jamk.fi

Teach9

Teaching Application

James Pearce

Bachelor's thesis
December 2016
Technology, communication and transport
Degree Programme in Software Engineering

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s) Pearce, James	Type of publication Bachelor's thesis	Date December 2016
		Language of publication: English
	Number of pages 28	Permission for web publication: (X)
Title of publication **Teach9** Teaching Application		
Degree programme Degree Programme in Software Engineering		
Supervisor(s) Lappalainen-Kajan, Tarja		
Assigned by Ukkonen, Sirkku		

Abstract

There is already existing software to assist teachers with their teaching; however, compromises usually must be made when creating software, and the software cannot do everything every user requires. The assigner of the thesis, Sirkku Ukkonen, a student at the University of Jyväskylä, needed a new teaching application to fulfill her specific needs.

The task was to create a teaching application for the assigner to use in her Master's thesis. The objective of the teaching application was to allow increased student participation in the planning of lessons and courses.

Django was chosen as the framework for the application. It was intended to facilitate fast development; however, learning Django and its tools was more time consuming than expected. It therefore took more time than originally planned to create an application that performed the basic functions required in a teaching environment; this left little time to focus on improving the existing tools and adding more to allow better student and teacher interaction.

The result is an application allowing the assigner to create, update and display teaching materials with enough communication between student and teacher; however, the application could be developed further.

Keywords/tags (subjects)
Python, Django, software development, software architecture, software

Miscellaneous

jamk.fi

Kuvailulehti

Tekijä(t) Pearce, James	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä Joulukuu 2016
	Sivumäärä 28	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: (X)
Työn nimi **Teach9** Ohjelmisto opetukseen		
Tutkinto-ohjelma Ohjelmistotekniikan koulutusohjelma		
Työn ohjaaja(t) Tarja Lappalainen-Kajan		
Toimeksiantaja(t) Sirkku Ukkonen		

Tiivistelmä

Opetuskäyttöön tarkoitettuja ohjelmistoja on jo olemassa, mutta usein on varaa parantaa tai täyttää erityistarpeita ohjelmistolla. Toimeksiantajana toimi Sirkku Ukkonen, Jyväskylän yliopiston maisteriopiskelija, pääaineenaan englannin kielen aineenopettajaopinnot. Hän halusi ohjelman maisterin tutkintonsa osaksi englanninkielisen opetusmateriaalin luomista, päivitystä ja toteuttamista varten.

Ohjelmiston tehtävänä oli opetusympäristönä luoda paremmat yhteydet opettajan ja oppilaan välille. Oppilaiden oli tarkoitus pystyä osallistumaan oppituntien suunnitteluun. Kaikkia toivottuja ominaisuuksia ei pystytty tekemään, mutta oppilaat pystyvät äänestämään ohjelmaa käyttäen, mitä opettajan ehdottamista opetustavoista he toivoivat tunneilla tehtävän. Tämän tarkoituksena oli motivoida oppilaita antamalla heille mahdollisuudet vaikuttaa tunnilla tehtäviin aktiviteetteihin ja käsiteltäviin sisältöihin.

MVC-mallia käyttävä Django valittiin ympäristöksi. Django on kehitetty Pythonia käyttäen. Djangon keskeisin ominaisuus on helppo ja nopea kehitys. Uuden ohjelmointikielen ja ympäristön oppiminen oli hankalampaa kuin alussa ajateltiin. Koska aikaa meni oletettua enemmän valittujen työkalujen opetteluun, ohjelmiston kehitys jäi vähäisemmäksi.

Ohjelmisto, joka pystyy toteuttamaan opetusympäristön vaatimukset, onnistui, mutta opettajan ja oppilaan välistä keskustelua parantavat ominaisuudet jäivät vähäiseksi.

Avainsanat (asiasanat)
Python, Django, ohjelmistokehitys, ohjelmistoarkkitehtuuri

Muut tiedot

Table of Contents

Acronyms

API	Application Programming Interface
ARGS	Arguments
KWARGS	Keyword arguments
CSS	Cascading Style Sheets
CRUD	Create, Read, Update, Delete
HTML	Hyper Text Mark-up Language
HTTP	Hyper Text Transfer Protocol
IDE	**I**ntegrated Development Environment
MVC	Model, View, Controller
OS	Operating System
UML	Unified Modelling Language
URL	Uniform Resource Locator

FIGURES

1 Introduction

This thesis deals with the requirements of the assigner to allow creating, displaying and updating teaching materials. The assigner is another student who studies at the University of Jyväskylä. The project is a part of the assigner´s Master's thesis. The teaching material was created in the form of a course.

The purpose of the project was to create a web application that the assigner can use to fulfil their personal teaching requirements. As an additional requirement, to give the application added value, it was decided to make the application flexible and allow any teacher to use it to create and update courses as well as display the courses to their students and peers.

There are many applications that perform similar functions; however, the assigner required one to use in their Master's thesis and to have an application that allows students more opportunities to be able to be a part of the course planning. Communication between the teacher and their students was to be the focus of the application's features.

The tools used were: Python, Django (a Python Web framework) and for the database SQLite. Python was chosen for the opportunity to learn and practice how to use it and because it is commonly used in applications. Django was a natural choice for the framework due to its ease of use and quick development methods.

The initial plan for the project was to allow students to have a forum within the application to allow communication with the teacher and other students on the course. Students could share their views of the course and its subjects on this forum; either privately with the teacher or publicly with everyone participating in the course.

”Three ways to foster autonomy, or self-regulated learning, are (1) allowing choice of materials, (2) changing teacher and learner roles, and (3) developing learner networks” (Candling and Byrnes 1995, quoted in Andrade and Evans 2013:51). Thus, the intention of allowing students to have more control on how they learn was to increase their motivation for learning and to improve the content of the course.

The time allotted for the project left the polishing of the application to a minimum and some “nice to have” features had to be cut. The key features were successfully implemented and no serious bugs were detected when the assigner tested the application.

2 Tools Used

2.1 Microsoft Visual Studio

Microsoft Visual Studio is an IDE from Microsoft. It has useful built-in functions and features that, for example allowed the easy creation of the virtual environment for the testing and development of the project. The tool's other notable useful features were its refactoring capabilities and IntelliSense.

"IntelliSense describes APIs as you type and uses auto-completion to increase speed and accuracy. Quick Info tool tips let you inspect API definitions, and squiggly lines let you know about issues, often showing them as you type." (Ref -Microsoft Visual Studio)

When creating the project with Microsoft Visual Studio it could be created directly into a virtual environment supported by the IDE. Not having the need of finding a suitable virtual environment was a great help at the beginning of the project. The virtual environment was difficult to activate if Microsoft Visual Studio was installed on the computer, this caused issues when developing the project on computers that didn't have it installed.

2.2 Django

Django is based on a MVC model. In Django, the names of some of the parts differ from those used in MVCs; however, they have the same function. Models are still called models in Django, a MVC view is called a template and the Controller is called a view.

"Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source." (Ref -Django project)

Using Django was simple to start with and with some experience it could be a very powerful tool. The learning curve was slightly steep to get more out of Django and Python if you had no previous knowledge. After becoming more familiar with the tools getting the project to the required quality was quite simple.

Microsoft Visual Studio's support of the version of Django used (1.1.0) was not optimal. Django's commands were only partially supported. Some of the newer commands were missing and there were deprecated commands listed. All the deprecated commands that were listed caused errors or did nothing when executed.

2.3 Python

"Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants, on the Mac, and on Windows 2000 and later." (Ref -General Python FAQ)

Getting familiar with using Pythons syntax did not take long and the user experience was pleasant after a while but getting the most out of it required some work. Having a high-level programming language was useful and gave a lot of options when solving the problems. There is plenty of room for improvement in the application and Python supports this.

Python is considered by users of www.quora.com to be slower than equivalent compiled languages (Ref -Python speeds). This should not be an issue with the application.

2.4 SQLite

SQLite version 3; also, known as SQLite3, was used for the projects database because Python 2.7 supports it without additional work and it did everything required for the prototype. If the project were to be used in production, then swapping to a more robust database should be easy with Django and it should be upgraded to a PostgreSQL database or something similar.

"SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle." (Ref -Python documentation on SQLite)

2.5 Other Tools

Other tools used were: Pip, Bootstrap, JQuery and Git. Pip is a Python package installation tool and so it had little to do with the development of the project; however, it made it a great deal simpler to get and update other tools installed. For example, Bootstrap and Django. One can also use it to get specific versions of Python packages to guarantee compatibility. Pip was installed by default in the virtual environment used by Microsoft Visual Studio and there was no reason to use another tool instead of it.

Bootstrap was used to give the project a better-looking appearance without having to spend a lot of time doing it and makes the application more mobile friendly. Being mobile friendly was not a focus of the project, however having the option that one could access the application with a mobile device successfully is a great feature.

JQuery is used for scripting minor tools on the HTML side of the project. The use of JQuery outside of what it is used in Django's admin tools was limited to the date picker for the courses and lessons.

Git is a version handling tool. The use of version handling tools is vital in software development and was luckily well used during the development of the project because it saved time near the end of the project after a hard drive failure. Had there not been any version handling a lot of the work would have been lost.

3 Requirements

3.1 Original

At first the requirements for the project were a bit vague and required some time to obtain full details of what the assigner wanted the application to do. This was the assigner's first time working with software development and it caused a lot of problems when conveying the strengths and limitations of the different tools that could be used.

The requirements boiled down to the assigner being able to create a single course with the application and then being to be able to present the course to her potential students using it. The main goal was to understand how her course would be set up and to make an application that can be used to create it.

Implementing features to allow students and the teacher to communicate within the application were planned; however, most of these features were removed because they would take too much time to implement.

