# CHAPTER 1

INTRODUCTION

## EBooks

EBooks are becoming a popular alternative medium for physical books. People want to read their favorite books across different devices and platforms. With the invention of wide range of personal computers, laptops, tablets and smart phones we now have a pretty diversified range of display form factors. Developing software that runs perfectly on each of these devices and platforms become a cumbersome job.

## EPUB FORMAT

EPUB (short for electronic publication) is a [free](http://en.wikipedia.org/wiki/Free_standard) and [open](http://en.wikipedia.org/wiki/Open_standard) [e-book](http://en.wikipedia.org/wiki/E-book) standard by the [International Digital Publishing Forum](http://en.wikipedia.org/wiki/International_Digital_Publishing_Forum) (IDPF). Files have the extension .epub.

EPUB is designed for [reflowable content](http://en.wikipedia.org/wiki/Reflowable_document), meaning that an EPUB reader can optimize text for a particular display device. EPUB also supports fixed-layout content. The format is intended as a single format that publishers and conversion houses can use in-house, as well as for distribution and sale. It supersedes the [Open eBook](http://en.wikipedia.org/wiki/Open_eBook) standard. In August 2009, the IDPF announced that they would begin work on maintenance tasks of the EPUB standard.

EPUB format essentially is a zip archive that contains a collection of html, style sheets and images.

## BENEFITS OF EPUB

* Open format
* Well supported libraries for manipulation
* Lots of ebooks available in this format
* Easy to manipulate book contents
* Zip container - No complex packaging

## FEATURES

* [Free](http://en.wikipedia.org/wiki/Free_standard) and [open](http://en.wikipedia.org/wiki/Open_standard)
* [Reflow able](http://en.wikipedia.org/wiki/Reflowable_document) (word wrap) and resizable text
* Inline [raster](http://en.wikipedia.org/wiki/Raster_graphics) and [vector](http://en.wikipedia.org/wiki/Vector_graphics) images
* Embedded [metadata](http://en.wikipedia.org/wiki/Metadata)
* [DRM](http://en.wikipedia.org/wiki/Digital_rights_management) support
* Support for alternative renditions in the same file
* Use of out-of-line and inline [XML](http://en.wikipedia.org/wiki/XML) islands to extend the functionality of EPUB

# CHAPTER 2

SYSTEM STUDY

## International Digital Publishing Forms

The International Digital Publishing Forum (IDPF) is the global trade and standards organization dedicated to the development and promotion of electronic publishing and content consumption.

The work of the IDPF promotes the development of electronic publishing applications and products that will benefit creators of content, makers of reading systems, and consumers.  The IDPF develops and maintains the EPUB content publication standard that enables the creation and transport of reflowable digital books and other types of content as digital publications that are interoperable between disparate EPUB-compliant reading devices and applications.

The IDPF welcomes book, newspaper, journal and magazine publishers, booksellers, software and reading system developers, authors, and other groups interested in digital reading to join our organization.

Digital publishing has come a long way in the last two years, with eBooks now outselling hard covers. EPUB has been a key standard at the heart of the eBook revolution. EPUB 3, which converges digital publishing with HTML5 and modern Web Standards, and delivers robust accessibility and global language support, is a monumental update to the standard.

## The goals of the IDPF are to

* Promote industry-wide adoption of electronic publishing through standards development, conferences, best practices, and demonstrations of proven technology.
* Develop, publish, and maintain common standards (e.g. EPUB) relating to electronic publications and promote the successful adoption of these specifications.
* Encourage interoperable implementations of EPUB publications and reading systems and provide a forum for resolution of interoperability issues.
* Identify, evaluate, and recommend standards created by other bodies related to electronic publishing.
* Provide a forum for the discussion of issues and technologies related to electronic publishing.
* Accommodate differences in language, culture, reading and learning styles, and individual abilities.

## EXISTING SYSTEMS

To address the problems in distributing eBooks, different organizations come with different solutions. Every solution had its own advantages and disadvantages. Most of these solutions are proprietary, patented and closed source. Some formats are not particularly suitable for reading eBooks.

Reader software is needed to read the contents of the eBooks. Most of the organizations that developed proprietary file formats for eBooks have also developed eBook readers to read the books. While these readers are good at reading the organization’s eBook file format, it fails miserably in features and cross compatibility. Some readers provide rich features but they need native software installation and this prevents the software to be installed across different platforms.

## Some of the proprietary eBook file formats are

|  |  |  |
| --- | --- | --- |
| **Name** | **Format** | **File Extension** |
| KF8 (Amazon Kindle) | Kindle | .azw; .kf8 |
| Broadband eBooks (BBeB) | Sony media | .lrf; .lrx |
| Microsoft LIT | Microsoft Reader | .lit |
| Comic Book Archive file | compressed images | .cbr, .cbz, .cb7, .cbt, .cba |
| Compiled HTML | Microsoft Compiled HTML Help | .chm |
| DjVu | DjVu | .djvu |
| iBook (Apple) | iBook | .ibooks |
| Portable Document format | Adobe Portable Document Format | .pdf |

## Existing readers

* Adobe pdf reader
* Microsoft LIT reader
* Mobi pocket reader
* Amazon Kindle reader

## Drawbacks

* Book format is closed source, patented and proprietary
* EBook format distribution is less
* Platform dependent
* Need to install a software to read the eBooks
* Dictionary lookup might be available but look up is intrusive
* No mechanism to see sample images for “words”
* Need to install different software for different devices

## PROPOSED SYSTEM

Design an eBook reader that is based on open standards and free software. It should be portable across platforms and devices. There should be no need of installing new software at the user’s side to be able to read the books. Dictionary look up and image look up should be available. The reader should work on both desktop, tablets and smart phone mobile devices. Provide an option to learn meanings of all words in the page. When learning meaning alone is not sufficient, give option to show first 3 Google image result. By using these features the reader learns all the words before reading the page. But it will not interrupt the book reading. Based on these requirements, we arrived at the following proposal.

### The reader should be

1. Web based
2. Should work on any browser
3. Fully written in JavaScript
4. Scalable across platforms
5. Uses Google dictionary API and Google image API
6. Support caching at the browser to avoid downloading books each time
7. User can upload eBooks to their account and read anywhere.

