

Romain GUILLOT
Informatique
Polytech Grenoble
Rapport de stage 2019

Developing A Service Platform for Electronic Therapy for Stutterers

Version : Française (FR)

Tome principal
ET
Annexe

Année universitaire : 2018 - 2019
Periode du stage : 20 mai 2019 - 9 août 2019

Remerciements

Avant de commencer, je tiens à remercier toutes les personnes qui ont contribué au bon déroulement de mon stage en m'accompagnant financièrement, moralement ou physiquement.

Je souhaite tout d'abord remercier ma superviseuse de l'université d'accueil, Dr Noreen Izza Arshad, ainsi que toute son équipe pour leur accueil dans au sein du laboratoire et pour leurs conseils sur la vie en Malaisie.

Je remercie également Jean François Monin d'avoir accepté d'être mon tuteur et d'avoir été à l'écoute en cas de besoin durant ces 12 semaines de stage.

J'adresse aussi mes remerciements à la région Auvergne-Rhône-Alpes pour m'avoir aider financièrement grâce à la bourse Explo'ra, sans laquelle ce voyage à plus de 10000km de chez moi aurait demandé beaucoup plus de sacrifices. Je remercie également mes parents qui ont pris en charge financièrement tous les frais supplémentaires, et sans oublier leur soutien moral.

Enfin, je souhaite remercier Thibaut Arnoux, Maxime Lordez et Oussama Belkacemi, amis et étudiants de Polytech Grenoble pour ces 3 mois de collocations et de bonne humeur. Je remercie encore plus chaleureusement Oussama Belkacemi, plus connu sous le nom de *Little Oussama*, pour avoir laissé ma carte bancaire se faire détruire par un distributeur de billet me permettant de me commander une nouvelle carte bancaire toute neuve.

Résumé

Dans le cadre de ma 4ème année d'étude en informatique à Polytech Grenoble, j'ai effectué mon stage de fin d'année à *Universiti Teknologi Petronas* en Malaisie. Ce stage s'est déroulé sur 12 semaines où j'ai été amené à concevoir et à développer une application mobile.

Le but de ce stage était de réaliser une application mobile visant à aider les personnes souffrant de bégaiement à corriger ce trouble de la parole. L'application est aussi destinée aux orthopédistes souhaitant suivre la progression de leurs patients utilisant l'application.

Ce document donne une vision globale du projet et des solutions apportées durant ce stage.

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Table des matières

1	Introduction	6
1.1	Contexte	6
1.2	Sujet proposé	6
1.3	But du rapport	7
1.4	Organisation du rapport	7
2	Projet réalisé	8
2.1	Analyse du marché et des besoins, élaboration du cahier des charges . .	8
2.1.1	Résumé de cahier des charges de l'application	8
2.2	Choix des technologies	10
2.2.1	Création de l'application	10
2.2.2	Stockage de données dans le cloud	12
2.3	Gestion de project	13
2.3.1	Planning prévionnel	13
2.3.2	Gestion de version (git)	13
2.4	Modélisation de l'interface graphique	14
2.5	Architecture de l'application	15
2.5.1	Interface graphique et noyau fonctionnel	15
2.5.2	Architecture des exercices	17
2.6	Developpement de l'application, tâches à finir, futur de l'application . .	19
2.6.1	Fonctionnalités developpées	19
2.6.2	Et ensuite ?	20
2.7	Utilisation du cloud	21
2.7.1	Authentification	21
2.7.2	Synchronisation des exercices	21

<i>TABLE DES MATIÈRES</i>	4
2.7.3 Sécurité	21
3 Développement durable	22
4 Bilan et conclusion	23
Annexes	25
A Description du sujet	26
B Stutter Manager v3	27
C Étude comparative	28
D Software requirements specification	29
E Software requirements specifications - Exemple d'un cas d'utilisation	30
F Diagramme de Gantt prévisionnel	31
G Exemple release Github	32
H Diagramme IHM	33
I Wireframes	34
I.1 Page principales	34
I.2 Exercices	34
I.3 Divers	36
J Diagramme de classe	37
K Stuttherapy - Captures d'écran	39

Table des figures

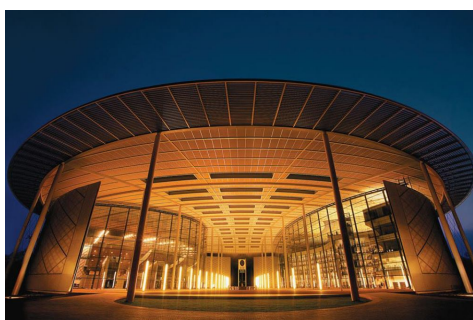
1.1	Universiti Teknologi PETRONAS	6
2.1	Diagramme cas d'utilisation	9
2.2	Sprint Scrum	13
2.3	Exemple du flowchart entre les pages <i>Exercice Homepage</i> et <i>Therapy Feed</i>	14
2.4	Exemple du flowchart entre les pages <i>Exercice Homepage</i> et <i>Wireframe de la page principale d'un utilisateur bogue</i>	15
2.5	<i>Behaviour subject</i> comportement	16
2.6	Universiti Teknologi PETRONAS	17
2.7	<i>Behaviour subject</i> comportement	17
2.8	De la structure à l'interface	18
2.9	Génération visuelle des réglages de l'exercice	19
3.1	Températures moyennes minimales et maximales en Malaisie, dans la capitale	22

Introduction

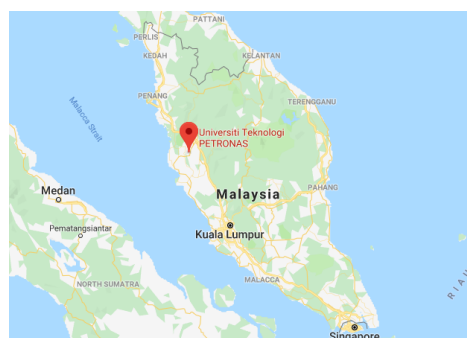
1.1 Contexte

Dans le cadre de ma 4^{ème} année d'étude à Polytech Grenoble en informatique, j'ai effectué mon stage de 12 semaines à Universiti Teknologi PETRONAS (UTP) à Seri Iskandar, en Malaisie. Ce stage s'est déroulé du **20 mai 2019** au **9 août 2013**.

À l'occasion de la mise en place d'un futur partenariat entre Polytech Grenoble et UTP, différents sujet de stage ont été transmis aux étudiants de Polytech par le bureau des relations internationales. Au final, nous étions 4 étudiants : j'ai été accompagné de 2 autres étudiants venant de la spécialité INFO (Informatique) et 1 étudiants de IESE (Informatique et Electronique des Systèmes Embarqués).



(a) Bibliothèque de l'université



(b) UTP, Seri Iskandar, Malaisie

FIGURE 1.1 – Universiti Teknologi PETRONAS

1.2 Sujet proposé

Le sujet proposé a pour objectif de développer une application mobile dans le but d'aider les personnes atteintes de bégaiement à surpasser ce trouble de la parole.

Une application similaire a déjà été développée durant 3 années par des étudiants dans le cadre de leurs *Final year project*. Comme illustré dans l'annexe B, cette application propose les fonctionnalités suivantes :

- Des exercices pour apprendre à contrôler son flux de parole ;
- La possibilité de visualiser sa progression pour chaque exercice ;
- Des informations concernant le bégaiement.

Cette application était disponible sur les appareils Android via le Play Store, la plateforme de téléchargement d'application développée par Google. Suite à un trop grand nombre de retours de bug concernant l'application, celle-ci a dû être retirée du Play Store.

Le sujet qui m'a été proposé est de recommencer depuis le début le développement de cette application, en reprenant donc les fonctionnalités précédemment présentées.

Dans sa version finale, l'application devra donc proposer différents exercices pour apprendre à contrôler son flux de parole ainsi que la possibilité de visualiser sa progression pour chacun des exercices accompagné de graphique illustrant la progression de l'utilisateur. De plus, l'application pourra être utilisée par des orthophonistes pour accéder à la progression de leurs patients. Ils pourront aussi donner des retours et des commentaires.

Aucune autre contrainte (technologies, organisations, etc.) m'a été imposée.

1.3 But du rapport

Ce rapport est destiné à tous ceux qui souhaitent avoir un aperçu global du projet et de ce qui a été réalisé durant ces 12 semaines. En particulier, ce rapport est un bon point d'entrée pour tous ceux qui souhaitent continuer le développement de *Stuttherapy*. Le rapport est disponible en version anglaise et française.

1.4 Organisation du rapport

Le rapport est tout d'abord constitué de cette partie, l'introduction où j'ai présenté le contexte dans laquelle le stage a été effectué et où j'ai donné une description succincte du sujet proposé.

Ensuite, la deuxième partie décrit le travail effectivement réalisé lors de ce stage, notamment tout le processus de réflexion, de recherches, de conception, d'organisation, de développement, de tests et finalement de production.

La troisième partie est consacrée aux connaissances que j'ai acquises et améliorées durant ces 12 semaines. Je reviendrai sur les erreurs que j'ai commises, leurs causes, leurs conséquences.

Avant de conclure ce rapport, une page sera consacrée au développement durable, en vertu de la loi Grenelle 1 de 2009 sur l'environnement.

