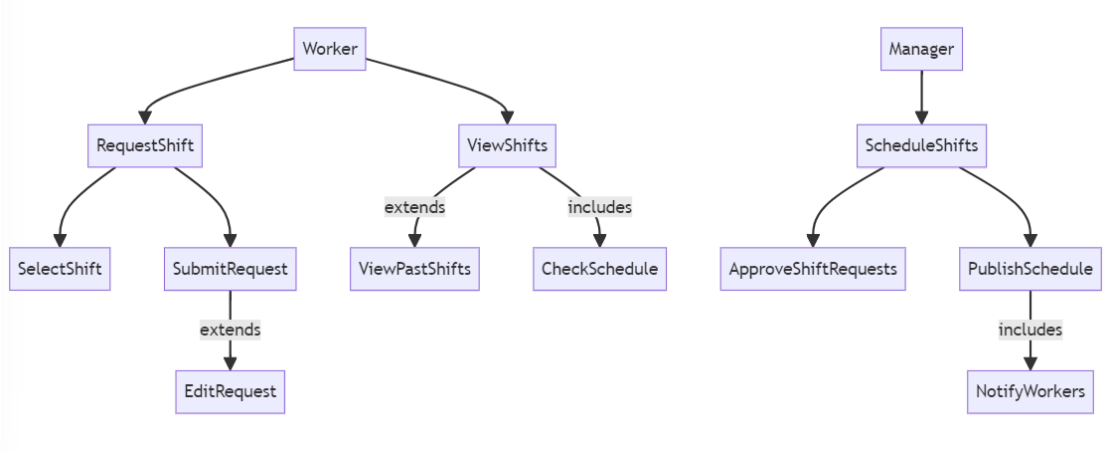
This document provides a comprehensive representation of EasyShifts using Unified Modeling Language (UML) diagrams.

1. **Use Case Diagram:**
    - Presents a high-level overview of the project's functionalities from a user's perspective, identifying key actors and their interactions.
2. **Class Diagram:**
    - Illustrates the static structure of the system, showcasing the classes, their attributes, and relationships.
3. **Object Diagram:**
    - Provides a snapshot of the system at a specific moment, showing instances of classes and their relationships.
4. **Activity Diagram:**
    - Captures the dynamic aspects of the system by visualizing workflows and processes.
5. **Sequence Diagram:**
    - Displays the interactions between objects over time, illustrating the sequence of messages exchanged. It helps to understand the dynamic behavior of the system during specific scenarios.
6. **State Machine Diagram:**
    - Describes the different states a particular object can transition through during its lifecycle - the system's response to events.
7. **Entity-Relationship Diagram (ERD):**
    - Depicts the database schema and relationships between entities, providing a clear understanding of the data model and DB structure.

