Háskólinn í Reykjavík

Hugbúnaðarfræði

T-303-HUGB

Sprint 1
Project Overview Report

Students:

Emil Örn Kristjánsson

Jóhann Ingi Skúlason

Jón Skeggi Helgason

Ríkharður Friðgeirsson

Steinar Snær Guðjónsson

Valdís Bæringsdóttir

Teacher:

Grischa Liebel

Date:

01.09.2020

Table of contents

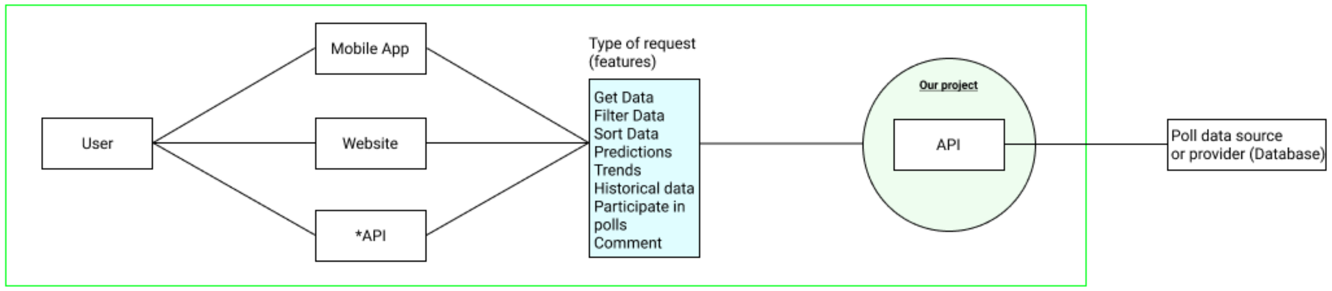
1 Introduction

The aim of the project is to create a backend that supports the management and use of polling data for elections. The backend should be general enough so it can be used for all sorts of frontends for all sorts of polling data. This project will use an agile process, similar to SCRUM, where the project is divided up to five sprints. Each sprint is 2-weeks long and this report is part of the first Sprint. The first sprint is a planning sprint to perceive the system. This is the first sketch of the overview of the system, including possible users, features and scenarios.

2 The System

2.1 A Domain Model

Before designing a backend it is very important to look at the entire system. As shown in the flow chart here below the system is larger then just the backend or the API shown in green in figure 1.



Mynd 1: A domain model for the system.

The goal of the project is to implement a backend/API that handles processing of data requests from users and sends them back its findings in the most optimal form for a frontend to display.

2.2 Users

To understand what the system needs it is important to know what the user needs. There is a big potential user base for a platform that hosts polling information. The users will be anyone who has an interest in elections, such as voters, candidates, campaign managers, journalists, politicians and political enthusiasts to name a few. These users might approach the data from different interfaces. Most users would likely access the data from a website or a mobile app. *The last interface is noted with a star sign because some users, such as data analysts or journalists, might want to request the raw data without a frontends implementation to be able to process the data in their own way. These users might use an API platform such as Postman for these kinds of requests.

2.3 Features

There are a lot of different kinds of data these user might want to see. Therefore, there are multiple possible types of requests the user should be able to make. Firstly this system should give users a platform for viewing election and polling information. They should be able to see polling data, participate in polls, view trends and interact with other users. All the different features are shown in the "Types of requests (features)" box in figure 1. For a detailed list of what the users should be able to do see the list of user stories here.

3 Scenarios

The flow of entire system put together works as following; a user needs certain information and uses a frontend, a website or an API platform (e.g. Postman), to requests that certain data from the backend/API. The API processes and returns the requested data, which is delivered to the user through the frontend they are using. See figure 1 again.

Different kinds of the system flow are called scenarios and the most important ones are listed here below.

Scenario 1:

INITIAL ASSUMPTION: A voter visits a site that is using our API to view the current political status of his/her country
NORMAL: The voter visits the site and sees the information gathered from the most recent polls from his/her country displayed in the way that the website chooses to implement it. The voter uses the site's widget controls to see how the polls were in the past.
WHAT CAN GO WRONG: The data on the website is wrong because the API is feeding it outdated or wrong information.
OTHER ACTIVITIES: The voter decides to view polling data from a specific region in his/her country.
SYSTEM STATE ON COMPLETION: The voter has viewed the current polling data from the database and was able to see his/her country's current political status.

Scenario 2:

INITIAL ASSUMPTION: A campaign manager visits a site using our API and views the difference in poll data between candidates.
NORMAL: The campaign manager looks up the section on the website where it displays the difference between the candidates. The website sends a request to our API to collect both the total votes each candidate has gotten and the percentage of total votes each candidate has. The API sends the data back to the website and it displays it to the campaign manager.
WHAT CAN GO WRONG: The API could be missing data or the data it has is outdated or wrong. The API wont receive the request from the website. The API fails to send the data back to the website.
OTHER ACTIVITIES: The campaign manager might download the poll data as json from the website or use an external tool.
SYSTEM STATE ON COMPLETION: The campaign manager is able to see the difference in poll data between candidates and see who is ahead of an election and by how much.

Scenario 3:

INITIAL ASSUMPTION: A journalist/analyst using a website or external tool to request latest polling data and current trends on the current presidential elections.
NORMAL: Journalists/analysts using a website request the latest results from polls and receive them as a bar graph generated by the webserver with data from the API.
WHAT CAN GO WRONG: The data on is wrong because the API is feeding outdated or wrong information.
OTHER ACTIVITIES: The journalist/analyst could download the data as json from the website or use an external tool.
SYSTEM STATE ON COMPLETION: Journalists/analysts were able to receive the relevant data.

Scenario 4:

INITIAL ASSUMPTION: A user is logged in to the website and comments on the latest policy of a candidate.
NORMAL: The user views a candidate and sees what his/her policies are, listed by latest first. The user views a policy and comments on it.
WHAT CAN GO WRONG: The user can be banned from commenting or commenting be disabled by site admin.
OTHER ACTIVITIES: The user deletes a comment.
SYSTEM STATE ON COMPLETION: A user is logged in. Comment is added to the database along with username and time.