UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

COMPUTER SCIENCE HONOURS
FINAL PAPER
2016

Title: Dig-it 2.0: A Prototype System for Managing Mathematics Content

Author: Ethan Marrs

Project Abbreviation: DIGIT

Supervisor(s): Melissa Densmore, Michelle Kuttel

Category	**Min**	**Max**	**Chosen**
Requirement Analysis and Design	0	20	20
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	0
System Development and Implementation	0	15	15
Results, Findings and Conclusion	10	20	15
Aim Formulation and Background Work	10	15	10
Quality of Paper Writing and Presentation	10		10
Quality of Deliverables	10		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	0	10	
Total marks		**80**	**80**

Dig-it 2.0: A Prototype System for Managing Mathematics Content

Ethan Marrs
Computer Science Honours
University of Cape Town
mrreth002@myuct.ac.za

ABSTRACT
This paper presents the development of a prototype application for managing mathematical content. The project is based on Dig-it, Praekelt.org's mobile learning platform. Dig-it is planning to scale nationally, and a prototype was created to trial new features that might enable the platform to sustainably manage content in future. Among many things, the paper discusses an approach to the design of an e-learning Content Management System, with emphasis on automating workflows and allowing for effective arrangement of syllabus content. This involved an aim to reduce the reliance on tools external to the platform. The project was evaluated using a few metrics, including a user evaluation using the System Usability Scale. The evaluation showed that the prototype successfully provided a way to manage mathematics content, and stakeholder feedback indicated that the prototype had met its functional requirements, providing ideas and approaches for future work on the Dig-it platform.

Keywords
Computer Science; E-Learning; Education; Workflow Automation; CMS; Syllabus Management

1. INTRODUCTION
The education space has been a focus of computer science innovation for some time now. In South Africa, a number of initiatives have researched technology as a platform for learning – so called "e-learning". These projects have differed greatly in scope and goals, with some operating as for-profit ventures while others are backed by Non-Governmental Organisations (NGOs).

A common target of these projects has been STEM fields, particularly mathematics given South Africa's problematic international ranking in the subject [18]. This background has led to the creation of Dig-it, a non-profit mathematics platform operated by Praekelt.org and recently funded by Investec. Dig-it provides a mobi-site for high school students where they are provided questions on a weekly basis and rewarded with airtime for reaching certain goals. A common goal for students is answering more than 60% of questions correctly. Currently, Dig-it operates on a small scale with roughly 1500 active students using the platform from grades 10 through to 12.

1.1 Problem Statement
In 2017, Dig-it is scaling to the national level, aiming to service over 100 000 students in 3 provinces. For a number of reasons, Dig-it's current workflow and codebase need to be adjusted to cater for this expansion. The primary concerns regarding the current system are the creation of mathematical content, the management of content and curricula, and optimizing the delivery of this content to students.

1.2 Proposed Solution
The proposed solution was to create a prototype which demonstrates possible features for the project to scale in future. The prototype was built from the ground up, focussing on our areas of interest and research questions. This project is a proof of concept, and a potential reference design.

1.3 Research Questions
While this is not a research project, this paper plans to answer the following research questions:

- To what degree can a Content Management System (CMS) reduce the reliance on external tools for coordinating e-learning content workflows?
- Which visual methods prove to be effective in conveying and manipulating complex syllabus information?

1.4 Project Work Allocation
The overall project can be divided into two sections. The first section focuses on content creation and delivery. This involves incorporating an open-source WYSIWYG equation editor to allow non-technical content creators to generate mathematics questions, as well as reducing the load size of web pages delivered to learners on the platform. The second section focusses on content management. This involves designing and developing a CMS that allows users to easily understand and organize course content. The latter section is the focus of this paper moving forward. The two sections have been integrated to create a single, unified application.

1.5 Aims
The primary aims of the project are to:

- Allow for automatic or semi-automatic content availability, which involves making questions available to students at certain points in the syllabus
- Improve the arrangement and selection of questions

This paper refers to supplementary materials that can be accessed via the UCT Computer Science online publications website: http://pubs.cs.uct.ac.za/. The abbreviation for this project is **DIGIT**. All references made to the online appendices in this paper refer to these materials.

- Reduce dependencies on tools external to the platform
- Create an inclusive system that manages the moderation of questions before their release

2. BACKGROUND AND RELATED WORK

This section explores some related e-learning projects, as well as other relevant areas of the literature.

2.1 Mobile E-Learning

As mentioned previously, Dig-it exists largely due to the education issues facing South Africa today. Dig-it's approach to improving mathematical literacy is informed by the prevalence of mobile phones. The literature indicates that mobile phone proliferation and access to GSM networks is high in South Africa. In 2013, Tomlinson et al. identified that 93% of South Africans are connected to a GSM network [21]. Hence, the Dig-it platform has taken a mobile-first approach to reach as many students as possible, especially those in poorly resourced environments [11]. While the focus of this project is the CMS, the student-facing aspects do need to take the mobile-first nature of the current platform into account.

2.2 Related E-Learning Projects

E-learning is often described as a medium of instruction that is presented on a digital device [10]. This is a broad definition, and so e-learning can be categorised in many ways. Perhaps the most relevant categorization for our purposes, is whether the learning is synchronous or asynchronous. Synchronous e-learning occurs when the pace is set by an instructor, whereas asynchronous e-learning is self-paced [10]. Dig-it would be categorised as synchronous e-learning due to the structured progression of the syllabus. Each week, certain questions are made available to students based on the progression of the national curriculum.

There are a few relevant projects that are worth mentioning in the context of a mathematics e-learning systems:

- Khan Academy[2] is a world-renowned e-learning platform provided to students of all ages for free. Khan Academy provides a large amount of learning material in mathematics, but also features a large set of practice questions for each section covered by their course content. Their model is asynchronous, with emphasis on students pacing themselves and supplementing their learning in brick-and-mortar institutions. Khan Academy is particularly relevant since they support mobile phones, which is the focus of Dig-it.
- Wolfram Alpha Problem Generator[3] is an advanced mathematics practice application aimed at students. The platform is also asynchronous. Wolfram Alpha's most impressive feature is their fully worked solutions to randomly generated problems, allowing students to understand exactly where they went wrong. This is an advanced feature, which Dig-it does not currently have.
- Siyavula[4] is a South African company that provides asynchronous e-learning products for primary and high school students. Essentially, this takes the form of a question bank which students can use to solidify their learning [19]. Probably the most impressive aspect of their platform is the implementation of adaptive practice, where question difficulty is modified based on the student's performance, preventing frustration or boredom [19]. Questions also make use of randomisation to vary the content presented to students. While Siyavula's model is asynchronous, students tend to focus on content that relates to what they are currently learning. Thus, Siyavula is probably the most similar platform to Dig-it at the moment.

