UML Diagrams for Smart Campus Navigation and Facility Booking System

1. Use Case Diagram

Primary Actors:

- **Student**: Navigate campus, book study rooms
- **Faculty/Staff**: Book facilities, access advanced features

- **Administrator**: Manage system, facilities, and users
- **Visitor**: Basic navigation and information access

Use Cases:

Navigation Module:

- Search Buildings/Rooms
- Get Directions
- View Interactive Map
- Find Accessible Routes

Booking Module:

- Search Available Facilities
- Make Reservation
- Modify Booking
- Cancel Booking
- View Booking History
- Set Recurring Bookings

Administration Module:

- Manage Facilities
- Manage Users
- Configure Booking Rules
- Generate Reports
- System Configuration

```
@startuml
left to right direction
skinparam packageStyle rectangle

actor Student
actor "Faculty/Staff" as Faculty
actor Administrator
actor Visitor
```

```
rectangle "SCNFBS System" {
  usecase "Search Buildings/Rooms" as UC1
  usecase "Get Directions" as UC2
  usecase "View Interactive Map" as UC3
  usecase "Find Accessible Routes" as UC4
  usecase "Search Available Facilities" as UC5
  usecase "Make Reservation" as UC6
  usecase "Modify Booking" as UC7
  usecase "Cancel Booking" as UC8
  usecase "View Booking History" as UC9
  usecase "Set Recurring Bookings" as UC10
  usecase "Manage Facilities" as UC11
  usecase "Manage Users" as UC12
  usecase "Configure Booking Rules" as UC13
  usecase "Generate Reports" as UC14
  usecase "System Configuration" as UC15
}

Student --> UC1
Student --> UC2
Student --> UC3
Student --> UC4
Student --> UC5
Student --> UC6
Student --> UC7
Student --> UC8
Student --> UC9

Faculty --> UC1
Faculty --> UC2
Faculty --> UC3
Faculty --> UC4
Faculty --> UC5
Faculty --> UC6
Faculty --> UC7
Faculty --> UC8
Faculty --> UC9
Faculty --> UC10

Administrator --> UC11
Administrator --> UC12
Administrator --> UC13
Administrator --> UC14
Administrator --> UC15
Administrator --> UC1
Administrator --> UC2
```

```
Administrator --> UC3
Administrator --> UC4
Administrator --> UC5
Administrator --> UC6
Administrator --> UC7
Administrator --> UC8
Administrator --> UC9
Administrator --> UC10

Visitor --> UC1
Visitor --> UC2
Visitor --> UC3
Visitor --> UC4
@enduml
```

