Examination Scheduling System

Presented by:

Alyamamah Ahmad Mohammad Alajarmah (32101002079)

Mona Ahmad Mohammed Haniyeh (32101002105)

Supervised by:

Dr.Raed Kaleel

"Submitted in partial fulfillment of the requirements for bachelor's degree. "

2025

Prince Abdullah Bin Ghazi Faculty of Information Technology and Communication

Al-Balqa Applied University

ACKNOWLEDGMENT

Praise be to God, by whose grace good deeds are accomplished, and by whose grace and guidance this work was completed. We extend our sincere thanks and appreciation to everyone who contributed to the completion of this research and its publication in this form.

First, we express our deep gratitude and appreciation to His Excellency Dr. Raed Khalil, the project supervisor, for his valuable guidance, patience, meticulous follow-up, and generous efforts, which were the primary drivers of this achievement.

We also extend our sincere thanks to His Excellency Dr. Osama Dargham, Dean of the College, for his unlimited support and constant commitment to overcoming all difficulties and providing the ideal academic environment that helped us complete this work.

We extend our special thanks and appreciation to His Excellency Dr. Khaled Al-Kharabsheh, Head of the Department, for his academic and administrative support, his valuable comments that enriched this research, and his guidance that helped us develop it.

We must not forget to express our deepest gratitude to our esteemed family for their patience, moral support, and sacrifices, which have been a true asset to our research journey.

We would like to express our sincere gratitude to our supervisors, examiners, and faculty members for their generous presence and supportive efforts, which we greatly appreciate. Thank you for being part of this academic journey.

This achievement represents the culmination of a journey of constructive collaboration and wise guidance, where our minds and hearts joined together to transform an idea into a tangible reality. It is not just a piece of research we present; it is a testament to team spirit and willpower. We see this achievement as a launching pad toward broader horizons, drawing inspiration from it and drawing the determination to face future challenges with confidence and determination, always striving to advance ourselves and serve our community.

DECLARATION

We, the undersigned, proudly present the "**Examination Scheduling System**" as our capstone project, submitted to the distinguished Information Technology Department at Prince Abdullah bin Ghazi College. This endeavor is a testament to our dedicated efforts and pursuit of the Bachelor of Science degree in [Software Engineering].

This comprehensive system represents our original work, meticulously developed under the esteemed guidance of Dr. Raed Khalil. All external sources, concepts, and inspirations have been scrupulously acknowledged through appropriate citations, ensuring academic integrity and transparency.

We solemnly affirm that this project, in its entirety, has not been previously submitted to any other academic institution or university for the conferral of any degree or qualification.

Signed

Alyamamah Alajarmah
(32101002070)

Mona Haniyh
(32101002105)

ABSTRACT

This graduation project (*Examination Scheduling System*)presents a web-based system designed to efficiently manage and organize university course and exam schedules. The system allows administrators to add exams, check for scheduling conflicts, and display organized timetables for both students and faculty. Through the integration of front-end technologies (HTML, CSS, JavaScript) and back-end scripting (PHP), the platform ensures user-friendly interaction and real-time conflict detection. The primary objective is to streamline academic scheduling, minimize human error, and enhance the overall administrative workflow in educational institutions.

Table of Contents

- **Cover Page ..... 1.**
- **Acknowledgment ..... 2.**
- **Declaration ..... 3.**
- **Abstract ..... 4.**

Chapter 1: Introduction ..... 9.

- 1.1 Introduction
- 1.2 Software Development Life Cycle
- 1.3 Extreme Programming Methodology

Chapter 2: Planning Phase ..... 17.

- 2.1 Introduction to Planning Phase
- 2.2 Project Scope and Setting Boundaries

Chapter 3: Requirements Analysis ..... 34.

- 3.1: Introduction to Requirements Analysis Phase
- 3.2 The Sprint3.3 Product Backlog Creation for Project
- 3.4 The Sprint (1)
- 3.5 The Sprint (2)
- 3.6 The Sprint (3)
- 3.7 The Sprint (4)
- 3.8 The Sprint (5)
- 3.9 The Sprint (6)
- 3.10 The Sprint (7)

- 3.11 Burndown Chart
- 3.12 User Story Mapping

Chapter 4: Design Phase ..... 75.

- 4.1 Introduction to Design in XP
- 4.2 Guiding Principles and Techniques in XP Design
- 4.3 Database Design
- 4.4 Architectural Design
- 4.5 User Personas
- 4.6 Conceptual Class Diagram (Inferred)
- 4.7 Use Case Diagram / Key User Stories

. 4.7.1 Key Interaction Sequences

- 4.8 Site Map
- 4.9 High-Fidelity Representation (UI Design)

Future Improvement ..... 104.

References ..... 106.

List of Figures

Figure	1-1	SDLC Phases	10
Figure	1-2	Planning Phases	16
Figure	2-2	Gantt chart	33
Figure	1-3	Requirements analysis	34
Figure	2-3	Extreme Programming (XP)	37
Figure	3-3	Requirements Analysis Process in XP Methodology	40
Figure	4-3	User Story Structure	47
Figure	5-3	User Story Card for Administrator Authentication	50
Figure	6-3	User Story Card for Department Head Authentication	51
Figure	7-3	User Story Card for Department Management	52
Figure	8-3	The Planning Game in XP	53
Figure	9-3	User Story Card for Course Management	55
Figure	10-3	User Story Card for Examination Creation	58
Figure	11-3	User Story Card for System Configuration	64
Figure	12-3	Burndown Chart	69
Figure	13-3	User Story Map	71
Figure	14-3	Requirements Validation approaches	72
Figure	1-4	design phase	74
Figure	2-4	ER-diagram	82
Figure	3-4	Persona 1	85
Figure	4-4	Persona 2	86
Figure	5-4	Class Diagram	90
Figure	6-4	Use Case Diagram	91
Figure	7-4	Admin Login Sequence	94
Figure	8-4	Schedule Exam Sequence	95
Figure	9-4	Site Map	96

List of Tables

Table	1-2	Time schedule	
Table	2-2	Theme vs Epic vs User story	

Chapter 1: Introduction

1.1: Introduction

This project introduces an enhanced Examination Management System designed for Al-Balqa Applied University, specifically supporting the Prince Abdullah bin Ghazi Faculty of Information and Communication Technology. The primary objective of this system is not merely to automate, but to significantly improve the efficiency of examination scheduling. A key focus is enabling the possibility of scheduling multiple examinations on the same day, thereby helping to shorten the overall duration of the examination period. By leveraging web technologies (PHP, HTML, CSS, JavaScript) and a centralized MySQL database, the system offers a robust platform to achieve this optimization.

The system provides distinct functionalities for Administrators and Department Heads. Administrators manage foundational data like departments, courses, instructors, and classroom details. Department Heads can schedule exams for their departments, specifying dates, times, types (Midterm, Final), courses, instructors, and rooms. Crucially, the system incorporates an automated conflict detection mechanism. This feature is essential for the goal of schedule compression, as it meticulously checks for potential issues like double-booked instructors or classrooms, or conflicting exams for related courses, ensuring that multiple exams can be scheduled per day without logistical problems.

This Examination Management System aims to enhance the existing scheduling process by providing tools for greater efficiency and optimization. By facilitating the scheduling of multiple exams per day through reliable conflict checking, it seeks to significantly shorten the examination period, reduce administrative burden associated with complex scheduling, minimize errors, and improve resource utilization within the Prince Abdullah bin Ghazi Faculty of Information and Communication Technology at Al-Balqa Applied University. This contributes to a more streamlined and efficient examination process for the benefit of administrators, faculty, and students.

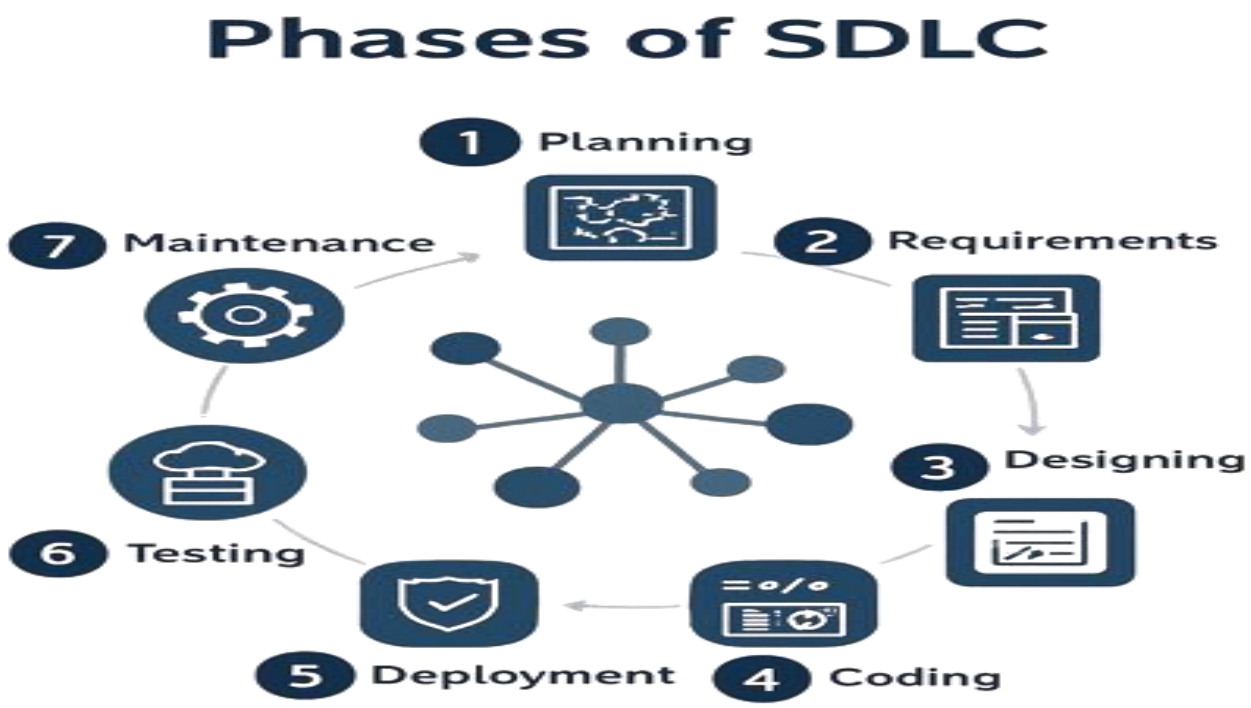


Figure 1-1: SDLC Phases

1.2: Software DEVELOPMENT Life Cycle

The Software Development Life Cycle (SDLC) provides a structured framework for developing the Examination Management System, ensuring a methodical approach from conception to deployment and maintenance. The process begins with the **Planning Phase**, where the project's scope, objectives, and feasibility are defined. For this system, the goal was to automate exam scheduling in academic institutions, addressing inefficiencies in manual processes. Feasibility analysis considered

technical, operational, and resource factors, leading to the decision to develop a web-based application for administrators and department heads.

Next, the **Requirements Analysis Phase** involves gathering detailed functional and non-functional requirements. Stakeholders like administrators and department heads were identified, along with their needs, such as managing courses, instructors, classrooms, and detecting scheduling conflicts. The database schema and ER diagram were key outputs of this phase, defining the system's data structure.

The **Design Phase** translates requirements into a technical blueprint, including system architecture, database design, and user interface. The Examination Management System adopted a three-tier web architecture: presentation (HTML, CSS, JavaScript), application logic (PHP), and data (MySQL). The database design outlined tables like Admin, Exam, and Instructor, while UI design focused on intuitive navigation and forms.

In the **Implementation Phase**, developers wrote the actual code, creating PHP scripts for backend logic (e.g., conflict detection, database interactions) and frontend interfaces using HTML, CSS, and JavaScript. The database was built using SQL scripts, bringing the design to life.

"After the system goes through the main phases of **(SDLC**), the new model of the system becomes more efficient, reliable, and user-friendly, and the developers can maintain it in an easy way. It aims to reduce maintenance costs in the long term, which enhances the user experience and increases security."

1.3: Extreme Programming (XP)

Extreme Programming (XP) is an Agile software development methodology that emphasizes customer satisfaction, rapid feedback, and high-quality code through disciplined engineering practices. Unlike traditional approaches, XP focuses on frequent releases, continuous improvement, and adaptability to changing requirements. It is particularly effective for projects with dynamic or unclear requirements, making it suitable for fast-paced development environments.

The specific development methodology employed for the Examination Management System project is not explicitly documented within the provided files. However, based on the nature of academic projects, the structure of the codebase (separation of concerns with HTML, CSS, PHP, SQL), and the typical workflow for developing such systems, it is plausible to infer that an **Iterative Development Model** was likely

followed, possibly incorporating elements of prototyping, especially for the user interface components.

An Iterative Development model contrasts with a strict Waterfall approach by breaking down the software development process into smaller, repeated cycles or iterations. Each iteration builds upon the previous one, incorporating feedback and refinements, and resulting in a functional subset of the final system. This approach is well-suited for projects where requirements might evolve or become clearer as development progresses, which is often the case in academic settings where students are learning and refining their understanding simultaneously.

Here's how an Iterative approach likely guided the SDLC phases previously discussed for the Examination Management System:

- **Initial Planning and Requirements**: The core requirements (managing exams, courses, rooms, instructors, basic conflict avoidance) would be identified initially, similar to the first two SDLC phases. However, unlike a rigid Waterfall model, not all requirements needed to be exhaustively detailed upfront.
- **Iteration 1 (Core Functionality)**: The first iteration might focus on setting up the basic database structure (perhaps involving tables for Departments, Instructors, Courses, Classrooms) and implementing core CRUD (Create, Read, Update, Delete) operations for these entities. This would involve creating the initial database schema (examinationsystem1.sql), developing the

corresponding PHP scripts for backend logic, and building simple HTML forms and display pages. This produces a basic, functional skeleton of the system.examinationsystem1.sql

- **Iteration 2 (Exam Scheduling & Basic UI)**: Building on the first iteration, the next cycle could introduce the Exam table and the functionality to schedule exams (AddExam.html, process_exam.php). The user interface for department heads (department-Heads.html, depheadlogin.php) might be developed or refined during this stage. Early versions of exam lists (exams_list.php) could also be implemented.

- **Iteration 3 (Conflict Detection)**: With core scheduling in place, a subsequent iteration would focus on the more complex logic for conflict detection (check_conflict.php, conflicts.php, store_non_conflicting.php). This involves refining the database queries and PHP logic to identify clashes based on instructor availability, room bookings, and potentially time overlaps. The UI would be updated to display these conflicts (conflicts.html).

- **Further Iterations (Refinement, Testing, UI Polish)**: Later iterations would involve refining existing features based on testing and feedback, improving the user interface and user experience (polishing CSS like AddExam.css, Courses.css, etc.), enhancing security aspects (like password handling in login.php), and performing more comprehensive testing across the integrated system. The inclusion of separate login pages (loginpage.html, depheadlogin.html) suggests distinct user roles were considered and likely implemented iteratively.

This iterative process allows for flexibility. For instance, the exact rules for conflict detection might be refined over several iterations as the complexity is better understood. Similarly, the user interface could be prototyped and adjusted based on usability feedback during development. While it follows the general sequence of the SDLC phases (planning -> analysis -> design -> implementation -> testing), these phases are revisited in each iteration for specific parts of the system.

Compared to a strict Waterfall model, where each phase must be fully completed before the next begins, the Iterative model allows for parallel work streams within an iteration (e.g., designing, coding, and testing a specific feature) and provides opportunities to adapt to challenges or new insights discovered during development. This makes it a practical and common choice for student projects, enabling the development of a functional system incrementally while managing the learning curve and project constraints effectively.

Chapter 2: Planning Phase

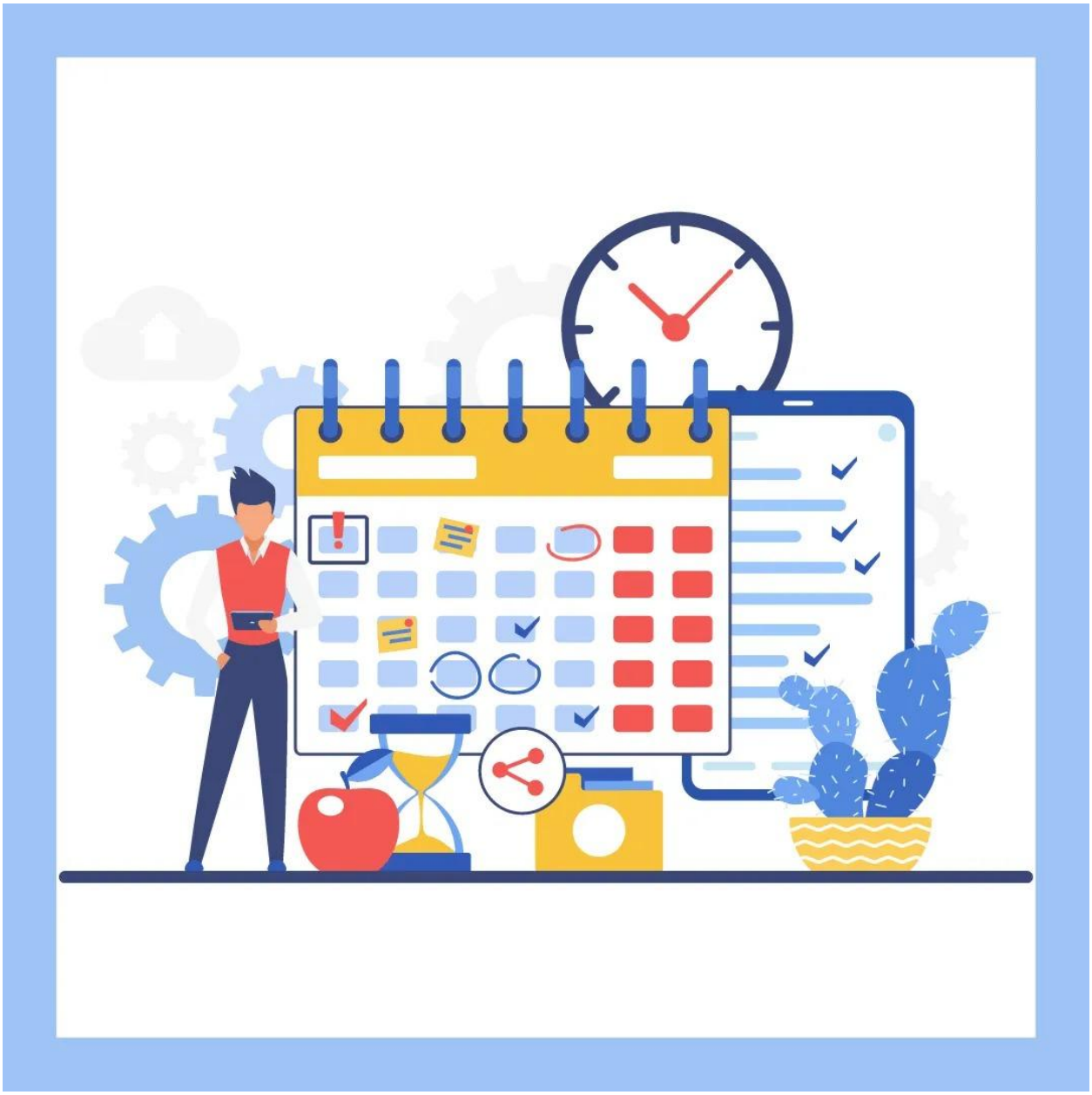


Figure 1_2: Planning Phases

2.1 Introduction to Planning Phase

The journey of developing any successful software system commences with a crucial and foundational stage: the Planning Phase. This initial phase within the

Software Development Life Cycle (SDLC) serves as the bedrock upon which all subsequent development activities are built. It is during planning that the project's vision is conceptualized, its viability is assessed, and a strategic roadmap is formulated to guide the development team towards achieving the desired objectives. For the Examination Management System project, undertaken as part of a graduation requirement, the planning phase holds particular significance. It involves not only defining the technical parameters but also aligning the project with academic goals, understanding the specific problems within the institutional context that the system aims to solve, and establishing a clear scope to ensure the project remains manageable within the given timeframe and resources.

Effective planning involves a systematic process of identifying the core problem, defining clear and measurable objectives, determining the project's scope and boundaries, assessing potential risks, estimating required resources (including time and effort), and outlining the initial project schedule. It translates the abstract idea of an automated examination scheduling system into a concrete plan of action. This phase requires careful consideration of various factors, including the needs of the end-users (such as department heads and administrators), the technical constraints and opportunities, and the overall strategic goals the system intends to support within the academic institution. Neglecting or rushing the planning phase often leads to scope creep, budget overruns, missed deadlines, and ultimately, a final product that fails to meet the intended requirements or deliver the expected value. Therefore, dedicating adequate time and effort to meticulous planning is indispensable for mitigating risks and ensuring the successful development and implementation of the Examination Management System.

2.2 Project Scope and Setting Boundaries

Defining a clear project scope and establishing firm boundaries are fundamental activities within the planning phase, crucial for managing expectations, controlling

project resources, and ensuring the development effort remains focused on achieving the core objectives. The scope outlines the specific features, functionalities, deliverables, and limitations of the project. For the Examination Management System, setting the scope involves explicitly stating what the system will encompass and, just as importantly, what it will exclude, thereby preventing scope creep – the uncontrolled expansion of project requirements beyond the initial agreement.