2.3 Workflow Automation

One of the aims of the project is to reduce the dependencies on external tools. This falls under the field known as "workflow automation" – increasing operational efficiency by supporting internal workflows, integrating disparate systems and migrating manual tasks to electronic platforms [20]. In general, "operational efficiency" includes metrics such as cost of operations, service quality and service speed [4]. For the current Dig-it system, this is relevant, as a few external tools are used in conjunction with the main system. Some are not free, and often, many of the features are not used. This is discussed in more detail in section 3.

An important part of workflow automation is the identification of user roles and tasks. Roles are important, since in many systems, a single user may hold many roles [14]. The importance of users and roles indicate that task analysis and use cases will be important in understanding the requirements of the new workflow.

3. CURRENT PROCESSES AND TOOLS

Currently, Dig-it content is managed through a customised administration interface. The process of managing content also involves workflows that are external to the Dig-it platform. Figure 15 in the appendix describes the current system, which involves Asana for tracking content creation, Google Sheets for managing syllabi, and email for coordinating the moderation of questions.

What Figure 15 should indicate, is that the workflow is currently quite complicated, with multiple tools being used for different purposes. In this section, current processes and tools will be analysed with the aim being to identify functionality that needs to be provided more succinctly in the new design.

3.1 Delivering Questions to Students

Once a student has been registered on the Dig-it platform and placed in a class, they will be able to answer questions. Normally, students are limited to 3 questions per day, on week days. The platform ensures that students receive questions randomly from the "question pool" to prevent cheating. If students consistently work through all the available questions, they will receive questions related to where they should be in the national syllabus. However, if they fall behind, the pool of questions grows and students may not receive questions relevant to their class work. Currently, questions need to be made available to students manually through the CMS. This is time consuming, and error-prone.

3.2 Asana

Asana[5] is a management tool designed to help teams communicate and schedule tasks, with specific emphasis on project

[2] https://www.khanacademy.org/

[3] https://www.wolframalpha.com/problem-generator/

[4] http://www.siyavula.com/

[5] https://asana.com/

communication. It is an expansive tool, with many advanced features and settings. From discussions with the Dig-it project manager and content uploader, it is apparent that Asana provides Dig-it with functionality for tracking dates and tasks. This almost always relates to deadlines around when course content should be finalised. While Praekelt.org is using the primary functions of Asana – task management – it appears as though only a small portion of the overall functionality is being utilised. For instance, Asana includes messaging between team members, yet Praekelt.org utilises the Slack application for this. Specifically, Dig-it stakeholders require Asana in order to:

- Keep track of deadlines for the creation, moderation, upload, and publishing of content
- Understand whether content management is meeting the scheduled deadlines
- Determine when content for a particular topic has not been completed

3.3 Google Sheets

As a part of Google's online offering to businesses, document management and online editing are provided through Google Drive[6]. This essentially provides functionality similar to Microsoft's Office suite, except in a completely web-based form. The Dig-it team makes extensive use of this tool currently, with thousands of questions stored in Microsoft Word documents in the cloud. This will be replaced by the content creation aspect of this project, as discussed by Nathan Begbie. More relevantly, the Praekelt.org team makes use of spreadsheets to track information on course content. Specifically, the Praekelt.org team uses Google Sheets to:

- View a breakdown of each topic in a grade
- Track when questions need to be made available to students
- Manage the moderation process and the status of created content

3.4 Email Communications

Internally, Prakelt uses a few tools for communication. As mentioned previously, Asana is used for some aspects of the Dig-it course management. However, email and a chat application called Slack are used as the primary form of internal communications. From discussions with the current content manager for the Dig-it project, the use of email is necessitated by the moderation process. As indicated by Figure 15 in the appendix, there is a "back and forth" between the content moderator and content creator. This usually involves corrections and other changes to the content which are indicated in an email. This is certainly not ideal, as these communications might involve multiple emails with attachments and references to external documents. Specifically, the Praekelt.org team needs email communications in order to:

- Indicate corrections of created content
- Relay queries to a specific content creator or moderator
- Identify that content is ready for consumption by students

[6] https://www.google.com/drive/

4. DESIGN

Given the software engineering nature of the project, design was an important aspect of the project's planning and incremental improvement. Early on is the planning stages, it was decided that user involvement was key to the success of the project, and so a form of User-Centred Design was adopted. As mentioned previously, this required regular meetings with Praekelt.org stakeholders involving demonstrations, sketches, clarifications and discussions.

Overall, the content management aspect of the project had three primary design focusses. The first was the technical design of the application, involving system architecture, design patterns and programming decisions. The second focus was on realising the user stories defined by stakeholders, and thus the information flows that are involved. The final focus was on the user interface and visual aspects of the system – how the design could best aid the user in constructing syllabi and content.

4.1 Design Methodology

4.1.1 User-Centred Design

User-Centred Design (UCD) is a broad term that describes design processes which attempt to involve end-users in how the design progresses [2]. The way that users are involved may differ greatly between the different UCD approaches. More intensive processes may partner users with designers throughout the design lifecycle. Less intensive processes may involve users at specific points in the design process, such as requirements gathering or usability testing [1]. This project has taken the latter approach given the nature of the work. Stakeholders at Praekelt.org have been involved through meetings to gather requirements, give feedback on designs and evaluate outcomes.

4.1.2 Task Analysis

To ensure that the prototype matched stakeholder expectations, the design process has involved task analysis. This is the process of classifying and understanding human performance in work scenarios [3]. To achieve this, we were required to identify the key users of the application, the tasks they needed to perform, and what their environment is like. The output of the task analysis can be viewed in Figure 1 and Figure 2, in the use case diagrams.

4.2 Requirement Analysis

4.2.1 Analysis of the Existing Codebase

As mentioned in section 1.2, our initial proposal for the project indicated that we would be building features and making changes on top of the existing Dig-it codebase. While we had begun investigating the current design and structure of the code, we were not fully aware of the state of the project.

The existing codebase is quite large, with almost 1500 Git commits to date and over 13 contributors. In fact, the project has been under development since April 2014 [16]. The project has quite an odd structure – for historical reasons. Initially, Praekelt.org embarked on a project known as MobileU, which aimed to be a very broad Minimum Viable Product (MVP) framework for building mobile education sites [16]. This framework was then used for a new project called OnePlus, which would eventually be renamed Dig-it. Dig-it then made use of a large portion of the MobileU code, through imports and inheritance [17]. Ultimately, Dig-it was the only project to make use of MobileU, and this has resulted in Praekelt.org maintaining two separate code bases that are highly coupled. This violates an

important principle in software architecture: separation of concerns. In this case, the codebase has low cohesion and high coupling, which could be problematic moving forward [9]. Since the project was started some time ago, the project dependencies have aged significantly. The project relies on a very old version of the Django Web Framework and other libraries, which excludes some important features.

After contemplating these facts, we determined that continuing work on the existing codebase would run the risk of turning the project into a maintenance task. We might have spent much of our time understanding existing code and fixing errors, instead of prototyping new features. Since the project was always framed as a prototype, we decided (along with stakeholders at Praekelt.org) that the best path forward would be to start from scratch and focus on our areas of interest.

