Technical 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

SAXION UNIVERSITY OF APPLIED SCIENCES

Table of contents

[Introduction 2](#_Toc121095999)

[General Overview and approach 2](#_Toc121096000)

[Design Consideration 3](#_Toc121096001)

[System architecture 4](#_Toc121096002)

[Database Design 4](#_Toc121096003)

# Introduction

As part of “Client on board” class the team of authors were tasked to undertake a project of creating an application requested by a client. The client “ProfitFlow” had requested for the authors to build a Web application that would simulate the kind of environment the client currently was using. This application had to be capable of retrieving the status of converters, as well as generating a ticket if the status did display an error. To solve this task the authors had to integrate API’s of converter companies. Each company had their own respective API that authors had to research and integrate to retrieve the converter status’s. This document describes how the authors planned to implement the proposed solution from the Functional Design. These decisions are argued and supported by the authors’ reasoning and the clients preferences.

# General Overview and approach

To achieve the given task, the teamuthors had to work in a developer team, meaning the tasks needed to be defined and distributed amongst the team members. To achieve the needed result the team of authors used such work frameworks as SCRUM to provide a organized way of working on and solving problems.

The group also utilized GitLab as a shared work environment where group members can add and edit code on their local machine and eventually merge the code with the main code base of the group.

Project development was split in 4 sprints including Sprint 0, where every sprint is dedicated to a development period of the application.

Each sprint has its own set of backlog items, excluding Sprint 0, since during that period most of the initial setup for the start of the project is done.

# Design Consideration

The client had given the team of authors a set of preferences describes in the case document. Client had stated the it would be appreciated if the end product that the teams delivers would consist of frontend and backend that are both written in TypeScript. TypeScript is a superset of JavaScript, meaning both languages are similar. Since authors had previous experience in web development with Java Script it would not require time to research Type Script as it uses most of the same concepts as Java Script.

A PostgreSQL database was to be created to store data retrieved. To retrieve the data gotten from the converters and users a database is needed to store the data in a well-structured manner. The team of authors had decided to host the database on a web server via Microsoft Azure for ease of access and taking account that the alternative being the virtualization application “Docker” is highly disliked by the development team.

A CRON job was also requested to be a part of the application. CRON is a job scheduler on Unix-like operating systems. Users who set up and maintain software environments use cron to schedule jobs, also known as CRON jobs, to run periodically at fixed times, dates, or intervals.

One of the clients’ requirements was for the author’s to use a CRON job. The authors chose to use a CRON job for retrieving the statuses of the converters, since CRON can be scheduled to execute scripts. The CRON job can be used to fetch the converter’s statuses daily or hourly, depending on the clients preferences.

# System architecture

# Diagram Description automatically generatedDatabase Design

The above figure displays the authors’ current proposed database design for the application.