From Awareness to Action: Empowering Communities through Disaster Risk Reduction Education

Irfan Ali (011191067), Sumaiya Akter Mim (011191029),
S.M. Salam Rahman (011191032), Sadia Rahman (011191026)

United International University
United City, Madani Avenue, Badda, Dhaka -1212, Bangladesh

1 Project Scenario

Introducing "CommunityShield Academy", is a special project that creates a web application to make Bangladesh is safer during cyclones and other disasters. It uses smart computer skills to provide real-time weather updates, show safe escape routes, and give emergency contact numbers. The web application also connects volunteers and organizations with people who need help during disasters. It teaches everyone how to get ready for emergencies through fun videos and interactive activities. The project is led by clever software engineers who work together to make the web application effective and user-friendly. They use advanced technologies to give accurate weather forecasts and helpful information to keep people safe. The web application acts as a hub, bringing together volunteers and organizations to work as a team and provide support during emergencies. The project also focuses on teaching people about different types of disasters and how to prepare for them. The web application has interactive lessons, quizzes, and localized content that is easy to understand and remember. This way, everyone can learn how to stay safe and be prepared. In Bangladesh, where cyclones and disasters pose threats, "CommunityShield Academy" steps in as a guiding light. Using advanced technology and expert knowledge, it ensures the safety of all citizens. The user-friendly web application offers real-time weather updates, safe escape routes, and emergency contacts. Through interactive education, "CommunityShield Academy" empowers people to face disasters confidently. With software engineers and disaster experts working together, it strengthens Bangladesh, creating a safer future for everyone. Together, we protect and uplift our nation.

2 Architecture Model

For the development of the "CommunityShield Academy" web application, geared towards enhancing safety during disasters, a strategic approach known as "Microservices Architecture" is being employed. Instead of crafting a single, monolithic program, the project involves creating multiple smaller components that function collectively, resembling a well-coordinated team. These components, referred to as "microservices," have distinct roles, such as providing real-time weather updates or illustrating escape routes.

Think of it as building a big project with different tools. They have tools for things like weather updates, escape routes, and emergency contacts. All these tools work together to finish the whole project. Some tools help users, some teach important things, and one even helps volunteers connect with people who need help. To ensure smooth interaction among these microservices, a facilitator known as the "API" has been introduced. This role can be likened to a conductor overseeing an orchestra, with the microservices working harmoniously. The "API" coordinates communication to ensure that the microservices collaborate effectively. Additionally, advanced methodologies and technologies are being utilized to seamlessly integrate these microservices, mirroring the seamless collaboration of a skilled team.

This way of building things lets them make changes without big problems. But they need to make sure all the parts understand each other. Even though it might be different from the usual way, this method helps them create a strong web application. With this approach, they're helping communities get ready and take action when disasters happen.

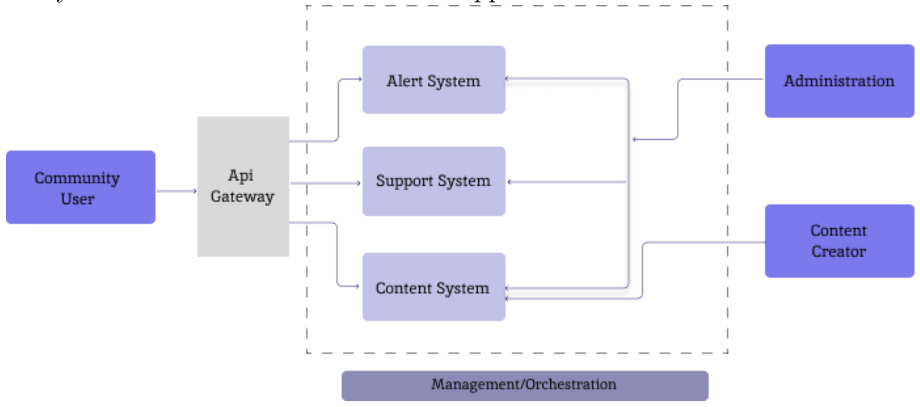


Figure 1: Microservices Architecture

The microservices architecture diagram depicts a dynamic system with three core microservices: the Alert System, the Support System, and the Content System, interconnected with various user actors, including Administrators, Community Users, and Content Creators. Each component in this architectural framework serves a specific purpose and collaborates to provide a rich user experience. The Administration Actor oversees system management, including configuring alerts, monitoring support requests, and content management. Community User Actors are regular users, while Content Creator Actors contribute content. The Alert System communicates critical notifications to both administrators and community users, ensuring everyone is informed. The Support System provides user assistance.

