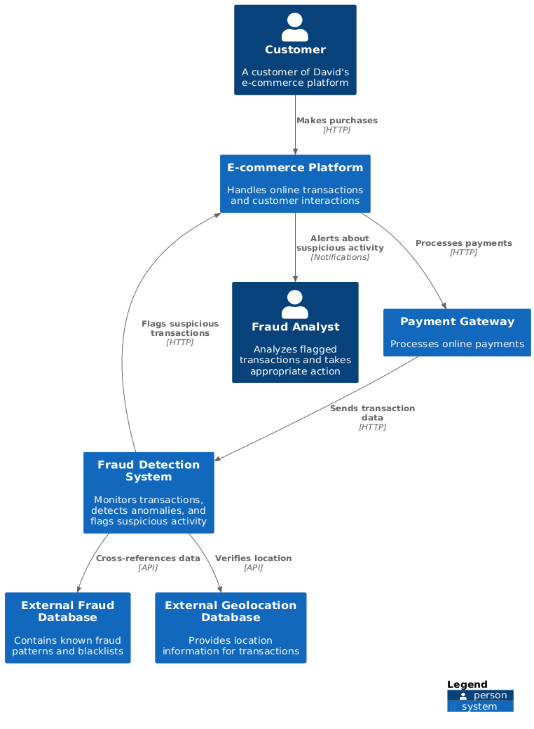
**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**



Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API’s etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1 | User Interface | How user interacts with application e.g. Web UI, Mobile App, Chatbot etc. | HTML, CSS, JavaScript (React/Angular/Vue), responsive design libraries |
| 2 | Transaction Data Processing | Logic for collecting, cleaning, and preparing transaction data for analysis. | Python with libraries like Pandas, NumPy, Scikit-learn |
| 3 | Machine Learning Model | Purpose of Machine Learning Model for anomaly detection and fraud prediction. | Python with libraries like TensorFlow, PyTorch, Scikit-learn, XGBoost, LightGBM |
| 5 | Fraud Alert & Notification System | System for triggering alerts based on model predictions and notifying relevant personnel. | Python with libraries like Gradio,Flask |
| 6 | Database | Data Type, Configurations etc. | PostgreSQL, MySQL, NoSQL databases like MongoDB or Cassandra |
| 7 | Cloud Database | Database Service on Cloud for high availability and scalability. | AWS RDS, Azure SQL Database, Google Cloud SQL, or other relevant cloud database services |
| 8 | File Storage | File storage requirements for transaction data, user profiles, and model artifacts. | AWS S3, Azure Blob Storage, Google Cloud Storage, or other relevant cloud storage services |
| 9 | API Availability | API for integrating our app into any existing services. | Python Libraries like Flask/Gradio/Django |
| 11 | Infrastructure (Server/Cloud) | Application Deployment on Local System/Cloud, Local Server Configuration, Cloud Server Configuration. | Cloud platforms like AWS, Azure, GCP, or container orchestration platforms like Kubernetes |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1. | Open-Source Frameworks | Gradio, Pandas,Numpy, Scikit-learn | Gradio for User Interface and API  Pandas and Numpy for processing the data  Scikit-Learn for Machine Learning |
| 2. | Scalable Architecture | Justify the scalability of architecture (3 – tier, Micro-services) | Cloud-based architecture with scalable infrastructure and services, containerization using Docker and Kubernetes for efficient resource utilization |
| 4. | Availability | Justify the availability of application (e.g. use of load balancers, distributed servers etc.) | Use of load balancers, redundant servers, and auto-scaling mechanisms to ensure high availability |
| 5. | Performance | Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc. | Efficient caching mechanisms, and load balancing to ensure performance and handle high traffic volumes |