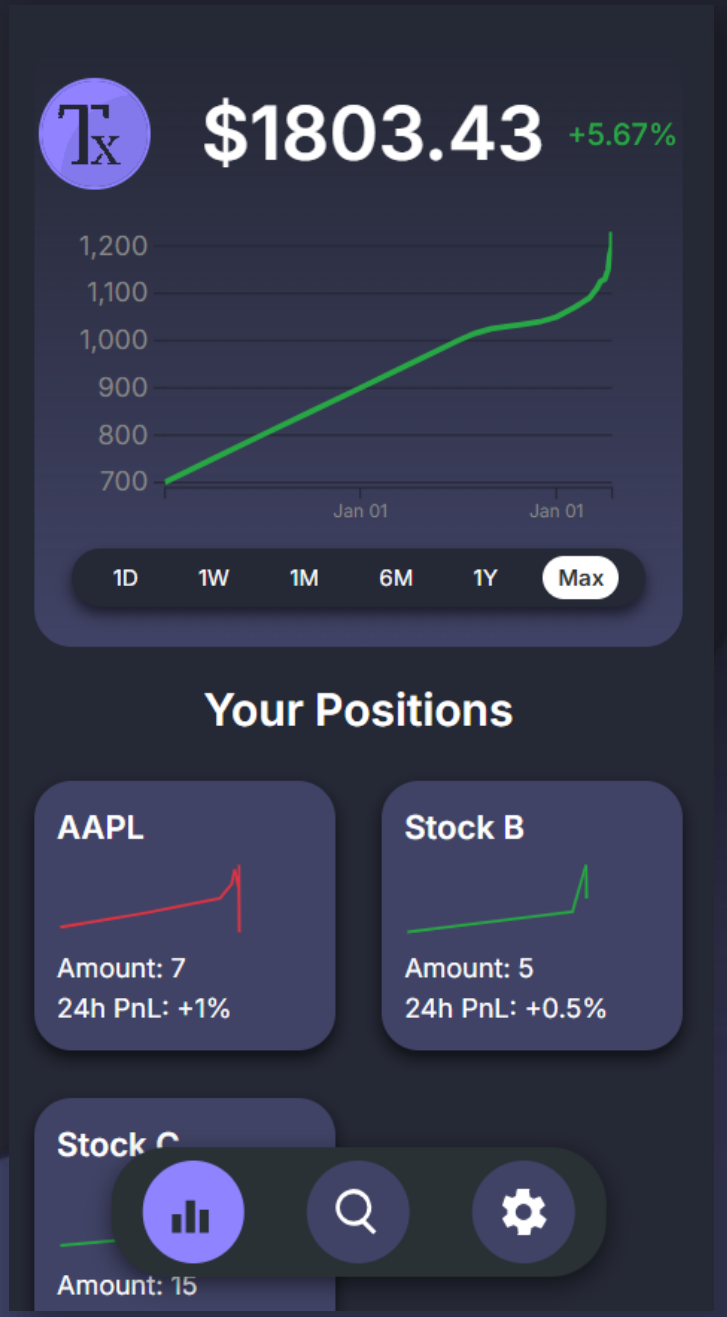
Presenting: Thales Exchange

Simulated Stock Exchange

The basic mechanism + idea

- A simulated stock exchange to enable students to trade against each other
- Helping users to gain experience with stock markets and trading
- Users can place bids or asks in order books, which are then matched
- Displays the history of stock markets



The personas

Travis

- History 1st year

-knows nothing about trading

-wants to learn the basics of the stock market

Sofia

-CS 3rd year

-interested in quant

-wants to test out bots against other players

Myles

-E&M 1st year

-knows about discretionary investing

-wants to practice fundamental investing on a simulated market

```python
def sell_stock(self, stock_id: int, price: float, vol: int):
    # add money to balance
    self.balance += price * vol

    # remove stock from portfolio
    ticker = OrderBook.get_ticker_by_id(stock_id)
    if not ticker:
        raise ValueError("Stock does not exist")
    if vol <= 0:
        raise ValueError("Must sell a positive amount")

    if not ticker in self.portfolio:
        raise ValueError(f"Seller {self.username} does not own the
    else:
        if self.portfolio[ticker] < vol:
            raise ValueError(f"Seller {self.username} has insuffi
        self.portfolio[ticker] -= vol
        if self.portfolio[ticker] == 0:  # if stock is no longer
            del self.portfolio[ticker]  # remove from portfolio

def display_portfolio(self) -> str:
    res = f"Portfolio of {str(self)}:"
    for ticker in self.portfolio:
        res += f"\n  {ticker}:\t  {self.portfolio[ticker]}"
    return res

def display_balance(self) -> str:
    return f"Balance of {str(self)}:\t  {self.balance}"
```

The order book

- Built in Python
- Follows the Jane Street spec of order books
- Handles Limit and Market orders