4.2.2 Determining Scope

Once the decision was made to start from scratch, it was important to determine the scope of our prototype. Even before this stage, it was understood that not all the current features of Dig-it would be included in the prototype. The features that were specifically highlighted as within the project scope are:

- Syllabus management
- Question management
- Delivering questions to users and recording answers
- Automated "liveness" of questions

The features that were specifically highlighted as out of scope were:

- Leaderboards
- Special events
- Gamification

These decisions were made to allow for us to focus on the important aspects of the project as determined by our research questions and aims.

4.2.3 General Design

The design for the prototype is based on the idea that the structure of a syllabus should be determined by the weeks in a year. By using ISO week numbers (each week is numbered between 1 and 52, occasionally 53 in leap years) instead of specific dates, content can be reused in the next year with few changes to the underlying data.

This means that a syllabus is made up of topics with specific start and end weeks. Topics are then broken into blocks, which are then associated with questions. Blocks are a week long, but could be of arbitrary length, hence their naming. Students will be delivered questions that are at most two weeks behind the current date, introducing the idea of a 2 week "moving window" of content. This helps prevent the predicament of students falling behind in the syllabus, as described in section 3.1. The final concept introduced is that of a question creation task. This is a task set up to request the creation of questions for a given topic. This then encapsulates the content creation and moderation processes. The implementation of these concepts through Django models (see section 4.4.1) and hence database tables, is described visually in the model diagram, available in the online appendices.

4.2.4 Use Cases

The use cases for the Dig-it prototype were developed collaboratively with the Dig-it project manager and content uploader. Some use cases were identified through the scheduled project meetings, while others were identified during a specific use case session. The minutes of these meetings can be found online, alongside the output from the use case session.

The new Dig-it workflow was designed alongside the use cases, and is described in Figure 16 in the appendix. Two use case diagrams have been created to describe the requirements of the system. The first is the simplest, and describes the student's view of Dig-it. This is indicated in Figure 1.

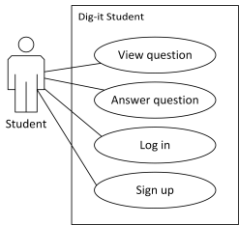

Figure 1: Dig-it Student Use Case Diagram

The second use case diagram relates to the administration aspect of the system, and involves five key actors: syllabus creator, content creator, content moderator, project manager and time. The syllabus creator has a focus on structuring the syllabi through the creation of topics, subjects and blocks. The content creator and moderator need to create and modify content for the system. They share much of the same use cases, since their primary aim is effective content creation. Currently, these roles are filled by contracted teachers external to the project. The project manager needs to have an overview of the system and manage users. Lastly, time can be considered an actor, since the system needs to be able to send reminders about upcoming tasks and missing content warnings to administrators. These use cases are indicated in Figure 2.

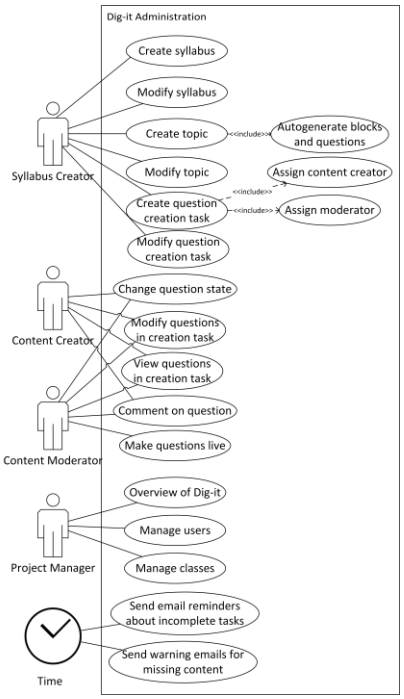

Figure 2: Dig-it Administration Use Case Diagram

4.3 System Architecture and Design

4.3.1 Architecture

The prototype architecture is described in Figure 3. Users interact with the Dig-it application, which interacts with a database. Dig-it has been structured to have two modules (or as Django describes them, "apps"). The "core" module encapsulates all the administrative features and the models for storing the data. The "student" module encapsulates all student-facing features such as answering questions and signing up on the platform.

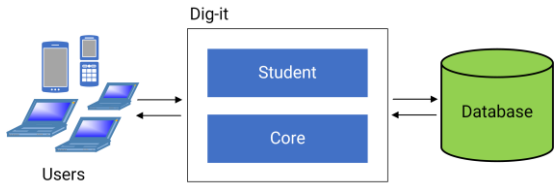

Figure 3: Dig-it Architecture Diagram

4.3.2 New Workflow

To meet the design objective of automating aspects of the original system and unifying processes, a new workflow needed to be designed.

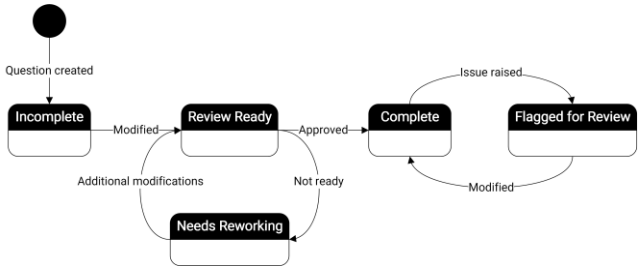

Figure 4: State Machine Diagram for Question States

An important part of this new workflow involves the state of questions. Figure 4 describes how questions can progress from the state of "incomplete" to "complete". A question transitions between these states based on interactions between the content creators, content moderators and platform manager.

4.3.3 Automated Liveness

The current Dig-it application requires manual intervention by administrators to put questions into a live state. This means that at specific dates, an administrator must identify the allocated questions, successfully select them and make them available to students. This task needs to be tracked using external tools, and has led to errors and omissions in the past.

```
• Select questions answered by student < 2 weeks ago
• Select questions from the syllabus scheduled < 2 weeks ago
• Question pool = scheduled - answered
• Randomly deliver a question from the question pool
```

Figure 5: Pseudo Code for Question Delivery

Figure 5 describes the pseudo code for the new automated liveness design for the Dig-it prototype. The new design asserts that we should be able to serve a question to a student based solely on the question's relationship to the curriculum and the student making the request. No manual intervention is necessary – once a question has been completed, the system will automatically serve it to students at the correct time.

4.4 Software Design Patterns

To speed up development and ensure that the application runs reliably, a few key software design patterns were used. Some of these patterns are mandated by the Django Web Framework, whereas others are optional.

4.4.1 Model-View-Controller

At the core of the Django web framework is the familiar Model-View-Controller (MVC) pattern, although with some caveats. Somewhat confusingly, Django refers to views as templates and controllers as views (Model-Template-View) – however, the pattern is almost the same. The main distinction is that the view decides which data is presented, not necessarily how it is presented [12]. The MTV pattern and how it integrates with the web browser and database, is described in Figure 6. While the MTV pattern might have its caveats, from a programming and design perspective, the difference is not a large matter. This paper will make use of Django's MTV definition from this point onward.

