Berserker Chat: Specification, Design and Implementation Report

Dan Tomosoiu, Tony Lau, Hector Grebbell
Berserker
DIM3
Student numbers
{1102486T, 1102266L, 1007414G}@student.gla.ac.uk

ABSTRACT

This report details the design and implementation of Berserker Chat – a web-based instant messenger and file sharing application which allows many users to collaboratively chat and share files. Berserker Chat is built using the Django Web Framework [1] which is written in the Python programming language.

1. AIM OF APPLICATION

The main aim of the application is to provide a service which allows users to communicate with each other in a real-time chat environment, not dissimilar to other chat applications which the user may have used before.

1.1 Goals and Objectives

The main goals and objectives are listed below:

1. Anonymous use of the application to chat with others in public chat rooms
2. Ability for a logged-in user to have a private chat with another logged-in user
3. Public chat rooms based on category/interests, e.g. public chat room for users interested in photography
4. Creation of public chat rooms based on interest which allow other people to find the created room, join and chat
5. Ability for file sharing between users – files are uploaded and can be downloaded by other users

1.2 Functionality

More specifically, the desired functionality of the web application is as follows:

1. Allow user to login and logout
2. Start a public chat
3. Start a private chat
4. Choose a chat by category
5. Allow user to filter chats specifically by searching for a category
6. A display of the recent chats that the user has been involved in
7. A file sharing interface in the chat room for users to share files with each other

DIM3 2013, Glasgow, UK
ACM X-XXXXX-XX-X/XX/XX.

1.3 Assumptions

- It is assumed that the application is to be used by adults and as such, no filtering of inappropriate language before they are displayed to other users in the chat rooms has been undertaken.
- The application will be displayed to the user in English.

1.4 Constraints

- The Django web framework must be used to build the application
- The application must be completed within the time allotted for the DIM3 course
- The application is to be designed and implemented by 3 people.

1.5 Reflections on scope and design goals

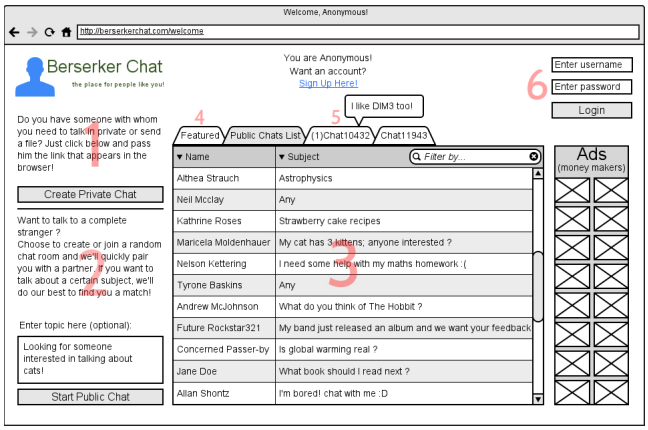
This application has considerable complexity, particularly in the methods which have to be used to achieve as close to instantaneous display of messages as they are sent by the user. The categorisation of chat rooms, adds to the complexity, as does the file sharing capability.

The team feels that the design goals are realistic for the most part. The functionality for users to chat to each other in different chat rooms is realistic and achievable. The only part which may not be implemented is the file sharing capability. The actual functionality implemented is discussed in Section 5.

This type of application is wholly suitable for distribution across the web. By distributing it over the web, anyone with access to an internet browser can use it to communicate with other people across the world. Furthermore, the categorisation of chat rooms allows people to talk to other like-minded people with similar interests.

2. CLIENT INTERFACE

The wireframe of the user interface of the main screen is shown below:



1. This allows the user to create a private chat. When the 'create private chat' button is clicked, a link appears in the browser which the user can click to redirect them to the correct chat room. The private chat link can be passed to others: only those with the link can chat in that specific chat room.

2. This allows the user to start a public chat. When the 'start public chat' button is clicked the user is redirected to a random public chat room. If the user specifies a topic, then the user is redirected to a room for that topic, if it exists. If it does not exist, the room for the topic is created and the user can invite others to join, or wait for other people with the same interest to join.

3. This shows the list of public chats and the name of the person who started it. Clicking on a particular chat subject takes the user into that chatroom.

4. The featured tab shows the chats which have had the most users and are therefore very popular. This allows the user to see what the main topics of chat are.

