

FUNCTIONAL DESIGN

PROJECT CLIENT ON BOARD

Date: 2022/11/24

Group: Brave Alligators

Authors:

Adrian Krantz – 524202

Darius Bejan - 516471

Georgs Jakubovskis - 517100

Hai Ha Pham - 493611

Tiffany Deng – 452428

Sadra Samadzadeh – 519407

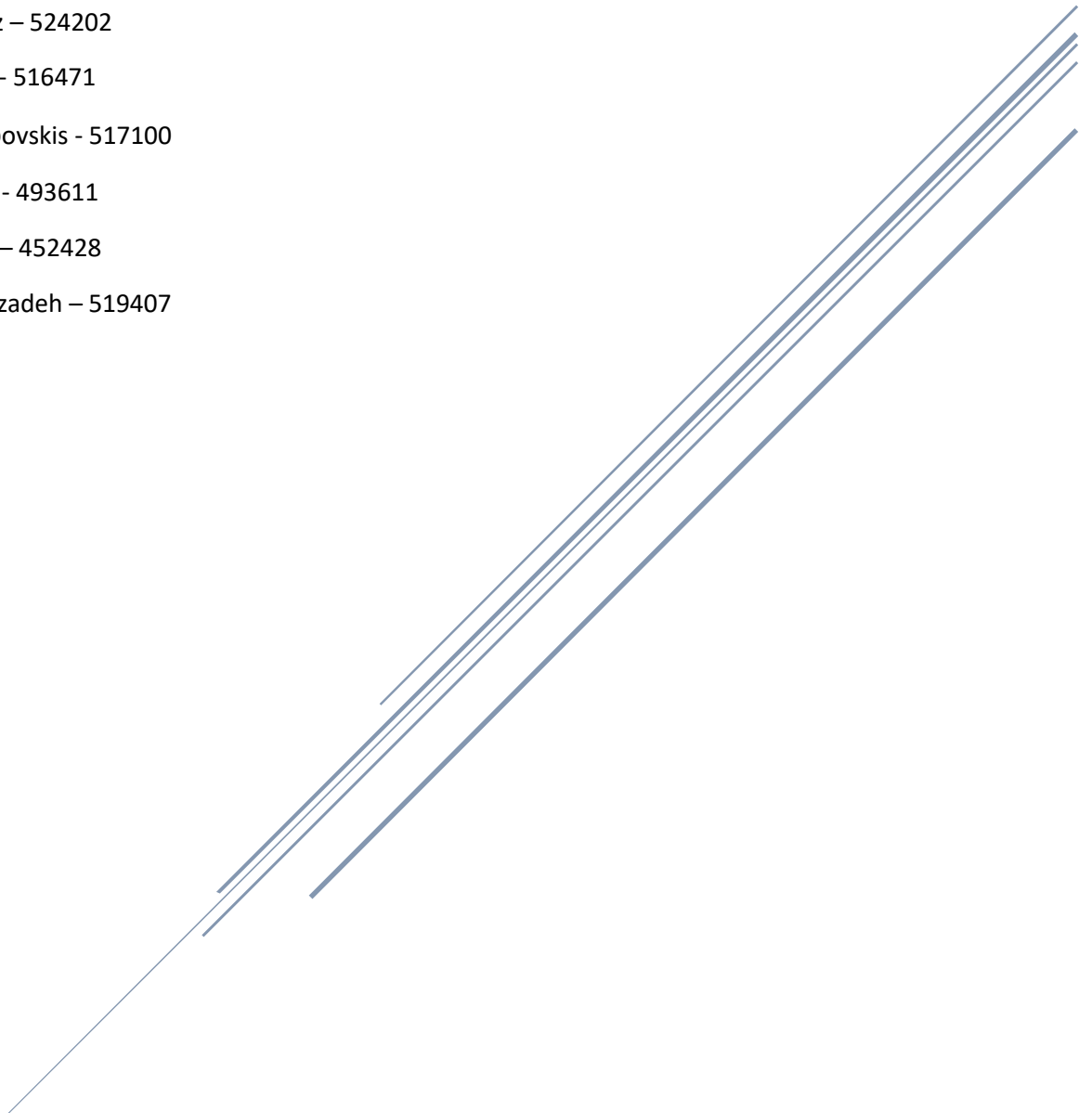


Table of contents

Introduction	2
Requirements.....	3
Functional.....	3
Non-functional	3
User stories (user requirements)	4
Wireframes	5
Login page	5
Register page.....	6
Customer CRUD page	6
Company CRUD page	7
Ticket CRUD page	8
Converter CRUD page.....	8
Log page	9
Customer dashboard page	9
.....	9
Navigation schema.....	10

Introduction

As part of the course of “Client on Board” the group of authors were faced with a challenge of working in a 6 person team on a bigger project for a legitimate client “Profit Flow”. Profit Flow is an IT company offering smart solutions for specifically the solar panel market. The client has requested for the team of authors to design and engineer an application that is capable of performing an important function of monitoring the components of the solar panels called “converters”.

A converter is a device that regulates the electricity from a solar panel into a current that is allowed and accepted by home devices. With a faulty converter the solar panels cannot function properly, and if a fault occurs it is very difficult and time consuming to check if the converter functions properly or not.

The application will be used by 3 main user classes those being:

- Customer
- Company Admin
- Global Admin

To solve this problem the team used knowledge previously learned in past learning experiences such as REST API for fetching data from the web, and other frameworks used in web application development, as well as new concepts learned during the research phase.

The preferred development language was requested by the client to preferably to be “Typescript”. The team had no previous experience of working with typescript, and during the project research was also done to make sure the team members were competent in the usage of TypeScript. To make sure the project gets updated in a structured matter the team had agreed upon git etiquette which is a set of rules to be followed when making changes in ones local repository, this in return provided the team with the ability to make changes to the project without having such issues as merge conflicts.

