AQUA WIZARDS

SURFERS'SENSE

MAINTENANCE DOCUMENT v1.1

Latest update : 11 June 2020

Team : AQUA WIZARDS
Member : Akshay Ijantkar
Nishant Tripathi
JIA XIE

Table of Contents

1 System Overview

1.1 Goals of the System and Target Audience

Surfers' Sense is mobile compatible web application which helps novice surfer to plan safe surf journey along with support services which will keep them updated about what's happening in surfing world and help them in the emergency situation. The application was designed and built keeping in mind surfer at rudimentary level, but it can be useful for surfers and swimmers of all experienced levels. It provides following functionalities:

- 3 days forecast for all Victorian beaches of Shark Attack Probabilities using Machine Learning (ML) model. Model trained on historic shark attack data combining it with aggregated historic weather and tides data for that particular location and date where shark attack incidents have happened.
- 3 days forecast for all Victorian beaches of Shark Sighting Probabilities using Statistical Model wherein it calculates similarity score of the given day by comparing aggregated weather and tide data of the date and location which you intend to calculate with aggregated historic weather and tides data where shark attack has happened.
- Suggesting necessary surf gears for surfing according to weather conditions like sea water temperature.
- Simplified Surf report which includes wave period, wave height, tides status and sunrise sunset timings.
- Personalized Surfboard specifications recommendation using physical parameters like age, weight, fitness, etc. along with surfing skill levels.
- Tailored surfing news related to topics which surfers will be interested into.
    - Topics which will give surfers word of cautions like Shark Attack, Jelly fish and Rip Current.
    - Topics which will motivate surfers like Surfing competitions and Surfing life.
- Emergency support service which will help surfers in case of emergency like injuries to navigate with nearby hospitals as per their current location. It just not recommends nearby hospitals with generic information like address, website, phone no. and google rating but also tells user responsiveness score (Percentage of patients seen on time by that hospital in the past) and also gives list of services provided by the hospital.

1.2. Product Link

https://surfersense.tk/

Access Credentials: Username: user | Password: EnterSite123@

1.3. Product Video:

Access Link: https://www.youtube.com/watch?v=1cWaSNg8Heg

2 System Organization

2.1 System Architecture

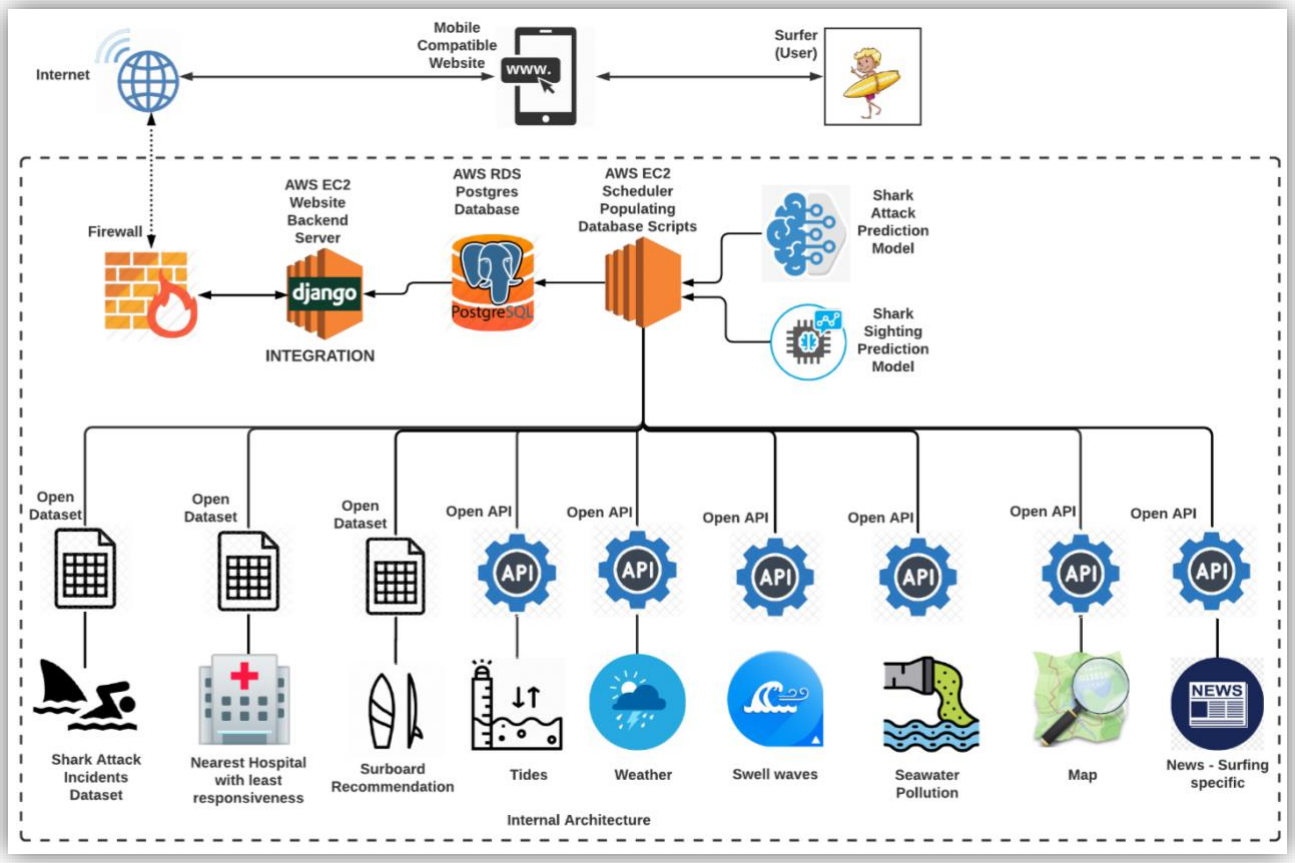

Figure 1: System Architecture

Mainly consists of 3 components which are: AWS EC2 instance 1 - Scheduler, AWS EC2 instance 1 -Django and AWS RDS – Postgres Database.

2.1.1 AWS RDS – Postgres Database

It consists of tables and views. We have categorized tables in two ways:

Static Tables: In which data remains constant for and doesn't change dynamically for a given scope of target audience.

Beach table: Consists of beach information like name, coordinates, etc. in Victoria.

Hospital table: Consists of hospital information like name, phone no. website, google rating, responsiveness score, etc. in Victoria.

Service table: Consists of all services that can be provided by any hospital in Victoria.

Hospital Services table (Bridging Table): Consists of Hospital ID (as Foreign Key) and Service ID (as Foreign Key), it maps all services that can be in given hospital.

Dynamic tables: In which data dynamically changes for a given scope of audience as per update frequency of each table in the system.

Tide Height table: Consists of 24-hour logged data of tide heights and status (RISE or FALL) of all Victorian beaches.

Weather table: Consists of 24-hour logged data of weather attributes like temperature, humidity, etc. of all Victorian beaches.

Astronomy table: Consists of sunrise and sunset timing information, of all Victorian beaches.

Sea water quality table: Consists of 24-hour log Ph value of all Victorian beaches.

Swell table: Consists of 24-hour logged data of wave height, wave period, water and air temperature for all Victorian beaches.

Extremes Tide table: Consist of daily status of extreme tide like High and Low tide of all Victorian beaches.

News table: Consists of news attributes like title, description, link, image, etc.

2.1.2 EC2 Instance 1 – Scheduler

It makes all API requests and fetches JSON responses, parses required data from JSON then cleans the data then does Missing Value Imputation if needed then transforms into tabular format and finally inserts it into Postgres DB everyday using Crontab Linux Scheduler.

