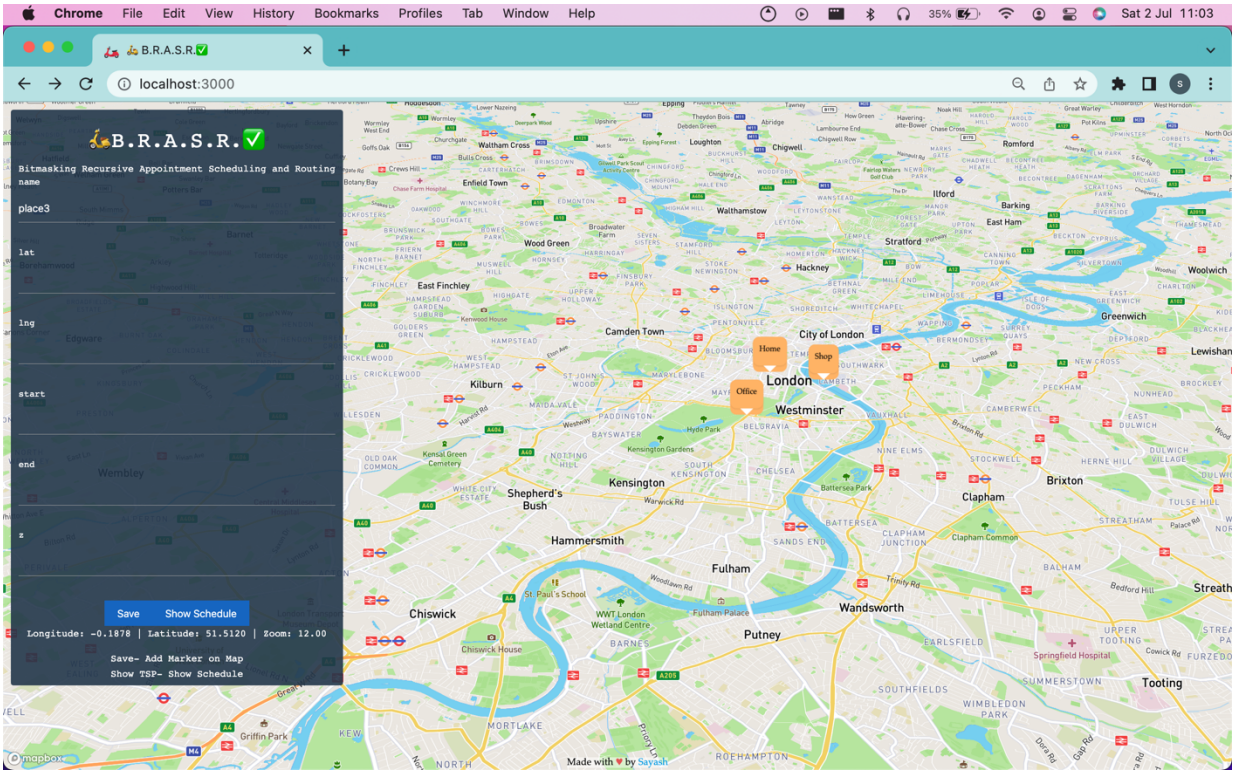
TEAM SAYASH- SAYASH RAAJ, IIT MADRAS 3RD YEAR

🛵B.R.A.S.R.✅

Bitmasking Recursive Appointment Scheduling and Routing



DOCUMENTATION

- Deliverables
- Working
- Outlook, Teams, Google Calendar, etc. all calendars integration details
- Example JSON Request and JSON Response
- API model
    - API scheduling
    - API Calendar (Outlook, Google Calendar, etc. all calendars) integration
- UI integration
    - Maps integration
    - UI Calendar (Outlook, Google Calendar, etc. all calendars) integration
- Pseudocode for 🛵B.R.A.S.R.✅ algorithm

[redacted] **a solution for Appointment Scheduling and Routing**

Appointment scheduling and routing solution helps in reducing manual, time-taking processes such as maintaining data. security, etc.

This solution helps in tracking locations and addresses. It has been integrated successfully with online maps to improve efficiency, which is not possible in a manual offline process.