## ADVANTAGES

* No need to install new software
* Browser is available on almost all Personal computers, tablets and mobile devices
* User can read books anywhere
* Powerful Image lookup reduces difficulty in understanding unknown words
* EPUB format is widely supported and lots of eBooks are distributed in EPUB format

# CHAPTER 3

SYSTEM SPECIFICATIONS

## WEB SERVER HARDWARE SPECIFICATIONS

* PROCESSOR : Intel(R) Xeon(R)
* Cores : 8
* SPEED : 3.4 GHz
* RAM : 16 GB
* HARD DISK : 20 GB (MIN)

## SOFTWARE SPECIFICATION:

* Python 2.7
* Python Bottle 0.11
* Python epub 0.5.5
* Python NLTK
* Beaker 1.6.4
* Apache 2.2
* NGINX 1.2.8
* Git 1.7.4.1
* Linux 2.6.32
* Twitter Bootstrap 2.3.1
* Mod\_wsgi
* Sqlite3
* JQuery 1.9.1
* Any Modern HTML5 compatible browser

# CHAPTER 4

SOFTWARE DESCRPTION

## CHOOSING TECHNOLOGIES

### EPUB

* Open format
* Well supported libraries for manipulation
* Lots of ebooks available in this format
* Easy to manipulate book contents
* Zip container - No complex packaging

### HTML5

HTML5 is the latest version of Hypertext Markup Language for creating web pages. It’s very easy to learn even for a beginner. The interest about HTML5 is increasing day by day and the number of

web professionals adopting this technology is also increasing rapidly. Latest standard of **HTML** is HTML5 and it is an output of joint efforts from **W3C** and **WYSWYG**. It introduces more efficient markup to eliminate the use of external plugging like Adobe Flash Player. HTML5 also reduces the use of scripting languages and it’s more **SEO** friendly. It is device independent and supported by latest versions of all major web browsers.

In particular, HTML5 adds many new syntactic features. These include the new <video>, <audio> and <canvas> elements, as well as the integration of scalable vector graphics (SVG) content (that replaces the uses of generic <object> tags) and [MathML](http://en.wikipedia.org/wiki/MathML) for mathematical formulas. These features are designed to make it easy to include and handle [multimedia](http://en.wikipedia.org/wiki/Multimedia) and [graphical](http://en.wikipedia.org/wiki/2D_computer_graphics) content on the web without having to resort to proprietary [plugins](http://en.wikipedia.org/wiki/Plug-in_%28computing%29) and [APIs](http://en.wikipedia.org/wiki/Application_programming_interface). Other new elements, such as <section>, <article>, <header> and <nav>, are designed to enrich the [semantic](http://en.wikipedia.org/wiki/Semantic_web) content of documents.

### PYTHON

* Easy to learn and develop.
* Code should be clean and maintainable.
* Easy to deploy application.
* Well supported across platforms and Open source.

Python is a [general-purpose](http://en.wikipedia.org/wiki/General-purpose_programming_language), [high-level programming language](http://en.wikipedia.org/wiki/High-level_programming_language) whose design philosophy emphasizes code [readability](http://en.wikipedia.org/wiki/Readability). Python's syntax allows programmers to express concepts in fewer [lines of code](http://en.wikipedia.org/wiki/Lines_of_code) than would be possible in languages such as [C](http://en.wikipedia.org/wiki/C_%28programming_language%29),and the language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple [programming paradigms](http://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](http://en.wikipedia.org/wiki/Imperative_programming) and [functional programming](http://en.wikipedia.org/wiki/Functional_programming) styles. It features a fully [dynamic type](http://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](http://en.wikipedia.org/wiki/Memory_management) and has a large and comprehensive [standard library](http://en.wikipedia.org/wiki/Standard_library).

### APACHE

**Apache** is a widely deployed [open source](http://en.wikipedia.org/wiki/Open_source) [web server](http://en.wikipedia.org/wiki/Web_server) that supports many languages. Perfect for serving [HTTP](http://en.wikipedia.org/wiki/HTTP), HTTPS protocols. It is licensed under liberal Apache license and it runs on [Unix](http://en.wikipedia.org/wiki/Unix), [Linux](http://en.wikipedia.org/wiki/Linux), [BSD variants](http://en.wikipedia.org/wiki/BSD), [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X), [Solaris](http://en.wikipedia.org/wiki/Solaris_%28operating_system%29), [AIX](http://en.wikipedia.org/wiki/AIX), [HP-UX](http://en.wikipedia.org/wiki/HP-UX), and [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows)

* Very stable
* Reliable
* Easy to deploy
* Process based server
* Works well with python
* Open source

### SQLITE

**SQLite** is a [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) contained in a small (~350 [KB](http://en.wikipedia.org/wiki/Kilobyte)) [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) programming [library](http://en.wikipedia.org/wiki/Library_%28computer_science%29). In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it.

SQLite is [ACID](http://en.wikipedia.org/wiki/ACID)-compliant and implements most of the [SQL](http://en.wikipedia.org/wiki/SQL) standard, using a dynamically and weakly typed SQL [syntax](http://en.wikipedia.org/wiki/Syntax) that does not guarantee the [domain integrity](http://en.wikipedia.org/wiki/Integrity_constraints).

SQLite is a popular choice as [embedded database](http://en.wikipedia.org/wiki/Embedded_database) for local/client storage in [application software](http://en.wikipedia.org/wiki/Application_software) such as [web browsers](http://en.wikipedia.org/wiki/Web_browser). It is arguably the most widely deployed [database engine](http://en.wikipedia.org/wiki/Database_engine), as it is used today by several widespread browsers, [operating systems](http://en.wikipedia.org/wiki/Operating_system), and [embedded systems](http://en.wikipedia.org/wiki/Embedded_system), among others. SQLite has many [bindings](http://en.wikipedia.org/wiki/Language_binding) to programming languages.

Unlike [client–server](http://en.wikipedia.org/wiki/Client–server) database management systems, the SQLite engine has no standalone [processes](http://en.wikipedia.org/wiki/Process_%28computing%29) with which the application program communicates.

Instead, the SQLite [library](http://en.wikipedia.org/wiki/Library_%28computing%29) is [linked in](http://en.wikipedia.org/wiki/Linker_%28computing%29) and thus becomes an integral part of the application program. (In this, SQLite follows the precedent of [Informix SE](http://en.wikipedia.org/wiki/IBM_Informix#Other_Products) of [c. 1984](http://www.iiug.org/faqs/informix-faq/ifaq01.htm.1#1.2)) The library can also be called dynamically.

The application program uses SQLite's functionality through simple [function calls](http://en.wikipedia.org/wiki/Subroutine), which reduce [latency](http://en.wikipedia.org/wiki/Latency_%28engineering%29) in database access: function calls within a single process are more efficient than [inter-process communication](http://en.wikipedia.org/wiki/Inter-process_communication).

SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform [file](http://en.wikipedia.org/wiki/Computer_file) on a host machine. It implements this simple design by [locking](http://en.wikipedia.org/wiki/Lock_%28computer_science%29) the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

* Easy configuration
* No need of dedicated db server
* Open source
* Supports transactions
* Light weight

### jQuery

It is a multi-browser JavaScript library designed to simplify the client-side scripting of HTML. jQuery is the most popular JavaScript library in use today.

jQuery is free, open source software. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications.

jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets.

The modular approach to the jQuery library allows the creation of powerful dynamic web pages and web applications.

* DOM element selections using the multi-browser open source selector engine Sizzle, a spin-off out of the jQuery project
* DOM manipulation based on CSS selectors that uses node elements name and node elements attributes (id and class) as criteria to build selectors
* Events
* Effects and and Animation
* Utilities - such as user agent information, feature detection
* Compatibility methods that are natively available in modern browsers but need fall backs for older ones - For example the inArray() and each() functions.

## DESIGN GOALS

* Book reader should be web based.
* It should support mobile and desktop.
* Use open format for eBook.
* Application should be responsive for different form factors
* DRY
* Make use of existing tools and libraries wherever appropriate

# CHAPTER 4

PROJECT DESIGN

## MODULE (SERVER)

In server module we perform the task such as user registration, it is needed to enter in the process. User details send to the database through the database access layer after finishing the user signup details.

Session contains user login details and it provides the information of the user reading book to the user mail box. It bookmarks the page of which book the user read and what until the user read.

EPUB files are uploaded from the browser. After this it will extract to the EPUB processor. Initially all the EPUB files are zip file. While EPUB files are in zip file the user cannot read eBook content.

Using EPUB processor we can convert the zip file to unzip file. Now the user can read the book content in the web server. The unzip EPUB file are stored in the database. User can get the unzip files directly from the database while open the book.

## MODULE (CLIENT)

In client side the figure shows the design details, content display details. For process the page content server content handle the content show, it works by the content processor.

In JavaScript we right the coding to get the content of the page by using book object. After get this the user can view the content of the page.

Book reader displays the book we can view the previous page and next page content.

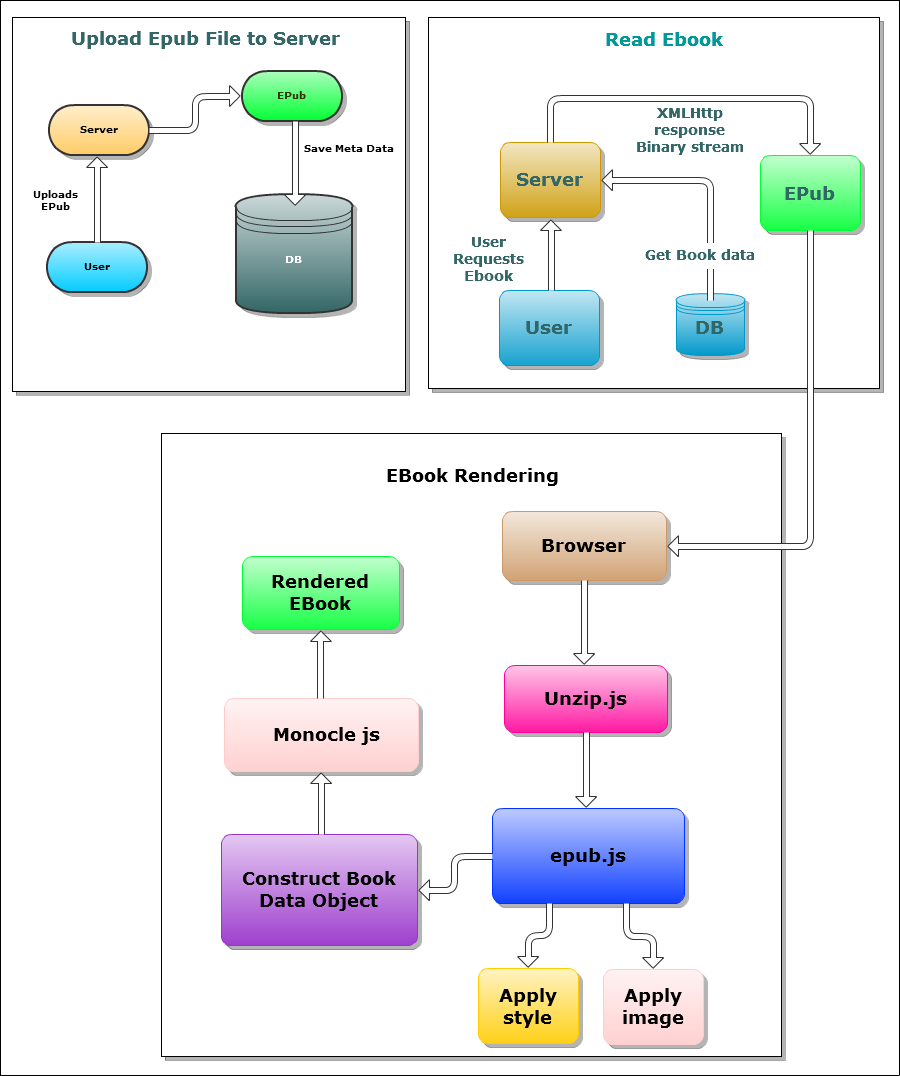
To get the meaning of the particular words in Wikipedia from the Google browser it use web service manager. It shows the excerpt information from the browser. EPUB image and styles also get from the using by the content processor, it display eBook image and styles.

User need to bookmark the page which the page finished in the eBook, it send to the user mail from the session. EBook should get facility to move the page to next page as well as previous page.

We can able to resizing the page using controls and change the font size

of the page content using custom controls. While view the page content it should be in better alignment to read the book, these are processor are handle by the content processor.

## WORD FLOW

****

# CHAPTER 4

SYSTEM TESTING

## OBJECTIVES

Testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

If testing is conducted successfully, it uncovers errors in the software. As a secondary benefit, testing demonstrates that software functions appear to be working according to specification, that performance requirements appear to have been met.

In addition, data collected as testing is conducted provide a good indication of software reliability and some quality as a whole.

The objectives of testing are:

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding an as yet undiscovered error.
* A successful test is one that uncovers an as yet undiscovered error.