The primary scope of the Examination Management System project is to design, develop, and implement a web-based application that automates the process of scheduling academic examinations and managing related resources within an educational institution, specifically focusing on the needs of department heads and potentially administrators. This includes:

- **Resource Management:** The system will provide functionalities for managing core entities essential for exam scheduling. This includes maintaining records of academic departments, courses offered (with details like course number, name, credit hours), instructors (with their department affiliations), and classrooms (including room number, type, and capacity).

- **Exam Scheduling:** The system will allow authorized users (primarily department heads) to create, view, update, and potentially delete examination schedules. This involves capturing key details for each exam, such as the associated course, assigned instructor, designated classroom, date, start time, end time, and exam type (e.g., Midterm, Final).

- **Conflict Detection:** A core functional requirement within the scope is the implementation of a mechanism to automatically detect and flag potential scheduling conflicts. The system will check for common conflicts, such as

double-booking of classrooms, assigning the same instructor to multiple exams occurring simultaneously, and potentially other institution-specific constraints if defined (though the provided files primarily suggest instructor and room conflicts).

- **User Roles and Access Control:** The system will implement basic role-based access control, distinguishing between different user types (e.g., Department Heads with access to their department's exams and potentially an Administrator role with broader oversight, although the admin role seems less developed in the provided files compared to the department head role).

- **User Interface:** The scope includes developing a functional web-based user interface (UI) that allows users to interact with the system effectively for performing the tasks mentioned above. This involves creating forms for data entry, tables for displaying information, and navigation menus.

Setting boundaries is equally critical. The following aspects are considered **out of scope** for the current phase of this graduation project:

- **Student Interface/Access:** The system, in its current defined scope, does not include features for student access. Students will not be able to log in, view their personalized exam schedules, or register for exams through this system.

- **Online Examination Conduct:** This system is purely for scheduling and logistical management. It does not include functionalities for creating, administering, or grading exams online.

- **Integration with Other University Systems:** Direct, real-time integration with other institutional systems (like Student Information Systems (SIS), Human Resources systems, or university-wide timetabling systems) is not part of the current scope. Data like course lists or instructor details are assumed to be managed within this system or imported manually/via batch process.

- **Advanced Reporting and Analytics:** While basic lists of exams and conflicts will be available, sophisticated reporting features or advanced analytics on exam data are considered outside the initial scope.

- **Mobile Application:** The development is focused on a web-based application accessible via standard browsers; a dedicated native mobile application is not included.

By clearly defining these inclusions and exclusions, the project maintains a manageable focus, allowing for the successful delivery of core functionalities within the typical constraints of a graduation project. This defined scope provides a clear target for the subsequent design, implementation, and testing phases.

2.2.1 Business Profile

The Examination Management System is designed to operate within the specific context of an academic institution, such as a university or college. The core "business" of such an institution revolves around providing education, fostering research, and managing the academic progression of its student body. A critical component of the academic process is the assessment of student learning, primarily conducted through examinations scheduled at regular intervals, typically mid-term and final exams, as indicated by the system's data structure. The administration and scheduling of these examinations represent a significant operational function that directly impacts students, faculty, and administrative staff.

The effective management of examinations is vital for maintaining academic integrity, ensuring fairness, and facilitating the smooth execution of the academic calendar. This process involves coordinating numerous variables: a diverse range of courses across multiple departments (like Computer Science, Software Engineering, Computer Information Systems, as seen in the database), a large pool of instructors with varying schedules and assignments, a finite number of physical classrooms with different capacities and equipment (labs, lecture halls), and specific time constraints dictated by the academic calendar and institutional policies. Key stakeholders in this process include Department Heads, who are often responsible for coordinating exams within their respective departments; Instructors, who prepare and invigilate exams; Administrators, who may oversee the overall scheduling process and resource allocation; and Students, who are the ultimate recipients of the examination schedule and whose academic progress depends on its timely and conflict-free execution.

Traditionally, many academic institutions rely on manual or semi-automated methods for exam scheduling, often involving spreadsheets, paper-based systems, or disparate departmental tools. These methods are frequently inefficient, labor-intensive, and highly susceptible to errors. Common challenges include scheduling conflicts (e.g., an instructor or a room being assigned to multiple exams simultaneously), inefficient use of classroom space, difficulties in communicating schedules accurately and promptly, and significant administrative overhead in managing changes and resolving issues. These inefficiencies not only consume valuable time and resources but can also lead to frustration among staff and students, potentially disrupting the examination period. The Examination Management System project directly addresses these operational pain points by proposing a centralized, automated solution designed to streamline the scheduling process, minimize conflicts, optimize resource utilization, and improve overall efficiency within this specific operational domain of the academic institution

2.2.2 Definition & Description of the App The Examination Management System is defined as a specialized web-based software application designed to automate and manage the complex logistics of scheduling academic examinations within an educational institution. It serves as a centralized platform accessible through standard web browsers, intended primarily for use by administrative staff and department heads responsible for overseeing or coordinating the examination process. The application integrates database management, server-side processing, and a user-friendly interface to provide a comprehensive solution for handling exam-related data and workflows.

At its core, the application functions as an information system focused on the entities crucial to exam scheduling: courses, instructors, classrooms, departments, and the exams themselves. It replaces manual tracking and coordination efforts with a structured digital environment. The system is built upon a relational database (likely MySQL, given the PHP context) that stores detailed information about these entities and their relationships, as meticulously defined in the database schema and ER diagram. This structured data storage is fundamental to the system's ability to manage information consistently and perform complex checks, such as conflict detection.

The application's architecture follows a typical multi-tier model common for web applications. A presentation tier, constructed using HTML, CSS, and potentially JavaScript, provides the graphical user interface (GUI) through which users interact with the system. This includes forms for inputting data (e.g., adding a new exam schedule), tables and lists for displaying information (e.g., viewing existing exams or available classrooms), and navigational elements. The application logic tier, implemented using a server-side scripting language like PHP, handles the core processing tasks. This includes receiving user requests from the browser, validating input data, interacting with the database to retrieve or store information, implementing the business rules (most notably the conflict detection algorithms),

and preparing the data to be sent back to the presentation tier. The data tier consists of the relational database management system (RDBMS) that persistently stores and manages all the application data.

In essence, the Examination Management System acts as a dedicated digital assistant for exam administrators and coordinators. It aims to simplify the tasks of inputting exam requirements, allocating appropriate resources (instructors and rooms), visualizing the schedule, and proactively identifying and highlighting potential scheduling conflicts before they disrupt the examination process. By providing a single, shared platform, it enhances coordination between departments and ensures that scheduling decisions are based on accurate, up-to-date information regarding resource availability and existing commitments.

2.2.3 The Purpose of the Product The fundamental purpose of developing the Examination Management System is to address the inherent complexities and inefficiencies associated with the manual or semi-automated scheduling of academic examinations within educational institutions. The system is conceived as a targeted solution designed to streamline administrative workflows, enhance operational efficiency, and improve the overall reliability and integrity of the examination process. Its primary goals are multi-faceted, aiming to deliver tangible benefits to various stakeholders, particularly administrative staff and department heads tasked with the demanding responsibility of coordinating exam logistics.

First and foremost, the system aims to significantly reduce the administrative burden and time commitment required for scheduling exams. By automating tasks such as data entry, resource allocation checks, and conflict detection, it frees up valuable staff time that would otherwise be spent on tedious manual coordination, cross-referencing spreadsheets, and resolving scheduling clashes through lengthy communication chains. This automation allows personnel to focus on higher-value activities and ensures the scheduling process is completed more swiftly and efficiently.

Secondly, a critical purpose of the product is to minimize, if not eliminate, scheduling errors and conflicts. Manual systems are notoriously prone to human error, leading to situations like double-booking classrooms, assigning instructors to overlapping exams, or scheduling exams for courses outside designated time slots. Such errors can cause significant disruption and stress for both faculty and students. The Examination Management System, with its built-in validation rules and automated conflict-checking algorithms (as suggested by files like check_conflict.php), serves the purpose of proactively identifying and preventing these issues, thereby ensuring a smoother and more reliable examination period.

Thirdly, the system is intended to optimize the utilization of institutional resources, specifically classrooms and instructor time. By providing a clear, centralized view of resource availability and commitments, the application enables schedulers to make more informed decisions, ensuring that classrooms of appropriate size and type are allocated effectively and that instructor workloads are managed appropriately during the exam period. This efficient resource allocation contributes to cost-effectiveness and better operational management.

Finally, the purpose extends to improving communication and coordination among different academic units and stakeholders involved in the examination process. By providing a single, authoritative source of information for exam schedules and resource assignments, the system facilitates transparency and reduces the likelihood of miscommunication or reliance on outdated information. This centralized approach ensures that all relevant parties, particularly department heads, have access to consistent and accurate data, fostering better collaboration and planning. In essence, the purpose of the Examination Management System is to transform the traditionally challenging task of exam scheduling into a more manageable, accurate, and efficient process, ultimately contributing to the smoother operation of the academic institution and a less stressful experience for everyone involved.

2.2.4 Strategic Planning and SWOT Analysis

Strategic planning for the Examination Management System involves aligning its development and implementation with the broader operational goals of the host academic institution, primarily focusing on improving administrative efficiency and the reliability of the examination process. While a graduation project might not involve the same level of extensive strategic planning as a large commercial venture, conducting a SWOT analysis (Strengths, Weaknesses, Opportunities, Threats) is a valuable exercise. It helps to understand the project's internal capabilities and limitations, as well as the external factors that could influence its success and adoption within the institutional environment.

Strengths: Internally, the project possesses several key strengths. Its primary strength lies in directly addressing a well-defined and persistent operational challenge faced by many academic institutions: the complexity and error-proneness of manual exam scheduling. By automating this process, the system offers a clear value proposition in terms of potential time savings for administrative staff and reduced scheduling conflicts. The focused scope, deliberately defined to be manageable within a graduation project timeframe, is another strength, increasing the likelihood of successful completion of core functionalities. Furthermore, the utilization of widely adopted and accessible web technologies like PHP, MySQL, HTML, and CSS represents a strength in terms of technical feasibility, availability of development resources, and ease of potential deployment on standard institutional web infrastructure. The system's design as a centralized platform inherently promotes better coordination compared to disparate manual methods.

Weaknesses: Despite its strengths, the project also has internal weaknesses, largely stemming from the constraints typical of an academic project. The intentionally limited scope, while a strength for feasibility, is also a weakness when viewed against comprehensive commercial solutions; the absence of features like a student interface, direct integration with other university systems (like SIS), or advanced reporting capabilities limits its overall utility in a fully integrated institutional context. The quality

of the user interface and overall user experience might be a potential weakness if not given sufficient attention during development, potentially impacting user adoption. The system's reliance on initial manual data entry for courses, instructors, and rooms could be cumbersome for institutions with large datasets. Additionally, aspects like security hardening and performance optimization might be less robust compared to enterprise-grade software, potentially requiring further work before widespread, critical deployment.

Opportunities: Externally, the project taps into significant opportunities. There remains a considerable opportunity within the educational sector, as many institutions, particularly smaller ones or specific departments within larger ones, may still rely on outdated or inefficient methods for exam scheduling. This presents an opportunity for the system, even in its current form, to offer a substantial improvement. Looking ahead, the system presents numerous opportunities for future expansion and enhancement. Functionality could be added iteratively, such as incorporating student access, developing mobile interfaces, integrating with SIS for automated data synchronization, or adding more sophisticated analytical reporting features. Successful implementation provides an opportunity to significantly improve the satisfaction of administrative staff and faculty by reducing workload and stress associated with scheduling. It also creates an opportunity to gather valuable data on resource utilization (e.g., classroom usage patterns) that could inform better long-term planning.

Threats: Finally, the project faces external threats that need consideration. A primary threat is potential resistance to change from administrative staff who are accustomed to existing procedures, however inefficient they may be. Overcoming this inertia might require effective communication, training, and demonstrating clear benefits. The existence of established commercial Examination Timetabling or broader Academic Management software poses a competitive threat, as these systems often offer a wider range of features and dedicated support, although they typically come at a higher cost. Changes in institutional IT policies, budget constraints, or strategic directions could pose a threat to the adoption, deployment, or long-term maintenance

of the system. Ensuring data privacy and compliance with relevant regulations is crucial, and failure to do so represents a significant threat. Lastly, like any software project, it faces technical threats such as unforeseen development challenges, compatibility issues with server environments, or difficulties in securing ongoing maintenance and support after the initial project completion.

2.2.5 App's Requirements To ensure the Examination Management System effectively fulfills its purpose and meets the needs of its intended users, a clear set of requirements must be defined during the planning phase. These requirements encompass both the specific functions the system must perform (functional requirements) and the quality attributes or constraints under which it must operate (non-functional requirements). Based on the project's scope, objectives, and the analysis of the provided system components, the key requirements can be articulated as follows.

Functionally, the system must provide robust capabilities for user authentication and authorization. Specifically, it needs to allow designated users, primarily Department Heads, to securely log into the system using unique credentials (User ID and Password), validating these against stored records. Once authenticated, users should only have access to functionalities pertinent to their role. A core functional area is resource management, which necessitates providing interfaces for authorized users to perform Create, Read, Update, and Delete (CRUD) operations on essential entities. This includes managing lists of courses offered within departments, maintaining records of instructors and their departmental affiliations, and administering details of available classrooms, including their capacity and type (e.g., lab, lecture hall), as indicated by pages like Courses.php and Rooms.php. The most critical functional requirement revolves around examination scheduling itself. The system must empower Department Heads to schedule new examinations by inputting comprehensive details such as the course, instructor, classroom, date, start and end times, and the type of exam (Midterm or Final), facilitated through interfaces like AddExam.html and processed by scripts like process_exam.php. Complementing creation, users must be able to view lists of already scheduled exams (exams_list.php) and possess the ability to modify or remove existing exam entries

(delete_exam.php). Integral to scheduling is the automated conflict detection mechanism. The application is required to intelligently check for and prevent common scheduling conflicts upon the creation or modification of an exam. This involves verifying that the assigned instructor is not already scheduled for another exam during the same time slot and ensuring the selected classroom is available and not double-booked. The system must clearly present any detected conflicts to the user, enabling them to make necessary adjustments, supported by logic potentially found in check_conflict.php and displayed via conflicts.html. Beyond these core functions, several non-functional requirements define the system's quality attributes. Usability is paramount; the user interface must be intuitive, straightforward, and easy to navigate for administrative staff who may not possess advanced technical skills. Clear form layouts, logical workflow, and informative feedback, including error messages, contribute to this requirement. Performance is another key consideration; the system should exhibit reasonable responsiveness, particularly during potentially resource-intensive operations like conflict checking, ensuring users do not experience undue delays. Reliability is crucial for an administrative system of this nature; data must be stored accurately and retrieved consistently, and the conflict detection logic must function correctly to be trustworthy. The system should maintain high availability, especially during peak scheduling periods. Security requirements include protecting user accounts via password authentication (ideally with secure password storage mechanisms) and enforcing role-based access control to prevent unauthorized data access or modification. Finally, maintainability, supported by well-structured code and clear separation of concerns (as suggested by the distinct HTML, CSS, and PHP files), is important for potential future updates or troubleshooting, while basic scalability considerations should allow the system to handle the volume of data typical for a department or faculty within an academic institution.

2.2.6 Performing Project Estimation Project estimation is a vital component of the planning phase, involving the prediction of the time, effort, and resources required to complete the project successfully. While estimation in a commercial setting often involves complex models and financial considerations, within the context of a

graduation project like the Examination Management System, the primary focus is typically on estimating the effort required (usually measured in person-hours or calendar time) and ensuring the project scope is achievable within the academic timeframe and with the available resources (primarily the student developer's time, access to development tools, and potentially institutional IT resources like web hosting for testing or deployment). Estimating the effort for the Examination Management System involves breaking down the overall project into smaller, manageable tasks derived from the defined requirements. These tasks would encompass activities across the SDLC phases, including detailed requirements refinement, database design and implementation, backend development (PHP scripting for CRUD operations, session management, conflict logic), frontend development (HTML structure, CSS styling, potential JavaScript interactions), unit and integration testing, documentation writing (like this report), and potentially setting up a demonstration environment. Each identified task is then assessed for complexity and the estimated time required for completion, considering the developer's familiarity with the required technologies (PHP, MySQL, HTML, CSS) and the inherent challenges of specific components, such as implementing the conflict detection logic. Given that this is an academic project, the estimation process must also account for factors like the student's learning curve, concurrent academic commitments (other courses, exams), and the fixed deadlines imposed by the academic calendar for project submission and defense. Therefore, estimations often incorporate buffer time to accommodate unforeseen challenges or delays. The resources required are generally modest, including a development computer, standard software development tools (text editor/IDE, web browser, local server stack like XAMPP or WAMP), and access to documentation and online resources. The primary resource constraint is the developer's available time and effort. The goal of estimation in this context is less about precise budget forecasting and more about validating the feasibility of the project scope against the available time and ensuring a realistic plan can be formulated. It helps in identifying potential bottlenecks early on and allows for adjustments to the scope or plan if necessary. The outputs of this estimation process typically inform the creation of a project schedule, often visualized using tools like a Gantt chart, which outlines the timeline for completing the various tasks and milestones. 2.2.6.1 Time Constraints for the Examination Management

System Project A critical aspect of project estimation, particularly within an academic setting, is the explicit recognition and management of time constraints. Unlike commercial projects that might have flexible timelines or the possibility of extension based on budget, graduation projects operate under rigid temporal boundaries dictated by the academic calendar. The Examination Management System project is subject to these inherent constraints, which significantly influence its planning, scope, and execution strategy. The most overarching time constraint is the fixed duration allocated for the graduation project, typically spanning one or two academic semesters. This period encompasses all project phases, from initial planning and requirements gathering through design, implementation, testing, documentation, and final presentation or defense. Specific deadlines are usually imposed by the institution or department for key milestones, such as the submission of a project proposal, interim progress reports, final report submission, and the final project demonstration or defense. These non-negotiable deadlines create critical pressure points throughout the project lifecycle and necessitate careful scheduling to ensure timely completion of deliverables. Furthermore, the actual development time available to the student developer is constrained by concurrent academic responsibilities. Students undertaking graduation projects are typically enrolled in other courses, preparing for regular examinations, and managing other academic or personal commitments. This means that the total calendar time allocated for the project does not translate directly into full-time development effort. Instead, development work must be scheduled around these competing demands, effectively reducing the number of dedicated hours available per week. This constraint underscores the importance of efficient time management, realistic task estimation, and potentially prioritizing core functionalities to ensure a viable product is delivered within the limited timeframe. These time constraints necessitate a pragmatic approach to project planning. The scope must be carefully managed to be achievable within the available period, and the project schedule, often visualized in a Gantt chart, must realistically reflect the deadlines and the part-time nature of the development effort. Contingency planning for potential delays, whether due to technical challenges or unforeseen interruptions in the student's schedule, also becomes crucial within this constrained temporal framework. Effectively navigating these time constraints is

essential for the successful completion of the Examination Management System project.

2.2.6.1: Time Constraints for Examination Management System*:*

Phase	Start Date	End Date	Duration (days)
Requirements Analysis	2025-02-03	2025-02-21	18
Design	2025-02-24	2025-03-07	11
Development - Phase 1	2025-03-10	2025-03-21	11
Development - Phase 2	2025-03-24	2025-04-04	11
Development - Phase 3	2025-04-07	2025-04-18	11
Integration & Delivery	2025-04-21	2025-04-30	9

Table 1.2: Time schedule

2.2.6.2 Gantt Chart

A Gantt chart serves as a visual representation of the project schedule, illustrating the sequence, duration, and dependencies of tasks across the project timeline. For the Examination Management System project, the Gantt chart provides a clear visualization of how the development activities are distributed across the academic semester, helping to ensure that all necessary tasks are completed within the available timeframe and that critical deadlines are met. The chart divides the project into distinct phases, each represented by a different color for easy identification: Planning & Requirements (grey), Design (blue), Implementation (green), Testing (pink), Documentation (yellow), and Submission & Defense (salmon). Within each phase, specific tasks are scheduled with defined start and end dates, creating a comprehensive roadmap for the project's execution. The Planning & Requirements phase, scheduled for the first three weeks of January 2025, encompasses the initial project setup, detailed requirements gathering, and preliminary estimation. This is followed by the Design phase in early February, focusing on database design and system architecture. The Implementation phase, the most extensive, spans from

mid-February to late March, covering database setup, user authentication, the core exam scheduling module, and the critical conflict detection logic. Testing activities are scheduled for late March through early April, ensuring thorough validation of the system's functionality. Documentation and preparation for deployment run concurrently in mid-April, leading to the final submission and project defense at the end of April. This structured timeline ensures that adequate time is allocated to each critical aspect of the project, with particular emphasis on the implementation of core functionalities like exam scheduling and conflict detection. The chart also visually highlights the project's critical path and potential bottlenecks, allowing for proactive management of time constraints. By adhering to this schedule, the project aims to deliver a functional Examination Management System that meets the defined requirements within the academic timeframe.