Scheduler No.	Timing	Freq.	Purpose
1	1 am	Every Night	Requests data from Weather and Tides API and receives 24-hour log JSON response Parses required data from response json and then transform it to tabular form. Missing Value Imputation is achieved if needed and then inserts 24-hour log table to Postgres DB Calculates all aggregates over 24 hours of all weather and tides attributes These aggregated attributes will be used as features for Machine Learning model Inference to get Shark Attack Probability These aggregated attributes will be also used as features for Statistical model inference which compares current aggregated weather and tide data with historic aggregated weather and tide data and calculates Shark Sighting Probability using cosine similarity. This Shark Attack and Ssighting Probability along with aggregated weather and tides data for each beach for a particular day is also inserted in Postgres DB.
2	2 am	Every Night	Requests data from Astronomy, Swell and Sea water quality API and receives 24-hour log JSON response. Parses required data from response json and then transform it to tabular form. Missing Value Imputation is achieved if needed and then inserts 24-hour log table to Postgres DB.
3	3 am	Every Night	Requests data from News API for topics like Shark Attack, Jelly fish, rip current, Surfing Competition and Surfing life and receives JSON response. Filtering relevant news by using keyword search. Transforming relevant news data from JSON to tabular form and inserts it into Postgres DB.
4	4 am	Every Night	Deleting rows of the 5th day from the current day from the following tables: weather, tide, extremes tide, swell, astronomy, sea water quality tables. So, at a given time DB will have 7 days of data from tables which are dynamic to save space of Postgres DB which has space limitation.

Table 1: Each Scheduler Purpose

2.1.3 AWS EC2 Instance 2 - Django Server

In this Django server will be running and all integration has happened. It connects User Interface of the web application with Postgres DB.

- Django server and DB are connected using Django's default PostgreSQL engine. All needed for these to work in sync is psycopg2 python library that can be downloaded from the internet.
- All the url are codes in the urls.py files present in the SurfersBible folder
- Corresponding views are created with same function name in views.py file.
- Each view function renders a webpage that are stored in html format in the static folder of the web application.
- In views, python libraries like pytz are extensively used to get the current time for Australia.
- After time calculation is completed, a list of beaches and user given latitude, longitude are provided to Bing distance matrix API for distance calculations.
- From these calculated distances nearest 10 beaches are selected.
- All selected beach ids are used to get information from various tables in the database using beach id and given date time as search fields.
- Similar procedure is followed for emergency services feature as well.
- Remaining webpages are static or are made dynamic by the use of JavaScript which functions independently on the client side.
- Apart for them, the Nginx server is used to serve the static file and provide routing to the site name.
- Gunicorn library from python is used to connect to the wsgi HTTP python client and post 80 of the AWS EC2 instance.
- Supervior Unix technology is used for gunicorn as background process for all the time.