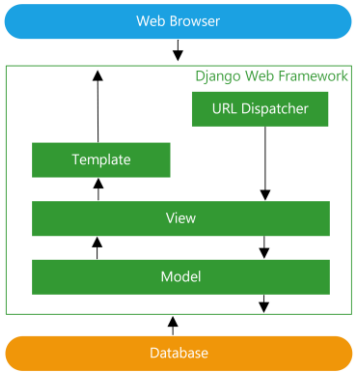

Figure 6: The Django Model-Template-View Pattern [12]

Django models form an object-orientated abstraction of the database tables, allowing for easier manipulation at the application level, and change over time. The changes that are made to the models need to be captured in migrations, which handle the database transformations. Django allows for automatic generation of migrations in most cases. Django views manage the selection of data, making use of the models and Object Relational Mapping (ORM) to achieve this. Finally, the templates present the data, describing the format and technologies used. This involves standard HTML web pages interspersed with Django template tags. These tags insert data that was obtained in the view into the HTML page.

4.4.2 Object Relational Mapping

In Django, the main reason for using models is to gain access to the Object Relational Mapping (ORM) that is tied to them. Essentially, this allows for developers to query the database using python code, without having to write Standard Query Language (SQL) queries. The ORM achieves this by mapping commands or chains of commands to SQL strings, and then executing the resulting SQL. ORMs are somewhat controversial in computer science, especially on larger projects [15]. However, for a project of this size, the Django ORM would enable faster development iterations, and is thus a necessary design pattern.

4.5 User Interface

In keeping with the simplified agile methodology we adopted, the user interface went through a number of iterations. Initially, these were low-fidelity sketches without implementation. Later in the project, the sketches were implemented and iterated on further. The designs were improved in consultation with stakeholders at Praekelt.org. Full documentation of design output can be found in the online appendices.

4.5.1 Iteration 1 (Sketches)

These sketches were done very early in the project, without much attention to detail. Their main purpose was to highlight important flows in the application, and begin to conceptualise the screen design.

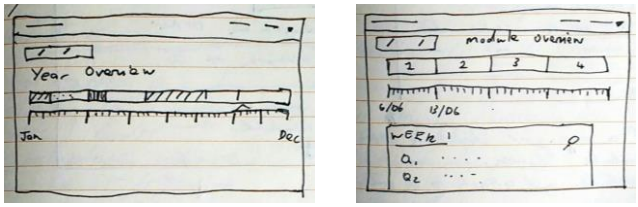

Figure 7: Early design sketches

4.5.2 Iteration 2 (Sketches)

These sketches were possibly the most formative for the eventual design, where many of the design decisions were retained. These sketches were drawn and redrawn on a whiteboard, and they involved more detail than the previous sketches. Most importantly, these sketches were reviewed by stakeholders and led to design changes.

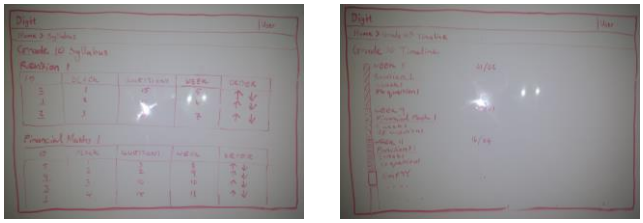

Figure 8: Whiteboard design sketches

4.5.3 Iteration 3 (Implemented)

In this iteration, the designs were implemented in code. There were some changes and additions to the layout, based on feedback from the previous iteration. The colour palette for the project was also decided. Minor modifications would be made before the project was finalised, but these designs were close to the final result.

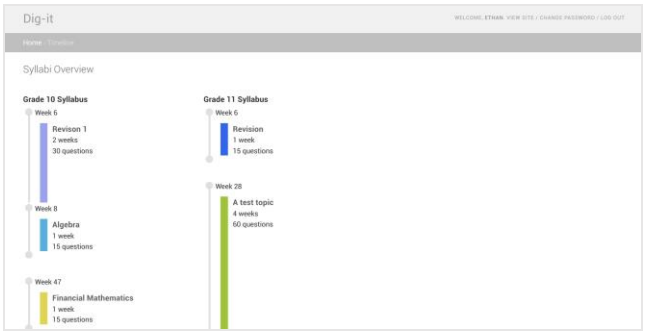

Figure 9: Implemented Design

5. DEVELOPMENT AND IMPLEMENTATION

This section discusses the implementation process for the Dig-it prototype. This includes how the initial design decisions were implemented, and how data structures and algorithms feature in the application.

5.1 Software Development Methodology

In the early stages of planning the project, how our work would relate to the Dig-it platform was unclear. After some discussion with stakeholders, it was determined that given the timeline and research background of our work, the project would best be defined as a prototype. This would potentially serve as a model for another approach to Dig-it, a proof-of-concept for better management of the platform, or even a starting point for future work.

To ensure that the project received critique and guidance, regular meetings were scheduled with Praekelt.org stakeholders in the Dig-it project. This ensured that feedback was incorporated into the design lifecycle of the project. We had initially planned to use the scrum methodology to coordinate development, but after meeting with Praekelt.org, it was apparent that the time constraints of important stakeholders would make this difficult. Regular meetings and sprint reviews would not be possible. Instead, we adopted a simplified agile methodology. This approach ensured that we incorporate client feedback into the development lifecycle, while allowing more time flexibility. Essentially, this approach involved development iterations where the output would be incremental improvements to the prototype.

5.2 Programming Languages and Frameworks

Praekelt.org almost exclusively writes software in the Python language for their projects. Thus, the existing Dig-it system, which was developed some time ago, made use of Python 2. Praekelt.org also makes extensive use of the Django Web Framework[7] to speed up the software development process and provide a consistent technological base for their applications.

Both project members have experience with Python and Django from past projects, and this was a natural choice for the development of the project. The Python/Django stack's ability to enable faster prototyping and development was also deemed to be useful in this project. The latest versions of Python (version 3.5) and Django (version 1.10) were used. Given that the project is web-based, standard web technologies have also been used. This includes HTML, CSS and JavaScript. The stack used can also be said to allow for high portability, given that Django runs on Windows, MacOS and Linux.

5.3 External Libraries

5.3.1 Testing

Testing was implemented using standard Python and Django testing functionality, as well as the PyTest[8] library. PyTest was used to improve upon standard Python testing features. For this project, the primary advantage was test reporting, such as detailed reports of test coverage which are not provided by the standard testing libraries. PyTest was used for the project's unit tests,

[7] https://www.djangoproject.com/

[8] http://doc.pytest.org/en/latest/

which run locally and externally on Travis CI. Coverage reports and evidence of build reports are provided in the online appendices.

