Spring 2024

Software Requirements Specification

SECUREGUARD: REAL-TIME SECURITY OBJECT DETECTION AND FACIAL RECOGNITION SUITE

Gladwin Chan

Bregwin Jogi

Sripalan Nishanth

Nicholas Lee

Executive Summary

Background

Current security camera systems are prone to errors and inefficiencies due to reliance on simply recording the footage and reviewing it in case of suspicious events. Similarly, attendance systems for schools and workplaces require manually checking to see if students arrived or entering a pin to confirm check in times for part time employees.

The proposed project aims to resolve both issues through developing an advanced security system with real-time facial recognition and object detection, including fire, animals, weapons and more. This will reduce the need for a human surveillance, increased security, reduce errors, and improve user convenience.

Description

The proposed application is a real-time facial recognition system designed to enhance security and attendance tracking. Traditional methods relying on manual check-ins, keycards etc. are prone to errors. This application provides seamless, accurate and efficient identification capabilities to sectors such as small businesses, homes and those interested in improving home security. This system will be hosted locally, and users will access via a web application to detect objects, faces and other foreign objects.

Company Value Add

Implementing a security system with real-time facial recognition, we position our company at the cutting edge of security technology. This offers users an automated software solution to reduce manual surveillance and reduce errors and improve user convenience. This offers a competitive edge by enhancing operational efficiency and reducing security risks.

End-User Value Add

The application simplifies the check in process, monitors unauthorized access, enhances security, and operates more efficiently than manual processes, allowing users to focus on core activities without being inundated with worrying about security and/or attendance tracking.

Scope

What is Included

The project will include:

- Real-time object detection for people, animals, weapons, and fire hazards.
- Facial recognition to compare facial features that match with the database.
- Attendance/Check-in system using the facial recognition.
- Local Recognition Program and online server/dashboard for adding users to be detected and configuring security settings.

What is Not Included

The project will not include:

- Physical security hardware
- Developoment of a mobile application
- Multi-factor authentication
- Implementation with third-party systems

Justification

Today's world we value security, as it prevents unauthorized personnel from gaining access to sensitive information and areas. By developing a real time facial recognition application, this will be a great learning experience for the team as the project would take lots of time and effort to build because developing facial recognition application at a college level will be challenging. Many challenges will occur throughout the project such as understanding and implementing complex algorithms using image processing like OpenCV, and PyTorch. To add on, the security needs to be accurate to reduce the risk of unauthorized access. Therefore, lots of testing and debugging will be needed throughout the project timeline. This project of developing a real time facial recognition application will enhance the teams technical skills in software development, image processing and cybersecurity justifying the substantial value of 2 course credits.

Section 1 – Introduction and Overview

1.1 Document Authors

- Gladwin Chan
- Bregwin Jogi
- Sripalan Nishanth
- Nicholas Lee

1.2 Document Revision History

WEEK	DATE	Revisions
3	May 22, 2024	Updated Section 1 Introduction & Overview, 2.1 Project Proposal, and 2.2 Stakeholders and Users
4		
5	June 11, 2024	Updated Section 2.3 Requirements
6	June 17, 2024	Updated Section 2.4 Project Scope, 2.5 System Risks, 2.6 Operating Environment, and 2.7 UI/UX Interface Mockups
7		
8		
9		

1.3 Document Purpose

The Software Requirements Specification (SRS) document serves as a comprehensive guide, outlining project requirements, objectives, and constraints. It covers project architecture, timeline, and limitations, ensuring clarity in defining features and quality attributes. Additionally, it includes a detailed timeline with milestones and strategies to address constraints. Assumptions are documented to manage expectations. The purpose of the document is ultimately guiding project development while aligning with stakeholder needs.

1.4 Audience

The intended audience can be broken into two categories: intended document audience and intended application audience.

1.4.1 Intended Document Audience

Project Sponsor, Development Team for reference

1.4.2 Intended Application Audience

Small Businesses, Schools, Home Security oriented individuals

1.5 Group Agreement

Team # 4

Project Title

SecureGuard: Real-Time Object Detection and Facial Recognition Suite

Project Time Frame

The project time frame would total 27 weeks:

- Planning – 3 weeks
- Analysis & Design – 10 weeks
- Implementation – 11 weeks
- Maintenance – 2 weeks

Team Members

Gladwin Chan

Bregwin Jogi

Sripalan Nishanth

Nicholas Lee

Team Leadership

Sripalan Nishanth

Team Functions/Roles

Database Developer

Front End Developer

Back End Developer

Systems Developer

QA/QC Tester

Development Environment

- Information will be shared through MS Teams, OneDrive, email, and in-person/online meetings.
- GitHub: brokoli777/SecureGuard (github.com)
- IDE: Visual Studio Code
- Project Management Tools: Notion, Jira, Trello, GitHub PM <URL>

Team Meetings

- One meeting one-line – Wednesday (12:25PM) with professor partial participation.
    - Week 1 – 7: 15 minutes, Week 8 – 13: 30 minutes
- One meeting in-person – Monday (3:20PM – 5:05PM) B1024 with professor partial participation

Team Problems

1. Meeting deadlines:
    - Ensure all tasks are completed on time by setting clear deadlines, regularly check ins, providing support for one another, and allocating enough time for each task.
2. Equal participation:
    - Ensure all team members contribute equally by assigning responsibility, and open communication.
3. Differing opinions
    - Foster a collaborative environment by respecting others idea and actively listening to each other.

Team Commitment

Example: The undersigned members agree to work together on the project until the end of the PRJ666 next Semester. They recognize that as a team and individually they are equally responsible for the quality of all deliverables.

Name	Date	Signature
Sripalan Nishanth	2024-05-22	
Gladwin Chan	2024-05-22	
Bregwin Jogi	2024-05-22	
Nicholas Lee	2024-05-22	

Section 2 – Project Overview

2.1 Project Proposal

2.1.1 Project Background

Current security camera systems are prone to errors and inefficiencies due to reliance on simply recording the footage and reviewing it in case of suspicious events. Similarly, attendance systems for schools and workplaces require manually checking to see if students arrived or entering a pin to confirm check in times for part time employees.

The proposed project aims to resolve both issues through developing an advanced security system with real-time facial recognition and object detection, including fire, animals, weapons and more. This will reduce the need for a human surveillance, increased security, reduce errors, and improve user convenience.

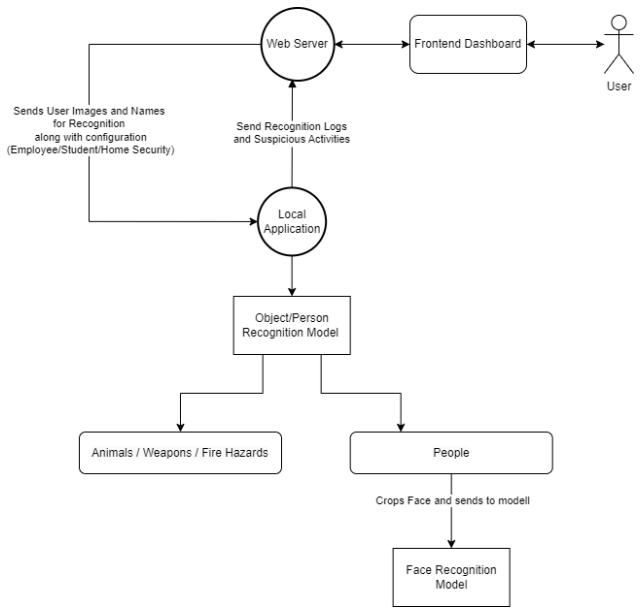
2.1.2 Problem Statement

Current security and attendance tracking systems are prone to errors, fraud, and inefficiencies due to reliance on manual check-ins, keycards, and passwords. These traditional methods lack real-time, accurate identification capabilities, leading to potential security breaches and administrative burdens. The proposal aims to develop a real-time facial recognition application to provide seamless, accurate, and efficient security and attendance management, thereby enhancing security, reducing errors, and improving user convenience.

2.1.3 Product Vision

The proposal is to develop a real time object detection and facial recognition application to enhance security and attendance tracking. Users will upload their photos and the application will detect and recognize faces as well as harmful/dangerous events (knives, weapons, fire hazard) real-time with advanced algorithms and a secure database.

This system will be used to automate check-ins and check-outs, provide accurate identification, generate alerts for unauthorized access, and maintain a comprehensive log of events. The solution leverages object detection and facial recognition technology and secure database management to deliver a seamless, efficient and secure user experience.



2.2 Stakeholders and Users

The main stakeholders for the Secure Guard real-time facial recognition application would be:

1. Developers/Technical Team: Their focus lies in enhancing system capabilities and addressing the deficiencies present in today's security systems. It is the development team that will come up with new functionalities and ensure seamless integration within the application. The primary focus is to guarantee optimal detection across all functionalities.

2. Users/Administrators: They rely on the application for both authentication, workplace hazard detection, and depends on its accuracy. It serves not only for attendance tracking but also enhances security measures.

3. Members: They want to ensure that their personal information (images, name, email etc.) is being stored securely and that the application detects them accurately without recognizing them as a different person.

2.3 Requirements

The following requirements tables list the requirements; sorted by each functional area of the system. Requirements are divided into a top level of classification: Functional and Non-Function Requirements.

Functional: Things the system must do, tasks user can complete within the system

Non-Functional: Properties the system must have: Operational, Performance, & Security Requirements

2.3.3 Requirements Prioritization

Each requirement is classified under a level of priority within the scope of the project: *<A good place to start is with is the following, feel free to modify this to match your project>*

- (MH) MUST HAVE:
    - Real-time object detection for people, animals, weapons, and fire hazards.
    - Facial recognition technology for accurate identification.
    - Secure database management for storing photos, user information, recognition logs.
    - Database with event logs for object detection timestamps.
    - User dashboard for managing settings and adding users.
    - Local recognition models for real-time detection.
- (SH) SHOULD HAVE:
    - Footage recording and uploading to an online server.
    - Informing users about unauthorized access or dangerous events.
- (NH) NICE TO HAVE:
    - Analytics and insights generation from event logs.

2.3.1 End-User Interface (Online Web App/Dashboard)

2.3.1.1 General Requirements

Functional Requirements

Req. #	Requirement	Priority
R01	Each page must have header.	MH
R02	Each page must have a footer.	MH
R03	Each page should have a navigation bar.	MH
R04	404 error page must indicate that the requested page was not found.	MH
R05	System automatically logs user out after 30 minutes of inactivity, with an option to extend 2 minutes before timeout.	NH
R06	The web application will be built using React.	MH
R07	The web application will have consistent UI using libraries such as Bootstrap and/or React UI.	MH
R08	The API requests will be handled through Express.js	MH

Non-Functional Requirements

Req. #	Requirement	Priority
R09	Each page must have header including the logo and a navigation bar.	MH
R10	Each page must have footer with contact information.	MH
R11	Each page must have responsive web design.	MH
R12	Each page must load within 3 seconds.	MH
R13	The user interface must be user-friendly for ease of use.	MH

2.3.1.2 Login and Registration

Functional Requirements

Req. #	Requirement	Priority
R14	Login page form should include a space to input email address, password, username, a link to registration page, and a forget password option	MH
R15	Users can login via email or username	SH
R16	First Name is required, must be alphabetic characters, between 2-50 characters	MH
R17	Users can also use their google account for authentication.	MH
R18	Last Name is required, must be alphabetic characters, between 2-50 characters	MH
R19	Username is required, must be unique, alphanumeric characters, between 5-15 characters	MH
R20	A valid email address is required and must be unique	MH
R21	The password is required and must be 8 to 20 characters long. It should include at least one uppercase letter, one lowercase letter, one digit, and one special character.	MH
R22	Password must be confirmed, the second password must match the first password	MH
R23	Users must be able to recover password using their email	MH
R24	Users must be able to recover username using their email	MH
R25	Security Question 1: User can select from predefined options	MH
R26	Security Answer 1: User's answer must be between 3-50 characters	MH
R27	Security Question 2: User can select from predefined options	MH
R28	Security Answer 2: User's answer must be between 3-50 characters	MH
R29	Security Question 3: User can select from predefined options	MH
R30	Security Answer 3: User's answer must be between 3-50 characters	MH
R31	User must agree to terms and conditions	MH
R32	Passwords may not be re-used on the same account	MH
R33	User must receive email confirmation to create account	MH

Non-Functional Requirements

Req. #	Requirement	Priority
R34	System will use secure hashing to store passwords	MH
R35	Display user-friendly error messages for any validation failures	SH
R36	Ensure registration form is accessible with screen reader	SH
R37	Ensure forms are keyboard navigable	MH
R38	Account will lockout after 3 unsuccessful attempts to log in	SH
R39	Locked accounts can be recovered via users' email	MH

2.3.1.3 User Profile Page

Functional Requirements

Req. #	Requirement	Priority
R40	The profile page must display user information including name and email	MH
R41	Users must be able to edit their personal information (name and email).	SH
R42	Users must be able to add their profile picture by uploading new image.	NH
R43	User must be able to change their password.	SH
R44	The profile must provide an option to enable or disable 2FA.	NH
R45	The profile page must display user's account activity log (last login).	NH
R46	The profile page must provide option for users to delete their account.	NH
R47	The profile page must have a link to redirect to user image upload page.	MH
	The profile page must have option for user to manage notifications.	MH

2.3.1.4 User Image Upload Page

Functional Requirements

Req. #	Requirement	Priority
R48	The system should support common image formats such as JPG, JPEG and PNG.	MH
R49	Users should be able to upload images from their local machine.	MH
R50	Users should be able to delete uploaded images if needed.	SH
R51	The page must display a preview of image uploaded before final upload.	SH
R52	The page must ensure uploaded images adhere to size and dimensions.	SH
R53	The system should compress images reduce file size.	NH
R54	The page must provide feedback regarding suitability of uploaded image.	NH
R55	The system should notify users if image is successfully or unsuccessfully uploaded.	MH
R56	Cloudinary/Supabase can be used to store User Images for the server.	MH

Non-Functional Requirements

Req. #	Requirement	Priority
R57	The system should be able to handle image uploads from multiple users concurrently without performance issues.	MH
R58	The system should store images in database while maintaining image quality.	MH
R59	The uploaded images should be scanned for viruses.	NH
R60	The upload functionality should be done through a secure and encrypted protocol.	SH

2.3.1.5 Security Events Log

Functional Requirements

Req. #	Requirement	Priority
R61	The events log must be presented in a tabular format for easy viewing.	MH
R61	The page must display a list of events captured from camera, including timestamps (date and time), descriptions (name and event).	MH
R62	The page can display relevant video clips or images captured from camera.	NH
R63	Users must be able to filter all events by date, time, name and type of events.	MH
R64	Users must be able to sort by date, time, name, and type of events.	MH
R65	By default, the logs are sorted by date and time.	MH
R66	Users should be able to export logs into different file formats including CSV, PDF and text.	MH
R67	The page should provide search functionality to find events.	MH
R68	There must be either be pagination for the events logs or dynamically display more logs when the user scrolls.	MH
R69	Different types of events can be displayed in different colors. (Green for Authorized Individuals, Red for Unauthorized People or Fire Hazards, Yellow for others)	NH

Non-Functional Requirements

Req. #	Requirement	Priority
R70	The data should be only be accessible with proper authorization.	MH
R71	The table should be responsive allowing for easy viewing across different devices.	MH
R72	The event log page should take less than 2 seconds to load.	MH
R73	The exported event logs should take less than 2 seconds to generate.	MH

2.3.1.6 Check In/Check Out Page

Functional Requirements

Req. #	Requirement	Priority
R74	System must be able to check the user in and out.	MH
R75	System must be able to display a confirmation message when check-in is successful.	NH
R76	Users will receive notifications regarding their check-in and check-out status.	NH
R77	The systems events log will be updated with the check-in and check-out time.	MH
R78	Users can view their past check-in history.	NH

R79	Users can view their past check-out history.	NH
R80	Check-in and check-out notifications on mobile devices	NH
R81	Check-in and check-out records will show date, time and user	MH
R82	Page will display user, date and time for checking in and checking out	MH
R83	Admins must be able to see all check-ins and check-outs	MH

Non-Functional Requirements

Req. #	Requirement	Priority
R84	Ensure that the check-in and check-out process is secure, protecting user data and privacy.	MH
R85	The system should have a uptime of at least 99% to ensure users can check in and check out reliably	SH
R86	The system should provide messages during the check in/check out process	MH
R87	The system provides users with informative error messages and guidance on resolving issues during the check-in/check-out process	

2.3.1.7 About Us Page

Functional Requirements

Req. #	Requirement	Priority
R88	The page will be accessible through the header or footer of any page.	MH
R89	The page will provide information about the company's mission.	NH
R90	The page will provide information about the members of the company.	
R91	The page will provide contact information such as email address.	
R92	The page will include a brief history of the company and highlight any milestones.	
R93	The page will provide links to social media profiles of the company for further engagement.	NH

2.3.2 – Web Application Hosting

Functional Requirements

Req. #	Requirement	Priority
R94	The web application will be hosted on Vercel.	MH
R95	The hosting platform should be setup to send notifications regarding downtimes and deployment failures.	NH
R96	The hosting platform should have ability to track performance, resource usage, traffic and errors.	NH

R97	The database will be stored on Supabase (a service using AWS) to ensure data durability and accessibility.	MH
R98	The web application hosting environment should support continuous integration and deployment (CI/CD) pipelines.	SH

Non-Functional Requirements

Req. #	Requirement	Priority
R99	System should ensure high availability.	MH
R100	System should be able to handle high traffic.	SH
R101	The website must use HTTPS protocol.	MH

2.3.2 Local Application Requirements

Functional Requirements

Req. #	Requirement	Priority
R102	The local application will be developed in Python.	MH
R103	The faces for each person to be detected should be stored in file storage and will be received through the API connection (POST request) from the online dashboard.	MH
R104	For each person a folder will be created in which all the images for that particular person will be stored.	MH
R105	The system must use local recognition models to ensure real-time detection without relying on external servers.	MH
R106	Pytorch, Keras or Tensorflow can be used to load the model.	SH
R107	The application must implement two recognition models: one for object detection and one for facial recognition.	MH
R108	The application must be able to be run on Windows operating systems.	MH
R109	The application must be able to run on Mac OS.	MH
R110	The application will request for access to system's camera on startup.	SH
R111	The system must preprocess images to enhance recognition accuracy (e.g., noise reduction, normalization).	MH
R112	Flask will be used to send and receive API requests.	MH
R113	The object detection model must run continuously to detect for people, animals, weapons and other events such as fire.	MH
R114	The application will call the facial recognition model when the object detected is a person.	MH
R115	Each detected object/person and facial recognition information must be passed to the online server via REST API POST requests.	MH
R116	The application should log detection events locally as well for backup purposes.	NH
R117	The local application should be able to send and receive API requests to the web application. (two-way communication)	MH

R118	When an unauthorized person is detected (Person's face not part of the user system), the system must send the detected frame, cropped image to the online server. (fields specified in Network/API requirements)	MH
R119	For running on Windows, the python code code can be converted into an .exe file for ease of use.	SH

Non-Functional Requirements

Req. #	Requirement	Priority
R120	The application should have a user-friendly interface.	MH
R121	The application should handle any errors and provide meaningful error messages.	MH
R122	The output from the object and image recognition models must be send to the online server as quickly as possible (under 30s) once complete to avoid delays in detecting dangerous event and people.	MH

2.3.2 Object Detection/Recognition Model Requirements

Functional Requirements

Req. #	Requirement	Priority
R123	An Object detection model would be retrained on existing datasets along with new pictures taken and labelled.	MH
R124	The system must utilize OpenCV for image preprocessing tasks such as resizing, normalization, and noise reduction.	MH
R125	The best object detection models have to be decided through research. Current candidates include Yolov4, Resnet and MobileNet	MH
R126	The model must be able to detect People.	MH
R127	The model can detect Fire/Smoke.	MH
R128	The model can detect Animals (general) and weapons	NH
R129	The system can use OpenCV features for face detection.	MH
R130	Once a person is detected, a cropped image of their face must be created to be passed to the face recognition model. (can use OpenCV for it)	MH
R131	If a person is detected but the face can be detected, a special log must be passed to the online server ("Person Detected but unable to detect Face. This might be possible due to covering the face or that the face was not faced towards the camera")	MH
R132	New Training Images should be labelled using a labelling software (For example: Roboflow) with bounding boxes and what object it is.	SH
R134	For each category to be trained on, the images must be stored into different folders. Each must be labelled with bounding boxes around the object.	SH
R135	Google Collab or online software like roboflow can be used for training since the local machines don't have enough system resources required for the task.	SH