3.2 What was used

Because the project is more of a proof of concept and not a final product the easiest to use tools were selected for the project. The assigner would be able to create her course and the requirements for that course would be the most important ones. As an additional requirement, to give the application more value, it was decided to make the application usable by other teachers and not to hardcode the functionality of the application to suit only her course.

“Nice to have” functions would be added if time allowed for it. Most of these were dropped because of time constraints. These functions were mostly usability upgrades or related to the communication between the student and the teacher. If the assigner’s Master’s thesis requires it, they can they can be implemented later.

4 The Project

4.1 Original plan

The plan was originally; one week of testing the tools (about 40 hours of work), two weeks getting the project going (about 80 hours of work), three more weeks working on the project (about 120 hours of work) and finishing the application at a slower pace of three weeks finalizing the project and writing the project thesis (about 40 hours of work). This would have brought the total time for the project and thesis to about 280 hours, however, unexpected changes delayed the project.

The idea was to create six components for the application. To save time Django's built-in authentication was used for User and Administration needs and two of the six components were removed (Personal Forum and Public Forum).

Django's authentication and administrative tools are great and can be modified to suit the projects needs. Creating new user and administration tools would have been too much work and not worth the investment of time for the prototype. The first version of the UML diagram with the originally planned forums can be seen in Figure 1. The components used in the project were: Courses, Lessons, Assignments and Activities can be seen in Figure 2 with their Django fields.

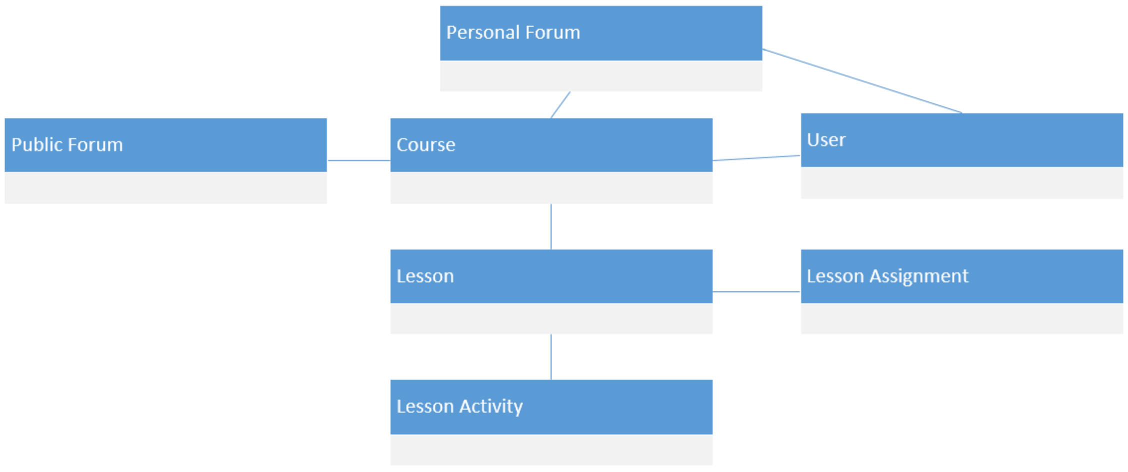


Figure 1. First version of the UML diagram for the project.

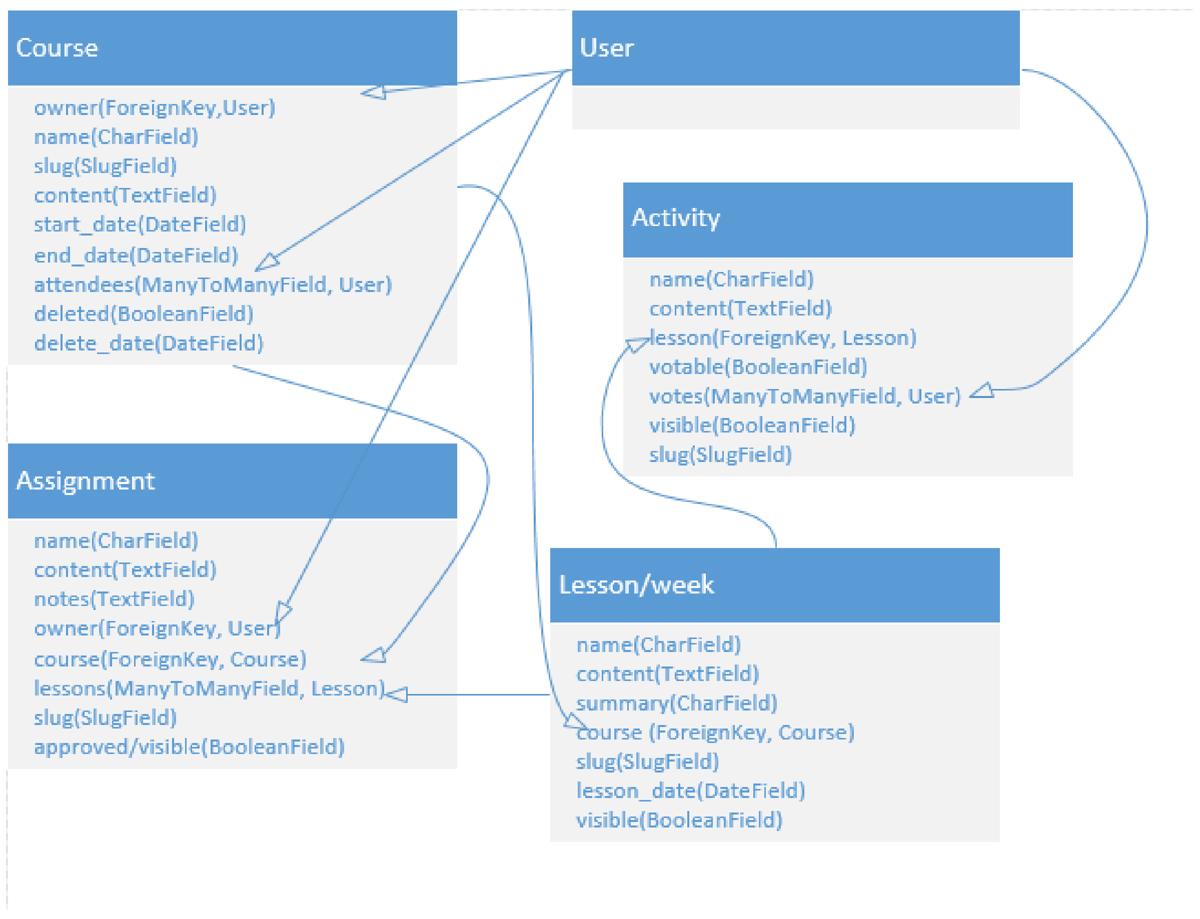


Figure 2. UML Diagram of the project with Django fields.

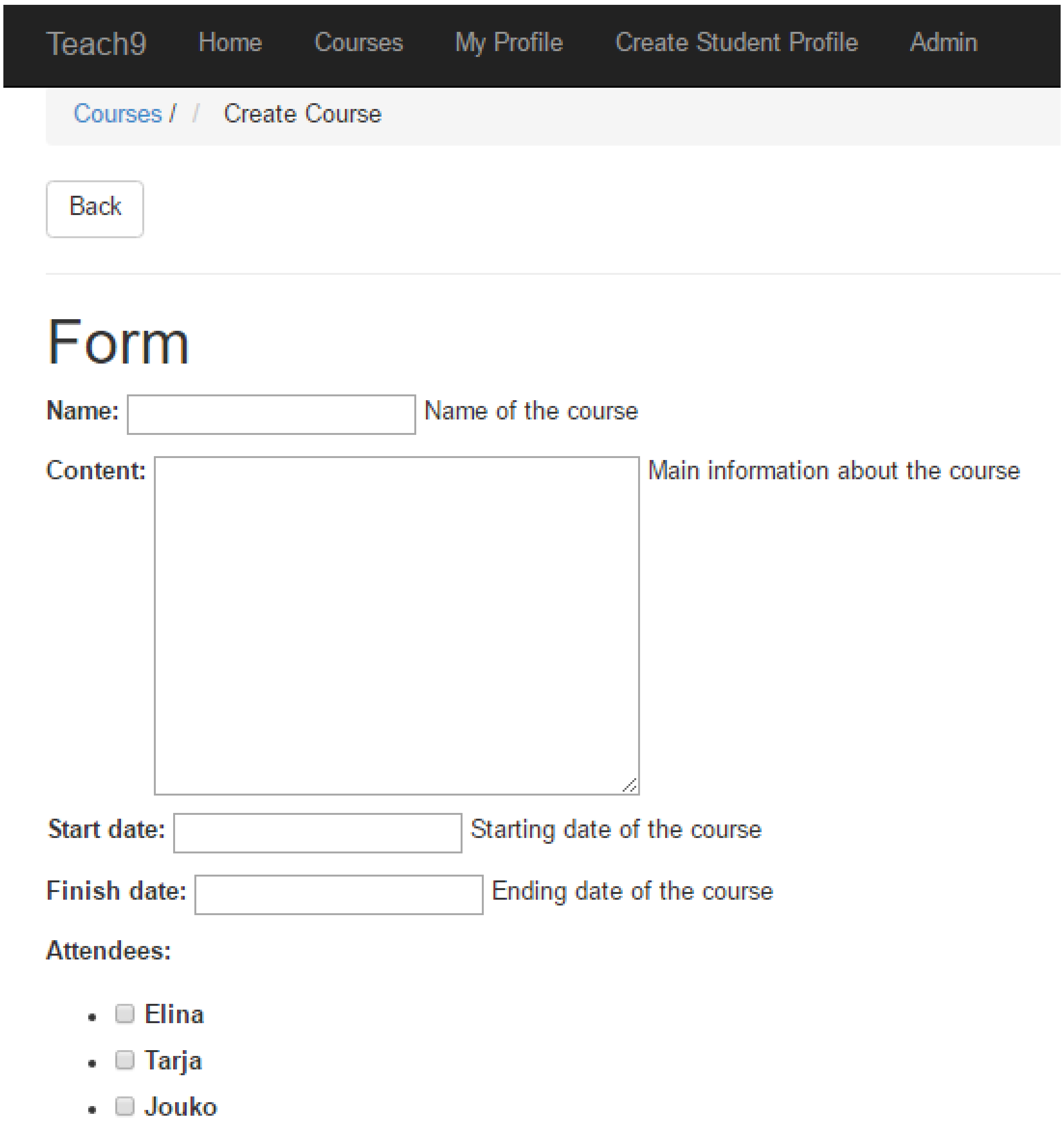
The fields in the model are mostly Django's built-in ones that allow Django to support multiple database architectures without having to make changes to the models, which was one of the reasons Django was selected for the project. More information on the fields can be found in the references.

