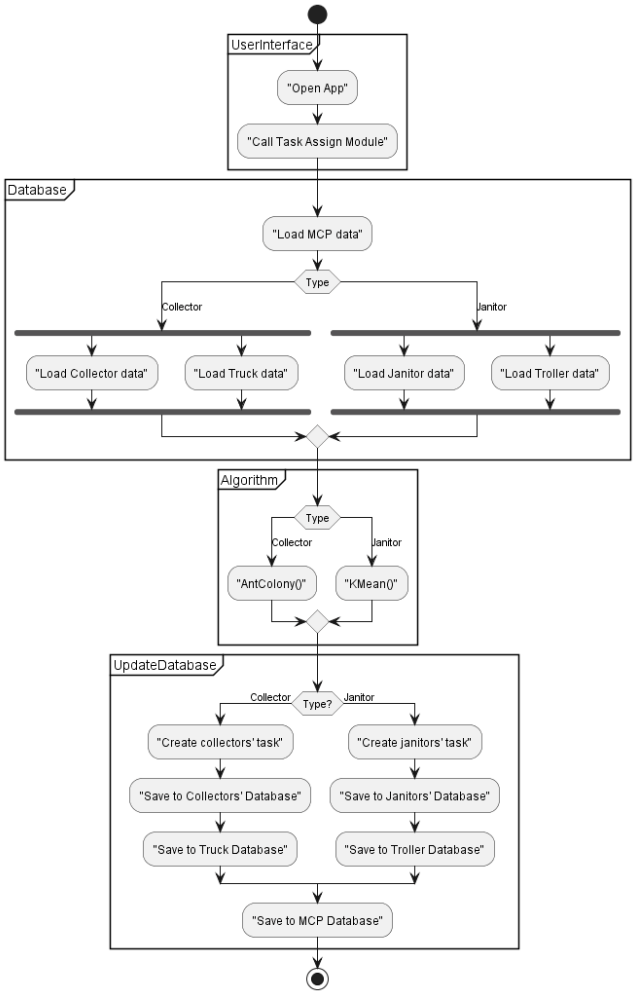
Task 2.1

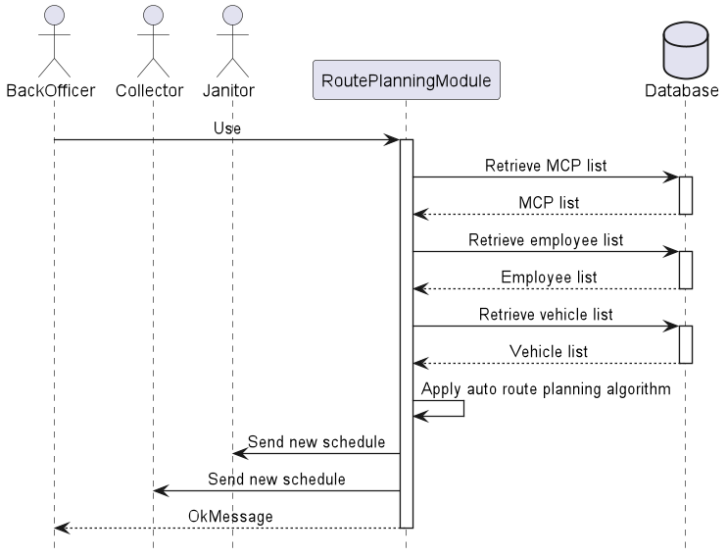


Task 2.2

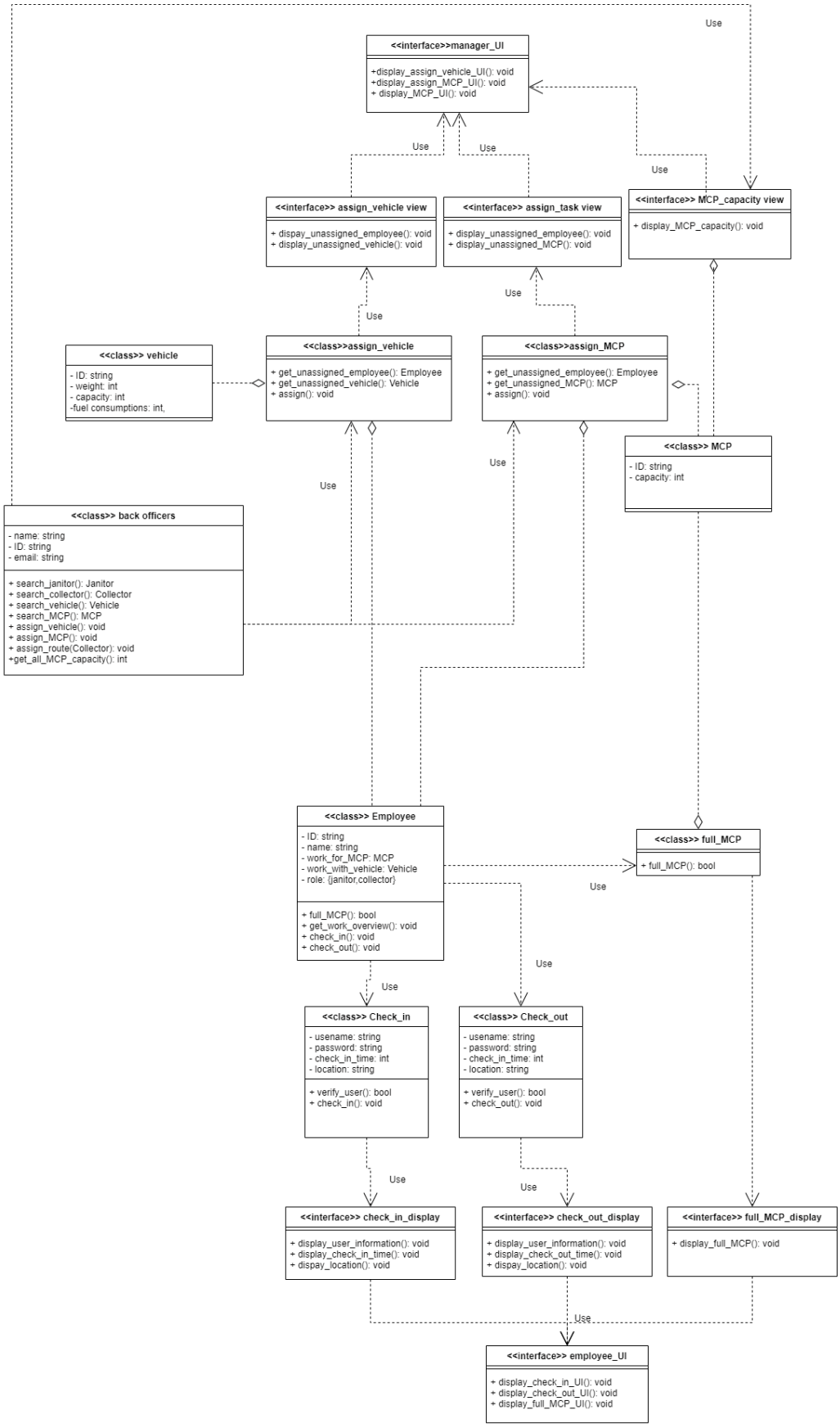
Route planning module includes the following steps:

- Retrieve information about MCPs (location, traveling distance to other MCPs…)
- Retrieve information about collectors and janitors (work days, location…)
- Retrieve information about vehicles (capacity…)
- From information about MCPs, employees, and vehicles, apply algorithm to automatically plan approriate routes and schedule for everyone.
- Message janitors and collectors about their new routes and schedule.

Sequence diagram:


BackOfficer
Collector
Janitor
RoutePlanningModule
Database
Use
Retrieve MCP list
MCP list
Retrieve employee list
Employee list
Retrieve vehicle list
Vehicle list
Apply auto route planning algorithm
Send new schedule
Send new schedule
OkMessage

Task 2.3


Use
<<interface>>manager_UI
+display_assign_vehicle_UI(): void
+display_assign_MCP_UI(): void
+ display_MCP_UI(): void
Use
Use
Use
<<interface>> assign_vehicle view
+ dispay_unassigned_employee(): void
+ display_unassigned_vehicle(): void
<<interface>> assign_task view
+ display_unassigned_employee(): void
+ display_unassigned_MCP(): void
<<interface>> MCP_capacity view
+ display_MCP_capacity(): void
Use
Use
<<class>> vehicle
- ID: string
- weight: int
- capacity: int
-fuel consumptions: int,
<<class>>assign_vehicle
+ get_unassigned_employee(): Employee
+ get_unassigned_vehicle(): Vehicle
+ assign(): void
<<class>>assign_MCP
+ get_unassigned_employee(): Employee
+ get_unassigned_MCP(): MCP
+ assign(): void
Use
Use
<<class>> MCP
- ID: string
- capacity: int
<<class>> back officers
- name: string
- ID: string
- email: string
+ search_janitor(): Janitor
+ search_collector(): Collector
+ search_vehicle(): Vehicle
+ search_MCP(): MCP
+ assign_vehicle(): void
+ assign_MCP(): void
+ assign_route(Collector): void
+get_all_MCP_capacity(): int
<<class>> Employee
- ID: string
- name: string
- work_for_MCP: MCP
- work_with_vehicle: Vehicle
- role: {janitor,collector}
+ full_MCP(): bool
+ get_work_overview(): void
+ check_in(): void
+ check_out(): void
<<class>> full_MCP
+ full_MCP(): bool
Use
Use
Use
<<class>> Check_in
- usename: string
- password: string
- check_in_time: int
- location: string
+ verify_user(): bool
+ check_in(): void
<<class>> Check_out
- usename: string
- password: string
- check_in_time: int
- location: string
+ verify_user(): bool
+ check_out(): void
Use
Use
<<interface>> check_in_display
+ display_user_information(): void
+ display_check_in_time(): void
+ dispay_location(): void
<<interface>> check_out_display
+ display_user_information(): void
+ display_check_out_time(): void
+ dispay_location(): void
<<interface>> full_MCP_display
+ display_full_MCP(): void
Use
<<interface>> employee_UI
+ display_check_in_UI(): void
+ display_check_out_UI(): void
+ display_full_MCP_UI(): void