

Angle Matching Web Test

Design Document

Version: 0.1

Date: 10/8/2019

Development Team: Jesse Malinosky
Emily Barnes
Emily Fliegel
Jacob Dominguez
Justin Serrano
Aidan Melchior
Alexander Little

T f a e C e b

Table of Contents	1
Terms and Definitions	2
Revision History	2
Purpose	3
High Level Design	3
<p>P A g e b a i c e h</p> <p>N A a A e d i e i c a b i i</p> <p>L N L S Q S Q</p>	
Web Pages	4
Page Navigation	10
Single Page Application	10
Server Side Rendering	11
<p>F D g a a i</p> <p>Signup</p> <p>Login</p> <p>Give an Exam</p>	<p>11</p> <p>11</p> <p>12</p>
<p>F E D g d e i</p> <p>Technology Stack</p>	<p>12</p>
<p>D g a c e d e i</p> <p>Spring Framework</p> <p>Voice Recognition</p>	<p>13</p> <p>14</p>
<p>A e c a a S i h e i</p> <p>I a d O</p>	
Input	14
Touch	14
Voice	15
Output	15
In Browser	15
Document Export	15
Database Schema	15

T D f e a d e i

The IT Glossary of Terms maintains the common terms in this document. Additional terms and definitions specific to this document are included below:

Term or Acronym	Definition
JLO	The <i>Judgement of Line Orientation</i> test
SPA	Single Page Application
PWA	Progressive Web Application
AMWT	Angle Matching Web Test (This Project)
Vue	The open-source JavaScript framework
Spring	The application framework and inversion of control container for the Java platform
IoC	Inversion of Control
AWS	Amazon Web Services

H e i i i

Revision	Date	Description	Author
0.1	10-8-2019	Initial Release	AMWT Dev Team
1.0	12-18-2019	Final Submission	AMWT Dev Team

--	--	--	--

P e

Judgment of Line Orientation (JLO) is a standardized test of visuospatial skills commonly associated with functioning of the parietal lobe in the right hemisphere. Traditionally the test is not of much relevance when testing for dementia. There does however, seem to exist some correlation between test performance and concentration. By developing a modern web application we can facilitate administration of JLO, providing researchers with various latent as well as expected data points.

L g D g e e i e h i

The project consists of a *SPA* developed using *Vue.js* and an accompanying *Spring MVC* based server-side application.

The client will enable users to:

- Register Accounts
- Take the JLO Test
- View Test Results
- Provide Researchers Result Access

The site's compiled source code as well as static assets will be served to users via (aws, netlify, etc). Users can then register an account and begin testing.

Users will take the JLO test interactively by using voice to describe the angles of lines shown. The client will rely on the server-side application to provide transcription of the recorded audio as well as to persist test results.

Once a test has been completed, users will immediately see the results (returned from the server) of the test. They can also select a specific test from a list of past tests and view its result. Tests can also easily be exported in a variety of formats.

The server will provide the following services:

- Voice Transcription
- Test Result Determination
- Result Persistence
- Authentication/Authorization

Project Architecture

Native Application vs Web Application

When developing the project's specification we considered whether we should implement the client as a native application on Android/iOS or a web application. Our application does not have strict performance requirements so we did not feel the need to go with a native application on that basis. Instead, we chose to go with a web-based application which enables us to support more platforms with much less time spent.

SQL vs NoSQL

The nature of the data we are storing will have an implicit schema, thus lends itself to a more structured datastore. With this in mind, we decided to rely on a traditional relational database. The properties of RDBMS will allow us to look for relationships between different data points in an efficient manner should we choose to do so.

Web Application

Page	Function
Home	Describe web application functionality.
Sign up	Allows a user to register a new account as a researcher
Interactive JLO Test	Allows a user to take the JLO test interactively
Test Result	Displays the results of a completed test
Past Tests	A list of completed tests
Settings	Allow user to tweak various settings and personal info fields

Harness the data within.

Easily administer the JLO test and derive deep, statistical insights from the results.

Get Started



Why?

JLO test provides all the functionality of the traditional JLO test, enhanced with multiple latent data points.



Audio Transcription

Cloud powered audio transcription, for convenient test administration.