Projet réalisé

2.1 Analyse du marché et des besoins, élaboration du cahier des charges

Afin de développer un produit qui puisse convenir aux besoins des personnes souffrant de bégaiement ainsi qu'aux orthopédistes, j'ai effectué quelques recherches pour définir les exercices utilisés par les orthopédistes pour aider leurs patients. Le projet devant être développé en seulement 12 semaines, j'ai décidé de concentrer mes recherches uniquement en ligne sans démarcher de réelles orthopédistes qui m'auraient permis de cerner plus précisément les besoins réelles mais qui m'aurait aussi pris beaucoup plus de temps.

J'ai également fait une étude comparative des applications actuellement disponible sur le marché des application Android, un tableau récapitulatif est disponible dans l'annexe C. Cette étude a relevé un manque d'application complète qui propose plusieurs exercices, les applications se concentrent souvent sur un seul exercice. Aussi, aucune application propose de faire le lien entre les bégues et leurs thérapeutes.

Afin de décrire complètement le but de l'application, le contexte d'utilisation dans lequel elle s'inscrit (*par qui ? comment ? pourquoi ?*), ses fonctionnalités et autres diverses exigences (sécurité, maintenabilité, etc), j'ai rédigé un *Software Requirements Specification (SRS)*. Un *SRS* est un document décrivant les exigences fonctionnelles et non fonctionnelles, les interactions de l'utilisateur sur le produit et les éventuelles contraintes à respecter (lois et réglementations, protocole à utiliser, limitations matériel, etc.). La table des matières de ce document est disponible dans l'annexe D. En particulier, ce document décrit précisément les interactions de l'utilisateur sur l'application récapitulées dans le diagramme de cas d'utilisation de la figure 2.1. L'annexe E contient un exemple de spécification d'un cas d'utilisation.

2.1.1 Résumé de cahier des charges de l'application

L'application est destinée à être utilisée par des personnes souffrant de bégaiement et par des orthopédistes.

Les bégues auront à disposition une liste d'exercices pour s'entraîner à mieux contrôler leur vitesse et leur rythme de parole. Chaque exercice sera paramétrable pour convenir aux besoins de l'utilisateur. En particulier les exercices pourront utiliser des ressources que l'utilisateur devra lire, ces ressources devront être récupérées via une

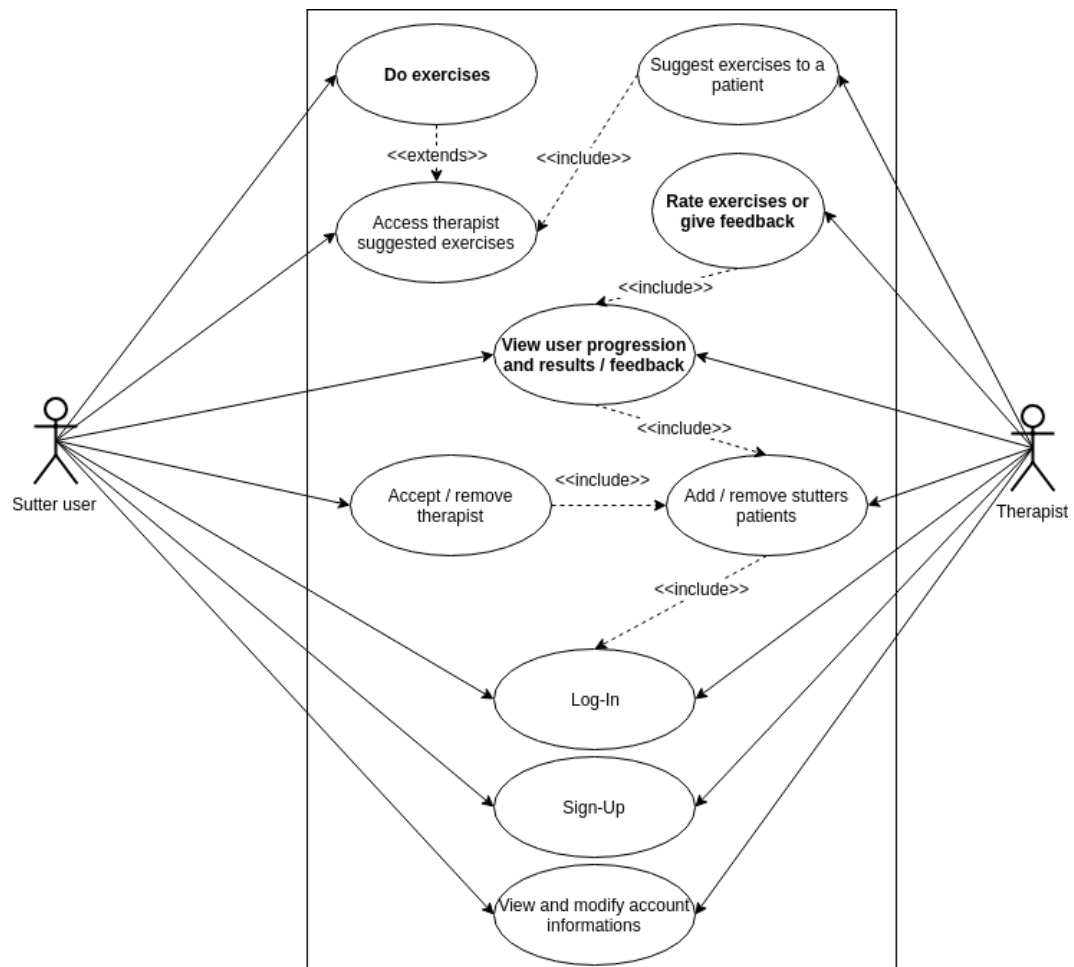


FIGURE 2.1 – Diagramme cas d'utilisation

collection de ressources partagé possédant plusieurs type de ressource : des mots, des phrases et des textes. Le bègue pourra choisir sur quelle type de ressources il veut s'entraîner pour chaque exercices. Les exercices pourront d'appuyer aussi sur d'autres éléments tel qu'un enregistreur de voix, un enregistreur vidéo, etc. L'utilisateur pourra choisir d'activer ou non ces enregistreurs. Par exemple voici 4 exercices types qui rentrent dans le cadre de l'application :

- **Metronome** : laisser parler le bègue avec un metronome (un metronome est un appareil émettant un signal visuel et sonore à interval de temps régulier) ;
- **Reading** : laisser parler le bègue librement sur les ressources de son choix (mots, textes, phrases) ;
- **Delayed Auditory Feedback (DAF)** : exercice où le bègue parle puis entend le retour de sa voix quelques dizaines de millisecondes après ;
- **Mirroring** : exercice où le bègue s'entraîne à parler avec un retour vidéo de lui pour analyser ses mouvements faciaux.

L'utilisateur bègue pourra accéder à sa progression. La progression est constituée de l'historique de tous les exercices effectués. Ces exercices contiendront l'éventuel enregistrement de sa voix ou de sa caméra, les ressources utilisées lors de cet exercice ainsi qu'un commentaire de son éventuel orthopédiste (voir ci-après). La progression de l'utilisateur pourra aussi être visualisée sous forme de courbe de pourcentage de réussite de prononciation des ressources de l'exercice. L'utilisateur bègue pourra décider de synchroniser des exercices dans le cloud. Pour ce faire il devra se créer un compte avec un nom, une adresse mail et un mot de passe. Une fois un exercice synchronisé dans le cloud, il sera accessible par son orthopédiste qui pourra alors ajouter un commentaire sur cet exercice. Ces utilisateurs peuvent donc ajouter un orthopédiste qui aura accès à tous leurs exercices synchronisés. Pour ajouter un orthopédiste ils devront connaître son identifiant personnel (voir ci-après). Ils pourront bien entendu révoquer l'accès de cet orthopédiste aux exercices à tout moment.

Pour utiliser les fonctionnalités de l'application, les orthopédistes devront se créer un compte (comme pour les bègues, avec un nom, un courriel et un mot de passe). Une fois le compte créé ils auront accès à leur identifiants personnels ainsi que la liste de tous les utilisateurs bègues les ayant ajoutés comme orthopédiste via leur application. Ces utilisateurs sont appelés *patients* pour l'orthopédiste. Un orthopédiste pourra visualiser la progression de ses patients. Il pourra aussi supprimer des patients de sa liste.

2.2 Choix des technologies

2.2.1 Création de l'application

Android et iOS sont les deux systèmes d'exploitation mobile les plus utilisés sur les smartphones en 2018, ils se partagent respectivement 74.45% et 22.5% du marché [1].

Il existe deux solutions pour développer une application mobile :

- *Application native* : Utilisation des SDK (*kit de développement logiciel*) spécifiques à chaque plateforme (iOS et Android) qui fournissent des outils de développement et de debuggages permettant de créer des applications pour ces plateformes ;
- *Application hybride* : Utilisation d'un *framework multi-plateforme* permettant la création d'application sur plusieurs plateformes avec le même code sans utiliser directement les SDK de ces dernières.

L'avantage des applications natives sur les applications hybrides sont :

- La disponibilité de toutes les, et notamment des dernières, fonctionnalités des différents systèmes d'exploitations ;
- La performance de l'application en accédant directement au système d'exploitation et ses fonctionnalités.