This document involves a thorough analysis of the problem and it’s context and a validated solution.

Requirements

Functional

ID	Requirement	MoSCoW
F1	A list of all customer's converters must be displayed	M
F2	Each converter shows the following details: <ul style="list-style-type: none">○ Status○ Device ID○ Description○ Expected throughput○ Company	M
F3	All client-side input must be validated before sending it to the server	M
F4	The server performs server-side validation	M
F5	A CRON job is used to get status updates from converters	M
F6	CRUD functionality for customers	
F7	CRUD functionality for converters	
F8	CRUD functionality for converters on customers	
F9	Retrieval and display of converter event history	
F10	Creation of tickets and notifying customers and companies of failure events	
F11	Notifying customers and companies if expected throughput is not met	
F12	API communication with converter supplier	
F13	Role-based access with customer, company administrator, and global administrator roles.	

Non-functional

ID	Requirement	MoSCoW
NF1	Frontend is built using Svelte	M
NF2	Backend is built using Node.js and Express	M
NF3	All response bodies must return valid JSON	M
NF4	Frontend and backend provide descriptive error messages	M
NF5	The user is automatically logged in after registration	M
NF6	All endpoints have good and bad weather tests	M
NF7	The API returns JSON along with an appropriate HTTP status code	M
NF8	The REST API must use the following REST principles: <ul style="list-style-type: none">○ URL paths represent resources not actions	M

	<ul style="list-style-type: none"> ○ HTTP requests are used as intended. GET requests never update a resource. ○ Query parameters are used for querying and should limit results. 	
NF9	The database uses PostgreSQL	M
NF10	At least a single API connection is to be made to one of the supplied brands for converters	M
NF11	Back-end development using NodeJS/Express and TypeScript, with potential use of serverless functions	M
NF12	The API has clear and concise documentation	M

User stories (user requirements)

