**COMSATS University Islamabad,   
Abbottabad Campus**

**SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)**

**for**

**Video Surveillance System**  
Version 1.0

***By***

**Zain Asif CIIT/FA20-BSE-136/ATD**

**Nabeel Saleem CIIT/FA20-BSE-122/ATD**

**Hammad Ur Rehman CIIT/FA20-BSE-126/ATD**

***Supervisor*Dr. Saad Mustafa**

***Bachelor of Science in Computer Science (2020-2024)***

**Table of Contents**

**Revision History 3**

**1.** **Introduction 5**

**2.** **Design Methodology and software process model 5**

**3.** **System Overview 5**

3.1 Architectural Design 6

3.2 Colloborative Relationships 7

**4.** **Design Models [along with descriptions] 8**

**5.** **Data Design 19**

5.1 Data Dictionary 21

**6.** **Algorithm & Implementation 22**

**7.** **Software Requirements Traceability Matrix 23**

**8.** **Human Interface Design 24**

8.1 Screen Images 26

**9.** **Appendix I 30**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

**Supervised by**

**Dr. Saad Mustafa**

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Introduction**

We are developing an AI system which ensures the safety of people by detecting the violence or suspicious activity. It divides video in chunks and the feed them into our Deep learningmodel which is trained on our dataset. Then our model process the video if it find any violence activity related to the classes we define like Shooting and Explosion then it will notify the authorized user.

**Scope:**

The scope of the project encompasses the development of a comprehensive video surveillance application with a focus on violence detection. Till now we are able to cover some of main module of our system. Like Registration and Authentication module where Admin can login and perform registration and authorization of user. Ensures secure and authorized access to the application. We also cover the Violence Detection module this module will Divides video into frames and feed to our Deep Machine Learning Model. This model will analyze these frame and if any violence activity is detected related to our define classes then our model inform the user.

**Design methodology and software process model**

When it comes to designing a machine learning system, choosing the right design methodology is super important for an effective and efficient development process. So, for our Project we are following the Object Oriented approach. Let me explain and justify why this choice makes sense:

**Modularity and Reusability:**

OOP allows for the encapsulation of code into reusable and modular objects. In Flutter, widgets can be organized as modular components, making it easier to manage various aspects of the application, such as UI elements, user interactions, and data handling.

**Flexibility and Scalability:**

OOP provides a flexible and scalable architecture, making it easier to accommodate changes and additions to the system. As the Flutter app and machine learning model evolve, the OOP design allows for the seamless addition of new features or improvements without disrupting the existing codebase.

**Clear Separation of Concerns:**

OOP encourages the separation of different concerns, such as user interface, data processing, and model training, making it easier to manage the complexity of the project.

**Software process model:**

Adopting an iterative software process model for the development of the proposed video surveillance application with violence detection offers several advantages. Here are some reasons why an iterative approach is suitable for this project:

**Evolutionary Nature of Machine Learning Models:**

An iterative approach allows for frequent model updates, incorporating new data and insights into the training process. This flexibility is crucial for enhancing the model's accuracy over time.

**Incremental Development of Features:**

The nature of the project involves multiple features such as registration, video processing, notification, and machine learning model integration. An incremental approach allows for the step-by-step development of these features.

**Early Detection of Issues:**

Regular testing and feedback during each iteration allow the team to identify and rectify issues promptly. This proactive approach minimizes the chances of major defects accumulating over time

**System overview**

**Functionality:**

The proposed project is an AI-based Violence Detection System designed to enhance public safety by identifying violent and suspicious activities in real-time through video surveillance. The system utilizes a Deep Machine Learning model, trained on a specific dataset, to analyze video chunks and detect potential threats. The process involves dividing videos into segments, processing them and notifying authorized users if violence or suspicious activities are detected.

**Key functionalities include:**

**Registration and Authentication:**

Admin login and register user for authorized access.

Admin manages user credentials, including updates, registrations, and deletions.

**Upload Video:**

Users can upload videos to detect violent activities.

**Notification:**

Alert notifications are sent to registered users when violent incidents are detected.

**Action Detection:**

Video frames are divided into patches, embedded for computer understanding, and feed to our model then our model based on its training try to detect violence activities.

