POLITÉCNICA
"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Matemáticas e Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de

Ingenieros Informáticos

TRABAJO DE FIN DE GRADO

DIGITISATION OF AUTUMN NATIONAL SWEDISH MATHEMATICS TEST FOR STUDENTS OF FIRST GRADE

Autor: Raúl Villora Valencia

Director: Anna Levén

LINKÖPING, DICIEMBRE 2018

INDEX

List of tables

List of figures

List of Scripts

Abstract

Nowadays more and more elements of education are becoming digital. Some schools and universities only teach with digital materials, and even some universities are conducting exams on online platforms.

Currently, in Sweden, paper exams are carried out to know the degree of knowledge of the students according to the contents taught during that period.

The project, of which the author of this work has been part, aims to and has drastically reduced the time that teachers devote to these exams by developing a real-time Single Web Application as a digital test to measure the contents taught in the first semester of Mathematics of Swedish schools.

The author of this work has developed parts of the application, such as: understanding and knowing the needs of users, configuring the architecture for the deployment of the application, developing different React components with the content of the questions, developing the structure of the flow of test questions, send to the server and save in the database the input of the components, solve the problems that arose when performing the test of the application with 300 students. Analysis of the data has not been included in the project.

This project has developed the first digital version of the *Autumn National Swedish Mathematics Test* for students of the *First Grade* and presents ideas for future projects such as the development of the teacher's interface for this kind of digital tests.

Resumen

Hoy en día, cada vez más elementos de la educación se están volviendo digitales. Algunas escuelas y universidades sólo enseñan con materiales digitales, e incluso algunas universidades están realizando exámenes en plataformas en línea.

Actualmente, en Suecia, se realizan exámenes en papel para conocer el grado de conocimiento de los alumnos en función de los contenidos que se imparten durante ese periodo.

El proyecto, del que ha formado parte el autor de este trabajo, pretende y ha reducido drásticamente el tiempo que los profesores dedican a estos exámenes mediante el desarrollo de una *real-time Single Web Application* como prueba digital para medir los contenidos impartidos en el *Primer Semestre de Matemáticas* de las escuelas suecas.

El autor de este trabajo ha desarrollado partes de la aplicación, tales como: comprender y conocer las necesidades de los usuarios, configurar la arquitectura para el despliegue de la aplicación, desarrollar diferentes componentes en *React* con el contenido de las preguntas, desarrollar la estructura del flujo de preguntas de prueba, enviar al servidor y guardar en la base de datos la entrada de los componentes, resolver los problemas que surgieron al realizar la prueba de la aplicación con 300 estudiantes. El análisis de los datos no se ha incluido en el proyecto.

Este proyecto ha desarrollado la primera versión digital de la *Prueba Nacional de Otoño de Matemáticas de Suecia* para estudiantes de *Primer Grado* y presenta ideas para futuros proyectos como el desarrollo de la interfaz del profesor para este tipo de pruebas digitales.

CHAPTER 1

1. INTRODUCCTION AND PROBLEM DESCRIPTION

The lack of knowledge about the effectiveness and level of precision in the traditional methodology for measuring the degree of knowledge in Mathematics has resulted in a wide variety of tests in Sweden. The Swedish National Agency for Education is the organisation in charge of the creation of those tests [1].

Performing these tests on paper requires a considerable amount of time due to tasks such as correcting the answers one by one, the student's handwriting and the loss of exams.

This time is even more significant in the early grades where the paper exam is conducted verbally in individual sessions since students are learning to read and write [2]. Therefore, this can lead to human errors during their development because the teachers take the results of the questions by hand. Furthermore, as the implementation of these paper tests requires a considerable amount of time and effort for their planning, implementation and correction, this makes it difficult to evaluate and monitor student progress more frequently.

On the other hand, it is fundamental to take into account that since there is no standard for carrying out paper tests, not all teachers perform the same test in the same way [2]. For example, while some teachers motivate and encourage the student during the exam, others provide a better explanation of each question. This means that not all students are under the same opportunities to take the test. Therefore, a standard would allow to define and clarify how the tests should be, and it would not result in different versions of the same test.

Nor is there a standard that defines the correct answers to each question. Consequently, the correction of each question depends on the criteria of each teacher, so the same answer has a different validation depending on the teacher [2]. As a result of the lack of standards, there is an inequality in the way students are assessed.

It is primary to bear in mind that all these factors determine the result obtained in the test by the students.

To sum up, it could be concluded that the implementation of the paper test requires an enormous amount of time and effort for their preparation, realisation and correction. This makes it difficult to evaluate and monitor the student's progress more frequently.

CHAPTER 2

2. OBJECTIVES AND METHODOLOGY

2.1 Approach

The Main Project of the start-up *Scientific Ed Tech* will focus on reducing and eliminating the problems described above. In order to achieve it, these are the main objectives of this Major Project:

- Reduce the efforts and time teachers spend in the traditional paper test
- Standardise the way tests are carried out
- Measure the correlation between manual and digital tests to determine the quality of the digital test
- Standardise and improve the analysis of test results

Below are the primary goals which are going to allow the Main Project of the start-up *Scientific Ed Tech* achieves its goals:

- Reduce the efforts and time teachers spend in the traditional paper test
- Create the first version of the digital test

To accomplish those goals, the author of this work will be involved in the development of a real-time Single Web Application at the start-up *Scientific Ed Tech*. The application is going to be a digital test which will be part of the Main Project of the start-up. This test will consist of a series of questions whose content will be based on the material taught in the *First Grade of Mathematics in Swedish* schools [2]. By performing this test, the Main Project of the start-up is going to be able to develop an analysis of the level of knowledge of the student in the different contents, allowing him/her to get better in those areas in which he/she shows the most significant lack and enhancing the rest.

The application is going to be a web application instead of desktop application since the first one provides more advantages with than the second one. First of all, the web application avoids the process of deploying the application in each client machine so each time the application is deployed by using *Google Cloud Platform* services, there is no need for the user to update the app as all users are always using the last version available. Another reason to take into account is that the application will be platform independent. Besides, it will be accessible from anywhere.

The choice of a *Single Page Application (SPA)* instead of a traditional web application or a multi-page application is mainly because a *SPA* does not require reloading the page during its use and this improves the user experience. In addition, from a technical perspective, the application is going to be using *Socket.IO*, a *JavaScript* library for real-time web applications, which enables bi-directional communication between the server and the client as well as between the server with *Google Speech API* in real-time. The socket is going to be created just after the authentication is completed which means the user and password are verified. Afterwards, one socket is going to be created for each user that has logged. If the application is a *SPA*, there is no need to reload the page. Thus, there is not going to be a socket disconnected which implies the communication between client and server will remain working. Otherwise, if it were instead a *Multiple Web Application*, once the user has logged the page will be reloaded in order to display the next content in the application. For example, the first question or instruction will be presented after the user has logged, which requires reloading the page. Therefore, the socket will be disconnecting, all the information will be lost, and the user will be logged out [3]-[5].

For the front-end development, the application is going to use *ReactJS* since it is one of the most popular frameworks nowadays. It was created by *Facebook* in 2013 and is continuously developing the open-source library [6]. Moreover, it has a large community which has been using the framework for a while, providing considerable documentation which is one of the main reasons for using *ReactJS* instead of other frameworks.

Besides, the server and the client of the application are going to be developed by using *JavaScript* language since the application is going to use *Node.js* which is a cross-platform *JavaScript* run-time environment [7]. Another benefit of using *Node.js* in the app is that it could also use the *NoSQL* database *MongoDB*. *MongoDB* provides an easy way to scale the application, and it also removes the complex object-relational mapping layer by allowing to create a flexible data model [8].

The application is going to be hosted in the *Google Cloud Platform* not only for of all the services it offers, more than 50 services including *Google Speech to Text* but also because it has pretty good official documentation of its products [9]-[11]. With the *Google Speech to Text API*, there will be able to know in real-time a transcription of the answer of the question, saving them for later showing the results of the verbal question to the teacher without the teacher having to ask and listen to the response of all the students. This aim to cut around between 60% to 70% of the time the teacher will have to spend if the test will have made in the paper, allowing the digital test to perform multiple time during the week.

The application is going to use *Docker*, as it will allow to separate the app from the infrastructure and handle as a controlled application. Furthermore, it will also contribute to the faster sending, testing and deployment of the code, as well as reducing the time between writing and executing code. Moreover, it will do this by blending the core containerisation functionalities with workflows and tools that will help manage and deploy the application. Last but not least, *Docker* containers can be used directly in *Kubernetes*, which enables them to run easily in the *Kubernetes Engine* [12]. *Kubernetes* is going to allow that a large number of containers work together, running container across a wide variety of machines, scaling up or down by adding or removing container on demand, keeping storage consistent with multiple instances of the application, launching new containers on different machines if something goes wrong, among others benefits [13][14].

The flow of questions is going to be divided into blocks. These blocks will represent a series of questions that measure one or more particular skills of the user. These blocks are also divided into three categories, as is done in the paper exam. In the traditional paper test, all students will start in the middle depending on their score the students will move on to the proper level. Meanwhile, in the digital test, the student is going to have to answer all the levels of all the questions, which implies more data will be gathered from each student. Hence, this will determine better accuracy in the measurement of the student skills.

On a technical level, these blocks are going to be *React* components which will help to develop the question flow and the randomisation of the questions, so the odds of two students starting with the same question will be low.

Moreover, the digital test is going to be divided into two blocks: the *Tap* and the *Verbal*. The first one is going to be related to *Tap* response format and the second one with the *Verbal* response format.

For developing those formats, there is going to be used two main the *React* components: The *Keyboard* component, which is going to use a keyboard-line as the way to record the response. The Speech component which is going to register the answer and get a transcript of it by using the *Google Speech to Text API* and that transcript is going to be the answer.

By using these components, there is going to be reduced the time the teacher spends in the individual session with the students, without having to do any one-by-one session with them.

Intending to understand each question and to know how to respond the different answer formats, all the students who are going to start with the same type of response will begin by following a series of instructions. By doing this, they will be able to understand precisely how to answer correctly to each question and what is explicitly asked of them at each moment. In the future, this is the way *Scientific Ed Tech* would like to be the standard way of doing the test.

In order to test the application, tests will be carried out with real users on a weekly basis. These tests will allow us to know what the user experience is like from the comfort in the use of the application to the absence of functionalities. Also, it will enable to identify the problems that arise when these tests are being carried out.

This application will be tested in the week of November 6th by an estimated number of 300 First Grade Mathematics students from *Swedish* schools.

One of the aims of that week is to obtain enough data so that the Main Project at Scientific EdTech can measure the correlation between paper and digital tests. In the week before November 6th, around 100 of the 300 students will also perform the paper test, and then the correlation of both tests will be determined by the Main Project at Scientific EdTech.

2.2 Test Content

One of the objectives of the Main Project of the start-up *Scientific Ed Tech* is to measure the correlation between traditional tests and digital tests.

With a view to achieving this goal, it is essential to recreate the eleven questions that constitute the traditional test in digital format. The table of content of the paper test (See Table 1 in Appendix A) shows the different levels of each question with its corresponding content. The answer format in all the questions in the paper test is verbal.

The author of this project has focused on the traditional paper examination that takes place in the fall. Questions 6 and 10 are included in the table of contents of the paper exam as they are only performed in the *Spring* exam (See Table 1 in Appendix 1) [2].

In the table of contents of the paper test (See Table 1 in Appendix A), the questions and content of the paper test were shown in the fall. In order to transfer the contents of the previous table (See Table 1 in Appendix A) to a digital format, this project has had the collaboration of Martin Hassler Hallstedt, PhD, CEO and founder of Scientific EdTech. Martin has worked on the creation of the digital test. To do so, Martin has reconstructed the traditional paper test into a digital format, keeping the question but enhancing its content.

This new table (See Table 2 in Appendix A) shows the number of the question together with the level to which it belongs, the number of items that each question will include and the response format in the digital test.

2.2 Build the requirements

For the purpose of meeting the objectives and develop the test in digital format, it is essential to specify the requirements. It must be considered that during the realisation of the project new requirements will arise, and some of the established ones could change:

- Authentication, registration, and management of users in real-time.
- Storage of the results, of the realisation of the test, in real time.
- Conducting the test with the content of the questions and the different response formats as described (See Table 2 in Appendix A).

2.3 User stories

Once the general requirements of the web application are known, it is necessary to know the needs of the users for the further development of the user interface. Knowing the needs of users allows creating friendly, simple user interfaces and where the user can operate naturally. Several sessions of *user stories* have been made to understand the user needs correctly.

