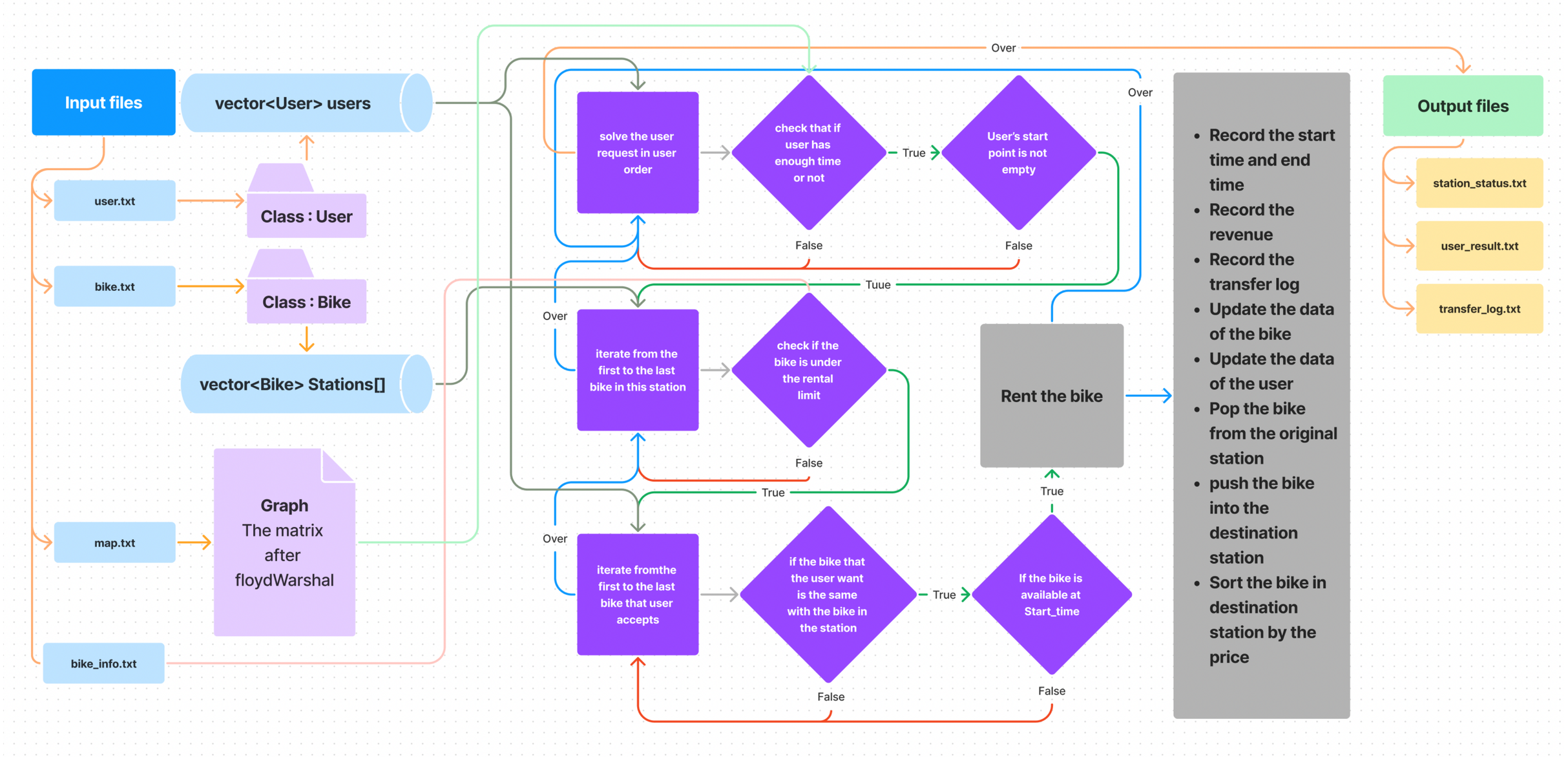
DS Final project report

109000109 謝秉軒

The structure of the code :



