“Unstable Bluff” Detection System Design Specification

Prepared by: “The Bluffers” - Samantha James, Joshua Olaoye, Antonio Rivera

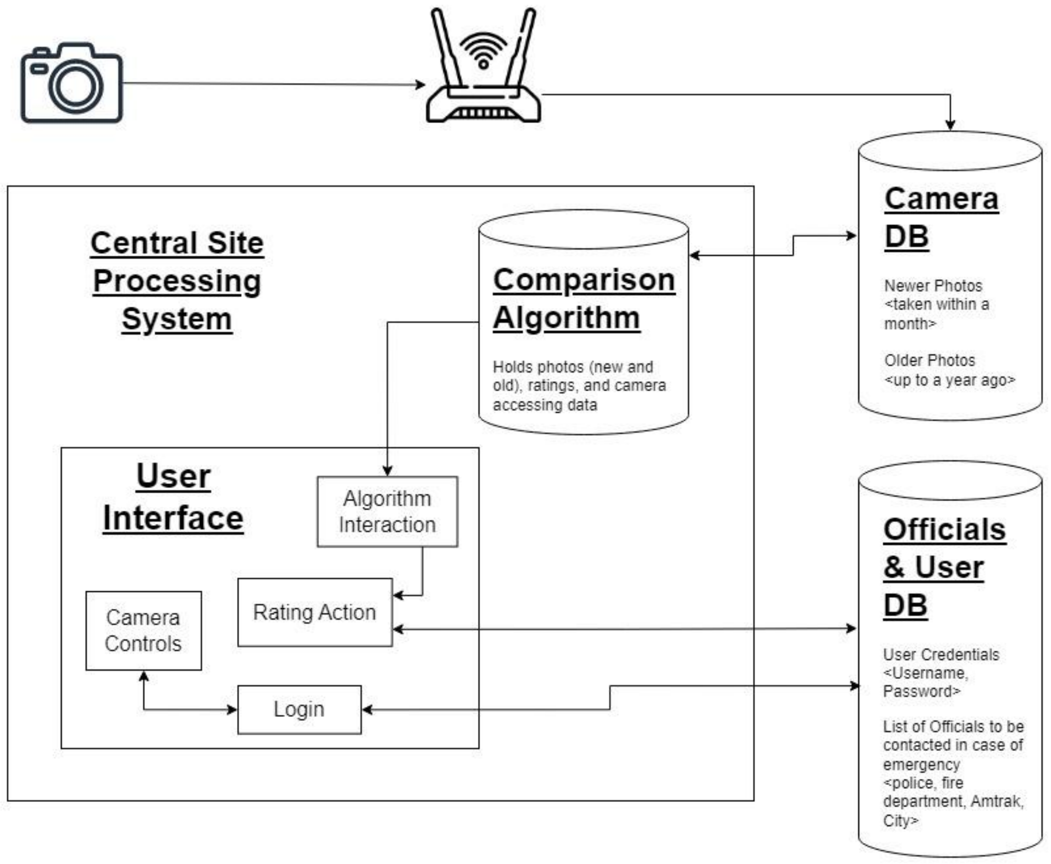
March 10, 2023

System Description

There are unstable bluffs, which are steep cliffs or banks, in the city of Del Mar along California's southern coast. A heavily trafficked train's tracks are located on a portion of these bluffs, which is very dangerous if these bluffs' stability is not monitored, potentially leading to injury and/or death of those on the train as well as people on the beach below the bluffs. This camera-based monitoring system's purpose is to constantly monitor the bluffs and detect when areas become unstable and alert authorities so that safety precautions can be implemented. This document will outline the software and architecture necessary for designing and later implementing this bluff monitoring system, which is divided into sections of software architecture overview and development plan and timeline.