Non-Functional Requirements

Req. #	Requirement	Priority

R136	The object detection model should have high accuracy detection rate (at least 70%).	MH
R137	The model should be scalable to accommodate additional categories in the future.	NH
R138	The model recognition speed must greater than or equal to 10 frames per second	MH

2.3.2 Facial Recognition Requirements

Functional Requirements

Req. #	Requirement	Priority
R139	The best object detection models have to be decided through research. Current candidates include Sface, Dlib and FaceNet.	MH
R140	The facial recognition system should support multiple face recognition (multiple people in single frame).	MH
R141	The facial recognition system should support recognizing faces under different lighting conditions.	MH
R142	When training the model or retraining existing model, different faces in multiple angles (preferably high level) must be added to the training dataset to increase accuracy when using security camera footage for detection.	SH
R143	The face recognition model must able to detect faces with glasses.	MH
R144	Faces Detected that are not part of the user images must be logged as “Unauthorized Person” to be passed to the server.	MH
R145	Faces that are recognized as a person added must be logged as “Recognized Person” along with their information and passed to the server.	
R146	Ensure the ML model is loaded when the application starts instead of when a person is detected since it takes a significant amount of time to load the model.	MH
R147	The model received a cropped face image of specified height and width from local application. If it is not the correct dimensions, the model throws an error.	SH
R148	The model loops through each image for each user stored in the local application and checks the similarity of the current image.	SH

Non-Functional Requirements

Req. #	Requirement	Priority
R149	The face recognition output confidence must be greater than 50% to be passed to the online server as recognized face. Otherwise, it should pass the output as “Unsure” along with the face with the highest probability of recognition.	MH
R150	The model recognition speed must greater than 2 frames per second	MH

2.3.3 Network / API Connection Requirements

Functional Requirements

Req. #	Requirement	Priority
R151	POST request from the local application to online server must have the following information: • ObjectID • ObjectName (Person, Animal, Weapon, Fire) • FaceDetected: Boolean (optional depending on if face is present) If detected: ○ FaceID ○ FaceName • ClientID • DateTime	MH
R152	POST Request from Online server to Local Application for passing Face Images to be recognized locally: Body: • FaceID • FaceName • FaceBuffer: (The image is passed as a buffer through the network) • FileType • DateTime • Client ID (to allow multiple admin users in the future)	MH
R153	For POST request from the local application to online server, the following are mandatory: ObjectID, ObjectName, DateTime, ClientID	SH
R154	For POST Request from Online server to Local Application for passing Face Images, all the fields mentioned are mandatory and must throw an error if fields are missing.	SH
R155	Authorization header must be send along with requests for data security.	MH
R156	The image recognition requests from the location application must be batched so that recognition logs from few seconds is sent together.	MH
R157	The date and time sent in requests must have either the timezonse send along with it or must be time-zone agnostic.	MH

R158	Health Check GET Request response for Local Application:
	Body: {
	dateTime:
	Status: “OK”
	Message: “Local Application Running”
	}
		MH

R159	If above request is not received from the local application, the server must display and error to user informing “Local Application is Offline or unable to be detected”	MH
R160	Health Check GET Request response for Online Server:	MH

	Body: {
	dateTime:
	Status: “OK”
	Message: “Online Server Running”
	}	

R161	If above request is not received from the online server, the local application must display an error to user informing “Server is Offline or unable to be detected”	MH
R162	All the API requests must have an authentication header along with it. If not, must throw an error	MH
R163	GET request for online dashboard to retrieve all the event logs. Secureguard.com/dashboard/events	MH
R164	GET request for online dashboard to retrieve all the persons to be detected for a particular admin. Secureguard.com/dashboard/person	MH
R165	GET request for online dashboard to retrieve only authorised people detected logs. Example: Secureguard.com/dashboard/events/?filter=authorized	MH
R166	GET request for online dashboard to retrieve unauthorised people detected logs (People whose faces were not part of images uploaded) Example: Secureguard.com/dashboard/events/?filter=unauthorized	MH
R167	GET request for online dashboard to retrieve animals detected logs. Example: Secureguard.com/dashboard/events/?filter=animals	MH
R168	GET request for online dashboard to retrieve fire hazards detected logs. Example: Secureguard.com/dashboard/events/filter=fire	MH
R169	GET request for online dashboard to retrieve fire weapons detected logs. Example: Secureguard.com/dashboard/events/filter=weapons	MH
R170	The GET requests to retrieve the event logs can have pagination passed as a query.	SH

R171	POST request for online dashboard to add a person to be detected
	Secureguard.com/dashboard/person
	{
	Name:
	Images: [ ]
	}
		MH

R172	PUT request for online dashboard to update a person to be detected
	Secureguard.com/dashboard/person
	{
	Name:
	Images: [ ]
	}
		MH

R173	DELETE request for online dashboard to update a person to be detected Secureguard.com/dashboard/person {	

	personID: }	

Non-Functional Requirements

Req. #	Requirement	Priority
R174	API requests will be asynchronous.	MH
R175	There must be proper error handling for requests from both servers for debugging. The error handler must print out the cause of error.	MH

2.3.5 Database Services

Functional Requirements

Req. #	Requirement	Priority
R176	The system must securely store face images and personal information including names and emails.	MH
R177	The images can be stored in Supabase or Cloudinary with the links send to the database.	MH
R178	The system must store other images for object recognition.	NH
R179	The database must support CRUD (Create, Read, Update, Delete) operations for all data.	MH
R180	The system must support backups and recovery procedures.	NH
R181	The system must ensure data integrity for all stored information.	SH
R182	The database must log user activity for security purposes.	MH
R183	The database must support efficient querying of images and personal information.	SH
R184	The database must support role-based access control for managing user permissions.	MH
R185	The database supports integration with external system through APIs.	SH
R186	The database should synchronize in real-time with web application.	SH
R189	System must store all check-in and check-out time records	MH
R190	System must store all users who have checked-in and checked-out	MH

Non-Functional Requirements

Req. #	Requirement	Priority
R191	The database must store both user information and detection logs	MH
R192	The system should have 90%+ uptime.	SH
R193	The system responds to database queries within 2 seconds.	SH
R194	The database should have low latency.	SH
R195	The system should have a maintenance window during off-peak hours	SH

R196	The database should store efficiently to optimize disk usage and performance.	MH
R197	The database should provide logs and error messages for troubleshooting.	MH
R198	The database should have sufficient capacity to store all data and images.	SH

2.3.6 – User Notifications

Functional Requirements

Req. #	Requirement	Priority
R199	The systems should send email notifications to clients' registered email addresses	MH
R200	Notifications should include clear subject line, indicating the nature of the event (e.g., "Animal Presence Detected,", "Fire Incident Alert")	MH
R201	Emails should provide clickable links or buttons in the email for quick access to more details or action items	MH
R202	Notification settings can be managed through the user account	SH
R203	Online user account notifications should be prominently displayed on the notification center	SH
R204	Users should be able to view all notifications on their online user account	MH
R205	Users should be able to click on notifications for full details and actions on their online user account	SH
R206	Users should be able to link a phone number for Text Message (SMS) notifications	SH
R207	SMS messages should include a URL or code for accessing more details	SH
R208	Online user notifications should allow push notifications for instant delivery	SH
R209	Dashboard alerts display prominently with colours for severity and summaries	NH
R210	Dashboard alerts should include acknowledgement/dismissal options	SH
R211	Notifications should have priority levels assigned based on severity	MH
R212	Popup notifications should be implemented for immediate attention	SH

Non-Functional Requirements

Req. #	Requirement	Priority
R213	Emails should be delivered within 5 minutes of triggering notification	SH
R214	Emails should use standardized email templates for consistency	SH
R215	Ensure prompt and reliable SMS delivery.	MH
R216	Notifications should have customized visual cues and actionable buttons	SH
R217	Customize notifications for each platform	SH
R218	Maintain data security and privacy in email communications	MH
R219	Optimize dashboard performance for smooth navigation	NH
R213	Ensure that all SMS communication channels are secure	SH
R214	Comply with SMS character limits and formatting guidelines.	SH

R215	Online accounts should ensure real-time (less than 1 second delay) or near-real-time (less than 5 seconds delay) update of notifications.	MH

2.4 Project Scope

The application to be delivered at the end of PRJ666 will allow the users to:

- Have real time facial recognition
- Person Detection
- Object detection: fire, weapons, animals
- Attendance Tracking
- Functional SecureGuard Application
    - Register
    - Admins to log activity
- Check in and check out notifications

The version of the application that will be released will include the following features:

- Online Dashboard
    - Ability to add individuals to be detected
    - Being able to view all recognition individuals
    - Registering as an Admin (account managing individual information)
    - Logging into the dashboard as an Admin
    - View all event logs
    - Filter based on the recognized individual's names.
    - Find detection information for each Member (Check In/Checkout) for each day
    - Filter based on objects (fire hazards, animals, weapons)
    - Edit individual Information – Add Name, Role and Upload images for each member for facial recognition.
- Local Application
    - Real time person detection
    - Facial recognition
    - Fire Hazard Detection
    - Weapon Detection
    - Animal Detection
    - Login UI for connecting application to online server for secure event logging
    - Download Individual Images from the webserver for local facial recognition.

The following features will not be included in the current version, but may be considered in a future version of the software:

- Email notifications for Notifications
- Improved facial recognition
- Advanced UI for the local application

Expected Timeline

- Project Start: Summer 2024 (May 26th, 2024)
- Project Completion: End of Winter 2024 (December 13th, 2024)

Estimated Cost

Total Cost: $0

As students, the team will work for free.

Free tools and servers will be utilized for development and deployment.

2.5 System Risks

All systems design and implementation processes have risk associated with them. The following is a list of risks that could potentially impact the ability for the application to be delivered on-time, within scope, and on budget.

RISK	RESPONSE
Implementing machine learning models for facial and object recognition is a new tool our team needs to learn, presenting a learning curve.	We will make sure we do thorough research and collaborate and support each other to ensure smooth learning process.
This development of a machine learning model and surveillance system may exceed our budget due to costs associated with computational resources, storage and hosting.	We will closely monitor expenses and as a group collectively agree on how much we are willing to spend and stay within that budget.
The facial recognition model needs to be accurate, otherwise it may result false positives and negatives, undermining the app's effectiveness.	We perform through testing and validation to ensure high accuracy and reliability. In addition, we will continuously monitor and address any issues.
API failure between local application and web application will disrupt functionality.	Develop API with error-handling in the event of API failure.
Hosting platform may experience downtime and disrupt the surveillance services.	Make sure to use a hosting platform that is reliable and has uptime guarantees, ensuring minimal disruption to the system.
Data privacy is a concern, it is imperative to safeguard customer information, photos and associated data.	Data will be encrypted, and role-based access controls will be implemented to protect customer information and photos.
Stored data may be lost due to disaster or human error, leading to permeant loss of customer information.	We will regularly backup our database to ensure redundancy. We will also use GitHub for version control of our code, enabling us track changes, and see previous versions.
The system is potentially vulnerable to hacks or malicious interference.	Implement strong security measures, use best practices to encrypt data.

As more users begin to use application, the application may face scalability issues such as increased server loads, and storage demands.	Implement a scalable architecture by using the cloud to address the increased server loads and storage demands efficiently.
Some third-party libraries and tools could cause some new compatibility issues and conflicts.	Implement compatibility testing so that third party libraries do not cause issues and conflicts and keep everything up to date.
Database performance issues from high volume or complex queries can delay response time.	Optimize the database queries to improve performance. Monitor and analyze database performance
Complex implementation of algorithms that might take time to understand.	Help other group members that are involved with complex algorithms and prior to coding, if one knows that there will be complex algorithms, start early to give enough time.
Deadlines are strict, as all deliverables must be complete by the end of the school term	Using tools such as GitHub projects and Gantt charts to visualize the timeline and any dependencies, the group can mitigate any bottlenecks or risks of incomplete deliverables by the expected delivery date. Consistent team meetings will help the group to avoid miscommunications and potential scope creep that could delay the project.
Potential issues with team members taking a semester earlier or later than the other members.	Our team is currently in agreement to maintain the same members for PRJ666 and align our enrollments, as either lacking a team member or replacing them with another member may delay deliverables and scope may have to be readjusted.
Potential risk that the application will require paid services to be functional as the current plan is to make use of only free applications and services.	As development progresses, the group may find that paid services are required to successfully provide a working application. In this case, the group will have a team meeting to discuss a budget and determine what we are comfortable allocating for the service.
Insufficient testing may lead to unintended behaviours, bugs or other issues that may lead to the application not functioning as intended.	The group should implement a rigorous testing suite, that utilizes industry standard methods such as Unit Testing, Integration Testing and Regression Testing. The group should also consider adopting test-driven development so that tests are created before the code is, to ensure sufficient coverage.

2.6 Operating Environment

- A custom API will be developed to facilitate data transfer between the local application and the server. This API will handle the transmission of event logs, image uploads, and face detection logs.

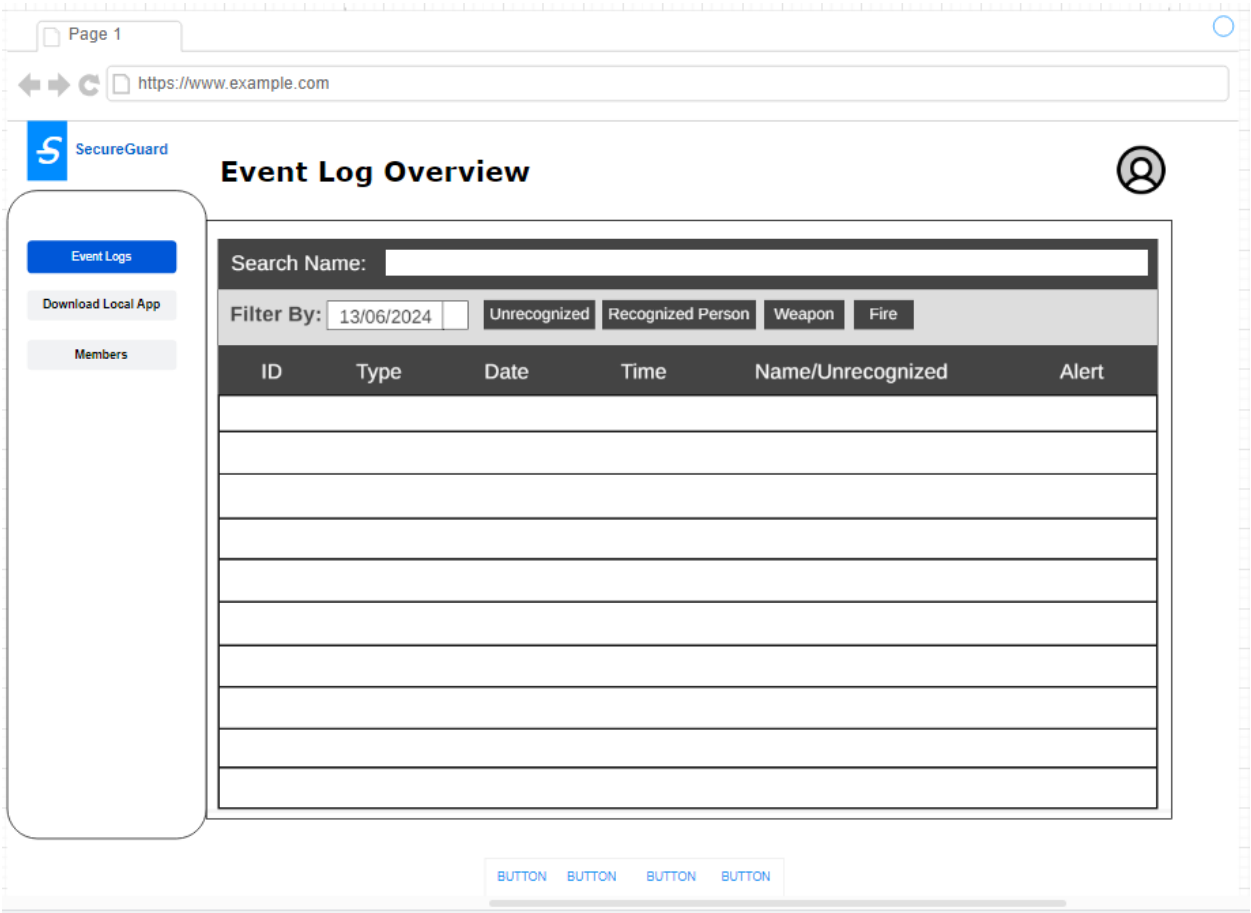
- The project requires custom hosting solutions. Image detection will be performed locally to enable real-time object and face detection. Therefore, an application will run on the user's computer connected to a camera (such as a security camera feed). The online server and website will be hosted on Vercel, Netlify, or Render, in that order of preference. The database will be hosted on Supabase, which also provides authentication services and stores user information.
- The online application is technically hosted on the cloud through the above services; however, no resources have to be managed as they are service providers on top of the cloud infrastructure.
- Enhanced security measures include encrypting API requests between the local application and online service using an Authorization Header. Users must log in when installing the local application to ensure it runs successfully. A sign-up and login page will be provided for users (Security Admins) to register and access the online dashboard.
- User hierarchy is not required, as only administrators will utilize the application. However, multiple administrators can be accommodated.
- The local application will initially support Windows, with potential support for macOS and Linux based on available development time. Whereas, the online dashboard, being web-based, allows users to check event logs from any device and location.
- The project has specific bandwidth requirements due to the large amount of data transferred between the local and online application. The local application requires an internet connection with speeds of at least 1 Mbps.
- Significant storage capacity is needed for the project, as the object detection and facial recognition models are large (approximately 250MB to 1GB). Additionally, facial recognition images must be stored on both the online server and the local application, with storage needs varying based on the number of individuals to be detected.
- Development will be using Visual Studio Code for Python (local application) and Node.js (online service).
- GitHub Actions will be utilized for Continuous Integration and Continuous Deployment to maintain the application.

2.7 UI/UX Interface Mock-ups

The following screenshots are an initial mock-up of the screens to be provided within the application. They are initially creating using wireframes and a content review, and later created with more defined graphics, look and feel, in addition to other user experience considerations.