**Context:**

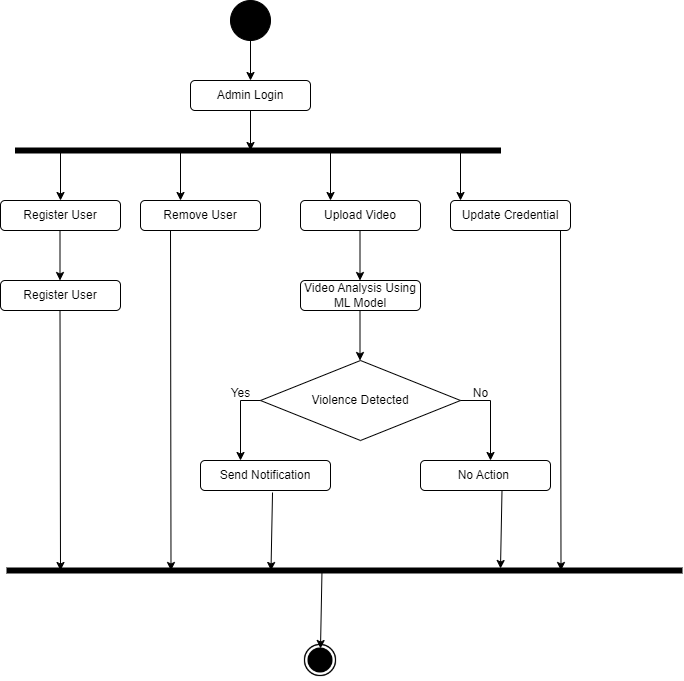
The project addresses the rising crime rates in Pakistan by providing a cost-effective, efficient, and intelligent solution to violence detection. Traditional surveillance systems may miss events or have delayed reactions, and hiring 24/7 human security is expensive. The proposed system aims to fill this gap by automating the detection process and alerting authorized users in real-time.

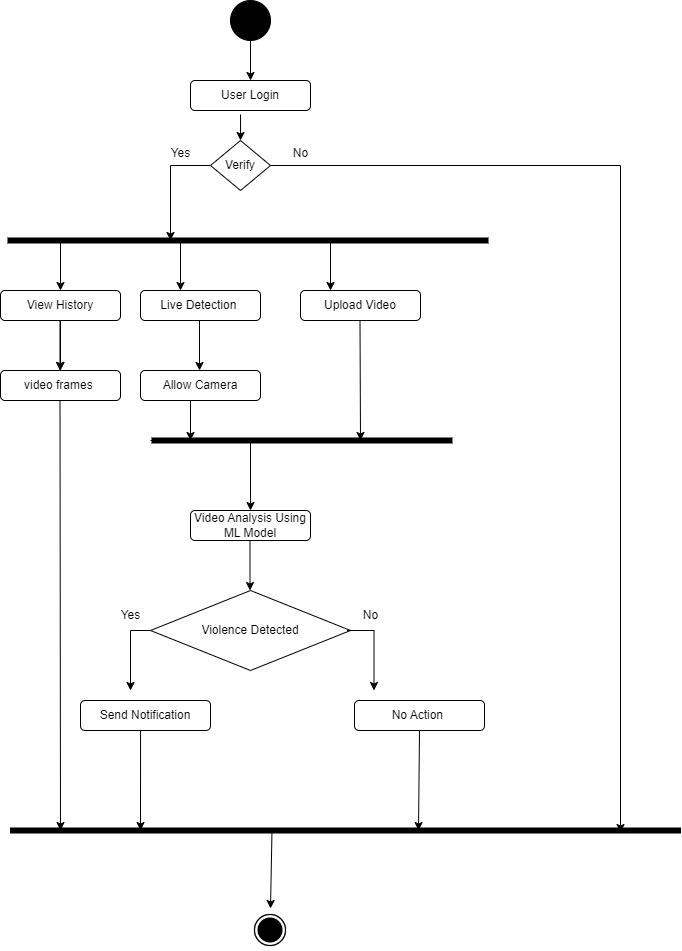
**Design:**

The system is designed as an Android-based application, ensuring accessibility for all the register user. It follows an iterative software process methodology, allowing for flexibility and scalability

