (see Section 4.3.2.7). This is displayed first as it is likely users will be returning to the platform to start a new session of learning.

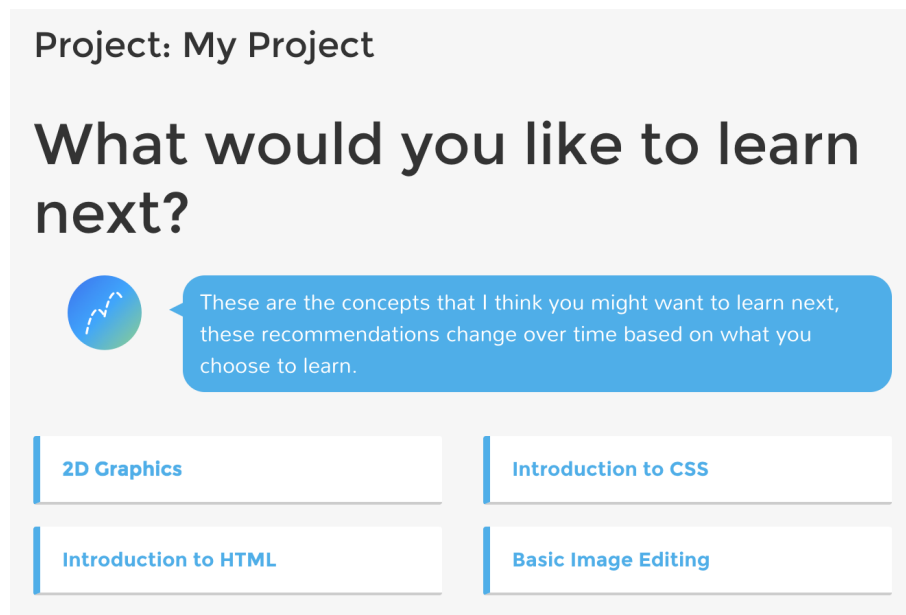


Figure 4.14: Project summary recommendations

Additionally users are able to search for particular content from the project summary (see Figure 4.15), selecting any module – including those they have already completed and those for which they have not satisfied the prerequisites. This is to ensure that content is as easy to find as possible and that users are free to choose to learn any content (see Section 2.1.1).



Figure 4.15: Project summary search field

Finally users can also view a listing of all the modules they have completed so far in the project (see Figure 4.16). This gives users an overview of the progress they have made and a sense of satisfaction if the list contains a large number of entries.

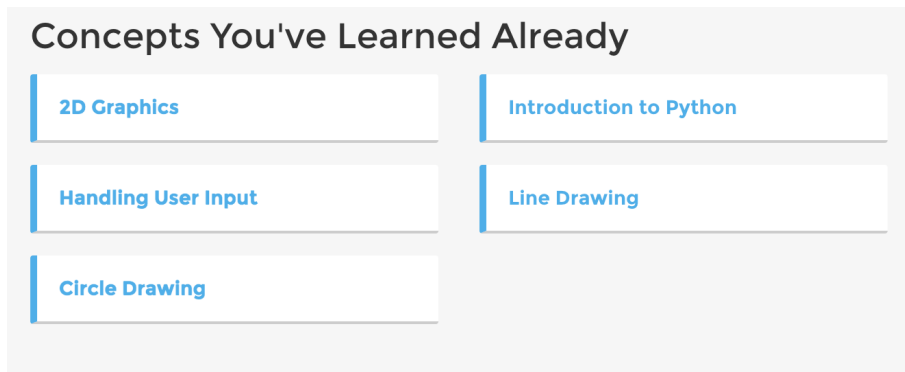


Figure 4.16: List of complete modules shown in the project summary

4.3.1.8 Content Consumption

Tasks: 5, 7, 8, 10, 11

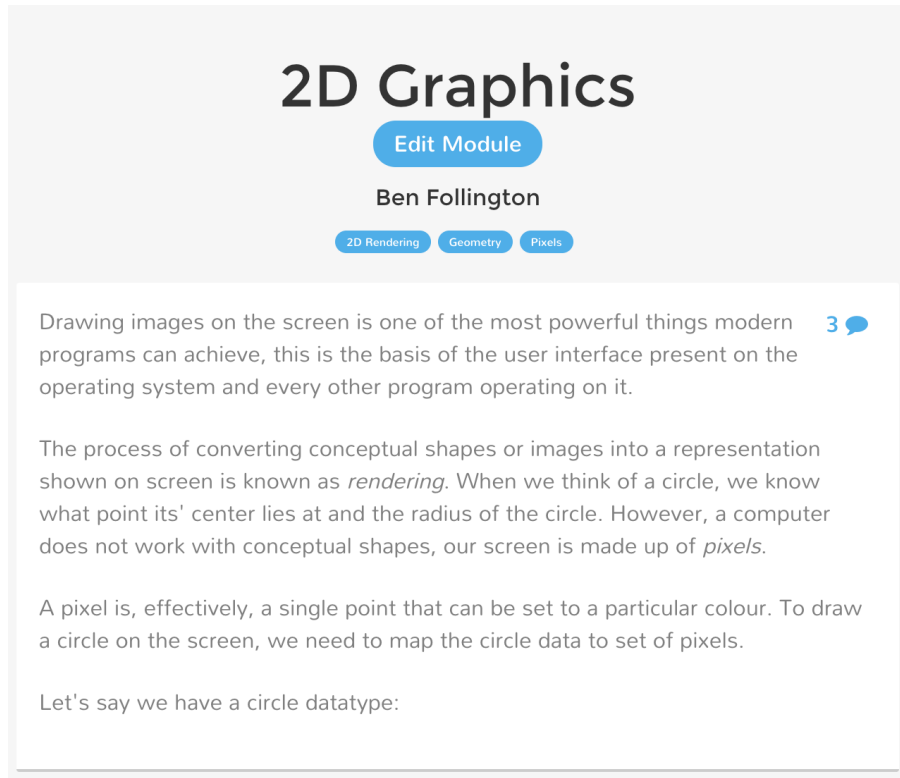


Figure 4.17: Title, author and background information regarding a learning module

All learning modules are displayed with the title, author and list of topics (see Figure 4.17) above the main body of the content. The body of the learning module is composed of a series of Content Blocks (see Section 4.2.5). Each of these blocks can be discussed using the comment panel (see Section 4.3.2.4 and Figure 4.18).

Upon completion of the module – indicated by the user clicking the finish button – users can enter feedback and receive their next set of recommendations as shown in Figure 4.19.

This explicit completion of a module is used to ensure users can leave the page without influencing their future recommendations. Automatically flagging a module as complete on the users part could lead to unexpected behaviour, which contradicts the Nielsen design principles of user control, freedom and error prevention [32].