L'inconvénient majeur est qu'il est nécessaire de développer, maintenir et deployer la même application 2 fois.

Les framework multi-plateformes permettent d'écrire des application pour iOS et Android avec les même code (bien qu'il est aussi possible d'écrire des partie pour une plateforme spécifique) permettant d'avoir un *time to market* (temps nécessaire pour rendre le produit sur le marché) plus court. Ces framework propose aussi souvent des solutions modernes permettant encore de réduire ce *time to market*. Les inconvénients de ces frameworks sont propres à chacun. Il en existe de nombreux. Pour commencer la comparaison j'ai choisis les frameworks activement développés avec une communauté importante et active. Les frameworks les plus populaires en 2019 sont **Flutter** (71 662 stars avec 422 contributeurs sur Github [2]), **React Native** (79 513 stars avec 1990 contributeurs [3]) et **Ionic** (38 705 stars avec 331 contributeurs [4]).

J'ai finalement opté pour **Flutter**, le framework multi-plateforme développé par Google. Flutter utilise le langage de programmation *Dart*, un langage orienté objet utilisant un *garbage collector*. J'ai choisi Flutter pour plusieurs raisons :



Performance Le code de l'application est compilé à l'avance en code ARM natif et non pas au moment de l'exécution comme React Native le fait, ce qui permet d'avoir des performances similaire que applications natives. Ça semble être le framework le plus performance du marché.

Composants graphiques Flutter gère entièrement le rendu des éléments graphiques sur son propre canvas sans utiliser les composants graphiques natifs aux plateformes (comme le fait React Native). On peut donc gérer au pixels près les éléments graphiques utilisés. L'application aura la même apparence indépendamment des versions des systèmes

d'exploitation. Cela permet de gérer précisément ce qu'il sera affiché sur tous les appareils.

2.2.2 Stockage de données dans le cloud

L'application doit proposer le partage de données entre les bégues et les thérapeutes. Pour ce faire les données doivent être stockées dans le cloud. J'ai choisi d'utiliser un système d'authentification pour les utilisateurs avec adresse mail et mot de passe pour qu'ils puissent gérer leurs données et que ces dernières soient uniquement accessibles aux personnes autorisées (et non pas sur un espace "publique").

J'ai choisi d'utiliser **Firebase**, en particulier la base de données noSQL **Firestore** et le système d'authentification de Firebase pour gérer les utilisateurs. J'ai choisi Firebase pour plusieurs raisons :



Facilité d'utilisation Google développe et maintient à jour un ensemble de plugins permettant d'utiliser les produits Firebase facilement et rapidement : *FlutterFire* [5]. Les plugins offrent les dernières fonctionnalités des produits Firebase.

Evolution de l'application sur d'autres plateformes Firebase est largement utilisé aujourd'hui, les produits Firebase peuvent être utilisés aussi bien sur les web que sur applications mobiles facilement grâce à un grand nombre de plugins développés par les communautés des différents framework / langages. Il peut aussi être utilisé directement en utilisant les API des produits avec les langages supportés.

Evolution de l'utilisation du Cloud au sein de l'application Pour l'instant, seuls *Firestore* et le système d'authentification de Firebase sont utilisés. Cependant Firebase intègre bien d'autres services pouvant potentiellement être utilisés par l'application. Parmi eux, on peut citer *Firebase Analytics* permettant d'analyser le comportement des utilisateurs sur l'application, *Firebase AdMob* pour intégrer des publicités au sein de l'application ou encore *Cloud Functions* pour exécuter des fonctions sur des serveurs. Tous ces produits peuvent être étroitement utilisés ensemble, par exemple *Cloud Firestore* peut automatiquement déclencher une fonction de *Cloud Functions* lors de l'ajout d'un nouveau document dans la base de données. Donc, **sans** modifier le code de l'application côté client, il est possible d'enrichir ou de modifier les fonctionnalités de l'application en utilisant les différents services Firebase (authentification, publicité, base de données, stockage, etc.).

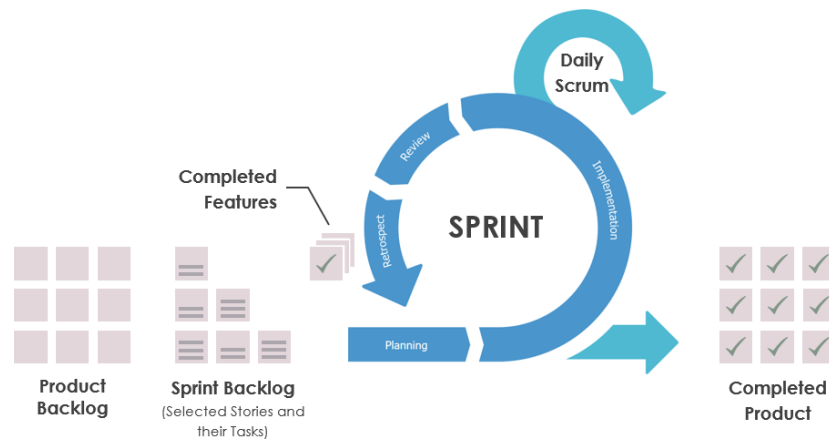


FIGURE 2.2 – Sprint Scrum

2.3 Gestion de project

2.3.1 Planning prévisionnel

J'ai tout d'abord créer un diagramme de Gantt (voir annexe [E](#)) pour m'aider à planifier les différentes tâches que j'ai à faire. Pour élaborer ce diagramme de Gantt je me suis inspirer de la notion de *sprints* de la méthode Agile Scrum [6]. Un sprint, tel qu'utilisé dans ma gestion de projet, est une phase de developpement de l'application d'une durée de 1 semaine. Au terme d'un sprint, les fonctionnalités développer dans la semaines doivent être terminées, testées et prêtent à être disponible en production.

Sur les 12 semaines de stage, 7 semaines ont été consacré au developpement de l'application. J'ai donc, en début de projet, définis les fonctionnalités de chacun des 7 sprints prévu. Les première semaines ont été consacré à la prise en main du sujet, aux recherches, à l'élaboration du cahier des charges et à la conception de l'architecture de l'application. Les dernières semaines ont été réservé au packaging final de l'application, au deploiement de l'application sur le *Play Store* et à ce rapport.

Le planning du projet final, après modification de celui-ci au cours du projet, est disponible dans la partie [Bilan et conclusion](#) de ce rapport.

2.3.2 Gestion de version (git)

J'ai choisis d'utiliser le système de gestion de version Git avec le service d'hébergement Github pour héberger le projet afin de donner un accès rapide et complet à mon travail à mon enseignant référent et à ma superviseure de stage.

Github m'a aussi permis de créer des point de sauvergarde du projet, appelé *release*, permettant de télécharger le code source des versions majeures de l'application

avec commentaires de ce qui à changé depuis la dernière version ainsi que le fichier d'installation de l'application pour les appareils Android. La capture d'écran de la *release* finale du projet est disponible dans l'annexe G.

2.4 Modélisation de l'interface graphique

A partir du cahier des charge des l'application, j'ai construit le flowchart (*représentation schématique d'un enchainement d'action*) de l'interaction utilisateur sur les différentes pages de l'application, voir l'annexe H. J'ai tout d'abord listé les différentes pages que doit composer l'application. Une page est représentée par un rectangle avec le nom de la page accompagné d'un bref descriptif. La navigation entre deux pages est représenté par une flèche directionnel vers la page de destination avec le déclencheur qui provoque le changement de page (représenté par un losange).

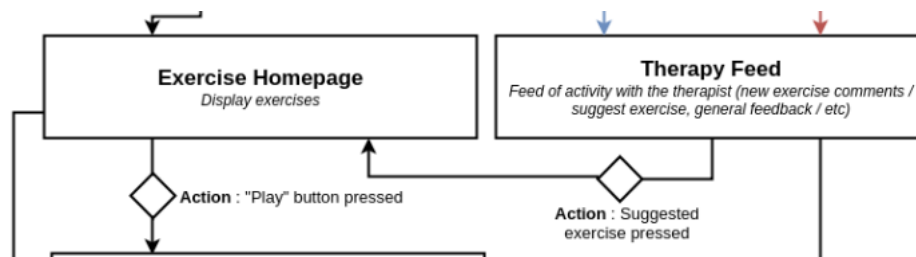


FIGURE 2.3 – Exemple du flowchart entre les pages *Exercice Homepage* et *Therapy Feed*

Une fois, les différentes pages de l'application définie, je me suis attelé a faire élaboré les *wireframes* de l'application. Un *wireframe* est une maquette fonctionnelle, c'est à dire une representation schématique de l'interface pour définir les différents composants de l'interface sans se soucier des règles de design comme les couleurs ou la typographie par exemple. Les *wireframes* se concentrent sur l'ergonomie de l'application et non sur le desgin de celle-ci. La figure 2.4 montre un exemple du *wireframe* de la page principale d'un utilisateur bègue. Tous les *wireframes* de l'application sont disponible dans l'annexe I.