Authors

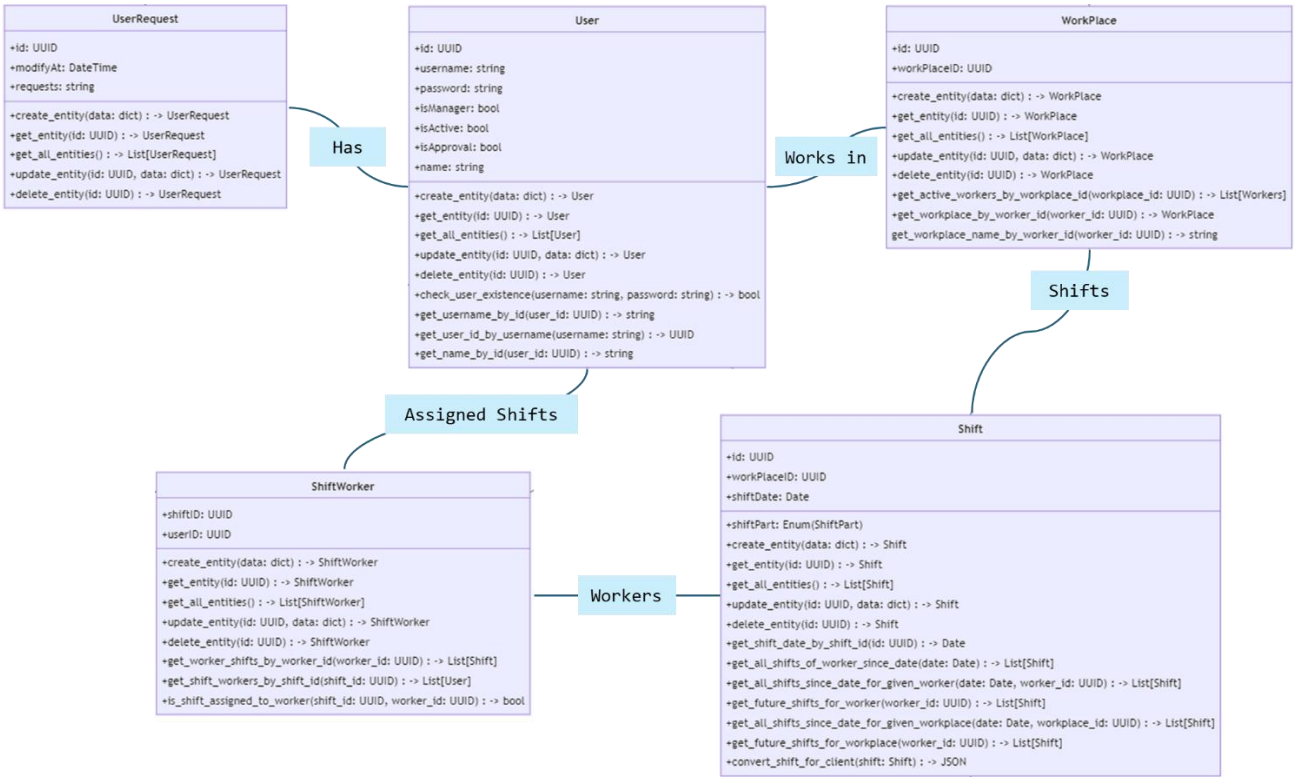
אלון משולם	207964487
אורי אקשטיין	213562069
שובל נחמיאס	211776356
נטע כהן	325195774
הלל יצחקי	322989674

