Handover Document

ST. MARCHMAN LIBRARY

Table of Contents

Purpose of this Document

The purpose of this document is to provide a comprehensive overview of the HansRoslinger application, allowing any future owners/contributors to understand the frontend and backend stack, as well as technologies we are utilising as well as how we are utilising them.

This document also contains further information on the installations required to begin working on HanRoslinger, basic application structuring, and common issues we've come across while developing this application, and how to troubleshoot them.

What is HansRoslinger?

HansRoslinger is a gestured-based presentation tool that allows individuals to create, upload and present datasets and images, with the unique ability of being able to present with their webcam positioned behind the data being presented, and being able to manipulate the data using built-in custom gestures.

The purpose of HansRoslinger is to make the process of presenting information as engaging as possible to the audience, and as intuitive as possible to the presenter. Bygone are the days of mindlessly clicking through a powerpoint presentation, with only your voice or a small video of you in the corner. HansRoslinger quite literally puts the presentation right in your fingertips.

ENABLE GESTURES
3,500
3,000
2,500
1,000
500
0
January
February
March
April
May
July
August
September
October
December
Bar sales over 204

Sir Hans Rosling using HansRoslinger (chatgpt generated)

HansRoslinger is accessible via a public URL `hansroslinger.website`, or via the Cloud Run URL

`https://hans-roslinger-961228355326.us-central1.run.app`

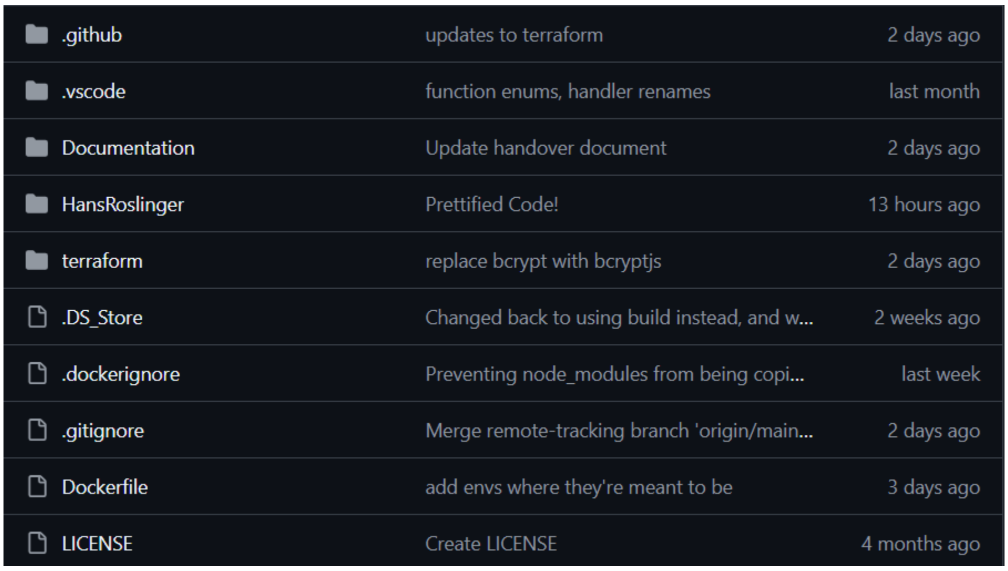
Please note that the production version is currently deployed in the us-central1 zone, this is because only US regions support domain mapping, which allowed us to use the custom URL. Doing the same with an Australia based region is possible however would incur additional costs

---

Repository Structure

- **Documentation/** — Project docs (CI/CD, tutorials, troubleshooting) and supporting images.