ID	Requirement	MoSCoW
US1	As a company administrator, I want to be able to see the history of events in an installed converter, so I can keep track of a converter's data	M
US2	As a customer, I want to be notified if my converter detects a failure so that I know if my solar panels are working	M
US3	As a customer, I want to be able to see the history of events in an installed converter, so I can keep track of converter's data	M
US4	As a customer, I want to be able to register with an email and a password so that I can use the application	M
US5	As company administrator, I want to be able to create, remove, update, and delete the converters of a customer, so I can maintain the website	M
US6	As a global administrator, I want to be able to create, remove, update, and delete the companies, so I can maintain the website	M
US7	As a customer, I want to login to the application so that I can keep track of my converters	M
US9	As customer, I want to be able to see the availability and status of the converters, so I can be sure the converters work properly	M
US11	As a global administrator, I want to be able to add, remove and modify company administrators so that I can maintain the application	M
US12	As a global administrator, I want to login to the application so that I can maintain the application	M
US13	As a company administrator, I want to login to the application so that I can keep track of my clients	M
US14	As a global administrator, I want to be able to do what company administrator and the customer can do, so I can maintain the system	M
US16	As a company administrator, I want the application to be able to communicate with different supplier's converters so that our customers can track their converters	M

US17	As a company administrator, I want to see the expected throughout put per converter, so I can make sure that the expectations are met	M
US18	As a customer, I want to see the expected throughout put per converter, so I can make sure that the expectations are met	M
US19	As company administrator, I want to be able to create, remove, update, and delete the tickets of a converter, so I can maintain the website	M
US20	As a company administrator, I want to be able to add, remove and modify customers so that I can maintain all my customers on the application	M


Wireframes

[Login page](#)



The wireframe shows a login page with a light gray background. At the top, the text "Log in" is centered. Below it are two input fields: "Email" and "Password". Under the "Password" field is a "Log in" button. At the bottom, there is a "Forgot?" link.

Register page

A registration form titled "Register" is centered on a light gray background. The form consists of four text input fields stacked vertically, each with a label inside: "Username", "Email", "Password", and "Confirm password". Below these fields is a "Register" button. The entire form is enclosed in a light gray rounded rectangle.

Register

Username

Email

Password

Confirm password







Register

Customer CRUD page

Ticket CRUD page

Desktop - 1

Tickets







ID	Name	Email	Role	Actions
				
				
				
				
				
				

