# Grievance Cell For Organization

1. Project Structure

Grievance\_cell/

├── Department/

│   ├── APIs.py

│   ├──crud.py

│   ├── models.py

│   └── schemas.py

├── User/

│   ├── APIs.py

│   ├──crud.py

│   ├── models.py

│   └── schemas.py

├── Grievances/

│   ├── APIs.py

│   ├──crud.py

│   ├── models.py

│   └── schemas.py

├── Comments/

│   ├── APIs.py

│   ├──crud.py

│   ├── models.py

│   └── schemas.py

├── auth.py

├── dependencies.py

├── roles.py

├── database.py

└── main.py

1. Roles & Authorization

| **Role Enum** | **Description** |
| --- | --- |
| user | Can sign up, log in, create grievances, add/view own comments/grievances |
| employee | Can view/apply to assigned grievances, add comments |
| admin | Can create departments and assign grievances to employees |
| super\_admin | Full access—manage departments, users, assignments |

1. API Endpoints
   1. Authentication Endpoints

| **Method** | **Path** | **Body** | **Response** | **Description** |
| --- | --- | --- | --- | --- |
| POST | /signup | { email, password, department\_id, role } | UserFull | Create user and return full details |
| POST | /login | Form: username, password | { access\_token, token\_type } | JWT login |

* 1. User Endpoints

| **Method** | **Path** | **Body** | **Response** | **Access Roles** | **Description** |
| --- | --- | --- | --- | --- | --- |
| POST | /users/ | UserCreate | UserFull | admin, employee, super\_admin | Create new user |
| GET | /users/ | — | List of UserLimited / UserFull | All roles | List users (limited info for user) |
| GET | /users/{id} | — | UserLimited or UserFull | All roles | Get user detail (limited if not self/admin) |

* 1. Department Endpoints

| **Method** | **Path** | **Body** | **Response** | **Access Roles** | **Description** |
| --- | --- | --- | --- | --- | --- |
| POST | /departments/ | DepartmentCreate | DepartmentOut | admin, super\_admin | Create a department |
| GET | /departments/ | — | List of DepartmentOut | employee, admin, super\_admin | List departments |

* 1. Grievance Endpoints

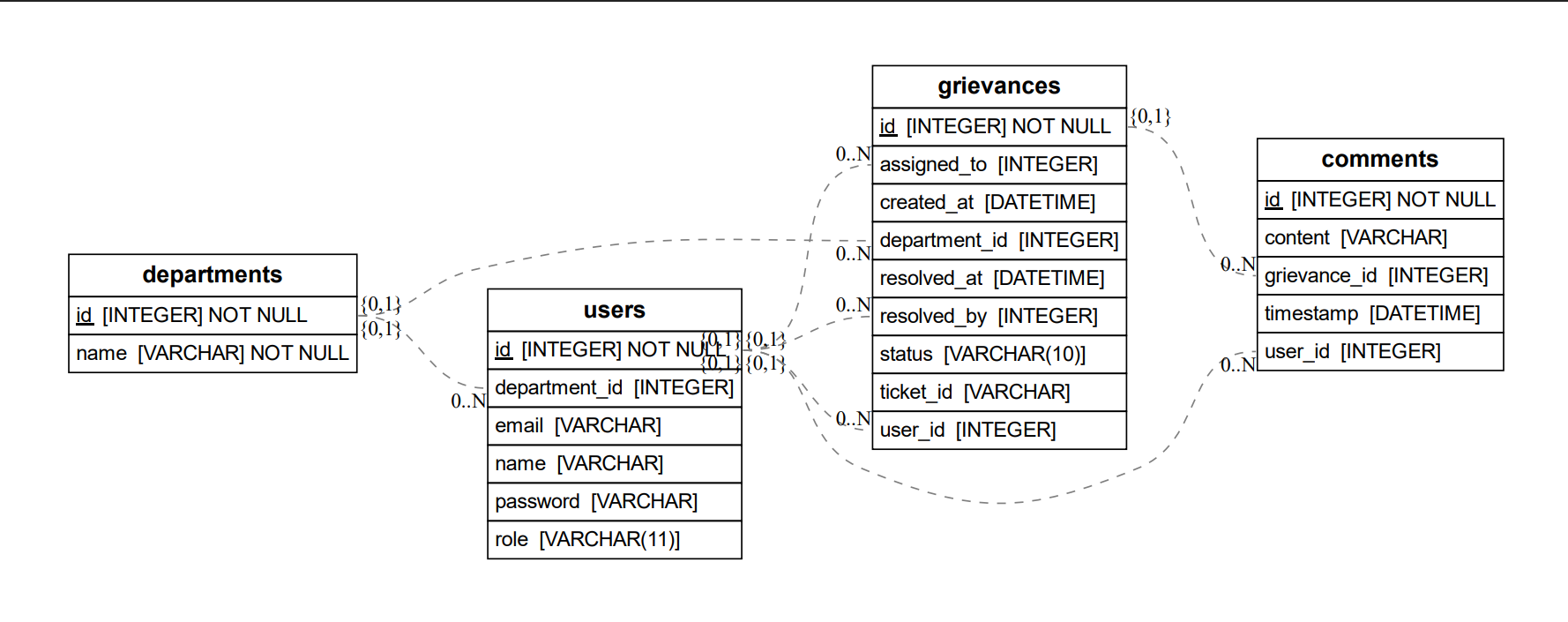
| **Method** | **Path** | **Body / Query** | **Response** | **Access Roles** | **Description** |
| --- | --- | --- | --- | --- | --- |
| POST | /grievances/ | GrievanceCreate | Grievance | user only | Create grievance; auto adds ticket\_id, status pending |
| GET | /grievances/ | — | List of Grievance | All roles (filtered) | Users see own, employees see assigned, admin/super admin see all |
| POST | /grievances/assign | — | 204 No Content | admin, super\_admin | Distributes pending grievances evenly to employees |
| POST | /grievances/{id}/resolve | ResolverSchema | Full Grievance | employee, admin, super\_admin | Mark grievance solved/not\_solved, sets resolved\_by, resolved\_at |

