Owen Lindsey

Professor Sparks, James

09/28/2024

CST-391

Milestone 2

Aircraft Maintenance Management Application Instructor Feedback:

Milestone 1 feedback:

1) You do not document how users are created, managed and their role in the application (login?). The category Users only appears in the database

2) On the dashboard you show 'Flight Time' but it is not in the database or in the UI.

3) Watch out for misspellings and typos. In Actions, you have 'veiw' buttons.

4) Inconsistent capitalization: You have a 'Maintenance page for aircraft 4'

Overall, good work and an interesting project idea.

Fix the issues mentioned in the 'refined submission'. Points taken will increase on the refined proposal on matters discussed here that you did not fix.

Aircraft Maintenance Management Application:

Application description:

This application will allow an aircraft maintenance organization manager their fleet. The system provides the following key features;

1. **Fleet Overview:**
   - *Display a comprehensive list of all aircraft under the organization's responsibility.*
   - *Show each aircraft the most recent maintenance and important details.*

2. **Detailed Maintenance History:**
   - *Access a complete maintenance history for each aircraft*
   - *View all maintenance records, including date, details, and responsible technician.*
   - *Edit existing maintenance entries or add new ones as needed.*

3. **Performance Metrics Dashboard:**
   - *Visualize key performance indicators using various graph types.*
   - *Monitor and analyze metrics such as flight hours, fuel efficiency, and maintenance frequency.*
   - *Input new performance data and generate reports on total flight times, average oil consumption, and other relevant statistics.*

4. **Role-Based Access Control:**
   - *Implement different access levels for Admin and User roles to ensure data security and appropriate feature access.*

5. **User Roles and Privileges:**

   - *The application implements a role-based access control system with two primary key roles: Admin and User.*

6. **Admin Privileges:**

   - *Full access to all features of the aircraft maintenance system.*
   - *Ability to add, modify, or delete user accounts.*
   - *Permission to make critical changes to aircraft data or maintenance records.*
   - *Access to system-wide analytics and reports.*
   - *Capability to manage and assign roles to other users.*

7. **User Privileges:**

   - *View aircraft maintenance records and performance metrics.*
   - *Update non-critical information.*
   - *Access basic analytics and reports.*
   - *Perform routine data entry and updates within their assigned scope.*

Aircraft Maintenance Management Application:

Functionality Requirements (User Story description):

1. **Aircraft Performance Tracking**

- *As a User, I want to add and update aircraft performance metrics so I can keep track of each aircraft's operational efficiency.*

- As an Admin, I want to manage all performance tracking data across the fleet.

2. **Maintenance Record Management:**

- *As a User, I want to select an aircraft and update its maintenance information so that I can keep maintenance records current and accurate.*

- *As an Admin, I want to review, approve, and make critical changes to records when necessary.*

3. **Fleet Expansion:**

- *As an Admin, I want to add new aircraft to our hangar spaces whenever a new aircraft is acquired or arrives for maintenance, so that our fleet inventory stays up to date.*

4. **Visual Analytics:**

- *As a User, I want to view and interact with graphs that visually display maintenance performance metrics (such as maintenance downtime, flight hours, and engine hours) so that I can quickly assess the status and trends of the fleet.*

- *As an Admin, I want access to more detailed and system-wide analytics for fleet management.*

Aircraft Maintenance Management Application:

Functionality Requirements (User Story description):

5. **Data Entry and Editing:**

- *As a User, I want to input new data and edit existing information maintenance records and performance metrics to ensure our database remains accurate and current.*

- *As an Admin, I want to have override capabilities for all data entries and the ability to manage data input and permissions.*

6. **Reporting**

- *As a User, I want to generate basic reports on aircraft performance and maintenance history to support regular operations.*

- *As an Admin, I want to create and access reports across the entire fleet.*

7. **User Management:**

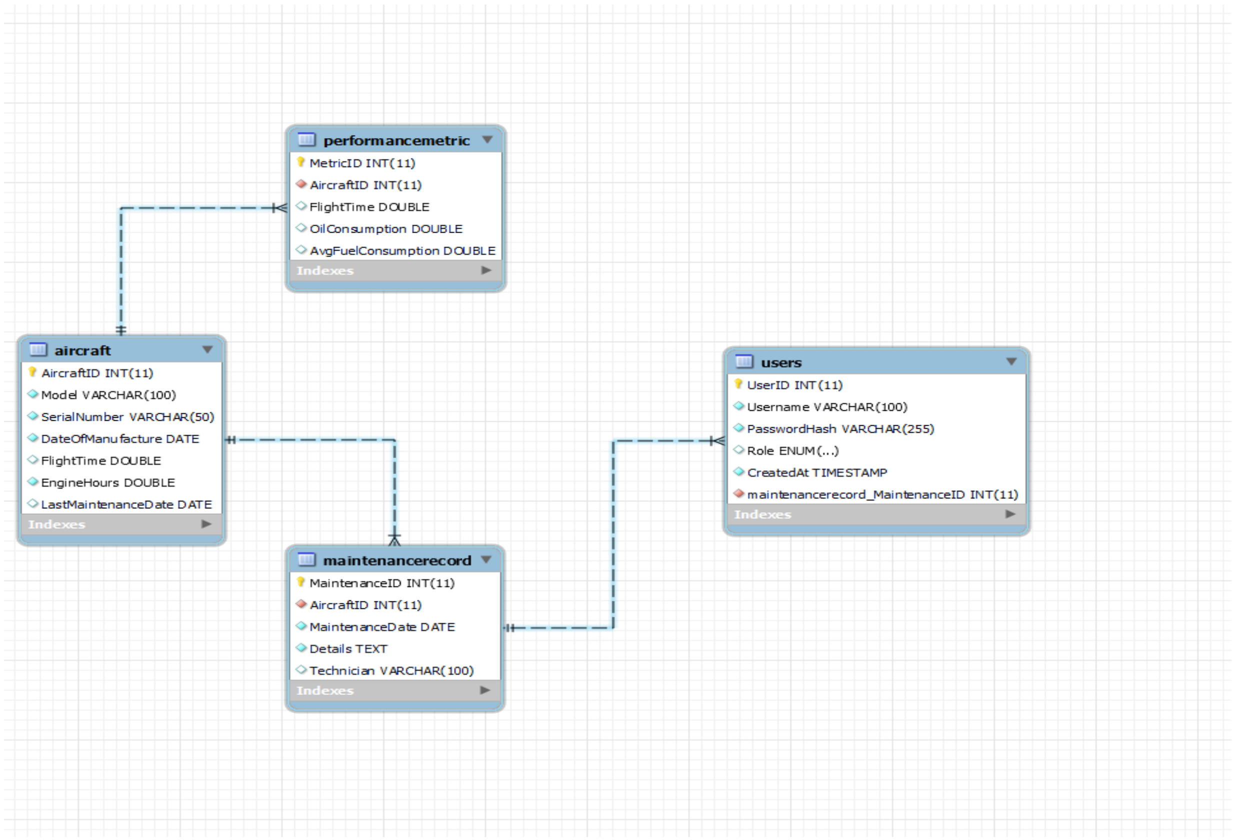
- *As an Admin, I want to create, modify, and delete user accounts, and assign appropriate roles to ensure proper access control within the system.*

8. **Authentication and Authorization:**

- *As a User or Admin, I want to log in securely to access my role specific features and ensure the safety of sensitive aircraft maintenance data.*

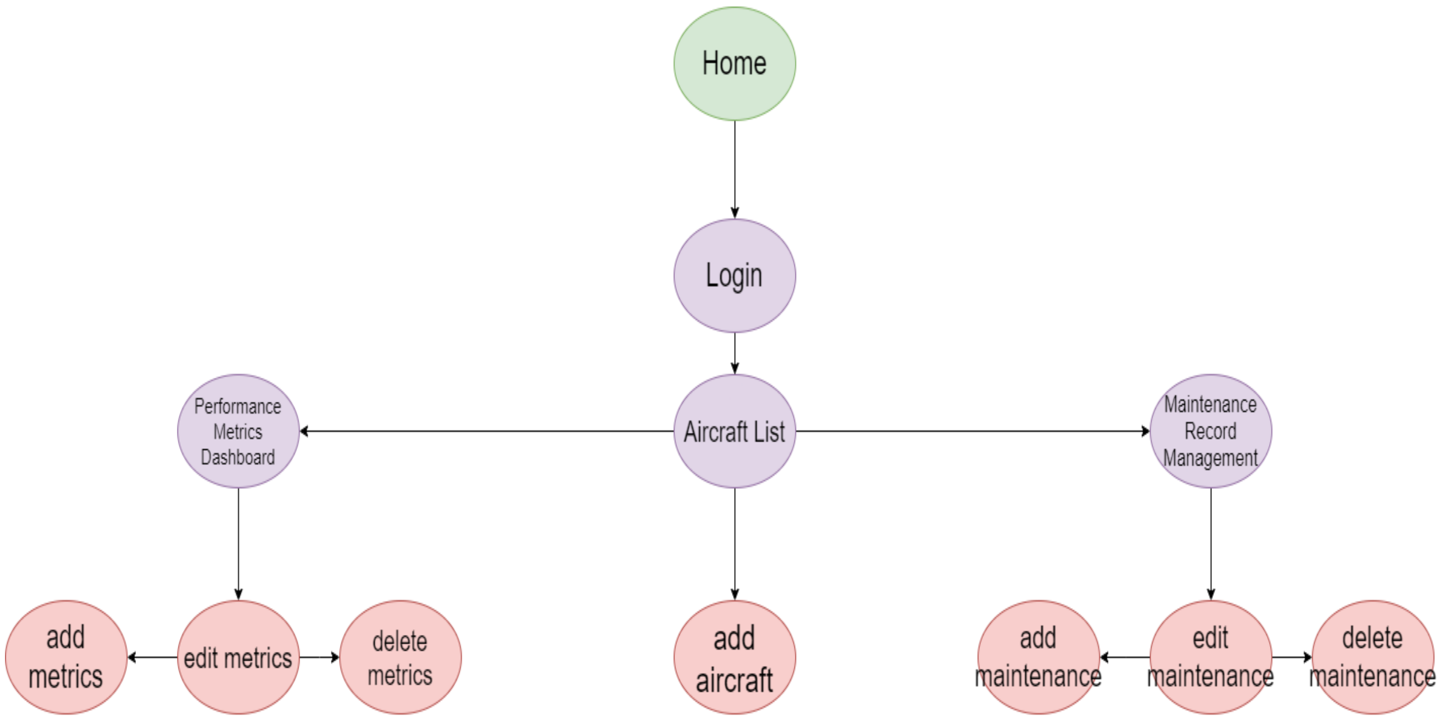
Database Design for Aircraft Maintenance Management Application:

ER diagram -



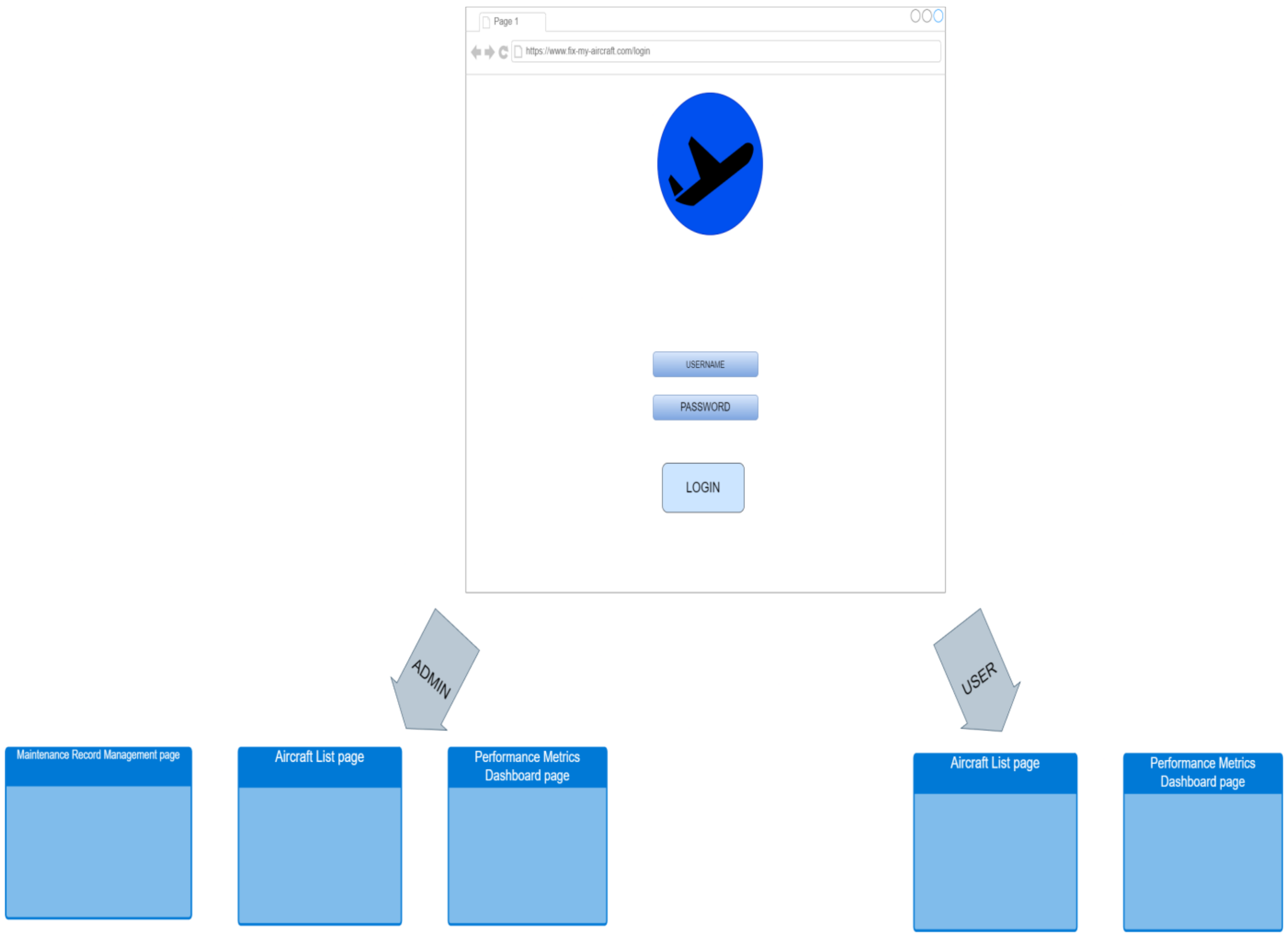
Sitemap for Aircraft Maintenance Management Application:

Sitemap:



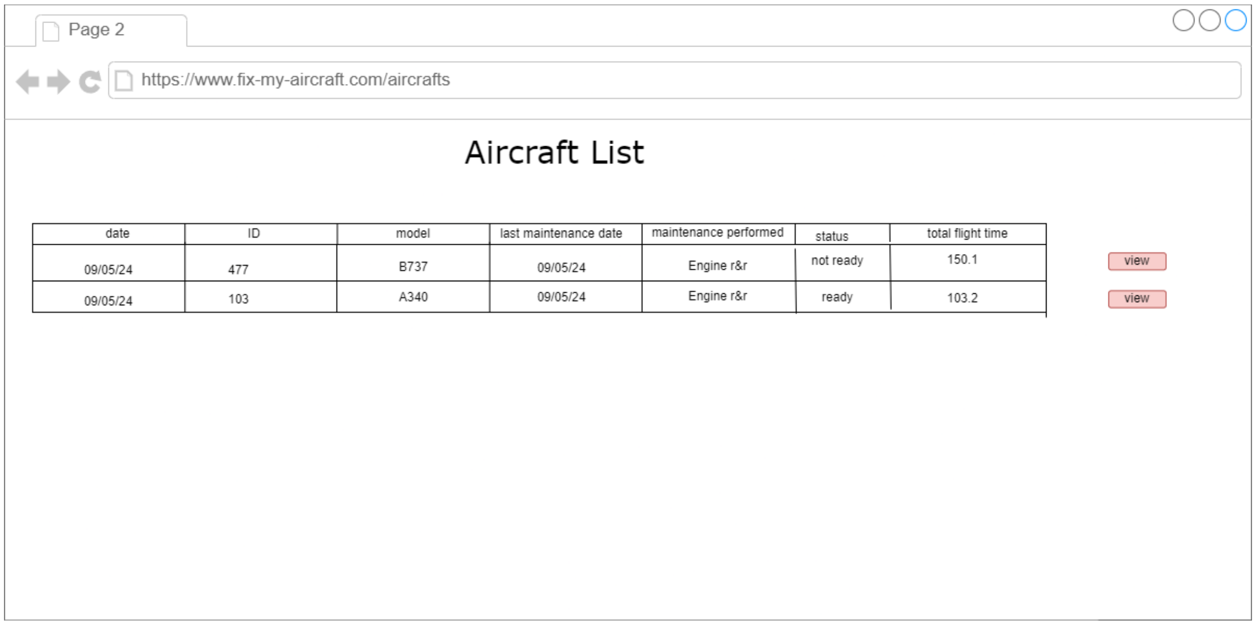
Wireframes for Aircraft Maintenance Management Application:

Page 1) Login Wireframe:


Page 1
https://www.fix-my-aircraft.com/login
USERNAME
PASSWORD
LOGIN
ADMIN
USER
Maintenance Record Management page
Aircraft List page
Performance Metrics Dashboard page
Aircraft List page
Performance Metrics Dashboard page

Wireframes for Aircraft Maintenance Management Application:

Page 2) Aircraft List Wireframe:

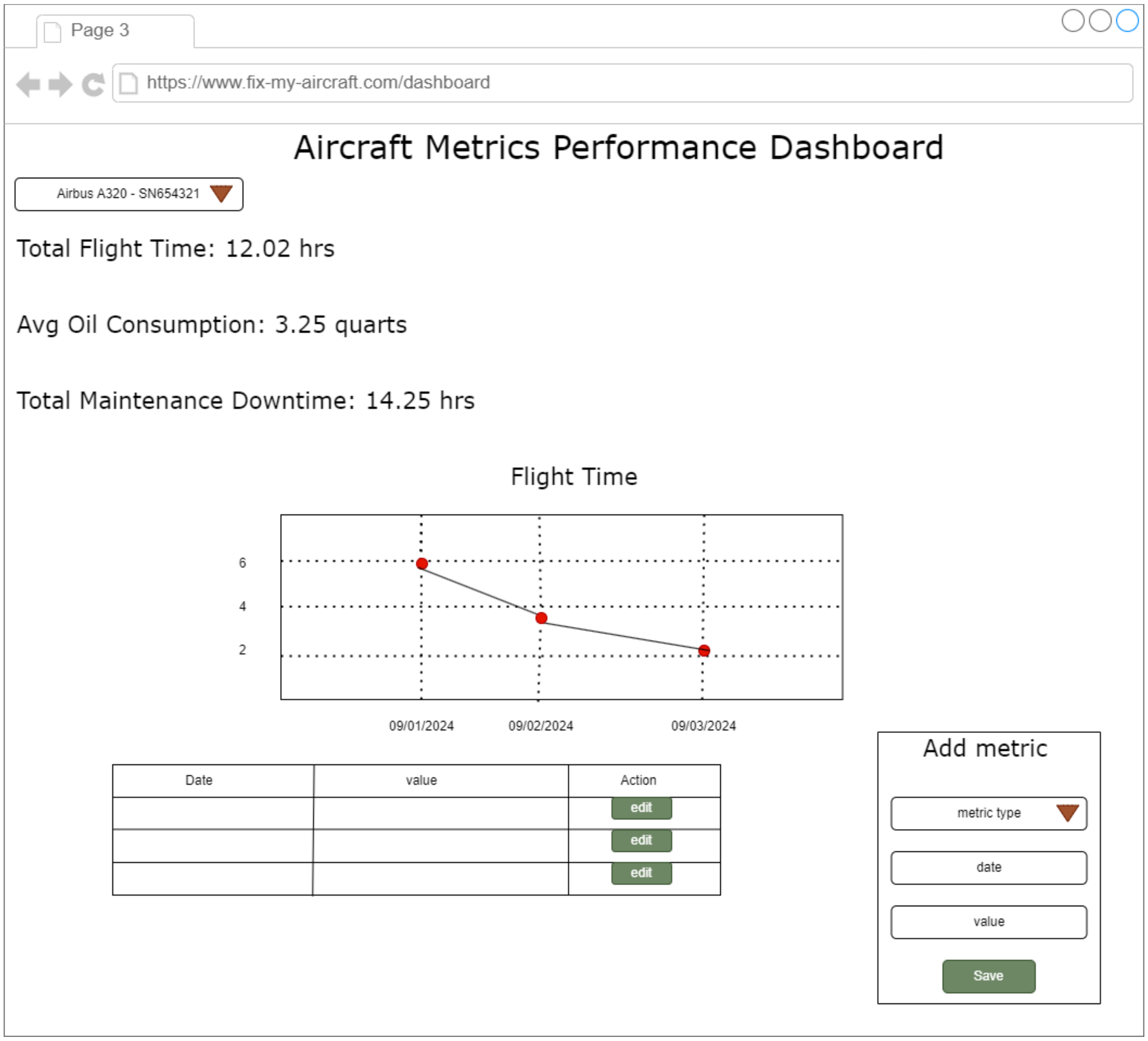


Aircraft List

date	ID	model	last maintenance date	maintenance performed	status	total flight time	
09/05/24	477	B737	09/05/24	Engine r&r	not ready	150.1	view
09/05/24	103	A340	09/05/24	Engine r&r	ready	103.2	view

Wireframes for Aircraft Maintenance Management Application:

Page 3) Aircraft Metrics Performance Dashboard Wireframe:



Aircraft Metrics Performance Dashboard

Total Flight Time: 12.02 hrs

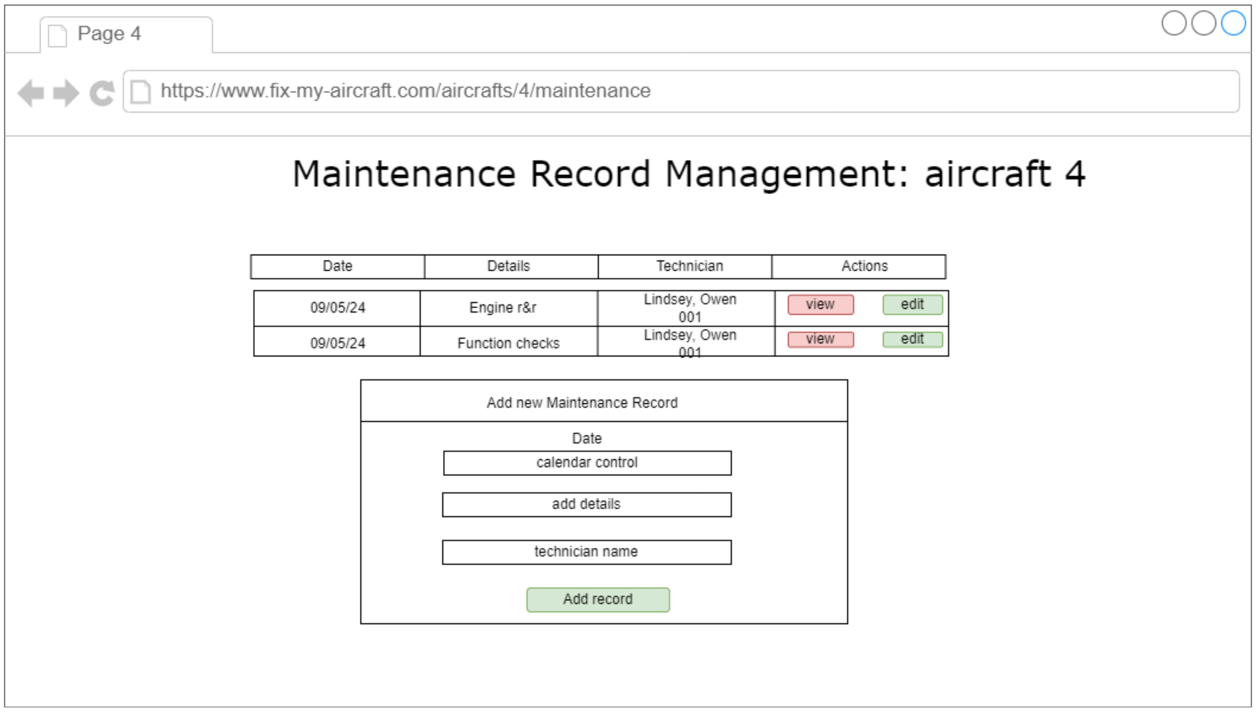
Avg Oil Consumption: 3.25 quarts

Total Maintenance Downtime: 14.25 hrs

Date	value	Action
		edit
		edit
		edit

Wireframes for Aircraft Maintenance Management Application:

Page 4) Maintenance Record Management Wireframe:



Page 4

https://www.fix-my-aircraft.com/aircrafts/4/maintenance

Maintenance Record Management: aircraft 4

Date	Details	Technician	Actions
09/05/24	Engine r&r	Lindsey, Owen 001	view edit
09/05/24	Function checks	Lindsey, Owen 001	view edit

Add new Maintenance Record

Date

calendar control

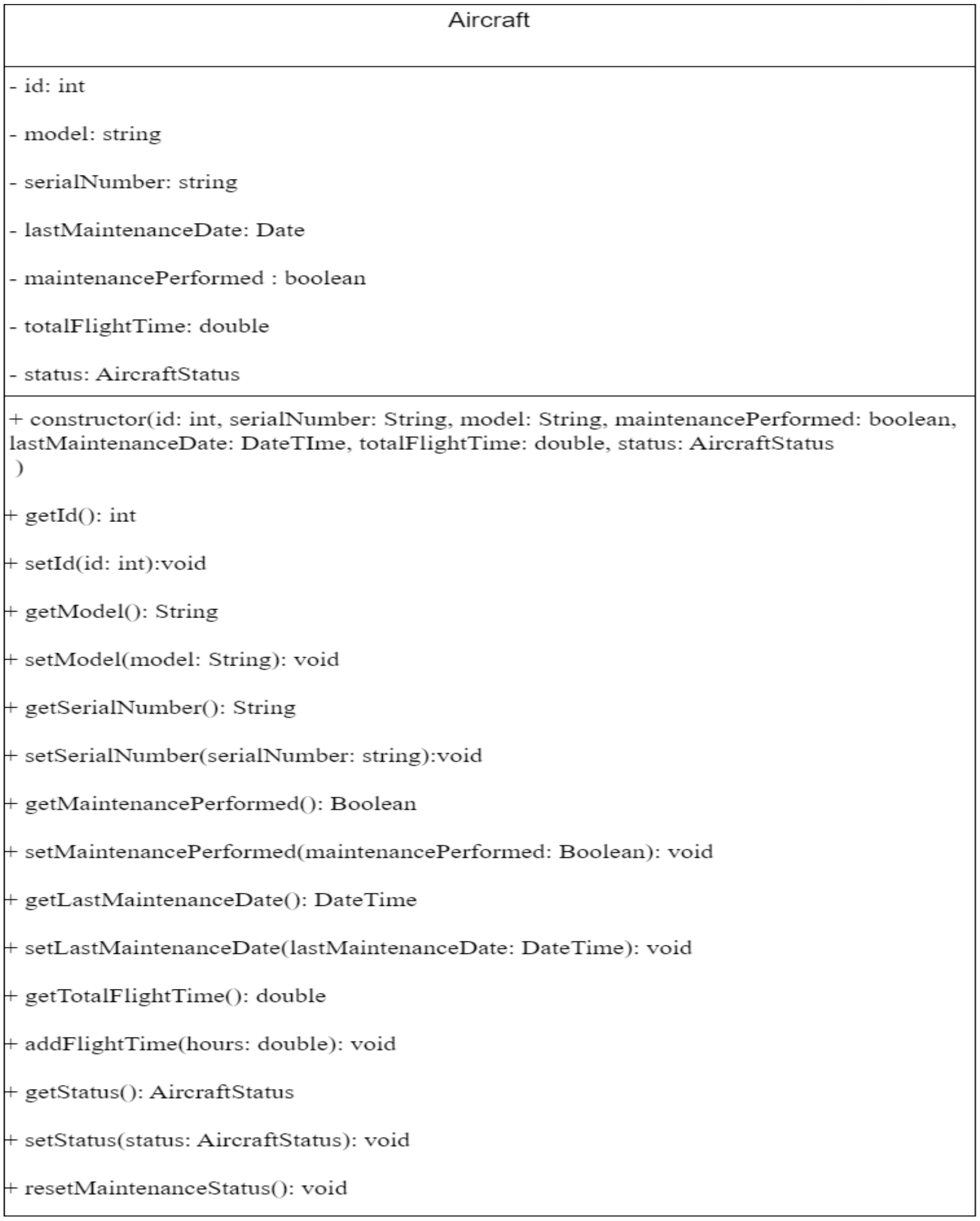
add details

technician name

Add record

UML classes for Aircraft Maintenance Management Application:

aircraft class:



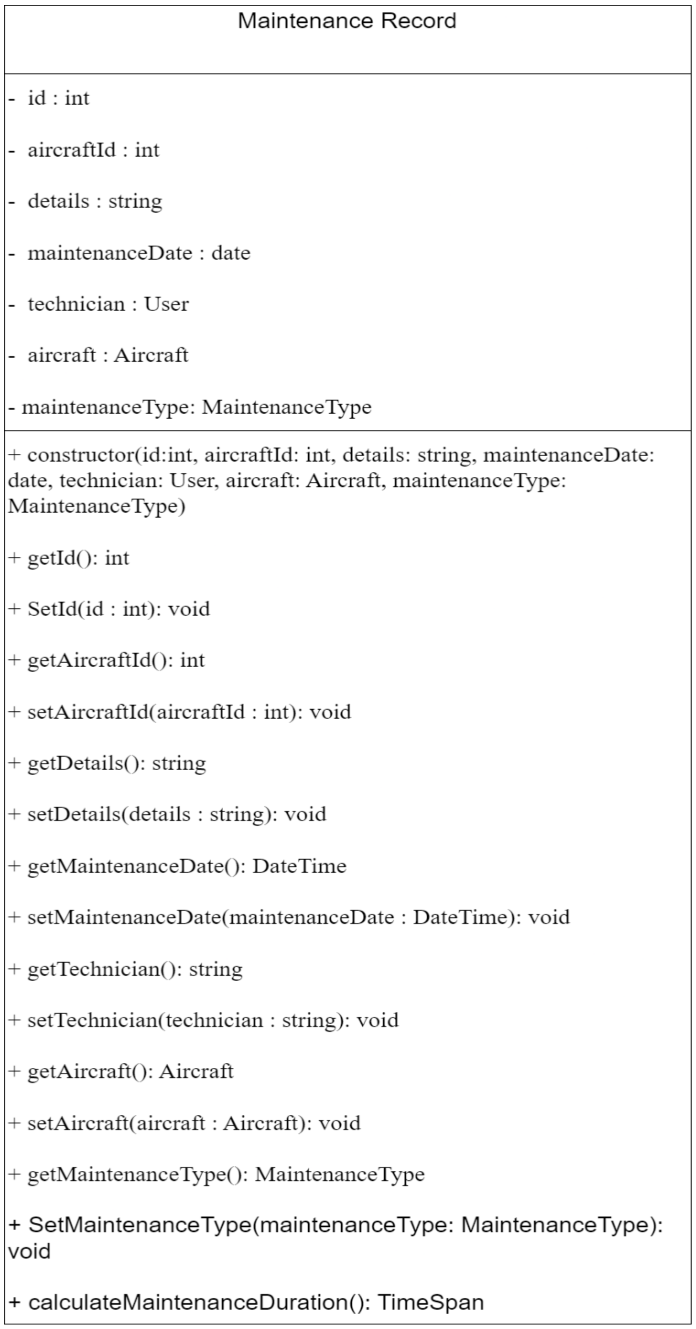
Aircraft

- id: int
- model: string
- serialNumber: string
- lastMaintenanceDate: Date
- maintenancePerformed : boolean
- totalFlightTime: double
- status: AircraftStatus

+ constructor(id: int, serialNumber: String, model: String, maintenancePerformed: boolean, lastMaintenanceDate: DateTIme, totalFlightTime: double, status: AircraftStatus)
+ getId(): int
+ setId(id: int):void
+ getModel(): String
+ setModel(model: String): void
+ getSerialNumber(): String
+ setSerialNumber(serialNumber: string):void
+ getMaintenancePerformed(): Boolean
+ setMaintenancePerformed(maintenancePerformed: Boolean): void
+ getLastMaintenanceDate(): DateTime
+ setLastMaintenanceDate(lastMaintenanceDate: DateTime): void
+ getTotalFlightTime(): double
+ addFlightTime(hours: double): void
+ getStatus(): AircraftStatus
+ setStatus(status: AircraftStatus): void
+ resetMaintenanceStatus(): void

UML classes for Aircraft Maintenance Management Application:

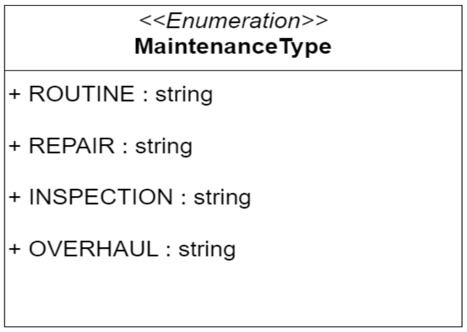
MaintenanceRecord:



Maintenance Record

- id : int
- aircraftId : int
- details : string
- maintenanceDate : date
- technician : User
- aircraft : Aircraft
- maintenanceType: MaintenanceType

+ constructor(id:int, aircraftId: int, details: string, maintenanceDate: date, technician: User, aircraft: Aircraft, maintenanceType: MaintenanceType)
+ getId(): int
+ SetId(id : int): void
+ getAircraftId(): int
+ setAircraftId(aircraftId : int): void
+ getDetails(): string
+ setDetails(details : string): void
+ getMaintenanceDate(): DateTime
+ setMaintenanceDate(maintenanceDate : DateTime): void
+ getTechnician(): string
+ setTechnician(technician : string): void
+ getAircraft(): Aircraft
+ setAircraft(aircraft : Aircraft): void
+ getMaintenanceType(): MaintenanceType
+ SetMaintenanceType(maintenanceType: MaintenanceType): void
+ calculateMaintenanceDuration(): TimeSpan

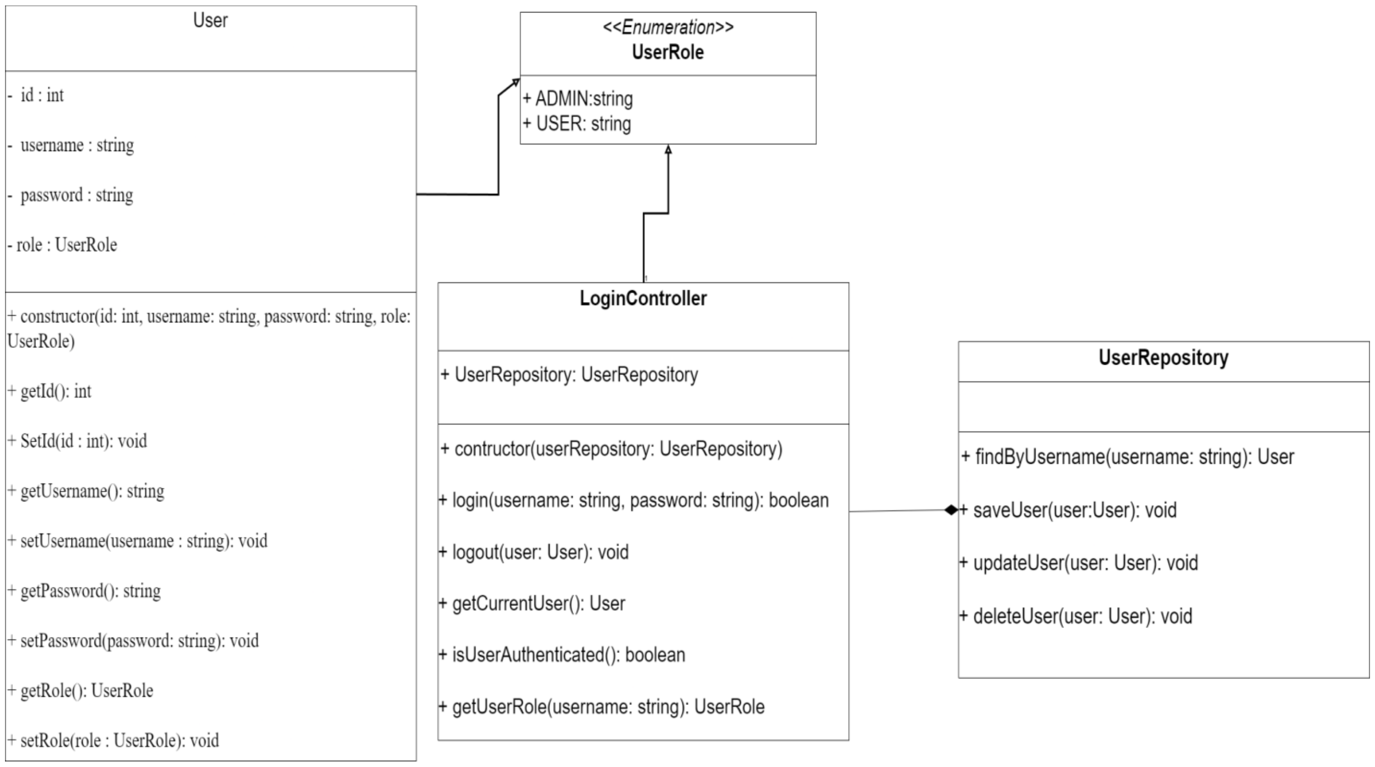


<<Enumeration>>
MaintenanceType

+ ROUTINE : string
+ REPAIR : string
+ INSPECTION : string
+ OVERHAUL : string

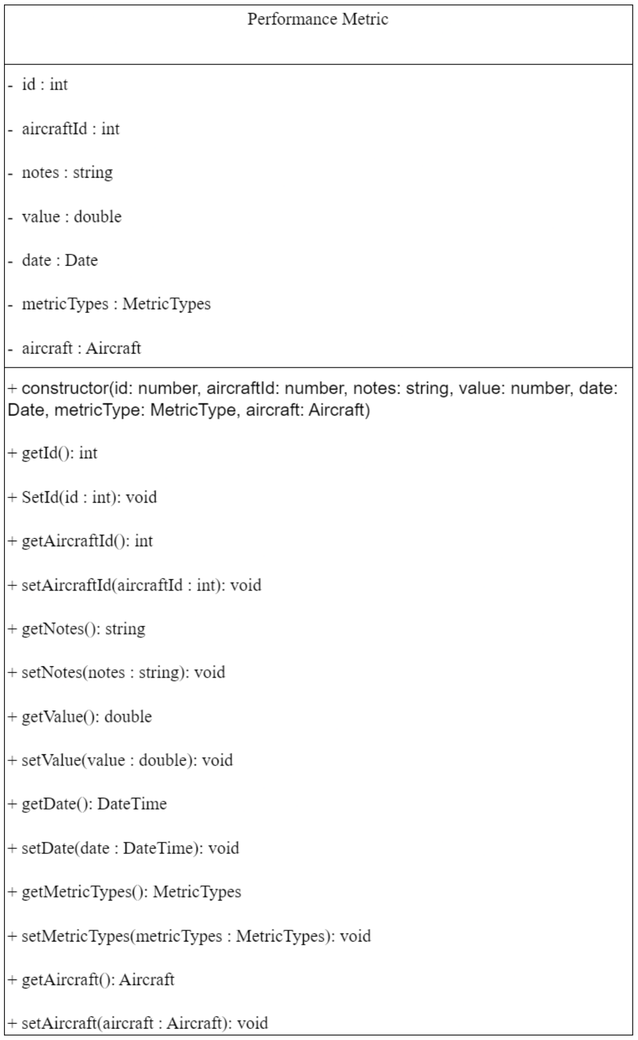
UML classes for Aircraft Maintenance Management Application:

User:

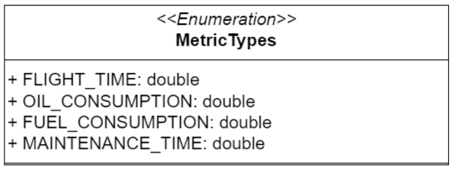

User
- id : int
- username : string
- password : string
- role : UserRole
+ constructor(id: int, username: string, password: string, role: UserRole)
+ getId(): int
+ SetId(id : int): void
+ getUsername(): string
+ setUsername(username : string): void
+ getPassword(): string
+ setPassword(password: string): void
+ getRole(): UserRole
+ setRole(role : UserRole): void
<<Enumeration>>
UserRole
+ ADMIN:string
+ USER: string
LoginController
+ UserRepository: UserRepository
+ contructor(userRepository: UserRepository)
+ login(username: string, password: string): boolean
+ logout(user: User): void
+ getCurrentUser(): User
+ isUserAuthenticated(): boolean
+ getUserRole(username: string): UserRole
UserRepository
+ findByUsername(username: string): User
+ saveUser(user:User): void
+ updateUser(user: User): void
+ deleteUser(user: User): void

UML classes for Aircraft Maintenance Management Application:

PerformanceMetric:



Performance Metric
- id : int
- aircraftId : int
- notes : string
- value : double
- date : Date
- metricTypes : MetricTypes
- aircraft : Aircraft
+ constructor(id: number, aircraftId: number, notes: string, value: number, date: Date, metricType: MetricType, aircraft: Aircraft)
+ getId(): int
+ SetId(id : int): void
+ getAircraftId(): int
+ setAircraftId(aircraftId : int): void
+ getNotes(): string
+ setNotes(notes : string): void
+ getValue(): double
+ setValue(value : double): void
+ getDate(): DateTime
+ setDate(date : DateTime): void
+ getMetricTypes(): MetricTypes
+ setMetricTypes(metricTypes : MetricTypes): void
+ getAircraft(): Aircraft
+ setAircraft(aircraft : Aircraft): void



<<Enumeration>>
MetricTypes
+ FLIGHT_TIME: double
+ OIL_CONSUMPTION: double
+ FUEL_CONSUMPTION: double
+ MAINTENANCE_TIME: double

Risks of Aircraft Maintenance Management Application:

1. Security Risks:

1.1 User Authentication and Authorization:

Risk:

- *Inadequate user validation and oversight in permission management.*

Impact:

- *Unauthorized access to sensitive aircraft data within the application.*

Mitigation:

- *Implement a simple but effective role-based access control system.*

- Thoroughly test user permissions for both admin and regular user roles.

Risks of Aircraft Maintenance Management Application:

1. Security Risks:

1.2 Input Validation:

Risk:

- *Vulnerability to basic injection attacks due to improper input handling*.

Impact:

- Potential for data corruption or unauthorized access.

Mitigation:

- *Implement basic input validation and sanitization for all form fields*.
- Use prepared statements for database queries to prevent SQL injection attacks.

Risks of Aircraft Maintenance Management Application:

2. Data Management:

2.1 Data Integrity:

Risk:

- *Incorrect data entry or manipulation in maintenance records.*

Impact:

- Inaccurate reporting and potential logical errors in the application.

Mitigation:

- *Add basic data validation rules to the user interface.*
- *Implement simple error checking for critical fields (date formats, numeric ranges).*

Risks of Aircraft Maintenance Management Application:

3. Project Management:

3.1 Scope and Time Managment:

Risk:

- *Project scope could become too ambitious for the class time frame.*

Impact:

- Incomplete features or rushed implementations.

Mitigation:

- *Clearly define core features required for the assignment.*
- *Prioritize functionality over optimization initially.*
- *Keep up with an organized timeline.*

Risks of Aircraft Maintenance Management Application:

4. Performance Considerations:

4.1 Basic Application Performance:

Risk:

- *Inefficient code leading to slow performance, especially for data intensive operations*.

Impact:

- Poor user experience and potential issues during project demonstrations.

Mitigation:

- *Focus on writing clean, efficient code for core functionalities.*

- *If time allows, implement basic optimization for data retrieval and display.*

Aircraft Maintenance Management Application API:

1. Aircraft endpoints:

- ***GET:*** */aircrafts*
- ***GET:*** */aircraft/{id}*
- ***POST:*** */aircrafts*
- ***PUT:*** */aircrafts/{id}*
- ***PUT:*** ***/****aircrafts/{id}*
- ***DELETE:*** */aircrafts/{id}*

2. Maintenance endpoints:

- ***GET:*** */aircrafts/{id}/maintenances*
- ***GET:*** */aircrafts/{id}/maintenances/{maintenanceId}*
- ***POST:*** */maintenance*
- ***PUT:*** */aircrafts/{id}/maintenances/{maintenanceId}*
- ***DELETE :*** */aircrafts/{id}/maintenances/{maintenanceId}*

Aircraft Maintenance Management Application API:

3. Metrics endpoints:

- ***GET:*** */aircrafts/{id}/metrics*
- ***GET:*** */aircrafts/{id}/metrics/{metricId}*
- ***POST:*** */aircrafts/{id}/metrics*
- ***PUT:*** */aircrafts/{id}/metrics{metricId}*
- ***DELETE:*** */aircrafts/{id}/metrics/{metricId}*

4. User endpoints:

- ***GET:*** */users/*
- ***GET:*** */users/{id}*
- ***POST:*** */users*
- ***PUT:*** */users/{id}*

- ***DELETE:*** */users/{id}*

5. Authentication endpoints:

- ***POST:*** */auth/login*

- ***POST:*** */auth/logout*