2. Class Diagram

Core Classes:

```
@startuml
class User {
  +userId: String
  +username: String
  +email: String
  +firstName: String
  +lastName: String
  +role: UserRole
  +isActive: Boolean
  +createdAt: DateTime
  +updatedAt: DateTime
  +authenticate(password: String): Boolean
  +hasPermission(permission: String): Boolean
  +getBookingHistory(): List<Booking>
}

enum UserRole {
  STUDENT
  FACULTY
  STAFF
  ADMINISTRATOR
  VISITOR
}

class Building {
  +buildingId: String
  +name: String
```

```
  +code: String
  +address: String
  +latitude: Double
  +longitude: Double
  +floors: Integer
  +isAccessible: Boolean
  +entrances: List<Entrance>
  +rooms: List<Room>
  +getRoomsByType(type: RoomType): List<Room>
  +getAvailableRooms(startTime: DateTime, endTime: DateTime): List<Room>
}

class Room {
  +roomId: String
  +number: String
  +name: String
  +type: RoomType
  +capacity: Integer
  +floor: Integer
  +isAccessible: Boolean
  +equipment: List<Equipment>
  +building: Building
  +bookings: List<Booking>
  +isAvailable(startTime: DateTime, endTime: DateTime): Boolean
  +getBookingsForDate(date: Date): List<Booking>
}

enum RoomType {
  STUDY_ROOM
  LECTURE_HALL
  LABORATORY
  CONFERENCE_ROOM
  SPORTS_VENUE
  LIBRARY_SPACE
  MEETING_ROOM
}

class Equipment {
  +equipmentId: String
  +name: String
  +type: String
  +isWorking: Boolean
  +room: Room
}

class Booking {
```

```
  +bookingId: String
  +user: User
  +room: Room
  +startTime: DateTime
  +endTime: DateTime
  +purpose: String
  +status: BookingStatus
  +isRecurring: Boolean
  +recurrencePattern: String
  +createdAt: DateTime
  +updatedAt: DateTime
  +validateBooking(): Boolean
  +sendConfirmation(): void
  +sendReminder(): void
}

enum BookingStatus {
  PENDING
  CONFIRMED
  CANCELLED
  COMPLETED
  NO_SHOW
}

class BookingRule {
  +ruleId: String
  +name: String
  +description: String
  +userRole: UserRole
  +roomType: RoomType
  +maxDurationHours: Integer
  +maxAdvanceDays: Integer
  +maxConcurrentBookings: Integer
  +isActive: Boolean
  +applies(user: User, room: Room): Boolean
}

class MapPath {
  +pathId: String
  +startPoint: Location
  +endPoint: Location
  +waypoints: List<Location>
  +distance: Double
  +estimatedTime: Integer
  +pathType: PathType
  +isAccessible: Boolean
```

```
  +instructions: List<String>
  +calculateRoute(): void
}

enum PathType {
  WALKING
  ACCESSIBLE
  INDOOR
  OUTDOOR
  MIXED
}

class Location {
  +locationId: String
  +latitude: Double
  +longitude: Double
  +floor: Integer
  +building: Building
  +description: String
  +isAccessible: Boolean
}

class Entrance {
  +entranceId: String
  +name: String
  +location: Location
  +building: Building
  +isAccessible: Boolean
  +isMain: Boolean
  +operatingHours: String
}

class Notification {
  +notificationId: String
  +user: User
  +type: NotificationType
  +title: String
  +message: String
  +isRead: Boolean
  +sentAt: DateTime
  +relatedBooking: Booking
  +send(): void
}

enum NotificationType {
  BOOKING_CONFIRMATION
```

```
  BOOKING_REMINDER
  BOOKING_CANCELLATION
  SYSTEM_MAINTENANCE
  FACILITY_UPDATE
}

class Report {
  +reportId: String
  +name: String
  +type: ReportType
  +generatedBy: User
  +generatedAt: DateTime
  +dateRange: DateRange
  +data: String
  +generate(): void
  +export(format: String): File
}

enum ReportType {
  FACILITY_UTILIZATION
  USER_ACTIVITY
  BOOKING_STATISTICS
  PEAK_HOURS
  REVENUE
}

' Relationships
User ||--o{ Booking : makes
Room ||--o{ Booking : reserved
Building ||--o{ Room : contains
Room ||--o{ Equipment : has
User ||--o{ Notification : receives
Building ||--o{ Entrance : has
BookingRule ||--o{ User : applies_to
BookingRule ||--o{ Room : restricts
Report ||--o{ User : generated_by
MapPath ||--o{ Location : connects
Building ||--o{ Location : located_at
@enduml
```

3. Sequence Diagrams

3.1 Facility Booking Process

```
@startuml
participant User
```

```
participant "Web Interface" as UI
participant "Booking Controller" as BC
participant "Room Service" as RS
participant "Booking Service" as BS
participant "Notification Service" as NS
participant Database

User -> UI: Search for available rooms
UI -> BC: searchAvailableRooms(criteria)
BC -> RS: findAvailableRooms(dateTime, type, capacity)
RS -> Database: query available rooms
Database -> RS: return room list
RS -> BC: available rooms
BC -> UI: display available rooms
UI -> User: show room options

User -> UI: Select room and time slot
UI -> BC: createBooking(roomId, startTime, endTime)
BC -> BS: validateBooking(booking)
BS -> Database: check room availability
Database -> BS: availability status
BS -> Database: check user booking limits
Database -> BS: booking limits
BS -> BC: validation result

alt Booking Valid
  BC -> BS: saveBooking(booking)
  BS -> Database: insert booking record
  Database -> BS: booking saved
  BS -> NS: sendConfirmation(booking)
  NS -> User: email confirmation
  BC -> UI: booking confirmed
  UI -> User: success message
else Booking Invalid
  BC -> UI: booking failed
  UI -> User: error message
end
@enduml
```