4.2 Components of the project

The largest components in the application are users and courses. The other components are either linked directly to one of the larger ones or are part of a larger one. The user component uses Django's built in authentication and it was used as is with no modifications. One could have implemented custom functions in it; however, this was not required in the project.

Users can be students, teachers or administrators. Students can only attend courses. Teachers can create student accounts, attend courses, create courses and manage their own courses. Administrators can see and manage all courses and create any type of user account using Django's administrative tools.

Courses are what teachers create and maintain, they are meant to encompass all the teaching material for one course. Courses have a name, starting date, ending date, attendees and content that explains what the course is about. The next two parts of a course are the lessons and assignments. Because all courses are different, the elements composing a course must be flexible.



Form

Name: Name of the course

Content: Main information about the course

Start date: Starting date of the course

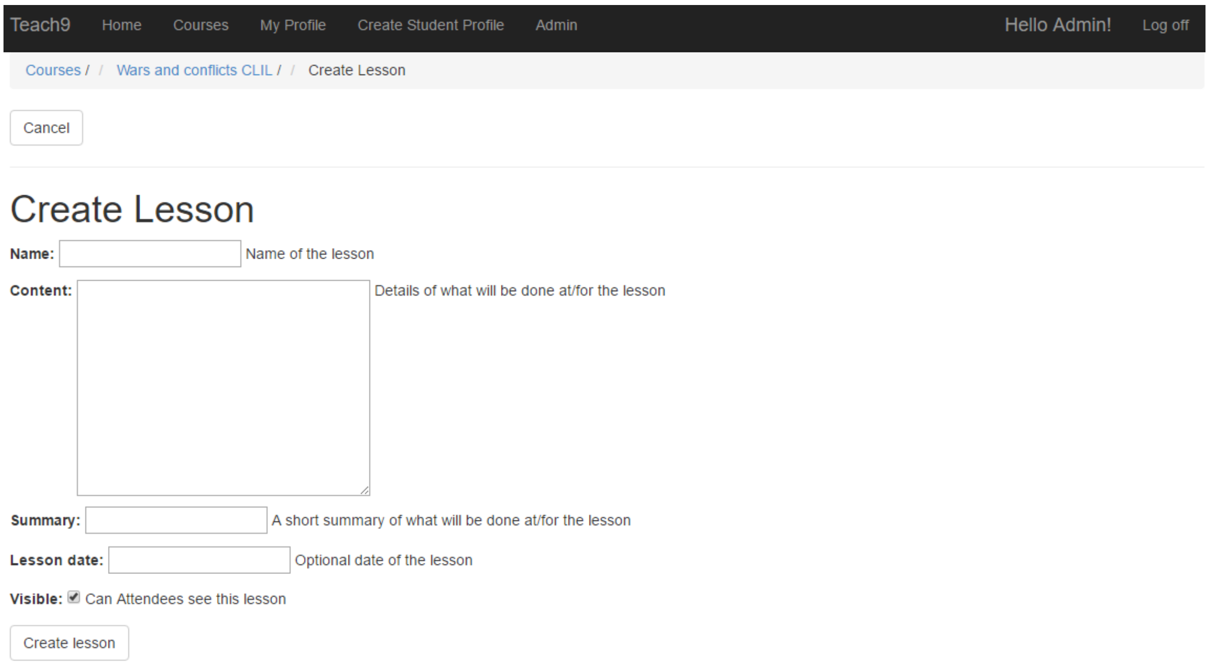
Finish date: Ending date of the course

Attendees:

- ☐ Elina
- ☐ Tarja
- ☐ Jouko

Figure 3. Create Course form

Lessons are used to create a rhythm for the learning and to manage when and what is on the agenda. Lessons; like courses, have a name and content. They additionally have a summary of the content that is shown on the list of lessons, an optional lesson date for when the lesson is and finally the option of is it visible to attendees of the course. Lessons that are not set to be visible to attendees can only be seen by the teacher.



Create Lesson

Name: Name of the lesson

Content: Details of what will be done at/for the lesson

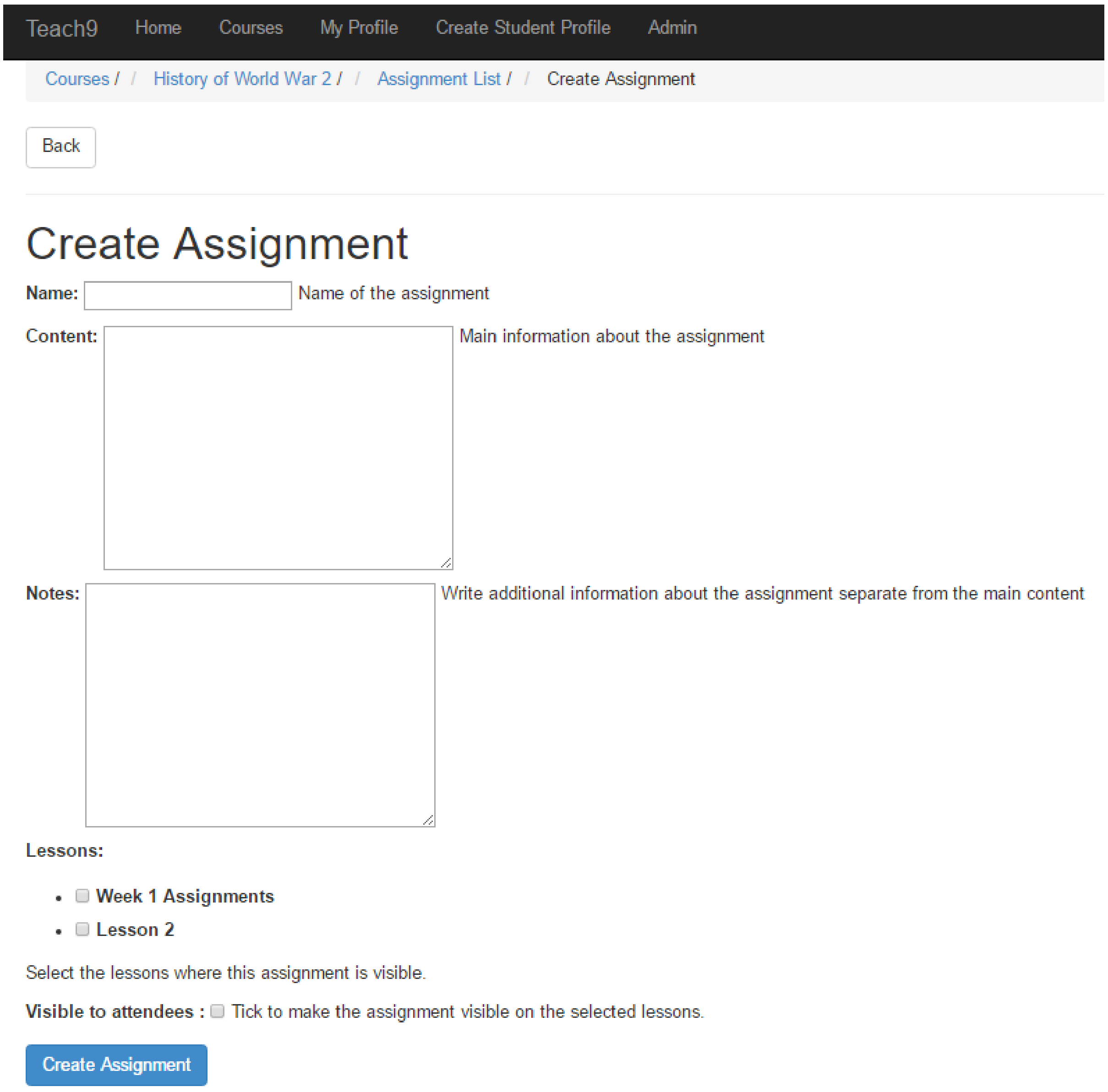
Summary: A short summary of what will be done at/for the lesson

Lesson date: Optional date of the lesson

Visible: ☑ Can Attendees see this lesson

Figure 4. Create lesson form

Assignments are used to create tasks and homework for students that they can return using the application. The assignment model is used also for the returning of assignments. The form presented to the user when creating and editing is tied to the relationship of the user to the course. The teacher created assignments are tied to the course and can be assigned to lessons to be displayed. Student created assignments are tied to lessons and are only visible to the teacher.