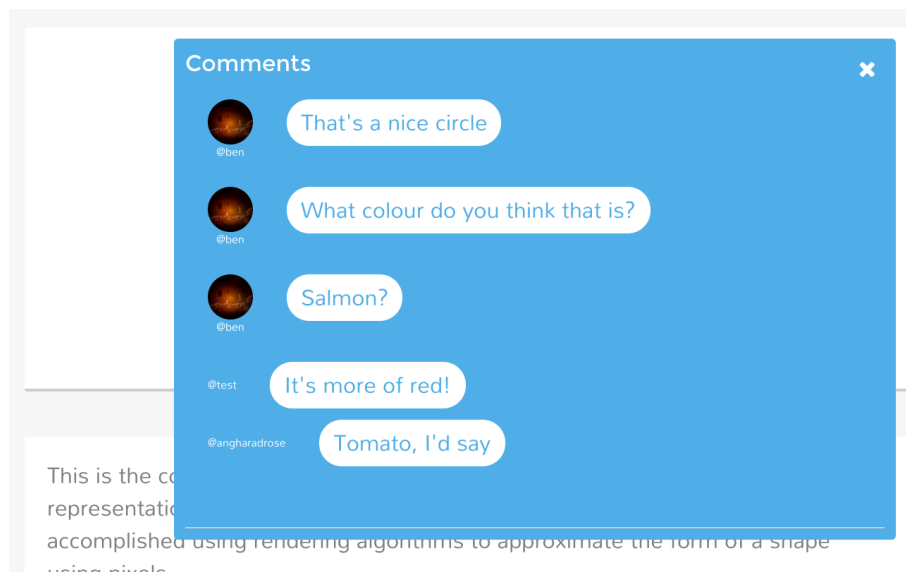


Figure 4.18: Commenting interface

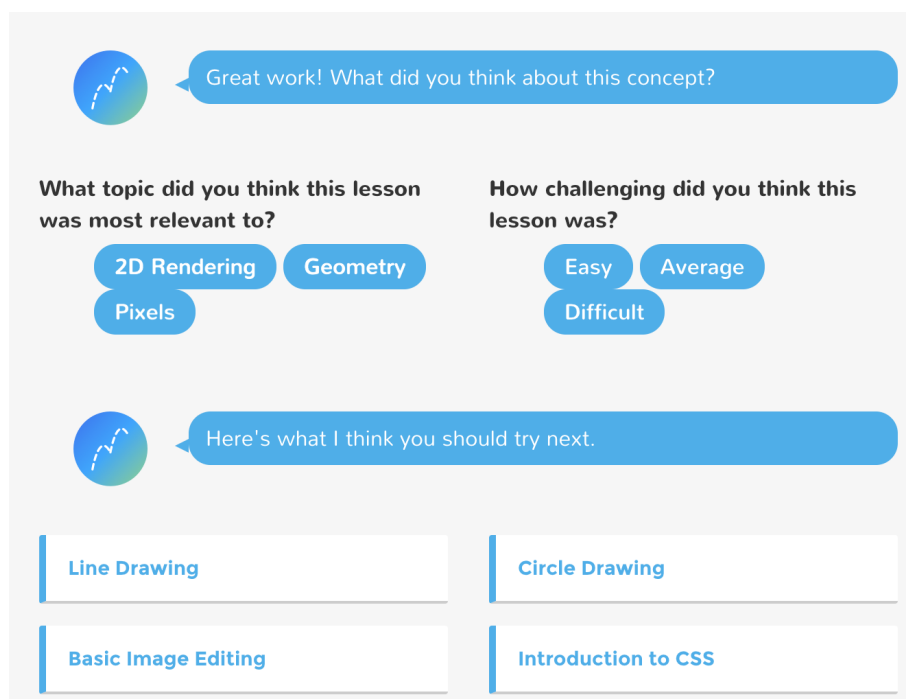
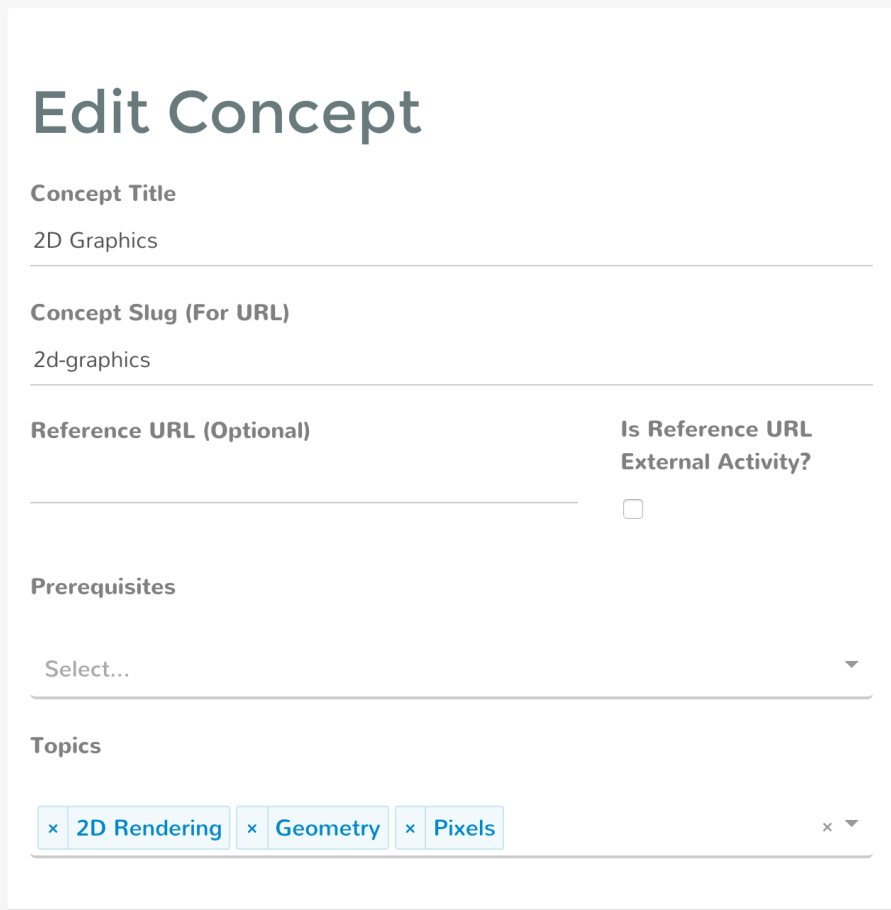


Figure 4.19: Module completion feedback and Recommendations

4.3.1.9 Content Editor

Tasks: 13, 14



The screenshot shows a web form titled "Edit Concept". It contains several input fields and a checkbox. The "Concept Title" field contains "2D Graphics". The "Concept Slug (For URL)" field contains "2d-graphics". The "Reference URL (Optional)" field is empty. To the right of this field is a checkbox labeled "Is Reference URL External Activity?". Below these fields is a "Prerequisites" section with a dropdown menu currently showing "Select...". At the bottom is a "Topics" section with a list of tags: "2D Rendering", "Geometry", and "Pixels", each with a small 'x' icon to its left. A small 'x' icon is also visible at the end of the topics list.

Figure 4.20: Content editor interface

The content editor (see Figure 4.20) allows an author to compose learning modules by creating and arranging content blocks. To make the authoring process as easy as possible several features have been included:

Drag and Drop

The content blocks can be rearranged through drag and drop as this is one of the most physically relatable ways to organise a list. As stated by the Nielsen design principles, design should mirror the real world whenever possible [32].

Identical Edit and Consumption View

A potential challenge of many online authoring systems is the discrepancy between the content during editing and during consumption. This is often mitigated through a preview process, however this can also prove cumbersome. To attempt to alleviate this issue and aid the production of content, all content blocks are displayed identically for both the author and the student.

Parameterisation Testing

The creation of parameterised content closely resembles the process of programming. Each block of content is effectively a function which takes project metadata as a parameter and returns the final output for a student. Given the potential complexity of this process debugging tools have been included in the content editor. Authors can simulate the metadata of a student at any time using the JSON input field above the main content (see Section 4.6 for parameterisation detail).

4.3.2 Design Patterns

A description of the significant design elements and the reasoning behind their inclusion follows. Where relevant design choices refer to the task they are intended to facilitate.

4.3.2.1 Message Bubble

When a section of the system is introduced to a user, a “message bubble” is displayed with the platform’s logo. These messages deliver basic tutorial information in conversational format as indicated in Figure 4.21.

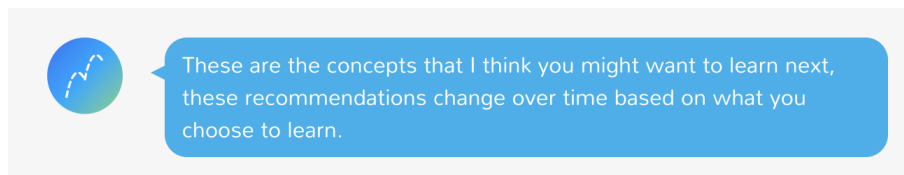


Figure 4.21: The message bubble is used to explain the platform to users in a conversational manner

From the original Nielsen design principles [32], it has been shown that users respond positively to designs that are already familiar. In the modern era of digital interaction, messaging has become ubiquitous [33]. In turn conversational voice has begun to replace formal voice in user interfaces, as designers begin to target their work towards particular cultures [34]. Users often respond more positively to conversational language, as it is more relatable [35].

4.3.2.2 Input Fields

Throughout the platform all input fields are styled to be consistent. As shown in Figure 4.22 the input fields resemble traditional paper form fields with a simple label and underline to indicate the need for user input. Each input field also supports display of a small descriptive sentence beneath the input (see Figure 4.22).

This was chosen over more common patterns such as tooltips, blocks of prose or “help icons” to fit with the overall conversational and friendly design of the system. This solution combines the immediacy of prose with the proximity of tooltips and is easily ignored if users do not require assistance.

Email

me@website.com

The email you signed up with

Figure 4.22: A sample input field used on the Login Page. Input fields include a label, placeholder text and an optional explanation field.

4.3.2.3 Topic Cloud

When a user creates a new project in the system, they are presented with an input field to name the project and two questions prompting users to select a range of topics to learn about.

These topics resemble a “tag cloud” and are designed to avoid implying any ordering or relationship between topics. Topics can be selected through a click and in response to this they become enlarged through an animation. This gives users feedback that their click was accepted by the system and allows them to see all the currently selected items at a glance. This is done to ensure users understand the current state of the form, in accordance with the Nielsen design principles [32]. Topic clouds are used during project creation and are visible in Figure 4.13.

4.3.2.4 Inline Commenting

While users are working through learning modules they have the ability to leave comments on each content block, as well as read the comments left by other users in the past. To indicate that there are comments to be read a “speech bubble” icon, along with a count of the comments, is displayed. If there are no comments to display then an “add” icon is displayed instead (see Figure 4.23).

When this icon is clicked the comment panel (see Figure 4.18) animates out from the icon, making the source of the panel clear to the user. The comment panel itself is designed to resemble common messaging interfaces. As discussed in Section 4.3.2.1, messages have become ubiquitous and reusing the same design pattern reduces the cognitive overhead for users, as they already understand the interface.

