**Environmental Monitoring and Prediction System**

Problem Statement and Requirements

Problem: Users want to know the status of the environment when they want to. Users want to be more informed of the various environmental impact that is happening around them. Users can use this information to determine where to volunteer, where to travel and visit, and where to move.

Problem Solution: The system aims to solve the users need to know what the environmental conditions are. If the user wants to subscribe to predictions and alerts, the system will allow the user to do so. The system will aim to help users access environmental conditions based on already established probes in the air, water, and forests.

The system functionalities will be: Air Quality, Water Quality, Forest Quality, Predictions, Ability to subscribe, and Ability to register as a user.

The target users will be:

- Young people: ages 15 – 25, male and female

- The youth has a trend of caring for the environment and at that age they are able to be more proactive and this will help them to advocate for the environment.

- People who travel: ages 25 – 35, male and female

- People who travel are concerned about their health and safety, as well as their ability to make the most out of their travel experience. This will help them make informed decisions as to where to travel.

- People looking to buy homes or move: ages 35 – 50, male and female

- People who are looking to move are looking to see if where they are moving to is a good location. Environmental factors are of concern, as they may not enjoy moving to a place where there are major environmental issues. Often, if they have kids or plan on having kids, they want a good environment.

Business goals are that the system shall be able to give users a quick response to the information they request. The system shall be able to generate new registered users and subscribers. The system shall have a clean and easy to use interface to appeal to a large market.

Non Functional Requirements

Performance Requirements: Scalability

* 100 concurrent users or 100 TPS: Vertical scaling can be used in this case. Vertical scaling involves adding more power to existing resources, like upgrading server CPU, RAM, and storage.
* 1000 concurrent users or 1000 TPS: Optimizing SQL database queries would be a good way to handle this increase in users. Indexing makes columns faster to query by creating pointers to where data is stored within a database. Caching stores data that is frequently accessed, and usually rarely changed, closer to end users. Can help reduce the load on the database and servers. It can also help improve latency which leads to a faster system.
* 10k concurrent users or 3000 TPS: Horizontal scaling is when more hardware nodes are added to the system, like adding more servers. It is used instead of vertical scaling. If one server goes down, other servers can be used, making less downtime and less performance issues. Horizontal scaling is cost effective and helps application resilience.
* 100k concurrent users: Cloud storage would be the best way to handle this many users. Cloud storage has data in various logical pools that are then distributed over a number of servers. This is perfect for large numbers of users or increased growth because the storage is there, and all that needs to be done is to buy more storage as needed. Load balancing software can also be used to help scalability at this size.

Performance Requirements: Response Time

* The response time shall be within 100–500 milliseconds, as that is ideal for user experience.
* One way to ensure this is through optimizing the database, which is improving the database and making sure that it can handle large amounts. This is done through close monitoring and testing.
* Another way to ensure this is through configuring caching, which is storing data in a temporary location so it is faster to access next time it is called. This is done through monitoring and seeing which variables are called more than others.

Performance Requirements: Throughput

Throughput is the amount of material or items (eg data) passing through the system.

To make sure throughput is able to pass through the system, review the existing workflow and make sure that there are no issues with the amount of data. Eliminate bottlenecks by testing and seeing which areas cause bottlenecks and find solutions. Reduce equipment downtime by choosing reliable equipment and by making sure back up systems are in place. Reduce parts rejection rate by having quality control and eliciting feedback to ensure the customer is happy with the end product. Improve employee training so that everyone is on the same page as to what needs to be done. Use automation in testing as it makes it faster to test and find issues.

Security Requirements: Authentication

A close-up of a line

Description automatically generated

In this case, HTTP is being used. HTTP is a secure application layer protocol that transfers information between devices on the network. It is used for communicating between clients and servers.

Security Requirements: Authorization

Role-based access control (RBAC): to differentiate between a guest and a registered user. RBAC assigns permissions to users based on their role within an organization and grants access based only on the role of the user within an organization.

Security Requirements: Data Encryption

A diagram of a computer network

Description automatically generated

SSL/TLS certificates are electronic documents used to certify the ownership of a public key. They allow secure communications on networks. Servers present their certificates to clients during connections. Using encryption techniques to ensure data is protected when being sent and at rest. End-to-end encryption is one technique that would help the project become more secure. It keeps the messages more secure by keeping the message a secret from everyone, including the service.

Maintainability Requirements: Code Modularity

Code modularity involves dividing a system into smaller, independent modules or components, each with a specific function. This will be done through the use of UML Diagrams.

Maintainability Requirements: Documentation

Documenting code enhances readability, maintainability, and longevity, ultimately saving time, reducing errors, and improving overall software quality.

Determine your audience, define the scope, use a standardized structure, write descriptive titles and explanations, document parameters and return values, document any changes made and when, have a backup.