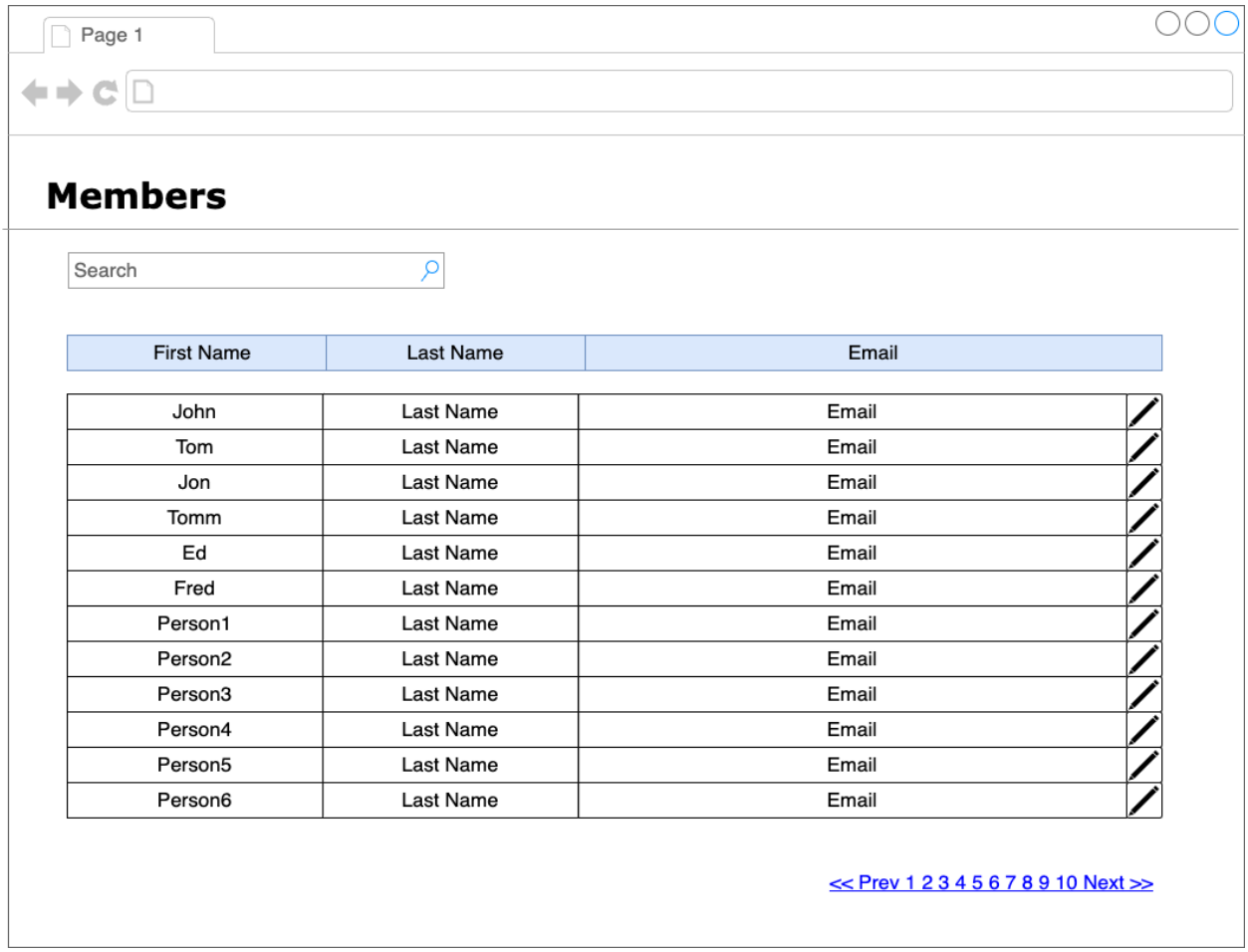
2.7.1

Home/Event Page:



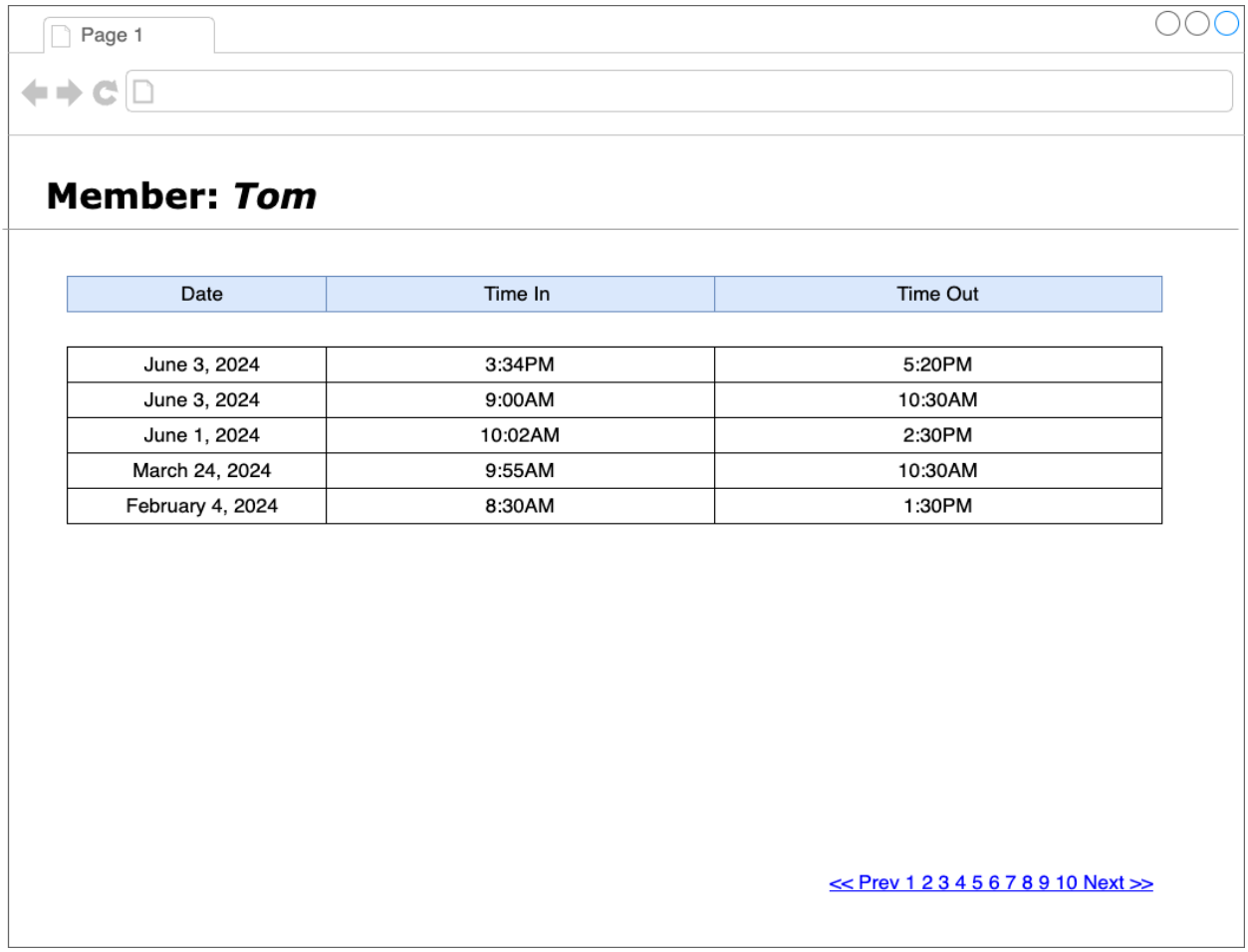
2.7.2

All Members Page:



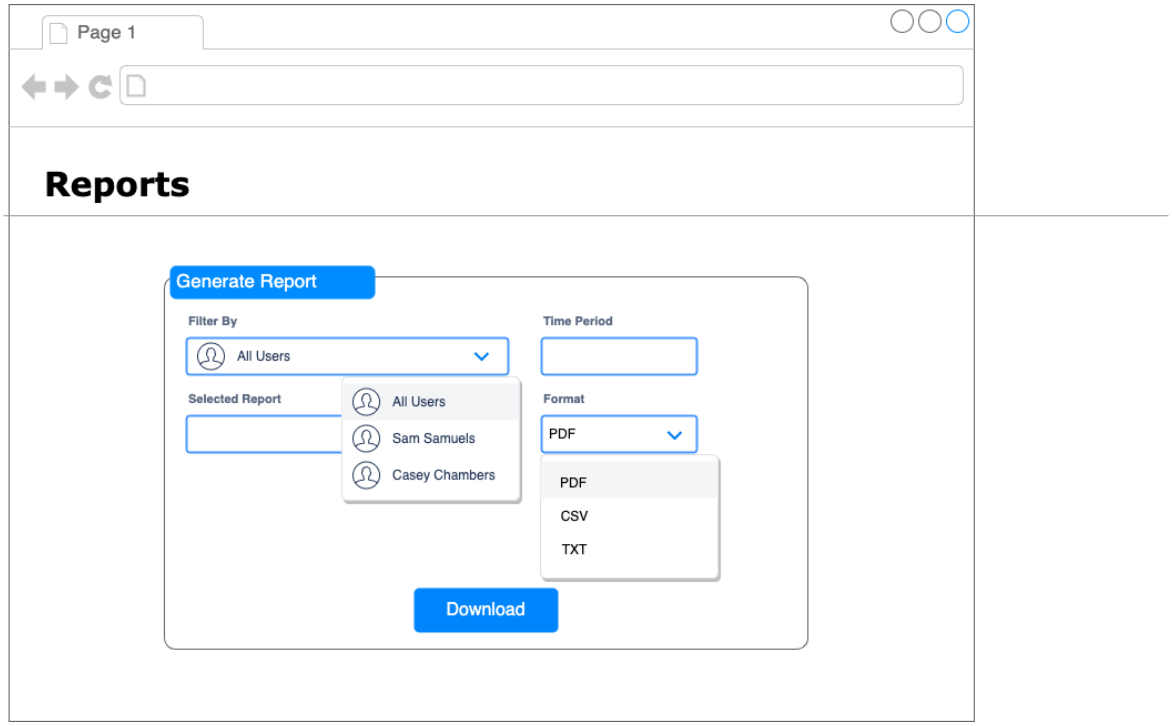
2.7.3

Individual Members Page (Check-in/Check-out):



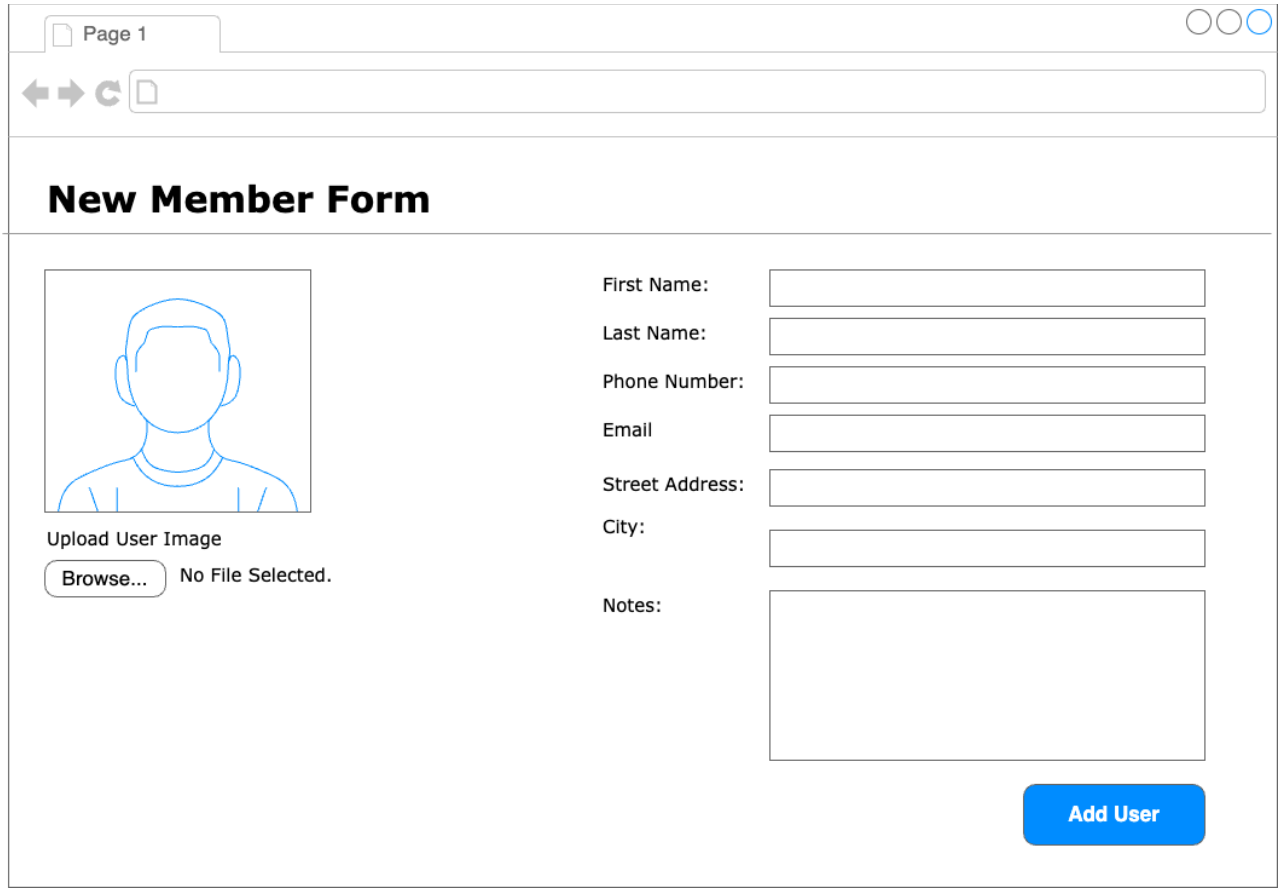
2.7.4

Generate Reports:



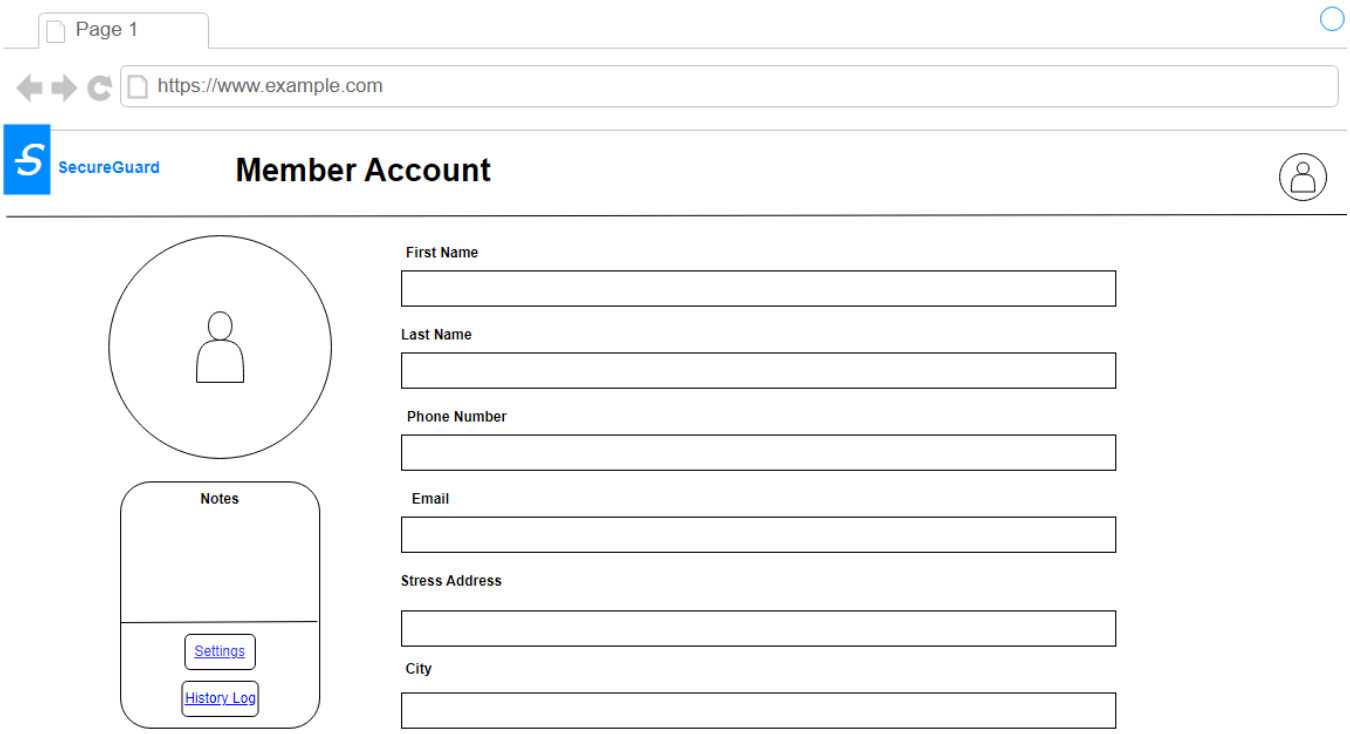
2.7.5

New User Form:



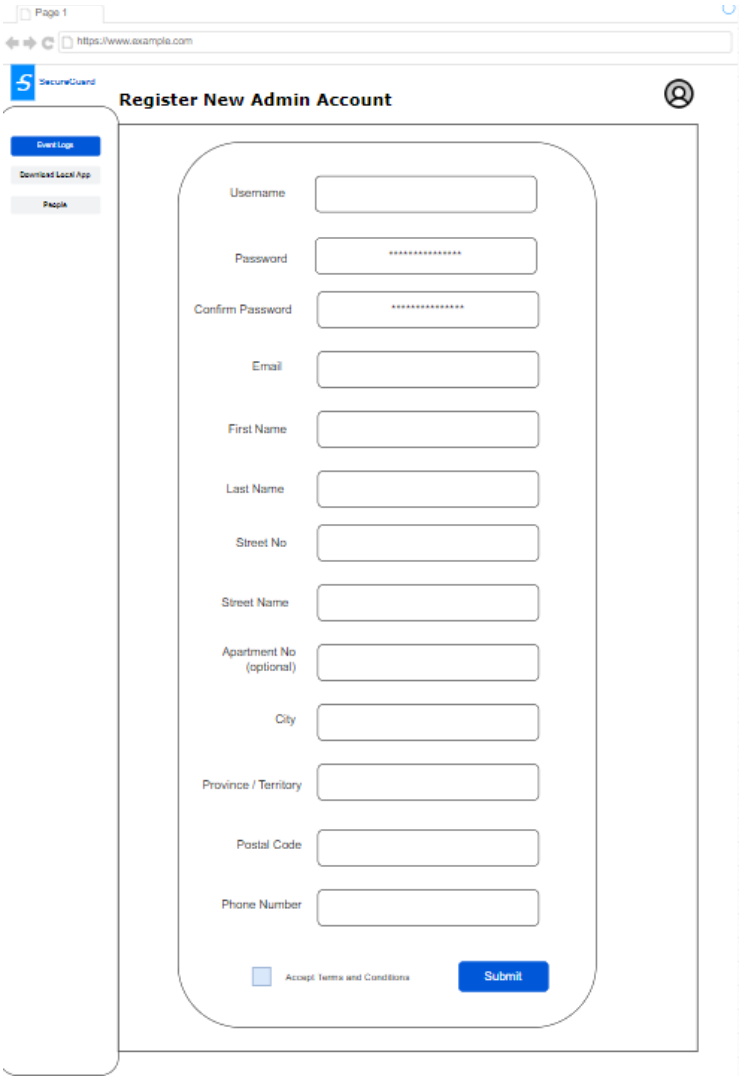
2.7.6

Member Account Page:



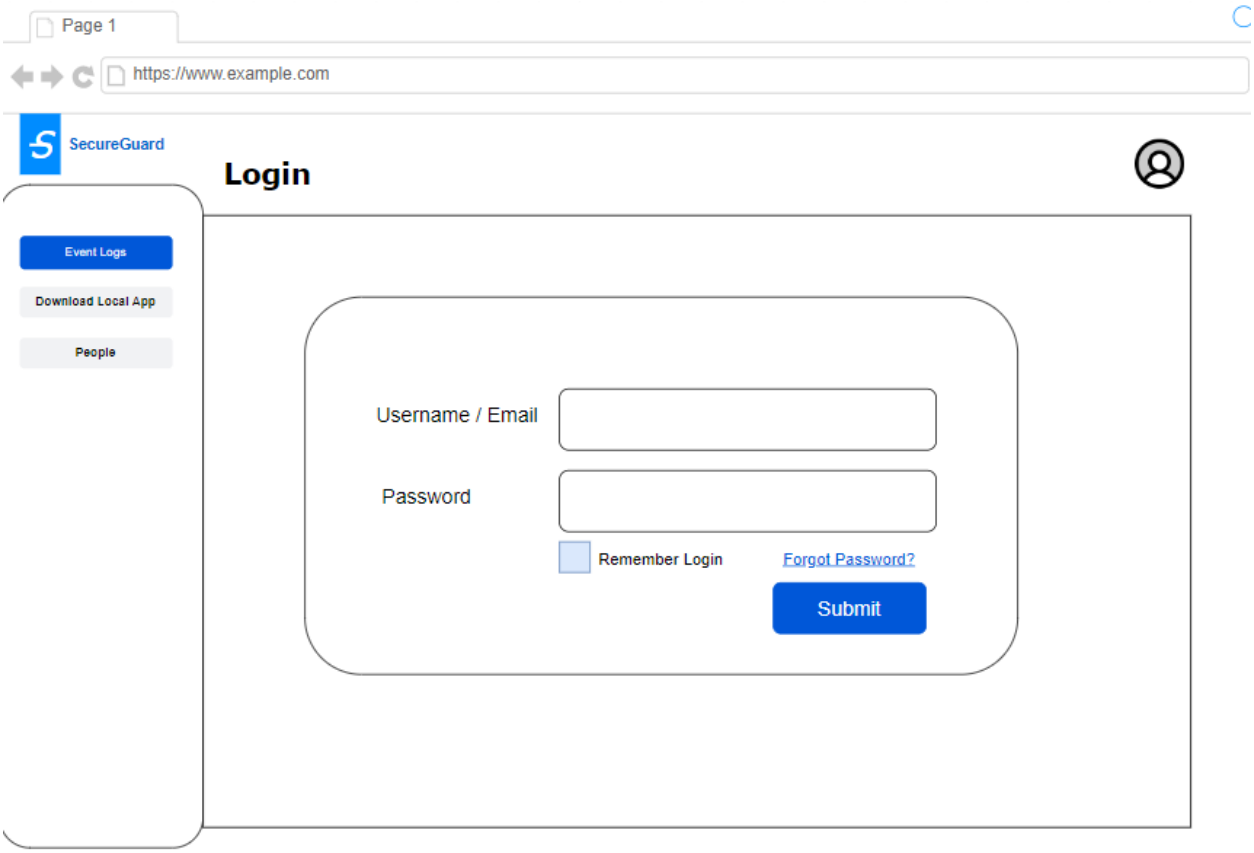
2.7.7

Register New Admin Account:



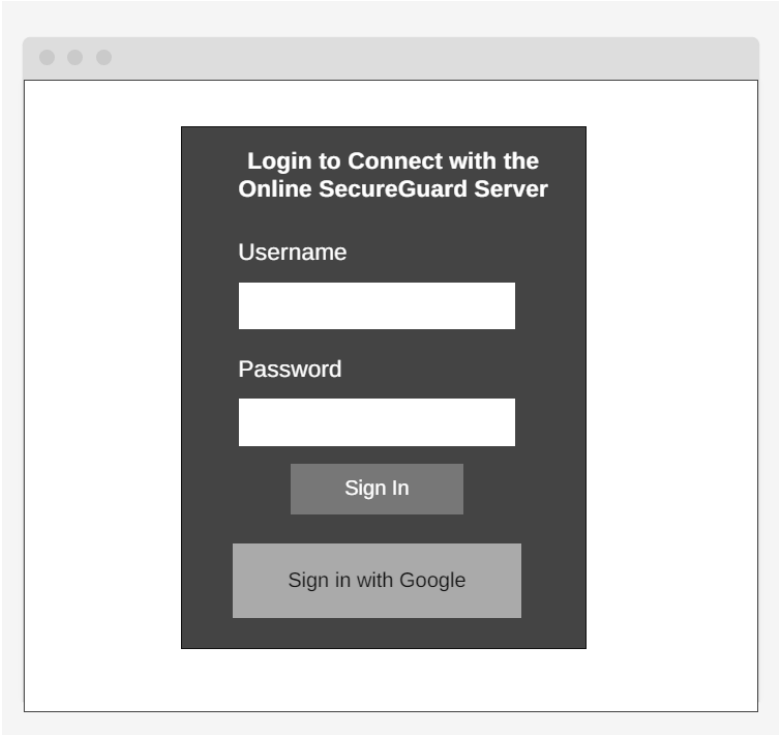
2.7.8

Login Admin:

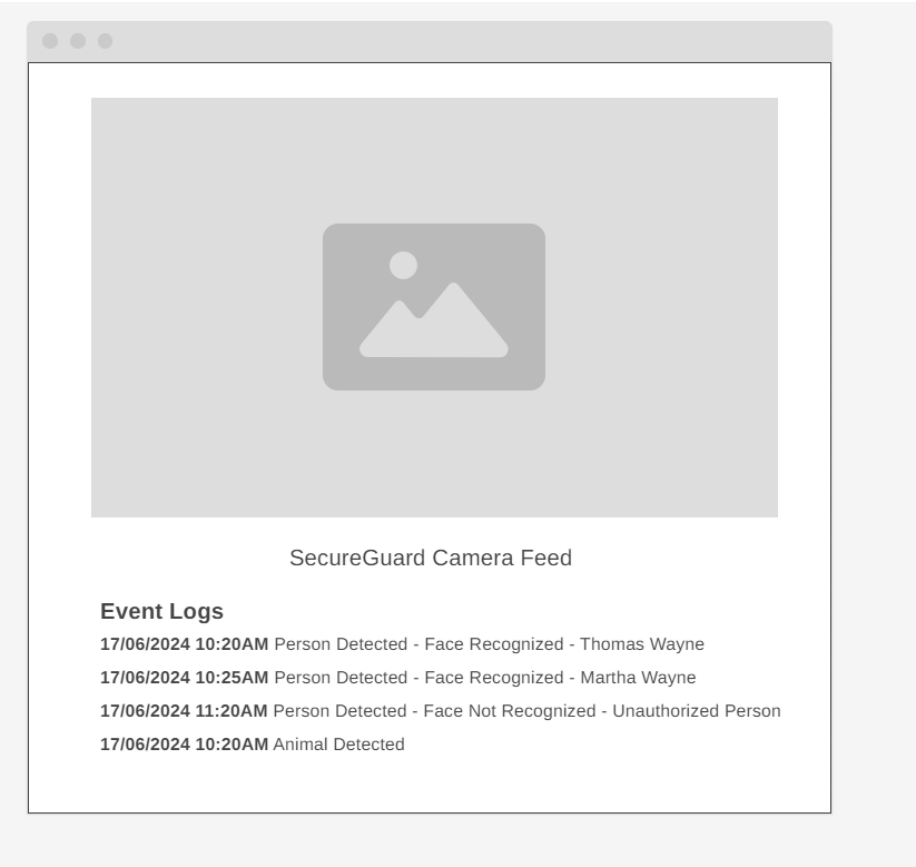


2.7.9

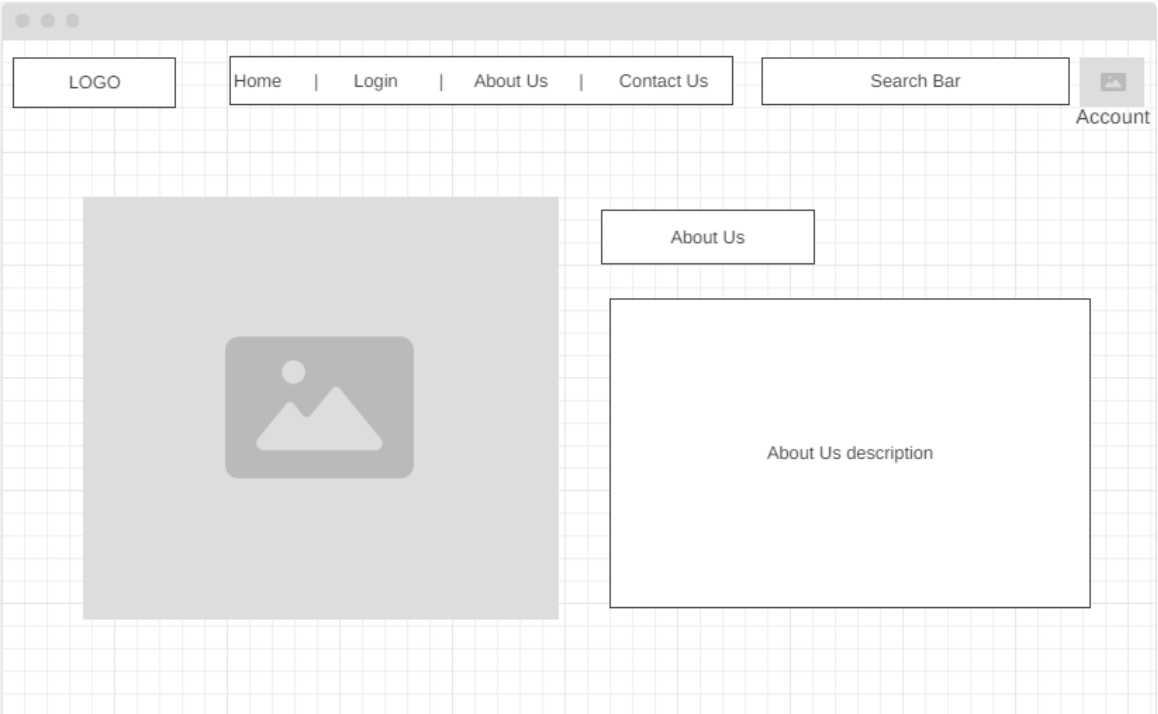
Local Application Login:



2.7.10 Local Application Running



2.7.11 About Us Page

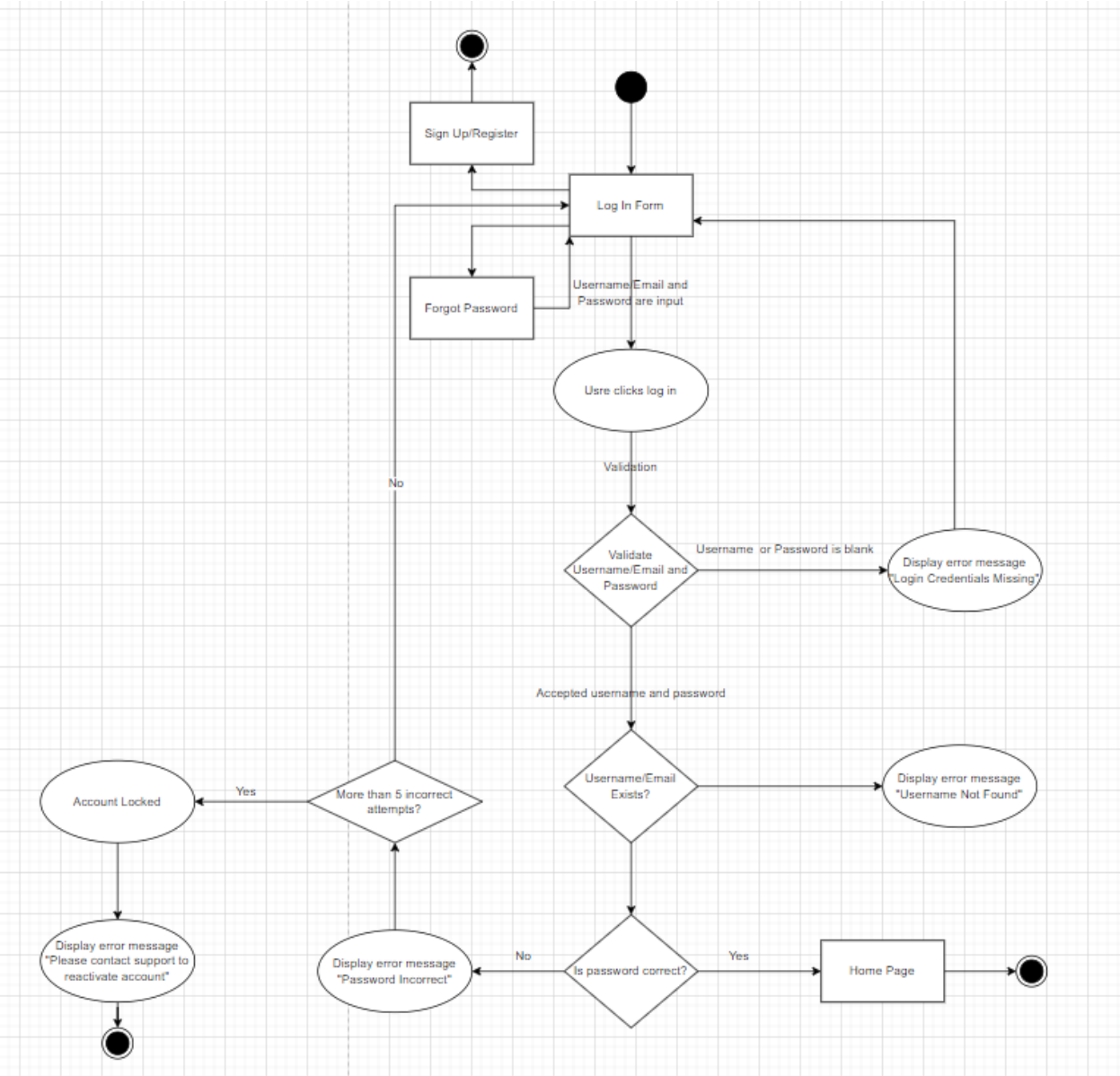


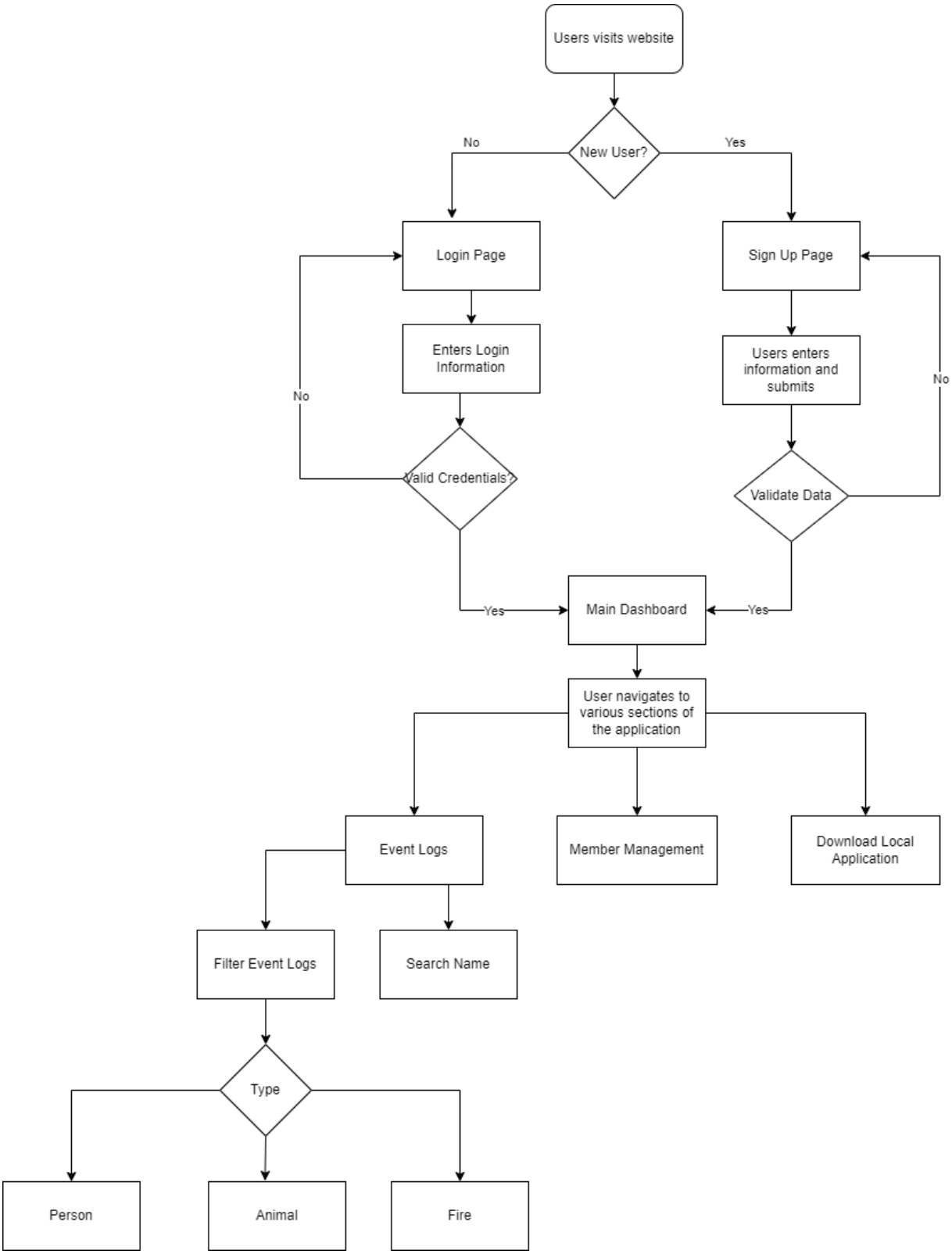
Section 3 – Process and Data Modelling

3.1 Workflow Diagrams

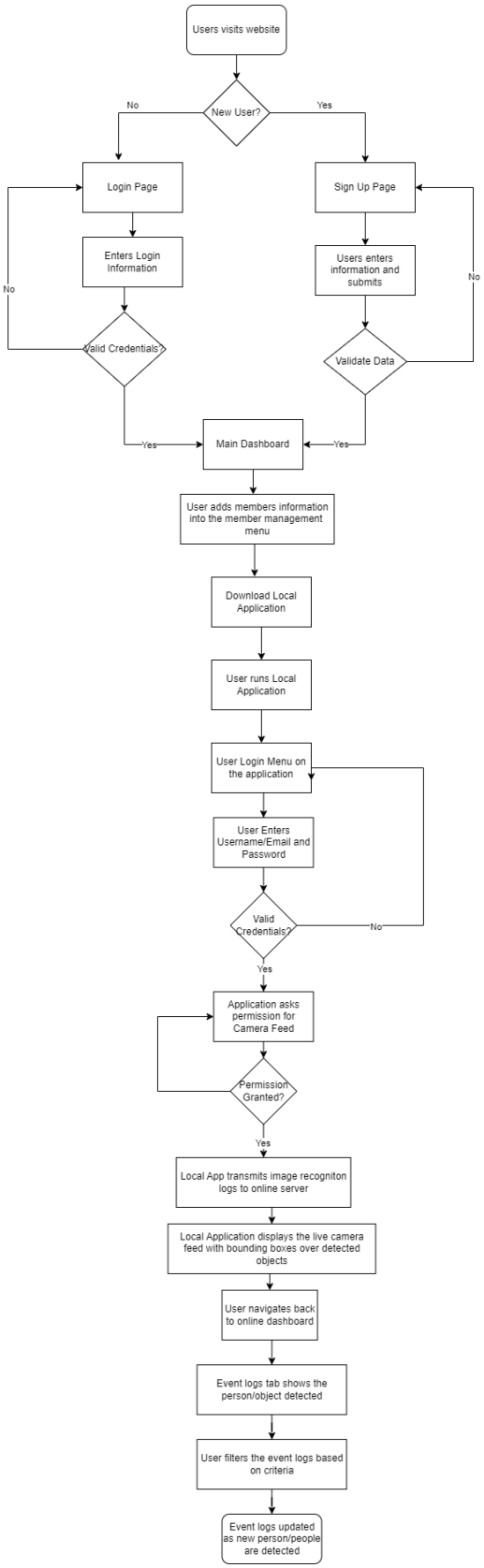
3.1.1 Application Navigation

Login Workflow




Users visits website
New User?
No
Yes
Login Page
Sign Up Page
Enters Login Information
Users enters information and submits
No
No
Valid Credentials?
Validate Data
Yes
Main Dashboard
Yes
User navigates to various sections of the application
Event Logs
Member Management
Download Local Application
Filter Event Logs
Search Name
Type
Person
Animal
Fire