2.2 Major Functions

2.2.1 Shark Attack Prediction using Machine Learning

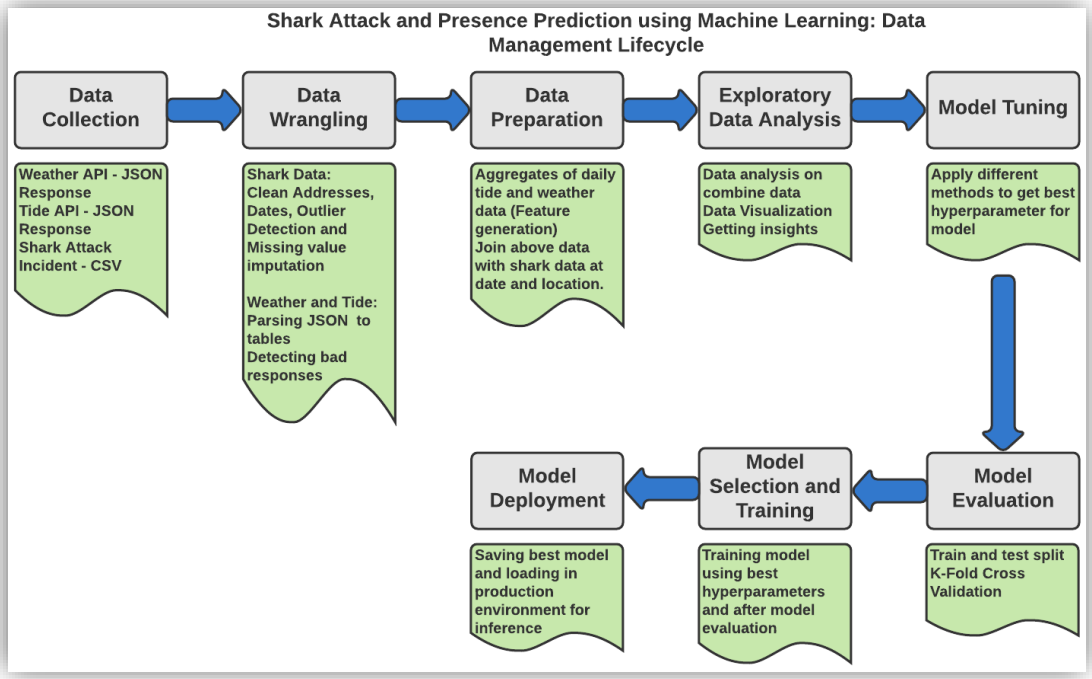

Figure 2: Machine Learning Cycle in the product

2.2.1.1 TRAINING PHASE:

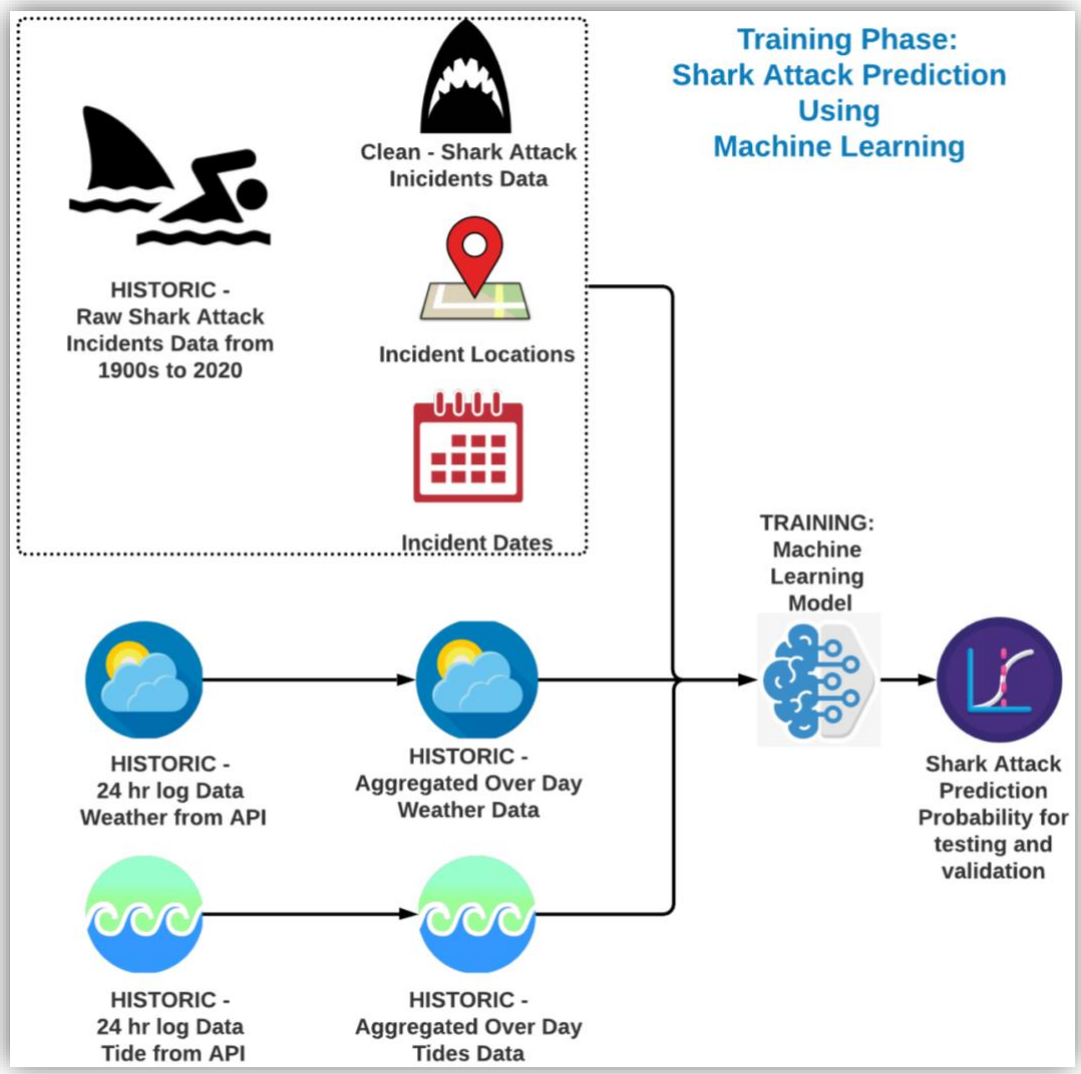

Figure 3: ML Training phase Flow

- Using Shark attack incident data from 1900 to 2020 which consists of all details of shark attack like Fatal (Target Variable), Date, Time of the day, Age of the victim, location where incident happened, etc.
- After doing thorough Exploratory Data Analysis we realized useful columns will be date, Fatal and location of the incidents, which will be used further in model building.
- From our research we found that Shark Attack and movements near the shore as some relations with weather and tides, so going with this concept we proceeded further.
- We collected Historic data of weather and tide for that particular date and location where all shark attacks as happened in the past.
- But we got 24-hour log data of weather and tide for the entire day, so we thought using aggregated of the weather and tides over a complete day.
- Following are the aggregates which we thought of using over a day: 'max','min','mean','median','std','var','sem'

- This aggregated weather and tides attributes over a day will be used as features and target column will be Fatal from Shark attack data for making Machine Learning Model.
- After trying different machine learning models, we finalized Catboost Classifier which gives 87% accuracy on train/test split validation and 84.5% accuracy on K-FOLD validation.
- We also did Bayesian hyperparameters tuning to get best set of hyperparameters for our final model.

2.2.1.2 INFERENCE PHASE:

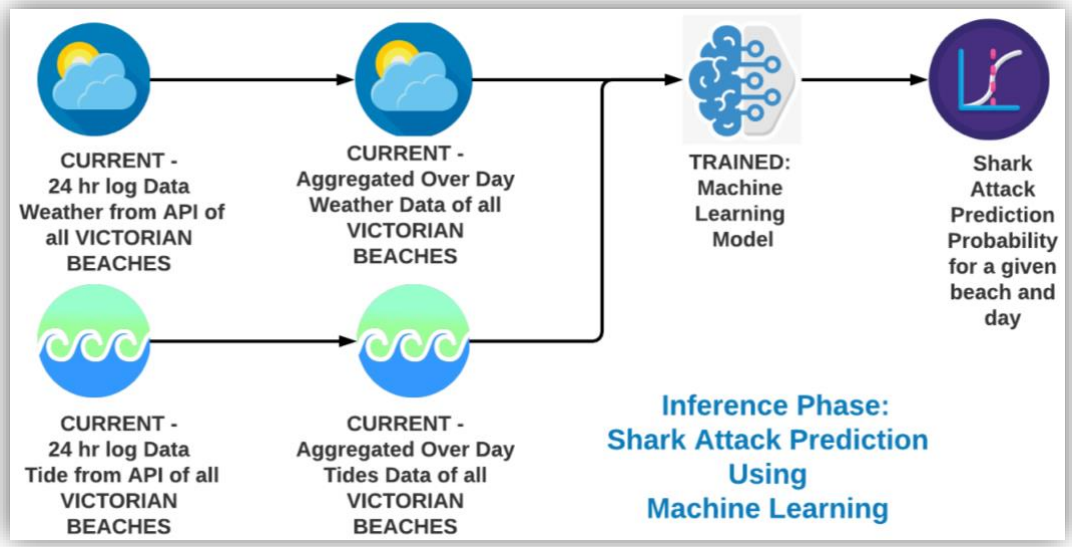

Figure 4: ML Inference Phase Data Flow

- Once model is trained, we used this model for predicting Shark Attack probability on day to day basis using weather and tides data for all Victorian beaches.
- For all Victorian beaches we collect 24-hour logged data of weather and tide data.
- Taking aggregates of the above data over a day which will be used as features for our trained Catboost ML model.
- Inference on daily aggregated weather and tides data using the ML model to get Shark Attack Probability.

