Contents

[Route Map 2](#_Toc15371121)

[High level Idea 2](#_Toc15371122)

[Phase 1: 2](#_Toc15371123)

[Phase 2: Additional Data Integrations 2](#_Toc15371124)

[Phase 3: Missing People Tracker 2](#_Toc15371125)

[Nice to have 2](#_Toc15371126)

[Problem: 4](#_Toc15371127)

[Solution: 4](#_Toc15371128)

[Options: 4](#_Toc15371129)

[Minor Problems and solutions: 4](#_Toc15371130)

[Uses: 5](#_Toc15371131)

[Phase 1: Identify Flood Level and send alerts 5](#_Toc15371132)

[Processing Data 5](#_Toc15371133)

[Phase 2: Additional Data Integrations and Uses: 5](#_Toc15371134)

[Phase 3: Missing People Tracker 6](#_Toc15371135)

[High Level Architecture 6](#_Toc15371136)

[Detailed Architecture 6](#_Toc15371137)

[Technical Details 7](#_Toc15371138)

[Phase1: Image Categorization and Alerting. 7](#_Toc15371139)

[Flow Chart for flood level detection: 8](#_Toc15371140)

[Flow Chart for Navigation Assistance: 9](#_Toc15371141)

[Data Model 10](#_Toc15371142)

[Videos 10](#_Toc15371143)

[Introduction video of 3 mins 10](#_Toc15371144)

[Introduction Video 10](#_Toc15371145)

[Project & Technical Demo and Short Demo 10](#_Toc15371146)

[Image Analysis Demo using Python 10](#_Toc15371147)

[Watson Assistant and Facebook in Action part 1 and part 2 10](#_Toc15371148)

# Route Map

|  |  |
| --- | --- |
| Phase 1:  i. Identify Flood Level  ii. Send alerts (🖄🖅/Email/SMS/Whatsapp)  iii. Build a mapping solution  iv. Provide Navigation Assistance on Facebook Messenger. | Done (Used Watson Visual Recognition)  Done (Used MQTT/simple python and Twilio)  Done  Done (Used Watson Assistant and Facebook Messenger) |
| Phase 2: Additional Data Integrations  i. Find/Track Rescue centers/temporary rehabilitation camps and provide info on chat.  ii. Monitor Social Media and New platforms  iii. Identify open ATMs/Commercial Places  iv. Integrate Satellite/Radar/Drone info.  v. Analyze Image for background objects like famous buildings/structures. | Future Implementation |
| Phase 3: Missing People Tracker | Future Implementation |
| Phase 4: Nice to have | Future Implementation |

# High level Idea

The important goal is to use the existing resources as much as possible, from Traffic Cameras to Facebook Messengers.

### Phase 1:

#### Identify Flood Level

Flood level detection by looking at height of submerged electric poles (using color code), traffic cones, road dividers, distance markers, standard vehicles (typres/fully submerged/no flood), People (ankle deep, waist deep).

Traffic cameras, security cameras can be used to collect info.

Analyze the depth of water level.

#### Send Alerts

Plot a route from point A to point B based on mode/vehicle type and water depth.

#### Build Mapping Solution

Identify the shortest path of water to water outlet like a river to pump out water

Send data to Govt. Authorities, volunteer team by email/sms/whatsapp.

People can provide images too, and they can be identified from GPS location or well-known buildings/structures.

#### Provide Navigation Assistance.

Using Facebook Messenger people can reach out to Watson Assistant to get flood level information.

### Phase 2: Additional Data Integrations

i. Find/Track Rescue/temporary rehabilitation camps centers and provide info on chat.

ii. Information can be pulled from online search/news sites/social networking.

iii. Identify open ATMs/Shops/ Medicine Stores, and share the info

iv. Build a solution to get data from Satellite/Radars/Drones and add the information to phase 1 and process it.

v. Analyze Image for background objects like famous buildings/structures/distance/kilometer markers/direction boards to find the exact navigation point.

### Phase 3: Missing People Tracker

i. GPS like identifier to track the people willing to help and needing help. Location sharing manually.

ii. Identifying people from facial recognition when they are in a crowd, thus allowing authority to trace missing people complaints.

### Nice to have

Find potential areas where tree branches or such things can cause power disruption.

Water/Wind speed calculation from moving objects.