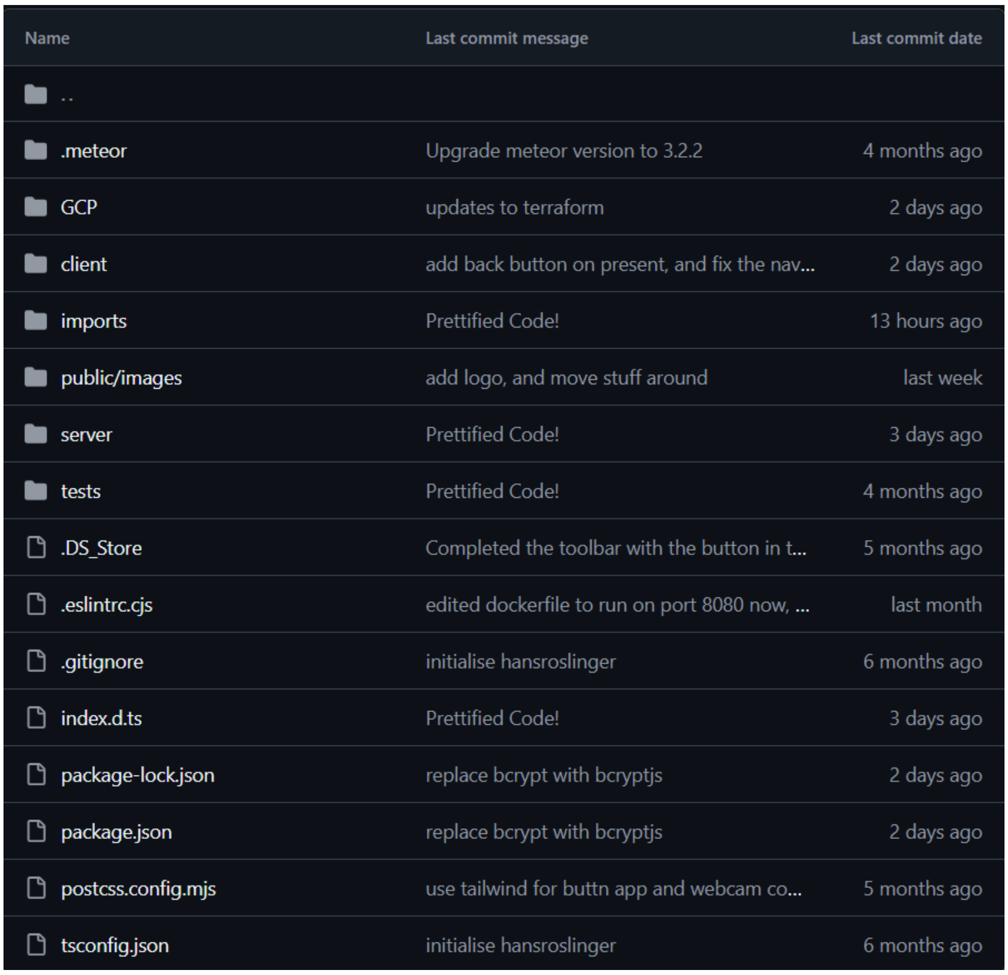
- **HansRoslinger/** — Meteor + React application source (client, server, shared imports, public assets, GCP helpers, tests).
- **terraform/** — Infrastructure as code for GCP (root configs, reusable module, per-env tfvars, state files).



.github	updates to terraform	2 days ago
.vscode	function enums, handler renames	last month
Documentation	Update handover document	2 days ago
HansRoslinger	Prettified Code!	13 hours ago
terraform	replace bcrypt with bcryptjs	2 days ago
.DS_Store	Changed back to using build instead, and w...	2 weeks ago
.dockerignore	Preventing node_modules from being copi...	last week
.gitignore	Merge remote-tracking branch 'origin/main...	2 days ago
Dockerfile	add envs where they're meant to be	3 days ago
LICENSE	Create LICENSE	4 months ago

HansRoslinger/ Structure

- **client/** — React/Meteor client entry and global HTML/CSS; bootstraps the UI.
- **server/** — Meteor server startup, methods, publications, and integrations.
- **imports/** — Shared app code: UI components/pages, handlers, gestures, and data layer.
- **public/** — Static assets served as-is (e.g., images/HandoverDocumenticons).
- **GCP/** — Google Cloud helpers/config (e.g., bucket utilities).
- **tests/** — App tests and helpers.



Name	Last commit message	Last commit date
..		
.meteor	Upgrade meteor version to 3.2.2	4 months ago
GCP	updates to terraform	2 days ago
client	add back button on present, and fix the nav...	2 days ago
imports	Prettified Code!	13 hours ago
public/images	add logo, and move stuff around	last week
server	Prettified Code!	3 days ago
tests	Prettified Code!	4 months ago
.DS_Store	Completed the toolbar with the button in t...	5 months ago
.eslintrc.cjs	edited dockerfile to run on port 8080 now, ...	last month
.gitignore	initialise hansroslinger	6 months ago
index.d.ts	Prettified Code!	3 days ago
package-lock.json	replace bcrypt with bcryptjs	2 days ago
package.json	replace bcrypt with bcryptjs	2 days ago
postcss.config.mjs	use tailwind for buttn app and webcam co...	5 months ago
tsconfig.json	initialise hansroslinger	6 months ago