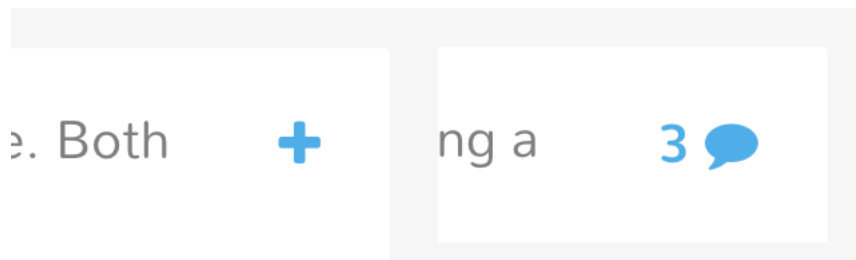


Figure 4.23: Left: "Add Comment" icon, Right: "Read Existing Comments" icon

The inline approach to comments has several benefits over a combined comments section or a separate discussion board:

- It is immediately clear which content comments pertain to
- Comments can be viewed on the same page as the content being discussed, helping to affirm content
- Users can add comments immediately when they feel the need to do so

This is a pattern that is gaining momentum across the web, with Medium²⁷, Livefyre²⁸ and Carnival²⁹ gaining popularity.

4.3.2.5 Avatar and User Identification

Users are identified by a username and an avatar (if one is defined) throughout the system. When a user is referred to by username it is prepended with the @ symbol. This is a defacto standard on the internet to imply that a term is a username. Originally established by Twitter, this convention has spread to the majority of platforms where users may want to refer to one another.

The originally intended meaning on Twitter was that a message was directed "at" a user, but this shifted to a method of typesetting the username over time. It is possible to implement usernames without this pattern, however as with most design choices on the platform familiarity and clarity are assist in retaining user engagement.

²⁷<http://medium.com>

²⁸<http://web.livefyre.com/>

²⁹<http://carnivalapp.io/>

4.3.2.6 Content Types

All learning modules are constructed from the four basic Content Blocks: Markdown, Image, Math and Code. Each of these have particular display characteristics outlined below, but it is important to note that all four types are edited using the same interface:

For writing Markdown, Math, Code examples or entering image URLs the Ace editor³⁰, developed by Cloud9, is embedded in the content editor (see Section 4.3.1.9).

Code Display In addition to being used for editing code samples, the Ace editor is also used for displaying code to the user. This simplifies the process of content creation for authors as the code entered in the editor is displayed identically to the end user (see Figure 4.24).

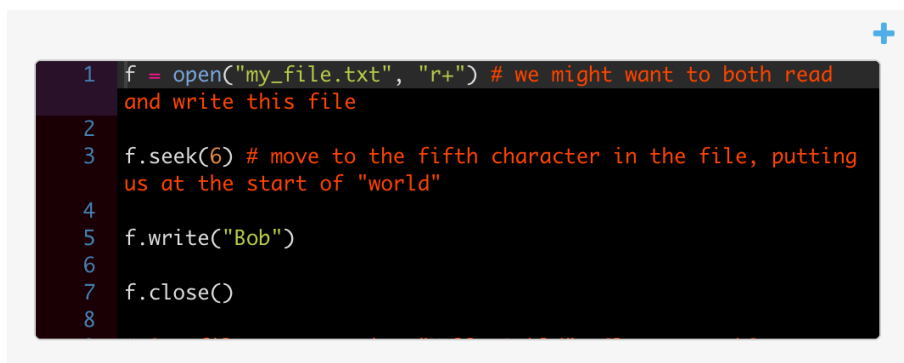
A screenshot of the Ace editor interface. The editor has a dark background with a light blue border. The code is displayed with syntax highlighting: line numbers 1-8 on the left, and code text in various colors (green for keywords, orange for strings, white for comments). The code is: 1 f = open("my_file.txt", "r+") # we might want to both read and write this file 2 3 f.seek(6) # move to the fifth character in the file, putting us at the start of "world" 4 5 f.write("Bob") 6 7 f.close() 8 A small blue plus sign is in the top right corner of the editor area.

Figure 4.24: An example of a code sample as displayed to a user

In similar systems code samples are often displayed using a separate system. These support syntax highlighting but do not provide advanced features such as bracket matching and code folding. Using Ace for display allows users to imagine they are actually editing the code themselves, as they would in a real development environment. Once again, this choice was made to enhance the consistency, familiarity and clarity of the system.

Image Display When an image is displayed it is centered on screen and restricted in size. This is to prevent images overwhelming the content and to enforce consistency between all images.

³⁰<https://ace.c9.io>

Markdown Display Markdown content is displayed using the same baseline font size as the rest of the application. The font used throughout the platform is Nunito³¹ and was selected for its readability and playful tone. In general, sans-serif fonts have been found to be more readable on a computer display [36], and a more “serious” font choice can severely affect the user impression of the system [37].

When displaying the resulting text to users, markdown is converted to HTML using the popular Marked³² library.

Math Display The math content type is intended to display a single equation, and as such equations are displayed in the center of a content block. This resembles the typical presentation of equations in academic writing and draws attention to the equation in a similar way to an image. Equations are written using LaTeX syntax, as this is the common standard for typesetting mathematical content. To render the source LaTeX into HTML, the KaTeX³³ library from Khan Academy is used. KaTeX was chosen over the more common MathJax³⁴ as its performance is far better, despite supporting a smaller syntax set.

Embedded Content In place of the standard series of content blocks, learning modules can also be composed of external content. In this case the body of a learning module is replaced with an inline frame (iframe). This effectively embeds another webpage within the platform, see Section 4.3.2.9 for more detail. Visually this is chosen over opening the external reference itself to indicate to the user that they remain within the platform.

4.3.2.7 Recommendations

Recommendations are displayed to users on both the project summary and at the conclusion of each learning module. Whenever these recommendations are displayed it is in a set of four choices (see section 4.5.3) laid out horizontally (or in a grid view when the page is condensed), see Figure 4.19 for example usage.

³¹<https://www.google.com/fonts/specimen/Nunito>

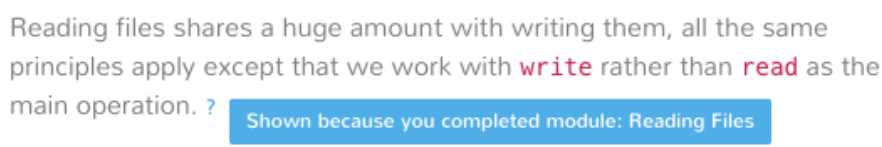
³²<https://github.com/chjj/marked>

³³<http://khan.github.io/KaTeX/>

³⁴<https://www.mathjax.org>

Recommendations are displayed from left to right, with the most relevant recommendation on the left. For English speakers this ordering is implied by the traditional left-to-right reading direction. However, the goal of this layout is to retain user control and allow users to contradict the recommendation system and move along a different path. For each recommendation, users can hover over the card to display information on the reason the module was recommended. This is to inform the user and give them the freedom to overrule a recommendation made by the system.

4.3.2.8 Parameterisation Indicator



Reading files shares a huge amount with writing them, all the same principles apply except that we work with **write** rather than **read** as the main operation. ?

Shown because you completed module: Reading Files

Figure 4.25: Parameterisation is indicated by a "?" symbol, and shows a tooltip on hover

When content is parameterised, a user may wonder why a particular piece of text has been inserted. To give users some insight into the function of the system inserted content is tagged with a question mark icon (see Figure 4.25). When a user hovers their cursor over this icon the reason for the insertion is displayed. This allows a user to either choose to focus on the content, if the reason *does* apply to them, or ignore it if they feel it is irrelevant or inaccurate.

By giving users the ability to learn more about the system's functionality they retain their control over their learning. This aligns with the Nielsen design principle of giving users visibility into the system status [32].

4.3.2.9 External Content

Learning modules can consist of external content. Authors can enter a URL to be displayed in place of the body of a learning module. This allows a large portion of content on the Internet to be used in the system with minimal effort. Given the wide availability of online tutorials this facilitates unification of content on the web and reduces the effort required to discover it.

This enables the system to function as a content aggregation platform as well as an integrated learning system. This also helps to combat content duplication and reproduction by facilitating the discovery of educational material that students may not otherwise find.

External content embedded in a learning module is annotated with topics and fits into the recommendation system via the same interface as regular content. It also retains the same feedback options post-completion and provides a set of recommendations. This addresses many of the issues associated with third-party tutorials as explored in Section 2.4.2.

4.3.3 Implementation

Using React and Flux, the UI of the application is implemented declaratively using functional programming techniques. This facilitates the component-based design used throughout the application.

When a page is loaded, the page's controller dispatches a series of actions that trigger retrieval of data from the REST API. These actions can be expressed declaratively using decorator annotations as shown in Listing 4.4.

```
...
@data(
  props => [
    fetchProject(props.project),
    fetchModule(props.module),
    fetchTopics(),
    fetchNextModules(props.project, props.module),
    fetchContents(props.module)
  ]
)
export class ViewModulePage extends React.Component {
  ...
```

Listing 4.4: Sample data annotation declaration. This lists the actions that must be performed before rendering this component.