A *user story* is a brief description of the functionality from the user's perspective. The story is told from the user's point of view and describes the feature, event, or functionality in a simple language, ending with the primary purpose of the feature. The user stories facilitate the use of the application by users, and it will help in the creation of project requirements [15].

In the creation of the *users' stories*, the collaboration of several members of the start-up and the feedback from teachers have been told. The latter has reported on how the tests are carried out on paper, allowing the development of a digital version of the same.

Through the use of user stories, there have determined the future requirements of the user interface that will cover the needs of users. To do so, it is necessary to put in the shoes of both users and describe in detail what kind of needs arise during the development of the test. For example, teachers need to have control of what is always going on in the classroom or the needs that arise for students when taking the test.

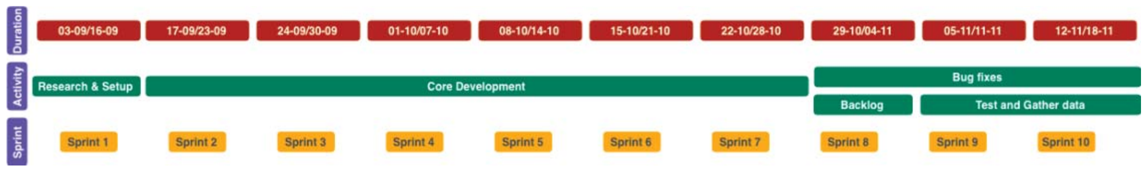
The *user stories* have been made of by both users (See Table 3 in Appendix A; Table 4 in Appendix A). However, the author of this project will work in the development of the student user interface.

After several sessions of *user stories*, several requirements have been found that include both the user interface of the student and the teacher during the different steps in which the process of conducting the test can be divided (See Table 3 in Appendix A; Table 4 in Appendix A).

2.4 RoadMap

Once the general requirements of the application and the requirements of the user interface are known, the next step is to organise the tasks to meet the requirements. For this, the project is going to use the *Scrum* methodology with weekly sprints (See Figure 1 RoadMap). This allows having a better record of the tasks that have been developed during the project, increasing the value of the project, and allowing us to have better control over it. In addition, it allows having adaptive planning, an evolutionary development, and the continuous improvement of the product.

Figure 1 RoadMap



The first sprint has a longer duration than the rest of the sprints. This is due to the fact that it is one of the most essential and critical phases in the development of this project since it is necessary to take into account that in the first sprint the project begins to be built and the ideas about its structure clarified. In this first sprint, it is essential to carry out an in-depth investigation of the tools, frameworks, programming languages, libraries and other resources that are necessary to create the architecture of the project. It is also essential to understand the advantages they offer one over the other, always taking into account the requirements of the project when making the decision. After that, the architecture that allows deployments of the application will be built.

From the second to the seventh sprint, the rest of the application will be created: server, client, communication between the both, calls to other APIs, among others. In addition, internal tests of the application will be carried out both by the developer (project's author), as well as tests with a reduced number of users.

The eighth sprint focuses on correcting errors and making a list of the pending tasks of the previous sprints called backlog as well as perform the tasks of that backlog.

The ninth and the tenth sprint are still correcting errors but will also test the application with a total of 300 users, and all the data generated during this test will be gathered in a CSV file.

CHAPTER 3

3. DIGITAL TEST DEVELOPMENT

3.1 Defining the digital test architecture

The software architecture plays a crucial role in this project. For this reason, the architecture design is made by keeping in mind the goals described in the previous chapters.

At first, it was considered whether the final architecture of the application (See Figure 2 Final architecture) was going to be for a web application or a native application. For this, first, there had to know what kind of application was going to be developed.

The advantages of developing an application were that development would be quicker and save cost. Also, deployment, distribution and updates are more comfortable because there is no need to update the app in an app store which can take three days or more, the changes will be immediate. Moreover, the web application will be accessible on all different platforms.

Although, some native capabilities of the devices will not be able to use anymore, and the application would only work if there is an internet connection. In addition to that, there would be the risk that certain features of the application did not work correctly in the application [16].

Finally, it was decided that a web application would be developed due to the short time available to create the application, as well as the average revision time, which is more than three days. This last one is one of the main advantages that this application needs. Since if a test is performed and some errors happen, these problems can be solved, the application updated and immediately after deploying the new version, so the test can continue immediately after it is fixed [17].

The next step is to know how the architecture of the application will be. For the client, since it is going to be a web application, there are several alternatives among the possible frameworks to use. Among all of them, select the three most popular currently: *Vue*, *React* and *Angular*. Finally, it was decided to use the *React* framework as it is easy to learn thanks to its simple syntax, the high level of flexibility and responsiveness, the downward data binding and the virtual *DOM* which allows arranging documents in different formats [18].

Meanwhile, not only the learning curve of *Angular* is far sharper but also the area covered by the *API* framework is massive, meaning that users must familiarise themselves with a higher number of concepts before becoming productive. *Angular's* complex nature derives heavily from its design goal of only aiming at large and very complex applications. Nevertheless, this makes the framework more challenging for the least experienced developers [19].

Otherwise, *React*, and *Vue* has many similarities, as both use a virtual *Document Object Model*, both give active and modular components and keep the focus on the main library. Although, *Vue* in not as widely used as *React* as the second one is used among top technological leader. Besides, *Vue* lacks full *English* documentation [18]-[20].

As for back-end, there were two options for building the server: *JavaScript* using *NodeJS* or using *C#* with *ASP.Net*. *NodeJS* allows to develop a web application in a single language, *JavaScript*, which helps reduce the context shift involved in the development of client and server as well as allows to easily share code among client and server, e.g. reusing the same code. In addition, *JavaScript Object Notation* is an open-standard file format and is native to *JavaScript*, which is the language employed in several *NoSQL* databases, so interfacing with them becomes a matter of course. Besides, *NodeJS* is much lighter than *C#* code. However, with *NodeJS* there is no control over the compilation process [7].

Regarding the choice of the database, there are two options: either *NoSQL* or *SQL*. It was chosen that it was No*SQL* due to the short time the project had to develop and thus *NoSQL* avoid the complexity of creating the structure of the data as well as the rapid scalability of the databases of the scale. Among all the *NoSQL* databases, *MongoDB* was chosen primarily for being the leading database among the *Fortune 500* and *Global 500* companies [21]. In addition, MongoDB and Node.js are usually combined since both shared the use of JavaScript object notation.

Furthermore, *Mongoose*, which is an *ODM* (*Object Data Modelling*) library, gives an accurate modelling environment for the data. Also, Mongoose strengthens the structure of the data and also wraps the Node.js native driver which is one the ways to connect *MongoDB* and *Node.js*. Besides, the primary purpose of adding Mongoose to the project is because it allows defining schemas for the project collections. Those collections will be enforced at the *ODM* layer by *Mongoose* [22].

To create a desktop application in the browser can be used *Comet* and *Ajax*. To use either solution, one needed to rely on server and client hacks the same way in order to maintain open sockets for a long time and forge a long-lasting connection. Although, this leads to resource allocation problems on the servers. In addition, long polling sends unneeded queries and have a constant flow of open and closed connections for servers to handle. One solution would be to place additional protocols over *Comet* or *Ajax*, but simplicity is not in that solution. However, the use of *WebSockets* offers the ability to use an updated HTTP request and send data based on messages, which means using a single connection, and the ability to send data back and forth between client and server. Also, to provide a more secure of communication, there is the possibility to layer another protocol on top of *WebSockets* [23]. A few years ago, the majority of the browser did not support *WebSockets*. However, today there is 93.18% of users that have a browser compatible with *WebSockets* [24].

Modern browsers can use *WebSockets* to manage communication between the browser and the server. In addition, *Socket.IO* provides an abstract layer on *WebSockets* as well as other *JavaScript* transports on the client and node side. Moreover, this technology will transparently resort to other *WebSockets* solutions as long as *WebSockets* is not implemented in a web browser and the same *API* is maintained.

As it is mention before with *WebSockets*, it can establish a persistent connection, which allows bidirectional communication between client and server. An alternative is to use *REST* based *APIs* which are built upon *HTTP*, where the client requests a resource or page, and the server responds. However, in that scenario, there is nowhere for a server to forward data without the client requesting it first. This means the client will have to keep up asking for changes in regular intervals, polling, which is not real-time communication. By using *WebSocket*, there is the feature of a broadcast new message received to every client connected [25].

Socket.IO is a *WebSockets API* which uses feature detection to choose whether the communication is going to be made with *WebSocket, AJAX, long polling, Flash*, … allowing to create real-time apps which work everywhere a snap. It also provides fail-overs to other protocols if WebSocket is not supported on the browser. Moreover, it streamlines the *WebSocket API* and combines the *APIs* of its fallback transports which includes *WebSocket, Flash Socket, AJAX long-polling, AJAX multipart streaming, IFrame* and *JSON polling.*

Socket.IO does not provide a fallback mechanism to long polling when *WebSockets* are not available. Instead, the fallback to long polling is provided by a protocol where *Socket.IO* is in the top of, *Engine.IO* which provides multiple underlying transports, in order words WebSockets and long polling. This allows it to deal with different browser capabilities as well as deal with different proxy capabilities, with seamless switching between transports. Furthermore, it gives an *API* for *Node.JS* which seems way more as the client-side *API.*

This server-side approach enables both client-side and server-side APIs to be unified. By using Node.JS, it can create a standard HTTP server and then transfer the server instance to Socket.IO where it can be created the connection, disconnect and send messages to listeners in a similar way as it is done on the client side. [26].

Redis, which is an in-memory data structure store, has multiple applications and supports a wide variety of data structures [27].

Redis, Socket.IO and *Node.JS*, allow this project to achieve the benefits of enforcing a single web socket connection per user. This allows that each user will be login to test application just once until that user disconnects then that user can log in again. This is going to be possible by using *Redis* to lock and unlock resources: the user sessions [28].

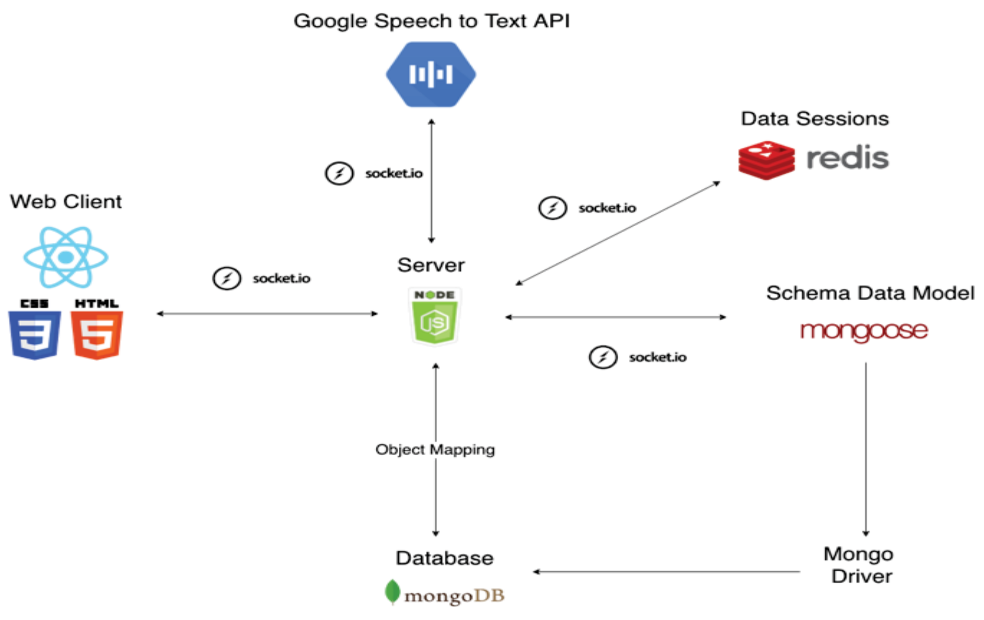
Docker is an accessible technology for containerising the application. *Docker* provides a container which is the standard unit of software where it packages up the code and the entire collection of its dependencies. This provides the application to run quickly and steadfastly in every computing environments. *Docker* also provides container images which are lightweight, able to operative independently from other hardware or software and an executable package of software which contains the entire thing required to execute the application including the code, a runtime, libraries, environment variables and configuration files. *Docker* will allow to combine, develop, deploy, scale and administer containers throughout *Docker* hosts [29].

