CHAPTER ONE

Introduction

1.1 Background of Study

Timetable scheduling is a critical administrative task in universities, involving the assignment of courses, instructors, classrooms, and timeslots each term. Developing an effective timetable is fundamental to the healthy functioning of any academic institution. However, university scheduling is notoriously complex due to numerous constraints limited available hours, multiple courses, instructor workloads, and room capacities. For example, Alghamdi *et al.* (2020) note that factors such as room availability, number of subjects, and lecturer assignments make the timetabling problem very complex. Creating a conflict-free schedule for all students and faculty is one of the major challenges most institutions face. In many universities (including Bayero University Kano, BUK), this process has historically been manual. Public universities often spend days manually assigning classes to instructors and rooms. Such manual methods are labor-intensive and prone to human error, and they tend to fail as academic programs and student bodies grow. Prior research observes that "manual scheduling is often labor-intensive and prone to human errors, particularly in large institutions with multiple departments". When academic complexity increases, manual methods frequently produce scheduling conflicts and inefficient use of rooms. These general findings imply that BUK's current scheduling process likely suffers similar issues: heavy administrative workload, frequent timetable clashes, and underutilized resources.

1.2 Problem Statement

At BUK, timetabling is largely done by hand (via faculty meetings and ad-hoc software), which is time-consuming and error-prone. Administrators invest substantial effort and time each semester to produce the class schedule. Because of the many courses, instructors, and rooms involved, this manual process often results in conflicts (e.g., instructors double-booked or students scheduled for overlapping classes) and idle resources (empty classrooms or underused faculty time). In practice, BUK staff have limited ability to quickly explore alternative schedules or handle last-minute changes. The inefficient manual process "fails to scale with increasing academic complexity and often results in scheduling conflicts and inefficient room allocation". In conclusion, BUK's

timetable generation is slow, frustrating, and cannot guarantee an optimal schedule under all constraints.

To overcome these problems, this project proposes an *automated timetable generation system* for BUK. The system will use computational algorithms to automatically assign courses, instructors, and rooms to timeslots, respecting all institutional constraints (faculty availability, classroom capacity, course requirements, etc.). Automated scheduling has proven highly effective: for instance, Farinola & Assogba (2025) developed an AI-based timetable generator that "significantly reduced scheduling time, eliminated course conflicts, and improved allocation efficiency". Diallo & Tudose (2024) similarly showed that automating faculty scheduling can optimize resource usage and eliminate clashes. Based on these insights, the proposed solution will embed a genetic-algorithm or similar optimization engine into a user-friendly system. By systematically considering all constraints, the automated system will produce conflict-free timetables much faster than manual methods. In turn, this will save administrative time, improve resource utilization, and deliver more reliable schedules for students and instructors.

1.3 Aim and Objectives

The aim of this project is to develop an automated timetable generation system for Bayero University Kano that efficiently schedules courses, instructors, and classrooms while satisfying all academic constraints. The specific objectives are:

1. Analyze the current manual timetable scheduling process at BUK and identify all relevant constraints and requirements (e.g., faculty, classroom availabilities, course loads).
2. Design a system architecture (web-based) that allows administrators to input courses, lecturers, rooms, and constraints, and that can automatically generate timetables.
3. Implement optimization algorithms (such as a genetic algorithm with appropriate fitness criteria) to compute optimized schedules that avoid conflicts and maximize resource usage.
4. Incorporate user-defined preferences and constraints (e.g., instructor time preferences, departmental priorities) into the scheduling process.

5. Evaluate the performance of the automated system by comparing its schedules to the existing manual timetables (measuring factors like scheduling time, number of conflicts, and resource utilization).

1.4 Significance of the Study

This project is important for students, staff, and the university as a whole. For administrators and faculty, an automated system will greatly reduce the workload of generating timetables. By automating the assignment of courses and rooms, the system will cut down the weeks of manual effort now required. Prior work notes that an automated timetable tool "will reduce the time required by administrative staff and faculty, allowing them to focus on more strategic activities". Students will benefit from the increased reliability of their schedules: conflicts (overlapping classes or missing rooms) will be minimized, and course schedules can better respect student enrollment patterns. The university will see improved resource utilization (fewer empty classrooms and better lecturer time allocation). Overall, a well-designed automation system will streamline academic operations, reduce errors, and enhance satisfaction among all stakeholders. In the long run, this contributes to a smoother academic environment and allows the university to manage growth more effectively.