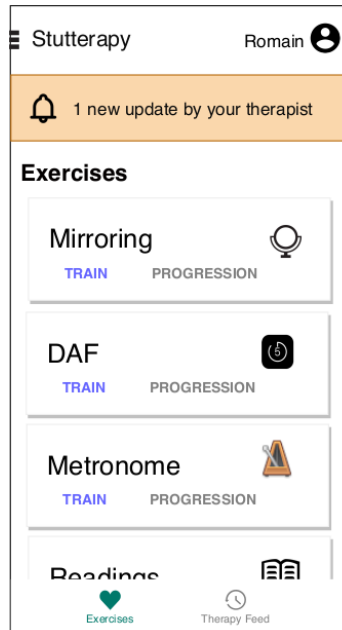


FIGURE 2.4 – Exemple du flowchart entre les pages *Exercice Homepage* et *Wireframe de la page principale d'un utilisateur bègue*

2.5 Architecture de l'application

Dans cette partie je vais m'attarder premièrement sur l'architecture choisie pour la communication entre l'interface graphique et le noyau fonctionnelle de l'application, puis enfin je décrirais l'architecture adoptée pour implémenter les différents exercices au sein de l'application.

Le diagramme de classe de l'architecture du projet avec les différentes classes qui ont été utilisées lors du projet est disponible en annexe J.

2.5.1 Interface graphique et noyau fonctionnel

La partie graphique de l'application et la partie fonctionnelle doivent bien entendu être distinctes. De plus la partie fonctionnelle doit être indépendante de la partie graphique. Cela permet par exemple de déployer une application web en construisant seulement une nouvelle interface graphique en utilisant la même partie fonctionnelle.

L'architecture de l'application est un choix crucial dans le développement d'application, la modification des données de la partie fonctionnelle doit se refléter sur la partie graphique. Un composant graphique dans le monde de Flutter est appelé *widget* (bouton, texte, layout, etc.). La programmation mobile est asynchrone et réactive, des données

peuvent être modifiée, par exemple, après un appel réseau, par un utilisateur extérieur ou encore par un autre appareil, et les widgets doivent réagir à ces changements.

J'ai choisis d'ajouter une couche de providers entre le niveau graphique et le niveau fonctionnelle. Ces providers fournissent les données nécessaire au monde graphique par l'intermédiaire de *stream* (équivalent aux *observable* dans ReactiveX). Un stream peut être vu comme un tuyau unidirectionnel ici, le provider insère des éléments dans ce tuyau, et la partie graphique à accès à la deuxième extrémité du tuyau pour accéder à l'élément qu'il y a dans le tuyau. Chaque nouvel élément inséré dans le *stream* remplace l'éventuel ancien élément.

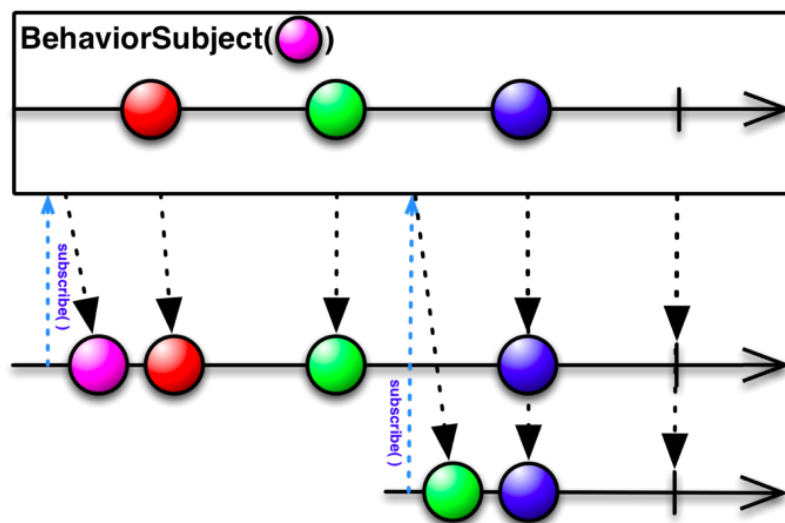


FIGURE 2.5 – *Behaviour subject* comportement

Il existe le widget *StreamBuilder* qui permet d'écouter la sortie d'un *stream*. A chaque nouvel élément dans le stream les fils de ce widget seront reconstruit et redessiner à l'écran. Par exemple, si le champ texte de l'arbre de widget de la figure 2.6a dépend d'une donnée de la partie fonctionnelle qui est susceptible de changer au cours du temps et donc qui est disponible via un *stream*, il suffit d'ajouter comme parent à ce widget texte le widget *StreamBuilder* qui va avoir accès à l'extrémité du tuyau contenant la donnée à afficher. Ce widget et sa descendance se reconstruire automatiquement à chaque fois que le contenu du stream change (càd que le provider met une nouvelle donnée à afficher dans le tuyau).

Cette architecture permet de reconstruire seulement les parties de l'interface graphique qui ont **besoin** d'être reconstruite suite à la modification ou à l'ajout d'une donnée, qui se passe donc du côté fonctionnelle de l'application. C'est une architecture bien plus performante qu'une solution basique qui consisterait à reconstruire tout l'arbre de widget à partir de la racine à chaque changement.

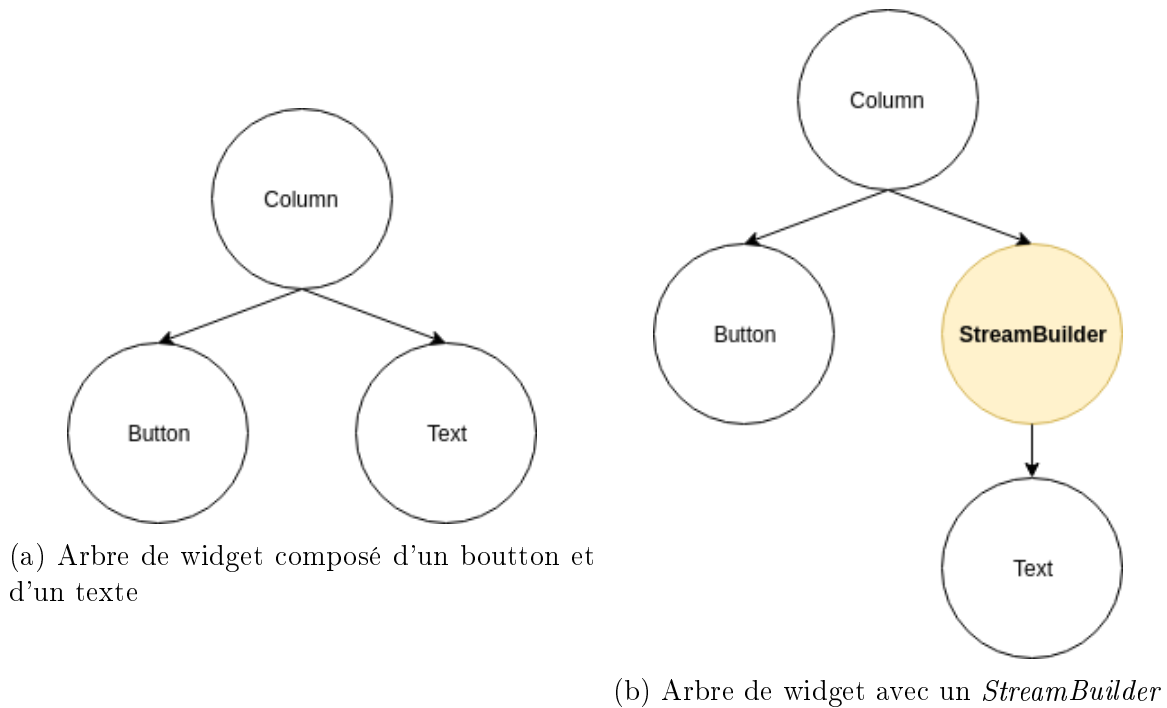
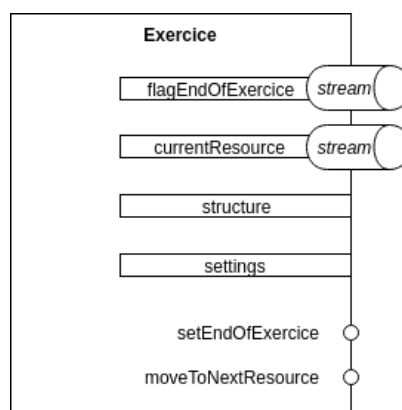


FIGURE 2.6 – Universiti Teknologi PETRONAS

2.5.2 Architecture des exercices

Un point intéressant à concevoir était la structure des exercices au sein de l'application. Les exercices sont différents mais partagent tout de même beaucoup de point commun, il était nécessaire de trouver une architecture qui permette d'implémenter différents exercices rapidement sans redeveloppé toute la base de de l'exercice à chaque fois.

FIGURE 2.7 – *Behaviour subject* comportement

L'architecture choisie est la suivante : un exercice possède simplement les attributs suivants :

- Une **strucutre** : la liste des éléments constituant l'exercice. Plusieurs éléments peuvent être utilisé par un exercice, par exemple : un enregistreur vocal, un afficheur de ressource (mot, phrase, texte), un metronome, etc. ;
- Des **réglages** : les réglages de l'exercice (choisi par l'utilisateur ou directement imposé par l'exercice). Ces réglages sont identifié par un identifiant unique et associé à une valeur (booléen, entier, etc.) ;
- La **resource courrante** : la resource actuellement utilisé par l'exercice (mot, phrase, texte) ;
- Un **flag** pour signaler la fin de l'exercice.