2.2.2 Shark Sighting Prediction using Statistical Model

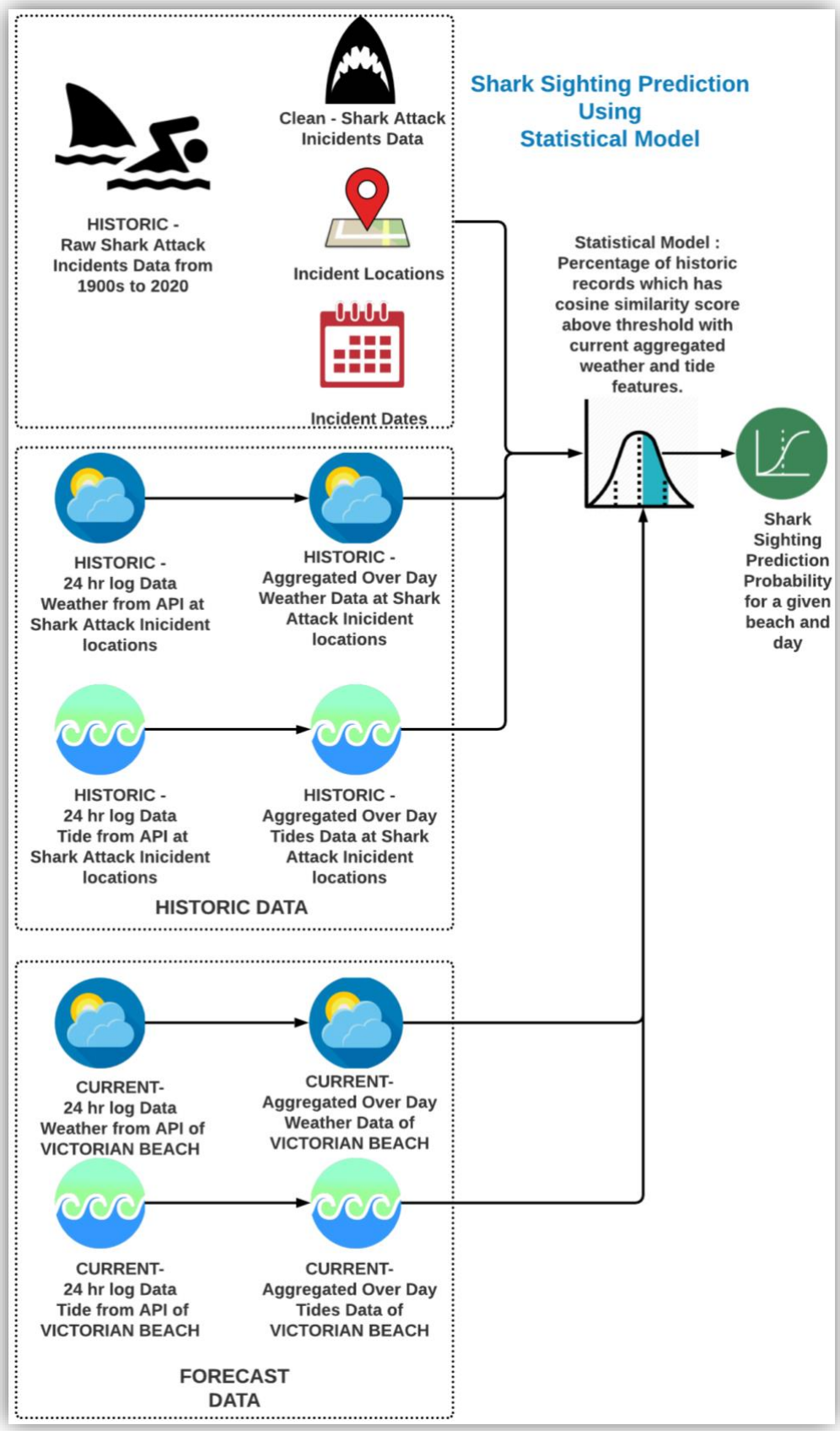

Figure 5: Shark Sighting using Statistical Model

- From the process of getting shark attack probability we have the table which consist of historic shark attack incidents data (required columns date and location only).
- We also calculate aggregated weather and tides data over a day as features for getting Shark Attack probability.
- For getting Shark Sighting Probability we used data from above two steps as follows:
    - Irrespective whether Shark Attack was fatal or not we can say that during all the incidents shark was sighted near the sea shore.
    - Considering above thing, we calculated Shark Sighting Probability by calculating percentage of current aggregated features (Daily aggregated weather and tides data) which has cosine similarity above threshold (in our case we kept 0.9) compare to historic aggregated weather and tides data features (where shark attack happened that means shark was spotted) using cosine similarity as statistical model.

2.2.3 Personalized Surfboard Specification Recommendation

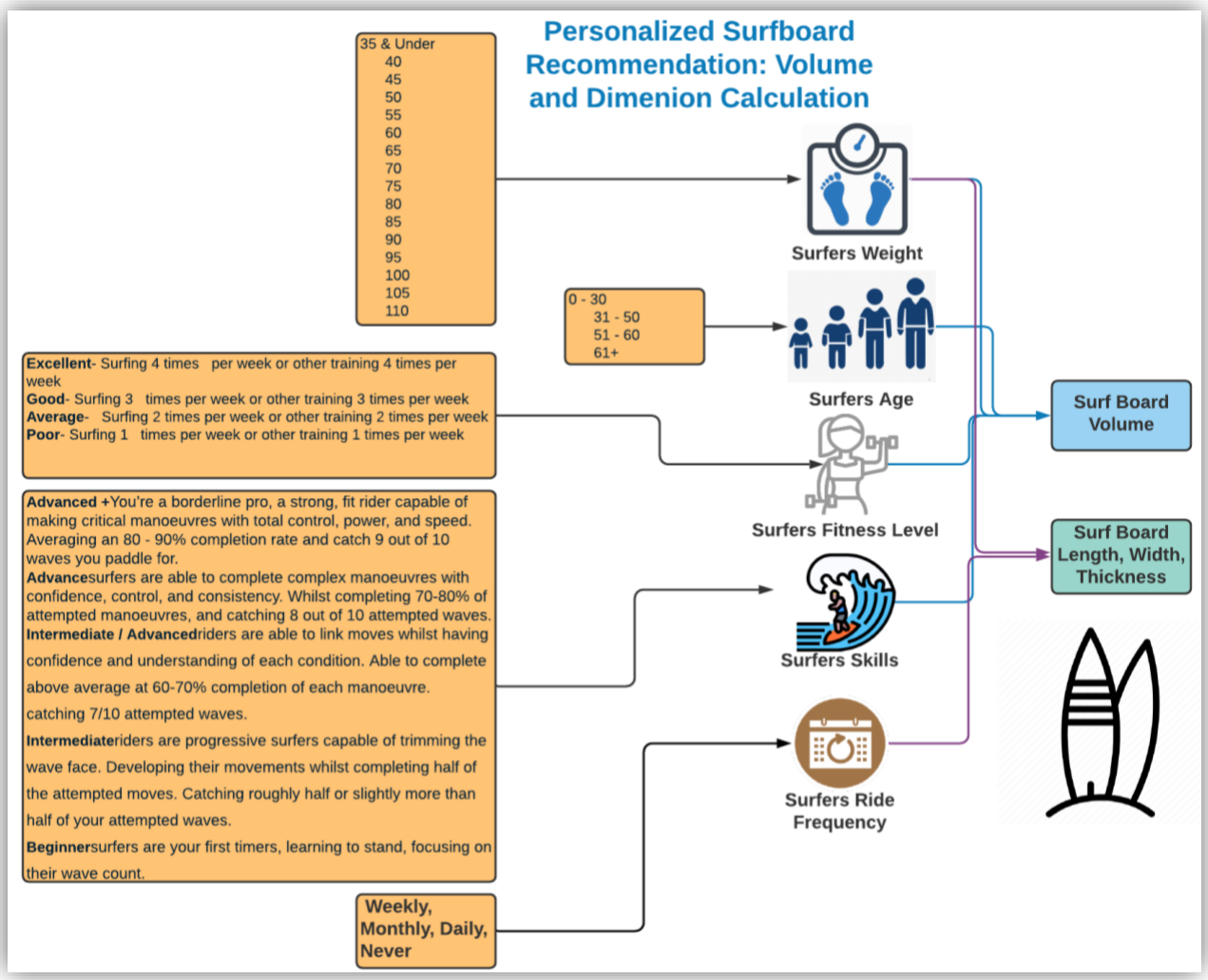

Figure 6: Personalised Surfboard Recommendation Flow

- We have lookup tables which relates Surfboard dimensions with surfers' weight and surfers ride frequency.
- We have lookup tables which relates Surfboard Volume with Surfers' weight, age, fitness level and skills.
- Hence, we calculated personalized surfboard for users considering physical parameters and skills.

