Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

646-21 Visual Acuity Screening Tool Technical Guide

December 2022

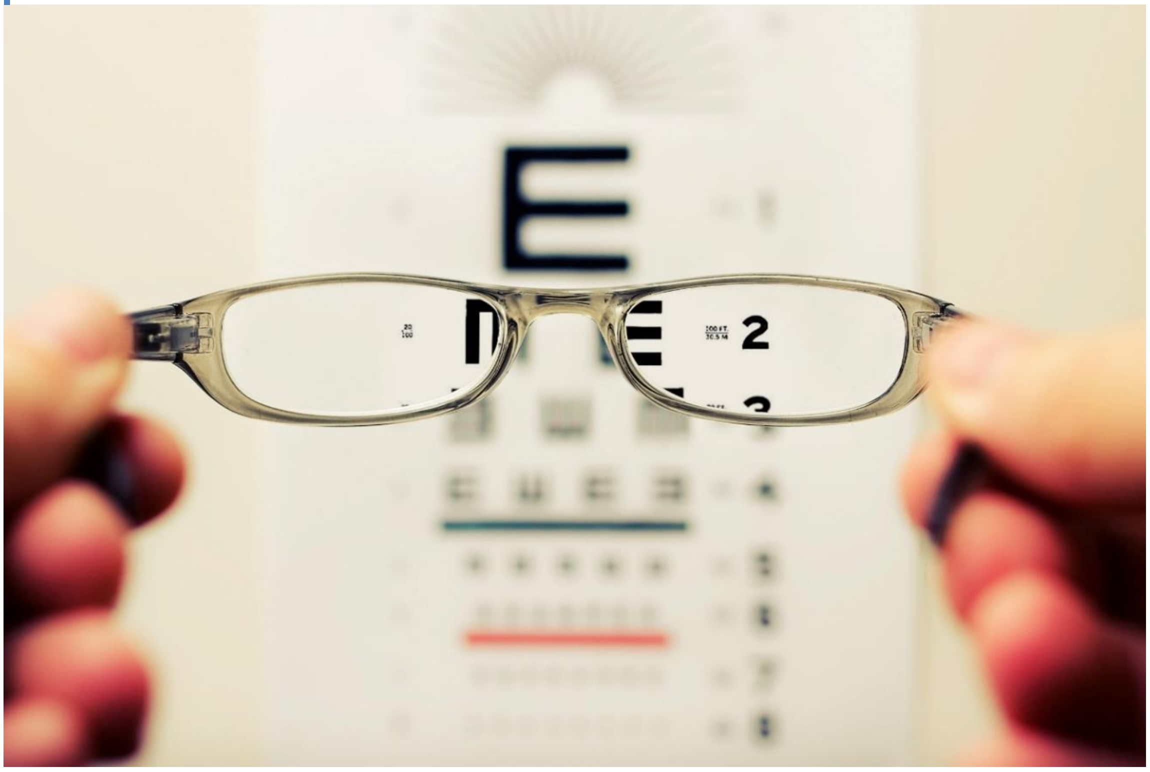


Photo by David Travis on Unsplash

Authors: Gustin Gwenaëlle, Hagenbuch Matthieu, Tapparel Océane, Flament Quentin

Professors: Gard Sébastien, Cotting Alexandre

External Client: Steinfeld Katia

Table of Content

1. Description

A. Project

Visual Acuity Screening Tool is an application designed to perform vison test digitally.

There is a main screen where a stimulus (a Landolt C) appears in a defined angle and position, and a second screen used as a controller to select the desired answer with four buttons.

The results must be stored in a database and the application must have the possibility to run offline.

B. Developer Team

The team behind this project is composed of 4 students from the HES-SO:

Gwenaëlle Gustin	gwenaelle.gustin	gwenaelle.gustin@students.hevs.ch
Océane Tapparel	oceane.tapparel	oceane.tapparel@students.hevs.ch
Matthieu Hagenbuch	matthieu.hagenbuch	matthieu.hagenbuch@students.hevs.ch
Quentin Flament	Qflament	quentin.flament@students.hevs.ch

C. Versioning

Version	Who	When	What
1.00	Flament Quentin	15.10.2022	Creating the template
2.00	Tapparel Océane	12.11.2022	To be completed according to sprint 1 and 2
3.00	Hagenbuch Matthieu	12.12.2022	Refactor + add updated content

2. General Information

A. Important notice

- Browser

The project was built and tested with **Google Chrome** as browser.