Note : pour faire simple j'ai omis certaines informations comme la date de création, l'intitulé, etc).

Structure Comme expliqué précédemment, la structure d'un exercice est consitué de plusieurs éléments (concrètement ces éléments sont les valeurs d'un type énuméré). Côté graphique, il y a un convertisseur de ces valeurs en oject graphique. Par exemple l'élément *metronome* sera associé à un widget qui affiche un métronome avec un signal visuel et un signal sonor à tempo régulier. Ces objets graphiques ont accès aux attribue et à certaines fonctions de l'exercice. Par exemple le widget permettant d'afficher une resource (mot, phrase, texte), a accès à la resource courante de l'exercice et peux appeller la fonction `moveToNextResource` pour changer la resource courante. Ces widgets peuvent aussi accéder aux réglages de l'exercices.

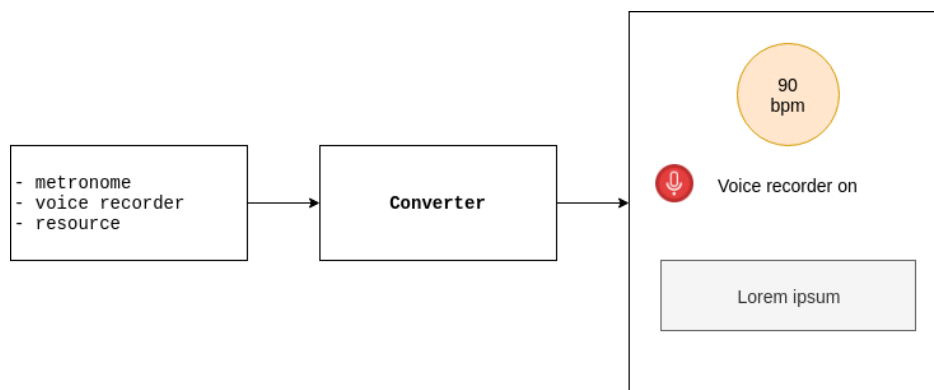


FIGURE 2.8 – De la structure à l'interface

Réglages Chaque exercice peut définir sa liste de réglages. Ces réglages sont destiné à être utilisé par les widget associé aux éléments de la structure des exercices. Par exemple l'exerice du metronome peut définir un réglages `metronome_bpm` qui va permettre au widget affichant le metronome de régler la valeur du tempo suivant ce réglage. Comme pour les éléments de la structure des exercices, il y a un convertisseur des réglages en

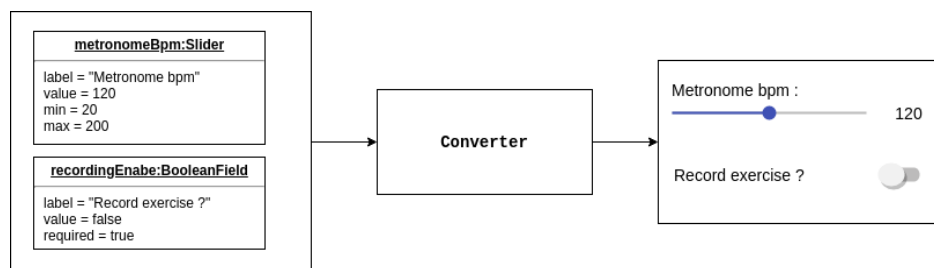


FIGURE 2.9 – Génération visuelle des réglages de l'exercice

Cette architecture permet de construire facilement et rapidement un large panel d'exercices. Pour structurer un nouvel exercice il suffit de piocher dans la collection déjà existante si tous les éléments nécessaire y sont disponible. Dans le cas échéant, il suffit juste de développer l'élément qu'il manque.

2.6 Développement de l'application, tâches à finir, futur de l'application

2.6.1 Fonctionnalités développées

L'application se nomme **Stuttherapy**. Voici la liste (presque) exhaustive des fonctionnalités développées durant les 7 semaines consacrées au développement :

- **Pour un utilisateur bègues :**
 - Trois différents exercices disponibles : *metronome*, *mirroring* et *reading* (voir [Résumé de cahier des charges de l'application](#) pour plus d'informations concernant le but de ces exercices) ;
 - Progression pour chacun des exercices disponible sous deux formes :
 - Liste de tous les exercices effectués avec les informations et données de l'exercice en question : date de création, éventuel enregistrement audio ou vidéo et les éventuels mots dont l'utilisateur n'a pas réussi à prononcer correctement ;
 - Une représentation graphique *ligne* du pourcentage de réussite de prononciation des mots au cours des différents exercices. L'utilisateur peut changer la fenêtre de temps du graphique pour afficher sa progression hebdomadaire, mensuelle ou annuelle.
 - Accès à sa liste de mots *enregistrés*. L'utilisateur peut, si il le souhaite, indiquer les mots avec lesquels il a eu des difficultés lors des exercices, ils seront alors automatiquement enregistrés dans cette liste ;
 - Synchroniser et récupérer des exercices dans le cloud : l'utilisateur peut choisir de synchroniser ou desynchroniser les exercices qu'il veut dans le cloud grâce à un compte utilisateur. Il peut récupérer les exercices synchronisés dans

- le cloud sur son appareil (en cas de changement de téléphone par exemple) ;
- Ajouter un orthopédiste autorisé à avoir accès à ces exercices synchronisés dans le cloud : l'utilisateur peut, à partir de l'identifiant de l'orthopédiste, l'autoriser à accéder à sa liste d'exercices synchronisés dans le cloud et à y ajouter des commentaires. Il peut aussi le supprimer à posteriori.
- **Pour un orthopédiste :**
 - Visualisation des exercices synchronisés des utilisateurs bègues l'ayant ajouté en tant qu'orthopédiste (un tel utilisateur est appelé *patient* pour l'orthopédiste)
 - Suppression d'un patient

Des captures d'écran de l'application sont disponibles dans l'annexe [K](#).

2.6.2 Et ensuite ?

L'application est entièrement fonctionnelle et aucun bug connu n'a été identifié. Toute les fonctionnalités décrites dans le *software requirements specification (SRS)* rédigés en début de projet ont été développées. Il reste cependant quelques améliorations à fournir pour coller complètement à la spécifications de celles-ci données dans le *SRS* :

- Acutellement, lorsque l'utilisateur accède à son historique d'exercices effectués, les ressources utilisées lors de celui-ci ne sont pas affichées, seuls le date d'entraînement, l'éventuel enregistrement vocal / vidéo et les mots non prononcé correctement sont disponibles ;
- Lorsque un utilisateur synchronise un exercice dans le cloud, l'éventuel enregistrement vocal ou vidéo de l'exercice n'est pas sauvegardé dans le cloud. Seul l'URI du fichier local est sauvegardé dans la base de donnée, donc seul l'utilisateur avec le fichier déjà présent dans son téléphone peut y avoir accès. Il faudrait sauvegarder ces enregistrements dans un espace de stockage (par exemple *Firebase Stockage* pour être consistant avec les solutions acutellement utilisées) et ajouter le chemin d'accès vers ce fichier dans le document de la base de donnée correspondant à l'exercice. Il faudrait alors réfléchir au cout mémoire (et donc économique) que cela engendrerait. Il serait surrement nécessaire de fixer un quota de sauvegarde par utilisateur et utiliser une compression pour stocké ces enregistrements audios et vidéos ;
- La possibilités de supprimé définitivement sont compte utilisateur du cloud ainsi que ces données ;
- Proposer à l'utilisateur les options de connexions via les réseaux sociaux (*Facebook*, *Twitter*, etc.). Cela est facilement réalisable avec le système d'authentification utilisé actuellement : *Firebase Authentication*.

Il y a 3 types de tests à utilisé lorsqu'on developpe une application Flutter pour être sûr que les fonctionnalités implémentées fonctionnent bien et apparaissent correctement dans l'interface graphique :

- **Les tests unitaires** : pour tester une fonction, une méthode ou une classe ;

- **Les tests d'intégration** : pour tester une partie plus large de l'application, pour vérifier que tout marchent bien ensemble ;
- **Les tests de widget** : pour tester que l'interface graphique se comporte bien comme prévu.

Quelques tests unitaires ont été écrit, cependant toutes les fonctionnalités de l'application ne sont pas couvertes par ces tests. Il sera donc **nécessaire** de compléter ces tests avant d'ajouter de nouvelles fonctionnalités à l'application. Il n'y a malheureusement pas de tests d'intégration ni de test de widget à l'heure où j'écris ces lignes.

2.7 Utilisation du cloud

2.7.1 Authentification

2.7.2 Synchronisation des exercices

2.7.3 Sécurité

security rules
ddsfd

Développement durable

A l'heure où la France s'inquiétait de l'impact des climatiseurs pour refroidir nos pièce sur la planète, la Malaisie, et notamment UTP ne semblent par encore être préoccupé par ce problème.

