Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	15 October 2022
Team ID	PNT2022TMID53057
Project Name	Project – Personal Expense Tracker
Maximum Marks	4 Marks

Personal Expense Tracker

Technical Architecture:

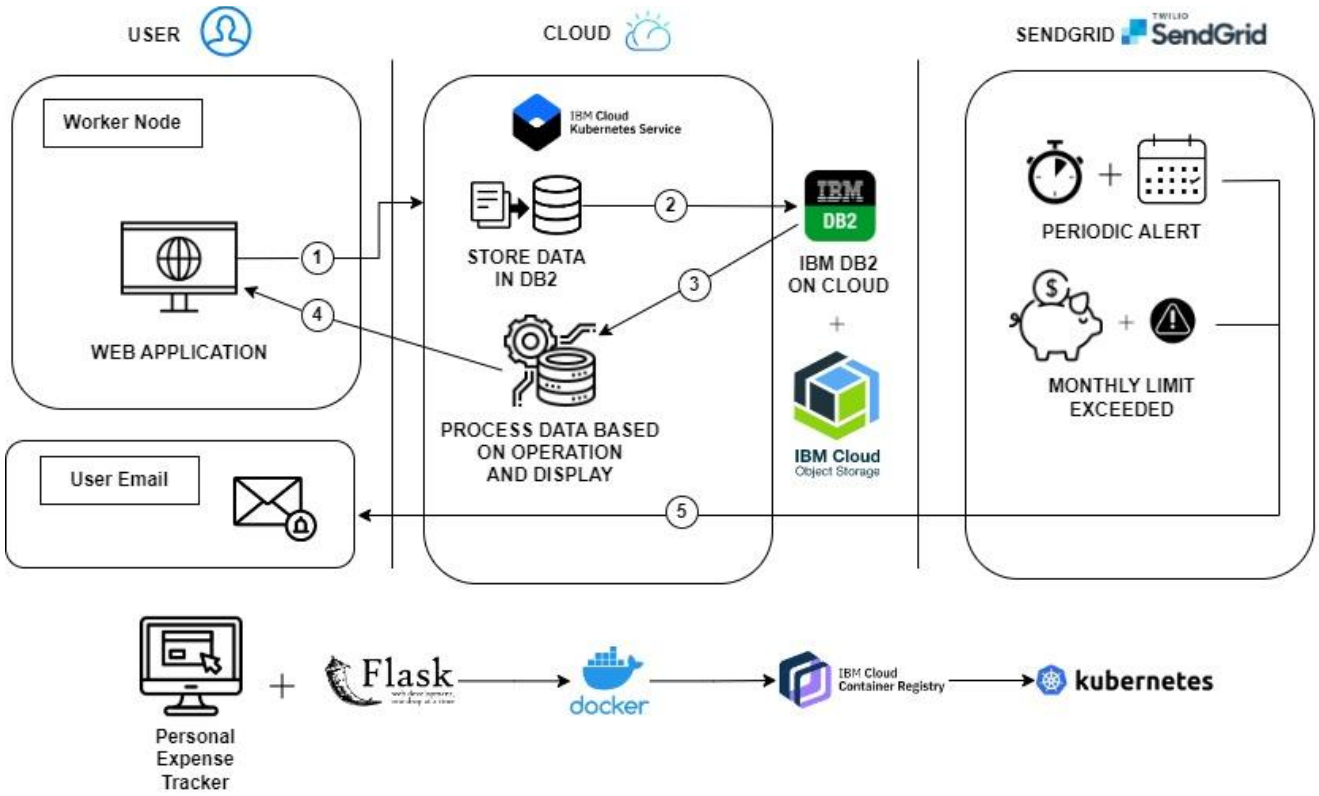

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user users the Web UI to carry out his/her operations.	HTML, CSS, Javascript, Flask, Python
2.	Application Logic – Creating an account	The user interacts with the Web UI to create an account. The account details are stored in a secured manner in IBM DB2.	Flask App running using Kubernetes Cluster, IBM DB2
3.	Application Logic – Logging in	The user interacts with the Web UI to login into the application. The user details are verified by cross-checking with data available in DB2.	Flask App running using Kubernetes Cluster, IBM DB2
4.	Application Logic - Creating an Expense	The user interacts with the Web UI to create an expense. The user chooses a certain category and the enters the amount spent. This is logged in DB2 under the user's record.	Flask App running using Kubernetes Cluster, IBM DB2
5.	Application Logic – User Balance Initialization	The user interacts with the Web UI to initialize the wallet balance which gets updates in DB2.	Flask App running using Kubernetes Cluster, IBM DB2
6.	Application Logic – Modifying an Expense	The user interacts with the Web UI to modify an existing expense. The expenses are displayed to user by fetching it from DB2 and an expense is updated in DB2 as per the user's modifications.	Flask App running using Kubernetes Cluster, IBM DB2
7.	Application Logic – Create a category of expense	The user interacts with the Web UI to add a category of expense which gets registered in DB2. Hence, when the user creates an expense next time, the category list is fetched from DB2, reflecting the addition of the category as performed by the user.	Flask App running using Kubernetes Cluster, IBM DB2
8.	Application Logic – Summary of Expenses	The user interacts with the Web UI to view a summary of expenses in a graphical or textual manner. This data to create the summary is obtained from DB2.	Flask App running using Kubernetes Cluster, IBM DB2
9.	Application Logic – Setting monthly expenditure limit	The user interacts with the Web UI to set the monthly expenditure limit based on which notifications and alerts are created.	Flask App running using Kubernetes Cluster, IBM DB2
10.	Application Logic – Adding a periodic expense	The user interacts with the Web UI to add a periodic expense, i.e. an expense that the user is	Flask App running using Kubernetes Cluster, IBM DB2, SendGrid service

