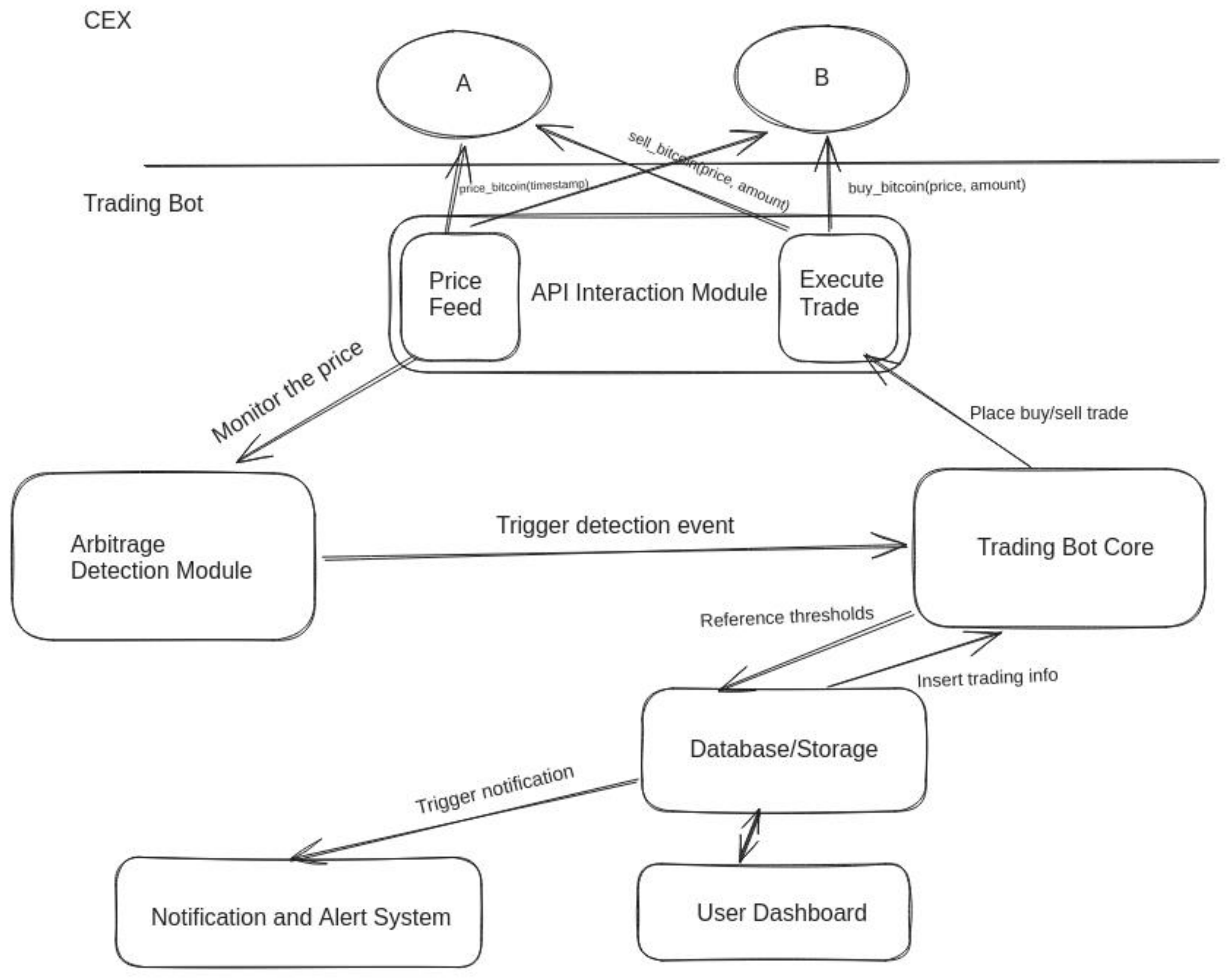
Part1: System Design

Here is a high-level overview of the system architecture, along with a diagram illustrating the components and their interactions.



Application Architecture Components:

1. **API Interaction Module**
   Function:
   - These are modules that interact with the Centralized exchange APIs.
   - Retrieves real-time Bitcoin prices from ExchangeA and ExchangeB using the `price_bitcoin(timestamp)` API.

- Executes buy and sell orders using the `buy_bitcoin(price, amount)` and `sell_bitcoin(price, amount)` APIs.

Considerations:

- Needs to be efficient and fast to capture real-time price differences and consider about latency to fetch price from each exchange.
- Must be highly reliable and secure. Needs to handle API rate limits and potential downtimes.

2. **Arbitrage Detection Module**
   Function:
   - This component constantly monitors the price of Bitcoin across A and B using **API Interaction Module**.
   - It detects arbitrage opportunities (e.g. significant price differences between A and B 10,000USD per BTC in A,   9,000USD per BTC in B)

   Considerations:
   - Processes the incoming price data to identify price discrepancies.
   - Algorithms that analyze processed data to pinpoint potential arbitrage opportunities.

3. **Trading Bot Core**
   Function:
   - It is the decision-making engine.
   - It receives signals from the Arbitrage Detection Module and decides whether to execute trades.
   - It uses **API Interaction Module** to place buy and sell orders.

   Considerations:
   - Requires sophisticated algorithms to quickly decide whether to execute and the amount of Bitcoin to trade.

4. **Database/Storage**
   Function:
   - Stores historical price data, trade logs and other relevant information.
   - Useful for analytics, monitoring past history and improving the arbitrage strategy.

   Considerations:
   - Requires robust database management for accurate record-keeping.

5. **User Dashboard**
   Function:
   - Provides a visual interface for monitoring bot activity and performance
   - Allows users to set parameters such as which exchanges(A or B) to monitor, thresholds for arbitrage, etc.

   Considerations:
   - Needs to be user-friendly but is not critical for the bot's core functionality.

6. **Notification and Alert System**
   Function:
   - Sends alerts or notifications based on certain triggers, like successful trades or significant arbitrage opportunities.

   Considerations:
   - Important for monitoring bot activities and taking action in case of anomalies.

Design Choices and Trade-offs:

1. **Modularity**
   - **Advantage:** Easy to integrate new exchanges.
   - **Trade-off:** Requires initial effort to integrate with new exchange's APIs.
2. **Real-time Data Processing**
   - **Advantage:** Quick detection of arbitrage opportunities.
   - **Trade-off:** Higher computational resources.
3. **Autonomy vs Control**
   - **Advantage:** Can operate autonomously for efficiency.
   - **Trade-off:** May need manual intervention for unusual market conditions. This includes the modification of thresholds to manage arbitrage trading.
4. **Data Storage:**
   - **Advantage:** Valuable for analytics and decision improvement.
   - **Trade-off:** Data storage and management costs.
5. **User Interface Dashboard**
   - **Advantage:** Improved usability (User can modify configurations easily.)
   - **Trade-off:** Development and maintenance resources.
6. **Notification and Alert System**
   - **Advantage:** Keeps users informed and engaged.
   - **Trade-off:** Potential information overload if not finely tuned.