Drone Flight Management System

Name
Krishna

SFSU Student ID
123456789

Github Username
KaiserMighty

Checkpoint #	Date Submitted
Milestone 1	02/20/2024
Milestone 2	03/05/2024
Milestone 3	04/01/2024

Table of Contents

Project Description

The Drone Flight Management System is primarily designed for military applications, yet it can be adapted for civilian use with minimal adjustments. The database provides decision-makers and organizational leaders with real-time insights into their Remotely Operated Aircraft (RPA) fleet activities. It facilitates mission planning, maintenance scheduling, crew management, and flight monitoring.

Pilots and Sensor Operators have access to upcoming flight schedules, while Maintainers can track pending maintenance requests. Planners can customize mission plans to meet organizational objectives, and leadership can assign Remotely Operated Aircrafts to missions, while also accessing comprehensive maintenance and flight history for each crew and Remotely Operated Aircraft.

Our database consolidates relevant information for user convenience, eliminating the need for multiple disparate programs and services. However, specialized tools still offer more in-depth analysis and functionality. Integration with weather services enables dynamic flight plan adjustments based on current conditions, enhancing operational safety. Similarly, integration with maintenance services provides users with simplified maintenance logs for efficient fleet management.

Planners can create detailed mission plans and flight paths with an unlimited number of waypoints and tasks, facilitating precise mission execution. Additionally, the system supports mid-flight retasking, allowing Remotely Operated Aircrafts to be reassigned to different missions and flight plans dynamically.

Given the highly sensitive nature of the data within the database, stringent security measures are implemented to mitigate the risk of security breaches. These measures aim to safeguard the integrity, confidentiality, and availability of the data. In the event of a security breach, the system is equipped to minimize the impact and extent of damages.

Functional Database Requirements

1. **Remotely Piloted Aircraft (RPA)**
    a. An RPA must have at most one pilot assigned to it.
    b. An RPA must have at most one sensor operator assigned to it.
    c. An RPA shall have maintenance conducted on it by multiple maintainers.
    d. An RPA may be classified as either available, mission planning, in-flight, or in maintenance appointment.
    e. An RPA shall be assigned to multiple missions.
    f. An RPA shall be assigned to multiple flight plans.
    g. An RPA shall have many maintenance appointments.
    h. An RPA shall have many sorties.
    i. An RPA shall be associated with at most one organization at a time.
    j. An RPA shall be assigned to at most one base at a time.
    k. An RPA shall have at most one payload.
    l. An RPA shall have one flight log.
    m. An RPA shall have one maintenance log.
2. **Organizations**
    a. An organization shall have multiple RPAs assigned to it.
    b. An organization shall have multiple pilots assigned to it.
    c. An organization shall have multiple sensor operators assigned to it.
    d. An organization shall have multiple maintainers assigned to it.
    e. An organization shall have multiple planners assigned to it.
3. **Pilots**
    a. A pilot shall have one account.
    b. A pilot shall be assigned to at most one RPA at any given time.
    c. A pilot shall belong to at most one organization at a time.
4. **Sensor Operators**
    a. A pilot shall have one account.
    b. A sensor operator shall be assigned to at most one RPA at any given time.
    c. A sensor operator shall belong to at most one organization at a time.
5. **Maintainers**
    a. A pilot shall have one account.
    b. A maintainer belongs to only one organization at a time.
    c. A maintainer shall have multiple RPAs assigned for maintenance.
    d. A maintainer shall belong to at most one base at a time.
6. **Maintenance Appointment**
    a. A maintenance appointment shall include many RPAs.
    b. A maintenance appointment shall include many maintainers.

7. **Maintenance Log**
    a. A maintenance log shall belong to one RPA
    b. A maintenance log shall have many maintenance appointments.
8. **Flight Log**
    a. A flight log shall belong to one RPA
    b. A flight log shall have many flights.
9. **Payload**
    a. A payload shall belong to many RPAs.
    b. A payload shall belong to many missions.
    c. A payload shall belong to many flight plans.
10. **Planners**
    a. A planner shall belong to at most one organization.
    b. A planner shall create many missions.
    c. A planner shall create many flight plans.
11. **Bases**
    a. A base shall have multiple RPAs.
    b. A base shall have multiple maintainers.
12. **Missions**
    a. A mission shall have many planners.
    b. A mission shall have many RPAs assigned to it.
    c. A mission shall have at most one flight plan.
13. **Flight Plans**
    a. A flight plan shall be assigned to multiple missions.
    b. A flight plan shall have many planners.
14. **Sortie**
    a. A flight shall have at least one RPA assigned to it.
    b. A flight shall have at least one pilot assigned to it.
    c. A flight shall have at least one sensor operator assigned to it.
    d. A flight shall have at most one payload assigned to it.
    e. A flight shall have at least one mission assigned to it.
    f. A flight shall have at least one flight plan assigned to it.
15. **Accounts**
    a. An account may be a pilot, sensor operator, maintainer, or planner.
    b. An account shall be logged into by many devices.
16. **Devices**
    a. A device can belong to at most one account.
    b. A device can log into an account.