Installation Guides

Follow these steps in order to set up the required software for HansRoslinger

1. Install Meteor using the following step
2. CD Into HansRoslinger directory and run `meteor npm install` to install all dependencies
3. Install MongoSH in order to perform local development with MongoDB
4. Install Gcloud CLI in order to perform testing with GCP
5. Install Terraform in order to provision cloud infrastructure

Once all of these steps have been followed to completion, you should be able to run the following commands to run HansRoslinger

```
# This will start hansroslinger
cd HansRoslinger
meteor

# in another terminal, you can access the local MongoDB Database using the
following
cd HansRoslinger
meteor mongo

# in another terminal, you can access information about the data stored on gcloud
```

```
with the following command
gcloud --help
# or you access it via the Google Cloud Console
https://console.cloud.google.com/welcome?invt=Ab6sAg&project=hansroslinger-468011
```

Suggested Tutorials/Documentation

If you are not familiar with the services above, feel free to take a look at the following documentation/tutorials to gain some familiarity with these services.

Meteor

- Technical Tutorial Plan
- Introduction to Meteor

MongoSH

- Introduction to MongoDB
- Introduction to MongoDB Atlas

Google Cloud Services

- Google Cloud Overview
- Cloud Run Overview
- Artifact Registry Overview
- Storage Bucket Overview

Terraform

- What is Terraform
- Getting started with Terraform

Technology Stack

HansRoslinger has been primariliy developed using the meteor full stack framework, which combines a number of different front end and back end frameworks to deliver an end to end web application.

Frontend Stack

HansRoslinger makes use of the React library in combination with Material UI (MUI) for the definition and use of UI component.

More information can be found about React here. More information can be found about Material UI here.

Each webpage is split into it's own React file in the UI directory, with additional handling logic and components contained within subdirectorys in the same location.

Backend Stack

All of the backend logic written for HansRoslinger is contained within the Meteor application in the form of TypeScript. TypeScript provides the benefit of strong in-time type hinting, which has improved troubleshooting as we developed this software.

More Information can be found about typescript here.

MongoDB Database

In Code

All persistent data for HansRoslinger (excluding user authentication information such as JWT which is stored in cookies) is stored in a NoSQL MongoDB database.

MongoDB Collections are defined directly in Meteor, example.

They are then published on start up of the server component of Meteor, example

And subsribed to directly in the client side backend logic in order to upload and retrieve documents from, example

MongoDB Atlas

MongoDB Atlas is used to host our database online, With the free tier allowing us to store 512MB worth of documents

- If you expect more space being required in the future you may need to look into payed plans, or enterprise systems

More information about MongoDB here.

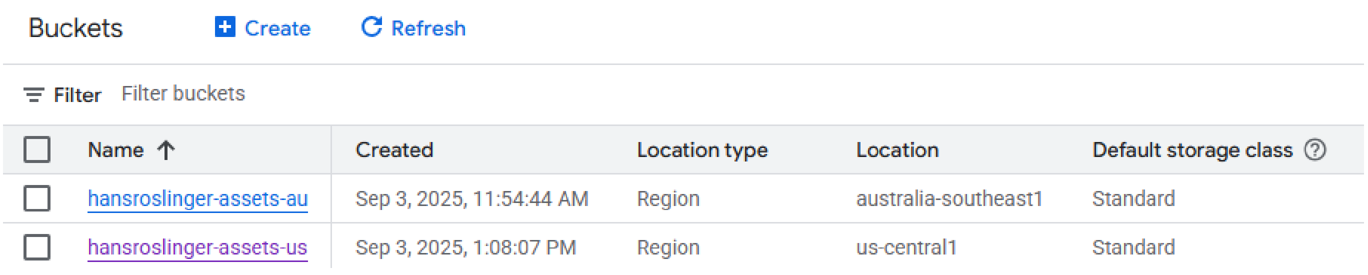
In order to connect to MongoDB Atlas we make use of a connection URL which specifies a path to our database, as well as sensitive login information for the account. This is stored via a Github Secret.

You can look into how this MongoDB Atlas database is connected to meteor through this guide here.

Google Cloud Bucket

All images uploaded by the user to HansRoslinger is stored in a google cloud bucket. Buckets allows for cheap and robust uploading of images, witih links to the files being stored as documents in the MongoDB Database.

