Project Name: Hotel Management System

Assignment No 2

Group Assignment

Group Member:

21-3881 Muhammad Taha Jawaid

21k-3842 Naresh Kumar

21k-3840 Israr Ali

21k-3841 Ahmad Noor Khan

21k-3825 Sohaib Baig

21k-3871 Mohsan

1. Project Feasibility

Technical Feasibility

- The system will be built using a web-based architecture (HTML, CSS, JavaScript, MySQL, PHP)
- Features include secure login/logout, real-time booking, guest management, billing, laundry, and food service tracking.

Operational Feasibility

- Hotel staff will manage reservations, bills, laundry, and food orders easily from a centralized dashboard.
- Guests can view and manage their bookings through a simplified UI.

Economic Feasibility

- Low-cost development using open-source technologies.
- Reduces the need for manual labor, saving operational costs.
- Increases revenue through better room allocation, digital billing.

Project Vision

To develop a smart, user-friendly Hotel Management System that enables seamless room booking, task tracking, and comprehensive billing operations to enhance guest experiences and hotel staff productivity.

Business Case

Market Demand

- Increasing trend of hotels automating operations post-COVID.
- Many budget and mid-size hotels lack an integrated platform for managing reservations, laundry, food, and billing.

Revenue Streams

- One-time software purchase for hotels.
- Monthly/Annual subscription with premium support.

2. Project Initiation

Agile Charter

1. Executive Summary

The Hotel Management System (HMS) is a digital platform developed to automate and streamline hotel operations including room bookings, guest information management, receptionist activity tracking, and billing. Developed using Agile methodology, this project is aimed at delivering incremental value through iterative development, feedback collection, and continuous improvement. The system provides user authentication, dashboards for operational visibility, and efficient resource management while maintaining data accuracy and billing transparency.

2. Project Purpose and Justification

This project addresses the need for a reliable and efficient system to manage hotel operations. Manual processes often result in operational delays and errors. Automating core activities such as reservations, guest record management, and billing can significantly improve customer satisfaction, ensure transparency, and enhance productivity of hotel staff. The HMS provides secure and non-editable billing, dynamic dashboards, and integrates extra hotel services into guest records, allowing seamless operations.

3. Business Objectives

1. To build a secure login/logout and authentication system.
2. To implement guest and reservation management features with CRUD operations.
3. To calculate and manage receptionist commission based on services and billings.
4. To generate non-editable, automated billing for transparency.
5. To integrate service management for laundry, food, and cabs.

4. Project Description

The HMS application allows hotel receptionists and administrators to manage daily hotel tasks through a centralized web-based platform. The functionalities include guest check-in/check-out, room assignment, reservations, billing automation, and performance tracking. Receptionists receive commissions based on completed reservations and associated service usage. Hotel services can be tracked per reservation. The system ensures that billing is automated and not editable by any user.

5. Objectives and Success Criteria

Sprint	Key Deliverable	Success Criteria
1	Dashboard creation	Login/logout fully functional with operational dashboard displaying basic statistics.
2	Guest, Room & Reservation Modules	Add/update guests and room types; create and edit reservations.
3	Services, Billing & Commission	Services (laundry, cab, food) integrated; billing auto-generated; commission tracking.
4	Testing & Final Release	The system is 90% tested, feedback integrated, and deployed for demonstration.

6. Requirements

- Secure login/logout functionality.
- CRUD operations for guest records and room types.
- Add/edit reservations.
- Add extra services to reservations (e.g. laundry, food, cab).
- Track receptionist commissions.
- Automated, secure billing system.
- Dashboard showing reservations, room types, and performance statistics.

7. Constraints

- Limited time frame due to academic deadlines.
- Restricted to free development tools and platforms.
- No access to real hotel management APIs or datasets.

8. Assumptions

- Team members will contribute equally.
- Scope and requirements will remain stable once development begins.
- Feedback from instructors will be timely and constructive.
- All modules will be developed using open-source or free tools.

9. Preliminary Scope Statement

In Scope:

- Authentication module
- Guest and room management
- Reservation and service tracking
- Dashboard
- Billing and commission system

10. Risks

Risk	Likelihood	Impact	Mitigation Strategy
Time management issues	High	High	Use sprints and regular stand-ups

Technical errors in billing	Medium	High	Use unit testing and peer code reviews
Team member unavailability	Medium	Medium	Reassign tasks or have backup contributors
Misunderstood requirements	Medium	High	Continuous feedback and prototyping

11. Deliverables

- Functional HMS Web Application
- Documentation for each module and user flow
- Final Sprint Review Report
- Demonstration-ready system with valid test cases
- GitHub repository with all commits and tags per sprint

12. Budget Summary

Item	Estimated Cost (PKR)
Internet and Power Sharing	2,000
Software Tools & Design	1,000
Printing and Reports	1,500
Miscellaneous	2,000
Hosting	3,000
Total Estimated Budget	9,500

13. Approval Requirements

- All planned features must be functional.
- Codebase properly documented and uploaded to GitHub.
- Sprint backlog, velocity chart, and sprint review documents must be submitted.
- Approval by mentor/supervisor after successful demonstration.

