Atlas

•••

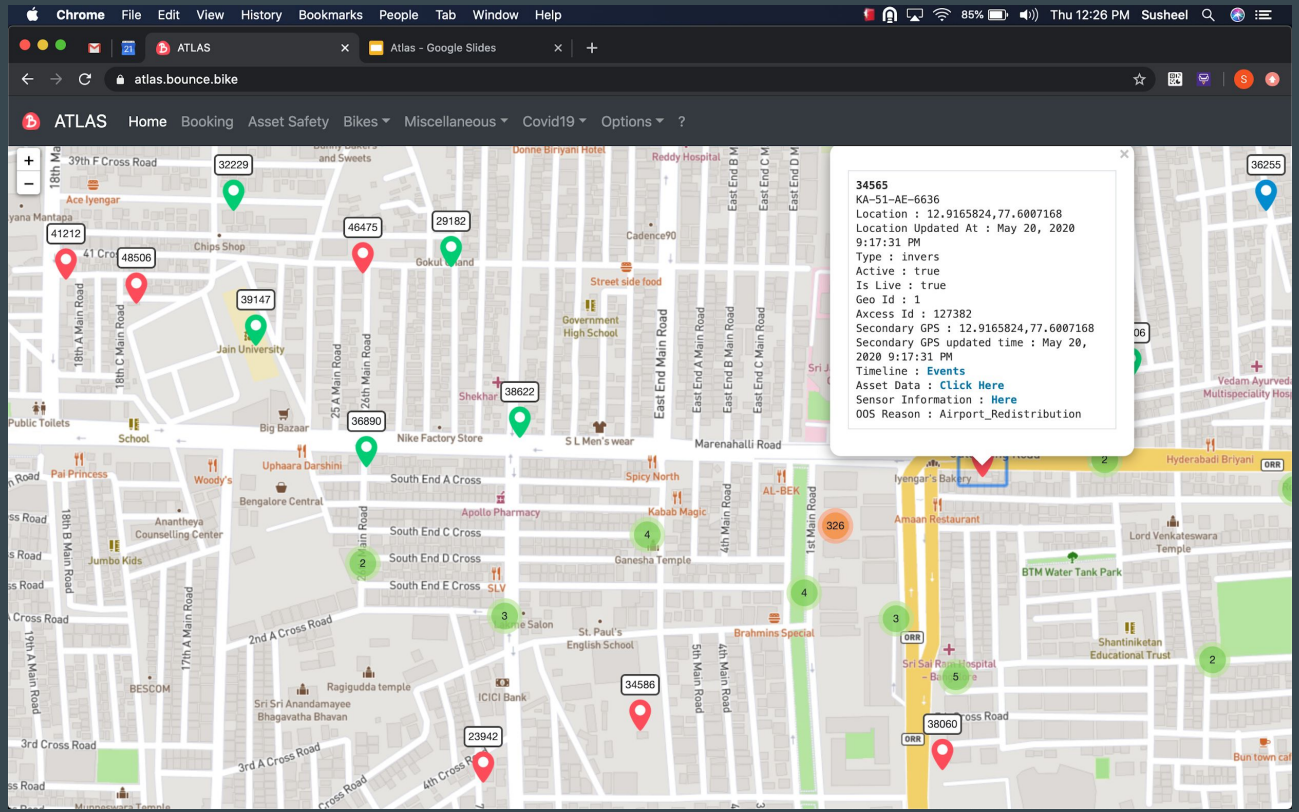
An OpenSource Internal Admin-Panel Framework

github.com/bounceshare/atlas

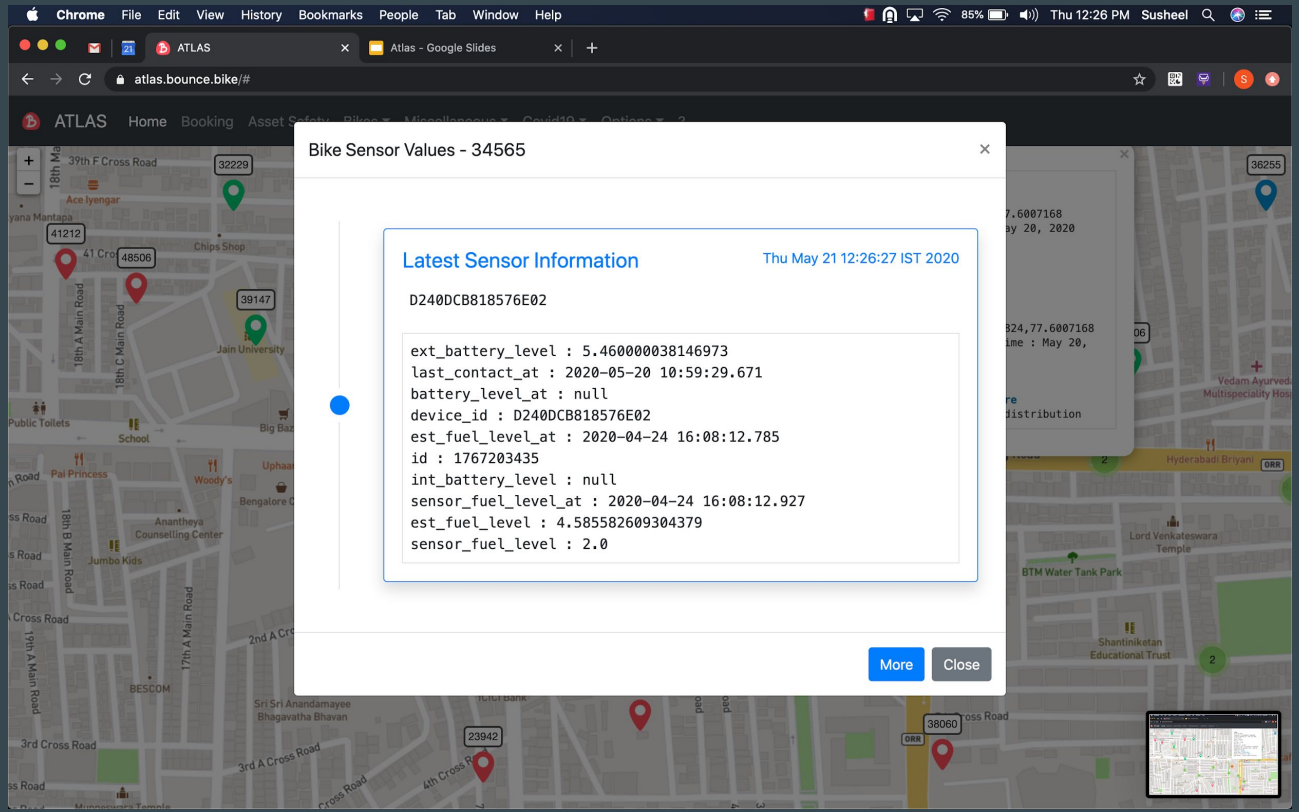
Features

- Map Controls - Markers, Fences, Circles & Paths
- GeoJSON Support
- Postgres CRUD UI
- Customizable Search Controls
- Config Driven
- Access Control
- Standalone Deployment Support
- Mobile Responsive
- AtlasSDK
- Backend Driven (so devs don’t have to fiddle around with UI code)

