**Speedruntimers.com**

**Background:**

Speed running is a hobby of many video game enthusiasts, in which an individual or a group of individuals try to play a video game as fast as possible to beat each other’s records. It primarily thrives on streaming sites such as Twitch.tv, a popular website dedicated to streaming video games. An important factor in racing is time, because, as seen in the Olympics, records can be broken by fractions of seconds. There are many timer solutions available for speed runners each with unique features to enhance the speed running experience. A major feature in these timers is splits. A game could take hours to complete so it is very helpful to know how long you are along in the game, which is where splits come in. Splits are user-set points pre-set in the timer which correlate to certain points in the game to let the runner know how he/she is comparing to their previous runs. For example, if somebody was speed running Mario, they could have “splits” at the end of each stage. They would start the timer when they started the game and then press a button on the timer at the end of each stage. The timer would then display the time they finished the stage, how far behind or ahead they are from their previous records, and how much of the run they have to go. A problem with these timers and splits is that there is no central repository in which to obtain these splits. A person either has to make their own or have someone send them one.

A popular event in the speed running community is having group races. Group races are usually held on a site called speedrunslive.com. Speedrunslive.com is a website that displays current races, has a ranking system, and an IRC (internet relay chat) server that controls many of the functions needed to have a group race, such as “join race”, “start”, and “finish”. However there is no convenient way to see how you are comparing to the other users in the race, and even more, everyone could be using different splits.

**Description:**

My project aims to solve most of the problems associated with speed running timers and group races. This project has three major components: the website (speedruntimers.com) itself, a browser and stand-alone timer client, and a websockets server.

The planned features for the website are:

* A user account system with login
* An interactive website
* A split database where users can upload/download/modify splits
* Ability to upload/export splits in the file formats of other timers
* Organization of group races where spectators can also view current splits

The planned features for the local/web client:

* All the functionalities of competing timers, such as hot keys, customizable pictures, basic timer feature, graph view, etc
* Ability to sign in through website account and upload/download/modify splits directly from the client
* Ability to use/export splits in the file formats of other timers
* Modular graphics system with support for custom plugins and reorganization of layout
* Ability to have a custom sound play when split

The planned features for the websockets server are:

* Ability to have group races
* Ability to see other members times
* Ability for spectators to view the races
* Possible integration with speedrunslive.com or twitch.tv.

**Possible Outcomes/Goals:**

My hope for this project is for it to have a successful community that finds use in my efforts. In the future I may monetize the site and earn revenue from ads, but right now I am only aiming to make having races and speed runs as easy and as accessible as possible.

I also anticipate that this will teach me a lot as a programmer and help me with projects in the future. This will hopefully guide me to become more conscious of the things I code and better understand the design process.

This is definitely an ambitious project and I do not expect to include every single feature because of time constraints but I do plan to have at least a basic timer program, a split database/website, and the ability to have group races.

**Methods/Steps:**

This is the rundown of the order of events (likely to change/flexible):

1. Create Website

-Create basic URI system

-Make basic split repository/database

-Spruce up website with lots of CSS, JavaScript, and jQuery

-Add user account system

1. Create Timer

-Create skeletal class system for timer with room to grow

-Make the basic split and timer module

-Create more useful modules

-Allow users to sign in on client

1. Make websockets server

-Create basic websockets handlings and server logic

-Fine tune to integrate with website accounts

1. Polishing and publishing

-Link website and client to websockets

-Add last additional

-Refactor code

-Make site and application easy to maintain

-Possibly add APIs for the applications for the community to get involveds

-Bug testing

-Publish site

**Research/Sources:**

The following is what languages/libraries I will be using in my project:

* Flask: a Python library for creating web-servers
* HTML/CSS/JavaScript/jQuery: for creating an interactive website
* SQL server: database for users and splits, accessed via Flask
* Java: for creating the timer client
* Some websockets server implementation (sockets.io, ratchet, jWebServer, etc): to make the websockets server

I already know all these libraries and languages (with the exception of the websockets server), but I can learn a lot more of Flask and JavaScript. If I run into any issues in development I will use various internet resources and documentations.

**Checkpoints/Interim Due Dates:**

I am planning for this project to be an all year project, broken into many sections. These sections will be:

1. Designing the website with interactivity, split database, and user system
2. Creating the timer client
3. Making a websockets server and implement this functionality into the client
4. Adding additional features, polishing everything, and publishing the website

**Closing remarks:**

I plan on continuing this project past the due date and/or when I finish. I see this as a starting point, version 1.0. I anticipate this growing into something big, and if I cannot continue it, I will make it as easy as possible for the community to continue it for me.