14. Authorization

This Agile Project Charter is acknowledged and approved by the team members and the assigned supervisor. Final sign-off will be taken at the time of project completion and demonstration.

3. Plan Release 1

Breakdown of Epics

Epic 1: User Authentication & Authorization

- Login and logout functionality with role-based access.

Epic 2: Room Reservation Management

- Add, update, and manage reservations.

Epic 3: Guest Management

- Add and update guest profiles.

Epic 4: Dashboard & Home Screen

- Show user greeting, reservation stats, billing count.

Epic 5: Room Type & Availability

- Define and display room types and availability.

Epic 6: Billing Automation

- Generate immutable bills upon reservation completion.

Epic 7: Service Add-ons Management

- Enable receptionist to add services.

Estimate Stories with Poker Planning

(Team Size: 6 Members)

Story	Story Points
User login/logout implementation	3
Create reservation (basic info + room selection)	5
Update reservation details	3
Add guest profile	3
Update guest profile	2
Dashboard metrics (reservations & bills)	5
Calculate and display available rooms	5
Auto-generate bills upon reservation	8
Add services to bill	5
Track receptionist commissions	5

Create Release Plan

Release 1: Core Functionalities

Login/logout, reservations, guests, billing, room types, dashboard

Release 2: Performance

Receptionist commissions, performance tracking, services, security

4. Iteration 0 (Pre-development Phase)

Architectural Spikes

Frontend and Backend created using the following languages

- PHP
- HTML
- CSS
- Database in MySQL
- JavaScript

Prepare for Iteration 1

Implement basic authentication module

Define user stories & acceptance criteria

Setup project management board

5. Iteration 1 - 4 (Development Sprints)

Sprint 1: Authentication & Core Setup

Stories:

Login/logout

Role-based access

Initial dashboard (user greeting, reservation count)

Activities:

Code & unit tests

Setup navigation bar and user context

Backend login API + session management

Testing:

Auth flow test

Session persistence test

Acceptance testing

Sprint 2: Reservation & Guest Management

Stories:

Add/update reservation
Add/update guest
Track reservation per receptionist

Activities:

Backend model integration
CRUD UI for reservation/guest
Testing update flows

Testing:

UI form validation
Backend integration tests

Sprint 3: Billing & Room Management

Stories:

Define/display room types
Show available rooms
Generate bills automatically
Prevent bill edits

Activities:

Implement room catalog
Room availability logic
Billing microservice

Testing:

Bill generation test
Permission test for editing

Sprint 4: Testing and Final Release

Stories:

Add optional services to bill
Calculate receptionist commission
Count processed bills per receptionist

Activities:

UI service add-on section
Backend for commissions
Final styling + responsiveness

Testing:

Cross-browser testing
Edge cases for service billing

Data integrity testing
UI/UX verification

Conduct Code Reviews

- **Frequency:** Daily or every time a Pull Request (PR) is raised.
- **Tools:** GitHub or GitLab Merge Request Review
- **Checklist:**
    - Code readability and naming conventions.
    - Proper validations and error handling.
    - Test coverage and passing test cases.
    - No hard-coded credentials or values.
- **Involvement:** At least one peer and one senior dev must approve each PR.

Execute Functional & Integration Testing

- **Functional Testing:**
    - Ensure each module performs its intended function (e.g., bill is generated only after reservation).
- **Integration Testing:**
    - Validate interactions between modules (e.g., reservation links correctly with guest and billing).
    - Test service add-ons workflow (e.g., add laundry service → bill updates automatically).

Testing & Validation

Frontend Testing (HTML/CSS/JS)

Manual UI Testing

- **Objective:** Ensure visual elements work as expected and user experience is smooth.
- **Activities:**
    - Check all buttons, forms, and links.
    - Validate layout across different screen sizes (mobile, tablet, desktop).
    - Confirm CSS styles load correctly in all supported browsers.

Cross-Browser Testing

- **Browsers Tested:** Chrome

Functional Frontend Testing Scenarios

- Login & logout buttons redirect appropriately.
- Forms for reservation/guest info submit and clear correctly.
- Feedback for incorrect inputs (e.g., empty fields, invalid formats).

- Room cards display the correct type, price, and image.

