DAFTAR PUSTAKA

Badan Pusat Statistika . (2017). *Badan Pusat Statistika Kabupaten Bombana.* Bombana: Katalog BPS: 1403.7406.

Geosriwijaya. (2018, November 5). *Pengertian dan Fungsi Web Geographic Information System (WebGIS).* Diambil kembali dari GN consulting: https://geosriwijaya.com/2018/11/pengertian-dan-fungsi-web-geographic-information-system-*WebGIS*/

GISlearning. (2012, april 28). *Model Data GIS.* Diambil kembali dari gislearning.wordpress: https://gislearning.wordpress.com/2012/04/28/model-data-gis-2/

Haryana, K. S. (2008). Pengembangan Perangkat Lunak Dengan Menggunakan PHP. *Jurnal Computech & Bisnis*, Vol.2 No 1. 14-21.

Irwansyah, E. (2013). *Sistem Informasi Geografis: Prinsip Dasar Dan Pengembangan Aplikasi.* Yogyakarta: DigiBook.

Iskandaria. (2012). *Contoh Pengujian Black Box.* Diambil kembali dari kafegue.com: http://kafegue.com/contohpengujian-black-box-testing/

Kadir, A. (2002). *Pemrograman Web Mencakup: HTML, CSS, Javascript & PHP.* Yogyakarta: Penerbit Andi.

Khalil, S. M. (2015). Vector Data Model In GIS And How it Underpins A Range Of Widely Used Spatial Analysis Techniques. *Dutse Journal of Pure and Applied Sciences*, 1. 122 - 132.

Koesheryatin, S. T. (2014). *Aplikasi Internet Menggunakan HTML, CSS, dan Javascript.* Indonesia: Elex Media Komputindo.

Kuncoro, R. (2012). *Cara Menggunakan dan Memakai Oath Analysis (Analisis Jalur).* Bandung: Alfabeta.

Meliyanti, F. (2015). Efektivitas Penggunaan Leaflet Terhadap Peningkatan Pengetahuan Remaja Tentang HIV/AIDS di SMP Negeri 2 Ogan Komering Ulu.

Munawar. (2005). *Pemodelan Visual dengan UML.* Yogyakarta: Graha Ilmu.

Nugraha, S. S. (2016). Perencanaan Sistem Informasi Pengolahan Administrasi Keuangan Sekolah Menengah Pertama Islam 95 Terpadu Assalam Garut. *Jurnal Algoritma Sekolah Tinggi Teknologi Garut*, ISSN: 2302-7339,13(1), 320-332.

Nugroho, B. (2008). *Aplikasi Pemrograman Web Dinamis Dengan PHP dan MySQL.* Yogyakarta: Gava Media.

Prahasta, E. (2002). *Sistem Informasi Geografis: Tutorial ArcView.* Bandung: Informatika.

Putra, A. C. (2019, agustus 5). *Pengantar Database NoSQL dan MongoDB.* Diambil kembali dari candra.web.id: http://www.candra.web.id/pengantar-*database-NoSQL*-dan-*MongoDB*

Radar, O. (2011, juli 6). *What Is Node.js ?* Diambil kembali dari radar.oreilly.com: http://radar.oreilly.com/2011/07/what-is-node.html

Roger S, P. (2001). *Software Engineeering (A Practitioner's Appoarch)*. USA: Hill Higher Education.

Setyamidjaja, D. (2008). *Bertanam Kelapa dan pengolahannya*. Yogyakarta: Kanisius.

Shaff, A. (2020, januari 19). *Mengenal GeoJSON*. Diambil kembali dari goprau.com: https://goprau.com/index.php/artikel/5?judul=Mengenal+Geo*JSON*

Vandriansyah, D. (2008). *Filsafat ilmu komunikasi: suatu pengantar*. Jakarta: Indeks.

Warisno. (2007). *Budi Daya Kelapa Genjah.* Yogyakarta: Kanisius.