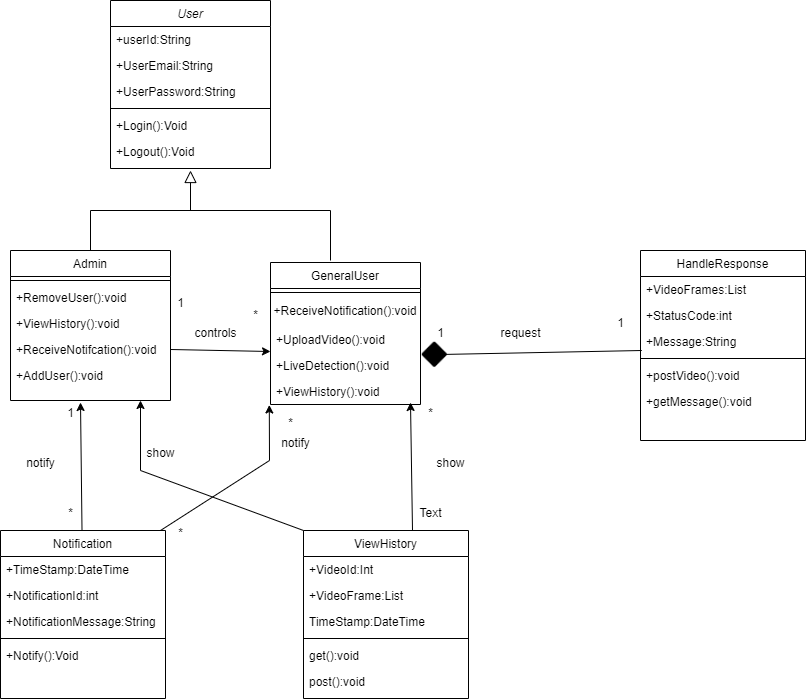
**Process flow/Representation**

**By Admin perspective:**



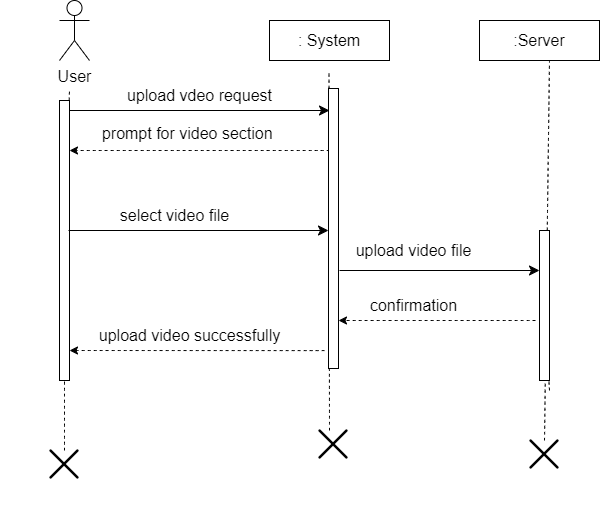
**By User perspective:** ****

**Class Diagram**

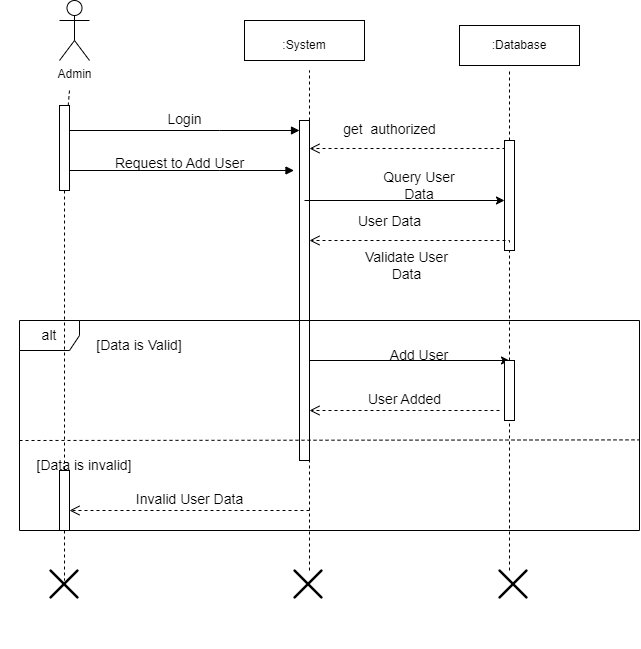


**Sequence Diagram**

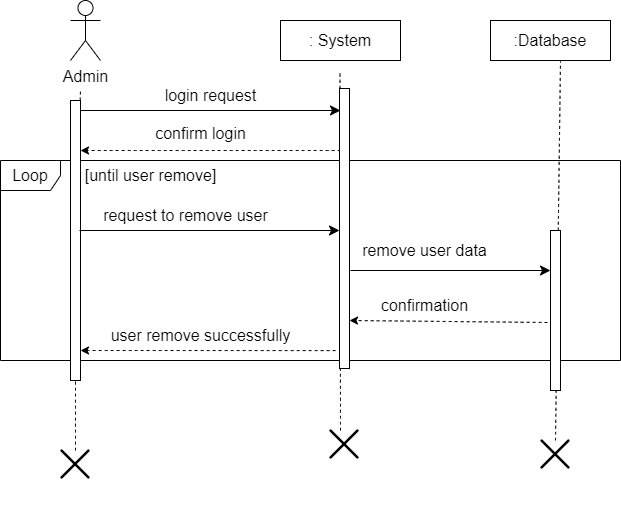