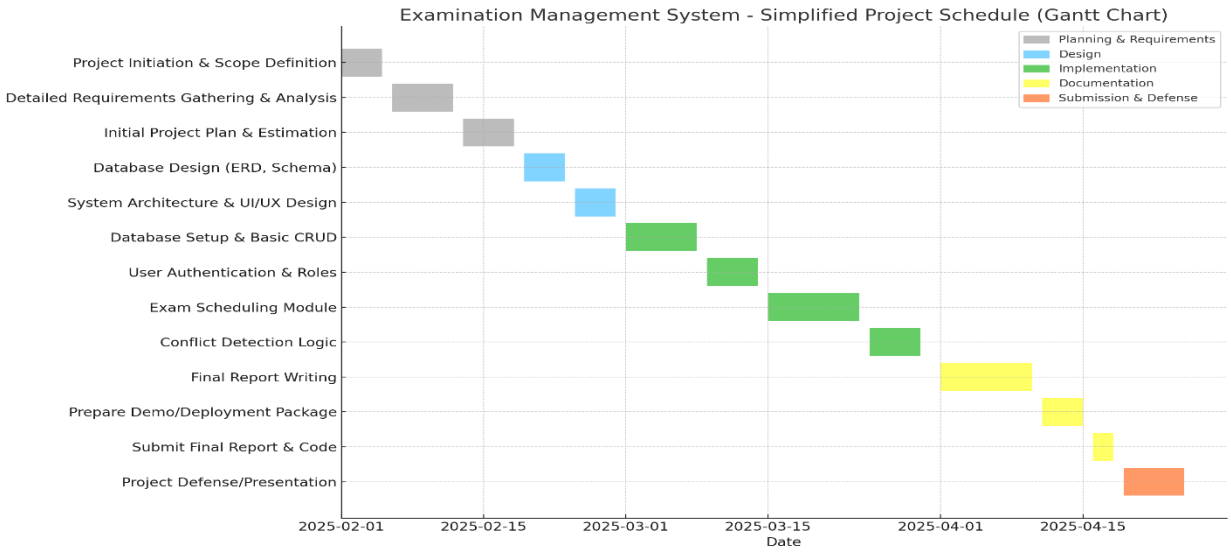


Figure 2_2: Gantt chart

Chapter3: Requirements analysis

Figure 1-3: Requirements analysis

3.1: Introduction to Requirements Analysis Phase

Requirements analysis is a critical phase in the software development lifecycle, particularly when employing the Extreme Programming (XP) methodology. This phase serves as the foundation upon which the entire Examination Management System is built. Unlike traditional methodologies that emphasize comprehensive documentation upfront, XP approaches requirements analysis as an iterative and collaborative process that evolves throughout the project lifecycle.

The requirements analysis phase for the Examination Management System focused on identifying, clarifying, and prioritizing the functional and non-functional requirements through continuous stakeholder collaboration. This approach aligns with XP's core values of communication, simplicity, feedback, courage, and respect, ensuring that the development team maintains a clear understanding of what needs to be built while remaining adaptable to changing requirements.

In the context of our Examination Management System, the requirements analysis phase was characterized by:

1. **Direct stakeholder involvement:** Department heads, administrators, instructors, and students participated in requirements gathering sessions, providing immediate feedback on proposed features.

2. **Incremental requirements definition:** Rather than attempting to define all requirements at the outset, requirements were identified and refined incrementally throughout the development process.

3. **User story-driven development:** Requirements were captured as user stories, focusing on the user's perspective and the value delivered by each feature.

4. **Test-first approach:** Acceptance criteria were defined before implementation, establishing clear expectations for when a requirement is considered complete.
5. **Continuous validation:** Requirements were continuously validated through frequent demonstrations, ensuring alignment with stakeholder expectations.

This chapter details the requirements analysis process employed for the Examination Management System, including the XP methodology's specific practices, the creation and management of user stories, and the planning and execution of development iterations.

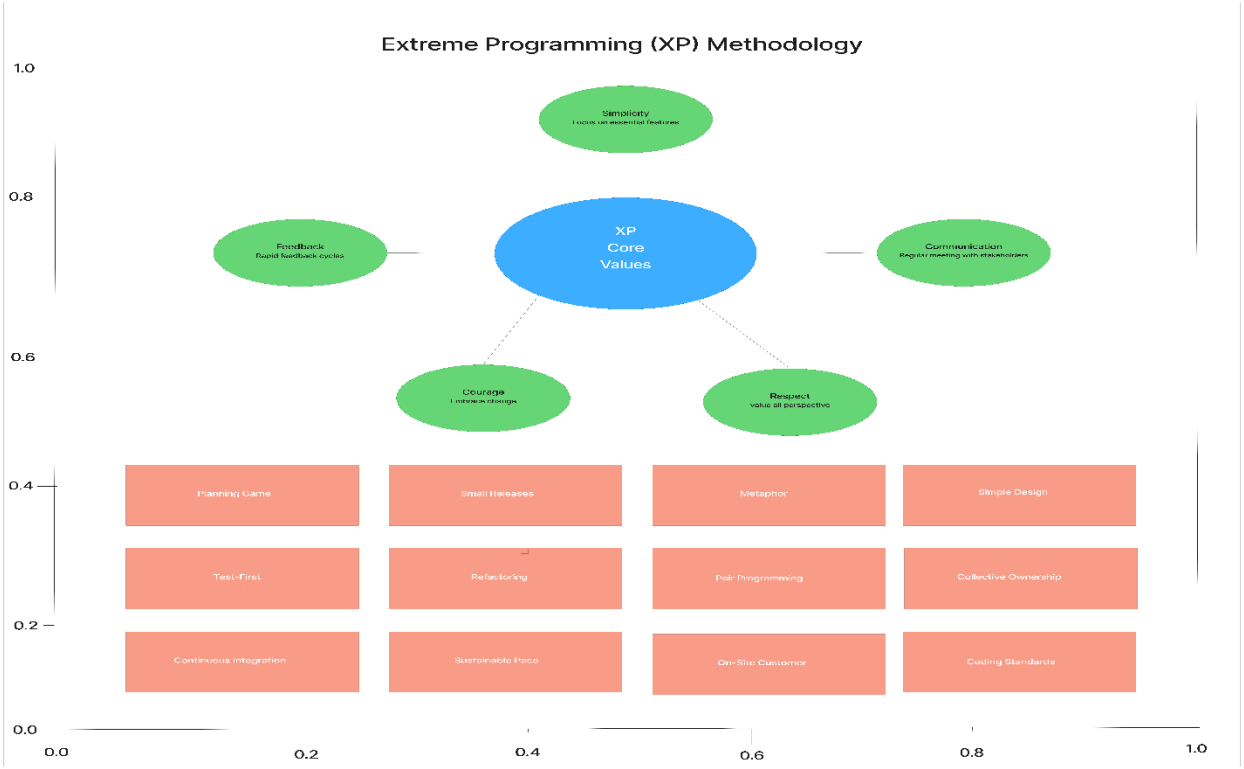


Figure 2-3: Extreme Programming (XP) Methodology showing core values and practices applied to the Examination Management System project

3.2 The Sprint

In Extreme Programming (XP), iterations are typically referred to as "timeboxes" rather than sprints (which is more common in Scrum). However, the concept serves a similar purpose: to divide the development process into manageable, fixed-length periods during which specific work is completed and made ready for review. For our Examination Management System project, we adopted the term "sprint" while maintaining XP's core practices and values.

3.2.1 Stages of Sprint Life Cycle in XP

While XP doesn't formally define a sprint lifecycle as Scrum does, our project incorporated a structured approach to iterations that combined XP practices with elements of sprint management.

Each sprint in our project followed these stages:

1. **Planning Game:** At the beginning of each sprint, the team engaged in the XP Planning Game, where:

- Business representatives (department heads and administrators) defined priorities
- Technical team members provided estimates and identified technical constraints
- User stories were selected for the iteration based on business value and technical feasibility
- Acceptance tests were defined to clarify requirements

2. **Development**: During this stage, the team:

- Practiced pair programming for all production code
- Followed test-first development (writing tests before code)
- Maintained collective code ownership
- Integrated code continuously (multiple times per day)
- Refactored code to maintain simplicity and clarity

3. **Daily Stand-ups**: Brief daily meetings where team members:

- Shared what they accomplished the previous day
- Discussed what they planned to work on today
- Identified any obstacles or impediments

4.**Customer Review:** At the end of each sprint:

- Working software was demonstrated to stakeholders
- Immediate feedback was gathered
- Acceptance tests were verified
- New insights were incorporated into future planning

5.Retrospective: The team reflected on:

- What went well during the sprint
- What could be improved
- Action items for process improvement in the next sprint

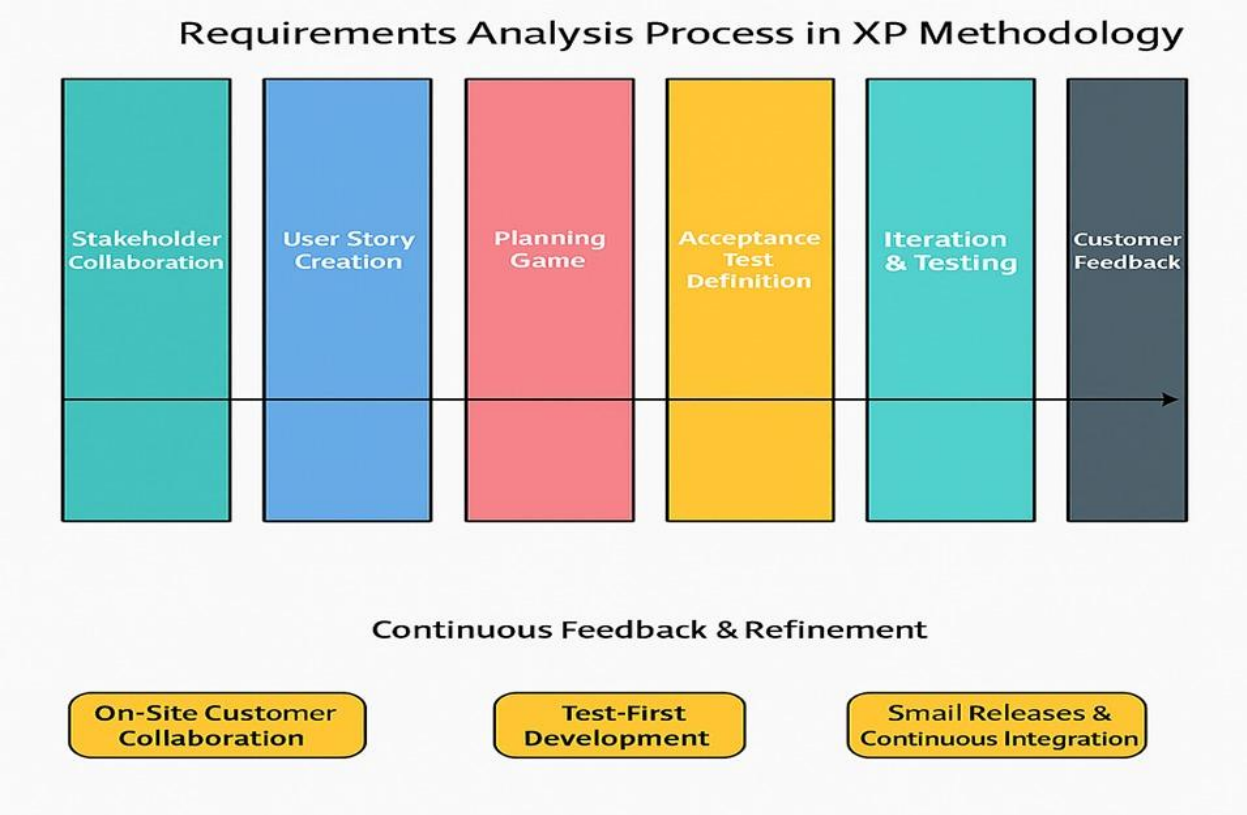


Figure 3-3: Requirements Analysis Process in XP Methodology showing the iterative flow from stakeholder collaboration to customer feedback

The sprint lifecycle provided structure while maintaining XP's emphasis on adaptability, continuous feedback, and close customer collaboration. For the Examination Management System, sprints were set to two weeks in duration, providing frequent opportunities for stakeholder feedback while allowing sufficient time for meaningful development progress.

3.3 Product Backlog Creation for Project

In XP, the product backlog is typically less formalized than in other agile methodologies like Scrum. Instead, XP focuses on maintaining a prioritized list of user stories that represent customer requirements. For our Examination Management System project, we adopted a structured approach to backlog creation that maintained XP's emphasis on simplicity while providing sufficient organization for effective planning.

The product backlog for the Examination Management System was organized hierarchically, from high-level themes down to specific user stories:

3.3.1 Theme

In the context of our Examination Management System, themes represent high-level objectives or strategic goals that guide development efforts. Themes provide a way to group related epics and user stories, creating a coherent narrative around major system capabilities.

The primary themes identified for the Examination Management System were:

1. User Authentication and Access Control

- Ensuring secure access to the system with appropriate role-based permissions

- Managing user profiles and account information

2. Department and Course Management

- Organizing academic departments and their relationships
- Managing course offerings, instructors, and student enrollments

3. Examination Scheduling and Administration

- Creating and managing examination schedules
- Allocating resources (rooms, proctors, materials)
- Handling scheduling conflicts

4. Result Management and Reporting

- Recording and calculating examination results
- Generating reports and analytics
- Publishing results to stakeholders

5. System Administration and Configuration

- Managing system settings and configurations
- Monitoring system performance and usage

- Performing maintenance tasks

Themes provided strategic alignment for development efforts, ensuring that all features contributed to the overall vision of the Examination Management System.

3.3.2 Epic

Epics represent significant pieces of functionality that deliver substantial business value but are too large to complete within a single sprint. In our project, epics served as containers for related user stories, providing a middle layer between high-level themes and detailed user stories.

Key epics identified for the Examination Management System included:

1.Authentication System

- User registration and account creation
- Login and session management
- Password recovery and security features
- Role-based access control

2.Department Management

- Department creation and configuration
- Staff assignment and management
- Department resource allocation

3.Course Management

- Course creation and configuration
- Instructor assignment
- Student enrollment
- Course scheduling

4.Examination Creation

- Exam template creation
- Question bank management
- Exam configuration and settings

5.Examination Scheduling

- Calendar management
- Room allocation

- Proctor assignment
- Conflict detection and resolution

6. Result Processing

- Grade entry and calculation
- Result verification and approval
- Result publication
- Appeals management

7. Reporting and Analytics

- Standard report generation
- Custom report creation
- Data visualization and dashboards
- Export capabilities

Each epic was further broken down into specific user stories that could be completed within a single sprint, allowing for incremental delivery of value.

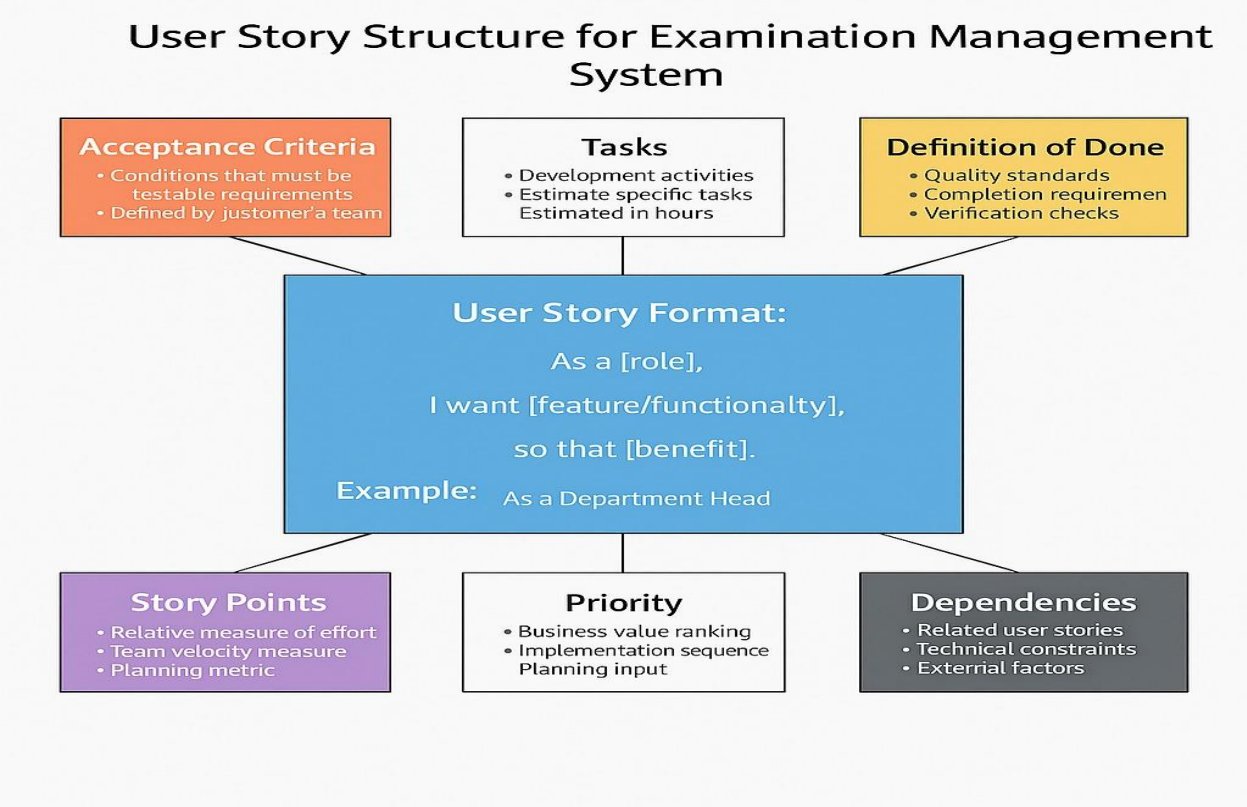


Figure 4-3: User Story Structure for Examination Management System showing the components of a well-formed user story

The following table provides a comparison of Themes, Epics, and User Stories as used in our Examination Management System project:

Aspect	Theme	Epic	User Story
Definition	A high-level grouping of related Epics and Stories, representing a broad goal or strategic objective.	A large, complex requirement that represents a significant piece of functionality or feature.	A small, specific requirement that represents a single piece of work to deliver value to the user.
Scope	Very high-level, encompassing multiple epics.	High-level, covering multiple smaller stories.	Granular, focusing on a specific task or goal.
Time to Complete	Long-term, may span an entire project or product lifecycle.	Longer duration, possibly spanning multiple sprints.	Shorter duration, typically completed in a sprint.
Purpose	Organizes work into meaningful categories to align with business objectives.	Helps organize and group related work.	Drives the actual implementation and delivery of features.
Detail Level	Broad, providing guidance on overall priorities and objectives.	General, not enough detail for direct implementation.	Detailed, includes acceptance criteria and specifics for implementation.
Relationship	Encompasses multiple epics that align with its goal.	Can be broken down into multiple User Stories.	A part of an Epic, contributing to its completion.
Examples	"User Authentication and Access Control" for the Examination System	"Authentication System" including login, registration, and access control	"As an Administrator, I want to create user accounts with specific roles so that I can control system access"

Table 2.2: **Theme vs Epic vs User story**

3.4 The Sprint (1)

3.4.1 Sprint Goal

The goal of Sprint 1 was to establish the foundation of the Examination Management System by implementing the core authentication functionality and basic department management features. This would

provide a secure framework upon which subsequent features could be built while delivering immediate value to administrators.