Fire intensity calculation using temperature sensitive cameras (difference in temperature to try and identify life).

# Problem:

In India, people go to different places to estimate the impact/damage during disasters. Higher authorities use Helicopters to estimate the same. While the prior is time taking, the later is costly and not so accurate. Both have a major flaw; they are not real time.

# Solution:

If we can have a system that can give us continuous data regarding an issue, it will be much helpful in planning.

## Options:

1. We can use drones to estimate how much area is covered in water (or snow). This would work when the weather is not extreme. This way needs people to operate drones, and yet will not be able to provide continuous data. Flying drones all the time is a costly affair too.
2. In modern world, cameras are there in various places. From traffic cameras, security cameras in malls, ATMs, Stadiums, some Schools, Function halls, etc., to people carrying cameras. Even forest officers deploy cameras in different parts of the forest to track poachers. If we can get data from them, we can continuously monitor the situation.
3. Drones can still be used as all areas may not have the required method of capturing or sending data. Additionally, it can be used to spot people on roof tops and plan rescue operations.

|  |  |
| --- | --- |
| 🛩🛰 | 📷 |
| 🕴🚍⛳🚘🏍 | |

### Minor Problems and solutions:

1. In extreme weather conditions, there is a chance for *power outage*. Some traffic cameras are being equipped with solar panels. Malls/ATMs, etc. have their power backups. More solar panels are needed.
2. Networkcould also be a problem. We can use a way to send the data when common network solution fails. Use of radio waves is a possible solution. Project Owl uses drones. Instead, if we can have repeaters on traffic posts, electric posts, etc., it would serve as a backup. Some ATMs generally keep working in extreme weather conditions too.

## Uses:

Municipal authorities, sanitary workers, traffic department and many other people can look at this real time data and advice common people/volunteers/trapped people to the best course of action.

If the data is real time, municipal authorities can take preventive steps to clear the blocks and reduce the effect.

Forest official can take action to track and vacate trapped animals.

Electricity Boards can monitor areas where power is out, plan a path and attend to them. Preventive steps like nests/dead trees/branches, that may impact during rains can be found and fixed. Further if we can find areas of power leaks, they can be fixed.

But people can’t keep analyzing pictures all the time.

## Phase 1: Identify Flood Level and send alerts

So if we can have a system that analyzes water levels in real time, it would help even if the disaster is not predicted, for example leakage from canals, dams, accidental fires in remote areas, etc.

*How do we analyze the data (images or videos)?* We can use modern technology to feed the images/videos to cloud and data be analyzed and sent to relevant authorities.

*How do we find out the water level in an area?* If we are to consider traffic cameras, then they take images of traffic, and they cover vehicles. From the level of water of standard vehicles like Bus, Trucks/Lorries to cars, for example, the tyres, etc., we can tell how much water is on the road. Water at body level of people can also serve as a source.

But what if the road is *deserted?* We can calculate water level from traffic cones, road dividers, signal posts, electric poles and other non-moving objects.

*What about forests?* Trees need to be marked to aid this. Or we can use *comparative* logic to identify the levels.

While tyres of standard vehicles like Bus have a specific dimension, Traffic cones, road dividers, and signal posts are marked. Using this we can find out the level of water.

*Use of Geometry/Trigonometry?* While this is a solution, we can train AI on several use cases and see if that is feasible.

Some advanced AI like PowerAI is capable of identifying objects in images and its movement in videos.

*What if traffic cameras are not there?* We can take feed from *people* too, to add to our data. GPS and time and date marked images would be best.

But sometimes *GPS is inaccurate*. In such cases, if we can analyze the objects in background, like well-known/unique buildings it will help.

### Processing Data

Data needs to be sent to Govt. Authorities/Media in emails, to help them help others, and trapped people trying to escape the disaster. SMS/Whatsapp can be sent to Volunteers/Victims.

We need to plot the captured and processed data on maps to benefit the users.

Facebook Messenger can be used to provide chat assistance to users.

Water levels in stadiums/schools/function halls can be tracked and rescue centers, temporary rehabilitation camps can be established.

Once data is captured, we need to use it efficiently. If we can overlay this data on common available tools like google maps, it will help people in time of need.

