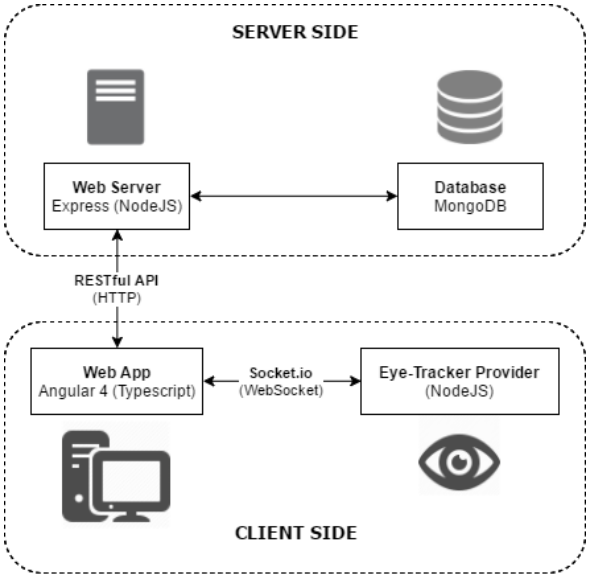
Development Planning

Eye-tracking augmented website - Clément Jacquet

CURRENT ARCHITECTURE



The app architecture consists of 4 independent units that communicate over the network.

Server side, the Web server, built with the `ExpressJS` framework (`NodeJS`), provide both the Web app (route `'/'`) and a RESTful API that enable interaction with the `MongoDB` database (route `'/api'`).

Client side, once the Web app is retrieved from the Web server, it runs in the user's browser, using the new framework `AngularJS 4`, and receives eye gaze location on the screen through the eye-tracker provider, a `NodeJS` program running locally and that is using eye tracker device API and drivers to retrieve information. In the future, there could be several eye-tracker providers connected that are not forcibly running on the user's computer.

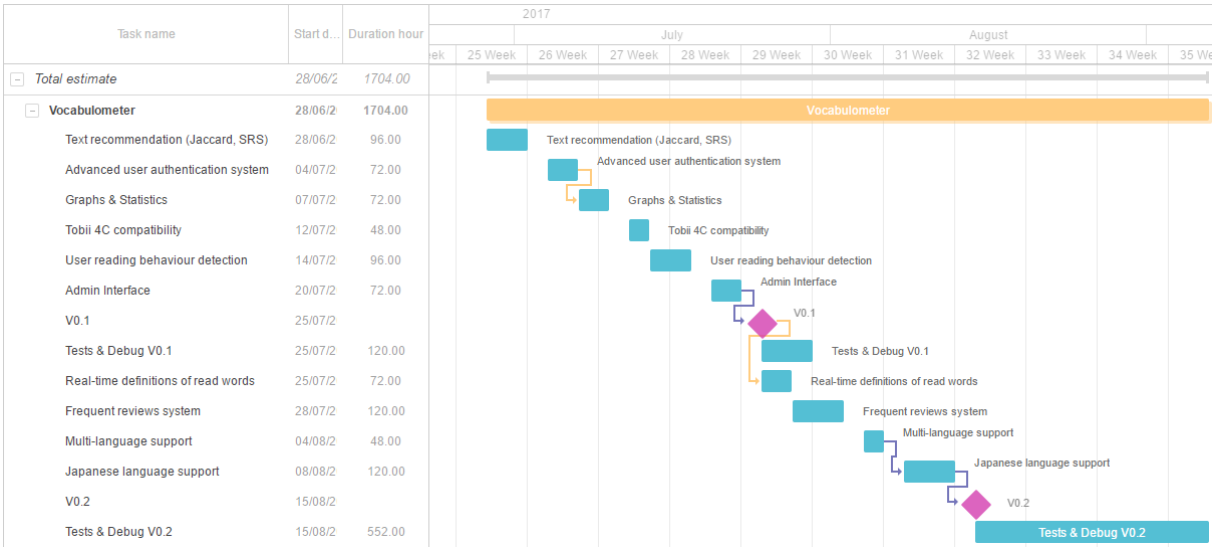
The project has been built based on a prior prototype conceived by University of Bordeaux students. Following choices has been made :