← 1 2 3 4 5 →

Converter CRUD page

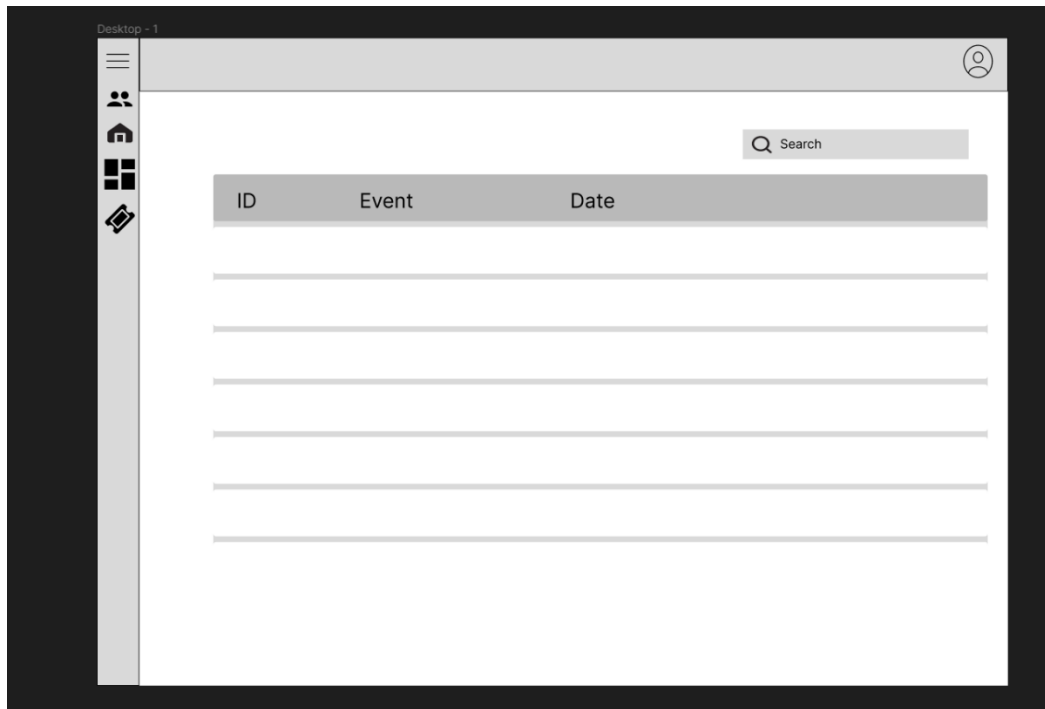
Desktop - 1

Converters

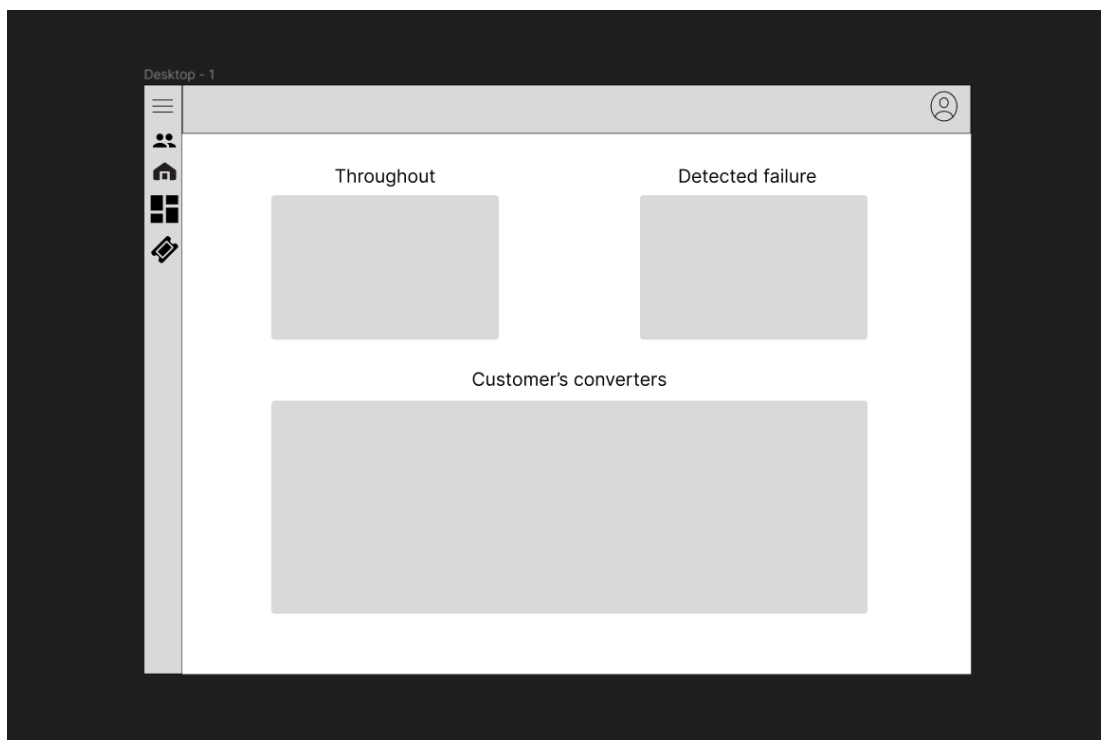
ID	Name	Email	Role	Actions
				
				
				
				
				
				

← 1 2 3 4 5 →

Log page



Customer dashboard page



Navigation schema