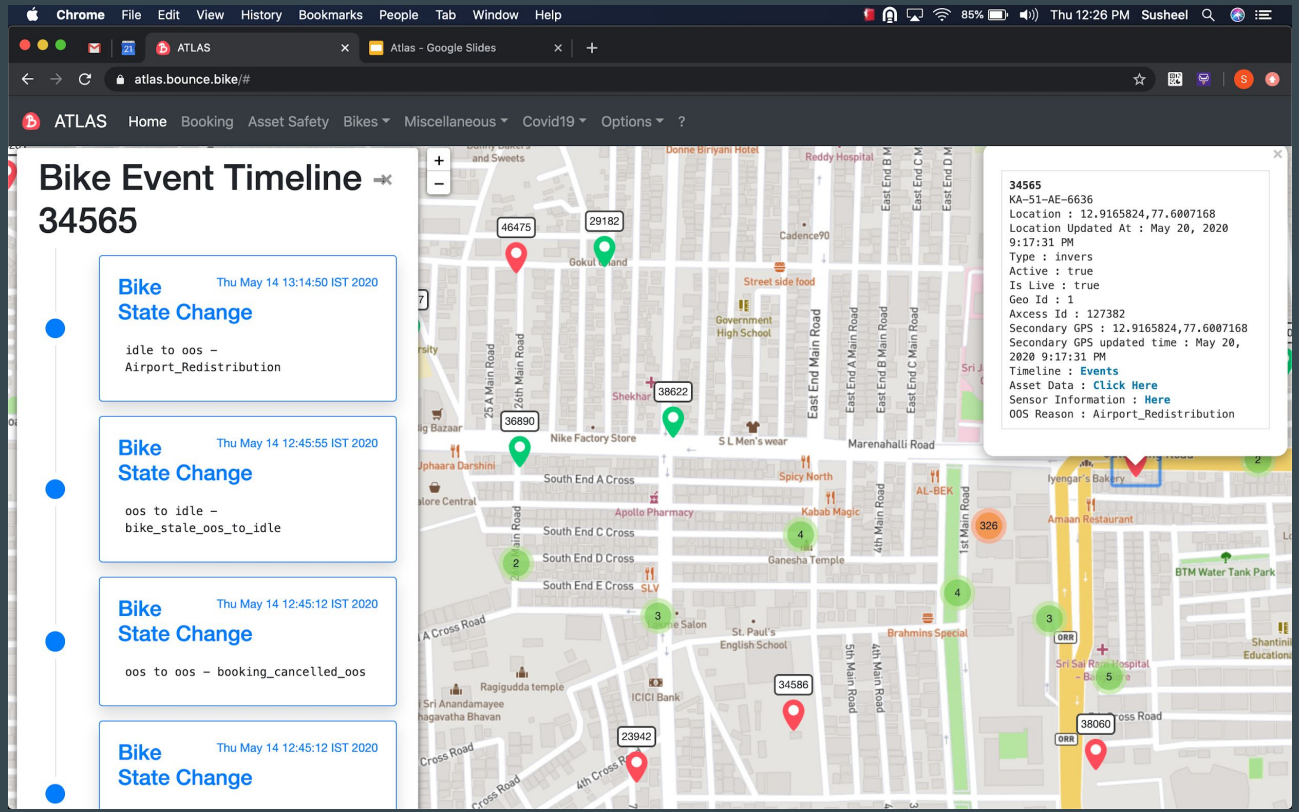
Screenshots



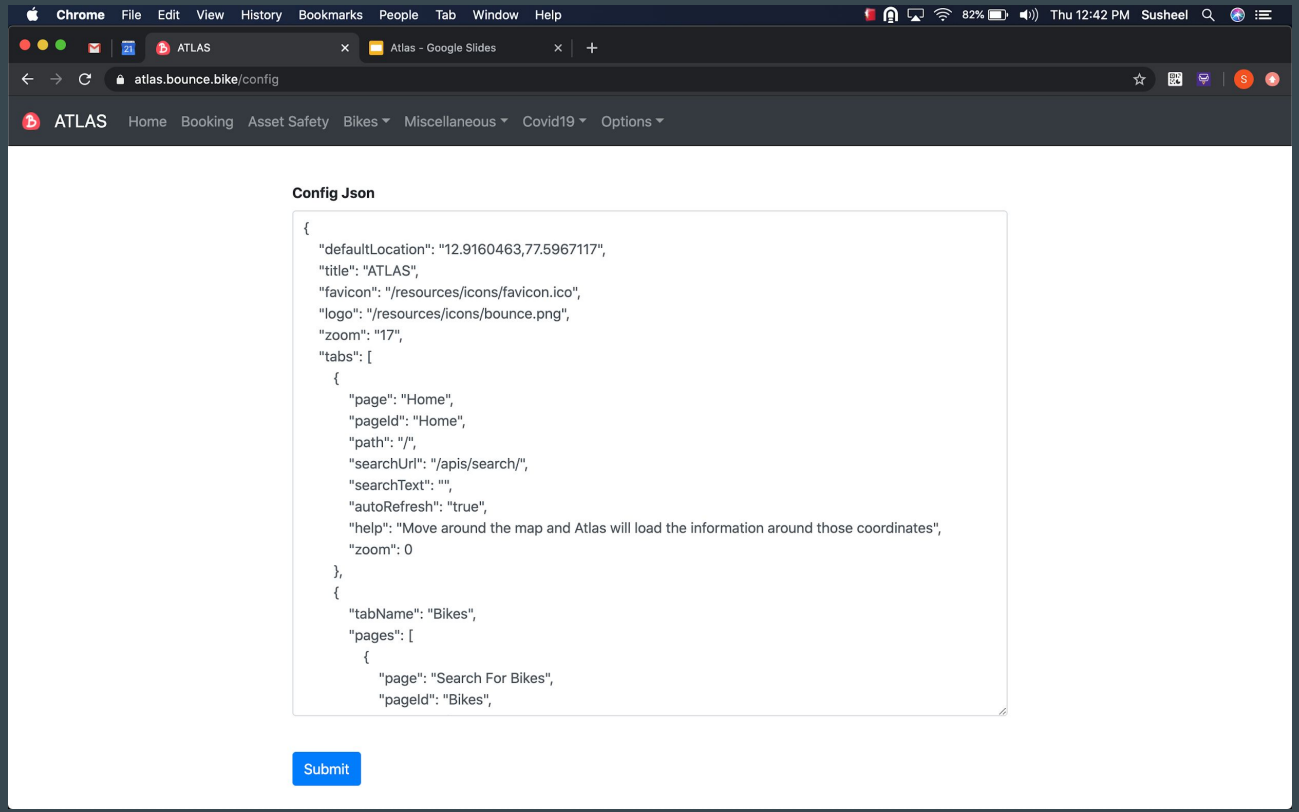
Screenshots



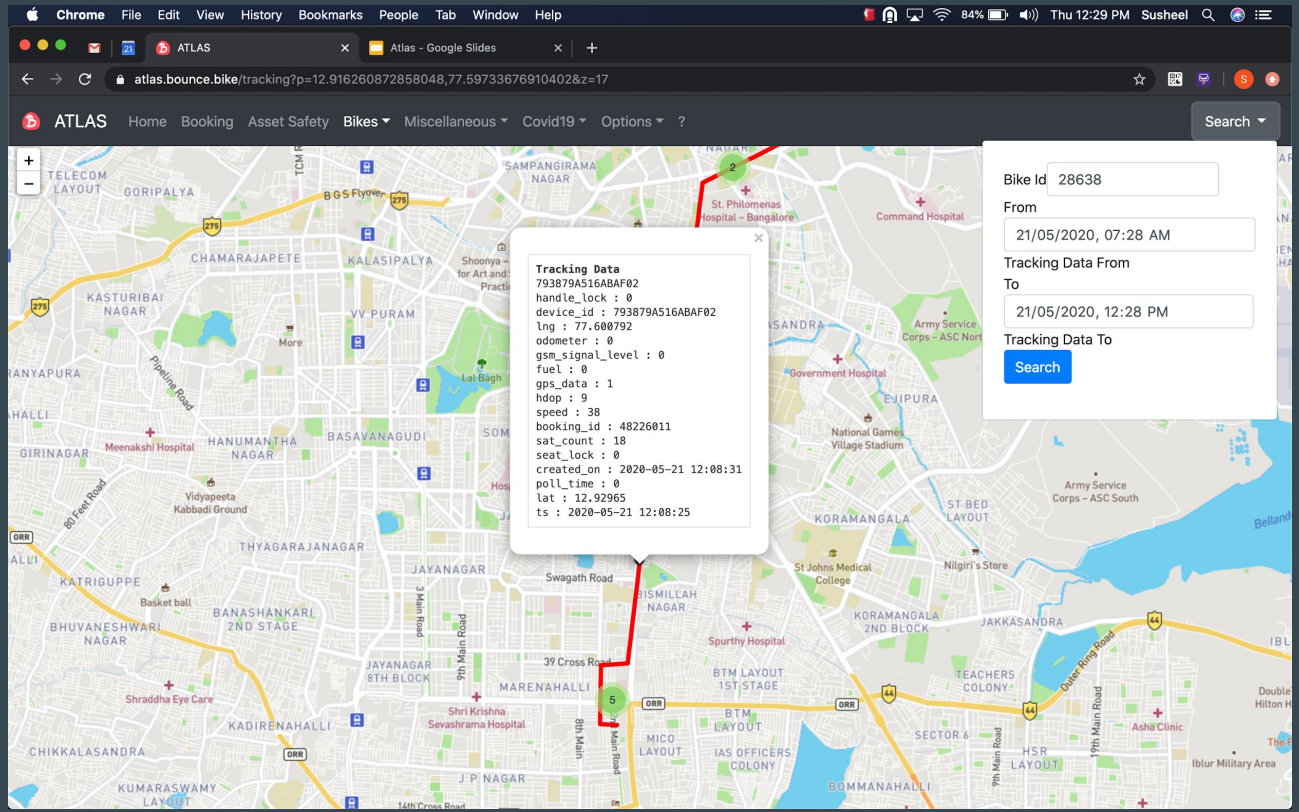
Screenshots



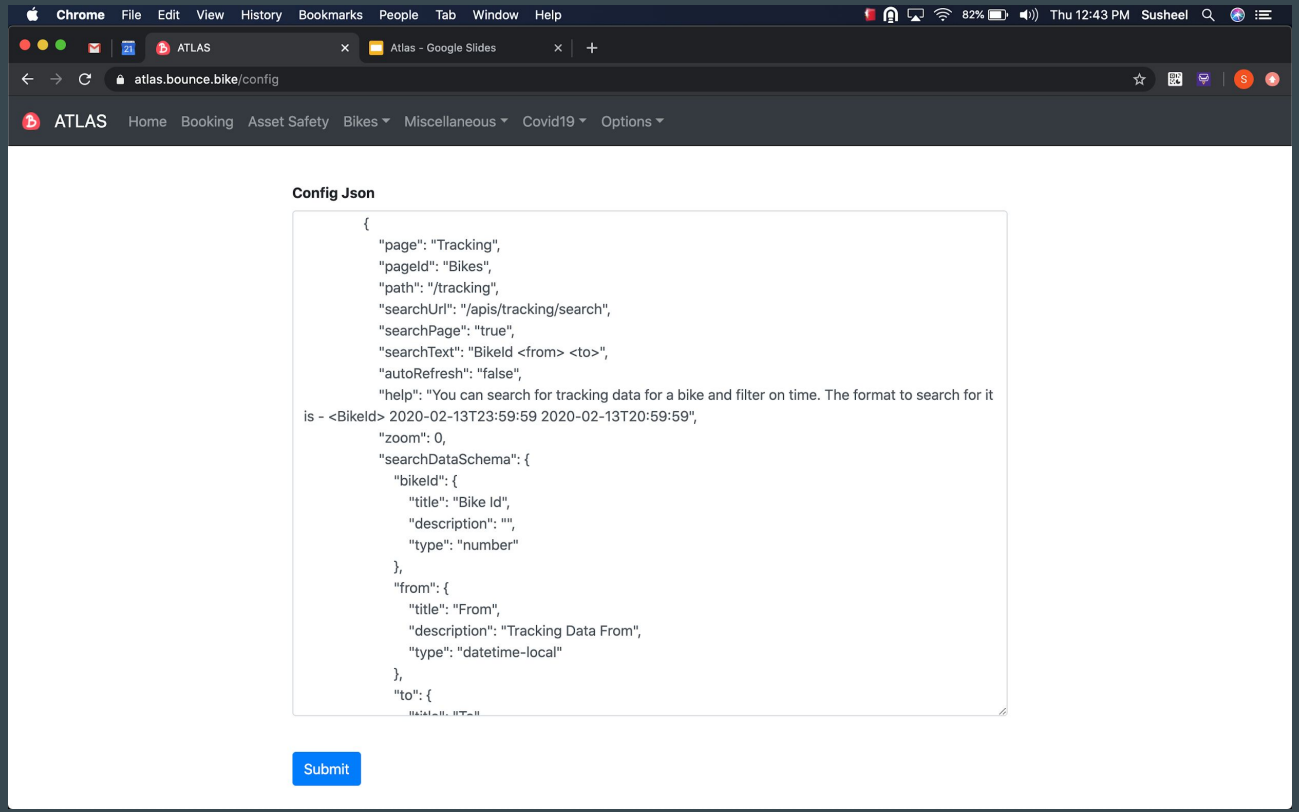
Screenshots



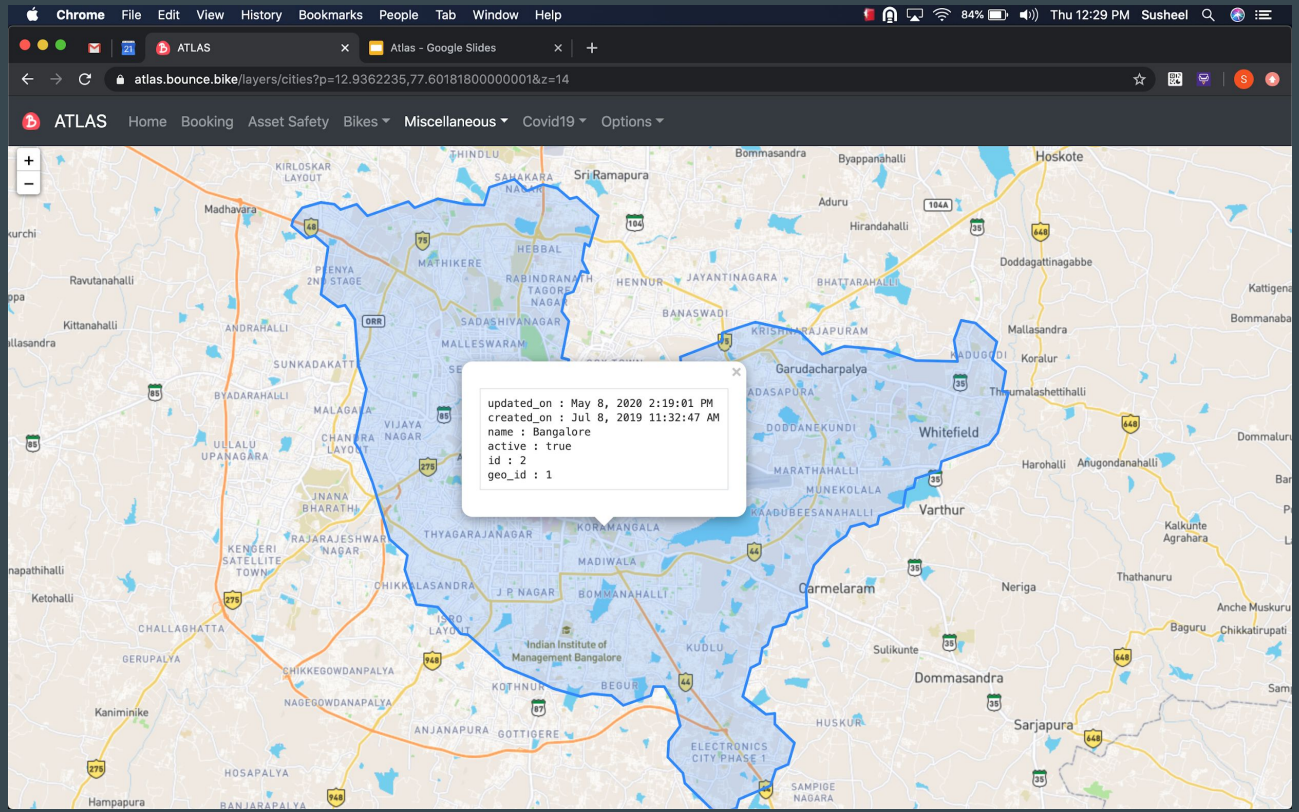
Screenshots



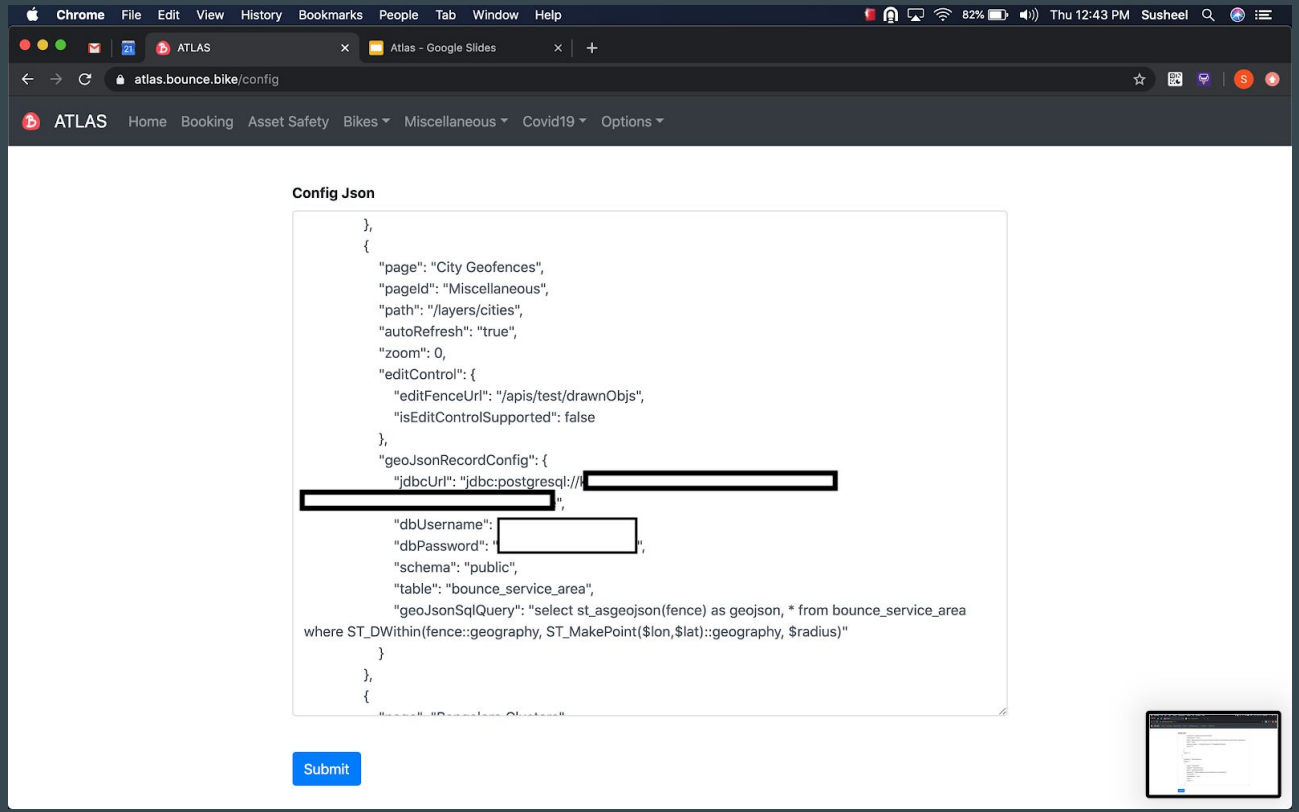
Screenshots



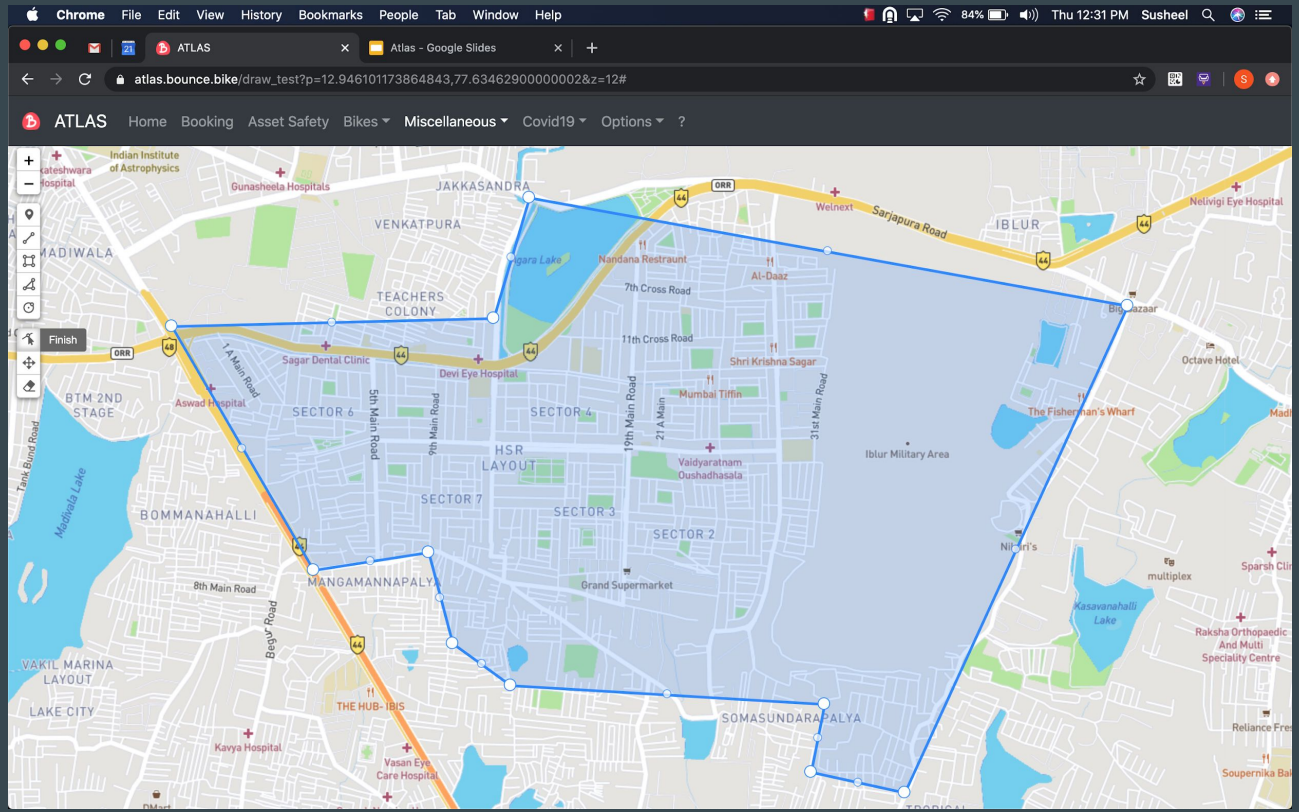
Screenshots



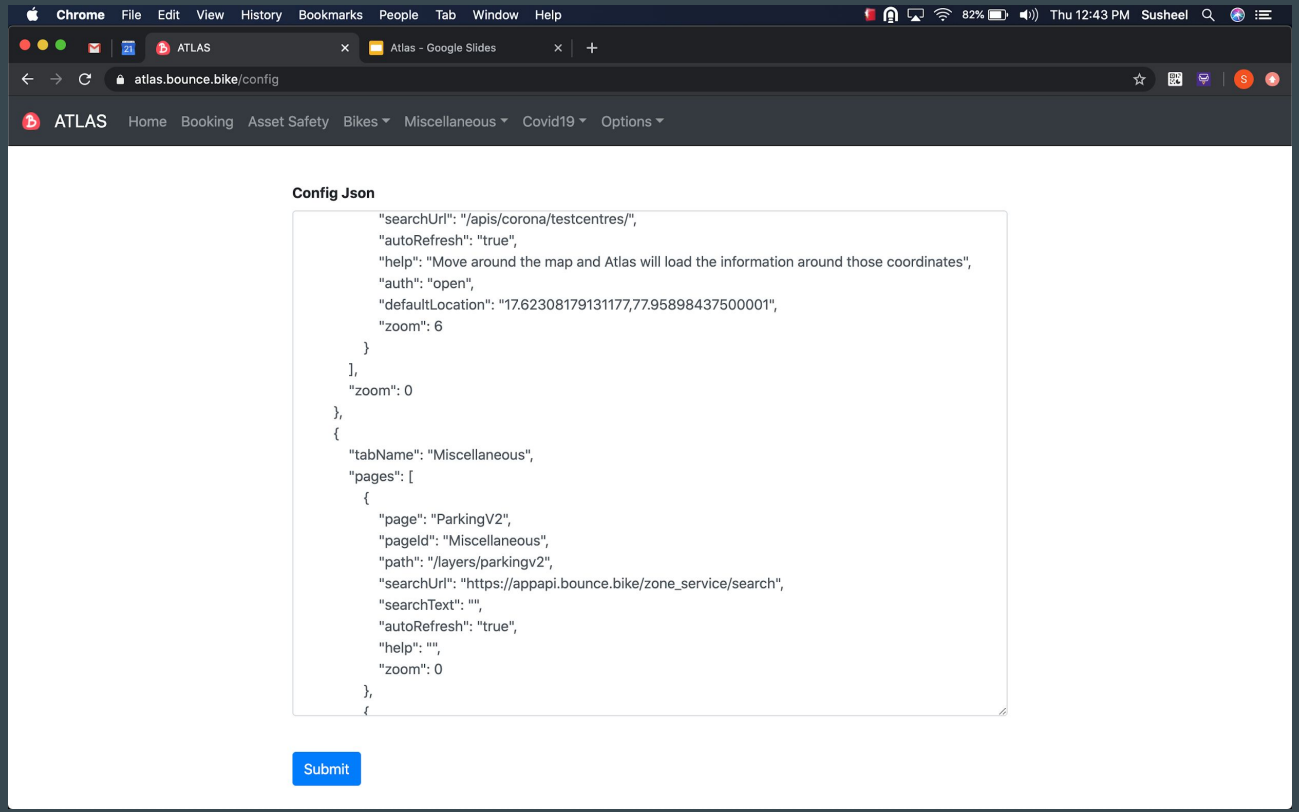
Screenshots



