**Chapter 4**

**Design**

This chapter outlines the design decisions for this Level 4 project. As it was previously described, WEAVE is based on top of the standalone Java application IWE. This affects to a great extent these decisions. Following the well-established software engineering principle of reusability, guidance for the design of WEAVE was to reuse any good aspects of IWE’s design while improving its weaknesses.

**4.1. Storage of Data**

The scope of this Level 4 project does not include the creation of a web based author interface. This means that the design of WEAVE must ensure an easy and efficient way for importing and storing the existing worked examples. In addition, WEAVE must support uploading updates to existing examples. As described in **Chapter 2**, the examples are stored in the form of XML files. To remind yourself of the structure of these files, please refer to **Figure 2.2**.

During the exploration process of IWE, a weakness of storing the information about the examples in the XML files used by IWE was identified. Since these files are easy to access and modify by the authors of worked examples, and this is typically much easier than modifying the worked examples using the author interface, one may be tempted to make changes to the examples manually. However, references to some objects may be present in more than one of these XML files. For example, in the Documents.xml file fragments are stored as individual elements identified by a fragment id. When the steps for the example are defined in the Processes.xml file, each fragment is referred to with its id. A problem with storing information about the examples in files is that if an element is modified in one of the files, consistency about this element must be ensured. It is trivial having to find the same feature across multiple files and in the end consistency and validity are not guaranteed.

Django provides object relational model (ORM) database functionality. This is an efficient way to manage objects and their relationships and is a preferred method for managing the data for the worked examples because it guarantees consistency. The relationships between different objects are expressed via foreign keys. The entity relationship diagram is presented on Figure 4.1. Using a database adds a level of reliability that the data is valid because internal integrity checks are made before saving an object to the database. Further advantage is that the Django administrator interface allows an easy means of exploring and modifying the examples in a consistent and safe manner.

**4.2. Authentication**

A huge discussion point was how to authenticate teachers and their students due to the privacy and ethical issues discussed in **Section 3.3.2**.

One possibility was students to create their own accounts and give their usernames to the teacher. However, this approach could potentially result in various complications.

* Students would need to be explicitly instructed that their accounts should not reveal their true identity.
* Students would not have any benefit of having their own accounts and only the teacher will be the one who would use their usernames for something meaningful, i.e. to check their progress.
* This approach solves identification at an individual level but each student needs to belong to a group as well.

A second option was considered, in which the teacher would create an account for each of their classes. For this account, they would need to create usernames for their students. Keeping in mind that a teacher would often have more than one classes and that each class consists of twenty to thirty students, the following problems arise:

* The teacher would need to create a lot of accounts and this could be a trivial and time consuming task.
* The teacher would need to ensure they will be able to match each of their students to their id since they must not use any names due to the privacy issues mentioned above.

To go around the privacy issues and the problems with the options described above, a third approach based on the general idea of the second one was adopted. Teachers will create their own accounts. In these accounts, they will be able to create groups for their classes. On creation of the group, the teacher needs to specify the number of the students in this class. WEAVE will then generate random ids for these students. An id consists of two random letters followed by a single digit. While being short enough to be easily remembered, such an id ensures that the privacy of students is protected due to its random nature. Furthermore, the number of possible combinations for all student ids is large enough so that it will be highly unlikely that students will be able to “guess” one of their classmate’s student id and work with the examples on their behalf.

Talking to a lead teacher, Peter Donaldson, who is part of the PLAN C project, a potential inconvenience of this approach was identified. Using WEAVE for longer periods than one academic year could lead to a significant increase of the groups. In addition to the growth of the number of groups, some teachers might prefer to use the same name for their classes across years. To resolve these issues, a further classification of groups by the academic year the group belongs to was adopted.

Mr. Donaldson was generally happy with the idea that teachers select the number of students for each group at the creation of this group. However, he pointed out that it is possible for a student to arrive in a class at a later stage than the beginning of the academic year. Using the selected approach would have required that the teacher creates a new group just to add one student only. This could be very problematic, because data about the same students would be spread across two different groups and most of the students will be given two student ids which could become really confusing for both teachers and students. To avoid these problems, the option for teachers to update a group was added to the design decisions for the authentication part. In addition, groups can also be deleted in case of creation of unneeded groups.

**4.3. Presentation of Data**

The main goal of the teacher interface is to present to the teacher data associated with a particular group or student. Three different types of data are recorded from the student interface:

* Time at each step.
* The direction of the transition to each step i.e. is the student going backwards or forwards to a step.
* Answers to questions.