1.5 Scope

The timetable automation system will focus on scheduling regular course offerings (lectures and labs) at BUK according to the academic calendar. **In scope:** The system will generate semester timetables by assigning courses and instructors to available time slots and classrooms. It will consider hard constraints (e.g., no instructor in two places at once) and selected soft constraints (instructor time preferences) as part of the optimization. The output will be a conflict-free schedule for the relevant departments or faculties of BUK, accessible via a user interface or printout. **Out of scope:** This project will not cover exam or invigilation scheduling; it will not handle student course registration or financial/payroll processes. It also will not perform real-time adjustments during the term (any changes will require re-running the system or manual edits). Integration with other campus systems (e.g., learning management) is beyond the current scope. The system is intended as a decision-support tool for BUK's academic staff, not as a fully integrated ERP solution.

1.6 Summary

This chapter has introduced the context and motivation for the BUK Timetable Automation System. It has reviewed the complexity of university scheduling and identified the key issues with BUK's current manual approach. An automated solution was proposed as a way to address these challenges. The chapter stated the project's aim and specific objectives, highlighted the significance for students and staff, and clarified the system's scope. These considerations will guide the design and development of the timetable system.

CHAPTER TWO

Literature Review

2.1 Introduction

This chapter reviews recent scholarly work on automated timetable scheduling in higher education. It surveys literature on web-based scheduling tools, optimization algorithms (such as genetic algorithms and integer programming), and AI techniques applied to course and exam timetabling.

2.2 Related Works

Authors (Year)	**Research Focus**	**Methods / Technology**	**Key Findings / Contribution**
Alghamdi *et al.* (2020)	Review of university timetable optimization algorithms	Survey of metaheuristics (GA, PSO, etc.)	Reviewed several optimization algorithms for course timetabling, discussing how each handles constraint; highlighted the complexity and NP-hard nature of the problem.
Elen (2022)	Exam scheduling	Genetic Algorithm	Developed a GA-based system for scheduling final exams at Karabuk University; obtained complete, conflict-free exam timetables in simulations.
Shokouhifar *et al.* (2023)	University course scheduling optimization	Integer programming (GAMS)	Formulated an exact mixed-integer model maximizing professors' preferred time slots; implementation on real data reduced teaching days to 3 per week and balanced faculty loads.
Diallo & Tudose (2024)	Scheduling teaching activities in a faculty	Evolutionary algorithm (GA), multi-objective optimization	Built an automated system using GA and multi-objective criteria to schedule courses and instructors; achieved conflict-free schedules while optimizing resource usage and stakeholder satisfaction.

Authors (Year)	**Research Focus**	**Methods / Technology**	**Key Findings / Contribution**
García Yáñez *et al.* (2024)	Web-based course timetabling (Mexican university)	Web application (UI + algorithms)	Created a web platform for inputting instructor availability and course requirements; manual errors and conflicts were greatly reduced, and the system was designed to integrate future optimization algorithms.
Romaguera *et al.* (2024)	Web-based course scheduling (Philippines)	Enhanced Genetic Algorithm (heuristic mutation)	Designed a web-based course timetabling tool with an "enhanced" GA; it optimized the use of classroom resources using fewer rooms, producing efficient timetables that satisfy hard and soft constraints.
Farinola & Assogba (2025)	AI-based timetable generator	Genetic Algorithm	Proposed an "Explicit AI" timetable generator using a GA to fully automate scheduling; testing with real data showed the approach significantly reduced scheduling time, eliminated conflicts, and improved allocation efficiency.
Respati & Ramadhani (2025)	Lecture scheduling (Indonesia)	Genetic Algorithm (web app)	Developed a web-based lecture scheduling application for an engineering faculty using GA; generated conflict-free schedules for 34 courses and 345 lecturers, demonstrating GA's effectiveness in reducing workload and improving efficiency.
Carawana *et al.* (2025)	Secure web scheduling system	GA + Facial Recognition	Built a web scheduling system integrating GA optimization with OpenCV facial authentication; it achieved an 0% reduction in manual scheduling time and consistently

Authors (Year)	Research Focus	Methods / Technology	Key Findings / Contribution
			produced conflict-free timetables, with high user satisfaction.
Saw *et al.* (2025)	AI-optimized academic timetabling	GA + Machine Learning	Developed an "Adaptive Scheduler" that combines GA with machine learning in a web app; the system automatically generates optimal timetables based on faculty/student constraints, significantly cutting manual effort and enabling continuous improvement through feedback.