Kubernetes is a software system which provides easily deploys and manage containerised applications on top of it. *Kubernetes* relies on the features of *Linux* containers to run applications without the need of having any details of these applications and without having to deploy these applications in every host manually. Besides, it enables to deploy the application always in the same way even if the cluster contains just one node or millions of them. Besides, the *Docker* containers can be directly used in *Kubernetes*, which allows them to be run in the *Kubernetes Engine* smoothly [23].

Once almost the entire architecture is almost defined, the next step is to know which platform will host the application. To make that decision, one of the first things to keep in mind is how good *Speech to Text* services are on those platforms. *Speech to Text* technology uses machine learning to produce audio transcriptions, and automatic transcription of the speech context is becoming faster, cheaper and more accurate. Four main businesses offer speech to text *APIs*: *Microsoft, Google, IBM* and *Amazon*.

In the presentation made in *Saturn 2018*, it shows the degree of maturity of *Google's* service conditions has a degree of maturation greater in Discourse to the text than any of the other companies [30]. Because of this and because *Google Cloud Platform* offers proper documentation of services, reasonable price, different storage classes according to needs, high durability and good support for *Kubernetes* and *Docker* were the main factors to choose the *Google Cloud Platform* [31].

Figure 2 Final architecture



3.2 Setup the deployment system

After knowing the final architecture of the project, the next step is to set up the environment to the application so that the application can be deployed to *the Google Cloud Platform.* The first deployment will be a *React* application created from a dotnet template since in the first deployment is just needed a simple application with the same underlying architecture as the project will have. After creating the *React* application, the application has to be containerised the web application in a *Docker* image and then deploy it to the *Google Kubernetes Engine*. To do that there will be followed these steps [32]

1. Creation of the *Dockerfile*, which contains the instructions on how the image is built, by using the *Docker* documentation [33]
2. Create the image container of the application and tag it as the first version. The result is a docker image which consists of read-only layers each of which represents a *Dockerfile* instruction.
3. Upload the container image to the private *Container Registry*.
4. Run the container locally by testing the container image using the local *Docker Engine* and then preview the web application. This is step is optional, and at the same time, it provides if there have been errors in the steps behind.
5. Create a container cluster to run the container image since in *GKE* each cluster consists not only of at least cluster master but also of nodes which are multiple worker machines. Both run the *Kubernetes* cluster orchestration system.
6. To deploy and manage the app from the *GKE* cluster is necessary to communicate that with the *Kubernetes* cluster management system. The deployment will manage several copies of the app known as replicas and will schedule them to run on individual nodes inside the cluster created earlier.
7. The final step is to expose the application to the internet as the containers which are run by the *GKE* are not accessible from the Internet. To make them accessible is required that they have an external IP address which will be created by the *GKE*. *GKE* will also create a *Load Balancer* service which is the process of distributing network traffic across multiple servers, ensuring that non-a single server bears too much demand which increases the application responsiveness.

Because this process can be challenging to follow for new people if they have to do it for the first time, there has created documentation in *Confluence* about it. Also, when an update is required for the web application, it has also been documented about how to deploy a new version of the app in *Confluence*.

3.3 Development of the core of the digital test

Once the architecture is known, and new versions of the application can be deployed to the *GCP*, the next step is to figure out how to make the application to be a *Single Web Application*.

For not using routing in a *Single Web Application*, one of the alternatives is to use *React* states to show and hide different components [34].

The state in *React* is similar to props as both are plain *JavaScript* objects and hold information which influences the output of render [35]. The props are read-only which means if a component is declared as a function or a class, it should never change its props. Meanwhile, states allow *React* components to transform outflow through time in reaction to the user's actions and any other thing. Besides, the state is private, and the component entirely controlled it [32].

The application uses one state of each different component that has to display. This state allows the application to show the component that is needed, according to the question order in the table of the test content, hiding the rest of the components.

Using *React* components states helps the question test flow to be carried out in a *Single Web Application* which means there are no needs of refreshing the page.

After knowing how to make the application an *SWA*, the next step is to build the two main components of the application in *React*, which are: the component *Speech*, is the component of the questions with the verbal format, and the component *Keyboard*, is the component of questions with tap format. Both have to have a simple interface that allows the student after a brief prior explanation to quickly understand how these components work and to be focused on the test.

3.3.1 Development of the Speech component

The *Speech* component is the first *React* component that has been created and will be used in questions with the verbal format. This component arises from the need to evaluate students orally, as is done in the paper test.

It differs from the traditional format because the teacher does not have to be present either to ask the question or to take the students' answer. This allows the teacher to be aware of a more significant number of students where the test is being done. Also, the answers are digitalised, and the process of proof correction is automated. This affects the time the teacher uses to correct it.

3.3.1.1 Speech component Version 1

Before having performed internal tests of this component with users, two versions were created: the *Click* version and the *Hold* version, with the same functionality. The difference between both lies in the user's interaction with the component.

In both versions, the question that must be answered is made external to the application.

3.3.1.1.1 Click Version

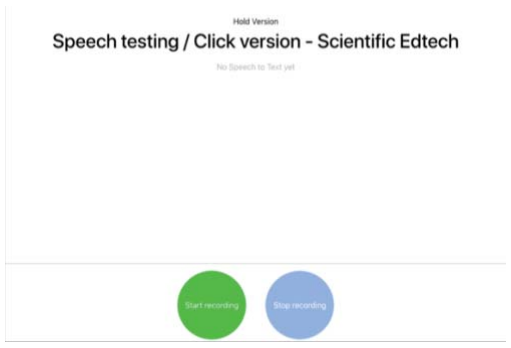
While the question is being asked externally to the application, the user will see the image interface (See Figure 4 Speech Component Version 1 Click Version). To answer the question, the user clicks on the *green* button, after pressing it the user responds, and the response is recorded locally in a temporary file. When the user has finished answering the question, the student has to press the *blue* button.

After this, the user can re-record a new response, in case it is not by the one given, or when the visibility of the *submit* button has been activated, the user can go to the next question by pressing this last button.

Technically, when the user presses the green button, a call to *the Google Speech to Text API* starts with the data that is being recorded. A transcript of these is received in text format in real-time, along with the percentage of accuracy of each of the words that the *Google API* has captured. At the same time, this audio data is stored in a temporary audio file that will be sent to the database just when the user presses the *submit* button.

When the user has finished answering the question, the user must click on the *blue* button. This transcript will be sent to the database where it will be saved when the user clicks on the *blue* button.

Figure 3 Speech Component Version 1 Click Version



3.3.1.1.2 Hold Version

In this version, once the question has been asked, the user presses and keeps pressed the green button to record the answer, releasing it only when it has finished responding. In the image (See Figure 4 Speech Component Version 1 Hold Version) this component is shown. The technical part that gives functionality to this component is the same as that described above in the *Click* version.

In the user interface tests, the following conclusions were reached:

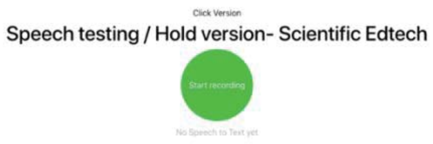
The users did not understand that the button to start recording had two different colours. The reason for using different colours was because the *Google Speech to Text API* needs at least 2 seconds to understand the context in which the user is speaking.

Also, it was noted that users started talking before the button will change colour. To solve this problem, a *pling* sound was added that tells the student that the user can start to respond.

On the other hand, the language and the answers that the users were going to respond as a context were added reducing the errors that could be produced by the speech recognition API.

After several tests with the users, it was determined that the users understand better the mechanism of the *Hold* version. Therefore, the use of the *Click* version was discarded.

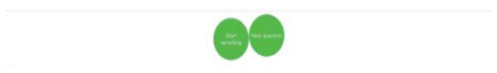
Figure 4 Speech Component Version 1 Hold Version



3.3.1.2 Speech component Version 2

In the next version, the visual format of the component was changed, changing the button to record the answer to the bottom along with the submit button. In addition, in order to simplify the interface of the component, the description of it was eliminated, as shown in the image (See Figure 5 Speech Component Version 2)

Figure 5 Speech Component Version 2

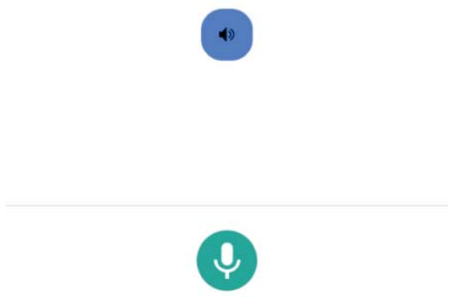


3.3.1.3 Speech component Version 3

In this version it was decided, after several internal tests with users, to introduce in this component the audio reproduction and the possibility to stop and listen to the audio again through the implementation of the audio component.

Besides, the CSS of the send answer button was changed since an arrow was more intuitive as a send response button and go to the next one than the current one, as shown in the image (See Figure 6 Speech Component Version 3)

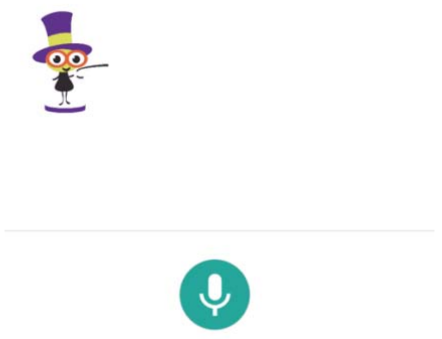
Figure 6 Speech Component Version 3



3.3.1.4 Speech component Version 4

In this version, after performing several tests with users, the absence of a component to which users will pay attention was observed. The questions were formulated in the audio reproduction, and meanwhile, the users stared at the support person who was in the test. This resulted in the loss of user attention. To solve this problem, an animated component was introduced, as shown in the image (See Figure 7 Speech Component Version 4). He simulated through the movement of the mouth that it was the amination who formulated the questions *(this animated component has not been created by the author of this work but is part of the project. The author of this project has worked on the integration of this animated component by adding in certain questions the audios and the animation match).*

Figure 7 Speech Component Version 4



3.3.1.5 Speech component Version 5

In this version, the way to use the component is changed (See Figure 8 Speech Component Version 5 without having pressed the record button; Figure 9 Speech Component Version 5 having pressed the record button). Until now the component started its functionality when it was pressed and ended when the user released it. This generated an error, which is explained in the section of *Week Test in schoo*ls. To solve it, two buttons are added, to activate their functionality the user has to click them, one to start recording the user's response and another to finish the recording of the response.

Figure 8 Speech Component Version 5 without having pressed the record button

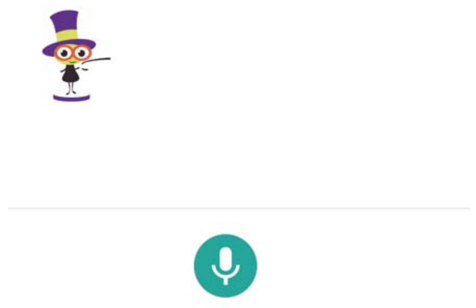
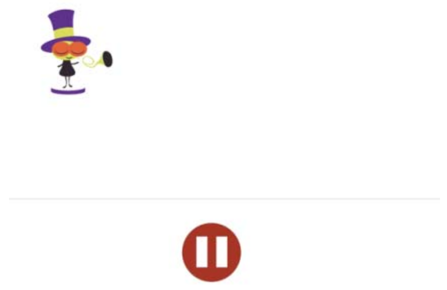


Figure 9 Speech Component Version 5 having pressed the record button



3.3.2 Keyboard component

This is the second component that has been created and arises from the need for answers to tap-type questions, that is, it will be used in those questions in which the student is required to know the symbol of the number referred to in the ask and/or know the relationship between the sound of that number and its position in the number-line.

3.3.2.1 Keyboard component Version 1

In this version, after asking the user the question, the user will be able to respond to it by pressing the button or buttons with the symbol or symbols that the user considers as a response. In addition, the user has the possibility, after having pressed at least one numeric symbol, to eliminate any of the numeric symbols that have selected. For this, the user must press the symbol that the user does not want in response. When the user considers that it is his definitive answer, the user must press the button *lämma* (See Figure 10 Keyboard Component Version 1 without any numeric symbol pressed; Figure 11 Keyboard Component Version 1 having pressed numeric symbols).

At a technical level, when the user presses a key, an array is generated with the numeric symbol that has selected, and when the user eliminates it, the symbol also does it from the array. On the other hand, when the user presses the *lämma* button to send the response, it is sent to the server and from the server to the database.