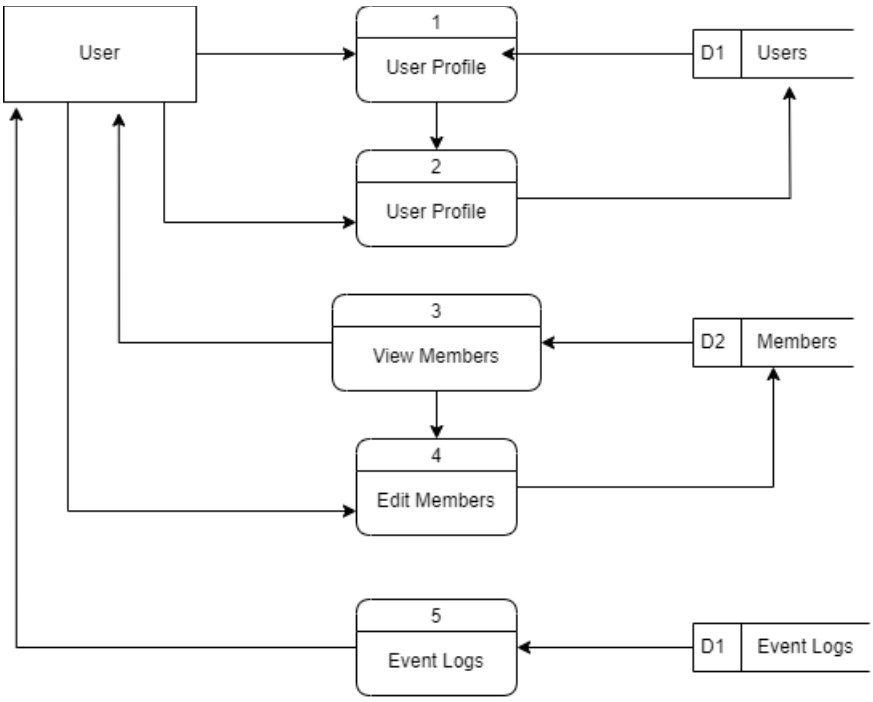
3.1.2 End User Workflow



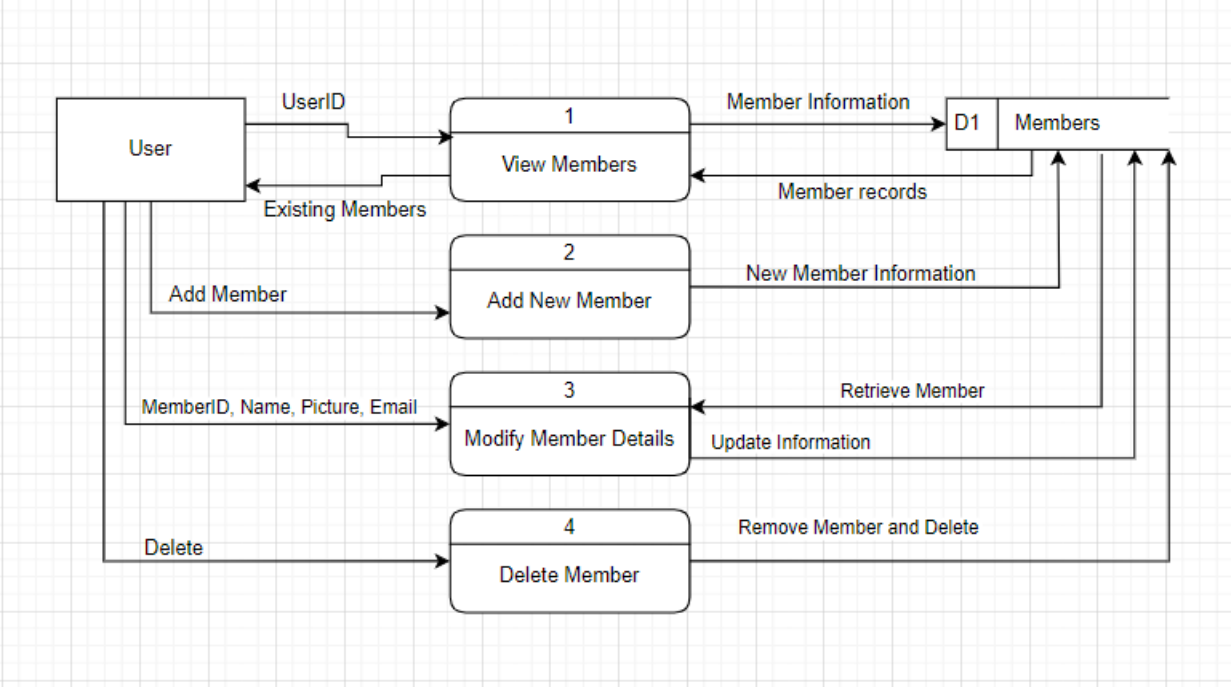
3.2 Data Modelling and Flow

3.2.3 Data Flow Diagrams

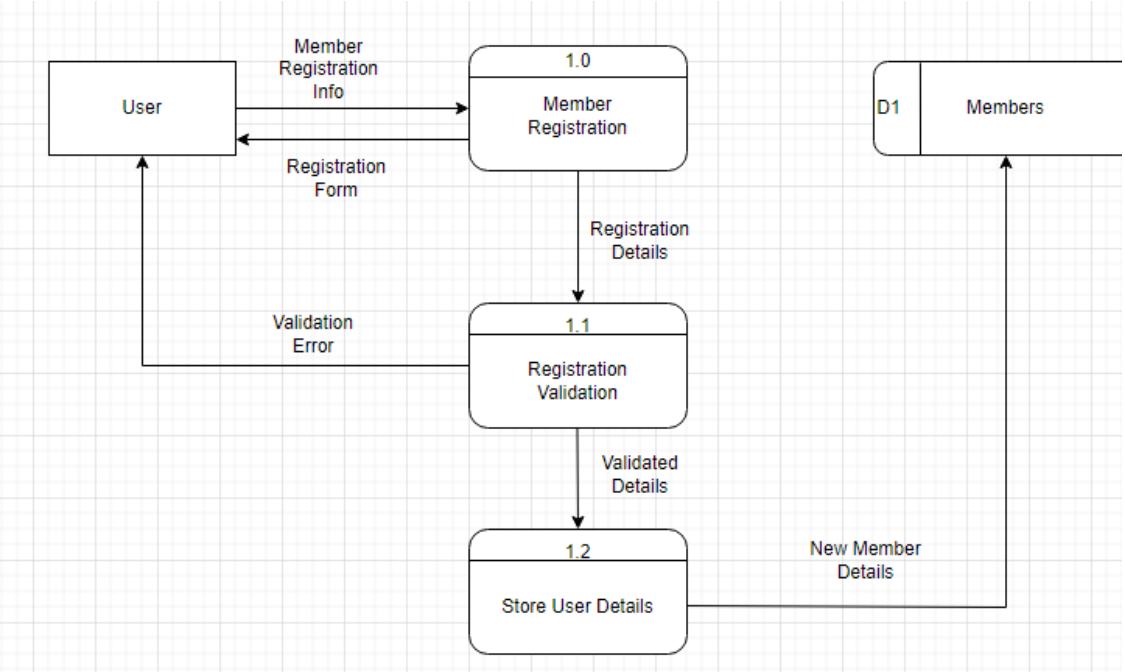
User Navigation Dataflow



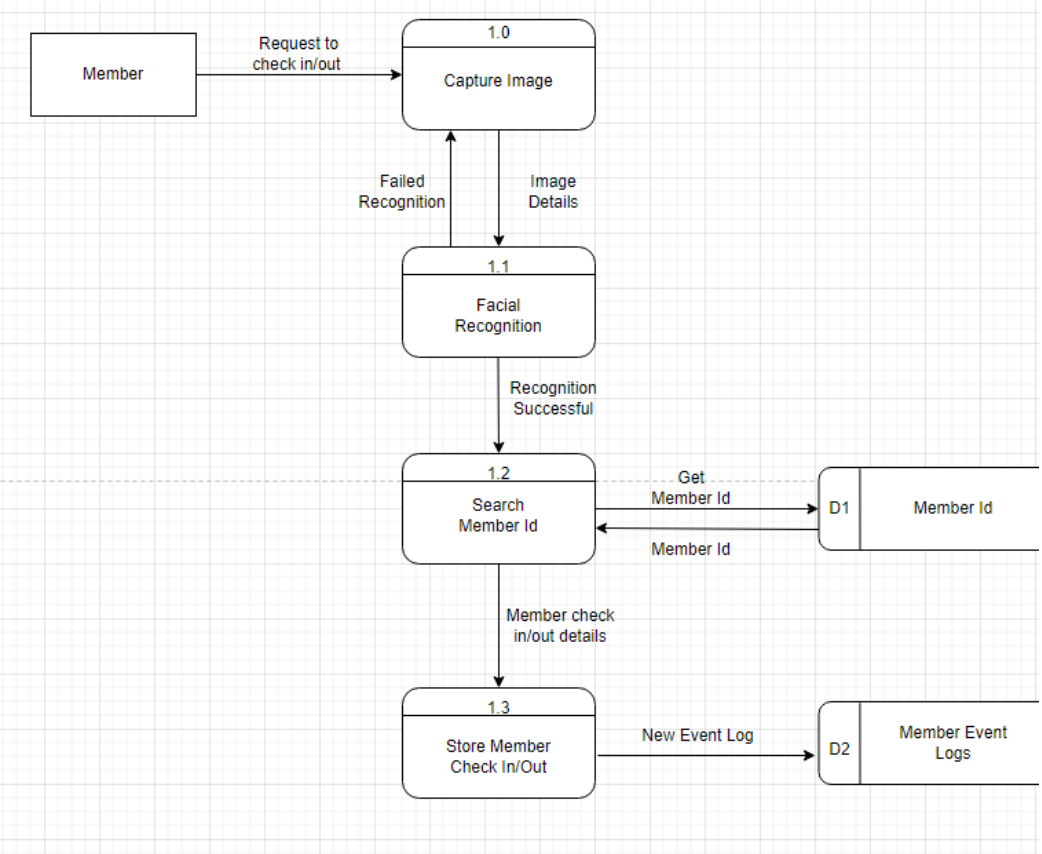
Member Management



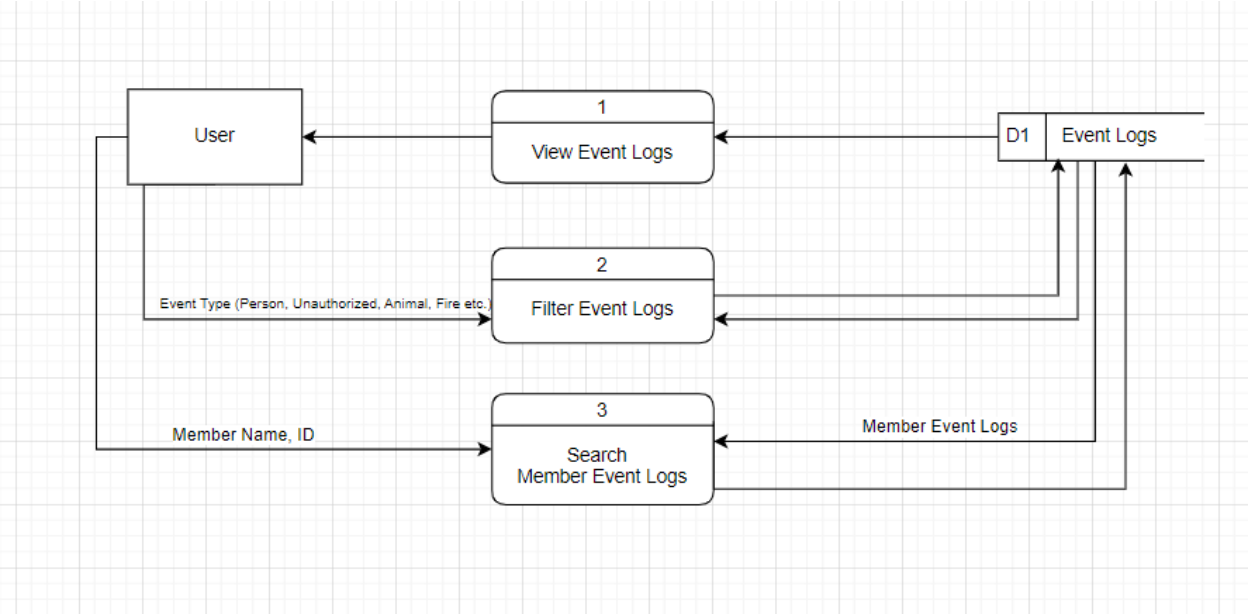
Member Registration

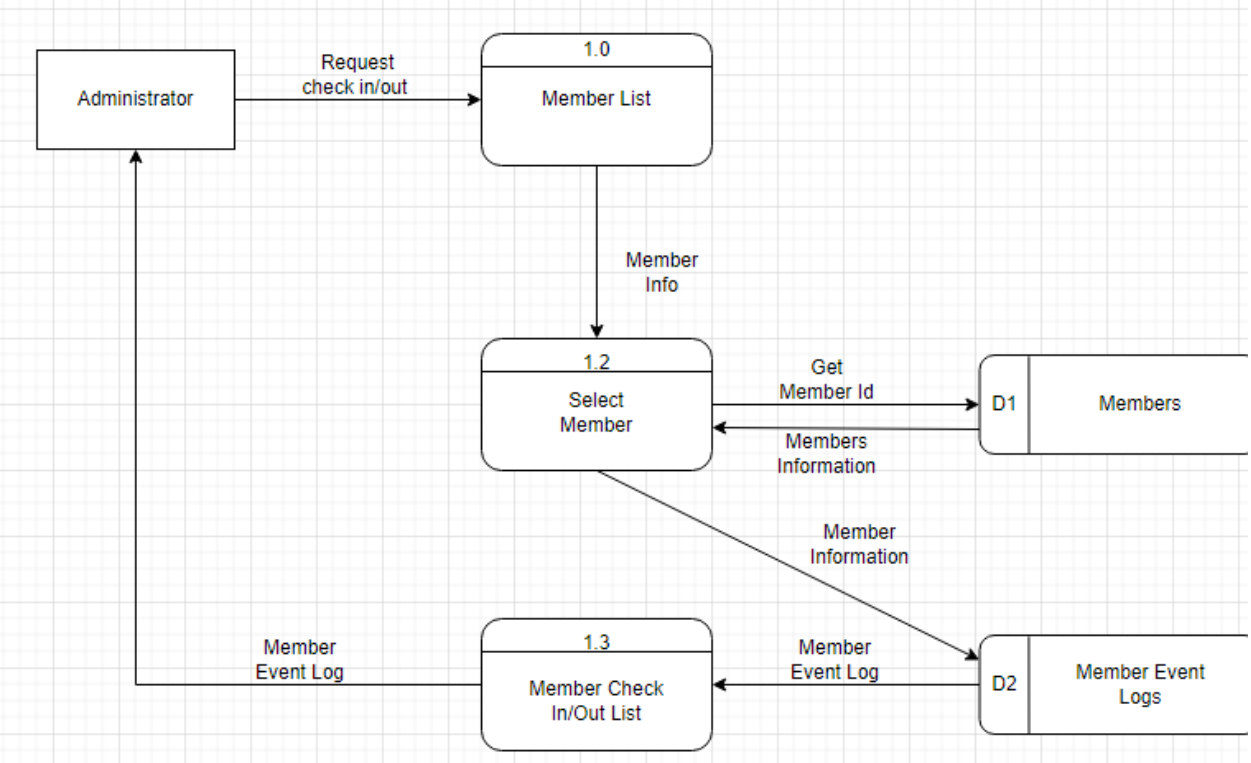


Event Log (Storing member event logs)

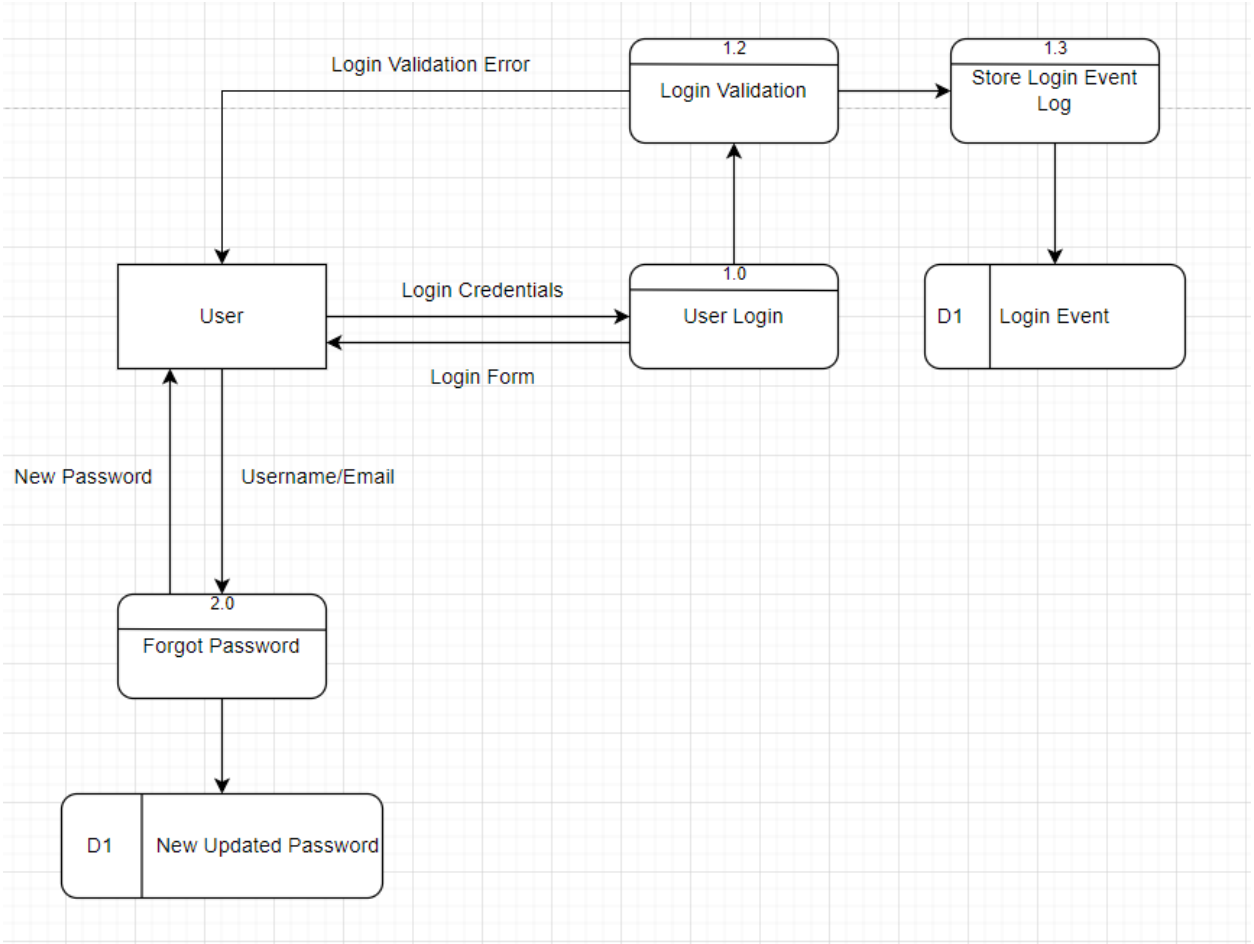


Event Logs (Retrieving Event Log Information)





Login DFD:



3.3 Use Case Scenarios

The following Use Case Scenarios (UCSs) have been identified. The following table is a summary of the UCSs followed by detailed descriptions.

Use Case Scenario #	Description
UC1	User Login (Online Dashboard)
UC2	User Dashboard
UC3	Search for Member Account
UC4	Add New Member
UC5	Update Existing Member Information
UC6	Delete Member
UC7	Review Event Logs
UC8	Filter Event Logs based on Type
UC9	View Unauthorised People Detected
UC10	Run Local Application
UC11	Admin Registration
UC12	Search Events Log

UC13	View member check in/ check out page

3.3.1 UC1 - User Login (Online Dashboard)

Scenario Name:

User Login

Actors:

User

Stakeholders and Interests:

- Admin Users: Want to access their dashboard securely and quickly.
- Members: Want to ensure only authorized users can access their info.

Description:

This use case describes how a user logs into the online dashboard by providing their credentials.

Assumptions, Constraints, and/or Pre-Conditions:

- User has a registered account.
- User is on the login page of the web application.

Trigger – Starting Point:

User opens the login page and attempts to log in.

Relationships:

- UC2: User Dashboard

Normal Flow of Events:

1. User navigates to the login page.
2. User enters their email address and password.
3. User clicks the "Login" button.
4. System validates the credentials.
5. Upon successful validation, user is redirected to their dashboard (UC2).