LAMPIRAN

Berikut implementasi *code* dalam pembuatan *website*:

Admin

layout.server.js

```
import { factory } from '$db/collection/factory';
import { redirect } from '@sveltejs/kit';

/** @type {import('./$types').LayoutServerLoad} */
export async function load({ cookies }) {
  if (!cookies.get('sessionId')) {
    throw redirect(303, '/auth/login');
  }

  return {
    factories: await factory
      .find({})
      .project({
        _id: { $toString: '$_id' },
        coordinate: 1,
        location: 1,
        owner: 1,
        type: 1,
        price: 1,
        photos: 1,
        production: 1
      })
      .toArray()
  };
}
```

page.server.js

```
import { district } from '$db/collection/district';
import { factory } from '$db/collection/factory';
import { fail, redirect } from '@sveltejs/kit';
/** @type {import('./$types').PageServerLoad} */
export async function load({ parent }) {
  const { factories } = await parent();
  const districts = district.find({}).project({ _id: 0, name: 1 }).toArray();

  return {
    factories: factories,
    districts: districts
  };
}

/** @type {import('./$types').Actions} */
export const actions = {
  create: async ({ request }) => {
    const formData = await request.formData();

    const owner = formData.get('owner');
    if (!owner) {
      return fail(400, { name: owner, ownerIsMissing: true, insertFailed: true });
    }

    let doc = {
      owner: { name: owner, contact: {} },
      coordinate: {},
      location: {},
      type: {},
      price: []
    };

    const lat = formData.get('latitude');
    const long = formData.get('longitude');
    if (lat && long) {
      doc.coordinate = {
        lat: parseFloat(lat),
        long: parseFloat(long)
      };
    }

    const district = formData.get('district');
    const subdistrict = formData.get('subdistrict');
    const street = formData.get('street');
    if (district && subdistrict && street) {
      doc.location = {
        district,
        subdistrict,
        street
      };
    }

    const email = formData.get('email');
    if (email) {
      doc.owner.contact['email'] = email;
    }

    const telp = formData.get('telp');
    if (telp) {
      doc.owner.contact['telp'] = telp;
    }

    const white = formData.get('white');
    const black = formData.get('black');
    doc.type = {
      white: white === 'on',
      black: black === 'on'
    };

    const prices = formData.get('prices');
    if (prices) {
      prices.split(' ').forEach((e) => {
        const price = parseInt(e);
        if (price) {
          doc.price.push(price);
        }
      });
    }

    try {
      await factory.insertOne(doc);
    } catch (err) {
      return fail(400, { insertFailed: true });
    }

    return { insertSuccess: true };
  },
  update: async ({ request }) => {
    const formData = await request.formData();

    const owner = formData.get('owner');
    if (!owner) {
      return fail(400, { name: owner, ownerIsMissing: true, insertFailed: true });
    }

    let updateDoc = { 'owner.name': owner };

    const lat = formData.get('latitude');
    const long = formData.get('longitude');
    if (lat && long) {
      updateDoc['coordinate.lat'] = parseFloat(lat);
      updateDoc['coordinate.long'] = parseFloat(long);
    }

    const district = formData.get('district');
    const subdistrict = formData.get('subdistrict');
    const street = formData.get('street');
    if (district && subdistrict && street) {
      updateDoc['location.district'] = district;
      updateDoc['location.subdistrict'] = subdistrict;
      updateDoc['location.street'] = street;
    }

    const email = formData.get('email');
    if (email) {
      updateDoc['owner.contact.email'] = email;
    }

    const telp = formData.get('telp');
    if (telp) {
      updateDoc['owner.contact.telp'] = telp;
    }

    const white = formData.get('white');
    const black = formData.get('black');
    updateDoc['type.white'] = white === 'on';
    updateDoc['type.black'] = black === 'on';

    const prices = formData.get('prices');
    if (prices) {
      let price = [];
      prices.split(' ').forEach((e) => {
        const p = parseInt(e);
        if (p) {
          price.push(p);
        }
      });
      updateDoc['price'] = price;
    }

    try {
      const res = await factory.updateOne({ 'owner.name': owner }, { $set: updateDoc });
      if (res.modifiedCount === 0) {
        return fail(400, { updateFailed: true });
      }
    } catch (err) {
      return fail(400, { updateFailed: true });
    }

    return { updateSuccess: true };
  },
  delete: async ({ request }) => {
    const formData = await request.formData();

    try {
      const result = await factory.deleteOne({ 'owner.name': formData.get('name') });
      if (result.deletedCount !== 1) {
        return fail(400, { deleteFailed: true });
      }
    } catch (e) {
      return fail(400, { deleteFailed: true });
    }

    return { deleteSuccess: true };
  },
  logout: async ({ cookies }) => {
    cookies.delete('sessionId');
    throw redirect(303, '/auth/login');
  }
};
```