Figure 10 Keyboard Component Version 1 without any numeric symbol pressed

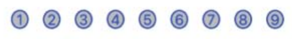
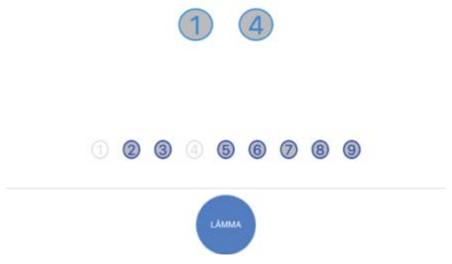


Figure 11 Keyboard Component Version 1 having pressed numeric symbols



3.3.2.2 Keyboard component Version 2

In this version, changes were made to the *CSS* of the component. When carrying out several tests with users, it was determined that they be familiar with the use of the iPad, for this reason, a keyboard was created that resembled the *iPad*.

Users are very familiar to see in the games and applications of the *iPad*, the button to send the answer in the lower right corner of the *iPad*, so it was also implemented.

In the previous version, the only way to delete a number was by clicking on it. This was not intuitive to the users. One of the reasons is that on the *iPad* keyboard there is a specific button to delete characters. For this reason, the feature of being able to eliminate the numbers by pressing, and the delete characters button was added.

With the aim that users can listen again to the question they have just been asked, the button of the audio component is added.

Figure 12 Keyboard Component Version 2 without any numeric symbol pressed

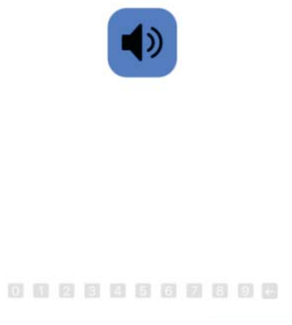
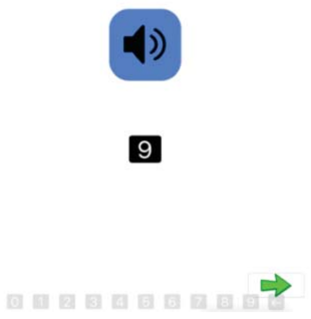


Figure 13 Keyboard Component Version 2 having pressed numeric symbols



3.3.2.3 Keyboard component Version 3

In this version, visual improvements are still being made. The keys are enlarged keeping the size of the numeric symbols which provides greater visibility.
The position of the keyboard is changed to a higher part, easily distinguishable from the send response key, which is still in the lower right part of the screen.

A dividing line is added between the keyboard and the numbers that have been selected. This facilitates the user to distinguish between the selected number and the keyboard (See Figure 14 Keyboard Component Version 3 without any numeric symbol pressed; Figure 15 Keyboard Component Version 3 having pressed numeric symbols).

Figure 14 Keyboard Component Version 3 without any numeric symbol pressed

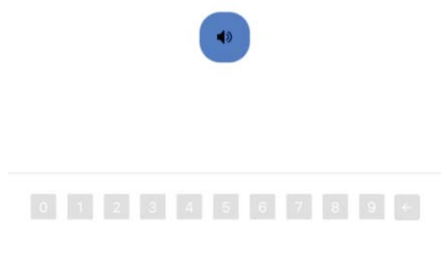
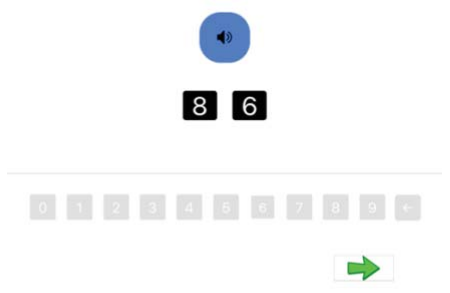


Figure 15 Keyboard Component Version 3 having pressed numeric symbols



3.3.2.4 Keyboard component Version 4

In this version, the *CSS* of the application has been restructured creating a more straightforward and cleaner interface. The button has been changed to delete the last selected numeric symbol. Instead of being an arrow has been changed to an *X* because on the *iPad* keyboard has this symbol. This change makes it easier for users to understand the functionality of this button.

It also includes an active component that takes the functionality of the audio button. For this reason, the button is deleted.

Furthermore, to prevent the user from pressing too many numeric symbols and given that the maximum number of digits that is asked is three symbols, this number was limited to the maximum number of symbols that users could select (See Figure 16 Keyboard Component Version 4 without any numeric symbol pressed; Figure 17 Keyboard Component Version 4 having pressed numeric symbols).

Figure 16 Keyboard Component Version 4 without any numeric symbol pressed

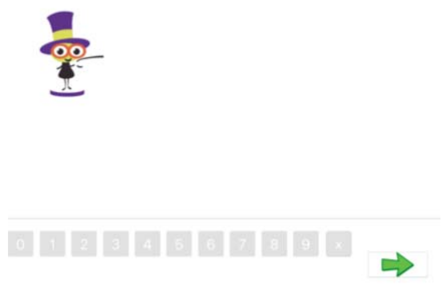
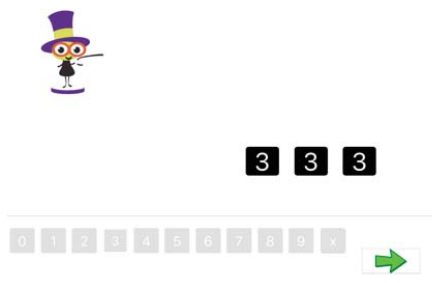


Figure 17 Keyboard Component Version 4 having pressed numeric symbols

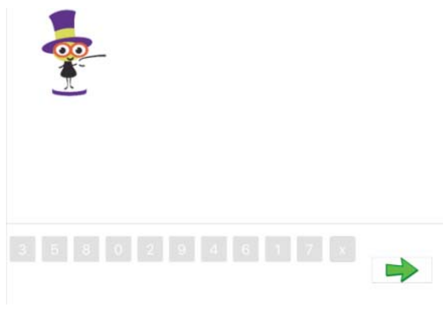


3.3.2.5 Keyboard component Shuffle Version

In order to know if the students have knowledge of the position of the symbols in the number-line, there was created a version of the Keyboard component with the number-line shuffle in this specific order: 3 5 8 0 2 9 4 6 1 7 (See Figure 18 Keyboard Component Shuffle Version without any numeric symbol pressed).

This version of the Keyboard is going to be used in the Q04 and the Q05.

Figure 18 Keyboard Component Shuffle Version without any numeric symbol pressed



3.3.3 Creation of a single instance of Socket.IO

The project needs to use Socket.IO in different components, and a way to implement this is by using Redux as Middleware and use one library for using Redux and WebSocket [36]. However, as it requires refactoring most of the project, instead there was implemented another solution which to create a single of Socket.IO in the app [37][38]. To do so there is going to build a context instance and then store it in a separate file. This will help the component to use Socket.IO easily as it will just have to import it when need it. After that in the main wrapper component a Socket.IO instance is created, and it uses a Provider to send the Socket.IO instance to the component tree, this will provide to any component access to the Socket.IO instance.

3.3.4 Starting the question flow

After having the two main components created, the next step is the creation of the first version of the question flow. To do this, the Start component was introduced (See Figure 19 Start Component). This component appears once the user has logged in and it is the waiting area where the students will wait for the teacher to give them instruction to start. This allows all students to start the test at the same time.

Figure 19 Start Component

3.3.5 Login component

One of the specified requirements was the creation of users and their authentication. To this end, an asynchronous authentication of the users was implemented on the server, and a *Login* component was created in the client (See Figure 20 Login Component). If the user did not correctly enter the password and/or the username that was going to be given to a beginning before starting the test, the user could not log in. Once the user has logged in, a socket is detected that will allow the communication of information between client and server. In case the user tries to access again from another tab, a message appears that this user is already logged in and will prevent it from being logged in again (See Figure 21 Login Component when the user is not authorised)

Figure 20 Login Component

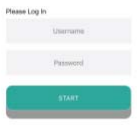
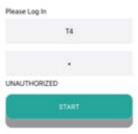


Figure 21 Login Component when the user is not authorised



3.3.6 Introducing the instructions

After several tests, there was figure out that it will be easier for the students if they have some instructions before they start in the verbal and tap part since half of them did not what to do sometimes during. Sometimes this was produced because they did not understand the difference between clicking and holding. In order to fix problems, there was decided to make an instruction on how they should proceed step by step. To do this, there were created two scripts of the instructions (See Script 1 in Appendix C; Script 2 in Appendix C).

Two new components have been built to create that script: Learning Keyboard and Learning Speech. These components will be displayed before the touch or the verbal question.

These are the Learning Keyboard instructions briefly:

While the audio is played, the user will see Keyboard and animation explaining that the user that now will learn how to click a number and encourage the user to click 3 and then the arrow (See Figure 22 First Learning Keyboard instruction is played; Figure 23 Answer of the First Learning Keyboard instruction). Then if there is no answer or the answer is not the correct one, there will play the same audio again.

Figure 22 First Learning Keyboard instruction is played

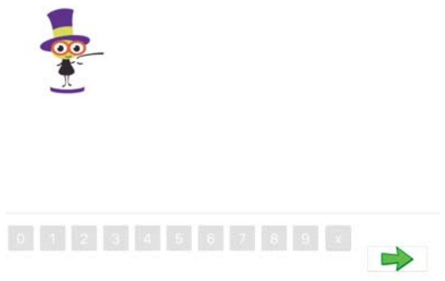
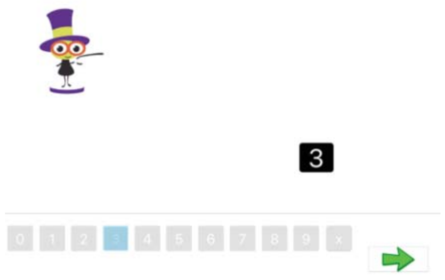


Figure 23 Answer of the First Learning Keyboard instruction



After the answer that first instruction is typed correctly the user is going to see that an eight is displayed in the screen as the same time as it is explained that if the answer the user has given can be changed. In order to make the user learn this, the user is encouraged to delete the number 8 and press 5 and if it was the new answer of the user. (See Figure 24 Second Learning Keyboard instruction is played; Figure 25 Answer of the Second Learning Keyboard instruction). Then if there is no answer or the answer is not the correct one, there will play the same audio again.

Figure 24 Second Learning Keyboard instruction is played

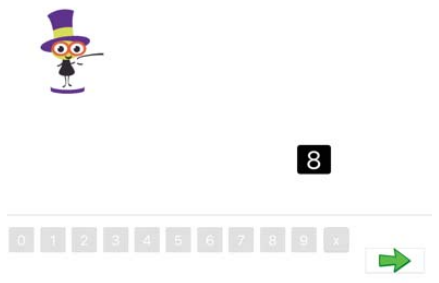
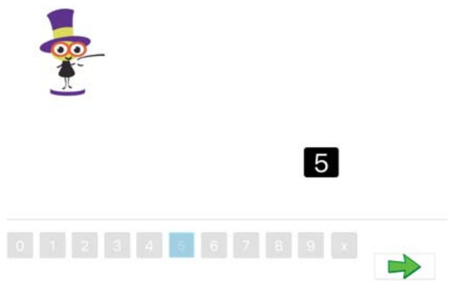
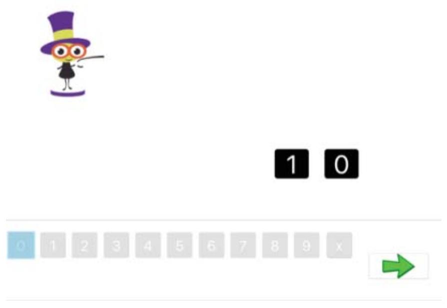


Figure 25 Answer of the Second Learning Keyboard instruction



After the answer that first instruction is typed correctly the user is going to learn that can press more than select more than one number of each question. To do this, the user is going to be encouraged to press the number 1 and the number 0. (See Figure 26 Answer of the Third Learning Keyboard instruction) Then if there is no answer or the answer is not the correct one, there will play the same audio again. If the user types the correct answer the first question of the Tap format is going to be displayed.

Figure 26 Figure 26 Answer of the Third Learning Keyboard instruction



These are the Learning Speech instructions briefly:

The first audio is played explaining to the user that now will learn to talk to the computer and that the user has to the user's finger on the round button and that the iPad will listen to the user answer after the beep sound is played. Meanwhile, the audio is played the user will see the text *Learning to talk to the computer* and white screen (See Figure 27 First Learning Speech Instruction is played). After the audio is finished of playing the round button will be displayed (See Figure 28 Answer of Learning Speech Instruction) and when the user presses the round button a beep button will sound (See picture Figure 29 Beep sound of the Learning Speech Instruction).