3 C4 model

The C4 model is a well-established approach for visualizing the architecture of a software system from various perspectives. It provides a clear and concise way to represent how the system is organized into different containers and components, as well as their relationships. Furthermore, the model can illustrate how the system interacts with its users and external dependencies.
At its core, the C4 model comprises a series of hierarchical software architecture diagrams. These diagrams offer multiple levels of abstraction, enabling software architects and development teams to grasp the system's structure at different levels of detail. The C4 model typically consists of the following levels:

- System Context Diagram (Level 1): This is the highest-level view, portraying the software system as a single entity or a box, often called "the system," with its interactions with external users and other systems. It provides an overview of the system's boundaries and its major external components.

- Container Diagram (Level 2): The container diagram zooms into the system and breaks it down into high-level containers, which could be web servers, databases, mobile apps, microservices, or other major building blocks. It emphasizes how these containers interact with each other and with external users.

- Component Diagram (Level 3): The component diagram goes deeper into each container and shows the smaller parts, called components, that make up the container. Components are like building blocks of the system. This diagram helps to understand how these building blocks work together inside the container to do their job. It shows how they team up to make the container do what it's supposed to do.

- Code Diagram (Level 4): In certain variations of the C4 model, there's a code-level view that reveals the key classes or modules within each component, offering developers a detailed understanding of how the system's components are implemented and how they interact at the code level.

3.1 Context Diagram

The context diagram for CommunityShield Academy illustrates the interactions between the system and its external entities. Community Users can register, receive weather alerts, and access content. Administrators post quizzes, offer support, and manage content, while Content Creators contribute blogs and videos. This diagram provides a concise overview of the system's functionality and its engagement with users, administrators, and content creators, emphasizing its educational and community-oriented services.

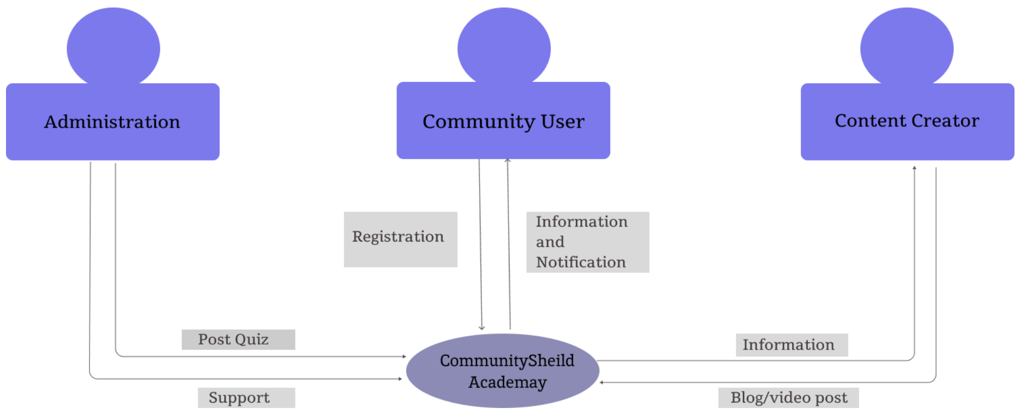


Figure 2: Context Diagram

3.2 Container Diagram

The CommunitySheild Academy container diagram demonstrates that the system has been zoomed in at CommunitySheild System. Here, community Users interact with the content system and system support through web applications and mobile applications, and they also get emails from the email system. Admin additionally interacts with the Content System and System Support.

