Honours Project Report

iTextbook
Interactive Textbook Authoring, Reading and Assessment Tool for Introductory Computer Programming

By:
Julian Kanjere
jkanjere@cs.uct.ac.za

Supervised By:
Gary Stewart
gstewart@cs.uct.ac.za

October 2012

<table>
<thead>
<tr>
<th>Category</th>
<th>Min</th>
<th>Max</th>
<th>Chosen</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Requirements Analysis and Design</td>
<td>0</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>2 Theoretical Analysis</td>
<td>0</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>3 Experiment Design and Execution</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>4 System Development and Implementation</td>
<td>0</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>5 Results, Findings and Conclusion</td>
<td>10</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>6 Aim Formulation and Background Work</td>
<td>10</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>7 Quality of Report Writing and Presentation</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>8 Adherence to Project Proposal and Quality of Deliverables</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>9 Overall General Project Evaluation</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total Marks</strong></td>
<td><strong>80</strong></td>
<td></td>
<td><strong>80</strong></td>
</tr>
</tbody>
</table>

UNIVERSITY OF CAPE TOWN
Department of Computer Science
Abstract
The proliferation of internet based activity such as e-Commerce and e-Learning in the place of traditional means such as shopping in physical storefronts and classroom based learning can be attributed to various characteristics of the internet. These include its ubiquitous nature and lower costs of storage and delivery of online goods vs. physical goods.

Taking a closer look at e-Learning, it takes on many different forms, from online study groups to use of online educational resources and interactive online textbooks. Of particular interest is the concept of interactive online textbooks, whose potential is yet to be fully exploited.

This report discusses the design and development of a web based version of the iTextbook, an Interactive Online Textbook Authoring, Reading and Assessment Tool for Introductory Computer Programming. Included in this report is a motivation for the project based on current status of e-Learning and overview of present day authoring, reading and assessment tools for introductory computer programming.

It concludes with a discussion of key findings based on evaluation of the iTextBook conducted with a sample of the target users – computer science students and lecturers as well as a list of possible future enhancements. The findings demonstrate that the iTextBook serves as a great tool to teach programming with intuitive authoring interfaces and analytics functionality for lecturers. For students, the findings suggest that it is a useful all-in-one tool for reading and taking programming assessments although the navigation is unclear in some instances and the interface is a bit bland. As such, some of the future enhancements identified include aesthetic improvements to the interface and an ask-a-question feature for private student and lecturer interaction.

Keywords
Authoring, Assessment, Design, E-Learning, Interactive Online Textbooks, LMS, Programming

Acknowledgements
Firstly I would like to extend my sincere gratitude to Mr Gary Stewart, my project supervisor, for all his feedback and support. Thank you for keeping me on the right track during the course of the project.

Next, I would like to thank Riyaadh Kajee, Leen Remmelzwaal as well as all the University of Cape Town (UCT) students who took part in the evaluations of the iTextBook, giving up part of their spare time to give feedback on this project.

I would also like to thank my family, whose sacrifices and endless support has spurred me thus far.

Last but not least, I would like to thank my Lord and Saviour, Jesus Christ for leading me every step of this journey. To him be all the glory.
# Table of Contents

Abstract .................................................................................................................. ii

Keywords ............................................................................................................... ii

Acknowledgements .............................................................................................. ii

List of Figures ...................................................................................................... v

List of Tables ........................................................................................................ vi

Glossary .................................................................................................................. vii

1. Introduction ...................................................................................................... 1
   1.1. Motivation for Research .............................................................................. 1
   1.2. System Overview ....................................................................................... 1
   1.3. Ethical Considerations ............................................................................. 2
   1.4. Report Structure ....................................................................................... 2

2. Background ...................................................................................................... 3
   2.1. Introduction ............................................................................................... 3
   2.2. The Rise of e-Learning and Interactive Online Textbooks (IOTs) ......... 3
       2.2.1. IOT Design Guidelines ..................................................................... 5
       2.2.2. IOT Models ..................................................................................... 6
   2.3. Features of Introductory Computer Programming IOT ....................... 7
       2.3.1. Authoring .......................................................................................... 7
       2.3.2. Accessing Learning Materials .......................................................... 8
       2.3.3. Programming Assessments ............................................................... 9
   2.4. Previous Work .......................................................................................... 10
   2.5. Conclusion ................................................................................................ 13

3. Design .............................................................................................................. 14
   3.1. Analysis ..................................................................................................... 15
       3.1.1. Interview with Computer Science Lecturer ...................................... 15
       3.1.2. Ethnographic Study .......................................................................... 15
       3.1.3. Low Fidelity Prototypes ................................................................... 16
   3.2. Proposed Solution ..................................................................................... 17
       3.2.1. Lecturer Features ............................................................................. 18
       3.2.2. Student Features ............................................................................. 20
       3.2.3. User Interface Design ...................................................................... 22
       3.2.4. High Fidelity Prototypes .................................................................. 22

4. Implementation ................................................................................................ 27
   4.1. Web Development Framework ................................................................. 27
   4.2. Database .................................................................................................... 29
List of Figures

Figure 1 - iTextBook System Overview ........................................................................... 2
Figure 2 - Apple iBooks Bookshelf Metaphor ................................................................. 6
Figure 3 - iBook's Lecturer Authoring User Interface ..................................................... 8
Figure 4 - Mendeley Desktop End User Interface ........................................................... 8
Figure 5 - Overview of User Centred Design ................................................................ 14
Figure 6 - Use Case Diagram for Lecturer ................................................................. 19
Figure 7 - Use Case Diagram for Student ................................................................... 21
Figure 8 - iTextbook Web System Architecture ........................................................... 27
Figure 9 - Django Project Structure ............................................................................. 28
Figure 10 - Entity Relationship Diagram for iTextBook Web ........................................ 30
Figure 11 - Combined Server ......................................................................................... 31
Figure 12 - Dual View Interface ..................................................................................... 32
Figure 13 - Book favicon for iTextBook ....................................................................... 33
Figure 14 - Admin Courses Home ................................................................................ 34
Figure 15 - Admin Add Course ..................................................................................... 34
Figure 16 - Admin Add Topic ......................................................................................... 35
Figure 17 - Admin Add Subtopic .................................................................................. 35
Figure 18 - Subtopic Video on Left Side of Dual View and Notepad on Right Side .......... 36
Figure 19 - Discussion Forum / Comments Feature on Right Side of Dual View ............ 36
Figure 20 - Admin Analytics Options ....................................................................... 37
Figure 21 - Admin Usage Analytics Feature .................................................................. 38
Figure 22 - Admin User Coding Question Use ............................................................... 38
Figure 23 - Admin Subtopic MCQ Analytics ................................................................. 39
Figure 24 - Search Results (in order of decreasing abstraction) ..................................... 40
Figure 25 - User Specific Help ....................................................................................... 40
Figure 26 - Codepad located on Right of Dual View ...................................................... 41
Figure 27 - Admin Add Coding Question ................................................................. 42
Figure 28 - Coding Question ......................................................................................... 42
Figure 29 - Automated Feedback for Coding Question ................................................. 43
Figure 30 - Admin Add Multiple Choice Question ....................................................... 44
Figure 31 - Admin Attempt Multiple Choice Question .................................................. 45
Figure 32 - Multiple Choice Question Feedback ........................................................... 46
Figure 33 - User Login .................................................................................................. 47
Figure 34 - Admin Add User ......................................................................................... 47
Figure 35 - Admin Manage Users ............................................................................... 48
Figure 36 - Scaling Django using dedicated servers ....................................................... 56
List of Tables

Table 1 - Interactive Online Textbooks vs. Hardcover Textbooks .............................................. 4
Table 2 - Summary of Programming Interactive Online Textbooks ................................................ 10
Table 3 - Heuristic Evaluation of Low Fidelity Prototype 1 ........................................................... 16
Table 4 – User Feedback of Low Fidelity Prototype 2 ................................................................. 17
Table 5 - Design Principles used for iTextBook Development ....................................................... 22
Table 6 - High Fidelity Prototype 1 Student Evaluation Comments and Resolves .................... 23
Table 7 - High Fidelity Prototype 1 Lecturer Evaluation Comments and Resolve ....................... 24
Table 8 - High Fidelity Prototype 2 Lecturer Evaluation Comments and Resolve ....................... 25
Table 9 - Specifications for Web Server ....................................................................................... 32
Table 10 - Statistics for Required Students Computer Skills ....................................................... 52
Table 11 - Statistics for Student Functionality Evaluation ............................................................ 53
Table 12 Low Fidelity Prototypes 1 ................................................................................................. 60
Table 13 - Low Fidelity Prototypes 2 ............................................................................................. 63
Table 14 High Fidelity Prototype 1 (Student) ................................................................................. 66
Table 15 - High Fidelity Prototype 1 (Admin / Lecturer View) ...................................................... 69
Glossary

Admin Short for Administrator which refers to an individual with CRUD rights when accessing a computer system

Amazon Multinational electronic commerce company that started off as an online bookstore but has since diversified to sell DVD’s, CD’s, software, video games etc.

Bug Flaw in a computer program that causes it to behave in an unpredictable manner

C A general purpose programming language

Client An application or a system that consumes a service made available by a server on the network

Codepad Textbox on a web page in which one can paste source code for a computer program, execute it and get output within the web browser

Coding Involves designing, writing and debugging the source code for a computer program

Command Line Interface in which a user issues commands to a computer in the form of successive line of text

Course Unit of teaching that typically lasts for one academic term

CQ Acronym for Coding Question that requires the user to enter source code to solve the problem

CRUD Acronym for Create, Read, Update and Delete

CSS Style sheet language used to describe presentation semantics of a web page

Database Organized collection of data

Django Open source Web 2.0 application framework written in Python that encourages rapid application development

Dropbox File hosting service run by Dropbox Incorporated that’s offers remote storage, file synchronization and access via a web interface or client software

Dual View Split screen interface with two different views displayed simultaneously on the left and right hand side of a screen

ERD Acronym for Entity Relationship Model which is an abstract way to describe a database graphically

FLV Acronym for Flash Video which is a proprietary file format used for video files that are delivered over the internet using Adobe Flash Player

HCI Acronym for Human Computer Interaction which is the study of interaction between people and computers

IDE Acronym for Integrated Development Environment which is a software application that provides aggregated facilities to support a computer programmer in software development activities

Java Object Oriented Programming language originally developed at Sun Microsystems now owned by Oracle Corporation
LMS Acronym for Learning Management System which is a web based portal used by lecturers to administer course materials, communicate with students as well as monitor and grade their progress, whilst students use it for learning, collaboration and communication

MCQ Multiple Choice Question for which the user has to select one correct answer from the list of possible answers

MP4 Multimedia format commonly used to store digital video and audio streams

MVC Acronym for Model-View-Controller which software architecture pattern which separates the modelling of a domain, its presentation and actions based on user input into three separate classes which are Model, View and Controller respectively

Object An instantiation of a real world entity

OER Acronym for Open Educational Resources which refers to an Internet-driven worldwide community effort through which educational resources can be accessed free of charge across the globe

ORM Acronym for Object-Relational Mapping which is a programming technique for converting data between incompatible type systems in object oriented programming languages

PDF Acronym for Portable Document Format which is an open standard for electronic document exchange invented by Adobe Systems in 1992

Perl High level, interpreted feature-rich programming language

Plugin Set of software components that extend functionality of a software application

Python Interpreted, interactive object oriented extensible programming language

Scope Work that needs to be accomplished in order to deliver a software product

Server Physical computer dedicated to provide one or more specialised services to users of other computers connected on the same network

Skype Software application used for voice, video and instant messaging communication between 2 or more parties over the internet

Tablet Mobile computer that is larger than a mobile phone and integrated into a flat touch screen device

UCT Acronym for University of Cape Town situated in Cape Town, South Africa. Founded in 1829

Use Case Diagram A diagram used in software engineering to display the interactions between a role / actor and a system to achieve a goal. The actor can be a Human or an external system

Web browser Software application for accessing information resources on the World Wide Web

Web Framework Software framework designed to support development of dynamic websites

WingIDE A Python Integrated Development Environment
1. Introduction

The arrival of 21st century dubbed as the “digital age” (Skills, Burkhardt, Gunn, Dawson, & Coughlin, 2003), has seen rapid advancements in internet based technologies and possibilities which the world has begun to tap into such as e-Commerce, e-Government and e-Learning to mention all but a few. These advancements are in part explained by Moore’s law (Voller & Porté-Agel, 2002) on rapid pace of technology innovation which also postulates the reduced cost of new technologies over time. One should not have to look far to appreciate the penetration and possibilities presented by the internet, common everyday examples are Skype and Amazon (Javalgi, Cutler, & Todd, 2004).

Of particular interest is the notion of e-Learning (Bhuasiri, Xaymoungkhoun, Zo, Rho, & Ciganek, 2012) which is defined as an innovative approach to education delivery via technology enabled delivery of information. Institutions world over are finding different ways of jumping onto this web based education bandwagon. This is attested to by increased uptake and use of Learning Management Systems (LMS) which are web based portals used by lecturers to administer course materials, communicate with students as well as monitor and grade their progress, whilst students on the other hand use an LMS for learning, collaboration and communication (Tsolis et al., 2010).