Data Export

Export test results in for convenient analysis

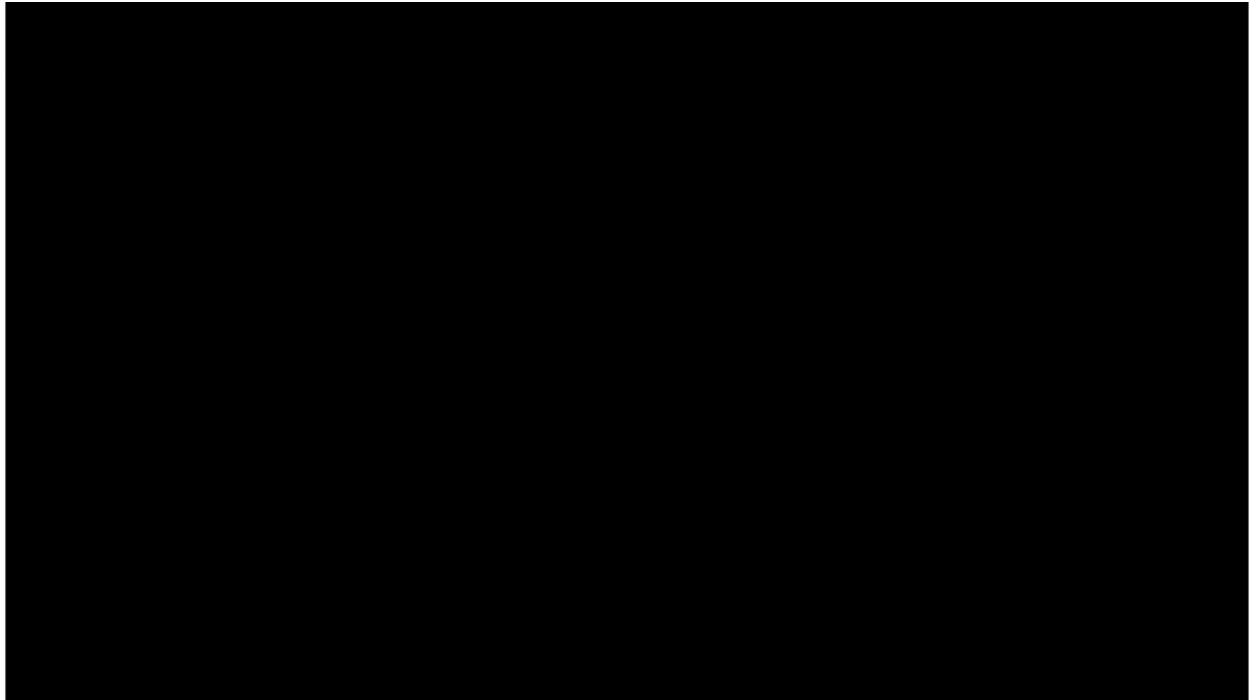


Enhanced Test Results

Enhanced insight through the use of latent data points

How it works

JLO test makes administering tests a breeze. Get up and running in minutes.



Angle Matching Web Test

Home Login

Doctor Sign Up

Please enter the following information:

First Name

Last Name


Email

Password

Retype Password

Submit







Already have an account? [Login](#)

