Rooms Creation Module - Implementation Table

🎯 Scope: ROOMS CREATION ONLY

Note: Booking, payment, and 15-minute session locking will be handled in booking module.

BACKEND IMPLEMENTATION

1. Database Models (Already Exist ✅)

Component	File	Status	Notes
RoomTypes	`BACKEND/app/models/sqlalchemy_schemas/rooms.py`	✅ Complete	Has: id, type_name, max_adult, max_child, price, square_ft, images
Rooms	`BACKEND/app/models/sqlalchemy_schemas/rooms.py`	✅ Complete	Has: id, room_no, room_type_id, status, amenities
RoomAmenities	`BACKEND/app/models/sqlalchemy_schemas/rooms.py`	✅ Complete	Has: id, amenity_name
RoomAmenityMap	`BACKEND/app/models/sqlalchemy_schemas/rooms.py`	✅ Complete	Many-to-many relationship

Current Schema Check:

```
-- Rooms table already has these columns:
room_id (PK)
room_no (unique)
room_type_id (FK)
room_status (enum: AVAILABLE, BOOKED, MAINTENANCE, FROZEN, HELD)
freeze_reason
hold_expires_at -- (for future booking sessions)
created_at
updated_at
is_deleted
```

2. Pydantic Schemas (Request/Response Models)

Component	File	Status	Required For
RoomTypeCreate	`BACKEND/app/schemas/pydantic_models/room.py`	✅ Exists	Admin: Create room type
RoomTypeResponse	`BACKEND/app/schemas/pydantic_models/room.py`	✅ Exists	API response for room types
RoomCreate	`BACKEND/app/schemas/pydantic_models/room.py`	✅ Exists	Admin: Create individual room
RoomResponse	`BACKEND/app/schemas/pydantic_models/room.py`	✅ Exists	API response for rooms

Component	File	Status	Required For
RoomUpdate	`BACKEND/app/schemas/pydantic_models/room.py`	✅ Exists	Admin: Update room
AmenityCreate	`BACKEND/app/schemas/pydantic_models/room.py`	⚠️ Check	Admin: Create amenity

What to verify:

```
# Should exist in room.py:
class RoomTypeCreate(BaseModel):
    type_name: str
    max_adult_count: int
    max_child_count: int
    price_per_night: float
    square_ft: int

class RoomCreate(BaseModel):
    room_no: str
    room_type_id: int
    # Optional: amenity_ids: List[int]

class AmenityCreate(BaseModel):
    amenity_name: str
```

3. CRUD Operations (Database Layer)

Operation	File	Status	Implementation
Create room type	`BACKEND/app/crud/rooms.py`	✅ Exists	`create_room_type()`
Get room types	`BACKEND/app/crud/rooms.py`	✅ Exists	`get_room_types()`
Create room	`BACKEND/app/crud/rooms.py`	✅ Exists	`create_room()`
Get rooms	`BACKEND/app/crud/rooms.py`	✅ Exists	`get_rooms()` with filters
Update room	`BACKEND/app/crud/rooms.py`	✅ Exists	`update_room()`
Delete room	`BACKEND/app/crud/rooms.py`	✅ Exists	`delete_room()` (soft delete)
Create amenity	`BACKEND/app/crud/rooms.py`	⚠️ Check	`create_amenity()`
Map amenity to room	`BACKEND/app/crud/rooms.py`	⚠️ Check	`map_amenity_to_room()`
Unmap amenity	`BACKEND/app/crud/rooms.py`	⚠️ Check	`unmap_amenity_from_room()`

What each does:

```
# Create room type
create_room_type(session, RoomTypeCreate) → RoomTypes

# Create individual room
create_room(session, RoomCreate) → Rooms

# Link amenity to room
map_amenity_to_room(session, room_id, amenity_id) → RoomAmenityMap

# Get available rooms (filters by status, freeze, type)
get_rooms(session, room_type_id=None, is_freezed=None, status=None) → List[Rooms]
```