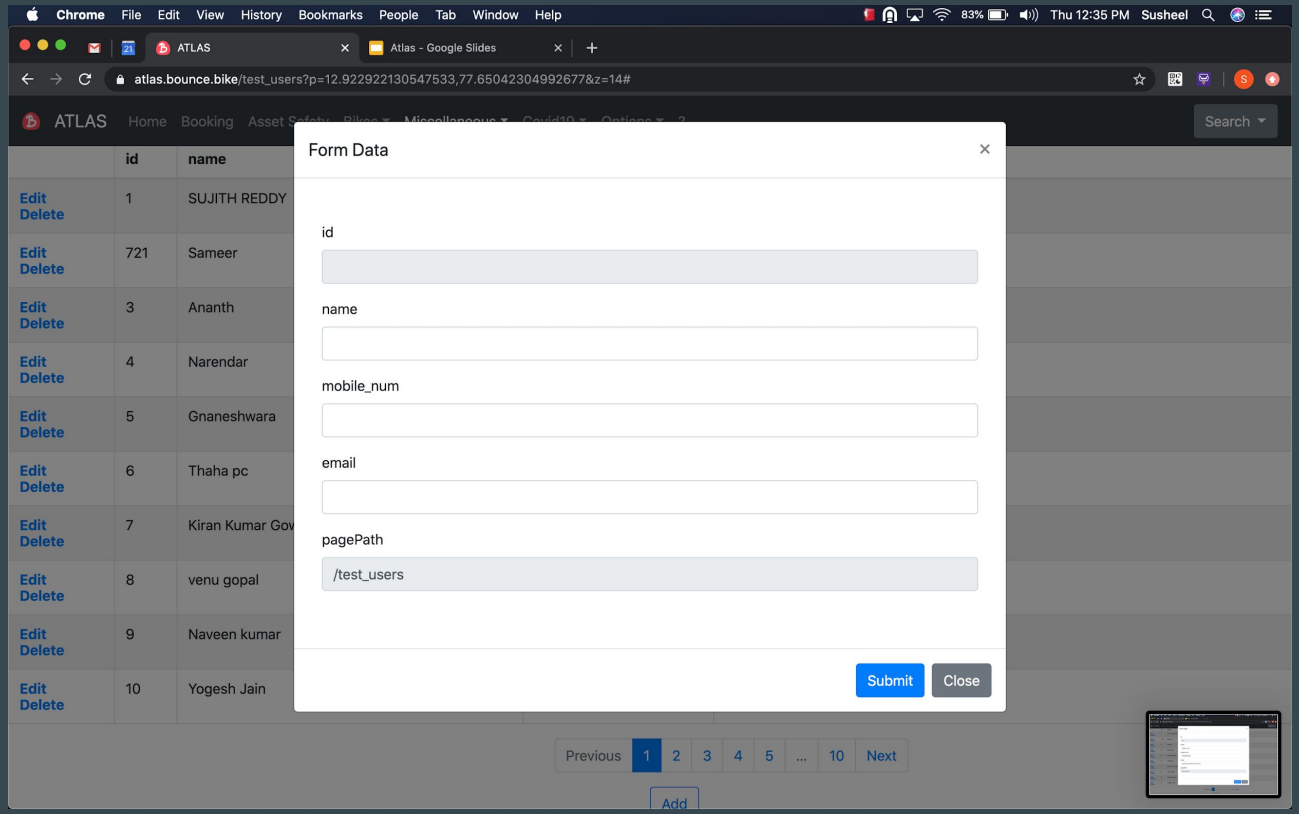
Screenshots



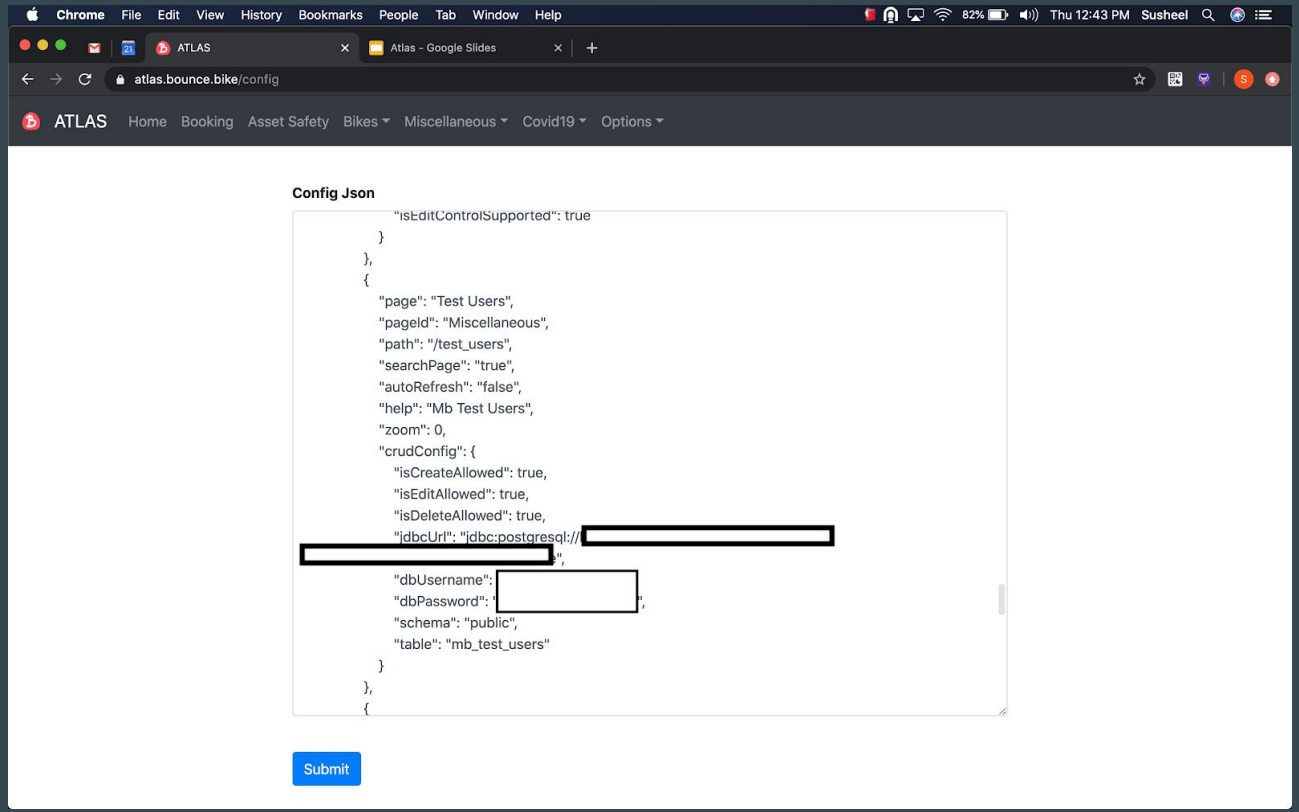
Screenshots



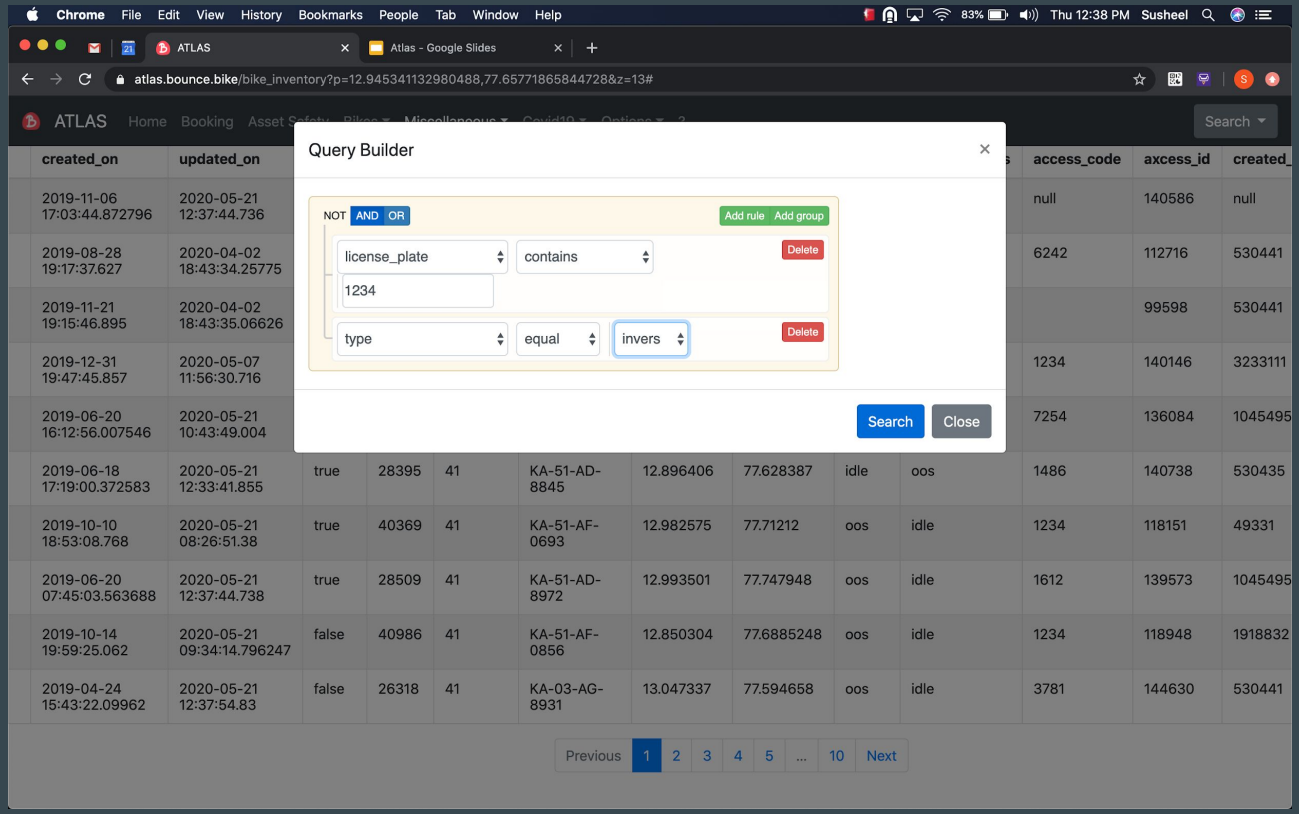
Screenshots



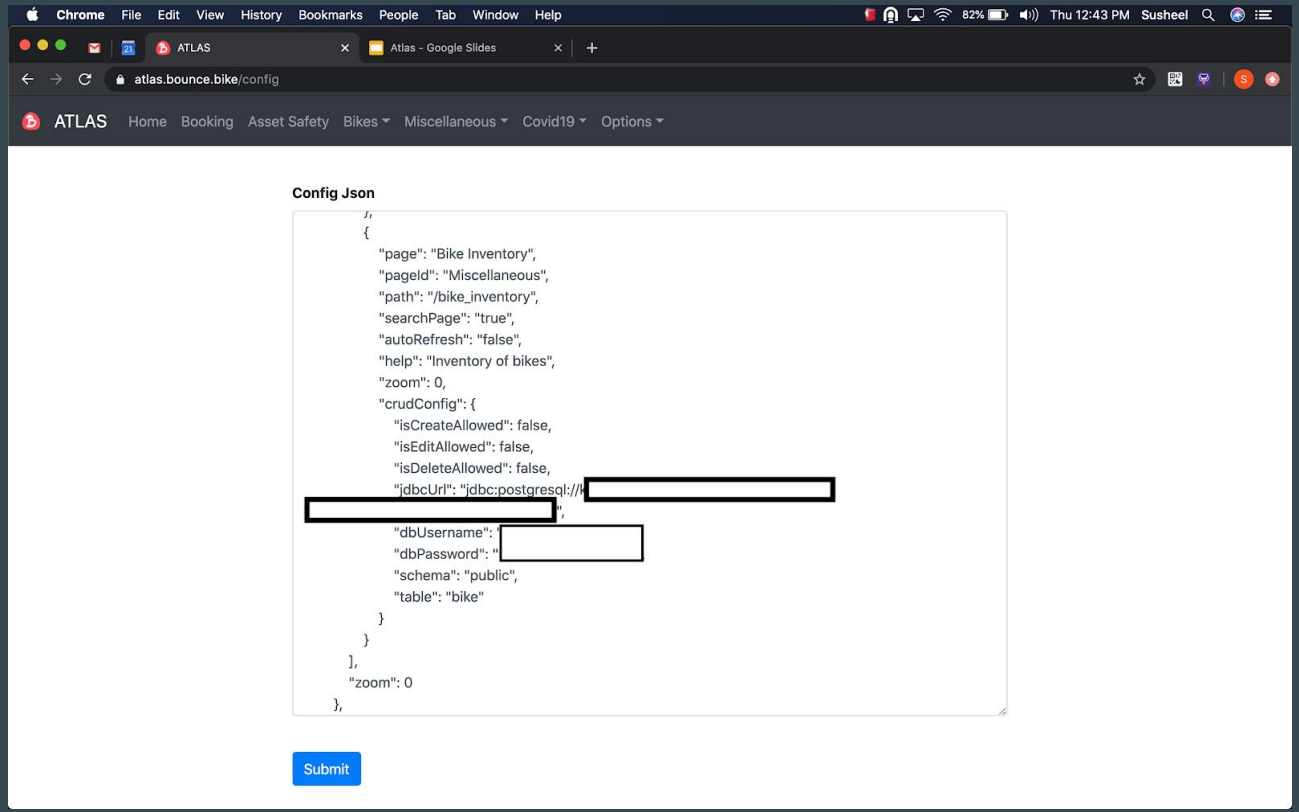
Screenshots



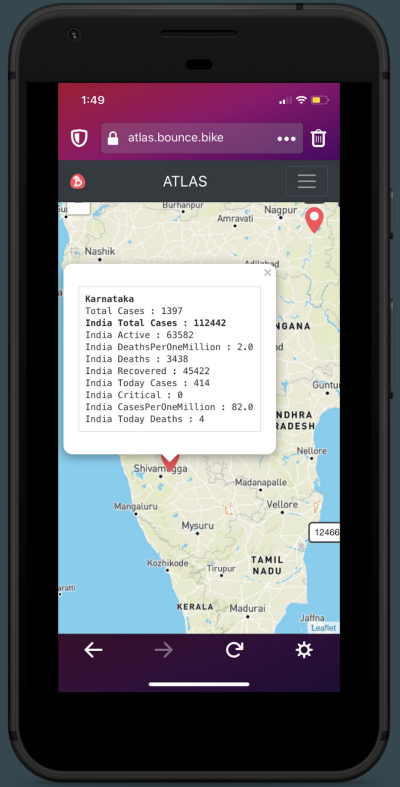
Screenshots

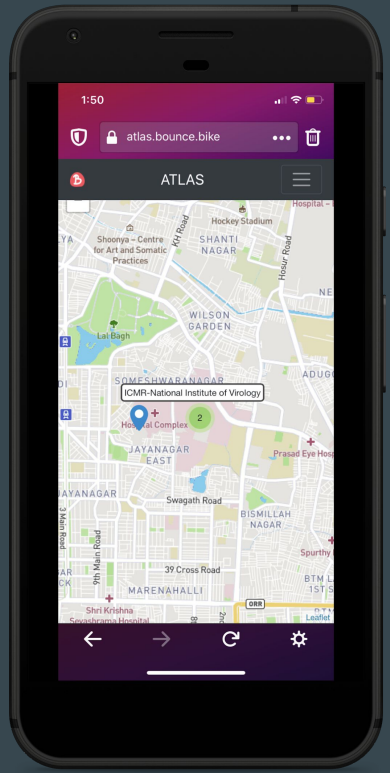


Screenshots



Works on Mobile (Responsive)





Tech Stack

- JAX-RS using Jersey deployed on Jetty
- Leaflet.js + jQuery + Bootstrap 4