Sub-Flows:

- S-1: Forgot Password
    1. User clicks on "Forgot Password."
    2. System prompts the user to enter their registered email.
    3. System sends a password reset link to the provided email.
    4. User follows the link and resets their password.

Alternate/Exceptional Flows:

1. User enters incorrect credentials.
    - System displays an error message.
    - User can try logging in again or use the "Forgot Password" option.
2. User account is locked after multiple failed login attempts.
    - System displays an account locked message.
    - User must contact support or use the account recovery options.

3.3.2 UC2 - User Dashboard

Scenario Name:

User Dashboard

Actors:

User

Stakeholders and Interests:

- Admin Users: Want to ensure users can access relevant data and functionalities easily.
- Members: Want to ensure their personal information is being handled properly.

Description:

This use case describes how a user accesses and interacts with their dashboard after logging in.

Assumptions, Constraints, and/or Pre-Conditions:

- User has successfully logged in.

Trigger – Starting Point:

User is redirected to the dashboard after logging in.

Relationships:

- UC1: User Login
- UC4: Add New Member
- UC5: Update Existing Member Information
- UC6: Delete Member
- UC7: Review Event Logs
- UC8: Filter Event Logs based on Type
- UC9: View Unauthorized People Detected

Normal Flow of Events:

1. User is redirected to the dashboard.

2. User navigates through various sections like event logs, and member management and the link to download the local application.
3. User views their profile information and settings.

Sub-Flows:

- S-1: View Personal Information
    1. User clicks on the "Profile" section.
    2. System displays the user's personal information.
- S-2: Manage Settings
    1. User clicks on the "Settings" section.
    2. System displays available user profile settings for the user to modify.
- S-3: Manage Settings
    1. User clicks on the "Settings" section.
    2. System displays available settings for the user to manage.

Alternate/Exceptional Flows:

1. System fails to load the dashboard.
    - System displays an error message.
    - User can try refreshing the page or contact support.

3.3.3 UC3- Search For Member Account

Scenario Name:

Search for Member Account

Actors:

Administrator

Stakeholders and Interests:

- Administrator: Wants to search for member account.

Description:

This use case describes how an administrator searches for a specific member account.

Assumptions, Constraints, and/or Pre-Conditions:

- Administrator logged in and has access to the member management section.

Trigger – Starting Point:

Admin wants to search for a member to edit details.

Relationships:

- UC2: User Dashboard
- UC5: Updating Existing Member Information
- UC6: Delete Member

Normal Flow of Events:

1. User navigates to the member management section.
2. User clicks on search bar.
3. User enters a member to be searched.
4. System validates that search bar field is not empty.
5. System validates that member is found in database.
6. System displays members that match.

Sub-Flows:

- S-1: Input Member Details
    1. User enters member details like name, email, and profile picture.
    2. System validates each field.

Alternate/Exceptional Flows:

1. System is unable to find a member that matches in database.
    - System displays an error message.
    - User re-enters the data.

3.3.4 UC4 - Add New Member

Scenario Name:

Add New Member

Actors:

User

Stakeholders and Interests:

- User: Want to manage member information efficiently.
- Members: Expect their information to be accurately recorded and updated.

Description:

This use case describes how an user adds a new member to the system.

Assumptions, Constraints, and/or Pre-Conditions:

- User is logged in and has access to the member management section.

Trigger – Starting Point:
User wants to add a new member.

Relationships:
- UC2: User Dashboard

Normal Flow of Events:
7. User navigates to the member management section.
8. User clicks on "Add New Member."
9. System displays a form to input member details.
10. User enters member details (name, picture and email) and submits the form.
11. System validates the information and adds the new member to the database.
12. System displays a confirmation message.

Sub-Flows:
- S-1: Input Member Details
    1. User enters member details like name, email, and profile picture.
    2. System validates each field.

Alternate/Exceptional Flows:
2. User enters invalid data.
    - System displays validation errors.
    - User corrects the data and resubmits the form.
3. System encounters an error while adding the member.
    - System displays an error message.
    - User refreshes the page and re-enters the data.

3.3.5 UC5 - Update Existing Member Information

Scenario Name:
Update Existing Member Information

Actors:
Administrator

Stakeholders and Interests:
- User: Want to ensure member information is up-to-date.
- Member: Expect their information to be accurately updated.

Description:
This use case describes how an administrator updates the information of an existing member.

Assumptions, Constraints, and/or Pre-Conditions:
- User is logged in and has access to the member management section.

Trigger – Starting Point:
Administrator wants to update a member's information.

Relationships:
- UC2: User Dashboard

Normal Flow of Events:
1. User navigates to the member management section.
2. User selects a member to update.
3. System displays the member's current information.
4. User updates the necessary fields and submits the form.
5. System validates the new information and updates the member in the database.
6. System displays a confirmation message.

Sub-Flows:
- S-1: Edit Member Details
    1. User updates details like name, email, and profile picture.
    2. System validates each field.

Alternate/Exceptional Flows:
1. User enters invalid data.
    - System displays validation errors.
    - User corrects the data and resubmits the form.
2. System encounters an error while updating the member.
    - System displays an error message.
    - User retries and refreshes the page.

3.3.6 UC6 - Delete Member
Scenario Name:
Delete Member

Actors:
User

Stakeholders and Interests:

- User: Want to manage user membership efficiently.
- Members: Expect their information to be accurately recorded and updated.

Description:

This use case describes how an User deletes a member from the system.

Assumptions, Constraints, and/or Pre-Conditions:

- Administrator is logged in and has access to the member management section.

Trigger – Starting Point:

Administrator wants to delete a member.

Relationships:

- UC2: User Dashboard

Normal Flow of Events:

1. User navigates to the member management section.
2. User selects a member to delete.
3. System prompts for confirmation.
4. User confirms the deletion.
5. System removes the member from the database.
6. System displays a confirmation message.

Sub-Flows:

- S-1: Confirm Deletion
    1. System prompts the User for confirmation.
    2. User confirms the deletion.

Alternate/Exceptional Flows:

1. User cancels the deletion.
    - System aborts the process and retains the member information.

3.3.7 UC7- Review Event Logs

Scenario Name:

Review Event Logs

Actors:

User

Stakeholders and Interests:

- User: Want to review and manage event logs for security purposes.

Description:

This use case describes how users review the event logs captured by the system.

Assumptions, Constraints, and/or Pre-Conditions:

- User is logged in.

Trigger – Starting Point:

User wants to review event logs.

Relationships:

- UC2: User Dashboard

Normal Flow of Events:

1. User/Administrator navigates to the event logs section.
2. System displays a list of event logs.
3. User/Administrator reviews the event logs.

Sub-Flows:

- S-1: Filter Logs
    1. User/Administrator applies filters to the event logs (e.g., date, event type).
    2. System displays filtered logs.

Alternate/Exceptional Flows:

1. System fails to retrieve logs.
    - System displays an error message.
    - User/Administrator retries and refreshes the page.

3.3.8 UC8- Filter Event Logs based on Type

Scenario Name:

Filter Event Logs

Actors:

User

Stakeholders and Interests:

- Users: Want to find specific events easily.

Description:

This use case describes how users filter event logs based on event type.

Assumptions, Constraints, and/or Pre-Conditions:

- User is logged in and has access to event logs.

Trigger – Starting Point:

User wants to filter event logs by type.

Relationships:

- UC7: Review Event Logs

Normal Flow of Events:

1. User navigates to the event logs section.
2. User selects a filter option (e.g., Type: Person, Animal, Fire Hazard, Unauthorised).
3. System displays logs that match the selected filter.

Sub-Flows:

- S-1: Apply Filters
    1. User selects multiple filter criteria (e.g., date range, member name).
    2. System displays logs that match all the selected criteria.

Alternate/Exceptional Flows:

1. No logs match the filter criteria.
    - System displays a "No logs found" message.
    - User adjusts the filter criteria.

3.3.9 UC9 - View Unauthorized People Detected

Scenario Name:

View Unauthorized People Detected

Actors:

User

Stakeholders and Interests:

- Users: Want to ensure immediate action upon detecting unauthorized People.
- Members: Want to ensure that they are secure.

Description:

This use case describes how users/administrators view logs of unauthorized people detected by the system.

Assumptions, Constraints, and/or Pre-Conditions:

- User is logged in.

Trigger – Starting Point:
User wants to view logs of unauthorized people detected.

Relationships:
- UC2: User Dashboard
- UC7: Review Event Logs

Normal Flow of Events:
1. User navigates to the "Unauthorized People Detected" section.
2. System displays a list of logs showing unauthorized people detected.
3. User reviews the logs.

Sub-Flows:
- S-1: View Details
    1. User clicks on a specific log entry.
    2. System displays detailed information about the incident (e.g., time, location, image).

Alternate/Exceptional Flows:
1. System fails to retrieve logs.
    - System displays an error message.
    - User refreshes the page.

3.3.10 UC10 - Running Local Application

Scenario Name:
Running Local Application

Actors:
User

Stakeholders and Interests:
- Users: Want to run the local application smoothly and efficiently.
- Developers: Want to ensure the local application runs correctly without issues.

Description:
This use case describes how a user runs the local application to manage their data and interact with the system.

Assumptions, Constraints, and/or Pre-Conditions:
- User has the local application installed on their machine.
- User has valid login credentials if authentication is required by the local application.
- The local application is properly configured and connected to the necessary services or databases.

Trigger – Starting Point:

User wants to launch and use the local application.

Relationships:

- UC1: User Login (the local application requires authentication)
- UC2: User Dashboard
- UC3: Search for Member Account
- UC4: Add New Member
- UC5: Update Existing Member Information
- UC6: Delete Member
- UC7: Review Event Logs
- UC8: Filter Event Logs based on Type
- UC9: View Unauthorized People Detected

Normal Flow of Events:

1. User launches the local application.
2. If authentication is required, the user is prompted to log in (refer to UC1).
3. After successful login (if required), the user is presented with the main camera feed of the local application.
4. The local application displays a live camera feed with bounding boxes over detected people/animal/events along with current event logs.

Sub-Flows:

- S-1: Application Download
    1. User is logs into the dashboard website.
    2. User navigates to the “Download Local Application” and clicks the download button.

Alternate/Exceptional Flows:

1. Application fails to launch.
    - System displays an error message.
    - User can troubleshoot.
2. User enters incorrect credentials (if authentication is required).
    - System displays an error message.
    - User can retry logging in.
3. Application encounters an error while performing an action.

- System displays an error message.
- User can retry the action or contact support.

Post-Conditions:

- User successfully uses the local application to view the live camera feed and the events detected by the ML models.
- Any changes made by the user (e.g., adding a new member) in the online dashboard are synced with the local dashboard.

3.3.11 UC11 – User Registration

Scenario Name:

User Registration

Actors:

User

Stakeholders and Interests:

- User: Want to create an user account to manager members and review event logs.

Description:

This use case describes how user setups up a user account to have access to the admin dashboard

Assumptions, Constraints, and/or Pre-Conditions:

- User has their user information

Trigger – Starting Point:

User wants to create an administration account.

Relationships:

- UC2: User Dashboard

Normal Flow of Events:

4. User navigates to the "Registration" section.
5. System displays a registration form.
6. User submits and system validates.

Sub-Flows:

- S-1: View Details
    1. User clicks on a submit button.
    2. System validates information before storing user info in database.

Alternate/Exceptional Flows:

2. System fails to retrieve logs.

- System displays an error message.
- User doesn't leave information blank

3.3.12 UC12 – Search Events Log

Scenario Name:

Search Events Log

Actors:

User

Stakeholders and Interests:

- User: Want to efficiently search to identify specific events.

Description:

This use case describes how users and administrators search for specific event logs through the search bar.

Assumptions, Constraints, and/or Pre-Conditions:

- User is logged in.
- User has access to events log sections.

Trigger – Starting Point:

User wants to view logs of unauthorized people detected.

Relationships:

- UC7: Review Event Logs

Normal Flow of Events:

1. User navigates to the events log section.
2. User selects filter or search option.

Sub-Flows:

- S-1: Apply Filters
    1. User selects filters based on their search criteria.
    2. System display logs that matches selected criteria.

Alternate/Exceptional Flows:

1. System display an error message.
2. User retries and refreshes page.

3.3.13 UC13- View member check-in/check-out

Scenario Name:

View member check-in/check-out

Actors:

User

Stakeholders and Interests:

- User: Wants to view member check-in and check-out information.

Description:

This use case describes how a user will view check-in and check-out information for a specific member.

Assumptions, Constraints, and/or Pre-Conditions:

- User logged in and has access to the member management section.

Trigger – Starting Point:

User wants to view a specific member check-in and check-out details.

Relationships:

- UC2: User Dashboard
- UC3: Search for Member Account

Normal Flow of Events:

1. User navigates to the member management section.
2. User clicks on search bar.
3. User enters a member to be searched.
4. System validates that search bar field is not empty.
5. System validates that member is found in database.
6. User clicks the member of interest.
7. System displays check-in/check-out page.

Sub-Flows:

- S-1: Input Member Details
    1. User enters member details like name, email, and profile picture.
    2. System validates each field.

Alternate/Exceptional Flows:

1. System is unable to find a member that matches in database.
    a. System displays an error message.

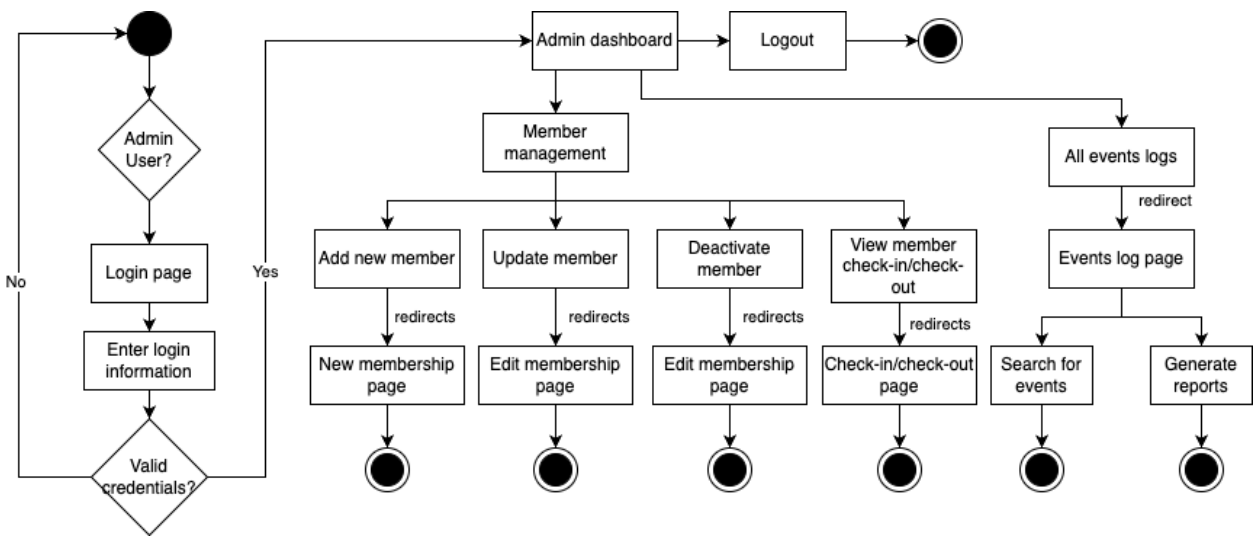
b. User re-enters the data.

3.4 Activity Diagrams

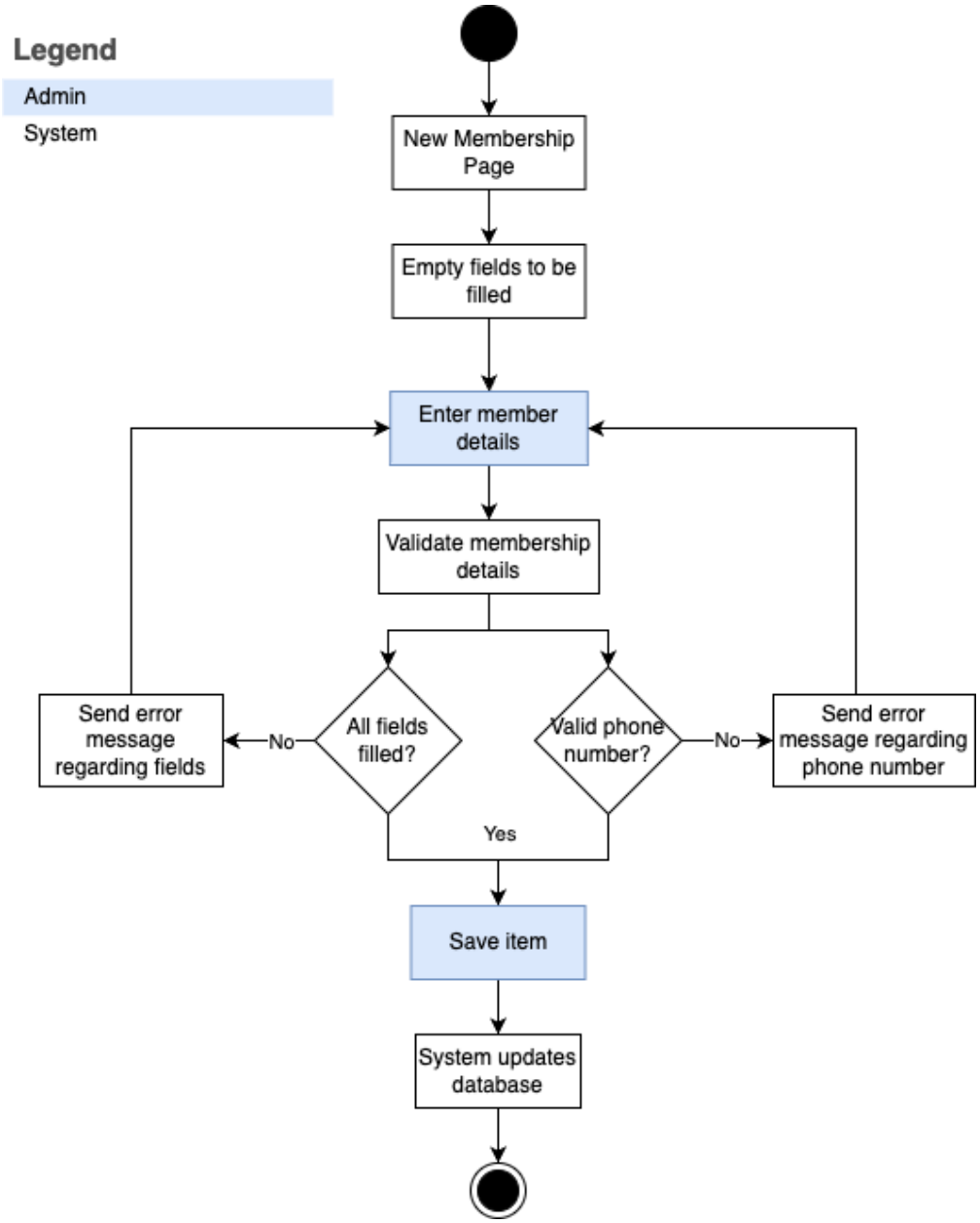
The following is a summary table of the Activity Diagrams provided followed sub-sections of the actual diagrams.