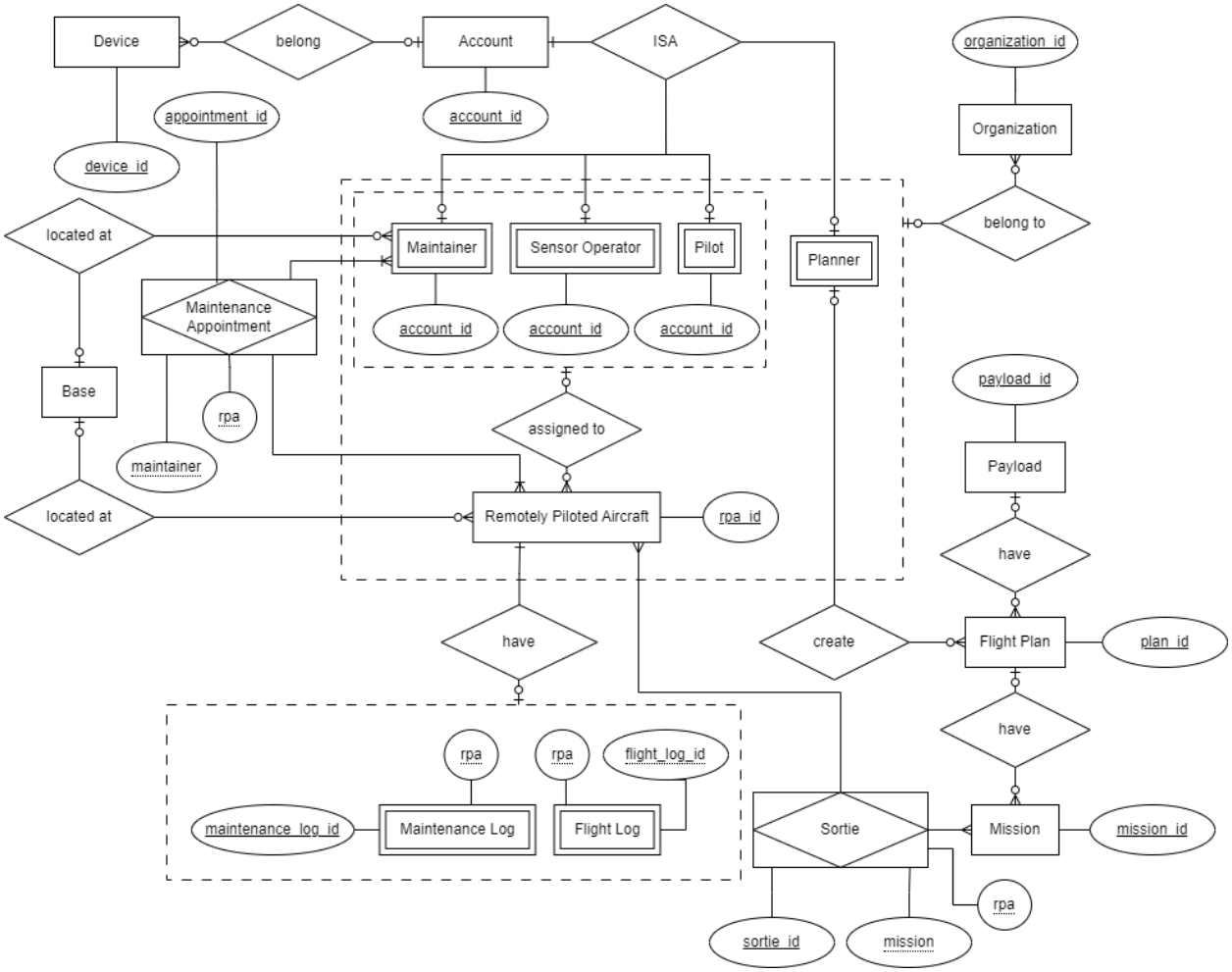
Non-functional Database Requirements

1. **Performance**
    a. The database shall support many concurrent users without significant degradation in performance.
    b. The database shall respond quickly to user prompts with minimal latency.
    c. The database shall support query optimization with appropriate indexing.
    d. The database shall support real time tracking of RPA flights with minimal latency.
    e. The database shall support real time tracking of RPA maintenance with minimal latency.
2. **Security**
    a. The database shall only use encrypted passwords.
    b. The database shall be backed up everyday at 23:59 CST.
    c. The admins shall control access to the database for each user.
    d. The database shall support audit logs for tracking user activity.
    e. The database shall regularly conduct penetration testing and patch out any security vulnerabilities.
3. **Scalability**
    a. The database shall be scaled horizontally through distributive computing to support large data loads.
    b. The database shall be scaled vertically through upgrading hardware to handle increased demands.
    c. The database shall support dynamic resources allocation based on current workload.
    d. The database shall support global servers to minimize latency for users if they are abroad.
    e. The database shall scale down if the workload is low to remain cost effective and optimize resource utilization.
4. **Capability**
    a. The database shall support a wide range of RPAs regardless of manufacturer or model.
    b. The database shall interface with the RPA's telemetry computer to make real time flight tracking possible.
    c. The database shall interface with maintenance programs and systems to make real time maintenance possible.
    d. The database shall be extended to be made compatible for manned aircraft if needed.
    e. The database can interface with weather applications to notify planners and pilots of weather conditions along their flight plan.

5. **Environmental**
    a. The database shall be executed on distributed servers.
