1) Business Challenges and Objectives

Group Business Overview: In science and engineering labs, limited resources like analytical instruments and rooms need orderly scheduling and use by students, and administrators must arrange maintenance, calibration, and shutdowns.

Current Pain Points: Resource scarcity causes conflicts; no one is accountable for no-shows or late arrivals; schedules conflict with maintenance; it is hard to combine utilization rates and historical records.

System Improvement Process (Collection → Processing → Reporting):

Collection: Actions including reservations, waitlist entries, maintenance, actual usage, and penalties.

Processing: Resource+time occupancy mutual exclusion checks; waitlist conversion to confirmed slots; eligibility (quota) validation; violation logging.

Reporting: Resource utilization rates, individual usage totals, violation statistics, maintenance occupancy, etc.

2) Entity List (Name, Business Definition, Attributes, Primary Relationships)

USER (User, Supertype)

Attributes:

UserID

FirstName

MiddleName

LastName

Email

Phone

UserType

STUDENT

Attributes:

Major

ADMIN

Attributes:

Department

AccessLevel

QUOTA_PLAN

Attributes:

PlanID

StartDate

ExpirationDate

AccessLevel

PenaltyPoints

Status

LOCATION

Attributes:

LocationID

Building

RoomNo

Capacity

OpenHours

INSTRUMENT

Attributes:

InstrumentID

Name

Model

SerialNo

AccessLevel

Notes

OCCUPANCY

Attributes:

OccupancyID

Date

StartTime

EndTime

OccupancyType

Status

RESERVATION

Attributes:

BookedAt

CheckInTime

CheckOutTime

MAINTENANCE

Attributes:

Type

Priority

Notes

WAITLIST

Attributes:

WaitListID

Desired Start

Desired End

Requested At

Status

PENALTY_RULE

Attributes:

PenaltyRuleID

ViolationType

Description

Points

PENALTY

Attributes:

PenaltyID

Reason

CreatedAt

Notes

IsActive

SCHEDULE

Attributes:

InstrumentID

Date

StartTime

EndTime

Status

3)Reasons for Entity Selection (One by One)

USER (User, Supertype)

Serves as the unified abstraction for all natural person accounts (identity, contact information, status). Numerous relationships (qualifications, appointments, maintenance, penalties) originate from "people," necessitating unified authorization, deactivation, or statistical tracking, thus justifying its existence.

STUDENT (Student, USER subtype)

Business roles create different rules and relationships (for example, creating appointments, registering for waitlists, receiving penalties). Making students a subtype makes it clear "who can do what" and supports usage and violation statistics by student.

ADMIN (Administrator, USER subtype)

Only administrators can schedule maintenance or suspensions, and their actions differ from students. Making it a subtype shows who owns maintenance creation, management duties, and audit responsibility, so business relationships are clear.

QUOTA_PLAN (User Quota Record)

Quotas have time limits, access level, penalty points and history (like temporary subway passes). Defining it as its own entity shows historical changes, current validity, and links with reservation eligibility; keeping it only as a USER attribute blocks history tracking and may cause redundancy and errors.

LOCATION

Locations have independent management value. If we treat them only as text attributes of INSTRUMENT, we lose the ability to model "multiple resources at the same location," which then confuses later constraints.

INSTRUMENT (Resource/Equipment)

This is the core entity for reservations, maintenance, and usage. Keeping it as an entity allows location tracking, statistics, maintenance, and key relationships with SCHEDULE, and WAITLIST; putting occupancy or maintenance only on the user side would add redundancy and make later relationship queries harder.

OCCUPANCY (Occupancy, supertype)

This unifies the idea of "who occupies a resource for what purpose during a time period." Without this supertype, RESERVATION and MAINTENANCE would each define time windows and mutual exclusion rules, which causes repetition and confusion.

RESERVATION (Reservation, Occupancy Subtype)

This means "confirmed occupancy entitlement." It links to students and drives check-in, actual usage, and penalty rule triggers. It is not just a state on other entities; it is an independent business object.

MAINTENANCE (Maintenance, Occupancy Subtype)

This sets time windows for "system/administrator occupancy" (repairs, calibration, cleaning, facility closure). Listing it with Reservation as an occupancy subtype clearly shows mutually exclusive and prioritized rules at the conceptual level.