Kecamatan

layout.server.js

```
import { district } from '$db/collection/district';

/** @type {import('./$types').LayoutServerLoad} */
export async function load() {
  return {
    districts: await district
      .find({})
      .project({
        _id: { $toString: '$_id' },
        name: 1,
        coordinate: 1,
        production: 1
      })
      .toArray()
  };
}
```

page.server.js

```
import { district } from '$db/collection/district';
import { fail } from '@sveltejs/kit';

/** @type {import('./$types').PageServerLoad} */
```

```
export async function load({ parent }) {
  const { districts } = await parent();

  return {
    districts: districts
  };
}

/** @type {import('./$types').Actions} */
export const actions = {
  create: async ({ request }) => {
    const formData = await request.formData();
    const name = formData.get('name');
    if (!name) {
      return fail(400, { requiredIsMissing: true, field: 'name' });
    }

    let doc = { name: name };

    const lat = parseFloat(formData.get('latitude'));
    const long = parseFloat(formData.get('longitude'));

    if (!lat || !long) {
      return fail(400, { requiredIsMissing: true, filed: 'coordinate' });
    }

    doc['coordinate'] = { lat, long };

    try {
      const res = await district.insertOne(doc);
      if (!res.insertedId) {
        return fail(400, { insertFailed: true });
      }
    } catch (err) {
      return fail(400, { insertException: true });
    }

    return { insertSuccess: true };
```

```
},
update: async ({ request }) => {
  const formData = await request.formData();
  const name = formData.get('name');
  if (!name) {
    return fail(400, { requiredIsMissing: true, field: 'name' });
  }

  let doc = { name: name };

  const lat = parseFloat(formData.get('latitude'));
  const long = parseFloat(formData.get('longitude'));

  if (!lat || !long) {
    return fail(400, { requiredIsMissing: true, filed: 'coordinate' });
  }

  doc['coordinate.lat'] = lat;
  doc['coordinate.long'] = long;

  try {
    const res = await district.updateOne({ name: name }, { $set: doc });
    if (res.modifiedCount === 0) {
      return fail(400, { updateFailed: true });
    }
  } catch (err) {
    return fail(400, { updateException: true });
  }

  return { updateSuccess: true };
},
delete: async ({ request }) => {
  const formData = await request.formData();
  const name = formData.get('name');

  try {
    const res = await district.deleteOne({ name: name });
    if (res.deletedCount !== 1) {
        return fail(400, { deleteFailed: true });
      }
    } catch (err) {
      return fail(400, { deleteFailed: true });
    }

    return { deleteSuccess: true };
  }
};
```

Produksi

page.server.js

```
import { district } from '$db/collection/district';
import { productionYear } from '$db/collection/productionYear';
import { fail } from '@sveltejs/kit';

/** @type {import('./$types').PageServerLoad} */
export async function load({ parent }) {
  let { districts } = await parent();

  districts = districts.map((district) => {
    district.production = district.production.find((p) => p.year === 2020);
    return district;
  });

  return {
    districts,
    productionYear: await productionYear
      .find({})
      .project({
        _id: { $toString: '$_id' },
        year: 1
      })
      .sort({ year: 1 })
      .toArray()
  };
}
```

```
/** @type {import('./$types').Actions} */
export const actions = {
  update: async ({ request }) => {
    const formData = await request.formData();
    const name = formData.get('name');
    const year = parseInt(formData.get('year'));
    const landArea = parseFloat(formData.get('area'));
    const productionCapacity = parseFloat(formData.get('cap'));
    const coconutProduction = parseFloat(formData.get('coconut'));

    if (!name || !year || !landArea || !productionCapacity || !coconutProduction) {
      return fail(400, { updateFailed: true, reason: 'Invalid data' });
    }

    const query = { name: name, 'production.year': year };
    const updateDoc = {
      'production.$.land_area': landArea,
      'production.$.production_cap': productionCapacity,
      'production.$.coconut': coconutProduction
    };

    try {
      let res = await district.updateOne(query, { $set: updateDoc });

      if (res.modifiedCount === 0) {
        res = await district.updateOne(
          { name: name },
          {
            $push: {
              production: {
                year: year,
                land_area: landArea,
                coconut: coconutProduction,
                production_cap: productionCapacity
              }
            }
          }
        );
      }

      if (res.modifiedCount === 0) {
        return fail(400, { updateFailed: true, reason: 'No Matched Query' });
      }
    } catch (err) {
      return fail(400, { updateExceptopn: true });
    }

    return { updateSuccess: true };
  }
};
```