2.2.4 Tailored Surfing News

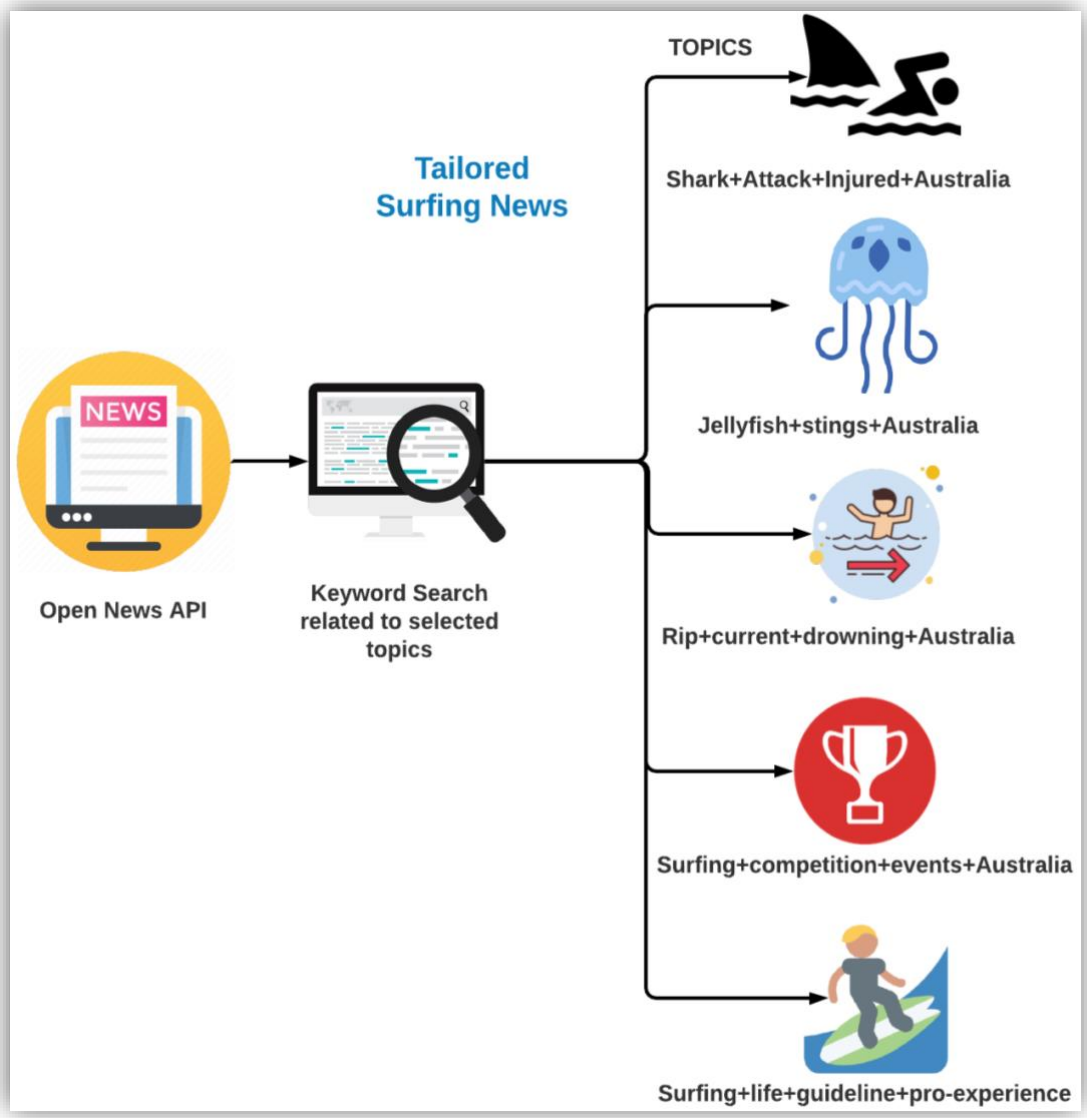

Figure 7: Tailored Surfing News Flow

- We did research and found list of topics which surfers of all experience level will be interested into like Shark Attack, Jelly fish and Rip current (reason behind many drowning cases in Australia) which can give word of cautions to all surfers. On other hand topics like Surfing Life and Surfing Competitions which can motivate surfers about what's happening in surfing world.
- Using the above topics, we made list of relevant keywords for each topic.
- We filtered incoming news using keyword searching to make the news more relevant to selected topics.
- Once we have all relevant news, we inserted into Postgres DB.

2.2.5 Emergency Service Support

- We have open dataset which consists of all hospitals in Australia with responsiveness score (Percentage of patients seen on time in the past during urgency).
- Out of that we selected only hospitals in Victoria.
- Using Google geocoding API, we fetched formatted address and coordinates for all hospitals.
- Using Google places API, we fetched hospital information like website link, phone no., Google rating, etc.
- Using open dataset, we have list of all services which hospitals provides.

