стовпці: id, валютну пару, таймфрейм, час й ціну відкриття й закриття позиції й прибуток зароблений за одну виконану угоду. Реалізований варіант вікна повністю відповідає спроєктованому.

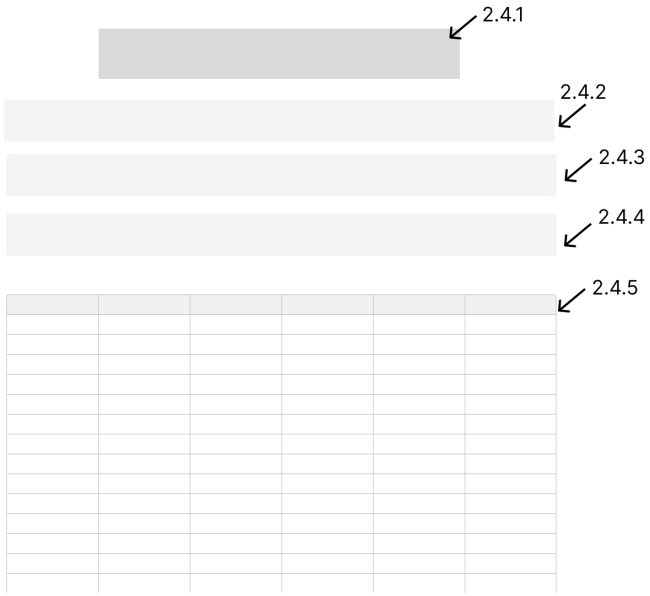


Рисунок 4.9 – Прототип вікна для результатів live-торгівлі

Таким чином, торговий агент містить 9 вікон, які спроєктовані так, щоб задовольняти всі потреби для дій користувача й наладчика системи, а також мати інтуїтивно-зрозумілий й ефективний для роботи інтерфейс.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проаналізовано предметну галузь, обрано ефективні методи для створення системи, спроєктовано прототипи й реалізовано додаток торгового агенту. Реалізований торговий агент виконує основну функцію торгівлі в режимі реального часу для користувачів, які хочу заробити пасивного гроші, але при цьому не розуміються на трейдингу. А також додано функцію бектестування для покращення параметрів торгівлі для наладчика й аналізу виконаної торгівлі.

В результаті отримано торговий агент, який виконує функції, що були поставлені на початку. Звичайно, ця торгова система ще не готова до загального використання, але базова реалізація присутня. Торговий агент виконує свою основну функцію – це пасивний заробіток шляхом виконання автоматизованої торгівлі з використанням методів машинного навчання.

Звичайно торговий агент можна ще довго покращувати й оснащувати додатковими функціями. Наприклад, по-перше, перенесення системи на сервер з MT5 для загального користування додатку деякими окремими користувачами одночасно. По-друге, додавання більшої кількості валютних пар для можливості вибору більш підходящої для різних етапів ринку. По-третє, прив’язування до додатку систем оплати для вводу й виводу грошей з системи. По-четверте, використання торгівлі в реальному часу на рахунку з реальними грошима, який приєднаний до брокера, бо під час розробки роботи був використаний демо-варіант торгівлі, який має абсолютні однакові графіки з реальним акаунт, різниця полягає тільки в «несправжніх» грошах на демо-акаунті.

Таким чином, в ході виконання кваліфікаційної роботи було створено торговий агент, який може використовуватись для торгівлі в режимі реального часу для користувачів, які прагнуть отримувати пасивний дохід без вкладення свого часу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) 2022 triennial central bank survey. *Bank for international settlements*. URL: https://www.bis.org/statistics/rpfx22_fx_annex.pdf (date of access: 23.04.2025).

2) Chen J. Forex market: definition, how it works, types, and trading risks. *Investopedia*. URL: https://www.investopedia.com/terms/forex/f/forex-market.asp (date of access: 23.04.2025).

3) Grossbard J. Forex trading statistics. *CompareForexBrokers*. URL: https://www.compareforexbrokers.com/trading/statistics/ (date of access: 25.04.2025).

4) What is forex trading? CMC Markets. URL: https://www.cmcmarkets.com/en/forex/what-is-forex-trading (date of access: 29.04.2025).

5) What is forex and how does it work?. *IG*. URL: https://www.ig.com/en/forex/what-is-forex-and-how-does-it-work (date of access: 21.04.2025).

6) The complete guide to trading strategies and styles. *IG*. URL: https://www.ig.com/au/trading-strategies/the-complete-guide-to-trading-strategies-and-styles-190709 (date of access: 24.04.2025).

7) Должикова К. І. Використання методів машинного навчання для торгівлі на ринку Forex. *Радіоелектроника та молодь у XXI столітті. Том 6* : Матеріали XXIX Міжнар. молодіж. форуму, м. Харків. 2025. С. 24–25.

8) What is fundamental analysis & how to use it in trading? *Online Trading Broker*. URL: https://www.axi.com/int/blog/education/fundamental-analysis (date of access: 01.05.2025).

9) Technical and fundamental analysis: a beginners guide. *Corporate Finance Institute*. URL: https://cfi.trade/en/blog/trading/technical-and-fundamental-analysis-a-beginners-guide (date of access: 02.05.2025).

10) The ultimate guide to forex analysis in 2025. *Trendo*. URL: https://fxtrendo.com/blog/244/Forex-analysis-types-the-best-analysis-method (date of access: 02.05.2025).

11) Leaman K. Exploring the potential of machine learning in trading. *AvaTrade*. URL: https://www.avatrade.com/blog/trading-tools-technologies/machine-learning-trading (date of access: 03.05.2025).

