Asset Management App

Project Overview:

This project involves creating an Asset Management System, which will manage assets, inventories, asset tracking, and reporting functionalities. The system will allow administrators to add, update, and remove assets, while users can view asset information, availability, and other details. Agile methodology will be used to ensure iterative development, regular feedback, and flexibility throughout the project.

Phase 1: Project Initialization

1. **Objective**:
    - Set the foundation of the project and prepare the team and stakeholders for the Agile process.
2. **Key Deliverables**:
    - Project Scope and Vision Document
    - High-level requirements for the Asset Management System
    - Initial product backlog creation
    - Team structure and roles defined
3. **Key Activities**:
    - Conduct project kickoff meeting with stakeholders.
    - Define key goals and objectives for the system.
    - Identify stakeholders and gather initial requirements.
    - Develop a high-level project roadmap outlining major milestones.
    - Set up a product backlog in a project management tool (e.g., Jira, Trello).
4. **Tools**:
    - Jira and Trello for backlog and task management
    - Visual Studio and Android Studio for coding
    - AWS for cloud services and backend management

Phase 2: Sprint 0 - Foundation Setup

1. **Objective**:
    - Prepare the team and technical infrastructure for the development process.
    - Conduct sprint planning for Sprint 1.
2. **Key Deliverables**:
    - System architecture diagram
    - User stories for the first sprint
    - Sprint planning meeting and backlog grooming session
    - Establish development environment
3. **Key Activities**:
    - Define core system architecture (backend, frontend, database).
    - Identify key technologies (React Native, Node.js, PostgreSQL, etc.).
    - Set up the development environment and repository (e.g., GitHub).
    - Review user stories, estimate them, and create a sprint backlog.
4. **Tools**:
    - GitHub (code repository)
    - Node.js (backend)
    - Gaphql (API
    - PostgreSQL (database)
    - React Native (frontend)

Phase 3: Sprint Cycles (Sprint 1 to N)

High-Level Project Plan:

Agile Strategy:

The project will follow Scrum methodology with the following components:

- **Sprints**: 2-week iterations.
- **Daily Standups**: Quick sync meetings to track progress and discuss blockers.
- **Sprint Planning**: Before each sprint, we will plan the backlog items for the next sprint.
- **Sprint Review & Retrospective**: After each sprint, we will demo the completed work and discuss what worked well or could improve.

Sprint 1: User Account Creation and Base Setup on AWS (2 weeks)

Goal:

- Establish the basic user registration and authentication system.
- Set up the AWS infrastructure to support the asset management application.
- Ensure the project foundation is built for scalability.

Key Deliverables:

- **User Registration & Login System**: Allow users to create accounts, login, and reset passwords.
- **AWS Infrastructure Setup**: Use AWS services such as S3, RDS, Lambda, and API Gateway for the backend.
- **Database Setup**: Set up an initial database (likely PostgreSQL) for user accounts and initial asset data schema.
- **Authentication**: Implement user authentication using AWS Cognito or another secure authentication service.
- **Basic UI/UX**: Create simple pages for registration and login, ensuring it is mobile-friendly.

Tasks:

1. **AWS Cognito Setup**:
    - Configure user pools for user authentication and registration.
    - Enable features like email confirmation, password reset.
2. **Database Initialization**:
    - Design the user schema (UserID, Email, Password, Role, etc.).
    - Set up an RDS instance for the database.

3. **Create User Registration API**:
    - Use AWS Lambda and API Gateway to create serverless functions.
    - Create endpoints for user registration and login.
4. **Frontend Setup**:
    - Implement basic React/React Native interface for registration and login forms.
    - Connect forms to the AWS backend.
5. **Security**:
    - Implement basic security measures, including data encryption and validation.
6. **CI/CD Pipeline Setup**:
    - Set up AWS CodePipeline or other CI/CD tools to deploy the frontend and backend code.

Sprint 2: Asset Types and Inventory Setup (2 weeks)

Goal:

- Build the functionality for asset types and inventories.
- Allow users to categorize assets by type and track inventory data.
- Implement QR code functionalities to simplify asset identification and tracking.

Key Deliverables:

- **Asset Type Setup**: Create a list of asset types like laptops, equipment, etc.
- **Inventory UI**: Develop a page where users can view asset types and filter inventory.
- **Inventory Data**: Allow users to add basic asset details (type, description, status, etc.).
- **QR code generation:** Generate QR Codes in bulk and assign them to assets.
- **QR code Scanning**: Scan and identify assets using QR Code with option to Multi-asset QR code scanning.

Tasks:

1. **Asset Type Database Setup**:
    - Expand the database schema to include asset categories, availability status, etc.
2. **API for Asset Types**:
    - Create APIs for adding and retrieving asset types using AWS Lambda.
3. **Frontend:**
    - Implement a UI for managing asset types (React or React Native).

- Create forms for asset type addition.

4. **Inventory View**:
    - Build the frontend for users to view and manage asset inventory.
5. **QR Code Generation for Assets:**
    - Automatically generate QR codes when a new asset is created.
    - Display the QR code in the asset details page for easy identification.
    - Ensure the QR code contains necessary asset information for tracking.
6. **Multi-Scan QR Code Functionality:**
    - Implement a multi-asset scan feature to update the status of several assets at once using QR codes.
    - Allow batch actions, such as bulk check-ins/outs of assets via QR code scanning.
    - Enable the scanning of QR codes for individual asset details viewing and updates.

Sprint 3: Asset Management Module (2 weeks)

Goal:

- Build the core asset management features, allowing users to manage assets effectively.
- Enhance user interface to support asset-related operations efficiently.

Key Deliverables:

- **Asset Creation**: Users can add new assets with details like serial number, purchase date, warranty, etc.
- **Asset Tracking**: Track the current location, assignment, and status (available, in use, rented).
- **Editing/Deleting Assets**: Allow users to edit or remove asset records.

Tasks:

1. **Database Schema Expansion**:
    - Expand the asset schema to include all necessary fields.
2. **Full CRUD Functionality for Assets:**
    - Implement create, read, update, and delete operations for asset management.
    - Allow editing and viewing of asset details (type, location, serial number, purchase date, etc.).
    - Ensure assets can be assigned to users or locations.
3. **Advanced Search and Filters:**
    - Implement search functionality to find assets based on type, serial number, status (e.g., Available, In Use), and location.
    - Enable filtering for specific asset types and other attributes.

4. **Asset Management API**:
    - Implement APIs for asset creation, updating, and deletion.
5. **Assigning Assets to Users and Locations:**
    - Develop functionality to assign assets to specific users or locations.
    - Provide a history of asset assignment and movement.
6. **User Interface Enhancements:**
    - Create an intuitive UI for asset management, making it easy to perform CRUD operations.
    - Integrate QR code functionality within the asset management interface, ensuring seamless use.
7. **Testing and User Feedback:**
    - Test all asset management functionalities, including QR code generation and scanning.
    - Collect feedback from users regarding the ease of use for managing assets and scanning operations.

Sprint 4: Reporting and Asset Lifecycle Management (2 weeks)

Goal:

- Setup user tasks and build reporting features and track the lifecycle of assets from acquisition to disposal.

Key Deliverables:

- **User tasks:** Allow admins/system to create asset tasks for users
- **Asset Reports**: Generate reports for asset allocation, purchase date, warranty expiry, etc.
- **Lifecycle Tracking**: Track the lifecycle events of an asset (assigned, maintained, decommissioned).
- **Export Reports**: Allow users to export reports as CSV or PDF.

Tasks:

1. **Setup asset-based user tasks:**
    - Ability to create, assign, and manage tasks related to specific assets (e.g., maintenance, inspection).
    - Integration of task due dates and reminders within the system.
    - Track task progress and completion history for assets.
2. **Frontend UI for asset reports and task management:**
    - User-friendly interface to view asset-related tasks.

    - Notifications for task deadlines and overdue tasks.
3. **Reporting Database**:
    - Implement database functions to gather data for reporting.
4. **Reporting API**:
    - Create an API to generate asset reports.
5. **Frontend for Reports**:
    - Create a reporting dashboard where users can customize and view reports.
6. **Export Functionality**:
    - Allow users to export asset data in various formats.
7. **Lifecycle Event Tracking**:
    - Track asset changes over time, from purchase to decommissioning.

Sprint 5: Notifications and User Roles (2 weeks)

Goal:

- Implement a notification system for asset events (e.g., warranty expiry) and set up user roles for the system.

Key Deliverables:

- **Notifications**: Notify users of important asset events via email or in-app messages.
- **User Roles**: Define different roles like Admin, User, and Viewer with specific permissions.

Tasks:

1. **Notification System**:
    - Use AWS SNS or SES for email and push notifications.
    - Implement notifications for events such as asset expiration, new asset assignment, etc.
2. **User Role Management**:
    - Create roles and permissions for different user levels.
    - Ensure that only admins can manage assets and view all data.
3. **UI for Notifications**:
    - Add a section where users can view recent notifications.

Sprint 6: Final Testing and Optimization (2 weeks)

Goal:

- Test the full application for bugs, performance, and usability.

- Optimize for speed and user experience.

Key Deliverables:

- **Testing**: Conduct full end-to-end tests across all modules.
- **Optimization**: Ensure the system is optimized for large-scale data.
- **Bug Fixes**: Resolve any critical issues before the final release.

Tasks:

1. **User Testing**:
    - Conduct usability testing with sample users.
    - Gather feedback and improve UI/UX based on user input.
2. **Load Testing**:
    - Test the application with large data sets to ensure scalability.
3. **Bug Fixing**:
    - Fix any critical issues or bugs found during testing.
4. **Performance Optimization**:
    - Ensure the system runs efficiently on AWS, with optimized Lambda functions, API Gateway performance, and database queries.