Any engineering product can be tested in one of the two ways

1. Knowing the specific function that a product has been designed to perform, tests can be performed that demonstrate each function is fully operational **Black box testing.**

2. Knowing the internal workings of product, tests can be conducted to ensure that a product performs according to specification and all internal components have been adequately exercised **White Box Testing**

## UNIT TESTING

The first level of testing is called Unit Testing. In this different modules are tested against the specification proceed the modules. The goal is to test the internal logic of the modules.

## SYSTEM TESTING

The system developed has been successfully tested under the system testing and all system elements are found to carry out their allocated tasks correctly. There are the various types of testing strategies implemented for the system developed.

## INTEGRATION TESTING

Integration Testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit-testing modules and build a program structure that has been dictated by design.

## ACCEPTANCE TESTING

At the culmination of integration, software is completely assembled as a package, interfacing errors have been uncovered and final series of software test ‘Validation test’ begins.

# CHAPTER 5

CONCLUSION

In this project we have successfully developed an EPUB reader based on HTML and browser/server technologies that are fully compliant to EPUB format. The whole project provides a solid foundation for an eBook reader that uses only open source technologies and free software. The web application is hosted at webfaction hosting service and it can be accessible through Internet.

This project itself is an open source software and anyone can fork this from github. Full source code of the project is available at github at the following url <https://github.com/janujohn/cuneiform>

## FUTURE ENHANCEMENTS

* Support multiple languages
* Machine learning algorithm to detect word learning curve
* Bookmarking of pages
* Note taking support
* EBook upload via web accessible URL in addition to conventional file upload
* Social media integration

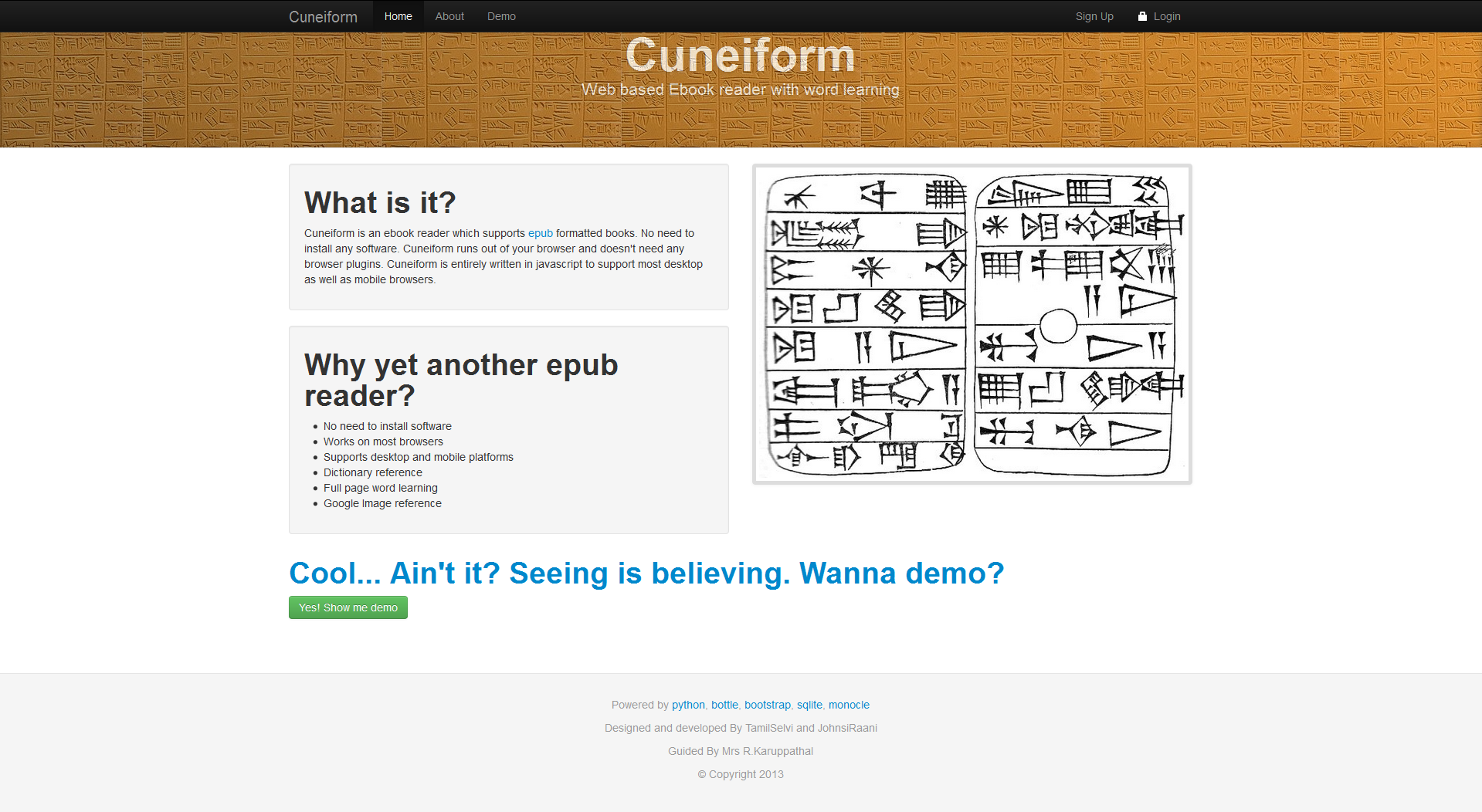
**Reference:** idpf Epub 3.0 standard <http://code.google.com/p/epub-revision/>  
**Author:** International Digital Publishing Forum (IDPF)