Software Architecture Overview

Software Architecture Diagram

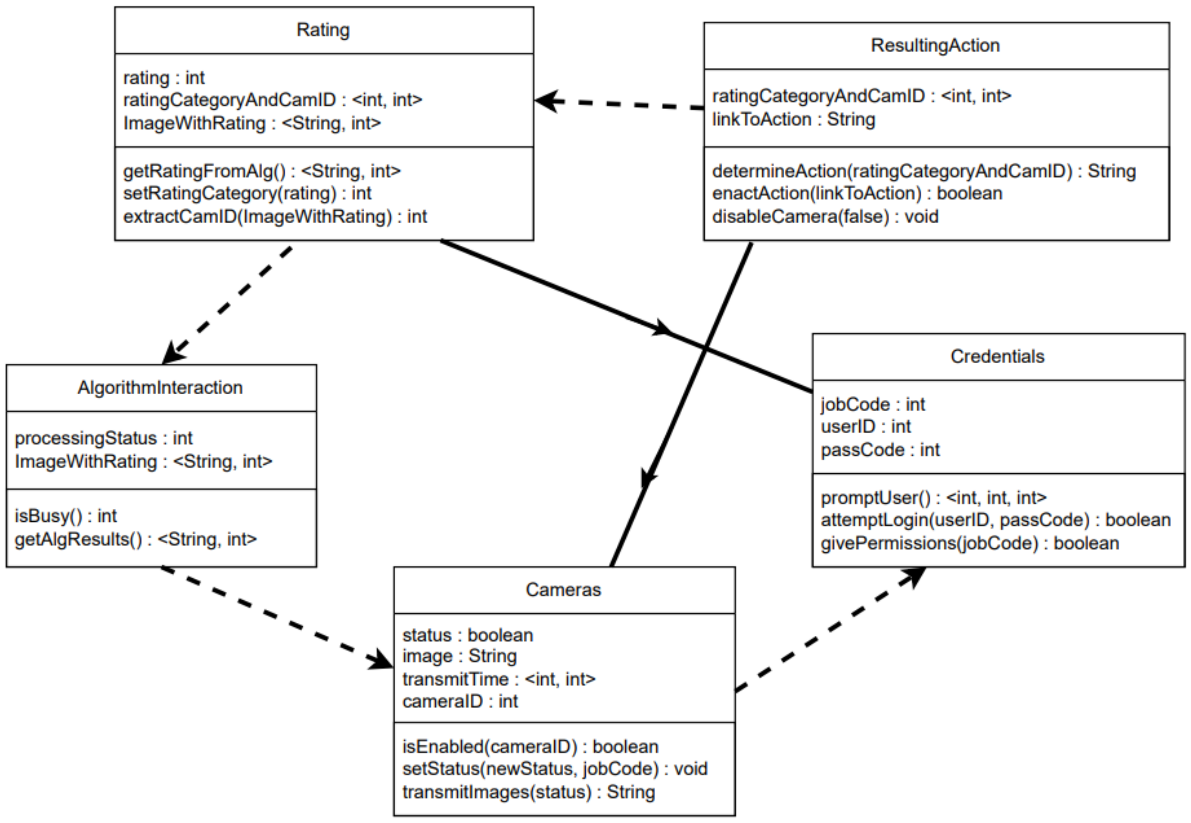


Central Site Processing System
Comparison Algorithm
Holds photos (new and old), ratings, and camera accessing data
User Interface
Algorithm Interaction
Rating Action
Camera Controls
Login
Camera DB
Newer Photos <taken within a month>
Older Photos <up to a year ago>
Officials & User DB
User Credentials <Username, Password>
List of Officials to be contacted in case of emergency <police, fire department, Amtrak, City>

Software Architecture Design Description

In this SWA diagram, a clear picture of how the system will be implemented is on display. Starting with the camera icon, this represents the six cameras placed along the cliffside, monitoring the change in stability. These cameras are then uploaded via Wifi to the nearby router, which transfers the data into the Camera Database. This database is what holds all the photos, new and old, and will be accessed by the Central Site Processing System. This system has a Comparison Algorithm implemented to extract the data from the Camera Database. This algorithm holds the photos, ratings, and camera accessing data, along with actually assigning the rating. The output of this algorithm is then sent to the Algorithm Interaction feature of the User Interface. It is from here that the user can see the photos with the rating attached to them. Depending on the rating, it is then sent to the Rating Action section of the system, where it will access the Officials and User Database, which contains the contact information of important officials in the area that need to be contacted if there is going to be a slide. There are also camera controls which interact with the login feature, which access the the Officials and User Database to see if the user who is attempting to log in has valid credentials to access the system. It is sort of daisy chained in the sense that in order to access the camera control feature, it first needs to interact with the login feature, which needs to interact with the database. The reason why the Rating Interaction does not require login access is because if the rating is high enough, it should show a warning regardless if anyone is signed in. This is prevent disaster from occurring because a user is having trouble logging in.

UML Class Diagram


Rating
rating : int
ratingCategoryAndCamID : <int, int>
ImageWithRating : <String, int>
getRatingFromAlg() : <String, int>
setRatingCategory(rating) : int
extractCamID(ImageWithRating) : int
ResultingAction
ratingCategoryAndCamID : <int, int>
linkToAction : String
determineAction(ratingCategoryAndCamID) : String
enactAction(linkToAction) : boolean
disableCamera(false) : void
AlgorithmInteraction
processingStatus : int
ImageWithRating : <String, int>
isBusy() : int
getAlgResults() : <String, int>
Credentials
jobCode : int
userID : int
passCode : int
promptUser() : <int, int, int>
attemptLogin(userID, passCode) : boolean
givePermissions(jobCode) : boolean
Cameras
status : boolean
image : String
transmitTime : <int, int>
cameraID : int
isEnabled(cameraID) : boolean
setStatus(newStatus, jobCode) : void
transmitImages(status) : String