- **Minimum task DONE ✅**
  - Build a restful API to schedule and confirm appointments based on the description mentioned DONE ✅
  - An interface to clearly see the agenda/calendar for the day with the drive time between appointments DONE ✅
  - Integration with Outlook/Teams calendar DONE ✅
- **Intermediate task DONE ✅**
  - Create a smart solution rearranging the appointments to reduce drive time DONE✅
- **Advanced task DONE ✅**
  - Integration with Teams or Outlook to reflect the agenda DONE ✅
- **EXTRA CREATIVE task DONE ✅**
  - Can be integrated with ANY calendar (Outlook, Teams, Google, Yahoo, Apple, etc)
  - Average appointment time can be customised for ALL locations individually
  - Stateless deAuth approach for null security, privacy-first approach

Tech Stack

- REST API
- Frontend Framework
- Custom-built scheduling and routing algorithm
  - Uses a 32-bit Bitmasking Recursive approach using an NP-hard solution framework
  - Recursively finds the shortest path w.r.t time with given constraints
  - Uses graph-algorithmic approach
  - Uses bitmasking to store mid-execution state of algorithm

Outlook, Teams, and all calendars integration

ICS-based approach- Has business-centric efficient approach

An out of the box solution, using ICS files has many advantages in this integration activity

Stateless management of activities. Auth integration not required, anonymous user calendar generation

Secure with null privacy concerns as opposed to sharing auth details/token to integrate calendar

Shareable calendar, can be shared inside and outside office team, including vendors, friends, clients, potential clients, etc

ICS file is generated and downloaded **automatically** in the **UI based approach**

Using **GET** request on server url http://localhost:5000/calendar for **API approach**

🦾B.R.A.S.R.✅ How it works-

Is the **best and most efficient way** of solving the problem.

Using custom-built modifications on the age-old original Travelling Salesman Problem (TSP), which is considerably the most famous NP-hard problem, we have built the 🦾B.R.A.S.R.✅ algorithm

Solution Flow

User enters locations to visit, entering information in a simple form- clicks Save button

Form response is appended to a global array consisting of locations added so far

User clicks Show Schedule button, global array is sent as a JSON array of objects as body request to the Cross-Platform server


```
[
    {
        "id": "place0",
        "coords": {
        "lat": 51.51198245486377,
        "lng": -0.1278277598563
        },
        "start": 0,
        "end": 1000,
        "z": 0
    },
    {
        "id": "place1",
        "coords": {
        "lat": 51.503120589264064,
        "lng": -0.15282095066100
      },
        "start": 0,
        "end": 1000,
        "z": 0
    },
    {
        "id": "place2",
        "coords": {
        "lat": 51.503341807681544,
        "lng": -0.11952824596429
      },
        "start": 0,
        "end": 1000,
        "z": 0
    }
]
```


actual request.body sent to server

The Cross-Platform Modular Server parses the JSON array in the input parsing node

Server send an internal request to perform the calculative task using the 🦾B.R.A.S.R.✅ algorithm

- Calculative function sends an external POST request to Time Travel Matrix API, obtaining information on least distance and time between each node
- Parses external API response into an n x n matrix, n = number of nodes
- Constructs a dense network as opposed to a sparse network of nodes
  - Number of edges = nC2 = n(n-1)/2 nodes
- 🦾B.R.A.S.R.✅ algorithm runs on constructed dense network, returns object

Server reformats output as a union of request.body parameters and 🦾B.R.A.S.R.✅ output