Various e-Learning channels are currently being used to teach introductory computer programming courses such as online learning environments, quizzes, videos, electronic marking to mention all but a few. Integration of these online delivery channels gives birth to what is described in this report as the iTextBook. The iTextBook is an interactive online textbook authoring, reading and assessment tool for introductory computer programming. It is designed for use by lecturer’s who will use it to create online course material structured in book format and students who will use it to access the published material as well as take related computer programming assessments online. The iTextBook will be available as a web based application as well as a standalone Android tablet application. This report describes the design, development and implementation of the web based version of the iTextBook whilst the Android Tablet Application developed by Riyaadh Kajee is discussed in his report.

1.1. Motivation for Research

From the literature review conducted, the findings suggested that presently there is no web based application that offers a combination of interactive computer programming course authoring, access to material for reading and practical assessments with real time and customised feedback for Python 3 and Java. Most of the applications available typically cater for one of the aforementioned functionalities or in the event that it provides all the features, none is specific to Python 3 and Java. One example of a tool that provides all the functionality listed above is Ceilidh (Foubister, Michaelson, & Tomes, 1997) developed originally by the University of Nottingham for Standard ML and not Python 3 or Java. This warranted the need for the iTextBook.

1.2. System Overview

The main components making up the iTextBook are the web based version, the Android Tablet application and a common database. An abstract view of these is shown in Figure 1 below:
The Web Version and the Android Tablet Application share the same database and are otherwise standalone in every other respect and function independently of one another. The web version is referred to hereafter as the iTextBook Web.

1.3. Ethical Considerations
The prototypes developed for the iTextBook Web were tested on University of Cape Town (UCT) 1st year students taking the introductory programming course in Python and their lecturers. The objective of this testing was to evaluate usability and usefulness of the tool. Prior to carrying out the testing, Ethics Clearance was obtained from the Science Faculty Research and Ethics Committee at the UCT. Furthermore, to engage the 1st year students, permission was obtained from the Department of Student Affairs (DSA) at UCT. In accordance with DSA ethics regulations, all participants in the testing of the iTextBook Web were required to sign consent forms (see Appendix C1) and their identities were encoded in order to preserve their privacy.

1.4. Report Structure
The remainder of this report consists of an introduction and peek at the concept of e-Learning and Interactive Online Educational Material. This is followed by the design and implementation process of the iTextBook Web - from paper based prototypes to the fully functional final system and concluded by a discussion of evaluation findings and future enhancements.
2. Background

2.1. Introduction

It is a common sight in most tertiary academic institutions to see students making use of a desktop computer, laptop, tablet or other electronic device to complement their academic endeavours. The common ground shared by the use of these devices is their use to access online educational material such as Interactive Online Textbooks (IOT) as well as other academic material from centralised repositories managed by the institutions. These repositories are typically in the form of LMS, which have been earlier defined.

IOT are digital in nature and can take one of two forms – identical representation of an actual textbook or a logical ordered sequence of academic course materials put together by a lecturer for the purpose of teaching. IOT are increasingly being used across the globe and their uptake is accelerated by the global reach, interactive features (Weber & Specht, 1997) and ubiquitous nature of the web; variety of as well as easy to use authoring tools and affordable if not free online publishing. Furthermore, IOT are often cheaper, accessed more easily and have features such as embedded video, audio and search that can never be offered in traditional printed textbooks (Dittawit & Wuwongse, 2011). This interactive nature of IOT is not only convenient but visually and verbally stimulating (Zadoks, 1996).

With this increasing popularity, one of the considerations is to ensure continued if not improved usability, accessibility and effectiveness of IOT together with other online educational resources. Some guidelines addressing this issue are discussed in section 2.2.1 of this background chapter.

This background chapter discusses the rise of e-Learning as well as the transition from use of printed textbooks to IOT that facilitate knowledge sharing. This is followed by a review of design issues and guidelines of IOT (from a lecturer’s and a student’s perspective that are often overlooked if not ignored) and brief analysis of existing Interactive Textbook Authoring, Reading and Assessment Tools for teaching / learning computer programming. These tools are analysed from a content management perspective and programming assessments features available. Finally, this chapter concludes with a recap of major themes stemming from the discussion and best features thereof.

2.2. The Rise of e-Learning and Interactive Online Textbooks (IOTs)

As earlier mentioned, the global and ubiquitous nature of the internet has served as an enabler for e-Learning. E-Learning can be summed up as anytime anywhere learning and is positively influenced by factors such as availability of open source LMSs such as Sakai (http://www.sakaiproject.org/about-sakai) and Moodle (https://moodle.org/about/) as well as availability of Open Educational Resources (OER). These LMSs allow for course administration and easy content creation by lecturers and content access by students. However, one must also be cognisant of factors that have hindered the rapid uptake of e-Learning in Africa at large and South Africa for a local context. These include high structural and cost factors for setting up internet facilities especially in Africa(Oyelaran-Oyeyinka & Nyaki Adeya, 2004) as well as high costs of mobile devices such as tablets.

There is a common ground between IOT and LMS – both are e-Learning delivery channels involving a common subset of users – lecturers and students. IOT consist of hyper marked text and multimedia learning material made available online, serving a function similar to that of the printed textbook – imparting knowledge to the reader (Dittawit & Wuwongse, 2011). This learning material is created and/or uploaded by the lecturer for use by students.
Below is a comparison of IOT with Hardcover Textbooks, from which some of the advantages stand out (Dittawit & Wuwongse, 2011).

Table 1 - Interactive Online Textbooks vs. Hardcover Textbooks

<table>
<thead>
<tr>
<th>Feature</th>
<th>Interactive Online Textbook (IOT)</th>
<th>Hardcover Textbook</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>Highlight, Copy/Paste</td>
<td>Highlight, Photocopy</td>
</tr>
<tr>
<td>Mobility</td>
<td>Can be accessed online or stored on a portable reader such as the iPad which is lightweight</td>
<td>Physically heavy</td>
</tr>
<tr>
<td>Cost</td>
<td>Relatively cheap</td>
<td>Expensive, sometimes in short supply</td>
</tr>
<tr>
<td>Learning Material</td>
<td>Dynamic text, video, audio, images</td>
<td>Static text and images</td>
</tr>
<tr>
<td>Collaboration</td>
<td>Some books allow for collaboration with student formed groups or with the lecturer</td>
<td>No collaboration</td>
</tr>
<tr>
<td>Adaptability</td>
<td>Can be customised to suite preference of student such as layout and navigation through material</td>
<td>Non-customisable</td>
</tr>
<tr>
<td>Data Gathering</td>
<td>Can be used to collect data on student study patterns which can be useful to the lecturer in determining the need for supplementary information or editing a part thereof. However, this has possible ethical implications</td>
<td>At best buying patterns can be observed but these are not useful to giving insight into use</td>
</tr>
<tr>
<td>Search</td>
<td>Quick and instant search via search box</td>
<td>Manual and time consuming search via index</td>
</tr>
</tbody>
</table>

Sadly, not all IOTs incorporate the features listed in the table above, some fail to take full advantage of their electronic form and often appear mostly identical to their printed book counterparts (Dittawit & Wuwongse, 2011). Possible reasons for this include a lack of appropriate authoring tools available to the lecturer or in the case that the tools are available, a lack of proficiency in the use thereof. This calls for easy to use and intuitive interfaces for authoring tools – one of the goals of the iTextBook Web.

Use of IOT is also encouraged by the premise of web based education - classroom and platform independence. This results in increased convenience for the student and as such, it is common to presume that IOT will automatically phase out hardcover textbooks. However, although more convenient and cheaper, research seems to suggest that most students are still using hardcover/printed books and not IOT (Shepperd, Grace, & Koch, 2008). This could be attributed to the poor design of IOT, connectivity costs to access the IOT and more notably the powerful conservative social and
organisational forces holding back the change (Press, 2000). The next section will address some of the design guidelines that should be considered with respect to design of IOT and other forms of OER.

### 2.2.1. IOT Design Guidelines

The common users of IOT are the lecturer and the student, who both require that any form of IOT be user friendly. Some of the goals for IOT from a lecturer's end include intuitive and relatively straightforward book authoring and publishing. Likewise, for the student, the material must be easily accessible via a user friendly interface with intuitive navigation and minimal distraction. This would in turn contribute towards a learner centric and interest-based interaction that encourages knowledge pulling (Tsolis et al., 2010) by the student. This is in contrast with the traditional teacher-centric and course centric environment that is promoted by hardcover textbooks and some of the present day Learning Management Systems (LMS) resulting in knowledge push (Tsolis et al., 2010). This view was earlier discussed by Gunawardena & Adamchik (2003), as they highlighted that commonly used LMSs such as Moodle, Sakai and Blackboard are excellent tools for transferring information but not knowledge since they have limited collaboration and students cannot take notes or make persistent changes (e.g. highlight) to the online reading material within context of the LMS.

Brusilovsky (1996) drew up some guidelines which can be used to achieve the earlier mentioned goals of IOT. These guidelines are also useful in the transformation of a printed book to an electronic one which is not achieved in a one-to-one manner (Weber & Specht, 1997). Wilson (2002) reiterates that it is imperative to make use of best practice guidelines in the design of electronic textbooks as the availability of teaching material in electronic format for Higher Education is continually increasing.

Listed below are a few of the basic yet often overlooked guidelines:

- Book page numbers must be included; they serve as a navigation aid.
- Table of Contents (TOC) must be preserved. This is because it primarily serves as a navigation tool and suggests the order in which the material should be covered when assuming that all students are of the same ability.
- Curriculum Sequencing (CSq) must be catered for during the design of the IOT. The student must be able to determine the CSq to use which can be the TOC, the recommended order set by the lecturer or his own personal discretion. IOT need not follow the linear structure that is followed especially in novels (Crestani & Melucci, 2003) since connections between chapters in textbooks can be made without following the TOC sequentially.
- Web based learning systems must cater for different types of students with different levels of knowledge (Brusilovsky, 2003). In the case of printed textbooks and sadly in some IOTs, the lecturers have assumed an average user persona, drawn up an optimal learning path and presented their curriculums with one type of user in mind (Weber & Specht, 1997). However, students are hardly identical and have different abilities which can be catered to using the Worldwide Web unlike in the case of the printed textbook. This difference in ability amongst students can frustrate the learning process for a student that is either well versed in the subject matter or is poorly versed with the material.
- Metaphors must be included in design of IOT. A metaphor is a concept that users are familiar with in the real world that can be used to make Human Computer Interaction more intuitive and easier to understand (Hackos and Redish, 1998). Common metaphors that can be considered in the design of IOT include:
o Traffic lights metaphor to show links/sections of the book that can be viewed that can be visited by using the colours Red, Green and Orange. However, with regards to use of Traffic Light Metaphors, one must consider the meaning of colour if the IOT is for a specific culture, country or region

o Textbook Metaphor which may be implemented in different forms such as including a book cover, flipping pages in a similar way to a hardcover textbook, maintaining page numbers and/or TOC among other conventions used for the hardcover textbook.

o Book shelf metaphor to show books that the student has accessed in his library such as is used in the Apple iBook’s application

Figure 2 - Apple iBooks Bookshelf Metaphor

- Visual Book Metaphor to represent printed book characteristics such as dimensions, thickness and book cover to mention all but a few (Crestani & Melucci, 2003)
  - Avoid visual clutter such as unnecessary web links and/or online adverts (Cestani & Melucci, 2003). In this regard, the rule of thumb is that if it does not appear in the book and is of non-educational value then it should not appear on the interactive textbook interface.

Most of these guidelines are considered in the development of the iTextBook Web.

### 2.2.2. IOT Models

After noting the goals and guidelines of an IOT, the implementation is often based on a combination of models that categorises the learning material and in some instances the different system actors who are usually the lecturer and the student. Some of the common models are discussed below.

#### 2.2.2.1. Domain Model (Learning Objects)

This model is composed of the actual learning (course) material (Bontchev & Vassileva, 2009). The learning material is organized into a repository of information objects known as learning objects. According to (Gunawardena & Adamchik, 2003) a learning object (LO) is an integrated module that in the case of programming would contain the core theory, code examples and review questions. Learning objects are described by an XML document based on Sharable Content Object Reference Model (SCORM 1.2) standards and stored in a database. SCORM is an emerging standard for creating, sharing and extending learning objects independent of the platform or the audience (Gunawardena & Adamchik, 2003). The Domain Model also supports for definition of two types of relationships between LO’s, is_prerequisite_of and is_related_to (Moundridou & Virvou, 2001). From an iTextBook Web point of view, the learning objects will
be materials that are used for the unit of learning – subtopics. A subtopic will consist of notes, videos and assessments.