2.3 Entity Relationship Diagram

2.3.1 Shark Attack and Sighting Prediction – ER Diagram

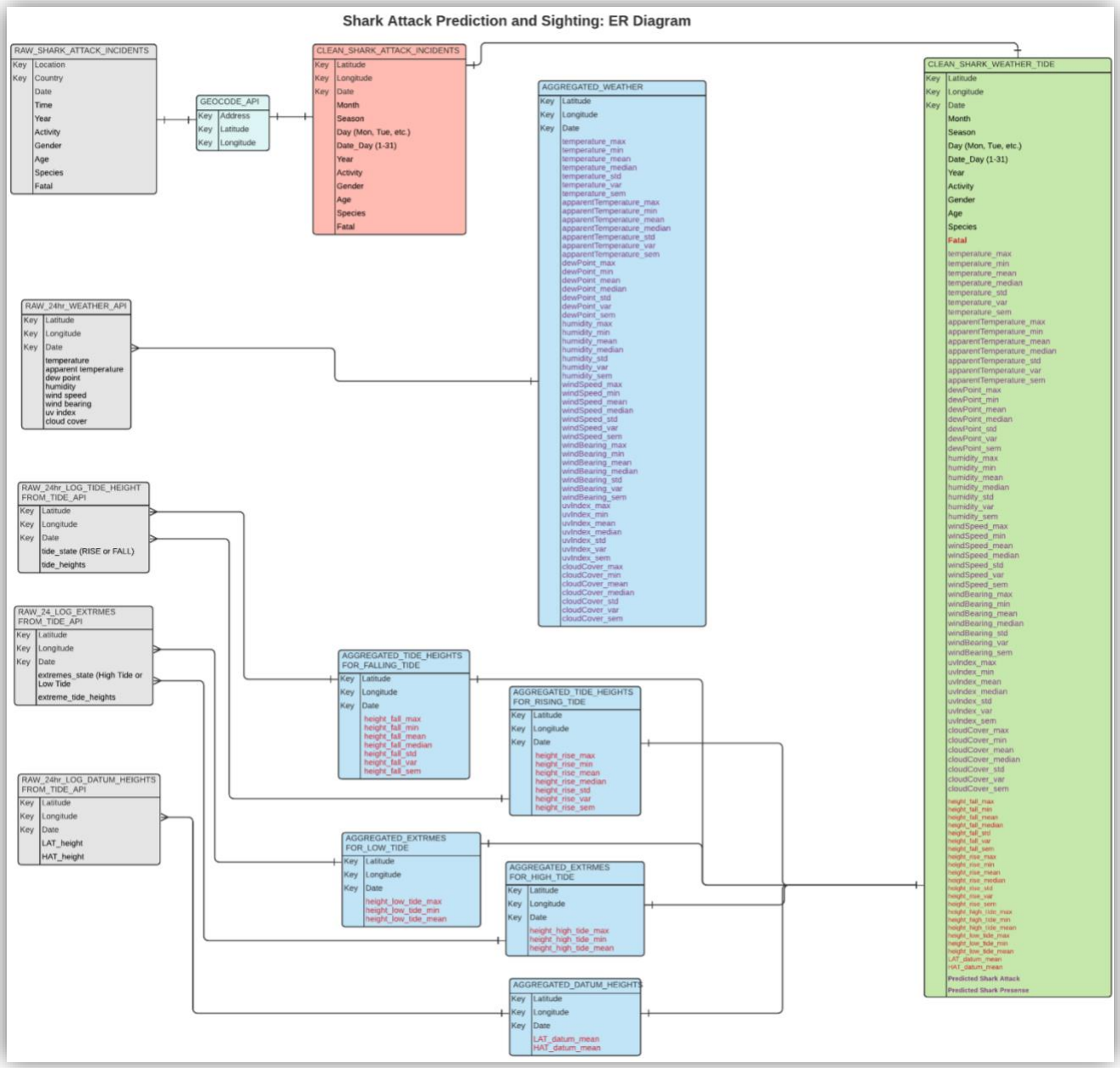

Figure 8: ER Shark Attack and Sighting Prediction ERD

2.3.2. Iteration 2 – Epics 3, 4, 5: ER Diagram

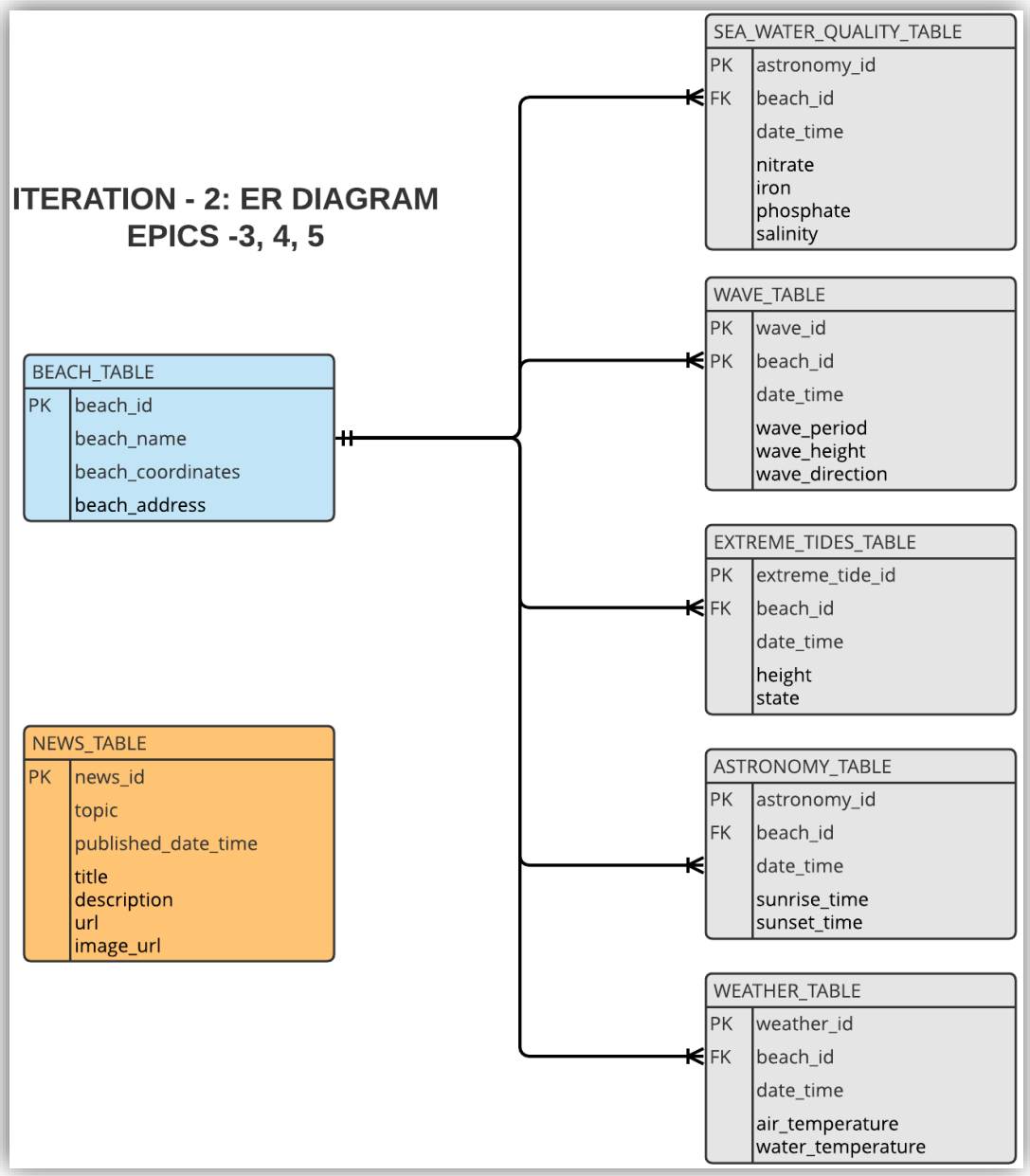

Figure 9: Iteration 2 ERD

2.3.3 Emergency Support Service – ER Diagram

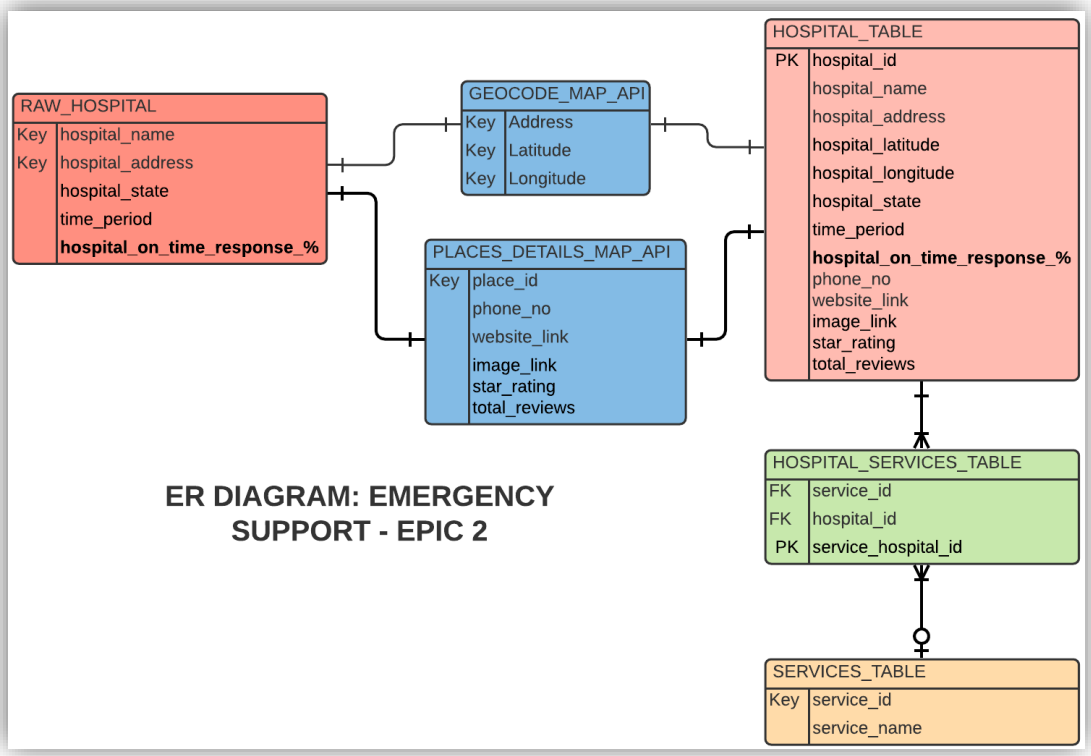

Figure 10: Emergency Support Sevice ERD

2.4 Use Case Diagram

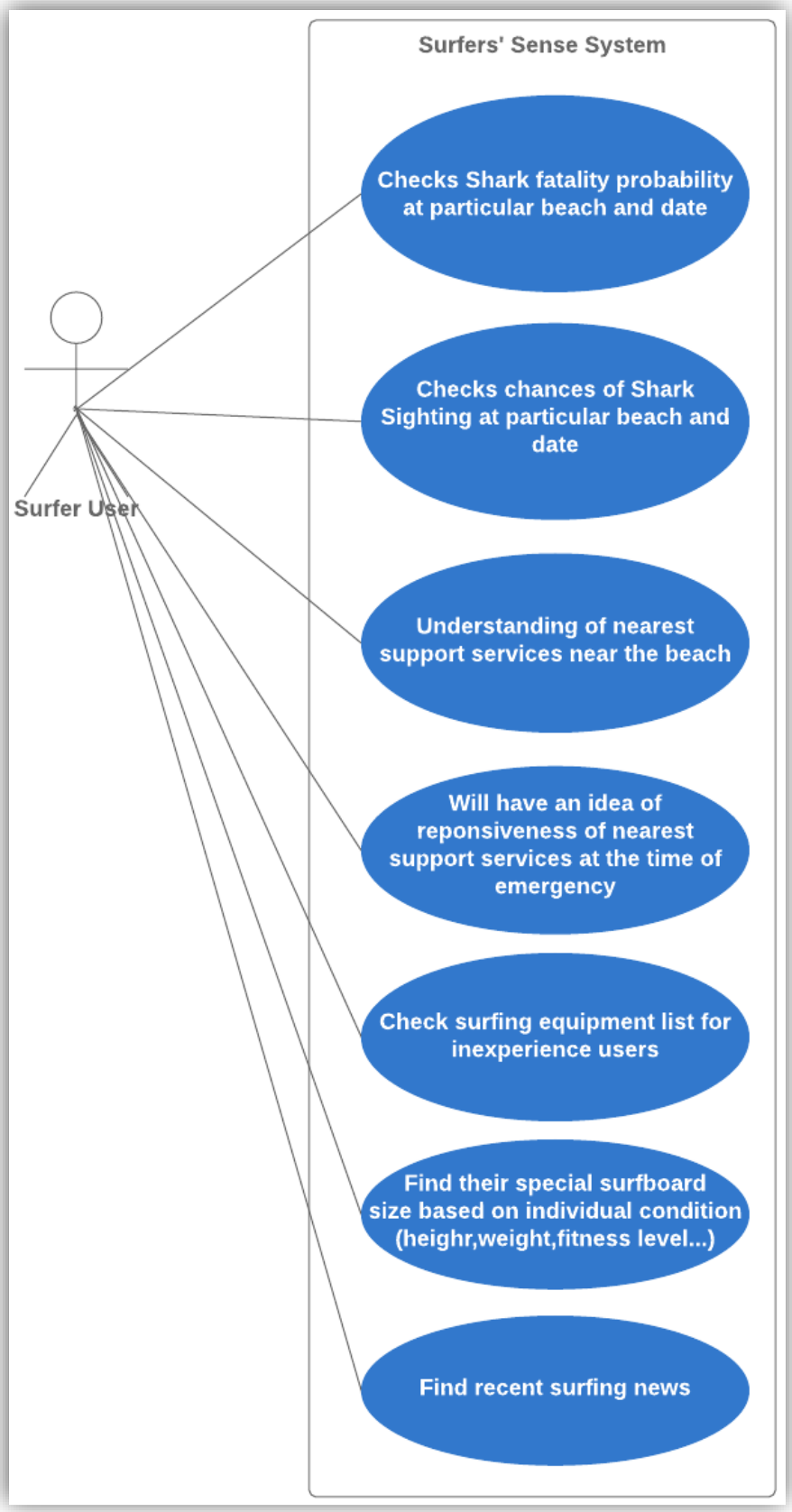

Figure 11: Use Case Diagram

3 Security

3.1. Internal Access Authority

- To do effective version control of the system and avoid any potential disorders, the internal access authority will be identified.
- Only the solution architect will have access to system's root folders on premise or cloud portal.

3.2. Access to Data

- Surfers' Sense database is physically segmented from the application layer.
- Only the application server is allowed to send requests and access data in the database.
- All other accesses have been prohibited through explicit firewall rules on the database server.

3.3. Measures

- SQL injection is possible as we have fields where the user will be entering the location or address of their house and the radius in which they want us to search the best beach for them. Raw SQL queries or scripts through one of these fields on site can be harmful and will be handling this by using POST requests and by limiting the database access to the application.
- Only developers should have access to the server for regular update purposes and needs to happen via SSH protocol to protect the prediction model and various datasets. As development will be carried on local computers and later deployed on the server.
- User Data will be critical for the recommending and this data will include address, name, radius, etc. This data will be transferred using SSL security protocol to the server.
- As users are entering information on the site, Cross-Site Scripting can be used to access personal information of the user. To prevent this all the codes that will be accessing the input fields will be made as explicit as possible.

4 Equipment and Computing Environment

- Surfers' Sense utilizes multiple web technologies and programming languages to offer the services listed above.
- However, it does not need any special hardware equipment to deliver on its functionalities.
- Surfers' Sense is currently deployed as an Application service on AWS EC2 instance.
- The current environment details and dependencies are listed below:

4.1 AWS EC2 for Django Server (Web application Deployed) and Scheduler Scripts – Free Tier:

- OS: Ubuntu Server 16.04 LTS (HVM)
- System Architecture: 64-bit x86
- Type: t2.micro
- RAM Size: 1 GB
- SSD Size: 8 GB
- Processor: 1.6 GHz
- Bandwidth: 165 MB

4.2 AWS RDS – Postgres Database – Free Tier:

- Allocated Storage: 20 GB
- Engine type: Postgres SQL
- 1 CPU
- RAM Size: 1 GB
- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro Instance
- 20 GB for automated backup storage and any user-initiated DB Snapshots.

4.3 Deploying Surfers' Sense Website in AWS:

Step 1: Create a account on AWS Free Tier and launch basic t2.micro instance with Ubuntu 18.04 Amazon Machine Image. (link)

Step 2: In the Security session download a new key pair for connecting with the instance. For more details follow this link.

Step 3: Create a python virtual environment using the command python3 -m venv env

Step 4: Activate the virtual environment using the command source env/bin/activate.

Step 5: Install Django, psycopg2, pandas, numpy, and gunicorn using pip3 install library name.

