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

| | |
|---|---|
| Date | 03 October 2022 |
| Team ID | PNT2022TMID40167 |
| Project Name | Personal Expense Tracker Application |
| Maximum Marks | 4 Marks |

**Personal Expense Tracker Technical:**

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

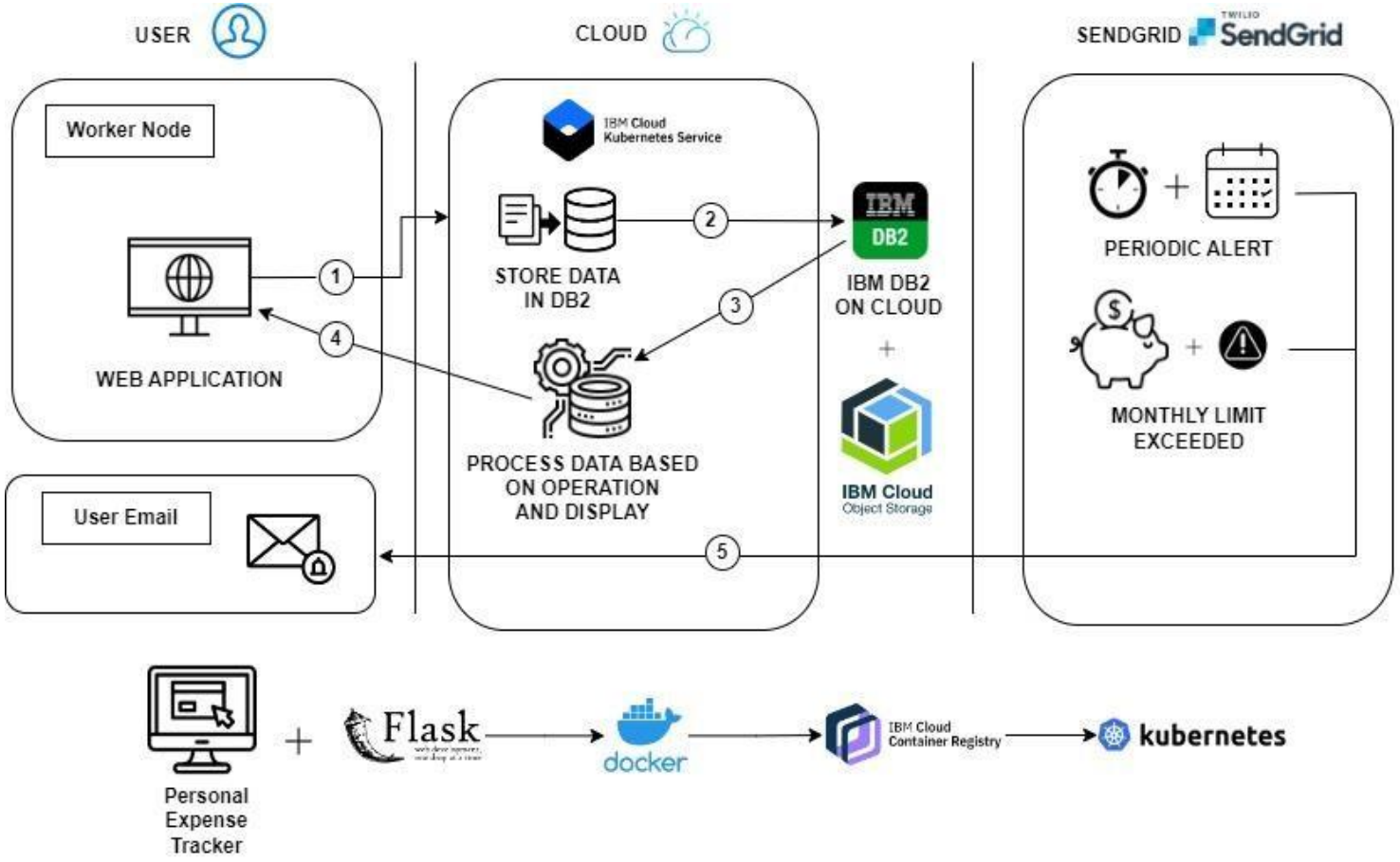

**Table-1 : Components & Technologies:**

| 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 crosschecking 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. | File Storage | IBM block storage used the financed data of the user | IBM Block Storage or Other Storage Service or Local Filesystem |

| | | | |
|---|---|---|---|
| 15. | External API- SendGrid | The SendGrid service will be used to alert users of various notifications etc as defined by the user. | SendGrid Service |
| 16. | 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 |
| 17. | Infrastructure (Server / Cloud) | Application deployment on local system or server | Local, Cloud Foundry, Kubernetes, etc. |

**Table-2: Application Characteristics:**

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1. | Open-Source Frameworks | Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It has become one of the most popular Python web application frameworks. | Python Flask Framework |
| 2. | Security Implementations | This application provides high security to the user financial data. It can be done by using the container registry in IBM cloud. | Container registry, Kubernetes cluster |
| 3. | Scalable Architecture | Containerized application is deployed to rapidly increase scale on demand | Docker |
| 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. | IBM Cloud Object Storage, Kubernetes, Docker Images, IBM DB2, SendGrid |

**References:**


https://c4model.com/

https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/

https://www.ibm.com/cloud/architecture

https://aws.amazon.com/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d