Drone Survey Management
Assignment Documentation

SUBMITTED BY

RIPUNJAY CHOUDHURY

Email id: ripunjaychoudhury42@gmail.com

Contact no: 9864832057

Table of Contents

1. Introduction

The Drone Survey Management Platform is a web application designed to facilitate planning, execution, and monitoring of drone survey missions. It serves drone operators and facility managers with tools for mission planning, fleet management, monitoring, and reporting.

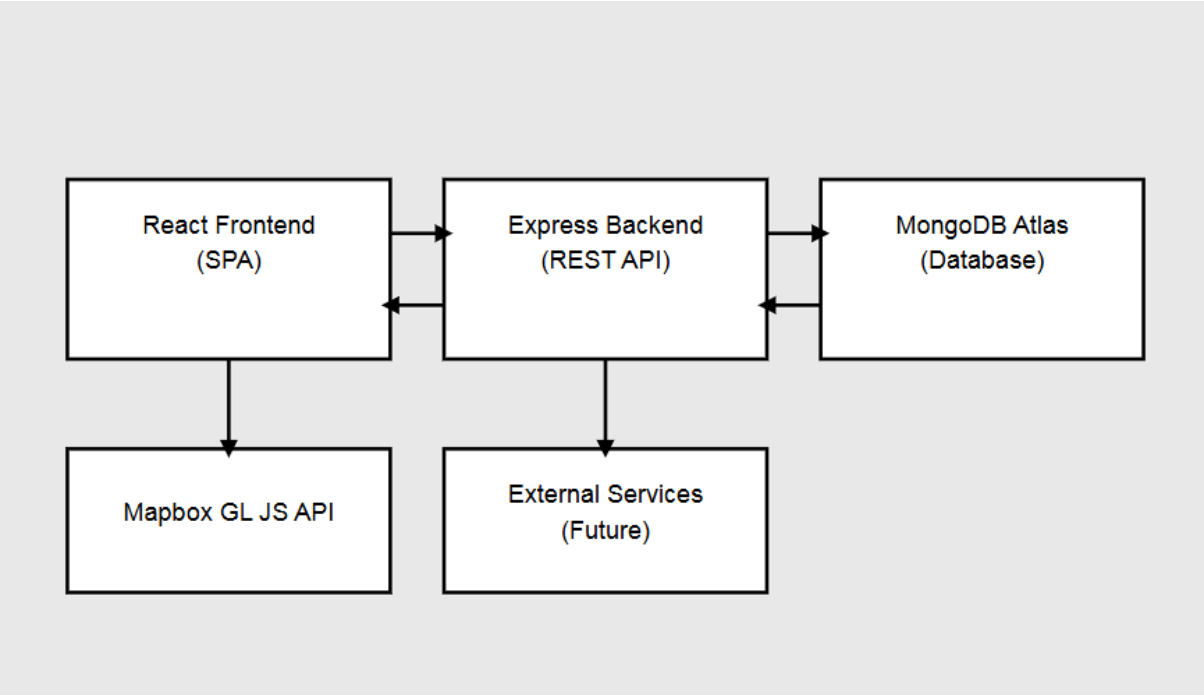
1.1 Purpose & Scope

- Mission planning with geospatial capabilities
- Drone fleet management
- Real-time mission monitoring
- Survey reporting and analytics
- User and role management

2. System Architecture

The platform employs a three-tier architecture:

2.1 Architecture Diagram



2.2 Key Components

- **Frontend Layer**: React SPA with responsive design
- **Backend Layer**: Express.js REST API with JWT authentication
- **Data Layer**: MongoDB Atlas database with geospatial capabilities

2.3 System Interfaces

- **User Interface**: Web browser-based interface
- **API Interface**: RESTful endpoints for frontend-backend communication
- **Database Interface**: Mongoose ODM for database interactions
- **External Service Interfaces**: Mapbox GL JS API integration