Backend Testing (PHP)

Unit Testing in PHP

- **Tool: PHP Unit**
- **Test Examples:**
  - Validate reservation form input before insertion.
  - Ensure bill generation functions return correct totals.
  - Confirm login checks session and credentials properly.

Database Validation

- Manually check your MySQL tables for:
  - Correct insertions (guests, reservations, services).
  - Proper relational integrity (e.g., bill → reservation → guest).
- Use PhpMyAdmin or MySQL CLI.

Acceptance Testing for Each Sprint

- **Each module will be tested based on real-world user flows:**
  - Receptionist logs in and creates a reservation.
  - Receptionist updates service used by the guest, bill updates automatically.
  - System displays count of bills and reservations for the receptionist only.
- **Client UAT Checklist includes:**
  - Screens load properly without errors.
  - Workflows are logical and error-free.
  - Data is secure and accurate (e.g., no bill editing).

Agile Meetings

1. Daily Standups

- **Duration:** 15 minutes max
- **Format:**
  - What was done yesterday?
  - What will be done today?
  - Any blockers?
- **Goal:** Maintain momentum and quickly address obstacles

Iteration Retrospective

- **Timing:** End of every sprint (every 3 weeks)
- **Format:**
  - What went well?
  - What didn’t go well?

    - What can we improve for next sprint?
- **Outcome:** Action items to be implemented in next sprint

6. Project Close-Out

Final Activities

Conduct Final User Acceptance Testing (UAT):

1. UAT Objective

Ensure the hotel reservation system meets the business requirements and is ready for production by validating all major user-facing functionalities.

2. Key Features to Test

Based on your features, the following will be tested:

- Login/Logout
- Booking
- Reservation
- Add Guest
- Dashboard Homepage
- Auto Generate Billing
- Type of Available Rooms

3. UAT Participants

- Front Desk Agent
- Hotel Manager
- IT Support
- Customer (Simulated/Test User)

4. Test Schedule

- Day 1–2: Functional testing (login, booking, dashboard)
- Day 3–4: Scenario-based testing (guest additions, room availability)
- Day 5: Billing and end-to-end flow

5. UAT Entry Criteria

- The system is fully developed and internally tested
- Test data is available (rooms, users, guests)
- All major bugs from internal QA are resolved

6. UAT Exit Criteria

- All critical test cases passed
- Minor issues documented with workarounds
- Sign-off from business users

Address final bug fixes and enhancements:
Bug fixes:

Feature	Issue	Fix Implemented
Login/Logout	Login screen doesn't show error on invalid credentials	Added proper error messaging with retry logic
	Logout button doesn’t always clear session	Ensured session/token invalidation on logout
Booking	Booking button clickable multiple times	Added debounce/throttle to prevent multiple submissions
Reservation	Incorrect time zone on reservation timestamps	Standardized time zone handling across backend/frontend
Add Guest	Form allows submission with missing required fields	Added frontend + backend validation
Billing	Incorrect tax calculation if bill is generated manually	Auto bill generation but receptionist can add food service charges manually
Room Types	Room availability not updated after cancellation	Added real-time room status refresh logic

Enhancements:

Reservaion: Add ability to modify reservation dates.
Billing: Detailed bill breakdown per day and service.
Room Types: Filter by room amenities (e.g., AC, sea view, suite)

Obtain client sign-off:

A formal client sign-off document confirms the client has reviewed the system, verified that it meets the agreed-upon requirements, and approves it for deployment or production use.

Post-Launch support plan:

Phase 1: Monitoring system performance:

Objectives

- Ensure system stability under real user load
- Identify and resolve performance bottlenecks
- Monitor uptime, latency, and resource usage
- Collect usage data to inform future enhancements

Monitoring Workflow

1. Set up monitoring dashboards
2. Monitor logs and system metrics daily
3. Weekly review of performance reports

Phase 2: Gather user feedback and iterate:

Objectives

- Collect qualitative and quantitative feedback from real users
- Identify usability issues, feature gaps, and improvement areas
- Prioritize and implement enhancements for better user experience and performance

Who to Involve

- Front Desk Staff / Receptionists (daily users)
- Hotel Managers
- IT Support / Maintenance
- Customers/Guests (if system has guest-facing elements)