Use case diagram



Mermid

```

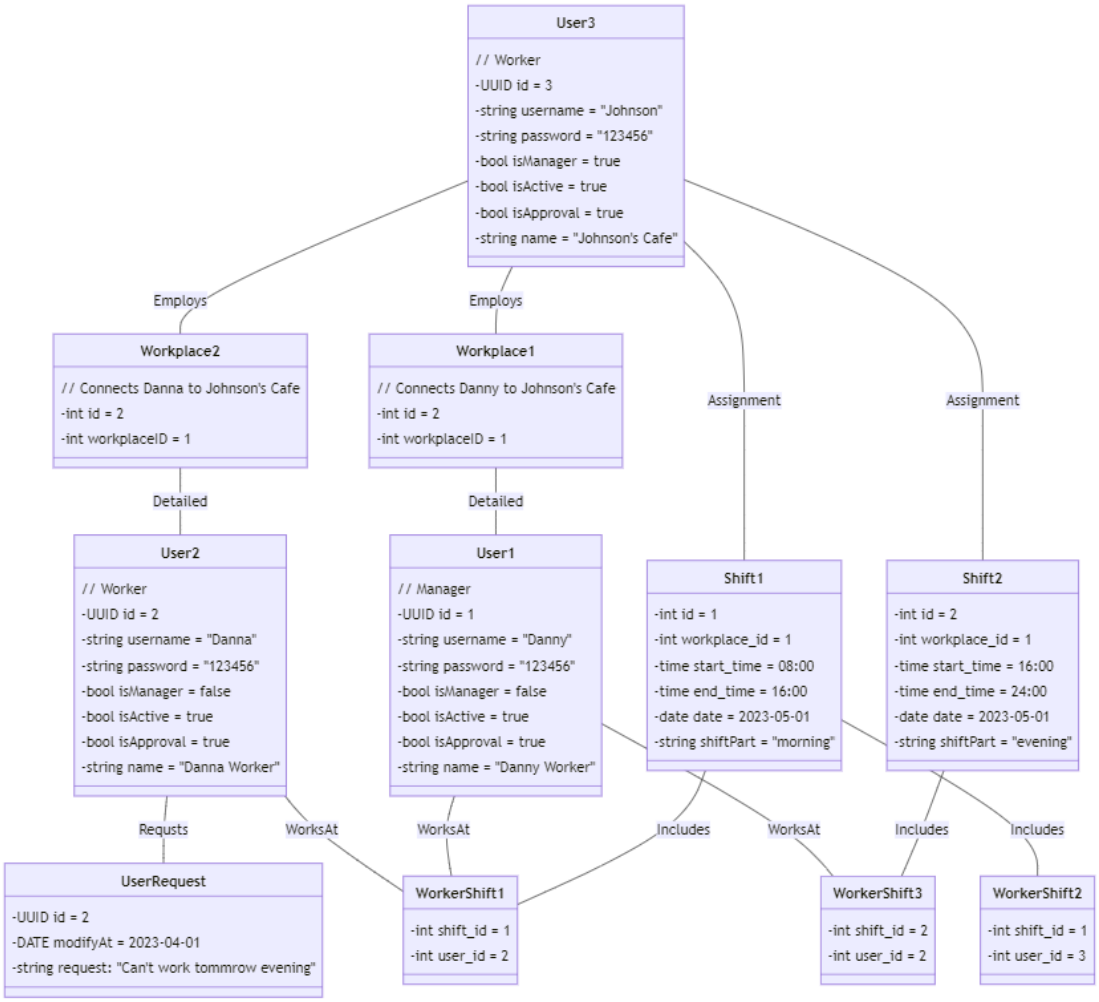
```mermaid
graph TD
    W[Worker] --> RequestShift
    W --> ViewShifts
    M[Manager] --> ScheduleShifts
    ScheduleShifts --> ApproveShiftRequests
    ScheduleShifts --> PublishSchedule
    RequestShift --> SelectShift
    RequestShift --> SubmitRequest
    ViewShifts -- extends --> ViewPastShifts
    ViewShifts -- includes --> CheckSchedule
    SubmitRequest -- extends --> EditRequest
    PublishSchedule -- includes --> NotifyWorkers
```
```

Class diagram



Mermid

```

```mermaid
classDiagram
    class User {
        +id: UUID
        +username: string
        +password: string
        +isManager: bool
        +isActive: bool
        +isApproval: bool
        +name: string
        +create_entity(data: dict) -> User
        +get_entity(id: UUID) -> User
        +get_all_entities() -> List[User]
        +update_entity(id: UUID, data: dict) -> User
        +delete_entity(id: UUID) -> User
        +check_user_existence(username: string, password: string) ->
bool
        +get_username_by_id(user_id: UUID) -> string
        +get_user_id_by_username(username: string) -> UUID
        +get_name_by_id(user_id: UUID) -> string