12) Jansen S. Machine learning for algorithmic trading : навчальний посібник. 2nd ed. 2020. 820 p.

13) ML algorithms in the markets. *Medium*. URL: https://medium.com/@thornexdaniel/ml-algorithms-in-the-markets-ddbff48c7e0 (date of access: 05.05.2025).

14) Ullrich C., Reese D., Chalup S. Foreign exchange trading with support vector machines. *Studies in classification, data analysis, and knowledge organization*. URL: https://link.springer.com/chapter/10.1007/978-3-540-70981-7_62 (date of access: 06.05.2025).

15) Yıldırım D. C., Toroslu I. H., Fiore U. Forecasting directional movement of forex data using LSTM. *Financial innovation*. 2021. Vol. 7, no. 1. URL: https://jfin-swufe.springeropen.com/articles/10.1186/s40854-020-00220-2 (date of access: 06.05.2025).

16) Gaussian smoothing in time series data. *Medium*. URL: https://medium.com/data-science/gaussian-smoothing-in-time-series-data-c6801f8a4dc3 (date of access: 04.05.2025).

17) How random forests can improve macro trading signals. *Macrosynergy*. URL: https://macrosynergy.com/research/how-random-forests-can-improve-macro-trading-signals/ (date of access: 06.05.2025).

18) Gaussian filter - rules, settings, strategy, returns - quantifiedstrategies.com. *QuantifiedStrategies*.URL: https://www.quantifiedstrategies.com/gaussian-filter/ (date of access: 10.05.2025).

19) Regmi S. Gaussian smoothing in time series data. *Medium*. URL: https://medium.com/data-science/gaussian-smoothing-in-time-series-data-c6801f8a4dc3 (date of access: 11.05.2025).

20) Red S. KNN machine learning strategy: trend prediction trading system based on k-nearest neighbors. *Medium*. URL: https://medium.com/@redsword_23261/knn-machine-learning-strategy-trend-prediction-trading-system-based-on-k-nearest-neighbors-3aa36a9374c3 (date of access: 13.05.2025).

21) Gunay D. KNN algorithm. *Medium*. URL: https://medium.com/@denizgunay/knn-algorithm-3604c19cd809 (date of access: 15.05.2025).

22) Heikin-Ashi technique. *Corporate Finance Institute*. URL: https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/heikin-ashi-technique/ (date of access: 16.05.2025).

23) Simple moving average (SMA). *Fidelity*. URL: https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/sma#:~:text=Simple%20Moving%20Average%20(SMA)&text=It%20is%20simply%20the%20average,used%20to%20determine%20trend%20direction (date of access: 19.05.2025).

24) Красько Я. RSI relative strength index | bikotrading.pro. *Bikotrading.pro*. URL: https://bikotrading.pro/rsi-indikator (date of access: 21.05.2025).

