Technology Feasibility Analysis

Date: 10/26/2017

Project sponsor: Barbara Jenkins

Team faculty mentor: Ana Paula R. Steinmacher

BRAINSTIM STUDIOS

Vincent Messenger (Lead)
Nathan Franklin

Table of contents

1. Introduction

Alzheimer's Disease results in almost 2 million deaths each year. It affects 6% of people over 65 years old. Research has shown that engaging in intellectual activities may reduce your risk of Alzheimer's Disease. Brain stimulation is a method of Alzheimer's Disease prevention that has become easy and fun to do with specially designed games. Current solutions such as Luminosity make use of this concept by presenting users with stimulating brain challenges. The idea is that by giving users a fun way to challenge their brains, users can keep their brains healthy and reduce the risk of Alzheimer's Disease. There is currently one aspect current brain stimulating games don't incorporate.

Research has also shown that regular stimulating social interactions can also reduce the risk of Alzheimer's Disease. Barbara Jenkins, our sponsor, has created a fast-paced word game that incorporates social gameplay in order to provide users with maximum potential for increasing their brain health. The game generates random letter sets with which the user will have three minutes to construct as many words as possible. Words will be constructed in a grid like fashion, which allows words to overlap. Once a game is finished, the user's score is calculated and they can compare their score and words with other users of the game.

There are different game modes that present users with different scoring systems. This forces users to adapt the way they think to the challenge that is presented. The game will generate ten letter sets per gametype everyday that each user can complete. Once a given letter set has been completed, users can then compare their scores with other users of the game. On top of this, the user will have the option to play unlimited practice games, where unique letter sets will be generated at the beginning of each game. However, these practice games are not eligible for community comparison.

Currently, the game takes place with a pencil and paper. Our team tested this in our first meeting and discovered the same issues Barbara has told us about. Manually keeping track of time ended with us shocked to find out over six minutes had passed by the time we were all happy with our word combinations. Upon calculating our scores, a few of us had misspelled words, and calculating our scores correctly was not as easy as we thought it would be. Currently the way Barbara compares scores with other players is by emailing both letter sets and scores to other participants. This is problematic because it is time consuming and does not give current players an easy way to compare their challenge solutions. In addition, game scores should not be viewable by someone until they have completed that letter set. With just email, there is no way to prevent someone who has not played the tileset from viewing an email containing scores and words constructed. This paper version relies on trust; the user keeps track of the three minute time limit and calculates their own score.

All of these problems will be solved by implementing this game as a web application. There are a few great differences to improve the game over the pen and paper version. Connected letter tiles of length two or greater will be validated against the same list of valid words for all users, changing colors to tell notify the player of an acceptable word. This word validation will happen any time a letter tile is placed, moved or removed from the play grid. A score calculator will also be updated as users construct words onto the board. Scores will be uploaded to a database, where they can be retrieved and viewed. From this, users will be able to view high scores for specific letter sets. We will also provide users with a way to create communities of players, so they can filter high scores to just those they wish to see.

In this technology feasibility analysis document, we discuss the main elements to this project that we will implement. The team needs to investigate all possible technologies for each aspect of the project, because changing a technology after development has begun will result in wasted time and effort. Throughout this paper, we will analyze our technology options for user authentication, server language, database, word validation, and Javascript frameworks.

2. Technological Challenges

Based off the project requirements, the team has identified several challenges that should be evaluated before work on the web application can begin. The team will need a secure method for authenticating users, as well as a database to store user data. To facilitate communication between the frontend and the authentication and database services, the team will need to create a server. Therefore, the team should evaluate which programming language we should use to create this server. Since this web application will be a timed game, we need a quick and reliable way to validate words that users create. Lastly, the team needs to find a suitable Javascript framework to communicate with the server and to provide tools for UI manipulation.

3. Technology Analysis

In this section, we have our in-depth analysis for each of the requirements that we proposed earlier in this document. For each requirement, we compare our top three technology choices, and justify our final decision.

3.1 Authentication

In order to keep track of individual user game scores, we will need to give the user the ability to create an account and authenticate into the account at later times. When the user

arrives at the web application, they will have to login in order to see any of their user data. Once logged in, the user will also have the option to sign out of the system.