Activity Diagram #	Description	Related UCS #
AD1	Admin Dashboard	UC2
AD2	Admin Adding New Member	UC4
AD3	Admin Edit Member & Delete Member	UC5, UC6
AD4	Admin Search Users	UC3
AD5	Search Event Logs	UC7
AD6	Admin Registration	UC11
AD7	Running Local Application	UC 10

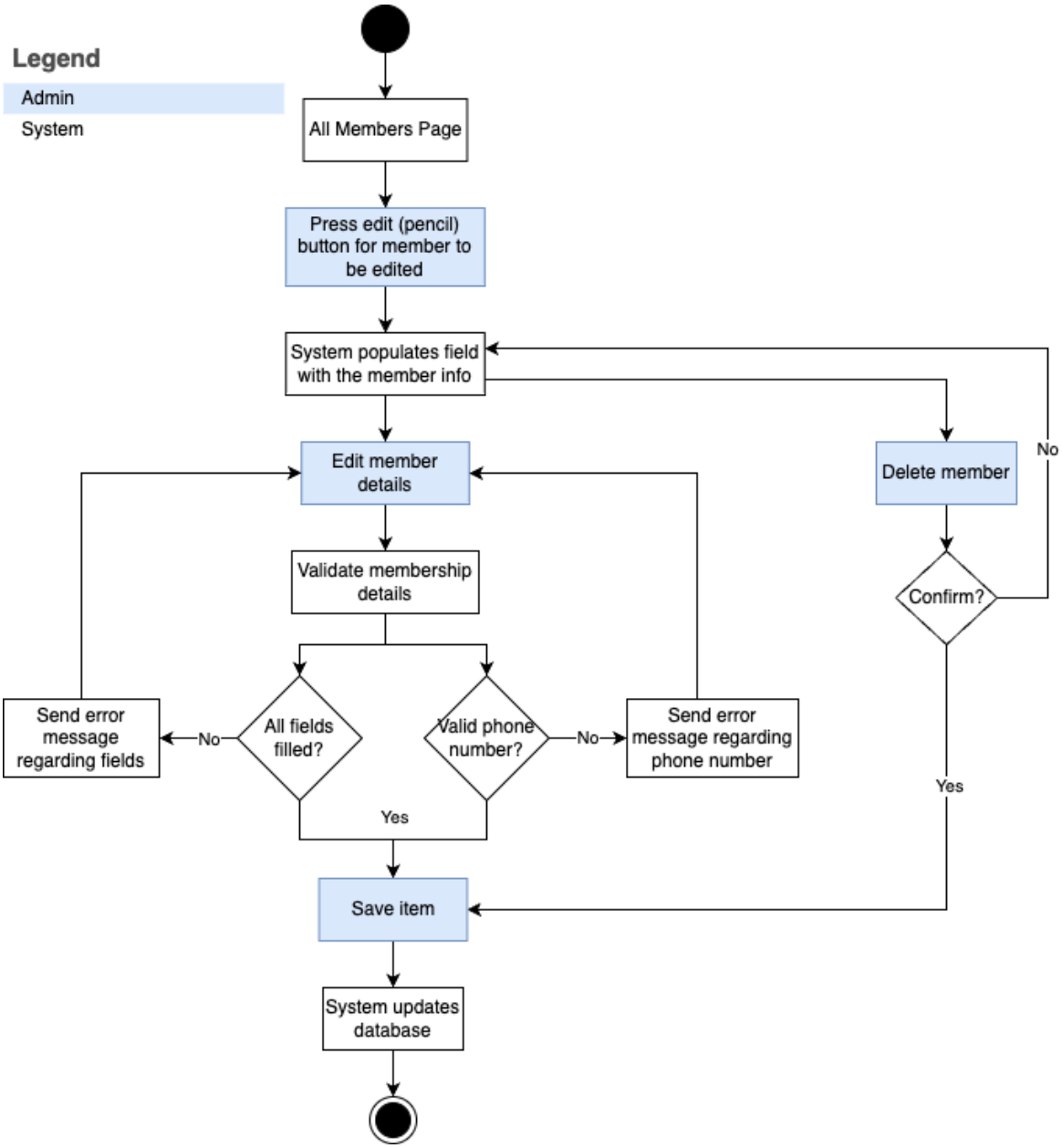
3.4.1 AD1 - Admin Dashboard



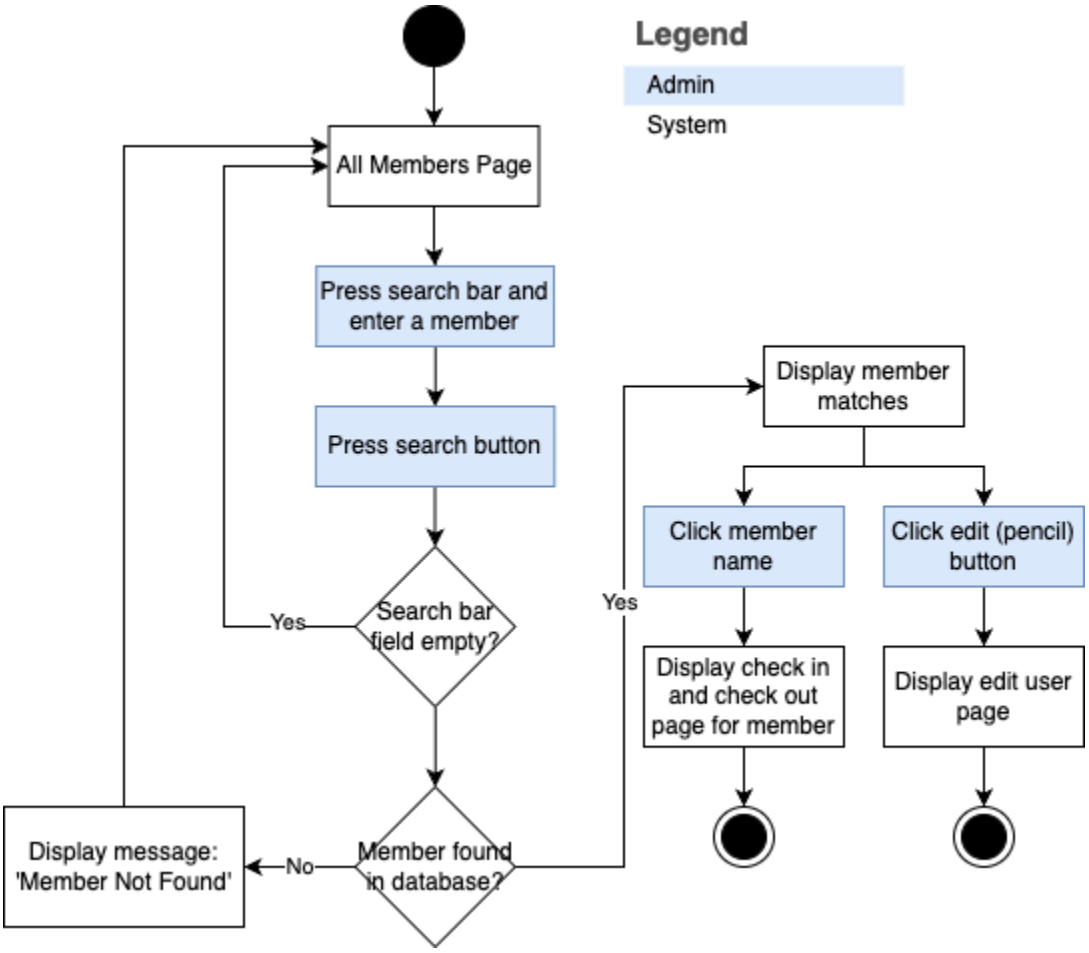
3.4.2 AD2 - Admin Adding New Member



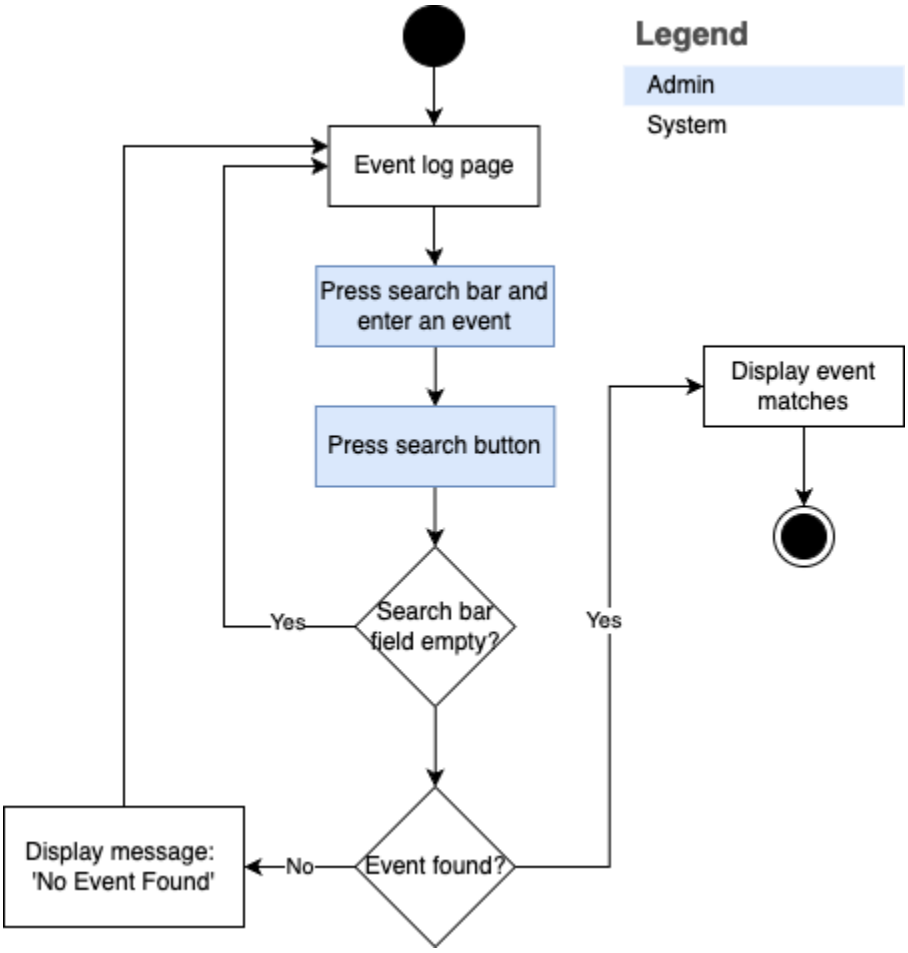
3.4.3 AD3 - Admin Edit Members & Delete Members



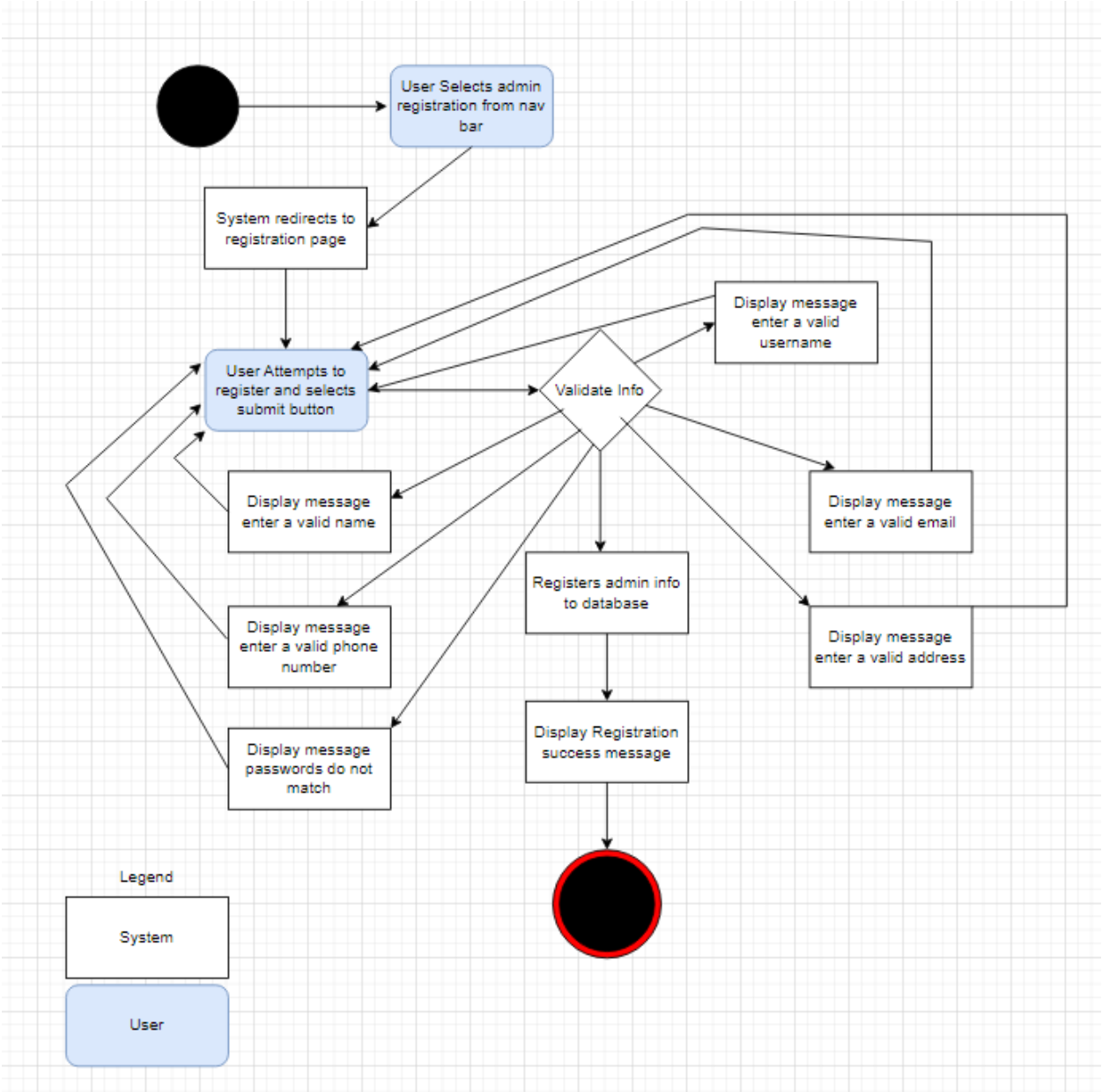
3.4.4 AD4 - Admin Search Members



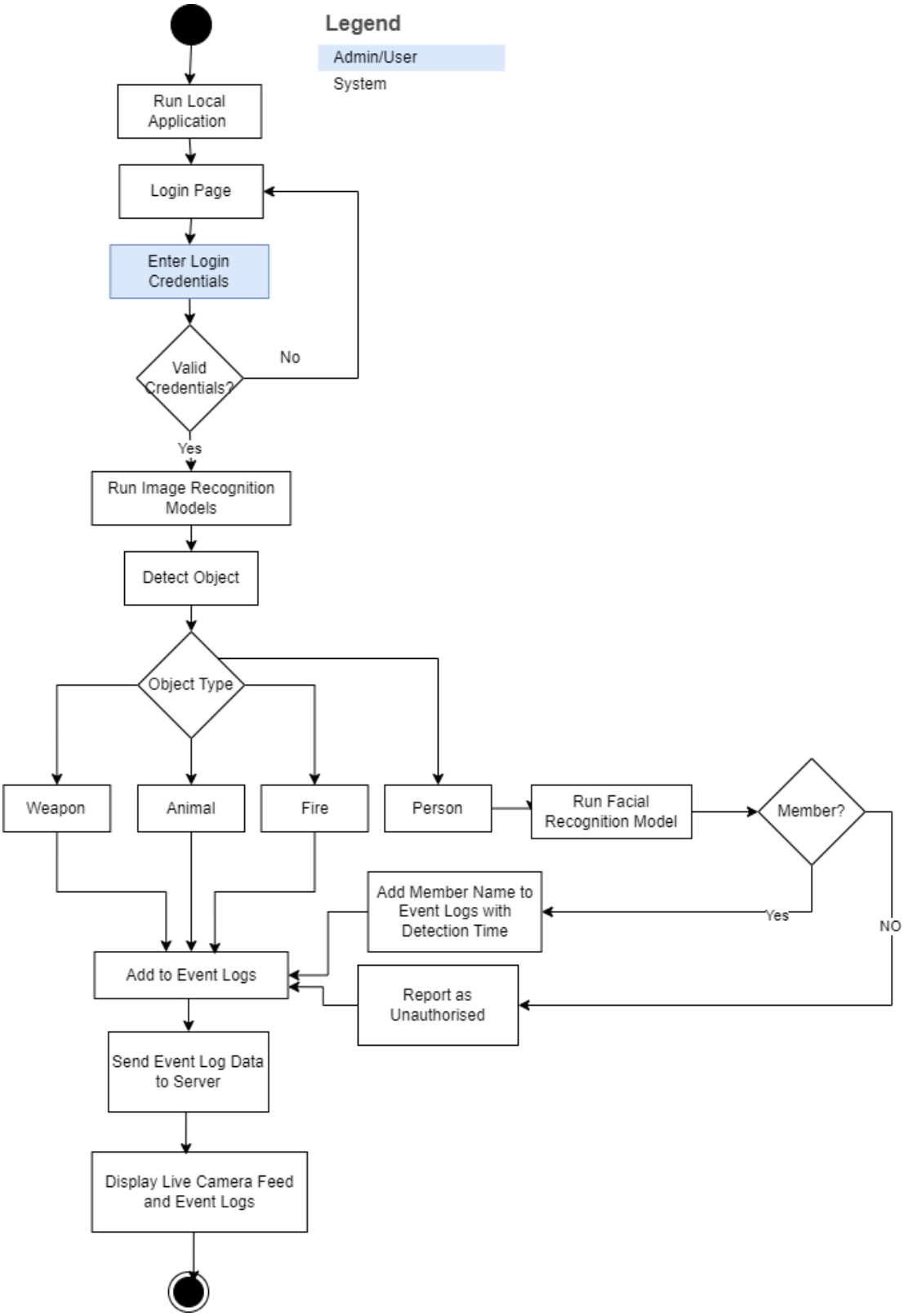
3.4.5 AD5 - Search Event Logs



3.4.6 AD6 – Admin Registration



3.4.7 AD7 – Running Local Application

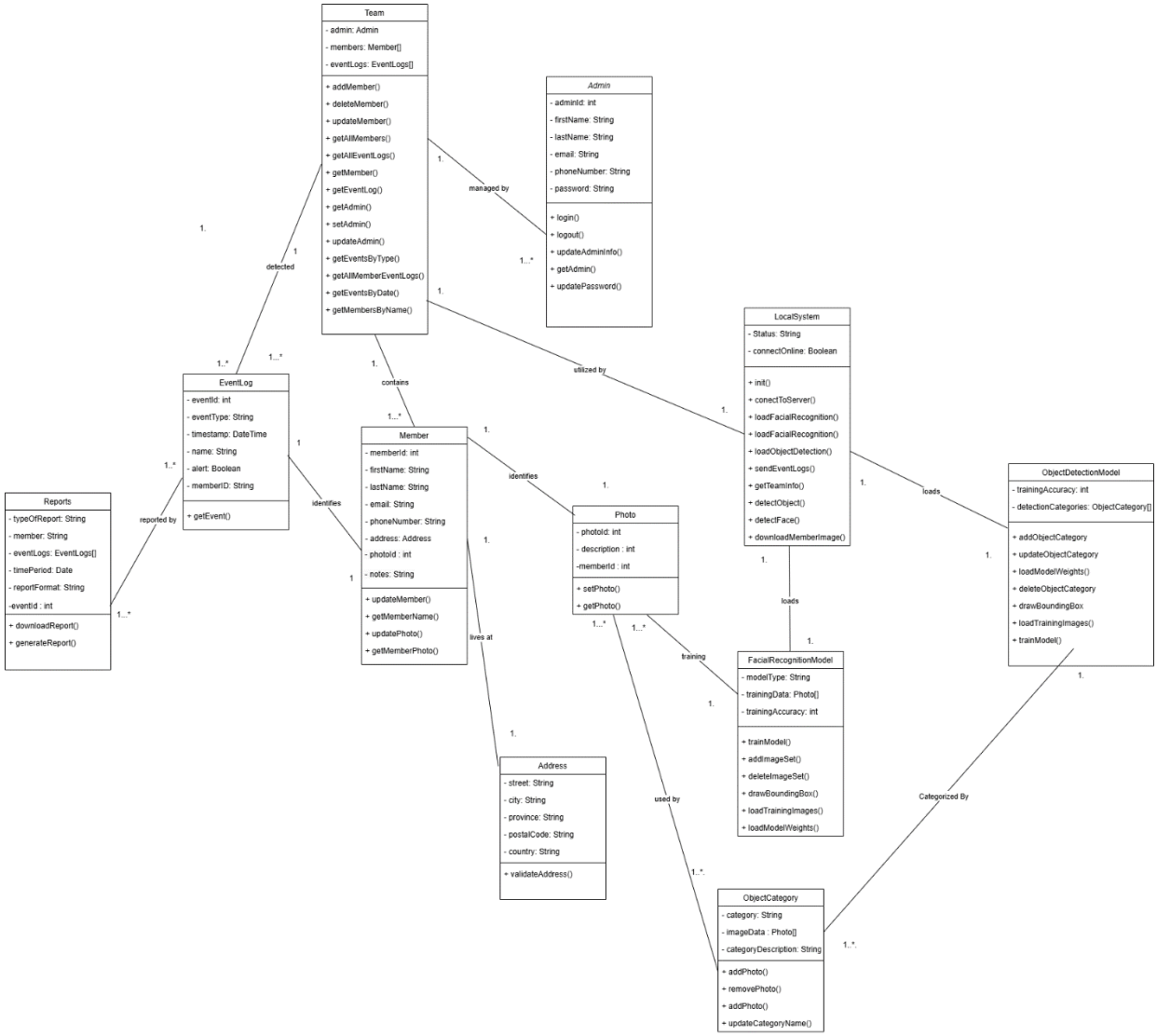


3.5 Business Rules

The following is a list of Business Rules that must be met through the design of the SecureGuard application. Each rule is described below and associated with the corresponding Activity Diagrams, Use Case Scenarios, and User-Interface Mock-up.

