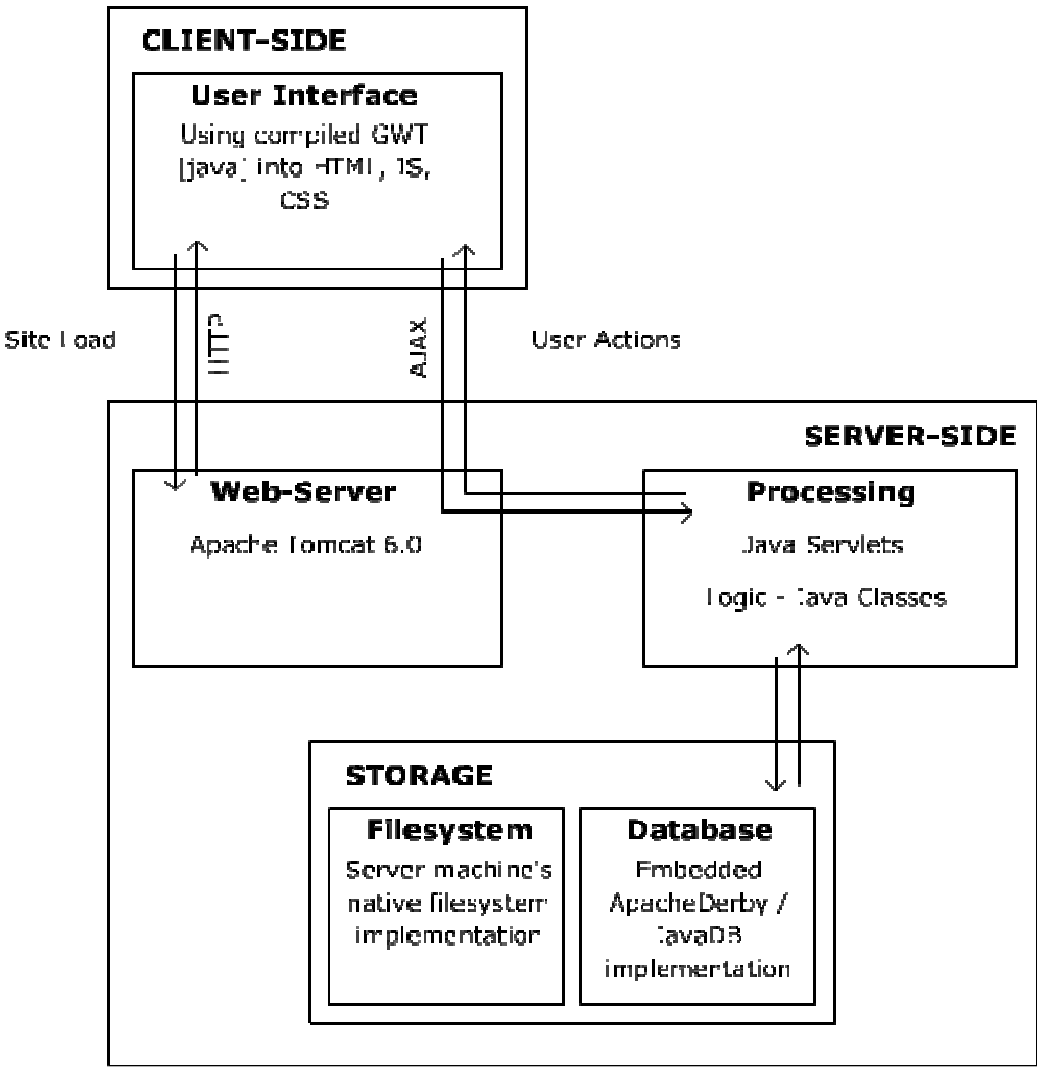
Mario’s Party

Design Document

Contents

Overview

Our project is split into two main parts; the client-side and server-side. The client-side portion of the application contains the user interface. The server-side is split into three different parts, the web-server, processing and storage.