* 1. Comments API

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /comments/ | Post a new comment |
| GET | /comments/grievance/{grievance\_id} | View comments on grievance |

1. Database

In this Project we incorporated SQLite Database



Entity Relationship Diagram For Grievance.db

* 1. Users Tables

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | Integer PK | Auto-incrementing user ID |
| email | String | Unique user email |
| password | String | Hashed password using bcrypt |
| role | Enum | Role in the system (user, employee, etc.) |
| department\_id | ForeignKey | Links to departments.id |

**Relationships**:

* Many users → One department
  1. departments Table

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | Integer PK | Unique identifier for each department |
| name | String | Department name |

**Relationships**:

* One department → Many users
* One department → Many grievances
  1. Grievances Table

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | Integer PK | Unique grievance ID |
| title | String | Short description of grievance |
| description | String | Full grievance explanation |
| ticket\_id | String | Auto-generated unique ticket ID (UUID) |
| status | Enum | pending, resolved, not\_resolved |
| created\_at | DateTime | Timestamp when the grievance was created |
| resolved\_at | DateTime | Timestamp when resolved |
| user\_id | ForeignKey | Who created the grievance (User) |
| department\_id | ForeignKey | Which department it's assigned to |
| assigned\_to | ForeignKey | Which employee is assigned (nullable) |
| resolved\_by | ForeignKey | Who resolved the grievance (nullable) |

**Relationships**:

* One grievance → One user
* One grievance → One department
* One grievance → One assigned employee (nullable)
* One grievance → One resolver (nullable)
  1. Comments Table

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | Integer PK | Unique comment ID |
| content | Text | Comment body |
| timestamp | DateTime | When the comment was made |
| user\_id | ForeignKey | Author of the comment |
| grievance\_id | ForeignKey | Related grievance |

1. CRUD

5.1 User

This module manages user-related operations, including user creation, authentication, and fetching user records. It integrates department assignment logic and password hashing for security.

**Functions:**

* **get\_password\_hash(password: str) -> str**  
  Hashes a plain-text password using bcrypt for secure storage.
* **verify\_password(plain: str, hashed: str) -> bool**  
  Verifies a plain-text password against the hashed version.
* **create\_user(db: Session, user: schemas.UserCreate)**  
  Registers a new user. If no department is specified, it assigns or creates a default "OTR" department. The password is securely hashed before storing.
* **authenticate\_user(db: Session, email: str, password: str) -> Optional[models.User]**  
  Validates user credentials during login. Returns user object if authentication succeeds.
* **get\_users(db: Session, role\_filter: List[str] = None)**  
  Retrieves all users. If role\_filter is provided, returns only users matching the specified roles.
* **get\_user(db: Session, user\_id: int)**  
  Fetches a single user by their unique ID.