Teach9 Home Courses My Profile Create Student Profile Admin

Courses / History of World War 2 / Assignment List / Create Assignment

Back

Create Assignment

Name: Name of the assignment

Content: Main information about the assignment

Notes: Write additional information about the assignment separate from the main content

Lessons:

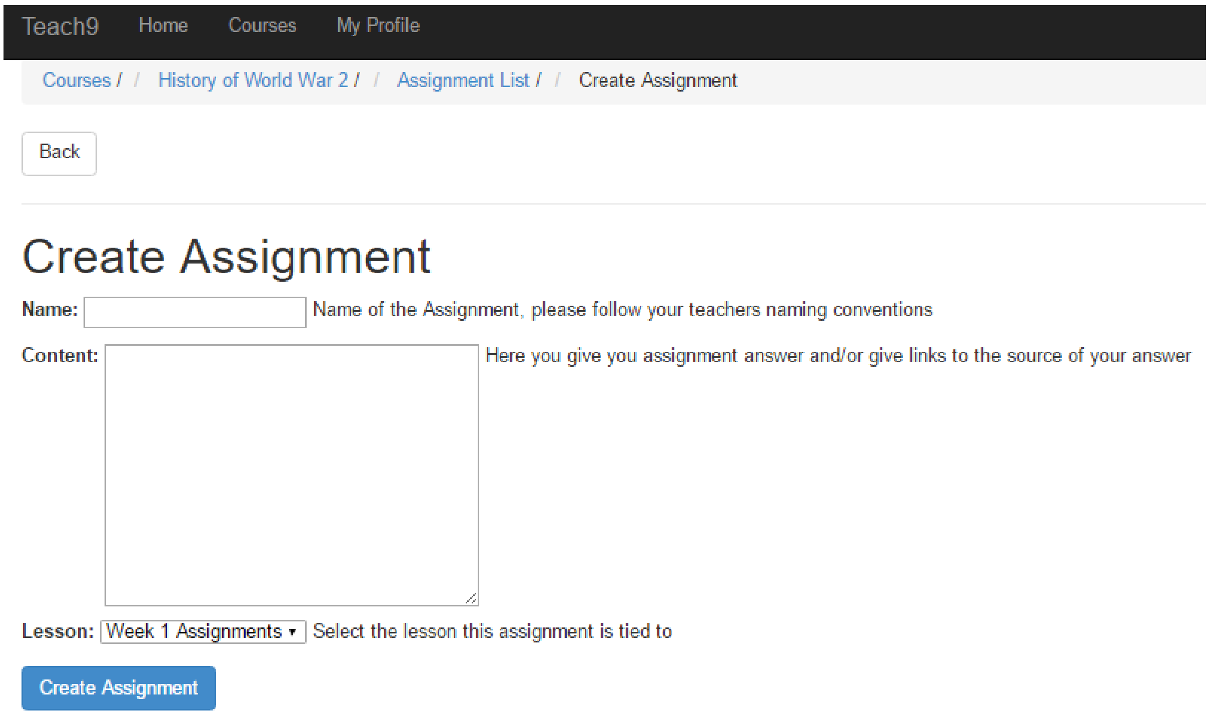
- Week 1 Assignments
- Lesson 2

Select the lessons where this assignment is visible.

Visible to attendees : Tick to make the assignment visible on the selected lessons.

Create Assignment

Figure 5. Create Assignment form for teachers


Teach9
Home
Courses
My Profile
Courses / History of World War 2 / Assignment List / Create Assignment
Back

Create Assignment

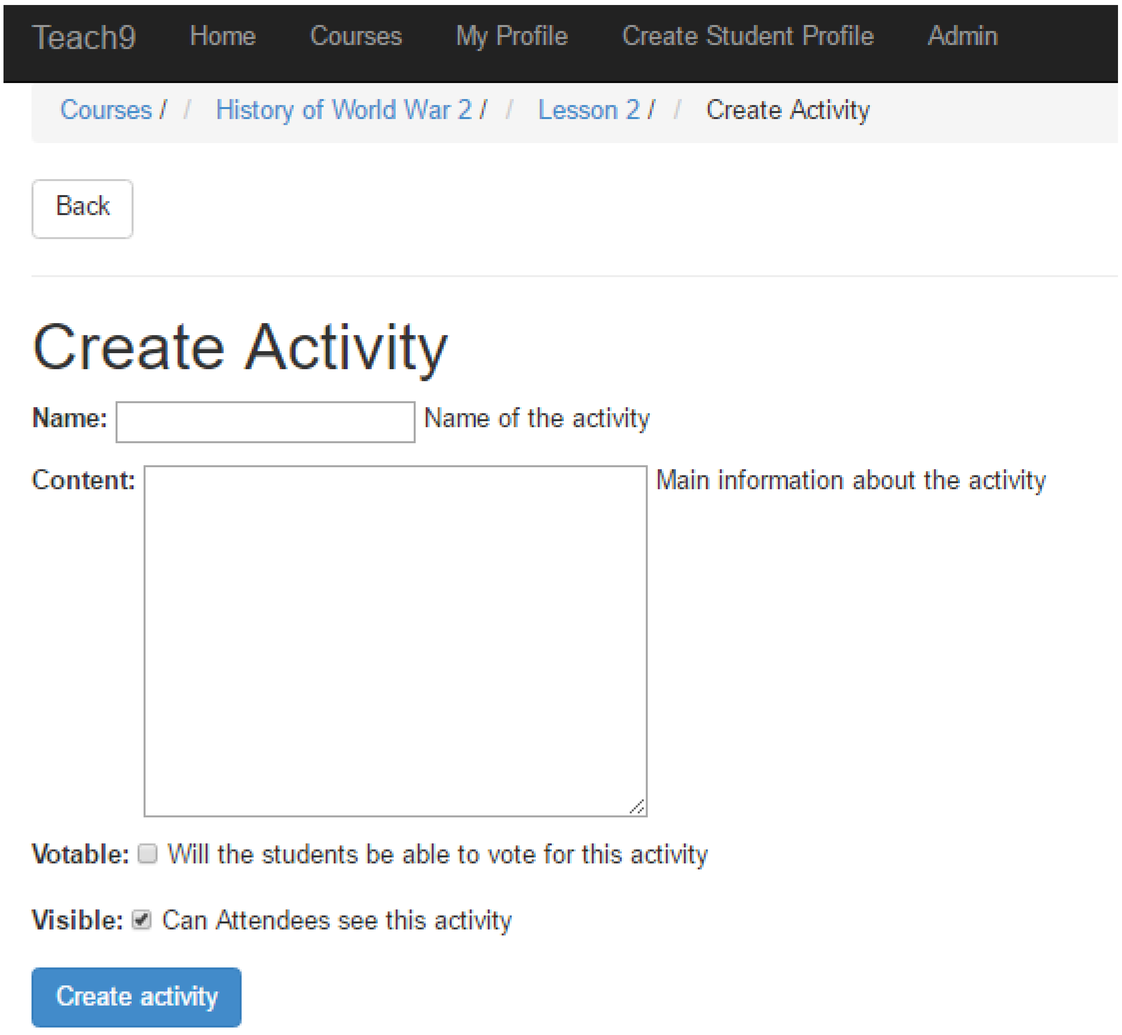
Name: Name of the Assignment, please follow your teachers naming conventions

Content: Here you give you assignment answer and/or give links to the source of your answer

Lesson: Week 1 Assignments ▾ Select the lesson this assignment is tied to

Figure 6. Create Assignment form for students

The final implemented component in the application are activities. They are tied to lessons and are meant to be used as themes or events for the lesson. A voting functionality allows the teacher to see what the students want to do during the lesson or how they want to learn a subject. To allow voting the teacher needs to activate it.


Teach9 Home Courses My Profile Create Student Profile Admin

Courses / History of World War 2 / Lesson 2 / Create Activity

Back

Create Activity

Name: Name of the activity

Content: Main information about the activity

Votable: ☐ Will the students be able to vote for this activity

Visible: ☑ Can Attendees see this activity

Create activity

Figure 7. Create Activity form

4.3 How the project went

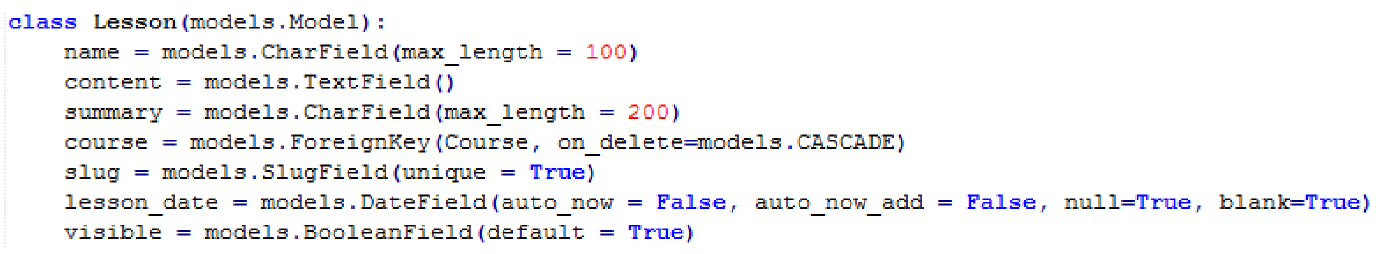
Not being able to do the originally planned eight hours of work a day caused the project to take longer to complete. The work hours it took to complete the project were quite close to the originally planned hours. Getting to know the tools and understanding how they can be used was the only part that took a great deal more time than expected and during the project new ways of doing the required work were learnt.

In the beginning of the project videos were the main source of possible tools to use in the project and learning those tools. Once the tools were chosen their respective wikis were a great source of learning and tips.