Maintainability Requirements: Testing Strategies

* Front-End Testing Strategy: making sure everything looks and functions for the user
* Static Testing Strategy. A static test evaluates the quality of a system without actually running the system
* Cross-platform testing: testing across platforms to ensure application can be used by all

Use Case Diagram

A diagram of a process

Description automatically generated

The purpose of a use case diagram is to demonstrate the different ways that a user can interact with a system. It is a visual representation of the interactions between actors, users or external systems, and a system. It is used to show functionality of the system from a user point of view. In this case, a user can either be a registered user or a guest. Both can access air quality, water quality, forest quality, and check predictions. A registered user can subscribe to updates. A guest user can choose to create an account. The database stores login information from the user and subscription status. The database stores air quality, water quality, forest quality, and predictions.

Domain Model

A diagram of a flowchart

Description automatically generated

A domain model is a graphical representation of ideas and how they flow in the system, and not software. Domain models show entities or things, and how they are related. A domain model can tell us the relationships between entities.

In the Environmental System entity, there is a user. The user can be a registered user or a guest. A guest can create an account. Both a registered user and a guest can choose their location, which is then validated, then they access the environmental services, which is the homepage of the application. From there, they can choose to see air quality, water quality, forest quality, predictions, or subscribe. Air quality gets its information from air probes that are close to the location specified. Water quality gets its information from air probes that are close to the location specified. Forest quality gets its information from air probes that are close to the location specified. Prediction gets its information from air, water, and forest probes, then makes a prediction based on the information gathered, and from the algorithm prediction generator.

Class Diagram

A diagram of a computer flowchart

Description automatically generated

A class diagram shows the static structure of the system. It shows classes and the relationship between classes. It shows structural and behavioral features within a system.

A guest has the option to create an account. When the guest creates an account, they enter a username and password, and verify the username and password. From there, they become a registered user. As a registered user, they have a username and password. A guest and a registered user both have the option to put in their location and view the environmental services. Location is verified.

From the environmental services, the user can choose to see air quality, water quality, forest quality, or prediction. Choosing air, water, or forest quality, these are then sent to Air Quality, Water Quality, or Forest Quality. These take the location and find the correct probe in which to pull the information from. When these receive the information from the respective probes, they then take the raw data they receive from the probes, and put it into a readable format, and send it back to the environmental services, so the user can then read it.

For the prediction, the prediction grabs water, air, and forest quality, puts it into a prediction generator which is an algorithm, and that generates a prediction based on the inputs. This is sent back to the environmental service so the user can read it. The user can also subscribe to updates. When subscribing to updates, the user puts in their email address and the frequency in which they want to receive updates.

Sequence Diagram

A diagram of a company

Description automatically generated

A sequence diagram shows all the messages in a system. It is used to show the interactions between objects in the sequential order that those interactions occur.

In this environmental service system, there is a user. The user can login. The database validates the login. The user can subscribe to updates. The database confirms subscription status. The user can update their location. The database confirms the location.

The user can request water quality. The database can tell if the location is bad, and sends an error. If the location is valid, the database sends the location to the water probe. The water probe can then send the water quality back to the database, which then sends the readable water quality to the user.

The user can request air quality. The database can tell if the location is bad, and sends an error. If the location is valid, the database sends the location to the water probe. The air probe can then send the air quality back to the database, which then sends the readable air quality to the user.

The user can request forest quality. The database can tell if the location is bad, and sends an error. If the location is valid, the database sends the location to the forest probe. The forest probe can then send the forest quality back to the database, which then sends the readable forest quality to the user.

The user can request prediction. The database can tell if the location is bad, and sends an error. If the location is valid, the database sends the location to the prediction generator. The prediction generator then requests water quality from the water probe. The water probe sends water quality to the prediction generator. The prediction generator then requests air quality from the air probe. The air probe air quality to the prediction generator. The prediction generator then requests forest quality from the forest probe. The forest probe sends forest quality to the prediction generator. From there, the prediction generator uses its algorithm to generate a prediction. The prediction generator then sends its prediction to the database, which then sends it in a readable format to the user.

State Diagram

A diagram of a flowchart

Description automatically generated

State diagrams are used to model the behavior of objects in a system over time. They illustrate how an object moves through different states and its response.

For the state diagram, the user can choose to login or continue as a guest. Both are buttons on the application. When the user logs in, they can then validate the login by pressing a button. If the login fails, an error message is sent to the user. If the login is successful, they can then see the environmental services home screen. The guest can see the environmental services home screen after continuing as guest. From the environmental home screen, there is the option to see the categories or subscribe. From the subscribe button, the user can then validate the subscription settings. If the subscription settings are valid, they are taken back to the environmental services home screen. From the categories section, the user can then choose air quality button, water quality button, forest quality button, or prediction. These 4 options also have a back button, for the user to return to the categories option.

