Groupwork: Logic Coverage for Your Project

Team 7

- 312551074 陳柏翰
- 312551080 紀軒宇
- 312553008 陳沛圻

Spec

Line module

```
def create_line_user(user_id, line_id)
def get_user_by_line_id(line_id)
```

Account module

```
def create_record(user_id, date, item, cost, category, comment)
def show_recent_record(user_id, num=5, days=3, type='num')
def search_record(user_id, date_from, date_to=None)
def update_record(user_id, record_id, date = None, item=None, cost=None,
category=None, comment=None)
def delete_record(user_id, record_id)
def export_record(user_id, method='this_month')
```

Message parser module

```
def parse_message(message)
```

Main module

```
def handle_message(event)
```

Note

- 對每個我們 spec 中的 function，我們將設計測試以滿足 PC, CC, 和 CACC。
- 假設有一 predicate P = A && B，測試資料表達方法為: `T t t`: P 為 true, A 為 true, B 為 true
- 針對此 groupwork 所設計的測資皆包含於 Github Repo `test` 資料夾中的測資，測試結果如下

```
============================================================================
======================= test session starts
============================================================================
=========================
platform linux -- Python 3.10.12, pytest-8.2.0, pluggy-1.5.0
rootdir: /home/bhchen/NYCU_program/NYCU_Software_Testing_Final/test
plugins: cov-5.0.0, mock-3.14.0
collected 96 items

accounting_test.py ....................................
[ 41%]
line_test.py .......
[ 48%]
main_test.py ......................
[ 72%]
message_parser_test.py ..........................
[100%]

============================================================================
======================== 96 passed in 0.55s
============================================================================
=========================
```

Line module

def create_line_user(self, line_id):

Source code:

```
def create_line_user(self, line_id):
        success = False
        user = None
        error_message = None


        # check if line_id is valid
        if not isinstance(line_id, str):
            error_message = 'invalid line_id parameter'
            return success, user, error_message

        # DB related
        conn = sqlite3.connect(self.db_name)
        cursor = conn.cursor()

        # check if line_id is already in the database
        cursor.execute('SELECT * FROM user WHERE line_id = ?', (line_id,))
        row = cursor.fetchone()
        if row != None:
            error_message = 'line_id already exists'
            return success, user, error_message
```

```
        try:
            # insert new user
            cursor.execute('INSERT INTO user (line_id) VALUES (?)',
(line_id,))
            new_user_id = cursor.lastrowid

            # get user_id and create_date from new data
            cursor.execute('SELECT * FROM user WHERE user_id = ?',
(new_user_id,))
            row = cursor.fetchone()
            user = User.User(user_id=row[0], line_id=row[1],
create_date=row[2])
            success = True
        except Exception as e:
            error_message = str(e)
            conn.rollback()

        conn.commit()
        conn.close()
        # DB related end

        return success, user, error_message
```

對於此 function，我們有兩個 predicate

- not isinstance(line_id, str)，以下簡稱 P1
  - A: isinstance(line_id, str)
  - P1: !A
  - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
  - 測試資料設計如下
    - T f
      - create_line_user(123456789)
    - F t
      - create_line_user("U123456789")
      - mock 資料庫使 fetchone() 回傳 None
- row != None，以下簡稱 P2
  - A: row != None
  - P2: A
  - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
  - 測試資料設計如下
    - T t
      - create_line_user("U123456789")
      - mock 資料庫使 fetchone() 回傳 User 資料為非 None
    - F f
      - create_line_user("U123456789")
      - mock 資料庫使 fetchone() 回傳 User 資料為 None

def get_user_by_line_id(self, line_id)

Predicate 1

```
# check if line_id is valid
if not isinstance(line_id, str):
    error_message = 'invalid line_id parameter'
    return success, user, error_message
```

- A: `isinstance(line_id, str)`
- P1: `!A`
- PC
    - A: False, P1: True
        - get_user_by_line_id(123456789)
    - A: True, P1: False
        - get_user_by_line_id("U123456789")
- CC
    - A: True, P1: True
        - get_user_by_line_id("U123456789")
    - A: False, P1: True
        - get_user_by_line_id(123456789)
- CACC
    - **A**: False, P1: True
        - get_user_by_line_id(123456789)
    - **A**: True, P1: False
        - get_user_by_line_id("U123456789")

Predicate 2

```
# find user by line_id
cursor.execute('SELECT * FROM user WHERE line_id = ?', (line_id,))
row = cursor.fetchone()
if row == None:
    error_message = 'user not found'
    return success, user, error_message
```

- A: `row == None`
- P2: `A`
- PC

    - A: True, P2: True
        - get_user_by_line_id("U123456789")
    - A: False, P2: False
        - get_user_by_line_id("U123456789")

- CC

    - A: True, P2: True
        - get_user_by_line_id("U123456789")
    - A: False, P2: False
        - get_user_by_line_id("U123456789")

- CACC

    - **A**: True, P2: True
        - get_user_by_line_id("U123456789")
    - **A**: False, P2: False
        - get_user_by_line_id("U123456789")

Accounting module

def create_record(self, user_id, date, item, cost, category, comment)

Source code:

```
    def create_record(self, user_id, date, item, cost, category, comment):
        success = False
        record = None
        error_message = None

        # check if user_id is valid
        if not isinstance(user_id, int):
            error_message = 'invalid line_id parameter'
            return success, record, error_message

        # check if date is valid
        if not isinstance(date, str):
            error_message = 'invalid date parameter'
            return success, record, error_message

        # check if item is valid
        if not isinstance(item, str):
            error_message = 'invalid item parameter'
            return success, record, error_message

        # check if cost is valid
        if not isinstance(cost, int):
            error_message = 'invalid cost parameter'
            return success, record, error_message

        # check if category is valid
```

```
        if not isinstance(category, str):
            error_message = 'invalid category parameter'
            return success, record, error_message

        # check if comment is valid
        if not isinstance(comment, str):
            error_message = 'invalid comment parameter'
            return success, record, error_message

        # DB related
        conn = sqlite3.connect(self.db_name)
        cursor = conn.cursor()

        # check if user_id exists
        cursor.execute('SELECT * FROM user WHERE user_id = ?', (user_id,))
        row = cursor.fetchone()
        if row == None:
            error_message = 'user_id does not exist'
            return success, record, error_message

        try:
            # insert new record
            cursor.execute('INSERT INTO record (user_id, date, item, cost,
category, comment) VALUES (?, ?, ?, ?, ?, ?)', (user_id, date, item, cost,
category, comment))
            new_record_id = cursor.lastrowid

            # get record_id and create_date from new data
            cursor.execute('SELECT * FROM record WHERE record_id = ?',
(new_record_id,))
            row = cursor.fetchone()
            record = Record.Record(row[0], row[1], row[2], row[3], row[4],
row[5], row[6], row[7])
            record.date = str(record.date)
            record.create_date = str(record.create_date)
            success = True
        except Exception as e:
            error_message = str(e)
            conn.rollback()

        conn.commit()
        conn.close()
        # DB related end

        return success, record, error_message
```

對於此 function，我們有七個 predicate

- not isinstance(user_id, int):，以下簡稱 P1
    - A: isinstance(user_id, int)
    - P1: !A

    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - create_record('1', '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - Expected: error_message = 'invalid line_id parameter'
        - F t
            - create_record(1 , '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - mock 資料庫使 fetchone() 分別回傳 User 資料和 Record 資料皆為非 None
            - Expected: success = True, record = Record(1, '20240101', 'apple', 20 , 'food', 'good_to_eat')