Figure 27 Learning Speech Instruction is played

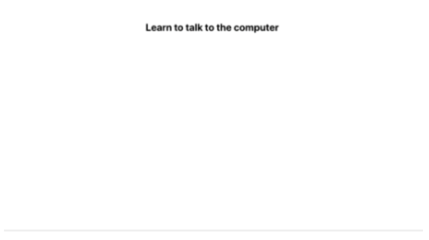


Figure 28 Answer of Learning Speech Instruction

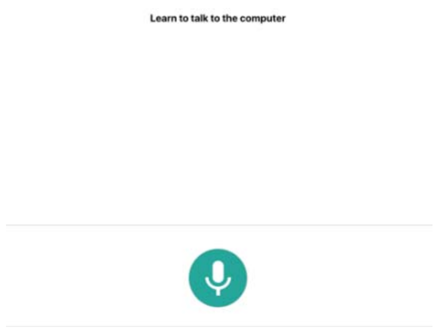
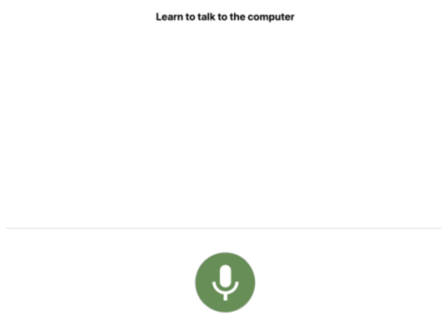


Figure 29 Beep sound of the Learning Speech Instruction

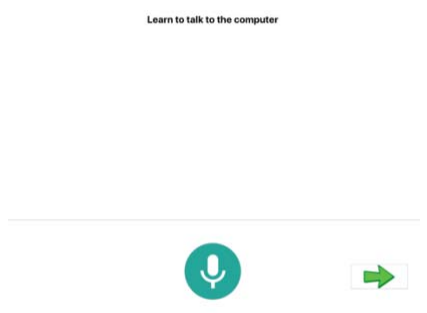


If the user does not press the button within 6 seconds, there will be audio encouraging the user to press the button. If the user does not hold the button long enough, there will be audio encouraging the user to hold the button more time so that the iPad can listen to the user. If the user does not release the button within 7 seconds, there will be audio encouraging the user to release the button.

After user holds the button successfully and then releases it, then it will play audio saying that the user has done magnificent work. Then the following audio instruction will be played encouraging the user to hold the button and say *åtta* at the same time (See Figure 27 Learning Speech Instruction is played). After the audio is finished of playing the round button will be displayed (See Figure 28 Answer of Learning Speech Instruction) and when the user presses the round button a beep button will sound (See Figure 29 Beep sound of the Learning Speech Instruction).

If the user does not press the button within 6 seconds, then there will be audio encouraging the user to hold the button and wait to hear the sound before answering. If the student answers incorrectly, then there will be audio encouraging the user to repter *åtta* while holding the button and after the beep sound is played. If the student responds *åtta* successfully there will be played audio saying to the user, well done and encouraging the user to press the arrow (See Figure 30 Arrow of the Learning Speech). After the user has pressed the arrow the first question of the Speech format will be displayed.

Figure 30 Arrow of the Learning Speech



A question flow will be created that will use the *React* states in each case to hide or show each component. This first version of the question flow goes as follows:

1. The user enters their username and password, only if they are correct the user logs (See Figure 20 Login Component; Figure 21 Login Component when the user is not authorised).

2. The Start Session component is hidden, and the Start component is displayed (See Figure 19 Start Component).

3. When the user presses the START button, the user starts the test, and this component is hidden.

4. The Learning Keyboard component is displayed (See Figures from 22 to 26), and the instructions are explained according to Script 1 (See Script 1 in Appendix C).
5. Once the user has finished the instructions, this component is hidden.

6. The Keyboard component is shown as this is the first test question.

7. Once all the questions of the Tap format are finished, the Keyboard component is hidden (See Figures from 10 to 18).

8. The Learning Speech component is displayed (See Figures from 27 to 29), and the instructions are explained according to Script 2 (See Script 2 in Appendix C).

9. Once the user has finished the instructions, this component is hidden.

10. The Speech component is shown to be the first question of the test (See Figures from 3 to 9).

11. Once all the questions in the Verbal format have been completed, this component is hidden.

12. The user at this point has finished the test, and a message is displayed informing about it.

3.3.7 Measuring the time

One of the requirements mentioned above was to know the time that each user used during the test. Knowing this time will allow confirming if the digital test decreases and in what percentage, the time that students must spend in the realisation of the test, as well as the amount of time that each teacher saves for the traditional test.

For the purpose of carrying out the analysis of the duration of the test, the total duration of the test will be measured, from when the user presses the START button until the latter sends the answer of the last question. This information will be stored in the database next to the identifier of the session to which it belongs, allowing knowing what day the test was performed and how long that user took.

With a view to measuring the duration of each question, the duration from the beginning of the question is measured, that is before the audio of that question is reproduced, until the user presses the button to send the answer to that question.
With the intention of measuring the response time of each question, it is measured from when the audio has finished until the user has clicked on the button to send the answer to that question. With this time, it is going to be known if the user is taking longer than the average and in future developments can make recommendations for similar exercises for the user to improve significantly in this type of exercise.

By measuring the response time of a question and the total duration of that question, it can be known if there have been technical problems during the performance of the test and if these have influenced the response time of the user.

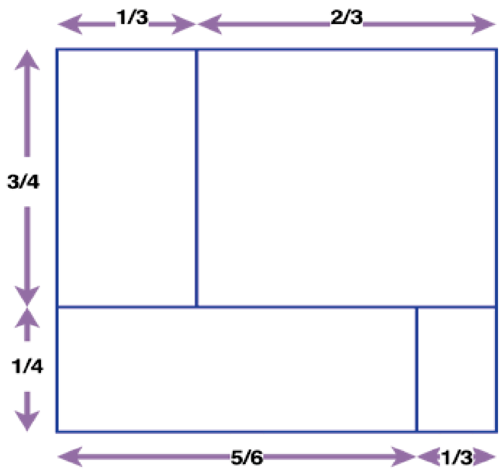
These times will be stored in the database next to the type of answer format of that question and the question number to which they correspond.

Additionally, the time it takes each user to perform the instructions of both formats is also measured. Knowing these times together with the answers that the user is giving during the realization of these instructions, allows knowing if some instructions need to be changed because there is a high percentage of users that fail in the interpretation of some instruction or if the time spent by a high percentage of users in the instructions is very high, so it should be reduced.

3.3.8 New CSS Architecture

A new architecture has been implemented (See Figure 31 New CSS architecture) in the CSS using the Flexbox Layout module or Flexible Box [39]. The purpose is to provide a more efficient way to design, align and distribute the space between the elements of a container, even when its size is unknown and/or dynamic. This allows the components that are currently in the application to have a similar architecture. Therefore, changing from one component to another will not mean a tremendous aesthetic change, since when using this architecture, all the components will be placed according to that architecture. This allows the user not to notice a sizeable visual change in the location of the elements when moving from one component to another. This will allow in the future that each time a change in the CSS is made, either because it has to change certain elements or functionalities of a component, the creation of a new component or any other type of change is much easier to execute. This will be possible because the CSS will be better organised, and it will be easier to make any change without having to refactor all the CSS.

Figure 31 New CSS architecture



3.3.9 Speech Picture & Keyboard Picture component

In the elaboration of the flow of questions, it was decided that multiple questions needed to show images. In particular questions Q07, Q08, Q11 and Q13. To solve this, two new components were created in *React* that allow displaying an image in both response formats. In addition, the *Keyboard* component and the *Speech* component were modified, including the new functionality of inserting images, giving rise to the component *SpeechPicture* (See Figure 32 SpeechPicture Component) and *KeyboardPicture* (See Figure 33 KeyboardPicture Component). To add this functionality an Image component was developed that added the name image that had been passed as a property to this component.

Figure 32 SpeechPicture Component

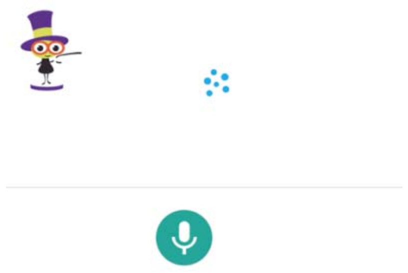
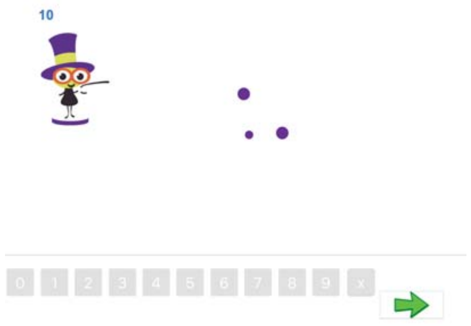


Figure 33 KeyboardPicture Component



3.3.10 Version two of the verbal instructions

After performing user testing, there was observed that the majority of the student who carried out the test were stuck in the verbal instructions. This means the students were able to do the rest of the test, the tap part, without having any problems. For solving this problem, a new version of the verbal instructions was done (See Script 3 in Appendix C) and the whole structure of the verbal part was refactored to be able to perform the new instruction version. The new instruction version was the double size than the first one, and it takes double time to go through.

These are the Learning Speech instructions briefly:

A box is shown while the first audio is played explaining to the user that that box is sometimes shown (See Figure 34 Audio box) That box is the audio allows box which appears, in the iOS devices, if the user did not say anything in more than five minutes. When that first audio is finished, the following audio will be played encouraging the user to click the button *Tillåt*. Meanwhile, a hand will be shown above the button which is about to click it (See Figure 35 Audio box with a hand clicking the button *Tillåt*).

Figure 34 Audio box

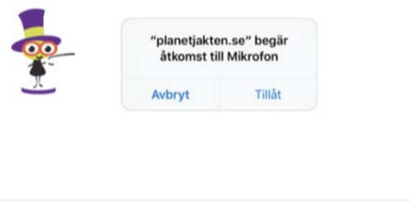
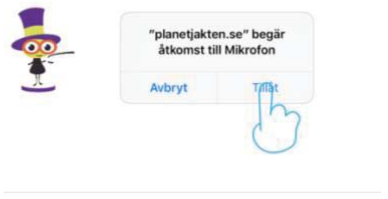


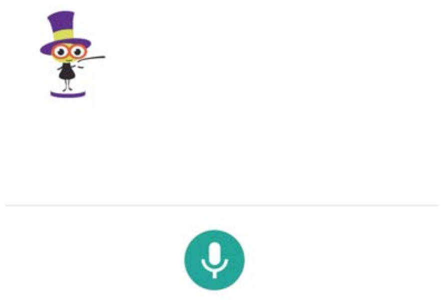
Figure 35 Audio box with a hand clicking the button Tillåt



If the student pushes the cancel button, *Avbryt*, there will be played audio encouraging the user to click the button *Tillåt*. If the user clicks correctly the allow button, there will be audio explaining to the user that has always to click the allow button.

After the user has clicked the allow button, there will be displayed a green round button while audio is played encouraging the user to press it (See Figure 36 Answer button is shown).

Figure 36 Answer button is shown



Afterwards, the teacher in the left corner explain to the user that will start learning to talk to the computer (See Figure 37 Learning to talk to the computer). Then a hand with a finger up will appear to click the round button (See Figure 38 Answer button with a hand above it). After that a pling-sound is played, the button will become active, and audio will be played encouraging the user to keep the user's finger on the button all the time (See Figure 39 Answer button with a hand above it, pressing the button).