1. Building a Angular4 app based on Angular1 prototype's web app, to take advantage of the simpler structure and new functionalities of Angular4 ;
2. Using MongoDB as word storage rather than MySQL, as the nature of the data fits better the Document Oriented MongoDB DMBS, and as its query language is more expressive, making it easier to compute complex statistics than with SQL.
3. Following the prior point, using NodeJS with ExpressJS for the strong link between NodeJS and MongoDB, and the easiness to build RESTful APIs with ExpressJS.

DONE

1. Loading English texts from the MongoDB database ;
2. Detecting which words are "read" and saving them ;
3. Text display settings for debug

GANTT Figure



The figure above shows a simple Gantt summarizing the different tasks and their realization time estimations. Spaces between tasks represent week-ends. The development is split in two parts, corresponding to two versions, called V0.1 and V0.2. This split would enable to obtain more flexibility, in case after a month of development needs have changed.

Specifications in V0.1 are considered to have priority compared to V0.2 specifications, that corresponds to more advanced functionalities, whose realization time is even more complex to assess.

After each version, tests with multiple users will be conducted, to detect eventual bugs and collect enhancement propositions. After that the V0.1 is reached, tests and debug are planned to only take a week, and following what enhancement propositions have been made, V0.2 specifications could change.

From a macroscopic point of view, with such estimations, the **V0.1** would be completed over the **07/25**, and the **V0.2** over the **08/15**. Considering my internship officially ends the **09/06**, that would leave around 3 weeks for the final tests of the app with different users, collect comments about the app user experience and apply modifications, in parallel of resolving bugs, writing a maintenance manual and at the end eventually presenting a demonstration of the app.

SPECIFICATIONS

1. Text recommendation (Jaccard) **[4 days]**

   In the same way as the prototype, recommend texts where the Jaccard with the user's vocab is high if the user wants to train, or is low if the user wants to learn new words. Thresholds could be determined by using classification, in order to let the system knows what is a "hard" or an "easy" text.

2. User authentication system **[3 days]**

   Use user authentication to provide multiple users experience

3. Graphs & Statistics **[3 days]**

Add a view where statistics appear with figures, graphs. Statistics computed could be the count of known words, at which level the user knows it, how many words did he read in a day or in a week...

4. Tobii 4C compatibility **[2 days]**

   For the moment, the app is only compatible with the Tobii EyeX. The new model, the Tobii 4C, should be supported too, as it is the successor of the EyeX and would provide a better user experience for nearly the same price.

5. User reading behaviour detection **[4 days]**

   For the moment, words are considered read if a fixation occured nearby. However, the user could be just looking at random locations on the screen and not pay attention to the words themselves. Detecting whether the user is reading would enable us to tell if a word is truly read.

6. Admin interface **[3 days]**

   There isn't any interface to manage users and texts in the database, so every DB operation has to be done manually with a request to the DB. A web admin interface would enable to interact in an easier way with it, and make tasks as adding a new text into the database easier.

7. Real-time defintions of read words **[3 days]**

   Definitions of words just read would enable the user to understand quicklier the text he is reading.

8. Frequent reviews system **[5 days]**

   Implementing a Spaced Repetition System (SRS) would provide a better training experience to users.

9. Multi-Language support **[2 days]**

   Only English is supported for now. This task would consist into changing the DB structure and the website to easily support multiple languages

10. Japanese Language support **[5 days]**

    European languages use spaces to delimit words, but that's not the case for Japanese. The support for Japanese language is tougher, as delimiting words is totally different. Therefore it is a new task in itself.

LINKS

Trello : `https://trello.com/b/zNe0KT7W/vocabulometer`
Gantt : `https://app.ganttpro.com/shared/token/6727b078ffc30090c148ccde44ba37ad91a0867377777c207146#!/app/home`