5.3.2 Scheduled Tasks

Scheduled reminders and warnings were implemented through the Celery[9] library. Celery is an asynchronous task queue that allows for the scheduling of tasks. Implemented alongside Django Celery Beat[10], these libraries allow for administrators to determine when email reminder and warning tasks should be scheduled. Celery was used since it interfaces well with Django, and successfully abstracts the complexity of task scheduling.

5.3.3 Web Framework

A few libraries were used that extend the functionality of Django. The first library was Django Ordered Model[11], which extends the functionality of standard models by providing relative ordering. This was useful when ordering blocks within topics. The second library was Django Markdown Deux[12], which provides markdown rendering functionality by extending Django's template tags. Markdown is a common and simple form of text markup that gets converted into html, and was used to give syllabus creators flexibility in describing content requirements.

5.3.4 Frontend

A few libraries were used reduce the amount of JavaScript needed, and to provide additional frontend functionality. JQuery[13] was used to make event handling and HTML document traversal simpler. Event listeners were used on several of the administration screens. In addition to this, a small colour script called Random Color[14] was used to generate attractive random colours in visual elements of the admin interface.

5.3.5 Documentation

The documentation for the project was produced using the Sphinx[15] library. Sphinx is a library that produces documentation for python projects. It allows for the use of themes, and has a wide variety of customization options. Sphinx was especially useful for this project since it utilises python comments to automatically generate documentation. This meant that commenting code as the project was developed was also writing the documentation, reducing duplication of effort. This form of documentation should make maintenance of the project simple, as all major functions and classes are clearly described.

5.4 Tools

5.4.1 Version Control System

The Git[16] distributed Version Control System (VCS) was used to manage the codebase. Specifically, we followed the centralised workflow, which involves developers working individually and then synchronising with a central repository [8]. Standard Git practices, such as branching and merging, were used to achieve this. Github[17] was used to host the central repository in the cloud and integrate with our testing tools.

5.4.2 Continuous Integration Tests

To ensure that application builds were tested on every new code merge, we made use of the Travis CI[18] testing platform. This integrated with Github to run the test suite every time code was pushed to the central repository, and report failing tests.

5.4.3 Integrated Development Environment

The Integrated Development Environment (IDE) used for this project was Jet Brains' PyCharm[19]. This provided advanced checks and code completion, which helped speed up development.

5.4.4 Project Management

To manage tasks and deadlines, the lightweight online tool Trello[20] was used. Trello tracks the progression of tasks on user created boards, which is intuitive, and is not time consuming. Hence, the tool suited our workflow on the project.

5.5 Databases and Data Formats

5.5.1 SQLite

Django provides an approach to databases that attempts to abstract most technology-specific code. In other words, smaller projects would most likely be able to swap out database technologies by simply changing a few lines of code relating to the database connector. For this reason, the underlying database technology was not a focus of this project, and SQLite[21] was used.

SQLite databases have the benefit of being stored as a single file, allowing for portability and fast setup times. It also meant that for this project, we could easily provide a working application with sample data without having to run heavy database background processes.

5.5.2 JSON

For some of the visual aspects of the project, it was required that client-side JavaScript be used to build the interface. This meant that data needed to be transmitted from the Django view to the template, where JavaScript would then use the data. To do this easily, the JSON[22] format was used. JSON is simple to read and generate, and works natively in JavaScript.

5.6 Algorithms and Data Structures

The prototype utilised a few key data structures heavily throughout the codebase. This usually took place in the Python code, where data was held in memory after being retrieved from the database. Django provides a standard data structure known as a queryset which allows for a query to be constructed without actually fetching the data – the database is only hit when the queryset is evaluated [13]. Additionally, standard hash maps and arrays were used to manage small amounts of data, including

[9] http://www.celeryproject.org/

[10] https://github.com/celery/django-celery-beat

[11] https://github.com/bfirsh/django-ordered-model

[12] https://github.com/trentm/django-markdown-deux

[13] https://jquery.com/

[14] https://github.com/davidmerfield/randomColor

[15] http://www.sphinx-doc.org/en/latest/

[16] https://git-scm.com/

[17] https://github.com/

[18] https://travis-ci.com/

[19] https://www.jetbrains.com/pycharm/

[20] https://trello.com/

[21] https://sqlite.org/

[22] http://www.json.org/

passing data between the Django views and the JavaScript frontend.

Probably the most important algorithm in the prototype is the delivery of questions to students. This is described in Figure 5. Despite the complexity around identifying which questions have already been answered, and which questions are within the 2 week "rolling window", the algorithm will still operate in linear time. Also, thanks to Django's lazy queryset evaluation, the algorithm will avoid fetching unneeded data.

5.7 Student Screens

Since the Dig-it prototype is primarily focussed on syllabus management, the student functionality is quite simple. Students can sign up, login and answer questions. Figure 10 shows the most important screens students are presented.

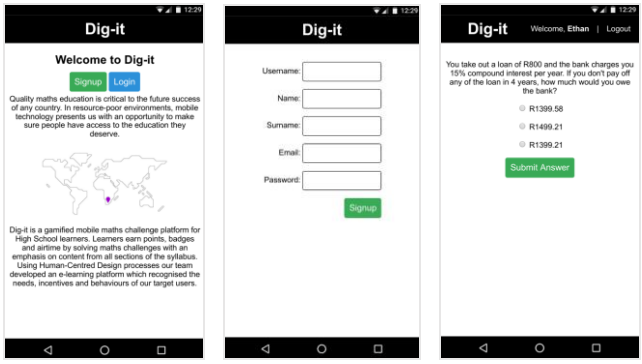

Figure 10: Student Welcome, Signup and Quiz Screens

The delivery of questions to the user is much more complicated, although from the user's perspective, it seems quite simple. The content delivery is discussed in Nathan Begbie's side of the project, although it integrates with the overall system.

5.8 Curriculum Management

Managing the curriculum is done through a customised CMS that provides for standard Create, Read, Update and Destroy (CRUD) operations on stored data, and tasks specific to the platform. Syllabi are created through standard creation forms, and are linked up to grades and classes. Students are then allocated to classes. The syllabus creator can then create topics for the syllabus, which autogenerates the required number of questions. A task can then be made to request the creation of content, assigning moderators and creators to fill in the content for the blank questions. Figure 11 shows the administration home page for Dig-it, below.

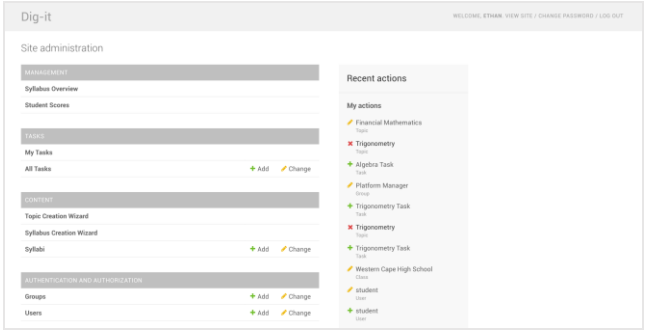

Figure 11: The Dig-it Administration Home Page

5.9 Content Creation

The creation of mathematical content, handled by Nathan Begbie's portion of the project, has been integrated into the overall workflow of the Dig-it administration. In addition to editing questions, comments can be left by any of the administrators, allowing for collaboration between the content creators and moderators. This is indicated in Figure 12, below.