Produksi

page.server.js

```
import { factory } from '$db/collection/factory';
import { productionYear } from '$db/collection/productionYear';
import { fail } from '@sveltejs/kit';

/** @type {import('./$types').PageServerLoad} */
export async function load({ parent }) {
  let { factories } = await parent();

  factories = factories.map((factory) => {
    factory.production = factory.production.find((p) => p.year === 2020);
    return factory;
  });

  return {
    productionYear: await productionYear
      .find({})
      .project({ _id: { $toString: '$_id' }, year: 1 })
      .sort({ year: 1 })
      .toArray(),
    factories: factories
  };
}
/** @type {import('./$types').Actions} */
```

```
export const actions = {
  update: async ({ request }) => {
    const formData = await request.formData();
    const production = parseInt(formData.get('production'));
    const year = parseInt(formData.get('year'));
    const owner = formData.get('owner');
    const query = { 'owner.name': owner, 'production.year': year };
    const updateDoc = { 'production.$.result': production };

    try {
      let res = await factory.updateOne(query, { $set: updateDoc });
      if (res.modifiedCount === 0) {
        res = await factory.updateOne(
          { 'owner.name': owner },
          {
            $push: {
              production: {
                year: year,
                result: production
              }
            }
          }
        );

        if (res.modifiedCount === 0) {
          return fail(400, { updateFailed: true, reason: 'No Matched Query' });
        }

        console.log(res);
      }
    } catch (err) {
      console.log(err);
      return fail(400, { updateFailed: true, reason: 'Exception' });
    }

    return { updateSuccess: true };
  }
};
```

User

page.server.js

```
import { kecamatan } from '$db/collection/kecamatan';


/** @type {import('./$types').PageServerLoad} */
export async function load() {
  return {
    kecamatan: await kecamatan.find({}).toArray()
  };
}
```

Api

District

Name

```
import { district } from '$db/collection/district';
import { json } from '@sveltejs/kit';


/** @type {import('./$types').RequestHandler} */
export async function GET({ params }) {
  const data = await district.findOne(
    { name: params.name },
    {
      projection: {
        _id: { $toString: '$_id' },
        name: 1,
        coordinate: 1
      }
    }
  );

  return json(data);
}
```

Production

```
import { district } from '$db/collection/district';
import { json } from '@sveltejs/kit';
/** @type {import('./$types').RequestHandler} */
```

```
export async function GET({ params }) {
  const doc = await district.findOne(
    {
      name: params.name
    },
    {
      projection: {
        _id: { $toString: '$_id' },
        name: 1,
        production: {
          $elemMatch: { year: parseInt(params.year) }
        }
      }
    }
  );

  return json(doc);
}
```

Production

```
import { district } from '$db/collection/district';
import { json } from '@sveltejs/kit';


/** @type {import('./$types').RequestHandler} */
export async function GET({ params }) {
  let districts = await district
    .find({})
    .project({
      _id: { $toString: '$_id' },
      name: 1,
      production: {
        $elemMatch: { year: parseInt(params.year) }
      }
    })
    .toArray();

  districts = districts.map((district) => {
    district.production = district?.production?.[0] ?? {};
    return district;
  });

  return json(districts);
}
```

Factory

Name

```
import { factory } from '$db/collection/factory';
import { json } from '@sveltejs/kit';

/** @type {import('./$types').RequestHandler} */
export async function GET({ params }) {
  const data = await factory
    .find({
      'owner.name': params.name
    })
    .project({
      _id: { $toString: '$_id' },
      owner: 1,
      coordinate: 1,
      location: 1,
      type: 1,
      price: 1
    })
    .toArray();

  return json(data[0]);
}
```

Production

```
import { factory } from '$db/collection/factory';
import { json } from '@sveltejs/kit';

/** @type {import('./$types').RequestHandler} */
export async function GET({ params }) {
  const doc = await factory.findOne(
    {
      'owner.name': params.name
    },
    {
      projection: {
        _id: { $toString: '$_id' },
        owner: 1,
        production: {
          $elemMatch: { year: parseInt(params.year) }
        }
      }
    }
  );

  return json(doc);
}
```