Despite the fact we have tested the project on Mozilla Firefox too, we highly recommend that you use only Google Chrome to run the application.

- Algorithm

The algorithm used in the project is implemented with all the default parameters. The results are scaled to match the criteria (a value between -0.3 to 1).

To continue this project, the algorithm must be configured and controlled in the right way to work properly.

B. Technology

- ### Language

The choice of technologies involved a webapp with a mandatory offline operation.

After studying the possibilities, we have chosen to use **React JS.** React allow to build a progressive web app, which allows us to use the local storage of the browser to work offline.

Node.js is used as the runtime environment and must be installed to use React.

- ### IDE

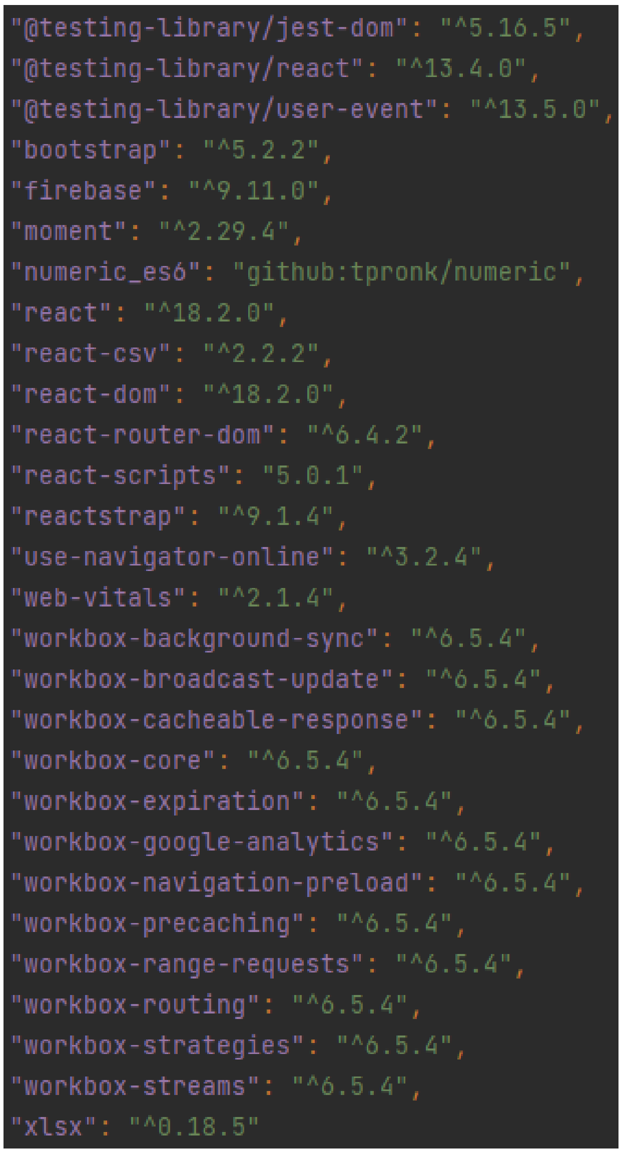
All the developers are using Webstorm and are using npm as a node package manager.

- ### Project creation

The project was created with a template that contains the offline functionalities via the command:

```
npx create-react-app my-app --template cra-template-pwa
```

- ### Libraries



```
"@testing-library/jest-dom": "^5.16.5",
"@testing-library/react": "^13.4.0",
"@testing-library/user-event": "^13.5.0",
"bootstrap": "^5.2.2",
"firebase": "^9.11.0",
"moment": "^2.29.4",
"numeric_es6": "github:tpronk/numeric",
"react": "^18.2.0",
"react-csv": "^2.2.2",
"react-dom": "^18.2.0",
"react-router-dom": "^6.4.2",
"react-scripts": "5.0.1",
"reactstrap": "^9.1.4",
"use-navigator-online": "^3.2.4",
"web-vitals": "^2.1.4",
"workbox-background-sync": "^6.5.4",
"workbox-broadcast-update": "^6.5.4",
"workbox-cacheable-response": "^6.5.4",
"workbox-core": "^6.5.4",
"workbox-expiration": "^6.5.4",
"workbox-google-analytics": "^6.5.4",
"workbox-navigation-preload": "^6.5.4",
"workbox-precaching": "^6.5.4",
"workbox-range-requests": "^6.5.4",
"workbox-routing": "^6.5.4",
"workbox-strategies": "^6.5.4",
"workbox-streams": "^6.5.4",
"xlsx": "^0.18.5"
```