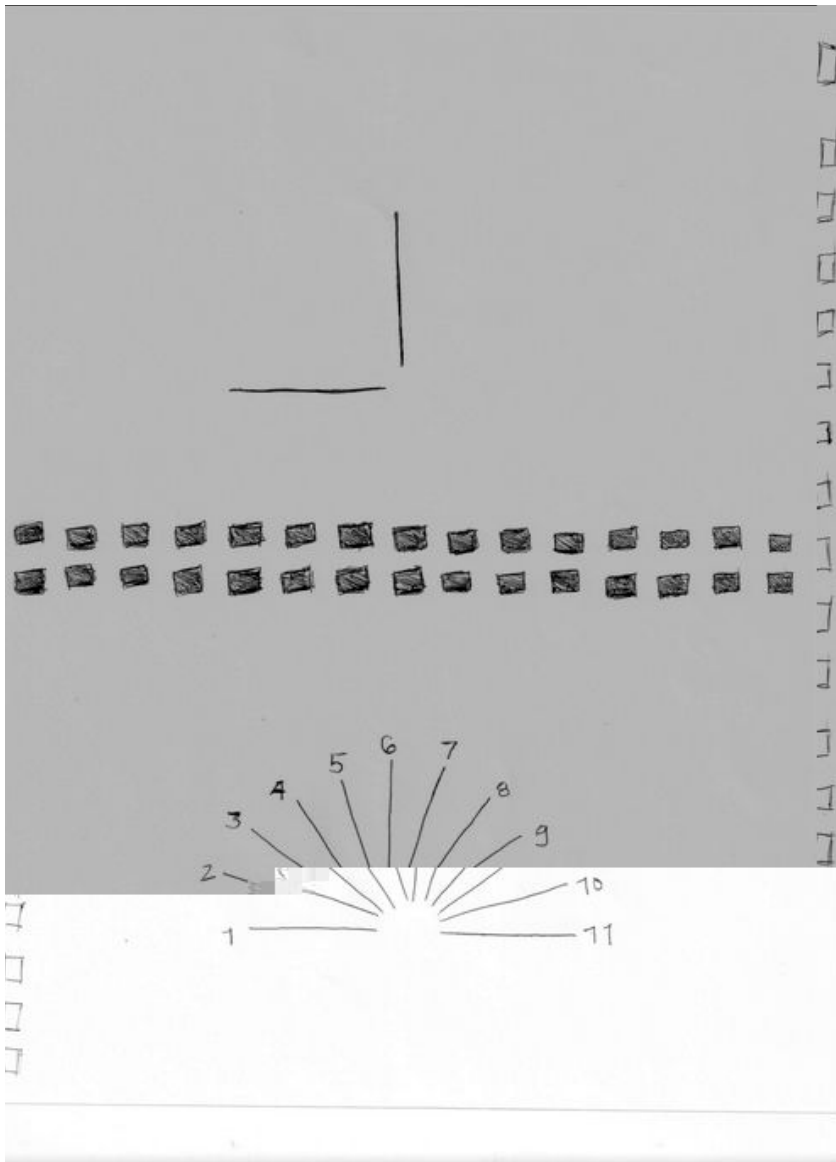
 Angle Matching Web Test

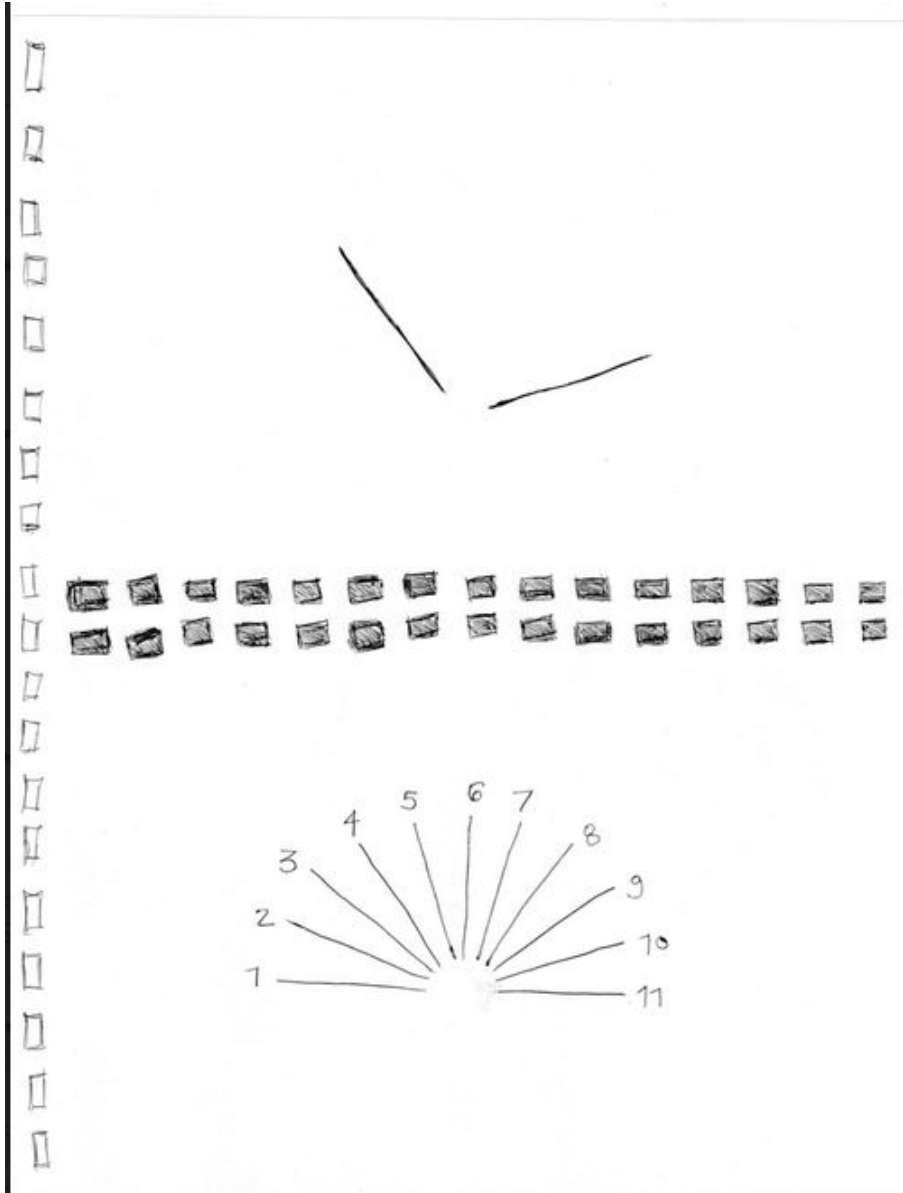
[Home](#) [Web Test](#)