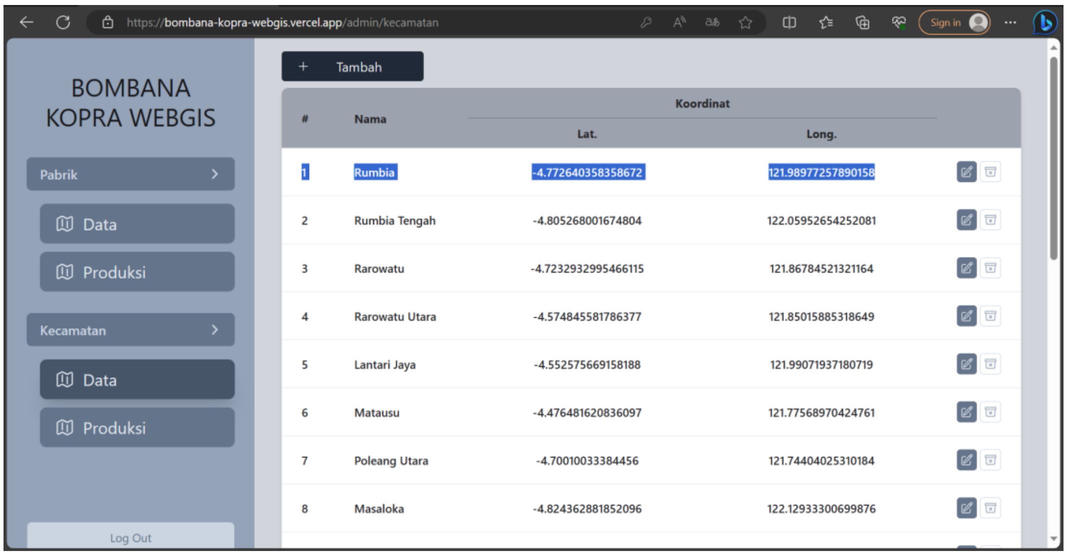
Production

```
import { factory } from "$db/collection/factory";
import { json } from '@sveltejs/kit';

/** @type {import('./$types').RequestHandler} */
export async function GET({ params }) {
  let factories = await factory
    .find({})
    .project({
      _id: { $toString: '$_id' },
      owner: 1,
      production: {
        $elemMatch: { year: parseInt(params.year) }
      }
    })
    .toArray();
  factories = factories.map((district) => {
    district.production = district?.production?.[0] ?? {};
    return district;
  });
  return json(factories);
}
```

Auth/login

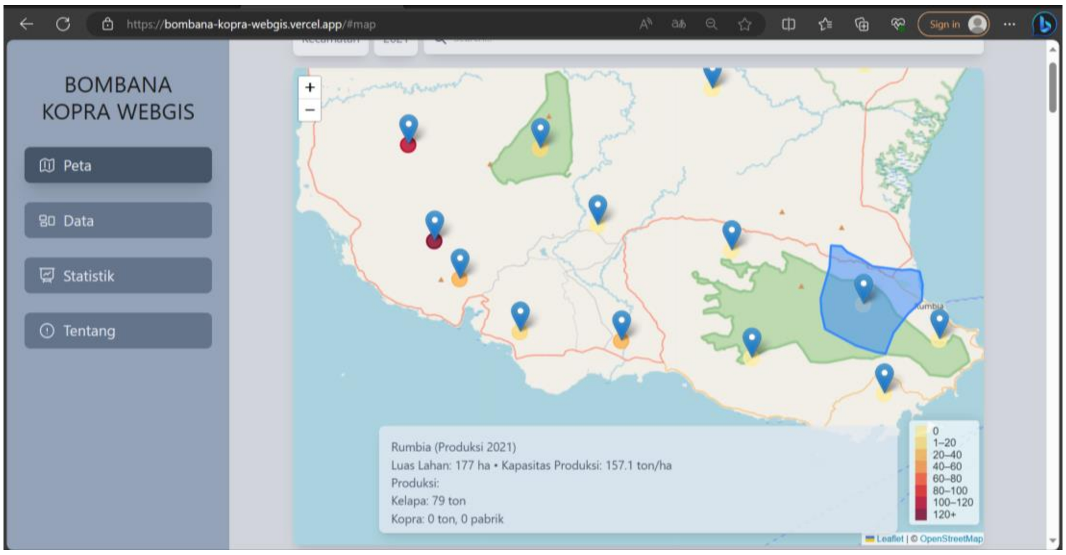
```
import { fail, redirect } from '@sveltejs/kit';


const users = {
  superadmin: 'Super Admin'
};


export async function load({ cookies }) {
  if (users[cookies.get('sessionId')]) {
    throw redirect(303, '/admin');
  }
}


/** @type {import('./$types').Actions} */
export const actions = {
  default: async ({ cookies, request }) => {
    const formData = await request.formData();
    const username = formData.get('username');
    const password = formData.get('password');

    if (!username || !password) {
      return fail(400, { missingField: true });
    }

    if (!users[username]) {
      return fail(400, { invalidUsername: true });
    }

    if (password !== users[username]) {
      return fail(400, { invalidCredential: true });
    }

    cookies.set('sessionId', username, { path: '/' });
    throw redirect(303, '/admin');
  }
};
```