Libraries details :

@testing-library/*:	Generated by default
Bootstrap:	Frond-end framework to manage the UI (must be installed for Reactstrap)
Firebase:	To use Firestore database and other Firebase tools
Moment:	Date formatting
Numeric:	Use for an optimized calculation for the JSQuest+ algorithm (Kuroki, D., & Pronk, T. ,2022)
React:	Generated by default
React-csv:	Added for csv export
React-dom:	Generated by default
React-router-dom:	Added to navigate between pages
React-scripts:	Generated by default
Reactstrap:	Use for the user interface (css)
Use-navigator-online:	Detect when the browser is online or offline
Web-vitals:	Generated by default
Workbox-*:	Those libraries are generated by default with the progressive web app creation. They are used to communicate with the local storage and allows the app to work offline.
Xlsx:	Spreadsheet data parser and writer for the export in xlsx format

C. Code Repository

The source code was upload on GitLab. Repository link: here

The database is managed with Firestore from Firebase.

The only file that is not on the online repository is the env.local file, the app need it to work with Firebase (details in the next chapter).

The technical guide (this) and the user guide are available on the Git repository too.

For all access requests, you can send an email to gwenaelle.gustin@students.hevs.ch.

D. User and Password information

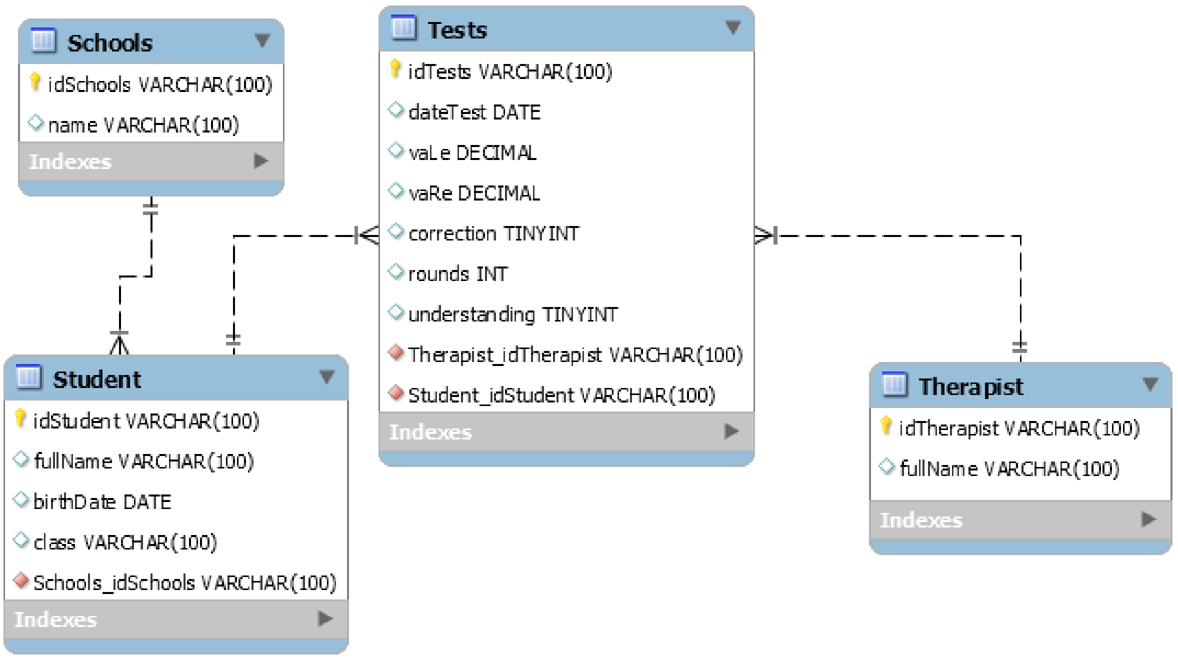
There is no authentication system implemented yet. Although the database is designed for a future feature.

We imagined that the login would link the visual tests to the connected therapist. Therefore, the tests are currently linked to the default therapist Katia Steinfield.