2.2.2.2. Learner Model
The Learner Model consists of the student’s personal details including their user name and password, their preferences as well as a knowledge stereotype which may be any one of novice, beginner, intermediate or expert. This stereotype is initially formed after a preliminary test (student modelling) posed to the student or determined by the lecturer and may change overtime as a result of the students interactions with the system. This model is often used together with the domain model to track which material has been covered as well as the students’ performance for particular topic (Moundridou & Virvou, 2001).
A variation to this learner model is the Episodic Learner Model which diagnoses a students complete and incomplete solutions to problems and gives them personalized assistance(Weber & Specht, 1997). This knowledge of the student is stored as a collection of episodes.

2.2.2.3. Instructor Model
Maintains information obtained explicitly by the instructor, such as the desired difficulty of the course and / or determining how student’s level of knowledge will be calculated (Moundridou & Virvou, 2001).

2.3. Features of Introductory Computer Programming IOT
The aim of the iTextBook project is to develop a tool that can be used to teach and learn introductory computer programming. Therefore, the features of interest with regards to designing such a tool are authoring functionality for lecturers and learning functionality for students which include accessing of the learning materials and taking of programming assessments.

2.3.1. Authoring
Most authoring tools make use of a user friendly, intuitive interface from which the lecturer can upload learning material in a logical sequence and edit the content in a relatively simple manner. One such example of this is the e-book Authoring application by Apple Inc., iBook’s Lecturer as shown below.
2.3.2. Accessing Learning Materials

Standard end user / student functionality found in most reading tools includes easy reading of material, navigation via scroll bars, table of contents and pagination. Extra functionality includes the ability to take personalised notes (annotations) as illustrated in the Mendeley Desktop application.
2.3.3. Programming Assessments

Programming assessments are typically in the form of Multiple Choice Questions (MCQ’s) and Programming Exercises referred to hereafter as Coding Questions (CQ’s).

MCQ’s present the user with a question and choice of mutually exclusive solutions for which there is typically one correct answer. MCQ’s usually provide instant feedback upon submission and are a common sight in online assessment platforms.

CQ’s on the other hand, present the user with a problem to solve that requires writing some source code. The evaluation of the CQ can then be done by use of dynamic or static techniques which check the correctness and optimality thereof (Foubister et al., 1997). These are explained below.

2.3.3.1. Static Code Analysis

Static code analysis involves checking against programming style conventions such as Python PEP 8 Style guide (http://www.python.org/dev/peps/pep-0008/). Common static code analysis metrics include programming style such as use of whitespace, use of comments, keywords, ordered keywords, number of functions, number of single variables (Blumenstein, Green, Nguyen, & Muthukkumarasamy, 2004). Static code analysis for python can be carried out by industrial quality tools such as PyChecker (http://pychecker.sourceforge.net/), PyLint (http://www.logilab.org/857/). Checkstyle and PMD (Edwards, 2003) are static code analysis tools for Java that perform automated checks for indentation, brace usage, object instantiations to mention all but a few.

2.3.3.2. Dynamic Code Analysis

Dynamic code analysis involves executing the code and analysing the output and behaviour thereof. The metrics used for this include efficiency determination which is accomplished by measuring of the execution time of a piece of code and code coverage. Code coverage tests functions, statements, decisions, conditions, and parameters within a piece of code (Corps & Derrick, n.d.). Code coverage tools for python include Coverage.py (http://nedbatchelder.com/code/coverage/) and Pester (http://jester.sourceforge.net/). Some of the common code coverage tools for Java include CodeCover (http://codecover.org/) and JMockit Coverage (http://jmockit.googlecode.com/svn/trunk/www/tutorial/CodeCoverage.html).
2.4. Previous Work

This section briefly highlights Authoring /Reading / Assessment Tools that are used to teach programming and one used to teach algebra. The latter though not specific to programming still serves as a useful reference point as it is used to teach a specific domain.

Table 2 - Summary of Programming Interactive Online Textbooks

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description &amp; Platform</th>
<th>Lecturer Features</th>
<th>Student Features</th>
<th>Model</th>
</tr>
</thead>
</table>
| ELM – ART2 art2.ph-freiburg.de/Lisp-Course (Weber & Specht, 1997) | • Example based programming in LISP  
• Web Based Client, WWW-server CL-HTTP | • Login to Web Based Lecturer Interface  
• Content Editing – Images, Text  
• Creation of Interactive Problems such as Multiple Choice Questions (MCQ, these problems are used to check if user possesses prerequisite knowledge and for evaluation tasks)  
• Robot Colour Metaphor for curriculum sequencing | • Login to Web Based Student Interface  
• User profiling at 1st use  
• Interactive Problem Solving – programming exercises and evaluation (dynamic / static), MCQ  
• Navigation using TOC or navigation bar on each page  
• Drawback is limited interaction –e.g. lack of annotation feature | Episodic Learner Model |
| Adopta (Bontchev & Vassileva, 2009) | • Java Programming  
• Web Based Client, Database, XML, HTML | • Login to Web Based Lecturer Interface  
• Content Editing – Images, Text  
• Creation of Interactive Problems | • Login to Web Based Student Interface  
• User profiling at 1st use  
• Interactive Problem Solving  
• Navigation using TOC or navigation bar on each page  
• Drawback is the only assessment catered for is MCQ and lack of programming exercises and evaluation (dynamic / static) | Learner and Domain |
| Adaptive Book | • Java Programming | • Login to Web Based Lecturer | • Users can highlight, annotate, | Learner, |
| (Gunawardena & Adamchik, 2003) | • Desktop Client, WWW server | Interface  
• Content Editing – Images, Text  
• Creation of Interactive Problems  
• Content creation for all students or particular user or a group of users | attach files  
• Interactive Problem Solving  
• Collaboration – notes on chapter from buddy list, classmates, post questions for lecturer  
• Easy Navigation  
• Bookmarking  
• Study sheets  
• Drawback is it only has a desktop client program and not web based, Limited Interactive Problem Solving – lack of programming exercises and evaluation (dynamic / static) | Instructor, Domain |
| eTutor (Remmelzwaal, 2010) | • Python and C Programming | • Login to Web Based Lecturer Interface  
• Content Editing – Images, Text  
• Creation of Interactive Problems  
• Content creation for all students or particular user or a group of users | • Interactive Problem Solving In form of MCQ’s and programming exercises with dynamic and static evaluation | Learner, Instructor, Domain |
| Automark | • Java Programming  
• Unix Server | • Login to Web Based Lecturer Interface  
• Uploading of assessment marking templates only and not course material | • Login to Web Based Student Interface  
• Dynamic and Static Code Assessment  
• Limited to submission of assessments only and no functionality to read course materials | Learner, Instructor, Domain |
| Web Based Centre for Automated Testing (Web Cat), (Virginia Tech) | • Unix server  
• Automated assessment of Java programs | • Login to Web Based Lecturer Interface  
• Uploading of assessment marking | • Limited to submission of assessments only and no functionality to read course materials | Instructor, Domain |
| Try (Reek, 1989)          | Assessment of Pascal programming assessments using UNIX shell | Uploading of assessment marking templates only and not course material | Login to Student Interface  
Dynamic and Static Code Assessment  
Limited to submission of assessments only and no functionality to read course material |
|---------------------------|-------------------------------------------------------------|---------------------------------------------------------------|------------------------------------------------------------------|
| Web Based Authoring Tool for Algebra Related Domains (WEAR), (Virvou & Moundridou, 2000) | WEAR is a Web-based Intelligent Tutoring System Authoring tool used for problem construction and solving in Algebra-related domains  
Web Client,HTML & MySQL database | Login to Web Based Lecturer Interface  
Lecturer prepares html files for topics to be contained in the book, Upload each file to WEAR server, and specify title, difficulty level, position in hierarchy, edit prerequisite relationship and related_to relationship  
Lecturer may also create MCQ for individual topics  
User friendly interface for the teacher  
Data mining of student/student interaction with IOT  
Can be utilised for the creation of courseware in other non-Algebra domains (Moundridou and Virvou, 2001) | Login to Web Based Student Interface  
MCQ to stereotype student during 1st use  
Learner Model stores whether read and knowledge level; topic states e.g. visited and known, visited and not known  
Interactive Problem Solving  
Navigation using TOC or navigation bar on each page  
lack of programming exercises and evaluation (dynamic / static)  
Learner, Domain and Instructor |
2.5. Conclusion

E-Learning is becoming part and parcel of daily pedagogy activities in the world over. The convenience it presents through easy access to academic content, easy administration of the content thereof and communication amongst participants of an academic activity is significant. Although in some cases, internet connectivity infrastructure or costs may slow down its uptake, it is nonetheless imperative that educators and stakeholders begin to look at ways of enhancing the effect of e-Learning and taking full advantage thereof.

One such avenue is through IOT whose usefulness for knowledge transfer is beneficial to both lecturers and students as highlighted earlier in this background chapter. For the lecturers, information creation and publishing is available instantly at the click of a button. For the student, there is increased convenience as compared to using printed textbooks as well as opportunities for learning by collaboration and by interactive assessments.

However, these advantages do not necessarily mean that IOT automatically replaces the printed textbook as there are cognitive and social forces at play that have delayed uptake of IOT. To curb the effect of these social forces, best practises must be followed in the design of IOT for them to be effective and meet their goals which include creating an easy to use and accessible knowledge sharing tool. These best practises are also useful in creating digital IOT which are representations of existing hardcopy textbooks as the conversion is not as one would imagine being a one to one mapping, the digital nature of IOT must be fully utilised.

This background chapter ended with an overview of three IOT’s for programming and one for algebra with the intention of drawing the best features which will serve as a reference point in the design of IOT’s for introductory computer programming from this point on. These best features include easy navigation, interactive problem solving, web accessibility, collaboration and annotation. With regards to interactive problem solving, IOT can be used to carry out dynamic and static CQ evaluation in addition to the standard MCQ evaluations.

The effectiveness, accessibility and use of IOT’s to teach programming and other domains can be improved by incorporating the design guidelines and features discussed in this chapter.
3. Design

During the development of the iTextBook Web, the design methodology employed was User Centred Design (UCD) which ensured that the potential users of the application were involved in the design thereof and that their experience and usability needs documented (Jones et al., 2005). Whilst employing UCD, prototypes of the iTextBook Web were developed and iterated as necessary incorporating the user’s feedback until the final system described in the implementation chapter was produced. Prototyping is known to be a good and quick technique to work through ideas – promoting, rejecting and refining them whilst eliminating ambiguity that would otherwise result in the final product if it is not used (Jones, & Marsden, 2005). Prototyping is used to create mock ups of a system that usually evolve into the final system.

The primary users of the iTextBook Web are Computer Science (CS) students and CS lecturers, who for purposes of testing and revising the design of the iTextBook were UCT 1st year students taking the introductory computer programming course (CSC1010H) together their lecturer(s) and lecturers teaching assistants. The users were involved in design of the iTextBook Web via applied ethnographic study as they were carrying out computer programming exercises and direct observation when asked to carry out tasks on the different prototypes of the system, giving comments in the process. The diagram below illustrates process of User Centred Design.

![Diagram of User Centred Design](image)

Figure 5 - Overview of User Centred Design

The advantages of using UCD are that design is kept in-line with user requirements and customer satisfaction is gained when the final system is implemented (Vredenburg, Smith, & Carey, 2002). This proved true with the development the iTextbook Web.

The design was also based on some general Human Computer Interaction (HCI) principles such as affordance and visibility which are discussed under the client design section. Put in other words, HCI is simply Man-Machine Interaction and determines the effectiveness of any computer system (Karray, Alemzadeh, Saleh, & Arab, 2008). This effectiveness, as argued by Karray et al. (2008), is best
achieved when there is a balance between usability and functionality, hence the decision to engage HCI principles in development of the iTextBook Web in order to cater to the usability aspect.

Another technique used for the evaluation of the prototypes was heuristic evaluation provided by Gary Stewart and Professor James Gain, both CS lecturers in the Computer Science Department at UCT. The advantage of heuristic evaluation is that it is quick and cheap as there is no need to evaluate a reasonable sized sample as the experts’ advice is considered truth. This proved to be useful to the development of the iTextBook Web considering the time frames and resources available to conduct wide scale user testing.

The design of the database and server was done after careful consideration of the typical deployment environment, the data-types to be stored as well as standard speed, performance, reliability requirements.

The rest of this design chapter will look at techniques employed to perform requirements analysis and gathering as well as the subsequent incorporation of those requirements into low fidelity prototypes. These low fidelity prototypes were tested on the users and their feedback is summarised in table form together with actions taken to incorporate the feedback. The chapter concludes with a list and brief description of the collated requirements, the proposed solution and resultant high fidelity prototypes.

3.1. Analysis

According to Dennis, Wixom and Roth (2009), the objective of any systems analysis and design endeavour is to study and understand the requirements for development of a new system or redevelop an existing one to make relevant to a context. The next few sections describe the various system analysis activities that were carried out for iTextBook Web project.

3.1.1. Interview with Computer Science Lecturer

A face to face interview was carried out with an Introductory Computer Programming lecturer from the Computer Science Department at UCT. The requirements gathered from this interview are listed below.