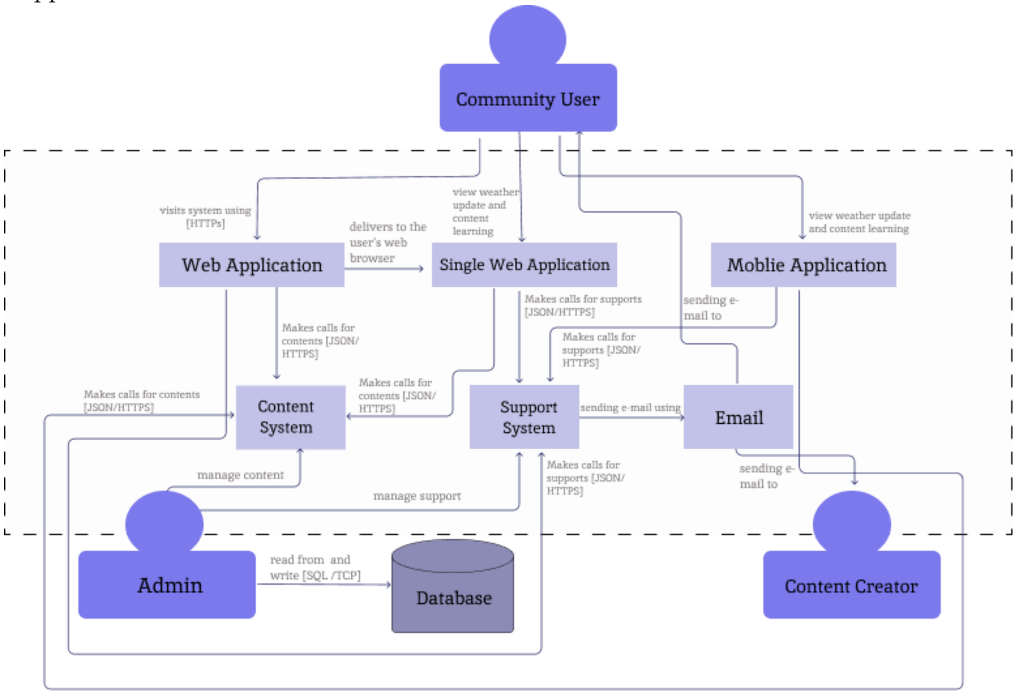

Community User
visits system using [HTTPs]
view weather update and content learning
view weather update and content learning
Web Application
delivers to the user's web browser
Single Web Application
Moblie Application
Makes calls for contents [JSON/ HTTPS]
Makes calls for supports [JSON/HTTPS]
sending e-mail to
Makes calls for supports [JSON/ HTTPS]
Makes calls for contents [JSON/HTTPS]
Content System
Makes calls for contents [JSON/ HTTPS]
Support System
sending e-mail using
Email
manage content
manage support
Makes calls for supports [JSON/ HTTPS]
sending e-mail to
Admin
read from and write [SQL /TCP]
Database
Content Creator

Figure 3: Container Diagram

3.3 Component Diagram

We have shown component diagram for the content system and support system.

3.3.1 Component Diagram for Content System

We updated the component diagram to show that the ”Homepage” is a part of the ”Content System.” Its job is to fetch and display content on the main page of the platform, making it easier for community users to access educational materials.