    b. The database shall be easily migratable in the advent of new technologies and services.
    c. The database shall be physically located at secure locations.
    d. The database shall be executed in containers to maximize portability and security to the host.
    e. The database shall have redundant distributed servers to minimize performance impact and data lose in the event of failure of any one server.
6. **Coding Standards**
    a. The database shall use consistent coding and naming conventions throughout.
    b. The database shall gracefully handle errors without detrimentally affecting the rest of the database.
    c. The database shall use object oriented programming to support modularity and scaling.
    d. The database shall be well documented for easy understandability for users, and admins alike.
    e. The database shall be coded in the most optimal time and space complexity manner to maximize performance.
7. **Media Storage**
    a. Sensor footage and images may be saved into the database if required.
    b. Maintainers shall include images or media if required for their work.
    c. Planners shall include maps and visual representations of flight plans for readability.
    d. Flight telemetry log can be saved on the database for each flight.
    e. Any stored media should be backed up to multiple locations for failure redundancy.
    f. A wide range of media formats shall be supported based on the specifications of each RPA.
8. **Privacy**
    a. Non-admin users are not allowed access into mission data or history to prevent bad actors from leaking sensitive information.
    b. The database does not store any private information about the users by nature.
    c. Users are only compartmentalized into only being able to access their section of the database (maintainers cannot access mission plans, etc).
    d. Data shall be anonymized and pseudonymized wherever possible to minimize damage in the event of a leak.

e. Only the minimum amount of data required shall be stored in order to minimize the risk of security breaches.