Input files :
Take bike_info.txt
for example

```
char path_bike_info[100] = "testcases/";
strcat(path_bike_info, selectedCase.c_str());
strcat(path_bike_info, "/bike_info.txt");
ifstream ifs_bike_info(path_bike_info, ios::in);
if (!ifs_bike_info.is_open())
{
cout << "Failed to open file.\n";
}
string bike_type_template;
float bike_ini_price_template;
while (ifs_bike_info >> bike_type_template >> bike_ini_price_template)
{
discount_price = stof(bike_type_template);
rental_limit = bike_ini_price_template;
break;
}
while (ifs_bike_info >> bike_type_template >> bike_ini_price_template)
{
Bike_info bike_info(bike_type_template, bike_ini_price_template);
bike_infos.push_back(bike_info);
}
ifs_bike_info.close();
```

合成路徑中的變數

將整理好的資料 Push 進 vector 中

Build a graph :

```
map.addEdge(S1, S2, distance);
```

新增路徑

```
floydWarshal(map.arr());
```

使用 floydWarshal 將整個 Adjacency Matrix 變成 all-shortest path

```
void floydWarshal(int **cost)
{
for (int k = 0; k < V; k++)
{
for (int i = 0; i < V; i++)
for (int j = 0; j < V; j++)
if (cost[i][k] + cost[k][j] < cost[i][j])
cost[i][j] = cost[i][k] + cost[k][j];
}
}
```

```
The matrix after floydWarshal:
    0  138  109   42   74
  138    0   68   96   93
  109   68    0  151  155
   42   96  151    0  116
   74   93  155  116    0
```

The main rental logic of this system :

```
for (auto user = users.begin(); user != users.end(); user++)
{ // solve the user request in user order
select = false;
if (user→End_Time - user→Start_Time > map.get_value(user→Start_Point, user→End_Point))
{ // check that if user has enough time or not
if (!Stations[user→Start_Point].empty())
{ // if the station where user start is not empty
for (auto bike_in_the_station = Stations[user→Start_Point].begin(); bike_in_the_station !=
Stations[user→Start_Point].end(); bike_in_the_station++)
{ // iterate from the first to the last bike in this station
if (bike_in_the_station→Rental_count < rental_limit)
{ // check if the bike is under the rental limit
for (auto user_accept_bike_type = user→Accept_Bike_Types.begin(); user_accept_bike_type !=
user→Accept_Bike_Types.end(); user_accept_bike_type++)
{ // iterate from the first to the last bike that user accepts
if (user_accept_bike_type→compare(bike_in_the_station→Type) == 0)
{ // if the bike that the user want is the same with the bike in the station or not
if (bike_in_the_station→TimeIndex < user→Start_Time)
{ // III. If no bike is available at Start_time, the rental request should be rejected with no charge.
(we assume users do not wait for bikes.)
```

Output files :
Take station_status.txt
for example

```
char path_station_status[100] = "result/";
strcat(path_station_status, selectedCase.c_str());
strcat(path_station_status, "/station_status.txt");
ofstream out_station_status(path_station_status);
for (int i = 0; i < V; i++)
{
if (!Stations[i].empty())
{
sort(Stations[i].begin(), Stations[i].end(), [](const Bike &lhs, const Bike &rhs)
{ return lhs.ID < rhs.ID; });
for (auto it = Stations[i].begin(); it != Stations[i].end(); it++)
{
out_station_status << "S" << it→Station << " " << it→ID << " " << it→Type << " "
<< it→Rental_price << " " << it→Rental_count << endl;
}
}
}
out_station_status.close(); // 關閉檔案
```

合成路徑中的變數

將資料一筆一筆輸出

心得 ：

一開始看到這個作業的時候覺得很難，感覺有很多input 和 output 要處理，邏輯上又很複雜。所以不知要從何下手。接著在處理 Input 的時候花了比較多的時間，但把所有資料都整理好，裝入對應的容器後，思路就變得比較清晰了。我開始畫邏輯判斷上的流程圖，並把他們實作出來，最後在幾次的糾錯後就順利完成此項作業了！
這次的作業模擬真實情境的設定我很喜歡，不僅讓我收穫良多，也讓我很有成就感！