**Upload Video Sequence Diagram:**

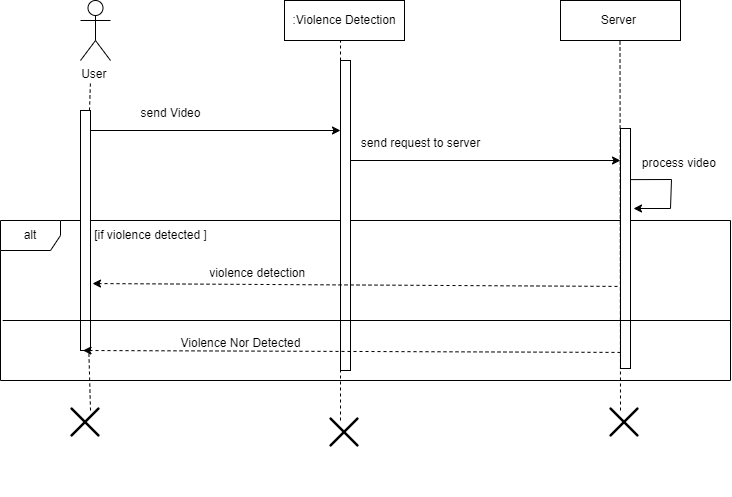
****

**Add User Sequence Diagram:**

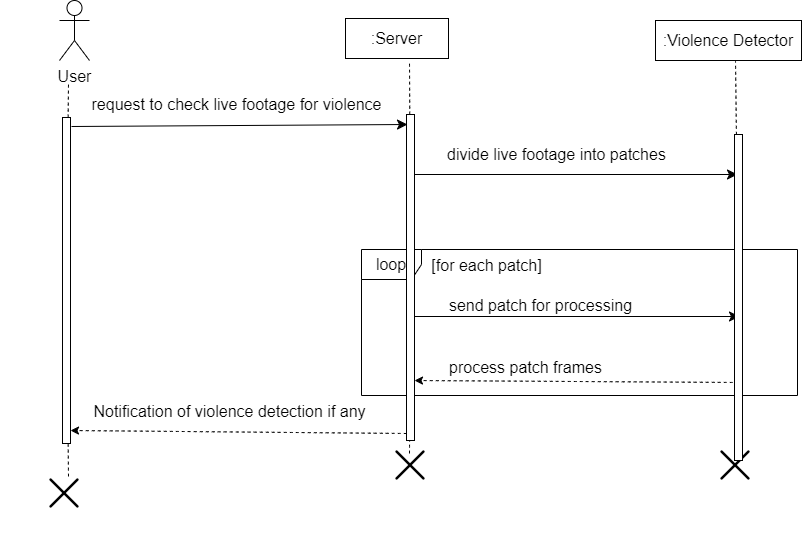
****

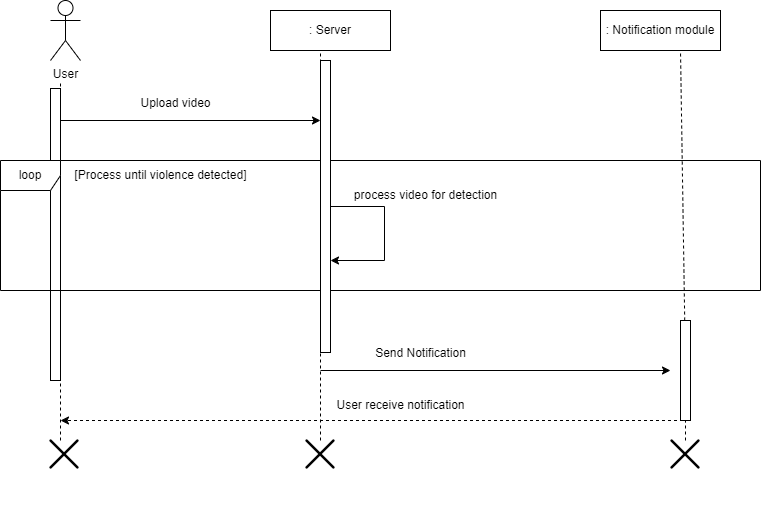
**Remove User Sequence Diagram:**

**Violence Detection Sequence Diagram:**

****

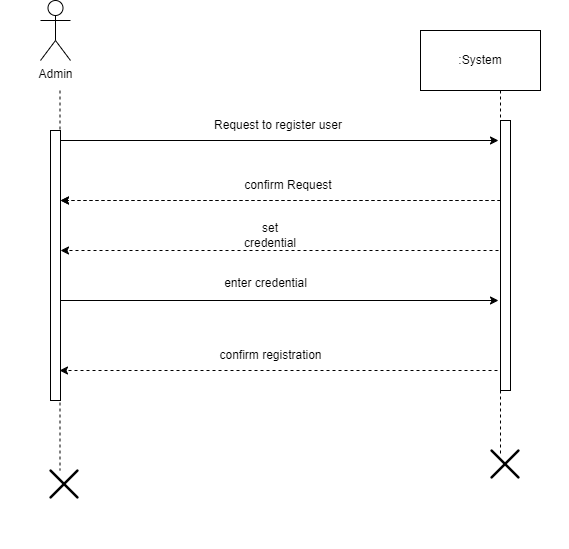