- not isinstance(date, str):，以下簡稱 P2

    - A: isinstance(date, str)
    - P2: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - create_record(1 , 20240101, 'apple', 20 , 'food', 'good_to_eat')
            - Expected: error_message = 'invalid date parameter'
        - F t
            - create_record(1 , '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - mock 資料庫使 fetchone() 分別回傳 User 資料和 Record 資料皆為非 None
            - Expected: success = True, record = Record(1, '20240101', 'apple', 20 , 'food', 'good_to_eat')

- not isinstance(item, str):，以下簡稱 P3

    - A: isinstance(item, str)
    - P3: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - create_record(1 , '20240101', 123, 20 , 'food', 'good_to_eat')
            - Expected: error_message = 'invalid item parameter'
        - F t
            - create_record(1 , '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - mock 資料庫使 fetchone() 分別回傳 User 資料和 Record 資料皆為非 None
            - Expected: success = True, record = Record(1, '20240101', 'apple', 20 , 'food', 'good_to_eat')

- not isinstance(cost, int):，以下簡稱 P4

    - A: isinstance(cost, int)
    - P4: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f

            - create_record(1 , '20240101', 'apple', '20' , 'food', 'good_to_eat')
            - Expected: error_message = 'invalid cost parameter'
        - F t
            - create_record(1 , '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - mock 資料庫使 fetchone() 分別回傳 User 資料和 Record 資料皆為非 None
            - Expected: success = True, record = Record(1, '20240101', 'apple', 20 , 'food', 'good_to_eat')

- not isinstance(category, str):
    - A: isinstance(category, str)
    - P5: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - create_record(1 , '20240101', 'apple', 20 , 123, 'good_to_eat')
            - Expected: error_message = 'invalid category parameter'
        - F t
            - create_record(1 , '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - mock 資料庫使 fetchone() 分別回傳 User 資料和 Record 資料皆為非 None
            - Expected: success = True, record = Record(1, '20240101', 'apple', 20 , 'food', 'good_to_eat')

- not isinstance(comment, str):
    - A: isinstance(comment, str)
    - P6: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - create_record(1 , '20240101', 'apple', 20 , 'food', 123)
            - Expected: error_message = 'invalid comment parameter'
        - F t
            - create_record(1 , '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - mock 資料庫使 fetchone() 分別回傳 User 資料和 Record 資料皆為非 None
            - Expected: success = True, record = Record(1, '20240101', 'apple', 20 , 'food', 'good_to_eat')

- row == None:
    - A: row == None
    - P7: A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T t
            - create_record(2 , '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - mock 資料庫使 fetchone() 回傳 User 資料為 None
            - Expected: error_message = 'user_id does not exist'

        - F f
            - create_record(1 , '20240101', 'apple', 20 , 'food', 'good_to_eat')
            - mock 資料庫使 fetchone() 回傳 User 資料和 Record 資料皆為非 None
            - Expected: success = True, record = Record(1, '20240101', 'apple', 20 , 'food', 'good_to_eat')

def show_recent_record(self, user_id, num=5, days=3, type='num'):

Source code:

```
    def show_recent_record(self, user_id, num=5, days=3, type='num'):
        success = False
        records = None
        error_message = None

        if not isinstance(user_id, int):
            error_message = 'invalid user_id parameter'
            return success, records, error_message

        if not isinstance(num, int):
            error_message = 'invalid num parameter'
            return success, records, error_message

        if not isinstance(days, int):
            error_message = 'invalid day parameter'
            return success, records, error_message

        if type!='num' and type!='days':
            error_message = 'invalid type parameter'
            return success, records, error_message

        # DB related
        conn = sqlite3.connect(self.db_name)
        cursor = conn.cursor()

        # check if user_id exists
        cursor.execute('SELECT * FROM user WHERE user_id = ?', (user_id,))
        row = cursor.fetchone()
        if row == None:
            error_message = 'user_id does not exist'
            return success, records, error_message

        if type == 'num':
            cursor.execute('SELECT * FROM record WHERE user_id = ? ORDER BY
create_date DESC LIMIT ?', (user_id, num))
        elif type == 'days':
            cursor.execute('SELECT * FROM record WHERE user_id = ? AND
create_date >= date("now", "-' + str(days) + ' day") ORDER BY create_date
DESC', (user_id,))

        rows = cursor.fetchall()
        records = []
```

```
        for row in rows:
            record = Record.Record(row[0], row[1], row[2], row[3], row[4],
row[5], row[6], row[7])
            record.date = str(record.date)
            record.create_date = str(record.create_date)
            records.append(record)
        success = True

        return success, records, error_message
```

對於此 function，我們有 6 個 predicate

- if not isinstance(user_id, int):，以下簡稱 P1
    - A: isinstance(user_id, int)
    - P1: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - show_recent_record('1', 3 , 1 , 'num')
            - Expected: error_message = 'invalid user_id parameter'
        - F t
            - show_recent_record(1 , 3 , 1 , 'num')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
- if not isinstance(num, int):，以下簡稱 P2
    - A: isinstance(num, int)
    - P2: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - show_recent_record(1 , '3' , 1 , 'num')
            - Expected: error_message = 'invalid num parameter'
        - F t
            - show_recent_record(1 , 3 , 1 , 'num')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
- if not isinstance(days, int):，以下簡稱 P3
    - A: isinstance(days, int)
    - P3: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f

            - show_recent_record(1 , 3 , '1' , 'num')
            - Expected: error_message = 'invalid day parameter'
        - F t
            - show_recent_record(1 , 3 , 1 , 'num')
            - Expected: success = True, len(records) = 3

- if type!='num' and type!='days':，以下簡稱 P4
    - A: type != 'num'
    - B: type != 'days'
    - P4: A && B
    - PC:
        - T t t
            - show_recent_record(1 , 3 , 1 , 123)
            - Expected: error_message = 'invalid type parameter'
        - F t f
            - show_recent_record(1 , 3 , 1 , 'num')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
    - CC:
        - F t f
            - show_recent_record(1 , 3 , 1 , 'days')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
        - F f t
            - show_recent_record(1 , 3 , 1 , 'num')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
    - CACC:
        - T t t (A as Major)
            - show_recent_record(1 , 3 , 1 , 123)
            - Expected: error_message = 'invalid type parameter'
        - F f t (A as Major)
            - show_recent_record(1 , 3 , 1 , 'num')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
        - F t f (B as Major)
            - show_recent_record(1 , 3 , 1 , 'days')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3

- if row == None:，以下簡稱 P5
    - A: row == None
    - P5: A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T t

        - show_recent_record(2 , 3 , 1 , 'num')
        - mock 資料庫使 fetchone() 回傳 User 資料為 None
        - Expected: error_message = 'user_id does not exist'
    - F f
        - show_recent_record(1 , 3 , 1 , 'num')
        - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
        - Expected: success = True, len(records) = 3

- if type == 'num': ... elif type == 'days': ...，以下簡稱 P6
    - A: type == 'num'
    - B: type == 'days'
    - C6: A || (!A && B) 簡化後為 A || B
    - PC:
        - 無法完成 PC，因在實作上執行到此時，type 只會是 'num' 或 'days' 其中一個，故 predicate 必定為 true
    - CC:
        - T t f
            - show_recent_record(1 , 3 , 1 , 'num')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
        - T f t
            - show_recent_record(1 , 3 , 1 , 'days')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
    - CACC:
        - A as major
            - T t f (pick this)
            - F f f
        - B as major
            - T f t (pick this)
            - F f f
        - T t f (A as Major)
            - show_recent_record(1 , 3 , 1 , 'num')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
        - T f t (B as Major)
            - show_recent_record(1 , 3 , 1 , 'days')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3

def search_record(self, user_id, date_from, date_to=None):