3.4.2 Create Sprint Backlog

The Sprint 1 backlog was created during the Planning Game session, where business stakeholders and technical team members collaborated to select and prioritize user stories based on business value and technical dependencies. The following user stories were selected for Sprint 1:

3.4.2.1 User Story Cards

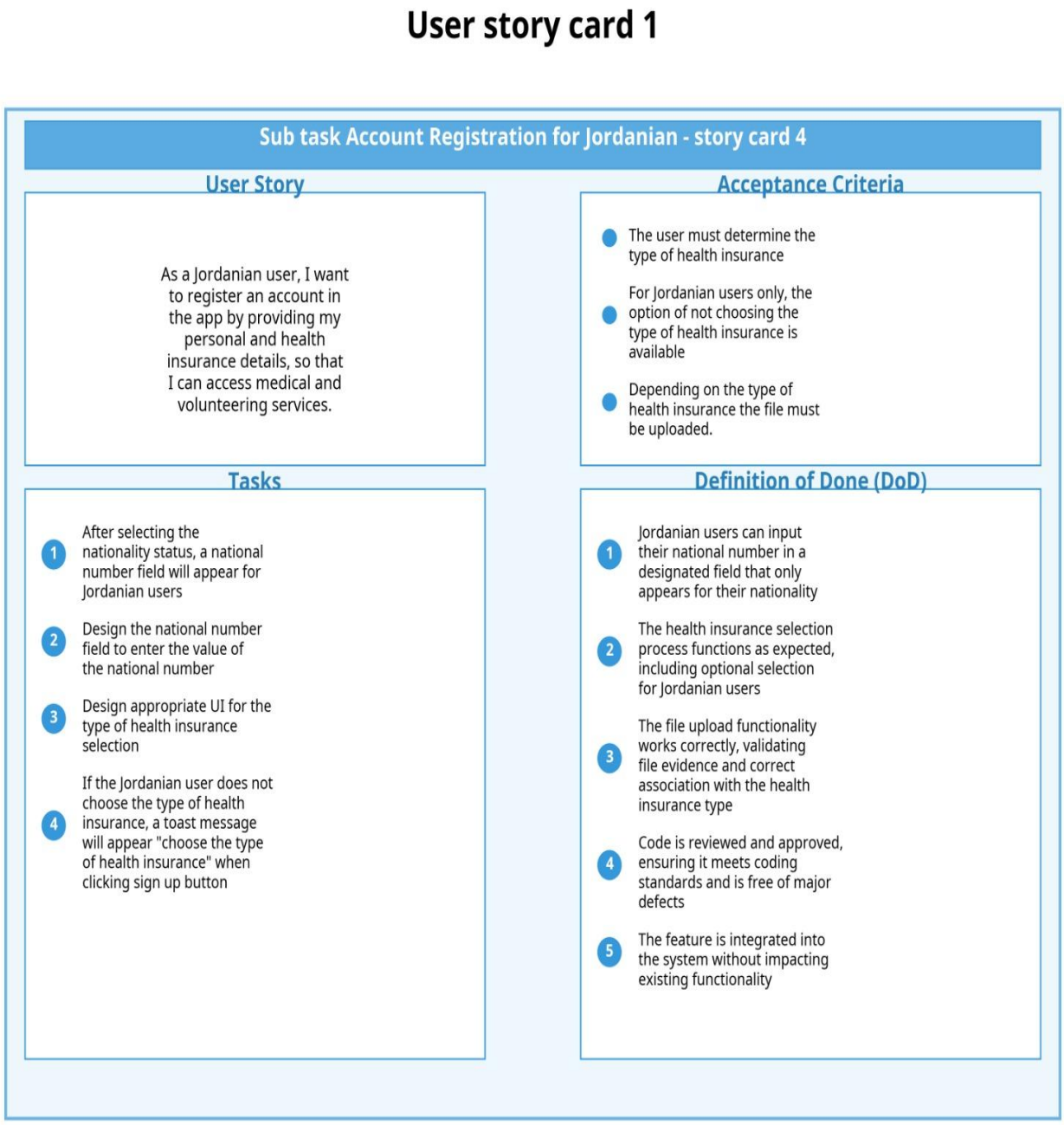


Figure 5-3: User Story Card for Administrator Authentication

User story card 2

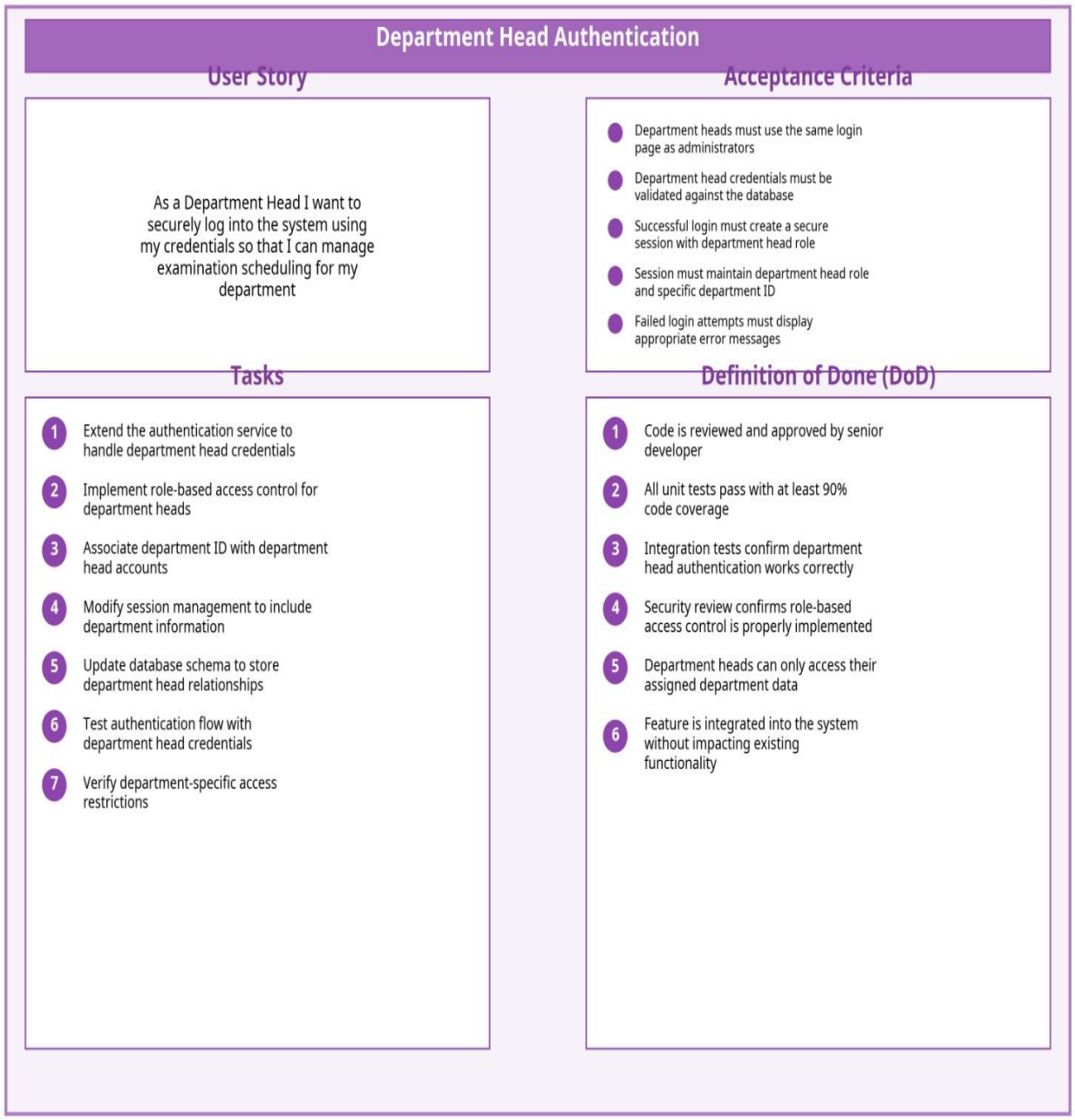


Figure 6-3: User Story Card for Department Head Authentication

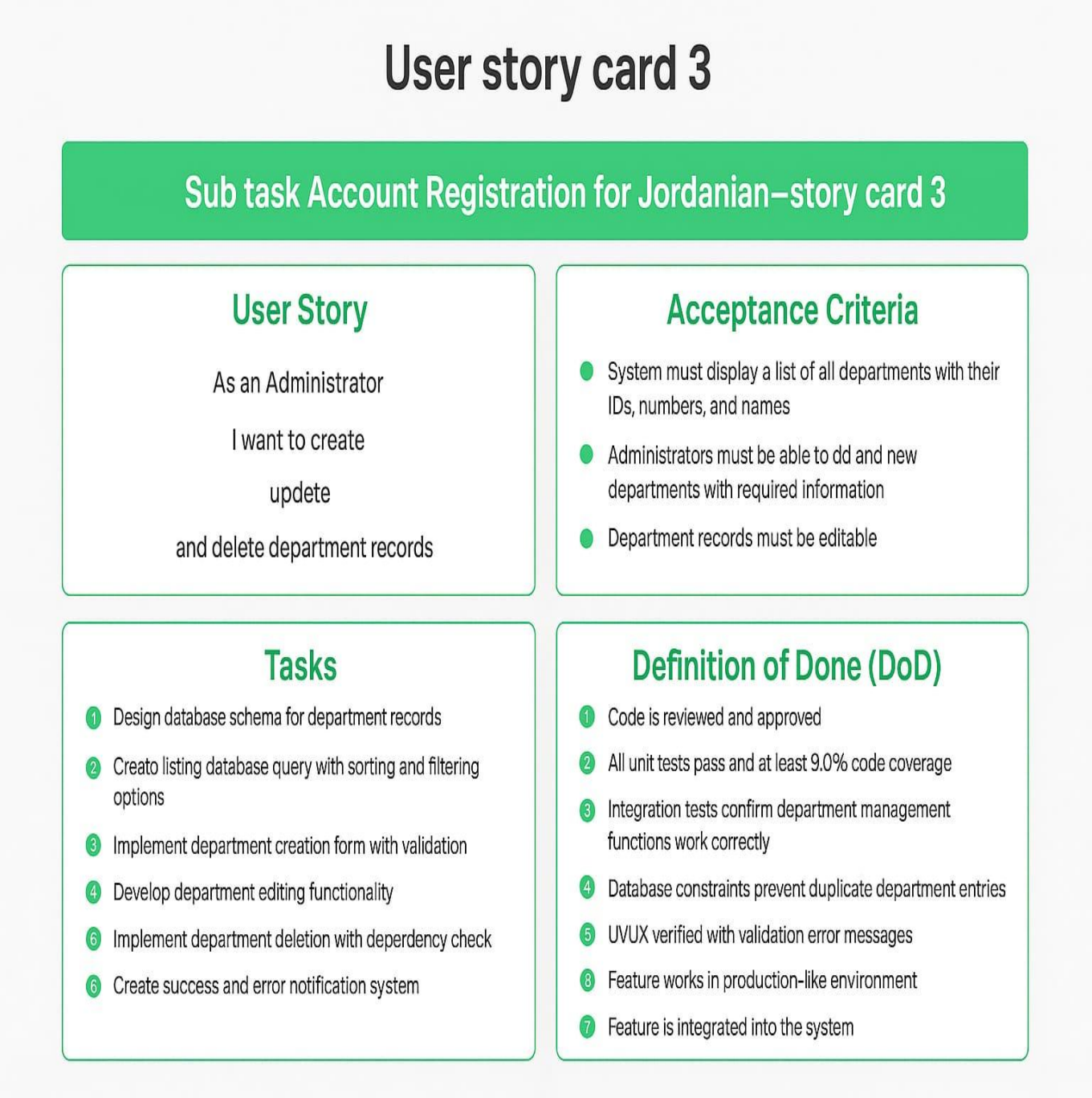


Figure 7-3: User Story Card for Department Management

The Planning Game session for Sprint 1 involved detailed discussions about each user story, with business representatives clarifying requirements and technical team members providing estimates. The team used the XP practice of "Yesterday's Weather" to determine their capacity for the sprint, based on previous performance.

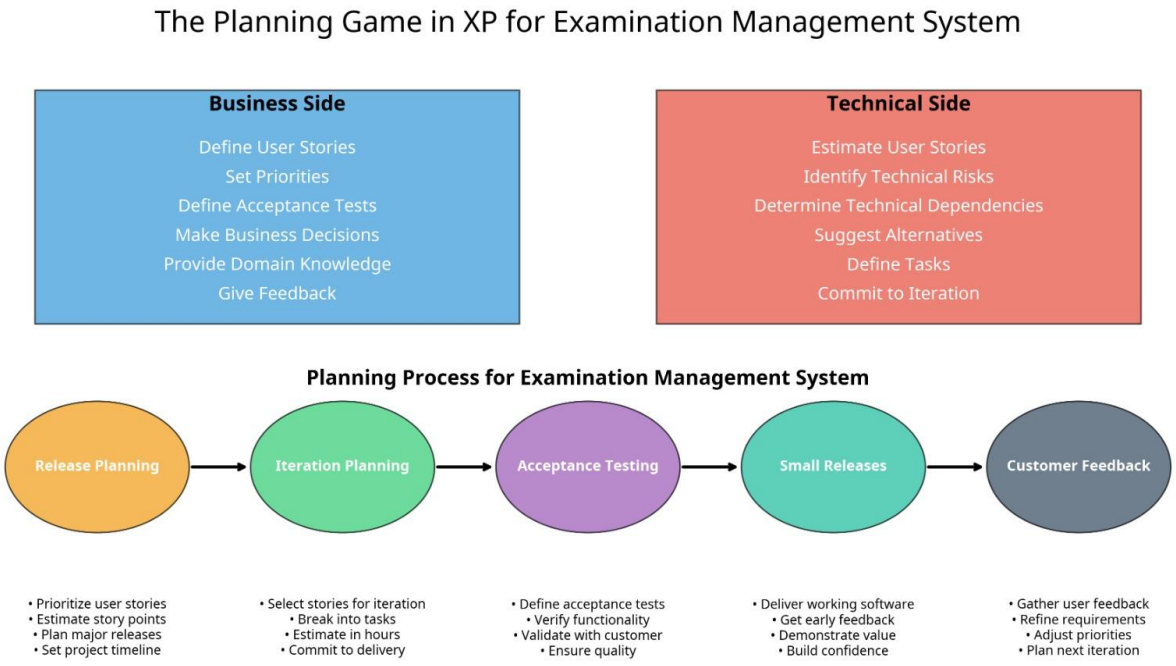


Figure 8-3: The Planning Game in XP showing business and technical sides collaborating on requirements

3.5 The Sprint (2)

3.5.1 Sprint Goal

The goal of Sprint 2 was to extend the system's capabilities by implementing course management functionality and establishing the foundation for examination scheduling. This would allow department heads to begin organizing their courses and preparing for examination planning.

3.5.2 Create Sprint Backlog

During the Planning Game for Sprint 2, the team selected user stories focused on course management and initial examination setup. The selection was influenced by feedback from Sprint 1 and refined priorities from business stakeholders.

3.5.2.1 User Story Cards

User story card 1

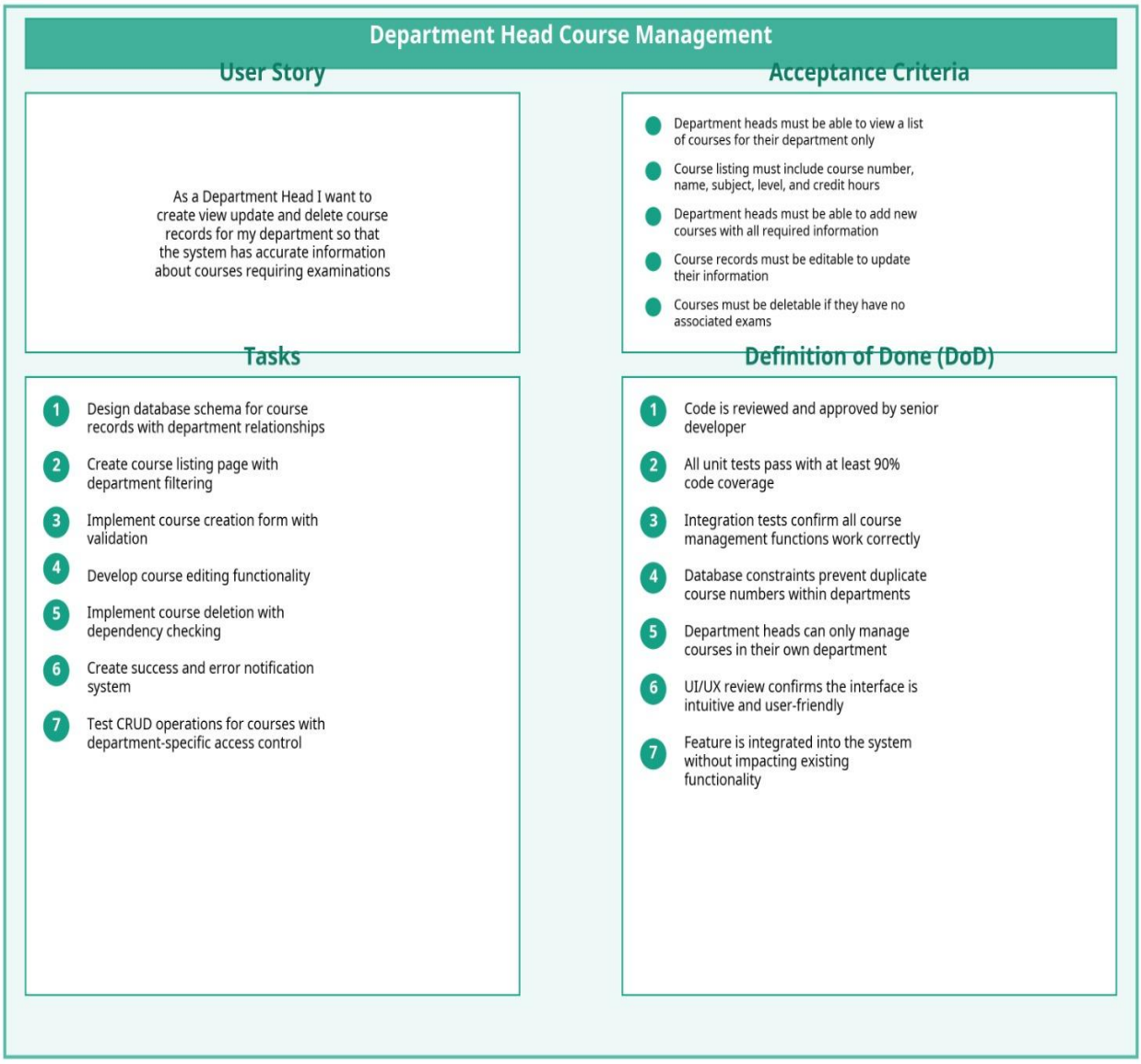


Figure 9-3: User Story Card for Course Management

Additional user stories in Sprint 2 included:

1. Instructor Management

- As a Department Head, I want to add instructors to the system and assign them to courses, so that course responsibilities are clearly defined.
- Acceptance Criteria:
  - Department Head can add new instructors with their details
  - Department Head can assign instructors to specific courses
  - Department Head can view all instructors in their department
  - System prevents assignment conflicts (e.g., scheduling conflicts)

2. Student Enrollment

- As a Department Head, I want to enroll students in courses, so that they can be included in examination planning.
- Acceptance Criteria:
  - Department Head can add students to courses individually or in bulk
  - System validates student information before enrollment
  - Department Head can view enrolled students for each course
  - System prevents duplicate enrollments

3.Course Schedule Management

- As a Department Head, I want to define course schedules, so that examination planning can account for course timing.
- Acceptance Criteria:
  - Department Head can set days, times, and locations for course sessions
  - System detects and prevents scheduling conflicts
  - Department Head can view complete course schedules
  - Schedule information is available for examination planning

3.6 The Sprint (3)

3.6.1 Sprint Goal

The goal of Sprint 3 was to implement the core examination creation functionality, allowing administrators and department heads to define examination templates, question banks, and examination settings.

3.6.2 Create Sprint Backlog

The Sprint 3 backlog focused on examination creation features, building on authentication, department, and course management foundations established in previous sprints.

3.6.2.1 User Story Cards

Examination Creation User Story Card

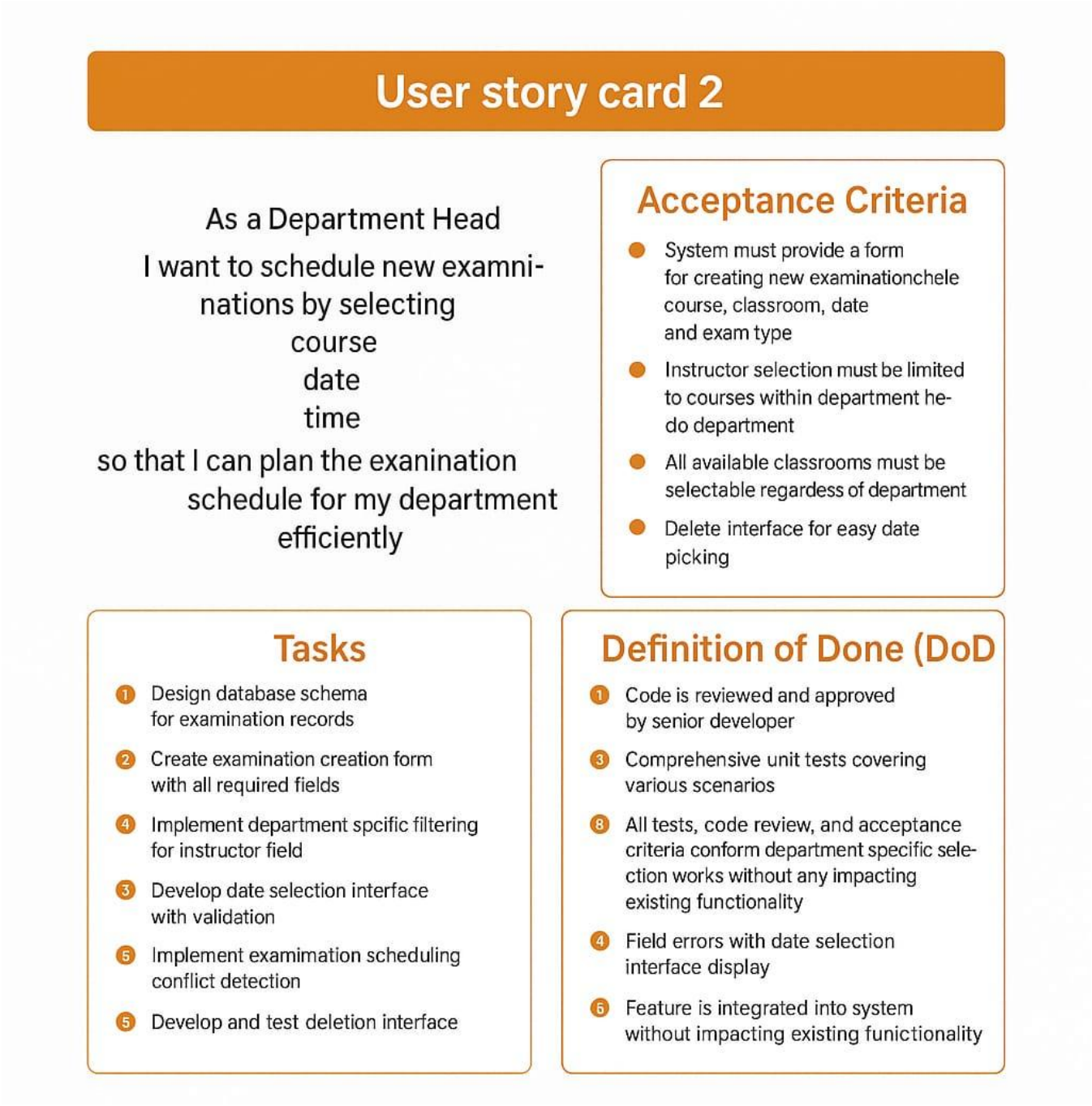


Figure 10-3: User Story Card for Examination Creation

Additional user stories in Sprint 3 included:

1.Question Bank Management

- As an Instructor, I want to create and manage question banks for my courses, so that I can reuse questions across multiple examinations.
- Acceptance Criteria:
  - Instructor can create question banks with different question types
  - Instructor can categorize questions by topic and difficulty
  - Instructor can search and filter questions
  - Questions can be reused in multiple examinations

2.Examination Template Creation

- As a Department Head, I want to create examination templates with standardized formats, so that examinations maintain consistency across the department.
- Acceptance Criteria:
  - Department Head can define template structure and formatting
  - Templates can include department branding and standard instructions
  - Templates can be applied to new examinations
  - Templates can be shared across the department

3.7 The Sprint (4)

3.7.1 Sprint Goal

The goal of Sprint 4 was to implement examination scheduling functionality, allowing department heads to plan examination dates, allocate resources, and manage potential conflicts.