The `data` annotation accepts a function that maps the properties of a component to an array of actions. These actions are dispatched when the component is mounted on the page. Before the a page is rendered, each of these declared dependencies are satisfied and passed into the application data store. Requests to the API are performed simultaneously; each request creates a promise when dispatched and the completion of all requests is detected using `Promise.all` to consolidate these promises.

The data received in response to these actions is passed through the reducers defined using Redux. Once this data has been placed in the application data store, it can be accessed by the component. This is also performed declaratively using a second annotation shown in Listing 4.5.

```
@connect(  
  state => (  
    {  
      modules: state.module.items,  
      projects: state.project.items,  
      contents: state.content.items,  
      topics: state.topic.items,  
      account: state.account.currentAccount.data  
    }  
  )  
)  
)
```

Listing 4.5: Sample of the connect annotation this gives the mapping of global state to component properties.

This `connect` annotation accepts two functions. The first of these maps the app state to a series of properties which are in turn passed to the component. These values become accessible through `this.props` during rendering, e.g. `state.module.items` is mapped to `modules`, which becomes readable through `this.props.modules`.

On each page load the following sequence of events occurs:

1. Initial page load
2. Dispatch data retrieval actions
 - These are dispatched concurrently
 - Each action returns a promise that resolves upon completion
3. Responses are received and processed (see Section 4.3.3.1)
4. All promises resolve
5. Component is rendered using store data

This architecture minimises repeated code for components, prevents internal data management (by creating a uniform data storage interface) and allows declarative definition of data dependencies.

4.3.3.1 Flux Implementation

In the system the Flux (see Section 4.1.2.2) data flow pattern is used. This is facilitated through Redux³⁵, an implementation of the design pattern. Redux takes a functional programming inspired approach to Flux and relies on three core concepts:

1. State Tree

Rather than the concept of stores promoted by Flux, Redux combines all stores into a single state tree, represented in JSON. This tree itself is immutable and can only be influenced by a Reducer. In the platform this tree is divided into subtrees with each subtree storing one particular type of entity. On the frontend the following entities are stored:

- Learning Modules
- Projects
- Topics
- Accounts
- Content Blocks
- Notifications

2. Actions

³⁵<http://redux.js.org/>

Actions in Redux are almost identical to the specification given by Flux. They are simple data types that contain either parameters or results of a request and are processed by Reducers to update the state tree.

3. Reducers

Reducers are a functional programming approach to action responses. They accept the previous application state and an action and return the updated state. Reducers must be pure functions, that is to say, they must use only the data passed to them to create the output and must always return the same output given the same input.

Reducers can be composed and nested to create arbitrarily complex state trees. This pattern and implementation have several key benefits:

- Synchronous processing of actions
- Visualisation of the state tree
- Immutability of state data
- Predictable responses to actions

Which together help to prevent race conditions and ambiguous states in the user interface.

```
// Initial state
state = {
  items: [
    "one",
    "two"
  ]
}

...

// Action creation method
function addItem(item) {
  return {type: ADD_ITEM, item: item};
}

// Dispatch action to add an item
dispatch(addItem("three"));

...

// Reducer definition
(prevState, action) => {
  switch (action.type) {

    case ADD_ITEM:
      return Object.assign({}, state, {
        items: prevState.items.concat(action.item)
      });
  }
}

...

// Final state
state = {
  items: [
    "one",
    "two",
    "three"
  ]
}
```

Listing 4.6: An example state tree action and reducer system using Redux with psuedo correct syntax.

4.3.3.2 Content Editor Implementation

As discussed in Section 4.3.1.9 each content block in the content editor is displayed identically to both the author and the student. This is accomplished by recycling the same React component in both use cases. By simply disabling the `editable` flag on a content block component it can be shown to students without any authoring tools.

Each of these content blocks can also be reordered using a drag-and-drop system. This functionality is provided by `dragula`³⁶ with a custom React binding. When saving a module each content block is responsible for persisting its own content. Once all modules have been saved the parent component inspects the DOM to extract the ordering of the blocks within the module.

This method of persistence allows content blocks to preserve their comments even if the containing learning module is completely rearranged or the content block's content is edited.

4.4 Content writing

For the system to be effectively evaluated and tested a library of content was developed. This was comprised of a range of programming and technical topics selected to give both significant breadth and depth to the system. This allows students using this proof-of-concept implementation to find their distinct areas of interest within the library.

Many learning modules also include examples of external activities, code snippets, LaTeX style math rendering and referencing content from around the internet. This is intended to demonstrate the capability of the content-editor which was used to create all content (see Section 4.3.1.9). The learning modules created are outlined in Table 4.1 below.

³⁶<http://bevacqua.github.io/dragula/>

Table 4.1: Listing of the learning modules in the system.

Module Title	Topics
Line Drawing	2D Rendering, Geometry, Pixels
Axis-Aligned Bounding Box	Collision Detection, Game Movement
Collision Detection	Collision Detection, Game Movement
Introduction to Programming	Programming Basics, Programming, Python
Introduction to Python	Python, Programming Basics
Handling User Input	Python, User Input, Programming Basics
Reading Files	Python, String Manipulation, File Handling
Writing Files	File Handling, Python, String Manipulation
Introduction to HTML	HTML, Web Development, Page Layout
Introduction to CSS	CSS, Web Development, Page Layout
String Manipulation	String Manipulation, Programming Basics, Programming
2D Graphics	2D Rendering, Geometry, Pixels
Circle Drawing	2D Rendering, Geometry, Pixels
Introduction to Pointers	Programming Basics, Memory Management
Basic Image Editing	Image Editing, 2D Rendering

These topics and areas of interest were determined through analysis of other MOOCs targeting programmers, first year university topics and popular areas of development. Content was created with a primary focus on breadth of content, rather than depth. This allowed more diverse and thorough testing of the recommendation system and more realistic user testing. It should be noted that due to both the subject matter of these modules and the parameterisation system available to authors, content requires some background in programming to create.

4.5 Recommendation System

As outlined in Section 2.2, there are three general categories of recommendation system. The project employs a hybrid approach to content recommendation. Throughout this section the relevance “score” will be used to describe how relevance is computed. This is an value that increases as two learning modules are more relevant to one another.

4.5.1 Content Driven Aspects

To judge how related two individual learning modules are, the system first compares which topics (see Section 4.2.3) overlap between the modules. The underlying assumption behind this is that the more topics shared between the modules, the more relevant to one another they are.

Module to Module

The relevance score of two modules to one another is computed by comparing their respective sets of topics. It should be noted that modules receive a relevance score boost for each topic they share but do not receive a penalty for topics that are not shared. This allows more “lateral” recommendations, promoting exploration of a breadth as well as a depth of learning.

The specific details of this process are detailed in Section 4.5.3.

Module to Project

In addition to this inter-module measure, there is also a calculation of how relevant a given module is to the user’s current Project (see Section 4.2.2). The users project contains a map of topics the user has encountered to a “topic score” for the user with each topic. When a user completes a module, the topic score for each topic that module is tagged with is increased. These updated scores are then stored in the project. When calculating a module’s relevance, the score for each of its topics is retrieved from the current project and used to boost the module’s relevance score.

From this it can be understood that one unit of topic score corresponds to completion of one module associated with that topic. Following from this, one unit of relevance score is equivalent to one unit of topic score.

Prerequisites

Modules can also be related to one another by defining prerequisites. A module that is a prerequisite for others is assumed to be related to its dependent modules. When a prerequisite is completed the modules that depend on it are given a large relevance score boost.

4.5.2 Collaboration Driven Aspects

The collaboration driven aspects of the recommendation system rely on recording information both passively and actively from users.

Each time a user moves from one learning module to another, by selecting a recommendation, the system records this as a Transition (see Section 4.2.6). Each transition provides a small boost in relevance for the module in the same situation for every other user on the platform. Over time this allows users to completely override the system's recommendations.

After the completion of a learning module users can optionally give active feedback to the system. This is accomplished via two questions they can answer:

1. What topic did you think was most relevant to this module?

Users can choose from a list of all topics the module is tagged with, and this nomination allows the system to assign a weighting to each topic-module combination. By default a module has an equal weighting between all topics, but this shifts over time as more users give feedback.

This impacts the recommendation system by multiplying the score a user has in a topic by its weighting. The goal of this feedback is to crowdsource the "true" weighting of a learning module's topics.

2. How challenging did you find this module?

Users can select from "Easy", "Average" or "Difficult". This facilitates calculation of an average module difficulty. A more difficult module will be recommended later in a student's learning path. This is accomplished by assigning a penalty to a module's relevance score, corresponding to its difficulty.

4.5.3 Combination and Interpretation

Whenever a set of recommendations is presented to the user, all of these facts are combined to produce a set of 4 modules to choose from. The user is given a small number of modules to make the decision simple, but to retain control of their learning project. This is an important factor for implementation of the connectivist principles (see Section 2.1.1). The exact number shown was chosen by experimentation and visual appropriateness and other numbers of recommendations could be tested (see Section 6.2 for Future Work).