WAITLIST (Standby)

This is an associative entity for "student-resource-preferred time window," with its own attributes (Desired, Requested, Status). If we treat it as a reservation state, we cannot manage the queue, sort it, or control release strategies during the "pending confirmation" phase.

PENALTY_RULE (Penalty Rule)

This separates "event → score adjustment" settings from their instances, so we can revise, create, and delete rules. If we mix it with penalty entities, we cannot trace the original rules.

PENALTY (Penalty Ledger)

This records each violation (timestamp, score, reason) and supports accumulation and review later. If we keep only the "current score," we cannot explain how the score was formed or handle disputes.

SCHEDULE(Penalty Ledger)

This keeps track of all the schedules of every instrument, this is what allows the system to function as the composite key for the attributes of the instrumentid, date, starttime and endtime, tells us if the instrument is reserved during that time frame.

4) Related Lists (Name/Description, Participating Entities, Cardinality & Optionality, Trigger Rules)

Student Plan Binding: STUDENT—QUOTA_PLAN

STUDENT(1) ↔ QUOTA_PLAN(0..N)

Generates historical records when users obtain/change quotas.

Locations contains Instruments: LOCATION—INSTRUMENT

LOCATION(M) ↔ INSTRUMENT(0..N)

Location may contain one or more instruments and an Instrument is present at atleast 1 Location.

Occupancy for schedule: SCHEDULE—OCCUPANCY

SCHEDULE (1) ↔ OCCUPANCY(1)

Occupancy for a specific time can have only one schedule. For one schedule there can only be one occupancy.

Occupancy Event: Occupancy Super/Subtype: OCCUPANCY→{RESERVATION, MAINTENANCE}

Mutually exclusive, fully overlapping

Each belongs to its subtype when creating reservation/maintenance.

Student Creates Reservation: STUDENT—RESERVATION

STUDENT(1) ↔ RESERVATION(0..N)

Reservation generated after student submission and conflict check pass.

Administrator Schedules Maintenance: ADMIN—MAINTENANCE

ADMIN(1) ↔ MAINTENANCE(0..N)

Administrator publishes maintenance/calibration windows.

Reservation Corresponds to Actual Usage: RESERVATION—USAGE

RESERVATION(1) ↔ USAGE(0..1)

Arrival and departure constitute one usage; no usage if absent/cancelled.

Waitlist Registration : STUDENT—WAITLIST—INSTRUMENT

STUDENT(1) ↔ WAITLIST(0..N); INSTRUMENT(1) ↔ WAITLIST(0..N)

Register for waitlist when target time slot is occupied.

Waitlist Conversion (Optional): WAITLIST→RESERVATION

WAITLIST(0..1) ↔ RESERVATION(0..1)

Converted upon slot availability per policy. Reservation need not have a waitlist and a waitlist maybe cancelled before turning into a reservation

Rule-Triggered Ledger: PENALTY_RULE—PENALTY

PENALTY_RULE(1) ↔ PENALTY(0.. N)

Records events like late arrival/no-show/late cancellation.

Ledger Assigned to Student: STUDENT—PENALTY---RESERVATION

STUDENT(1) ↔ PENALTY(0..N) ↔ RESERVATION (1)

Records score changes under the student's name.

User Super/ Subtypes: USER→{STUDENT, ADMIN}

Mutually exclusive, fully covering

Role determined after account creation/authorization.

5) Key Design Decisions (Conceptual)

Why include OCCUPANCY (supertype for occupancy)

It unifies how we show “resource × time” occupancy facts, avoids two tracks where maintenance is a state and reservations are events, and lets us declare “time mutual exclusivity within the same resource” in one place.

Why USER uses super/subtypes

STUDENT and ADMIN have different roles and permissions (create reservations vs schedule maintenance). Using ISA shows these differences clearly in the conceptual diagram.

Why LOCATION is an independent entity

Locations have their own attributes (capacity/open periods/hierarchy) and a one-to-many link with resources. Separation supports “multiple resources per location” rules and statistics.

Why retain QUOTA_PLAN

It shows historical eligibility/quotas and what is “currently valid,” which we need to decide whether booking is allowed.

Why use RULE+PENALTY ledgers for penalties

Separating rule setup from penalty posting helps auditing, traceability, and rule changes.