3.7.2 Create Sprint Backlog

The Sprint 4 backlog focused on examination scheduling features, building on the examination creation capabilities developed in Sprint 3.

3.7.2.1 User Story Cards

1.Examination Calendar Management

- As a Department Head, I want to view and manage an examination calendar, so that I can plan examinations without conflicts.
- Acceptance Criteria:
  - Department Head can view a calendar of all scheduled examinations
  - Calendar displays room allocations and proctor assignments
  - Department Head can filter calendar by course, instructor, or date range
  - Calendar highlights potential conflicts

2.Room Allocation

- As a Department Head, I want to allocate rooms for examinations based on capacity and equipment needs, so that examinations have appropriate facilities.
- Acceptance Criteria:
  - Department Head can view available rooms with their capacities and features
  - System suggests suitable rooms based on examination requirements
  - System prevents double-booking of rooms
  - Department Head can manually override suggestions when necessary

3.Proctor Assignment

- As a Department Head, I want to assign proctors to examinations, so that examinations are properly supervised.
- Acceptance Criteria:
  - Department Head can view available proctors and their schedules
  - System suggests proctor assignments based on availability
  - System prevents scheduling conflicts for proctors
  - Department Head can manually assign or reassign proctors

3.8 The Sprint (5)

3.8.1 Sprint Goal

The goal of Sprint 5 was to implement result management functionality, allowing instructors to record examination results, calculate grades, and prepare results for publication.

3.8.2 Create Sprint Backlog

The Sprint 5 backlog focused on result management features, building on the examination scheduling capabilities developed in Sprint 4.

3.8.2.1 User Story Cards

1.Result Entry

- As an Instructor, I want to enter examination results for students, so that grades can be calculated and published.
- Acceptance Criteria:
  - Instructor can enter results individually or import in bulk
  - System validates entries against possible score ranges
  - Instructor can save partial results and complete entry later
  - System maintains an audit trail of result entries and changes

2.Grade Calculation

- As an Instructor, I want the system to calculate final grades based on examination results and grading schemes, so that grading is consistent and accurate.
- Acceptance Criteria:
  - System applies predefined grading schemes to raw scores
  - Instructor can review calculated grades before finalization
  - System supports different grading schemes (curve, absolute, etc.)
  - Instructor can manually adjust grades with justification

3.Result Verification

- As a Department Head, I want to verify examination results before publication, so that accuracy is ensured.
- Acceptance Criteria:
  - Department Head can review result summaries and statistics
  - System flags unusual patterns or potential errors
  - Department Head can approve or return results for revision
  - System maintains verification status and history

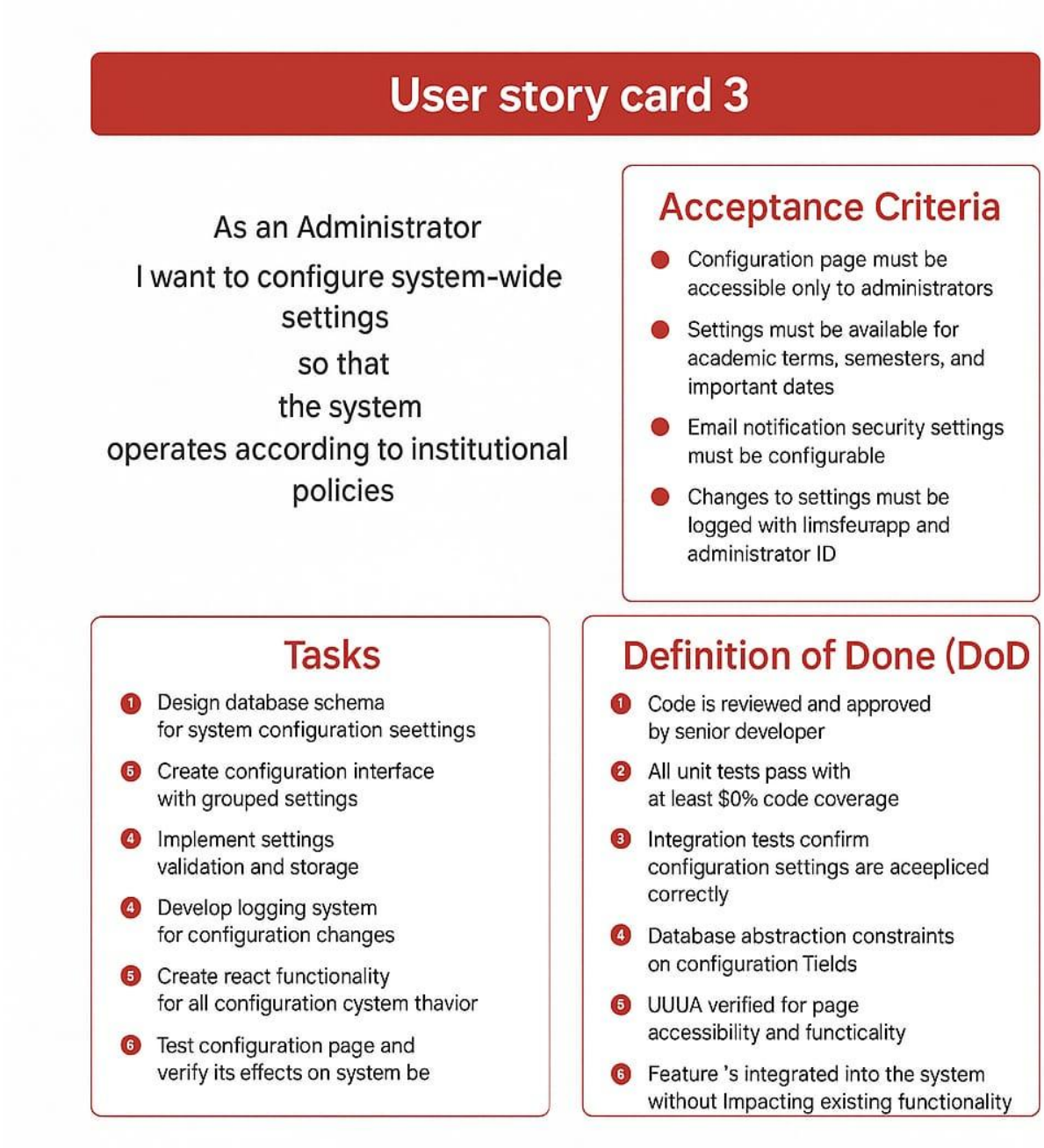


Figure 11-3: User Story Card for System Configuration

3.9 The Sprint (6)

3.9.1 Sprint Goal

The goal of Sprint 6 was to implement result publication and notification functionality, allowing approved results to be published to students and relevant stakeholders.

3.9.2 Create Sprint Backlog

The Sprint 6 backlog focused on result publication features, building on the result management capabilities developed in Sprint 5.

3.9.2.1 User Story Cards

1.Result Publication

- As a Department Head, I want to publish approved examination results, so that students can access their grades.
- Acceptance Criteria:
  - Department Head can select verified results for publication
  - System publishes results according to predefined schedule
  - Published results are available to authorized users only
  - System maintains publication history and status

2.Student Result Access

- As a Student, I want to view my examination results, so that I can track my academic progress.
- Acceptance Criteria:
  - Student can view only their own results
  - Results display grade, score, and relevant statistics
  - Student can view historical results across semesters
  - System provides appropriate context for interpretation

3.Result Notifications

- As an Administrator, I want the system to send notifications when results are published, so that stakeholders are informed promptly.
- Acceptance Criteria:
  - System sends notifications through multiple channels (email, SMS, in-app)
  - Notifications are sent only to relevant stakeholders
  - Notification content complies with privacy requirements
  - Administrator can configure notification templates and timing

3.10 The Sprint (7)

3.10.1 Sprint Goal

The goal of Sprint 7 was to implement reporting and analytics functionality, providing administrators and department heads with insights into examination performance and system usage.

3.10.2 Create Sprint Backlog

The Sprint 7 backlog focused on reporting and analytics features, completing the core functionality of the Examination Management System.

3.10.2.1 User Story Cards

1.Standard Reports

- As an Administrator, I want to generate standard reports on examination results and system usage, so that I can monitor institutional performance.
- Acceptance Criteria:
  - Administrator can select from predefined report templates
  - Reports include appropriate visualizations and summaries
  - Reports can be filtered by department, course, date range, etc.
  - Reports can be exported in multiple formats (PDF, Excel, etc.)

2.Custom Report Builder

- As a Department Head, I want to create custom reports on examination data, so that I can analyze specific aspects of academic performance.
- Acceptance Criteria:

- Department Head can select data fields and filtering criteria Department Head can choose visualization types
- Custom reports can be saved for future use
- Reports respect data access permissions

3.Analytics Dashboard

- As an Administrator, I want to view analytics dashboards with key performance indicators, so that I can quickly assess system and academic performance.
- Acceptance Criteria:
  - Dashboard displays relevant metrics and trends
  - Administrator can customize dashboard layout and content
  - Dashboard updates in real-time or on schedule
  - Dashboard elements are interactive for deeper analysis

3.11 Burndown Chart

The Burndown Chart is a key visual tool in agile methodologies that tracks the completion of work throughout the project. For our Examination Management System project, the burndown chart tracked the remaining story points across all seven sprints, providing visibility into project progress and helping to identify potential issues early.

Burndown Chart - Examination Management System

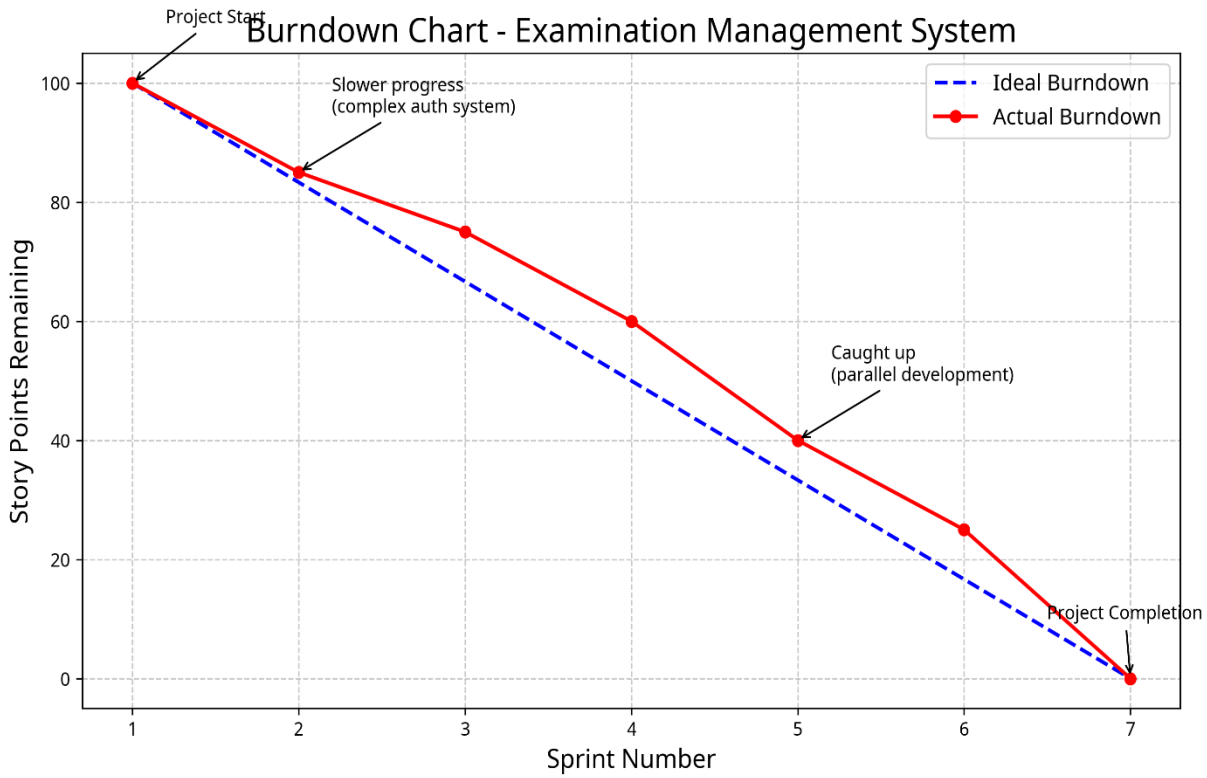


Figure 12-3: Burndown Chart showing planned vs. actual progress across all sprints

The burndown chart reveals several insights about the project's progress:

1.Initial Alignment: The project started on track, with actual progress closely matching the ideal burndown during Sprint 1.

2. Mid-project Challenges: During Sprints 2-4, the team encountered some challenges with the complex authentication system and examination scheduling features, resulting in slightly slower progress than planned.

3.Recovery through Parallel Development: In Sprint 5, the team implemented parallel development strategies, allowing them to catch up to the ideal burndown line.

4.Consistent Completion: The final sprints proceeded smoothly, with the project completing on schedule by the end of Sprint 7.

The burndown chart served as both a tracking tool and a communication aid, helping stakeholders understand project status and informing decisions about resource allocation and priority adjustments.

3.12 User Story Mapping

User Story Mapping is a technique that helps visualize the user's journey through a system and organize user stories in a way that provides a holistic view of functionality. For the Examination Management System, we created a user story map to organize stories by user activities, tasks, and releases.

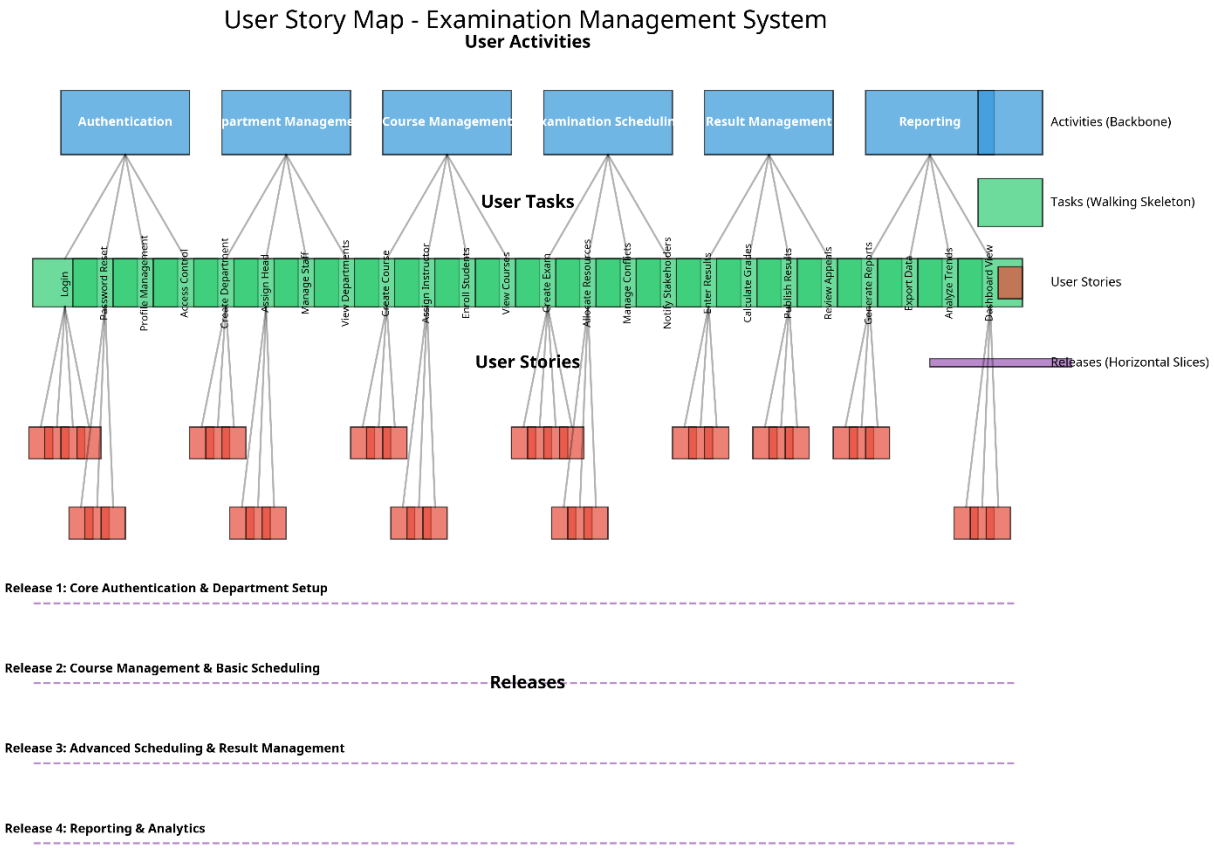


Figure 13-3: User Story Map showing the organization of user stories by activities, tasks, and releases

The user story map for the Examination Management System was organized as follows:

1.Backbone (User Activities): The top level represented major user activities such as Authentication, Department Management, Course Management, Examination Scheduling, Result Management, and Reporting.

2.Walking Skeleton (User Tasks): The middle level broke down each activity into specific tasks that users perform, such as Login, Create Department, Assign Instructor, Schedule Exam, Enter Results, and Generate Reports.

3.User Stories: The bottom level contained the detailed user stories associated with each task, prioritized by value and dependency.

4.Releases (Horizontal Slices): The map was divided horizontally into planned releases, showing which functionality would be delivered in each release:

- Release 1: Core Authentication & Department Setup
- Release 2: Course Management & Basic Scheduling
- Release 3: Advanced Scheduling & Result Management
- Release 4: Reporting & Analytics

The user story map provided a visual representation of the entire system, helping stakeholders understand the scope and sequence of development while ensuring that the team maintained focus on delivering end-to-end value in each release.

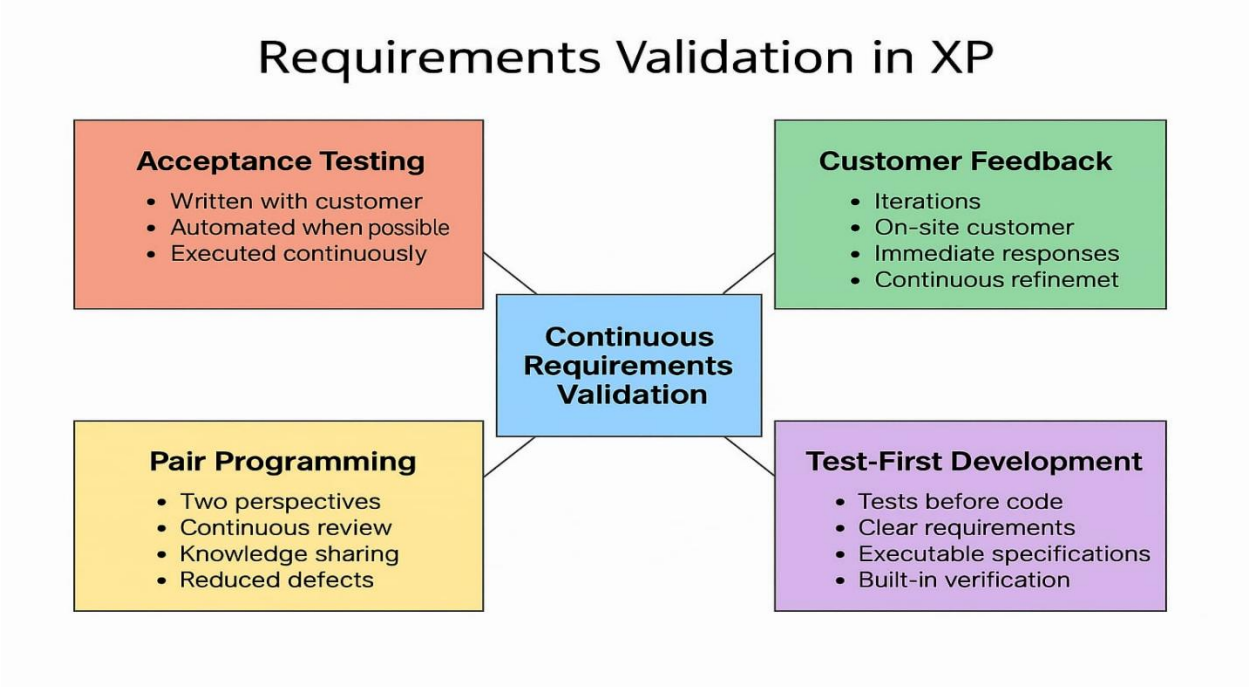


Figure 14-3: Requirements Validation approaches in XP methodology as applied to the Examination Management System

Throughout the requirements analysis phase, the team employed XP's validation practices to ensure that requirements accurately reflected stakeholder needs:

1.Acceptance Testing: Each user story included acceptance criteria that were translated into automated tests wherever possible, providing clear verification of functionality.