Sending Alerts  
Based on results provided by Watson Visual Recognition system the solution is able to send alert:  
• To any navigation application that subscribes to the provided alert service – providing the flood geolocation, and by extend the application can provide an itinerary to avoid floods  
• To messenger application, based on Watson ChatBot providing advices to people to avoid floods  
• To different relevant authorities, by sending emails or using any other channels to contact them

## Phase 2: Additional Data Integrations and Uses:

We can encourage shops/group of shops to use cameras, so that, in time of disaster, if *shops are open*, people can find out that their shops are open for business. They can help even if when they are closed, by marking the levels on walls from road level.

This can be applied to ATMs too. If we have standard *ATM messages* (like “Cannot connect to server”, “Out of money”, etc.) in front of ATM cameras, people can find out the nearest functional machine and go to it. ATMs giving status by self-diagnostic is a possibility.

Other sources of data? If we can source data from news channels, online search, Social Media, etc., it would be beneficial too.

## Phase 3: Missing People Tracker

During floods and evacuations, people are taken to different locations, and it becomes difficult to find family members.

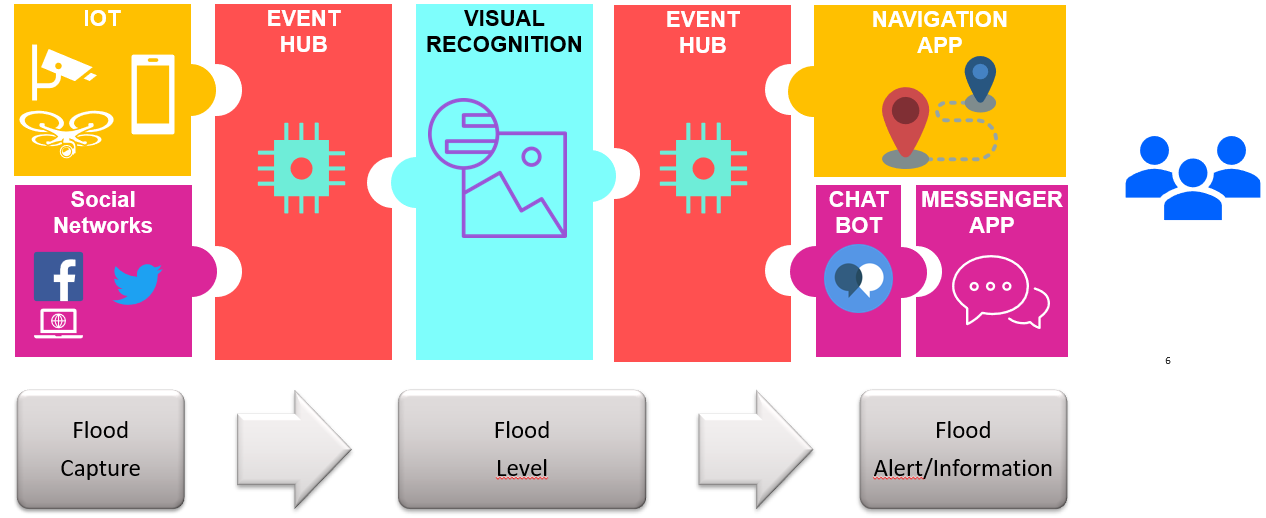
We can build an app to track the people using facial recognition.

Authorities can provide a photo of a person to the system, and over time, the system can identify and provide the information.