When researching various web application authentication methods, we were looking for options that would be easy to setup and maintain, provide robust authentication functionality, and that were secure. We found a couple highly recommended methods, which were Firebase and Passport.js. Based on our research, we believe that both of these methods would work for our web application. However, there is also the option of creating our own authentication that should be explored.

3.1.1 Firebase

Firebase is a service that is backed by Google. That means that it is built on Google's infrastructure and provides automatic scalability. It has an entire suite of products that can mix and matched to provide robust functionality to users. One of these features is user authentication. It's also important to note that Firebase is cloud based. It also provides workflows for resetting user passwords and other admin-like requirements.

3.1.2 Passport.js

Passport.js can be easily added into web applications to provide user authentication. It currently provides 307 different strategies for authenticating uses, including social sign-on and numerous Open Authorization (OAuth) providers. However, based on our research, it seems that there is still some configuration setup required to get the authentication functioning. Also, the user data would live on the server that is serving the application, which could increase the cost to host the application.

3.1.3 Creating Own Authentication

Creating our own authentication is an option that just isn't practical for our project. We need a secure way to store our user data, which our team could definitely achieve. However, it would be unwise for our team to spend the time to create a robust authentication library that provides the features that are already provided by other established methods such as the ones listed above.

3.1.4 Conclusions

Our chosen method for authentication is Firebase. This option is the easiest to integrate into web applications, and it provides many features that the team can also use. Firebase will also take the least amount of time to implement and maintain, which is important to the team so we can focus on bigger pieces of the project.

	Ease of Setup/Maintenance	Robust Functionality	Security
Firebase	x	x	x
Passport.js	x		x
Create Own Auth			x

3.1.5 Proving Feasibility

Our basic plans for proving the feasibility of using Firebase are to simply implement user login and logout functionality in a web app. We have researched many tutorials of setting up Firebase in a web application, which includes the Firebase documentation, and we have determined that Firebase will be quick and easy to implement using whichever Javascript framework we choose to use.

3.2 Server Language

For this project, the team will need a server for communicating with the authentication service, database, and web applications running on individual client devices. It will also be responsible for serving the web application front-end code.

There are a few things to consider when choosing a language with which to write a web application server. We need the server language to be easy to develop and maintain, we need available external libraries, and we need the language to be easily compatible with Javascript. Based on team experience and research, we have narrowed our server language options to Java and Javascript Node.js. There is also the option of creating a serverless web application that should be explored.

3.2.1 Java

Java is a strongly typed language that has been used to create application servers since it's emergence in the late 1990s. It has a sizable development community, which means there are many external libraries that can be used. One benefit to using Java is that we could take advantage of multithreading. However, since our server logic will not be overly complex, utilizing this sort of functionality might be unwise for our web application.

3.2.2 Node.js

Node.js is a server language written entirely with Javascript. It is extremely simple to write a server to serve a web application's frontend code in Node.js. Also, it can be achieved without the overhead of creating classes to communicate with each other, which is a requirement in Java. Also, because Node.js is written with Javascript, it is readily compatible with Javascript libraries and data structures. Another benefit to using Node.js is that we could easily import external libraries using Node Package Manager (npm). It is also important to note that certain members of the team already have extensive experience with using Node.js to serve web application frontend code and data.

3.2.3 Serverless Web Application

A serverless web application is possible to achieve if there is little need for a server, but there are definite negative impacts due to this. For one, all logic would take place on the client's device, which could potentially lead to security risks. Since we are building a web game that incorporates community score features, it is a bad idea to allow communication with our database to be handled by only the client's device. This would also expose details about the administration section of the web application that should be kept hidden. By bypassing any type of server, we also place restrictions on functionality that can be explored with the web application.

3.2.4 Conclusions

Our chosen option for our server programming language is Node.js. Based on our research, Node.js would fit our needs better than Java. Node.js has the benefit of being able to effortlessly use and serve Javascript Object Notation (JSON) data, and research has shown that Node.js is quicker at performing the sort of tasks we would be building into our server logic. Also, since both the client and server will be written in the same language, if we ever need to move logic from the client to the server or vice versa, we won't have to deal with rewriting logic in a different language.