Figure 37 Learning to talk to the computer

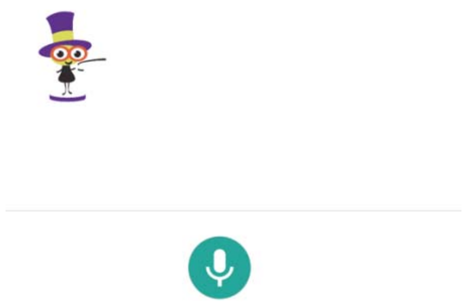


Figure 38 Answer button with a hand above it

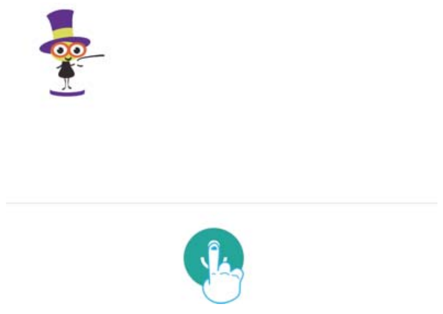
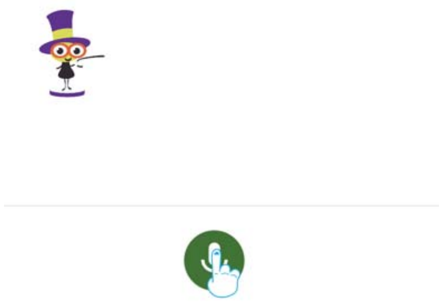
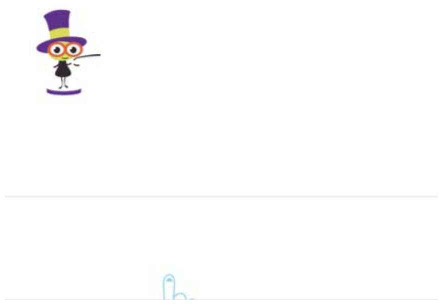


Figure 39 Answer button with a hand above it, pressing the button



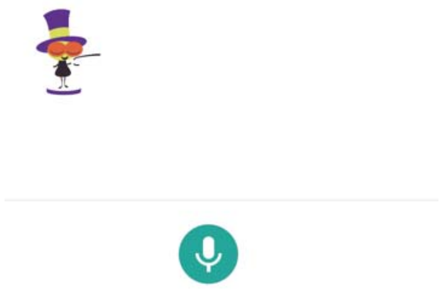
After the pling-sound is played an audio counting till twelve and saying the animation can count whole way up to 120. When audio has finished the finger from the button is removed (See Figure 40 Answer button with a hand leaving the button) and the button will no longer be active.

Figure 40 Answer button with a hand leaving the button



The following training question will be shown, and audio will encourage the user to put the finger on the button, hold the finger in the button and after the pling-sound say *åtta* (See Figure 41 Animation is moving the mouth and audio is explaining the question). After the audio is finished the button for answering the instructions will be displayed (See Figure 36 Answer button is shown). If the user does press the within six seconds, audio will encourage the user to hold down the button, wait for the pling-sound and say åtta. If the user does release the button within seven seconds, there will be audio encouraging the user to release it. If the user responds incorrectly or it is not record anything, there will be played audio repeating the instructions of this question.

Figure 41 Animation is moving the mouth and audio is explaining the question



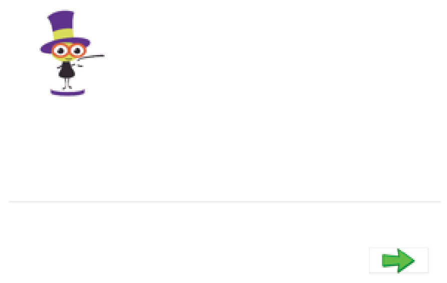
If the user does it correctly then the following question will be displayed and an audio will be played encouraging the user to put the finger on the button, hold the finger and after hearing a pling-sound say *1 2 3 4 5 6 7 8* (See Figure 41 Animation is moving the mouth and audio is explaining the question).

After the audio is finished the button for answering the instructions will be displayed (See Figure 36 Answer button is shown). If the user does press the within six seconds, audio will encourage the user to hold down the button, wait for the pling-sound and say *1 2 3 4 5 6 7 8*. If the user does release the button within seven seconds, there will be audio encouraging the user to release it. If it is not record anything, or the user responds anything then there will be displayed the following the instruction and an audio will encourage the user to hold down the button all the time, wait till a pling-sound is played and say *4 5 6 7 8 9 10* (See Figure 41 Animation is moving the mouth and audio is explaining the question).

After the audio is finished the button for answering the instructions will be displayed (See Figure 36 Answer button is shown). If the user does press the within six seconds, audio will encourage the user to hold down the button, wait for the pling-sound and say *4 5 6 7 8 9 10*. If the user does release the button within seven seconds, there will be audio encouraging the user to release it. If it is not record anything or the user responds anything then there will be displayed the following the instruction and an audio will encourage the user to hold down the button all the time, wait until a pling-sound is played and count down from five, saying *5 4 3 2*. (See Figure 41 Animation is moving the mouth and audio is explaining the question).

After the audio is finished the button for answering the instructions will be displayed (See Figure 36 Answer button is shown). If the user does press the within six seconds, audio will encourage the user to hold down the button, wait for the pling-sound and say *5 4 3 2*. If the user does release the button within seven seconds, there will be audio encouraging the user to release it. If it is not record anything or the user responds anything, then there will be displayed the first question of the Speech format will be displayed after the user click the arrow button (Figure 42 Arrow button of the Learning Speech version 2)

Figure 42 Arrow button of the Learning Speech version 2



3.3.11 Building shuffling questions structure

Once the tests were completed, the main component of the application was refactored entirely to address the need to include the complete flow of questions and the randomisation of both parts. This refactorisation was done in the following way:

First of all, there was created a *JavaScript* file including all questions of the test in different arrays. Those arrays were arranged following the structure of having an array with all the questions of the same level of the same question. Then another array was wrapping all the arrays of a question, with all the levels of that question contained in that array. Then all those arrays which have the same answer format are included in another array for both format: visual and tap questions. The last array is another array which contains both arrays.

After having all the question structure, the following step is to know which question have to be shuffled and how. At first, there was the requirement of shuffle the question in the following way:
The questions one, two, three and seven which are corresponding to the verbal format are not going to be randomised. This means they are going to have the order define from one to seven and the fixed order within each question.

The questions four, five, eleven and thirteen which are corresponding to the verbal format are not going to be randomised within each question. The question eight is going to randomise among all items without taking into account the level at which they correspond. The question fourteen is going to have a fixed order in the levels, from lower level to higher level. Although, it is going to be randomised within each level. Besides, the order of all the tap questions is going to be shuffle too.

The next step is to create two new methods in the main wrapper *React* component which controls whether this has to show the user's instructions of one of both formats or the question of one the formats or the test has finished since the user has gone through the whole test. To do this, after the start button is clicked the user is going to see one of both format instructions. Then after the instructions of that format are finished the user is going to go through the question test of the same format as the instructions were. Following this, all the questions, of that format, are going to display one after another. When the last question of that format is submitted, the other format instructions are going to be displayed as the first format instruction did and then the same for the questions of that format. Lastly, when the last question of this second format is submitted the final component are going to show, letting the user know that the test has ended.

That question flow is possible because the question flow array is defined before showing any instructions or questions of the test it allows the application to know which one the actual and following question is. This is used in those two methods mention before to know which content needs to show in the next question such as the audio and/or the image of that question.

After testing with the shuffle questions version of the application, it is observed that the randomisation at this point, where the number of the student who is going to do the test is not going to be more twenty in the same classroom it is unnecessary to carry out in many of the questions. So, the last version of the questions which are going to be shuffle is the question eight which is going to shuffle all of them at once and the question fourteen which is going to be with the fixed order of the level but randomise the questions of each level.

CHAPTER 4

4. TESTING THE DIGITAL TEST

4.1 Week Test in Swedish schools

From 6 to 14 November the test was carried out in different Swedish schools. At first, all the schools were going to take the test on the same day. However, this could lead to problems due to the number of users who were going to perform the test. It was therefore decided to do it in several days.

The factors which have produced these errors could not be mitigated before the week of testing due to the following reasons:

First of all, stress tests of the server were not performed due to lack of time. Secondly, the number of students taking the test was higher than in the internal tests, so the problems that might arise were unpredictable.

The following describes the problems that arose and how they were mitigated or solved:

One of the first problems that arose during the tests in the schools was finding devices with a version not compatible with *WebRTC*. *WebRTC* is an *HTML5* specification where there can add media communications in real-time directly between the browser and the devices [40]. *WebRTC* allows voices and video communication to work on web pages. This problem arises because the application is only compatible with the version of *iOS 11* and higher. Although, in some schools where several tests were performed, it was not possible to update or install another browser, which would have solved the problem.

In the first two days of the test, several problems allowed the server to restart every time it detected an error. It was enough that one of the students who was doing the test had a problem that was not addressed and the server would restart. This caused all the students who were doing the test at that moment to disconnect from the server. The most common error that arose was: *transport close* [41]. Different solutions were tried to solve this problem: implement *sticky-cluster* on the server use *Heroku* as a platform to deploy the application without using *load balancer*, use the *Google App Engine* service to do the deployment of the application without using load balancer, and the final solution, which was the one that worked, was to use the *Google Compute Engine* service without load balancer [42]-[45].

If a student presses the button that closes the *iPad*, the user will be disconnected from the test. This is still happening since there is no way for making the socket to go on working once the device has been blocked. When a user performs this action will be disconnected, and when that student returns to the test page, the student will be notified with a pop-up message that must notify the teacher that there has been a technical problem.

If a student changes windows in the browser or goes to another application, the student will disconnect from the test. This is still happening to prevent the student visit another page during the test. When a user performs this action will be disconnected, and when that student returns to the test page, the student will be notified with a pop-up message that must notify the teacher that there has been a technical problem.

If a student pressed the audio button too many times, in the instructions, it would play as many times as it was clicked. This problem is fixed by detecting when an instruction is playing and preventing the playing of other instructions at the same time even if the user presses many times a button which leads to playing other audio. This helps that just one instruction can be played at the same time.

In case a student recorded an answer, and after a while decided to go on with the test by pressing the arrow button and sending the answer or try it to record another answer, it would appear an audio timeout error. The error was caused because the user clicks the button, so the audio was streaming to the *Google Speech to Text API* and when the user has finished recording the answer and release the button not always the audio was stopped of being streamed. This makes the audio goes on streaming and causes the error audio timeout. This error was fixed by changing the press version of the *Speech* component to a click version where the user clicks when the user wants to record the answer and click again when wants to stop it.

There was fixed the problem of measuring the time in the first question of the verbal and the tap question. This was produced because the method was sending the questions time at the wrong time.
A small percentage of students still did not understand how to send the answer in the *Speech* instructions. For this reason, it was changed the order of question flow, putting the Tap response first ensure at least gather the answer of that response format before the server fell in the *Speech* part.

In some cases, there were problems related to the variable *Context* that was solved with the creation of *callbacks*, which makes that variable not be called before it is created.

Sometimes, there were problems with the bandwidth connection the centre had. This made: that it took too long to ask some questions as some content was not loaded quickly, answers were not sent and other problems derivate from the low speed of the network. As a solution, there was used a hotspot connection with a mobile network offering higher speed and reducing the possibility of creating errors in the server. Besides, in some centres due to the use of the *Mobile Device Management* control system that offers total control over all the devices where it is installed, it was not possible to record the answer of the *Speech* part and send the audio.

In the last sessions, the students could perform the keyboard version altogether, and even some of them managed to do the complete test without being disconnected from the server.

CHAPTER 5

5. CONCLUSIONS AND FURTHER WORK

5.1 Conclusions

The project describes a new way of measuring mathematical skills through the construction of a digital test. The digital test developed measures the degree of knowledge of all the content taught in the *First Grade of Mathematics* of *Swedish* schools.

This project will serve as a precedent for future research projects since it will benefit teachers by reducing the time, they spend on all those time-consuming tasks when conducting paper tests, such as proofreading and individual sessions. Moreover, it will allow them to perform the digital tests more frequently, learning more about student progress. In addition, it will also be beneficial for students, since everyone will perform a standard digital exam and the answers will be standardised.

Last but not least, it is essential to carry out the analysis of the data and to carry out more digital tests with the users to determine with precision the percentage of time that each teacher could reduce when performing the digital test.

For further details of *Raul's* work see Appendix B at the end of the project.

5.2 Further Work

This project has been focused on the content taught in the first semester of the *First Grade of Mathematics in Swedish* schools. So other future projects can focus on adding the questions and the content of the *Spring National Swedish Mathematics Test for students of First Grade.*

Besides, it can be added the contents which are taught during the *Spring* semester and are not measured in the *Spring National Swedish Mathematics Test for students of First Grade* such as:

- The equivalence symbol-important-sound-text-spatial (line of numbers) -number-writer
- The mathematical symbols <,>, =
- The clock by identifying: whole and half hours, in half an hour
- The knowledge of problem-solving
- Discriminate geometric figures between triangle, circle and rectangle
- Measurements with a rule
- Odds and even numbers
- Knowledge and management of the currency

The problems that are detailed below can be solved in future projects:

The current user interface is clean and simple so that the user can understand at a glance where the navigation and input elements are and the concordance between them. As more users complete this digital test, more information will be collected, including different ideas to improve the current interface.