Step 6: Clone the repository from GitHub using the link.

Step 7: Install Nginx using the following command: sudo apt-get install -y nginx

Step 8: Add inbound rules for the EC2 instance using following link

Step 9: Install Supervisor using the following command: sudo apt-get install -y supervisor

Step 10: After step 9, create a gunicorn.conf file in the folder /etc/supervisor/conf.d using nano command. Add the configuration details given in the below snapshot. Here the AwsDemo has to be replaced with surfers_bible/Website_iteration2/iteration2 and TestProject.wsgi has to be replaced with surfers_bible/Website_iteration2/iteration2 iteration2.wsgi.

Step 11: Create folder using the command sudo mkdir /var/log/gunicorn

Step 12: run following commands simultaneously sudo supervisorctl reread and sudo supervisorctl update.

Step 13: Create nginx configuration file in /etc/nginx/sites-available using command sudo nano Django.conf

Step 14: Inside django.conf file write the following lines of code. Here the AwsDemo has to be replaced with surfers_bible/Website_iteration2/iteration2 and after server_name give the IP address of the instance or the website name.

Step 15: once the configuration is saved, hard link is using following command sudo ln django.conf /etc/nginx/sites-enable

Step 16: The following commands simultaneously sudo nginx -t , sudo service nginx restart and visit the website link in the browser.

5 Software

The following tools are required to setup development environment on a standard Linux Machine.

5.1. Technology Stack

Project Section	Technology Stack	Version
Data Science	Python - Anaconda and Jupyter Notebook	3.6
	Postman (Collaboration Platform for API Development) – Creating API request in required format	--
Back -End	Django (Work of loading html files and connection to the database)	3.0.7
	Nginx (Gives does all the redirection to and from the website)	1.19.0
	Gunicorn (Used for connecting with the wsgi HTTP servers and unix)	20.0.4
	Supervisor (Technology that keeps running all of above in backend)	4.2.0
Front End	HTML	5
	CSS	4
	Bootstrap	4.5
	JavaScript	9
	Bootstrap Studio	5.1.0
	ATOM text editor	1.48.0

5.2. Python Data Science Libraries and its purpose in the project

Python for Data Science Libraries	Purpose
Pandas	Data Wrangling, Data Transformation, Dealing with Data Frames
NumPy	High-performance multidimensional array and basic tools to compute with and manipulate these arrays
Matplotlib	Plotting Graphs to interpret results and testing and Exploratory Data Analysis
Sklearn	Machine Learning Pre processing tasks like Train Test split, K Fold Validation, to calculate cosine similarity for Shark Sighting Probability etc.
Requests	For making HTTP request for all API and getting response
JSON	Converting HTTP response JSON to python dictionary to parse the data.
re	Python Regex for data parsing and cleaning
Datetime	To deal with datetime columns and also to read system datatime for scheduler.
Dateutil	To convert datetime from one time zone to another time zone, e.g.: All API responses is in UTC so I need to convert it to AEST time.
Catboost	Machine Learning Model used to predict Shark Attack Probability
psycopg2	Python driver for PostgreSQL, to write SQL statements in Python, Connecting to RDS DB, commit, execute SQL, etc.

5.3 Technical Expertise Needed- Operations and Developer:

5.3.1 Developer Team:

Position	Skills
Data Scientist	Python, Machine Learning, Data Cleaning, Deep Learning, Data Visualization, Statistics, SQL
Back-End	Django, Python, AWS, NGINX, RDS, Postgres, SQL
UI Developer	HTML, CSS, JAVASCRIPT, BOOTSTRAP
UX Developer	Design Thinking, UX writing, User empathy, UX research.