4. Service Layer (Business Logic)

Service Function	File	Status	Usage
`create_room_type()`	`BACKEND/app/services/rooms.py`	✅ Exists	Validates & creates room type
`list_room_types()`	`BACKEND/app/services/rooms.py`	✅ Exists	Lists all room types (cached)
`create_room()`	`BACKEND/app/services/rooms.py`	✅ Exists	Validates & creates single room
`list_rooms()`	`BACKEND/app/services/rooms.py`	✅ Exists	Lists rooms with filters
`get_room()`	`BACKEND/app/services/rooms.py`	✅ Exists	Gets single room details
`update_room()`	`BACKEND/app/services/rooms.py`	✅ Exists	Updates room info
`delete_room()`	`BACKEND/app/services/rooms.py`	✅ Exists	Soft-delete room
`bulk_upload_rooms()`	`BACKEND/app/services/rooms.py`	✅ Exists	Excel import
`add_amenity()`	`BACKEND/app/services/rooms.py`	⚠️ Check	Create & map amenity
`remove_amenity()`	`BACKEND/app/services/rooms.py`	⚠️ Check	Unmap amenity

Key validation in services:

```
# When creating room:
✓ room_no must be unique
✓ room_type_id must exist
✓ Check permissions (ROOM_MANAGEMENT:WRITE)

# When creating room type:
✓ type_name must be unique
✓ Prices & dimensions must be positive
✓ Adult/child counts must be valid
```

5. API Routes (Endpoints)

Endpoint	Method	Path	Status	Usage
Create room type	POST	`/room-management/types`	✅ Exists	Admin creates room type
List room types	GET	`/room-management/types`	✅ Exists	Get all types (cached 5 min)
Create room	POST	`/room-management/rooms`	✅ Exists	Admin creates single room
List rooms	GET	`/room-management/rooms`	✅ Exists	Get rooms with filters
Get room details	GET	`/room-management/rooms/{id}`	✅ Exists	Get single room info
Update room	PUT	`/room-management/rooms/{id}`	✅ Exists	Update room details
Delete room	DELETE	`/room-management/rooms/{id}`	✅ Exists	Delete room
Create amenity	POST	`/room-management/amenities`	✅ Exists	Admin creates amenity
List amenities	GET	`/room-management/amenities`	✅ Exists	Get all amenities
Map amenity	POST	`/room-management/rooms/{id}/amenities/{amenity_id}`	✅ Exists	Link amenity to room
Unmap amenity	DELETE	`/room-management/rooms/{id}/amenities/{amenity_id}`	✅ Exists	Remove amenity from room

Permission check on all routes:

```
# Scopes required:
POST/PUT/DELETE → ROOM_MANAGEMENT:WRITE
GET → No special scope (or ROOM_MANAGEMENT:READ)
```

6. Error Handling

Error Scenario	Status Code	Response
Room type name already exists	400	`{"detail": "Room type already exists"}`
Room number already exists	400	`{"detail": "Room number already exists"}`
Room type not found	404	`{"detail": "Room type not found"}`
Room not found	404	`{"detail": "Room not found"}`
Amenity not found	404	`{"detail": "Amenity not found"}`

Error Scenario	Status Code	Response
No permission	403	`{"detail": "Not authorized"}`
Invalid input	422	Validation error details

7. Caching Strategy

Data	Cache Key	TTL	Usage
Room types	`room_types:all`	5 min	List endpoint
Room list	`rooms:list:{filters}`	1 min	List endpoint with filters
Single room	`room:{room_id}`	2 min	Get single room
Amenities	`amenities:all`	10 min	Admin form dropdown