Source Code:

```
    def search_record(self, user_id, date_from, date_to=None):
        success = False
```

```
        records = None
        error_message = None

        if not isinstance(user_id, int):
            error_message = 'invalid user_id parameter'
            return success, records, error_message

        if not isinstance(date_from, str):
            error_message = 'invalid date_from parameter'
            return success, records, error_message

        if date_to != None and not isinstance(date_to, str):
            error_message = 'invalid date_to parameter'
            return success, records, error_message

        # DB related
        conn = sqlite3.connect(self.db_name)
        cursor = conn.cursor()

        # check if user_id exists
        cursor.execute('SELECT * FROM user WHERE user_id = ?', (user_id,))
        row = cursor.fetchone()
        if row == None:
            error_message = 'user_id does not exist'
            return success, records, error_message

        if date_to == None:
            cursor.execute('SELECT * FROM record WHERE user_id = ? AND date
= ?', (user_id, date_from))
        else:
            cursor.execute('SELECT * FROM record WHERE user_id = ? AND date
>= ? AND date <= ?', (user_id, date_from, date_to))

        rows = cursor.fetchall()
        records = []

        for row in rows:
            record = Record.Record(row[0], row[1], row[2], row[3], row[4],
row[5], row[6], row[7])
            record.date = str(record.date)
            record.create_date = str(record.create_date)
            records.append(record)
        success = True

        return success, records, error_message
```

對於此 function，我們有 5 個 predicate

- if not isinstance(user_id, int):，以下簡稱 P1
    - A: isinstance(user_id, int)
    - P1: !A

    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - search_record('1', '20240101', '20240103')
            - Expected: error_message = 'invalid user_id parameter'
        - F t
            - search_record(1, '20240101', '20240103')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆，當中有 3 筆符合條件
            - Expected: success = True, len(records) = 3

- if not isinstance(date_from, str):，以下簡稱 P2

    - A: isinstance(date_from, str)
    - P2: !A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T f
            - search_record(1, 20240101, '20240103')
            - Expected: error_message = 'invalid date_from parameter'
        - F t
            - search_record(1, '20240101', '20240103')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆，且當中有 3 筆符合條件
            - Expected: success = True, len(records) = 3

- if date_to != None and not isinstance(date_to, str):，以下簡稱 P3

    - A: date_to != None
    - B: isinstance(date_to, str)
    - P3: A && !B
    - PC:
        - T t f
            - search_record(1, '20240101', 20240103)
            - Expected: error_message = 'invalid date_to parameter'
        - F f t
            - search_record(1, '20240101', '20240103')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆，且當中有 3 筆符合條件
            - Expected: success = True, len(records) = 3
    - CC:
        - F t t
            - search_record(1, '20240101', '20240103')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3
        - F f f
            - search_record(1, '20240101', None)

            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆，但僅有 1 筆符合條件
            - Expected: success = True, len(records) = 1
    - CACC:
        - A as major
            - T t f
            - F f f (pick this)
        - B as major
            - T t f
            - F t t (pick this)
        - F f f (A as Major)
            - search_record(1, '20240101', None)
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆，但僅有 1 筆符合條件
            - Expected: success = True, len(records) = 1
        - F t t (B as Major)
            - search_record(1, '20240101', '20240103')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3

- if row == None:，以下簡稱 P4
    - A: row == None
    - P4: A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T t
            - search_record(2, '20240101', '20240103')
            - mock 資料庫使 fetchone() 回傳 User 資料為 None
            - Expected: error_message = 'user_id does not exist'
        - F f
            - search_record(1, '20240101', '20240103')
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆
            - Expected: success = True, len(records) = 3

- if date_to == None:，以下簡稱 P5
    - A: date_to == None
    - P5: A
    - 因此 predicate 只有一個 clause，故此處 CC = PC = CACC
    - 測試資料設計如下
        - T t
            - search_record(1, '20240101', None)
            - mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆，且當中有 1 筆符合條件
            - Expected: success = True, len(records) = 1
        - F f
            - search_record(1, '20240101', '20240103')

- mock 資料庫使 fetchone() 回傳 User 資料為非 None，且 Record 資料有 3 筆，且當中有 3 筆符合條件
- Expected: success = True, len(records) = 3

update record

```
def update_record(self, user_id, record_id, date=None, item=None,
cost=None, category=None, comment=None):
        success = False
        record = None
        error_message = None

        # check if user_id is valid
        if not isinstance(user_id, int):
            error_message = 'invalid user_id parameter'
            return success, record, error_message

        # check if record_id is valid
        if not isinstance(record_id, int):
            error_message = 'invalid record_id parameter'
            return success, record, error_message

        # check if item is valid
        if item != None and not isinstance(item, str):
            error_message = 'invalid item parameter'
            return success, record, error_message

        # check if cost is valid
        if cost != None and not isinstance(cost, int):
            error_message = 'invalid cost parameter'
            return success, record, error_message

        # check if category is valid
        if category != None and not isinstance(category, str):
            error_message = 'invalid category parameter'
            return success, record, error_message

        # check if comment is valid
        if comment != None and not isinstance(comment, str):
            error_message = 'invalid comment parameter'
            return success, record, error_message

        # check if date is valid
        if date != None and not isinstance(date, str):
            error_message = 'invalid date parameter'
            return success, record, error_message
        # DB related
        conn = sqlite3.connect(self.db_name)
        cursor = conn.cursor()

        # check if user_id exists
```

```
        cursor.execute(
            'SELECT * FROM record WHERE user_id = ? AND record_id = ?',
(user_id, record_id))
        row = cursor.fetchone()
        if row == None:
            error_message = 'the record of this id does not exist'
            return success, record, error_message
        # update record
        try:
            cursor.execute('UPDATE record SET item = COALESCE(?, item),
cost = COALESCE(?, cost), category = COALESCE(?, category), comment =
COALESCE(?, comment), date = COALESCE(?, date) WHERE user_id = ? AND
record_id = ?',
                           (item, cost, category, comment, date, user_id,
record_id))
            conn.commit()
            cursor.execute(
                'SELECT * FROM record WHERE user_id = ? AND record_id = ?',
(user_id, record_id))
            row = cursor.fetchone()
            record = Record.Record(
                row[0], row[1], row[2], row[3], row[4], row[5], row[6],
row[7])
            record.date = str(record.date)
            record.create_date = str(record.create_date)
            success = True
        except Exception as e:
            error_message = str(e)
            conn.rollback()
        conn.close()
        return success, record, error_message
```

有 8 predicates

- if not isinstance(user_id, int):
  - predicate coverage
    - T
      - ('123', 1, None, None, None, None, None)
    - F
      - (123, 1, None, None, None, None, None)
  - clause coverage (same as predicate coverage)
    - (not isinstance(user_id, int)) **(True)** (T)
      - ('123', 1, None, None, None, None, None)
    - (not isinstance(user_id, int)) **(False)** (F)
      - (123, 1, None, None, None, None, None)
  - correlative active clause coverage (same as predicate coverage)
    - (not isinstance(user_id, int)) **(True)** (T)
      - ('123', 1, None, None, None, None, None)
    - (not isinstance(user_id, int)) **(False)** (F)
      - (123, 1, None, None, None, None, None)

