

Full Stack Application - T3A2-A



Scaler

Support App

By Celine Roure

Overview

- 1 - Description
- 2 - Data flow diagram
- 3 - App Architecture Diagram
- 4 - User Stories
- 5 - Wireframes
- 6 - Planning methodology



Description

Purpose:

The product: Scaler is a musical plugin for musicians, to create progressions with a virtual instrument or an external one. It allows to create MIDI (Musical Instrument Digital Interface) effects. It worked in collaboration with a DA: Digital Audio Workstation.

The goal: Scaler is sold on a specific musical e-shop: Plugin boutique, among others products. It also have an informative website with a community forum. After discussions with the owner, he told me he would like **an support app for the customers who bought the product to be able to login and explain the issue he could face**. Then a technician would be able to exchange with him and find a solution.



Functionalities and Features

All users	Scaler Users	Admin
Log In	Create an account	Dashboard with all the tickets
Log out	Dashboard displaying history, current ticket(s)	Select a ticket and respond to a user
	Create a ticket	Change the status of ticket
	Update a ticket	
	Delete a ticket	

Target audience:

It is a niche product: made for Musicians, music lovers, professional or amateur that already use DAW and create their own music.

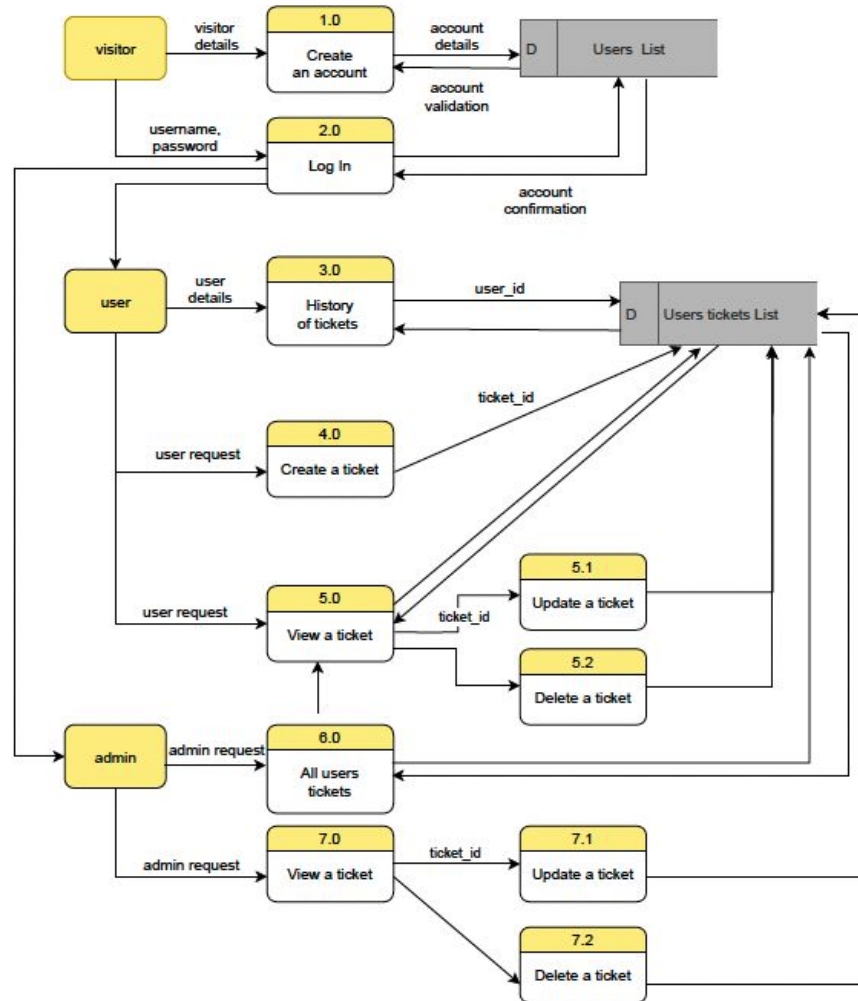
Only the existing users of the plugin could access to this support platform

Tech stack:

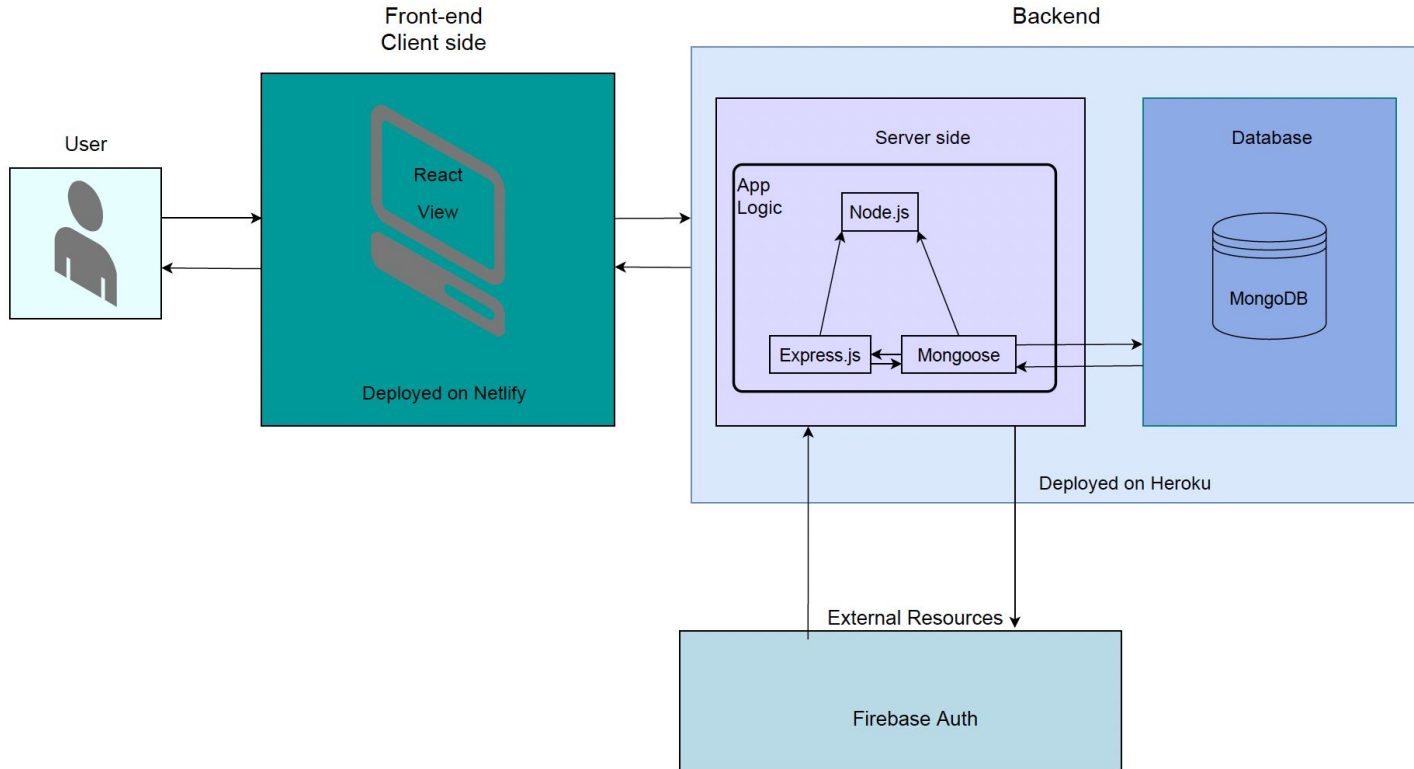
- Planning Management: Trello board
 - Wireframes: Balsamiq
 - Data flow diagram and App Architecture Diagram: Diagram.io
 - Backend: MERN with Express, Mongoose, Nodemon, Firebase for auth
 - Frontend: React, with React-dom, Axios, JWT
 - Testing: Manual testing, Jest, Cypress ?
 - Deployment: Heroku to deploy MERN and Netlify to deploy React
- 

Data flow diagram

Gane and Sarson
methodologie



Application Architecture Diagram



User: The app that the user interacts with

React view: It is a JavaScript library, specialised in building User Interface. It will render the single page support app