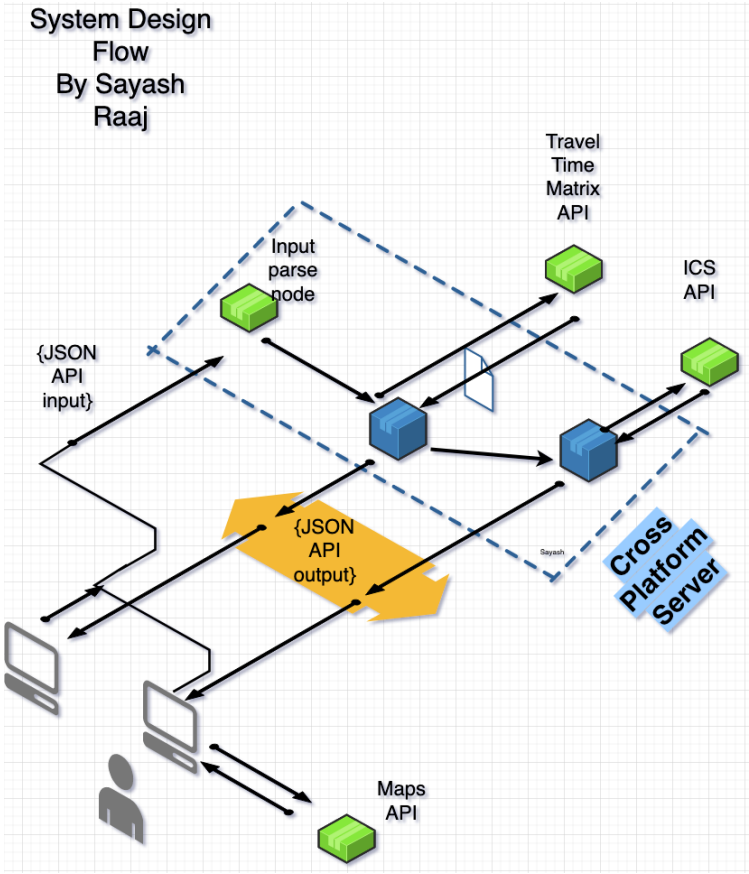
JSON output

Server sends an internal request to ICS generating function

- External POST request sent, response obtained with necessary ICS configuration
- ICS calendar file generated

Server reformats output for ICS file obtained

ICS files can be imported, shared via email etc, and integrated with any modern calendar, including but not limited to Outlook, Teams, Google Calendar, Apple Calendar, Yahoo Calendar, etc.



API Scheduling JSON Response

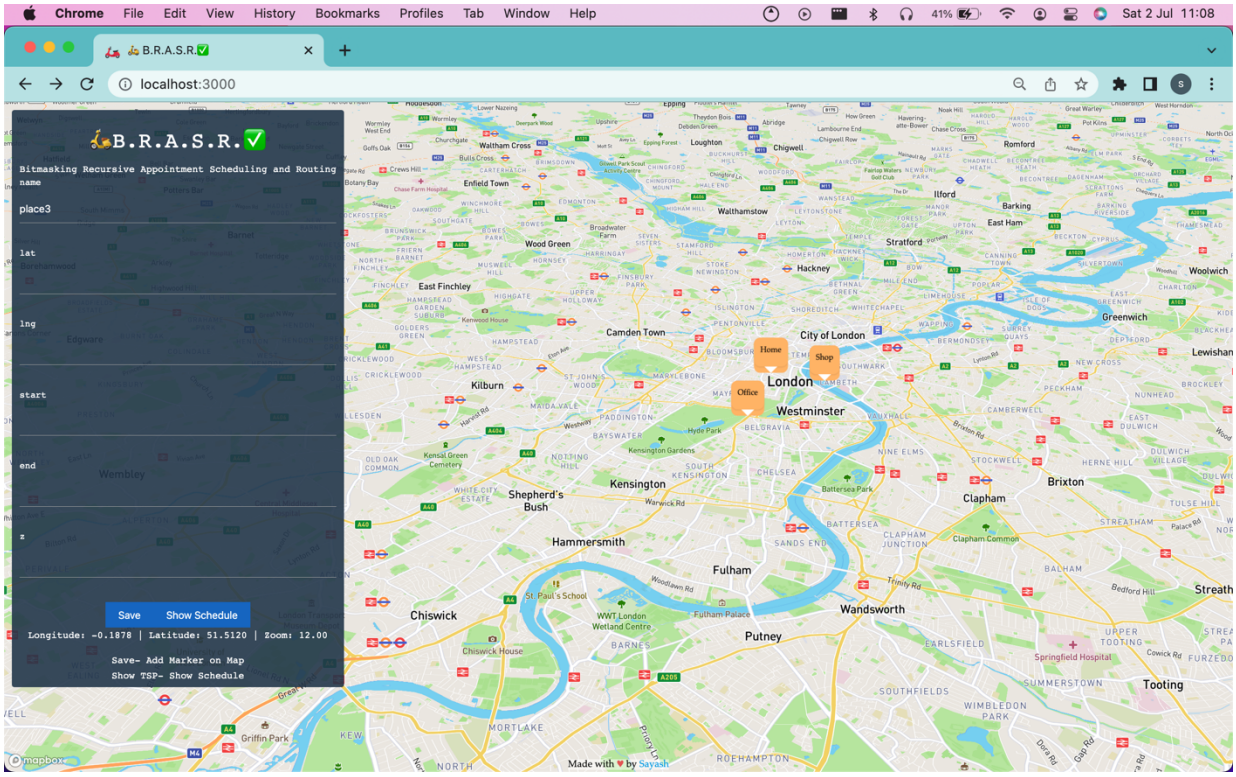

```
{
    "ans": 3108,
    "vec": [
        {
            "day": 0,
            "name": "place0",
            "start": "2022-07-02T13:18:09.862Z",
            "end": "2022-07-02T13:18:09.862Z"
        },
        {
            "day": 0,
            "name": "place1",
            "start": "2022-07-02T13:32:50.862Z",
            "end": "2022-07-02T13:32:50.862Z"
        },
        {
            "day": 1,
            "name": "place0",
            "start": "2022-07-03T13:18:09.862Z",
            "end": "2022-07-03T13:18:09.862Z"
        },
        {
            "day": 1,
            "name": "place2",
            "start": "2022-07-03T13:29:22.862Z",
            "end": "2022-07-03T13:29:22.862Z"
        }
    ]
}
```