**Live Detection Sequence Diagram:**

****

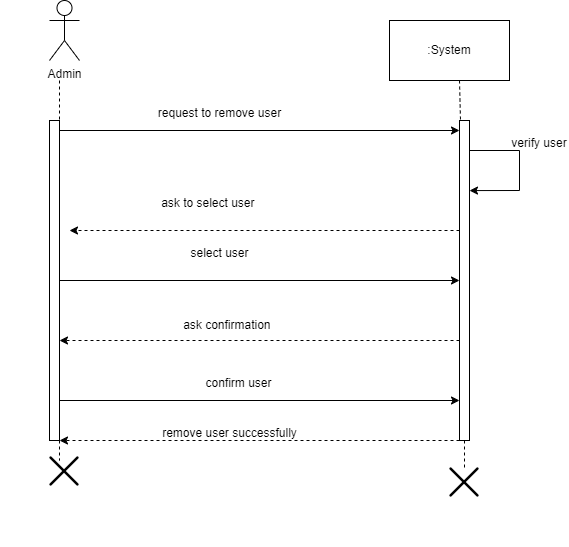
**Notification Sequence Diagram:**

**System Sequence Diagram**

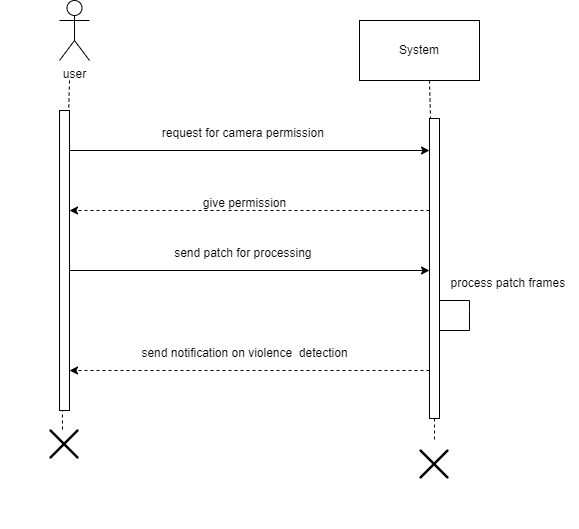
**Add User System Sequence Diagram:**



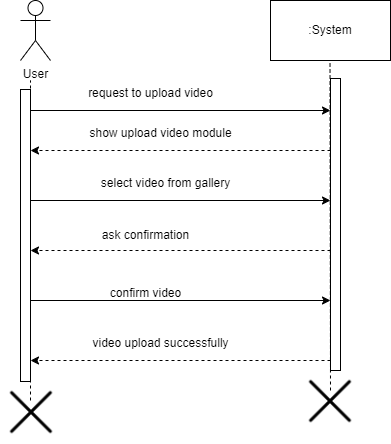
**Remove User System Sequence Diagram:**