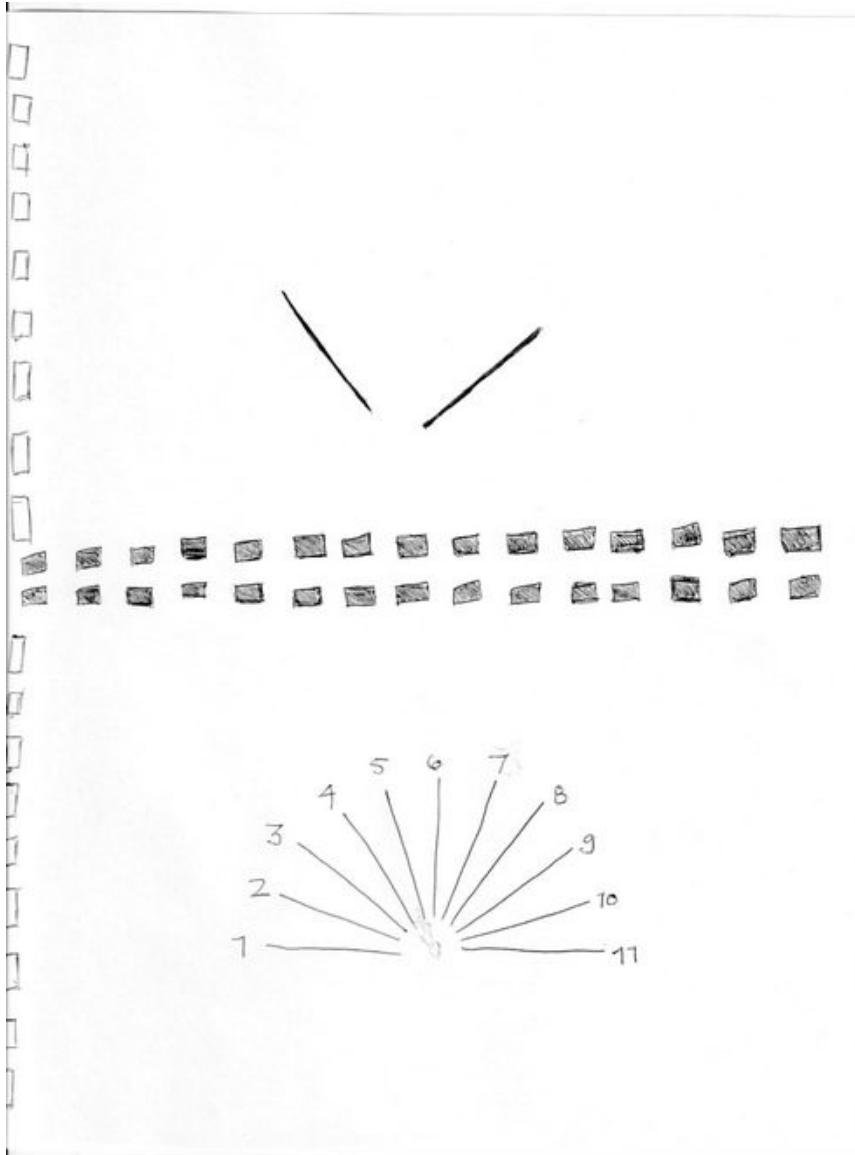
Logout

Dr.[Name]