API Calendar JSON Response

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//sebbo.net//ical-generator//EN
BEGIN:VEVENT
UID:d6b5eaf0-fb34-4994-8260-8486a2076c4a
SEQUENCE:0
DTSTAMP:20220702T172419Z
DTSTART:20220702T163217Z
DTEND:20220702T163217Z
SUMMARY:
LOCATION:Home
END:VEVENT
BEGIN:VEVENT
UID:68a6dfd7-6201-40d1-8262-5c0c2fc94445
SEQUENCE:0
DTSTAMP:20220702T172419Z
DTSTART:20220702T164658Z
DTEND:20220702T164658Z
SUMMARY:
LOCATION:Office
END:VEVENT
BEGIN:VEVENT
UID:fbcaafd2-4692-4d90-83ef-aabf99b909f1
SEQUENCE:0
DTSTAMP:20220702T172419Z
DTSTART:20220703T163217Z
DTEND:20220703T163217Z
SUMMARY:
LOCATION:Home
END:VEVENT
BEGIN:VEVENT
UID:26228bb1-4b4a-4422-ae1d-a2e300f01412
SEQUENCE:0
DTSTAMP:20220702T172419Z
DTSTART:20220703T164330Z
DTEND:20220703T164330Z
SUMMARY:
LOCATION:Shop
END:VEVENT
BEGIN:VEVENT
UID:fed62d25-7f47-46f2-bd4c-4a70b1ae2819
SEQUENCE:0
DTSTAMP:20220702T172419Z
DTSTART:20220704T163217Z
DTEND:20220704T163217Z
SUMMARY:
LOCATION:Home
END:VEVENT
```

