User Event Aggregation CLI

Context

You are tasked with developing a command-line utility for a social media app's backend service. This utility will aggregate user activity events to generate daily summary reports. These reports should include the number of posts a user has made, the number of likes they've received, etc., all on a per-day basis. Additionally, the utility should be capable of updating these summary reports in real-time as new events are added to the dataset.

You may use any language of your choice. Please make sure to include a README.md file in case any special instructions are needed to run. Also, include details about test cases that you have used to test your code.

Task

Part 1

Write a command-line utility that reads a list of user events from a JSON file, aggregates the events, and writes the daily summary reports back to another JSON file.

Input File Format:

```
[
    {"userId": 1, "eventType": "post", "timestamp": 1672444800},
    {"userId": 1, "eventType": "likeReceived", "timestamp": 1672444801},
    ...
]
```

Example Input and Aggregation:
If input.json contains:

```
[
    {"userId": 1, "eventType": "post", "timestamp": 1672444800},
    {"userId": 1, "eventType": "likeReceived", "timestamp": 1672444801},
    {"userId": 1, "eventType": "likeReceived", "timestamp": 1672444802},
    {"userId": 1, "eventType": "post", "timestamp": 1672531200},
    {"userId": 2, "eventType": "comment", "timestamp": 1672531201},
    {"userId": 2, "eventType": "post", "timestamp": 1672531202}
]
```

The output should be:

```
[
    {"userId": 1, "date": "2023-01-01", "post": 1, "likeReceived": 2},
```

```
    {"userId": 1, "date": "2023-01-02", "post": 1},
    {"userId": 2, "date": "2023-01-02", "comment": 1, "post": 1}
]
```

You should be able to run the command line utility something like this:

```
$ ./aggregate_events -i input.json -o output.json
```

Part 2

Extend your utility to support real-time aggregation. Each time a new event is added to the input JSON file, the utility should update the corresponding daily summary report without reprocessing all the previous events.

For example:

After the initial execution, if the input.json file is updated with a new event, running the utility again should update output.json without losing the previously aggregated information.

```
$ ./aggregate_events -i input.json -o output.json --update
```