5.2 **Department Module – crud.py**

This module handles creation and retrieval of department records.

**Functions:**

* **create\_department(db: Session, department: schemas.DepartmentCreate)**  
  Creates and stores a new department using the provided department name.
* **get\_departments(db: Session)**  
  Retrieves all departments stored in the database.

5.3 **Grievance Module – crud.py**

This module is responsible for all grievance-related database actions, including creating, assigning, retrieving, resolving, and transferring grievances.

**Functions:**

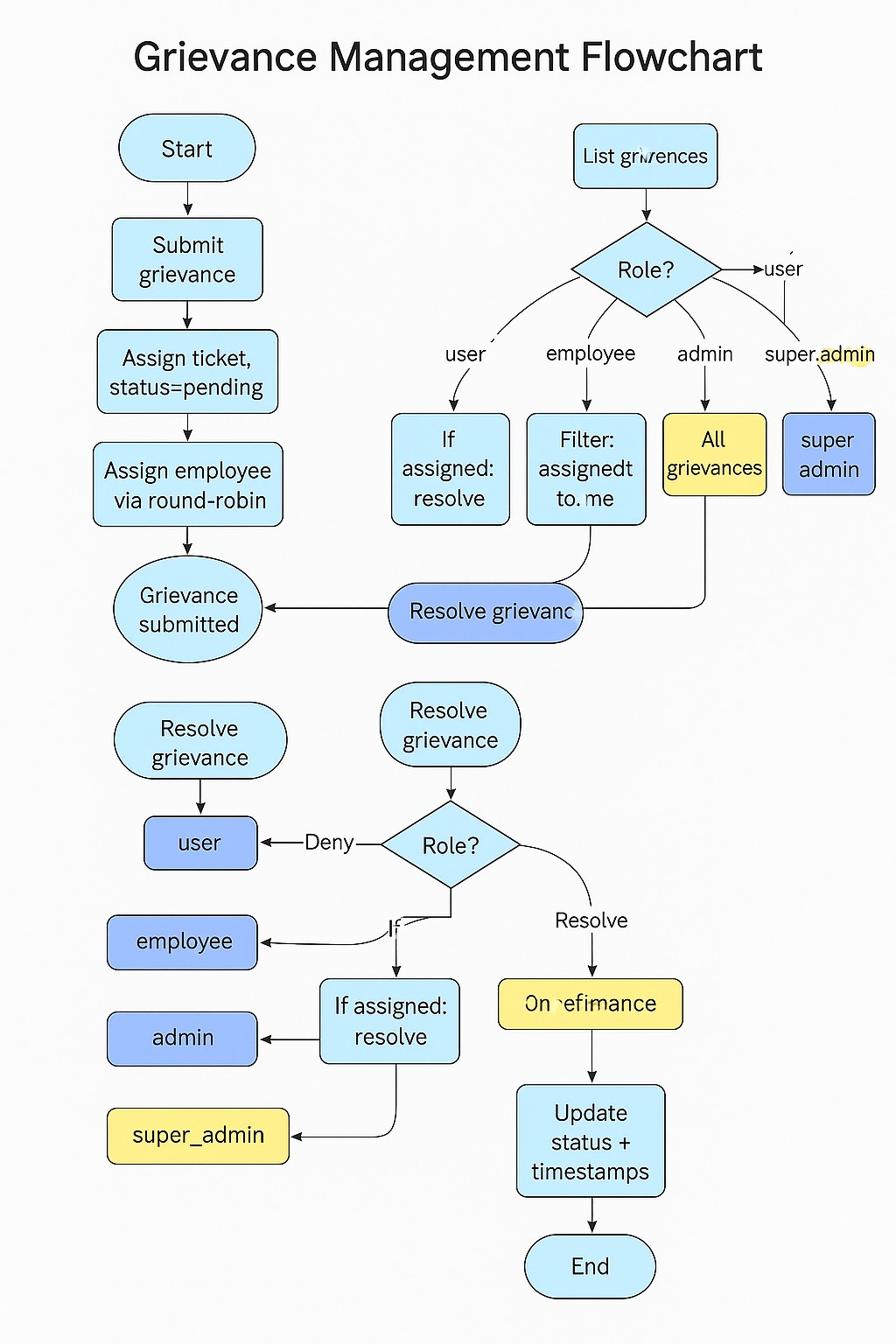
* **create\_grievance(db: Session, grievance: schemas.GrievanceCreate, user\_id: int)**  
  Creates a new grievance entry linked to a user and department. Generates a unique ticket ID for tracking.
* **assign\_grievances\_to\_employees(db: Session)**  
  Distributes all unassigned grievances to employees in a round-robin fashion.
* **get\_grievances\_by\_user(db: Session, user\_id: int)**  
  Retrieves all grievances submitted by a specific user.
* **get\_grievances\_by\_employee(db: Session, employee\_id: int)**  
  Retrieves grievances assigned to a specific employee.
* **get\_all\_grievances(db: Session)**  
  Retrieves all grievances in the system without filters.
* **resolve\_grievance(db: Session, grievance\_id: int, resolver\_id: int, solved: bool = True)**  
  Updates the status of a grievance to either resolved or not resolved. Also records who resolved it and when.
* **transfer\_grievance\_department(db: Session, grievance\_id: int, new\_department\_id: int)**  
  Transfers a grievance to another department using the grievance’s numeric ID.
* **transfer\_grievance\_by\_ticket\_id(db: Session, ticket\_id: str, new\_department\_id: int)**  
  Transfers a grievance to another department using the grievance’s ticket ID.