Les températures extérieures étant très élevées en Malaisie, tous les batiments (gare, restaurants, hotels, etc.) et les voitures sont équipés de climatiseurs. Cependant le réglages de ces derniers sont souvent bien trop froid ($< 20^{\circ}\text{C}$) et nous oblige à porter des vetements chauds et long à l'intérieur des batiments, malgré la temparature dépassant largement les 30°C à l'extérieur sur l'ensemble de notre séjour.

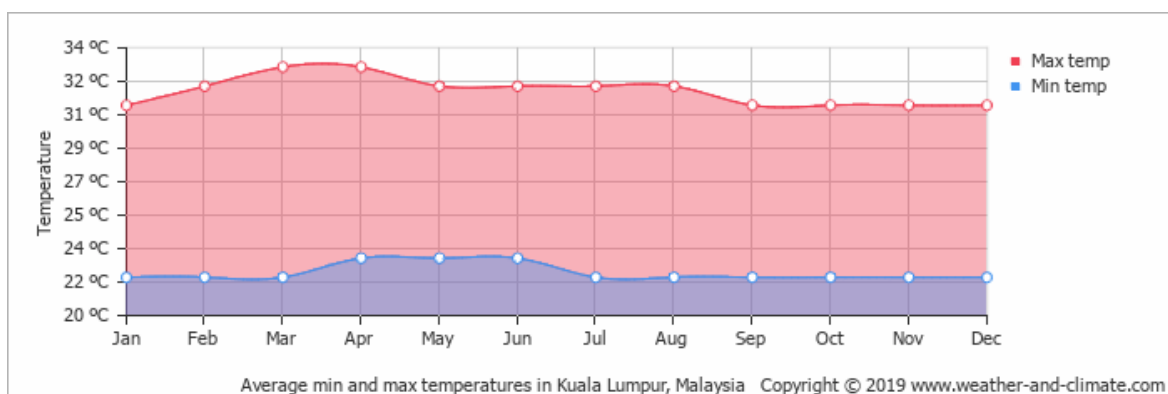


FIGURE 3.1 – Temperatures moyennes minimales et maximales en Malaisie, dans la capitale

Malgré l'usage excessifs des climatiseurs, les batiments récents de l'université ne sont en général pas dotés d'une isolation thermique comparable aux batiments récents construit en France. Pour ne prendre qu'un exemple, je travaillais dans la bibliothèque universitaire, cette dernière possède une immense façade vitrée bordée de porte, elles aussi vitrées, tout le long laissant la fraîcheur du batiments se faire ressentir sur plusieurs mètre à l'exterieur.

D'après le site de l'université [7], une attention particulière est mise en oeuvre pour favoriser la diversité des genres au sein de l'université, que ce soit au niveau des étudiants, des employés et du personnel pédagogique.

Bilan et conclusion

Bibliographie

- [1] Martyn Casserly. iphone vs android market share. <https://www.macworld.co.uk/feature/iphone/iphone-vs-android-market-share-3691861/>, 2019.
- [2] Flutter. Github - flutter. <https://github.com/flutter/flutter>, 2019.
- [3] Facebook. Github - react native. <https://github.com/facebook/react-native>, 2019.
- [4] Ionic. Github - ionic. <https://github.com/ionic-team/ionic>, 2019.
- [5] Google. Flutterfire - documentation. <https://firebaseopensource.com/projects/flutter/plugins/>.
- [6] Scrum.org. What is a sprint in scrum? <https://www.scrum.org/resources/what-is-a-sprint-in-scrum>.
- [7] UTP. Engineering gender diversity. <https://www.utp.edu.my/Pages/The-University/Publications/UTP-Impact/Engineering-Gender-Diversity.aspx>, 2019.

Annexes

Description du sujet

Internship Student Name: Romain Guillot

Title : Developing A Service Platform for Electronic Therapy for Stutterers

Duration : May 2019 till July/August 2019

Students' email: romain.guillot@etu.univ-grenoble-alpes.fr

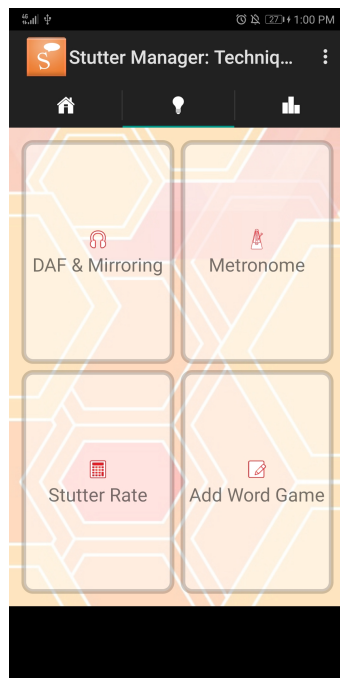
Job Descriptions:

1. To develop a mobile application for the stutterers to conduct electronic speech exercises.
The mobile app should have the following features:
 - a. Metronome exercises & results
 - b. Mirroring exercises & results
 - c. Delayed Auditory Feedback exercises & results
 - d. Stutter Rate exercises & results
 - e. Add word games exercises & results
 - f. Dashboard that charts progress by exercises (by day, week, month and year)
 - g. Dashboard that charts overall progress
2. To conduct a black box testing of the features developed
3. To prepare a documentation for administrator
4. To prepare an electronic user manual for the developed mobile application
5. To design a video that explains about the mobile application
6. To upload the tested mobile application in Google Playstore

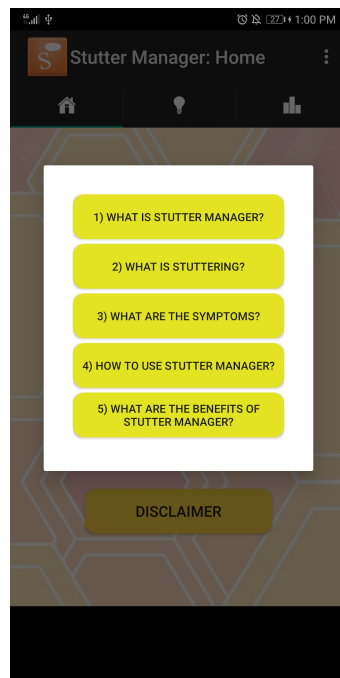
Description du projet proposé par ma superviseure Dr. Noreen Izza Arshad

Stutter Manager v3

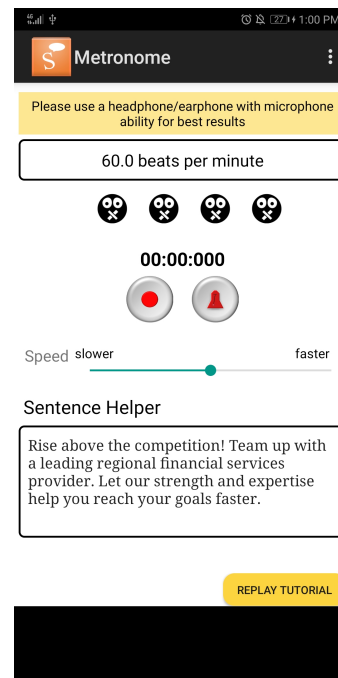
27



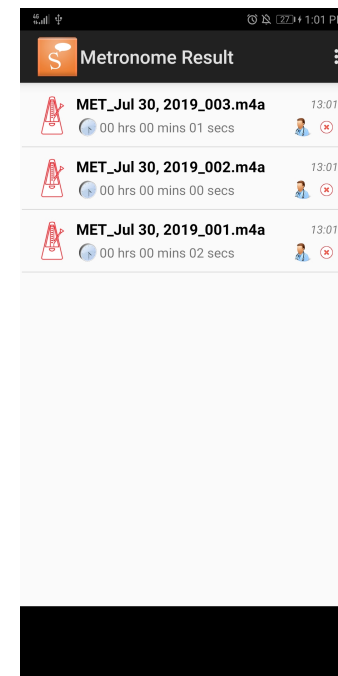
(a) Page principale (exercices)



(b) Informations générales



(c) Exercice : Metronome



(d) Progression de l'exercice metronome

Captures d'écran de Stutter Manager v3

Étude comparative

	Stuttering Speech Therapy	<u>Stamurai</u>	Stutter Help	Stuttherapy
Economic model	<i>Free</i>	<i>Free+in-app</i>	<i>Free</i>	<i>TBD</i>
General informations about stuttering (definition, causes, categories, etc)		✓		✓
Guide to learn how to breath, how to regulate its flow, etc.		✓		✓
Metronome	✓			✓
Delayed auditory feedback	✓	<i>In-App</i>	✓	✓
Mirroring	✓			✓
Reading exercises (text or word)		✓		✓
Progression		✓		✓
Communication with a therapist				✓
Voice recognition				

Étude comparative des applications disponibles sur le Play Store en comparaison avec les fonctionnalités prévues pour Stuttherapy

Software requirements specification

Table of contents

- Stuttertherapy : Software requirements specification
 - Change history
 - Table of contents
 - Introduction
 - Purpose
 - Scope
 - Definitions, acronyms, and abbreviations
 - References
 - Overview
 - General description
 - Product perspective
 - Product functions
 - Constraints
 - Specific requirements
 - Data Synchronisation
 - Use-case : Sign-Up
 - Use-case : Log-In
 - Use-case : View and modify account informations
 - Use-case : Handle stutter patients
 - Use-case : Handle allowed therapists
 - Use-case : View exercices results / progression / feedbacks
 - Use-case : Communicate with patient
 - Use-case : Do exercises
 - Use-case : Suggest exercises to a patient
 - Use-case : Access therapist suggested exercises
 - Conclusion
 - Appendix
 - A : Shared storage database

Table des matières du *Software requirements specification*

The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

Wikipedia - Software Requirements Specification

La spécification des exigences logicielles définit les exigences fonctionnelles et non fonctionnelles. Elle peut inclure un ensemble de cas d'utilisation décrivant les interactions de l'utilisateur que le logiciel doit fournir à l'utilisateur pour obtenir une interaction parfaite.

Traduction du passage ci-dessus

Software requirements specifications -

Exemple d'un cas d'utilisation

Use-case : Handle stutter patients

	Description
Function	Add patient (stutter user) to follow to track its progression / give feedbacks / ...
Requirements	Internet connection - Therapist account
Description	<p>A therapist is able to add stutters to follow to :</p> <ul style="list-style-type: none"> - Track their progressions - Give results, feedback of exercises done by the stutters - Give exercise to stutters <p>To add a patient the therapist has to have the stutter user key (a unique key associated to the account). This key is available in the app, in the account informations. The stutter informations is not directly available for the therapist, the stutter have to accept the therapist request, to summarize there are 4 states after the request :</p> <ul style="list-style-type: none"> - Request pending : the request has be sent but the stutters user do not yet approved ; - Request approved : stutters accept the request, the therapist can access to the stutters informations ; - Request refused : the stutters reject the request, the therapist cannot access to the stutters informations ; - Request revoke : the stutter user revoke the access, this state is possible only if the stutter user aproved the request before. <p>The therapist can see at any time the list of his patients and delete them of his list of patients.</p>

Spécification du cas d'utilisation **Handle stutter patients**

31



Exemple release Github


Latest release

2.0

2433d3b

Verified





Stable version 2

 develob released this 5 days ago

Authentication system with Firebase Auth
Therapist account : patients list / patients progress / remove patient / progress feedback
Progress charts
Bug fix and minor improvements:

- startup : overflow UI fix
- add alert dialog to erase local progress
- AM/PM date format
- Metronome audio signal
- recording permissions first startup caused crash, now it's fix
- migration to AndroidX to use Firebase Auth

▼ Assets 4

 bundle.aab	17.6 MB
 stuttherapy.apks	57.9 MB
 Source code (zip)	
 Source code (tar.gz)	

Release finale de l'application

Diagramme IHM

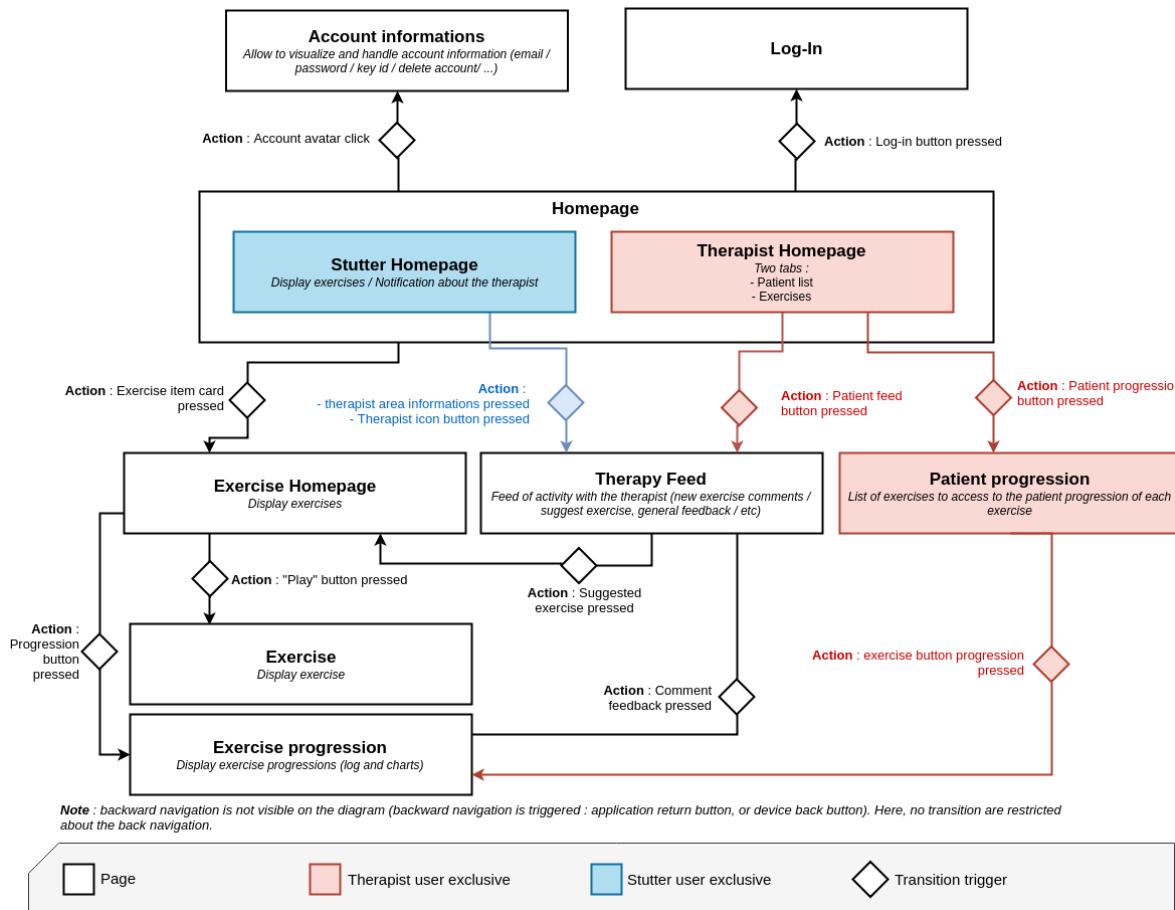
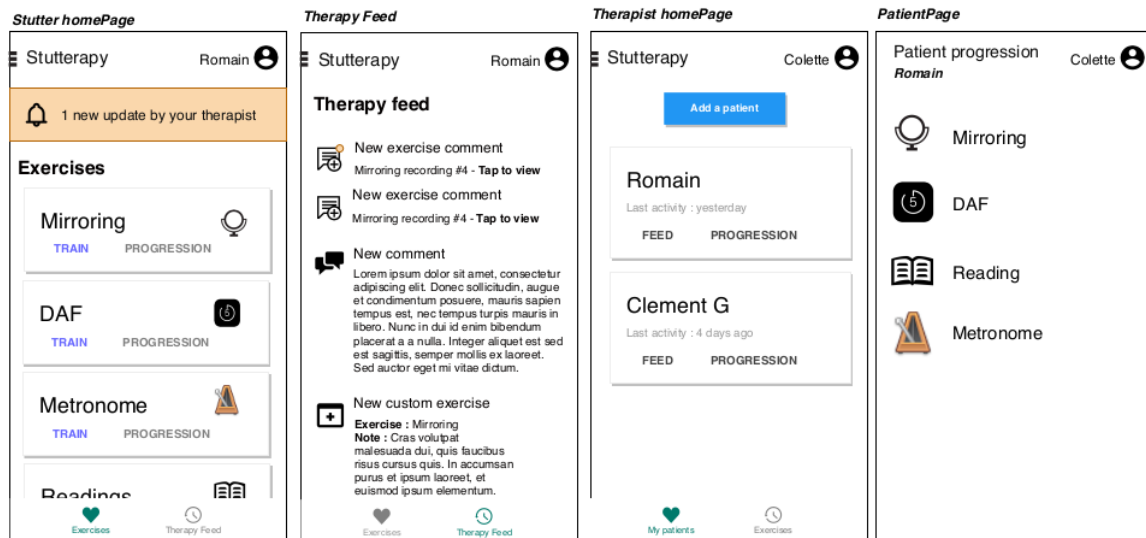