A high level overview of the algorithm used to compute recommendations is given below. A psuedocode version of this algorithm can be found in Appendix A.

4.5.3.1 Algorithm

1. The last module a user completed is selected (referred to as `last_module`)
2. For each topic the user has experience with, a map (`topic_lookup`) is created between topic id and topic score
3. For each learning module (`learning_module`) in the system, the relevance score is computed using `last_module` and `topic_lookup`:
 1. The number of transitions (`transitions`) between `last_module` and `learning_module` are counted
 2. For each topic (`topic`) `learning_module` is tagged with:
 1. The relevance score is increased by the topic score a user has for `topic`
 2. The number of times `topic` has been nominated as the most relevant topic (`relevance_rating`) is determined
 3. The relevance score is increased by $0.5 * \text{relevance_rating}$
 3. For each topic shared between `last_module` and `learning_module` the relevance score is increased by 2
 4. If `last_module` is a prerequisite for `learning_module` then the relevance score is multiplied by 1.1
 5. The relevance score is increased by $0.25 * \text{transitions}$
4. Modules are sorted by descending relevance score
5. The first four modules that have had their prerequisites satisfied and have not yet been completed by the user are selected

The result of this process simply delivers the final recommendations to the frontend application via an API call.

All scaling factors are opinionated and tailored to yield the correct results in testing. Through experimentation on a large dataset and user group these could be significantly refined (see Section 6.2).

4.6 Parameterisation Functionality

Within a learning module, content can be parameterised when displayed to users. This is the process of altering the content in response to a set of input parameters. When a learning module is rendered in the browser, it is passed a JSON payload (see Listing 4.7) containing the user's current relationship with various topics as well as a list of modules they have previously completed.

```
{
  "learning_modules": {
    "reading_files": true
    "writing_files": true
    "introduction_to_python": true
  },
  "topic_scores": {
    "python": 3,
    "file_handling": 2
  },
  "misc_value_a": 123
}
```

Listing 4.7: An example JSON payload that could be passed to the parameterisation system.

This data can be interpreted and used to customise content displayed within the learning module. This was accomplished by adding Handlebars.js³⁷ into the rendering pipeline. Handlebars complements the goals of a markdown based content editor as it is both readable and concise. Additionally, the syntax used by Handlebars does not conflict with that of markdown or HTML. An example of markdown combined with Handlebars can be seen in Listing 4.8.

³⁷<http://handlebarsjs.com/>

```
// Handlebars template

My name is {{name}}, I have {{houses.length}} houses:

{{#each houses}}
- {{address}}
{{/each}}

{{#if isRich}}
You've got to be rich to afford {{houses.length}} houses.
{{/if}}

// Source data
{
  "name": "Ben",
  "houses": [
    "123 Fake Street",
    "122 Fake Street"
  ],
  "isRich": true
}

// Final output

My name is Ben, I have 2 houses:

- 123 Fake Street
- 122 Fake Street

You've got to be rich to afford 2 houses.
```

Listing 4.8: An example of handlebars combined with markdown syntax.

Handlebars also supports custom helper methods. To accomplish more semantic authoring of content several of these have been defined:

```
{{#hasDone <MODULE_NAME>}}
```

hasDone prints the content it contains if a user has completed a specific learning module.

```
{{#hasNotDone <MODULE_NAME>}}
```

`hasNotDone` prints the the content it contains if a user has not completed a specific learning module.

```
{{#hasExperienceWith <TOPIC_NAME> <TOPIC_SCORE_THRESHOLD>}}
```

`hasExperienceWith` prints the content it contains if a user has a topic for the topic `TOPIC_NAME` that is greater than or equal to `TOPIC_SCORE_THRESHOLD`.

```
{{#hasNoExperienceWith <TOPIC_NAME> <TOPIC_SCORE_THRESHOLD>}}
```

`hasNoExperienceWith` prints the content it contains if a user has a topic for the topic `TOPIC_NAME` that is less than or equal to `TOPIC_SCORE_THRESHOLD`.

An example usage of these methods can be found in Listing 4.9.

To export data for later or for use in another program, writing to a file is very useful.

```
{{#hasDone "Reading Files"}}
```

Reading files shares a huge amount with writing them, all the same principles apply except that we work with `write` rather than `read` as the main operation.

```
{{/hasDone}}
```

```
{{#hasNoExperienceWith "Python" 10}}
```

Python actually makes writing files as simple as possible, so don't be scared, this should be easy to follow.

```
{{/hasNoExperienceWith}}
```

```
{{#hasExperienceWith "Python" 10}}
```

Working with files in python is fairly straightforward, as you know by now.

```
{{/hasExperienceWith}}
```

For our purposes, we're going to focus on writing text files rather than working with binary.

Listing 4.9: A sample usage of the custom handlebars functionality taken from the Writing Files module in the system.

For text based content rendering occurs as illustrated by Figure 4.26.

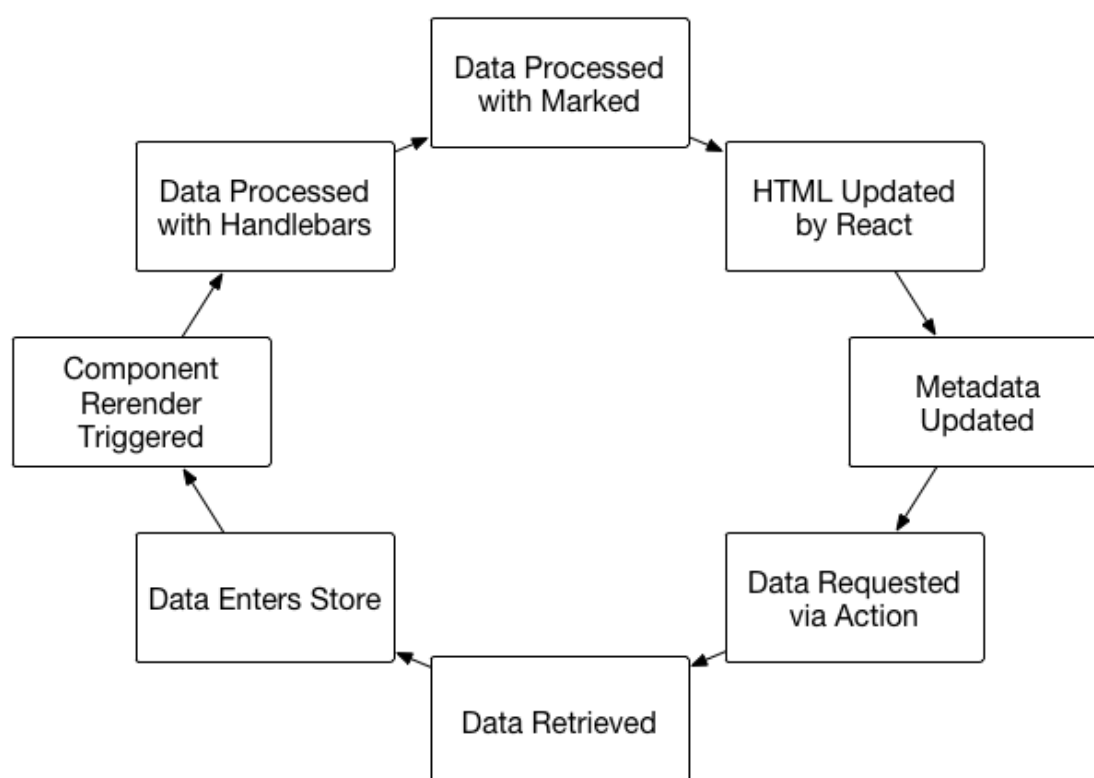


Figure 4.26: Rendering cycle for parameterised content

Additionally, the system exposes a REST API endpoint to facilitate both the storage and retrieval of metadata that can be used to parameterise content. This metadata is included in the JSON payload delivered to the front-end. Each request to this API returns the JSON payload a response which can once again be fed into the rendering pipeline detailed in Figure 4.26.

Due to the Flux-React-based architecture (see Sections 4.1.2.1 and 4.1.2.2) learning modules can respond to changes in metadata immediately without reloading the page or interrupting the user. Following from this, activities within learning modules can communicate with the platform API and cause learning modules to change while the user works through them.

Chapter 5

Verification and Validation

With a user facing platform it is important to evaluate not only the correctness of code but the usability of the interface. Accordingly both code (unit) testing and user testing were employed to evaluate this project.

5.1 Code Testing

The user interface depends on the dataflow system functioning correctly. These sections of the application are error prone due to their potentially complex nature. Accordingly the stores, reducers and actions were unit tested to verify correct behaviour. This was accomplished using the Jest¹ Javascript testing framework.

Tests are written in ES2015 and transpiled prior to execution, the general syntax for a test can be seen in Listing 5.1. These tests instantiate the data store, dispatch actions and verify that the store is updated correctly.