- if not isinstance(record_id, int):
    - predicate coverage
        - T
            - (1, '123', None, None, None, None, None)
        - F
            - (1, 123, None, None, None, None, None)
    - clause coverage (same as predicate coverage)
        - (not isinstance(record_id, int)) **(True)** (T)
            - (1, '123', None, None, None, None, None)
        - (not isinstance(record_id, int)) **(False)** (F)
            - (1, 123, None, None, None, None, None)
    - correlative active clause coverage (same as predicate coverage)
        - (not isinstance(record_id, int)) **(True)** (T)
            - (1, '123', None, None, None, None, None)
        - (not isinstance(record_id, int)) **(False)** (F)
            - (1, 123, None, None, None, None, None)
- if item != None and not isinstance(item, str):
    - predicate coverage
        - T
            - (1, 1, None, 123, None, None, None)
        - F
            - (1, 1, None, '123', None, None, None)
    - clause coverage
        - (item != None) **(True)** and (not isinstance(item, str)) **(True)** (T)
            - (1, 1, None, 123, None, None, None)
        - (item != None) **(True)** and (not isinstance(item, str)) **(False)** (F)
            - (1, 1, None, '123', None, None, None)
        - (item != None) **(False)** (F)
            - (1, 1, None, None, None, None, None)
    - correlative active clause coverage
        - (item != None)為Major
            - (item != None) **(True)** and (not isinstance(item, str)) **(True)** (T)
                - (1, 1, None, 123, None, None, None)
            - (item != None) **(False)** and (not isinstance(item, str)) **(True)** (F) (不可能發生)
        - (not isinstance(item, str))為Major
            - (item != None) **(True)** and (not isinstance(item, str)) **(True)** (T)
                - (1, 1, None, 123, None, None, None)
            - (item != None) **(True)** and (not isinstance(item, str)) **(False)** (F)
                - (1, 1, None, '123', None, None, None)
- if cost != None and not isinstance(cost, int):
    - predicate coverage
        - T
            - (1, 1, None, None, '123', None, None)
        - F
            - (1, 1, None, None, 123, None, None)
    - clause coverage

        - (cost != None) **(True)** and (not isinstance(cost, int)) **(True)** (T)
            - (1, 1, None, None, '123', None, None)
        - (cost != None) **(True)** and (not isinstance(cost, int)) **(False)** (F)
            - (1, 1, None, None, 123, None, None)
        - (cost != None) **(False)** (F)
            - (1, 1, None, None, None, None, None)
    - correlative active clause coverage
        - (cost != None)為Major
            - (cost != None) **(True)** and (not isinstance(cost, int)) **(True)** (T)
                - (1, 1, None, None, '123', None, None)
            - (cost != None) **(False)** and (not isinstance(cost, int)) **(True)** (F) (不可能發生)
        - (not isinstance(cost, int))為Major
            - (cost != None) **(True)** and (not isinstance(cost, int)) **(True)** (T)
                - (1, 1, None, None, '123', None, None)
            - (cost != None) **(True)** and (not isinstance(cost, int)) **(False)** (F)
                - (1, 1, None, None, 123, None, None)
- if category != None and not isinstance(category, str):
    - predicate coverage
        - T
            - (1, 1, None, None, None, 123, None)
        - F
            - (1, 1, None, None, None, '123', None)
    - clause coverage
        - (category != None) **(True)** and (not isinstance(category, str)) **(True)** (T)
            - (1, 1, None, None, None, 123, None)
        - (category != None) **(True)** and (not isinstance(category, str)) **(False)** (F)
            - (1, 1, None, None, None, '123', None)
        - (category != None) **(False)** (F)
            - (1, 1, None, None, None, None, None)
    - correlative active clause coverage
        - (category != None)為Major
            - (category != None) **(True)** and (not isinstance(category, str)) **(True)** (T)
                - (1, 1, None, None, None, 123, None)
            - (category != None) **(False)** and (not isinstance(category, str)) **(True)** (F) (不可能發生)
        - (not isinstance(category, str))為Major
            - (category != None) **(True)** and (not isinstance(category, str)) **(True)** (T)
                - (1, 1, None, None, None, 123, None)
            - (category != None) **(True)** and (not isinstance(category, str)) **(False)** (F)
                - (1, 1, None, None, None, '123', None)
- if comment != None and not isinstance(comment, str):
    - predicate coverage
        - T
            - (1, 1, None, None, None, None, 123)
        - F
            - (1, 1, None, None, None, None, '123')

    - clause coverage
        - (comment != None) **(True)** and (not isinstance(comment, str)) **(True)** (T)
            - (1, 1, None, None, None, None, 123)
        - (comment != None) **(True)** and (not isinstance(comment, str)) **(False)** (F)
            - (1, 1, None, None, None, None, '123')
        - (comment != None) **(False)** (F)
            - (1, 1, None, None, None, None, None)
    - correlative active clause coverage
        - (comment != None)為Major
            - (comment != None) **(True)** and (not isinstance(comment, str)) **(True)** (T)
                - (1, 1, None, None, None, None, 123)
            - (comment != None) **(False)** and (not isinstance(comment, str)) **(True)** (F) (不可能發生)
        - (not isinstance(comment, str))為Major
            - (comment != None) **(True)** and (not isinstance(comment, str)) **(True)** (T)
                - (1, 1, None, None, None, None, 123)
            - (comment != None) **(True)** and (not isinstance(comment, str)) **(False)** (F)
                - (1, 1, None, None, None, None, '123')
- if date != None and not isinstance(date, str):
    - predicate coverage
        - T
            - (1, 1, 20240103, None, None, None, None)
        - F
            - (1, 1, '20240103', None, None, None, None)
    - clause coverage
        - (date != None) **(True)** and (not isinstance(date, str)) **(True)** (T)
            - (1, 1, 20240103, None, None, None, None)
        - (date != None) **(True)** and (not isinstance(date, str)) **(False)** (F)
            - (1, 1, '20240103', None, None, None, None)
        - (date != None) **(False)** (F)
            - (1, 1, None, None, None, None, None)
    - correlative active clause coverage
        - (date != None)為Major
            - (date != None) **(True)** and (not isinstance(date, str)) **(True)** (T)
                - (1, 1, 20240103, None, None, None, None)
            - (date != None) **(False)** and (not isinstance(date, str)) **(True)** (F) (不可能發生)
        - (not isinstance(date, str))為Major
            - (date != None) **(True)** and (not isinstance(date, str)) **(True)** (T)
                - (1, 1, 20240103, None, None, None, None)
            - (date != None) **(True)** and (not isinstance(date, str)) **(False)** (F)
                - (1, 1, '20240103', None, None, None, None)
- if row == None:
    - predicate coverage
        - T
            - (1, 2, None, None, None, None, None) (if the record of this id does not exist)
        - F