5.4 **Comment Module – crud.py**

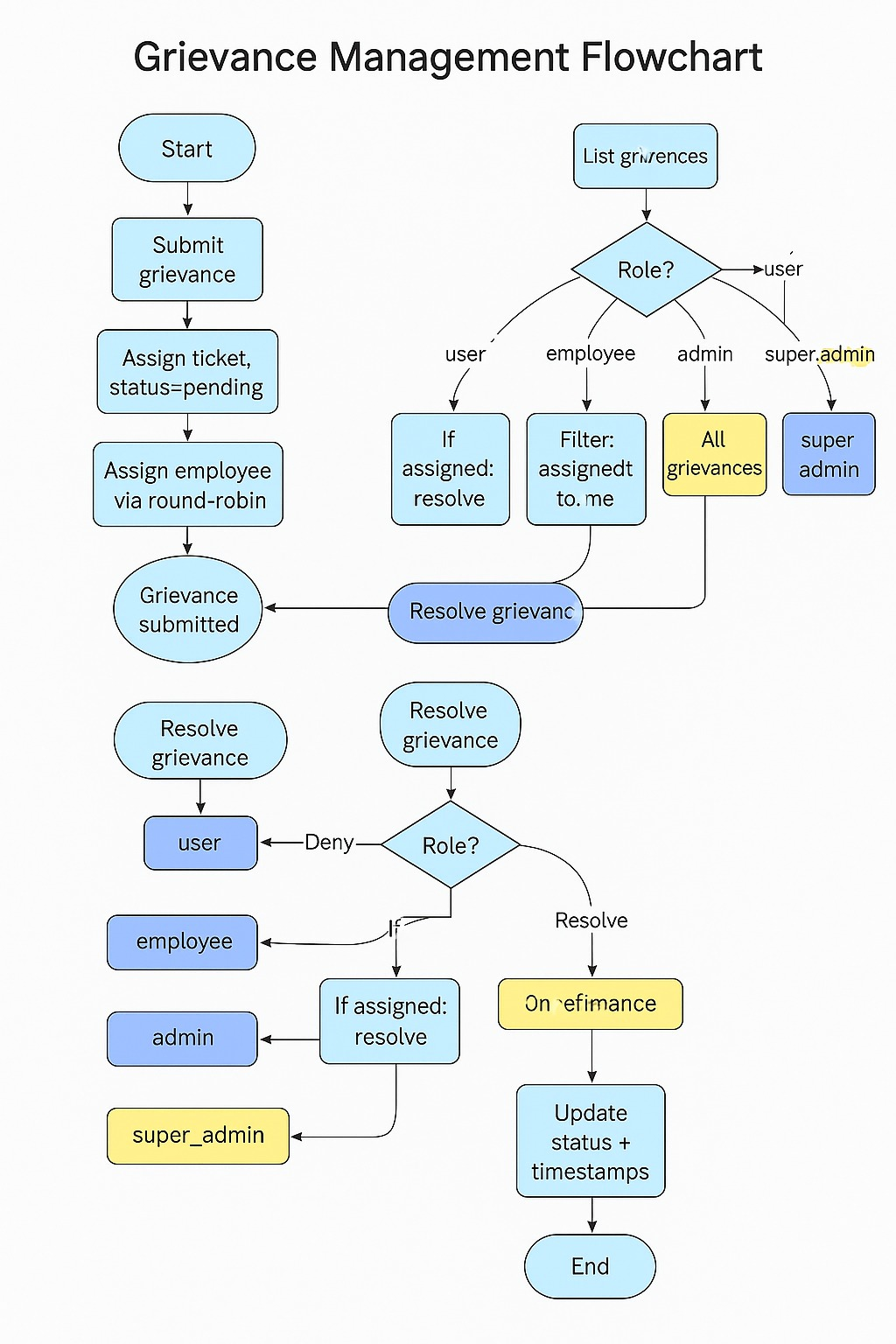
This module handles operations related to user comments on grievances.