Cache invalidation: Clear on create/update/delete

FRONTEND IMPLEMENTATION

1. Models/Interfaces (TypeScript)

Model	File	Status	Properties
RoomType	`FRONTEND/hbs/src/app/core/models/room.model.ts`	❌ Create	id, typeName, maxAdult, maxChild, pricePerNight, squareFt
Room	`FRONTEND/hbs/src/app/core/models/room.model.ts`	❌ Create	id, roomNo, roomTypeId, status, amenities, createdAt
Amenity	`FRONTEND/hbs/src/app/core/models/room.model.ts`	❌ Create	id, name
RoomCreateRequest	`FRONTEND/hbs/src/app/core/models/room.model.ts`	❌ Create	roomNo, roomTypeId, amenityIds[]
RoomTypeCreateRequest	`FRONTEND/hbs/src/app/core/models/room.model.ts`	❌ Create	typeName, maxAdult, maxChild, price, squareFt

Create file structure:

```typescript
// room.model.ts
export interface RoomType {
  id: number;
  typeName: string;
  maxAdultCount: number;
  maxChildCount: number;
  pricePerNight: number;
  squareFt: number;
  createdAt: string;
}

export interface Room {
  id: number;
  roomNo: string;
  roomTypeId: number;
  status: 'AVAILABLE' | 'BOOKED' | 'MAINTENANCE' | 'FROZEN' | 'HELD';
  amenities: Amenity[];
  createdAt: string;
  updatedAt: string;
}

export interface Amenity {
  id: number;
  name: string;
}
```

2. Services

Service	File	Status	Methods
RoomService	`FRONTEND/hbs/src/app/core/services/rooms/rooms.service.ts`	⚠️ Partial	`getRoomTypes()`, `getRooms()`, `createRoomType()`, `createRoom()`, `updateRoom()`, `deleteRoom()`
AmenityService	`FRONTEND/hbs/src/app/core/services/rooms/amenity.service.ts`	❌ Create	`getAmenities()`, `createAmenity()`, `mapAmenity()`, `unmapAmenity()`

RoomService Methods to Add:

```
// rooms.service.ts
export class RoomService {

  // Existing
  getRoomTypes(): Observable<RoomType[]>
  getRooms(filters?): Observable<Room[]>

  // New - for admin panel
  createRoomType(data: RoomTypeCreateRequest): Observable<RoomType>
  createRoom(data: RoomCreateRequest): Observable<Room>
  updateRoom(id: number, data: Partial<Room>): Observable<Room>
  deleteRoom(id: number): Observable<void>
  getRoom(id: number): Observable<Room>
}
```

AmenityService to Create:

```
// amenity.service.ts
export class AmenityService {

  getAmenities(): Observable<Amenity[]>
  createAmenity(data: { name: string }): Observable<Amenity>
  mapAmenityToRoom(roomId: number, amenityId: number): Observable<void>
  unmapAmenityFromRoom(roomId: number, amenityId: number): Observable<void>
}
```

3. Components (Admin Section)

Component	File	Status	Purpose
RoomTypeManagementComponent	`FRONTEND/hbs/src/app/features/admin/rooms/room-type-management/`	❌ Create	Create/edit room types
RoomManagementComponent	`FRONTEND/hbs/src/app/features/admin/rooms/room-management/`	❌ Create	Create/edit individual rooms
AmenityManagementComponent	`FRONTEND/hbs/src/app/features/admin/rooms/amenity-management/`	❌ Create	Create/map amenities
RoomsBulkUploadComponent	`FRONTEND/hbs/src/app/features/admin/rooms/bulk-upload/`	❌ Create	Excel upload for rooms

RoomTypeManagementComponent:

```
Input Form:
├─ Type Name (text)
├─ Max Adults (number)
├─ Max Children (number)
├─ Price per Night (currency)
├─ Square Footage (number)
└─ [Create] [Cancel] buttons

Room Types List:
├─ Table showing all types
├─ Edit link per row
├─ Delete button per row
└─ Last updated timestamp
```

RoomManagementComponent:

```
Create Room Form:
├─ Room Number (text, unique)
├─ Room Type (dropdown - from API)
├─ Amenities (multi-select)
├─ [Create] [Cancel] buttons

Rooms List:
├─ Table with columns: Room No, Type, Status, Amenities
├─ Edit link per row
├─ Delete button per row
└─ Filter by type & status
```