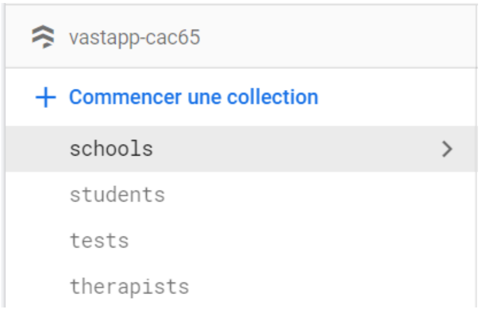
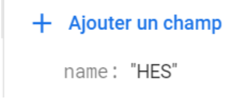
3. Data Model and Firebase

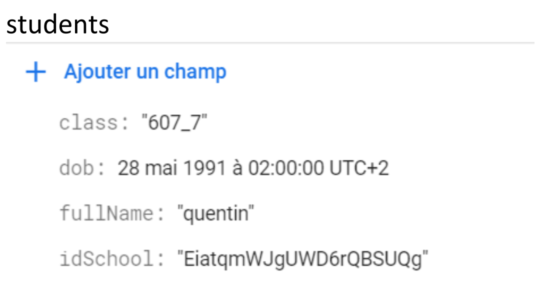
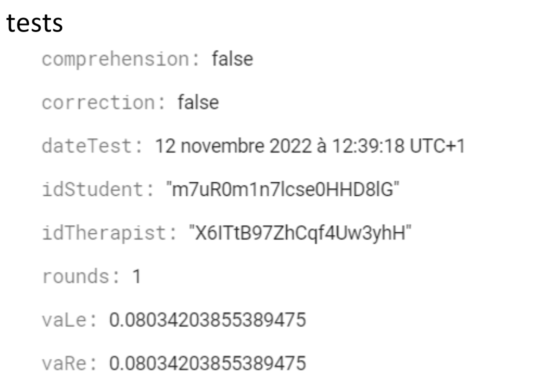
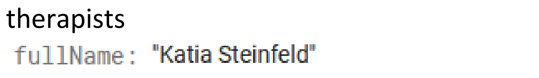
A. Design

Our data model has been designed with 4 different tables as the following way.



In Firebase :

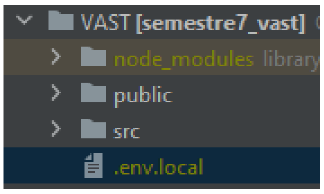
Element	Description
Collections  vastapp-cac65 / Commencer une collection / schools / students / tests / therapists	The model of the application contains 4 collections of documents - schools - students - tests - therapists
schools  Ajouter un champ / name : "HES"	- name (String)

students  Ajouter un champ class: "607_7" dob: 28 mai 1991 à 02:00:00 UTC+2 fullName: "quentin" idSchool: "EiatqmWJgUWD6rQBSUQg"	- class (String) - dob (timestamp) - fullName (String) - idSchool (String)
tests  comprehension: false correction: false dateTest: 12 novembre 2022 à 12:39:18 UTC+1 idStudent: "m7uR0m1n7lcse0HHD8lG" idTherapist: "X6ITtB97ZhCqf4Uw3yhH" rounds: 1 vaLe: 0.08034203855389475 vaRe: 0.08034203855389475	- comprehension (boolean) - correction (boolean) - dateTest (timestamp) - idStudent (String) - idTherapist (String) - rounds (number) - vaLe (number) - vaRe (number)
therapists  fullName: "Katia Steinfeld"	- fullName (String)

B. Implementation

- Add env.local

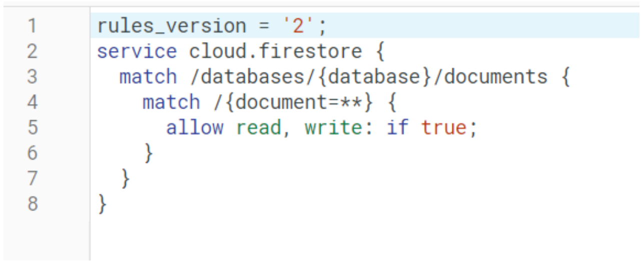
The env.local file that contains the configurations must be added in the project folder



