MARC 1 REVISED PROJECT PROPOSAL

PART 1:

Team Name: MARC 1

Team Members:

Member Name	**UNI**
Martin Ristovski	mr3986
Angel Garcia	ag4219
Rishav Kumar	rk3142
Cesare De Michelis	cd3195

Language: Python

Platform: Linux

Github Repository: https://github.com/martinristovski/marc1

We met with our mentor IA, Qingyang Wu, on October 19, 2021.

PART 2:

What will your service do? What kind of functionality or features will it provide?

Idea: Hosted Form Service

Our Form Hosting Service will allow developers to save the form data of their webpage in a secure environment without having to worry about hiring a back-end engineering team. Just use our API to save your data and then go through the submission.

Having taken into account the feedback given by the IAs we have decided to simplify the offering of our API. We will implement the forms hosting service with a standard implementation of a form, with a question / answer structure. The API will offer functionality for creation of forms and, once a form is created, forms can be filled out by the user and data submitted into our database.

Features for Developers:

1. Generate an API key which shall be used by developers in further requests
2. Register form by providing a template and application endpoint.

3. Update the fields and endpoints of the application on which the form is to be used.
4. Post their form responses to our service.
5. Can view the response received on their forms.

Who or what will be its users? What might they use the functionality for?

Our two groups of users will be developers and site visitors. Developers will interact with our service to host forms that can then be used on their websites. Site visitors will fill the data on developers application and then developers will call our service to save this information.

What kind of data will your service create or accumulate? What will the data be used for?

Our service will accept data the developers receive on their form in json format. This data can then be viewed by the developer for collating responses they received on their site.

You can view the development documentation at: Form-Service-API

PART 3:

How will you test that your service does what it is supposed to do and provides the intended functionality?

Our test suite will be divided into a three step process. The first suite will regenerate a unique API key for the developer. This will allow the developer to generate a form template for their application by providing a URL of the application and the template of their form. This will generate a form_id for the developer for using this form on their application.

The second suite validates that the request received for saving the form data received from developers website and will be stored in our database.

This saved form will then be used by the third step where these values are available for the developer.

How will you check that your service does not behave badly if its clients use it in unintended ways or provided invalid inputs?

We generate an API key whenever a developer registers on our service. In order to use our other APIs, developers should provide this API key as a mandate. If this API key is missing in any request, the request will be considered as “Unauthorized”.

For the purpose of authentication of requests that save data into our database, we will match the HTTP-Referer Header and form_id with the one registered to our application and the one sent in headers of the request. If they are a match, we check sanitization of the input parameters. Once these checks are validated the data is saved into the database.

How will you test that your service handles its data the way it's supposed to?

We will create a test suite that saves the data of forms into our database and then fetches the data back. We will apply assertions on the value of each field before being saved and the one fetched from the database. The test case qualifies as pass only if all the fields match.

Furthermore, we will make sure to test the idempotence of GET requests from the developer. We will run test suites to validate that the data remains consistent between multiple subsequent requests on the same data.

PART 4:

Frameworks: Flask, PyMySQL
Database: MySQL
Style Checker: Flake8
Static Analysis Bug Finder: Flake8
Test Runner: Unittest
Coverage Tracker: Coverage