Patient ID	Results (CSV)
P1	
P2	
P3	
P4	
P5	
P6	







P N g g a e a a i i

Single Page Application

The application can be implemented as a [SPA](#) in Vue.JS. Using the [vue-router](#) package, allows one to define which components will be rendered for a given route. The page will be dynamically rewritten when a user navigates to a given route. The experience will be similar enough to what those browsing a traditional static page are used to and will be relatively frictionless as a result. This is the default way most Vue.JS based pages are developed and is straightforward to implement.

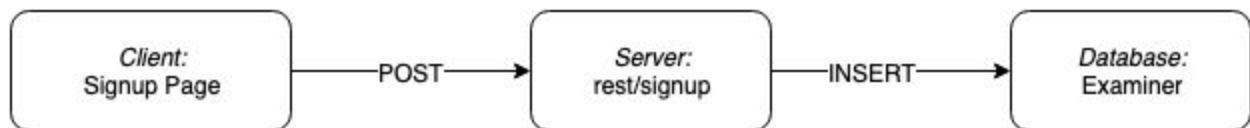
Server Side Rendering

Alternatively the website could be developed using Server Side Rendering. The most straightforward way to do this is using [Nuxt.JS](#). This provides some benefits. With an SSR based there will be less javascript running in a client's browser which brings a performance improvement.

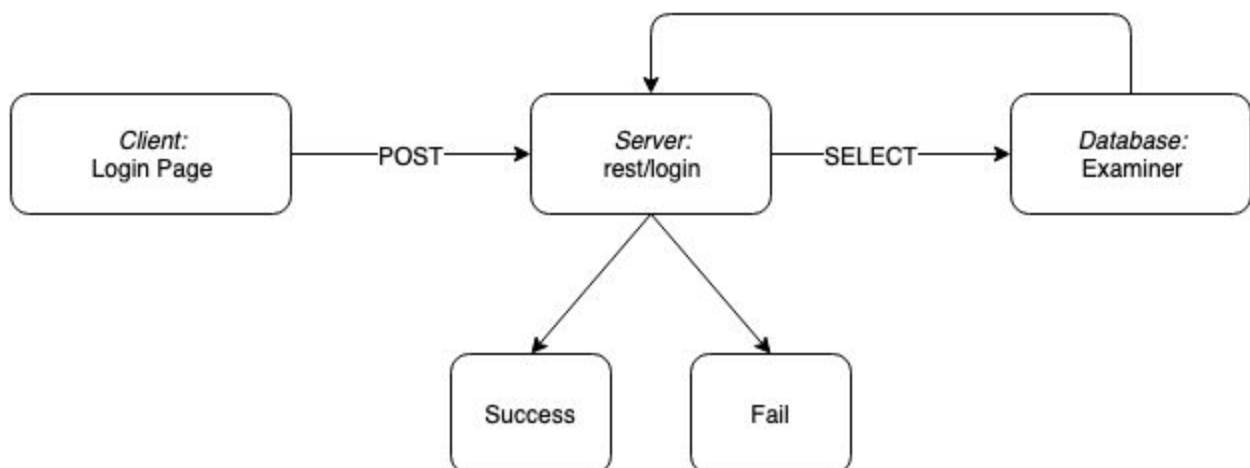
The application will be implemented as Single Page Application, as performance will be fine regardless and Server Side Rendering brings complexities we would like to avoid.