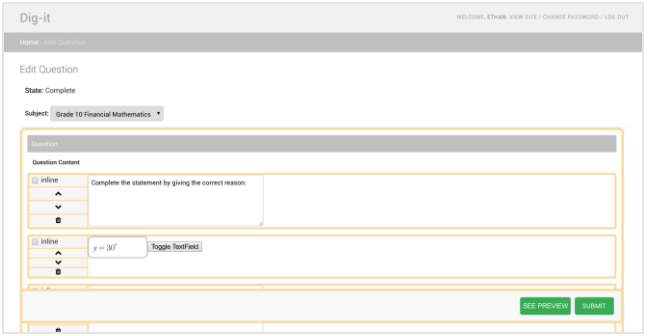

Figure 12: Editing Question Content

5.10 Content Moderation

Content moderation occurs on the question creation task screen displayed in Figure 13. Both content creators and content moderators will use this to navigate the content creation process. This screen displays all the questions in the associated topic, with descriptions for each week of work and the status of each question. From here, users can edit each question's mathematical content, answers etc.

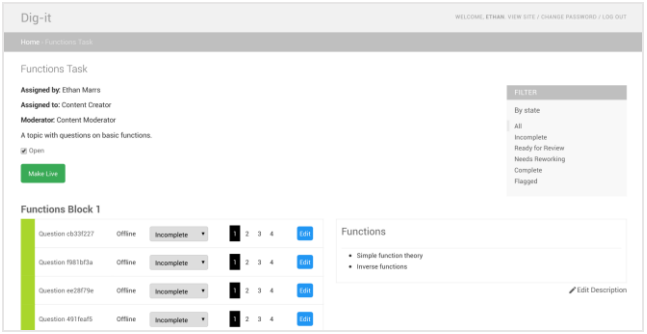

Figure 13: Management Screen for a Creation Task

6. TESTING AND EVALUATION

This section discusses the testing methodology of the project, as well as how the project was to be evaluated.

6.1 Automated Testing

An important aspect of modern software development is automated testing. While standard testing procedures such as following user flows are necessary, they cannot ensure full coverage of the system. Tests that are written in code can be run often, and can effectively raise flags when new code is added to the system.

Django encourages the use of unit tests as a core tenant of its development methodology. In keeping with this, unit tests were written to cover areas where functionality might behave incorrectly, or where subsequent code changes may fail to meet project requirements.

6.2 Usability and User Acceptance

Usability tests were performed informally during meetings with stakeholders throughout the project. A final formal evaluation session was held at the end of the project with various stakeholders, the aim being to gauge the success of the project and to demonstrate functionality.

To structure the evaluation of the system, a common usability scale was used known as the System Usability Scale (SUS). SUS was developed by John Brooke in 1996 to provide a global perspective on subjective usability assessments [6]. The assessment consists of 10 questions, each making use of a 5-point Likert scale ranging from "strongly disagree" to "strongly agree". SUS results in a score between 0 and 100, with the latter being a perfect score. Odd numbered questions in the assessment are phrased positively and increase the overall score, whereas even numbered questions are phrased negatively and reduce the overall score [6]. The questions for SUS are indicated in Table 1, below.

Table 1: System Usability Scale Questionnaire [6]

#	Question
1	I think that I would like to use this system frequently
2	I found the system unnecessarily complex
3	I thought the system was easy to use
4	I think that I would need the support of a technical person to be able to use this system
5	I found the various functions in this system were well integrated
6	I thought there was too much inconsistency in this system
7	I would imagine that most people would learn to use this system very quickly
8	I found the system very cumbersome to use
9	I felt very confident using the system
10	I needed to learn a lot of things before I could get going with this system

6.3 Evaluation Metrics

The metrics used to evaluate the success of the project are listed in Table 2, below. These metrics were stated in the initial project proposal before work on the project began.

Table 2: Metrics for Project Evaluation

#	Metric
1	Remove the need for a content uploader.
2	Reduce the reliance on external tools.
3	Ensure as much content management tracking information as possible is kept on the Dig-it platform.
4	Allow for automatic or semi-automatic release of questions on a schedule.
5	Stakeholder satisfaction with changes to the workflow of the platform.
6	Achieving test coverage of 80% for all code written.

7. RESULTS AND DISCUSSION

This section discusses the result of building the Dig-it prototype. This includes evaluating the project output in terms of the stated aims and metrics, as well as discussing the research questions.

7.1 General Feedback

Throughout the project, we received feedback on our progress. Mainly, this related to the flows within the application, and the tasks that needed to be completed. On more than one occasion, our meetings indicated that some of the functionality was confusing to use. This was one of our key findings, that although curriculum information appears easy to organise from a high-level perspective, the interface quickly becomes confusing for end users. We also received feedback that the more visual aspects of the system (such as timelines and year overviews) were well received, since the current system mostly uses basic CRUD forms.

7.2 Metrics for Project Success

The metrics by which to evaluate the project were determined in the project proposal, and are indicated in Table 2. Most of the metrics allow for a simple true or false evaluation, while some metrics required further analysis. Metric 1 described removing the content uploader role in the previous system. By design, this was achieved by ensuring that content is only created on the Dig-it platform. Figure 16 shows how the role was removed from the workflow. However, it should be noted that while the role was removed, some of the work of the content uploader is now shared by the content creators, moderators and syllabus creators. This includes the final go ahead for a question to be considered complete, and ensuring scheduling is correct.

Metrics 2 and 3 are, unfortunately, not very clearly defined, having been decided early in the project. To enable analysis of these requirements, section 3 identified the primary need for each tool being used. The functionality required in Asana has been met or replaced in some way by design. Content creation tasks assigned to content creators have a due date, which allows administrators to track the deadline of content creation and moderation. Automatic reminder emails are sent if tasks are still incomplete close to their due date, or if content is missing on the platform. Uploading content is no longer a required task since questions are now created on the platform itself. Similarly, publishing of content is no longer an issue – once content is moderated it can be allocated to the question pool and the "rolling window" will begin delivering questions to students when the time is right.

The need for Google Sheets has also been removed, with question tracking, moderation and high level views being provided in the CMS. Views such as the timeline of the syllabi should help prevent the need for long overview spreadsheets. The need for email in the moderation workflow has been reduced, with commenting on questions within the platform providing an arguably better tool for communication. Moderators and content creators can now discuss issues, with the content in question clearly visible. However, email communications would likely span more than just moderation communications, and so the system is not likely to replace this completely – only reduce the current reliance on email. Most importantly, the system has reduced the need for "back and forth" emails with large attachments.