AmenityManagementComponent:

```
Create Amenity Form:
├─ Amenity Name (text)
└─ [Create] button

Amenities List:
├─ List all amenities
└─ Delete button per row

Map Amenities Section:
├─ Select Room (dropdown)
├─ Select Amenities to add (multi-select)
├─ Current amenities list
└─ [Remove] button per amenity
```

4. Forms (Reactive Forms)

Form	Component	Status	Fields
RoomTypeForm	RoomTypeManagementComponent	❌ Create	typeName, maxAdult, maxChild, price, squareFt
RoomForm	RoomManagementComponent	❌ Create	roomNo, roomTypeId, amenityIds
AmenityForm	AmenityManagementComponent	❌ Create	name

Validation Rules:

```
// RoomTypeForm
typeName: required, minLength(3), maxLength(50)
maxAdult: required, min(1), max(10)
maxChild: required, min(0), max(10)
price: required, min(0.01)
squareFt: required, min(50)

// RoomForm
roomNo: required, minLength(1), maxLength(20), unique
roomTypeId: required
amenityIds: optional, array

// AmenityForm
name: required, minLength(2), maxLength(100), unique
```

5. Navigation & Routing

Route	Path	Component	Status	Access
Admin Rooms	`/admin/rooms`	RoomsAdminComponent	❌ Create	Admin only
Room Types Management	`/admin/rooms/types`	RoomTypeManagementComponent	❌ Create	ROOM_MANAGEMENT:WRITE
Room Management	`/admin/rooms/list`	RoomManagementComponent	❌ Create	ROOM_MANAGEMENT:WRITE
Amenity Management	`/admin/rooms/amenities`	AmenityManagementComponent	❌ Create	ROOM_MANAGEMENT:WRITE
Bulk Upload	`/admin/rooms/bulk-upload`	RoomsBulkUploadComponent	❌ Create	ROOM_MANAGEMENT:WRITE

Add to routing module:

```
const routes = [
  {
    path: 'admin/rooms',
    canActivate: [AuthGuard, PermissionGuard],
    data: { requiredPermission: 'ROOM_MANAGEMENT:WRITE' },
    children: [
      { path: 'types', component: RoomTypeManagementComponent },
      { path: 'list', component: RoomManagementComponent },
      { path: 'amenities', component: AmenityManagementComponent },
      { path: 'bulk-upload', component: RoomsBulkUploadComponent }
    ]
  }
];
```

WORKFLOW: Room Creation Process

Step 1: Admin Creates Room Type

1. Admin goes to /admin/rooms/types
2. Fills form (Type Name, Max Adults, Max Children, Price, Size)
3. Submits → POST /room-management/types
4. Backend creates RoomTypes record
5. Response shows new room type
6. Cache cleared automatically

Step 2: Admin Creates Individual Rooms

1. Admin goes to /admin/rooms/list
2. Selects "Create Room"
3. Fills form (Room Number, Room Type, Amenities)
4. Submits → POST /room-management/rooms
5. Backend creates Rooms record
6. Backend maps amenities (RoomAmenityMap entries)
7. Response shows new room
8. Room appears in list with status = AVAILABLE

Step 3: Admin Manages Amenities

1. Admin goes to /admin/rooms/amenities
2. Creates new amenity if needed → POST /room-management/amenities
3. Selects a room
4. Multi-select amenities to add
5. Submits → POST /room-management/rooms/{id}/amenities/{amenity_id}
6. Amenity linked to room

Step 4: Bulk Upload (Optional)

1. Admin goes to /admin/rooms/bulk-upload
2. Selects Excel file with room data (room_no, room_type_id, amenities)
3. Clicks upload → POST /room-management/rooms/bulk
4. Backend processes Excel
5. Creates rooms in batch
6. Returns success/error count

API CALLS SUMMARY

From Frontend to Backend

Action	Method	Endpoint	Body	Auth
Get room types	GET	`/room-management/types`	None	Bearer token
Create room type	POST	`/room-management/types`	RoomTypeCreate	WRITE perm