****

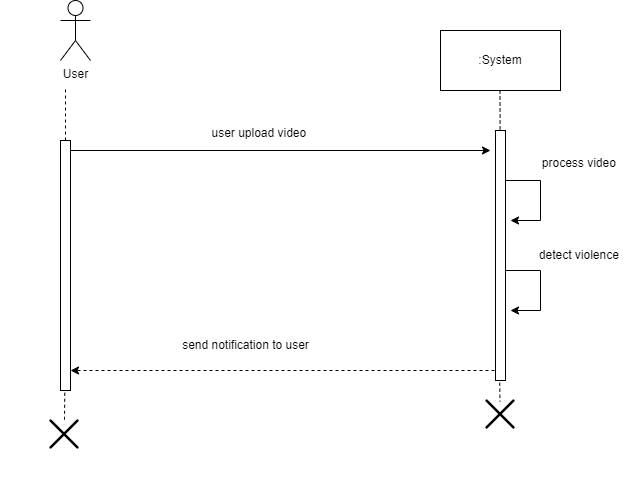
**Live Detection System Sequence Diagram:**

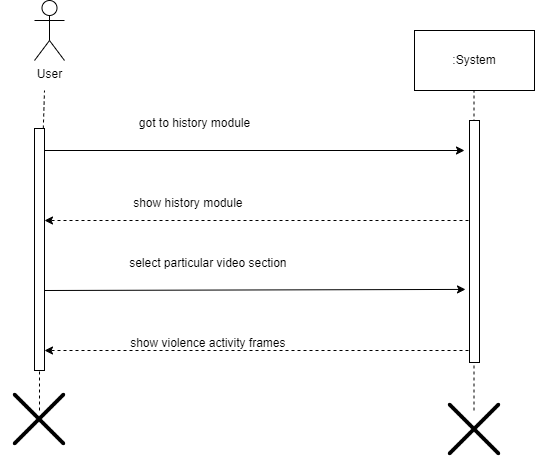
****

**Upload Video System Sequence Diagram:**

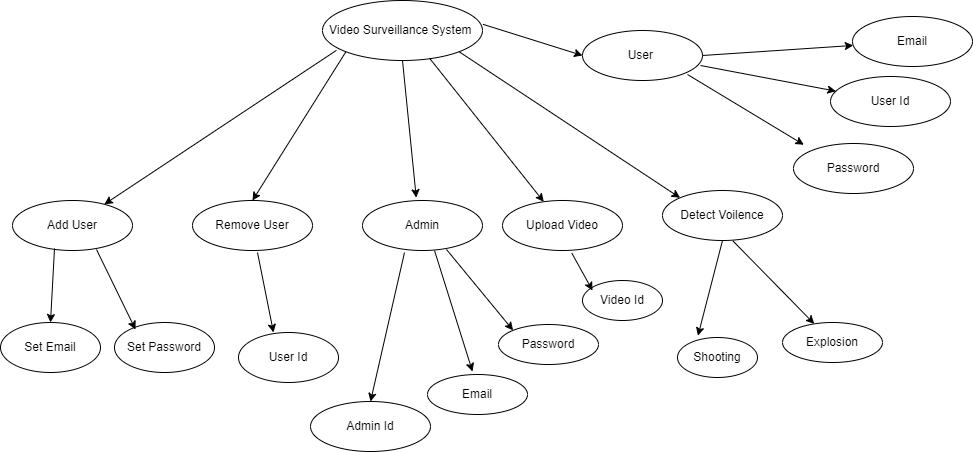
****

**Notification System Sequence Diagram:**

**View History System Sequence Diagram:**

****

**JSON Tree:**

****

**Data design**

Here is data design for the main modules of our project, focusing on the admin and user-related data. We'll consider the relationships between them and how data can be organized.

**Collections:**

**1. Admins**

Attributes:

* AdminID (Auto-generated or manually assigned)
* Username
* Email
* Password
* Other relevant admin details

**2. Users:**

Attributes:

* UserID (Auto-generated or manually assigned)
* Username
* Email
* Password (Hashed for security)
* Other relevant user details

**3. Videos:**

Attributes:

* Video ID (Auto-generated)
* UserID
* VideoURL
* Timestamp
* ViolenceDetectionResult
* Other video-related metadata

**Relationships:**

**1. Admin-User Relationship:**

An admin can have a role in managing users

**Data Organization:**

**1. Admins:**

* Admins are stored in the "Admins" collection.
* Indexed by AdminID.

**2. Users:**

* Users are stored in the "Users" collection.
* Indexed by UserID.

**Data dictionary**

1. **Admins Collection:**

* Attributes :AdminID, Username, Email, Password
* Type: String
* Description: Store complete information about admin like admin username password etc. . . .