5 Results

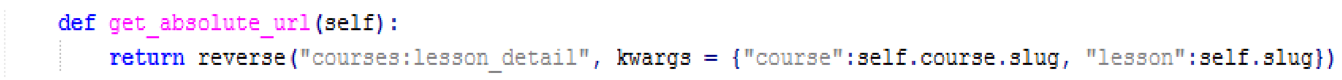
5.1 Project

The applications database is created using a Python file called Models and in it the models for “Courses”, “Lessons”, “Activity’s” and “Assignments” is defined using Django’s model fields. With these fields Django creates the database for the project automatically with the set parameters allowing simple and quick database creation and management. Models can also be given functions and some functions are used by default, like get_absolute_url that returns the URL of the model object. For a snippet of the Lesson model and its functions see Figure 8. and Figure 9.



```
class Lesson(models.Model):
    name = models.CharField(max_length = 100)
    content = models.TextField()
    summary = models.CharField(max_length = 200)
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    slug = models.SlugField(unique = True)
    lesson_date = models.DateField(auto_now = False, auto_now_add = False, null=True, blank=True)
    visible = models.BooleanField(default = True)
```

Figure 8. Lesson model

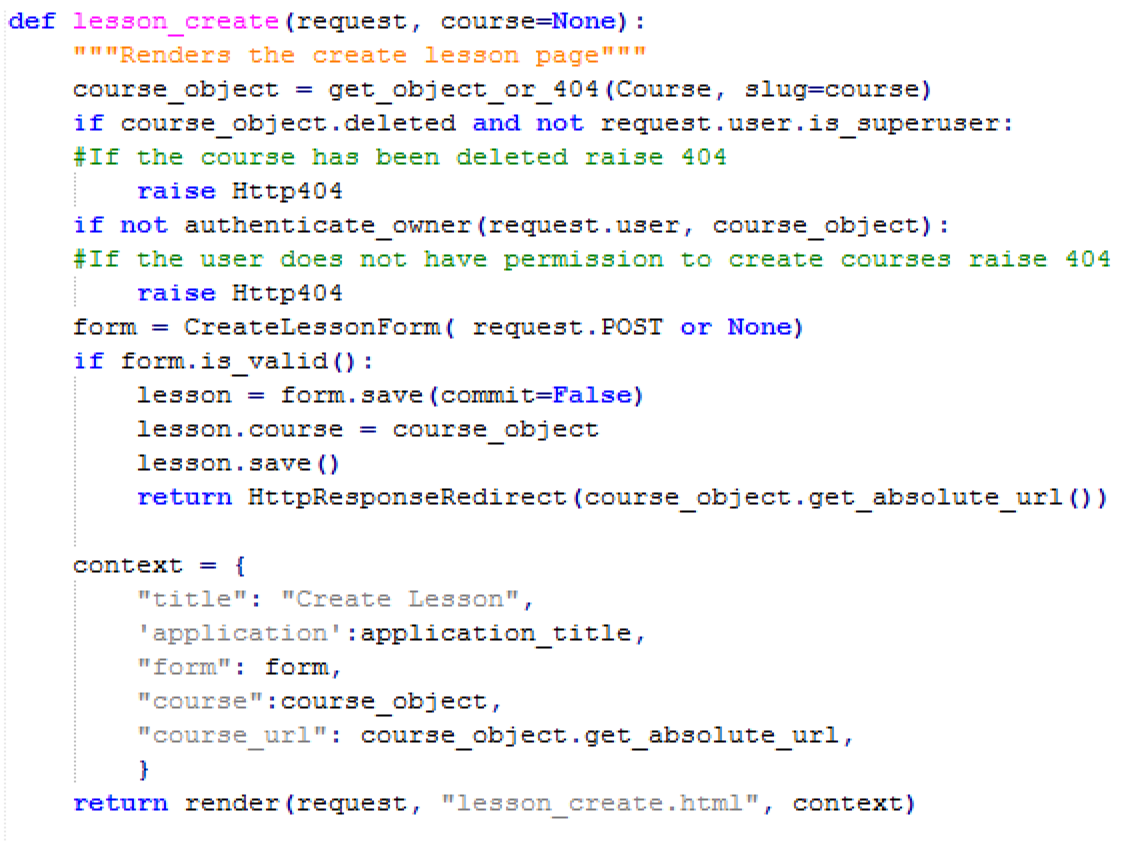


```
    def get_absolute_url(self):
        return reverse("courses:lesson_detail", kwargs = {"course":self.course.slug, "lesson":self.slug})
```

Figure 9. get_absolute_url function for a lesson model

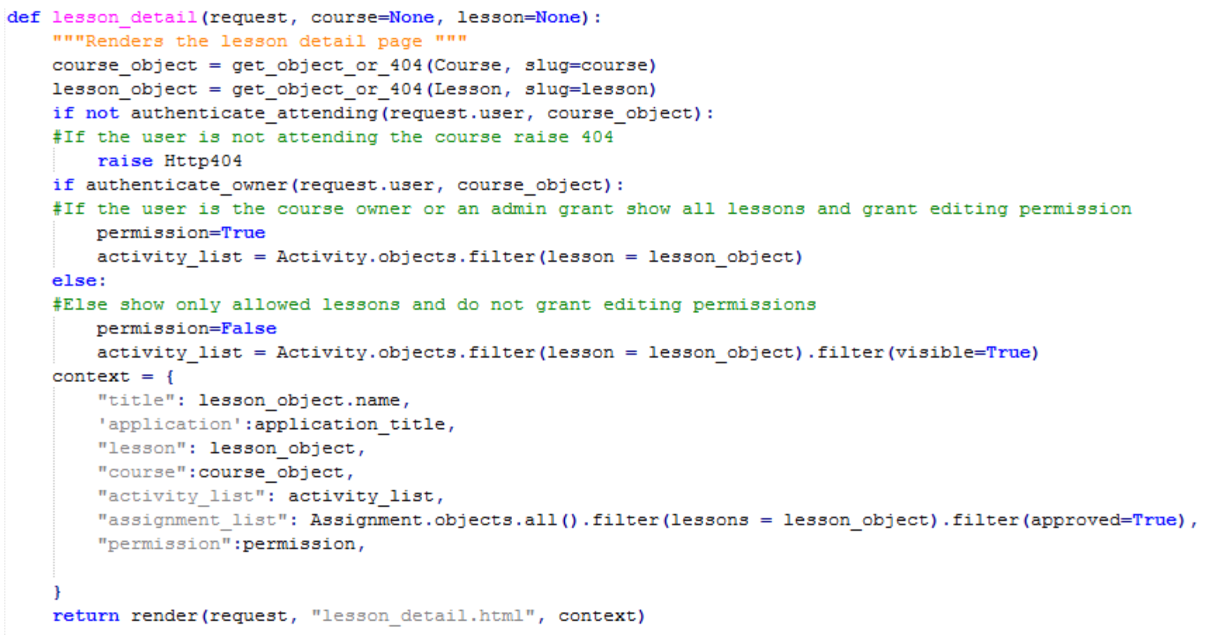
The controller of Django’s MVC model is confusingly called a view and in it the backend of the application is defined. There is plenty of built in functionality that comes with Django helping to speed up development and improves the quality of the software.

Because the models of the application follow the CRUD method each one of them has a separate view to handle creation, reading, updating and deletion of the objects in question, for examples see Figure 10. for creation of the lesson objects, Figure 11. for displaying lessons and Figure 12. for deleting lessons.



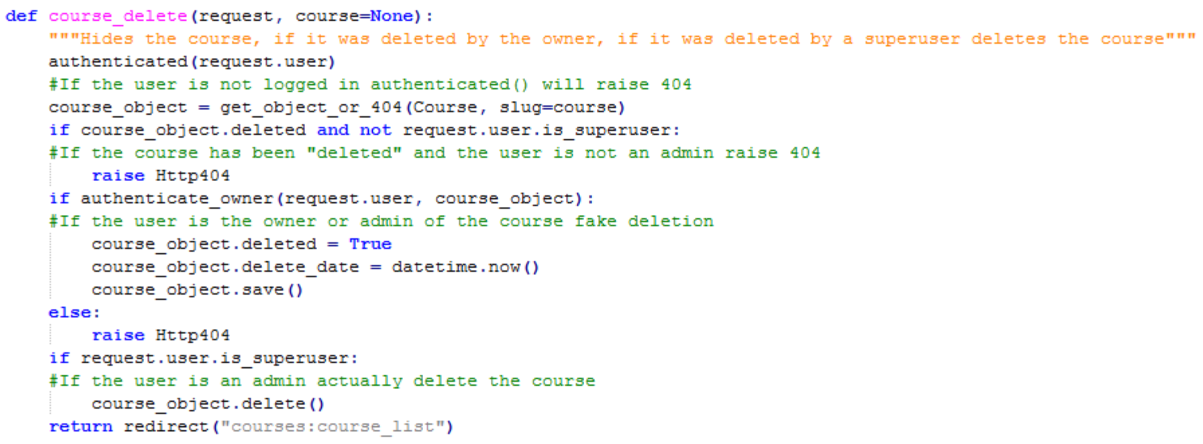
```python
def lesson_create(request, course=None):
    """Renders the create lesson page"""
    course_object = get_object_or_404(Course, slug=course)
    if course_object.deleted and not request.user.is_superuser:
    #If the course has been deleted raise 404
        raise Http404
    if not authenticate_owner(request.user, course_object):
    #If the user does not have permission to create courses raise 404
        raise Http404
    form = CreateLessonForm( request.POST or None)
    if form.is_valid():
        lesson = form.save(commit=False)
        lesson.course = course_object
        lesson.save()
        return HttpResponseRedirect(course_object.get_absolute_url())

    context = {
        "title": "Create Lesson",
        'application':application_title,
        "form": form,
        "course":course_object,
        "course_url": course_object.get_absolute_url,
        }
    return render(request, "lesson_create.html", context)
```

