**High Level Design For Trade Matching**

**Diagram

Description automatically generated**

**High Level Diagram for Matching**

**Different components of trade matching**

1. Models (Trade Data, Validation table, Matching table)
2. Services (Business logic for matching trades)
3. Controller (API request and response handler)
4. Repository (used for CRUD operations)
5. Swagger (connected to controller for doing API requests)

**API 1 (Create a new trade)**

* Post request from swagger will be handled through controller and based on logic in service class, trade data will be saved in the database (using repository).
* Validation and matching are done before saving the data in database.

**Validation will be of following type:**

* Based on valid parties, ID.
* Based on valid trade dates, notional amount.
* Seller and buyer can’t be same.
* Status will be unconfirmed by default for an unmatched trade.

**For Matching:**

* Matching will be done based on matching fields.
* If matching score will be 100%, status will be changed to confirmed in party as well as in counter party trade.
* If matching score will be less than 100%, then party and counter party trade reference number will be saved in a matching table with status “unconfirmed” and with its respective matching score percentage.

**API 2 (Search for trade)**

**Based on party name and status:** Search all the trades of a party with provided status. (Input string => confirmed, cancel, unconfirmed, exit)

* **Based on party name and trade reference number:**
* **If counter party existed:** Will return a trade along with the matching score.
* **If counter party doesn’t exist:** Will return the trade data.
* **If use input wrong string**: Data not present.

**API 3 (Update an existing trade)**

* **Version:** It will be incremented on every new updation (initially it is 1).
* **Updation states:**
* **For confirmed trade:** Status will be updated to “confirmed” and version will be incremented.
* **For cancelled trade:** If the user wants to cancel the trade the status will be updated to “cancelled” and version will be incremented. The existing trade will be searched based on trade reference number
* **For exit trade:** This state will be triggered on maturity date and the status will be updated to “exit” and version will be incremented.
* **For updating existing trade:** Based on party name and trade reference number the user can search the trade and update the desired records except status and version.