ДОДАТОК А

UML-схема для користувачів

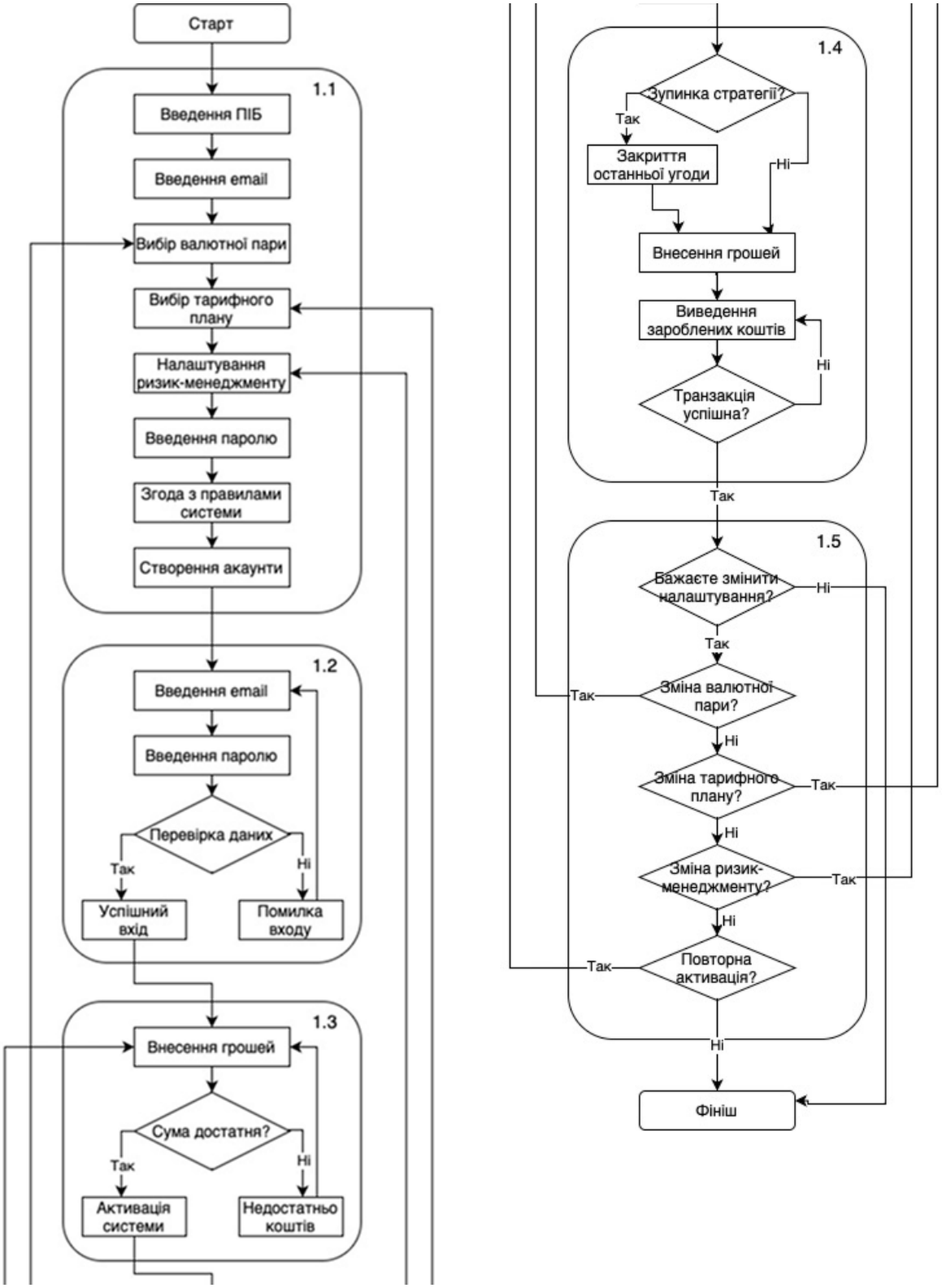


Рисунок А.1 – UML-схема для користувача

ДОДАТОК Б

Програмний код торгового агента

Лістинг Б.1 – Програмний код для підключення до MT5

```
class MT5Connector:
    def __init__(self):
        self._initialized = False
    def initialize(self):
        if not mt5.initialize():
            raise ConnectionError("Не вдалося виконати підключення до MetaTrader5")
        self._initialized = True
    def shutdown(self):
        if self._initialized:
            mt5.shutdown()
            self._initialized = False
    def _tf_constant(self, timeframe: str):
        mapping = {"M1": mt5.TIMEFRAME_M1,
            "M5": mt5.TIMEFRAME_M5,
            "M15": mt5.TIMEFRAME_M15,
            "M30": mt5.TIMEFRAME_M30,
            "H1": mt5.TIMEFRAME_H1,
            "H4": mt5.TIMEFRAME_H4,
            "D1": mt5.TIMEFRAME_D1,}
        if timeframe not in mapping:
            raise ValueError(f"Невідомий '{timeframe}'")
        return mapping[timeframe]
    def get_historical(self, symbol, timeframe, bars):
        if not self._initialized:
            self.initialize()
        bars = int(bars)
        if isinstance(timeframe, str):
            tf_map = {"M5": mt5.TIMEFRAME_M5,
                "M15": mt5.TIMEFRAME_M15,
                "M30": mt5.TIMEFRAME_M30,
```

Продовження лістингу Б.1

```
                    "H1": mt5.TIMEFRAME_H1,}
            timeframe = tf_map.get(timeframe,
mt5.TIMEFRAME_M15)
            rates = mt5.copy_rates_from_pos(symbol, timeframe,
0, bars)
            df = pd.DataFrame(rates)
            df['time'] = pd.to_datetime(df['time'], unit='s')
            df.set_index('time', inplace=True)
            self.shutdown()
            return df
        def get_historical_range(self, symbol: str, timeframe:
int, start: datetime.date, end: datetime.date) -> pd.DataFrame:
            if not self._initialized:
                self.initialize()
            dt_start = datetime.combine(start, dt_time.min)
            dt_end   = datetime.combine(end,   dt_time.max)
            rates  =  mt5.copy_rates_range(symbol,  timeframe,
dt_start, dt_end)
            df = pd.DataFrame(rates)
            if 'time' in df.columns:
                df['time']=pd.to_datetime(df['time'],unit='s')
                df.set_index('time', inplace=True)
            else:
                df.index = pd.to_datetime(df.index, unit='s')
            mt5.shutdown()
            return df
        def get_symbol_info(self, symbol: str):
            syms = mt5.symbols_get()
            if syms is None:
                if not mt5.initialize():
                    raise     ConnectionError("Не     вдалося
з’єднатися з MT5 для отримання списку символів")
                syms = mt5.symbols_get()
                if syms is None:
```

Продовження лістингу Б.1

```
                raise RuntimeError("Не вдалося отримати список символів від MetaTrader5")
            names = [s.name for s in syms]
            match = next((n for n in names if n.lower() == symbol.lower()), None)
            if not match:
                raise RuntimeError(f"Символ '{symbol}' не знайдено в обліковому записі")
            if not mt5.symbol_select(match, True):
                raise RuntimeError(f"Символ {match} знайдено, але не вдалось додати у MarketWatch")
            info = mt5.symbol_info(match)
            if info is None:
                raise RuntimeError(f"SymbolInfo для {match} повернув None")
            return info
        def get_symbol_tick(self, symbol: str):
            tick = mt5.symbol_info_tick(symbol)
            if tick is None:
                raise RuntimeError(f"Не вдалось знайти tick для {symbol}")
            return tick
```

Лістинг Б.2 – Реалізація методу генерації сигналів

```
    def generate_signals(df: pd.DataFrame, tp_pips: int = 40, sl_pips: int = 50, tp_atr_mult: float = None, sl_atr_mult: float = None, atr_period: int = 14,
                         gf_length: int = 40, gf_sigma: float = 10.0, knn_window: int = 5
    ) -> pd.DataFrame:
        df = df.copy()
        df.ffill(inplace=True)
        df['ATR'] = compute_atr(df, period=atr_period)
        df = compute_heiken_ashi(df)
```