Action	Method	Endpoint	Body	Auth
Get rooms	GET	`/room-management/rooms?room_type_id=&status=`	None	Bearer token
Create room	POST	`/room-management/rooms`	RoomCreate	WRITE perm
Get single room	GET	`/room-management/rooms/{id}`	None	Bearer token
Update room	PUT	`/room-management/rooms/{id}`	Partial	WRITE perm
Delete room	DELETE	`/room-management/rooms/{id}`	None	WRITE perm
Get amenities	GET	`/room-management/amenities`	None	Bearer token
Create amenity	POST	`/room-management/amenities`	AmenityCreate	WRITE perm
Map amenity	POST	`/room-management/rooms/{id}/amenities/{aid}`	None	WRITE perm
Unmap amenity	DELETE	`/room-management/rooms/{id}/amenities/{aid}`	None	WRITE perm

STATUS CHECK: What Already Exists vs What to Build

✅ Already Implemented (Backend)

- Database models (rooms, room types, amenities, mappings)
- Pydantic schemas (request/response models)
- CRUD operations
- Service layer with validation
- API routes (all endpoints)
- Permission guards
- Error handling
- Caching strategy

Action: Verify these exist and test with Postman

❌ Need to Build (Frontend)

Item	File(s)	Time	Priority
Models/Interfaces	`room.model.ts`	30 min	HIGH
Room Service enhancement	`rooms.service.ts`	30 min	HIGH
Amenity Service	`amenity.service.ts`	30 min	HIGH
Room Type Component	`room-type-management/`	1 hour	HIGH
Room Management Component	`room-management/`	1.5 hours	HIGH
Amenity Management Component	`amenity-management/`	1 hour	MEDIUM
Bulk Upload Component	`bulk-upload/`	1 hour	LOW
Routing setup	`admin-routing.module.ts`	30 min	HIGH

Item	File(s)	Time	Priority
Navigation menu item	`admin-menu/`	15 min	MEDIUM

Total frontend time: 5-6 hours

TESTING CHECKLIST

Backend Testing (Postman)

- ☐ POST /room-management/types → Create room type
- ☐ GET /room-management/types → List room types (check cache)
- ☐ POST /room-management/rooms → Create room
- ☐ GET /room-management/rooms → List rooms with filters
- ☐ GET /room-management/rooms/{id} → Get single room
- ☐ PUT /room-management/rooms/{id} → Update room
- ☐ POST /room-management/amenities → Create amenity
- ☐ POST /room-management/rooms/{id}/amenities/{aid} → Map amenity
- ☐ DELETE /room-management/rooms/{id}/amenities/{aid} → Unmap amenity
- ☐ DELETE /room-management/rooms/{id} → Delete room

Frontend Testing

- ☐ Room type form validates required fields
- ☐ Can create room type from UI
- ☐ Room types list populates from API
- ☐ Room form dropdown loads room types
- ☐ Can create room from UI
- ☐ Rooms list shows created rooms
- ☐ Amenities multi-select works
- ☐ Can delete rooms
- ☐ Permission errors handled gracefully

NEXT STEPS (For Implementation Tomorrow)

Morning (Backend Verification - 1 hour)

- ☐ Read through rooms routes
- ☐ Read through rooms service
- ☐ Test all endpoints in Postman
- ☐ Verify database schema matches documentation
- ☐ Check permission scopes are correct

Afternoon (Frontend Build - 5-6 hours)

- ☐ Create room.model.ts with all interfaces
- ☐ Enhance rooms.service.ts with new methods

- [ ] Create amenity.service.ts
- [ ] Build RoomTypeManagementComponent
- [ ] Build RoomManagementComponent
- [ ] Build AmenityManagementComponent
- [ ] Add routing
- [ ] Test complete flow

FUTURE: NOT IN SCOPE (Rooms Creation)

❌ Room search by date (for booking module)
❌ Room availability checking (for booking module)
❌ Room holding/locking (for booking module)
❌ Room images upload (for room type details)
❌ Room freezing/unfreezing (for admin management)

These will be in: Booking Module & Admin Module