3. Technical Stack

3.1 Frontend Technologies

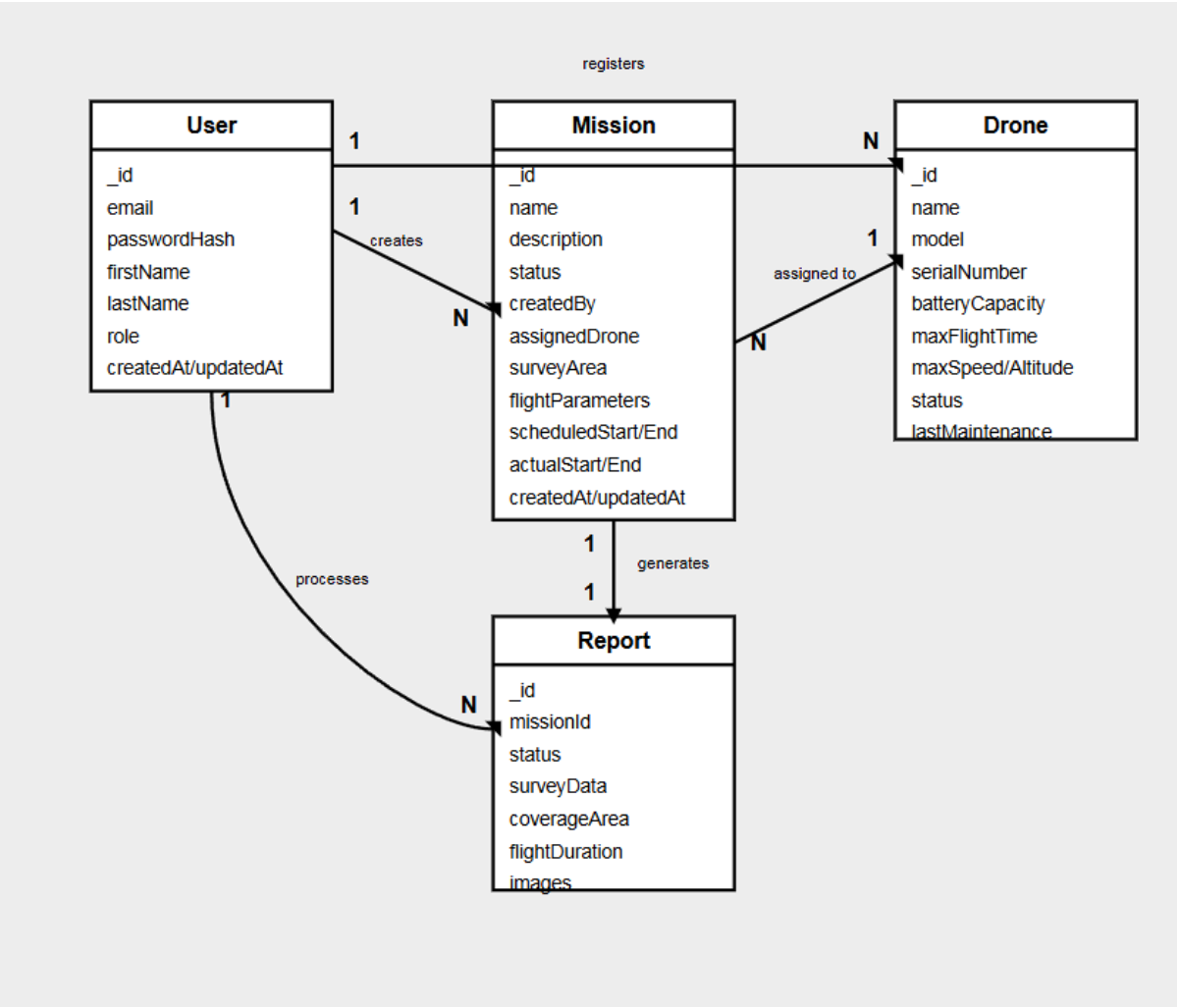
- React 19, React Router 7
- Tailwind CSS 4
- Zustand for state management
- Mapbox GL JS
- Vite build system

3.2 Backend Technologies

- Node.js,
- Express
- MongoDB with Mongoose
- JWT Authentication
- Bcrypt for password security

4. Database Design

4.1 ER Diagram



4.2 Key Entities

- **User**: Authentication and user management
- **Drone**: Fleet inventory and capabilities
- **Mission**: Survey planning and execution
- **Report**: Survey results and analysis

4.3 Data Relationships

- One User can create many Missions
- One Drone can be assigned to many Missions (over time)
- One Mission has one assigned Drone
- One Mission generates one Report