The buckets are available here



Name	Created	Location type	Location	Default storage class
hansroslinger-assets-au	Sep 3, 2025, 11:54:44 AM	Region	australia-southeast1	Standard
hansroslinger-assets-us	Sep 3, 2025, 1:08:07 PM	Region	us-central1	Standard

Github/CICD

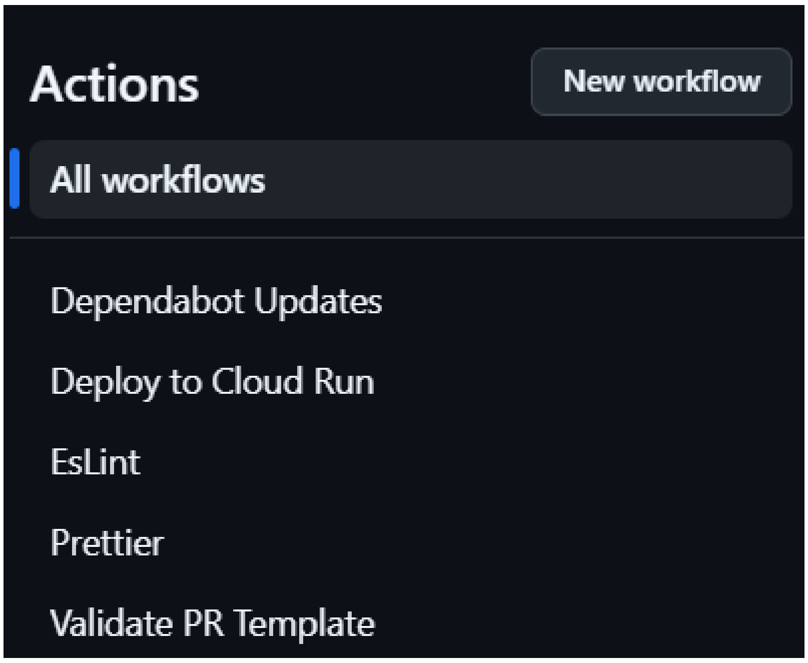
We utilise Github in to track developmnet, as well as to store documentation and run CICD workflows.

Documentation is found in this directory

CICD Workflows are found under .github and involve the following

- eslint which validates syntax for TypeScript and React (.tsx) files
- prettier which provides auto-linting when a PR is merged to main
- deploy which deploys the Meteor application to CloudRun, more information is available in the #Deployment section

We also have configured a pull request template in order to standardise the process of creating a pull request



How to Deployment to Production

Our application is deployed using Google Cloud Services in the form of a Cloud Run using request based billing. This reduces cost as you only pay for the time to fufill each request, and not the continuous deployment of the application., with the following steps being followed in order to deploy the application.

1. A DockerFile specifies the instructions to build the application into a container
2. A github workflow has been created which can be manually triggered (however this can be automated if needed) to deploy to GCP.

The steps executed to the pipeline are the following

1. Authenticate to the HansRoslinger GCP Project
2. Build the Docker Image and publish it to Google Artifact Registry
3. Deploy a CloudRun application using the provided Google Artifact Registry Image

If deployed successfully the HansRoslinger application should be accessible via this URL `https://hans-roslinger-961228355326.australia-southeast1.run.app/`

Information about CloudRun Information about request-based billing

Routing to Unique Domain Name

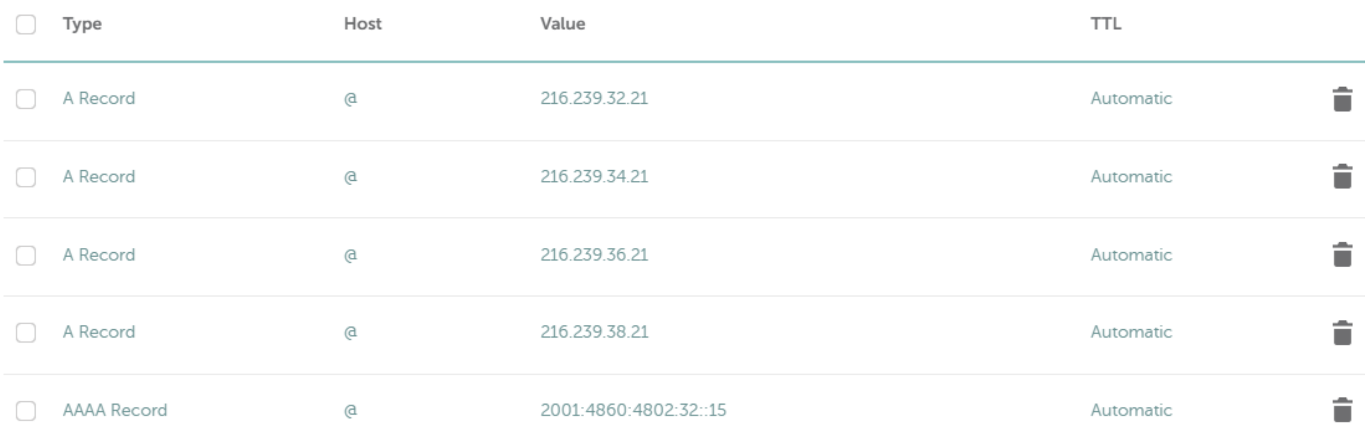
The HansRoslinger team has purchased the domain `hansroslinger.website` from namecheap.com