Unit tests relate to the following entities across 5 respective test suites:

- Account
- Content
- Module
- Project
- Topic

¹<https://facebook.github.io/jest/>

```
var SAMPLE_MODULE = {
  id: "123",
  name: "Test Module",
  topics: []
};

describe('Module State', function() {

  it('RECEIVE_MODULE', function() {
    var mod = require.requireActual('actions/Module');
    var store = getStore(redux);

    store.dispatch(
      mod.receiveModule(
        SAMPLE_MODULE.id,
        SAMPLE_MODULE
      )
    );

    expect(
      store.getState().module.items[SAMPLE_MODULE.id].isFetching
    ).toBeFalsy();

    expect(
      store.getState().module.items[SAMPLE_MODULE.id].data.name
    ).toBe(SAMPLE_MODULE.name);
  });
});
```

Listing 5.1: An abridged sample of a test for the module data store. This tests that a module is stored correctly after being received from the server.

5.2 User testing

To evaluate the both the user interface and the general user experience of the platform a round of supervised user testing was conducted. This testing focused on the general usability of the platform, the relevance of recommendations and whether the system would be useful to them or others in every day life.

Five self-proclaimed computer-literate users were recruited voluntarily based on their interest in the platform. These users were aged between 20 and 25 with no specific background in Software development or design. Before the study was conducted, ethics approval was obtained from the University of Queensland school of Information Technology and Electrical Engineering (ITEE) ethics committee. The information sheet and participant consent form are included in Appendix C.

Users worked through a series of tasks given below. For each task, users were given an initial explanation of what they should do, but did not receive guidance while completing the task. If a user was unable to continue they could ask for assistance.

While observations were recorded as users completed the tasks, users could also make specific comments at any point during the process. These tasks were:

1. Register a new account
2. Create a new project
3. Choose one of the recommended learning modules
4. Work through and complete at least three learning modules
5. Navigate to the current project
6. Search for the “Circle Drawing” module
7. Complete this module

Depending on time constraints users could opt to view the administration and content editing tools and give their comments. Upon completion of these tasks users were asked a series of questions (see Appendix B).

5.2.1 Results and Observations

Following testing, observations and user comments were collated and categorised. Each entry is accompanied by a potential course of action or consequence. All users who participated indicated they both enjoyed using the platform and that it had real-world potential.

5.2.1.1 Onboarding Process

The sign-up process is especially important to usability as it is the first impression users are given of the system and sets the tone for all activities that follow.

Registration

Some users had trouble locating the registration link on the login page, when asked this was apparently due to the ambiguous wording (“Create” vs. “Register”).

Courses of Action

The wording should be adjusted to include the word “Register” explicitly.

Message Bubble

All users read the initial message bubble in full and asked no questions regarding the purpose of projects.

Project Page Scrolling

Some users did not realise that they could scroll on the project creation page to view more content and spent some time idle before experimenting. When questioned about this, users suggested that the project name input field resembled an horizontal rule which indicated the end of the page.

Courses of Action

The width of input field could be reduced to avoid confusion. Alternative input field designs could also be considered potentially bearing more similarity to traditional text inputs.

An explicit prompt to scroll down could also be employed, however there is no set point which users will see the page cut off, due to the variability in screen sizes on modern computing devices. This makes placement of a scroll prompt challenging.

Previous Experience Nomination

While creating a new project some users did not realise that nominating topics they had “previous experience” with was optional. When questioned, users felt they still had to nominate at least one option due to the ambiguous wording (“... up to 3...”).

Courses of Action

The most obvious course of action is to clarify the wording explicitly by stating that the field is entirely optional. In general less text could be placed in the heading and more in the descriptive paragraph before the topic clouds. This would increase the motivation for a user to read this prose, rather than assume they understand how to use the form.

5.2.1.2 General Interface

Overall Appearance

All users indicated that they enjoyed the overall appearance of the platform, specifically stating the appeal of the colour scheme, whitespace and typography.

Explanations

Users commenting that the explanations given in text using Message Bubbles (see Section 4.3.2.1) for forms, input fields and UI elements were very helpful and avoid confusion when performing potentially complex interactions.

Commenting

Some users immediately opened and read comments in learning modules, while others began to ignore the comments as “they did not want to make any comments”. All users indicated that they enjoyed the animation and appearance of the comment panel, including the conversation style display of the discussion.

When the comment panel was opened, the text field was not immediately focused causing one user to type without actually entering a comment.

Courses of Action

Increase the distinction between the “Add Comment” prompt and the “Existing Comments” prompt. When the comment panel is first opened the input field should automatically be focused to allow a faster commenting process.

Current Project

Users were asked to return to the current project summary during the course of testing. It was suspected that this would be problematic as the navigation bar is used fairly infrequently. However, all users located the navigation item successfully. This is likely due to the familiarity users have with a horizontal navigation bar at the top of a webpage.

Lesson Completion

One user asked why a lesson must be explicitly finished by clicking a button. Following this the user attempted to reverse the “Finishing” process by hitting the “Back” button in the browser.

Courses of Action

Lessons could automatically complete if a user has scrolled a certain distance through the material. This could potentially alleviate the “undo” issue encountered by the user, as they would never realise the action had been performed.

Alternatively, an undo function could be implemented to allow a user to reverse the finish process. This would “unmark” the module from complete to prevent it being considered by the recommendation system.

5.2.1.3 Recommendations

Freedom

Users indicated that they appreciate the freedom of choice between all four recommendations and liked that there was no pressure to learn about a particular topic.

Weak Recommendations

Some users encountered a situation where the system could not find any strong recommendations. This is due to the limited pool of content created for testing but could also occur in well populated systems. They were confused by this and felt that perhaps the system was malfunctioning.

Courses of Action

Increasing the number of learning modules in the system will help address this concern. However, when the system is forced to make a “weak” recommendation the user should be informed. This could be displayed on hover in the “reason for suggestion” panel, or on the recommendation itself.

5.2.1.4 Content

Length

When asked users felt, in general, that the size of the learning modules provided was correct and appreciated the “bite-sized” nature of the learning. Some users suggested that they would prefer a system to have learning modules that were “too short” rather than “too long”.

Composition

Users indicated that they appreciated the rich media and logical layout of learning modules. Feeling that they could gain a deep understanding without being overwhelmed by information.

Chapter 6

Conclusion, Perspectives and Future Work

This project set out to produce an online learning platform capable of changing the way self-directed students learn. The platform was shaped by the connectivist principles and the existing approaches taken by current MOOCs. When commencing work on the platform, students indicate their interests and previous experience. This information is then used by the recommendation system to direct users toward content they may find relevant to themselves and their projects. Each learning module within the system can be discussed with other students and feedback can be given at the conclusion of each lesson.

Content can be created using the system's built-in editor or embedded from external sources. This gives authors powerful tools to create new content and a simple path to use existing material. Additionally, aside from the overall arrangement of content, the learning modules themselves can be parameterised using the information associated with user. This allows learning modules to connect ideas together without a predefined ordering.

The core goal of the system was to create a content discovery platform that alleviated the frustrations students can experience when self-teaching. By unifying the content delivery platform and automatically recommending content to students, they can focus purely on understanding content. The system was also designed to implement each of the connectivist principles (see Section 2.1.1), as these form the basis of most successful learning experiences.

From the user testing conducted (see Section 5.2.1) it was observed that users responded positively to not only the ideas behind the system but to the actual interface itself. Users indicated that they understood all the key interactions and fundamental functions of the system, often commenting positively, even on complex design choices.

Users also stated that they found the recommendations both appropriate and useful. None of the users who evaluated the platform found any issue with the lack of an overarching “course” concept and felt that the system had great potential to replace online tutorials.

Though users had no issue with the recommendation system, both development and testing showed that there is plenty of room for refinement of the recommendations (see Section 6.2.6). While users found no issue with the discussion system, comments suggest that deeper levels of collaboration could further improve the platform.

Currently, the system is usable by prospective students without any supervision. Students can register an account, set up their interests and their background and receive recommendations immediately. They can work through learning modules at their own pace, discussing content with other users and giving feedback to the platform regarding their experience.

The original goals of the project have been achieved and it has been shown through testing that this approach to online learning is both effective and has merit for self-directed learners. In a large-scale deployment of this system users could learn from a wide array of topics without ever having to intentionally search or plan their learning. They can focus on the relevant concepts and deepening their understanding. This project, in its current state, could be used as a substitute for a significant portion of third-party online tutorials.

Finally, the work and evaluation suggest that the concepts explored here do have real world applicability and should be explored further and expanded upon.

6.1 Limitations

As both the amount of user data and number of learning modules in the system grows over time, the time taken to compute recommendations will also increase. Currently recommendations are computed on demand. During testing, this was not an issue due to the scale of the dataset; however, in the future this computation could disrupt the user experience. There are several obvious avenues for improvement:

6.1.1 Caching of Results

Rather than recomputing the recommendations on demand, the system could instead generate the entire learning graph periodically and store this against each user. While this would increase the space required to store each user's information it would allow constant time lookups of recommendation data.