```
BEGIN:VEVENT
UID:a54a8a43-09fb-4a77-bc39-775756c5909b
SEQUENCE:0
DTSTAMP:20220702T172419Z
DTSTART:20220704T164109Z
DTEND:20220704T164249Z
SUMMARY:
LOCATION:place3
END:VEVENT
END:VCALENDAR
```

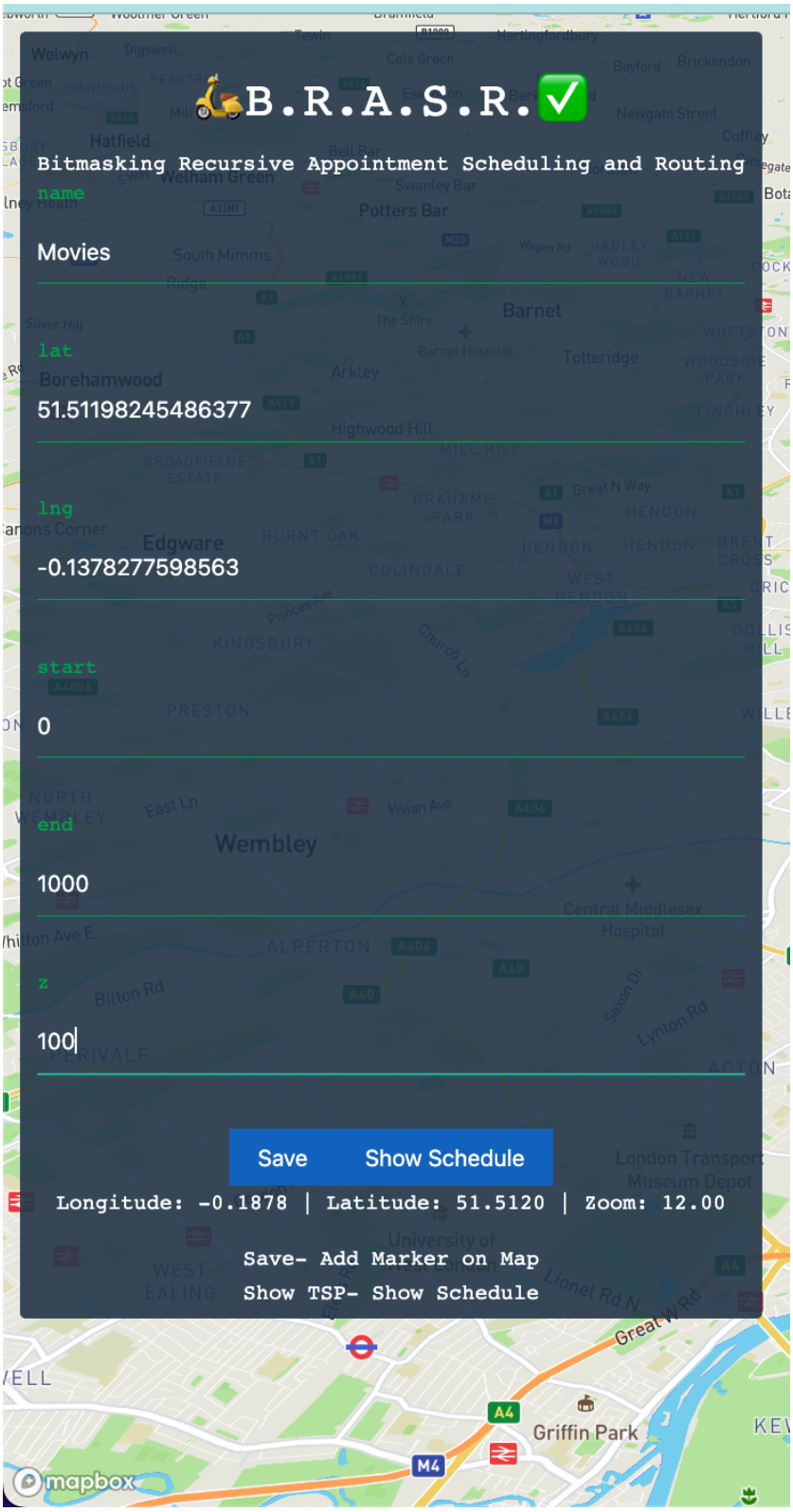
Maps integration



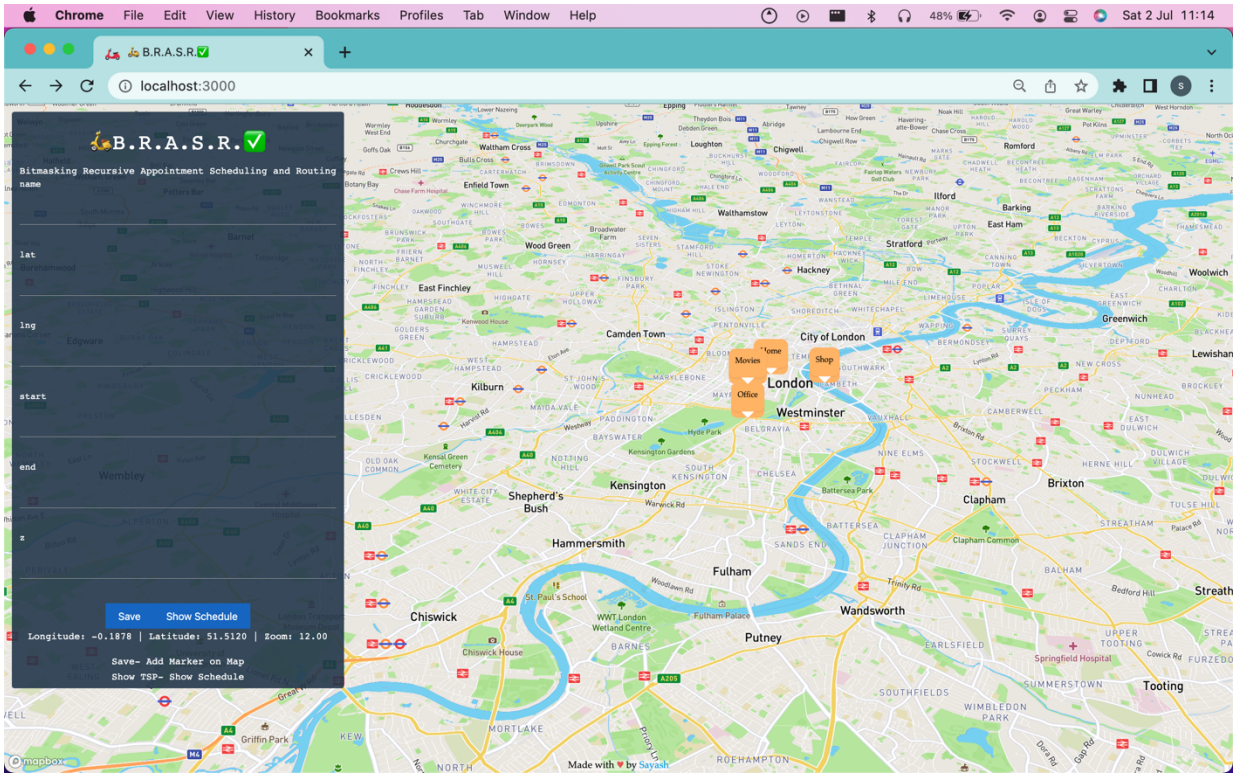
Maps integration displays all locations input by user

Home node is genesis node, first location input by user

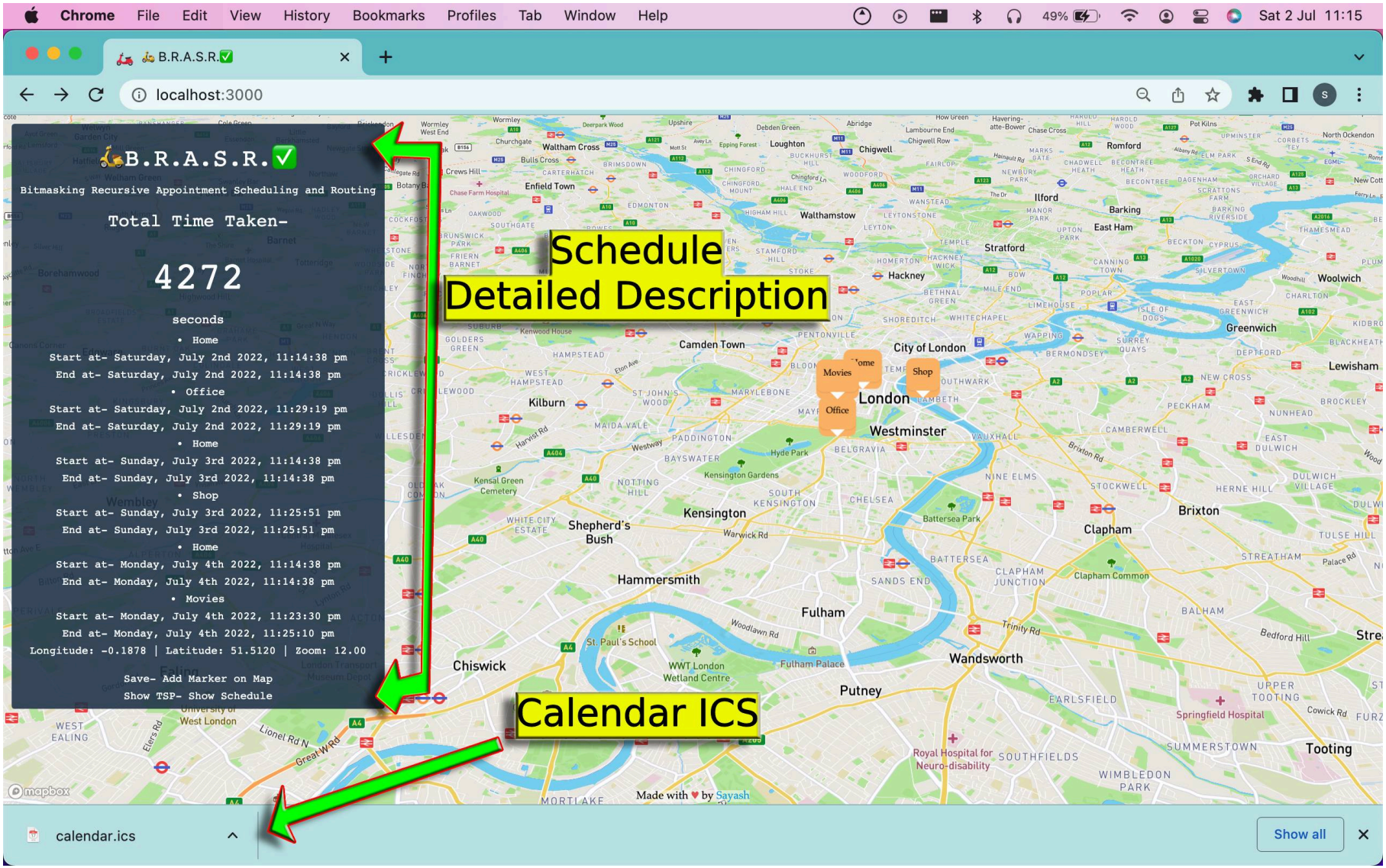
No limit on number of nodes or locations



Input form- appending locations input by user- node creation at instant



New node created and displayed, dynamic real-time Maps integration


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Sat 2 Jul 11:15
B.R.A.S.R.✅
localhost:3000
🛵B.R.A.S.R.✅
Bitmasking Recursive Appointment Scheduling and Routing
Total Time Taken-
4272
seconds
• Home
Start at- Saturday, July 2nd 2022, 11:14:38 pm
End at- Saturday, July 2nd 2022, 11:14:38 pm
• Office
Start at- Saturday, July 2nd 2022, 11:29:19 pm