3.2 Campus Navigation Process

```
@startuml
participant User
participant "Web Interface" as UI
participant "Navigation Controller" as NC
participant "Map Service" as MS
```

```
participant "Path Service" as PS
participant Database

User -> UI: Search for destination
UI -> NC: searchLocation(query)
NC -> MS: findLocations(query)
MS -> Database: search buildings/rooms
Database -> MS: location results
MS -> NC: locations found
NC -> UI: display search results
UI -> User: show location options

User -> UI: Select destination and request directions
UI -> NC: getDirections(startLocation, endLocation, preferences)
NC -> PS: calculateRoute(start, end, pathType)
PS -> Database: get map data and paths
Database -> PS: path information
PS -> PS: calculate optimal route
PS -> NC: route with turn-by-turn directions
NC -> UI: navigation instructions
UI -> User: display interactive map with route

User -> UI: Request accessible route
UI -> NC: getAccessibleRoute(start, end)
NC -> PS: calculateAccessibleRoute(start, end)
PS -> Database: get accessible paths and elevators
Database -> PS: accessible route data
PS -> NC: accessible route
NC -> UI: accessible navigation
UI -> User: show accessible route with elevators/ramps
@enduml
```

4. Component Diagram

```
@startuml
package "Frontend Layer" {
  component "Web Interface" as WebUI
  component "Mobile Interface" as MobileUI
  component "Interactive Map" as MapUI
}

package "API Gateway" {
  component "Authentication Service" as AuthAPI
  component "REST API" as RestAPI
  component "WebSocket API" as WsAPI
}
```

```
package "Business Logic Layer" {
  component "User Management" as UserMgmt
  component "Facility Management" as FacilityMgmt
  component "Booking Engine" as BookingEngine
  component "Navigation Engine" as NavEngine
  component "Notification Service" as NotificationSvc
  component "Reporting Service" as ReportingSvc
}

package "Data Access Layer" {
  component "User Repository" as UserRepo
  component "Facility Repository" as FacilityRepo
  component "Booking Repository" as BookingRepo
  component "Map Data Repository" as MapRepo
}

package "External Systems" {
  component "Email Service" as EmailSvc
  component "University Auth System" as UniAuth
  component "Campus Management System" as CampusMgmt
}

database "PostgreSQL Database" as DB
database "Map Data Store" as MapDB
database "Cache Store (Redis)" as Cache

' Frontend connections
WebUI --> RestAPI
MobileUI --> RestAPI
MapUI --> WsAPI

' API Gateway connections
AuthAPI --> UniAuth
RestAPI --> UserMgmt
RestAPI --> FacilityMgmt
RestAPI --> BookingEngine
RestAPI --> NavEngine
RestAPI --> ReportingSvc
WsAPI --> NavEngine

' Business Logic connections
UserMgmt --> UserRepo
FacilityMgmt --> FacilityRepo
BookingEngine --> BookingRepo
BookingEngine --> NotificationSvc
```

```
NavEngine --> MapRepo
NotificationSvc --> EmailSvc
ReportingSvc --> BookingRepo
ReportingSvc --> FacilityRepo

' Data Access connections
UserRepo --> DB
FacilityRepo --> DB
BookingRepo --> DB
MapRepo --> MapDB
BookingEngine --> Cache
NavEngine --> Cache

' External connections
FacilityMgmt --> CampusMgmt
@enduml
```

5. Activity Diagram - Facility Booking Workflow

```
@startuml
start
:User logs into system;
:Search for available facilities;
:System displays available options;
:User selects facility and time slot;

if (User eligible for facility?) then (yes)
  if (Time slot available?) then (yes)
    if (Booking rules satisfied?) then (yes)
      :Create booking record;
      :Send confirmation email;
      :Display success message;
      stop
    else (no)
      :Display rule violation message;
      :Return to facility selection;
    endif
  else (no)
    :Display "not available" message;
    :Return to facility selection;
  endif
else (no)
  :Display access denied message;
  stop
endif
@enduml
```

These UML diagrams provide a comprehensive design foundation for the Smart Campus Navigation and Facility Booking System, covering all major system components, user interactions, and data flows.