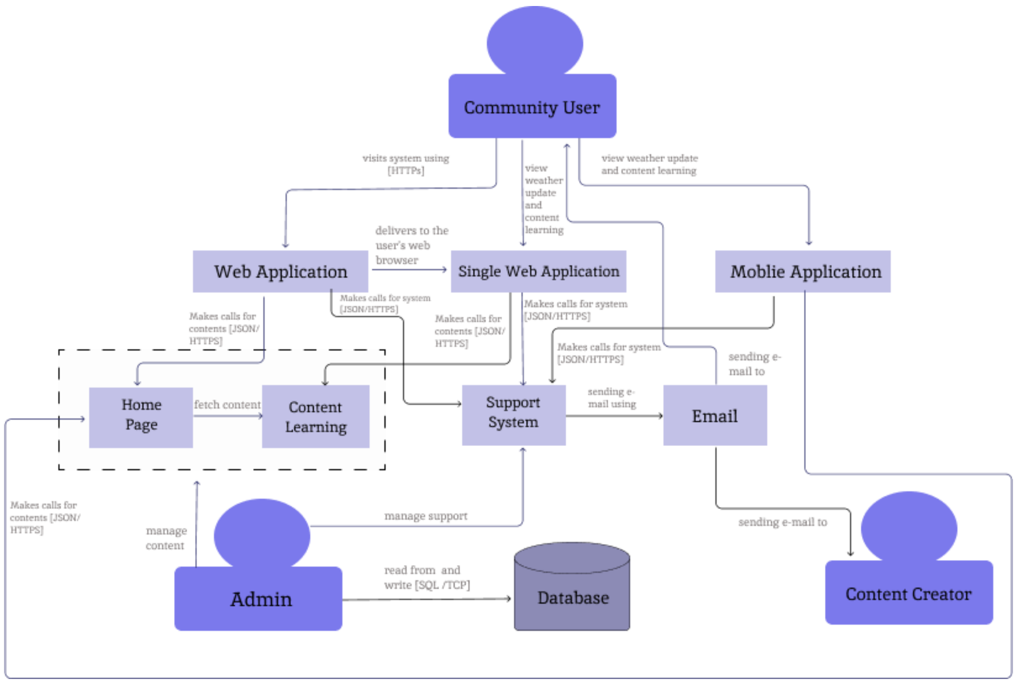

Community User
visits system using [HTTPs]
view weather update and content learning
view weather update and content learning
Web Application
delivers to the user's web browser
Single Web Application
Moblie Application
Makes calls for contents [JSON/ HTTPS]
Makes calls for system [JSON/HTTPS]
Makes calls for contents [JSON/ HTTPS]
Makes calls for system [JSON/HTTPS]
Makes calls for system [JSON/HTTPS]
sending e-mail to
Home Page
fetch content
Content Learning
Support System
sending e-mail using
Email
Makes calls for contents [JSON/ HTTPS]
manage content
manage support
sending e-mail to
Admin
read from and write [SQL /TCP]
Database
Content Creator

Figure 4: Component Diagram for Content System

- **Component Diagram for Content Learning:**

The component diagram of content learning demonstrates how community users interact with quizzes, blogs, and video tutorials through the home page. Here, content creators upload videos and blogs to the website, and the administrator also engages with the material. Email is sent to both Community Users and Content Creators.