		reminded of using SendGrid as per the user requirements.	
11.	Application Logic – Delayed Expense	The user interacts with the Web UI to define an expense that is to be paid at a later time. This is stored in DB2 and the user is alerted of the expense at the time defined by the user using SendGrid.	Flask App running using Kubernetes Cluster, IBM DB2, SendGrid service
12.	Database	The data types will be user dependent as the application is made to be customizable, i.e., apart from columns such as Username, Email, Password, and Wallet Balance columns such as delayed expense etc will be created as per user requirements.	IBM DB2
13.	Cloud Database	The above-mentioned database and its metadata will be present in IBM DB2 accessible to only the owners/developers of the application.	IBM DB2
14.	External API- SendGrid	The SendGrid service will be used to alert users of various notifications etc as defined by the user.	SendGrid Service
15.	Deployment	Application Deployment on Local System / Cloud Local Server Configuration: The application will run on the local server/client side to allow user to interact with Web UI. Cloud Server Configuration: The application will be hosted on the cloud for the user to user. This is done through containerization of the application using Docker, stored in the container registry, and will be run by Kubernetes.	IBM Cloud Registry, IBM Cloud Object Storage, IBM DB2, Docker, Kubernetes

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask is an open-source framework that is being used to develop the Web UI. Docker is also open	Flask is a micro web framework written in Python. It is classified as a microframework because it does not

S.No	Characteristics	Description	Technology
		source to a certain extent. SendGrid is also open source to a certain extent.	require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Docker offers certain components that are open-source and its main functionality are available for free. The same applies to the SendGrid service.
2.	Security Implementations	Access to the DB2 database system is managed by facilities that reside outside the DB2 database system (authentication), whereas access within the DB2 database system is managed by the database manager (authorization). Kubernetes expects that all API communication in the cluster is encrypted by default with TLS, and the majority of installation methods will allow the necessary certificates to be created and distributed to the cluster components.	IAM controls will be implemented for the access to various cloud applications. Security can be further enhanced using encryption of data in DB2, during the creation of the container image, etc.
3.	Scalable Architecture	3-tier architecture will be employed: Presentation tier – The UI is fixed for the application and thus scalability doesn't really apply to this tier. Application tier – This tier comprises of the Python logic that will be used to provide the main functionality to the application. For instance, the analysis of expenses made by the user to provide the user with the expense summary will be done in this layer. This layer can be scaled higher as the heart of this tier lies in the cloud, i.e. scaled architecture in terms of space, complexity etc, can be incorporated through the use of a new container image that is to be run by Kubernetes.	IBM DB2, IBM Cloud Object Storage, Kubernetes to run new container images incorporating scaled architecture, Docker to create the new container image that incorporate the scaled architecture.

S.No	Characteristics	Description	Technology
		Data tier – This tier contains all the data required. The data is stored majorly in IBM DB2 and IBM Cloud Object storage which have inherent scalability features. Thus, scalability is guaranteed here.	
4.	Availability	On a large scale, as majority of the application depends on the cloud, CDN's can be used and the storage can be distributed, i.e. data of the user would be stored in areas close to the user based on the availability of cloud servers which will enhance performance. Further, the application is heavily reliant on the cloud and thus would require constant connection to the internet, i.e. the cloud service for the user to access his/her data and use the application. If the application is widely used, load balancers can be enforced to ensure the efficient utilisation of storage and compute services.	IBM Cloud Object Storage, Kubernetes, Docker Images, IBM DB2, SendGrid
5.	Performance	The performance of the application would currently be limited as the services employed belong to the trial version and would thus enable only a certain number of users etc, i.e., the performance of the application depends on the storage and compute sources available. Further, a stable internet connection is essential in determining the performance of the web application. The area at which the user's data will also play an integral role in the performance of the application as the closer the data to the user the faster and more seamless use of the application. This performance can be made better using CDN's as they aim to solve the very same issue.	IBM Cloud Object Storage, Kubernetes, Docker Images, IBM DB2, SendGrid