# CHAPTER 6

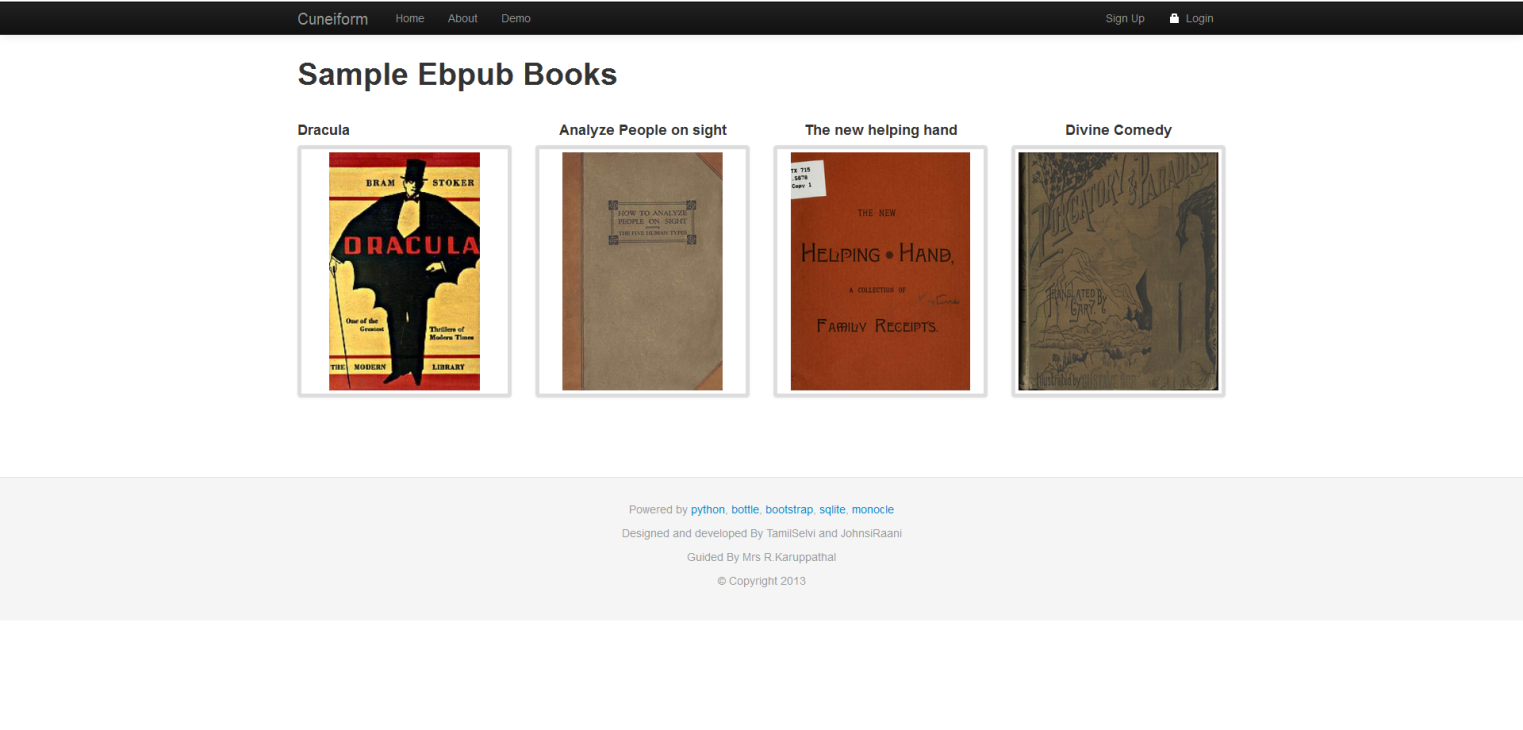
Appendix I

## SNAPSHOTS

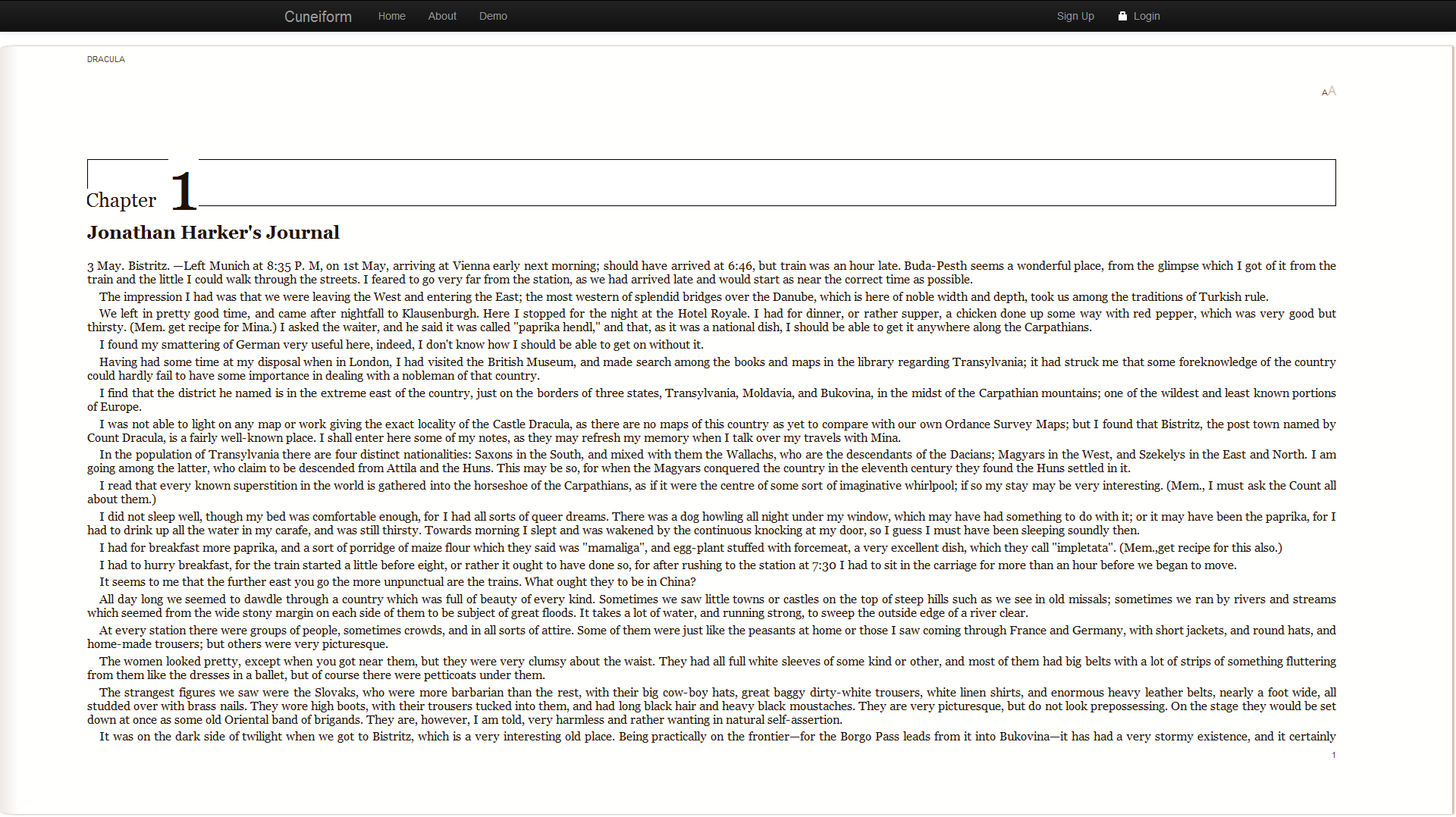
### Home page



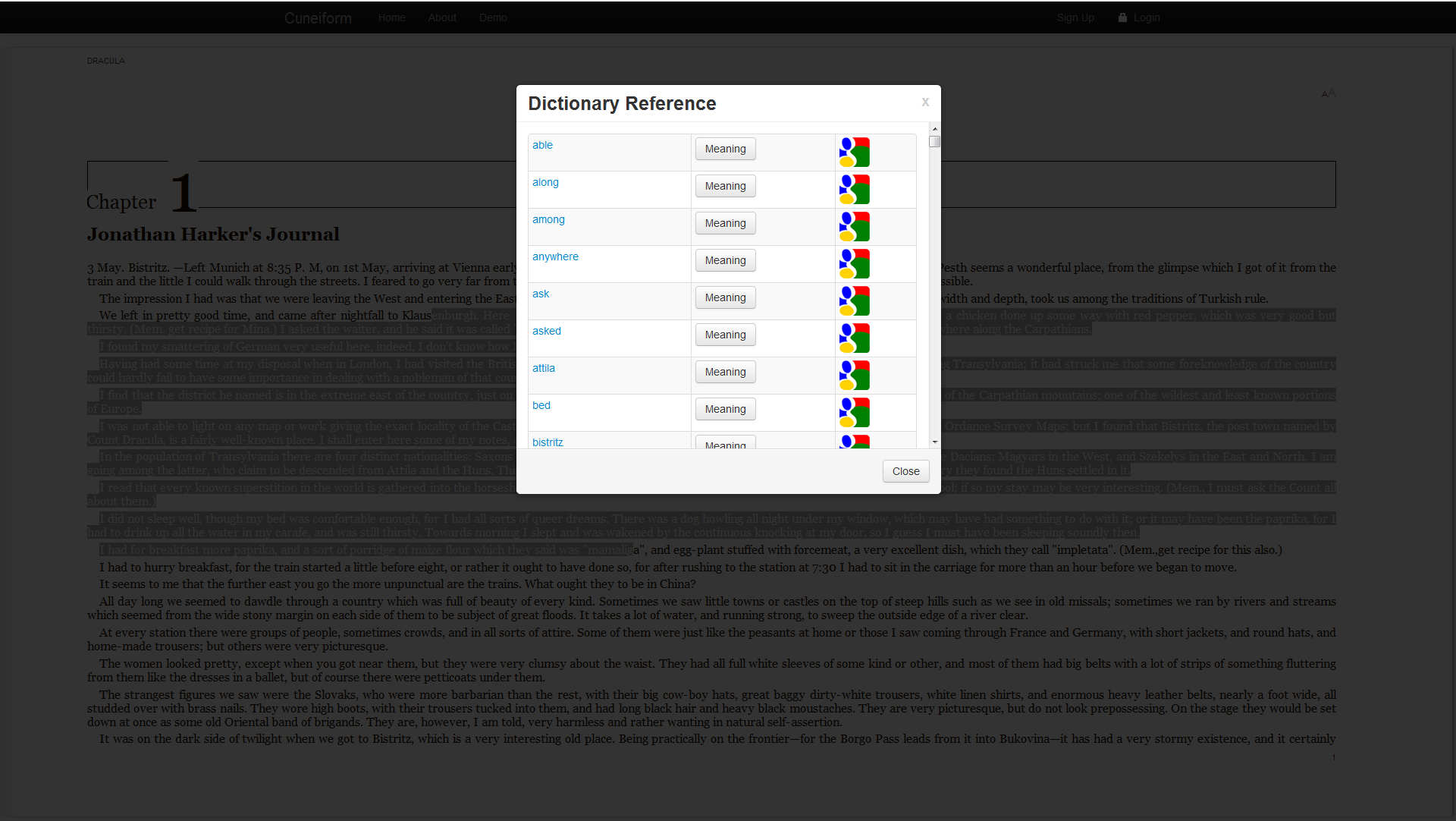
### Demo



### Reading a book



### Dictionary Reference



### Image lookup

### 5.pngSignup page

### 6.png

### Login page

### 7.png

### EPUB Upload Page

### 8.pngMyBooks Page

### 9.png

# CHAPTER 7

Appendix II

## SOURCE CODE

Here we have included only core files. All other source code are available at github (<https://github.com/janujohn/cuneiform>)

### app.wsgi

import os, sys

cwd = os.path.dirname(\_\_file\_\_)

sys.path.insert(0, cwd)

import settings

os.chdir(cwd)

from bottle import route, default\_app, static\_file, template, debug, request, hook, redirect, error

import beaker.middleware

import sqlite3

import hashlib, time, epub

import extractwords

from define import get\_meaning

session\_opts = {

'session.type': 'file',

'session.data\_dir': './session/',

'session.auto': True,

}

# Disable this once stable

debug(True)

# set beaker session in request.session

@hook('before\_request')

def setup\_request():

request.session = request.environ['beaker.session']

@route('/home')

@route('/')

def home():

return template('home', session=request.session)

@route('/signup', method=['GET','POST'])

def signup():

if request.method=='POST':

name = request.POST.get('name', '').strip()

email = request.POST.get('email', '').strip()

password = request.POST.get('password', '').strip()

password\_confirm = request.POST.get('password\_confirm', '').strip()

con = sqlite3.connect('cuneiform.sqlite3')