        - (1, 1, None, None, None, None, None) (if the record of this id exists)
    - clause coverage(same as predicate coverage)
        - (row == None) **(True)** (T)
            - (1, 2, None, None, None, None, None) (if the record of this id does not exist)
        - (row == None) **(False)** (F)
            - (1, 1, None, None, None, None, None) (if the record of this id exists)
    - correlative active clause coverage(same as predicate coverage)
        - (row == None) **(True)** (T)
            - (1, 2, None, None, None, None, None) (if the record of this id does not exist)
        - (row == None) **(False)** (F)
            - (1, 1, None, None, None, None, None) (if the record of this id exists)

delete record

```
def delete_record(self, user_id, record_id):
        success = False
        error_message = None

        # check if user_id is valid
        if not isinstance(user_id, int):
            error_message = 'invalid user_id parameter'
            return success, None, error_message

        # check if record_id is valid
        if not isinstance(record_id, int):
            error_message = 'invalid record_id parameter'
            return success, None, error_message

        # DB related
        conn = sqlite3.connect(self.db_name)
        cursor = conn.cursor()

        # check if the record of the user_id exists
        cursor.execute(
            'SELECT * FROM record WHERE user_id = ? AND record_id = ?',
(user_id, record_id))
        row = cursor.fetchone()
        if row == None:
            error_message = 'the record of this id does not exist'
            return success, None, error_message

        try:
            cursor.execute(
                'DELETE FROM record WHERE record_id = ? AND user_id = ?',
(record_id, user_id))
            conn.commit()
            success = True

        except Exception as e:
```

```
            error_message = str(e)
            conn.rollback()

        conn.close()
        return success, None, error_message
```

有 3 predicates

- if not isinstance(user_id, int):
  - predicate coverage
    - T
      - ('123', 1)
    - F
      - (123, 1)
  - clause coverage (same as predicate coverage)
    - (not isinstance(user_id, int)) **(True)** (T)
      - ('123', 1)
    - (not isinstance(user_id, int)) **(False)** (F)
      - (123, 1)
  - correlative active clause coverage (same as predicate coverage)
    - (not isinstance(user_id, int)) **(True)** (T)
      - ('123', 1)
    - (not isinstance(user_id, int)) **(False)** (F)
      - (123, 1)
- if not isinstance(record_id, int):
  - predicate coverage
    - T
      - (1, '123')
    - F
      - (1, 123)
  - clause coverage (same as predicate coverage)
    - (not isinstance(record_id, int)) **(True)** (T)
      - (1, '123')
    - (not isinstance(record_id, int)) **(False)** (F)
      - (1, 123)
  - correlative active clause coverage (same as predicate coverage)
    - (not isinstance(record_id, int)) **(True)** (T)
      - (1, '123')
    - (not isinstance(record_id, int)) **(False)** (F)
      - (1, 123)
- if row == None:
  - predicate coverage
    - T
      - (1, 2) (if the record of this id does not exist)
    - F
      - (1, 1) (if the record of this id exists)

- clause coverage(same as predicate coverage)
  - (row == None) **(True)** (T)
    - (1, 2) (if the record of this id does not exist)
  - (row == None) **(False)** (F)
    - (1, 1) (if the record of this id exists)
- correlative active clause coverage(same as predicate coverage)
  - (row == None) **(True)** (T)
    - (1, 2) (if the record of this id does not exist)
  - (row == None) **(False)** (F)
    - (1, 1) (if the record of this id exists)

export_record

```
def export_record(self, user_id, method='this month'):
        # method: may this_month, this_year, all
        # transit a csv file to the user
        success = False
        error_message = None
        link = None
        if not isinstance(user_id, int):
            error_message = 'invalid user_id parameter'
            return success, link, error_message

        if method != 'this month' and method != 'this year' and method !=
'all':
            error_message = 'invalid method parameter'
            return success, link, error_message

        # DB related
        conn = sqlite3.connect(self.db_name)
        cursor = conn.cursor()

        # check if user_id exists
        cursor.execute('SELECT * FROM user WHERE user_id = ?', (user_id,))
        row = cursor.fetchone()
        link = row
        if row == None:
            error_message = 'user_id does not exist'
            return success, link, error_message

        if method == 'this_month':
            cursor.execute(
                'SELECT * FROM record WHERE user_id = ? AND date >=
date("now", "start of month")', (user_id,))
        elif method == 'this_year':
            cursor.execute(
                'SELECT * FROM record WHERE user_id = ? AND date >=
date("now", "start of year")', (user_id,))
        elif method == 'all':
            cursor.execute(
```

```
                'SELECT * FROM record WHERE user_id = ?', (user_id,))

        rows = cursor.fetchall()
        conn.close()
        # write to csv file
        filepath = 'export_' + str(user_id) + '.csv'

        with open(filepath, 'w', newline='') as f:
            writer = csv.writer(f)
            writer.writerow(['record_id', 'user_id', 'date', 'item',
                             'cost', 'category', 'comment', 'create_date'])
            for row in rows:
                writer.writerow(row)
        # upload to file.io
        resp = Fileio.upload(filepath, expires="5m")
        success = resp['success']  # True if upload was successful
        link = resp['link']
        os.remove(filepath)
        if not success:
            error_message = 'upload failed'
        return success, link, error_message
```

有 5 predicates

- if not isinstance(user_id, int):
  - predicate coverage
    - T
      - ('123', 'this month')
    - F
      - (123, 'this month')
  - clause coverage (same as predicate coverage)
    - (not isinstance(user_id, int)) **(True)** (T)
      - ('123', 'this month')
    - (not isinstance(user_id, int)) **(False)** (F)
      - (123, 'this month')
  - correlative active clause coverage (same as predicate coverage)
    - (not isinstance(user_id, int)) **(True)** (T)
      - ('123', 'this month')
    - (not isinstance(user_id, int)) **(False)** (F)
      - (123, 'this month')
- if method != 'this month' and method != 'this year' and method != 'all':
  - predicate coverage
    - T
      - (1, 'this month')
    - F
      - (1, 'Hello')
  - clause coverage
    - (method != 'this month') **(True)** and (method != 'this year') **(True)** and (method != 'all') **(True)** (T)

            - (1, 'Hello')
        - (method != 'this month') **(True)** and (method != 'this year') **(True)** and (method != 'all') **(False)** (F)
            - (1, 'all')
        - (method != 'this month') **(True)** and (method != 'this year') **(False)** and (method != 'all') **(True)** (F)
            - (1, 'this year')
        - (method != 'this month') **(False)** and (method != 'this year') **(True)** and (method != 'all') **(True)** (F)
            - (1, 'this month')
    - correlative active clause coverage
        - (method != 'this month')為Major
            - (method != 'this month') **(True)** and (method != 'this year') **(True)** and (method != 'all') **(True)** (T)
                - (1, 'Hello')
            - (method != 'this month') **(False)** and (method != 'this year') **(True)** and (method != 'all') **(True)** (F)
                - (1, 'this month')
        - (method != 'this year')為Major
            - (method != 'this month') **(True)** and (method != 'this year') **(True)** and (method != 'all') **(True)** (T)
                - (1, 'Hello')
            - (method != 'this month') **(True)** and (method != 'this year') **(False)** and (method != 'all') **(True)** (F)
                - (1, 'this year')
        - (method != 'all')為Major
            - (method != 'this month') **(True)** and (method != 'this year') **(True)** and (method != 'all') **(True)** (T)
                - (1, 'Hello')
            - (method != 'this month') **(True)** and (method != 'this year') **(True)** and (method != 'all') **(False)** (F)
                - (1, 'all')
- if row == None:
    - predicate coverage
        - T
            - (2, 'this month') (if user_id does not exist)
        - F
            - (1, 'this month') (if user_id exists)
    - clause coverage(same as predicate coverage)
        - (row == None) **(True)** (T)
            - (2, 'this month') (if user_id does not exist)
        - (row == None) **(False)** (F)
            - (1, 'this month') (if user_id exists)
    - correlative active clause coverage(same as predicate coverage)
        - (row == None) **(True)** (T)
            - (2, 'this month') (if user_id does not exist)