6.1.2 Worker Queue

In addition to caching recommendation data, the computation of recommendations could be parallelised via a worker queue. A given users recommendation data is of suitable scale to be a single "job" for each worker and is free from race conditions. This would enable simple scalability and parallelisation of the data processing without interfering with the main web application performance.

6.2 Future Research & Potential Improvements

This project represents a fully featured exploration into the concept of a recommendation based, dynamic learning platform. While user testing showed very positive results there are many avenues for potential future work.

6.2.1 Insight Into Student Behaviour

To better create and maintain content within the platform authors require insight into how students use the system. Recording statistics and displaying them to authors via a dashboard or visualisation interface could greatly inform content decisions. Some examples of notable metrics are:

- Average time taken to complete module
- Number of module completions
- Percentage of attempts compared to completions

6.2.2 Collaborative Authoring

Collaboration has proved very effective for the learning experience [38] but research also suggests that there could be significant impact on the teaching perspective [39, 40]. Throughout the online landscape collaboration has proved useful for the creation of content in general. Prime examples of this include Google Documents¹ and Dropbox Paper².

This extends beyond having multiple editors a document and includes features such as commenting, task management and real-time cooperation. Additionally this could extend beyond teacher collaboration and involve students in the content creation process [39]. Students could provide feedback or suggest edits to content they found confusing when they first attempted to understand it.

6.2.3 Deep Collaboration

While the existing commenting system provides a useful method for students to discuss content and ask questions, it is less useful for broad discussion and the inclusion of rich media. Potential enhancements could include a separate forum where posts can be linked to particular learning modules and accessed from the main learning module body.

6.2.4 Assessment and Feedback

While the current metadata stored about students appears to be a sound predictor for recommendation, it is difficult to infer how well a student understood the concepts of a learning module. Further insight into a students understanding of, rather than simply exposure to, a topic could dramatically alter and improve their learning experience.

¹<https://www.google.com/docs/about/>

²<https://www.dropbox.com/paper>

This could be achieved through integration of formative assessments, such as multiple choice quizzes, into learning modules. These could take advantage of the current parameterisation system to adapt content immediately in response to a user's answers. Formative assessment would not only provide more insight into a student's learning, but has been shown to enhance the effectiveness of online learning in general [41].

6.2.5 Live Collaboration

It has been found that users learn best when part of a community [38] as discussed by the connectivist principles (see Section 2.1.1). While the current discussion system does help convey this impression to students, there are several other ways students could collaborate during the learning process.

Real-Time

To show students that other users are working through content in real time, a student's actions could leave "footprints" on the learning module. This could potentially include the current mouse position, an indication of how far other students have read or even an integrated chat system.

Group Work

Additionally, students could work through activities or formative assessment (see Section 6.2.4) together to further their understanding. These activities could include domain specific interactive tools, similar to those implemented by golabz³.

6.2.6 Recommendation System

While the recommendation system was rating positively by users there are many possible metrics that are not currently measured. Each potential metric requires experimentation and refinement of the combination algorithm. Running experiments with a large dataset could be used to greatly increase the accuracy and intelligence of the platform. Some potential metrics for consideration include:

- Time taken to complete a Learning Module
- Number of comments left by a user

³<http://www.golabz.eu/>

- Formative assessment results (see Section 6.2.4)
- Mouse activity on the page

Additionally future improvements could include more control for content curators and authors. Collections of learning modules could be defined which have a different set or weighting of recommendation factors, allowing for dynamic and fine-grained control of the recommendations given to students.

Appendix A

Recommendation Algorithm Pseudocode

```
def nextModules(project, learningModule = null)

    # Find the "last module" that was completed
    if learningModule != null:
        lastModule = learningModule
    else:
        lastModule = project.lastModule

    topicLookup = {}
    moduleLookup = {}

    # Create mapping of topic id to topic score
    for topicScore in project.topicScores:
        topicLookup[topicScore.topic.id] = topicScore.score

    learningModules = "All learning modules in the system"

    for learningModule in learningModules:
        moduleLookup[learningModule.id] = calculateScore(
            learningModule,
            topicLookup,
            lastModule
        )
```

```
# Sort map by the values in descending order
moduleLookup.sortByValue().reverse()

finalRecommendations = []
count = 0

for entry in learningModule:
    # Do not include a module again
    if !project.learningModules.contains(entry) and
        hasPrereqs(project, entry):

        finalRecommendations.append(entry)
        count++

    if count >= 4:
        break

return finalRecommendations
```

Listing A.1: Pseudocode declaration of the nextModules method which finds the recommendations to supply to a student.

```
def calculateScore(learningModule, topicScores, previousModule = null):  
  
    score = 0  
    transitions # Number of transitions between previousModule  
                # and learningModule (0 if previousModule is null)  
  
    for topic in learningModule.topics:  
        score += topicScores[topic]  
  
        relevanceRating # Number of times this topic has been  
                        # nominated as most relevant to this module  
  
        score += relevanceRating * 0.5  
  
        if previousModule != null:  
            if previousModule.topics.contains(topic):  
                score += 2  
  
            if previousModule.dependents.contains(learningModule):  
                score *= 1.1  
  
    score += 0.25 * transitions  
    return score
```

Listing A.2: Pseudocode declaration of the calculateScore method used to find the relevance score for a single learning module.

Appendix B

User Testing Question Sheet

Bearing in mind the fact this software platform is a prototype, there are a few questions we would like answered.

Overall, did you feel you understood how the platform is intended to function?

Did you think the learning modules were too short, too long or approximately the right length?

Did you personally feel the recommendations made sense based on your inputs?

Overall, did you enjoy using the platform?

Did you have any specific complaints, comments, thoughts or suggestions?

Appendix C

User Testing Information and Consent

School of Information Technology and Electrical Engineering
HEAD OF SCHOOL
Professor Paul Strooper

UNDERGRADUATE STUDENT
Ben Follington
Telephone 0438168458
Email benjamin.follington@uqconnect.edu.au
Internet www.itee.uq.edu.au
CRICOS PROVIDER NUMBER 00025B

Participant Informed Consent Form

Project Title: A Modular Approach to Software Education

Your written informed consent to participate in this study is needed by the researchers.
Please read the following statements, and sign if you agree with them:

The nature of this project has been explained to me and I have read and understood the Participant Information Sheet provided.

I agree to participate in the study as described in the Participant Information Sheet.

I understand that my participation in this study is voluntary and that I am free to withdraw from the study at any time, without penalty and without needing to provide any reason.

I understand that any personal data collected throughout this project will remain confidential. This data will be collected and stored in a deidentified form which will not reveal my identity.

I have been informed that I can contact the researcher if I would like feedback on this study.

Participant's Full Name: _____

Participant's Signature: _____ **Date:** ____/____/____

Researcher Name/s: Ben Follington
Researcher Title / Position: Undergraduate Student
Researcher Affiliation:

Supervisor Name/s: Jim Steel
Supervisor Title/s or Position/s: Dr
Supervisor Affiliation/s: School of ITEE

Participant Information Sheet

Project Title: A Modular Approach to Software Education

Voluntary participation

We seek your assistance in a research project on a new approach to online education. Your participation in this study is completely voluntary—you don't have to take part if you don't want to. If you do decide to take part in the project, you can withdraw from it at any time without giving a reason, and without any prejudice or penalty.

Purpose of the study

Ben Follington is conducting this research at The University of Queensland in a private room. The purpose of the project is to develop a new model for online learning software. The researcher is concerned with improving usability and ease of learning, and not with evaluating the people who take part or their ability to understand educational material. The research is being conducted for improvement and evaluation of the software platform.

What you'll be asked to do or to provide

The project will involve interacting with the software platform on the researcher's computer by reading and completing learning activities for up to 50 minutes. You are asked to work through the platform unassisted if possible, you will be asked general questions regarding the usability and intuitiveness of the platform. The researcher will also collect data on which learning activities were completed and in what order. The data will be collected on the researcher's laptop (for platform data) and on premade question sheets for feedback.

Any risks to you

The potential risks of participation are not beyond the risks of everyday living. There is a slight chance that participants may become bored or self-conscious using the platform but participants are free to leave the study at any time as indicated on the consent form.

Benefits to you

You will not receive remuneration for participation. Potential benefits to you are the satisfaction of contributing to knowledge.

How your data will be kept confidential

All information about you and all data collected from you will be kept strictly confidential in deidentified form. All data will be stored on the researcher's personal, password-protected laptop. No names, addresses or any other identifying information will be included in any report on the project. Publications and data collected (deidentified and de-sensitised) as a result from this study will be made available in UQ eSpace, UQ's public institutional repository to allow the dissemination of research.

Your right to withdraw at any point

If you do withdraw from the project after it has started, the materials that you have completed to that point will be deleted and will not be included in the project.

How any concerns can be handled

If you feel distressed by the research process either during the data collection or afterwards then you raise your concerns with the research or with any officer listed in the “ethical approval” paragraph below.