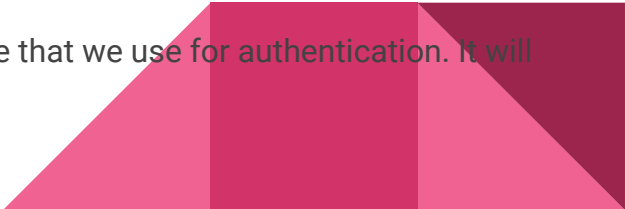
Node.js: It is a JavaScript runtime platform, allowing to create command line tools and server side scripts locally. It manages requests asynchronously.

Express.js: It creates a server locally in the development and testing phases. It is in charge of the HTTP requests and responses with CRUD operations routes

Mongoose: It determines the schema and model for the data

MongoDB: It is the document oriented database using JSON. It will be represented by collections of data. It interacts with Mongoose

Firebase: It is an external infrastructure, owns by Google. It is a realtime database that we use for authentication. It will works with the server side.



Personas

- Michael is a piano learner, 16yo, curious, interested in creating his own music, passionate about learning new skills, he would like to learn more about the features of the plug-in, So he can progress faster
- Charlie is a DJ amateur, a 30yo web designer, she has always love and listen a lot of music, she has been practicing on a DAW for around 5 years, she is looking now for a more specific tool to continue her journey into music
- Jane is a music teacher, she wants to be able to explain to her students the benefits, key features of using this product, So the more she knows, the better she could respond to her students
- John is a 45 yo music producer, he wants to ask questions about particular options So he can maximise his knowledge and utilization and he wants to resolve an issue asap so his work is not much impacted
- Zoe, 50yo, has been recently recruited by Scaler company. She is the admin of the app, she is very organised and likes to keep things simple



Users Stories

As a customer:


- I want to log in to the support platform, SO I can see my personal dashboard / history
 - I want to be able to create a new support ticket SO I can get an answer to her questions from an agent
- The tickets will have different categories: general info, feature info, problem, blocker
- I want to be able to update a ticket, SO I can refine my request to the agent
 - I want to be able to resolve a ticket (change the status) SO I can indicate to the agent that my problem is solved



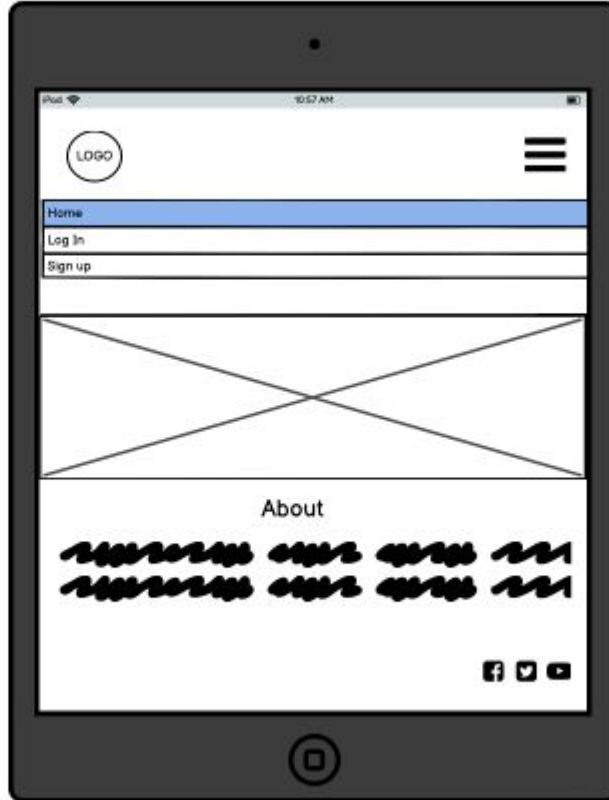
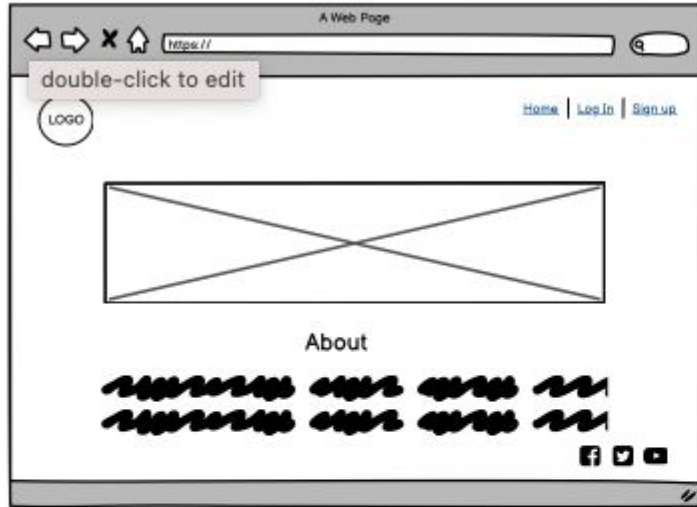
As an admin:

- I want to have an overview of all the tickets SO I can manage my load of work
- I want to be able to sort the tickets by date SO I can prioritize them
- I want to be able to interact with the customers SO I can help them
- I want to be able to resolve the ticket SO I can focus on others tasks

Future improvements:

- Admin has hired different agents to split the workload, she wants to assign the tickets to specific agents SO the customers can get an appropriate answer in the best time limit
 - Admin wants to remind customers with an automatic message if they didn't reply to an agent answer
- 

Wireframes - Home page



Sign Up view

A Web Page

https://

LOGO

home | Log In | Sign out

Sign Up

Username:

Email:

Product number:

Password:

Password confirmation:

Button

f t y

Pod 10:33 PM

LOGO

Home
Log In
Sign up

Sign Up

Username:

Email:

Product number:

Password:

Password confirmation:

Button

f t y

09:52 AM

LOGO

Home
Log In
Sign up

Sign Up

Username:

Email:

Product number:

Password:

Password confirmation:

Button

f t y

Log In view

A Web Page

https://

LOGO

[Home](#) | [Log In](#) | [Sign up](#)

Log In

Email

Password

☐ Remember me

[Sign Up](#)

[Forgot my password](#)

[f](#) [t](#) [v](#)

10:30 PM

LOGO

[Home](#)

[Log In](#)

[Sign up](#)

Log In

Email

Password

☐ Remember me

[Sign Up](#)

[Forgot my password](#)

[f](#) [t](#) [v](#)

09:52 AM

LOGO

[Home](#)

[Log In](#)

[Sign up](#)

Log In

Email

Password

☐ Remember me

[Sign Up](#)

[Forgot my password](#)