Contoh hasil penyesuaian titik koordinat meliputi *Longitude* (garis bujur) dan *Latitude* (garis lintang) Rumbia yang ada di *website* dengan di *google maps*.



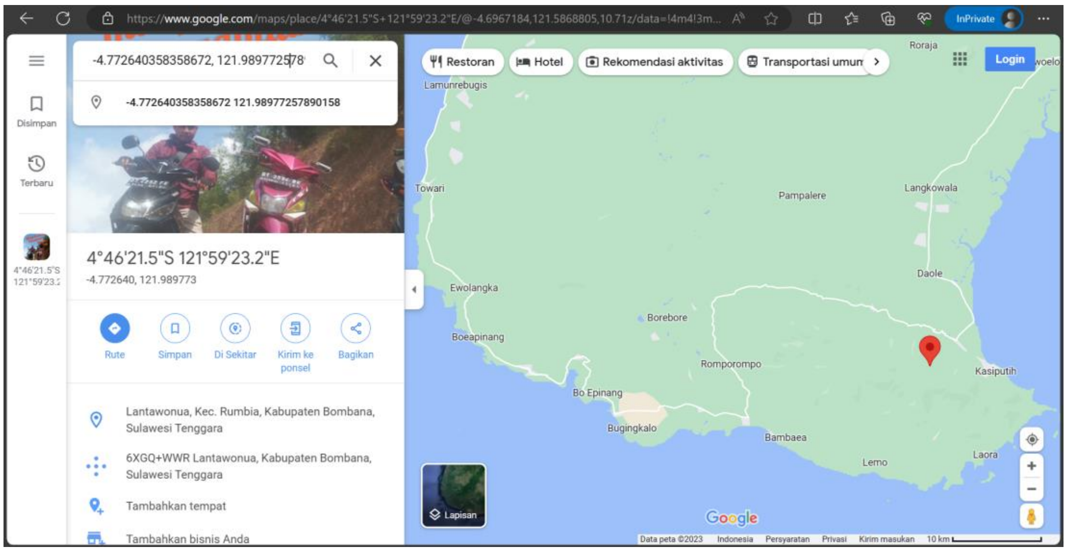
#	Nama	Koordinat	
		Lat.	Long.
1	Rumbia	-4.772640358358672	121.98977257890158
2	Rumbia Tengah	-4.805268001674804	122.05952654252081
3	Rarowatu	-4.7232932995466115	121.86784521321164
4	Rarowatu Utara	-4.574845581786377	121.85015885318649
5	Lantari Jaya	-4.552575669158188	121.99071937180719
6	Matausu	-4.476481620836097	121.77568970424761
7	Poleang Utara	-4.70010033384456	121.74404025310184
8	Masaloka	-4.824362881852096	122.12933300699876

Lampiran Gambar 1 Data Titik Koordinat Rumbia pada *Website*



Rumbia (Produksi 2021)
Luas Lahan: 177 ha • Kapasitas Produksi: 157.1 ton/ha
Produksi:
Kelapa: 79 ton
Kopra: 0 ton, 0 pabrik

Lampiran Gambar 2 Tampilan Titik Koordinat Rumbia pada Website



https://www.google.com/maps/place/4°46'21.5"S+121°59'23.2"E/@-4.6967184,121.5868805,10.71z/data=!4m4!3m...

-4.772640358358672, 121.98977257890158

4°46'21.5"S 121°59'23.2"E

-4.772640, 121.989773

Lantawonua, Kec. Rumbia, Kabupaten Bombana, Sulawesi Tenggara

6XGQ+WWR Lantawonua, Kabupaten Bombana, Sulawesi Tenggara

Lampiran Gambar 3 Tampilan Titik Koordinat Rumbia pada *GMaps*