Client		
client_id	Int	PK
username	String	Unique, Not null
email	String	Unique, Not null
balance	Double	balance >= 0, default 100
first_names	String	
last_name	String	

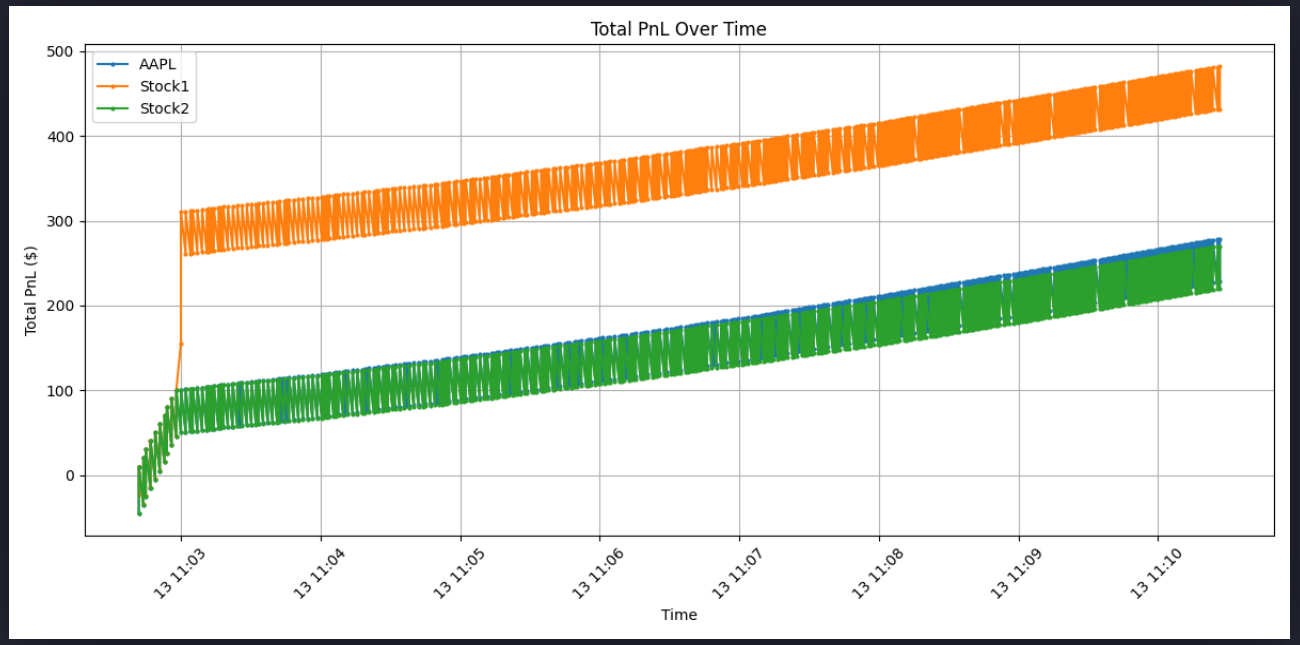
OwnedStock		
owner_id	Int	PK, FK
ticker	String	PK
average_price	Double	Not null, average_price > 0
total_vol	Int	Not null, total_vol > 0

Transactions		
transaction_id	Int	PK
bidder_id	Int	FK, Not null
bid_price	Double	Not null, bid_price >= ask_price
asker_id	Int	FK, Not null
ask_price	Double	Not null, ask_price > 0
vol	Int	Not null, vol > 0
ticker	String	Not null
time_stamp	String	Not null
transaction_price	Double	Not null, transaction_price = ask_price OR transaction_price = bid_price

The database

- Client: Records all user information
- OwnedStock: Records of all currently owned stock
- Transactions: Records of all completed trades

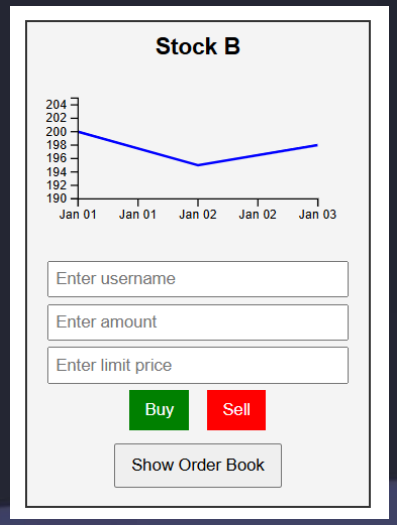
Market bots



- Price feed REST API
- Increase liquidity on exchange
- Limit orders which capture spread

Timeline, challenges and solutions

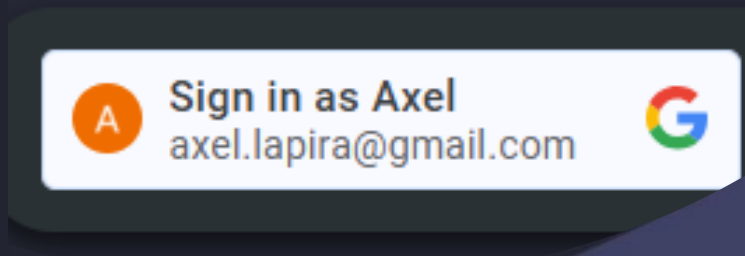
Front end MVP and Database setup → Order book → Front-end cleanup +backend → Log-in/Auth, Fixing bugs, backend-database link → Market maker bots → ...