- Authoring framework integrating text, video, quizzes and electronic marking for use by lecturers
- Hands on interactive learning experience for students- web based coding box and assessments
- User authentication framework
- Analytics functionality to track user activity and performance for use by lecturers

3.1.2. Ethnographic Study

An ethnographic study was carried out with the first year students in the introductory python programming class during one of their lab sessions. As discussed by Jones and Marsden (2005) ethnographic methods focus on producing an account of what is going in in real life situations by observing moment-by-moment behaviour of people interacting with others and their work environments over extended periods of time. This was carried out by observing the students activities and behaviours as they went about doing their lab exercises. In some cases, informal discussions were held with some of the students in order to understand some of their behaviours. The objective of this
study was to gather more requirements from the other set of users – the students and these are summarised below:

- Integrated / all-in-one learning environment – with coursework and coding box available on one interface as the students had to switch between IDE application (Wing IDE101), lab exercise questions and lecture notes
- Discussion forum as the students were often engaging each other in face to face conversation when needing advice
- Logging of user activity in order to track active learners and passive learners as some of the students in the lab session were not doing the lab exercises but doing other non-computer science related activities

3.1.3. Low Fidelity Prototypes
Low fidelity prototypes, also known as horizontal prototypes show as much functionality as possible but none of the functions are active (Jones et al., 2005). The low fidelity prototypes for the iTextBook Web were developed using Microsoft Excel based on the requirements gathered from the interview with the lecturer and the ethnographic study. The choice to use Microsoft Excel as the prototyping tool was that it can be easily and quickly used to duplicate interfaces on separate worksheets. This allowed for rapid prototyping which was certainly required due to the 6 week timeframe available to develop the iTextBook Web.

3.1.3.1. Low Fidelity Prototype 1 (Heuristic/Expert Evaluation)
The first low fidelity prototype (Appendix A1) was developed by making reference to existing systems as suggested by Professor Gary Marsden, another CS lecturer in the Computer Science Department at UCT. The existing systems used as reference included eTutor (Remmelzwaal, 2010) and Apple’s iBook (see Figure 2).

Heuristic evaluation for this prototype was provided by Professor James Gain and Gary Stewart. The evaluation identified the following issues to address:

<table>
<thead>
<tr>
<th>Comment / Issue</th>
<th>Resolve / Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluttered interface</td>
<td>Move Table of Contents from right hand pane to main screen area on home page</td>
</tr>
<tr>
<td>Poor navigation</td>
<td>Use Breadcrumbs Add navigation bars to the bottom of the screen</td>
</tr>
<tr>
<td>Lack of Dual View for course material and assessment screen</td>
<td>Redesign layout to allow for dual view of course material and assessment screen</td>
</tr>
<tr>
<td>Too little detail for low fidelity prototype</td>
<td>Add sample text and images to low fidelity prototype</td>
</tr>
</tbody>
</table>

3.1.3.2. Low Fidelity Prototype 2 (Student Evaluation)
The second low fidelity prototype (see Appendix A2) addressed the issues from the evaluation of the first low fidelity prototype. The evaluation was undertaken by 10 random first year students from introductory computer programming class at UCT. Each user was interviewed separately to prevent comments based on group think which is defined by Jones et al., (2005) unanimous agreement amongst test subjects with no dissent. Another reason for separate interviews was to ensure that other
users did not familiarise with the interface by watching others performing evaluations. Once seated, a student was presented with a paper based prototype and a series of tasks to perform on the paper based interface which were successively replaced upon successful execution. The evaluations were carried out in the Honours laboratory, located in the Computer Science Department at UCT. The rationale for this was to try and maintain an environment that is similar to that in the lab sessions that the students have – a typically noisy and alive computer lab environment.

During this evaluation, when carrying out tasks, the students were left to their own devices to ensure that their decisions and actions were not biased by assistance from the researcher. Whilst the user was carrying out tasks, their questions, expressions and body language were recorded. Upon completion of the tasks, the test subjects were required to answer post survey questions (see Appendix B5).

This evaluation identified the following issues to address.

Table 4 – User Feedback of Low Fidelity Prototype 2

<table>
<thead>
<tr>
<th>Comment / Issue</th>
<th>Resolve / Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficult to understand the difference between Chapter and Topic and to navigate between the two</td>
<td>Change naming convention from Chapter with Topics to Topic with Subtopics for easier understanding</td>
</tr>
<tr>
<td>Lack of language change option</td>
<td>iTextBook Web is developed for use in English, issue raised was out of scope</td>
</tr>
<tr>
<td>Save history so that if user logs out he can return to where he was, recently viewed option</td>
<td>Added to future work list</td>
</tr>
<tr>
<td>Explain what the user can do in the, help option</td>
<td>Make pdf downloads available from the web application that will have screenshots of the system and explanations</td>
</tr>
<tr>
<td>Links to more information on the web regarding a specific topic</td>
<td>The worldwide web has a plethora of information but as its quality cannot be guaranteed in most cases, the decision after consulting with the project supervisor was not implement this feature</td>
</tr>
<tr>
<td>Include ability to view slides and related material in full screen like YouTube and back to normal size when user presses escape</td>
<td>This was catered for the videos using the flow player plugin</td>
</tr>
<tr>
<td>Google search</td>
<td>This would detract from the web application which is meant to be self-contained</td>
</tr>
<tr>
<td>Compare code from different languages for same task</td>
<td>Decision was not to implement this as it was out of scope</td>
</tr>
<tr>
<td>Have a download option to download file with code submitted in codepad or in CQ</td>
<td>Implemented in High Fidelity prototypes and Final implementation - saving the code file at execution time</td>
</tr>
<tr>
<td>Please show current topic more clearly</td>
<td>Implemented in High Fidelity prototypes and Final implementation - Current topic displayed in bold in the web app and also in the breadcrumbs</td>
</tr>
<tr>
<td>Add chatroom or discussion box</td>
<td>Comments feature already exists for this purpose</td>
</tr>
</tbody>
</table>

These issues were documented and resolved in the high fidelity prototypes described in section 3.2.4 and shown in Appendix A3.

3.2. Proposed Solution

After performing the requirements analysis tasks that have been aforementioned, the proposed solution arrived at was a web accessible tool that could be used by lecturers to create and administer
courses in a textbook style, create programming assessments and perform analytics whilst the students could access the course material and take the assessments. To implement this solution, priority was placed on developing core functionality for the lecturers and students first and then usability issues such as user interface aesthetics second.

Implementing this core functionality involved considerations such as speed and reliability for the server as well as data types and Android compatibility for the choice of database. These are elaborated on in the Chapter 4 - Implementation. The core functionality is described in the next two subsections as lecturer features and student features, followed by user interface design and the high fidelity prototypes.

### 3.2.1. Lecturer Features

The collated features required for the lecturer are listed below:

- **i. Login / Logout**
  
  Authentication feature which upon successful verification of user credentials grants the lecturer super user rights which gave ability to Create, Read, Update and Delete (CRUD) Courses, Topics, Subtopics, Assessments and Users.

- **ii. Adding Courses/Books**
  
  Creating a course and adding the details for it such as the name, a description, an image and the programming language that it is based on.

- **iii. Creating Topics / Subtopics**
  
  Creating topics followed by subtopics for those respective topics. The creation of the subtopic involves the uploading of pdf slides, video files in MP4 or flv format as well as pdf textbook excerpts relevant to the subtopic. A decision was made to use PDF format for the slides / textbook as they are fairly easy to render in a browser and an open format unlike the proprietary Microsoft PowerPoint format.

- **iv. MCQ / CQ / Comments Usage Analytics**
  
  A feature to graphically represent the frequency of use of the MCQ, CQ and Comments with respect to a course in order to deduce the most active students vs. the least active. This could be useful in a scenario whereby a student can earn bonus marks at the discretion of the lecturer, who can then simply refer to the usage analytics to make decision based on usage frequency.

- **v. Subtopic MCQ / CQ / Comments Analytics**
  
  For a given subtopic, lecturers would want to view performance of students on MCQ’s, CQ’s as well as look at the students comments for that subtopic. Therefore an aggregation of all the users’ assessment attempts and scores as well as users’ comments relative to a subtopic was required.

- **vi. CRUD users**
  
  One of the lecturers raised the point that this system was to be typically deployed in a controlled environment and as such, there was no need for a registration facility to be made available to the students but that they are added by the lecturer instead. Therefore, the lecturer would be able to add users, delete users as well as make admin or remove from admin.
All these lecturer features are graphically represented in the use case diagram below. A use case diagram consists of a set of actions pertaining to a system related by a common actor and a goal (Booch, Rumbaugh and Jacobson, 1999).

Figure 6 - Use Case Diagram for Lecturer
3.2.2. Student Features

The collated features for the student are listed below:

i. **Login / Logout**

   Authentication feature which upon successful verification of credentials grants the student rights to make use of all the features listed below.

ii. **Access course, topics/subtopic slides, video, textbook,**

   This includes the ability to choose a subtopic, view the pdf slides, video and textbook excerpt uploaded by the lecturer.

iii. **Do MCQ**

    Option to take an MCQ assessment based on the subtopic learning material.

iv. **Do CQ**

    Option to take a CQ assessment based on the subtopic learning material.

v. **Codepad**

    A feature allowing a student to practise coding in a web browser text box. This code is not graded. The codepad would be ideal for students to try out pieces of code given as examples in the learning materials.

vi. **Comments**

    A collaboration feature in which logged in students can post comments that are unique to a subtopic. This encourages peer learning.

vii. **Annotation**

    A feature allowing for students to take online notes as they review learning material. Potentially useful for later review of material in one’s own understanding.

All these student features are graphically represented in the use case diagram below.
Figure 7 - Use Case Diagram for Student
3.2.3. **User Interface Design**

From the requirements gathering conducted, the need for an all-in-one interface from which a student could access course material and assessments became apparent. This led to a decision to design a dual view interface consisting of learning material on the left hand and assessment and related features on the right hand view. The left and right hand view would both consist of mutually exclusive menu options in order to limit what can be viewed in both views at any one time. Some of the design principles drawn up by Norman (1990) and observed during the user interface design process are listed in the table below.

Table 5 - Design Principles used for iTextBook Development

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affordance</td>
<td>The appearance of an object gives clues to its use</td>
<td>Choice of icons for the iTextBook Web based on affordance such as a red cross for delete, green plus for add</td>
</tr>
<tr>
<td>Constraints</td>
<td>User action restriction</td>
<td>User authentication for access to features</td>
</tr>
<tr>
<td>Principle of Least Astonishment</td>
<td>Consistent look and feel between different screens for the same user type</td>
<td>Use of one stylesheet and colour scheme used across all pages for the iTextBook Web as well as breadcrumbs to give hint on location</td>
</tr>
<tr>
<td>Principle of Least Effort</td>
<td>Simple and easy to use interface</td>
<td>Avoid clutter on the interface, where possible use icons instead of words</td>
</tr>
<tr>
<td>Fitts Law</td>
<td>Time required to rapidly move to a target area is a function of the distance to the target and the size of the target</td>
<td>Menu designed at the top of the page, which is easily accessible and menu buttons larger and more distinct than normal text</td>
</tr>
<tr>
<td>Hicks Law</td>
<td>Time taken by an individual to make a decision depends on the number of choices available</td>
<td>Simple uncluttered user interface, easy to focus attention on relevant detail</td>
</tr>
<tr>
<td>Visualisation Mantra</td>
<td>Present information in summarised form allowing for Overview, Filter and Detail on Demand</td>
<td>Table of Contents for course material</td>
</tr>
</tbody>
</table>

3.2.4. **High Fidelity Prototypes**

The high fidelity prototypes were developed by making use of the collated lecturer and student features, as well as issues identified during the evaluations of the low fidelity prototypes. Since the high fidelity prototypes were tangible software artefacts, backups were done using Dropbox as incremental changes occurred daily.