Продовження лістингу Б.2

```
        gf = GaussianFilter(length=gf_length, sigma=gf_sigma)
        df['Smoothed'] = gf.apply(df['HA_Close'])
        df['hl2'] = (df['high'] + df['low']) / 2
        df['kNN_MA']  =  df['hl2'].rolling(window=knn_window,
min_periods=1).mean()
        df['signal'] = 0
        cond_long1  =  (df['HA_Close']  >  df['Smoothed'])  &
(df['Smoothed'].diff() > 0) & (df['kNN_MA'].diff() > 0)
        cond_short1 = (df['HA_Close']  < df['Smoothed'])
        cond_short1  =  (df['HA_Close']  <  df['kNN_MA'])  &
(df['Smoothed'].diff() < 0) & (df['kNN_MA'].diff() < 0)
        df.loc[cond_long1,  'signal'] =  1
        df.loc[cond_short1, 'signal'] =  2
        df.loc[(df['HA_Close']       <    df['Smoothed'])    &
(df['Smoothed'].diff() < 0) \
               & (df['kNN_MA'].diff()   < 0), 'signal'] = -1
        df.loc[(df['HA_Close']        >    df['kNN_MA'])    &
(df['Smoothed'].diff() > 0) \
               & (df['kNN_MA'].diff()  > 0), 'signal'] = -2
        if tp_atr_mult is not None:
            df['tp_pips'] = (df['ATR'] * tp_atr_mult).round(1)
        else:
            df['tp_pips'] = tp_pips
        if sl_atr_mult is not None:
            df['sl_pips'] = (df['ATR'] * sl_atr_mult).round(1)
        else:
            df['sl_pips'] = sl_pips
        return df
```

Лістинг Б.3 – Реалізація класу TradeExecutor

```
class TradeExecutor:
    def __init__(self, mt5_conn, user_email):
        self.mt5 = mt5_conn
        self.user_email = user_email
```

Продовження лістингу Б.3

```
            self._create_trade_logs_db()
        def send_order(self, symbol, order_type, price, volume,
sl_price, tp_price):
            info = mt5.symbol_info(symbol)
            if info is None:
                return None
            point = info.point
            min_dist = info.trade_stops_level * point
            if sl_price:
                if abs(price - sl_price) < min_dist:
                    direction   =   1   if   order_type   ==
mt5.ORDER_TYPE_BUY else -1
                    sl_price = price - direction * min_dist
            if tp_price:
                if abs(tp_price - price) < min_dist:
                    direction   =   1   if   order_type   ==
mt5.ORDER_TYPE_BUY else -1
                    tp_price = price + direction * min_dist
            request = {'action': mt5.TRADE_ACTION_DEAL,
                'symbol': symbol,
                'volume': volume,
                'type': order_type,
                'sl': sl_price or 0.0,
                'tp': tp_price or 0.0,
                'type_time': mt5.ORDER_TIME_GTC,
                'type_filling': mt5.ORDER_FILLING_IOC}
            res = mt5.order_send(request)
            return res
        def _create_trade_logs_db(self):
            conn = sqlite3.connect("db/trade_logs.db")
            cursor = conn.cursor()
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS trade_logs (
                Id      INTEGER     PRIMARY KEY AUTOINCREMENT,
```

Продовження лістингу Б.3

```
                user        TEXT        NOT NULL,
                symbol      TEXT        NOT NULL,
                side        TEXT        NOT NULL,
                open_time   TEXT        NOT NULL,
                open_price  REAL        NOT NULL,
                lot         REAL        NOT NULL,
                tp_price    REAL,
                sl_price    REAL,
                close_time  REAL,
                close_price REAL,
                profit      REAL)
                """)
            conn.commit()
            conn.close()
        def log_open_trade(self, symbol, side, open_time, open_price, lot, tp_price, sl_price):
            now = datetime.now().isoformat(sep = '', timespec='seconds')
            with self._connect() as c:
                cursor = c.execute("""
                    INSERT INTO trades
                        (user, symbol, side, open_time, open_price, lot, tp_price, sl_price)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?)
                """, (self.user, symbol, side, now, open_price, lot, tp_price, sl_price))
                c.commit()
                return cursor.lastrowid
        def log_close_trade(self,trade_id,close_price,profit):
            now = datetime.now().isoformat(sep = '', timespec='seconds')
            with self._connect() as c:
                cursor = c.execute("""
                    UPDATE trades
```

Продовження лістингу Б.3

```
                    SET close_time=?, close_price=?, profit=?
                    WHERE id = ?
                """, (now, close_price, profit, trade_id))
                c.commit()
        def open_long(self, symbol, price, lot, tp_price,
sl_price):
            res = self.send_order(symbol, mt5.ORDER_TYPE_BUY,
lot, price, sl_price, tp_price)
            if res and res.retcode == mt5.TRADE_RETCODE_DONE:
                tid = self.log_open_trade(symbol, 'buy', price,
lot, tp_price, sl_price)
                self._open_trades[(symbol, "buy")] = tid
            return res

        def close_long(self, symbol):
            positions = mt5.positions_get(symbol=symbol) or []
            for pos in positions:
                if pos.type == mt5.ORDER_TYPE_BUY:
                    tick = mt5.symbol_info_tick(symbol)
                    res = self.send_order(symbol,
mt5.ORDER_TYPE_SELL, pos.volume, tick.bid, 0, 0)
                    if res and res.retcode ==
mt5.TRADE_RETCODE_DONE:
                        profit = (tick.bid - pos.price_open) *
pos.volume * pos.trade_contract_size
                        tid = self._open_trades.pop((symbol,
"buy"), None)
                        if tid:
                            self.log_close_trade(tid,
tick.bid, profit)
        def open_short(self, symbol, price, lot, tp_price,
sl_price):
            res = self.send_order(symbol, mt5.ORDER_TYPE_SELL,
lot, price, sl_price, tp_price)
```