- Freemarker for templating

Example - Listing Bikes on Map

```
@POST
@Path("/search/{path:.*}")
@Produces(MediaType.APPLICATION_JSON)
@Consumes({MediaType.APPLICATION_JSON})
@GoogleAuth
public void bikeListing(String inputString, @Suspended final AsyncResponse asyncResponse, @PathParam("path") String path) {
    logger.info("/apis/search");
    LayersApi layersApi = new LayersApi(inputString, asyncResponse, httpRequest, httpResponse, path);
    layersApi.onRequest();
}
```

Example - Listing Bikes on Map

```
public class LayersApi extends BaseApiHandler {

    private String path;

    public LayersApi(String inputString, AsyncResponse asyncResponse, HttpServletRequest httpRequest,
                     HttpServletResponse httpResponse, String path) {
        super(inputString, asyncResponse, httpRequest, httpResponse);
        this.path = path;
    }

    @Override
    public void onRequest(){
        try {
            super.onRequest();
        } catch (Exception e) {
            e.printStackTrace();
        }

        List<MarkerPojo> markers = Lists.newArrayList();
        List<FencePojo> fences = Lists.newArrayList();
        List<CirclePojo> circles = Lists.newArrayList();
        List<PathPojo> paths = Lists.newArrayList();

        double lat = input.optDouble( key: "lat",  defaultValue: 12.9160463);
        double lon = input.optDouble( key: "lon",  defaultValue: 77.5967117);
        int radius = input.optInt( key: "radius",  defaultValue: 5000);
        String searchQuery = input.optString( key: "searchQuery",  defaultValue: null);

        List<BikeRecord> bikes = QueryUtils.getBikes(lat, lon,  limit: 20000, radius,  status: null, searchQuery);
        List<MarkerPojo> bikeMarkers = RenderUtils.getBikesAsMarkers(bikes);
        markers.addAll(bikeMarkers);

        Map<Object, Object> response = Maps.newHashMap();
        response.put("markers", markers);
        response.put("fences", fences);
        response.put("circles", circles);
        response.put("paths", paths);
        if(TextUtils.isEmpty(searchQuery)) {
            response.put("autoRefresh", true);
        } else {
            response.put("autoRefresh", false);
        }

        sendSuccessResponse(asyncResponse, response);
    }
}
```