Figure 10. Create lesson view



```python
def lesson_detail(request, course=None, lesson=None):
    """Renders the lesson detail page """
    course_object = get_object_or_404(Course, slug=course)
    lesson_object = get_object_or_404(Lesson, slug=lesson)
    if not authenticate_attending(request.user, course_object):
    #If the user is not attending the course raise 404
        raise Http404
    if authenticate_owner(request.user, course_object):
    #If the user is the course owner or an admin grant show all lessons and grant editing permission
        permission=True
        activity_list = Activity.objects.filter(lesson = lesson_object)
    else:
    #Else show only allowed lessons and do not grant editing permissions
        permission=False
        activity_list = Activity.objects.filter(lesson = lesson_object).filter(visible=True)
    context = {
        "title": lesson_object.name,
        'application':application_title,
        "lesson": lesson_object,
        "course":course_object,
        "activity_list": activity_list,
        "assignment_list": Assignment.objects.all().filter(lessons = lesson_object).filter(approved=True),
        "permission":permission,

    }
    return render(request, "lesson_detail.html", context)
```

Figure 11. Display lesson view



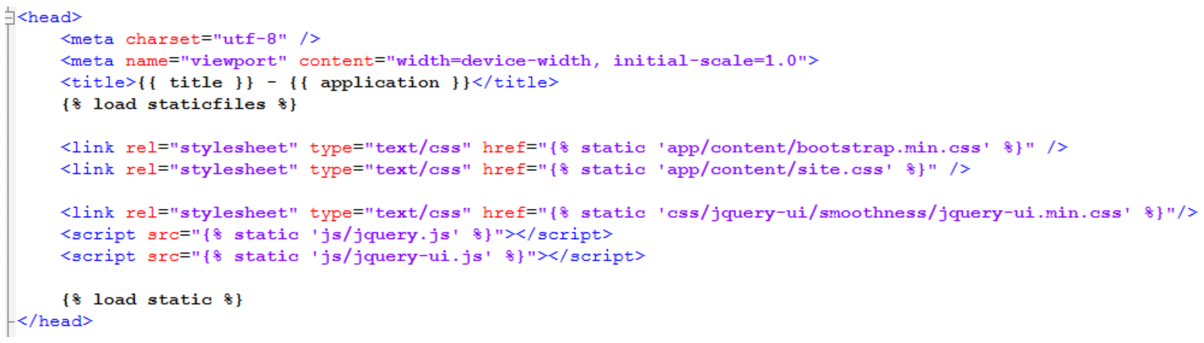
```python
def course_delete(request, course=None):
    """Hides the course, if it was deleted by the owner, if it was deleted by a superuser deletes the course"""
    authenticated(request.user)
    #If the user is not logged in authenticated() will raise 404
    course_object = get_object_or_404(Course, slug=course)
    if course_object.deleted and not request.user.is_superuser:
    #If the course has been "deleted" and the user is not an admin raise 404
        raise Http404
    if authenticate_owner(request.user, course_object):
    #If the user is the owner or admin of the course fake deletion
        course_object.deleted = True
        course_object.delete_date = datetime.now()
        course_object.save()
    else:
        raise Http404
    if request.user.is_superuser:
    #If the user is an admin actually delete the course
        course_object.delete()
    return redirect("courses:course_list")
```

Figure 12. Delete course view

The delete course view is special for it only fakes deleting the course if the teacher of the course deletes it. This is to prevent accidental deletion of courses. Fake deleted courses appear different to admins viewing the course list page. Admin can simply restore these fake deleted courses. All other deletion is permanent in the application.

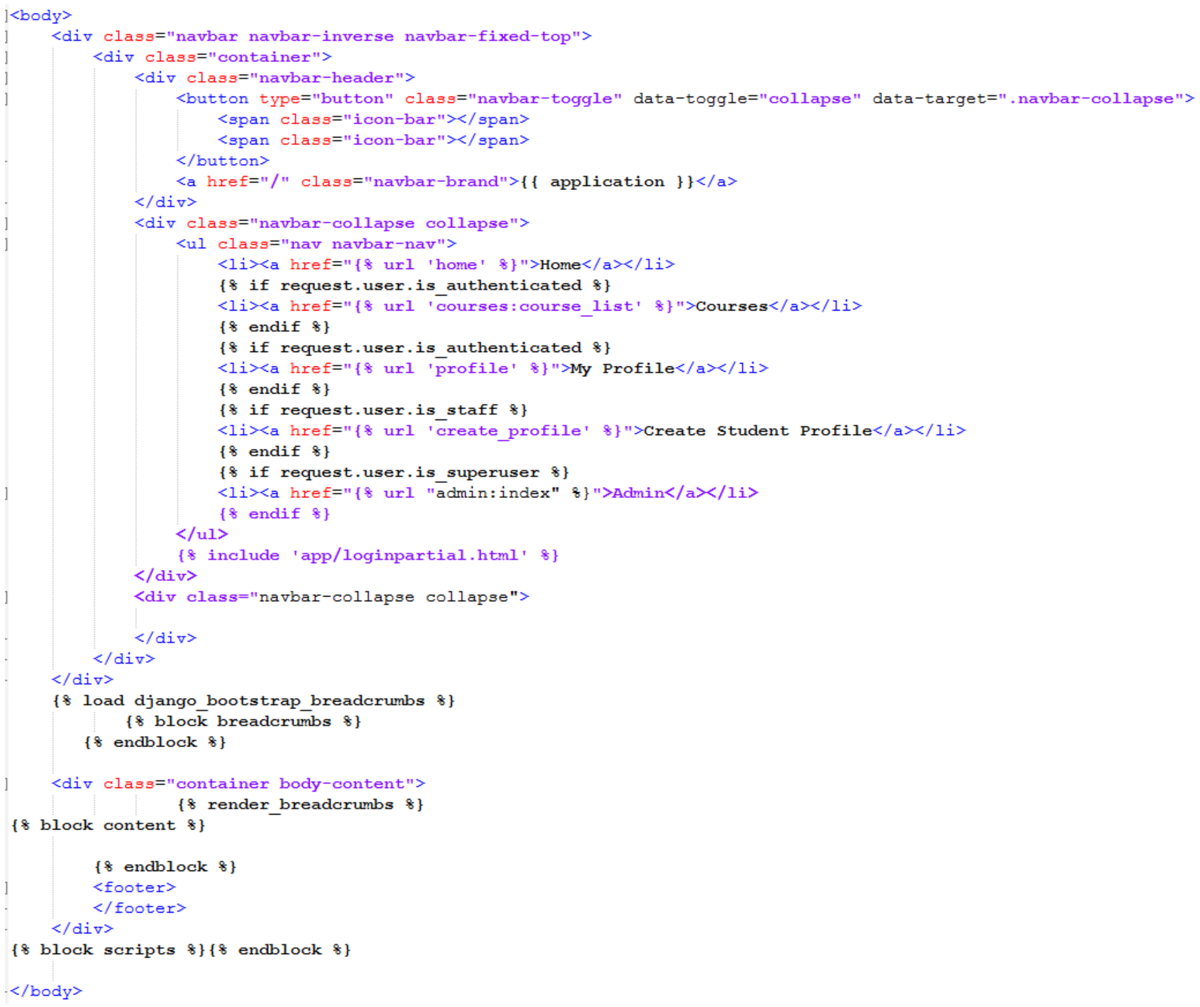
In the views user authorization is also performed by functions that first check if the user has logged in and then compares the HTTP requests logged in user to users who have access to the course. These would generally be either the teacher of the course or students attending the course. Users who have not logged in will always receive a ”404 page not found” error when trying to access pages other than the main page.

Templates are displayed to the user. Within this application this is done with a HTML file. The way the models and the views was setup there is a template for each model and its creation, displaying and updating. There is one Layout HTML that all the other templates extend as a base. The extending templates contain blocks that will be placed into the Layouts empty blocks. The Layout also contains the navigation bar and loads the static files. The Layout can be seen in Figure 13. and Figure 14. Using the Layout more dynamically or having more similar files is possible and would reduce the total amount of HTML files required in the project.