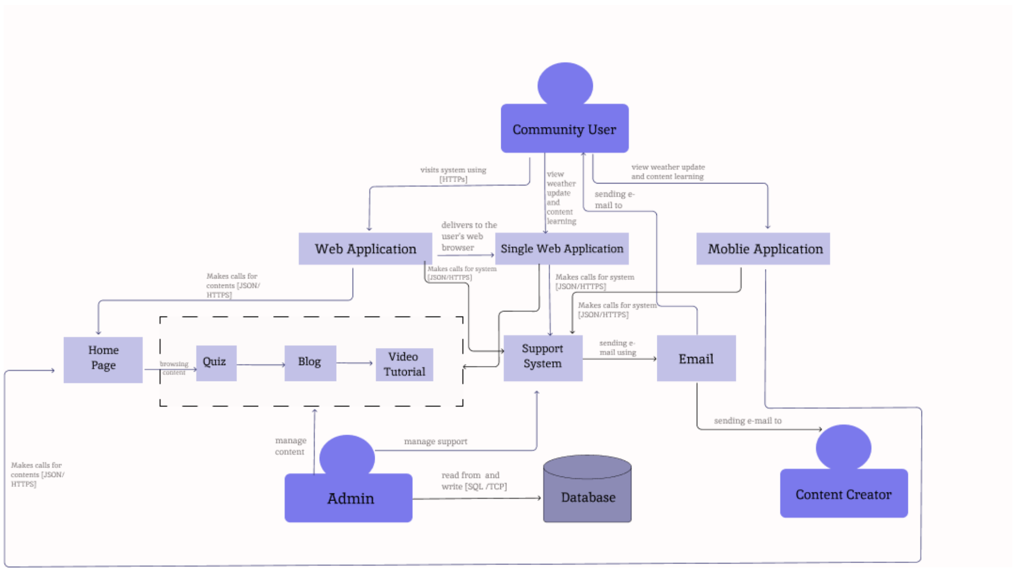
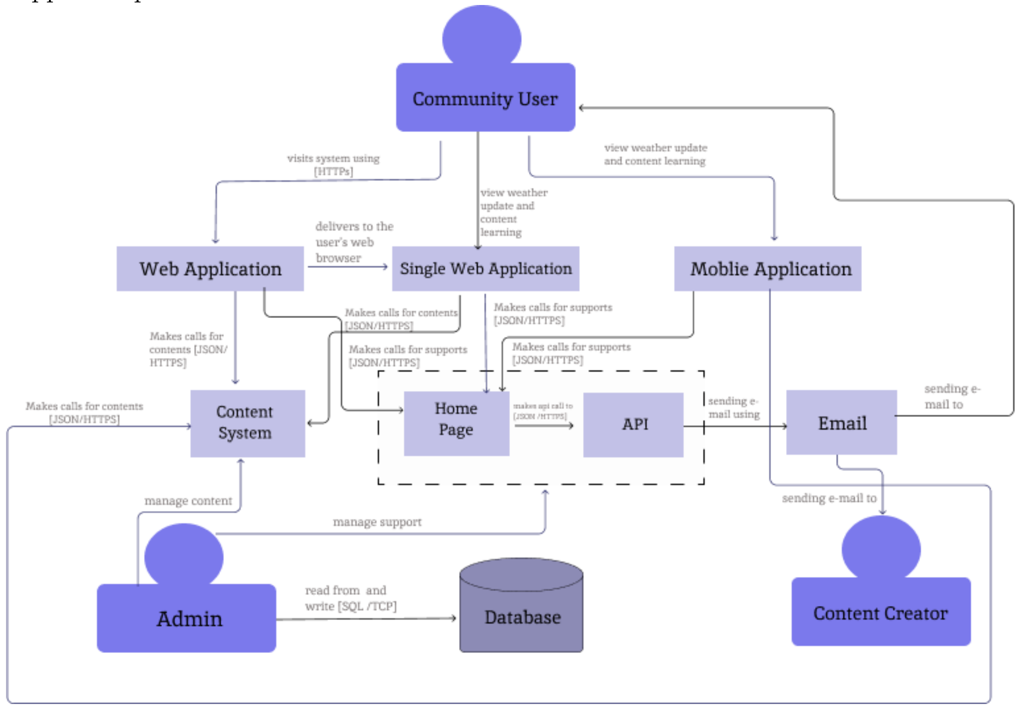

Community User
visits system using [HTTPs]
view weather update and content learning
sending e-mail to
view weather update and content learning
Web Application
delivers to the user's web browser
Single Web Application
Moblie Application
Makes calls for contents [JSON/ HTTPS]
Makes calls for system [JSON/HTTPS]
Makes calls for system [JSON/HTTPS]
Makes calls for system [JSON/HTTPS]
Home Page
browsing content
Quiz
Blog
Video Tutorial
Support System
sending e-mail using
Email
sending e-mail to
manage content
manage support
Makes calls for contents [JSON/ HTTPS]
Admin
read from and write [SQL /TCP]
Database
Content Creator

Figure 5: Component Diagram for Content Learning

3.3.2 Component Diagram for Support System

The homepage is like the control center of our support system. It uses an API, like a tool, to talk to other parts of the system and get information, especially regarding customer support requests. If we change anything on the homepage, it can affect how the whole system works and how well it communicates with other components, including those responsible for handling community user's support inquiries.



Community User
visits system using [HTTPs]
view weather update and content learning
view weather update and content learning
Web Application
delivers to the user's web browser
Single Web Application
Moblie Application
Makes calls for contents [JSON/HTTPS]
Makes calls for contents [JSON/HTTPS]
Makes calls for supports [JSON/HTTPS]
Makes calls for supports [JSON/HTTPS]
Makes calls for supports [JSON/HTTPS]
Makes calls for contents [JSON/HTTPS]
Content System
Home Page
makes api call to [JSON /HTTPS]
API
sending e-mail using
Email
sending e-mail to
sending e-mail to
manage content
manage support
Admin
read from and write [SQL /TCP]
Database
Content Creator

Figure 6: Component Diagram for Support System

- **Component Diagram for API:**

The component diagram of the API illustrates that the community user interacts with Real Time Weather Update and Safe Zone Route Suggest through the Home page. Admin also interacts with the Real-Time Weather Update and Safe Zone Route Suggest. Community Users and Content creators also got mail using the email system.