Продовження лістингу Б.3

```
            if res and res.retcode == mt5.TRADE_RETCODE_DONE:
                tid = self.log_open_trade(symbol, 'sell',
price, lot, tp_price, sl_price)
                self._open_trades[(symbol, "sell")] = tid
            return res
        def close_short(self, symbol):
            positions = mt5.positions_get(symbol=symbol) or []
            for pos in positions:
                if pos.type == mt5.ORDER_TYPE_SELL:
                    tick = mt5.symbol_info_tick(symbol)
                    res = self.send_order(symbol,
mt5.ORDER_TYPE_BUY, pos.volume, tick.ask, 0, 0)
                    if res and res.retcode ==
mt5.TRADE_RETCODE_DONE:
                        profit = (pos.price_open - tick.ask) *
pos.volume * pos.trade_contract_size
                        tid = self._open_trades.pop((symbol,
"sell"), None)
                        if tid:
                            self.log_close_trade(tid,
tick.ask, profit)
```

Лістинг Б.4 – Реалізація класу TradingEngine

```
class TradingEngine:
    def __init__(self, symbol: str, timeframe: str = "M15",
                 tp_pips: int = 40, sl_pips: int = 50,
                 atr_period:int=14, tp_atr_mult:float=None,
                 sl_atr_mult: float = None,
                 history_bars: int = 200,
                 polling_interval: float = 60.0,
                 lot: float = 0.1, user_email = None,
                 gf_length: int = 40,gf_sigma: float = 10.0,
                 knn_window: int = 5):
        self.symbol = symbol
```

Продовження лістингу Б.4

```
            self.timeframe = timeframe
            self.tp_pips = tp_pips
            self.sl_pips = sl_pips
            self.atr_period = atr_period
            self.tp_atr_mult = tp_atr_mult
            self.sl_atr_mult = sl_atr_mult
            self.history_bars = history_bars
            self.polling_interval = polling_interval
            self.gf_length = gf_length
            self.gf_sigma = gf_sigma
            self.knn_window = knn_window
            self.lot = lot
            self.mt5 = MT5Connector()
            self.mt5.initialize()
            self.executor = TradeExecutor(self.lot,user_email)
            self._last_signal = 0

            self._last_time = None
            self._running = False
            self._thread = None
        def start(self):
            if self._running:
                return
            self._running = True
            self._thread                                        =
threading.Thread(target=self._run_loop, daemon=True)
            self._thread.start()
        def stop(self):
            self._running = False
            if self._thread:
                self._thread.join(timeout=1)
        def _run_loop(self):
            while self._running:
```

Продовження лістингу Б.4

```
                df = self.mt5.get_historical(symbol=self.symbol,
timeframe=self.timeframe, bars=self.history_bars)
                  df.index = pd.to_datetime(df.index)
                  signals = generate_signals(df,
                      tp_pips=self.tp_pips,sl_pips=self.sl_pips,
                      tp_atr_mult=self.tp_atr_mult,
                      sl_atr_mult=self.sl_atr_mult,
                      atr_period=self.atr_period,
                      gf_length=self.gf_length,
                      gf_sigma=self.gf_sigma,
                      knn_window=self.knn_window)
                  row = signals.iloc[-1]
                  sig = int(row['signal'])
                  price = row['HA_Close']
                  tp = row['tp_pips']
                  sl = row['sl_pips']
                  if sig != self._last_signal:
                      if sig == 1:
                          self.executor.open_long(self.symbol,
price, self.lot, tp, sl)
                      elif sig == 2:
                          self.executor.close_long(self.symbol)
                      elif sig == -1:
                          self.executor.open_short(self.symbol,
price, self.lot, tp, sl)
                      elif sig == -2:
                          self.executor.close_short(self.symbol)
                      self._last_signal = sig
                  time.sleep(self.polling_interval)
          def run_backtest(self, start: datetime, end: datetime)
-> (pd.DataFrame, float):
              df   =   self.mt5.get_historical_range(symbol   =
self.symbol, timeframe=self.timeframe, start=start, end=end)
              df.index = pd.to_datetime(df.index)
```

Продовження лістингу Б.4

```
        df.ffill(inplace=True)
        df.bfill(inplace=True)
        df_sig = generate_signals(df,
            tp_pips=self.tp_pips,sl_pips=self.sl_pips,
            tp_atr_mult=self.tp_atr_mult,
            sl_atr_mult=self.sl_atr_mult,
            atr_period=self.atr_period,
            gf_length=self.gf_length,
            gf_sigma=self.gf_sigma,
            knn_window=self.knn_window)
        info = self.mt5.get_symbol_info(self.symbol)
        pip = info.point
        contract = info.trade_contract_size
        total_pnl = 0.0
        position = 0
        trades = []
        entry_price=entry_time = tp_price = sl_price = None
        for r in df_sig.itertuples():
            sig = r.signal
            price_close = r.HA_Close
            price_high = r.HA_High
            price_low = r.HA_Low
            tp_p = r.tp_pips
            sl_p = r.sl_pips
            if sig == 1 and position == 0:
                position = 1
                entry_price = price_close
                entry_time = r.Index
                tp_price = entry_price + tp_p * pip
                sl_price = entry_price - sl_p * pip
            elif position == 1:
                hit_tp = price_high >= tp_price
                hit_sl = price_low <= sl_price
                if hit_tp or hit_sl or sig == 2:
```