4.4 Database Schema

The MongoDB database uses the following document schemas:

User Schema

```
{
  _id: ObjectId,
  email: String,
  passwordHash: String,
  firstName: String,
  lastName: String,
  role: String,
  createdAt: Date,
  updatedAt: Date
}
```

Drone Schema

```
{
  _id: ObjectId,
  name: String,
  model: String,
  serialNumber: String,
  batteryCapacity: Number,
  maxFlightTime: Number,
  maxSpeed: Number,
  maxAltitude: Number,
  status: String,
  lastMaintenance: Date,
  maintenanceHistory: Array,
  createdAt: Date,
  updatedAt: Date
}
```

Mission Schema

```
{
  _id: ObjectId,
```

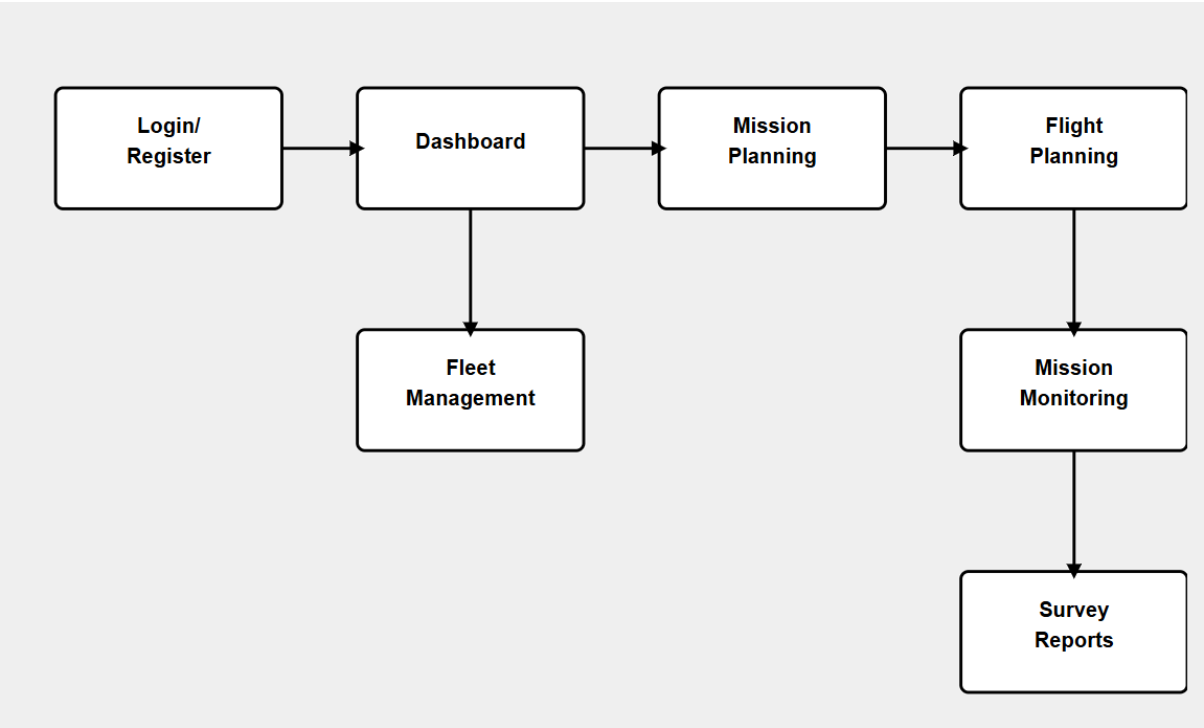
```
  name: String,
  description: String,
  status: String,
  createdBy: ObjectId,
  assignedDrone: ObjectId,
  surveyArea: {
    type: String,
    coordinates: Array
  },
  flightParameters: {
    altitude: Number,
    speed: Number,
    overlapPercentage: Number
  },
  scheduledStart: Date,
  scheduledEnd: Date,
  actualStart: Date,
  actualEnd: Date,
  createdAt: Date,
  updatedAt: Date
}
```

Report Schema

```
{
  _id: ObjectId,
  missionId: ObjectId,
  status: String,
  surveyData: Object,
  coverageArea: Number,
  flightDuration: Number,
  images: Array,
  generatedAt: Date
}
```

