## Requirement Engineering

- Mobile Device Managements
  - Access Control Managements
    - controlling access with assigning roles
    - different access dashboard for different roles
  - Logs Managements
    - need to control and analyze logs
  - Monitoring and Control
    - informing when system get into trouble
    - dashboard for monitoring and control data and logs and tasks
  - Update Managements
    - need to update app with sending notification and data

### Q&A

- TODO

## Technologies

- Server Application and API Application
  - Django
- Authentication
  - JWT
- Data Admin Dashboard
  - Django Admin Panel (customize with templates if required)
- Access Contorl Dashboard
  - Django Admin Panel (customize with permissions if required)
- Django Cache
  - Reddis
- SQL Database
  - PostgreSQL
- Build Management
  - Docker and Docker Compose (not install everything on your system)
- Log Management
  - ELK (Elasticsearch Stack)
- Log Entry
  - Kafka to Logstash
- Log Dashboard
  - Kibana
- Push Notification
  - fcm (firebase cloud messaging)
  - because it is not successful yet we use real-time communication
- Real-Time Communication
  - Websocket (Django Channels)

- Task Scheduling (for doing something in specific time like managing notifs)
  - Celery
- Task Dashboard
  - Flower

## Server Application and API Application

- Possible choices (most popular and having experience)
  - Node.js
  - GoLang
  - **Django**

### Node.js

- **Advantages:**
  - **Scalability:** Node.js is event-driven and non-blocking, which allows it to handle a large number of concurrent connections, making it highly scalable.
  - **Good and Enormous Libraries and Tools:** Node.js has a rich ecosystem with npm, providing a vast number of libraries and tools that can accelerate development.
  - **Fast for Web Tasks:** Node.js is lightweight and efficient, particularly for I/O-bound tasks, making it ideal for real-time applications.
- **Disadvantages:**
  - **Performance:** While Node.js is efficient for I/O tasks, it may not be as performant for CPU-intensive tasks due to its single-threaded nature.
  - **Time Implementation Overhead in Many Use Cases:** Some tasks that are straightforward in other languages can require more complex and time-consuming implementations in Node.js.
  - **Strange Coding Tricks Required:** To manage asynchronous code effectively, developers may need to employ patterns or techniques (like callbacks, promises, and async/await) that can sometimes complicate code readability and maintenance.

### GoLang

- **Advantages:**
  - **Scalability:** Go is designed with concurrency in mind, thanks to goroutines, making it well-suited for scalable applications.
  - **Performance:** Go is a compiled language, offering near C-level performance, which is ideal for high-performance applications.
  - **Simplicity:** Go's syntax and language constructs are simple and easy to learn, reducing the cognitive load on developers.
  - **Flexibility:** Go's standard library is powerful, and its native support for concurrent programming makes it versatile for various types of

applications.
  - **Concurrency:** Go's goroutines allow for concurrent tasks to be executed efficiently, which is a strong point for building scalable systems.
- **Disadvantages:**
  - **Lack of Libraries:** As a relatively new language, Go doesn't have as extensive a library ecosystem as Node.js or Python, which can lead to longer development times when specific tools or libraries are needed.
  - **Time Implementation Overhead in Many Use Cases:** While Go is efficient, certain advanced tasks may require more time to implement due to less mature libraries and frameworks compared to more established languages.

### Django

- **Advantages:**
  - **Very Easy and Rapid Development:** Django's "batteries-included" philosophy, with a wide array of built-in features, allows for rapid development.
  - **Ready to Use Apps and Panels:** Django comes with a built-in admin panel, authentication system, and other pre-configured apps that can significantly reduce development time.
  - **High-Level Abstractions:** Django's ORM and other high-level abstractions allow developers to write less boilerplate code, focusing more on the application logic.
- **Disadvantages:**
  - **Average Performance:** Django is built on Python, which, while powerful, is slower compared to languages like Go. This can impact performance in high-load scenarios.
  - **Average Scalability:** Django applications may require more careful optimization and potentially more resources to scale compared to solutions built in more performance-oriented languages.

## Log Management

- Possible choices
  - Datadog
  - Sentry
  - **ELK (Elasticsearch stack)**

### Datadog

- **Advantages:**
  - **Comprehensive Monitoring:** Datadog provides not just log management, but also metrics, traces, and real-time observability.

    - **Integration:** It integrates well with a wide range of services and technologies, making it versatile.
    - **Scalability:** Datadog is built to handle logs at scale, with real-time querying and alerting capabilities.
- **Disadvantages:**
    - **Cost:** Datadog can become expensive as the volume of logs and metrics increases.
    - **Complexity:** The depth of features can introduce a learning curve and may require more configuration effort.

**Sentry**

- **Advantages:**
    - **Error Tracking:** Sentry specializes in error tracking and can automatically capture exceptions from applications, providing detailed reports and context.
    - **Integrations:** It integrates easily with various development frameworks and languages.
    - **User Feedback:** Sentry also allows for user feedback to be captured directly alongside error logs, providing additional context for developers.
- **Disadvantages:**
    - **Focused Scope:** Sentry is more focused on error tracking and does not provide full log management or monitoring solutions like Datadog or ELK.
    - **Costs:** Like Datadog, Sentry's costs can increase with higher usage, particularly in high-volume applications.