CLIENT-SIDE

Graphical User Interface (GUI)

Purpose:
To provide a clean and engaging interface for users to intuitively interact with.

Priorities:
From highest to lowest priority, we prioritised our GUI decisions as follows:

1. Easy to use, without requiring much learning
2. Aesthetically pleasing
3. Allow for dynamic interactions

Outline of the design:
The GUI is the most important part of our application for the user, as it is the only bit of the software that they interact directly with. We need our GUI to be easy and fun to use so that the user has a productive and interesting learning experience.

Design Issues:

Issue 1: How to code our GUI

Option 1.1: Use a web interface
The advantages of using a web interface include the following:
It is easier to access the application on a wider range of devices
The group has more experience with writing web interfaces as opposed to any other type of interface.
The browser has built-in functionality to allow for uploading of images, allowing users to contribute towards a global collection.
It is easier to implement patches and updates without the user having to do it themselves, ie has better maintainability.

Option 1.2: Use a Swing interface
Allows the user to run the application without using third-party web-browser software (i.e. directly on the desktop).
Allows the program to have write access to the local machines (prevented by browser software for security reasons).

Decision 1: Option 1.1
We chose to use a web interface for the application because there were more positives than for using a swing interface. It is important for us to have sufficient knowledge about the way we code the GUI so that it can be created to a high standard. In order for our framework to be easy to use and have content that can be shared with others, it makes more sense for it to be a web application.

Issue 2: Whether to use Google Web Toolkit (GWT), another JavaScript library (such as Mootools or EXT-JS) or manually write all the corresponding JavaScript files

Option 2.1: Use GWT
Allows us to write the GUI using Java, and still port it across to a common web format (JavaScript) – without requiring the user to have an up-to-date version of the Java Applet/Web framework/plugins? (Revise this statement)
Allows us to run JUnit tests on the GUI, to help us to protect against 'buggy' interface.
Strict validation at compile-time reduces the likelihood of scripts containing syntax / language errors
Generated JavaScript is guaranteed to work across all major browsers.
Has many built-in widgets / interface tools
Has good online support

Option 2.2: Use a JavaScript library (Such as Mootools or ExtJS)
Allows us to write the JavaScript files manually, which is closer to our previous experiences writing interfaces for the browser.
Possibly reduces the physical amount of code required to create interface components (GWT appears to be quite lengthy).
Generated JavaScript is guaranteed to work across all major browsers.
Has many built-in widgets / interface tools
Has good online support

Option 2.3: Write the raw JavaScript from scratch
Doesn't rely on third-party software – less risk involved of third-party vendors dropping the projects or going out of business
Doesn't require external libraries

Decision 2: Option 2.1
We decided to use GWT at the direction of Bernhard. In addition to the useful functionality provided by GWT, the University (the client) does not appear to have a group of students well-versed in using GWT, and is looking to gain some expertise in this area.
It is also new to everyone in the group and should provide a good learning experience.

Issue 3: The layout of the image pages

Option 3.1: Put the picture in the centre of the page at the top with tags (both textual and audio) underneath

Option 3.2:

Decision 3:

SERVER-SIDE

Web-Server

Purpose:
This section describes the design decisions for the web-server we are going to use.

Priorities:
From highest to lowest priority, we prioritised our web-server decisions as follows:

1. Reliability
2. Speed
3. Implementation Costs
4. Language Implementation

Outline of the design:
The web-server is the software to communicate with the client pages. It deals with: getting data from the filesystem & database; processing of the data; sending and receiving data to and from the client.

Design Issues:

Issue 1: Which language provides the best logic implementation on the web-server

Option 1.1: Use Java
Most members in the group are more comfortable and used to the Java programming language. Java has a strong presence within the University and throughout the software engineering world.
Java has testing tools such as JUnit
Java has documentation tools such as JavaDoc
Java integrates well with several database implementations, specifically JavaDB (formerly Apache Derby).

Option 1.2: Use Python
Some members of the group have previous experience developing web-applications with Python
Python has a several web-specific frameworks available to help reduce implementation costs.
Python has quickly growing support from the online community
Python has testing and documentation tools available.