5. User Flows

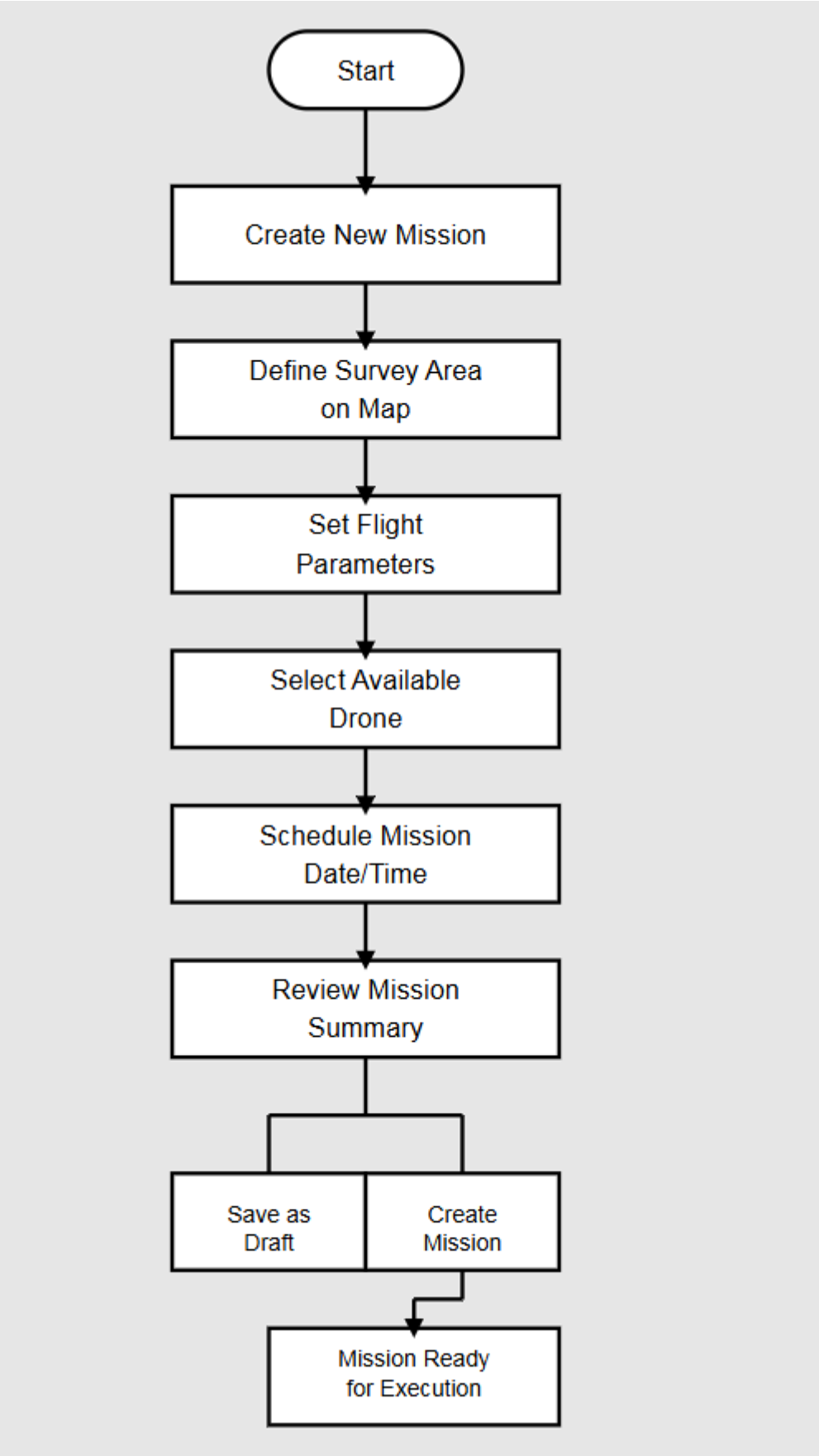
5.1 Main User Flow Diagram



5.2 Core Flows

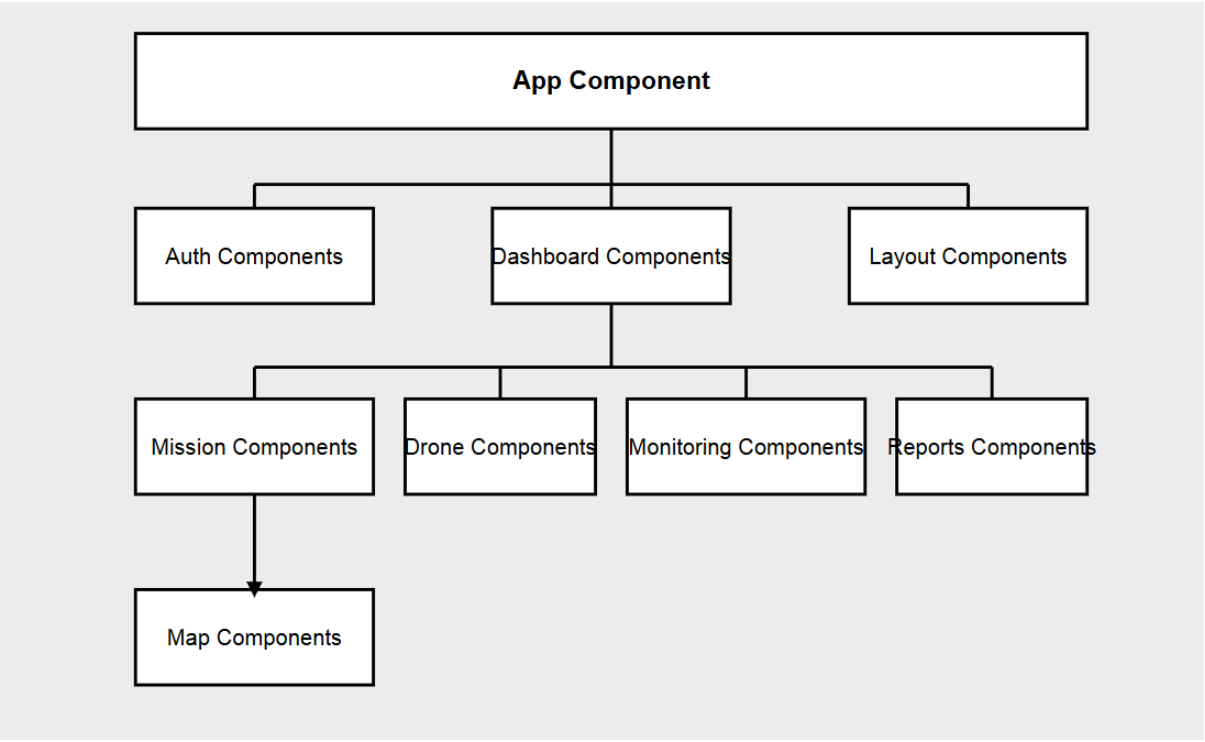
- Authentication (register, login, password reset)
- Mission planning process
- Fleet management
- Mission monitoring
- Report generation

5.3 Mission Planning Process



6. Component Structure

6.1 Frontend Component Hierarchy



6.2 Key Component Groups

- Authentication components
- Dashboard and layout components
- Mission management components
- Drone management components
- Monitoring and reporting components

6.3 Component Interaction Patterns

- Parent-child prop passing
- Context-based state sharing
- Zustand store interactions
- Event-based communication

7. API Structure

7.1 Core API Endpoints

- **/api/auth**: Registration, login, password management
- **/api/users**: User management
- **/api/missions**: Mission CRUD operations
- **/api/drones**: Drone fleet management
- **/api/reports**: Survey report access

7.2 API Authentication

- JWT token-based authentication
- Role-based access control
- Token refresh mechanisms
- Secure HTTP-only cookies

7.3 Request/Response Formats

- JSON-based communication
- Standard HTTP status codes
- Consistent error response format
- Pagination for list endpoints

8. Testing Documentation

8.1 Testing Approach

The application was validated using a combination of:

1. **Unit Tests**
    - Component-level tests for frontend React components
    - Function-level tests for utility and service functions
    - Model validation tests for MongoDB schemas
2. **Integration Tests**
    - API endpoint tests with mock requests
    - Database integration tests
    - Component integration tests
3. **End-to-End Tests**
    - User flow simulations
    - Cross-browser compatibility tests
    - Mobile responsiveness tests

8.2 Test Cases Summary

Area	Test Type	Description	Expected Outcome
Authentication	Unit	Validate JWT token generation	Token contains correct user data
Authentication	Integration	Login with valid credentials	User authenticated, token returned
Authentication	Integration	Login with invalid credentials	Error message returned
Mission Planning	Unit	Validate mission data model	Data validation works correctly
Mission Planning	Integration	Create mission with valid data	Mission created successfully
Mission Planning	Integration	Create mission with invalid data	Appropriate validation errors
Fleet Management	Unit	Drone status calculation	Status calculated correctly
Fleet Management	Integration	Fetch available drones	Only available drones returned
Mission Monitoring	Integration	Update mission status	Status updated correctly
Mission Monitoring	E2E	Simulate mission progress	UI updates with mission progress
Map Component	Unit	Flight pattern calculation	Flight paths generated correctly
Map Component	Integration	Draw survey area	Area drawn and saved correctly
Reports	Integration	Calculate survey area metrics	Metrics calculated accurately
UI/UX	E2E	Mobile responsiveness	UI adapts to different screen sizes

8.3 Testing Tools

- Jest for unit and integration tests
- React Testing Library for component tests
- ESLint and Prettier for code quality

9. Critical Explanations

9.1 How the Problem Was Approached

1. **User-Centered Design**:
    - Started with identifying key user personas (drone operators, fleet managers)
    - Mapped out critical user journeys and pain points to address
    - Designed interfaces focused on simplicity and operational efficiency
2. **Domain-Driven Design**:
    - Analyzed the drone survey domain to identify core entities and relationships
    - Created domain models that accurately reflect real-world drone operations
    - Used ubiquitous language across codebase reflecting industry terminology
3. **Iterative Development**:
    - Built core features first (authentication, mission planning, fleet management)
    - Added advanced features later (monitoring, reporting)
    - Validated with stakeholders after each iteration
4. **Component-Based Architecture**:
    - Designed reusable components for common UI elements
    - Separated concerns between data fetching and presentation
    - Created specialized components for complex interactions (mapping, mission control)

9.2 Trade-offs Considered

1. **Performance vs. Development Speed**:
    - Used Zustand instead of Redux for simpler state management with acceptable performance
    - Implemented virtualized lists for large data sets to maintain responsiveness
2. **Flexibility vs. Complexity**:
    - Created flexible mission scheduling system at cost of increased complexity
    - Standardized drone data model while allowing for manufacturer-specific attributes
3. **Real-time Updates vs. Backend Complexity**:
    - Initial implementation uses polling for mission updates
    - Future enhancement planned for true real-time communication
4. **Map Provider Selection**:
    - Selected Mapbox for better custom visualization capabilities
    - Trade-off: Less widespread familiarity but better suited for drone path visualization

9.3 Safety and Adaptability Strategy

1. **Safety Measures**:
    - Implemented comprehensive validation for all flight parameters
    - Added altitude and speed limits based on drone specifications
    - Created conflict detection for overlapping mission schedules
    - Implemented drone health monitoring with maintenance alerts
    - Added mission abort functionality for emergency situations
2. **System Adaptability**:

    - Used modular architecture to facilitate extension
    - Implemented feature flags for gradual rollout of new capabilities
    - Created abstraction layers for external services (mapping, weather)
    - Documented API interfaces for potential third-party integrations
    - Designed database schema to accommodate future data requirements
3. **Resilience Strategies**:
    - Implemented error handling and recovery mechanisms
    - Added logging for operations monitoring and debugging
    - Created data backup and restore capabilities
4. **Future-Proofing**:
    - Designed for future integration with regulatory compliance frameworks
    - Implemented versioned API to support backward compatibility
    - Documentation of all subsystems to support future development

10. Conclusion

The Drone Survey Management Platform represents a comprehensive solution designed with a focus on usability, safety, and extensibility. The architecture provides a solid foundation for current needs while allowing for future growth and adaptation to emerging requirements in the drone survey industry.

By following established design patterns and implementing robust testing practices, the system ensures reliable operation in mission-critical scenarios. The trade-offs made during development were carefully considered to balance immediate needs with long-term sustainability.

The planned future enhancements, including real-time communication, weather API integration, and advanced image processing capabilities, will further strengthen the platform's value proposition and maintain its competitive edge in the evolving drone management landscape.