    }
```
```

```
    class WorkPlace {
        +id: UUID
        +workPlaceID: UUID
        +create_entity(data: dict) -> WorkPlace
        +get_entity(id: UUID) -> WorkPlace
        +get_all_entities() -> List[WorkPlace]
        +update_entity(id: UUID, data: dict) -> WorkPlace
        +delete_entity(id: UUID) -> WorkPlace
        +get_active_workers_by_workplace_id(workplace_id: UUID) ->
List[Workers]
        +get_workplace_by_worker_id(worker_id: UUID) -> WorkPlace
        get_workplace_name_by_worker_id(worker_id: UUID) -> string

    }

    class UserRequest {
        +id: UUID
        +modifyAt: DateTime
        +requests: string
        +create_entity(data: dict) -> UserRequest
        +get_entity(id: UUID) -> UserRequest
        +get_all_entities() -> List[UserRequest]
        +update_entity(id: UUID, data: dict) -> UserRequest
        +delete_entity(id: UUID) -> UserRequest
    }


    class Shift {
        +id: UUID
        +workPlaceID: UUID
        +shiftDate: Date
        +shiftPart: Enum(ShiftPart)
        +create_entity(data: dict) -> Shift
        +get_entity(id: UUID) -> Shift
        +get_all_entities() -> List[Shift]
        +update_entity(id: UUID, data: dict) -> Shift
        +delete_entity(id: UUID) -> Shift
        +get_shift_date_by_shift_id(id: UUID) -> Date
        +get_all_shifts_of_worker_since_date(date: Date) -> List[Shift]
        +get_all_shifts_since_date_for_given_worker(date: Date,
worker_id: UUID) -> List[Shift]
        +get_future_shifts_for_worker(worker_id: UUID) -> List[Shift]
        +get_all_shifts_since_date_for_given_workplace(date: Date,
workplace_id: UUID) -> List[Shift]
        +get_future_shifts_for_workplace(worker_id: UUID) ->
List[Shift]
        +convert_shift_for_client(shift: Shift) -> JSON
    }
```

```
    class ShiftWorker {
        +shiftID: UUID
        +userID: UUID
        +create_entity(data: dict) -> ShiftWorker
        +get_entity(id: UUID) -> ShiftWorker
        +get_all_entities() -> List[ShiftWorker]
        +update_entity(id: UUID, data: dict) -> ShiftWorker
        +delete_entity(id: UUID) -> ShiftWorker
        +get_worker_shifts_by_worker_id(worker_id: UUID) -> List[Shift]
        +get_shift_workers_by_shift_id(shift_id: UUID) -> List[User]
        +is_shift_assigned_to_worker(shift_id: UUID, worker_id: UUID) -
> bool
    }

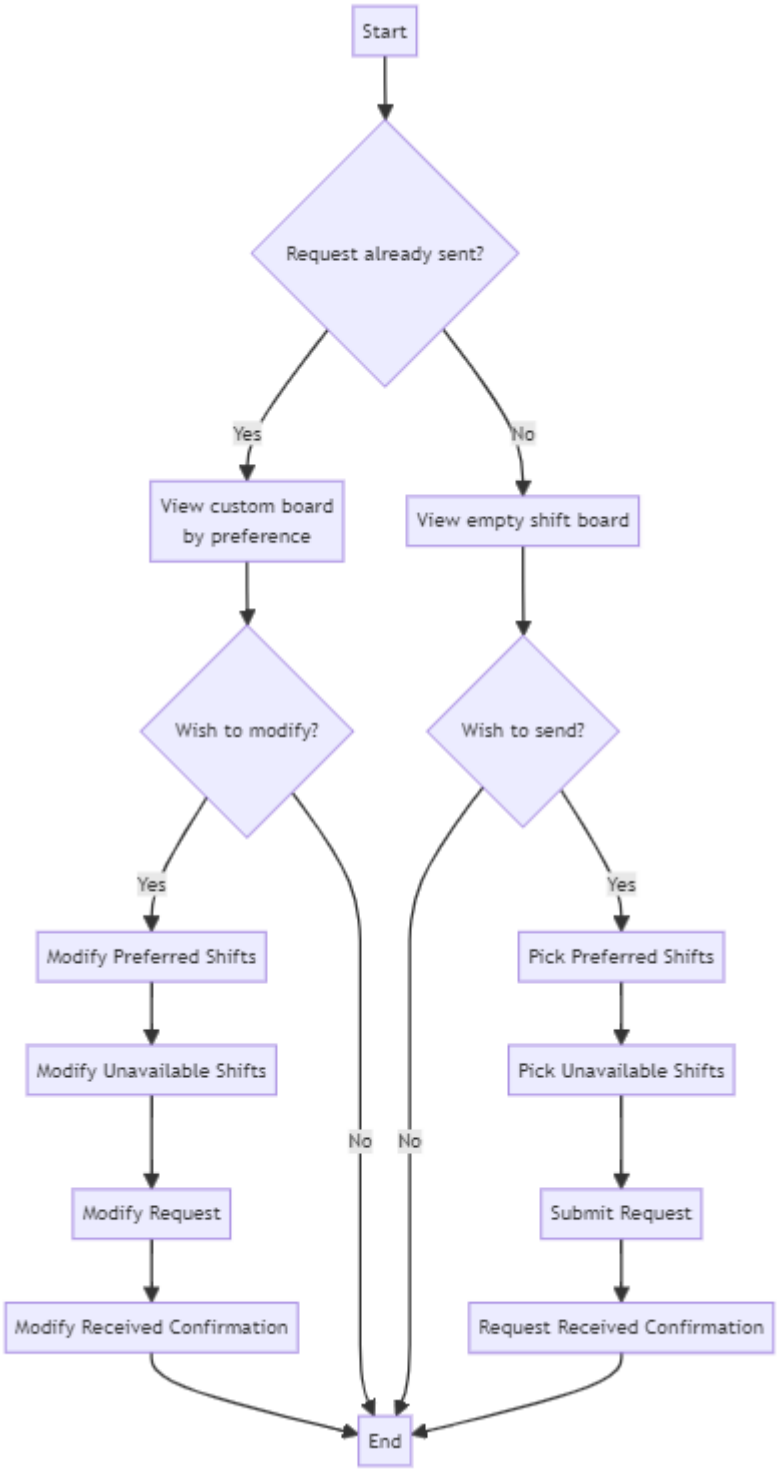

    User "1" -- "0..1" WorkPlace : WorksIn
    Shift "1" -- "*" ShiftWorker : Workers
    User "1" -- "0..*" UserRequest : Has
    WorkPlace "1" -- "*" Shift : Shifts
    User "1" -- "0..*" ShiftWorker : AssignedShifts
```

Object diagram



Mermid:

```

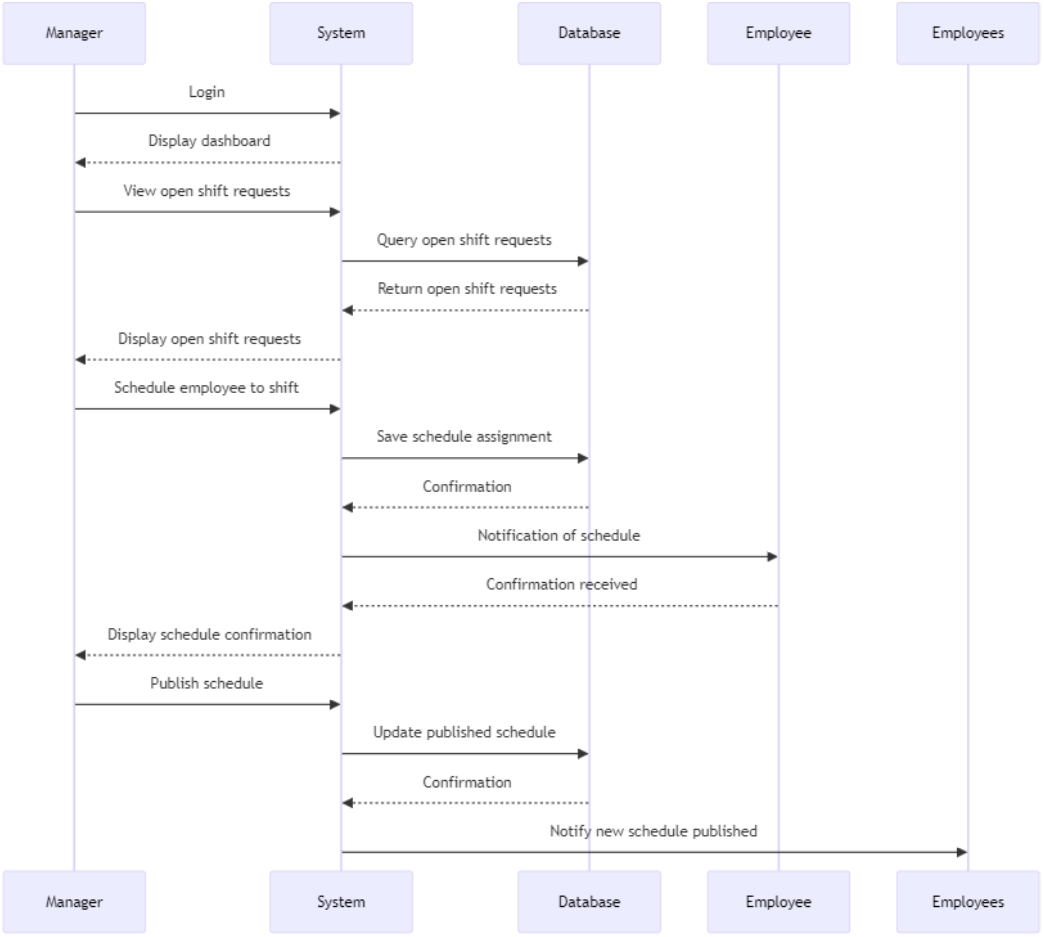
```
classDiagram
    User3 -- Workplace1: Employs
    User3 -- Workplace2: Employs
    Workplace1 -- User1: Detailed
    Workplace2 -- User2: Detailed
    User2 -- UserRequest: Requsts
    User3 -- Shift1: Assignment
    User3 -- Shift2: Assignment
    Shift1 -- WorkerShift1: Includes
    Shift1 -- WorkerShift2: Includes
    Shift2 -- WorkerShift3: Includes
    User1 -- WorkerShift1: WorksAt
    User2 -- WorkerShift1: WorksAt
    User1 -- WorkerShift3: WorksAt

    class User1 {
```
```

```
    // Manager
    -UUID id = 1
    -string username = "Danny"
    -string password = "123456"
    -bool isManager = false
    -bool isActive = true
    -bool isApproval = true
    -string name = "Danny Worker"
}
class User2 {
    // Worker
    -UUID id = 2
    -string username = "Danna"
    -string password = "123456"
    -bool isManager = false
    -bool isActive = true
    -bool isApproval = true
    -string name = "Danna Worker"
}
class User3 {
    // Worker
    -UUID id = 3
    -string username = "Johnson"
    -string password = "123456"
    -bool isManager = true
    -bool isActive = true
    -bool isApproval = true
    -string name = "Johnson's Cafe"
}


class Workplace1 {
    // Connects Danny to Johnson's Cafe
    -int id = 2
    -int workplaceID = 1
}
class Workplace2 {
    // Connects Danna to Johnson's Cafe
    -int id = 2
    -int workplaceID = 1
}

class Shift1 {
    -int id = 1
    -int workplace_id = 1
    -time start_time = 08:00
    -time end_time = 16:00
    -date date = 2023-05-01
    -string shiftPart = "morning"
```

```
    }
    class Shift2 {
        -int id = 2
        -int workplace_id = 1
        -time start_time = 16:00
        -time end_time = 24:00
        -date date = 2023-05-01
        -string shiftPart = "evening"
    }

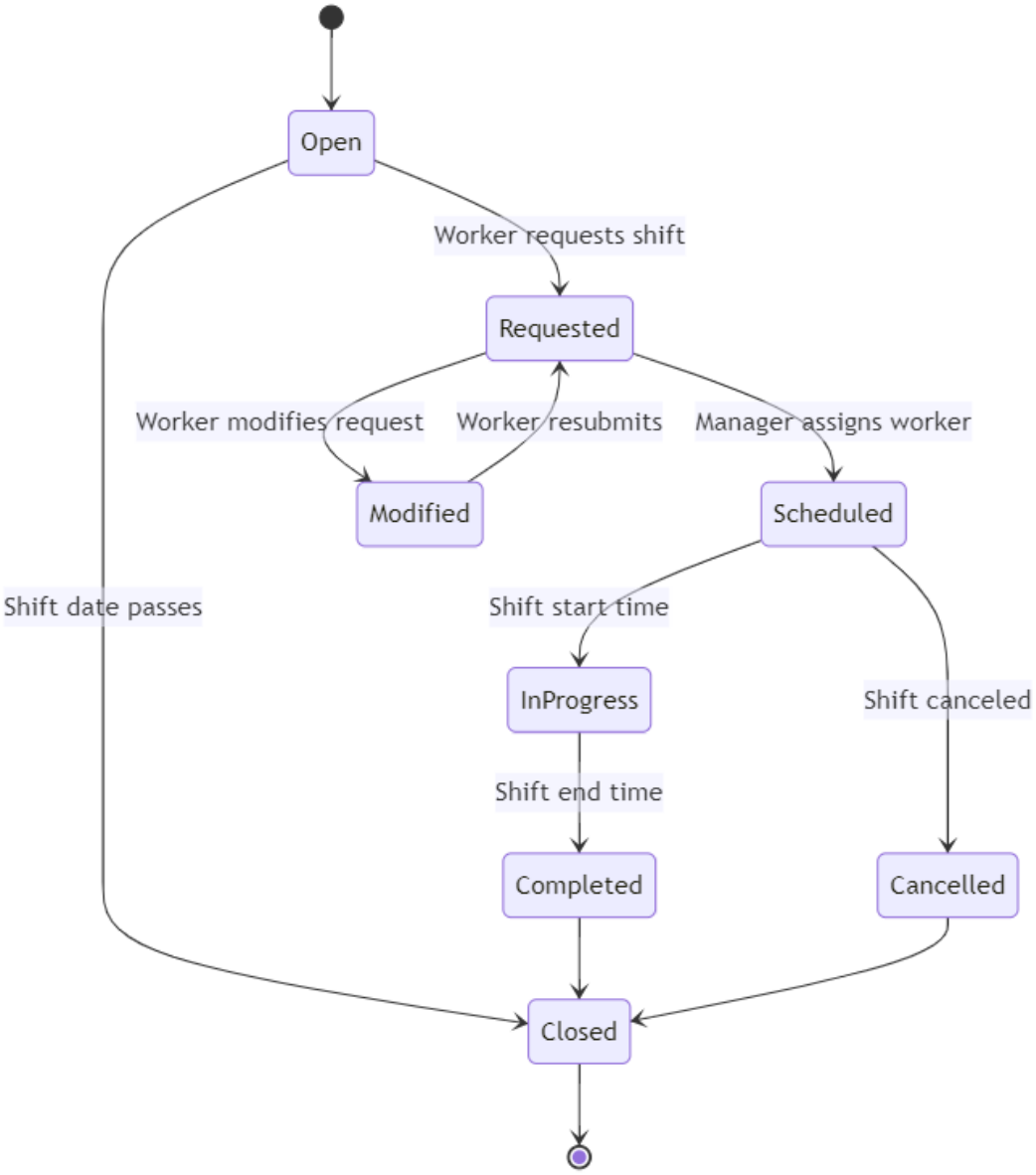
    class WorkerShift1 {
        -int shift_id = 1
        -int user_id = 2
    }
    class WorkerShift2 {
        -int shift_id = 1
        -int user_id = 3
    }
    class WorkerShift3 {
        -int shift_id = 2
        -int user_id = 2
    }

    class UserRequest {
        -UUID id = 2
        -DATE modifyAt = 2023-04-01
        -string request: "Can't work tommrow evening"
    }
```