Decision 1: Option 1.1
We decided to use Java as it is the language that the group in general was the most experienced in and comfortable with. Using Java also gives us the option to use the Google Web Toolkit (GWT).

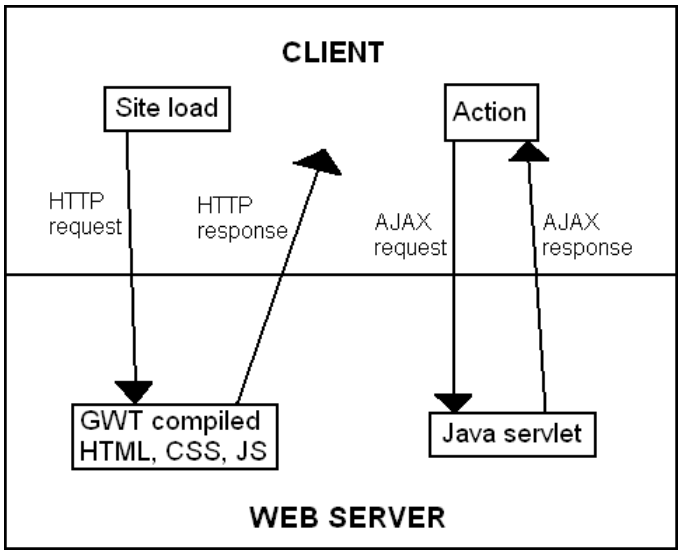
Issue 2: Which web-server implementation to use

Option 2.1: Create our own
This is the idea to build our own web-server from scratch. The server would implement provided java interfaces.
Some members of our group have experience creating kernels & web-servers in other programming languages

Option 2.2: Use an existing open source web-server (such as Apache Tomcat)
This is the idea to use an existing open-source Java server, such as Apache Tomcat.
This has the advantage of being already built – possibly reducing our overall production costs.
Significant online support in both documentation and implementation of the software – therefore it is likely that Apache Tomcat could prove to have better underlying design (in terms of scalability, reliability etc)
Some time required in order to become familiar with the API etc

Decision 2: Option 2.2
The decision was made to use Apache Tomcat. This was based on the general priorities mentioned above. It is likely to be more reliable than anything we would be able to make in the given amount of time. Coupled with scalability and responsiveness, these were our top priorities for a web-server. Tomcat has been around for some time and has been highly recommended online. It is constantly updated to provide consistent use with an ever changing environment. Any server we created would likely take too much maintenance to compete with Tomcat.

The following diagram shows how the client and web server will interact with each other:



Processing

i. Java Servlets
ii. Logic: Java Classes

Storage

Database

Purpose:

Priorities:
From highest to lowest priority, we prioritised our database decisions as follows:

Outline of the design:

IMAGE
id int [primary key]
location varchar

BOUNDING_BOX
id int [primary key]
image_id int [foreign key]
start_x int
start_y int
width int
height int

TEXTUAL_TAG
id int [primary key]
content varchar
dialect int [foreign key]

AUDIO_TAG
id int [primary key]
location varchar
dialect int [foreign key]

BOUDING_BOX_HAS_TEXTUAL_TAG
id int [primary key]
bounding_box_id int [foreign key]
textual_tag_id int [foreign key]

BOUNDING_BOX_HAS_AUDIO_TAG
id int [primary key]
bounding_box int [foreign key]
audio_tag int [foreign key]

TEXTUAL_TAG_HAS_AUDIO_TAG
id int [primary key]
textual_tag int [foreign key]
audio_tag int [foreign key]

DIALECT
id int [primary key]

name	varchar
language	int [foreign key]

LANGUAGE	
id	int [primary key]
name	varchar
charset	varchar (this might need to be a restricted set of options)

Design Issues:

File System

Purpose:

Priorities:
From highest to lowest priority, we prioritised our file system decisions as follows:

Outline of the design:

Design Issues: