***Abstract*—This report describes the development and implementation of an A/B testing platform using the Thompson Sampling algorithm with Bernoulli rewards. The project aims to significantly enhance decision-making processes in digital marketing by optimizing testing strategies and improving user engagement. Our product is flexible and can be adapted for use across various business platforms, providing real-time insights in business settings.**

I. Introduction

A/B testing is a fundamental tool in digital marketing. It can be applied to website optimization, email marketing, app development, and ad campaigns. Traditional A/B testing methods, while effective, are time-consuming and require large sample sizes to achieve statistical significance. This project aims to develop a tool that employs Thompson Sampling—a probabilistic, Bayesian approach for real-time analysis to identify the most effective option in any given scenario. Additionally, the project will enhance this tool into a recommendation engine capable of providing real-time, personalized recommendations. The project includes a microserver architecture(Database, API) which helps deploy it quickly in various environments, providing users with the opportunity to efficiently create and test experiments.

II. Problem Statement

The challenge of making optimal choices in real time is a significant issue in various domains such as e-commerce, content delivery, and interactive platforms. Traditional methods for decision-making often fail to adapt dynamically to new information, which is crucial in environments where user interactions and preferences change rapidly—such as high-throughput online settings and time-sensitive business environments. Traditional testing methods are not only slow but also struggle to provide the flexibility needed for high-frequency testing.

While advanced algorithms are capable of providing immediate feedback and fast adaptation, their practical implementation is often limited to large corporations, due to high resource demands. These challenges create an opportunity for businesses to use a mathematical approach based on the multi-armed bandit problem. This approach helps businesses efficiently choose between available options, explore and exploit these choices, and adapt the solution to various businesses regardless of their sizes and needs.

A. Specific Challenges

- **Dynamic Adaptation:** Develop a system capable of adapting its decision-making process based on continuous user interactions and feedback.
- **Optimization of Exploration vs. Exploitation:** Efficiently balance the exploration of new options and the exploitation of known ones to maximize user satisfaction and business outcomes.
- **Reduction in Testing Duration:** Minimize the time required to reach statistically significant conclusions in A/B testing scenarios.
- **Real-time Penalization:** Implement a recommendation engine that adjusts its suggestions in real time according to user preferences, providing personalized user experiences.

III. Literature Review

A. Docker

Docker is a platform that simplifies the packaging, distribution, installation, and execution of applications by using containers. Containers are isolated environments that provide a lightweight alternative to traditional virtual machines. The core component of Docker is the Docker Engine. It is a client-server-based application with a server-side daemon process. Clients communicate with the server using a REST API. For operational commands, Docker provides a command-line interface (CLI) that interacts with Docker daemons to manage Docker objects such as images, containers, networks, and volumes. Additionally, Docker provides Docker Compose, a tool for defining and running multi-container Docker applications. With Compose, you can use a YAML file to configure your application's services, networks, and volumes, and then create and start all the services from your configuration with a single command. This simplifies the process of managing complex container setups, making it easier to automate and scale applications across different environments. [**Merkel**].

The installation of Docker can be different based on the operating system, but Docker provides straightforward instructions for Windows, macOS, and various Linux distributions.

B. FastAPI

Fast API is a modern, fast web framework for building API's with Python. The framework is designed to be easy to use and makes it simple to create robust APIs. One of the features of Fast-API is its automatic generation of interactive API documentation using Swagger UI and ReDoc. This makes it easier for data scientists to visualize and interact with the API's endpoints [**Triangolo**].

Advanced features of FastAPI such as asynchronous request handling, allows it to handle a larger volume of requests per second than traditional synchronous code. This is particularly useful in environments where real-time data processing is crucial, such as in applications requiring immediate user feedback [**Jones**].

C. Bayesian A/B Testing

A/B testing is a method used in digital marketing to compare two versions of arms to understand which one is better. Users are assigned to the control or testing group randomly and based on user interaction insights are drawn.

By integrating Bayesian statistics into A/B testing frameworks, we can get more dynamic testing by continuously learning from user behaviour

Bayesian Update Formula: Bayesian statistics is an approach that is fundamentally different from traditional statistics. It is about updating our knowledge base using probabilities and evidence. it is named after Thomas Bayes an 18th-century mathematician and Presbyterian minister.

In Bayesian statistics we update our prior believes using update formula. The formula is given by:

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)}$$

where:

- θ: the mean of a Bernoulli random variable.
- X: the data collected.
- $p(\theta|X)$: the posterior distribution.
- $p(X|\theta)$: the likelihood.
- $p(\theta)$: the prior distribution.
- $p(X)$: the evidence, which is often challenging to compute directly.

Conjugate Priors: In Bayesian statistics, the prior distribution indicates what we know about the parameter before considering new data. After observing new data, this prior is updated using Bayes' theorem, resulting in the posterior distribution. Conjugate priors simplify the updating process. A prior is considered conjugate to a likelihood function if the posterior distribution is in the same family of distributions as the prior distribution. This property simplifies the mathematical computations for Posterior Distribution calculations, allowing for analytical solutions instead of numerical methods.

$$p(\theta|X) \sim \text{Beta}(a + \sum x_i, b + N - \sum x_i)$$

where a and b are the parameters of the prior Beta distribution, N is the number of trials, and $\sum x_i$ is the number of successes.

Epsilon-Greedy Algorithm: The Epsilon-Greedy algorithm is a simple method to balance exploration and exploitation:

- With probability ϵ, choose a random option (exploration).
- With probability $1 - \epsilon$, choose the best-known option (exploitation).

Thompson Sampling with Bernoulli Rewards: Thompson Sampling chooses the next option relying on the probability of it being the best. It uses the Bernoulli reward system. One of the advantages Is that Thompson sampling utilizes Beta-Bernoulli conjugacy.

$$\theta|X \sim \text{Beta}(a + \sum x_i, b + N - \sum x_i)$$

Advantages of Thompson Sampling with Bernoulli Rewards:

1) **Conjugate Prior:** As the Thompson sampling uses conjugacy it simplifies computations for posterior calculations.
2) **Real-Time Capability:** The distributions can be updated real-time.
3) **Proven Effectiveness:** Empirically out preforms other strategies by convergence and performance.

IV. Methodology

This section is about methodological approach for implementing real-time decision-making and recommendation engine which uses Containerized approach. The system is designed in a way to ensure scalability and efficiency.

A. Platform Architecture

The project uses Docker,which helps our system to maintain consistency across various development and production environments. It ensure that the application performs identically regardless of the underlying infrastructure.This is largely due to Docker's OS-agnostic nature, allowing it to run on any operating system without modification.

B. Container Configuration

The system architecture consists of three main Docker containers, each serving a distinct role: (Fig. 1) We used volumes so any data created inside a container would not be lost when the container is deleted. Also we utilized ports to allow Docker container to communicate with the outside environment as well as with other containers.

- **API Container:** FastAPI interacts with both the frontend and the database. It supports request validation, ensuring that the data received from users meets the expected format before processing. It also handles automatic API documentation, request validation, and serialization
- **Database Container:** Manages a PostgreSQL database, selected for its robustness and ability to handle large volumes of data with complex transactions. This container stores user data, preferences, and interaction logs.
- **PgAdmin Container:** Provides a web-based PostgreSQL database administration interface, helping with the manageability and monitoring of the database operations.

The interaction within the system unfolds as follows:

1) **User Interaction:** A user interacts with the frontend, which sends requests to the FastAPI application.
2) **Request Processing:** FastAPI receives the request, processes it, and if the request involves data (like fetching, updating, or inserting data), it communicates with the PostgreSQL database to execute the necessary operations.
3) **Database Management:** While the above interactions occur, pgAdmin can be used in parallel to monitor database performance, run maintenance tasks, or execute direct database manipulations as needed.
4) **Response Handling:** Once the necessary data is processed and retrieved from the database, FastAPI packages this data into a response and sends it back.

C. Database

The database schema for this project consists of three primary tables, each designed to manage different types of data and relationships:

Project Table: This table stores information on created projects. Each row includes project's unique

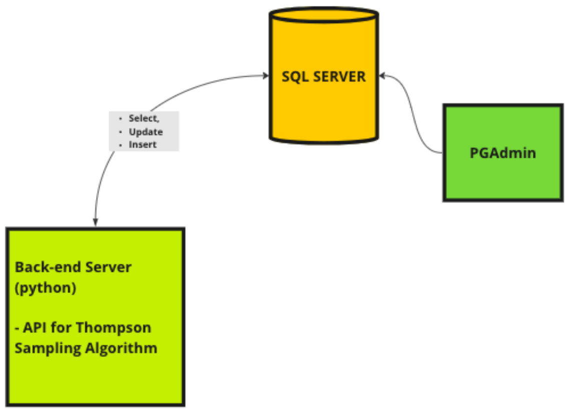

Fig. 1. Conceptual flowchart illustrating the deployment and scalability workflow in our Dockerized environment

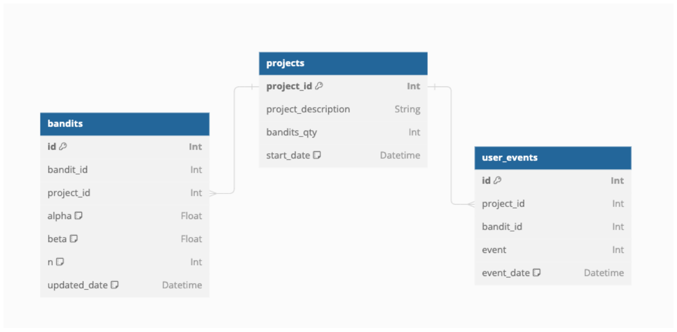

Fig. 2. ERD of the Database

identifier, description, number of bandits involved, and start date. It is main repository for all data.

Bandit Table: This table stores detailed information about each bandit participating in all available projects. It is associated with the project table through a foreign key. It includes a unique identifier for each bandit, the project ID they are linked to, and key statistical parameters: alpha, beta, and n values. This table is important for the analysis.

UserEvent Table: This table tracks the user interaction with the bandits. It stores each event with details such as a unique event identifier, the associated project and bandit IDs, the type of event, and the event's time.

D. Thompson Sampling Algorithm

The system is powered by Thompson Sampling algorithm, which is integrated within the FastAPI framework to manage real-time analysis and decision-making. This method allows the system to adapt to user interactions and preferences by adjusting the likelihood of each recommendation or A/B test option, based on the user feedback. Thompson Sampling is a Bayesian approach that takes a prior distribution over the probability of success for each option (arm). Then it updates this belief based on the received feedback, using the updated distribution to make decisions about which arm to pull next.

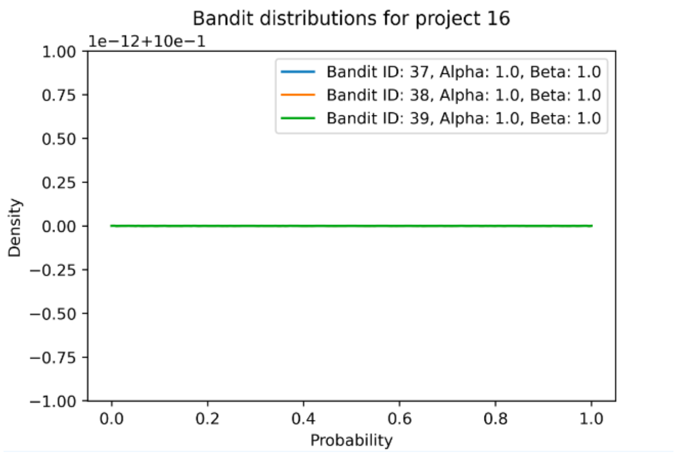

Fig. 3. Initial Beta distributions of bandits at the start of the experiment for project 16

E. Mathematical Formulation

The implementation of Thompson Sampling involves several key steps:

1) **Prior Distribution:** Initially, each arm's probability of success is modeled with a Beta distribution, often starting with parameters $\alpha = 1$ and $\beta = 1$, This using a uniform prior distribution to show that there is initial uncertainty. (Fig.3)

$$\text{Prior: } \theta_i \sim \text{Beta}(\alpha_i, \beta_i)$$

2) **Updating the Distribution:** After each pull of an arm i and observing the result x_i (success or failure), update the parameters of the Beta distribution for that arm:

$$\alpha_i \leftarrow \alpha_i + x_i, \quad \beta_i \leftarrow \beta_i + (1 - x_i)$$

where x_i is 1 for success and 0 for failure.(Fig4)

3) **Sampling and Decision Making:** At each decision point, sample a probability from the updated Beta distribution for each arm:

$$\hat{\theta}_i \sim \text{Beta}(\alpha_i, \beta_i)$$

Choose the arm with the highest sampled probability:

$$i^* = \arg\max_i \hat{\theta}_i$$

We can observe that throughout the course of the experiment, the distributions of the bandits evolve. Initially, the distributions have varied shapes and parameters. however, as the experiment continues, these distributions gradually change and become more distinct, indicating a clearer understanding of each bandit's potential. If we increase number of experiments the distributions will become more clear. (fig5)