            - (row == None) **(False)** (F)
                - (1, 'this month') (if user_id exists)
- if method == 'this_month':
    - predicate coverage
        - T
            - (1, 'this month')
        - F
            - (1, 'this year')
    - clause coverage
        - (method == 'this_month') **(True)** (T)
            - (1, 'this month')
        - (method == 'this_month') **(False)** (F)
            - (1, 'this year')
    - correlative active clause coverage
        - (method == 'this_month') **(True)** (T)
            - (1, 'this month')
        - (method == 'this_month') **(False)** (F)
            - (1, 'this year')
- if method == 'this_year':
    - predicate coverage
        - T
            - (1, 'this year')
        - F
            - (1, 'this month')
    - clause coverage
        - (method == 'this_year') **(True)** (T)
            - (1, 'this year')
        - (method == 'this_year') **(False)** (F)
            - (1, 'this month')
    - correlative active clause coverage
        - (method == 'this_year') **(True)** (T)
            - (1, 'this year')
        - (method == 'this_year') **(False)** (F)
            - (1, 'this month')
- if method == 'all':
    - predicate coverage
        - T
            - (1, 'all')
        - F
            - (1, 'this month')
    - clause coverage
        - (method == 'all') **(True)** (T)
            - (1, 'all')
        - (method == 'all') **(False)** (F)
            - (1, 'this month')
    - correlative active clause coverage

        - (method == 'all') **(True)** (T)
            - (1, 'all')
        - (method == 'all') **(False)** (F)
            - (1, 'this month')
- if not success:
    - predicate coverage
        - T
            - (1, 'this month') (if upload failed)
        - F
            - (1, 'this month') (if upload success)
    - clause coverage(same as predicate coverage)
        - (not success) **(True)** (T)
            - (1, 'this month') (if upload failed)
        - (not success) **(False)** (F)
            - (1, 'this month') (if upload success)
    - correlative active clause coverage(same as predicate coverage)
        - (not success) **(True)** (T)
            - (1, 'this month') (if upload failed)
        - (not success) **(False)** (F)
            - (1, 'this month') (if upload success)

`MessageParser.parse()`

Predicate 1

```
"""
Check if the message is string and not empty
"""
if not isinstance(user_message, str):
    error_message = 'wrong type'
    return success, param_list, error_message
elif user_message == '':
    error_message = 'empty message'
    return success, param_list, error_message
```

- A: `isinstance(user_message, str)`
- B: `user_message == ''`
- P1: `!A v B`
- PC
    - A: False, B: False, P1: True
        - parse(123)
    - A: True, B: False, P1: False
        - parse("!最近記帳 最近筆數 10")

- CC
    - A: True, B: True, P1: True
        - parse("")
    - A: False, B: False, P1: True
        - parse(123)
- CACC
    - **A**: False, B: False, P1: True
        - parse(123)
    - A: True, **B**: True, P1: True
        - parse("")

Predicate 2

```
"""
Split the message by space and check if the command is valid
"""
message_list = user_message.split(' ')
message_command = message_list[0]
if message_command not in self.command_map:
    error_message = 'invalid command'
    return success, param_list, error_message
```

- A: `message_command not in self.command_map`
- P2: `A`
- PC
    - A: True, P2: True
        - parse("123")
    - A: False, P2: False
        - parse("!最近記帳 最近筆數 10")
- CC
    - A: True, P2: True
        - parse("123")
    - A: False, P2: False
        - parse("!最近記帳 最近筆數 10")
- CACC
    - **A**: True, P2: True
        - parse("123")
    - **A**: False, P2: False
        - parse("!最近記帳 最近筆數 10")

Predicate 3

```
"""
Check if the pattern is valid
"""
command = self.command_map[message_command]
match = re.match(self.command_pattern[command], user_message)
if not match:
    error_message = 'invalid pattern'
```

- A: `match`
- P3: `!A`
- PC
    - A: True, P3: False
        - parse("!最近記帳 最近筆數 10")
    - A: False, P3: True
        - parse("!最近記帳 最近筆數 最近天數")
- CC
    - A: True, P3: False
        - parse("!最近記帳 最近筆數 10")
    - A: False, P3: True
        - parse("!最近記帳 最近筆數 最近天數")
- CACC
    - **A**: True, P3: False
        - parse("!最近記帳 最近筆數 10")
    - **A**: False, P3: True
        - parse("!最近記帳 最近筆數 最近天數")

Predicate 4

```
"""
Handle different commands
"""
if command == 'create_record':
    ...
    ...
elif command == 'show_recent_record':
    ...
    ...
elif command == 'search_record':
    ...
    ...
```

```
elif command == 'update_record':
    ...
    ...
elif command == 'delete_record':
    ...
    ...
elif command == 'export_record':
    ...
    ...
```

- A: `command == 'create_record'`
- B: `command == 'show_recent_record'`
- C: `command == 'search_record'`
- D: `command == 'update_record'`
- E: `command == 'delete_record'`
- F: `command == 'export_record'`
- P4: `A v B v C v D v E v F`
- PC
    - A: True, B: False, C: False, D: False, E: False, F: False, P4: True
        - parse("!記帳 20240101 breakfast 100 food delicious")
    - A: False, B: False, C: False, D: False, E: False, F: False, P4: False
        - will not happen due to *P2*
- CC
    - A: False, B: False, C: False, D: False, E: False, F: False, P4: False
        - will not happen due to *P2*
    - A: True, B: False, C: False, D: False, E: False, F: False, P4: True
        - parse("!記帳 20240101 breakfast 100 food delicious")
    - A: False, B: True, C: False, D: False, E: False, F: False, P4: True
        - parse("!最近記帳 最近筆數 10")
    - A: False, B: False, C: True, D: False, E: False, F: False, P4: True
        - parse("!查詢 20240101 20240102")
    - A: False, B: False, C: False, D: True, E: False, F: False, P4: True
        - parse("!修改記帳 12345678 日期 20240101 項目 lunch 金額 200 類別 food 備註 delicious")
    - A: False, B: False, C: False, D: False, E: True, F: False, P4: True
        - parse("!刪除記帳 12345678")
    - A: False, B: False, C: False, D: False, E: False, F: True, P4: True
        - parse("!匯出 本月")
- CACC

    - **A**: True, B: False, C: False, D: False, E: False, F: False, P4: True
        - parse("!記帳 20240101 breakfast 100 food delicious")
    - A: False, **B**: True, C: False, D: False, E: False, F: False, P4: True
        - parse("!最近記帳 最近筆數 10")
    - A: False, B: False, **C**: True, D: False, E: False, F: False, P4: True
        - parse("!查詢 20240101 20240102")
    - A: False, B: False, C: False, **D**: True, E: False, F: False, P4: True
        - parse("!修改記帳 12345678 日期 20240101 項目 lunch 金額 200 類別 food 備註 delicious")
    - A: False, B: False, C: False, D: False, **E**: True, F: False, P4: True
        - parse("!刪除記帳 12345678")
    - A: False, B: False, C: False, D: False, E: False, **F**: True, P4: True
        - parse("!匯出 本月")

Predicate 5

```
if method is None:
    ...
```

- A: `method is None`
- P5: `A`
- PC
    - A: True, P5: True
        - parse("!最近記帳 10")
    - A: False, P5: False
        - parse("!最近記帳 最近筆數 10")
- CC
    - A: True, P5: True
        - parse("!最近記帳 10")
    - A: False, P5: False
        - parse("!最近記帳 最近筆數 10")
- CACC
    - **A**: True, P5: True
        - parse("!最近記帳 10")
    - **A**: False, P5: False
        - parse("!最近記帳 最近筆數 10")