3.2.4.1. **High Fidelity Prototype 1**

This high fidelity prototype (Appendix A3) was based on feedback received from the evaluation of the paper based prototypes. Some of the issues / concerns raised by the students were relevant, although some were out of the scope of the application.
<table>
<thead>
<tr>
<th>Comment / Issue</th>
<th>Resolve / Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indentation should be 4 spaces not 8</td>
<td>Attempt to rectify this by tweaking the plugin for tabbing in web browser was unsuccessful due to time constraints</td>
</tr>
<tr>
<td>Full screen for left view / right view (with link to back to normal view and back to courses)/Zoom</td>
<td>Changed design to be responsive according to screen size instead of previously fixed 800x600px screen size, in order to maximise according to screen size. Full screen for either of the screen in the dual view was not implemented due to time constraints</td>
</tr>
<tr>
<td>When choosing course, the image should be click-able</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>Tool-tip text for courses</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>Better background for the application/theme</td>
<td>Simplistic approach was used to develop the iTextBook. As such, the background colour for the application remained as white because of its neutrality. Since colour has got different meanings in different contexts, the decision was to stick to white so that the application is relevant to a wide audience</td>
</tr>
<tr>
<td>Feedback when notes/annotations are saved</td>
<td>Successfully implemented – when notes are saved, a message in italics appears with a last saved message and time</td>
</tr>
<tr>
<td>Commenting on particular slide</td>
<td>This request was not implemented due to time constraints and because it was out of scope</td>
</tr>
<tr>
<td>Codepad button should say run and not submit code</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>Regarding CQ, if the question has no inputs(stdin) then that option should be greyed out/removed</td>
<td>Input text box for coding questions was removed after reworking of coding questions test cases as suggested by a lecturer during an admin functionality evaluation</td>
</tr>
<tr>
<td>Loading screen mask</td>
<td>Not implemented as the transitions from one screen to another in the iTextBook are almost instantaneous</td>
</tr>
<tr>
<td>CSS to show selected link</td>
<td>An attempt to implement this proved futile due to tweak required to Django templating. In the end, this was not implemented due to time constraints</td>
</tr>
<tr>
<td>Introduce more colour</td>
<td>Colourful icons added to login screen</td>
</tr>
<tr>
<td>Breadcrumbs for easier navigation</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>Utilize whitespace/bigger screen</td>
<td>Successfully implemented by using relative sizes in the css instead of using absolute sizes. This is a popular responsive design technique</td>
</tr>
<tr>
<td>Chat room</td>
<td>This was already available in the form of the discussion forum</td>
</tr>
<tr>
<td>Help</td>
<td>Successfully implemented by creating PDF manuals for students and admins which are accessible from within the application</td>
</tr>
</tbody>
</table>
Add music, games tabs | Not implemented as this is out of scope
---|---
Audio reading of notes | Not implemented as this is out of scope
video conferencing | Not implemented as this is out of scope
ask question/seek help option | Added to future work list

The admin/lecturer functionality for the same prototype was evaluated by a CS lecturer and the following comments were made. **Table 7 - High Fidelity Prototype 1 Lecturer Evaluation Comments and Resolve**

<table>
<thead>
<tr>
<th>Comment / Issue</th>
<th>Resolve / Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add course cancel button for forms to go back</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>When adding a topic, remove dropdown for course as this context should be fixed</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>When adding a topic / subtopic number for topic / subtopic should auto increment</td>
<td>Successfully implemented by automating in the background</td>
</tr>
<tr>
<td>Align form fields</td>
<td>Successfully implemented by tweaking Django templates</td>
</tr>
<tr>
<td>Does video work for mp4, flv</td>
<td>Flow player plugin supports many video formats including mp4 and flv</td>
</tr>
<tr>
<td>Textbook should be unique to subtopic</td>
<td>Successfully implemented by correcting the bug whereby textbook excerpt uploaded for the very 1st subtopic became default for all subtopics</td>
</tr>
<tr>
<td>Modify CQ interface by having a run code button as in codepad and then when user is satisfied they can submit the code for assessment</td>
<td>Not implemented due to time constraints – therefore the student only has a submit code button which submits immediately for assessment on 1st attempt</td>
</tr>
<tr>
<td>A CQ must have at least 3 test cases with input and output instead of 1 test case</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>Add button to go back to courses page in right hand section of dual screen view</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>Add CQ page displays error message if test case input not provided – remove required option</td>
<td>Bug fixed by validating form for adding a coding question to make sure that all the test cases are provided</td>
</tr>
<tr>
<td>When user clicks on codepad or coding question, the enter code here should be greyed out if not applicable or else removed</td>
<td>Enter code textbox place holder text removed</td>
</tr>
<tr>
<td>Codepad have a back to code button if you can’t get form with button to work with syntax highlighting, alternatively remove syntax highlighting as this is introductory programming</td>
<td>Back to code button not implemented due to time constraints, Syntax highlighting not removed as the argument is that it allows for easier review of code</td>
</tr>
<tr>
<td>User Analytics should be course specific</td>
<td>Successfully implemented by having Course/User analytics unique to a course</td>
</tr>
<tr>
<td>Questions being answered by subtopic analytics are have they understood the theory (MCQ Graph) and have they understood the practical (CQ graph). This is good</td>
<td>This was a compliment, no action required</td>
</tr>
</tbody>
</table>
3.2.4.2. **High Fidelity Prototype 2**

This prototype was tested by a CS teaching assistant, who for the purposes of this evaluation is considered to be an expert just like a lecturer. The following comments pertaining to the admin functionality and student functionality in some cases were raised.

Table 8 - High Fidelity Prototype 2 Lecturer Evaluation Comments and Resolve

<table>
<thead>
<tr>
<th>Comment / Issue</th>
<th>Resolve / Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin1 user had views identical to admin user but could not perform admin user actions</td>
<td>Successfully implemented by granting user super user rights and not just staff rights which are the different permission levels in Django</td>
</tr>
<tr>
<td>When adding a topic or a subtopic, one of the fields required was number which should be automated and not left for the admin to fill in</td>
<td>Successfully implemented by implementing it in the background and hiding from admin view</td>
</tr>
<tr>
<td>Set CQ / MCQ Where is the option to set CQ / MCQ on the course page</td>
<td>Successfully implemented by adding options to add CQ and add MCQ on the Table of Contents page for each subtopic. This view is available to admins only</td>
</tr>
<tr>
<td>Indentation on save code</td>
<td>Not implemented due to time constraints</td>
</tr>
<tr>
<td>Edit saved code</td>
<td>Not implemented due to time constraints</td>
</tr>
<tr>
<td>Remove company performance graphs sample from Google charts api</td>
<td>Successfully implemented by removing the sample bar chart that had been accidentally left during development</td>
</tr>
<tr>
<td>Course / User Analytics good</td>
<td>Compliment, no action required</td>
</tr>
<tr>
<td>Course / User Analytics makes use of pie charts, however the interpretation for pie charts is that of all available, this was used by this party. Therefore either rename CQ, MCQ and Comments Analytics to something to do with usage or consider using a bar chart instead</td>
<td>Successfully implemented by renaming CQ Analytics, MCQ Analytics and Comments Analytics to CQ Usage Statistics, MCQ Usage Analytics and Comments Analytics respectively</td>
</tr>
<tr>
<td>Rename Course / user analytics</td>
<td>Successfully implemented by renaming course / user analytics to usage analytics</td>
</tr>
<tr>
<td>MCQ Analytics subtopic are clear</td>
<td>Compliment, no action required</td>
</tr>
<tr>
<td>Subtopic mcq analytics could use pie chart to show percentages for a particular answer for each question e.g. Q:What is python 80% programming 20% snake</td>
<td>Not implemented due to time constraints, added to future work</td>
</tr>
<tr>
<td>Sorting based on user/code/score/feedback</td>
<td>Not implemented due to time constraints, added to future work</td>
</tr>
<tr>
<td>Subtopic CQ analytics is good, it makes sense</td>
<td>Compliment, no action required</td>
</tr>
<tr>
<td>HTML tooltip for MCQ, CQ analytics on course page etc.</td>
<td>Successfully implemented</td>
</tr>
<tr>
<td>Change min use to minimum use of above keyword Add CQ because Min use is ambiguous</td>
<td>Successfully implemented by renaming min use to minimum use</td>
</tr>
<tr>
<td>Keyword and min use must be optional Minimum use for keyword( in future uses, get context or set default to 1)</td>
<td>This was not implemented because the scoring for the coding questions involves keywords and cannot therefore be optional. At the very least, a coding question will have 1 keyword</td>
</tr>
<tr>
<td>Change keyword to keyword required in code</td>
<td>Successfully implemented by renaming keyword</td>
</tr>
<tr>
<td>Question</td>
<td>Implementation</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>How does Add CQ form show relationship between fields</td>
<td>No change implemented since different sections on a form demarcated using horizontal rule</td>
</tr>
<tr>
<td>Good use of icons and content organisation is good</td>
<td>Compliment, no action required</td>
</tr>
<tr>
<td>Basically avoid using abbreviations where there is space and use full text else use tooltip and at destination use full text</td>
<td>Guideline and Tooltip successfully implemented</td>
</tr>
</tbody>
</table>
4. Implementation

This chapter describes the final system, technology used to implement it, its components/applications and ends with a recap of the key features that it boasts of. The final system, a result of the evolution of the two high fidelity prototypes discussed in the Design chapter, was based on Django CMS 1.4 and SQLite3 database. The evolution was based on feedback received from the evaluations conducted. Prototype evolution is a common phenomenon in system design (http://newton.uor.edu/courses/sysanades/pdf/anaintro.pdf).

Below is a diagram of the final system architecture, with descriptions of the Web Server and Database to follow in this chapter.

![Diagram of the final system architecture](image)

**Figure 8 - iTextbook Web System Architecture**

4.1. Web Development Framework

Django CMS 1.4 was chosen as the web framework to use for the development of the iTextBook Web. The reasons for this include:

- Renown for use in Rapid Application Development as can be seen from its tagline “the web framework for perfectionists with deadlines” (https://www.djangoproject.com/)
- Django is a python based framework, in the event of challenges with python programming, ready expertise could be tapped into from the project supervisor – Mr Gary Stewart
- An opportunity to learn how to program in Python and Django, having never used them before the iTextBook project
Django CMS is pre-packaged with standard libraries to implement security such as csrf for web forms, authentication etc. Very little need to interact with the database directly since Django makes use of Object Relational Mapping (ORM).

Django CMS employs an MVC architectural pattern – Model (Database) Controller (Logic) Viewer (Templates/Interface) (http://msdn.microsoft.com/en-us/library/ff649643.aspx). This MVC pattern encourages modularity in software development as one separately writes code for the Model, Controller and Viewer. The Model ties in with the database, for which one has to write classes in Django which are then converted to objects in the database. The logic for the application and the code for accessing either the model or the view are written in the controller. Configuration of the viewer involves developing HTML based templates for the user interface which make use of data passed on from the controller.

Also, a project in Django such as the iTextBook Web can be broken up into separate applications, each one focusing on a specific task, also promoting modularity. The iTextBook Web was broken down into four main applications - an MCQ application, a CQ application, Book application and User application. These are described in detail in section 4.4.

The Django CMS project structure is illustrated below:

![Figure 9 - Django Project Structure](image-url)
4.2. Database

The design and implementation of the database was influenced by the consideration of the data and data types that were to be stored which included:

- User login detail in text form
- Lecture slides in PDF format
- Videos in FLV or MP4 format
- CQ’s in text format
- MCQ’s in text format
- Users saved code in file format (e.g. .py for python)

After reviewing the data types, a decision was made to make use of a database and file store combination to save the data. The choice for the database was SQLite3 since it is the natively used database for Android Tablet applications and is supported by Django CMS. This database was to be used for user authentication, storing locations for pdf, video and user code files, user comments, assessment feedback, and assessment attempt details. The filestore which is the local disk on the server was to be used for storing the actual pdf files, video and user code files as well as other files necessary for the running of the iTextBook Web. The naming convention for the files that would contain the code submitted by the user in the codepad or the coding assessment was to be saved as a combination of the student’s username and a random number for purposes of keeping a trail and avoiding duplication.

Also, during the design of the database, a simplistic and minimal approach was taken in order to avoid complex data tables and relationships which could prove to be challenging to maintain. A challenge was encountered with modifying the SQLite3 database structure, which required all the data to be dropped before making any adjustments since the Django Evolution plugin for database evolution does not work optimally with SQLite database. The work around to this was to manually add some database table fields using an SQLite database shell accessible via Linux terminal.

Entity Relationship Diagrams are used to graphically display the entities to be contained in a database and the relationships amongst them (Dawson & Parker, 1987). Below is an ERD diagram showing the main applications that make up the iTextBook Web – Book, Box, MCQ and User.

Each of the database tables in the ERD shown are explained under their respective application in the subsections that follow.
Figure 10 - Entity Relationship Diagram for iTextBook Web
4.3. Web Server
The requirements identified for the iTextBook web server included the ability to run the website development framework - Django, serve media files such as the video files, host the database as well as compile if necessary and execute programs created using the programming languages (such as Python 3.2 and Java) created using the codepad or when answering a coding question. This influenced the decision to use a Linux server with Python 3.2, Python 2.6 and Java installed. Two versions of Python are required as version 2.6 is required for the running of Django CMS (main version of python) and 3.2 for the Python course assessments. Furthermore, the server was configured to enable the Flow Video player plugin to render the videos as well as standard python search library – whoosh – for the indexing and searching of the iTextBook Web course material. This also required the configuration of an hourly cron job to update the search index.

Two possible server setups were envisioned:

- combined web and database server
- separate web and database server

However, since the development of the iTextBook was experimental, this meant it was low scale and as such, load balancing was not considered in the server design. Also, due to resources available – access to 1 desktop machine in the Honours Laboratory in the Computer Science Department at UCT, the combined web server and database server approach was chosen.

Figure 11 – Combined Server

4.3.1. Hardware
Initially during development of the iTextBook Web, two different machines were used in order to ensure compatibility on multiple architectures (x86 and x64) and to allow for location independent development. However problems arose with this setup due to different path names on the two machines which caused conflicts if changes were made using one machine and testing done on another machine.
The latter part of the development and the subsequent implementation of the final system were optimised on a desktop computer located in the UCT Computer Science Honours Laboratory.