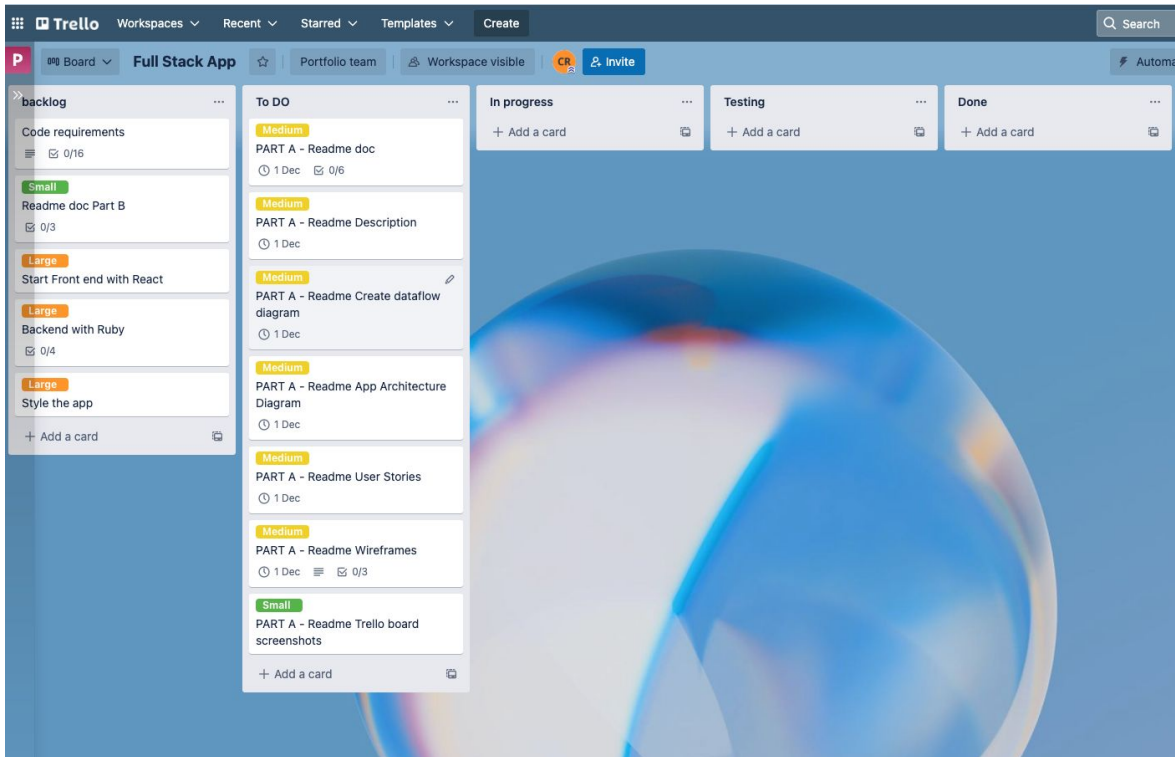
[f](#) [t](#) [v](#)

User Dashboard view



Planning methodology: Trello boards

<https://trello.com/b/bCOFFh9/full-stack-app>



P

Board

Full Stack App

Portfolio team

Workspace visible

CR

Invite

AutomationFilterShow menu

backlog

Code requirements

0/16

Small

Readme doc Part B

0/3

Large

Start Front end with React

Large

Backend with Ruby

0/4

Large

Style the app

+ Add a card

To DO

Medium

PART A - Readme Wireframes

1 Dec0/3

Small

PART A - Readme Trello board screenshots

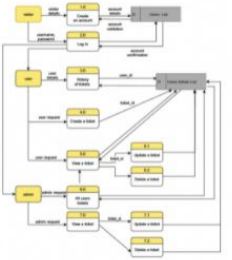
+ Add a card

In progress

Medium

PART A - Readme doc

1 Dec1/6



Medium

PART A - Readme Create dataflow diagram

1 Dec1

Medium

PART A - Readme App Architecture Diagram

1 Dec2

+ Add a card

Testing

+ Add a card

Done

Medium

PART A - Readme User Stories

1 Dec

Medium

PART A - Readme Description

1 Dec

+ Add a card



Trello

Workspaces

Recent

Starred

Templates

Create

Q Search

🔔

👤

P

Board

Full Stack App

Portfolio team

Workspace visible

CR

Invite

Automation

Filter

Show menu

backlog

Code requirements

0/16

Small

Readme doc Part B

0/3

Large

Start Front end with React

Large

Backend with Ruby

0/4

Large

Style the app

Add a card

To DO

Add a card

In progress

Medium

PART A - Readme doc

1 Dec 4/6

Medium

PART A - Readme Wireframes

1 Dec 0/3

Small

PART A - Readme Trello board screenshots

Add a card

Testing

Add a card

Done

Medium

PART A - Readme User Stories

1 Dec

Medium

PART A - Readme Description

1 Dec

Medium

PART A - Readme Create dataflow diagram

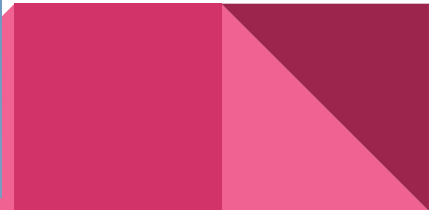
1 Dec 1

Medium

PART A - Readme App Architecture Diagram

1 Dec 2

Add a card



The background is a solid pink color. In the top right corner, there is a decorative pattern of overlapping geometric shapes, including triangles and squares, in various shades of pink and magenta.

Thank you