2.Customer Feedback: Regular demonstrations and on-site customer collaboration ensured immediate feedback and continuous refinement of requirements.

3.Pair Programming: All development work was performed in pairs, providing continuous review and multiple perspectives on requirement implementation.

4.Test-First Development: Tests were written before code, ensuring that requirements were clearly understood and verifiable.

Chapter 4: Design Phase

Figure1-4 : design phase

Chapter 4: Design Phase (XP Adaptation)

4.1 Introduction to Design in XP

The design phase within the context of the Exam Scheduling project, developed using Extreme Programming (XP), is not a distinct, monolithic stage but rather an ongoing activity integrated throughout the development lifecycle. Unlike traditional approaches that often mandate a comprehensive upfront design, XP embraces an emergent design philosophy. This means the system's structure and details evolve iteratively as the software is built, tested, and refactored. For the Exam Scheduling System, this implies that the design for managing entities like rooms, courses, exams, and conflicts was developed incrementally, focusing initially on the simplest solutions to meet immediate requirements derived from user stories.

This emergent approach is guided by XP's core values. Simplicity dictates creating the most straightforward design possible at any given moment, avoiding unnecessary complexity (YAGNI - You Ain't Gonna Need It). Communication ensures the team shares a common understanding, perhaps through practices like Pair Programming or a shared System Metaphor (even if implicit). Feedback, primarily through automated tests (like those encouraged by Test-Driven

Development) and user interactions, validates design choices and highlights areas needing refinement. Courage empowers the team to refactor the design aggressively to improve it or adapt to new insights without fear of breaking existing functionality. This iterative process, grounded in these values, aims to produce a design that is not only functional but also flexible, maintainable, and closely aligned with the actual needs of scheduling exams at Al-Balqa Applied University.

4.2 Guiding Principles and Techniques in XP Design

Several core principles and techniques guide the design process within Extreme Programming, ensuring that the evolving architecture remains simple, maintainable, and aligned with user needs. These practices were central to the development of the Exam Scheduling System.

- **Simple Design (YAGNI):** The most fundamental principle is to maintain simplicity. This involves implementing only the functionality required for the current iteration or user story, encapsulated by the phrase "You Ain't Gonna Need It" (YAGNI). In the Exam Scheduling System, this is evident in the directness of the PHP scripts handling CRUD operations for Rooms, Courses, and Department Heads. Each script focuses on its specific task without incorporating speculative features or overly complex layers. The design avoids premature

optimization and focuses on delivering the required functionality in the most straightforward manner possible.

- **Refactoring:** XP relies heavily on continuous refactoring to improve the internal quality of the code without altering its external behavior. As the Exam Scheduling System evolved, refactoring would have been essential to keep the codebase clean, remove duplication (like the potential issues suggested by the - Copy files), improve the clarity of variable and function names, and simplify complex conditional logic, particularly within areas like conflict checking. This ongoing process ensures the design remains adaptable and easy to understand, preventing the accumulation of technical debt.

- **Test-Driven Development (TDD):** Although explicit test files were not provided, TDD is a cornerstone of XP design. The practice involves writing automated tests before writing the production code. This forces developers to consider the code's interface and behavior upfront, leading naturally to modular and testable designs. For the Exam Scheduling System, TDD would involve writing unit tests for PHP functions (e.g., validating input, checking conflicts, interacting with the database) and potentially integration or functional tests for user interactions (e.g., logging in, adding an exam). The resulting test suite acts as a safety net, enabling confident refactoring and providing living documentation of the system's expected behavior.
- **System Metaphor (Inferred):** While not explicitly documented, a guiding metaphor like "Digital Scheduling Assistant" or "University

Resource Coordinator" could have helped unify the team's understanding. Such a metaphor provides a shared vocabulary and conceptual framework, making design discussions more intuitive and the codebase potentially easier to navigate. The current naming conventions (`ClassRooms`, `process_exam`, etc.) are functional but don't strongly suggest a consistently applied metaphor, indicating an area where explicit definition could enhance conceptual integrity.

These principles and techniques work together, allowing the design of the Exam Scheduling System to emerge and evolve effectively throughout the development process.

4.3 Database Design

The persistence layer of the Exam Scheduling System is handled by a relational database, identified as `examinationsystem` within the project's configuration (`db_connection.php`). The design of the database schema, inferred from the PHP scripts' interactions and SQL queries, appears straightforward and directly reflects the core entities required for the system's functionality. This aligns with the XP principle of implementing the simplest design that meets the current requirements, avoiding unnecessary complexity in the data model.

The key tables inferred to exist within the examinationsystem database are:

- **admin:**Stores login credentials for system administrators.

- AdminID (Primary Key, likely integer)
- username (VARCHAR)
- password (VARCHAR - Note: Storing plain text passwords is a security risk; hashing should be used in production).

- **departmenthead:** Manages information about department heads, who likely have specific roles in the exam process.

- id (Primary Key, likely integer)
- Name (VARCHAR, Full Name)
- user_id (VARCHAR, Username for login)
- Password (VARCHAR - Same security note as above applies)
- department (VARCHAR or Foreign Key to a departments table)
- status (VARCHAR or ENUM, e.g., 'Active', 'Inactive')
- last_access (TIMESTAMP or DATETIME)
- add_date (TIMESTAMP or DATETIME)

- **ClassRooms:** Contains details of physical rooms available for exams.

- ClassRoomNumber (Primary Key or Unique Key, likely VARCHAR or INT)
- RoomType (VARCHAR, e.g., 'Lab', 'Lecture Hall')
- Capacity (INT)
- created_at (TIMESTAMP or DATETIME)

- **Courses:** Stores information about academic courses requiring exams.

    - CourseID (Primary Key, likely integer)
    - CourseName (VARCHAR)
    - CourseNumber (VARCHAR or INT)
    - SubjectLevel (VARCHAR, e.g., 'Bachelor', 'Master') DepartmentID (Foreign Key, referencing a department entity) Level (VARCHAR, e.g., 'Undergraduate', 'Graduate')
    - CreditHours (INT)
    - created_at (TIMESTAMP or DATETIME)

    - **exams:** Central table for storing scheduled exam details.

        - ExamID (Primary Key, likely integer)
        - CourseID (Foreign Key, referencing Courses)
        - ExamDate (DATE)
        - StartTime (TIME)
        - EndTime (TIME)
        - ClassRoomNumber (Foreign Key, referencing ClassRooms)
        - (Potentially other fields like DepartmentHeadID, ExamType, Status)

        - **conflicts:** Likely used for logging or managing detected scheduling conflicts.

- Structure is less clear from provided files, but might include

(ConflictID, ConflictingExamID1, ConflictingExamID2, ConflictType, ResolutionStatus, Timestamp).

The relational nature of the schema allows for standard data retrieval and querying using SQL. The simplicity of the structure makes it relatively easy to understand and modify, which is advantageous in an agile environment like XP where requirements might evolve. The design directly supports the core functions of managing resources, scheduling exams, and checking for conflicts.

The following Entity-Relationship (ER) diagram visually represents the tables and their relationships based on the provided SQL schema:

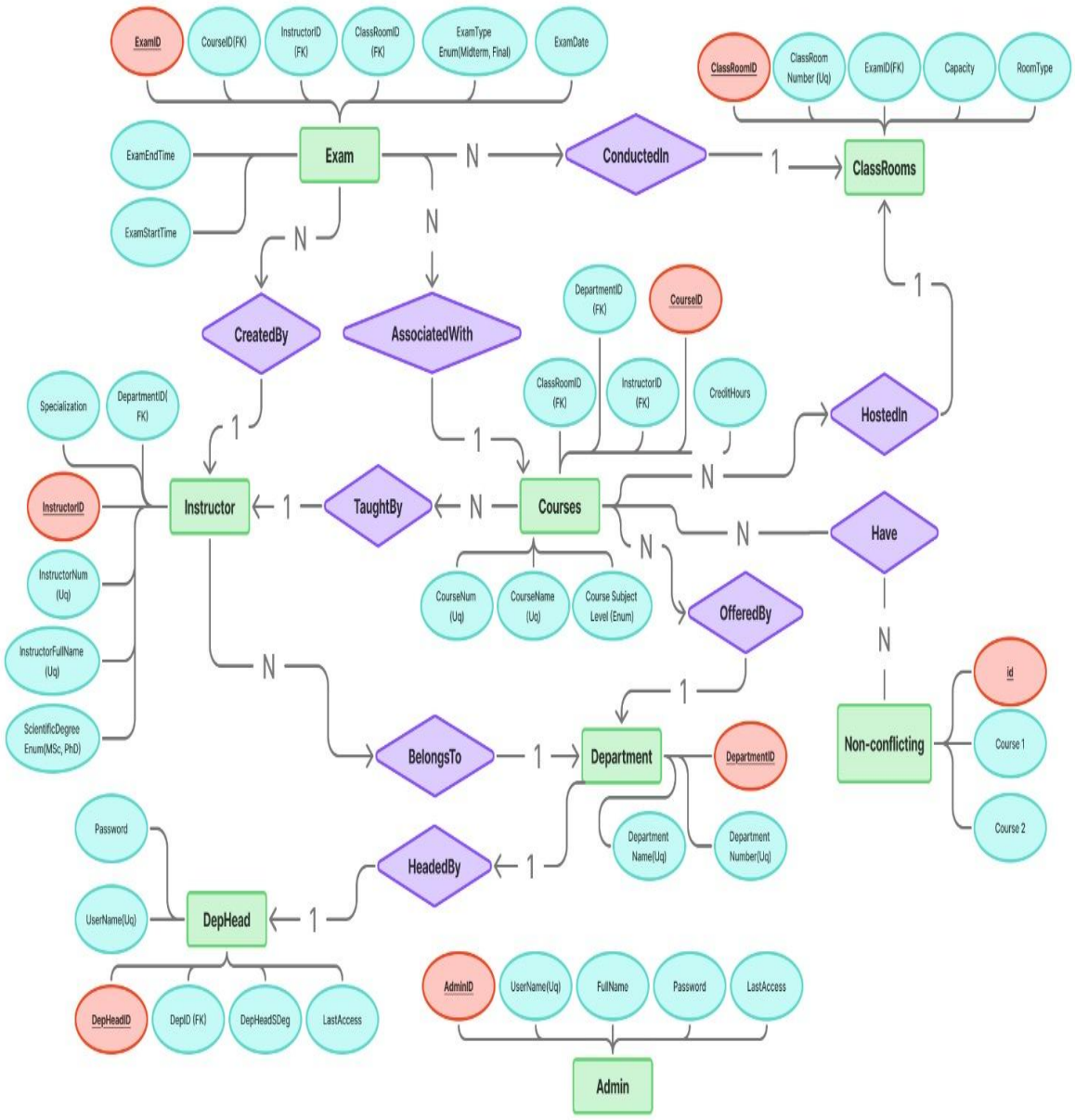


Figure 2 -4: ER-diagram

4.4 Architectural Design

The architectural design of the Exam Scheduling System, consistent with the emergent nature favored by XP, is primarily a multi-page web application built using PHP for server-side logic and standard web technologies (HTML, CSS, JavaScript) for the client-side presentation and interaction. The architecture facilitates the core requirements of managing university resources (rooms, courses, personnel) and coordinating the exam scheduling process, including conflict detection.

The system follows a client-server model. The client (user's web browser) interacts with the server via HTTP requests, typically initiated by navigating between pages or submitting forms. Key architectural components identified from the codebase include:

- **Presentation Layer:** Comprises the HTML files (.html, and HTML within .php files like home.php), CSS files (.css), and client-side JavaScript (embedded in HTML or in .js files like AddExam.js, depheadlogin.js). This layer is responsible for rendering the user interface, handling user input, performing client-side validation, and communicating with the backend via form submissions or asynchronous JavaScript (AJAX) calls using fetch.
- **Application Logic Layer:** Resides primarily within the PHP scripts (login.php, Rooms.php, Courses.php, departmenthead.php, process_exam.php, check_conflict.php, etc.). These scripts handle the core

business logic, process user requests, enforce rules (like conflict checking), interact with the database, manage sessions (session_start()), and prepare data to be sent back to the presentation layer (either as HTML redirects or JSON responses for AJAX).

- **Data Persistence Layer:** Managed through the MySQL database (examinationsystem) and accessed via the PHP scripts using the MySQLi extension. A dedicated script (db_connection.php) centralizes the database connection parameters, promoting better organization.

Key functional modules within this architecture are:

- **Authentication:** Separate flows for Admins and Department Heads.
- **Resource Management:** CRUD interfaces for Departments, Rooms, and Courses.
- **Exam Scheduling:** Functionality for adding, viewing, and potentially deleting exams.
- **Conflict Detection:** Logic to identify scheduling clashes.
- **Dashboard/Reporting:** Overview for administrators (home.php).

This architecture, while straightforward, effectively supports the application's requirements. Its modularity, based on distinct PHP/HTML/CSS files for different functions, allows for incremental development and easier maintenance, aligning with XP principles. The use of AJAX in certain modules (like Rooms and Department

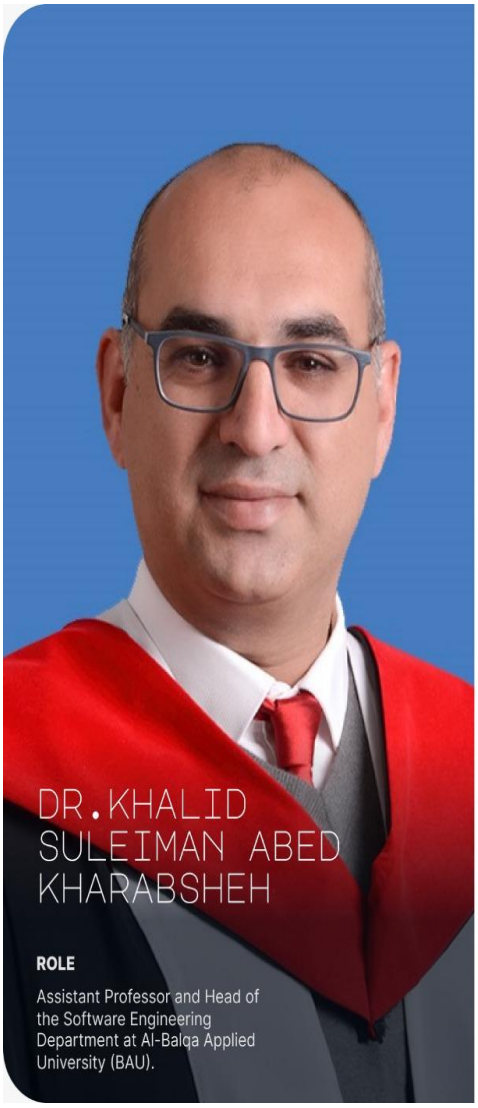
Heads management) enhances responsiveness by updating parts of the UI without full page reloads.

4.5 User Personas To better understand the users and guide the design from a user-centered perspective, the following personas represent the key individuals interacting with the Exam Scheduling System:

Persona 1: University Student

Figure 3 -4: Persona 1

Persona 2: Department Head (Software Engineering)



Profile:

Dr. Khalid Alkarabsheh is the accomplished Head of the Software Engineering Department at BAU, balancing his administrative duties with his role as an Assistant Professor and active researcher (focusing on empirical software engineering, software quality, and machine learning). He finds the traditional process of contributing to and verifying the exam schedule for his department cumbersome, requiring significant back-and-forth communication and manual checks to ensure his department's specific needs are met and no conflicts arise for his staff or courses. He wants a streamlined way to access the proposed schedule relevant to Software Engineering, easily verify its accuracy, and be quickly alerted to any potential issues or conflicts that require his attention, allowing him to dedicate more time to his research and teaching responsibilities rather than administrative overhead. He needs to trust that the system accurately reflects resource availability and constraints.

More Information

Background

Holds a Ph.D. in Computer Science/Software Engineering from the University of Santiago de Compostela (Spain, 2019), an M.S. in Computer Science from BAU (Jordan, 2005), and a Bachelor's degree from Yarmouk University (Jordan, 2002). Has extensive academic experience at BAU since 2006 (including Aqaba University College) and administrative experience as Head of Department since 2021.

Goals

- Ensure exams for the Software Engineering department are scheduled efficiently and without conflicts.
- Access accurate scheduling information relevant to the department's courses and instructors.
- Utilize the system to manage departmental scheduling responsibilities effectively.

Skills

Highly proficient with computer systems, software engineering principles, and academic software tools. Comfortable using web applications for administrative and academic tasks.

Figure 4 -4: Persona 2

4.6 Conceptual Class Diagram (Inferred)

While Extreme Programming prioritizes working code and comprehensive tests over extensive formal diagrams, a conceptual class diagram can still be useful for visualizing the main entities and their relationships within the system. This diagram is inferred from the structure of the PHP scripts, the database schema, and the overall functionality of the Exam Scheduling System. It represents the key logical components rather than a strict UML representation of PHP classes.

Key Inferred Classes/Modules:

- ***User (Abstract/Base Concept):***
  - Represents common attributes like username, password.
  - Could have subclasses Admin and DepartmentHead.
- ***Admin (Subclass of User):***
  - Responsibilities: Manages system resources, schedules exams, manages users.
  - Collaborates with: Database, RoomManager, CourseManager, DeptHeadManager, ExamScheduler, ConflictChecker.
- ***DepartmentHead (Subclass of User):***
  - Responsibilities: Views department-specific exam information, potentially participates in scheduling/conflict resolution for their department.
  - Collaborates with: Database, ExamScheduler (for viewing).

- ***RoomManager (Conceptual Module representing Rooms.php logic):***
    - Attributes: (Corresponds to ClassRooms table: roomNumber, roomType, capacity).
    - Responsibilities: Add, update, delete, retrieve room data.
    - Collaborates with: Database.
- ***CourseManager (Conceptual Module representing Courses.php logic):***
    - Attributes: (Corresponds to Courses table: courseName, courseNumber, subjectLevel, department, level, hours).
    - Responsibilities: Add, retrieve course data.
    - Collaborates with: Database.
- ***DeptHeadManager (Conceptual Module representing departmenthead.php logic):***
    - Attributes: (Corresponds to departmenthead table).
    - Responsibilities: Add, update, delete, retrieve department head data, toggle status.
    - Collaborates with: Database.
- ***ExamScheduler (Conceptual Module representing process_exam.php, exams_list.php, delete_exam.php logic):***
    - Attributes: (Corresponds to exams table: examDate, startTime, endTime).
    - Responsibilities: Schedule new exams, list exams, delete exams.
    - Collaborates with: Database, Course, Room, ConflictChecker.

- ***ConflictChecker (Conceptual Module representing check_conflict.php logic):***
    - Responsibilities: Detect scheduling conflicts based on rules (time, room, etc.).
    - Collaborates with: Database (querying exams, ClassRooms tables).
- ***Database (Represents the examinationsystem MySQL database):***
    - Responsibilities: Store and retrieve data for all entities.
    - Collaborates with: All manager/scheduler modules.

Conceptual Relationships:

- An Admin manages RoomManager, CourseManager, DeptHeadManager.
- An Admin uses ExamScheduler and ConflictChecker.
- A DepartmentHead views information related to ExamScheduler.
- ExamScheduler uses Course and Room information.
- ExamScheduler consults ConflictChecker before finalizing a schedule.
- All manager/scheduler modules interact with the Database.

This conceptual diagram provides a high-level overview of the system's structure and the interactions between its main logical parts, aiding in understanding the design without the overhead of detailed formal modeling.