**Functions:**

* **create\_comment(db: Session, comment: schemas.CommentCreate)**  
  Creates a new comment and automatically attaches the current timestamp to it.
* **get\_comments\_by\_grievance(db: Session, grievance\_id: int)**  
  Retrieves all comments associated with a specific grievance.

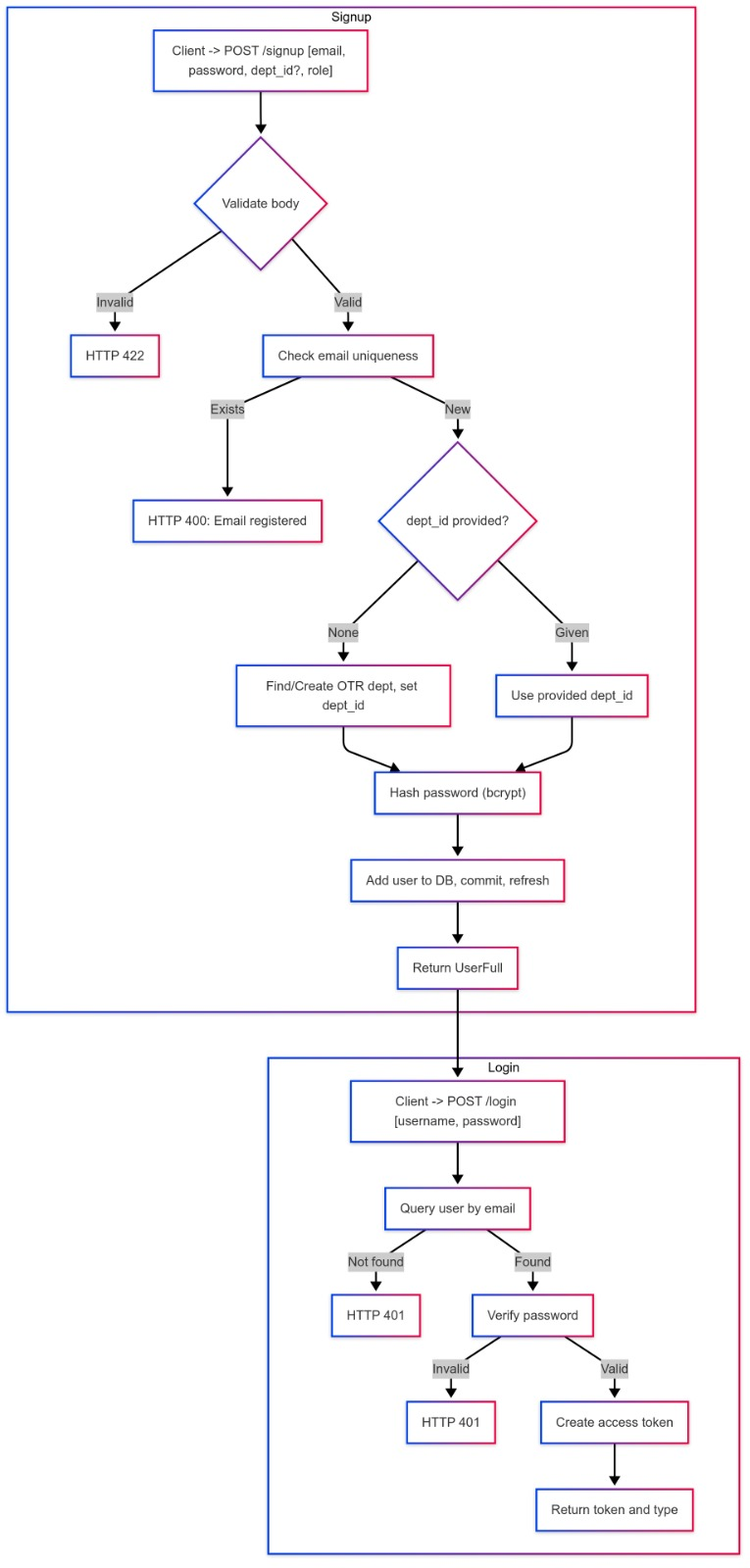