- Rules

All the documents in the database can be read and write

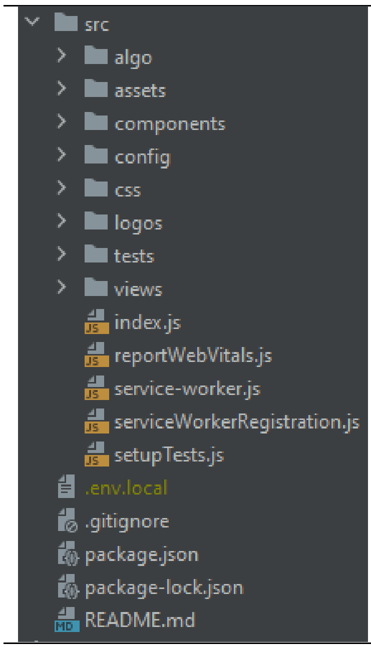
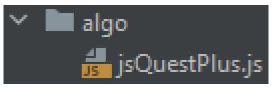
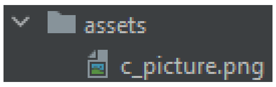
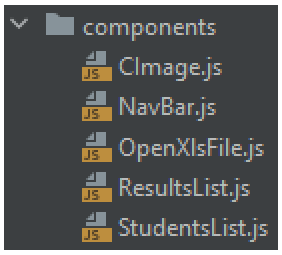
This configuration can be edited in https://console.firebase.google.com/project/vastapp-cac65/firestore/rules

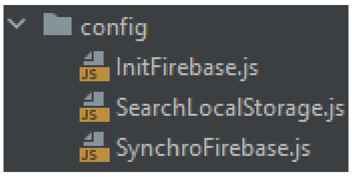
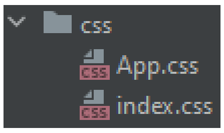
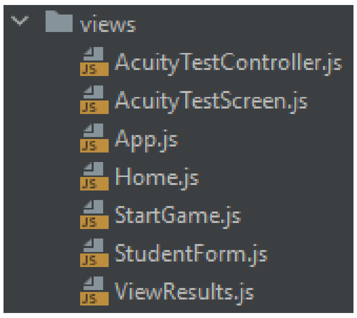


```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if true;
    }
  }
}
```

4. Code architecture

Our code is contained in a source file with different sub-folders

src folder	Description
 src / algo / assets / components / config / css / logos / tests / views / index.js / reportWebVitals.js / service-worker.js / serviceWorkerRegistration.js / setupTests.js / .env.local / .gitignore / package.json / package-lock.json / README.md	The files at the root of the src folder were generated with the project creation command. - **index.js:** where React communicate with the DOM and call the App view - **reportWebVitals.js:** By default with a React app - **Service-worker.js:** By default with a progressive React app, allow the app to run offline using the local storage - **Service-workerRegistration.js:** By default with a progressive React app, register and configure the service-worker.js - **SetupTests.js:** No used yet, we didn't delete this file for a future usage
Sub-folders	**Description**
src/algo  algo / jsQuestPlus.js	Contains the algorithm to determine the size of the C and return the results. This algorithm was developed by Kuroki, D., & Pronk, T. (2022). Further information on this GitHub : https://github.com/kurokida/jsQuestPlus
src/assets  assets / c_picture.png	Contains the Landolt C image that we use in the test screen and the controller.
src/components  components / CImage.js / NavBar.js / OpenXlsFile.js / ResultsList.js / StudentsList.js	Contains our functions that can be used in other classes - **CImage.js:** Display the next C in the screen. The angle (0°,90°,180°,270°) are equally distributed and the image don't appear twice in a row in the same position. - **NavBar.js:** similar navigation bar on all pages - **OpenXlsFile.js:** Upload an Excel file to tab NewStudent in local storage - **ResultsList.js:** The list of all the results stored in the local storage. Contain a button to export in a csv file - **StudentList.js:** form that displays the list of registered students and allows the user to select a student for the game

 config InitFirebase.js SearchLocalStorage.js SynchroFirebase.js	Contains the different links to firebase - **InitFirebase.js**: the configuration to link our application to firebase and queries methods - **SearchLocalStorage.js**: queries methods to interact with the local storage - **SynchroFirebase.js**: synchronise data from local storage with Firebase
 css App.css index.css	Application design - **App.css**: general design of the application - **Index.css**: application policy
 views AcuityTestController.js AcuityTestScreen.js App.js Home.js StartGame.js StudentForm.js ViewResults.js	Different pages of the application - **AcuityTestController.js:** displays the controller screen (called in a new tab). These are four buttons with an image of C from different angles - **AcuityTestScreen.js:** main screen for the test. It displays the C and show who perform it. There is a dev method who can show all the parameters and result from the algorithm - **App.js:** constants declaration and routes definitions - **Home.js:** home page that allows the user to go and take a test or see the results - **StartGame.js:** display the list of available students for the test. There is a button to add a new student - **StudentForm.js:** add a new student to the local storage and allow to start the test with him/her - **ViewResult.js:** display the ResultList.js component

.

5. Deployment

The application is currently deployed with Netlify and available at this link:

https://magical-klepon-7470b6.netlify.app