cursor = con.cursor()

cursor.execute('insert into users(name, email, password, date\_joined) values(?, ?, ?, strftime("%s", "now"))', (name, email, password))

userid = cursor.lastrowid

con.commit()

cursor.close()

redirect('/thanks')

return template('signup', session=request.session)

@route('/login', method=['GET', 'POST'])

def login():

if request.method=="POST":

email = request.POST.get('email', '').strip()

password = request.POST.get('password', '').strip()

con = sqlite3.connect('cuneiform.sqlite3')

c = con.cursor()

c.execute('select \* from users where email=? and password=?', (email, password))

result = c.fetchone()

if not result:

return template('login', error='User name or password is wrong. Please try again', session=request.session)

else:

signin\_user(result)

redirect('/mybooks')

return template('login', session=request.session)

@route('/ajax/learn', method='POST')

def learn():

if request.method=='POST':

text = request.POST.get('text', '').strip()

extracted\_words = extractwords.extract(text)

return {'error': 'false', 'words': extracted\_words}

@route('/ajax/get\_meaning')

def web\_definition():

word = request.GET.get('word', '').strip()

if word:

return {'web\_definition': get\_meaning(word), 'error': 'false'}

else:

return {'error':'true'}

@route('/logout')

def logout():

request.session.delete()

redirect('/home')

def signin\_user(user\_data):

request.session['userid'] = user\_data[0]

request.session['username'] = user\_data[1]

request.session['email'] = user\_data[2]

request.session['joined\_date'] = user\_data[4]