End at- Saturday, July 2nd 2022, 11:29:19 pm
• Home
Start at- Sunday, July 3rd 2022, 11:14:38 pm
End at- Sunday, July 3rd 2022, 11:14:38 pm
• Shop
Start at- Sunday, July 3rd 2022, 11:25:51 pm
End at- Sunday, July 3rd 2022, 11:25:51 pm
• Home
Start at- Monday, July 4th 2022, 11:14:38 pm
End at- Monday, July 4th 2022, 11:14:38 pm
• Movies
Start at- Monday, July 4th 2022, 11:23:30 pm
End at- Monday, July 4th 2022, 11:25:10 pm
Longitude: -0.1878 | Latitude: 51.5120 | Zoom: 12.00
Save- Add Marker on Map
Show TSP- Show Schedule
Schedule
Detailed Description
Calendar ICS
Movies
Office
Shop
London
Made with ♥ by Sayash
calendar.ics
Show all

Pseudocode for 🛵B.R.A.S.R.✅ algorithm

```
vis_mask = 32-bit mask state storage of all visited nodes
skipped_mask = 32-bit mask state storage of all skipped nodes
pos = node number mapped to location
ssf = current time so far
total = total time so far
day = current day

int BRASR(int vis_mask, int skipped_mask, int pos, int ssf, int total, int day)
    if(vis_mask == all nodes visited)
        if(skipped_mask  == no nodes were skipped) // termination condition
            return total time +(time from pos -> home)
        else
            vis_mask  = all nodes visited and not skipped
            mark home node as visited
            return BRASR(…, pos=0, ssf=0, total time)
    for( i=0; i<number of nodes; i++)
        if (node i is not visited)
            mark node i as visited
            if(current time + avg time for aptmnt. z_i > end time for aptmnt.)
                cant be completed today, mark as skipped for today
            else
                update current time to expected time at end of appointment
                ans = min(ans, BRASR(…, ssf=exp. end time))
    if(all nodes visited in loop)
        if(no nodes skipped)
            return ans
        vis_mask  = all nodes visited and not skipped
        increment day by 1, marking for next day
        ans = BRASR(…,pos=0, ssf=0, total time, day+1)

int main()
    return BRASR(1,0,0,0,0,0);
```