ThinkItThru!: Status Report

Jensen Schmidt, Zachary Travis, and Levi Wright

# Tasks and Objectives Updates

By implementing a chunk of our Class Diagram into the scripts of the site, we have created a Task and TaskList system and the base for the Objective’s system, which includes Daily Objectives. The Task system involves the ability to create Tasks and add them to the TaskList. Task Objects can be created with their name, time required to complete, priority (rated by the user upon creation as high, medium, or low priority), difficulty (again, rated by the user), due date, and optional subtasks. The Task currently can calculate the number of days until it is due and create a “queue score” which ranks it in terms of importance depending on the amount of work time remaining on the Task, the Task’s priority, and the due date. The task is added to the user’s TaskList, which then adds the Task in order based on the queue score. These Tasks can be referenced by index within the TaskList by other sections of the site.

The Objectives system is created with the user’s TaskList, and with later updates can be updated upon the changing of the TaskList. It separates each Task into different “Day” objects, each representing a date that the user has something due. It creates the day’s Daily Objectives List by calculating the average time the user needs to work each day until a Task’s due date. It begins with the nearest due date and ensures that all Tasks on that date will be completed by finding the average amount of time the user should be working per day to complete them. It then iterates onto the next day, where a new average work time per day is calculated. If the new average is less, it means that the further away work can start *after* the nearest work is completed. If it is greater, then more work needs to be done today to get a head start on those later Tasks, so the increase in time to work is added to work on that task in the Daily Objectives List. This system iterates through every Day object from nearest to furthest to ensure that the user knows what Tasks they must work on that day and for how long.

The current model of the website creates a Task List and adds Tasks onto it in order of queue-score-based importance. The Tasks can then be found and modified if needed. From this TaskList, Daily Objectives can be created. Currently, Tasks and Objectives can only be viewed through the console and all objects are created within the script itself in this test version.

# Database, Hosting, and Authorization Updates

The database has deviated from our original design, finding that JSON and Firebase Realtime Database does not have a lot of support for relational data. Originally (as seen in our ERD), we planned for the database to split up the necessary data it would need to reference in a single pull from the database. This was done to avoid too much data being pulled at once—a process called “normalization,” that is also intended to reduce data redundancy and improve data integrity. Having seen that Firebase Realtime Database opts to assign unique IDs to data objects (and does not include a clear function to use those same IDs in a separate data object in the JSON tree), we have decided to brainstorm more structures for the data.

While it is not inherently impossible to structure our database relationally, it would be somewhat difficult to go against the intended structure of Firebase’s Realtime Database. Our database structure is currently being remodeled and will have a few options to choose from—we will be deciding between moving forward with our current relational setup while instituting our own primary keys, or de-normalizing the data to work with Firebase’s software. Once this is done, we will modify the JavaScript to the necessary format for updating and creating data objects.

The JavaScript update and create functions still need to be finalized and incorporated with the task logic to provide the necessary data for the program to function. We have also realized a lacking element in our original design: we never asked the user for the free time they had available within the week. Several new data values are being added to the database and registration form to accommodate the user’s available time from Sunday to Saturday. In addition to storing the user’s free time, we will be adding a “total time worked” element to the task objects. We will also be taking away a few data items—in discussing the algorithm, we have concluded that the user no longer needs to input a “minimum” amount of time they want to work on a task and can instead delegate their time as they see fit. The subtasks will also be reduced to mainly extensions of the original task, rather than miniature versions of the tasks; for the time being, we will keep them as names and check marks associated with the task.

Our site is currently being hosted at a URL provided by Firebase (thinkitthru.web.app), however we have not released the current version of the site. Currently, we are working on the host files the site will run on, but once the database has been completely solidified (either relational or de-normalized) and the code shows to be functional, we will roll out our first deployment. Authorization through Firebase has also been established; however, we are looking to change it from the provided code and look more like the rest of the site’s design.