Sprint 7: Deployment and Go-Live (2 weeks)

Goal:

- Deploy the system to production and train users on its use.

Key Deliverables:

- **Production Deployment**: Deploy the system on AWS and ensure all services are up and running.
- **Training and Documentation**: Create documentation and train users to use the system effectively.

Tasks:

1. **Final Deployment**:
    - Deploy the latest version of the application to production.
2. **User Documentation**:
    - Create user manuals and tutorials.
3. **Training Sessions**:
    - Conduct training sessions for admins and users.

Project Milestones:

Milestone 1: User Account Creation & AWS Base Setup (End of Sprint 1)

- **Expected Completion**: Sprint 1 (2 weeks)
- **Key Deliverables**:
    - User registration and login system (with AWS Cognito).
    - Initial AWS infrastructure setup (S3, RDS, Lambda, API Gateway).
    - User authentication and authorization (Cognito with secure API).
    - Basic database schema for user accounts and security.

Milestone 2: Asset Type & Inventory Setup (End of Sprint 2)

- **Expected Completion**: Sprint 2 (2 weeks)
- **Key Deliverables**:
    - Asset type categorization (e.g., laptops, equipment).
    - Inventory interface allowing users to view, filter, and manage asset types.
    - Backend APIs for managing asset types and inventories.
    - Database schema extended to include asset types and availability statuses (Available, In Use, Rented, etc.).
    - QR Code Generation:
        - Automatic QR code generation for each asset upon creation.
        - Display QR code for each asset in the asset details page.
    - QR Code Scanning:
        - Implement multi-asset scanning functionality.
        - Use QR code scanning to view or update multiple asset statuses at once.
        - Allow asset status updates, such as checking in or out, via QR code scanning.

Milestone 3: Asset Management Module (End of Sprint 3)

- **Expected Completion**: Sprint 3 (2 weeks)
- **Key Deliverables**:
    - Full asset CRUD functionality (Create, Read, Update, Delete).
    - Advanced search filters: Search assets by various attributes (type, location, serial number, etc.).
    - Assign assets to users and track asset ownership or custodianship.

- Asset location tracking: View and update asset locations in real-time.
- Frontend UI for asset management:
  - User-friendly interface for creating, viewing, and editing assets.
  - Integration of QR code generation and scanning capabilities within the asset details UI.

Milestone 4: Reporting and Asset Lifecycle Management (End of Sprint 4)

- **Expected Completion**: Sprint 4 (2 weeks)
- **Key Deliverables**:
  - Tasks generation for assets
  - Frontend UI for asset reports and task management
  - Reporting functionality: Generate reports for asset allocation, purchase details, and warranty tracking.
  - Export asset reports in CSV or PDF formats.
  - Lifecycle event tracking: Track the status of assets (assigned, in repair, decommissioned).
  - Frontend UI for generating and viewing asset reports.

Milestone 5: Notifications & User Roles (End of Sprint 5)

- **Expected Completion**: Sprint 5 (2 weeks)
- **Key Deliverables**:
  - Notification system for asset-related events (e.g., warranty expiry, assignment).
  - Role-based access control: Admins, Managers, and Regular Users with different permissions.
  - In-app notifications and email alerts using AWS SNS or SES.
  - Frontend and backend support for managing and displaying user roles and permissions.

Milestone 6: Final Testing, Optimization, & Bug Fixing (End of Sprint 6)

- **Expected Completion**: Sprint 6 (2 weeks)
- **Key Deliverables**:
  - End-to-end testing of the entire system.
  - Load testing for scalability (ensuring the system handles a large number of assets).

- Optimizing backend performance (Lambda functions, API Gateway, and database queries).
- Resolving any outstanding bugs or critical issues.
- Usability testing and user feedback.

Milestone 7: Production Deployment and Go-Live (End of Sprint 7)

- **Expected Completion**: Sprint 7 (2 weeks)
- **Key Deliverables**:
  - Production deployment of the system on AWS (with full operational capacity).
  - Documentation for users and administrators.
  - User training sessions to ensure effective use of the system.
  - Final system handover to the business or stakeholders.

Milestone Summary:

1. **Milestone 1**: User account creation & AWS setup.
2. **Milestone 2**: Asset type & inventory setup with QR code generation and multi-scan capability.
3. **Milestone 3**: Full asset management module.
4. **Milestone 4**: Task creation, Reporting & lifecycle management.
5. **Milestone 5**: Notifications & user roles.
6. **Milestone 6**: Final testing & optimization.
7. **Milestone 7**: Production deployment & go-live.

Each milestone is aligned with a sprint, ensuring the Agile framework is followed and providing flexibility to adapt or adjust as needed between iterations.

This high-level plan provides the structure for executing the asset management project using Agile methodology, ensuring flexibility, regular feedback, and continuous improvements throughout the process.