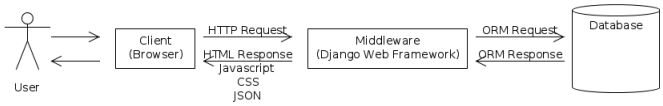
5. A tab shows the chats the user is currently participating in with a pop-up of the most recently received message.

6. Allows the user to login and logout of the site

- 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

The application is designed to use a 3-tier architecture as illustrated in the diagram below. Using this type of architecture allows a separation of concerns between the display of the content on the client browser, the web application framework, and the backend.



The 3-tier architecture consists of:

Client
The client (user) interacts with the Berserker Chat application through a web browser

Middleware
The middleware is where the Django web framework is positioned. It essentially glues together the client and the backend, providing views based on data contained within the database and also updating the database based on the interactions of the user at the browser interface level.

Database
The database stores information about the models which can be requested by the middleware by an Object Relational Model request. The database is implemented using SQLite3.

- N-Tier Architecture Diagram
- i.e. data flow diagram between the interface/client, middle ware, and backend services/data repos
- Describe the data model i.e. what data needs to be stored or persisted by the application?
- What are the relationships within the data model.
- i.e. use ER diagram and explain.
- Describe the backend services used (if any).
- Reflective Questions:
- How have you ensured that there is a separation of concerns?

- What other technology could have been used instead of django?

- What are the advantages of using a Web Application Framework over other technology?

- And, what are the disadvantages?

3.1 Reflections on Application Architecture

Separation of concerns is extremely important to ensure a consistent structure of the application and to ensure maintainability. It also allows development to happen in parallel, e.g. someone could be working on the HTML and CSS of a page, and someone could be working on the database backend. The separation of concerns in this application has been achieved by using the Model, View, Template pattern in Django.

This allows the display of the application on the user's browser to be separated from the business logic of the application. The view is responsible for mapping a URL to the data which should be displayed. The template controls how the data identified by a particular view is displayed to the user. The model is essentially a description or definition of the data that the application stores in a database. This maps easily to a relational database model. The view interacts with the model to obtain information for display using the template.

Other web technologies/frameworks could have been used instead of Django with some gaining a lot of traction and support from the web development community. Some of these are: Symphony, Ruby on Rails and Cakewalk.

Advantages and Disavantages of using Web Application Frameworks (WAFs)

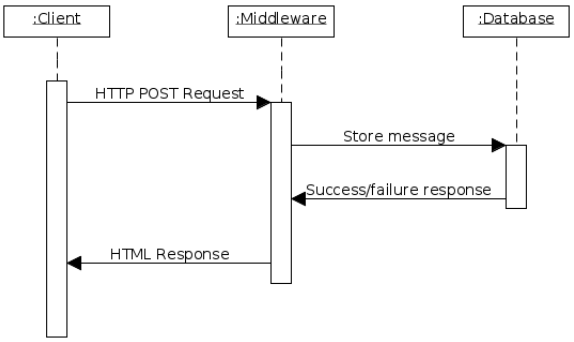
- Web application frameworks can decrease the time it takes to develop an application but also introduces the possibiity of a high learning curve to learn how to effectively and efficiently use the framework to enhance web development.

- Many WAFs are built upon some underlying design patterns, e.g. Django effectively uses the Model-View-Controller (MVC) pattern. This encourages the developer to adhere to software engineering best practices, which in turn allows high quality, maintanable applications to be built.

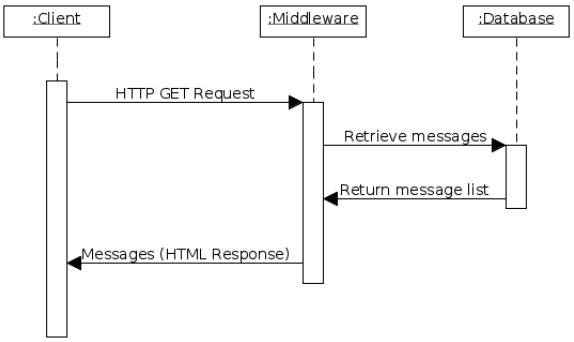
- Security issues and bugs found in the framework can affect all application that use that particular framework.

4. MESSAGE PARSING

4.1 Sending a chat message



4.2 Displaying chat messages



- 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. And our thanks to the peer reviewers for their feedback. Be sincere and be specific about how others have helped your group.