This project explains the requirements of the teacher's user interface. Future projects can focus on the development of the user interface with the teacher's control panel and how the teacher can collect the test results and have control over everything that happens during the digital test session.

It has been observed during the performance of the test, that few students did not clearly understand some of the instructions. This means that they cannot continue with the test because they got stuck in the first instruction and, sometimes, they are not able to answer the question correctly. The realisation of explicit and more precise instructions will allow all students to complete the test.

Improving the current architecture of the data schema lacks the measurement of some features.

Carry out the analysis of the current data to determine the time of the realisation of the test between the traditional and digital format as well as to measure the correlation between both tests.

APPENDIXES

Appendix A

Table 1 Oral assignments, Autumn Semester in year 1

Question number	**Question**	**Lower level**	**Mid-level**	**Higher level**
1	Count until I say stop	1-25	1-50	1-115
2	Start at X and keep counting until I says stop	3-12	9-20	26-80
3	Count backwards from 10	5-0	10-0	20-0
4	What number comes after	1, 4	7, 10	79, 99
5	What number comes before	2, 4	5, 9	50,72
7	Number consistency	Compare 2 amounts. Place 3 large objects in a pile	Compare 2 amounts. Place 3 large objects in a pile	Not tested on higher level.
		and 6 small objects in a pile.	*Put them close together.*	
		Are there the same number of objects in both piles?	How many objects are there?	

		How did you reason?	*Disperse the 6 objects.*	
		Note if the student counts the numbers	Now I have spread out the objects.	
		or compares the sizes.	How many objects are there now?	
		Note if the student needs to count the amount again and if the students can keep track of counted and not counted objects.		
8	Subitizing	*Show a dice.*	*Show a dice.*	*Use the 11 hidden objects.*
		Show the number 3 on the dice.	*Show the number 5 on the dice.*	I have hidden objects under here. I will let you look at
		How many dots are there?	How many dots are there?	them for a short period of time and then you must tell
		Show the number 4 on the dice.	*Show the number 6 on the dice.*	me how many objects you think there are. You will not

		How many dots are there?	How many dots are there?	have time to count them. Are you ready?
		Note if the student needs to "point count"	*Note if the student needs to "point count"*	*Lift the paper and show max 2 seconds. Cover.*
9	Name numbers/Combine training picture with number card	*Show a dice and the number cards 1-6.*	*Show the number cards 0-10.*	*Show the number cards 10-100 and 11-20.*
		Show a side on the dice.	*Point to the numbers and let the student name them*	*Point to the numbers and let the student name them*
		What number card matches the dice face?	Say the number that I am pointing to out loud.	Say the number that I am pointing to out loud.
11	More o less	*Show 3 objects*	*Show 6 objects.*	*Show 13 objects.*
		Here are 3 objects.	Here are 6 objects.	Here are 13 objects.
		How many would there be if there were 2 more?	How many would there be if there were 2 more?	How many would there be if there were 3 more?

		How did you reason?	How did you reason?	How did you reason?
		Here are 3 objects.	Here are 6 objects.	Here are 13 objects.
		How many would there be if there was 1 fewer?	How many would there be if there was 1 fewer?	How many would there be if there were 3 fewer?
		How did you reason?	How did you reason?	How did you reason?
12	Divide numbers	Show 5 objects.	*Show 5 objects.*	*Hide objects, for instance in your hand.*
		Divide the objects into 2 parts/piles.	Divide the objects into 2 parts/piles.	*Amounts between 7-10.*
		How many objects are there in each part/pile?	How many objects are there in each part/pile?	*For example number 7:*
		If the students can divide in one way it is acceptable.	Can you do this in more ways?	I have 7 objects here.
				Hide 4 objects.
				Show 3 objects.
				How many have I

				hidden? How did you reason?
				Continue with other numbers that you believe are appropriate.
				If the student needs to count with fingers you stop.
				If the student has automatized number facts (i.e., 4+3, 7-2)
				and can generalize it to other ways of dividing it you can stop.
13	Half/Double	*Show 4 objects.*	*Show 8 objects.*	What is half of 12?
		Divide the objects so we	Divide the objects so we	How did you reason?

		both get the same amount.	both get the same amount.	
		How many objects did we each get?	How many objects did we each get?	Lisa is 6 years old.
			What is half of 10? How did you reason?	Emir is twice as old as Lisa.
				How old is Emir?
				How did you reason?
				What is twice the amount of 7?
				How did you reason?

Table 2 Digital format of the questions of the traditional paper test created by Martin Hassler Hallstedt.

Verbal instruction	Number of items	Response type
Q1	1	Verbal
Q2 low	1	Verbal
Q2 mid	1	Verbal
Q2 hi	1	Verbal
Q3 low	1	Verbal
Q3 mid	1	Verbal
Q3 hi	1	Verbal
Q4-low-01	1	Tap
Q4-low-02	1	Tap
Q4-mid-01	1	Tap
Q4-mid-02	1	Tap
Q4-hi-01	1	Tap
Q4-hi-02	1	Tap
Q5-low-01	1	Tap
Q5-low-02	1	Tap
Q5-mid-01	1	Tap
Q5-mid-02	1	Tap
Q5-hi-01	1	Tap
Q5-hi-02	1	Tap
Q7-low	1	Verbal
Q7-mid-01	1	Verbal
Q7-mid-02	1	Verbal
Q8-low-01	1	Tap
Q8-low-02	1	Tap
Q8-mid-01	1	Tap
Q8-mid-02	1	Tap

Q9-low-01-06	6	Verbal
Q9-mid-00and07-10	5	Verbal
Q9-hi-10-100	10	Verbal
Q11-low-01	1	Tap
Q11-low-02	1	Tap
Q11-mid-01	1	Tap
Q11-mid-02	1	Tap
Q11-hi-01	1	Tap
Q11-hi-02	1	Tap
Q13-low-01	1	Tap
Q13-mid-01	1	Tap
Q13-mid-02	1	Tap
Q13-hi-01	1	Tap
Q13-hi-02	1	Tap
Q13-hi-03	1	Tap
Q14-low-01-06	5	Tap
Q14-mid-00and07-10	6	Tap
Q14-hi-10-100	10	Tap
TOTAL ITEMS	59	

Table 3 User stories with the steps the student will follow during the test

Student logs in	**Student starts the test**	**Student receives instructions**	**Student interruption**	**Student answers question**	**Student finishes the test**
As a student, I want to have a distinct and short login name, since I'll remember it better	As a student, I want to have a countdown that show when the test is going to start, because it will give me time to focus	As a student, I want to know when I need to listen, so that I don't lose my focus when new instructions are presented	As a student, I want to be able to go on the session after a break or an interruption	As a student, I want to know with a visual image, for example a symbol, in which kind of format I have to respond the question or if there is more than one	As a student, I want to know that I've finished the test
As a student, I want to not get disconnected if I am logged in before others	As a student, I want to I want to hear a start sound or see a symbol that signals when the test is going to start,	As a student, I want to know in which answer I can edit the answer before submitting it	As a student, I want to have something that help me forward even if I don't what to respond, because I might get upset	As a student, I want to I need to be kept motivated and avoid feeling helpless	As a student, I want to be kept interested for the whole duration of the test,

	because I want to when I have to be pay attention		and lose interest if I don’t know what to do	through the test, because the lack of feedback to my answers can be scary	even after I’m finished, because otherwise I might start to interference to my classmates
	As a student, I want to engage in something while I am waiting for the test to start, because I don’t want to interrupt other students				
	As a student, I want to hear a start sound or see a symbol when the start has begun				

Table 4 User stories with the steps the teacher will follow while the test is being made

Teacher: login and before test	**Teacher: students start the test**	**Teacher: student interruption**	**Teacher: student finish**
As a teacher, I want to select my own login and not use a password since it will be easier to remember for me	As a teacher, I want to push a button and know if all the students are ready to start the test	As a teacher, I want to know my student's situation so I can help them is they get stuck	As a teacher, I want a clear sign that let me know the last student has finished the test
As a teacher, I want the delivery of the students log in to smooth because I want to spend as less time as possible in this	As a teacher, I want a clear sign of every student having started the test so that I don't have to spend time checking every individual	As a teacher, I want to get warnings about students who are not responding or responding stereotypically (i.e. zeros) because I can help them to understand the test	As a teacher, I want the application to instruct students to notify me in case of problems, so I can feel confident in that they are progressing fine if I don't hear or see anything
As a teacher, I want the students to be able to log in without my help because helping many individual		As a teacher, I want any test interruptions recorded so I can take that into account when assessing the results	As a teacher, I want the app to be able to go on the session even if the hardware crashes or the student suddenly quits, so

students will take a lot of time		As a teacher, I want to know the progress of each student (starting, finished, almost finish it, not starting, …) so that I can have everything under control	that a student can jump right back in and go on with the test

Appendix B

Thereupon is going to be enumerated some of the tasks the author of this work has done:

• Create the user stories of both teacher and student.

• Look the drawbacks and benefits of adopting different technologies in the project carefully.

• Create a Docker image for the React application.

• Create a containerised web application; deploy a containerised web application in the Google Kubernetes Engine.

• Create documentation for the deployment of the application and about how to roll update of the deployed version of the app.

• Create a real-time Single Web Application

• Create different React components and the different versions of them.

• Create a new CSS architecture

• Measure answer time response, the whole question time and whole test time that each student spent in the different React components of the application.

• Create the question flow for the test application.

• Add questions and instructions as well as make them appear at a specific time and with the corresponding elements.

• Create the randomising question flow.

• Fix bugs and solve the problem that has arisen before and during the test with 300 users.

Appendix C

Script 1 Tap instructions version 1

1. Audio playing: *Nu ska du lära dig att klicka på siffror. Klicka på 3. Klicka sen på pilen.*

2.1. If the student does not respond within 7 seconds play the audio:

Klicka på siffran 3. Klicka sen på pilen.

2.2. If the student answers the instruction incorrectly play the audio:

Klicka på siffran 3. Klicka sen på pilen.

2.3. If the student clicks 3 and then the arrow play the following audio and go to the following instruction:

Bra!

3. Audio playing: *Om du klickar på fel nummer kan du ta bort det numret genom att klicka på knappen med ett kors. Ta bort 8 och klicka istället på 5*

4.1. If the student does not respond within 7 seconds play the audio:

Klicka på knappen längst ner med ett kryss. Klicka sedan fem.

4.2. If the student answers the instruction incorrectly play the audio:

Klicka på knappen längst ner med ett kryss. Klicka sedan fem.

4.3. If the student clicks delete, clicks 5 and then the arrow go to the following instruction.

5. Audio playing: *Nu ska vi skriva 10. Först klicka 1 och sedan 0*

6.1. If the student does not respond within 7 seconds:

Klicka först ett, och sedan noll

6.2. If the student answer incorrectly:

Klicka först ett, och sedan noll

6.3. If the student clicks 10 and then arrow

Snyggt!

Script 2 Verbal instructions version 1

1. Audio playing: *Hej! Nu lär du dig att prata med datorn. Sätt ditt finger på den runda knappen. Jag börjar börja lyssna efter pipen*

2. Beeping sound is played upon pressing

3.1. If student does not press button within 6 seconds, then play the following audio:

Tryck på knappen

3.2. If student does not hold the press long enough (until recording starts):

Håll nere knappen längre så jag kan lyssna

3.3. If student does not release the button within 7 seconds, then play the following audio

Släpp knappen!

4. After student holds the button successfully and then releases it, then play the following audio and go to the following instruction:

Bra jobbat! Släpp knappen

5. Play the audio*: Håll knappen nedtryckt och säg samtidigt "åtta"! Släpp sen knappen.*

6.1. If student does not press button within 6 seconds, then play the following audio:

Jag hörde inte. Håll nere knappen, efter tonen får du prata

6.2. If student does not release the button within 7 seconds, then play the following audio:

Släpp knappen!

6.3. If the student answers incorrectly, then play the following audio:

Jag hörde inte. Håll nere knappen, vänta på plinget och säg "åtta".