Debriefing at end of study

At the end of the study you will be asked whether you would like to discuss any aspect of it. We will be happy to discuss any concerns or issues at this time. You will also be invited to offer feedback to the research team members, either verbally or in writing, at the end of the study and you are welcome to offer feedback at any time later.

How you can find out about the overall results of the research

You will be asked if you would like a copy of the summary of research findings from the phase of the project in which you participated. If you wish to receive this information, we will ask you for your contact information so that we can send this information to you. This information will be kept on a project contact list database that can be accessed only by team members.

Ethical approval

This study adheres to the Guidelines of the ethical review process of The University of Queensland and the *National Statement on Ethical Conduct in Human Research*. Whilst you are free to discuss your participation in this study with the researcher (contactable on 0438168458 or via email at benjamin.follington@uqconnect.edu.au), if you would like to speak to an officer of the University not involved in the study, you may contact the ITEE Ethics Officer via email at: ethics@itee.uq.edu.au. You are free to discuss your participation in this study with Dr. Jim Steel.

Researcher Name/s: Ben Follington
Researcher Title / Position: Undergraduate Student
Researcher Affiliation:

Supervisor Name/s: Jim Steel
Supervisor Title/s or Position/s: Dr
Supervisor Affiliation/s: School of ITEE

Bibliography

- [1] S. L. Silver and L. T. Nickel, “Are online tutorials effective? A comparison of online and classroom library instruction methods,” *Research Strategies*, vol. 20, no. 4, pp. 389–396, Jan. 2005.
- [2] J. Mackness, S. F. J. Mak, and R. Williams, “The Ideals and Reality of Participating in a MOOC,” *Proceedings of the seventh International Conference on Networked Learning 2010*, pp. 266–274, 2010.
- [3] M. Kathleen Dunaway, “Connectivism: Learning theory and pedagogical practice for networked information landscapes,” *Reference Services Review*, vol. 39, no. 4, pp. 675–685, Nov. 2011.
- [4] G. Siemens, “Connectivism: A learning theory for the digital age.” ELearnSpace, 2004 [Online]. Available: <http://www.elearnspace.org/Articles/connectivism.htm>. [Accessed: 22-Mar-2015]
- [5] R. McGreal, *Online Education Using Learning Objects*. RoutledgeFarmer, 2004.
- [6] S. Downes, “Learning objects: Resources for distance education worldwide,” *The International Review of Research in Open and Distributed Learning*, vol. 2, no. 1, 2001.
- [7] D. M. Billings, “Using Reusable Learning Objects,” *The Journal of Continuing Education in Nursing*, vol. 41, no. 2, pp. 54–55, Feb. 2010.
- [8] R. McGreal, “A typology of learning object repositories,” in *Handbook on information technologies for education and training*, Springer, 2008, pp. 5–28.

- [9] R. K. Asim Ansari Skander Essegaier, “Internet Recommendation Systems,” *Journal of Marketing Research*, vol. 37, no. 3, pp. 363–375, 2000.
- [10] Xavier Amatriain, “Kdd 2014 Tutorial - the recommender problem revisited.” 2014 [Online]. Available: <http://www.slideshare.net/xamat/kdd-2014-tutorial-the-recommender-problem-revisited>
- [11] G. Dror, N. Koenigstein, and Y. Koren, “Web-Scale Media Recommendation Systems,” *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2722–2736, Sep. 2012.
- [12] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and others, “The YouTube video recommendation system,” in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 293–296.
- [13] S. S. Skiena, “Graph Traversal,” in *The Algorithm Design Manual*, London: Springer London, 2008.
- [14] S. R. Hiltz and M. Turoff, “Education goes digital: The evolution of online learning and the revolution in higher education,” *Communications of the ACM*, vol. 48, no. 10, p. 59, Oct. 2005.
- [15] M. Courtney and S. Wilhoite-Mathews, “From Distance Education to Online Learning: Practical Approaches to Information Literacy Instruction and Collaborative Learning in Online Environments,” *Journal of Library Administration*, vol. 55, no. 4, pp. 261–277, May 2015.
- [16] N. Case, “Sight & Light: How to create 2D visibility/shadow effects for your game.” 2014 [Online]. Available: <http://ncase.me/sight-and-light/>. [Accessed: 22-Mar-2015]
- [17] R. Whittaker, “2D Particle Engines.” 2014 [Online]. Available: <http://rbwhitaker.wikidot.com/2d-particle-engine-1>. [Accessed: 22-Mar-2015]
- [18] A. McAuley, B. Stewart, G. Siemens, and D. Cormier, “The MOOC model for digital practice,” 2010.
- [19] “Courses edX.” edX, 2015 [Online]. Available: <https://www.edx.org/course>. [Accessed: 22-Mar-2015]

- [20] “Khan Academy: Stories.” Khan Academy, 2015 [Online]. Available: <https://www.khanacademy.org/stories>. [Accessed: 24-Mar-2015]
- [21] “Treehouse: Library.” Treehouse, 2015 [Online]. Available: <http://teamtreehouse.com/library>. [Accessed: 24-Mar-2015]
- [22] N. Leavitt, “Will NoSQL databases live up to their promise?” *Computer*, vol. 43, no. 2, pp. 12–14, 2010.
- [23] “Sharding.” MongoDB, 2015 [Online]. Available: <https://docs.mongodb.org/manual/sharding/>
- [24] S. Günther, “Multi-dsl applications with ruby,” *IEEE software*, no. 5, pp. 25–30, 2010.
- [25] D. Spinellis, “The importance of being declarative,” *IEEE software*, no. 1, pp. 90–91, 2013.
- [26] “The Elegant Ruby Web Framework.” Padrino, 2015 [Online]. Available: <http://www.padrinorb.com/>. [Accessed: 11-Jan-2015]
- [27] “Minimizing browser reflow.” Google, 2015.
- [28] “A Javascript Library for Building User Interfaces React.” Facebook, 2015 [Online]. Available: <https://facebook.github.io/react/>. [Accessed: 11-Jan-2015]
- [29] “Flux Application Architecture for Building User Interfaces.” Facebook, 2015 [Online]. Available: <https://facebook.github.io/flux/>. [Accessed: 11-Jan-2015]
- [30] “MongoDB Data Model Design.” MongoDB, 2015 [Online]. Available: <https://docs.mongodb.org/manual/core/data-model-design/>. [Accessed: 11-Jan-2015]
- [31] H. Park and H.-D. Song, “Make E-Learning Effortless! Impact of a Redesigned User Interface on Usability through the Application of an Affordance Design Approach,” *Journal of Educational Technology & Society*, vol. 18, no. 3, pp. 185–196, 2015.

- [32] J. Nielsen, “Enhancing the explanatory power of usability heuristics,” in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 1994, pp. 152–158.
- [33] C. Foley, E. de Leaster, S. van der Meer, and B. Downes, “Instant Messaging as a Platform for the Realisation of a true Ubiquitous Computing Environment,” *Innovation and the Knowledge Economy Issues, Applications, Case Studies, Part*, vol. 2, pp. 1051–1060, 2005.
- [34] A. Marcus and E. W. Gould, “Crosscurrents: Cultural dimensions and global Web user-interface design,” *interactions*, vol. 7, no. 4, pp. 32–46, 2000.
- [35] S. J. Boyce, “Spoken natural language dialogue systems: User interface issues for the future,” in *Human factors and voice interactive systems*, Springer, 1999, pp. 37–61.
- [36] G. Bidjovski, “Interpretation of the Selected Text Fonts with Their Typographic Features from the Web Sites of the Computers of the Users,” *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 5, no. 12, p. 71, 2013.
- [37] K. E. Sulak, *What makes a font persuasive?: An eye-tracking study of perception in American and Chinese assessment of fonts*. 2012.
- [38] R. M. Felder and R. Brent, “Effective strategies for cooperative learning,” *Journal of Cooperation & Collaboration in College Teaching*, vol. 10, no. 2, pp. 69–75, 2001.
- [39] S. Govaerts, Y. Cao, N. Faltin, F. Cherradi, and D. Gillet, “Tutoring Teachers - Building an Online Tutoring Platform for the Teacher Community,” in *Immersive Education*, vol. 486, M. Ebner, K. Erenli, R. Malaka, J. Pirker, and A. E. Walsh, Eds. Cham: Springer International Publishing, 2015, pp. 39–51.
- [40] M. J. Rodríguez-Triana, A. Holzer, A. Vozniuk, and D. Gillet, “Orchestrating Inquiry-Based Learning Spaces: An Analysis of Teacher Needs,” in *Advances in Web-Based Learning-ICWL 2015*, Springer, 2015, pp. 131–142.
- [41] K. H. Wang, T. H. Wang, W.-L. Wang, and S. C. Huang, “Learning styles and formative assessment strategy: Enhancing student achievement in Web-based learning,” *Journal of Computer Assisted Learning*, vol. 22, no. 3, pp. 207–217, 2006.