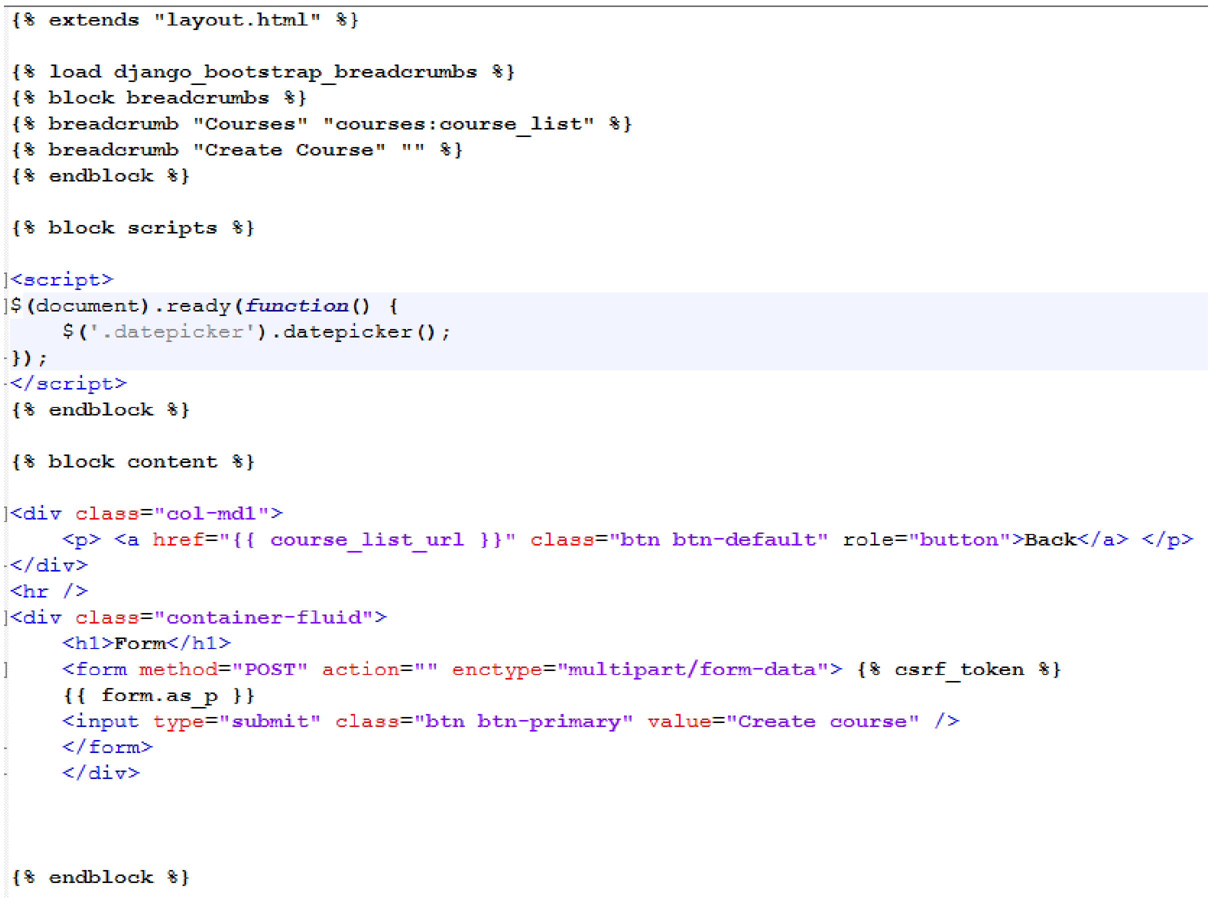
```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ title }} - {{ application }}</title>
    {% load staticfiles %}

    <link rel="stylesheet" type="text/css" href="{% static 'app/content/bootstrap.min.css' %}" />
    <link rel="stylesheet" type="text/css" href="{% static 'app/content/site.css' %}" />

    <link rel="stylesheet" type="text/css" href="{% static 'css/jquery-ui/smoothness/jquery-ui.min.css' %}"/>
    <script src="{% static 'js/jquery.js' %}"></script>
    <script src="{% static 'js/jquery-ui.js' %}"></script>

    {% load static %}
</head>
```

Figure 13. Layout <head>



```
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a href="/" class="navbar-brand">{{ application }}</a>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a href="{% url 'home' %}">Home</a></li>
                    {% if request.user.is_authenticated %}
                    <li><a href="{% url 'courses:course_list' %}">Courses</a></li>
                    {% endif %}
                    {% if request.user.is_authenticated %}
                    <li><a href="{% url 'profile' %}">My Profile</a></li>
                    {% endif %}
                    {% if request.user.is_staff %}
                    <li><a href="{% url 'create_profile' %}">Create Student Profile</a></li>
                    {% endif %}
                    {% if request.user.is_superuser %}
                    <li><a href="{% url "admin:index" %}">Admin</a></li>
                    {% endif %}
                </ul>
                {% include 'app/loginpartial.html' %}
            </div>
            <div class="navbar-collapse collapse">

            </div>
        </div>
    </div>
    {% load django_bootstrap_breadcrumbs %}
            {% block breadcrumbs %}
        {% endblock %}

    <div class="container body-content">
                {% render_breadcrumbs %}
{% block content %}

        {% endblock %}
        <footer>
        </footer>
    </div>
{% block scripts %}{% endblock %}

</body>
```

Figure 14. Layout <body>

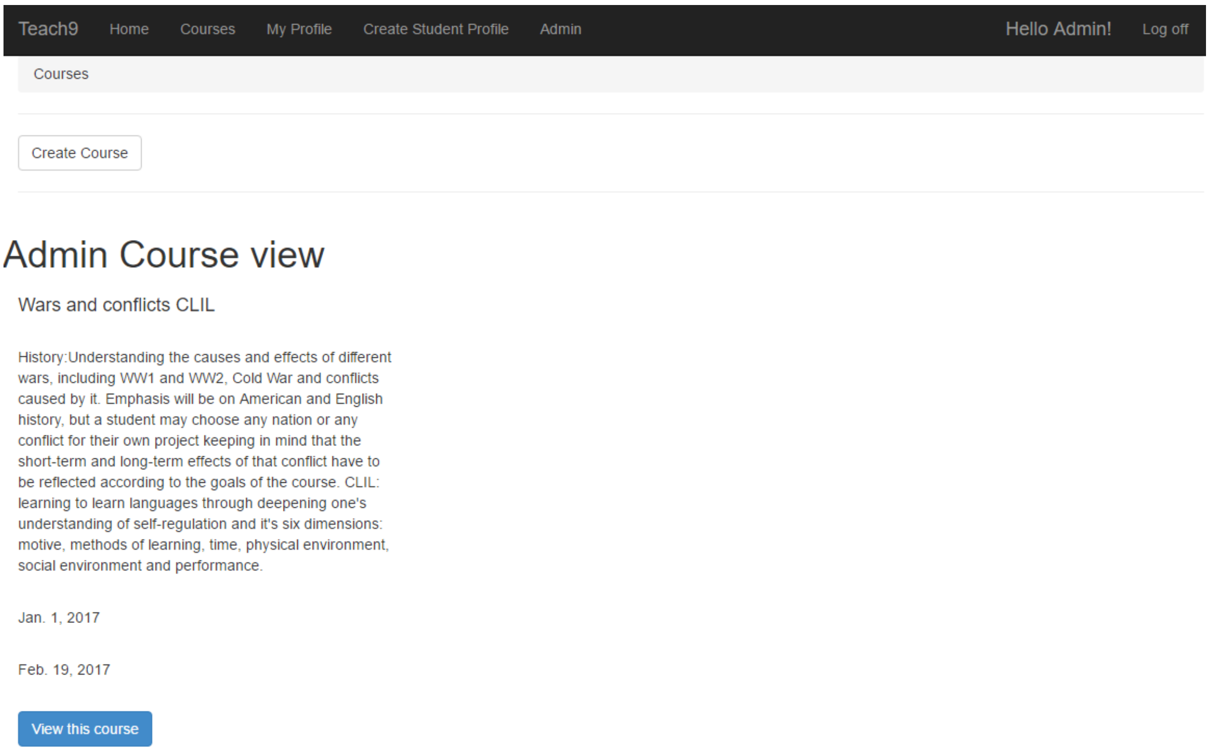


```
{% extends "layout.html" %}

{% load django_bootstrap_breadcrumbs %}
{% block breadcrumbs %}
{% breadcrumb "Courses" "courses:course_list" %}
{% breadcrumb "Create Course" "" %}
{% endblock %}

{% block scripts %}

<script>
$(document).ready(function() {
    $('.datepicker').datepicker();
});
</script>
{% endblock %}

{% block content %}

<div class="col-md1">
    <p> <a href="{{ course_list_url }}" class="btn btn-default" role="button">Back</a> </p>
</div>
<hr />
<div class="container-fluid">
    <h1>Form</h1>
    <form method="POST" action="" enctype="multipart/form-data"> {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" class="btn btn-primary" value="Create course" />
    </form>
    </div>


{% endblock %}
```

Figure 15. Create course template

5.2 An example of a course

An example of a course would be a course of teaching ninth graders, which was the original intention, and thus the name of the project is Teach9. The subject of the course was to be history and the course details would reflect this with information on what part of history the course will be about, e.g. World War 2. The example course will last eight weeks and have three contact lessons each week. The next task in the example was be to create a lesson for each contact period of the course. In addition to the contact lessons there will be additional entries to represent each week of the course. Thus, these week-long entries will have the assignments and homework for each week attached to them. Because most of the information of the courses and lessons is text, teachers can create courses that reflect their teaching methods.



Admin Course view

Wars and conflicts CLIL

History:Understanding the causes and effects of different wars, including WW1 and WW2, Cold War and conflicts caused by it. Emphasis will be on American and English history, but a student may choose any nation or any conflict for their own project keeping in mind that the short-term and long-term effects of that conflict have to be reflected according to the goals of the course. CLIL: learning to learn languages through deepening one's understanding of self-regulation and it's six dimensions: motive, methods of learning, time, physical environment, social environment and performance.