Figure 5 -4: **Class Diagram**

4.7 Use Case Diagram / Key User Stories

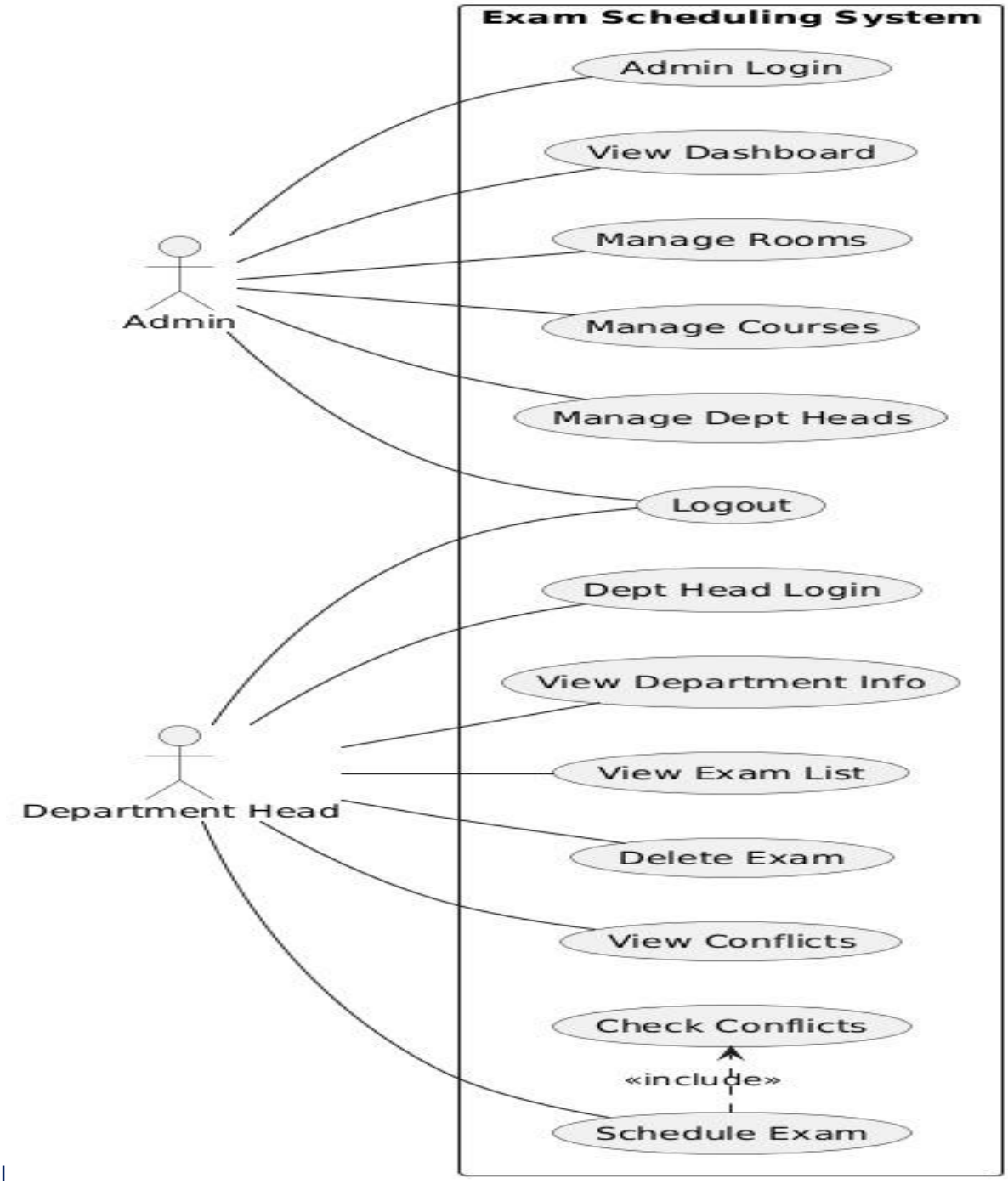


Figure 6 -4: Use Case Diagram

Key User Stories (Inferred):

For the System Administrator Persona:

- **Authentication:** "As an Admin, I want to securely log in to the system so that I can access administrative functions."
- **Dashboard:** "As an Admin, I want to see an overview dashboard with key statistics (room count, course count, department head count) and recent activity so that I can quickly understand the system's current state."
- **Room Management:** "As an Admin, I want to add a new classroom with its number, type, and capacity so that it becomes available for exam scheduling."
- **Room Management:** "As an Admin, I want to view a list of all classrooms so that I can manage them."
- **Room Management:** "As an Admin, I want to update the details (type, capacity) of an existing classroom so that the information remains accurate."
- **Room Management:** "As an Admin, I want to delete a classroom so that it is no longer used for scheduling."
- **Course Management:** "As an Admin, I want to add a new course with its name, number, level, department, and credit hours so that exams can be scheduled for it."
- **Course Management:** "As an Admin, I want to view a list of all courses."
- **Department Head Management:** "As an Admin, I want to add a new department head with their details (name, username, password, department) so that they can access the system."

- **Department Head Management:** "As an Admin, I want to view a list of all department heads and their status."
- **Department Head Management:** "As an Admin, I want to update the details of a department head."
- **Department Head Management:** "As an Admin, I want to activate or deactivate a department head's account."
- **Department Head Management:** "As an Admin, I want to delete a department head account."

For the Department Head Persona:

- **Authentication:** "As a Department Head, I want to securely log in to the system using my specific credentials so that I can access department-relevant information."
- **Information Access:** "As a Department Head, I want to view information relevant to my department (e.g., scheduled exams, assigned rooms) so that I can stay informed."
- **Exam Scheduling:** "As an Department Head, I want to schedule a new exam by selecting the course, date, start time, end time, and room so that the exam is officially planned."
- **Exam Management:** "As an Department Head, I want to view a list of all scheduled exams."
- **Exam Management:** "As an Department Head , I want to delete a scheduled exam."
- **Conflict Checking:** "As an Department Head, when scheduling an exam, I want the system to check for conflicts (e.g., room double-booking, time clashes) so that I can avoid scheduling problems."

- **Conflict Management:** "As an Department Head, I want to view detected conflicts so that they can be addressed."

These user stories, prioritized and implemented iteratively, would form the basis for the features observed in the Exam Scheduling System, driving the design in a user-focused, incremental manner characteristic of XP.

4.7.1 Key Interaction Sequences

Sequence diagrams help visualize the interactions between different components for specific scenarios. Here are examples for key processes:

Admin Login:

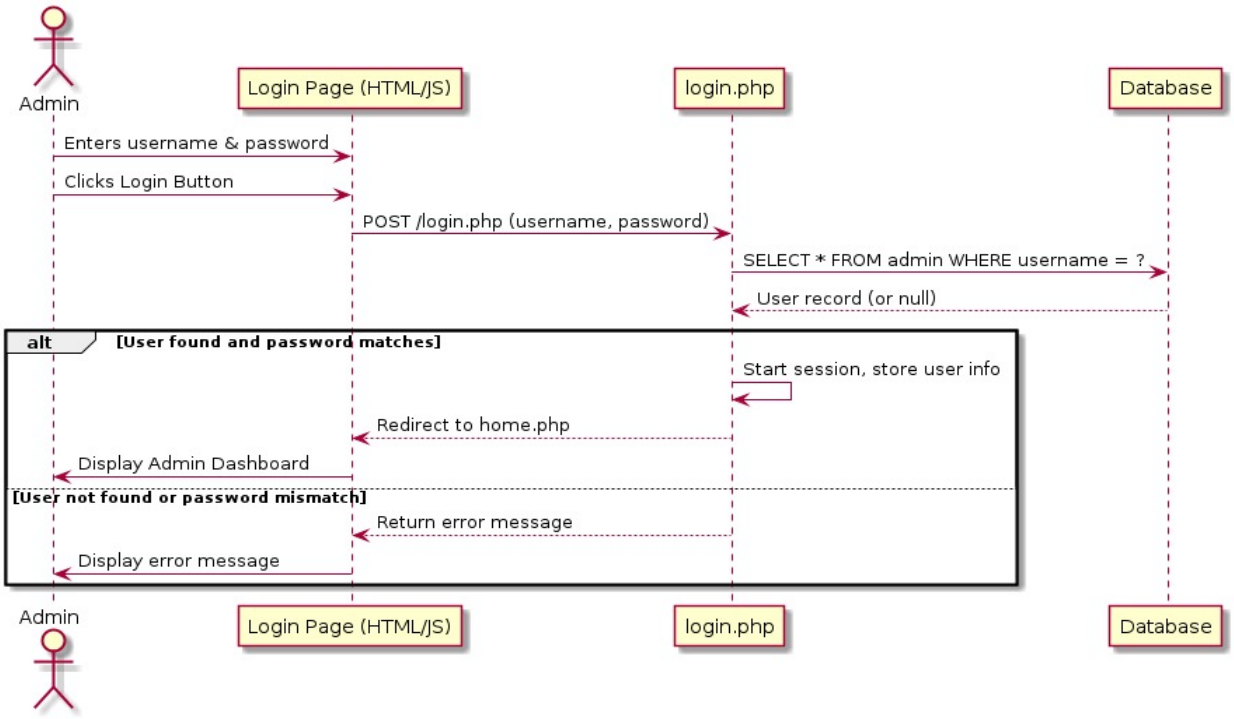


Figure 7 -4: Admin Login Sequence

Scheduling an Exam (including Conflict Check):

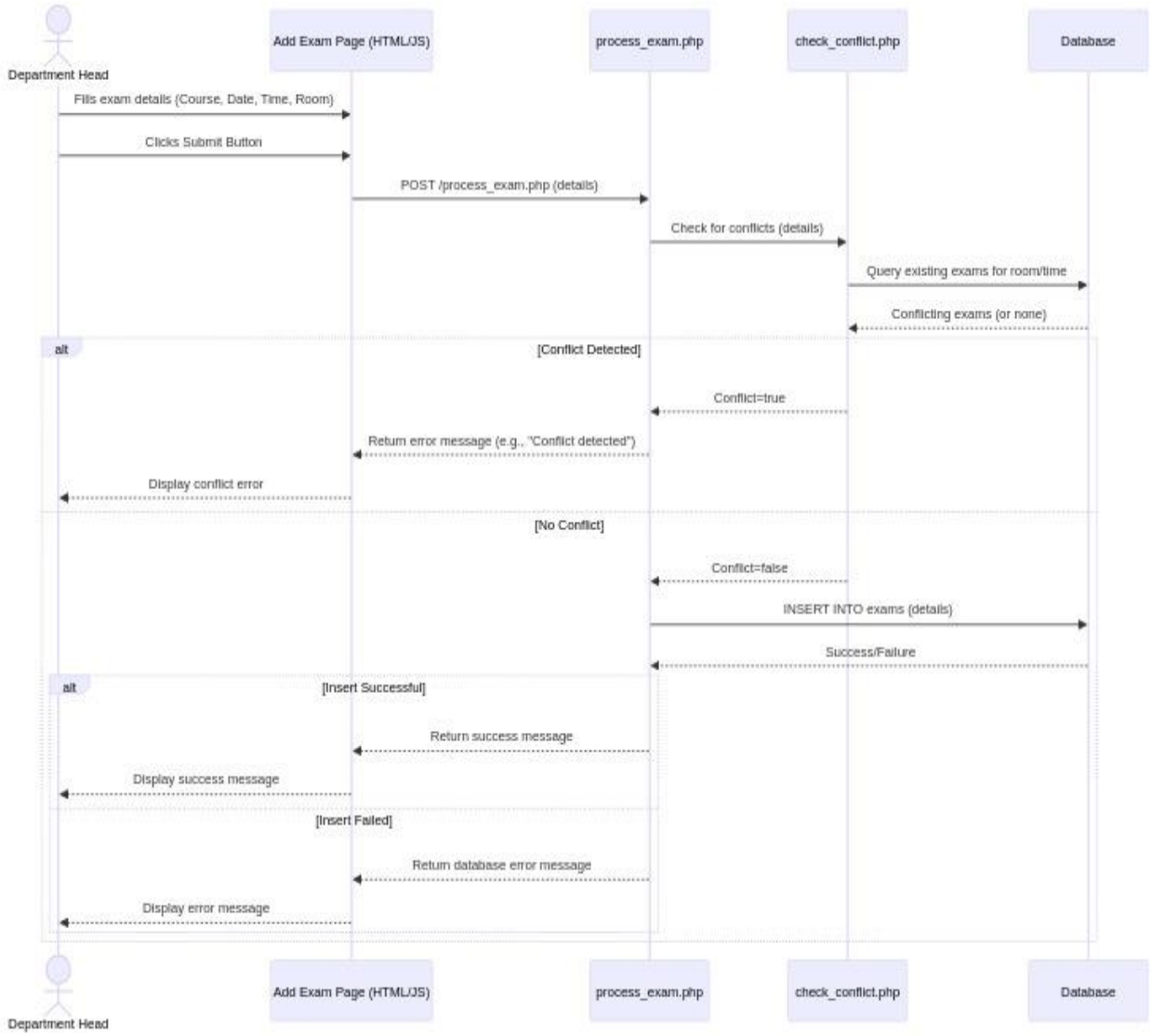


***Figure 8 -4* :Schedule Exam Sequence**

4.8 Site Map:

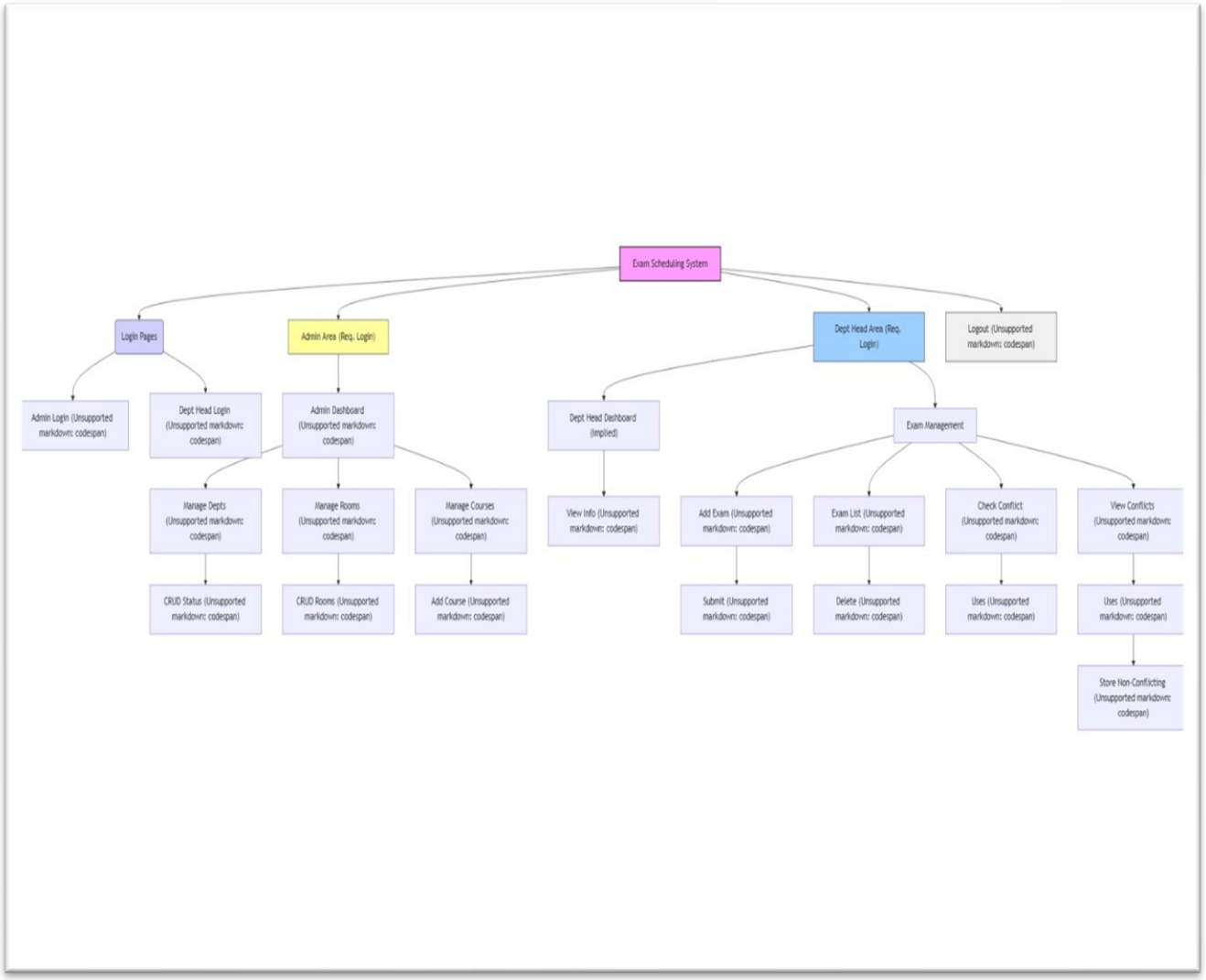
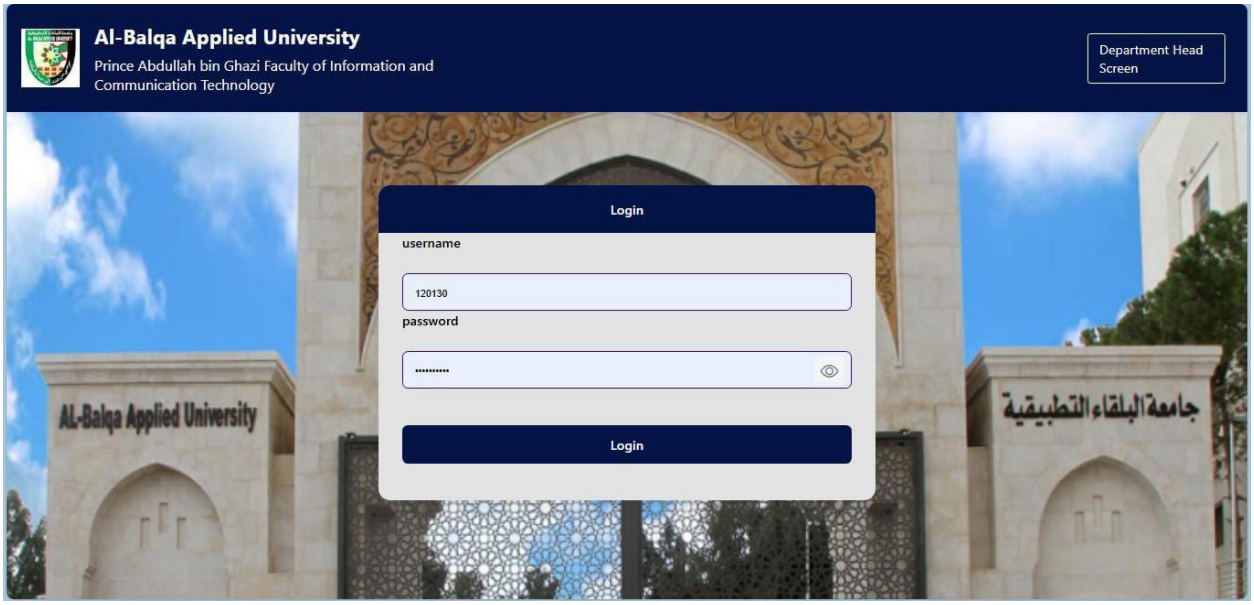
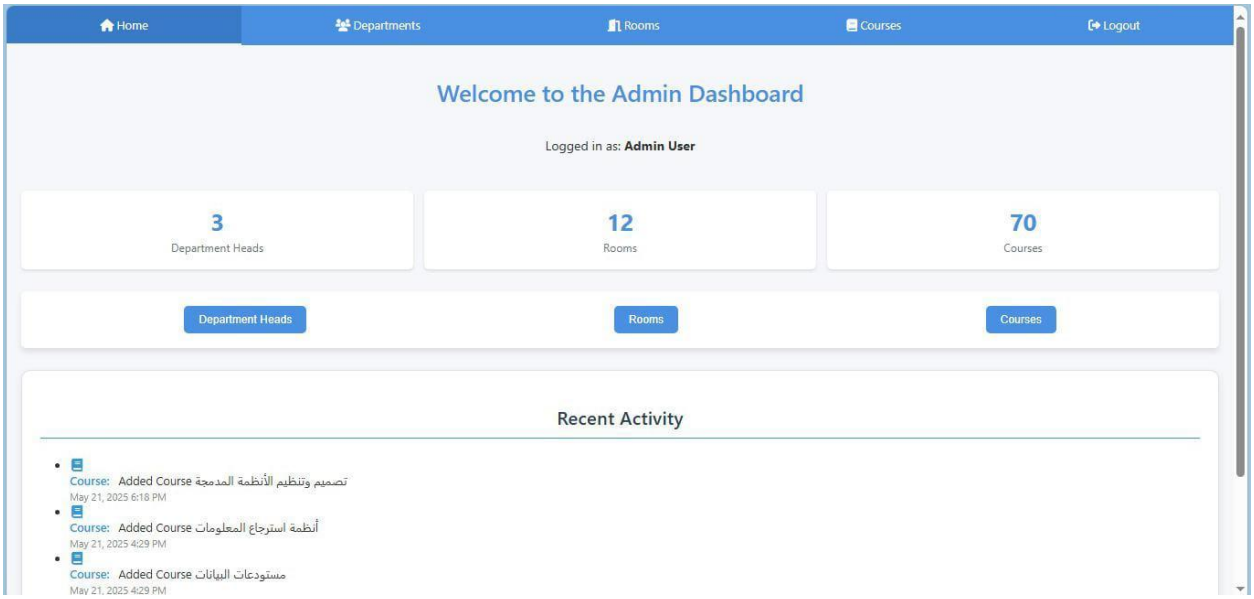