	Ease of Development and Maintenance	External Libraries	Ease of Javascript Compatibility
Java	x	x	
Node.js	x	x	x
Serverless App			x

3.2.5 Proving Feasibility

To prove the feasibility of using Node.js as our server language, the team plans to develop a server that will serve the frontend code for our web application. We will also add routes for user account creation, user login and logout, and saving user data to our database.

3.3 Database

User scores will be stored in a database every day, which will allow users to compare their solutions with other game players. High scores can be viewed on a worldwide scale, or by user-defined communities of users.

We researched different database technologies with a few key features in mind. Compatibility with Javascript means it will be easy to implement with the rest of the parts of the project, because Javascript is what we are using for the other main components of the project. We mainly focused on open source database technologies for cost reasons. The team is also looking for options that are easy and cheap to host. After comparing database technologies, we narrowed the decision to one of the top-rated SQL databases, one of the top rated non-SQL database, and Firebase.

3.3.1 MySQL

MySQL is the the top rated open source SQL database. It uses a relational database system which is great for complex queries. One key factor for including this in our evaluation is that each member of the team has experience with this technology. Because of this knowledge of MySQL, creating a database and constructing queries would be easier than learning a new database structure. However, since our user data is not overly complicated, using a SQL database could be unnecessary and lead to over complicating our application logic.

3.3.2 MongoDB

MongoDB is a database that is structured using Javascript Object Notation (JSON), so inherently it is compatible with Javascript. It is a NoSQL database, which means that SQL does not need to be used to retrieve stored data. Instead, the database is accessed using a Javascript-like syntax. Due to the absence of a structured query language, there are restrictions in terms of reporting that can be done, because it becomes more complicated to report from NoSQL databases. However, due to our specific requirements, our user data won't be overly complicated, and therefore we should have no issues retrieving data in any way we need.

3.3.3 Firebase

Firebase provides a real-time database that can be used with or without it's authentication functionality. This real-time database is a NoSQL database that stores its data in JSON. Since Firebase is provided by Google and is hosted as a cloud service, Firebase's real-time database is reliable. This also means the team would not have to find a place to host the database. It is important to note that Firebase provides free storage to a certain extent. When the project scales, the team would most likely need to eventually pay for additional storage. However, for the purposes of this project, we should be able to develop the web application without incurring any such costs.

3.3.4 Conclusions

Our chosen database is Firebase's real-time database. The features we decided to compare these database choices on are seen in the table below. Because the rest of the project will use Javascript, it is important for the database to work well with Javascript.Since Firebase offers free storage, we will be able to avoid incurring costs until the project scales. With MySQL and MongoDB, we would have find a solution for hosting the database, but Firebase comes as a cloud service that does not need to be hosted anywhere.

	Includes Host Server	Free to Use During Prototyping Phase	Compatible Javascript
MySQL		x	x
MongoDB		x	x
Firebase	x	x	x

3.3.5 Proving Feasibility

To prove the feasibility of using Firebase's real-time database, the team will setup endpoints to facillitate communication between the Javascript framework and the database. We will make it possible to save and edit certain pieces of user data from the frontend client.

3.4 Word Validation

Part of what makes this web app game better than the pen and paper style of gameplay that Barbara has been doing for over a year now is instant word validation. With every tile

placed on the game grid, any combinations of two or more tiles in a row will be checked for validity using a word list similar to the dictionary Scrabble™ uses.

Features to look at are how fast this validation happens; validation should happen quick enough to give the appearance of instant validation. This application has to run on tablets and smartphones, so available client-side resources will be limited compared to the computers we will be testing this application on. There are not a lot of available tools for word validation, but the tools we narrowed our options to are WordGameDictionary.com Validation API and Word-List. However, we will also be exploring the potential of creating our own word validation library.

3.4.1 Word Game Dictionary API

WordGameDictionary.com has an API that developers can use to validate words. The free version is for non-commercial use and allows up to 5000 requests per day. Because validation happens on a server and transfers tiny amounts of data, the validation can be almost instant and is not dependent on client device resources. We have deduced that the three-minute, thirteen tile game will average thirty word validation calls. There are ten available games per game mode each day, so we will need to be careful how we set our application up to send requests to this server. We need to plan our solution to prevent sending 5000 requests a day, so we can avoid incurring extra costs.