Metric 4 described the release of questions to students without manual intervention. This requirement has been met. Once a syllabus is set up, and questions are created and allocated to the question pool, where they will automatically be available for students to answer when they are within two weeks from the current day. The pseudo code for this process is described in Figure 5. The evaluation of metric 5 is described in the usability evaluation in section 7.3, since it required a more involved

process making use of SUS. Metric 6 mandated that we have automated test coverage for at least 80% of the code written. This requirement was met, with test coverage well above 80% for both the core and student Dig-it modules. Test coverage documentation can be found in the online appendices.

7.3 Usability Evaluation

A final demonstration of the project and usability session was conducted with stakeholders at Praekelt.org. As mentioned in section 6.2, we made use of SUS to structure this process. Three parties participated in the SUS evaluation after working through some scenarios on the platform. The results of the evaluation are indicated in Figure 14, below. The evaluation resulted in a SUS score of 81.6. According to the literature, the average SUS result is 68, and a score between roughly 73 and 85 can be considered to be "good" [7]. While the SUS score is only one metric, it does appear to suggest that the application was largely usable for its intended purpose. The full results of the evaluation, along with the scenarios, can be found in the online appendices.

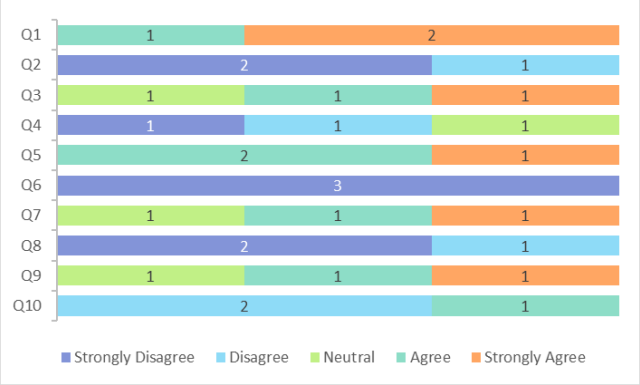

Figure 14: Results of the SUS Evaluation

The usability session also yielded some general feedback. Stakeholders mentioned that some aspects of the application were confusing at times. Especially for those with less knowledge of the current system, some of the tasks required explanation. However, it was noted that while the system was at times complex, it was "necessarily complex" for its purpose. Stakeholders also discussed how some of the features prototyped in the system might be incorporated into the current version of Dig-it, including the visual aspects for understanding curricula, and some of the new structural designs.

8. ETHICAL, PROFESSIONAL AND LEGAL ISSUES

Since this project resulted in coding a prototype from scratch, there were no concerns about the usage of the existing Dig-it codebase. Additionally, Dig-it is already open source and available online. The main concern regarding professional ethics, was the usage of Dig-it data. A number of real questions and answers were converted into the new storage format and used in the project for testing purposes. These questions and answers did not contain any user data, but are the property of Praekelt.org. For this reason, we were careful to ensure that no data was committed to the Git repository, or shared in any other way.

9. CONCLUSIONS

A prototype system for managing mathematical content was developed in an attempt to address some of the issues currently facing the Dig-it platform. A special emphasis was placed on reducing the reliance on tools external to the platform, and improving the arrangement of curricula for students. With regards to our research questions, it appears as though a custom CMS can certainly reduce the reliance on external tools, especially when their current usage is limited. For Dig-it, this included spreadsheets and task tracking software. The reliance on email as a communication tool was also reduced, but it was noted that email is used elsewhere in the organisation, and so it would be likely users will continue to fall back on its usage, regardless. Visual elements in the prototype were well received, especially when they provided an overview of Dig-it content that improved upon the usage of spreadsheets. Hence, timelines and overview screens proved to be the most effective in conveying complex curricula information to administrators.

The new approach to the design of the system, using the weeks of the year instead of fixed dates, allowed for the structuring of syllabi such that questions could be made available to students as soon as they were completed. This new structure did result in some confusion with the CMS interface however, despite iteration on the initial designs. This indicates that there is room for improvement in this area. Somewhat unexpectedly, the interface ended up being quite challenging both from design and technical perspectives.

Ultimately, we believe that we have met our overall aim for the project, to help provide new ideas and approaches for the Dig-it platform to scale effectively in future.

10. FUTURE WORK

This paper has focussed mainly on the content management implications for Dig-it in future. Through the implementation of a prototype, it seems clear that there are several possibilities for streamlining the platform and improving operational efficiency. One aspect of this is automating the creation of mathematical content. For a small platform, it would be possible to continue using a variety of tools to manage content creation, but it appears as though this process would become increasingly difficult as the platform scales.

The second aspect is how this content is managed sustainably. This prototype has demonstrated that incorporating functionality for content moderation is useful to the administrators of the platform, and that arrangement of questions on a single system can improve the usability of the overall platform. This may be an area of future work for the project. It can also be said that features that were specifically determined as out of scope, might be an area of future work for the prototype. This could include integrating leaderboards and gamification into the new application design and workflow.

The project also devised a method to allow for questions to be delivered to students without manual intervention. While this appeared to work in our testing, more research needs to be done to ascertain whether this is a suitable approach for a production system.

11. ACKNOWLEDGMENTS

Firstly, I'd like to thank my project partner, Nathan Begbie, for all his help and hard work on the project. I would also like to extend a special thanks to Lauren Kotze for acting as our primary liaison at Praekelt.org, and Milton Madanda for providing valuable advice. Lastly, I'd like to thank Melissa Densmore and Michelle Kuttel, our supervisor and second reader respectively, for their guidance and support on the project.

12. REFERENCES

[1] Abras, C., Maloney-Krichmar, D., Preece, J. 2004. User-Centered Design. Encyclopedia of Human-Computer Interaction. (2004), 1-2.

[2] Anderson, N. et al. 1988. User Centered System Design: New Perspectives on Human-Computer Interaction. *The American Journal of Psychology*. 101, 1 (1988), 148.

[3] Annett, J. and Stanton, N. 2000. *Task Analysis*. Taylor & Francis.

[4] Aversano, L. et al. 2002. Business process reengineering and workflow automation: a technology transfer experience. *Journal of Systems and Software*. 63, 1 (2002), 29-44.

[5] Begbie, N. and Marrs, E. 2016. Project Proposal: Optimising Mathematical Content Creation and Management Systems for Dig-it. 2016.

[6] Brooke, J. 1996. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*. 189, 194 (1996), 4-7.

[7] Brooke, J. 2013. SUS: A Retrospective. *Journal of Usability Studies.* 8, 2 (2013), 29-40.

[8] Chacon, S. 2009. *Pro Git*. Apress.

[9] Chapter 2: Key Principles of Software Architecture: 2016. *https://msdn.microsoft.com/en-za/library/ee658124.aspx*. Accessed: 2016- 11- 10.