Careful consideration was needed to reach to a solution that would visualise this data in a way which would enable the teacher to comprehend it easily and encourage further analysis. In addition, the ability the teacher to be able to view data both at a class and at an individual level further influenced the design decisions.

To start with, it has been decided that presenting the data in the form of graphs would be hugely beneficial to teachers as they would be able to identify patterns and any exceptional events for a particular worked example. These graphs should reveal information about the performance of the whole class as well as of individual students at each step of the example keeping in mind that some steps have questions. For this purpose, five different types of graphs were decided upon:

* Average Time. This graph would show the average performance of the students in the whole class on a particular example. Different steps will be represented on the x-axis of the graph by their step number. The average time spent on each step will be shown by the y-axis. Since the x-axis consists of the step number, this is not very informative to the teacher because they would need to look at the actual example to remind themselves about the context of the step. To avoid the need for that, hovering over the point representing the step will show the beginning of the explanation. Clicking on that point will open a window showing the whole text of the explanation, the average time spent on that step and how many students made a backwards transition to the step. As mentioned above, some steps contain a question, rather than an explanation. Such steps will be identified by a question mark in front of the step number on the axis label. Instead of showing an explanation on mouse hovering, a message encouraging teachers to click in order to see students’ answers is shown. Clicking on the point for that step will show the question, all the possible answers and how many and which pupils selected each option.
* Student Time. This graph is conceptually the same as the Average Time graph. Instead of showing information about the whole class, however, it shows the total amount of time spent at each step by a selected student.
* Student Answers. This graph shows the options for a selected question and the number of students who chose each option. Hovering with the mouse over each bar shows the list of students who selected the answer represented by this bar.
* Class Steps. This graph shows information about the time spent at a chosen step of an example by the students of a class. It is in the form of a bar chart where each bar represents a student’s attempt. This means that there might be more than one bar for each student if they have attempted the selected step more than once- each bar revealing information about the time spent by that student at a particular attempt of the step.
* Class Summary. This is a table showing summary information about the total time spent by a student at a particular example, how many times they returned to previous steps and the last step they reached. This would show the teacher how much effort did the student put in each example, how many problematic or unclear steps they encountered as well as whether they completed the example or which step they gave up on.

**4.4. Architecture**

The architecture of Weave consists of three distinct tiers as visualised on Figure 4.2.

* Presentation tier. This is the top level of the overall architecture also known as the client side web interface. It defines the appearance of the website by rendering HTML and CSS and provides means for users to interact with the application. The clients are in the form of web browsers. On every interaction, they send requests to the server in the form of Ajax GET or POST requests to ensure asynchronous communication between the client and the server.
* Django Middleware. This tier consists of two distinct components.
  + The first component of this tier serves as a communication point between the client and the database. In this tier the requests from the client are parsed and translated into ORM requests- a language understandable by the database. These requests are passed forward to get or store the information in the request from/in the database. After the backend generates the response, the middleware is responsible for passing it back to the presentation tier in the form of Http response.
  + The second component is the connection point between IWE and WEAVE. This is where the translation of the XML elements storing the examples in the form of ORM objects is taking place. An XML parser is used to separate the elements created by the author interface. The relationships between these elements are established and the database is populated with the objects defined by the parser.
* Data layer. This tier represents the database in which all the information used or generated by WEAVE is stored in the form of objects. On GET requests the backend tier responds with an object meeting the criteria specified in the ORM request. On POST requests, the database creates a new object with the features specified in the request and stores it in the database.

**4.5. User interface**

As already mentioned, WEAVE is intended for three distinct groups of users, each with their different needs. However, for the purpose of this Level 4 project, only two of these groups will influence the user interface. Addressing the needs of authors is beyond the scope of this project. To add examples to the system, they need to use the IWE author interface and contact the administrators of WEAVE with a request to add the newly created example to the application. This is why the section on the user interface is split into two subsections only, which describe the user interfaces for students and for teachers.

**4.5.1. Student User Interface**

A core purpose of this Level 4 project is to translate the student part of IWE into a more easily deployable online version. Careful consideration about the layout of the student interface of IWE is evident. The evaluation of IWE proved that the current interface is well accepted by students. A screenshot of this interface is presented on **Figure 2.3**. Generally, it has been decided to take advantage of Dr. Song’s findings and to reuse a very similar interface.