Class UML descriptions

This UML class diagram depicts the classes and associations between them that we need to design and begin to implement the bluff detection system. The Credentials class involves verifying qualified officials and maintenance workers' user logins and determining what actions they are allowed to do in the system. The promptUser() function collects the potential user's jobCode, userID, and passCode, which are all integer values, sets the fields equal to the inputs, and returns them as a triple, meaning all of those integers are connected to each other. The attemptLogin() and givePermissions() functions use the inputted credentials to determine whether the user should have access to the system and if yes, then what actions is that user allowed to perform within the system. The Cameras class has variables to hold the boolean status of the camera, whether it is enabled or disabled, the image in a String as a URL that includes data like cameraID and transmitTime attached to it within the String. The isEnabled() function uses the cameraID and returns a boolean as to whether that camera is enabled or disabled, of which this status is updated through the setStatus() function which requires a jobCode to ensure the proper users are doing this action. The transmitImages() function works if the status is enabled for a camera on which the function is called and if enabled, returns a String with the image URL. The AlgorithmInteraction deals with the interactions between the system and the algorithm that runs the images through comparison tests and determines their rating. There is a variable for processing status, an integer with precoded values for on, off, in progress of processing, and receiving/transmitting images so that the system only accesses the algorithm when it is properly available and this integer is returned from the isBusy() function. The imageWithRating pair of a String and an integer hold with data returned from the getAlgResults() function that accesses the String of the new image URL from the camera along with the rating that the algorithm assigned it. The Rating class separates the important information from the image URL and keeps it attached to the rating. The getRatingFromAlg() function accesses the data from the AlgorithmInteraction class and separates the rating from the pair in order to be accessed separately for other purposes. The setRatingCategory() uses the integer rating and determines what category of severity it falls into, which will later be used for what consequential action must be taken. The extractCamID() function takes in the ImageWithRating and copies over the cameraID integer value from within the String image URL to know which section of bluff it is referring to when it gets combined with the rating category into a pair. The ResultingAction class uses variables for the ratingCategoryAndCamID pair of integers and a linkToAction String. The determineAction() function uses the rating category to return the String action that must be implemented based on the severity of the situation determined by the comparison algorithm. This returned String populates linkToAction and is pulled from a list of predetermined links for different rating categories. The disableCamera() function sets the camera with the CamID from the data pair to disabled, so it sets their status to false. The enactAction() function opens the linkToAction which typically includes local officials' contacts in order to send emergency alert messages, then the function returns true once the action has been completed. These classes are interconnected and many of them rely on each other for functionality. The Rating class has an association to the Credentials class since a user must have proper credentials in order to set and access ratings, displayed by a solid line in the diagram. Also, the ResultingAction class has an association with the Cameras class since there is a function to disable cameras. Finally, there are many dependencies between the classes, which are displayed by dotted lines with arrows in the diagram. ResultingAction uses data from Rating, Rating uses data from AlgorithmInteraction, AlgorithmInteraction uses data from Cameras, and Cameras uses data from Credentials.

Development Plan and Timeline

In order to develop our “Unstable Bluff” detection system, we will have to create our central site processing system, including our user interface, and comparison algorithms, in addition to the databases for our camera images, officials, and other users. We will also have to set up the wireless camera transmission, and also the system for observing the states of each camera. We will allocate 4 months of time in order to create the detection system.

Sam will be responsible for creating the comparison algorithms. In order to create the comparison algorithms, she will work with officials to create a rating system to analyze the state of bluffs observed, which will done over 2 weeks. After this rating system is created, a machine learning algorithm to learn this system, and observe tells through the images given by the cameras will be created. This will take place over 2 months with appropriate testing being done to ensure the accuracy of the algorithm.

Antonio will be responsible for creating the user interface. One feature of the system is the ability to control the cameras set up along the bluffs. This will be the first task in creating the user interface and will allow users to control these cameras remotely, it will created within the first 3 weeks of the project. This user interface will include interaction with the comparison algorithm, allowing people who use the interface to view the ratings given by the algorithm. This will be created alongside the comparison algorithm over a period of 1 month, and will be a major part of the testing phase for the algorithm. Qualified officials and maintenance workers will also have the ability to login to the central processing system, so a login system will be created over a period of 3 weeks, alongside other tasks, and will give users the ability to access the system based on their credentials that are given during an account’s creation. The interface will also send messages to relevant officials when an at risk rating has been observed, which is a rating of 4 or 5.

The relevant officials contact information and affiliation will be included in the user/official database. Joshua will be responsible for creating the databases and setting up the cameras. Over 1 month six cameras will be set up along the bluffs, along with the necessary wireless transmission set up such as the routers. A database for the camera images will also be set up, with 2 sections one for holding newer photos(1 month or less) and also older photos(1 year or less), this task will be done over 2 weeks. In addition to the camera image database, a database including officials that could be contacted in case of an emergency, and that will also be used to store future user login information will be created. Alongside the login system, this task will be done over a period of 3 weeks, and will be tested during the comparison algorithm testing for its ability to effectively contact relevant officials.