Entity Relationship Diagram



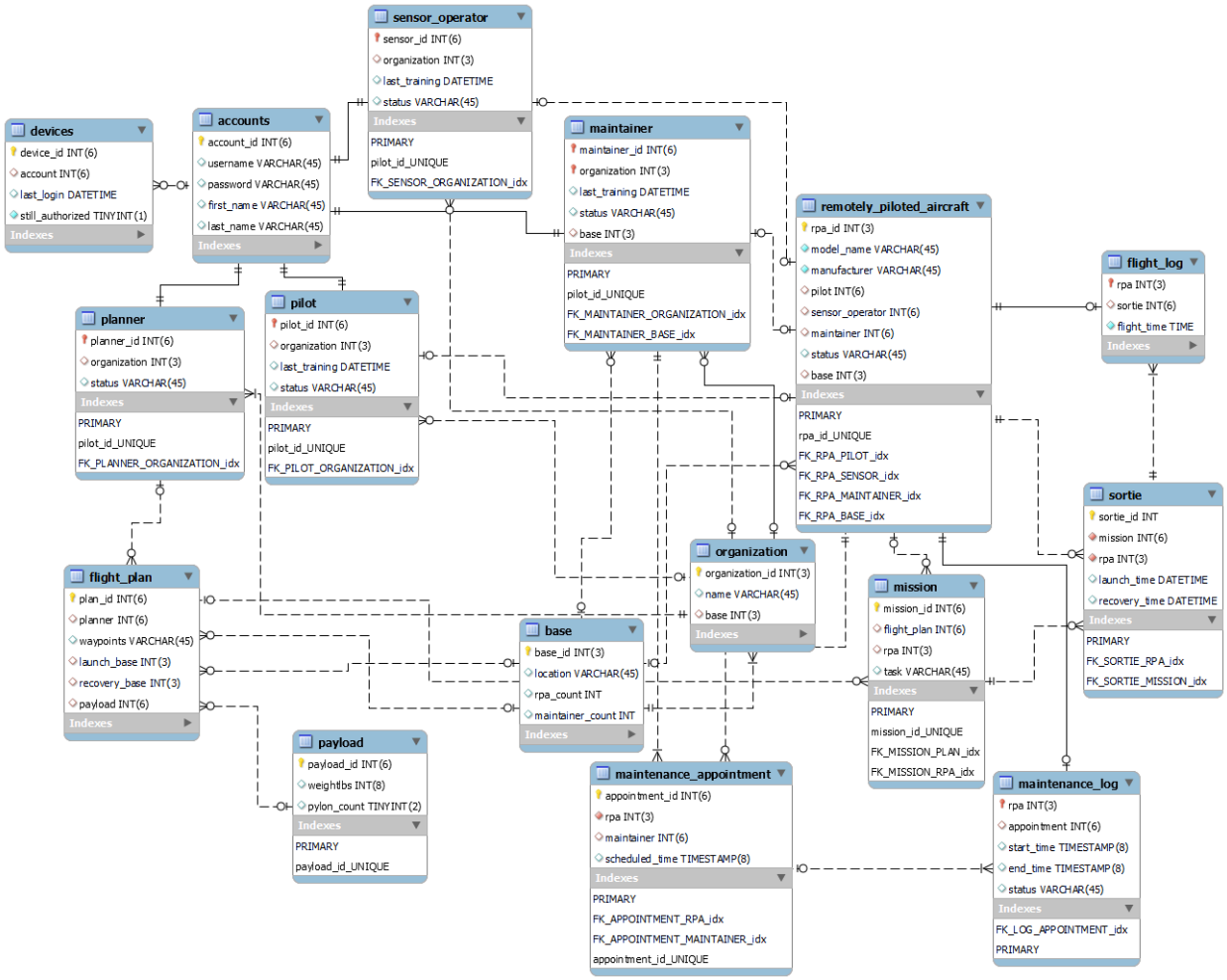
Entity Description

1. **Remotely Piloted Aircraft (Strong)**
    a. rpa_id: key, numeric
    b. model_name: composite, alphanumeric
    c. manufacturer: composite, alphanumeric
    d. status: enum, numeric
    e. base: key, numeric
2. **Account (Strong)**
    a. account_id: key, numeric
    b. username: composite, alphanumeric
    c. password: alphanumeric
    d. first_name: string, alphabetical
    e. last_name: string, alphabetical
3. **Pilot (Weak)**
    a. account: weak key, numeric
    b. organization: weak key, numeric
    c. last_training: multivalue, timestamp
    d. status: enum, numeric
4. **Sensor Operator (Weak)**
    a. account: weak key, numeric
    b. organization: weak key, numeric
    c. last_training: multivalue, timestamp
    d. status: enum, numeric
5. **Maintainer (Weak)**
    a. account: weak key, numeric
    b. organization: weak key, numeric
    c. last_training: multivalue, timestamp
    d. status: enum, numeric
    e. base: key, numeric
6. **Planner (Weak)**
    a. account: weak key, numeric
    b. organization: weak key, numeric
    c. status: enum, numeric
7. **Flight Plan (Strong)**
    a. plan_id: key, numeric
    b. waypoints: composite, alphanumeric
    c. launch_base: weak key, numeric
    d. recovery_base: weak key, numeric
    e. payload: weak key, numeric

8. **Mission (Strong)**
    a. mission_id: key, numeric
    b. plan: weak key, numeric
    c. rpa: weak key, numeric
    d. pilot: weak key, numeric
    e. sensor_operator: weak key, numeric
    f. task: composite, alphanumeric
9. **Sortie (Weak)**
    a. sortie_id: key, numeric
    b. mission: weak key, numeric
    c. rpa: weak key, numeric
    d. launch_time: multivalue, timestamp
    e. flight_time: multivalue, timestamp
10. **Maintenance Appointment (Weak)**
    a. appointment_id: key, numeric
    b. rpa: weak key, numeric
    c. maintainer: weak key, numeric
    d. schedule: multivalue, timestamp
11. **Flight Log (Weak)**
    a. rpa: weak key, numeric
    b. sortie: weak key, numeric
    c. flight_time: multivalue, timestamp
12. **Maintenance Log (Weak)**
    a. rpa: weak key, numeric
    b. appointment: weak key, numeric
    c. status: enum, numeric
    d. started: multivalue, timestamp
    e. ended: multivalue, timestamp
13. **Payload (Weak)**
    a. payload_id: key, numeric
    b. total_weight: numeric
    c. pylon_count: numeric
14. **Base (Weak)**
    a. base_id: key, numeric
    b. location: composite, alphanumeric
15. **Device (Weak)**
    a. device_id: key, numeric
    b. account: weak key, numeric
    c. last_login: multivalue, timestamp
    d. still_authorized: boolean