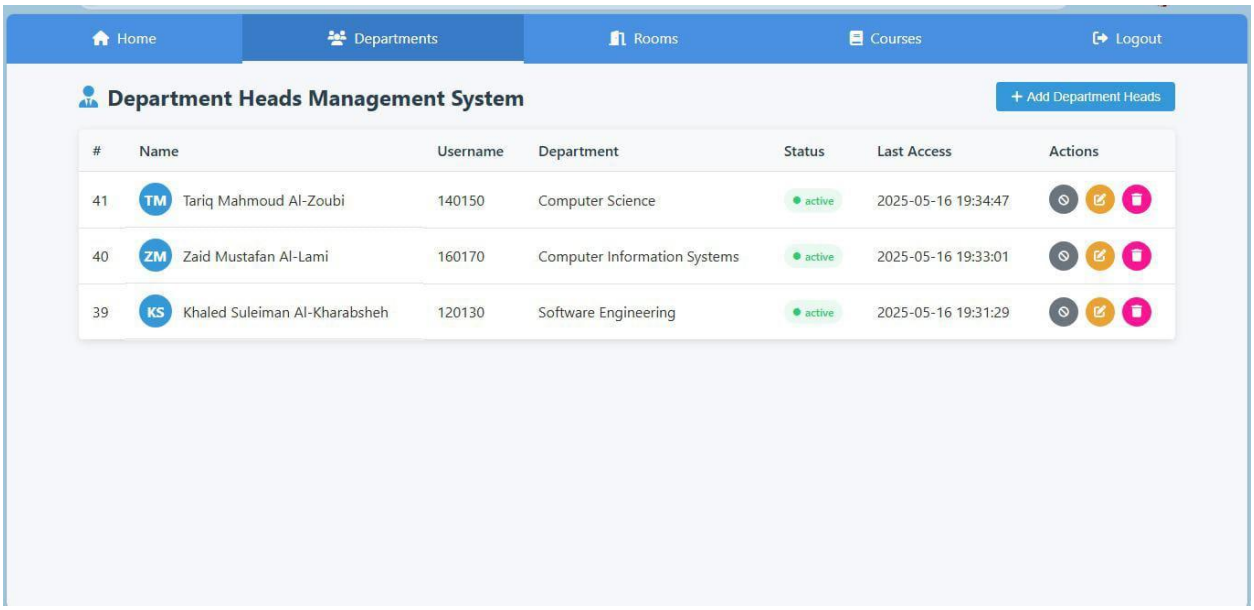
Figure 9 -4 **: Site Map**

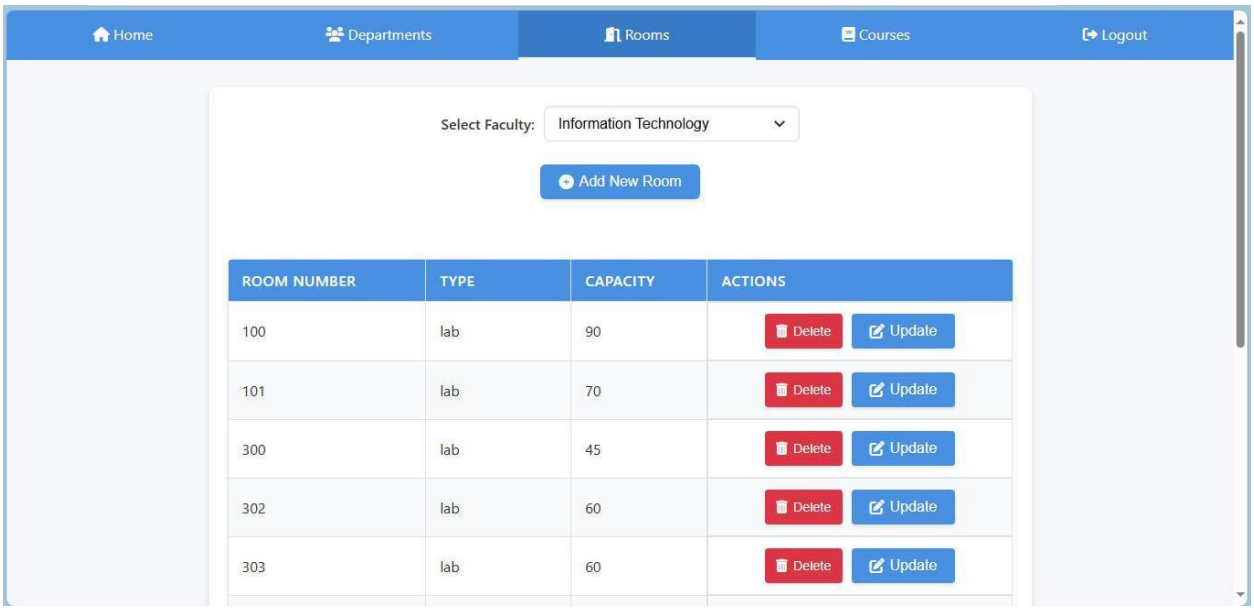
4.9 High-Fidelity Representation (UI Design)

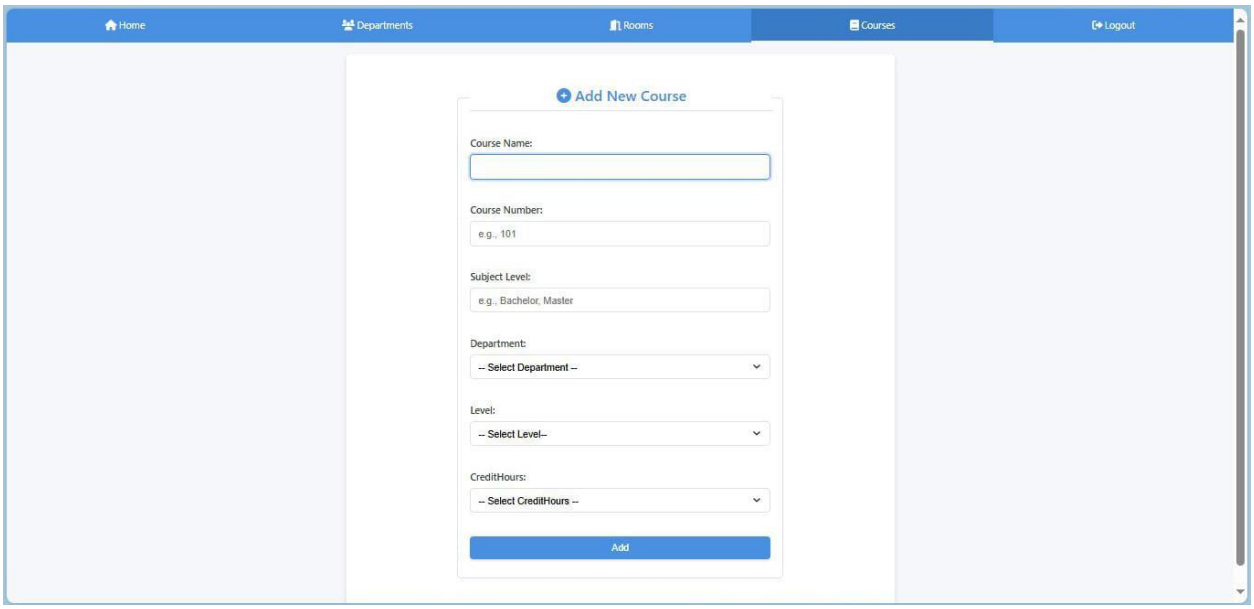
Admin Screen:



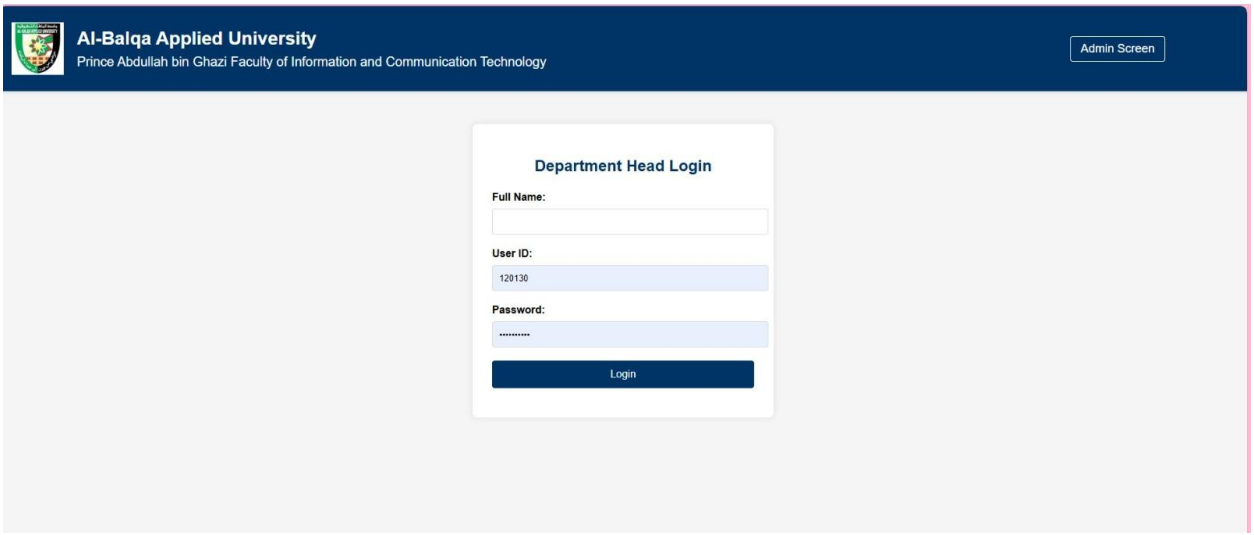


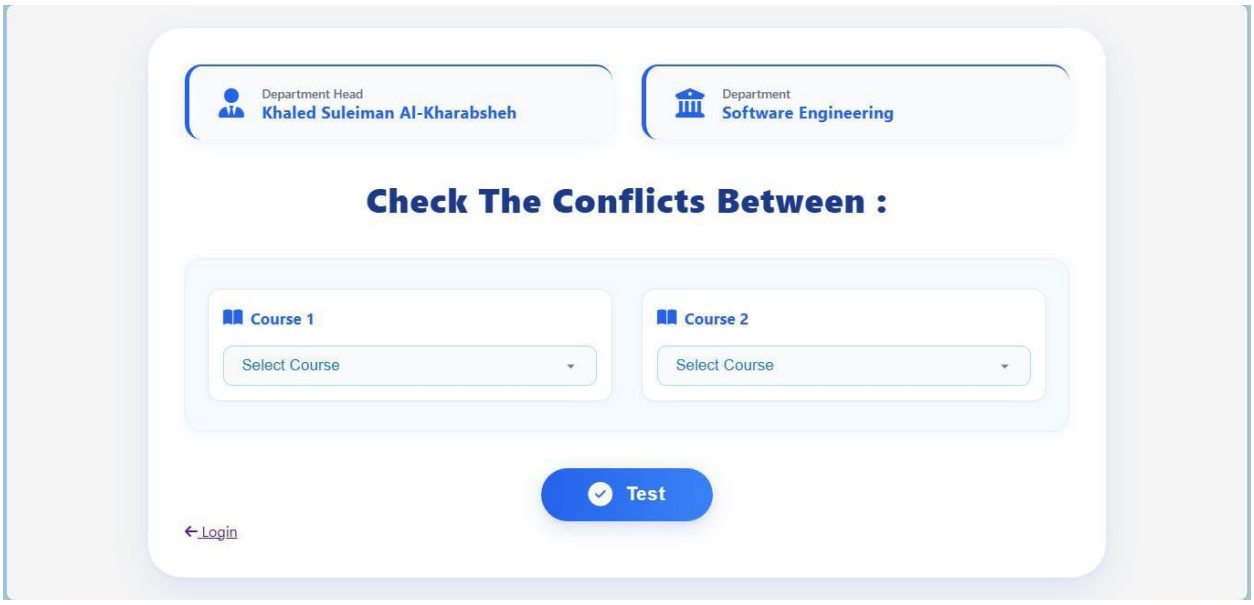

Home
Departments
Rooms
Courses
Logout
Department Heads Management System
+ Add Department Heads
# Name Username Department Status Last Access Actions
41 TM Tariq Mahmoud Al-Zoubi 140150 Computer Science active 2025-05-16 19:34:47
40 ZM Zaid Mustafan Al-Lami 160170 Computer Information Systems active 2025-05-16 19:33:01
39 KS Khaled Suleiman Al-Kharabsheh 120130 Software Engineering active 2025-05-16 19:31:29

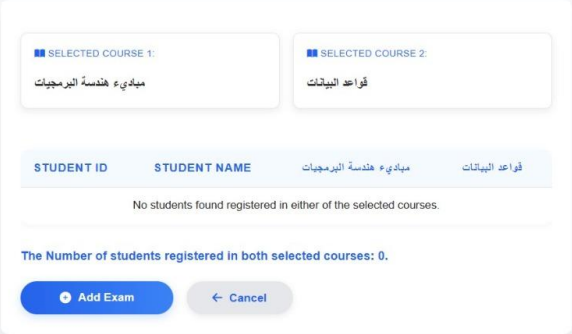

Home
Departments
Rooms
Courses
Logout
Select Faculty: Information Technology
Add New Room
ROOM NUMBER TYPE CAPACITY ACTIONS
100 lab 90 Delete Update
101 lab 70 Delete Update
300 lab 45 Delete Update
302 lab 60 Delete Update
303 lab 60 Delete Update


Home
Departments
Rooms
Courses
Logout
Add New Course
Course Name:
Course Number:
e.g., 101
Subject Level:
e.g., Bachelor, Master
Department:
-- Select Department --
Level:
-- Select Level--
CreditHours:
-- Select CreditHours --
Add

Department Head Screen:



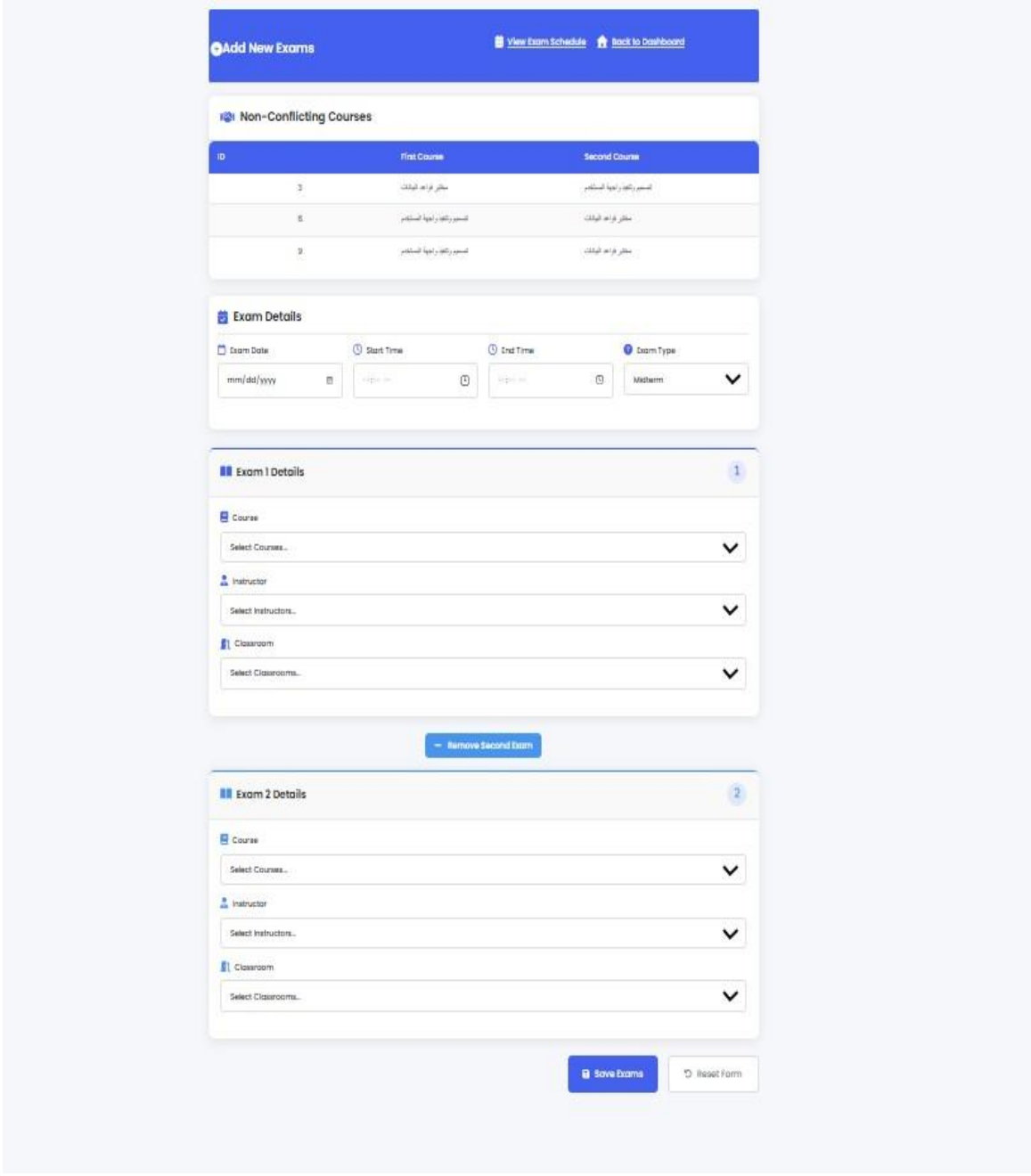




Exams List

Date	Time	Type	Course	Instructor	Classroom	Delete Exam
2025-05-22	04:34:00 - 07:34:00	Midterm	التفاضل والتكامل 2	Dr. Hiba Hayari	300	Delete

Back


Add New Exams
View Exam Schedule
Back to Dashboard
Non-Conflicting Courses
ID
First Course
Second Course
Exam Details
Exam Date
mm/dd/yyyy
Start Time
End Time
Exam Type
Midterm
Exam 1 Details
1
Course
Select Courses...
Instructor
Select Instructors...
Classroom
Select Classrooms...
Remove Second Exam
Exam 2 Details
2
Course
Select Courses...
Instructor
Select Instructors...
Classroom
Select Classrooms...
Save Exams
Reset Form

4.11 Future Improvements / Refactoring Opportunities

Consistent with the Extreme Programming principle of continuous improvement and refactoring, the current design of the Exam Scheduling System, while functional, presents several opportunities for future enhancements and code quality improvements. Addressing these would make the system more robust, maintainable, and potentially easier to extend.

Key areas identified for potential refactoring or future work include:

1. ***Code Duplication:*** The presence of files with names like AddExam - Copy.html, AddExam - Copy.css, AddExam - Copy.js, and check_conflict - Copy.css strongly suggests code duplication. This should be investigated and eliminated by consolidating functionality into single, reusable components or files. Duplication increases maintenance overhead and the risk of inconsistencies.
2. ***Security Enhancements:*** The inferred database schema suggests passwords might be stored in plain text (admin and departmenthead tables). This is a critical security vulnerability. Passwords must be securely hashed (e.g., using PHP's password_hash() and password_verify() functions) before storage.
3. ***Modularity and Separation of Concerns:*** While there's a basic separation (PHP for backend, HTML/CSS/JS for frontend), further improvements could be made. For instance:

    a. Consolidating common JavaScript functions into shared utility files instead of potentially repeating them across different pages.
    b. Refactoring large PHP scripts into smaller, more focused functions or potentially adopting a simple object-oriented approach for core entities (like Room, Course, Exam) to better encapsulate data and behavior.
    c. Ensuring stricter separation between data access logic (SQL queries) and business logic within PHP scripts.
4. ***Error Handling:*** Implementing more robust and user-friendly error handling on both the server-side (PHP) and client-side (JavaScript) would improve the user experience when issues occur (e.g., database connection errors, failed validation, unexpected exceptions).
5. ***Testing:*** Introducing an automated test suite (e.g., using PHPUnit for backend logic and a JavaScript testing framework for frontend interactions) would significantly improve confidence in future refactoring efforts and help catch regressions early, aligning strongly with TDD principles.
6. ***Conflict Checking Logic:*** The specific rules and efficiency of the conflict checking mechanism (check_conflict.php) could be reviewed and potentially optimized, especially if the number of exams and constraints grows.
7. ***User Interface/User Experience (UI/UX):*** While functional, the UI could be further refined based on user feedback, potentially improving workflows, enhancing accessibility, or providing more informative feedback messages.

Addressing these points iteratively, as part of the ongoing development process, would align with XP's emphasis on maintaining high internal quality and adapting the system over time.

4.12 References:

https://colaninfotech.com/blog/what-are-the-software-development-life-cycle-phases/ [1]

Life Cycle and Values of Extreme Programming [2]

https://bigpicture.one/blog/sprint-burndown-chart-101/ [3]

https://www.harness.io/blog/software-development-life-cycle-phases/ [4]

Thank You