F D g a a i

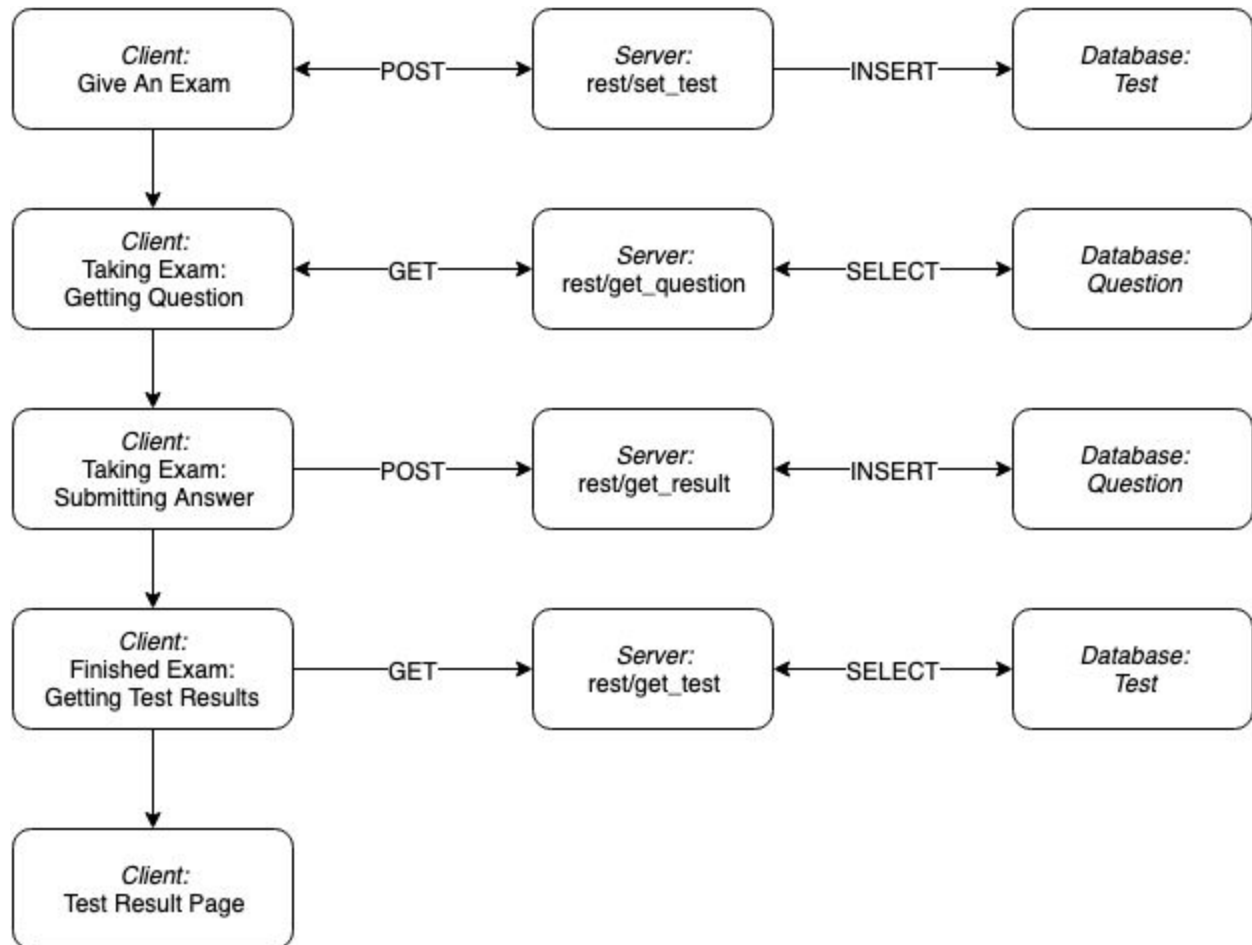
Signup



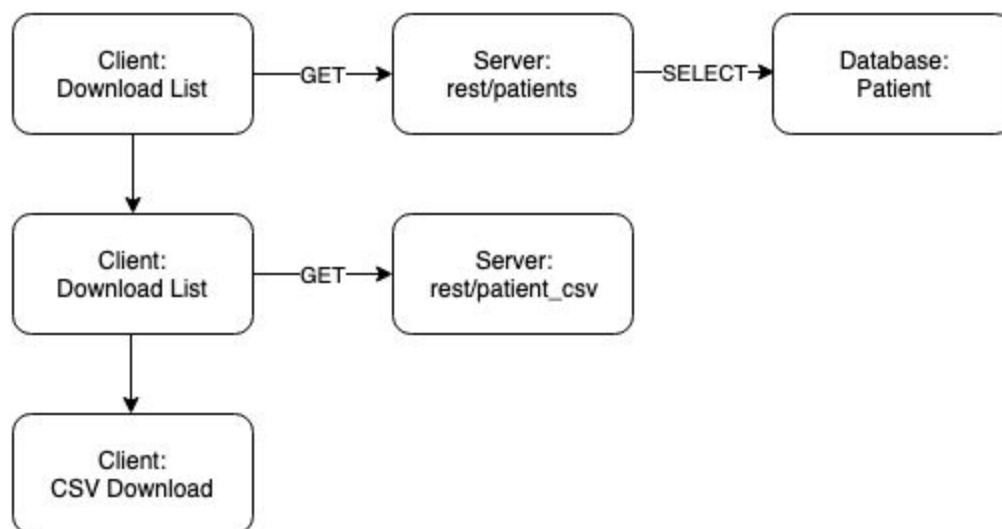
Login



Give an Exam



Download CSV



F E D g d e i

Technology Stack

Tech	Function
Vue.JS	Allows for the creation of clients in an efficient manner.
Vue Router	Allows for page navigation via route-based dynamic rendering
Vuetify	Provides UI components

D g a c e d e i

Spring Framework

At the core of all spring framework modules is Dependency Injection and IoC. This allows us to create a loosely coupled application which makes it much easier when testing. Along with these features is the option to build upon its core concepts with a number of spring modules. These bring in abstraction which will make our code easier to write and promotes further decoupling again making it easier to test.

Some of the Dependencies(excluding core features) we know we will be using are listed below.

Spring Dependencies

Dependency	Function
Spring Web	Build web, including RESTful applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
JPA	Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate
Spring mySQL Driver	mySQL JDBC driver
Spring Security	Highly customizable authentication and access-control framework for Spring applications.
Lombok	Helps reduce boilerplate code

Spring Boot

Spring boot provides auto configuration which allows us to easily start our project without having to worry about setting up all of the dependencies you would typically have to using normal Spring MVC.

Ec2 Server/S3 Bucket

The application will be hosted on Ubuntu using the provided AWS Server. The server will contain both our application and our SQL Database.

An Amazon S3 bucket will be used to store voice recordings taken during testing. These will then be sent to the provided voice recognition endpoint where we will use the resulting info to control the application.