Продовження лістингу Б.4

```
                        if hit_tp and hit_sl:
                            if abs(tp_price - entry_price) <
abs(entry_price - sl_price):
                                hit_sl = False
                            else:
                                hit_tp = False
                        diff = (tp_price - entry_price) if
hit_tp else (sl_price - entry_price) if hit_sl else (price_close
- entry_price)
                        profit = diff * contract * self.lot
                        total_pnl += profit
                        trades.append({
                            'entry time': entry_time,
                            'exit_time':r.index,'side':'Long',
                            'entry_price': entry_price,
                            'exit_price': tp_price if hit_tp
else sl_price if hit_sl else price_close,'profit': profit})
                        position = 0
                if sig == -1 and position == 0:
                    position = -1
                    entry_price = price_close
                    entry_time = r.index
                    tp_price = entry_price - tp_p * pip
                    sl_price = entry_price + sl_p * pip
                elif position == -1:
                    hit_tp = price_low <= tp_price
                    hit_sl = price_high >= sl_price
                    if hit_tp or hit_sl or sig == -2:
                        if hit_tp and hit_sl:
                            if abs(entry_price - tp_price) <
abs(sl_price - entry_price):
                                hit_sl = False
                            else:
                                hit_tp = False
```

Продовження лістингу Б.4

```
                        diff = (entry_price - tp_price) if
hit_tp else (entry_price - sl_price) if hit_sl else (entry_price
- price_close)
                        profit = diff * contract * self.lot
                        total_pnl += profit
                        trades.append({
                            'entry time': entry_time,
                           'exit_time':r.index,'side':'Short',
                            'entry_price': entry_price,
                            'exit_price': tp_price if hit_tp
else sl_price if hit_sl else price_close,'profit': profit})
                        position = 0
            trades_df = pd.DataFrame(trades)
            return df_sig, trades_df, total_pnl
```