**4.5.1.1. Home page**

The need for some additional features of this interface arises due to the fact that teachers need to be able to identify their students in order to monitor their interaction with the examples. To ensure that students are using the system in the intended way and to encourage them to use the details provided by their teacher, it has been decided that the examples will be hidden to them until they enter some authentication information or identify themselves as anonymous users.

Once the student has identified themselves, the area prompting the user for details is exchanged for the list of worked examples. Due to the fact that the system is required to accommodate examples created by many teachers across the UK, a filtering by the name of the worked examples functionality has been provided. In addition, the option to select a worked example appears in the top navigation bar. Having the same feature twice might seem repetitive at first. However, more careful consideration justifies this design decision. Substituting the detail specification area with the list of examples guides the user that they need to choose an example to work on. Having the same list of examples in the toolbar area contributes for faster navigation between examples.

In order to be able to exploit the worked examples viewer in an optimal way and to familiarise students with it, a tutorial appears on the main page. The idea for having a tutorial was borrowed from IWE. However, the way this tutorial was constructed there was identified as potentially ineffective at communicating all the information the pupil needs to know before working on examples due to the fact that it contains a lot of text which may discourage some of the pupils to read it. Furthermore, even if they read the tutorial, they may not understand what is referred to in the text because they may have not seen the worked examples viewer and its features in advance. A different approach was chosen for the tutorial on WEAVE. It is split into different steps describing an individual feature using minimal text and a screenshot of the feature.

**4.5.1.2. Page for viewing an example**

The page for viewing a worked example is very similar to the one used for the IWE student interface. However, due to constraints imposed by the size of the screens in schools, the design needed to be adjusted accordingly. The area for selecting an example (referred to as **Element 1** in **Figure 2.3**) is placed on the navigation bar with all the examples appearing in a drop down menu on request. This saves a significant portion of the screen which can be used for the problem specification instead. Another space consuming element is the bar showing the current step. In WEAVE this information is shown as a part of the explanation instead.

**4.5.2. Teacher Interface**

**4.5.2.1. Home page**

**4.5.2.1.1. Non-logged teachers**

The purpose of the teacher interface requires the teacher to be logged in. Therefore, on the first visit of the page the teachers are presented with a register and log in areas only.

**4.5.2.1.2. Logged Teachers**

The discussion on the design decisions on the authentication to WEAVE and the way usage data is presented to the teachers identifies the main sections of the interface for the logged in teacher. Options for all the activities a teacher can undertake via the teacher interface are present on their home page to avoid the need for transitions between different pages and to simplify navigation of the website. The main page is split into three areas.

* Area for registering, updating and deleting a group. These three options are provided in the same area on the screen. When the teacher selects the desired option, the elements for this area change accordingly. For example, when the user wants to create a group, they need to enter the group name and the number of students for that group. On update or deletion of a group, on the other hand, they select the group name from a dropdown list. The list of existing groups is shown to remind which group names are unavailable to this teacher. The textbox for entering the number of students accepts integer input only for error prevention purposes. On the submission of the request to create/update/delete a group, a message confirming the status of the action is show.
* View Group area enabling the teacher to select a group for which to view the student ids. Again, for simplicity and error prevention, the teacher selects the group via a dropdown list rather than typing its name.
* View Statistics icon which navigates the teacher to the statistics page.

**4.5.2.2. View Group Page**

Teachers are able to see the student ids for a class in the View Group page. This information is provided in a table form with columns for the student name and two identical columns with the student id. Teachers will be advised to print this group sheet and fill in the names of their students by hand. This would avoid any potential problems with storing identification information in the system. Teachers will also be encouraged to cut one of the columns for the student ids and hand them privately to each student.

**4.5.2.3. View Statistics Page**

Most of the design decisions for the different types of graphs are explained in **Section 4.3.** above. The teacher needs to select the particular group and the type of data they are interested in. If there is no data for that selection or the selection is invalid, an appropriate message appears on the screen. Otherwise, a graph is shown. This graph is downloadable to enable saving the data for statistics at different points in time and could be used for comparison by the teacher.

**4.6. Site Map**

The site map for this application is fairly straightforward. It is aimed to reduce the need for unnecessary navigation through pages. It is described how this is achieved in the section discussing the user interface above. The site maps for the two interfaces is shown on Figure X.

The following chapter will describe how the design decisions for WEAVE were implemented.