**Prototype Requirements**

**P06: Open Source Backend In Rust**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **24100305** | **Abdul Wahab** |
| **24100012** | **Muhammad Saad** |
| **24100085** | **Ahmed Mozammil Iqbal** |
| **24100121** | **Faraz Mansur Ahmad** |
| **24100296** | **Moiz Raza Amir** |

**Table of Contents**

[1.](#_gjdgxs) Introduction 3

[2.](#_30j0zll) Instructions 4

[3.](#_1fob9te) List of Requirements 5

[4.](#_3znysh7) Where to Access the Prototype 6

[5.](#_2et92p0) Review checklist 6

# Introduction

To provide context for our project, Backend as a Service (BaaS) solutions essentially abstract away the complexities of REST API such that the developer only needs to create the frontend and use the ready-made BaaS service’s methods to handle the backend. This increases a developer’s productivity as there is no need to write complex backend code as a result. Many known BaaS services come bundled with several available functionalities such as:

● Built-in REST API CRUD operations

● Out-of-the-box authentication

● File Storage

This makes BaaS solutions attractive for developers. There exist several BaaS services, such as ‘Firebase’ by Google. However, Firebase is closed source and any hosting of the database and other media is done by Google itself which some developers find problematic. Firebase also uses a proprietary data store called “Firestore” which makes data migration a hassle.

As such, there is a growing trend in self-hosting for reasons such as freedom and independence in hosting one’s own services, as well as having the ability to customize applications. Due to the increasing need of customizable services and providing transparency to users, Open Source projects are becoming popular. However, self-hosting open-source BaaS solutions can be tricky as there are several services that need to be configured for them to work securely and efficiently. Most of the existing BaaS solutions provide first-class support for usage as a service. However, they are hosted by the provider, and support for self-hosting in this domain is limited.

Hence we were motivated to create a lightweight backend similar to Firebase that is open source and can be self-hosted. [Pocketbase](https://pocketbase.io/) and [Supabase](https://supabase.com/) are close relatives of the idea, and are the references that will be used throughout the development of our project. The goal is to create a lightweight and fast backend while providing users well made documentation and a clean UI to easily navigate our service.

Unlike Pocketbase which uses Go and Supabase which uses Typescript, we will be writing our backend in Rust: a fast, systems programming language that performs orders of magnitudes better than both Go and Typescript in benchmarks. The potential users will mainly be developers. However when developers deploy our service as a backend for their softwares, System Admins will be able to use our provided User Interface to make any edits.

# Instructions

<

* Select a subset of system requirements and implement them. The end result of the prototype phase must be a working system with the selected set of requirements implemented completely. No mock-up screens will be accepted.
* While you may choose to implement Login/Logout functionality for prototype phase, you must also implement some core/business use cases of the system.
* Select the set of requirements keeping in mind that you have a total of three weeks for prototype development. I would ask you to add more requirements if I think that you can do more in the given duration.
* The prototype must be built using the tools and technologies which you have selected for your system development.
* Follow standard coding practices.
* By the end of the prototype development phase,
  1. You should have learnt development tools and technologies.
  2. You should have a clear idea of detailed technical architecture of your system. After the prototype phase, you will be required to define detailed technical architecture of your system.
* **Prototype Submission**
  1. Properly tested **working prototype** deployed on an online hosting platform.
  2. **Code** with proper comments uploaded in “prototype” folder of your project’s Github repository.
  3. **3-4 minutes video** that explains the functionality of your prototype—to be uploaded in “prototype” folder of your project’s Github repository.

# List of Requirements for Prototype

[Is it okay to develop the prototype without any authentication implementation?]

|  |  |
| --- | --- |
| **Requirements** | |
| **Sr#** | **Requirement** |
| 1 | As a developer, I need a modular and expansive SDK to use the system. |
| 2 | As a developer, I need to be able to create, read, update and delete records in the database. |

The following are the use cases that we will try to implement in the Prototype phase that come under the specified functional requirements above.

|  |  |
| --- | --- |
| **Use Cases For the Prototype Phase** | |
| **Sr#** | **Requirement** |
| 1 | Create a record through the API and UI |
| 2 | Update a record through the API and UI |
| 3 | Delete a record through the API and UI |
| 4 | Read a record through the API and UI |
| 5 | Read a list of records through the API and UI |
| 6 | Create a collection through the API and UI |
| 7 | Delete a collection through the API and UI |
| 8 | Start a Server to serve all the requests. Our server will be written in Rust. |
| 9 | Create index on a collection through the UI |
| 10 | Remove Index on a collection through the UI |

# Essentially our goal in the prototype phase is to implement CRUD functionality in our Rust server and simultaneously build the respective prototype version of our SDK and UI handling the CRUD functionality of the server connected through API requests.

# Where to Access the Prototype

Our project would not have any deployed instances. However, a demo can be hosted if it is required. [It should be hosted, so that you can know the complexities of hosting the actual project later.]

The project would consist of a single executable file contained within a zipped folder. The user is expected to download and deploy the tool to any provider of their choosing. For this purpose, a docker file would be provided that would help the user in deployment to major hosting providers.

Moreover the SDK will be downloadable through the npm registry. Further details on downloading the package will be provided later.

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| 4 | Saad, Wahab, Moiz |
| 3 | Ahmed Mozammil Iqbal, Faraz |
|  |  |
|  |  |