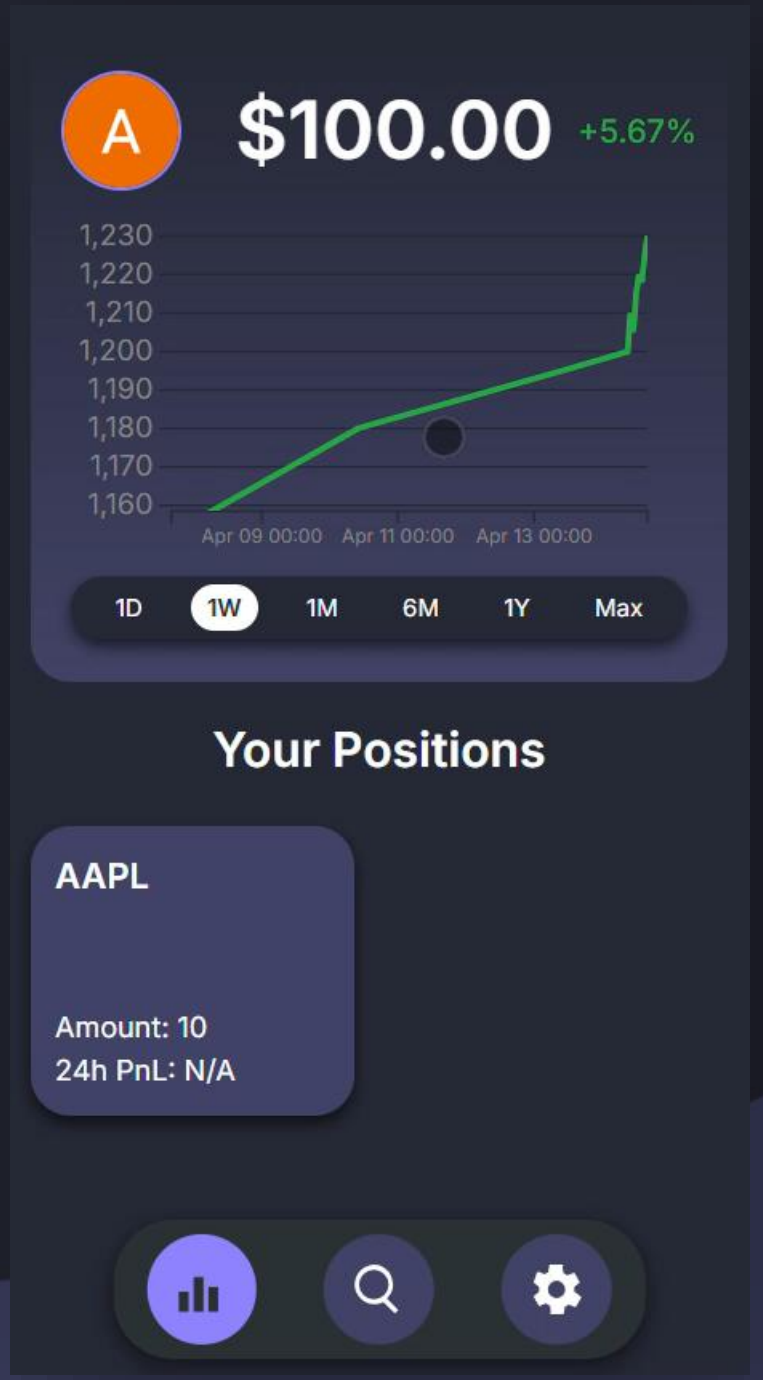

```
self.balance += price * vol

# remove stock from portfolio
ticker = OrderBook.get_ticker_by_id(stock_id)
if not ticker:
    raise ValueError("Stock does not exist")
if vol <= 0:
    raise ValueError("Must sell a positive amount")

if not ticker in self.portfolio:
```



Demo


A
$100.00
+5.67%
1,230
1,220
1,210
1,200
1,190
1,180
1,170
1,160
Apr 09 00:00
Apr 11 00:00
Apr 13 00:00
1D
1W
1M
6M
1Y
Max
Your Positions
AAPL
Amount: 10
24h PnL: N/A

Future plans

- Competitions!!
- Fine-tuning bot logic
- Order editing
- Fraud detection

Q&A and Thank You!