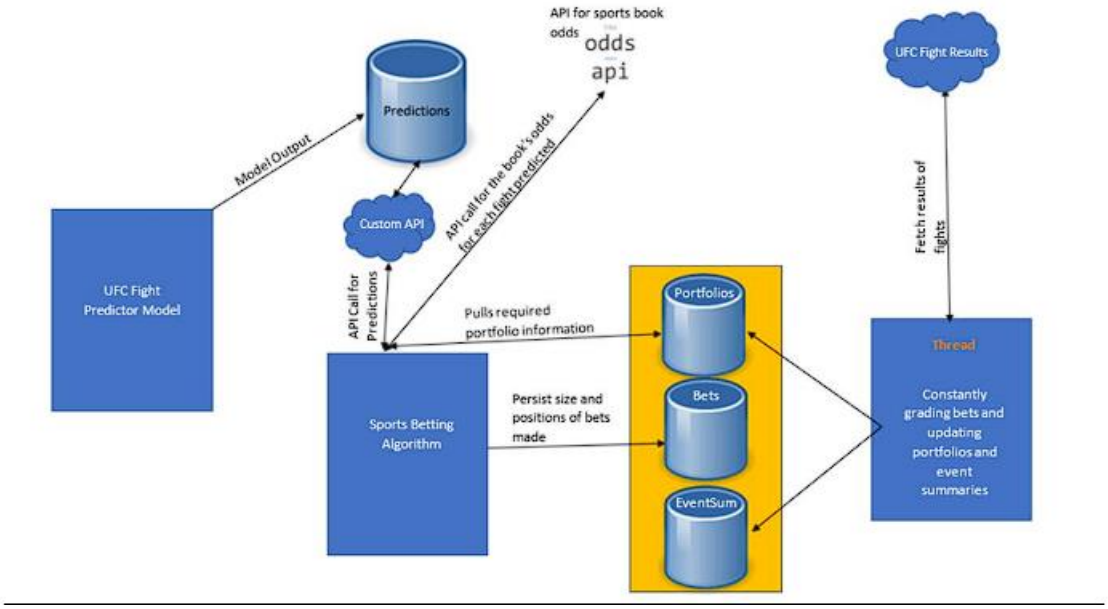
**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

**Technical Architecture:**

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

Certainly! Here is a table that outlines the components and technologies required for the "Octagon Oracle: Machine Learning-Powered UFC Fight Forecast" project:




| S.No | Component Description | Technology |
|---|---|---|
| 1 | User Interface | Web App (HTML, CSS, JavaScript) |
| 2 | Data Collection and Processing | Python (Pandas, Numpy) |
| 3 | Data Storage and Database | MySQL or NoSQL (e.g., MongoDB) |
| 4 | Machine Learning Models | Convolutional Neural Networks (CNN) |
| 5 | Web Application Framework | Django, Flask, or other web frameworks |
| 6 | Front-End Framework for Visualization | React, Angular, or Vue.js |
| 7 | Cloud Services for Scalability | AWS, Azure, or Google Cloud |
| 8 | External APIs | UFC Fight Data API, |
| 9 | File Storage | Local Filesystem |
| 10 | Data Processing and ETL | Python |
| 11 | Model Training and Evaluation | TensorFlow, Keras, or PyTorch |
| 12 | Predictive Analysis and Forecasting | Python (Scikit-Learn, XGBoost) |
| 13 | Version Control | Git and GitHub or GitLab |

Certainly, I can provide a more detailed table with specific items for each characteristic

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1 | **Open-Source Frameworks** | List of open-source frameworks used | TensorFlow, Keras, Scikit-Learn, Django, Flask, React, Apache Spark, Apache NiFi, ELK Stack, Git and GitHub/GitLab |
| 2 | **Security Implementations** | List of security/access controls implemented, use of firewalls, etc. | SHA-256 encryption, IAM Controls, OAuth 2.0, JWT, OpenID Connect, Secure APIs, OWASP best practices |
| 3 | **Scalable Architecture** | Justification for architecture scalability (e.g., 3-tier, Micro-services) | Microservices, Docker, Kubernetes, AWS Lambda, Horizontal Scaling, Auto-scaling |
| 4 | **Availability** | Justification for application availability (e.g., use of load balancers, distributed servers, etc.) | Load balancers, CDN, Distributed Servers, Redundancy, Failover |
| 5 | **Performance** | Design considerations for performance (e.g., number of requests per second, use of cache, use of CDNs, etc.) | Caching mechanisms, Content Delivery Networks (CDN), Load Testing, Performance Optimization, Database Indexing |