**2. Users Collection:**

* Attributes: UserID, Username, Email, Password
* Type : String
* Description: User collection store all the information related to user like user id user username password etc.

**3. Videos Collection:**

* Attributes: Video ID, VideoURL, ViolenceDetectionResult, Timestamp
* Type : String
* Description: Video collection store all the information related to video like video id, video URL, violence detection results etc.

**Algorithm & Implementation**

**Pseudocode for Add User:**

Admin enter his login credential

Set user email

Set user password

**Pseudocode for Remove User:**

Admin enter his login credential

Admin select specific user

Remove user

**Pseudocode for Upload video:**

User enter his credential

Select a video from gallery

User confirmed selected video

IF

Video is in allowed format

Video upload successfully

ELSE

Please change video format

**Pseudocode for Notification:**

IF

Violence activity detected

Send notification to the user

ELSE

No action will be perform

**Pseudocode for View History:**

System verify user

IF

Violence detected

User view the violence frames

ELSE ‘

User view blank page

**Pseudocode for Live Detection:**

User allow camera access

System convert video into small patches

Feed video frames into the model

Model analyse the frames

Detect violence

**Software requirements traceability matrix**

This section should contain a table that summarizes how each software requirement has been met in this document. The tabular format permits one-to-one and one-to-many relationships to be shown.

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. Number** | **Ref. Item** | **Design Component** | **Component Items** |
| FR-01 | Add user | Design Component | Component Items |
| FR-02 | Introduction | Design Component | Component Items |
| FR-03 | Scope | Design Component | Component Items |
| FR-04 | Design Methodology | Design Component | Component Items |
| FR-05 | Functionalities | Design Component | Component Items |
| FR-06 | Modules | Design Component | Component Items |
| FR-07 | Collaborative Relationship | Design Component | Component Items |
| FR-08 | Use Case Diagram | Use Case Diagram | Actors, Use Case IDs, Use Case Names, Triggers, etc. |
| FR-09 | Activity Diagram | Activity Diagram | Activities, Actions, Decisions, Flow of Activities |
| FR-10 | System Sequence Diagram | System Sequence Diagram | Actors, System Operations, Messages, Flow of Control |
| FR-11 | Sequence Diagram | Sequence Diagram | Actors, Objects, Messages, Lifelines, Interactions |
| FR-12 | Class Diagram | Class Diagram | Classes, Attributes, Methods, Relationships, etc. |
| FR-13 | JSON Tree | JSON Tree | JSON Structure, Keys, Values, Relationships |
| FR-14 | Data Dictionary | Data Dictionary | Entities, Attributes, Data Types, Constraints |
| FR-15 | Pseudo Code | Pseudo Code | Algorithmic Steps, Variables, Control Structures |
| FR-16 | Screen Images | Screen Design | Screenshots, UI Components, Navigation Paths |
| FR-17 | Conclusion | Design Component | Component Items |

**Human interface design**

**User-Centered Design:**

**Goal:**

Prioritize the needs and preferences of the users throughout the design process.

**Considerations:**

Conduct user research to understand the target audience.

Create user personas to represent different user types and their goals.

Gather user feedback through usability testing and iterate the design based on insights.

**Intuitive Navigation:**

**Goal:**

Enable users to navigate the application effortlessly and find relevant features.

**Considerations:** Implement a clear and well-organized menu structure.

Use intuitive icons and labels for navigation elements.

Minimize the number of clicks or steps required to perform common tasks.

**Consistent Design Language:**

**Goal:**

Maintain visual consistency across the application for a cohesive and professional look.

**Considerations:**

Establish a consistent color scheme, typography, and iconography.

Use standardized UI elements to create a unified design language.

Ensure a coherent design across different sections of the application.

**Clear Call-to-Action (CTA):**

**Goal:**

Guide users toward key actions and features within the application.

**Considerations:**

Use contrasting colors or visual elements to highlight primary CTAs.

Employ clear and concise language for buttons and prompts.

Prioritize important actions based on user goals.

**Engaging Visual Design:**

**Goal:**

Create a visually appealing and engaging interface to enhance user experience.

**Considerations:**

Use high-quality images and graphics that align with the application's purpose.

Balance visual elements to avoid clutter and maintain a clean design.

Consider incorporating branding elements to create a cohesive visual identity.

**Security and Privacy Considerations:**

**Goal:**

Build trust by clearly communicating how user data is handled and secured.