Over all Workflow of System

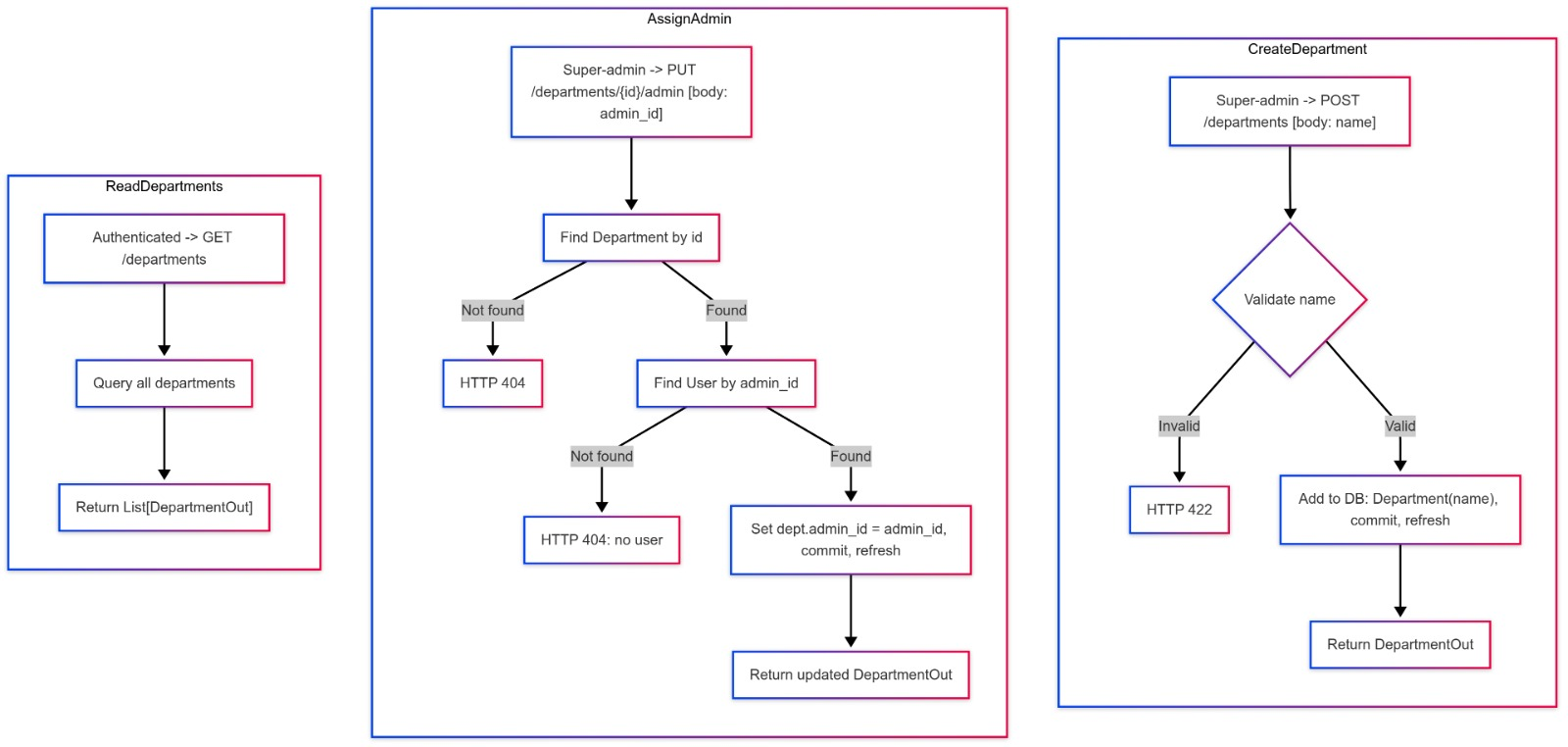


**Separated Workflow Of All Modules**

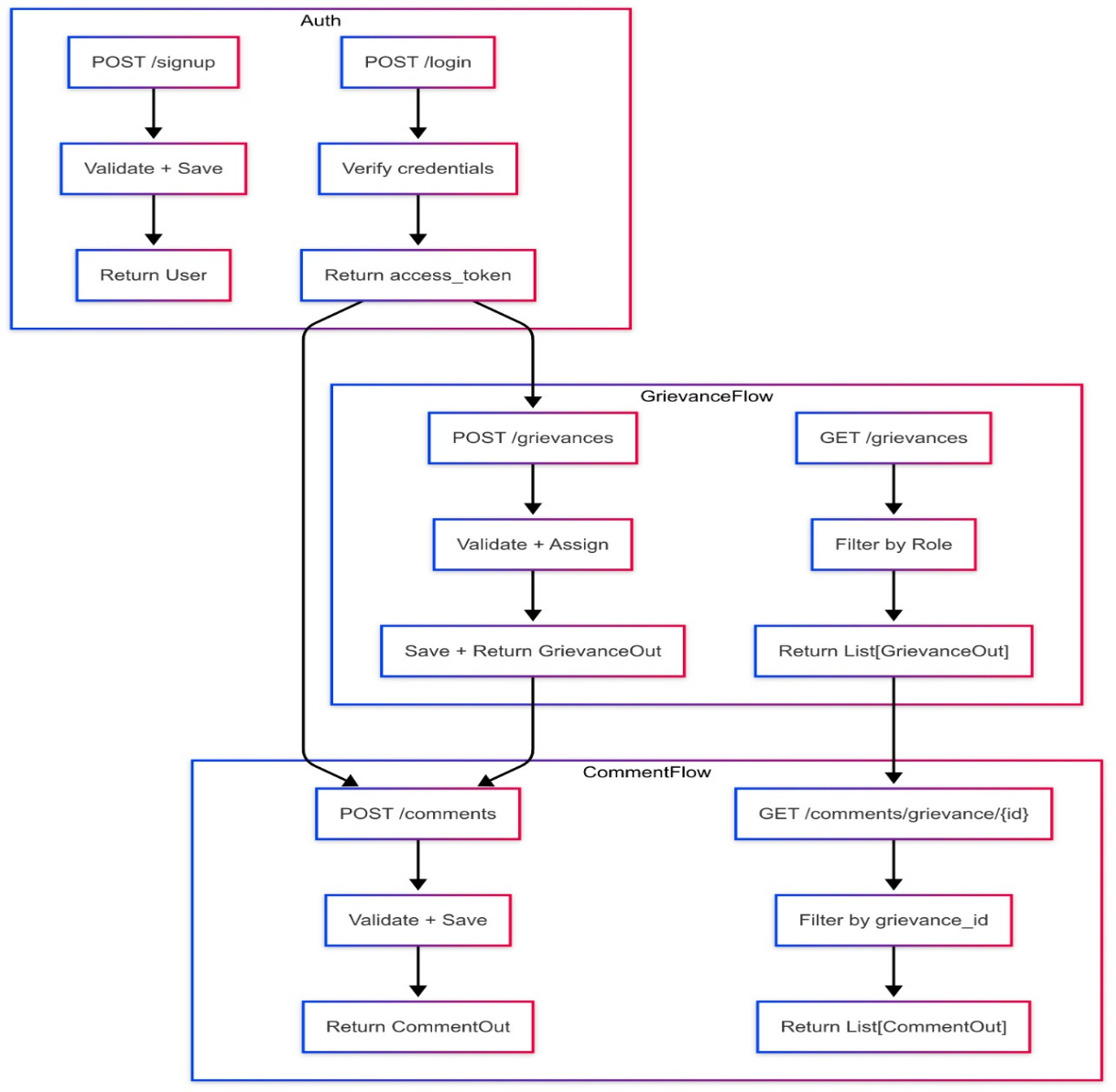
User Module