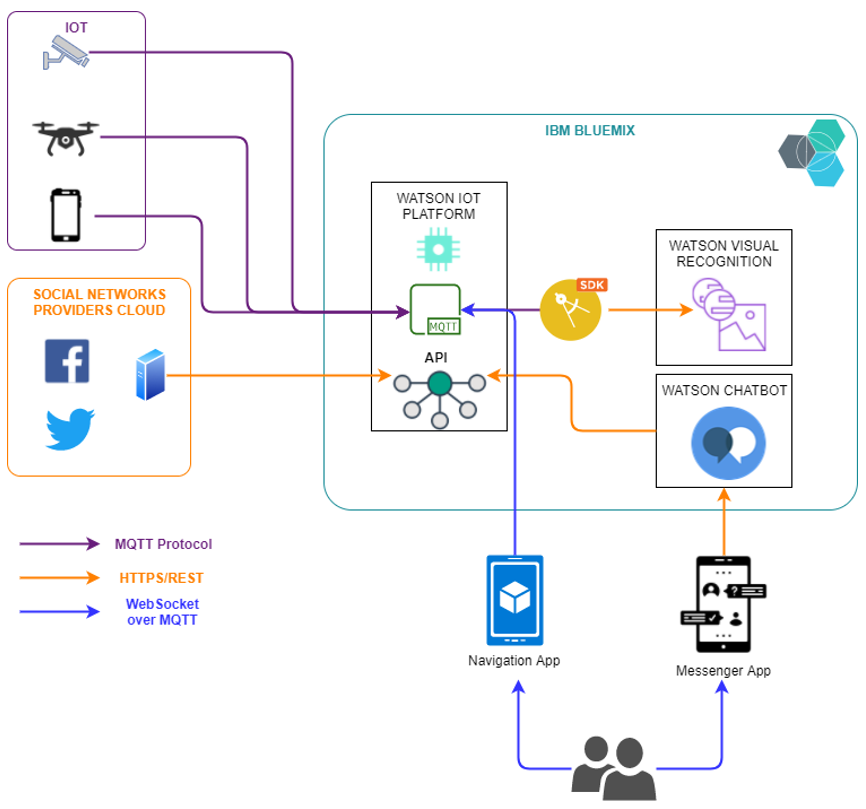
Alternative: For this, first, members of family can use an app to take images of their family members from different angels and upload them to cloud using codes. These codes could be *part* of name, id card, date of birth, email id, etc. This will help them to look for the missing people in times of need.

While, people should be able to provide the data to the system, getting data from the system could be given only to authority.

## High Level Architecture



## Detailed Architecture



# Technical Details

## Phase1: Image Categorization and Alerting.

Based on water at wheel level/human body level, I classified the classes in Watson Visual Recognition into the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Description↓/Severity🡪** | **No Flood** | **Manageable** | **Severe** |
| **Flood/No Flood 🡪** | noFlood | FloodArea |  |
| **Humans** | pPlNoFlood | pPlAnkleDeep | pPlWaistDeep |
| **Two wheeler** | 2WhlrNoFld | 2WhlrFld |  |
| **Three wheeler** | 3WhlrNoFld | 3WhlrFld |  |
| **Four wheeler light** | 4WhlrLightNoFld | 4WhlrLightFldTyre | 4WhlrLightFld |
| **Four wheeler heavy** | 4WhlrHvyNoFld | 4WhlrHvyFldTyre |  |

For each category, I added 10 or more, but in the negative category I added oceans, rivers, and this confused the model mainly in FloodArea. I have retrained it. Testing needs to be done.

Further, there is a minor difference in color of road and water, hence the model is getting confused sometimes. Additional training is required.



|  |  |  |
| --- | --- | --- |
| **Get Images 🡪** | **Image Analysis and Categorization 🡪** | **Flood level plotting** |
| Possible Source  Traffic Camera  Security Camera  ATM Camera  Pictures Taken from people  Online source  Store images in cloud | Get the images from cloud  Analyze it (using **Watson Visual Recognition**)  Classify/Categorize it | Get the details from json  Get further details like location  Plot the |
| What is the best place for devices/people to send images?  Can we have something like google drive/Facebook albums, etc? | What can we do to get better accuracy?  Adding more images may overfit/confuse the model.  Any alternative for Watson Visual Recognition? | We need to use google api |
|  |  | Assumption:  For now, Image received will have latitude/longitude to help us map the image. |

Navigation Assistance

|  |  |  |
| --- | --- | --- |
| **Get Source and Destination 🡪** | **Plot a map 🡪 Generate URL 🡪** | **Send URL** |
|  | We need to use google/other map api | Watson Chatbot and Facebook |