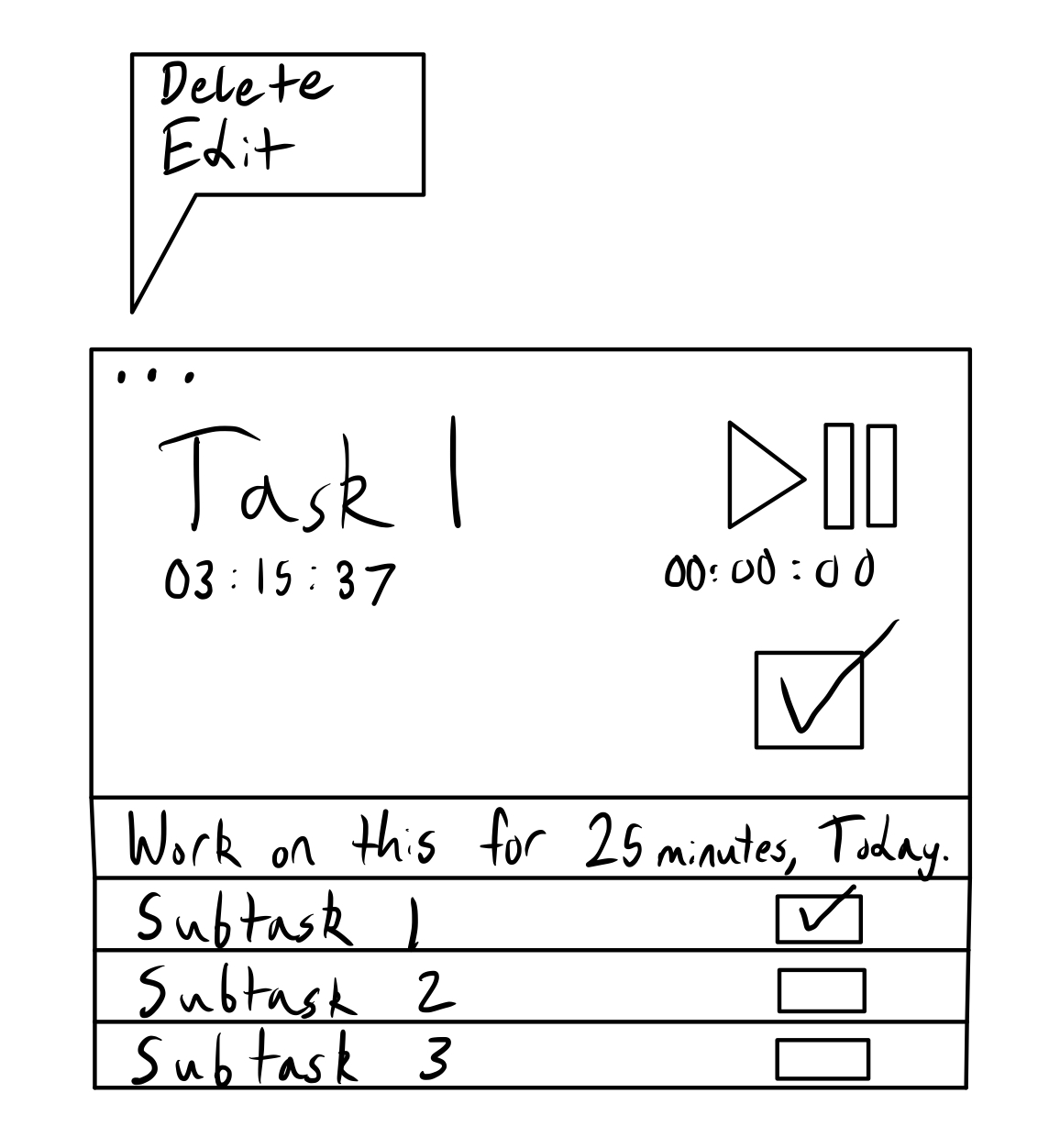
# Design Updates

ThinkItThrough!'s HTML/CSS/JavaScript design skeleton is materializing. The dashboard now features a navigation bar at the top, with an expandable hamburger menu. The menu is expanded using a JavaScript function to show/hide the container element with the links in it. Below, a "flexbox" list of dynamically generated task containers. Each task container itself will be a flexbox container, holding the main task view and subtask/additional information. Divisions inside the task containers will be implemented with a "grid" of dynamic properties. Dynamic properties and menu functionality are linked to JavaScript functions and data.

Icons are imported using the free and open-source web service Icon Noir. Using Icon Noir gives us access to over 1,500 commonly used icons and will save our group graphic design time. They are dynamically pulled from a file hosting platform, which requires an internet connection to be established. However, our application is a website, and this should be expected.

Graphics assets have been generated using hotpot.ai and midjourney.com. Some assets are created with a "seed" image, which greatly improves the quality/relevance of images. However, we have completely generated relevant, quality images using only word prompts. The image assets we use, color schemes, fonts, and other design choices consistently support the theme of "school/learning" and target our audience of college students.

We have finalized the look of the Daily tasks that will appear on-screen for the user’s dashboard:



As evident above, we have opted to include a main title for the task, the total work time we have in the database for the task, as well as the total work time for that day. Our timer is a play/pause symbol, with a checkbox below it to indicate that the task is done for the day. At the top left, there will be a symbol to indicate a menu to edit or delete the task, which will bring the user to a menu to edit the overall task that this daily task is derived from. Underneath this, we have decided to simplify subtasks: they will no longer function as miniature versions of the task objects, rather smaller aspects of the original task. Outside of the subtask change, our options menu has been reduced in size to the simple “delete” and “edit” functions.

# Summary of Things That Are Completed:

* Hosting through Firebase
* Database /\* We will make a final decision in changing, but for now the database is up \*/
* Task Logic
* Basic Overlay of Dashboard

# Summary of Things in Progress:

* Final Database thoughts decisions and reworks
* Updated functions to create and update data /\* Dependent on Database Rework \*/
* Adding new points of data
* Removing new points of data
* Connections page and adding discord links
* Gameboard page
* Game store and items
* Weekly Objectives
* User Logs

# Possible Major and Minor Obstacles:

Currently, our main obstacles pertain to the database re-structure and the trouble we have had with CSS and making a design look good for the structure we are aiming for. These could majorly impact us if decisions are not made soon, considering that’s the foundation of the site itself. Once the database is done, we will be looking into design elements and fleshing them out to accommodate our goals. Our next obstacles will be implementation of all these components together, and finally deploying the site to the URL that Firebase has given us.