6.4 if the student responds "8"/"åtta" successfully:

Bra, tryck på pilen

Script 3 Verbal instructions version 2

1. Audio playing: *Ibland visas den här rutan*

2. The allow box is shown

3. Audio playing: *Klicka alltid här dä*

4. The allow button is pressed

5. If student pushes the cancel button, then play the audio:

    Ajajaj, klicka här

  And then the allow button is pressed

6. If the student clicks correct the allow button

    Snyggt! Klicka alltid här

  And then the allow button is pressed

7. Play audio: *Tryck ner den runda knappen*

8. The teacher avatar pops up in the left corner saying:

    Hej! Nu lär du dig att prata med datorn

9. A hand with a finger up goes on the round (handGestureShow)

10. Pling-sound is played

11. The pointing finger moves to the button and then is played:

    ... håller fingret på knappen hela tiden

12. The button gives a clear activation signal when is touched by the finger and then the button is activated as described above all the time

13. After the pling-sound, is played the audio: *håller fingret på knappen hela tiden. Nu räknar jag så långt jag kan från en: 1,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 och jag fortsätter så långt jag kan. Om jag kan, kan jag räkna hela vägen upp till 120*

14. Button is active as described above all the time

15. When the audio is finished, the finger from the button is removed. The pointing finger is removed from the button, and the button is no longer active. Then handGestureRelease() and handGestureHide() and go to the next training question

16. Audio is played*: Sätt fingret på knappen, lyssna efter plinget, håll kvar fingret efter plinget och säg åtta*

17. If the student does not press the button within 6 seconds:

Audio is played: *Håll nere knappen, vänta på plinget och säg "åtta"*

Håll nere knappen, vänta på plinget och säg "åtta"

Arrow button bounces slightly to indicate that it should be pressed

18. If the student does not release the button within 7 seconds, then play audio:

Släpp knappen!

19. If the student responds incorrectly or it is not record anything, then play the audio

Hoppsan! Håll nere knappen hela tiden när du pratar, vänta på plinget, säg "åtta". Sen får du släppa knappen

20. If the student responds "8"/ "åtta" successfully, then go to the next training question

21. Audio is played: *Snyggt! Sätt fingret på knappen, lyssna efter plinget, håll kvar fingret efter plinget och säg "1 2 3 4 5 6 7 8"*

22. If student does not press the button within 6 seconds, then play the audio:

Håll nere knappen, lyssna efter plinget och säg "1 2 3 4 5 6 7 8"

23. If student does not release the button within 7 seconds, then play the audio:

Släpp knappen!

24. If the student does not respond anything, then play the audio:

Ojdå, Håll nere knappen hela tiden när du pratar, vänta på plinget, säg "1 2 3 4 5 6 7 8". Sen får du släppa knappen

25. If the student responds anything, then goes to the next question

26. Play the audio: *Sätt fingret på knappen, lyssna efter plinget, håll kvar fingret efter plinget och räkna uppåt från 4 till 10, säg så här: 4 5 6 7 8 9 10"*

27. If the student does not press the button within 6 seconds, then play the audio:

Håll nere knappen, lyssna efter plinget och säg "4 5 6 7 8 9 10"

28. If student does not release the button within 7 seconds, play the audio:

Släpp knappen!

29. If the student responds incorrectly:

Ojsan! Håll nere knappen hela tiden när du pratar, vänta på plinget, säg "4 5 6 7 8 9 10". Sen får du släppa knappen

30. If the student responds “4 5 6 7 8 9 10” successfully, then go to the next training question

31. Play the audio*: Snyggt! Sätt fingret på knappen, lyssna efter plinget, håll kvar fingret efter plinget och räkna neråt från 5, säg så här: 5 4 3 2*

32. If the student does not press the buton within 6 seconds, then play the audio:

Håll nere knappen, lyssna efter plinget och säg “5 4 3 2”

33. If the student does not release the button within 7 seconds, then play the audio:

Släpp knappen!

34. If the student does no respond anything, the play the audio:

Ojsan! Håll nere knappen hela tiden när du pratar, vänta på plinget, säg “5 4 3 2”. Sen får du släppa knappen

35. If student reponds anything, then go to the next question

REFERENCES

[1] “This is the Swedish National Agency for Education,” *Skolverket*. [Online]. Available: https://www.skolverket.se/andra-sprak-other-languages/english-engelska. [Accessed: 15-Oct-2018].

[2] “BEDÖMNINGSSTÖD I taluppfattning,” PDF. [Online]. Available: https://docplayer.se/47251716-Bedomningsstod-i-taluppfattning.html. [Accessed: 05-Sep-2018].

[3] Neoteric, “Single-page application vs. multiple-page application,” medium.com, 02-Dec-2016. [Online]. Available: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page- application-2591588efe58. [Accessed: 04-Sep-2018].

[4] “How To Choose Between Native, Hybrid or Web App For Your Business,” BuildFire, 16-Apr-2018. [Online]. Available: https://buildfire.com/choose-native-hybrid-web-mobile-app/. [Accessed: 14-Sep-2018].

[5] "Socket.IO — Docs", *Socket.IO*, 2018. [Online]. Available: https://socket.io/docs/. [Accessed: 10- Sep- 2018].

[6] "React (JavaScript library)", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/React_(JavaScript_library). [Accessed: 04- Sep- 2018].

[7] A. Young, B. Meck and M. Cantelon, *Node.js in action*. Shelter Island, NY: Manning Publications, 2017.

[8] "MongoDB and MySQL Compared", *MongoDB*, 2018. [Online]. Available: https://www.mongodb.com/compare/mongodb-mysql. [Accessed: 10- Dec- 2018].

[9] "Google Cloud Platform Services Summary | Google Cloud Platform Terms | Google Cloud", *Google Cloud*, 2018. [Online]. Available: https://cloud.google.com/terms/services. [Accessed: 12- Sep- 2018].

[10] 2018. [Online]. Available: https://www.netsolutions.com/insights/what-is-google-cloud-its-advantages-and-why-you-should-adopt-it/. [Accessed: 13- Dec-2018].

[11] “Cloud Functions - Features and Benefits | Google Cloud,” Google. [Online]. Available: https://cloud.google.com/functions/features/. [Accessed: 12-Sep-2018].

[12] "Introduction to Docker | Qwiklabs + google-run", *Qwiklabs*, 2018. [Online]. Available: https://google.qwiklabs.com/focuses/1029?parent=catalog. [Accessed: 14- Sep- 2018].

[13] “What is a Container,” Docker. [Online]. Available: https://www.docker.com/resources/what-container. [Accessed: 02-Oct-2018].

[14] "Kubernetes? Docker? What is the difference?", *The Containership Blog*, 2018. [Online]. Available: https://blog.containership.io/k8svsdocker/. [Accessed: 18- Jul-2018].

[15] "User Stories | Atlassian", *Atlassian*, 2018. [Online]. Available: https://www.atlassian.com/agile/project-management/user-stories. [Accessed: 19-Dec- 2018].

[16] 2017. [Online]. Available: https://buildfire.com/choose-native-hybrid- web-mobile-app/. [Accessed: 14- Sep- 2018].

[17]"Average App Store Review Times", *Appreviewtimes.com*, 2018. [Online]. Available: http://appreviewtimes.com/. [Accessed: 14- Sep- 2018].

[18] "ReactJS vs Angular5 vs Vue.js — What to choose in 2018?", *Medium*, 2018. [Online]. Available: https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in- 2018-b91e028fa91d. [Accessed: 15- Sep- 2018].

[19] "Comparison with Other Frameworks — Vue.js", *Vuejs.org*, 2018. [Online]. Available: https://vuejs.org/v2/guide/comparison.html. [Accessed: 06- Oct- 2018].

[20] "facebook/react", *GitHub*, 2018. [Online]. Available: https://github.com/facebook/react/wiki/Sites-Using-React. [Accessed: 15- Sep- 2018].

[21] "MongoDB – The Leading NoSQL Database", *MongoDB*, 2018. [Online]. Available: https://www.mongodb.com/leading-nosql-database. [Accessed: 15- Sep- 2018].

[22] "Categories," *Heroku Dev Center*. [Online]. Available: https://devcenter.heroku.com/articles/nodejs-mongoose. [Accessed: 16-Sep-2018].

[23] Lukša Marko, Kubernetes in action. Shelter Island, NY: Manning Publications Co., 2018.

[24] "Can I use... Support tables for HTML5, CSS3, etc," Can I use... Support tables for HTML5, CSS3, etc. [Online]. Available: https://caniuse.com/#feat=websockets. [Accessed: 17-Sep-2018].

[25] "Building a Node.js WebSocket Chat App with Socket.io and React," ITNEXT, 22-Feb-2018. [Online]. Available: https://itnext.io/building- a-node-js-websocket-chat-app-with-socket-io-and-react-473a0686d1e1. [Accessed: 24-Sep-2018].

[26] “Socket.io — The Good, the Bad, and the Ugly - DZone ...” [Online]. Available: https://dzone.com/articles/socketio-the-good-the-bad-and-the-ugly. [Accessed: 23-Sep-2018].

[27] “Introduction to Redis,” Redis. [Online]. Available: https://redis.io/topics/introduction. [Accessed: 02-Oct-2018].

[28] M. Tacke, “Enforcing a single web socket connection per user with Node.js, Socket.IO, and Redis,” Hacker Noon, 30-Jul-2018. [Online]. Available: https://hackernoon.com/enforcing-a-single-web-socket-connection-per-user-with-node-js-socket-io-and-redis-65f9eb57f66a. [Accessed: 24-Nov-2018].

[29] “Docker Documentation,” Docker Documentation, 13-Dec-2018. [Online]. Available: https://docs.docker.com/. [Accessed: 02-Oct-2018].

[30] A. Atanassova-Barnes, “How to ‘Talk’ to Your Software: Alexa, Google, Watson, and Cortana, a Side-by-Side Comparison of Cloud Speech Recognition APIs.” Plano, Texas, 07-May-2018.

[31] J. Vidal, “Google Cloud Storage: Pros/Cons and how to use it with Javascript,” medium.com, 23-Feb-2018. [Online]. Available: https://medium.com/dailyjs/google-cloud-storage-pros-cons-and-how-to-use-it-with-javascript-ea9ce60a94c0. [Accessed: 03-Oct-2018].

[32] “Deploying a containerized web application | Kubernetes Engine Tutorials | Google Cloud,” Google. [Online]. Available: https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app. [Accessed: 03-Oct-2018].

[33] “Best practices for writing Dockerfiles,” Docker Documentation, 13-Dec-2018. [Online]. Available: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/. [Accessed: 10-Oct-2018].

[34] “State and Lifecycle – React,” – A JavaScript library for building user interfaces. [Online]. Available: https://reactjs.org/docs/state-and-lifecycle.html. [Accessed: 12-Oct-2018].

[35] M. T. Thomas, React in Action. Manning Publications Company, 2018.

[36] Markerikson, “markerikson/redux-ecosystem-links,” GitHub. [Online]. Available: https://github.com/markerikson/redux-ecosystem-links/blob/master/middleware.md#sockets-and-adapters. [Accessed: 19-Oct-2018].

[37] I. Ovenden, “Redux WebSocket Integration – Ian Ovenden – Medium,” medium.com, 26-Jul-2017. [Online]. Available: https://medium.com/@ianovenden/redux-websocket-integration-c1a0d22d3189. [Accessed: 14-Oct-2018].

[38] “How to use a single instance of Socket.IO in your React app,” ITNEXT, 14-Jun-2018. [Online]. Available: https://itnext.io/how-to-use-a-single- instance-of-socket-io-in-your-react-app -6a4465dcb398. [Accessed: 07-Oct-2018].

[39] “A Complete Guide to Flexbox,” CSS-Tricks. [Online]. Available: https://css-tricks.com/snippets/css/a-guide-to-flexbox/. [Accessed: 04-Oct-2018].

[40] T. Levent-Levi, “What is WebRTC? • BlogGeek.me,” BlogGeek.me, 03-Dec-2018. [Online]. Available: https://bloggeek.me/webrtc/. [Accessed: 23-Oct-2018].

[41] Socketio, “Continual ‘transport close’ on client · Issue #3025 · socketio/socket.io,” GitHub. [Online]. Available: https://github.com/socketio/socket.io/issues/3025. [Accessed: 07-Nov-2018].

[42] Uqee, “uqee/sticky-cluster,” GitHub. [Online]. Available: https://github.com/uqee/sticky-cluster. [Accessed: 07-Nov-2018].

[43] “Categories,” Deployment | Heroku Dev Center. [Online]. Available: https://devcenter.heroku.com/categories/deployment. [Accessed: 07-Nov-2018].

[44] “Building Custom Runtimes | Custom runtimes for the App Engine flexible environment | Google Cloud,” Google. [Online]. Available: https://cloud.google.com/appengine/docs/flexible/custom-runtimes/build. [Accessed: 11-Nov-2018].

[45] “Categories,” Redis | Heroku Dev Center. [Online]. Available: https://devcenter.heroku.com/articles/heroku-redis. [Accessed: 09-Nov-2018].