ДОДАТОК В

Реалізовані інтерфейси торгового агенту

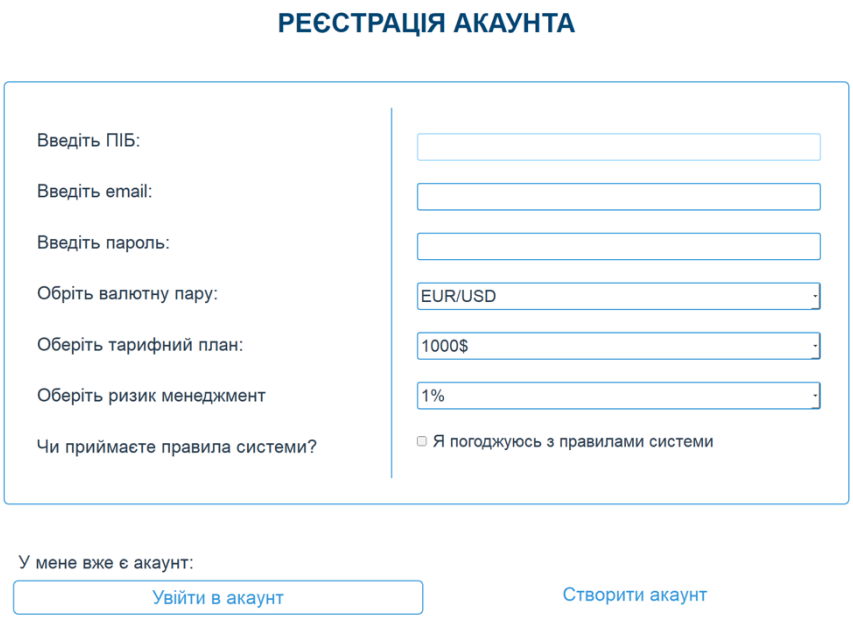


Рисунок В.1 – Вікно реєстрації акаунта

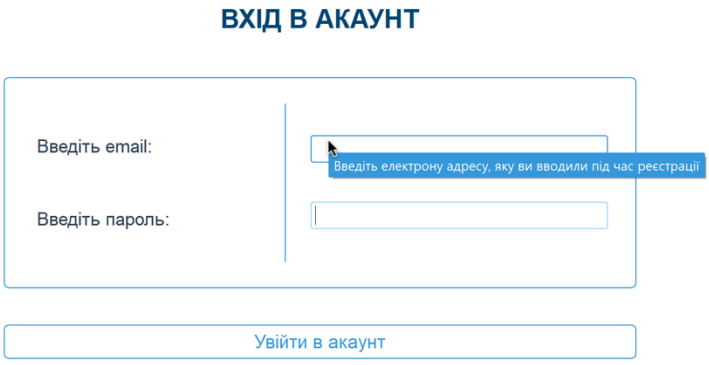


Рисунок В.2 – Вікно входу в акаунт

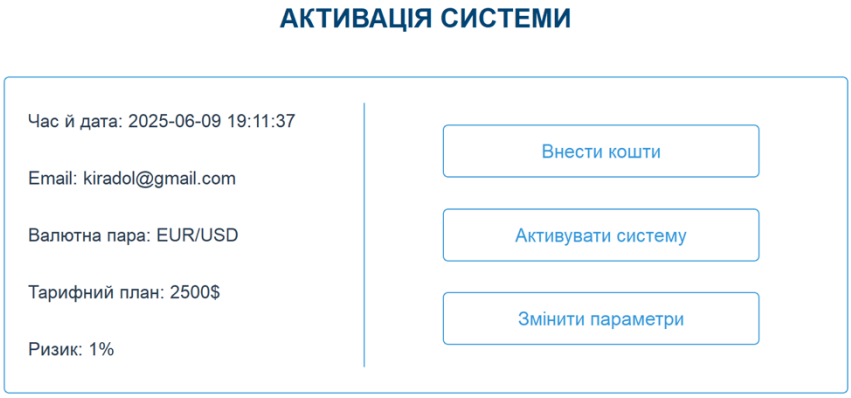


Рисунок В.3 – Вікно активації системи для користувача

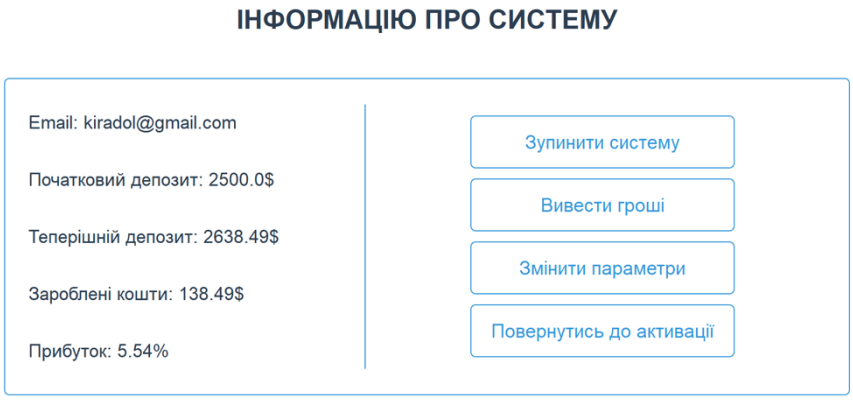


Рисунок В.4 – Вікно інформації про систему для користувача

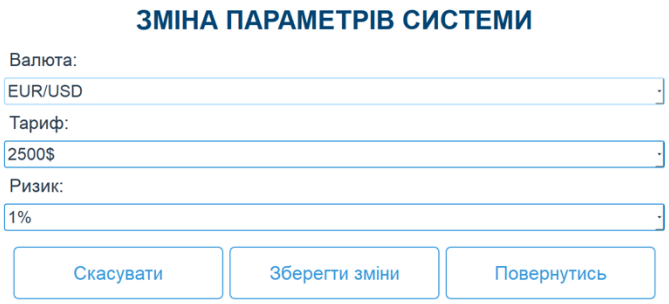


Рисунок В.5 – Вікно для зміни параметрів системи для користувача

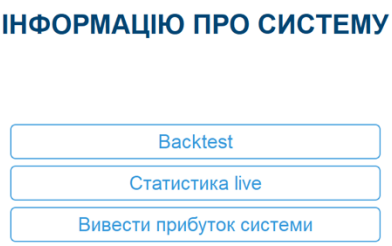


Рисунок В.6 – Вікно інформації про систему для наладчика

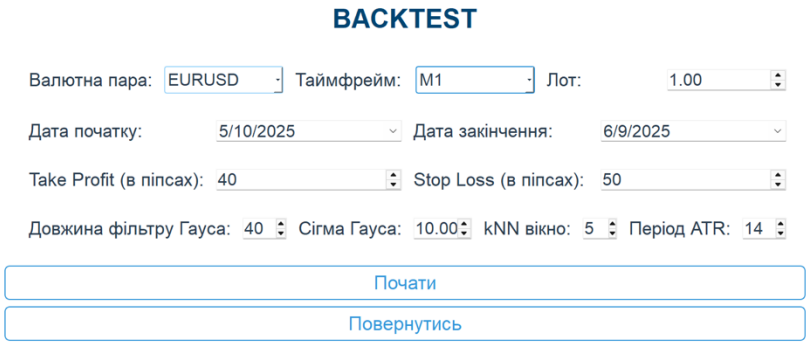


Рисунок В.7 – Вікно для вибору параметрів для проведення бектестування для наладчика