2.3 Literature Discussion

Recent research emphasizes algorithmic and AI-driven solutions for university scheduling. Saw *et al.* (2025) describe an *AI-Optimized Timetable Generator* that uses genetic algorithms combined with machine learning to allocate faculty, subjects, and classrooms. Their system automatically generates schedules satisfying multiple constraints (availability, preferences) and drastically reduces manual scheduling effort. In a similar vein, Carawana *et al.* (2025) developed a web-based scheduling system using a GA for optimization and added facial recognition for secure login. Their system reduced manual scheduling time by 0% while consistently producing conflict-free timetables. Farinola and Assogba (2025) also focus on GA-driven automation; their AI-based timetable generator eliminated all course conflicts and improved allocation efficiency when tested on institutional data. Respati and Ramadhani (2025) applied GA to lecture scheduling in a case study at Universitas Riau. Their web application scheduled 34 courses and 345 lecturers without any conflicts, demonstrating that GA can streamline scheduling and relieve administrative workload.

Other 2024 studies explore web platforms and enhanced heuristics. Romaguera *et al.* (2024) built a web-based course timetabling system that employs an "enhanced" genetic algorithm with heuristic mutation. This approach optimized the use of classrooms (using fewer rooms) and produced a timetable satisfying both hard (no conflicts) and soft constraints. Diallo and Tudose

(2024) developed an automated scheduling tool for faculty teaching activities. They used a multi-objective GA to generate faculty timetables that are conflict-free while maximizing room utilization and meeting instructor preferences. García Yáñez *et al.* (2024) report on a web platform for a Mexican university's scheduling needs. Their system allows instructors to enter their availability and course constraints. By capturing preferences in a user-friendly interface, the tool eliminated many errors inherent in the previous manual process and set the stage for algorithmic optimization.

Looking to earlier work, Shokouhifar *et al.* (2023) took an exact optimization approach. They formulated the university course scheduling problem as a large-scale integer linear model implemented in GAMS, with an objective to maximize professors' preferred time assignments. Their solution (solved on real college data) reduced the number of teaching days and greatly improved schedule compactness under complex constraints. Elen (2022) focused on exam timetabling: using a GA, he generated complete exam schedules for a vocational school that had no overlapping exams and satisfied all constraints. These earlier works complement the 2024–2025 studies by showing that both metaheuristic and exact methods can achieve optimal or near-optimal timetables in realistic settings.

Finally, Alghamdi *et al.* (2020) provide a broad review of optimization algorithms for university timetabling. They survey techniques like genetic algorithms, particle swarm, simulated annealing, etc., highlighting how each handles the many constraints of course scheduling1. Their review underscores that university timetabling is an NP-hard combinatorial problem requiring sophisticated solutions. Together, the reviewed works establish a foundation: modern timetable automation leverages advanced algorithms (GAs, hybrid methods, integer programming) and often uses web-based interfaces to capture requirements.

2.4 Summary

This chapter presented a review of recent research on automated academic scheduling. The literature shows a clear trend toward web-based systems and AI/optimization algorithms (especially genetic algorithms) to generate conflict-free timetables. A table summarized the key aspects of ten relevant works. These studies demonstrate effective methods for reducing scheduling time, handling constraints, and improving timetable quality. The insights gained will guide the design and implementation of the proposed automated scheduling system for BUK.

CHAPTER THREE

Methodology

3.1 Introduction

This chapter presents the methodological framework adopted for the development of the Bayero University Kano (BUK) Timetable Automation System. The methodology describes the structured approach used to carry out the various phases of the project including requirement elicitation, system analysis, design, and development. It begins with an overview of the project workflow that highlights the logical flow of activities from initial problem identification to system deployment and documentation. The System Development Life Cycle (SDLC) model adopted for this project is discussed, including the rationale behind its selection. This is followed by a detailed analysis of the existing manual timetable scheduling system at BUK, identification of the associated challenges, and the methods used to gather system requirements.

Furthermore, this chapter presents the requirements of the proposed system derived from stakeholders, followed by their modeling through use case diagrams, textual descriptions, and other structural analytical tools. The design of the proposed system, including activity diagrams, system architecture, and database Entity Relationship Diagrams (ERDs), is also extensively detailed. The chapter concludes with a summary of the methodological steps undertaken.

3.2 Project Workflow

The project workflow represents the sequence of major activities carried out during the development of the BUK Timetable Automation System. It ensures that the development process is organized, systematic, and aligned with the project objectives.