The specifications for this computer, configured to be a web server are listed below.

Table 9 - Specifications for Web Server

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESSOR</td>
<td>Pentium® Dual – Core</td>
</tr>
<tr>
<td>SPEED</td>
<td>2.2GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>2GB</td>
</tr>
<tr>
<td>HD</td>
<td>80GB</td>
</tr>
<tr>
<td>OS</td>
<td>Linux Ubuntu 11.10</td>
</tr>
<tr>
<td>ARCHITECTURE</td>
<td>x64</td>
</tr>
<tr>
<td>SCREEN RESOLUTION</td>
<td>1024x768</td>
</tr>
<tr>
<td>WEB BROWSER SOFTWARE</td>
<td>Google Chrome 22</td>
</tr>
</tbody>
</table>

### 4.4. User Interface

The user interface was implemented as a dual view / split screen such that at any point in time, the user has got the subtopic learning material (lecture slides, textbook notes or videos) available on the left side of the screen and the interactive features (notepad/annotations, discussion forum/commenting, cq or mcq) on the right side of the dual view (see Figure 12 below). The reasoning behind the dual view interface is that it enables the students to have hands on approach to learning by trying out the concepts noted on the learning material on the left side by using the features on the right side immediately without need to switch between different windows.

![Figure 12 - Dual View Interface](image-url)
The dual view was implemented by use of HTML iFrames. The favicon for the iTextBook Web used was of a book since the name of the application suggests that it is some form of a textbook, as shown below.

![iTextBook favicon](image)

**Figure 13 - Book favicon for iTextBook**

There was an attempt to use HTML5 for development of the interface but due to problems encountered in configuring the Flow video player plugin to work with in Django templates with HTML5, a decision was made to revert back to HTML 4 for which the plugin worked.

### 4.5. System Components

As earlier mentioned, a Django project is broken up into applications (see Figure 9) to promote modularity; hence in this case, the iTextBook Web was split into 4, namely:

- Book – Actual lecture slides and videos for a course as well as analytics functionality
- Box - Coding Questions
- MCQ – Multiple Choice Questions
- User – User registration, management and authentication

Using this approach, each application could be managed individually although related or dependent on another. These are subsequently listed and described.
4.5.1. Book

The purpose of this application was to mimic the real life textbook – to represent a course/book made up of topics and subtopics. The actual content/learning materials are in the form of pdf notes/documents and videos were associated with a subtopic and are located on the left side of the dual view interface.

**Authoring**

Using the admin features available on the interface (see Figure 14), the lecturer is able to author a course/book from scratch by defining a course (see Figure 15), followed by topics (see Figure 16) then within those topics, subtopics for which the lecturer will upload pdf notes, textbook excerpts relating to that specific subtopic as well as videos in mp4 of flv format (see Figure 17).

![Figure 14 - Admin Courses Home](image)

![Figure 15 - Admin Add Course](image)
A challenge that was encountered was configuring html forms in Django templates to handle file uploads which is different from image uploads. This was solved by making the form to be a multi-type form. In order to render the pdf notes or textbook excerpt, the web browser used by the student must have a pdf plugin enabled. Likewise, in order to render video in flv or mp4 format, the web browser used by the student must have flash player enabled in order to play the video that are rendered on the server side using an open source video player plugin called FlowPlayer which was configured during development of iTTextBook Web to work with Django template as shown in Figure 18.

The iTTextBook Web allows for more than 1 course to be authored, specifically providing support for Python3.2, Java, Perl and C. Challenges were encountered in configuring the execution of Java and C programs as they need to first be compiled and then executed. The workaround was to get the context of the current working directory when compiling and executing.

Implementing the course material at subtopic level is an application of the Domain model earlier discussed in the background chapter.
**Reading, Annotations, Comments**

This book application is used by students to view the learning materials (read) uploaded by the lecturer as well as take personal notes/annotations (see Figure 18) associated with the specific subtopic as well as make comments that appear publicly on the discussion forum, also respective to the subtopic in question.

![Figure 18 - Subtopic Video on Left Side of Dual View and Notepad on Right Side](image1)

Comments entered on the discussion forum are distinguished by using a different colour for the name for each comment, making it easy on the eye. Comments also show the time of posting and the most recent comment is shown at the bottom of the list as shown Figure 19 below.

![Figure 19 - Discussion Forum / Comments Feature on Right Side of Dual View](image2)

Both the personal notes and comments are saved in separate tables within the database. The notepad and the discussion forum are available on the right side of the split screen / dual view interface of the iTextbook Web.
All the other applications Box (for coding questions and codepad) and MCQ (for multiple choice questions) are based on subtopics, meaning that they are dependent on the Book application. This is the core of the application as the assessments and analytics ultimately depend on the material contained within this application.

**Analytics**

During the design phase, it was envisaged that an analytics application would be developed separately in addition to the four core applications earlier listed. However, it was discovered that the analytics was dependent on CQ, MCQ and Book, therefore it was included in the Book application since the Book application is the backbone of the iTextBook Web. On examination of the database tables for the Book application (Table 10 to Table 14), one will observe that there are no tables specifically for the analytics. This is because the data used for analytics can be derived from the already stored data on attempts and usage eliminating the need for redundant database tables. This also speeds up database access and ultimately the performance of the iTextBook Web.

The admin / lecturer analytics functionality occurs at 2 levels, usage and subtopic (CQ, MCQ or Comments) as shown below.

![Admin Analytics Options](image)

Figure 20 - Admin Analytics Options

A bar graph icon was chosen to represent the analytics feature, giving affordance to the different analytics features on the interface. The usage and subtopic levels are discussed below.

**Usage**

Formerly known as course/user analytics in earlier prototypes, this allows the lecturer to view frequency of usage of multiple choice questions feature, coding questions feature and commenting feature by the students. This is represented using 3 pie charts, one for each of the above mentioned features respectively, see Figure 19. The pie charts were implemented using Google charts API. An interesting feature is the drill down functionality whereby if the admin clicks on a slice of a pie, they get to see more detailed information regarding the actual activity performed by the student such as all the comments that they have submitted for that specific course as displayed in Figure 20. This drill down functionality required configuration of the charts using some JavaScript event handlers.
Figure 21 - Admin Usage Analytics Feature

Figure 22 - Admin User Coding Question Use
**Subtopic**

This allows the lecturer to view in tabular format, multiple choice question attempts, coding question attempts or comments made for a particular subtopic by all the students.

The selection is made by clicking on one of the three bar graph icons (CQ, MCQ, Comments) in a subtopic row as shown in Figure 20.

![Subtopic MCQ Analytics](image)

**Figure 23 - Admin Subtopic MCQ Analytics**

The analytics make use of the robot metaphor to display a student’s marks, red if a student’s MCQ score is 0, green if it is 1. This is also implemented for the CQ analysis, red if below 50%, orange if 50% and green if above 50%. This is to speed up visual queries to determine a student’s performance by the lecturer.

**Search**

Another feature incorporated in the Book application was the search feature. The search functionality implemented creates an index of courses, course descriptions, topics and subtopics such that any term that a user searches for is checked in all the aforementioned indexes. The search results are then presented to the user in order of decreasing abstraction, from the course to the subtopic level, from which he can choose as desired.
The search box was placed at the right of the on topmost bar as that is easily accessible and logout next to username as with standard conventions as shown in Figure 24 above. On the technical side, the search was implemented using a Django search plugin called haystack (http://haystacksearch.org/) and whoosh which is a fast, pure-python full text indexing, search and spell checking library (http://pypi.python.org/pypi/Whoosh/). A cron job was set up on the server to run hourly in order to update the index as new material is added by the lecturer.

**Help**

Lastly, a help feature was made available to both types of users, as earlier requested by one of the test subjects during evaluations. When the help button is selected, a pdf manual opens up which gives an overview of the features available depending on the type of user and permissions allowed for his role.

An additional installation document was created for system administrators / lecturers, which is bundled together with the iTextBook Web software. The tables for this Book application are shown in Figure 10, with the book prefix.
4.5.2. Box
This application provides the coding component of the iTextBook Web. It is used for the codepad and coding question features which are described below. Both of these execute user submitted code in the background on the web server and return the result to the web browser. This is accomplished by using the python “subprocess” module. Also, in order to enable indenting in the HTML text-area used for the codepad and coding question, allow-tab – a JavaScript plugin was used. In addition to this, syntax-highlighter JavaScript plugin was also configured in order to highlight user submitted code.

**Codepad**
The codepad is as the name suggests a coding notepad where students can enter their own code as they read the notes and execute it to get immediate feedback from the code that has just executed as if they had executed it from the command-line itself. For codepad, students can enter more than 1 input using on separate lines, the codepad is shown below in Figure 26.

![Codepad](image)

**Figure 26 - Codepad located on Right of Dual View**

**Coding Question**
The Box application also provides the Coding Question functionality which includes administration (see Figure 27), attempting of coding questions at subtopic level (see Figure 28) and automatic marking / feedback thereof and saving to the database as shown below.
The automatic marking and feedback is based on dynamic and static assessment of the student’s code as earlier introduced in the background chapter of this report. The dynamic assessment is accomplished by running the students code using test input from the database as set by the lecturer then comparing the output thereof with test output from the database and then scoring accordingly. Similarly, the static assessment is accomplished checking for comments and keywords (which are also set by the lecturer) in the student’s submitted code, from which the code quality and style can be
determined. Based on the outcome of the dynamic and static assessment of the code, the student is given some specific feedback as shown in Figure 29 below.

A student may attempt a coding question multiple times and for each attempt, the score, feedback and code are saved for later review. This is an application of the episodic learner model described earlier in the background chapter. Furthermore, a student also has the ability to download the code that he will have executed for both the coding question and in the codepad.

The scoring for a CQ is done as follows:

- if the code does not run at all, or results in an Error message, then a score of 0 is given and the rest of the checks are not performed
- If the code does run:
  - For each of the 3 test cases available, 5 marks are added to the total marks available and if the test case is passed, 5 marks are added to the user score
  - 5 marks are added to the total marks available for the comments check and if 1 or more comments are found in the user code, 5 marks are added to the user score
  - For each keyword that is associated with the question, 5 marks are added to the total marks available and for each keyword detected in the user code and used sufficiently 5 marks are added to the user score
  - The final result for the student is a percentage that is calculated as: (user score/total marks available )*100%

The major headings for the feedback on the CQ are displayed in bold in order to catch the users’ attention as shown in Figure 29.
The iTextBook can evaluate coding questions that are based on Python, C, Perl and Java programming languages. However, at the commencement of the development of the iTextBook, it was to be developed primarily for Python programming and so the support for Java, Perl and C are additional extras. However, due to time constraints, the codepad could not be configured to support Java, Perl and C and so the codepad only supports Python 3.2 although the coding questions can be set for Python, Java, Perl or C.

Both the codepad and coding questions are located on the right-hand side of the dual view interface and both show the code that the user has run with the syntax highlighted and lines numbered to make it easier for the student to go through the code.

The tables for this Box application are shown in Figure 10, with the box prefix.

4.5.3. MCQ

The MCQ application implements the Multiple Choice Question functionality at subtopic level. It allows for administration (see Figure 30), taking of multiple choice questions (see Figure 31), scoring thereof and saving to the database. A multiple choice question is given a series of possible answers for which one is selected by the lecturer as correct.

![Figure 30 - Admin Add Multiple Choice Question](image-url)
The student is allowed one attempt of a multiple choice question for which his attempt is saved and graded automatically, the interface is identical to that shown in Figure 31 above except it does not have the edit and delete options which are only for the admin / lecturer. After submitting an attempt, the questions are redisplayed in read-only mode (Figure 32) together with the student’s response, score, time of attempt and the correct answers. The correct answers are displayed in green to catch the student’s attention. Also, the student’s responses are green if correct or red if wrong, which is a use of the robot metaphor described earlier.
As in the case of coding question attempts described in the preceding section, the recording of the students MCQ attempts for later review is an application of episodic learner model, each attempt can be considered to be an episode.

Like the codepad and coding questions, the MCQ feature is available on the right hand side of the dual view interface.

The tables for this MCQ application are shown in Figure 10, with the mcq prefix.

4.5.4. User

The user application is provided as default in the Django CMS Web framework.

User Authentication

It comes with standard user authentication functionality, role based access control and security features. This was customised to the iTextBook Web as shown below in Figure 33. The roles for the iTextBook Web are lecturer and student, which seamlessly integrated into the default Django user roles.
The login page has got icons that give an idea into the features of the iTextBook Web. When the user hovers over each image, a description of that feature is given. The stock images are from colourbox.com and iconspedia.com.

**Adding Users**

However, the default configuration was not sufficient for the iTextBook Web and required the installation of an additional Django plugin called django-registration1.7 in order to handle new user registration by the lecturer / admin as shown in Figure 34 below.