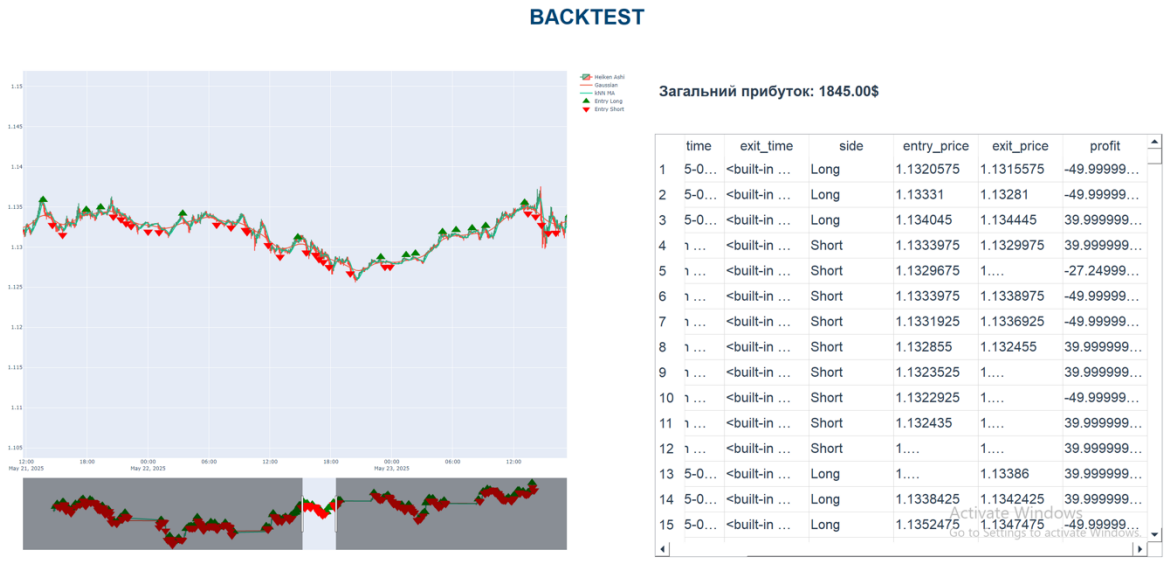
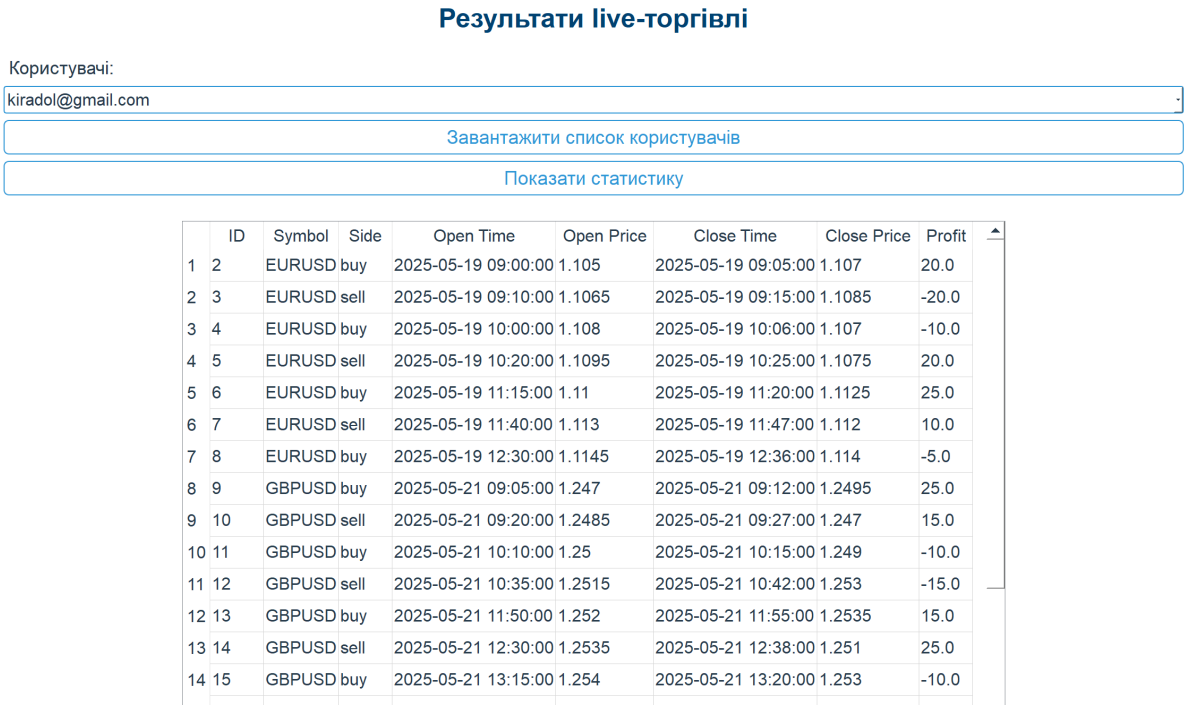


Рисунок В.8 – Вікно результатів бектестування для наладчика



	ID	Symbol	Side	Open Time	Open Price	Close Time	Close Price	Profit
1	2	EURUSD	buy	2025-05-19 09:00:00	1.105	2025-05-19 09:05:00	1.107	20.0
2	3	EURUSD	sell	2025-05-19 09:10:00	1.1065	2025-05-19 09:15:00	1.1085	-20.0
3	4	EURUSD	buy	2025-05-19 10:00:00	1.108	2025-05-19 10:06:00	1.107	-10.0
4	5	EURUSD	sell	2025-05-19 10:20:00	1.1095	2025-05-19 10:25:00	1.1075	20.0
5	6	EURUSD	buy	2025-05-19 11:15:00	1.11	2025-05-19 11:20:00	1.1125	25.0
6	7	EURUSD	sell	2025-05-19 11:40:00	1.113	2025-05-19 11:47:00	1.112	10.0
7	8	EURUSD	buy	2025-05-19 12:30:00	1.1145	2025-05-19 12:36:00	1.114	-5.0
8	9	GBPUSD	buy	2025-05-21 09:05:00	1.247	2025-05-21 09:12:00	1.2495	25.0
9	10	GBPUSD	sell	2025-05-21 09:20:00	1.2485	2025-05-21 09:27:00	1.247	15.0
10	11	GBPUSD	buy	2025-05-21 10:10:00	1.25	2025-05-21 10:15:00	1.249	-10.0
11	12	GBPUSD	sell	2025-05-21 10:35:00	1.2515	2025-05-21 10:42:00	1.253	-15.0
12	13	GBPUSD	buy	2025-05-21 11:50:00	1.252	2025-05-21 11:55:00	1.2535	15.0
13	14	GBPUSD	sell	2025-05-21 12:30:00	1.2535	2025-05-21 12:38:00	1.251	25.0
14	15	GBPUSD	buy	2025-05-21 13:15:00	1.254	2025-05-21 13:20:00	1.253	-10.0

Рисунок В.9 – Вікно для аналізу виконаних торгових угод в режимі реального часу для наладчика

ДОДАТОК Г

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	96 с.
	Інші документи	
2.	Презентаційні матеріали	16 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Должикова К.І.		20.06	Інтелектуальний торговий агент для торгівлі на ринку Forex	Шифр групи	Код напр./спец.
Перевірив		Поліїт М.Р.				ІТШІ-21-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					