Because we are primarily hosting our services in Google Cloud, and this provider differs from where we had purchased the domain from, we have to perform some additional steps in order to set up routing between our domain name and our Cloud Run deployment

1. setup name servers in name cheap to point to google domain services
2. authenticate that you own the said domain in google domain services - Once you have authenticated that you own the domain, you are able to control all future routing from Google Domain Services
3. Update domain specifications in google so that the domain points to the Cloud Run service
4. You will then need to set up A, AAAA and TXT DNS Records to point to your Cloud Run Service. While a Cloud Run is deployed to a region, this can have multiple zones, so you may need to configure multiple A DNS Records (4 in this case)

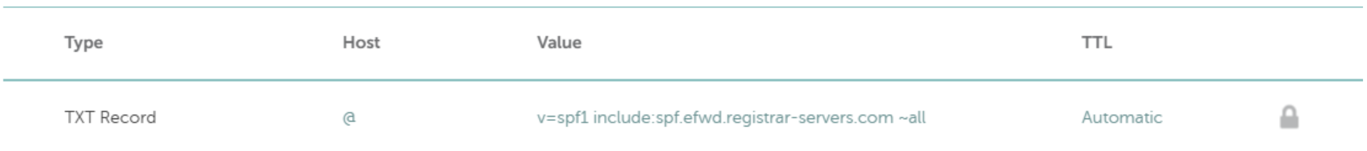
Purpose of DNS Records

> **A record:** Maps our domain name to the Google Server Zone IPv4 addresses. **AAAA record:** Maps our domain name to the Google Server IPv6 address. **TXT record:** Stores google site verification



Type	Host	Value	TTL
A Record	@	216.239.32.21	Automatic
A Record	@	216.239.34.21	Automatic
A Record	@	216.239.36.21	Automatic
A Record	@	216.239.38.21	Automatic
AAAA Record	@	2001:4860:4802:32::15	Automatic

A and AAAA Records



Type	Host	Value	TTL
TXT Record	@	v=spf1 include:spf.efwd.registrar-servers.com ~all	Automatic

TXT Records Records

Once this has been configured, google will automatically generate a SSL/TLS certificate to encrypt your domain, and shortly after you should be able to access your Cloud Run from your website.

Troubleshooting

While meteor is a comprehensive full stack framework, we have experienced countless issues with long build times, package issues, and modules not being found.

The following is a list of common issues that we encountered while developing HansRoslinger and how we fixed them.

Incorrect Node Version

Make sure you've downloaded the latest version of node as react-router requires at least `>=20.0.0`. Run `node -v` to check your version, if it is too low, uninstall node and then install from this website https://nodejs.org/en/download.

Module not found error

If a module not found error occurs when you are trying to run meteor, even after you have

- Validated that the import path is correct
- Validated that the data you are importing is being exported from the file

If you delete the `package-lock.json` file and then run `meteor npm install` from the `HansRoslinger` directory that should fix the issue

Meteor stuck on `Extracting meteor-tool@1.4.0-1`

This issue can cause meteor to spend sometimes multiple hours extracting tools, and sometimes even then it doesn't work.

This Github Issue describes the fix

1. Update git
2. run `npx meteor uninstall`, `npm uninstall -g meteor` or `choco uninstall meteor` depending on how you initially downloaded meteor
3. check that the `%appdata%/Local/.meteor` folder is removed (press Windows Key + R and type `%appdata%` to get to this directory)
4. remove the `HansRoslinger/.meteor/local` directory
5. remove the versions-file in `HansRoslinger/.meteor`
6. remove the `HansRoslinger/node_modules` directory
7. run `npx meteor install`, `npm install -g meteor --foreground-script` or `choco install meteor` depending on how you initially downloaded meteor
8. in the `HansRoslinger` folder, run meteor npm install
9. in the `HansRoslinger` folder, run meteor
10. This should hopefully have fixed the issue