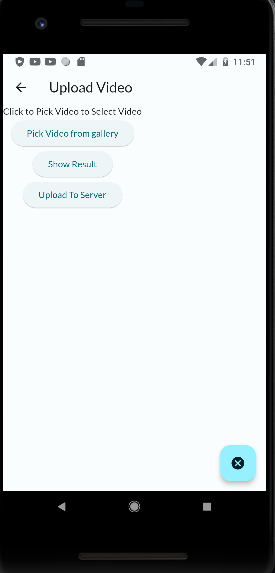
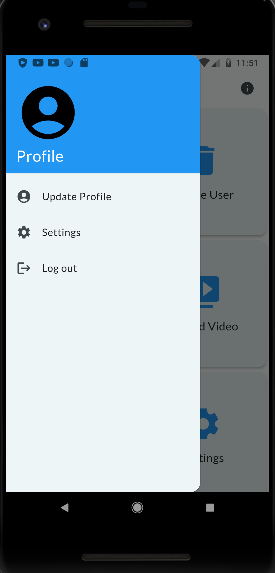
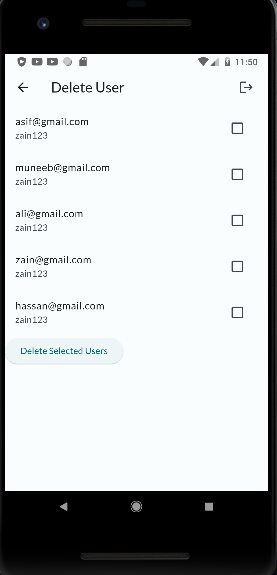
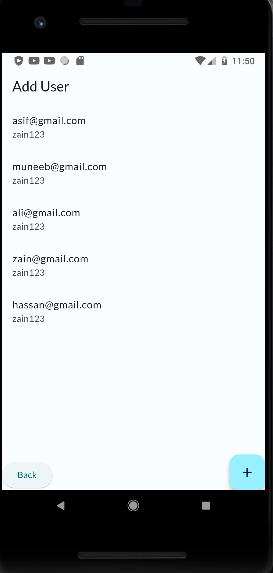
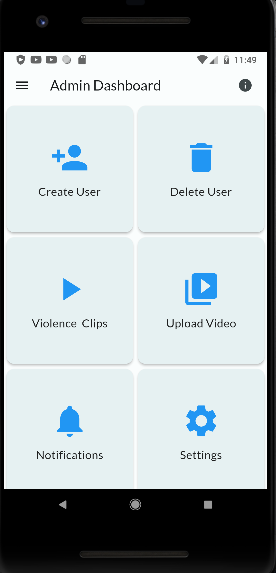
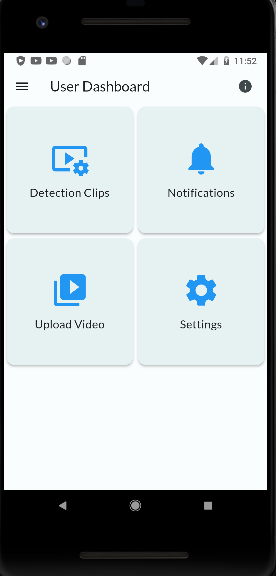
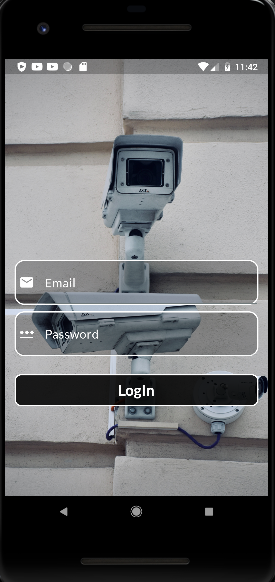
**Considerations:**

Clearly state privacy policies and terms of use.

Implement secure authentication methods.

Provide users with control over their data and account settings.

**Screen images**



**Appendix I**

* How to design using UML (OOP): For guidance please follow the instructions mentioned in the link: http://agilemodeling.com/artifacts/
* How and when to design ER diagrams: For guidance please follow the instructions mentioned in the link:

<http://people.inf.elte.hu/nikovits/DB2/Ullman_The_Complete_Book.pdf>

* Data flow diagrams: For guidance please follow the instructions mentioned in the link and book:
  + http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm
  + Software Engineering –A Practitioner’s approach by Roger Pressman
* Architecture diagram: For guidance please follow the instructions mentioned in the link and book:
  + Ian Sommerville – Software Engineering 9th Edition– Chapter 6