Predicate 6

```
if value is None:
    ...
```

- A: `value is None`
- P6: `A`
- PC
    - A: True, P6: True
        - parse("!最近記帳 最近筆數")
    - A: False, P6: False
        - parse("!最近記帳 最近筆數 10")
- CC
    - A: True, P6: True
        - parse("!最近記帳 最近筆數")
    - A: False, P6: False
        - parse("!最近記帳 最近筆數 10")
- CACC
    - **A**: True, P6: True
        - parse("!最近記帳 最近筆數")
    - **A**: False, P6: False
        - parse("!最近記帳 最近筆數 10")

Predicate 7

```
if date_to is None:
    param_list = [command, date_from]
else:
    param_list = [command, date_from, date_to]
```

- A: `date_to is None`
- P7: `A`
- PC
    - A: True, P7: True
        - parse("!查詢 20240101")
    - A: False, P7: False
        - parse("!查詢 20240101 20240102")
- CC
    - A: True, P7: True

        - parse("!查詢 20240101")
    - A: False, P7: False
        - parse("!查詢 20240101 20240102")

- CACC
    - **A**: True, P7: True
        - parse("!查詢 20240101")
    - **A**: False, P7: False
        - parse("!查詢 20240101 20240102")

Predicate 8

```
if key == '日期':
    date = value
elif key == '項目':
    item = value
elif key == '金額':
    cost = int(value)
elif key == '類別':
    category = value
elif key == '備註':
    comment = value
```

- A: `key == '日期'`
- B: `key == '項目'`
- C: `key == '金額'`
- D: `key == '類別'`
- E: `key == '備註'`
- P8: `A v B v C v D v E`
- PC
    - A:True, B:False, C:False, D:False, E:False, P8:True
        - parse("!修改記帳 12345678 日期 20240101 項目 lunch 金額 200 類別 food 備註 delicious")
    - A:False, B:False, C:False, D:False, E:False, P8:False
        - will not happen due to *P3*
- CC
    - A:True, B:False, C:False, D:False, E:False, P8:True
    - A:False, B:True, C:False, D:False, E:False, P8:True
    - A:False, B:False, C:True, D:False, E:False, P8:True
    - A:False, B:False, C:False, D:True, E:False, P8:True

    - A:False, B:False, C:False, D:False, E:True, P8:True
    - parse("!修改記帳 12345678 日期 20240101 項目 lunch 金額 200 類別 food 備註 delicious")

- CACC

    - **A**:True, B:False, C:False, D:False, E:False, P8:True
    - A: False, **B**:True, C:False, D:False, E:False, P8:True
    - A: False, B:False, **C**:True, D:False, E:False, P8:True
    - A: False, B:False, C:False, **D**:True, E:False, P8:True
    - A: False, B:False, C:False, D:False, **E**:True, P8:True
    - parse("!修改記帳 12345678 日期 20240101 項目 lunch 金額 200 類別 food 備註 delicious")

`main.handle_message(event)`

Predicate 1

```
line_success, line_user, line_error_message = my_line.create_line_user(
    event.source.user_id)
if not line_success and line_error_message == 'line_id already exists':
    line_success, line_user, line_error_message =
my_line.get_user_by_line_id(
        event.source.user_id)
```

- A: `line_success == True`
- B: `line_error_message == 'line_id already exists'`
- P1: `!A ^ B`
- PC

    - A: False, B: True, P1: True
        - event.source.user_id = "user_1", event.source.user_id = "user_1"
    - A: True, B: False, P1: False
        - event.source.user_id = "user_1", event.source.user_id = "user_2"

- CC

    - A: True, B: False, P1: False
        - event.source.user_id = "user_1", event.source.user_id = "user_2"
    - A: False, B: True, P1: True
        - event.source.user_id = "user_1", event.source.user_id = "user_1"

- CACC

    - **A**: True, B: True, P1: False
        - will not happen due to `my_line.create_line_user()`
    - **A**: False, B: True, P1: True
        - event.source.user_id = "user_1", event.source.user_id = "user_1"
    - A: False, **B**: False, P1: False

        - event.source.user_id = 123

Predicate 2

```
if not line_success:
    reply_message = "Create Line User error: " + line_error_message
    reply_message_request = ReplyMessageRequest(
        reply_token=event.reply_token,
        messages=[TextMessage(text=reply_message)]
    )
    line_bot_api.reply_message_with_http_info(reply_message_request)
    return reply_message_request
```

- A: `line_success == True`
- P2: `!A`
- PC
    - A: False, P2: True
        - event.source.user_id = 123
    - A: True, P2: False
        - event.source.user_id = "user_1"
- CC
    - A: True, P2: False
        - event.source.user_id = "user_1"
    - A: False, P2: True
        - event.source.user_id = 123
- CACC
    - **A**: True, P2: False
        - event.source.user_id = "user_1"
    - **A**: False, P2: True
        - event.source.user_id = 123

Predicate 3

```
parser_success, parser_param_list, parser_error_message = my_parser.parse(
    user_message)
if not parser_success:
    reply_message = "Parse error: " + parser_error_message
    reply_message_request = ReplyMessageRequest(
        reply_token=event.reply_token,
        messages=[TextMessage(text=reply_message)]
    )
    line_bot_api.reply_message_with_http_info(reply_message_request)
```

```
    return reply_message_request
else:
    my_line = line.lineFunction(db_name)
    my_accounting = accounting.accountingFunction(db_name)

    command = parser_param_list[0]
```

- A: `parser_success == True`
- P3: `!A`
- PC
    - A: True, P3: False
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - A: False, P3: True
        - event.message.text = 123
- CC
    - A: True, P3: False
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - A: False, P3: True
        - event.message.text = 123
- CACC
    - **A**: True, P3: False
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - **A**: False, P3: True
        - event.message.text = 123

Predicate 4

```
my_line = line.lineFunction(db_name)
my_accounting = accounting.accountingFunction(db_name)

command = parser_param_list[0]

"""
Turn the command string into callable function
"""
if command == 'create_record':
    ...
    ...

elif command == 'show_recent_record':
    ...
    ...

elif command == 'search_record':
```

```
    ...
    ...

elif command == 'update_record':
    ...
    ...

elif command == 'delete_record':
    ...
    ...

elif command == 'export_record':
    ...
    ...

else:
    reply_message = "Invalid command"
```

- A: `command == 'create_record'`
- B: `command == 'show_recent_record'`
- C: `command == 'search_record'`
- D: `command == 'update_record'`
- E: `command == 'delete_record'`
- F: `command == 'export_record'`
- P4: `A v B v C v D v E v F`
- PC
    - A: True, B: False, C: False, D: False, E: False, F: False, P4: True
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - A: False, B: False, C: False, D: False, E: False, F: False, P4: False
        - will not happen due to *MessageParser.parse()*
- CC
    - A: False, B: False, C: False, D: False, E: False, F: False, P4: False
        - will not happen due to *MessageParser.parse()*
    - A: True, B: False, C: False, D: False, E: False, F: False, P4: True
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - A: False, B: True, C: False, D: False, E: False, F: False, P4: True
        - event.message.text = "!最近記帳 最近筆數 10"
    - A: False, B: False, C: True, D: False, E: False, F: False, P4: True
        - event.message.text = "!查詢 20240101 20240102"
    - A: False, B: False, C: False, D: True, E: False, F: False, P4: True
        - event.message.text = "!修改記帳 12345678 日期 20240101 項目 lunch 金額 200 類別 food 備註 delicious"