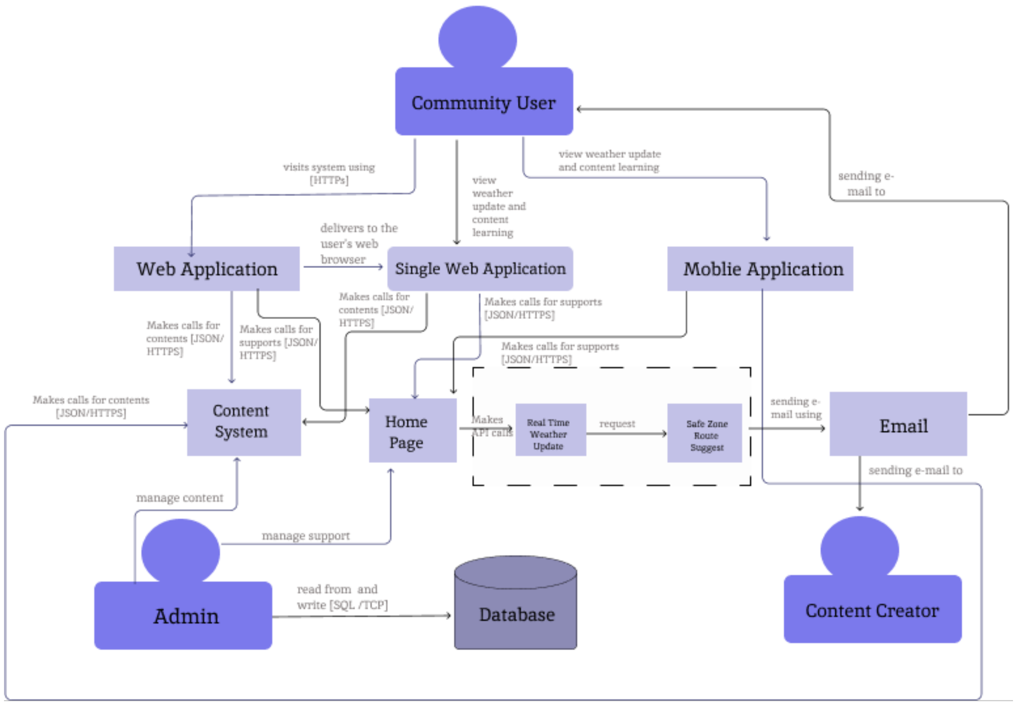

Community User
visits system using [HTTPs]
view weather update and content learning
view weather update and content learning
sending e-mail to
Web Application
delivers to the user's web browser
Single Web Application
Moblie Application
Makes calls for contents [JSON/ HTTPS]
Makes calls for supports [JSON/ HTTPS]
Makes calls for contents [JSON/ HTTPS]
Makes calls for supports [JSON/HTTPS]
Makes calls for supports [JSON/HTTPS]
Makes calls for contents [JSON/HTTPS]
Content System
Home Page
Makes API calls
Real Time Weather Update
request
Safe Zone Route Suggest
sending e-mail using
Email
sending e-mail to
manage content
manage support
Admin
read from and write [SQL /TCP]
Database
Content Creator

Figure 7: Component Diagram for API

3.4 Code Diagram for Quiz

Participants and scores in this Component from the content learning inherited the Quiz class. Participants participate in the quiz and answer the Quiz question using the answer() method and see the score of every attempt by the update() function.

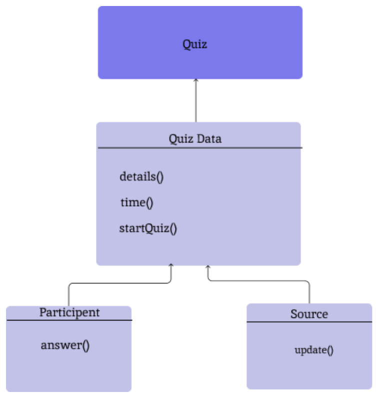


Figure 8: Code Diagram for Quiz

3.5 Code Diagram for Blogs

Comments and favorites inherit the Blog class in this learning component's component of content. The comment class can comment the blog using addcooment() and see the comment using the getComment method. Finally, through favorite, class can add as favorite blog using the function addfavourite().