From the air quality, if the system is able to retrieve air quality, the air quality is then shown. If there is an error retrieving the air quality, an error message is shown. From the water quality, if the system is able to retrieve water quality, the water quality is then shown. If there is an error retrieving the water quality, an error message is shown. From the forest quality, if the system is able to retrieve forest quality, the forest quality is then shown. If there is an error retrieving the forest quality, an error message is shown. From the prediction, if the system is able to retrieve prediction, the prediction is then shown. If there is an error retrieving the prediction, an error message is shown.

From the environmental services home screen, there is a button to logout. If the logout is valid, the user then exits the application. If there is an error logging out, an error message is shown. From the environmental services home screen, there is a button to exit the application. Once a user exits the application, the state is terminated.

Activity Diagram

A diagram of a company

Description automatically generated

An activity diagram shows the flow from one activity to another in a system. It can be used to understand the system and its components. It shows how a user can interact with the system, and the flow of the interactions.

The user opens the application. From there, the user can continue as a guest or login. If the user logs in, the login is validated and the user continues. If the login is unable to validate, there is an error message. The guest can choose to register. If the registration credentials are valid, the user then becomes a registered user and continues. If the registration credentials are invalid, an error message is shown. From there, the user can choose their location. The location is validated. If the location is invalid, an error message is shown. If the location is valid, the user continues to the environmental information chooser.

From there, the user can choose to see air information. The system retrieves air information. If there is an error retrieving air information, an error message is shown. If no errors, the air information is shown. The user can choose to see water information. The system retrieves water information. If there is an error retrieving water information, an error message is shown. If no errors, the water information is shown. The user can choose to see forest information. The system retrieves forest information. If there is an error retrieving forest information, an error message is shown. If no errors, the forest information is shown.

The user can choose to see predictions. From there, the system retrieves air information. If there is an error retrieving air information, an error message is shown. If no errors, the air information is retrieved for the prediction generator. The system retrieves water information. If there is an error retrieving water information, an error message is shown. If no errors, the water information is retrieved for the prediction generator. The system retrieves forest information. If there is an error retrieving forest information, an error message is shown. If no errors, the forest information is retrieved for the prediction generator. From there, a prediction is generated. Once generated, the prediction is shown to the user.

After this, the user can return to the home screen. From there, the user can exit the application. The user can also logout. If there is an issue logging out, an error message is shown. If logout was successful, the user can then exit the application. Once exiting the application, the application is now closed.

Component Diagram

A diagram of a computer

Description automatically generated

Component diagrams are used to model the physical aspects of systems and focuses on essential elements. In my component diagram, the data which is the username and password, flow into the component through the ports on the right hand side. The option to continue as a guest is also on the right hand side. The username and password is converted into a format that the internal components can use. The data passes through the user database for validation.

After validation, it then flows through the user, as well as the guest option. This is because both registered users and guests can access similar functions of the system.

After that, the user can now look into the environmental part through the environmental accessor. This takes the location that the user puts in. The location is validated. From there, it can grab the air quality through the air quality accessor. The air quality is linked to the air probe where it grabs the data. It can also see the water quality through the water quality accessor. The water quality is linked to the water probe where it grabs the data. It can also see the forest quality through the forest quality accessor. The forest quality is linked to the forest probe where it grabs the data. It can also see the prediction through the prediction accessor. The prediction is linked to the prediction generator where it grabs the data.

The prediction gets prediction details, and the output on the left hand side is the prediction. The forest probe gets forest quality details, and the output on the left hand side is the forest quality. The air probe gets air quality details, and the output on the left hand side is the air quality. The water probe gets water quality details, and the output on the left hand side is the water quality.

Deployment Diagram

A screenshot of a computer screen

Description automatically generated

Deployment Diagrams are used to visualize the hardware processes of the system. It shows the communication and relationship between the different hardware components and the placement of software files on the hardware. For my deployment diagram, I assigned various hardware components based on previous diagrams.

The hardware and software components are:

- Environmental application server, which hosts the application.

o The software components are the application server execution environment, which has the .jar file of the application.

- Presentation server, to host the user interface of the application.

o The software components are the application server execution environment, which has the user interface file of the application.

- User server, which allows users to login or continue as a guest.

o The software components are the user, if it is a registered user or guest and the login functionality.

- Database Server, which stores all the data for the environmental components: air quality, water quality, forest quality, and predictions.

o The software components are air quality, water quality, forest quality, and predictions.

-Water Quality Server, which is the server that gets the water quality information from a probe.

o The software components are the water quality.

-Air Quality Server, which is the server that gets the air quality information from a probe.

o The software components are the air quality.

-Forest Quality Server, which is the server that gets the forest quality information from a probe.

o The software components are the forest quality.

-Prediction Server, which is the server that makes predictions based off of information from air, water, and forest probes.

o The software components are the air quality, water quality, forest quality, and prediction generated.