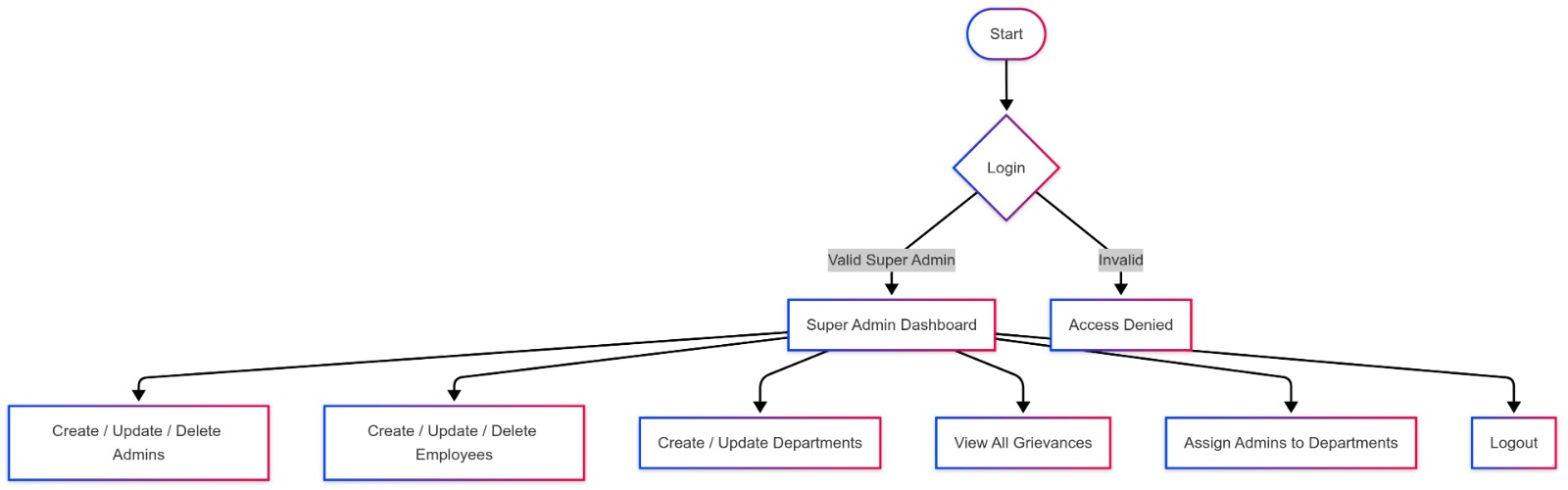
Department Module



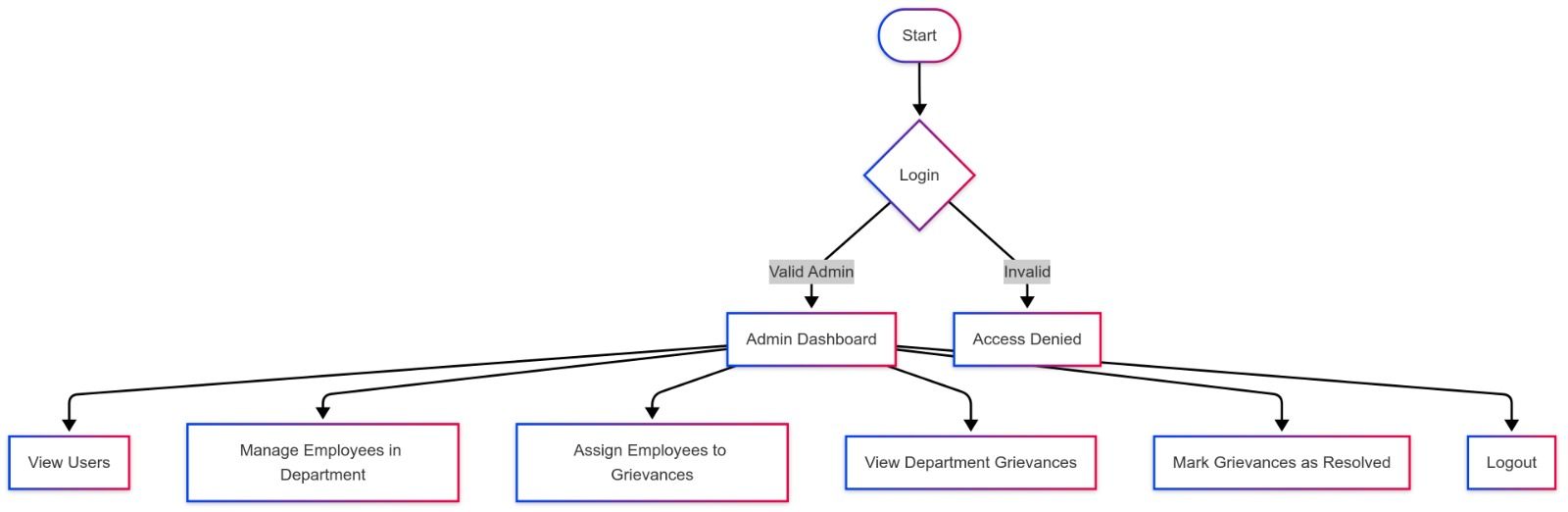
Comment Module



Super Admin Role



Admin Role



Employee role