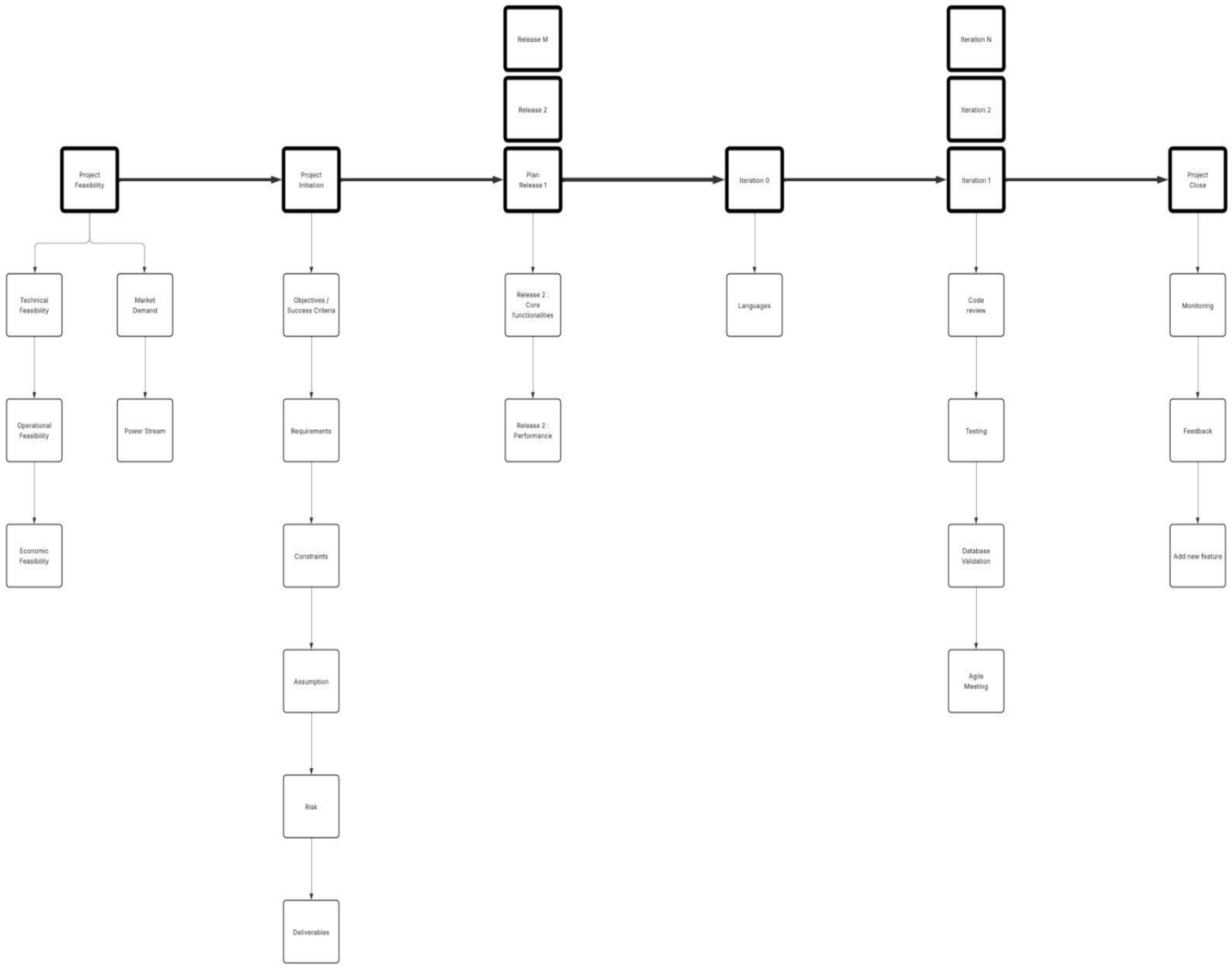
Feedback Iteration Workflow

1. **Collect & Tag Feedback**
   Use tags like usability, performance, bug,
2. **Analyze & Prioritize**
   Apply a simple prioritization model:
   - Urgent + High Impact = Fix Immediately
   - Low Impact + Easy = Quick Win
   - High Effort + Medium Impact = Schedule in future release

Phase 3: Scale and add new features:

Feature	Why Add It
Multi-property Support	For hotel chains or managing multiple branches
Role-Based Access Control	Define permissions (admin, front desk, housekeeping, accounting)
Online Payment Integration	Allow guests to pre-pay with Stripe, PayPal, or local gateways
Mobile App or PWA	Provide better access for staff and guests on mobile devices
Notifications & Alerts	Email/SMS alerts for bookings, cancellations, reminders

Language & Currency Support	For international guests and multi-region hotels



Project Feasibility
Technical Feasibility
Operational Feasibility
Economic Feasibility
Market Demand
Power Stream
Project Initiation
Objectives / Success Criteria
Requirements
Constraints
Assumption
Risk
Deliverables
Release M
Release 2
Plan Release 1
Release 2 : Core functionalities
Release 2 : Performance
Iteration 0
Languages
Iteration N
Iteration 2
Iteration 1
Code review
Testing
Database Validation
Agile Meeting
Project Close
Monitoring
Feedback
Add new feature