**AberDock BitTorrent Tracker Website**

|  |  |
| --- | --- |
| Report Name | Project Outline |
| Author (User Id) | Jack Ryan Davies (jrd15) |
| Supervisor | Dave Price (dap) |
|  |  |
| Module | CS39440 |
| Degree Scheme | G400 (Computer Science) |
|  |  |
| Date | February 9, 2022 |
| Revision | 0.1 |
| Status | Draft |

# Project description

The goal of this project is to provide a web interface which allows users to share files via the BitTorrent protocol.

It will do this by providing a browse and search system which allows users to find the content they are looking for, view a description of the content and download a .torrent file which may be personalised depending on if the user is logged into the site or not. Personalised torrent files will have a custom announcement URL which features the users “peer ID”, also known as a “PID”, which is used for gathering user statistics.

Content downloaded as a logged in user will update statistics on the users account, such as total download, total upload, download/upload history, currently active torrents and the users share ratio. These stats offer competition between users and can also be used to implement a “hit and run” system, which is designed to ban users who don’t seed the content they download and promotes the BitTorrent ethos of sharing content and keeping files online.

Not only will the site provide a library of torrents, but it will also run the announcement system, which is the component that torrent clients communicate with in order to receive a list of IP addresses of peers (which is needed in order to connect directly with other downloaders/seeders). This component is also responsible for gathering data from clients such as which file is being downloaded, how much the client has been downloaded and uploaded, how much is left to download, what client is being used and what state is the client in (downloading, seeding, disconnecting).

The login/registration component of the site will allow users to rate content, leave comments, customise the site based on preferences, bookmark and store history, and view personal statistics.

# Proposed tasks

In order to produce the desired product, I will need to do some additional research into creating customised Docker container images that allow for easy deployment of the software and to guarantee that the solution functions correctly in a verified environment. This will involve learning about the structure of docker-compose (used for deployment) as well as Dockerfile (used for creating images) [1].

My choice of language for the back-end is PHP, as this is what I am most familiar with and is used to power the majority of websites on the internet. It is heavily documented [2] and has support for both Linux and Windows operating systems. It shares a lot of similarities with Java, in the way object orientation is handled. One of the biggest drawbacks to using this language is that it can be pretty slow to execute (Nextcloud [3] is a great example of this) when used to develop complex applications and one workaround for this is to implement a caching system, which I am not familiar with. So I need to do research into optimising and configuring a caching system to aid in the projects goal in being as scalable as possible.

I plan on using MariaDB/MySQL for the database system, as the library I plan on using for executing database queries securely: Medoo, currently only supports this back-end. However, I intend on researching the performance differences between MariaDB and PostgresSQL, and if Postgres offers a noticeable uplift compared to MariaDB, this may warrant additional research into alternative libraries that offer similar functionality to Medoo. A caching system for the database, such as Redis [5] may be a viable solution in improving performance.

BitTorrent files are encoded using “Bencode” and the back-end needs a way of parsing this data into an array that can be used for modifying certain elements of the file (such as tracker URL’s) and re-encoding it. In the past I have used modified code provided by xbtit [6] in order to achieve this as it provided a foundation for learning how to properly handle this file type. For this project, I intend on writing my own class from scratch in order to achieve the same result whilst being written with higher standards. I have found a few sites that document the file structure [7][8].

The web interface component will be written using modern HTML5 standards, CSS and JavaScript (though my intention is to make the site fully functional without JS, to improve performance as well as make it usable for users who disable JavaScript due to security concerns). Whilst I have a strong understanding of developing responsive websites, browsers are implementing modern features that may be worth implementing, though the downside to this is compatibility with older software. There are online video tutorials by Kevin Powell [9] that are regularly uploaded, explaining new features as well as better ways of writing HTML5 and CSS. My goal is to use this source as well as read official HTML5 documentation [10][11] in order to develop a modern solution that has an optimal codebase.

# Project deliverables

My project should consist of the following items.

**Git repository**

The produced code and documentation will be available on a private Git repository. This will either be on a Gitlab or Github instance. The product in which I will use for this has not yet been determined. The Git repository will also list milestones and tasks that need to be undertaken.

**Diary**

A weekly diary will be produced in the form of a Markdown file. This will be available in the Git repository and I intend on producing a website that takes the content of this file and render it in an easy to digest format that is easily accessible, rather than needing to navigate a repository to view.

**Docker images**

Docker configuration files will be produced that will allow users to quickly and easily build and deploy up to date images of the produced solution. This should also make development more streamlined as images will be pre-configured with a known working setup. These will be stored in the Git repository.

**Storyboard of the web UI**

Initial plans and mockups of the web interface will be produced using GIMP and presentation software such as Google Slides or LibreOffice. This will give a clear guideline on how the interface is meant to appear and how it should be navigated. Having a pre-determined guideline will aid in producing a consistent and intuitive interface.

**Web interface and back-end code**

The main front-end solution that users will be interacting with. This will consist of multiple files, such as PHP, CSS, JS, JPEG, PNG and so forth. The completed solution will be mobile and desktop optimised, and it should implement accessibility features to make it reach a wider audience.

The back-end code will be written in PHP as well as libraries.

**Test table/automated testing**

A test table will be produced to test core functionality of the product. Tests will be written during the development phase of each component, so that bugs are caught early in the development cycle and addressed. The format in which this will be provided is Markdown, as this has the benefit of being easily modified and viewed on the web.

**Database**

A pre-made SQL dump of the sites database will be provided in the Git repository. This will be imported into the database on the first setup and used to update the database if structures change in future releases. The database will be used for storing torrent metadata as well as site and user statistics and credentials.

**Documentation of core components**

Core components will each have their own manual, likely in the form of PHPDoc. This will help developers in understanding the architecture of the various components. Documentation will be generated as HTML files, for easy digestion and viewing.

**Maintenance and setup manual**

A Markdown document explaining how to setup the solution from scratch as well as how to fix known issues that the user may encounter. This will be part of the git repository.

# Initial annotated bibliography

1. Docker documentation: <https://docs.docker.com/>
2. PHP documentation: <https://www.php.net/docs.php>
3. Nextcloud: <https://nextcloud.com/>
4. Medoo PHP library: <https://medoo.in/>
5. Redis: https://redis.io/
6. xbtit: <https://github.com/sir55/xbtit-3.x>
7. File Formats Torrent File: <https://fileformats.fandom.com/wiki/Torrent_file>
8. Theory.org BitTorrent Specification: <https://wiki.theory.org/BitTorrentSpecification>
9. Kevin Powell YouTube channel: https://www.youtube.com/@KevinPowell
10. W3 HTML5 specification: <https://dev.w3.org/html5/spec-LC/>
11. Mozilla HTML5 documentation: https://developer.mozilla.org/en-US/docs/Glossary/HTML5