Diagramme IHM

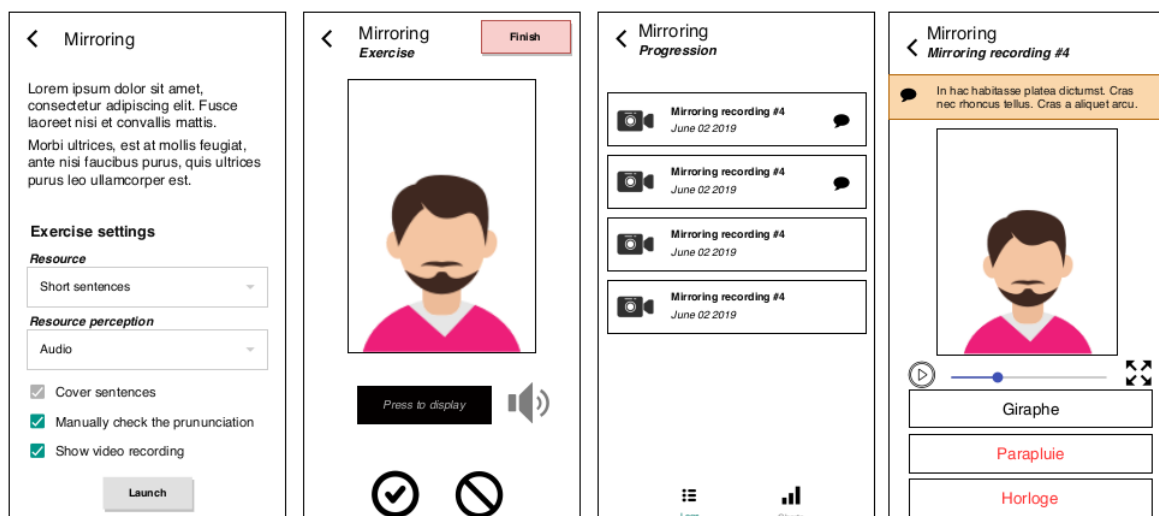
Wireframes

I.1 Page principales



Pages principales des bégues et des orthopédistes

I.2 Exercices



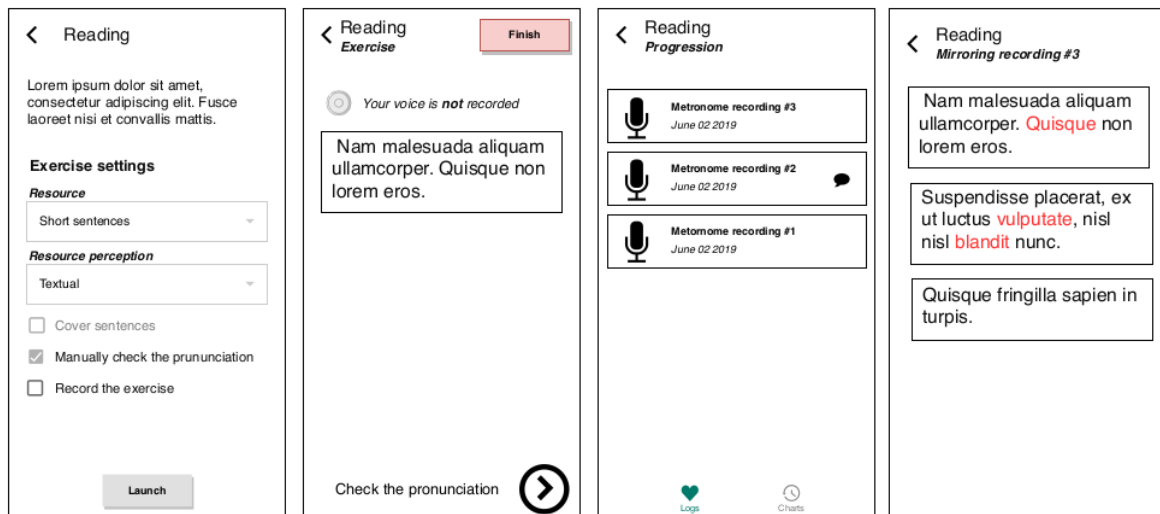
Exercice : Mirroring



Exercice : Metronome

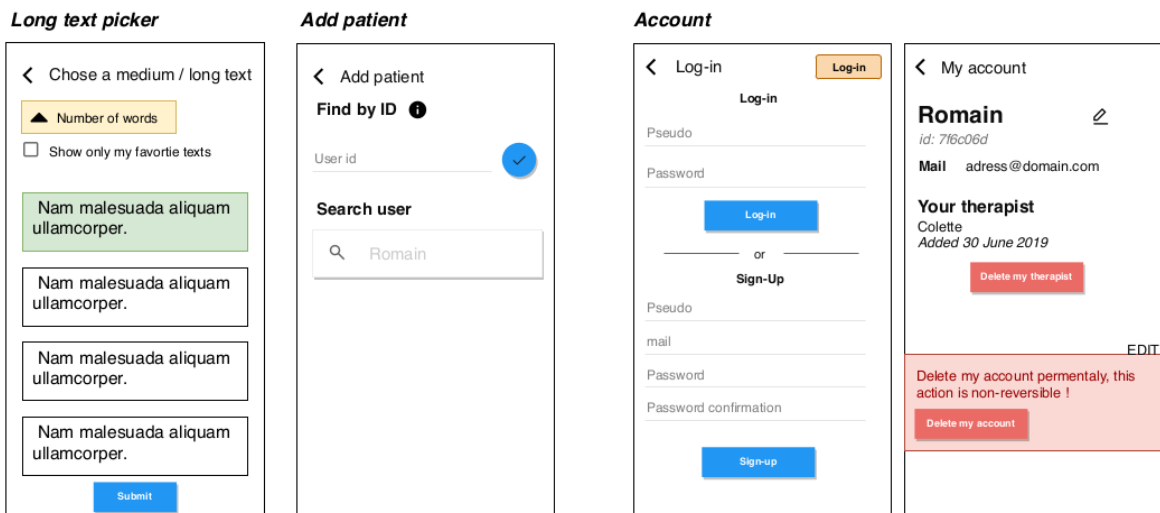


Exercice : DAF (delayed auditory feedback)



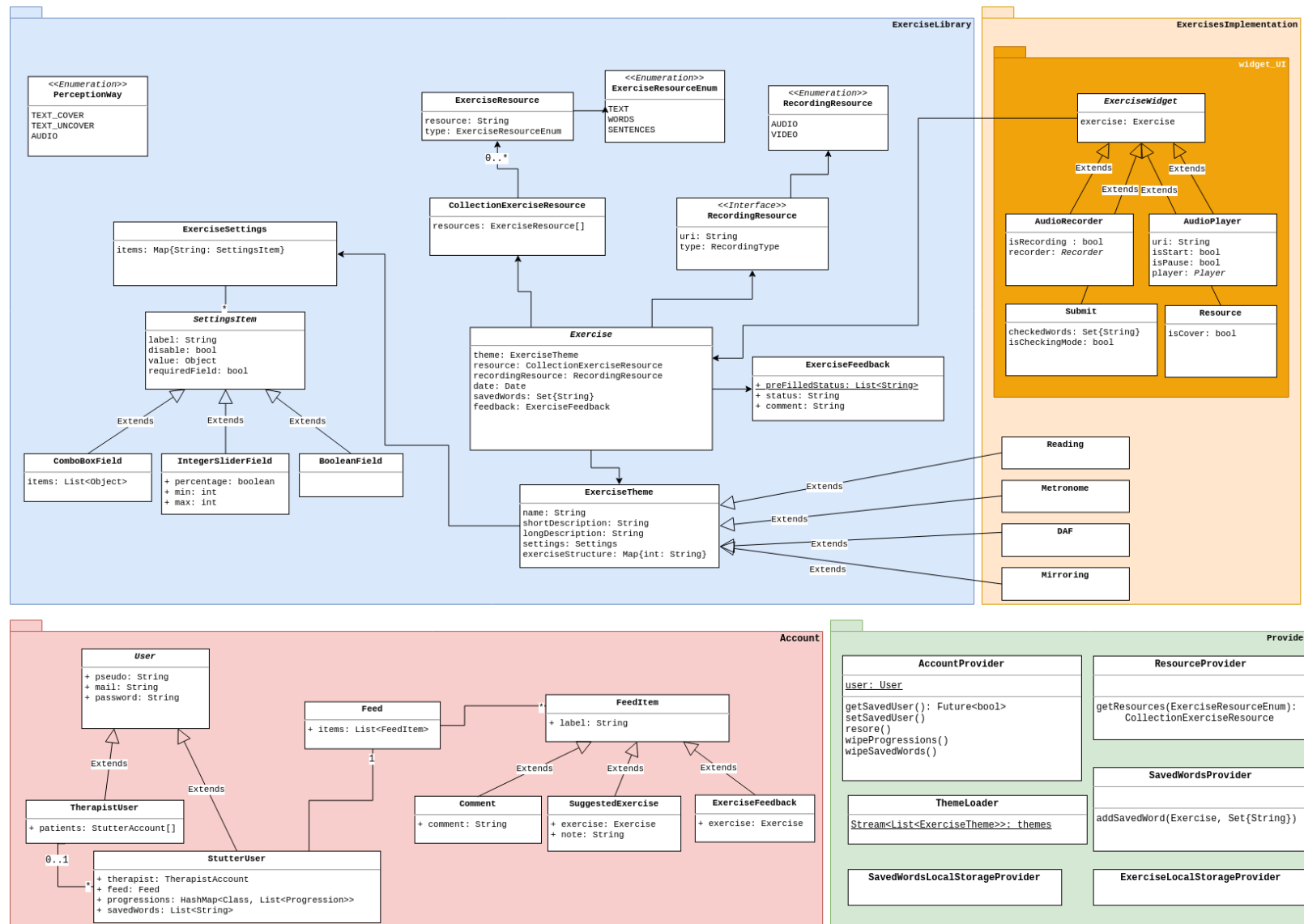
Exercice : Reading

I.3 Divers



Sélecteur de texte / ajout d'un patient / informations sur le compte

Diagramme de classe



Stuttherapy - Captures d'écran

Étudiant : Romain GUILLOT

Année scolaire : 2018 - 2019

Entreprise :

Adresse postale :

Téléphone :

Responsable administratif :

Téléphone :

Courriel :

Tuteur de stage :

Téléphone :

Courriel :

Enseignant-référent :

Téléphone :

Courriel :

Titre :

Résumé :