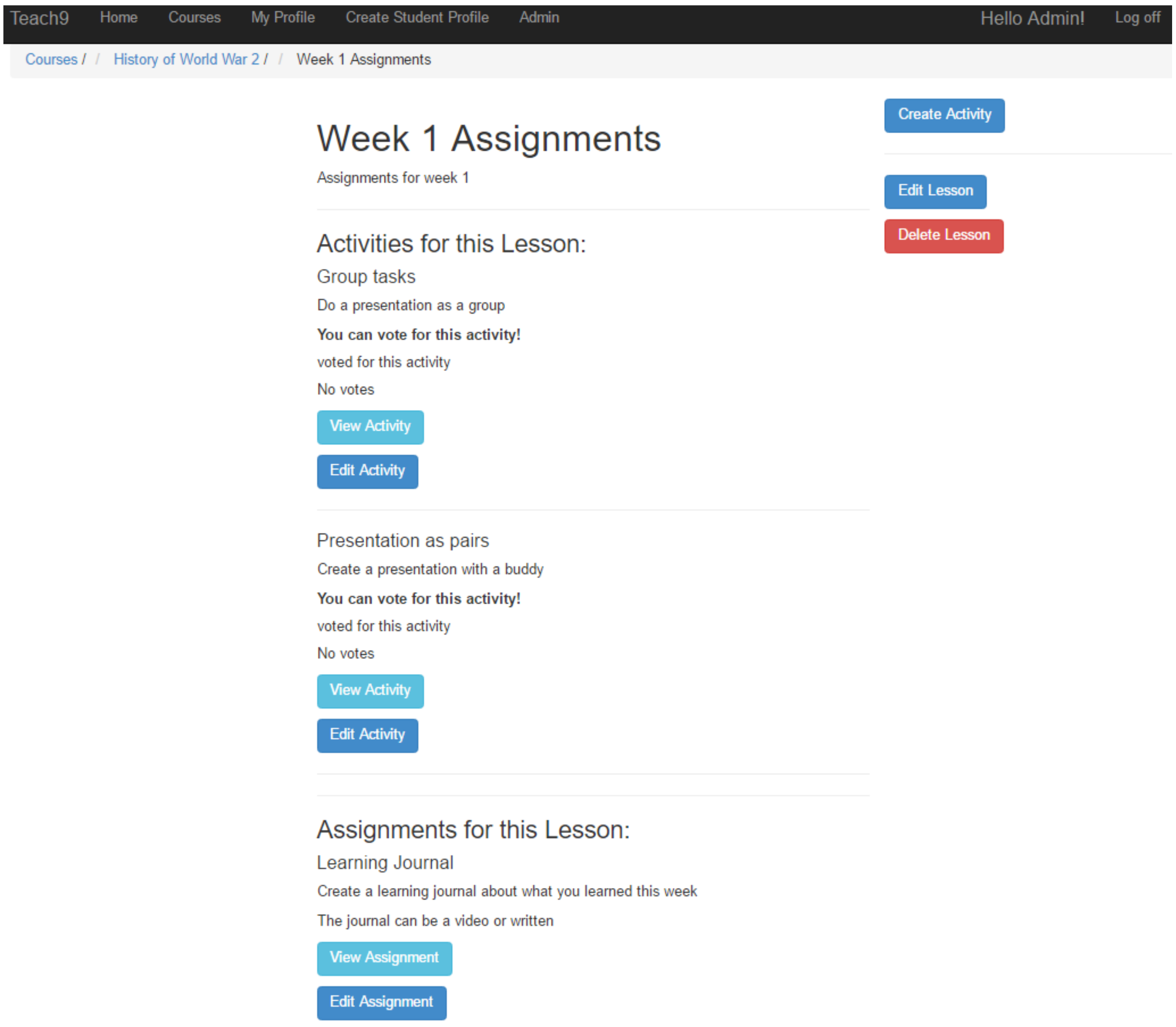
Jan. 1, 2017

Feb. 19, 2017

Figure 16. Admin Course View

Assignments for lessons could be, for example, learning journals where students must write what they have learned each week of the course because assignments can be referred to as a part of many lessons as the teacher sees fit; the teacher must only write one assignment about writing a learning journal and can then link it to each applicable lesson.

Now that the course, the lessons and the assignments for the course are completed the activities for each lesson can be focused on. As mentioned earlier, lessons will be created for each contact period it makes the most sense to have the activities assigned to them. The example lessons will have one or two activities that the students cannot vote on and will reflect what the teacher shall lecture about during that lesson. Some of the lessons will have activities that will consult the students on how they want to learn, for example: doing a group project and presenting it to the rest of the class, doing a project in pairs or some other way of studying that the teacher might come up with. See Figure 5. for an example of what a lesson could look like.



Week 1 Assignments

Assignments for week 1

Activities for this Lesson:

Group tasks

Do a presentation as a group

You can vote for this activity!

voted for this activity

No votes

Presentation as pairs

Create a presentation with a buddy

You can vote for this activity!

voted for this activity

No votes

Assignments for this Lesson:

Learning Journal

Create a learning journal about what you learned this week

The journal can be a video or written

Figure 17. Example of a lesson

5.3 Deployment

Deploying the project to a server to allow hosting the application and grant external access to it was more challenging that initially expected and revealed flaws in the product. Most of the issues came from Django and changing from development to production, since moving from development to production was not high on the priority list and was done at the last minute to allow testing from anywhere with an internet connection for the assigner.

The server chosen to host the application was from a cloud computing company. The OS on the server was the Linux distribution Debian 8.6. Working with Django in a Linux environment was not that much more different than working with it in a Windows environment. After installing PIP the procedure was pretty much the same, only few of the commands were different between Linux and Windows.

To properly deploy using Django the product should be more complete and have better security functions including a custom made 404 error page and handling. Handling the errors and issues that were part of the deployment are outside the scope of the project because part of the requirements Django wants is an external hosting tool and the IP of the hosting server. Separating the application server and the database would be another step in moving to a properly deployed application.

5.4 Successes

Delivering a functioning product was the definite goal and it was achieved: Although the application was deployed at the last moment and not to the best standards, it works and can be upgraded later.

Commenting in the project's code is adequate and is present where it is required. With thorough commenting of the code later development will be easier.

The project was not polished much; however, it still has decent performance and there were no complaints from the testers about implemented features, only about missing features.

5.5 Shortcomings

Learning the tools was more work than expected and caused issues with the flow of the project. The project's security is questionable and using the built-in admin tools to perform administration is less than desirable. The last-minute changes to allow deployment of the application were a hassle and when starting on the next project, even if the goal was not to create a deployable version, researching on what must be done before deployment would be carried out earlier in the development stages.

Using Pythons *args and **kwargs in the project's functions was lacking, however, the built-in functions for Django used them effectively and gave a glimpse on how they can be used and that they are very useful. There is minimal user input error correction, mainly on the dates of the course and lessons. Courses, for example, can end before they start; however, the work required to implement this was more than the value it would bring.

The templates would have been more useful with better ingenuity and smarter construction of the models: there could be less templates per model and CRUD action. Most of Django's functionality could have been improved upon by doing everything again with the insight that was gained during the project.

Not exactly a shortcoming automated testing would have been something that would be interesting to implement and see how it would affect the development of the application.

Removing most of the functionality that was supposed to bring the project uniqueness and value was a major shortcoming.

6 Discussion

6.1 What was learned

This project being my first solo project development a great deal was learned on how much work creating an application takes and how much can go wrong. It was, additionally, the first time creating an application for someone who has never had experience with software development.

Most of the learning on how to create software using Django was gained from inspecting the original functions of Django. It was inspiring and gave something to strive for seeing how software developers with more experience have created a framework for others to create on.

The lacking of exception handling was eye opening and will be a learning focus for future projects. Additionally, one cannot emphasise the importance of having the program validate all data from the user, some of it was left out of the project and it is evident how detrimental this was to the application.

6.2 Future development

Future development of the project will be continued as required to allow the successful integration of the application into the assigner's Master's thesis. The next updates would be aimed at increasing validation of data and improving the help page for the application giving the application easier usability for new users.

The implementation of the "nice to have" functions that were cut because of time restraints is possible if long term updates are carried out. From the "nice to have" functionalities that were cut the most important one that would bring the work additional uniqueness is improving the communications between the students and teachers with the forum.

Working on the overall look of the project, adding pagination, adding options to filter through the different fields, a custom made 404 error page that redirects to the main page are also things that could be added. In its current state the application does what is required of it, which is good and can be worked on to implement additional features.

6.3 Conclusion

"The idea is very good but it's a shame that some of the planned features were not deployed." (Toni, Pudas. 2016)

The object of the project was to create a functioning tool for the assigner to allow a way to create and display teaching material and this was achieved. The application will add value to the assigner's Master's thesis and was a good learning experience to all parties involved.

"It is a good prototype that can hopefully inspire similar projects in the future." (Sirkku, Ukkonen. 2016)

Teachers can create their courses with adequate flexibility. There are no obvious problems with the application; however, there are "nice to have" functions missing that would have to be implemented to give a reason to use the application in a school, for example, filtering of users when adding attendees to a course. The lack of a filtering option while adding attendees to a course will not be an issue while the number of users in the application is low.

The goal of the project was never to create an application for actual production. Thus, corners were cut to gain as many unique features as possible. Cutting corners is always a terrible idea in software development; however, in this project it was deemed worth it because the value lost to this was small and more value not gained.

The concept of the project is sound and although the project did not achieve much the exploration of giving students the tools to communicate with each other and their teachers is something worth doing.

"The project has some good ideas and should be developed further - especially the teacher - student interaction. That is something lacking in other applications." (John, Pearce. 2016)

References

Andrade, M. S. and N. W. Evans P.2013. Principles and practices for response in second language writing. Developing self-regulated learners. New York, NY: Routledge.

More on Django fields. Accessed on 8 November 2016. Retrieved from https://docs.djangoproject.com/en/1.10/ref/forms/fields

Microsoft Visual Studio. Accessed on 11 November 2016. Retrieved from https://www.visualstudio.com/vs/ide/

Information about Django. Accessed on 8 November 2016. Retrieved from https://docs.djangoproject.com/en/1.10/faq/

Django project. Accessed on 28 November 2016. Retrieved from https://www.djangoproject.com/

General Python FAQ. Accessed on 28 November 2016. Retrieved from https://docs.python.org/3/faq/general.html

Python documentation on SQLite. Accessed on 28 November 2016. Retrieved from https://docs.python.org/2/library/sqlite3.html

Pythons speed. Accessed on 1 December 2016. Retrieved from

https://www.quora.com/Why-is-Python-so-popular-despite-being-so-slow

Toni, Pudas. 2016. Student at JAMK.

Sirkku, Ukkonen. 2016. Assigner and student at University of Jyväskylä.

John, Pearce. 2016. Teacher at SeAMK.