**1. Communication**

* Identified key problems like manual room allocation, delayed complaints, and paper-based processes.
* Gathered input from stakeholders: students, hostel staff, provosts, university admin, and IT support.
* Defined target users and their roles clearly (e.g., student, staff, admin).
* Documented system goals to improve communication, transparency, and efficiency.

**2. Planning**

* System development planned in 3 phases:
  + **Phase 1:** Core features (login, room management, complaint system, notice board).
  + **Phase 2:** Enhancements (file attachments, lost & found, visitor log, reporting).
  + **Phase 3:** Advanced tools (stipend system, email alerts, mobile support).
* Timelines for each phase (4 to 6 weeks for Phase 1, etc.).
* Prioritized features based on importance and complexity.

**3. Modeling**

* Detailed system architecture using Django’s MVT (Model-View-Template) structure.
* Clear database models for each feature (e.g., StudentProfile, Room, Complaint, Notice, etc.).
* Defined user roles and permissions.
* Provided schema diagrams and data relationships (e.g., One-to-One, Foreign Keys).

**4. Construction**

* Technology stack includes Django, Bootstrap, PostgreSQL/MySQL, jQuery, etc.
* Used Django’s built-in admin, authentication, and ORM for fast development.
* Code structure organized into apps like authentication, rooms, complaints, etc.
* Included example models and functional requirements for every feature.
* Packages listed in requirements.txt for easy setup.

**5. Deployment**

* Development setup uses Django’s local server and SQLite.
* Production uses PostgreSQL, Gunicorn, Nginx, and HTTPS.
* Deployment considerations include:
  + Static/media file handling
  + Backup and recovery
  + Security and performance (e.g., caching, load balancing)
* Future plans for mobile app via Django REST API and cloud storage support.