Activity diagram



Mermid

```

```
graph TD
    A[Start] --> B{Request already sent?}
    B -- Yes --> C[View custom board\nby preference]
    C --> D{Wish to modify?}
```
```

```
    D -- Yes --> E[Modify Preferred Shifts]
    E --> F[Modify Unavailable Shifts]
    F --> G[Modify Request]
    G --> H[Modify Received Confirmation]
    D -- No --> I[End]
    B -- No --> J[View empty shift board]
    J --> K{Wish to send?}
    K -- No --> I
    K -- Yes --> L[Pick Preferred Shifts]
    L --> M[Pick Unavailable Shifts]
    M --> N[Submit Request]
    N --> O[Request Received Confirmation]
    O --> I
    H --> I
```

Sequence diagram



Mermid

```

```mermaid
sequenceDiagram
    Manager->>System: Login
    System-->>Manager: Display dashboard
    Manager->>System: View open shift requests
    System->>Database: Query open shift requests
    Database-->>System: Return open shift requests
    System-->>Manager: Display open shift requests
    Manager->>System: Schedule employee to shift
    System->>Database: Save schedule assignment
    Database-->>System: Confirmation
    System->>Employee: Notification of schedule
    Employee-->>System: Confirmation received
    System-->>Manager: Display schedule confirmation
    Manager->>System: Publish schedule
    System->>Database: Update published schedule
    Database-->>System: Confirmation
    System->>Employees: Notify new schedule published
```
```

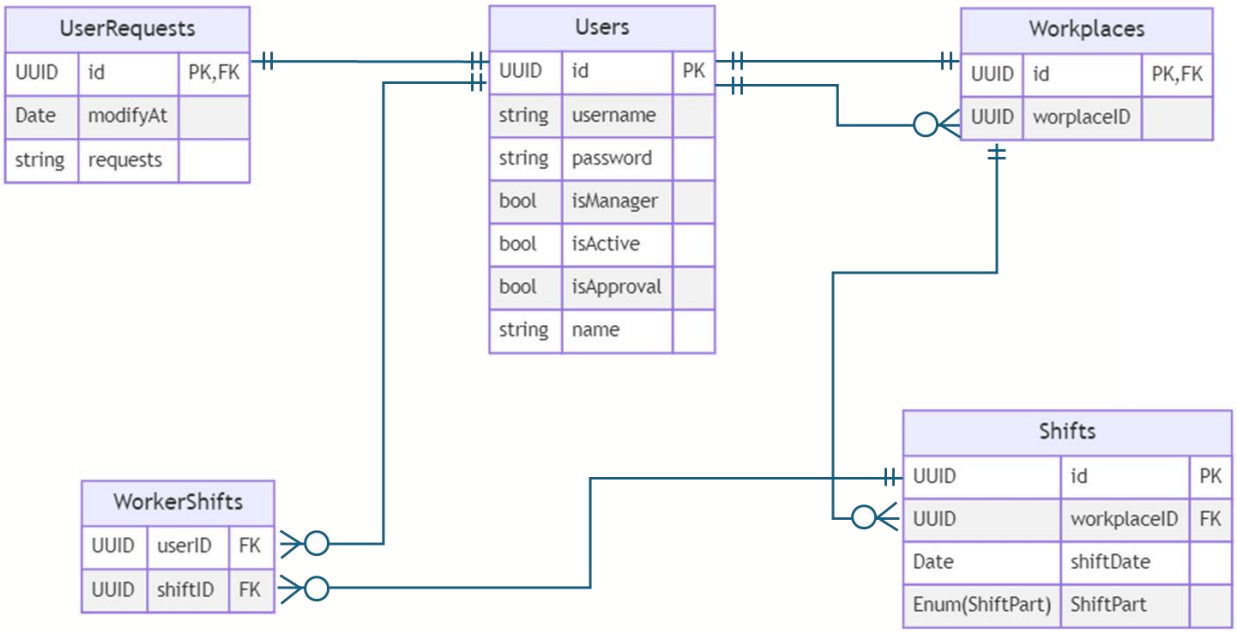
State machine diagram



Mermid

```
stateDiagram
[*]--> Open
Open --> Requested : Worker requests shift
Open --> Closed : Shift date passes
Requested --> Scheduled : Manager assigns worker
Requested --> Modified : Worker modifies request
Modified --> Requested : Worker resubmits
Scheduled --> InProgress : Shift start time
InProgress --> Completed : Shift end time
Completed --> Closed
Closed --> [*]
Scheduled --> Cancelled : Shift canceled
Cancelled --> Closed
```

ERD


UserRequests
UUID id PK,FK
Date modifyAt
string requests
Users
UUID id PK
string username
string password
bool isManager
bool isActive
bool isApproval
string name
Workplaces
UUID id PK,FK
UUID worplaceID
WorkerShifts
UUID userID FK
UUID shiftID FK
Shifts
UUID id PK
UUID workplaceID FK
Date shiftDate
Enum(ShiftPart) ShiftPart