Business Rule #	Description	Activity Diagram	Related UCS	UI Mockup
BR01	User must be registered in the system with a valid name (not containing special symbols)	AD6	UC11	UI2.7.7
BR02	User will receive notification if check in or check out was successful or not	AD07	UC09, UC10	UI2.7.1
BR03	User must check in and check out by having their face scanned by the camera	AD07	UC09, UC10	UI2.7.10
BR04	Only registered users can check in and check out	AD02	UC04	UI2.7.6, UI2.7.4
BR05	Users checked in and out reports will be logged by the system and checked by the admin	AD05	UC07	UI2.7.3
BR06	User must edit user profile within 24 hours if notified by member of info change	AD03	UC05, UC06	UI2.7.6
BR07	Facial recognition on non-users will not be allowed to check in or out	AD07	UC10	UI2.7.10
BR08	Only registered users should be retrievable during search operations for the purpose of editing or viewing user details.	AD4	UC3, UC5, UC13	UI2.7.3
BR09	Only Users with valid credentials can run the local application.	AD07	UC10	UI2.7.9
BR10	Users can only access their own event logs and member information. Each user's information must be secure and inaccessible to the other users.	AD1	UC2, UC3, UC7	UI2.7.1, UI2.7.2
BR11	Event Logs must be able to be filtered based on conditions for easy retrieval of data.	AD05	UC12	UI2.7.1

Section 4 – Domain Class

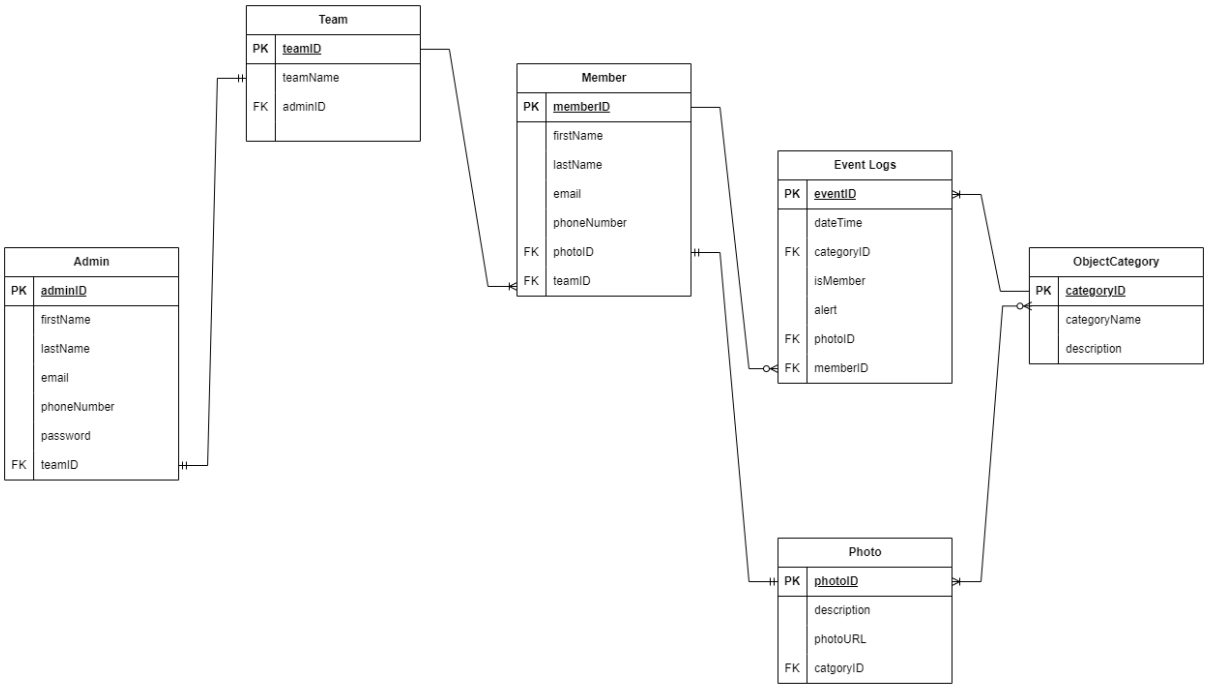


Section 5 – Database

Selected Database

The project will use a SQL relational database Supabase (Free Cloud Postgres service with additional features) for data storage.

5.1 ERD



5.2 Data Dictionary

SecureGuard Database

Table Name	Field Name	Field Label	Description	Field Size	Data Type/Format	Data Codes
Admin Table	adminID	Admin ID	Unique identifier for admin	10	integer	N/A
Admin Table	firstName	First Name	First name of admin	20	string	N/A
Admin Table	lastName	Last Name	Last name of admin	20	string	N/A
Admin Table	email	Email	Email of admin	30	string	N/A
Admin Table	phoneNumber	Phone Number	Street address of admin	20	string	N/A

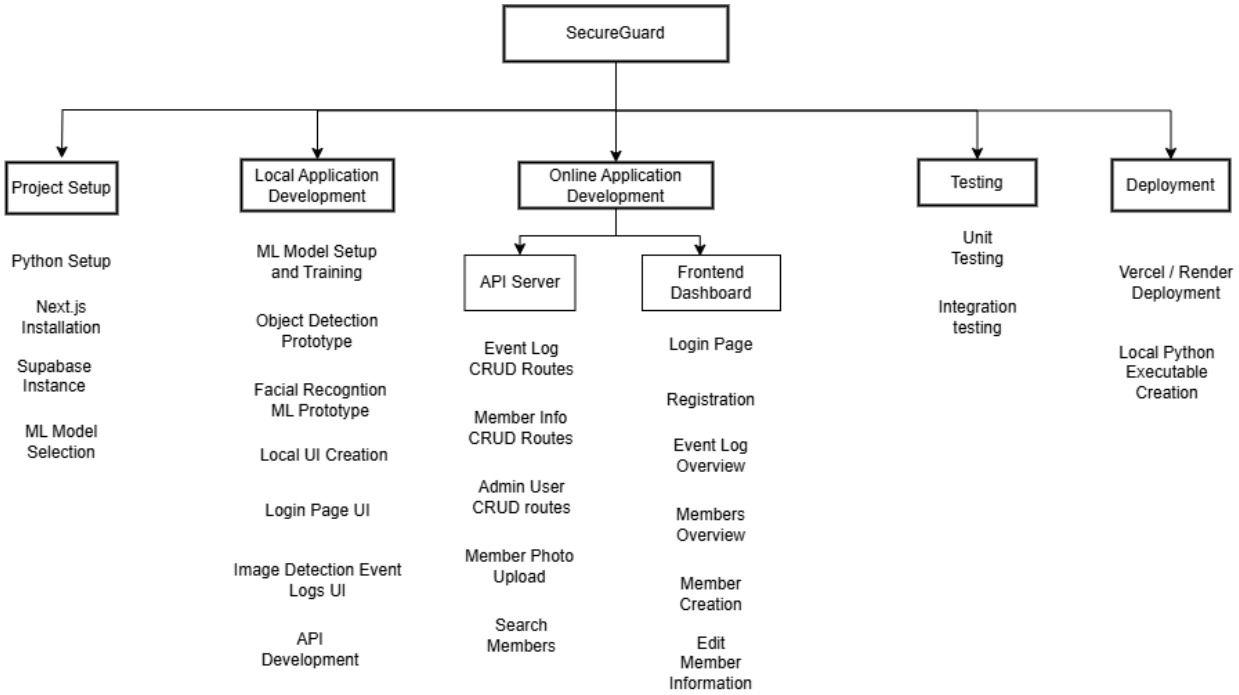
Admin Table	password	Password	Password for admin	50	string	N/A
Admin Table	teamID	Team ID	Foreign key to team table	10	integer	N/A
Team Table	teamID	Team ID	Unique identifier for team	10	integer	N/A
Team Table	teamName	Team Name	Name of team	30	string	N/A
Team Table	adminID	Admin ID	Foreign key to Admin Table	10	integer	N/A
Member Table	memberID	Member Identifier	Unique id for member	10	integer	N/A
Member Table	firstName	First Name	First name of member	20	string	N/A
Member Table	lastName	Last Name	Last name of member	20	string	N/A
Member Table	email	Email	Email of member	30	string	N/A
Member Table	phoneNumber	Phone Number	Member phone number	10	integer	N/A
Member Table	photoID	Photo ID	Foreign key to Photo Table	8	integer	N/A
Member Table	teamID	Team ID	Foreign key to Team Table	10	integer	N/A
Event Table	eventID	Event Log Identifier	Event log unique identifier	10	integer	N/A
Event Table	dateTime	Date and Time	Date and time of event	20	date	N/A

Event Table	categoryID	Category ID	Foregin key to ObjectCategory table	10	integer	
Event Table	isMember	Is Member	Boolean to see if recognized member	1	boolean	N/A
Event Table	alert	Alert	Alert for an event	20	string	N/A
Event Table	photoID	Photo ID	Foreign key to Photo table	10	integer	N/A
Event Table	memberID	Member ID	Foreign key to Member Table	`10	integer	N/A
Photo Table	photoID	Photo ID	Unique identifier of photo	10	integer	N/A
Photo Table	description	Description	Description of photo	50	string	N/A
Photo Table	photoURL	URL	URL of photo	50	string	N/A
Photo Table	categoryID	Category ID	Foreign key to ObjectCategory table	10	integer	N/A
Object Category Table	categoryID	Category ID	Object Category unique identifier	10	integer	N/A

Object Category Table	categoryName	Category Name	Name of the Object Category	20	string	N/A
Object Category Table	description	Description	Descripti on of Object Category	50	string	N/A

Section 6 – Project Management

6.1 Work Breakdown Structure



6.2 Milestones

Milestone 1 : Project Setup

Due Date: September 24th 2024

Project Installations

- Python Installation

- Next.js Installation

Database Setup

- Configure Supabase

Local Application Development

- Train and configure Machine Learning models
- Develop the frontend User Interface
- Implement backend functionality

Milestone 2: Online Application Development

Due Date: October 29th 2024

Backend Development

- Implement API Routes

Frontend Development

- Create the frontend dashboard

Milestone 3: Testing

Due Date: November 11th 2024

Testing Phases

- Local Application Testing
- Online Dashboard Testing
- Integration Testing

Quality Assurance

- Identify and fix bugs
- Perform updates as necessary

Milestone 4: Deployment

Due Date: December 2nd 2024

Deployment Tasks

- Deploy to Vercel
- Python Executable for Local Application

6.3 Acceptance Criteria

Milestone 1

- Python installed on the development environment.
- Next.js setup on the development environment.
- Supabase database is set up with the required tables and authentication.
- Local application runs without bugs.

Milestone 2

- All backend API routes are setup for authentication, event logs, user management and membership management.
- Frontend has pages for event log overview, user login page, new user registration, membership information, member update page, and new member setup, and application download.
- Frontend page is intuitive and functional for end users.

Milestone 3

- All functionalities have passed comprehensive testing including unit, integration and end-to-end tests.
- Bugs have been identified and resolved.

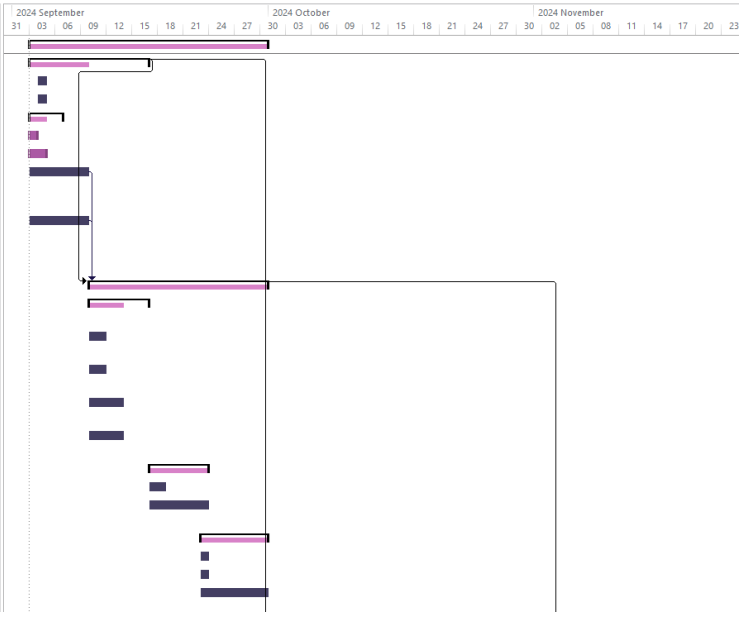
Milestone 4

- The web application is successfully deployed to Vercel.
- The Python application is packaged as a local executable.

6.4 Implementation Schedule

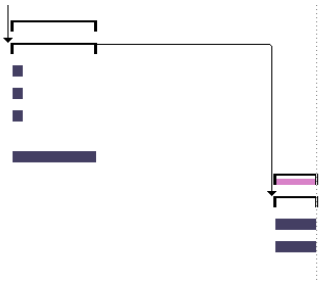
SecureGuard Project 1.mpp

	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Milestone 1**	**20 days**	**Tue 24-09-03**	**Mon 24-09-30**		
2	**Project Setup**	**10 days**	**Tue 24-09-03**	**Mon 24-09-16**		
3	Python Install and Setup	1 day	Wed 24-09-04	Wed 24-09-04		
4	Next.js Installation	1 day	Wed 24-09-04	Wed 24-09-04		
5	**Supabase Setup**	**4 days**	**Tue 24-09-03**	**Fri 24-09-06**		
6	Database Creation	1 day	Tue 24-09-03	Tue 24-09-03		
7	Authentication Setup	2 days	Tue 24-09-03	Wed 24-09-04		
8	Object Detection Model Setup (MobileNetSSD/YoloV4)	5 days	Tue 24-09-03	Mon 24-09-09		
9	Facial Recognition ML Model Selection and Setup (Sface/Face_Recognition/Dee Face)	5 days	Tue 24-09-03	Mon 24-09-09		
10	**Local Application Development**	**15 days**	**Tue 24-09-10**	**Mon 24-09-30**	**2,8,9**	
11	**ML Model Training and Configuration**	**5 days**	**Tue 24-09-10**	**Mon 24-09-16**		
12	Google Colab Retraining (Fire and Weapon	2 days	Tue 24-09-10	Wed 24-09-11		
13	Object Detector Class Development	2 days	Tue 24-09-10	Wed 24-09-11		
14	Object Detection Bounding Boxes	4 days	Tue 24-09-10	Fri 24-09-13		
15	Facial Recognition Class Development	4 days	Tue 24-09-10	Fri 24-09-13		
16	**Frontend UI**	**5 days**	**Tue 24-09-17**	**Mon 24-09-23**		
17	Login Page	2 days	Tue 24-09-17	Wed 24-09-18		
18	Camera Feed and Live Event Logs UI	5 days	Tue 24-09-17	Mon 24-09-23		
19	**Backend Development**	**6 days**	**Mon 24-09-23**	**Mon 24-09-30**		
20	Camera Feed Initialisation	1 day	Mon 24-09-23	Mon 24-09-23		
21	User Credentials Validatio	1 day	Mon 24-09-23	Mon 24-09-23		
22	Object Detection Model Initialisation	6 days	Mon 24-09-23	Mon 24-09-30		



	Task Name	Duration	Start	Finish	Predecessors	Resource Names
22	Object Detection Model Initialisation	6 days	Mon 24-09-23	Mon 24-09-30		
23	Facial Recognition Model Initialisation	6 days	Mon 24-09-23	Mon 24-09-30		
24	**Milestone 2**	**25 days?**	**Mon 24-09-30**	**Fri 24-11-01**		
25	**Online Application Developmer**	**24 days?**	**Tue 24-10-01**	**Fri 24-11-01**	**2**	
26	**Backend API routes**	**14 days?**	**Tue 24-10-01**	**Fri 24-10-18**		
27	GET All Members (/api/me	2 days	Tue 24-10-01	Wed 24-10-02		
28	GET Member by ID (/api/members/:id)	1 day	Thu 24-10-03	Thu 24-10-03	27	
29	POST New Member	1 day	Wed 24-10-02	Wed 24-10-02	30	
30	DELETE Member	1 day	Tue 24-10-01	Tue 24-10-01		
31	GET Member Photo (/api/member/:id/photo)	1 day	Fri 24-10-04	Fri 24-10-04	28	
32	PUT Update Member Information	1 day	Tue 24-10-01	Tue 24-10-01		
33	GET All Event Logs (/api/eventlogs)	1 day	Mon 24-10-07	Mon 24-10-07	31	
34	GET EventLog By Type (/api/eventlogs?type="")	1 day	Tue 24-10-01	Tue 24-10-01		
35	POST Event Log (/api/even	1 day	Thu 24-10-03	Thu 24-10-03	29	
36	POST Register New User	1 day	Fri 24-10-04	Fri 24-10-04	35	
37	PUT Update User Informati	1 day	Mon 24-10-07	Mon 24-10-07	36	
38	GET Authenticate user	1 day	Tue 24-10-08	Tue 24-10-08	37	
39	**Frontend Dashboard**	**10 days?**	**Fri 24-10-18**	**Thu 24-10-31**		
40	Event Log Overview	2 days	Fri 24-10-18	Mon 24-10-21		
41	Register New User	1 day	Tue 24-10-22	Tue 24-10-22	40	
42	User Information Page	1 day	Wed 24-10-23	Wed 24-10-23	41	
43	Add New Member	1 day	Thu 24-10-24	Thu 24-10-24	42	
44	Update Member	1 day	Fri 24-10-25	Fri 24-10-25	43	
45	Login Page	1 day	Mon 24-10-28	Mon 24-10-28	44	
46	Download Local Application Page	1 day	Tue 24-10-29	Tue 24-10-29	45	

	Task Name	Duration	Start	Finish	Predecessors	Resource Names
47	**Milestone 3**	**6 days**	**Mon 24-11-04**	**Mon 24-11-11**		
48	**Testing**	**6 days**	**Mon 24-11-04**	**Mon 24-11-11**	**10,25**	
49	Local Application Testing	1 day	Mon 24-11-04	Mon 24-11-04		
50	Online Dashboard Testing	1 day	Mon 24-11-04	Mon 24-11-04		
51	Integration Testing (Local-Online)	1 day	Mon 24-11-04	Mon 24-11-04		
52	Bug Fixes and Updates	6 days	Mon 24-11-04	Mon 24-11-11		
53	**Milestone 4**	**2 days?**	**Fri 24-11-29**	**Mon 24-12-02**		
54	**Deployment**	**2 days**	**Fri 24-11-29**	**Mon 24-12-02**	**48**	
55	Deploy to Vercel	2 days	Fri 24-11-29	Mon 24-12-02		
56	Create Python Executable for Local Application	2 days	Fri 24-11-29	Mon 24-12-02		



Section 7 – Client/Faculty Sign-off