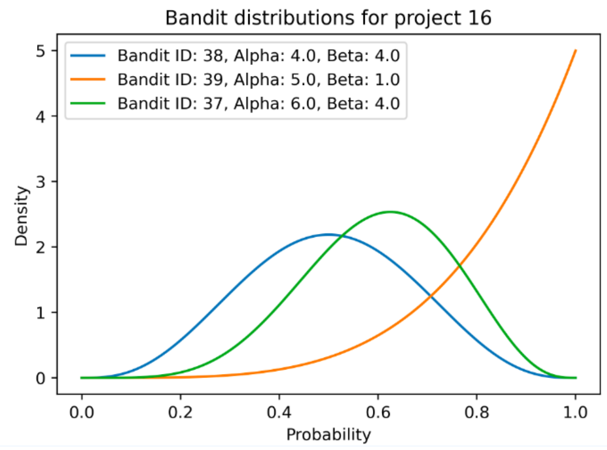

Fig. 4. Mid-experiment Beta distributions for project 16

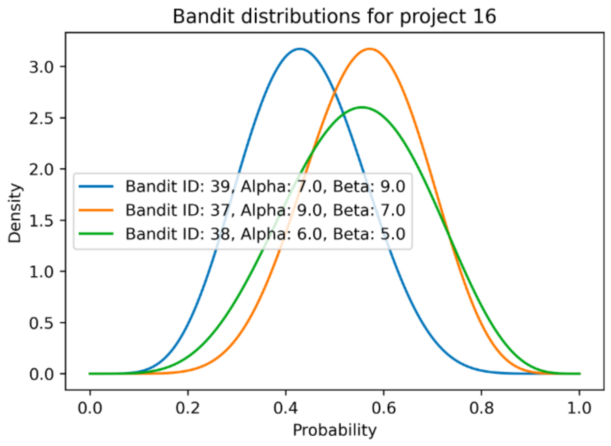

Fig. 5. Final Beta distributions at the conclusion of the experiment for project 16

V. Results

In this section, we present the outcomes of deploying our A/B testing platform, which is powered by the Thompson Sampling algorithm. We will explore the key functionalities of the system and the benefits they offer to users in a practical marketing problems.

A. Project Initialization and Management

The platform supports project management functionalities. Users can initiate new A/B testing projects, specifying essential details such as project descriptions and the number of testing variants. This setup not only makes the project creation process easier but also ensures that each testing scenario is uniquely identifiable and traceable, providing the opportunity of managing multiple projects.

B. Adaptive Bandit Selection

Utilizing the Thompson Sampling algorithm, the system selects the most effective variant or bandit for each project based on ongoing data analysis. This capability allows the platform to adapt to changing user preferences and interaction patterns in real-time, optimizing the selection process continuously and enhancing the overall effectiveness of the A/B testing.

C. Feedback Integration and Algorithm Optimization

A critical feature of our platform is its ability to include user feedback into the algorithm's learning process. This feedback influences the probability models of the variants, allowing the system to refine its predictions and choices based on actual user responses. Such an interactive feedback mechanism not only improves the accuracy of the algorithm but also makes the system more responsive to user needs and preferences.

D. Comprehensive Variant Evaluation

For more detailed analysis, the platform offers an endpoint that selects multiple top-performing variants from a project. This feature is crucial for comparing performances and understanding how different variants perform. It helps make strategic decisions by showing which options work best under certain conditions.

1) Plot Generation for Bandit Distributions: The platform includes an endpoint that generates plots of beta distributions for all bandits associated with a given project. This feature allows users to visually assess the effectiveness of each bandit in real time. By fetching the alpha and beta parameters of each bandit from the database, the system can plot their beta distributions, offering insights into how likely each bandit is to succeed based on past performance.

VI. Conclusion

Overall, the platform provides a sophisticated tool set for optimizing digital marketing strategies through targeted A/B testing. The combination of real-time adaptive testing, detailed feedback integration, and comprehensive data analysis capabilities makes it a valuable resource for organizations aiming to enhance user engagement and increase conversion rates through refined marketing tactics. The system's containerized architecture ensures that it can be adapted to a wide range of applications, from small-scale experiments to enterprise-level deployments.

References

[1] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014. Available: https://linuxjournal.com

[2] S. R. Tiangolo, "FastAPI official documentation," 2020. [Online]. Available: https://fastapi.tiangolo.com/. Accessed on: May 10, 2024.

[3] M. Jones, "Performance of Python Web Frameworks," *Journal of Web Development*, vol. 28, pp. 204-213, 2021.

[4] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A Tutorial on Thompson Sampling," *Foundations and Trends® in Machine Learning*, vol. 11, no. 1, pp. 1-96, 2018.

[5] W. R. Thompson, "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples," *Biometrika*, vol. 25, no. 3/4, pp. 285-294, 1933.

[6] O. Chapelle and L. Li, "An Empirical Evaluation of Thompson Sampling," in *Advances in Neural Information Processing Systems*, vol. 24, 2011.