DIM3 Web Application Report

Ross Eric Barnie,
Craig Cuthbertson,
Ross Taylor
Just. No.
DIM3
0901758, 1002386, 1002858
0901758b@student.gla.ac.uk
1002386c@student.gla.ac.uk
1002858t@student.gla.ac.uk

ABSTRACT

Provide a concise summary of the design of the application

1. AIM OF APPLICATION

- What is the purpose of the application?
- Eg. The application is an academic search engine called AcaSe and is it is based upon the PuppyIR Framework[**?**], which has been used to construct other such services[**?**, **?**]. The main purpose of this web application is to provide a customized interface to services such as Google Scholar and MS Academic Search.
- What are the assumptions about the aims and objectives?
- Describe the design goals and objectives of the application.
- What are the constraints of the project?
- Functionality List: i.e. what is the required and desired functionality?
- Reflective Questions:
- Is the scope of the application appropriate?
- Are the design goals realistic/achievable?
- How complex is the application?
- Is distribution across the web appropriate?

2. CLIENT INTERFACE

- Draw a wireframe of the user interface
- this may require several wireframes depending on the complexity of the application and the interfaces
- Describe the user interface.

- i.e. Label key input and output components: describe them.
- Provide a Walkthrough and explain the user interactions with application.
- i.e. use cases
- Describe the interactions associated with the dynamic components on the user interface.
- What calls are required to dynamically update the data on the client side?
- How does the user interface help the user achieve their goal, or complete their task?
- Is the user interface intuitive, appealing, usable, etc?
- What technologies are used on the client side?
- What are the reasons for your choices? i.e. what is the advantages and disadvantages of using this technology?
- What other options are there?

3. APPLICATION ARCHITECTURE

We used the N-tier diagram shown in figure 1 which shows the client (Web Browser), middleware (Django), the database and the external service (Youtube API), to design our architecture.

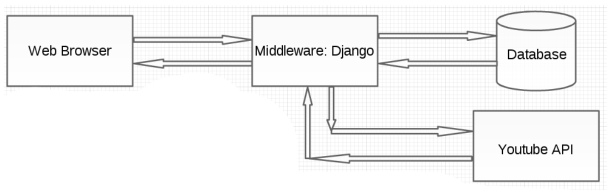

Figure 1: N-Tier Diagram

As can be inferred from the diagram, the clients sends requests for data to the middleware, this then retrieves the appropriate data from the database or the external services as required and the data is sent back through the middleware to the client.

Typically a request follows a structure similar to the following: the client makes a request for a web page; the middleware retrieves this request via specified url patterns which call particular view methods for the app; this view will then make calls to the database to retrieve appropriate data; the data will be taken by the view and then passed back to the web browser via a http response according to the template associated with the view; the client receives this and displays the information.

In order to design the data models for our application we created an Entity-Relationship diagram based on Chen notation [1]. This has been reconstructed and can be seen in figure 2.

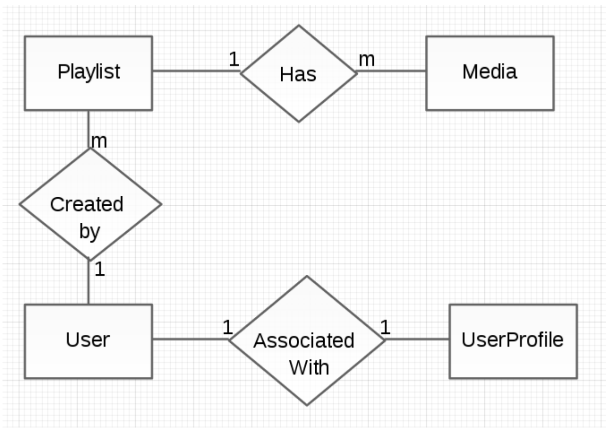

Figure 2: Entity-Relationship Diagram

- Playlist
  - Title: name of this playlist
  - Creator: reference to the User who created this playlist
  - Views: a count of the number of times this playlist has been viewed
  - Url: the title of this playlist encoded for use in a url
- Media
  - Playlist: reference to the playlist this media is associated with
  - Name: name of this media
  - Url: source url of this media
  - Source: the domain of this media eg. YouTube
  - Thumbnail: thumbnail image to be used for this media
- User
  - (default User model from Django, the following fields are the specific fields taken from this model)
  - Username
  - Password
  - Email
- UserProfile
  - User: reference to User assosciated with this UserProfile
  - Picture: image to be used as a profile picture

A key design element in consideration from the start of development was separation of concerns. Using many technologies (Django, Python, JavaScript, HTML, CSS etc.) it was paramount that each of these be kept as seperate as possible so that the source could remain readable and maintainable. Thanks to Django using the DRY (Don't Repeat Yourself) Principle [2] much of the separation of concerns was handled simply by using Django. However within this we set a static directory for the cascading stylesheets (CSS) and for images to be used for the thumbnail of the UserProfile data model. This kept the style of the HTML markup away from the markup itself and away from any javascript within that markup.

4. MESSAGE PARSING

- On the architecture diagram, Identify and label the main messages that will be parsed through the application.
- or alternatively (and preferably) include sequence diagrams to denote the sequence of communications parse between clients and servers.
- Describe the messages that are parsed back and forth through the application.
- For the main transactions - describe the payload of the messages
- i.e. What are the contents of the messages? i.e. include sample XML, XHTML, JSON, etc of one or two messages.
- What is the format of the messages?
- Why this format?
- What other formats could be used, what are the advantages and disadvantages of these other formats?

5. IMPLEMENTATION NOTES

- Views - What are the main views that you have implemented and what do they do?
- URL Mapping Schema - what is your URL mapping and schema?
- External Services - what external services does your application include and what handlers did you include?
- Functionality Checklist (which functionality is completed)
- Known Issues (what kind of works, what kind of errors to do you get)
- What technologies have been used and are required for the application. Include a list or table of all the technologies, standards, and protocols that will be required.

6. REFLECTIVE SUMMARY

For the Implementation Report Only:

- What have you learnt through the process of development?
- How did the application of frameworks help or hinder your progress?
- What problems did you encounter?
- What were your major achievements?

7. SUMMARY AND FUTURE WORK

- Summary of application and its current state.
- Include a list or table of all the technologies, standards, and protocols that will be required.
- What are the limitations?
- Plans for future development

8. ACKNOWLEDGEMENTS

Our thanks to the lecturers and demonstrators for their comments and suggestions. We would also like to thank the industry panel members for their comments and for taking the time to witness our demonstration.

9. REFERENCES

[1] P. P.-S. Chen. The entity-relationship modeltoward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, Mar. 1976.

[2] Cunningham and Cunningham, Inc. Dry principle. `http://c2.com/cgi/wiki?DontRepeatYourself`, 2013.