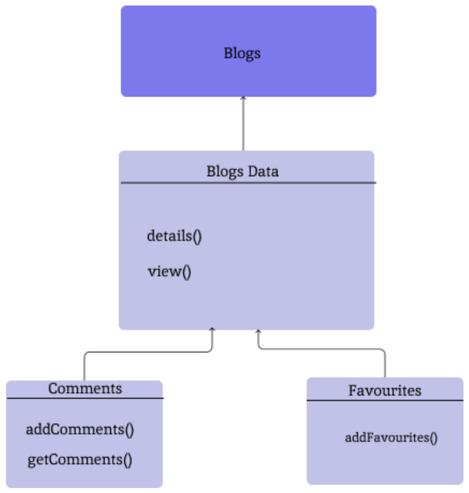


Figure 9: Code Diagram for Blogs

3.6 Code Diagram Video Content

Here favorite, comment and video control diagram inherits Video Content. From the favorite class, it can add a video content as favorite using addtoFavourite() method. Then from the comment class you can add comments using addComment() methods. Finally, a VideoControl diagram can be controlled using play() and share() methods.

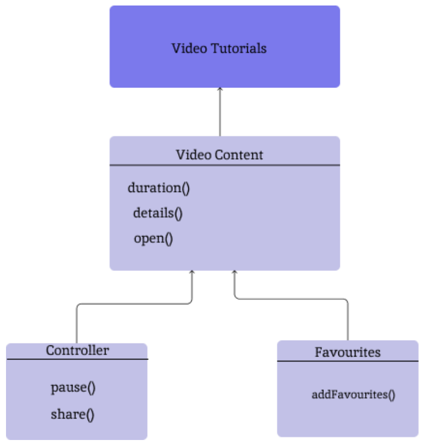


Figure 10: Code Diagram Video Content

3.7 Code Diagram for Real time Weather Update

Current Condition, Statistics Display, as well as Forecast Display classes also came from the Weather Data class in the Support System. Weather Data inherited Real Time Weather Update. Here, Support system uses Real Time Weather Update. Also, Current conditions update and display the data using Update() and display() methods.

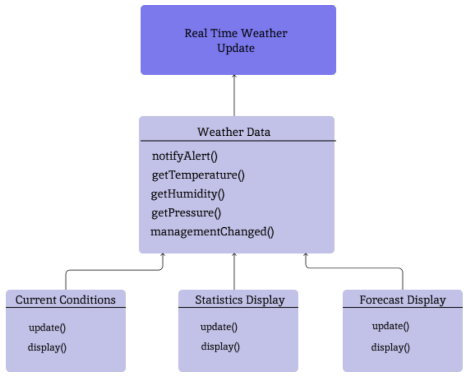


Figure 11: Code Diagram for Real time Weather Update

3.8 Code Diagram for Safezone Route Suggestions

Safe Zone Data inherited Safe Zone Route in this component. Then, Safe Zone Data is also inherited by Near Shelter and getResponse. Finally, Current conditions also inherit Near Shelter. Safe Zone Shelter notifies Near Shelter through alert() method and also sends response using getRoute() method.

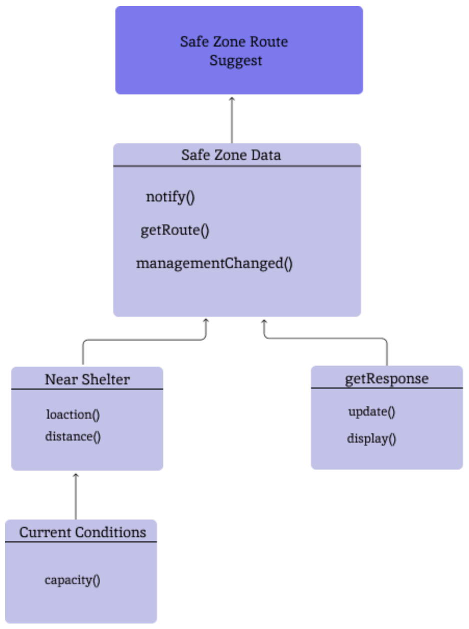


Figure 12: Code Diagram for Safezone Route Suggestions