Explanation of Workflow Stages

1. Problem Identification: This stage involved understanding the challenges associated with the existing timetable generation system at BUK. Interactions with timetable officers and academic staff revealed issues such as scheduling conflicts, delays, data inconsistencies, and poor resource distribution.
2. Literature Review: A comprehensive review of scholarly materials was conducted to explore existing solutions, optimization techniques, and genetic algorithm implementations

in academic timetabling systems. This provided foundational knowledge and guided the design of the proposed system.

3. Requirements Elicitation & Analysis: Multiple elicitation techniques, including interviews, document analysis, observation, and surveys, were used to gather information from stakeholders. The collected data was analyzed and converted into functional and non-functional requirements.
4. System Design: Models and diagrams such as use case diagrams, sequence diagrams, activity diagrams, and ERDs were developed during this stage. The design served as the blueprint for system implementation.
5. System Implementation: The system was developed using React.js for the frontend, Node.js and Express.js for the backend, PostgreSQL as the database, and a custom Genetic Algorithm to generate optimized timetables.
6. Testing: Different testing strategies, including unit testing, integration testing, system testing, and usability testing, were performed to verify the correctness and functionality of the system.
7. Deployment: The system was deployed on a cloud-based platform to enable accessibility to lecturers and administrators.
8. Documentation & Reporting: All development activities, methods, and outcomes were documented.

3.3 System Development Model

A System Development Life Cycle (SDLC) provides a structured approach for planning, developing, testing, and maintaining a software system. For this project, the Iterative Development Model was chosen due to its flexibility and suitability for optimization-driven software development.

3.3.1 Justification for Selecting the Iterative Model

The iterative model was selected based on the following considerations:

1. Evolving Requirements

   Timetable generation involves multiple constraints that may change over time. The iterative model supports continuous refinement of requirements as new insights emerge.
2. Need for Progressive Algorithm Enhancement

   The Genetic Algorithm (GA) used in the timetabling system must undergo several cycles of refinement to achieve optimal performance. Each iteration allows for improvement of the fitness function, mutation rate, and crossover techniques.
3. Early Feedback Integration

   The iterative model allows early versions of the system to be tested by faculty officers and lecturers, enabling prompt adjustments.
4. Reduced Development Risks

   Breaking down development into small cycles minimizes the chances of major design failures and enhances quality.

3.4 Analysis of Existing and Proposed Systems

This section compares the existing manual scheduling system at BUK with the proposed automated system. It includes an assessment of the current workflow, the elicitation process, requirements gathered, and the modeling of the system using analytical tools.

3.4.1 Description of the Existing System

The existing timetable generation system at BUK is a manual, paper-based or spreadsheet-based system handled by departmental timetable officers. The process involves:

1. Collecting course information from departments
2. Assigning lecturers to courses
3. Allocating venues manually
4. Checking for conflicts manually
5. Producing printed timetables

Problems Identified

1. High Error Rate: Scheduling conflicts such as course clashes and double-booked venues are common.
2. Time-Consuming: Creating a timetable manually can take several days to weeks.
3. Poor Resource Utilization: Lecture halls are not optimally assigned based on capacity and availability.
4. Difficulty Updating Schedules: Small changes require major restructuring of the timetable.
5. Lack of Transparency: Stakeholders often find it difficult to track decision-making.

3.4.2 Requirement Elicitation

To ensure that the system reflects real user needs, multiple elicitation techniques were used:

1. **Interviews**: Interviews were conducted with timetable officers, lecturers, and faculty administrators to gain insights into their challenges.
2. **Document Review**: Existing timetables, course allocation sheets, and venue lists were analyzed.
3. **Direct Observation**: Timetable creation meetings were observed to understand the workflow and problems.
4. **Surveys**: Google Forms were distributed to lecturers to collect data on availability, constraints, and preferences.

Findings from Elicitation

1. Users prefer a web-based system.
2. Lecturer availability is critical in conflict prevention.
3. Large venues are often wrongly allocated to small classes.
4. The system must generate optimized, conflict-free timetables.
5. Changes should be easily manageable.

3.4.3 Requirements Definition

Requirements are categorized into Functional and Non-Functional Requirements.

A. Functional Requirements

The system shall:

a. Allow administrators to manage departments, courses, lecturers, venues, and academic levels.
b. Allow lecturers to input their availability and constraints.
c. Accept input datasets containing course details, lecturer allocations, and venue information.
d. Automatically generate timetables using a Genetic Algorithm.

Prevent conflicts such as:

a. Lecturer scheduling clashes
b. Venue double bookings
c. Course-level conflicts
d. Display the generated timetable in an organized week-by-week format.
e. Allow printing and exporting of timetables.
f. Provide authentication (admin/lecturer access roles).
g. Allow updates to constraints and timetable regeneration.