-Azure Cloud Server, which is the server that hosts the application.

o The software components are the presentation server, database server, and environmental server.

-Workstation, which is the actual workstation of the application. This is where the application is running, like a computer or phone.

o The software components are the .exe executable file, which allows users to install and use the application.

Skeleton Classes

public class EnvironmentalService{

public static void main(String arg[]){

continueAsGuest(Boolean)

login(Boolean)

airQuality(int)

waterQuality(int)

forestQuality(int)

prediction(String)

subscribe(Boolean)

logout(Boolean)

exitApp(Boolean)

}

}

private Boolean continueAsGuest(Boolean guest){

if guest = True:

return True

if guest = False:

return False

}

private Boolean login(String Username, String Password){

validateUsername()

validatePassword()

if(validateUsername = True && validatePassword = True){

return True

}

else{

return False

}

}

private Int airQuality(){

getAirQualityFromProbe(){

return airQualityRaw

}

readableAir(airQualityRaw){

return int readableAir(airQualityRaw)

}

}

private Int waterQuality(){

getWaterQualityFromProbe(){

return waterQualityRaw

}

readableWater(waterQualityRaw) {

return int readableWater(waterQualityRaw)

}

}

private Int forestQuality(){

getForestQualityFromProbe() {

return forestQualityRaw

}

readableForest(forestQualityRaw) {

return int readableForest(forestQualityRaw)

}

}

private String prediction(){

getAirQualityFromProbe(){

return airQualityRaw

}

getWaterQualityFromProbe(){

return waterQualityRaw

}

getForestQualityFromProbe(){

return forestQualityRaw

}

generatePrediction(airQualityRaw, waterQualityRaw, forestQualityRaw) {

//generate prediction algorithm

readableAir(predictedAirQualityRaw){

return int readableAir(predictedForestQualityRaw)

}

readableWater(predictedWaterQualityRaw){

return int readableWater(predictedWaterQualityRaw)

}

readableForest(predictedForestQualityRaw){

return int readableForest(predictedForestQualityRaw)

}

return String generatePrediction(readableAir + readableWater + readableForest)

}

private Boolean subscribe(Boolean subscribeYesOrNo){

if subscribeYesOrNo = True {

getUserUsername(username)

getUserUsernameVerify(username)

getUserPassword(password)

getUserPasswordVerify(password)

if (getUserUsername == getUserUsernameVerify) && (getUserPassword == getUserPasswordVerify){

return True

}

else{

return False

}

else{

return False

}

}

private Boolean logout(Boolean logoutYesOrNo){

if logoutYesOrNo = True {

removeUsername()

removePassword()

}

return True

else{

return False

}

private Boolean exitApp(Boolean exitAppYesOrNo){

if exitAppYesOrNo = True {

closeApplication()

}

return True

else{

return False

}

Skeleton Database

A diagram of a diagram

Description automatically generated

1. For the skeleton code, I thought of the main classes in the application. I then went and expanded on them to provide details and insights on how the classes would act. For the main class, there is: continueAsGuest(Boolean), login(Boolean), airQuality(int), waterQuality(int), forestQuality(int), prediction(String), subscribe(Boolean), logout(Boolean), and exitApp(Boolean). I then assigned functions to each class based on how the classes should act.
2. For the skeleton database, I thought of each table needed in the database and the data that would be stored in them. For the user database, there is username, password, and location stored. For the subscription database, there is email address and frequency. For the water database that gets its data from a probe and stores it, there is location and water quality. For the air database that gets its data from a probe and stores it, there is location and air quality. For the forest database that gets its data from a probe and stores it, there is location and forest quality. For the prediction database, there is location, water quality, air quality, forest quality, and prediction made. I thought of example values that would be stored in these databases. I also thought of how the database tables would be linked together, as indicated by the arrows.

Design Patterns SOLID

A screenshot of a computer program

Description automatically generated

S: A class should have one and only one reason to change, meaning that a class should have only one job. For my project, I made sure airQuality, waterQuality, and forestQuality are all classes that have only one job.

O: Objects or entities should be open for extension but closed for modification. For my project, airQuality, waterQuality, and forestQuality all extend environmental categories, but cannot be modified by it.

L: Every subclass or derived class should be substitutable for their base or parent class. For my project, user implements registeredUser and Guest.

I: A client should never be forced to implement an interface that it doesn’t use, or clients shouldn’t be forced to depend on methods they do not use. In my project, waterQuality, airQuality, and forestQuality are all categories. All waterQuality, airQuality, and forestQuality have separate probes that they pull their data from. This is separated from the categories.

D: Entities must depend on abstractions, not on concretions. It states that the high-level module must not depend on the low-level module, but they should depend on abstractions. For my project, the database connection interface, waterQuality, airQuality, and forestQuality do not depend on each other.