[10] Clark, R. and Mayer, R. 2011. *E-learning and the science of instruction*. Pfeiffer. 8-12.

[11] Dig-it Learn More: 2016. *http://www.praekelt.org/digit/*. Accessed: 2016-11-15.

[12] Django Documentation – FAQ: 2016. *https://docs.djangoproject.com/en/1.10/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names*. Accessed: 2016-10-30.

[13] Django documentation – QuerySet API reference: 2016. *https://docs.djangoproject.com/en/1.10/ref/models/querysets/* . Accessed: 2016-11-15.

[14] Georgakopoulos, D. et al. 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*. 3, 2 (1995), 119-153.

[15] Object-relational Mappers (ORMs) - Full Stack Python: 2016. *https://www.fullstackpython.com/object-relational-mappers-orms.html*. Accessed: 2016-11-25.

[16] Praekelt/Mobileu: 2016. *https://github.com/praekelt/mobileu*. Accessed: 2016-10-30.

[17] Prakekelt/Oneplus: 2016. *https://github.com/praekelt/oneplus.* Accessed: 2016-10-30.

[18] SA almost dead last in maths and science education - WEF report: 2016. *http://www.news24.com/SouthAfrica/News/SA-almost-dead-last-in-maths-and-science-education-WEF-report-20151001*. Accessed: 2016-11-10.

[19] Siyavula Technology: 2016. *http://www.siyavula.com/technology-components.html*. Accessed: 2016- 11- 06.

[20] Stohr, E. and Zhao, J. 2001. Workflow Automation: Overview and Research Issues. Information Systems Frontiers. 3, 3 (2001), 281-296.

[21] Tomlinson, M., Rotheram-Borus, M. J., Swartz, L. and Tsai, A. C. 2013. Scaling Up mHealth: Where Is the Evidence? PLoS Medicine. 10, 2 (2013). DOI=10.1371/journal.pmed.1001382

13. APPENDIX

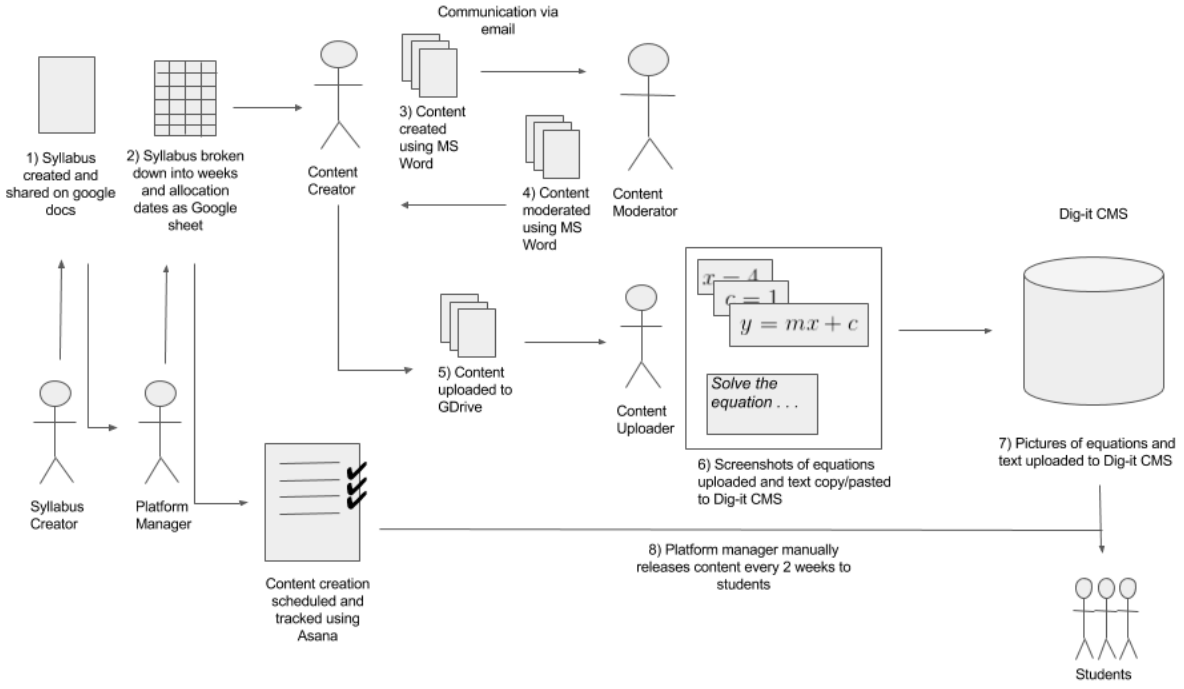

Figure 15: The Current Dig-it Workflow [5]

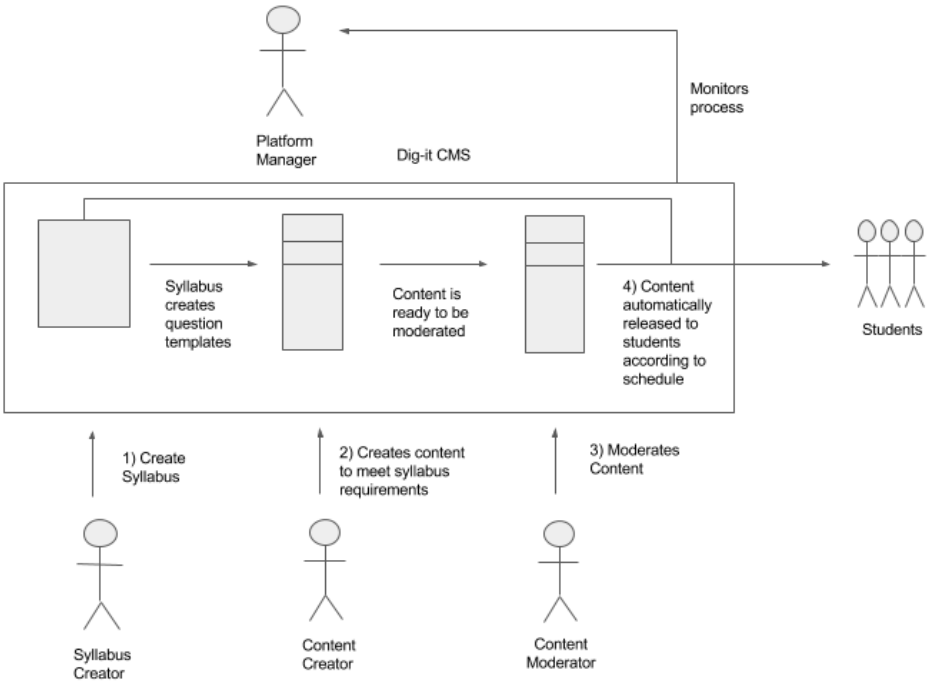

Figure 16: The New Dig-it Workflow [5]