## Flow Chart for flood level detection:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Assumption** | **Prerequisite** | **Technology** |
| Get Photo from bucket  **↓** | Name of file is image\_<latitude>\_<longitude>\_<datetime>  This is important, till we get real time data. | Solution checks for images in bucket at a set frequency,  Frequency of check increases/decreases based on flood level.  Sends images to processed buckets into different classifier folders.  This can be used to retrain the system, based on feedback or manually. | Cloud storage  Buckets |
| Calculate Flood Level  **↓** | For now, No Flood, Tyre (2,3,4 wheeler), Ankle, Waist, Dangerous levels |  | Watson Visual Recognition |
| Store Flood level  latitude, longitude, datetime in database  **↓** | We may get regular feed in cities and less feed in villages.  So we need to give date time of data too. |  | Python/DB |
| Based on level alert authority  Alert will have Flood level, latitude, longitude, datetime.  Map with pins/markers advisable. | For tyre and ankle level local authority is fine. Next level in cc advisable.  Inform nearest local authority as well.  As level increases higher authority need to be involved.  Authority will be able to deal based on priority. | Store authority name by region in db.  In India, its Village, City, District, State | MQTT/Python |
| Captured data can be used to add to existing data  And future planning during similar situations. | Future implementation. |  |  |
| Feedback system to verify and authenticate findings. |  |  |

## Flow Chart for Navigation Assistance:

|  |  |  |
| --- | --- | --- |
|  | **Assumption** | **Technology** |
| Get Navigation request  **↓** | Format <latitude>\_<longitude>\_<datetime> | May be Watson Assistant? |
| Process the request from database  **↓** |  | Sujoy to decide  Mapping solution and programming. |
| Send/show data authority  with date time and map with flood level markers |  | URL in Watson Assistant.  Webpage? |
| Feedback system to verify and authenticate data | Future implementation. |  |

## Data Model

FloodLevel info is used to send email/sms/whatsapp to **Subscribers** based on there enroll type and range.

It is also used for **Navigation** Assistance on chat.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Subscriber**   |  | | --- | | SubscriberId | | SubscriberType | | Name | | EmailId | | Mobile | | Whatsapp | | Voice | | Enroll (true/false) | | EnrollId | | Level | | Latitude | | Longitude | | Range (in km) | | Review | | Comments | | **Enroll**   |  | | --- | | EnrollId | | EnrollType  Ex. Email  Sms  Whatsapp |   **Authlevel**   |  | | --- | | AuthlevelId | | AuthlevelType  Ex. Govt. Authority,  Media,  Common People | | **FloodLevel**   |  | | --- | | TrackId | | ImageId | | Level | | Latitude | | Longitude | | Datetime | | Review | | Comments | | **Navigation**   |  | | --- | | Request\_Id | | Source\_Latitude | | Source\_Longitude | | Destination\_Latitude | | Destination\_Longitude | | Datetime | | Review | | Comments | |

# Videos

Please click on the description to launch the videos.

## [Introduction video of 3 mins](https://www.youtube.com/watch?v=8JRE7RArNl4&list=PLJZ7tzQf5MnhxtHVsozfgHsyf3MFtvmxy&index=8&t=0s)

<https://youtu.be/FysReNaKB0U>

## [Introduction Video](https://www.youtube.com/watch?v=CgMHSGOgnBg&list=PLJZ7tzQf5MnhxtHVsozfgHsyf3MFtvmxy&index=2&t=0s)

<https://youtu.be/CgMHSGOgnBg>

## [Project & Technical Demo](https://www.youtube.com/watch?v=b27EbEaVbZc&list=PLJZ7tzQf5MnhxtHVsozfgHsyf3MFtvmxy&index=5&t=0s) and [Short Demo](https://www.youtube.com/watch?v=1JtI4xT7ii8&list=PLJZ7tzQf5MnhxtHVsozfgHsyf3MFtvmxy&index=9&t=0s)

<https://youtu.be/b27EbEaVbZc> <https://www.youtube.com/watch?v=1JtI4xT7ii8&feature=youtu.be>

## [Image Analysis Demo using Python](https://www.youtube.com/watch?v=yD0ZUwU4pRU&list=PLJZ7tzQf5MnhxtHVsozfgHsyf3MFtvmxy&index=7&t=0s)

<https://youtu.be/yD0ZUwU4pRU>

## [Watson Assistant and Facebook in Action part 1](https://www.youtube.com/watch?v=Vir_ZGqQ_g4&list=PLJZ7tzQf5MnhxtHVsozfgHsyf3MFtvmxy&index=3&t=0s) and [part 2](https://www.youtube.com/watch?v=U40FcEYIrlg&list=PLJZ7tzQf5MnhxtHVsozfgHsyf3MFtvmxy&index=4&t=0s)

<https://youtu.be/U40FcEYIrlg> <https://youtu.be/Vir_ZGqQ_g4>