@route('/demo')

def demo():

return template('demo', session=request.session)

@route('/mybooks')

def mybooks():

if 'userid' not in request.session:

redirect('/login')

con = sqlite3.connect('cuneiform.sqlite3')

c = con.cursor()

c.execute('select \* from books where userid=? and is\_active=1 ', (request.session['userid'],))

books = c.fetchall()

return template('mybooks', session=request.session, active\_mybooks='active', books=books)

@route('/upload', method=['GET', 'POST'])

def upload():

if 'userid' not in request.session:

redirect('/login')

error = ''

if request.method=='POST':

#do something

upload = request.files.get('book')

name, ext = os.path.splitext(upload.filename)

if ext.lower()=='.epub':

# write it to books dir

save\_path = get\_book\_save\_path(request.session['userid'], upload.filename)

book\_path = '/'.join(save\_path)

with open(book\_path, 'w') as save\_file:

save\_file.write(upload.file.read())

book = epub.open\_epub(book\_path)

title = str(book.opf.metadata.titles[0][0])

author = str(book.opf.metadata.creators[0][0])

con = sqlite3.connect('cuneiform.sqlite3')

cursor = con.cursor()

cursor.execute('insert into books(userid, filename, title, author, is\_cover, is\_active, date) values(?, ?, ?, ?, ?, ?, strftime("%s", "now"))', (request.session['userid'], save\_path[1], title, author, 0, 1))

bookid = cursor.lastrowid

con.commit()

cursor.close()

redirect('/mybooks')

else:

error = "Error! Please upload only EPub format file"

return template('upload', session=request.session, active\_upload='active', error=error)

def get\_book\_save\_path(userid, filename):

upload\_dir = settings.app\_dir + '/static/books/' + str(userid)

if not os.path.exists(upload\_dir):

os.makedirs(upload\_dir)

filename = hashlib.sha1(str(userid) + filename + str(time.time())).hexdigest()

return [upload\_dir, filename]

@route('/about')

def about():

return template('about', session=request.session)

@route('/thanks')

def thanks():

return template('signup\_thanks', session=request.session)

@route('/demoreader')

def demoreader():

epub = request.GET.get('epub', '').strip()

title = epub.split('/')[1].replace('-', ' ').replace('.epub', '').title()

return template('reader', session=request.session, epub=epub, title=title)

@route('/reader')

def reader():

epub = request.GET.get('epub', '').strip()

con = sqlite3.connect('cuneiform.sqlite3')

c = con.cursor()

c.execute('select title from books where filename=?', (epub.split('/')[1],))

title = c.fetchone()[0]

return template('reader', session=request.session, epub=epub, title=title)

@route('/static/<filename:path>')

def send\_static(filename):

return static\_file(filename, root= cwd + '/static')

@error(404)

def error404(error):

return template('404', session={})

#application = default\_app()

application = beaker.middleware.SessionMiddleware(default\_app(), session\_opts)

### extractwords.py

# -\*- coding: utf-8 -\*-

import re, nltk

"""Extracts non trivial words for a raw text

Stops nltk's stopwords and single alphabets from a-z

Returns a list of extracted words

Authored by: JohnsiRaani.M <hi@johnsiraani.com>"""

nltk\_stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',

'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can',

'will', 'just', 'don', 'should', 'now',]

alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

others = ['st', 'nd', 'rd', 'th']

stopwords = nltk\_stopwords + alphabets + others

"""Extracts and returns non trivial words from given raw text

Skips ASCII incompatible encoded characters"""

def extract(raw):

filtered = []

word\_list = nltk.word\_tokenize(raw.lower().decode('utf-8'))

# compile regex for punctuation

punctuation = re.compile(r'[-.?!,":;\'`\[\]()\*%&@$^|0-9]')

word\_list = [punctuation.sub("", word).strip() for word in word\_list]

# skip empty string words

cleaned = [word for word in word\_list if word.strip() != '']

# skip stopwords

stopped = [word for word in cleaned if word not in stopwords]

# remove duplicates

stopped = sorted(list(set(stopped)))

for a in stopped:

try:

str(a)

filtered.append(a)

except UnicodeEncodeError:

pass

return filtered

### reader.js

Monocle.Events.listen(window, 'load', initFn);

function initFn() {

$('footer').css('display', 'none');

var epub = $('#epub\_location').val()

var client = new XMLHttpRequest();

client.onreadystatechange = function () {

if (client.readyState == 4 && client.status == 200) {

book = new JSEpub(client.responseText);

book.processInSteps(function(step, extras) {

if (step === 5) {cb\_book(book);}

});

} else if (client.readyState == 4 && client.status < 400 && client.status > 299) {

alert('Book could not be found');

} else if (client.readyState == 4) {

alert('Error getting book');

}

};

client.overrideMimeType('text/plain; charset=x-user-defined');

client.open("GET", epub);

client.send(null);

return;

function cb\_book(book) {

var fnmeta = function(key) {

return {

title: book.opf.metadata['dc:title']['\_text'],

creator: book.opf.metadata['dc:creator']['\_text']

}[key];

}

var comp = {};

var components = [];

for(var i=0;i<book.opf.spine.length;i++) {

var spineName = book.opf.spine[i]

href = book.opf.manifest[spineName]['href']

var doc = book.files[href];

var html = new XMLSerializer().serializeToString(doc);

comp[href] = html

components[i] = href;

}

var fncomponent = function(componentId) {

return comp[componentId];

}

var fncomponents = function() {

return components;

}

var toc=[]

/\*var toc = [];

for(var i=0;i<book.toc.length;i++) {

toc[i] = { title: book.toc[i].title, src: book.toc[i].fileName };

}\*/

var fncontents = function() {

return toc;

}

bookData = {

getComponents: fncomponents,

getContents: fncontents,

getComponent: fncomponent,

getMetaData: fnmeta

}

var readerOptions = {};

/\* PLACE SAVER \*/

var bkTitle = bookData.getMetaData('title');

var placeSaver = new Monocle.Controls.PlaceSaver(bkTitle);

readerOptions.place = placeSaver.savedPlace();

readerOptions.panels = Monocle.Panels.Marginal;

readerOptions.stylesheet = "body { " + "color: #210;" + "font-family: Palatino, Georgia, serif;" + "}";

window.reader = Monocle.Reader(

'reader',

bookData,

readerOptions,

function(reader) {

reader.addControl(placeSaver, 'invisible');

// SPINNER

var spinner = Monocle.Controls.Spinner(reader);

reader.addControl(spinner, 'page', { hidden: true });

spinner.listenForUsualDelays('reader');

// Hide loader after book loaded

Monocle.Events.listen('reader', 'monocle:loaded', function(){

$('#loader').css('display', 'none')

$('iframe').each(function(i,e){

e.contentDocument.onmouseup = getSelectionText;

})

});

// Because the 'reader' element changes size on window resize. we should notify it of this event.