    - A: False, B: False, C: False, D: False, E: True, F: False, P4: True
        - event.message.text = "!刪除記帳 12345678"
    - A: False, B: False, C: False, D: False, E: False, F: True, P4: True
        - event.message.text = "!匯出 本月"

- CACC

    - **A**: True, B: False, C: False, D: False, E: False, F: False, P4: True
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - A: False, **B**: True, C: False, D: False, E: False, F: False, P4: True
        - event.message.text = "!最近記帳 最近筆數 10"
    - A: False, B: False, **C**: True, D: False, E: False, F: False, P4: True
        - event.message.text = "!查詢 20240101 20240102"
    - A: False, B: False, C: False, **D**: True, E: False, F: False, P4: True
        - event.message.text = "!修改記帳 12345678 日期 20240101 項目 lunch 金額 200 類別 food 備註 delicious"
    - A: False, B: False, C: False, D: False, **E**: True, F: False, P4: True
        - event.message.text = "!刪除記帳 12345678"
    - A: False, B: False, C: False, D: False, E: False, **F**: True, P4: True
        - event.message.text = "!匯出 本月"

Predicate 5

```
if accounting_success:
    reply_message = ...
else:
    reply_message = ...
```

- A: `accounting_success == True`
- P5: `A`
- PC

    - A: True, P5: True
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - A: False, P5: False
        - event.message.text = "!刪除記帳 12345678" when `record_id 12345678` is not exist in db

- CC

    - A: True, P5: True
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - A: False, P5: False
        - event.message.text = "!刪除記帳 12345678" when `record_id 12345678` is not exist in db

- CACC
    - **A**: True, P5: True
        - event.message.text = "!記帳 20240520 Lunch 10 Food Comment"
    - **A**: False, P5: False
        - event.message.text = "!刪除記帳 12345678" when `record_id 12345678` is not exist in db

Predicate 6

```
if len(parser_param_list) == 1:
    ...
elif len(parser_param_list) == 2:
    ...
elif len(parser_param_list) == 3:
    ...
```

- A: `len(parser_param_list) == 1`
- B: `len(parser_param_list) == 2`
- C: `len(parser_param_list) == 3`
- P6: `A v B v C`
- PC
    - A: True, B: False, C: False, P6: True
        - event.message.text = "!最近記帳"
    - A: False, B: False, C: False, P6: False
        - will not happen due to *MessageParser.parse()*
- CC
    - A: True, B: False, C: False, P6: True
        - event.message.text = "!最近記帳"
    - A: False, B: True, C: False, P6: True
        - event.message.text = "!最近記帳 10"
    - A: False, B: False, C: True, P6: True
        - event.message.text = "!最近記帳 最近筆數 10"
- CACC
    - **A**: True, B: False, C: False, P6: True
        - event.message.text = "!最近記帳"
    - A: False, **B**: True, C: False, P6: True
        - event.message.text = "!最近記帳 10"
    - A: False, B: False, **C**: True, P6: True
        - event.message.text = "!最近記帳 最近筆數 10"

Predicate 7

```
if isinstance(parser_param_list[1], int):
    accounting_success, accounting_records, accounting_error_message = \
        my_accounting.show_recent_record(
            user_id=user_id, num=parser_param_list[1])
elif isinstance(parser_param_list[1], str):
    accounting_success, accounting_records, accounting_error_message = \
        my_accounting.show_recent_record(
            user_id=user_id, type=parser_param_list[1])
```

- A: `isinstance(parser_param_list[1], int)`
- B: `isinstance(parser_param_list[1], str)`
- P7: `A v B`
- PC
    - A: True, B: False, P7: True
        - event.message.text = "!最近記帳 10"
    - A: False, B: False, P7: False
        - will not happen due to *MessageParser.parse()*
- CC
    - A: True, B: False, P7: True
        - event.message.text = "!最近記帳 10"
    - A: False, B: True, P7: True
        - event.message.text = "!最近記帳 最近筆數"
- CACC
    - **A**: True, B: False, P7: True
        - event.message.text = "!最近記帳 10"
    - A: False, **B**: True, P7: True
        - event.message.text = "!最近記帳 最近筆數"

Predicate 8

```
if parser_param_list[2] == 'num':
    accounting_success, accounting_records, accounting_error_message = \
        my_accounting.show_recent_record(
            user_id=user_id, num=parser_param_list[1],
type=parser_param_list[2])
elif parser_param_list[2] == 'day':
    accounting_success, accounting_records, accounting_error_message = \
        my_accounting.show_recent_record(
```

```
            user_id=user_id, days=parser_param_list[1],
type=parser_param_list[2])
```

- A: `parser_param_list[2] == 'num'`
- B: `parser_param_list[2] == 'day'`
- P8: `A v B`
- PC
    - A: True, B: False, P8: True
        - event.message.text = "!最近記帳 最近筆數 10"
    - A: False, B: False, P8: False
        - will not happen due to *MessageParser.parse()*
- CC
    - A: True, B: False, P8: True
        - event.message.text = "!最近記帳 最近筆數 10"
    - A: False, B: True, P8: True
        - event.message.text = "!最近記帳 最近天數 10"
- CACC
    - **A**: True, B: False, P8: True
        - event.message.text = "!最近記帳 最近筆數 10"
    - A: False, **B**: True, P8: True
        - event.message.text = "!最近記帳 最近天數 10"

Predicate 9

```
if len(parser_param_list) == 2:
    accounting_success, accounting_records, accounting_error_message = \
        my_accounting.search_record(
            user_id=user_id, date_from=parser_param_list[1])
elif len(parser_param_list) == 3:
    accounting_success, accounting_records, accounting_error_message = \
        my_accounting.search_record(
            user_id=user_id, date_from=parser_param_list[1],
date_to=parser_param_list[2])
```

- A: `len(parser_param_list) == 2`
- B: `len(parser_param_list) == 3`
- P9: `A v B`
- PC

    - A: True, B: False, P9: True
        - event.message.text = "!查詢 20240101"
    - A: False, B: False, P9: False
        - will not happen due to *MessageParser.parse()*

- CC

    - A: True, B: False, P9: True
        - event.message.text = "!查詢 20240101"
    - A: False, B: True, P9: True
        - event.message.text = "!查詢 20240101 20240102"

- CACC

    - **A**: True, B: False, P9: True
        - event.message.text = "!查詢 20240101"
    - A: False, **B**: True, P9: True
        - event.message.text = "!查詢 20240101 20240102"

Predicate 10

```
if len(parser_param_list) == 1:
    accounting_success, accounting_link, accounting_error_message = \
        my_accounting.export_record(user_id=user_id)
elif len(parser_param_list) == 2:
    accounting_success, accounting_link, accounting_error_message = \
        my_accounting.export_record(
            user_id=user_id, method=parser_param_list[1])
```

- A: `len(parser_param_list) == 1`
- B: `len(parser_param_list) == 2`
- P10: `A v B`
- PC

    - A: True, B: False, P10: True
        - event.message.text = "!匯出"
    - A: False, B: False, P10: False
        - will not happen due to *MessageParser.parse()*

- CC

    - A: True, B: False, P10: True
        - event.message.text = "!匯出"
    - A: False, B: True, P10: True
        - event.message.text = "!匯出 本月"

- CACC

- **A**: True, B: False, P10: True
  - event.message.text = "!匯出"
- A: False, **B**: True, P10: True
  - event.message.text = "!匯出 本月"