After adding a new user, the new user receives an email with a registration link that is valid for 7 days. This required setting up a mail server, for which a local debugging mail server was run on the server for testing purposes. The lecturer is also able to make a user an admin as well as delete users as shown in Figure 35 on the next page.
Figure 35 - Admin Manage Users

The tables for this User application are shown in Figure 10, auth and registration prefix.
4.5.5. Summary of Key Features

4.5.5.1. Admin / Lecturer / Lecturer Features
- CRUD Courses/Books
- CRUD Topics and subtopics thereof,
- CRUD Subtopics which includes material relevant to the subtopic - pdf slides, video(MP4/flv), textbook(pdf),
- CRUD MCQ
- CRUD CQ
- View usage analytics at course/student level
- View Subtopic mcq/cq/comments analytics,
- CRUD users

4.5.5.2. Student Features
- Access course, topics/subtopic slides, video, textbook,
- Attempt MCQ
- Attempt CQ
- Try out code on Codepad
- Make Comments in Discussion Forum
- Make Notes / Annotation

4.5.5.3. Other Features
- Logging of user activity – every code execution is saved to database to allow for later reference
- Tooltip text
- Automatic Marking of code submitted for coding question and personalised feedback
- Lecturer and Student overview manual
5. Experiments and Testing

5.1. Systems Testing
During the development of the system, various types of testing were carried out to ensure correct functionality. These tests included:

- **Black box / Functional testing** - test cases to test each of the key system features were drawn and deviations were corrected unless otherwise stated in the design and implementation chapter
- **Link testing** – Testing of hyperlinks since iTTextBook Web is a website application that consists of links to navigate from one view to another. The links functioned as expected unless otherwise stated in the design and implementation chapter
- **White-box / Glass-box testing** – code coverage for the code contained in the controller was done manually by setting print statements at various points in order to trace which parts of the code were being executed and if it was producing the expected result. This proved particularly useful during the debugging phases

5.2. Experiments
Experiments involving the users were conducted as earlier mentioned throughout the design and implementation phases. However, of particular importance are the tests carried out on the high fidelity prototypes, since they evolved into the final implementation.

The manifold objectives were to test the following:

- Admin / Lecturer functionality
- Student functionality
- General features such as navigation and intuitiveness

The experiments were carried out by giving the users questionnaires that had tasks to perform on the iTTextBook Web as well as post-test questions and ratings to make. In addition to this, the students and lecturers were encouraged to “think out loud” and were directly observed as they carried out the tasks. Remuneration for the test participants was given in the form of potato crisp snacks.

The initial plan was to deploy the iTTextBook Web over a few weeks for a number of lab sessions in the first year computer labs but due to time constraints and the UCT academic calendar drawing to an end, this was not possible. Therefore the experiments were carried out over a few days in the Honours Laboratory at UCT to mimic as closely as possible the students normal environment in the computer labs. They were carried out over a few days due to the different availabilities, particularly the lecturers and teaching assistants.

Prior to carrying out the experiments, the test subjects were required to sign a consent form and asked to fill out a rating of their experience with use of web and touch devices (see Appendix B1 and B2). After this, they were given a brief description of the system. One of the requirements / tasks they had after successful login was to take a minute to familiarise with the iTTextBook Web interface.

Throughout the evaluation, the test subjects were allowed to ask questions if there was need for any clarity. At the end of the evaluation, the test subjects were required to answer some post test questions and ratings.
The student’s evaluation was carried out with two students at a time and lasted approximately 20-30 minutes for each evaluation. A total of ten students from the introductory programming computer course at UCT participated in the evaluations in one afternoon. As such, ten user accounts were created on the iTextBook Web, namely stud1 to stud10, these accounts would be used by the students (test subjects) to access the application. The student’s tasks consisted of accessing the content – user login, accessing subtopic pdf notes, navigating, video, commenting, annotating, downloading notes as well as carrying out an MCQ assessment and a CQ assessment (see Appendix B3).

The lecturer testing was conducted with one CS lecturer and one CS teaching assistant on separate days as they became available. Both were considered as experts. Their tasks involved using the authoring / administrative functions including add user, delete user, and make admin, view stats, set MCQ, set CQ (see Appendix B5). They were given administrator credentials in order to login.
6. Results and Evaluation
The experiments went well with the lecturers and the student participants being able to complete their tasks. The responses from the participants were then grouped into these three main categories – Authoring/Teaching, Required Skills and Learning and analysed statistically in order to make inferences from them. Using these statistical results as a guide, conclusions are drawn about the ease of use, functionality and practicality of the iTextBook Web.

6.1. Authoring/Teaching
Due to the nature of the discipline at hand – Computer Science, relevant experienced personnel to teach introductory computer programming are much less available as compared to other academic disciplines. Therefore, in order to test the admin functionality of the application, first preference was to test it using CS lecturers at UCT. However, when efforts proved futile to agree on evaluation times with some of the CS lecturers in question, the resort was to test the system on at least one CS lecturer and one CS teaching assistant. The reasoning behind this is that the lecturer and the teaching assistant are considered heuristic experts because they have prior exposure to CMS, LMS and web based tools.

One CS teaching assistant performed the tasks and evaluations as shown in the Admin Evaluation Tasks form in Appendix B5. The feedback from the teaching assistant can be summed as:

- CQ administrative feature lacks intuitiveness and requires labelling of different sections to make it easy especially for first time use
- Good use of space in presentation of course contents
- Course authoring features are straightforward

Another evaluation was carried out with a CS lecturer who teaches first year introductory computer programming, but this was think aloud and interactive in nature. It revealed that the content, assessment and analytics were satisfactory given the time available to develop the features.

Although one could argue that statistically significant conclusions cannot be drawn from these two evaluations as they are few, reasons for the contrary are:

- Due to the few lecturers available in CS field, it is a fair sample size to use
- Prior experience with LMS and proficiency in use of web tools qualifies both the lecturer and the teaching assistant as heuristic experts

Therefore, their feedback is taken as valid, a fair representative of the lecturer population and used to draw some conclusions.

6.2. Required Skills
Before the evaluations involving students were carried out, the students were asked to rate on a scale of 1 to 7 their computer literacy as well as daily use of the internet / web. The summarised results are displayed in the table below:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literacy</td>
<td>5.8</td>
<td>1.032796</td>
<td>10</td>
</tr>
<tr>
<td>Daily Web Use</td>
<td>5.7</td>
<td>1.418136</td>
<td>10</td>
</tr>
</tbody>
</table>
As can be seen the average literacy was 5.8 out of 7 whilst the daily web use was an average of 5.7 out of 7. If the mean for literacy and web use had been lower, this could have forced some customisation so that the iTextBook Web would cater for individuals with lower literacy and web exposure. The mean values for both Literacy and Daily Web Use are more than 80% and although the standard deviations for both of these measures suggest variability between the actual ratings and the mean ratings, one can assume that majority of the students tested are computer literate and use the web on a daily basis.

The results from these 2 ratings imply that the students have been exposed to computers before and that they are also familiar with web/internet since more than 50% use it on a daily basis. This means that their evaluations of the system are relevant unlike if they were novice computer or web users.

6.3. Learning

The rating used for the student features were from a scale of 1(strongly disagree) to 5(strongly agree). As can be seen from the table below, all the student features except for navigation had a mean rating of between 4.3 and 4.7 out of 5. This means that the students agreed that they found the personal annotation, commenting, CQ assessment, MCQ assessments and Dual View useful, relevant, and intuitive and agreed that the iTextBook was relevant to Intro Programming. Students were in between neutral-to-agreeing that the navigation was intuitive. This was improved upon by implementing the breadcrumbs, the back buttons with images as well as the renaming chapter/topic to topic and subtopic.

The personal annotation, discussion forum and navigation had standard deviations greater than 1; this suggests greater variability among the actual ratings from the mean ratings for the respective features.

CQ assessments, MCQ assessments, Dual View and Relevance to Introductory programming have standard deviation values below 1, which means that there is minimal variation amongst the actual ratings and the mean ratings for those features. This supports the cause of the evaluation that the aforementioned features are indeed useful in the context of learning introductory programming and intuitive in nature.

Table 11 - Statistics for Student Functionality Evaluation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal Annotation / Notes</td>
<td>4.4</td>
<td>1.264911</td>
<td>10</td>
</tr>
<tr>
<td>Discussion Forum / Commenting</td>
<td>4.3</td>
<td>1.251666</td>
<td>10</td>
</tr>
<tr>
<td>CQ assessments</td>
<td>4.7</td>
<td>0.674949</td>
<td>10</td>
</tr>
<tr>
<td>MCQ assessments</td>
<td>4.5</td>
<td>0.849837</td>
<td>10</td>
</tr>
<tr>
<td>Dual View</td>
<td>4.6</td>
<td>0.516398</td>
<td>10</td>
</tr>
<tr>
<td>Navigation</td>
<td>3.8</td>
<td>1.032796</td>
<td>10</td>
</tr>
<tr>
<td>Relevance to teaching Intro Programming</td>
<td>4.3</td>
<td>0.674949</td>
<td>10</td>
</tr>
</tbody>
</table>
The results from the ratings for the dual view, the mcq / cq assessments are consistent with some of the comments the students made regarding what they liked best such as “Having notes and code window on the same screen”, “Everything available on 1 page”, “Assessments and feedback”, “comments” and “Gives you feedback on what you did and did not do”.

6.4. General Comments
When asked to describe the application using 3 words, some of the words used by the students include “interactive”, “easy”, “reliable”, “fun”, “interesting”, “tricky”, “unique”, “different”, “extraordinary”. Most of these words used to describe the application are complimentary in nature, suggesting that the application is relevant to its cause. The students were certainly impressed with the realisation of the application having evaluated paper prototypes first and indicated that the final system was better than what they had seen on the paper prototype. Also, when asked whether they would recommend such an application to colleagues if it were deployed, all the students gave an affirmative answer.
7. Conclusion

The motivations for this research were to develop a web based interactive authoring, reading and assessment tool for introductory programming courses to be used by lecturers and students dubbed as the “iTextBook Web”. The lecturers would use the authoring functionality to create learning materials and assessments as well as perform administrative tasks. The students on the other hand would use the reading tool to access the materials in variety of formats as well as take programming assessments online with automated and instant feedback.

In order to develop the iTextBook Web, a user centred design approach was taken throughout the duration of the project. The initial requirements gathering and analysis was carried out by conducting interviews with the introductory computer programming course lecturers and an ethnographic study of the students during their computer programming lab sessions. This was followed by sketching of low fidelity paper based prototypes that were evaluated by two sets of users, the lecturers and the students. Feedback from both sets of users was crucial as the lecturers gave expert advice and the students gave feedback from their past experiences with learning introductory programming.

This valuable feedback served as input to the development of the iTextBook Web, whose final implementation was based on tweaking of two high fidelity prototypes that were a realisation of the low fidelity prototypes and their suggested changes. Evaluation for the two high fidelity prototypes was carried out with a sample of students from an introductory computer programming class at UCT, one CS lecturer and one CS teaching assistant. Most of the change suggestions from these final evaluations were successfully incorporated into the final implementation of the system, whilst the other requests not implemented are documented in the future work section of this report.

After conducting evaluations with the students and teaching staff at UCT, some conclusions were drawn regarding the iTextBook Web:

- Course authoring feature is intuitive and logical as it was given thumbs up by the CS lecturer and the CS teaching assistant who reviewed it
- The iTextBook Web intuitively and interactively offers access to materials in the form of pdf documents, videos and assessments all within a dual view. This is evident as the students expressed overall satisfaction with the arrangement of the materials as well as the intuitive nature of the dual view interface. Also, the CS teaching assistant expressed that the use of whitespace and logical ordering between topics and subtopics was noteworthy and enabled easy access to materials
- Than navigation between course material can be improved as the students were neutral as to the ease of navigation
- The assessment features and instant automated feedback of the iTextBook Web are useful judging from the mean ratings given by the students. This was the killer feature of the application
- There is a need for more flexibility in the CQ admin features to cater for a wider range of questions (such as questions that require more than 1 input)
- The analytics feature is a useful add-on to interactive computer programming assessments for use by lecturers. There is room for improvement by displaying more of the data in graphical form instead of tabular form

Therefore, overall, one can conclude that the iTextBook Web offers a useful and relevant combination of web based course authoring, interactive access to material for reading and practical assessments
with real time and customised feedback for introductory computer programming courses. The computer programming languages supported are Python 3, Perl, C and Java. However, minor adjustments need to be made for 100% support for C and Java as these were added at the very end of the development of the application.

Regarding the use of the iTTextBook Web, it can be used as a supplementary e-Learning tool to an introductory programming course or as a standalone application.

### 7.1. Future Work

This section lists some of the changes or issues raised by the students and admins during evaluation as well as some that were identified during the development of the iTTextBook but could not be implemented in the final system described in the implementation chapter. Reasons for not implementing some of these changes include time constraints, resources and in some cases, the requests were beyond the scope of the project.