Voice Recognition

A voice recognition endpoint was given that takes in a small wav file and outputs detected words spoken, and the times at which each word was spoken. This is implemented in our API. The command to use this remote endpoint in isolation is as follows:

```
curl -X POST
https://2vijq0kx1l.execute-api.us-east-1.amazonaws.com/api/recognize?apikey=${API_KEY} --upload-file path-to-some-audio-wav-file --header
"Content-Type:application/octet-stream"
```

Output: { String:word, long: time_start, long: time_stop}

Authentication

Implementing basic authentication security in our RESTful API in Spring starts with including Spring's [spring security starter](#). Extending WebSecurityConfigurerAdapter and overriding the configure(), which takes in an HttpSecurity object, and using this object, defines the types and levels of authentication in our application.

These tools included in spring security starter help to create a simple, working authentication system in our application.

Password security: Password will be hashed + salted using RSA before being stored.

Input

Touch

There will be actions available to the administrator via 2 buttons.

- One button will be pressed to signify that the patient is done answering the question. When the button is pressed the audio recording will be stopped and sent to the backend for evaluation.
- A second button will allow the examiner to advance the test to the next question.

Voice

- The client will record voice via a mobile browser. A single clip will be recorded for each questions and asynchronously sent to the backend. The backend will utilize the professor api to transcribe the audio into two angles that will be used as values..