**ELK (Elasticsearch) Stack**

- **Advantages:**
    - **Customizability:** ELK provides a highly customizable stack for log management, where Elasticsearch handles storage, Logstash or Beats handle data collection, and Kibana offers powerful visualization tools.
    - **Scalability:** Elasticsearch is designed to handle large volumes of log data, making it suitable for large-scale deployments.
    - **Cost-Effective:** Open-source versions of the ELK stack are free to use, offering a cost-effective solution for log management.
- **Disadvantages:**
    - **Operational Overhead:** Managing and scaling an ELK stack can be complex and resource-intensive, requiring expertise in deployment and maintenance.
    - **Performance:** High volumes of logs can require significant tuning to maintain Elasticsearch performance.

## Task Scheduling

- Possible choices
  - Cronjob
  - **Celery**

### Cronjob

- **Advantages:**
  - **Simplicity:** Cronjobs are straightforward and easy to configure for scheduling recurring tasks on Unix-based systems.
  - **Reliability:** Cron has been a part of Unix/Linux systems for decades and is extremely reliable for simple periodic tasks.
  - **Low Resource Consumption:** Cronjobs have minimal overhead, making them ideal for lightweight, periodic tasks.
- **Disadvantages:**
  - **Limited Functionality:** Cron is limited to time-based scheduling and does not handle task queuing or more complex workflows.
  - **Lack of Distributed Task Management:** Cronjobs are tied to a single machine, making them unsuitable for distributed task execution.
  - **Error Handling:** Cronjobs lack built-in error handling and retry mechanisms, requiring additional scripting to handle failures.

### Celery

- **Advantages:**
  - **Distributed Task Queue:** Celery allows for the execution of tasks across distributed systems, which is essential for scaling applications.
  - **Flexible Scheduling:** In addition to periodic tasks, Celery supports complex scheduling scenarios and real-time task execution.
  - **Retry Mechanisms:** Celery has built-in support for task retries in case of failures, improving the reliability of task execution.
  - **Integration:** Celery integrates well with Django and other Python web frameworks, providing seamless task management.
- **Disadvantages:**
  - **Complexity:** Celery is more complex to set up and maintain compared to Cron, requiring additional infrastructure like message brokers (e.g., RabbitMQ or Redis).
  - **Resource Intensive:** Running Celery with a message broker can consume more resources, making it overkill for simple tasks.
  - **Debugging:** Troubleshooting issues in Celery, especially in distributed environments, can be challenging.

## Real-Time Communication

- Possible choices
  - WebRTC

    - MQTT
    - **WebSocket**

**WebRTC**

- **Advantages:**
    - **Peer-to-Peer Communication:** WebRTC allows direct peer-to-peer communication, which reduces latency and server load.
    - **Media Streaming:** It is ideal for real-time audio and video communication, making it suitable for applications like video conferencing.
    - **Secure:** WebRTC includes built-in security features like encryption and data channel security.
- **Disadvantages:**
    - **Complex Implementation:** Setting up WebRTC can be complex, particularly with NAT traversal and establishing secure connections.
    - **Browser Compatibility:** While widely supported, minor differences in WebRTC implementation across browsers can cause compatibility issues.
    - **Scalability:** Managing large-scale WebRTC connections can be challenging and may require additional infrastructure like SFUs (Selective Forwarding Units) or MCUs (Multipoint Control Units).

**MQTT**

- **Advantages:**
    - **Lightweight Protocol:** MQTT is designed for low-bandwidth, high-latency environments, making it ideal for IoT and mobile applications.
    - **Efficient Message Routing:** MQTT's publish/subscribe model allows for efficient message routing, reducing the need for constant connections.
    - **Quality of Service Levels:** MQTT supports different QoS levels, allowing messages to be delivered reliably even in unreliable networks.
- **Disadvantages:**
    - **Not Ideal for Large Messages:** MQTT is not well-suited for large payloads or high-throughput data streams like video.
    - **Limited Browser Support:** MQTT requires additional libraries for browser support, unlike WebSockets which are natively supported.
    - **Broker Dependency:** MQTT communication relies on a central broker, which can become a single point of failure or a bottleneck.

**WebSocket**

- **Advantages:**
    - **Full-Duplex Communication:** WebSocket provides full-duplex communication over a single connection, which is ideal for real-time applications.

  - **Low Latency:** WebSocket connections have lower latency compared to HTTP-based solutions, making them suitable for time-sensitive data exchange.
  - **Wide Browser Support:** WebSocket is natively supported in modern browsers, simplifying implementation for web applications.
- **Disadvantages:**
  - **Scalability Challenges:** While WebSocket is powerful for real-time communication, managing a large number of concurrent WebSocket connections can be challenging and resource-intensive.
  - **Server Load:** WebSocket connections are