- Dedicated web, media and database servers for increased traffic (scalability)

![Diagram of dedicated servers](image)

**Figure 36 – Scaling Django using dedicated servers**

- Add colour to the user interface
- Auto save notes using ajax in case student forgets to save
- Ask-a-question feature for private student and lecturer interaction
- Subtopic mcq analytics could use pie chart to show percentages for a particular answer for each question e.g. Q:What is python 80% programming 20% snake
- For subtopic CQ analysis, enable sorting based on user/code/score/feedback
- For all forms have a red * for required fields
- Add kill words to the database such that when a user attempts to run code which includes a kill word, the code does not run
- Cancel all programs taking more than 2seconds to execute
- Have keywords added to a course when creating the course instead of adding keywords each time a CQ is created, when a CQ is created, one can then tick checkboxes for keywords to be used for that CQ
- Codepad can be preloaded with code from subtopic
- Configure the codepad to also support Java, Perl and C
8. References


Appendix A
Prototypes

A1 Low Fidelity Prototypes 1
Table 12 Low Fidelity Prototypes 1
Python is a powerful and expressive language that has very simple syntax. The language is used as a variety of disciplines: application, web, and game development. Python can do it all. Courses are based on Python 3.

Java is a class-based, object-oriented language that is platform independent meaning that code that runs on one platform does not need to be recompiled to run on another.

**Table 13 - Low Fidelity Prototypes 2**

### Low Fidelity Prototypes 2

**Login**

Username: [ ]

Password: [ ]

**Student Home Page**

**Student Table of Contents**

<table>
<thead>
<tr>
<th>Topic 1</th>
<th>Topic 2</th>
<th>Topic 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is Python</td>
<td>2 I/O Strings</td>
<td>3 Selection</td>
</tr>
<tr>
<td>a What is Python</td>
<td>a Basic I/O</td>
<td>4 Iteration</td>
</tr>
<tr>
<td>b Identifiers</td>
<td>b String Methods</td>
<td>Topic 1</td>
</tr>
<tr>
<td>c Functions</td>
<td>c Numbers</td>
<td>Topic 2</td>
</tr>
<tr>
<td>d Importing Modules</td>
<td>d Importing Modules</td>
<td>Topic 3</td>
</tr>
</tbody>
</table>

**Student Slide View and Coding Question**

```
What is Python cool?
```

`print (2+5)`

Clear Run

Notepad Comments Output

Save

### Personal Notes for Topic 1

Success|0.05sec|stdin=14

stdout=Out of Range

Save

### Student Slide View and Annotations/Notepad

I Love Python

Clear Run

Student Slide View and Comments

Play around with code samples here

```
print (2+5)
```

Clear Run

Notepad Comments Output

Save

### Student Slide View and Coding Question

What is Python cool?

- It is open source and freely available
- Comes with “batteries included” – vast set of standard and other free libraries
- Use in successful businesses and applications
  - Netflix
  - Google
- It is very popular and has an enthusiastic and active community

Evaluate what version of Python is installed

```
import random

number = 10

if number % 5 == 0:
    print("number is a multiple of 5")
else:
    print("number is not a multiple of 5")
```

Clear Run

Save

Contents 1 Python Basics...
### 1. Python Basics

Python is a powerful and expressive language that has very simple syntax. The language is used in a variety of disciplines: application, web, and game development. Python can do it all. Courses are based on Python 3.

<table>
<thead>
<tr>
<th>Book Title</th>
<th>Image</th>
<th>Description</th>
<th>Add</th>
</tr>
</thead>
</table>

### Loops

**For Loops**

<table>
<thead>
<tr>
<th>MCQ</th>
<th>Question</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Add</th>
<th>Finish</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MCQ</th>
<th>Question</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Add</th>
<th>Finish</th>
</tr>
</thead>
</table>

### MCQ

1. Python is considered to be a high-level programming language
   (A) - True  ()
   (B) - False ()
2. Each assembly language instruction corresponds to one machine language instruction
   (A) - True  ()
   (B) - False ()

### Coding Question

**Python Basics**

**Sub Topic 1**

- **What is Python**
- **Slides**: intro.ppt
- **Video**: intro.flv
- **PDF**: intro.pdf
- **MCQ**: mcq_pyth_basics1
- **Coding**: cq_pyth_basics1

**Sub Topic 2**

- **What is Python**
- **Slides**: intro.ppt
- **Video**: intro.flv
- **PDF**: intro.pdf
- **MCQ**: mcq_pyth_basics2
- **Coding**: cq_pyth_basics2

### Add MCQ

**Loops**

**For Loops**

<table>
<thead>
<tr>
<th>MCQ</th>
<th>Question</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Add</th>
<th>Finish</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MCQ</th>
<th>Question</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Add</th>
<th>Finish</th>
</tr>
</thead>
</table>

---

### Add Topic/Subtopic

**Student Slide View and Multiple Choice Question**

**Admin Add MCQ**

**Admin Edit Topic/Subtopic**

**Admin Manage Books/Add Book**

**Admin Home**
A3 High Fidelity Prototype 1
Table 14 High Fidelity Prototype 1 (Student)

iTextBook is an interactive and fun way to learn to code using rich media examples combined with online code execution and evaluation.

Table: 14 High Fidelity Prototype 1 (Student)

<table>
<thead>
<tr>
<th>Course Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>Introduction</strong></td>
</tr>
<tr>
<td>- What is python</td>
</tr>
<tr>
<td>- Identifiers</td>
</tr>
<tr>
<td>- Functions</td>
</tr>
<tr>
<td>- Importing Modules</td>
</tr>
<tr>
<td>2. <strong>I/O String Number</strong></td>
</tr>
<tr>
<td>- Basic I/O</td>
</tr>
<tr>
<td>- String Methods</td>
</tr>
<tr>
<td>- Numbers</td>
</tr>
<tr>
<td>3. <strong>Selection</strong></td>
</tr>
<tr>
<td>- seldad</td>
</tr>
<tr>
<td>4. <strong>Daily Bargain Hub</strong></td>
</tr>
<tr>
<td>- sdf</td>
</tr>
<tr>
<td><strong>Scorebook</strong></td>
</tr>
</tbody>
</table>

**Course (Home) Page**

Python 3 Contents

**Brief History of Python**
- Created by Guido van Rossum in early 1990s, still very active in its development
- Benevolent Dictator for Life (BDI)
- Open Source language, developed over time by large, active Python community
- Evolved to latest version, Python 3
- Managed by Python Software Foundation - www.python.org

**Subtopic Slides View and Notepad**
Subtopic Video View and Notepad

Subtopic Slides View and Comments

Subtopic Video View and Codepad

Subtopic Slides View and Multiple Choice Question
Brief History of Python

- Created by Guido van Rossum in early 1990s, still very active in its development
  - Benevolent Dictator for Life (BDFL)
- Open Source language, developed over time by large, active Python community
- Evolved to latest version, Python 3
- Managed by Python Software Foundation
  - www.python.org

CSC1010H
Python Basics
Gary Stewart
gstewart@cs.uct.ac.za
Table 15 - High Fidelity Prototype 1 (Admin / Lecturer View)

Login

Course Page

Add Course

Edit Course
Course Contents

Add Subtopic

Location: Course / Course

Topic: Liveness Analysis

Number: 

Name: 

PfSlides: Choose File

Video: Choose File

Add Subtopic

Add Topic

COMPILERS
Liveness Analysis

Gary Stewart
uct csc303x 2012

Register Allocation
- IR trees are used to determine instructions, but registers are not assigned
- Can we assign registers arbitrarily?
- What if:
  - mov ax, 14
  - add ax, 20
  - mov ax, ax
  - add ax, ax
- was translated to:
  - mov ax, ax
  - add ax, ax

Subtopic Slides View and Add Multiple Choice Questions
Subtopic Slides View and Edit Multiple Choice Questions

Subtopic Slides View and Add Coding Question

Add Users
Appendix B
User Experimentation Forms

B1 Consent Form

UNIVERSITY OF CAPE TOWN
COMPUTER SCIENCE DEPARTMENT
iTextBook High Fidelity Prototype Usability Testing

Consent Form

i. I voluntarily agree to participate in this evaluation
ii. I am aware that I may withdraw from the evaluation at any time without compromise to my well-being
iii. I agree to my responses being used for educational and research purposes
iv. I understand that my identity will be coded and as such my responses will be unidentifiable/anonymous
v. I have read this consent form, the information contained within and been given opportunity to clarify anything that I did not understand

Researcher: ________________________________

Name of Participant: ________________________________

Signature of Participant: ________________________________

Date: ________________________________
B2 Introduction and Background Information Form

Introduction

The iTextBook is an Interactive Authoring and Reading Tool for Introductory Programming. It may be used as a supplementary aid to an introductory computer programming course such as Introduction to Python or Introduction to Java. The iTextbook will be used by lecturers to create interactive online versions of the course containing lecture notes, videos and assessments such as multiple choice and coding questions which will be done online or on a tablet device. Students will therefore use the tool to learn, comment on lecture material as well as take the assessments.

Background Information

Please rate your level of computer literacy

Please rate how often you use the web

Please rate your experience with touch devices (Tablet Version)
B3 Student Evaluation Tasks
Questionnaire / Participant Tasks using the iTextBook Web Interface

NB: When you feel you have completed a task, please say so.

Task 1 – Login (username – stud , password – stud )

Task 2 – Select Python 3 Course

Task 3 – Select Introduction Topic then What is python subtopic

Task 4 – Task 11 relate to the topic / subtopic selected in Task 4

Task 4 – 30 Seconds to Familiarise with interface

Task 5 – Navigate through the slides

Task 6 – Play video

Task 7 – Make Notes

Task 8 – Add comments

Task 9 – Play with sample code, Execute and Download executed file

Task 10 – Do MCQ

Task 11 – Do Coding Question

Task 12 – Select the next subtopic under Introduction topic

Task 13 – Select IO String Topic then String Methods subtopic

Task 14 – Add comments

Task 15 – Select Java course

Task 16 – Go to courses page and freestyle

Task 17 – Logout
B4 Student Post Test Interview

Post Test Interview

1. Name three words or characteristics that describe this Web Application

2. What are the three things you like best about the Web Application?

3. What are the three things you like least about the Web Application?

4. If you could make one significant change to this Web Application, what change would you make?

5. Are there materials you would like to see added to the Web site / Tablet Application? Which ones?

6. Would you recommend this Web site to a colleague? To a friend?

7. Do you have any other questions or comments about the Web site / Tablet Application or your experiences with it?

8. I found the iTextBook to be relevant to learning introductory programming
9. I found the navigation between subtopics and topics to be straightforward

1 2 3 4 5

Strongly Disagree  Strongly Agree

10. I found the subtopic discussion forum / commenting feature useful

1 2 3 4 5

Strongly Disagree  Strongly Agree

11. I found the subtopic personal notepad useful

1 2 3 4 5

Strongly Disagree  Strongly Agree

12. I found the dual view between course materials and coding box useful

1 2 3 4 5

Strongly Disagree  Strongly Agree

13. I found the subtopic multiple choice quizzes useful

1 2 3 4 5

Strongly Disagree  Strongly Agree

14. I found the subtopic coding questions useful

1 2 3 4 5

Strongly Disagree  Strongly Agree
B5 Lecturer/Admin Evaluation Tasks
Questionnaire / Participant Tasks using the iTextBook Web Interface

NB: When you feel you have completed a task, please say so.

Task 1 – Login (username – admin, password – admin)

Take some time to familiarise with screen

Task 2 – Create a course for python

Task 3 – Add a Topic

Task 4 - Add a subtopic

Task 5 - Set an MCQ

Task 6 - Set a CQ

Task 7 – Task 11 relate to Python 3 (a previously created course) Select Courses -> Python3

Task 7 – View Course/User Analytics

Task 8 - View Subtopic MCQ Analytics

Task 9 - View Subtopic CQ Analytics

Task 10 - View Subtopic Comments Analytics

Task 11 – Go to courses page and freestyle

Task 12 – Logout
B6 Lecturer / Admin Post Evaluation Interview

Post Test Interview

1. Name three words or characteristics that describe this Web Application

2. What are the three things you like best about the Web Application?

3. What are the three things you like least about the Web Application?

4. If you could make one significant change to this Web Application, what change would you make?

5. Are there materials you would like to see added to the Web site / Tablet Application? Which ones?

6. Would you recommend this Web site to a colleague? To a friend?

7. Do you have any other questions or comments about the Web site / Tablet Application or your experiences with it?
8. I found the iTextBook authoring/admin functionality to be intuitive

1  2  3  4  5

Strongly Disagree  Strongly Agree

9. I found the iTextBook CQ admin functionality to be intuitive

1  2  3  4  5

Strongly Disagree  Strongly Agree

10. I found the iTextBook MCQ admin functionality to be intuitive

1  2  3  4  5

Strongly Disagree  Strongly Agree

11. I found the course/user analytics easy to understand

1  2  3  4  5

Strongly Disagree  Strongly Agree

12. I found the subtopic MCQ analytics easy to understand

1  2  3  4  5

Strongly Disagree  Strongly Agree

13. I found the subtopic CQ analytics easy to understand

1  2  3  4  5

Strongly Disagree  Strongly Agree

14. I found the subtopic comments analytics easy to understand

1  2  3  4  5

Strongly Disagree  Strongly Agree