5.3.2 Support Team:

Position	Skills
DB Admin	Postgres SQL, AWS, RDS
Tech Support	Python, AWS

6 Database Details

6.1. Dataset and APIs sources and descriptions

Access link: https://drive.google.com/file/d/1Ac0Hb0YgjQXEczloR0M-C7mOG54NtkGy/view?usp=sharing

6.2. API Request Management within usage limit

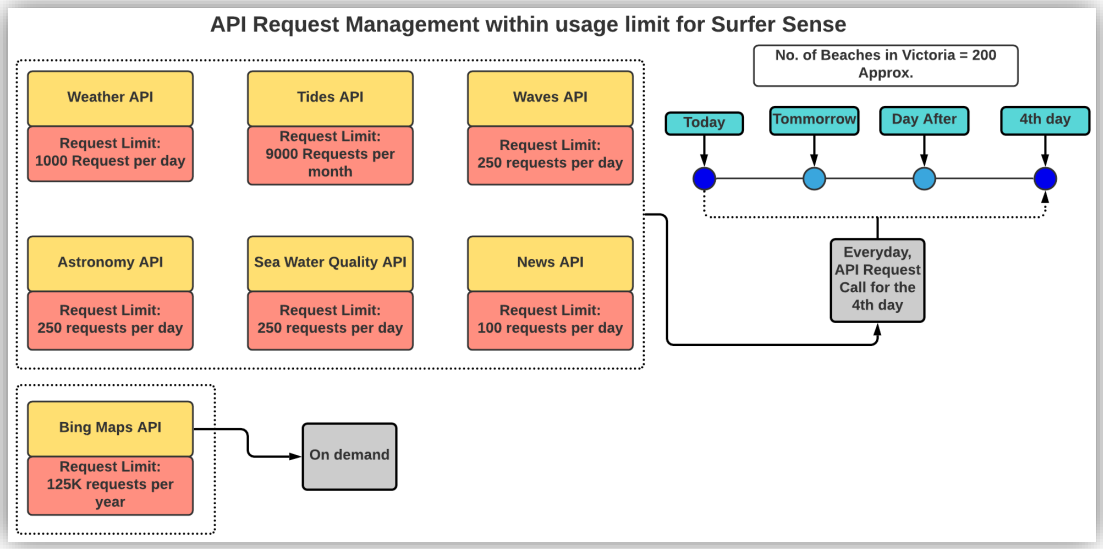

Figure 12: API Request Management

The above figure explains how API requests to populate database is managed in a such a way that it is used within free usage limit.

APIs with everyday update frequency:

- After thorough research and many trial and errors (I have spent like 3 days to come to this conclusion) we made all APIs like weather, tides, waves, astronomy, sea water quality and news in such a way that their update frequency in the system is daily keeping in mind usage limit of all APIs (Very tricky!).
- The trick was all the above-mentioned APIs will ask for data of the 4th day from the day it requests the data. For example: Asking date is 7th July then it will fetch the data of 10th of July, if asking data is 8th July then it will fetch the data of 11th July and so on.
- In this way we made all API compatible with same update frequency and secondly in this way we were able to make forecast of coming 3 days including today.

APIs with on demand update frequency:

- Bing Maps API – Request limit 125K requests per year: It was used in two cases one in Explore beach page and other in Emergency Support Service page.
- Usage in Explore beach page: When user clicks on submit then it fetches the user's current location from the browser and then calculates all distances from user's current location to all 200 beaches location and arrange in ascending order of distances.

- Usage in Emergency Support Service: When user clicks on allow location in Emergency Support Service then it fetches the user's current location from the browser and then calculates all distances from user's current location to all Victorian Hospital locations and arrange in ascending order of distances.

6.3 Adding API Keys:

API No.	Purpose	API Source	Scripts
1	**Weather**	**Darksky**	**Daily_Scheduler_Weather_Tide_API.py**
2	**Tide**	**Tide Hood**	**Daily_Scheduler_Weather_Tide_API.py**
3	**Sunrise Sunset**	**Storm Glass Astronomy**	**Daily_Scheduler_Swell_Pollution_Astro_Update.py**
4	**Wave Report**	**Storm Glass Weather**	**Daily_Scheduler_Swell_Pollution_Astro_Update.py**
5	**Water Quality**	**Storm Glass Biology**	**Daily_Scheduler_Swell_Pollution_Astro_Update.py**
6	**Custom News**	**OpenNews**	**Surfing_News_Scheduler.py**
7	**Distance Matrix**	**Bing Maps**	**View.py (Inside Shark function)**

7 Testing

7.1. Usability Testing

- Usability tests were conducted with multiple users after development of each iteration to determine bugs, flaws and ways to improve the system.
- A final usability testing was also done with multiple users after finishing all the iterations.

Testing Hours	Tester	Testing Type	Feedback	Recording Link
20 April 2020 13:00	**Jadzia**	UI Testing	Feedback: The color is look good and match the topic.	https://screencast-o-matic.com/watch/cYfYeGzd6o
27 April 2020 15:20	**Anthony**	Usability Testing	Feedback: **UI:** lovely website, I barely forget what my purpose is but indulge myself in this succinct and beautiful page. **UX:** perfect, I don't have to click too much. **Function:** we barely pay attention to these graphs, just basic information about the possibility of being attacked and the presence is ok. But, the phone number of hospitals is missing. What should I call?	https://screencast-o-matic.com/watch/cYfthjAd8f
18 May 2020 17:00	**Satabdi Dash**	Usability Testing	Feedback: Liked the entire concept of planning surf journey.	https://cloudstor.aarnet.edu.au/plus/s/jCGcf5N1AukE3Nf Password: **rLaTra^xqT9d**
18 May 2020 17:00	**Suraj**	Usability Testing	Feedback: Very easy to use, I use mobile to check it, cool!	https://drive.google.com/file/d/1BseD80RIMT8Xj7Ks1EBgTQCrlfHdPYDe/view?usp=sharing
20 May 2020	**Livia**	Usability Testing	Feedback: The things that need changes that are apparent (for example the use of checklist and the prediction indicators as discussed during our feedback session as well) need to be revised on priority, so that the user may not get confused. The UX needs to be such that the flow allows the user to realise all key functionality of the website and the last thing the user does is probably go to surf using the maps.	https://drive.google.com/open?id=130MlvIuOoTTDTcfkBMRIYjUNpCYHYUQ9
21 May 2020 15:20	**Anthony**	Usability Testing	Feedback: The website gave me a new idea of surfing, I have not tried it before, but I want to study surfing now!!	https://screencast-o-matic.com/watch/cYhrhFkJNu

21 May 2020 15:50	**Jerry**	Usability Testing	Feedback: why I need to reauthorize my location when I need an emergency service based on the beach I go not where I used my laptop or phone. Others are pretty cool.	https://screencast-o-matic.com/watch/cYhrhTkJ8d
21 May 2020 16:00	**Andy**	Usability Testing	Feedback: The whole interface is good, but some functions are not realized, maybe because of the network issue.	https://screencast-o-matic.com/watch/cYhr1QkdVc
22 May 2020	**Lila**	Usability Testing	Feedback: Details need to be improved to provide more value for users.	https://cloudstor.aarnet.edu.au/plus/s/hL8QbixqRCRaqVy Password: **m8DHANrxiDQ=**
23 May 2020	**Pruthvi Raju**	Usability Testing	Feedback: Some details need make improvement, but the idea is amazing.	https://cloudstor.aarnet.edu.au/plus/s/lZWSYJJC0oBuBAy Password: **Hui*LVxW1sko**
23 May 2020	**Dhananjay Adodra**	Usability Testing	Feedback: I like the website and I think it's useful for me before planning my travel.	https://cloudstor.aarnet.edu.au/plus/s/DS04eiHNervHA6I Password: **yhZugZ=YpC1Q**

7.2. Backup and Restoration Testing

GitHub was used as version controlling tool while development of Surfers' Sense and full back-up of the final version of Surfers' Sense is available to be downloaded from GitHub at the following URL:

https://github.com/Akshay-Ijantkar/surfers_bible

7.3. Integrity/Acceptance Testing

Access link: https://drive.google.com/file/d/17yW3G9ObTx-nhXnY-KbnksABMlaEsRhA/view?usp=sharing