Monocle.Events.listen(

window,

'resize',

function () { window.reader.resized() }

);

// MAGNIFIER CONTROL

var magnifier = new Monocle.Controls.Magnifier(reader);

reader.addControl(magnifier, 'page');

// BOOK TITLE RUNNING HEAD

var bookTitle = {}

bookTitle.contentsMenu = Monocle.Controls.Contents(reader);

reader.addControl(bookTitle.contentsMenu, 'popover', { hidden: true });

bookTitle.createControlElements = function () {

var cntr = document.createElement('div');

cntr.className = "bookTitle";

var runner = document.createElement('div');

runner.className = "runner";

runner.innerHTML = reader.getBook().getMetaData('title');

cntr.appendChild(runner);

Monocle.Events.listenForContact(

cntr,

{start: function (evt) {

if (evt.preventDefault) {

evt.stopPropagation();

evt.preventDefault();

} else {

evt.returnValue = false;

}

reader.showControl(bookTitle.contentsMenu);

}

}

);

return cntr;

}

reader.addControl(bookTitle, 'page');

// CHAPTER TITLE RUNNING HEAD

var chapterTitle = {

runners: [],

createControlElements: function (page) {

var cntr = document.createElement('div');

cntr.className = "chapterTitle";

var runner = document.createElement('div');

runner.className = "runner";

cntr.appendChild(runner);

this.runners.push(runner);

this.update(page);

return cntr;

},

update: function (page) {

var place = reader.getPlace(page);

if (place) {

this.runners[page.m.pageIndex].innerHTML = place.chapterTitle();

}

}

}

reader.addControl(chapterTitle, 'page');

reader.listen(

'monocle:pagechange',

function (evt) {

// $('iframe')[1].contentDocument.onmouseup = getSelectionText;

// if (!document.all) $('iframe')[1].contentDocument.captureEvents(Event.MOUSEUP);

chapterTitle.update(evt.m.page); }

);

/\*

reader.listen(

'monocle:turn',

function(evt) {

$('iframe')[1].contentDocument.onmouseup = getSelectionText;

if (!document.all) $('iframe')[1].contentDocument.captureEvents(Event.MOUSEUP);

});

\*/

// PAGE NUMBER RUNNING HEAD

var pageNumber = {

runners: [],

createControlElements: function (page) {

var cntr = document.createElement('div');

cntr.className = "pageNumber";

var runner = document.createElement('div');

runner.className = "runner";

cntr.appendChild(runner);

this.runners.push(runner);

this.update(page, page.m.place.pageNumber());

return cntr;

},

update: function (page, pageNumber) {

if (pageNumber) {

this.runners[page.m.pageIndex].innerHTML = pageNumber;

}

}

}

reader.addControl(pageNumber, 'page');

reader.listen(

'monocle:pagechange',

function (evt) {

pageNumber.update(evt.m.page, evt.m.pageNumber);

}

);

// Scrubber

var scrubber = new Monocle.Controls.Scrubber(reader);

reader.addControl(scrubber, 'popover', { hidden: true });

var showFn = function (evt) {

evt.stopPropagation();

reader.showControl(scrubber);

scrubber.updateNeedles();

}

for (var i = 0; i < chapterTitle.runners.length; ++i) {

Monocle.Events.listenForContact(

chapterTitle.runners[i].parentNode,

{ start: showFn }

);

Monocle.Events.listenForContact(

pageNumber.runners[i].parentNode,

{ start: showFn }

);

}

});

}

}

var t = '';

function getSelectionText(e) {

t = (document.all) ? $('iframe')[1].contentDocument.selection.createRange().text : $('iframe')[1].contentDocument.getSelection();

if(t=='') {

t = (document.all) ? $('iframe')[0].contentDocument.selection.createRange().text : $('iframe')[0].contentDocument.getSelection();

}

if(t!='') {

$('#learn').css('display', 'block')

$('#hidLearn').val(t);

}

else {

$('#learn').css('display', 'none')

$('#hidLearn').val('');

}

}

function getMeaning(element) {

word = $(element).attr('data-word')

if($(element).attr('data-meaning') == '') {

$.getJSON('/ajax/get\_meaning?word='+word, function(data){

$(element).attr('data-meaning', data['web\_definition']['webDefinitions'][0])

$(element).popover({

'title' : 'Definition for "<b>'+word+'</b>":',

'content': data['web\_definition']['webDefinitions'][0],

'html': true,

'placement': 'bottom'

});

$(element).popover('show');

});

}

}

$(document).ready(function(){

$('#learnButton').click(function(){

data = {'text': $('#hidLearn').val()};

var url = '/ajax/learn';

$.ajax({

dataType: "json",

url: url,

data: data,

type: 'POST',

success: function(data){

var table = '<table class="table table-striped table-bordered table-condensed">';

if(data['error'] == 'false') {

for(var i=0;i<data['words'].length;i++) {

table += '<tr>'

table += '<td><a href="#">'+data['words'][i]+'</a></td>'

table += '<td><a onclick="getMeaning(this)" data-meaning="" class="btn" data-word="'+data['words'][i]+'" >Meaning</a></td>'

table += '<td><a onclick="getImage(\''+data['words'][i]+'\')"><img src="/static/img/googleIcon.gif"></a></td>'

table += '</tr>'

}

table += '</table>'

$('#wordnet').html(table)

$('#learnModal').modal();

}

}

});

});

});

function getImage(word) {

url = 'https://ajax.googleapis.com/ajax/services/search/images?v=1.0&callback=processImage&q=' + word;

$.ajax({

url: url,

dataType: 'jsonp',

jsonpCallback: processImage

});

}

function processImage(data) {

if(data) {

var list = '<ul class="thumbnails">';

for(var i=0;i<data['responseData']['results'].length;i++) {

list += '<li class="span3"><div class="thumbnail"><img height="260" src="'+data['responseData']['results'][i]['url']+'" /></div></li>';

}

list += '</ul>';

$('#googleImage').html(list);

$('#ImageModal').modal();

}

}