We are planning to build a web application that covers the functionalities of the Garbage management system.

Frontend framework

We need a frontend framework that loads the components of our application dynamically. Since the application requires few libraries, we want the framework to be light. It demands the framework to create highly customized components based on its various states and reduce the dependency among the components for easy maintenance. On the other hand, the framework should facilitate easy learning. The framework should not add the additional overhead of learning a new programming language to concentrate more on the application's business logic. Hence, we will use **React** framework, which satisfies the above needs.

Backend framework

We need a backend technology that helps build a high-performance cross-platform application that has third-party libraries support, easy to scale, simple to learn and adapt, and must be cost-effective. Node.js provides all the necessary features that we require and, to speed up the development, we will use the Express.js framework. We can exchange roles and work on both the frontend and backend of the application. Node.js best suits the architecture of the application as it implements an asynchronous non-blocking event-driven system to handle multiple connections concurrently. We need the framework to scale when the scope of the application grows and Node.js provides that. Since our application has no heavy computational logic involved, **Node.js + Express.js** is the right backend framework.

Database technology

The Database supports persisting the information provided by the user. The two types of Databases are Relational and Non-Relational Databases. The performance of the Databases is measured based on four factors. They are Speed, Scalability, Structure, and Size. In our application, the data are small and related to each other. It is highly structured and organized. The application emphasis the need for Atomicity, Consistency, Isolation and Durability (ACID) properties. The application requires some data derived from joining the data from multiple entities. The application needs faster retrieval of data from the storage technology. Hence, we feel the Relational Database Management System meets the above-stated criteria. Even if our data grows faster in the future, it provides room for us to migrate data to NoSQL databases like MongoDB and keep the metadata still in Relational Database. We plan to utilize the **SQL storage service in the GCP** (Google Cloud Platform) for our web application.

References

[1] "Advantages of various frontend frameworks used in web.pdf," *Google Docs*, 2021. https://drive.google.com/file/d/16kCmnJQfzWSymllWZ_p4a-EMls6ieTsK/view (accessed May 15, 2021).

[2] "Disadvantages of various frontend frameworks used in web.pdf," *Google Docs*, 2021. https://drive.google.com/file/d/1p19y-mp0OLPvVw8AHxwqzH_XRKLkl92T/view (accessed May 15, 2021).

[3] "Advantages of various Backend frameworks used in web.pdf," *Google Docs*, 2021. https://drive.google.com/file/d/1Ldbw5rrp3LMOvlH1VilFS6__tFQff5zn/view (accessed May 15, 2021).

[4] "Disadvantages of various Backend frameworks used in web.pdf," *Google Docs*, 2021. https://drive.google.com/file/d/1lcpAuvl904LiSzCLC1Ew3hLOSzdUk_Fh/view (accessed May 15, 2021).

[5] Advantages and Disadvantages of various database technologies.pdf, "Advantages and Disadvantages of various database technologies.pdf," *Google Docs*, 2021. https://drive.google.com/file/d/100E2ri2VXUnyToSZPp8t8RKgKBIrRVyT/view (accessed May 15, 2021).