Output

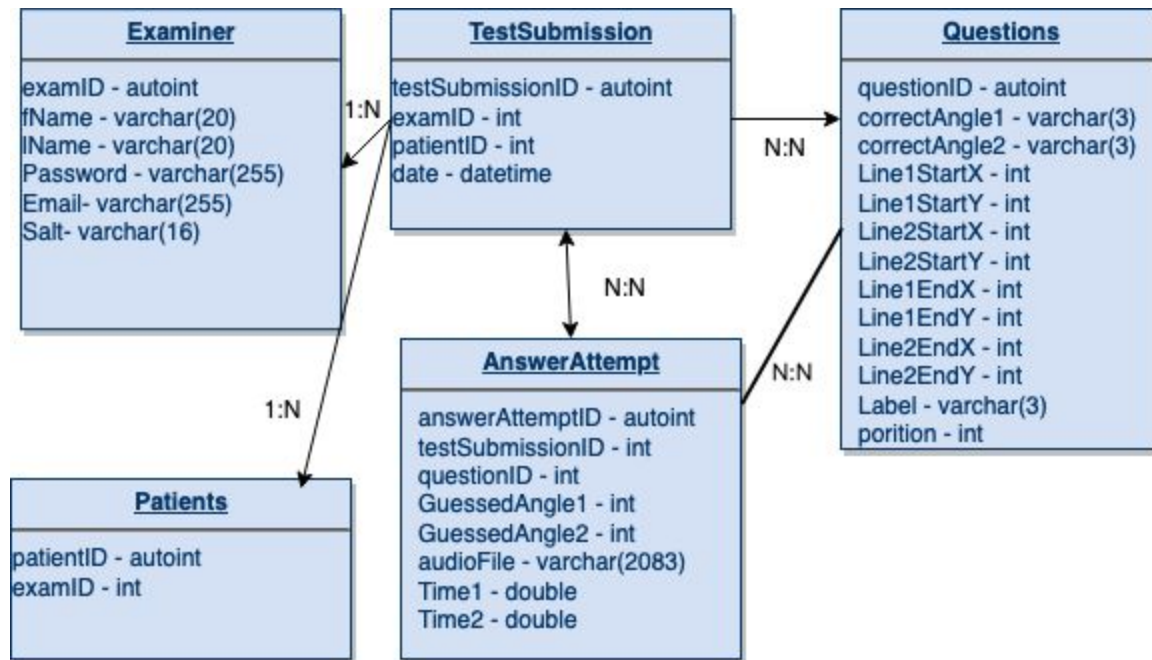
In Browser

- The user will be prompted with tests that are functionally equivalent to the traditional JLO test.

Document Export

- Test results can be exported in csv format.

Database Schema



RESTful Endpoints

- authenticate
 - Authenticated: No
 - Method: Post
 - Path: /auth/authenticate
 - Json In: { String: email, String: password }
 - Json Out: { String: email, String: firstName, String: lastName, String: token }
- register
 - Authenticated: No
 - Method: Post
 - Path: /auth/register
 - Json In: { String: firstName, String: lastName, String: email, String: password }

- existsEmail
 - Authenticated: No
 - Method: Post
 - Path: /auth/existsEmail
 - Json In: { String: email }
- patientAll
 - Authenticated: Yes
 - Method: GET
 - Path: /patient/all
- patientSpreadsheet
 - Authenticated: Yes
 - Method: GET
 - Path: /patient/spreadsheet
 - Params: patientID
- existsID
 - Authenticated: Yes
 - Method: POST
 - Path: /patient/existsID
 - Json In: { String: patientID }
 - Json Out: Boolean
- result
 - Authenticated: Yes
 - Method: POST
 - Path: /test/result
 - Params: file, testSubmissionID, questionID
 - Json Out: Boolean
- start
 - Authenticated: Yes
 - Method: POST
 - Path: /test/start
 - Json In: { String: patientID, Number: testID }
 - Json Out: { Number: testSubmissionID, Array: questions }

Sprint 2 Goals

Client

Task	Notes	Assigned To
Fix Client Side Audio	Potential Fix	Justin, Jesse
Make Layout Responsive		Jesse, Emily B
Exam Page	Draw Lines, Canvas . Communicate Results to backend. Proceed to next question	Emily B, Justin
Downloads Page	Request items to list from backend and display. Provide button to download result as csv	Justin, Emily B
Signup Page		Justin
Login Page		Justin
Home Page		Jesse
Initial Vuetify Integration		Justin
Continuous Integration		Jesse

Server		
Task	Notes	Assigned To
Endpoint: signup	Call function to create examiner, give access to site	Emily F, Jesse Malinosky
Endpoint: login	If { String:uName, String:Password} values check out, allow user access to site	Emily F
Endpoint: get_test	Return The questions for a	Alexander, Jesse

	given test and instantiate a new test submission	
Endpoint: get_patients	Return a list of all the patients for a given examiner	Alexander
Endpoint: get_patient		Aidan
Endpoint: get_result		Aidan
Confirm Working Database		Jesse
Authentication	Spring Session , Spring Security , Hash + Salt Passwords	Aidan, Jesse
CSV Export Service		Aidan, Jesse, Alexander, Emily F
Voice Transcription Service	Confirm that voice recognition endpoint uses received file, Set file size limits, Liberally set upload limits to prevent DoS,	Aidan
Audio File Storage		Jesse
Continuous Integration		Jesse