3.4.2 Word-List

Word-list is a github project with an MIT license and uses the same word list as the word game dictionary listed above. There is very little information about this project, and the validation part would still need to be implemented using this. This package basically provides a word list that developers can then use to build their own validation.

3.4.3 Create Own Validation Library

Making our own validation would be time consuming, but would be specific for our needs. There is a tournament word list to start with, but searching through every word looking for the passed in word would be inefficient. This option is not our preference, because the team would like to spend more time on other aspects of the user experience.

3.4.4 Conclusions

Using the word game dictionary will give us a cheat-resistant solution for word validation that will work in a quick manner without relying on the client device resources. While implementing our own word validator would be possible, making it run efficiently enough to appear to validate in real time on any client device would take more effort and time than we are willing to spend.

	Easy to Implement	Free after Scaling	Not Reliant on Client Device Resources
Word Game Dictionary API	x		x
Word-List		x	
Make our own		x	x

3.4.5 Proving Feasibility

For the purposes of creating a working game, we will use the word game dictionary api to effortlessly validate words for this project. Users will be able to construct words on their user interface and validate them against the Word Game Dictionary API.

3.5 Javascript Framework

There are many advantages to using a Javascript framework in a web application. Javascript frameworks save developers large amounts of time and effort by providing well-structured pre-built patterns and functions. It is also much more secure to communicate with servers using Javascript frameworks due to provided functionality.

When researching Javascript frameworks, we needed to find options that would work well with our technology decisions for authentication and database. We also were looking for an option that would have the smallest learning curve for our team. The last criteria we judged our options on was the possibility of external libraries. We narrowed our options down to Angular 4, ReactJS, and Vue.js.

3.5.1 Angular 4

Angular 4 is the third version of AngularJS. It is a MVC based Javascript library that was acquired by Google soon after its release in 2009. Angular provides ways to develop web applications supported across all platforms. One benefit is that since Angular and Firebase are both Google products, they have strong compatibility. Based off some tutorials we reviewed, it requires little effort to integrate Firebase into Angular. Another benefit is Angular has two-way binding, meaning it provides ways to keep frontend data synchronized between the database and frontend clients. Angular also has a large development community, with external libraries

being added every day. It is also important to note that members of the team have extensive experience using angular to create web applications.

3.5.2 ReactJS

ReactJS has grown in popularity in the development community in the past few years. Unlike Angular which is MVC based, ReactJS is more concerned with the task of creating user interfaces. ReactJS is compatible with both Firebase and Node.js, and the team has found that it is simple to add Firebase integration to a ReactJS application. Based off our research, it seems ReactJS has a relatively small learning curve, however since it is basically more of a view library, it could take time to get the team accustomed to best practices. It is also important to note that no members of the team have development experience using ReactJS.

3.5.3 Vue.js

Vue.js is a Javascript framework that in its core is only focussed on the view layer of the frontend client. However, it is advertised as incrementally adoptable, which means developers can integrate more robust Vue.js functionality as they find the need. This means that for simple web applications, Vue.js can be kept as a lightweight package, which is beneficial for speed and efficiency. Vue.js is compatible with both Firebase and Node.js, which would mean the team could use Vue.js with the other aspects of the web application. However, since Vue.js is a relatively new Javascript framework, there could be a bit of a learning curve for the team.