16. **Organization (Weak)**
    a. organizaion_id: key, numeric
    b. organization_name: composite, alphanumeric
    c. base: weak key, numeric
    d. assigned_pilots: generated, numeric
    e. assigned_sensor_operators: generated, numeric
    f. assigned_maintainers: generated, numeric
    g. assigned_planners: generated, numeric

Entity Establishment Relationship Diagram



Constraints Description

Table	FK	ON DELETE	ON UPDATE	Comment
Devices	account	SET NULL	CASCADE	Devices are linked to accounts, so updates cascaded. Since the lower bound is zero, delete is null.
Flight_Plan	planner	SET NULL	CASCADE	Update planner if edited, but set null if deleted. Flight plan can still be useful without its creator.
Flight_Plan	launch_base	SET NULL	CASCADE	Update existing plans if ids are moved, but set them to null if deleted. This should alert planners to assign new bases.
Flight_Plan	recovery_base	SET NULL	CASCADE	Update existing plans if ids are moved, but set them to null if deleted. This should alert planners to assign new bases.
Flight_Plan	payload	SET NULL	CASCADE	Update existing plans if ids are moved, but set them to null if deleted. This should alert planners to assign new payloads.
Maintainer	maintainer_id	CASCADE	CASCADE	Maintainers are child types of account, cascade any changes to the parent account.
Maintainer	organization	SET NULL	CASCADE	Maintainer is independent of organization, set null if deleted and just assign a new one.
Maintainer	base	SET NULL	CASCADE	Maintainer is independent of base, set null if deleted and just assign a new one.
Maintenance _Appointment	rpa	NO ACTION	NO ACTION	No action for accurate record keeping purposes.
Maintenance _Appointment	maintainer	NO ACTION	NO ACTION	No action for accurate record keeping purposes.

Maintenance_Log	rpa	CASCADE	CASCADE	Log is dependent on RPA, cascade all changes to parent RPA.
Maintenance_Log	appointment	NO ACTION	NO ACTION	No action for accurate record keeping purposes.
Mission	flight_plan	SET NULL	CASCADE	Cascade edits, but set null on delete. Mission can be assigned a new plan.
Mission	rpa	SET NULL	CASCADE	Cascade edits, but set null on delete. Mission can be assigned a new RPA.
Organization	base	SET NULL	CASCADE	Organization is independent of base, cascade edits but set null on delete and assign new base.
Pilot	pilot_id	CASCADE	CASCADE	Pilots are child types of account, cascade any changes to the parent account.
Pilot	organization	SET NULL	CASCADE	Pilots are independent of organization, set null if deleted and just assign a new one.
Planner	planner_id	CASCADE	CASCADE	Planners are child types of account, cascade any changes to the parent account.
Planner	organization	SET NULL	CASCADE	Planner are independent of organization, set null if deleted and just assign a new one.
Remotely_Piloted_Aircraft	pilot	SET NULL	CASCADE	RPAs are independent of pilots, set null if deleted and just assign a new one.
Remotely_Piloted_Aircraft	sensor_operator	SET NULL	CASCADE	RPAs are independent of sensor operators, set null if deleted and just assign a new one.
Remotely_Piloted_Aircraft	maintainer	SET NULL	CASCADE	RPAs are independent of maintainers, set null if deleted and just assign a new one.
Remotely_Piloted_Aircraft	base	SET NULL	CASCADE	Independent of bases, set null if deleted and assign a new one.

Sensor_Operator	sensor_id	CASCADE	CASCADE	Sensor operators are child types of account, cascade any changes to the parent account.
Sensor_Operator	organization	SET NULL	CASCADE	Sensor operators are independent of organization, set null if deleted and just assign a new one.
Sortie	rpa	NO ACTION	NO ACTION	No action for accurate record keeping purposes.
Sortie	mission	NO ACTION	NO ACTION	No action for accurate record keeping purposes.
Flight_Log	rpa	CASCADE	CASCADE	Log is dependent on RPA, cascade all changes to parent RPA.
Flight_Log	sortie	NO ACTION	CASCADE	Cascade edits, but no action on delete. Required for accurate record keeping.