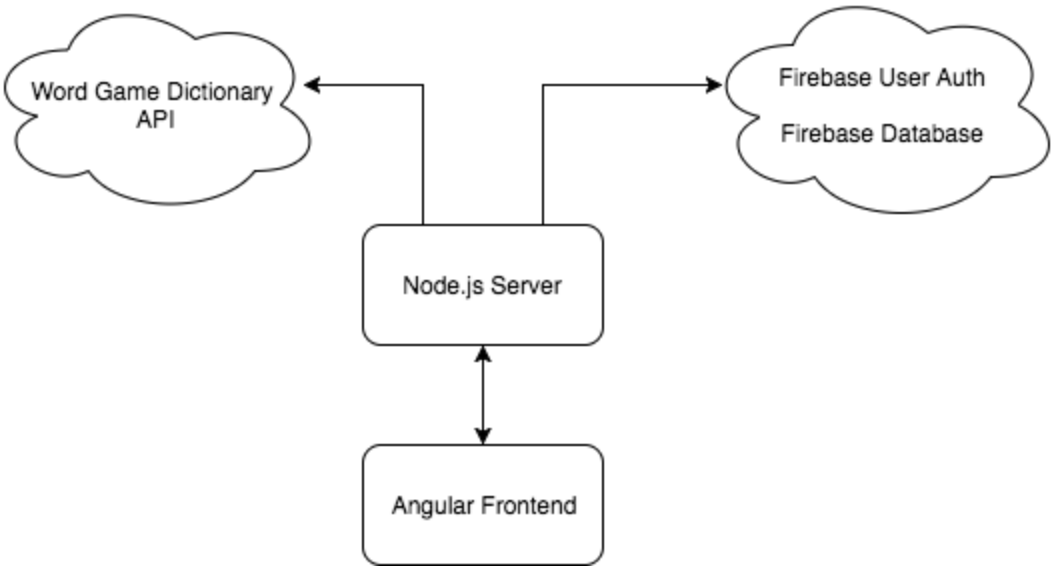
3.5.4 Conclusions

Our chosen option for a Javascript framework is Angular 4. Because Angular is an MVC-like Javascript framework, the team can take advantage of robust functionality for UI manipulation and communication with our various data services. The team can also make use of Angular's two-way binding to keep user data synchronized across devices. Also, since members of the team have extensive experience developing web applications using Angular, we are confident in our abilities to successfully use this framework in our project.

3.5.5 Proving Feasibility

Our plans for proving the feasibility of using Angular 4 in our web application are to use Angular to build a way to connect to the other pieces of the project. The Angular client will be served by the Node.js server. The team will build a way for users to create accounts and authenticate back into those accounts. We will setup endpoints on the server that will facilitate communication between Angular and our various data services. We will also build a way for users to construct words using generated letter sets that they can then authenticate against our chosen word validation API.

4. Technology Integration



Above is a diagram of our planned prototyped system. Frontend code will first be served from the Node.js server to the users web browser. The Angular client that is served to the user's web browser makes requests to the Node.js server in order to communicate with Firebase user authentication service and the Firebase Real-time Database service. The Angular frontend will also make requests to the Node.js server in order to communicate with the Word Game Dictionary API to validate words as user construct them with tiles.

5. Conclusion

In conclusion, our team BrainStim Studios will be developing a brain stimulating web application that will incorporate features to help prevent Alzheimer's Disease. We hope that by integrating a community based scoring system into our game, we will be able to increase the benefits and brain stimulation of our users in order to increase brain health and functionality.

We have researched and found solutions for the five main parts of our project. By doing the research early in the project, we will hopefully prevent the need to change technologies after we start implementing these parts of the project. Spending the time now to compare options and find the best suited technologies saves time, as opposed to finding out down the road that something is not going to work and having to throw out code to switch to a different technology better suited to our needs.

If we had unlimited time and manpower, making our own modules for user authentication and word validation would be our first choices so that the software would work exactly how it needs to. Because of the time constraint, we have found technology solutions that will allow us to spend more time in other crucial aspects of the web application. However, the team plans to develop this web application in such a way that modularization is possible, and technologies can be replaced if we find the need.

	Proposed Solution	**Confidence Level (1-5)** **1-strongly not confident** **3-neutral** **5-strongly confident**
User Authentication	Firebase	5
Server Language	Node.js	5
Database	Firebase	5
Word Validation	Word Game Dictionary	4
Javascript Framework	Angular 4	5

The above table gives an overview of our chosen technologies and our confidence level that we can successfully use these technologies in our project. Now that we have our chosen technologies that we know will integrate well with each other, we are excited to start prototyping WordScuffle. We are confident that we will be able to use these technologies to successfully create a web game to help prevent the onset of Alzheimer's Disease.