B. Non-Functional Requirements

Performance: Timetable generation should take less than 20 seconds.

Usability: Interface must be simple for non-technical users.

Scalability: Must support multiple faculties and hundreds of courses.

Security: User authentication and data protection are essential.

Reliability: The system must consistently produce correct outputs.

Maintainability: System should allow future enhancements.

Portability: Must run on all modern browsers.

3.4.4 Requirements Analysis

This involves modeling the system using UML diagrams and structured descriptions.

Use Case Diagram for the Proposed System

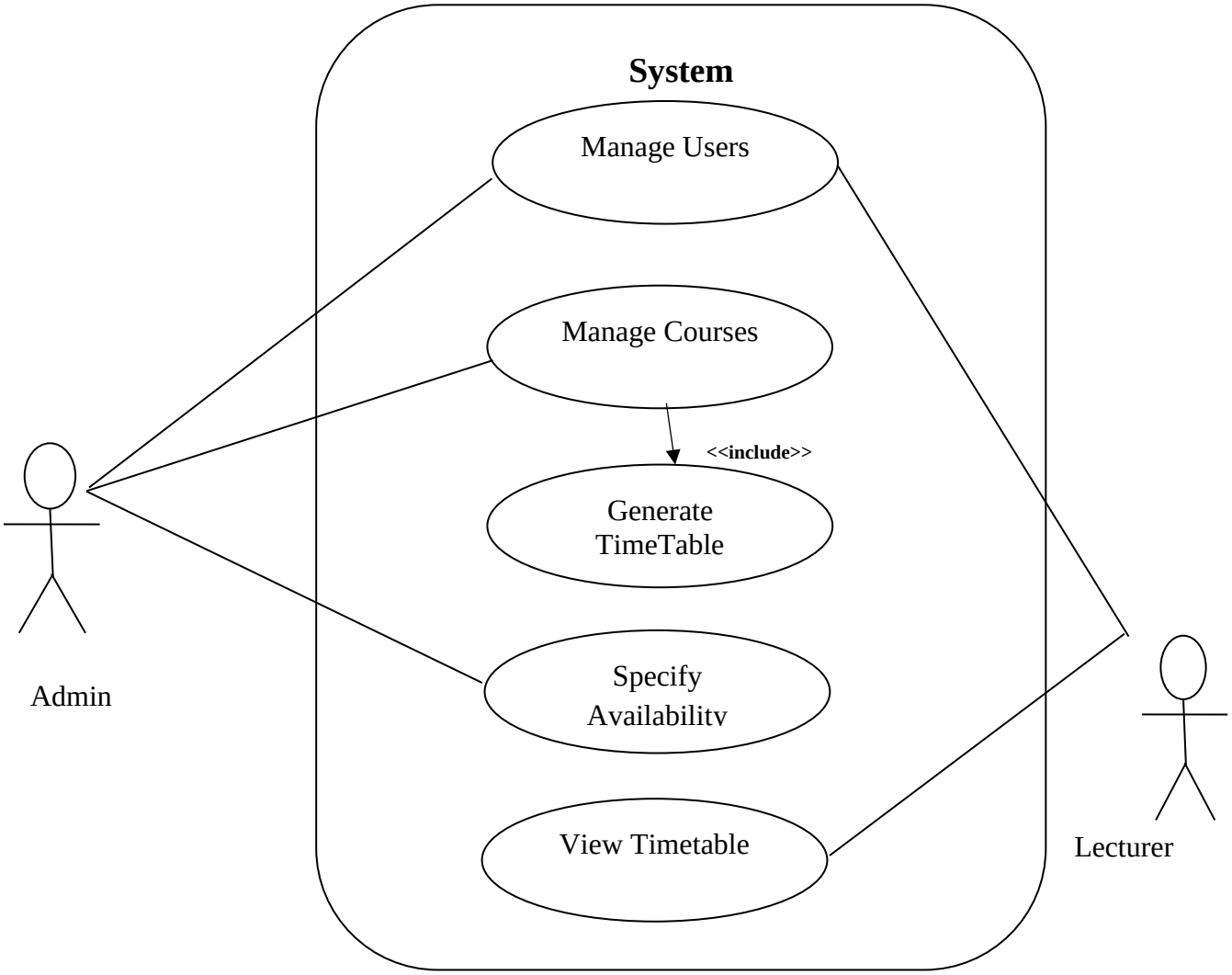


Figure 3.2: Use Case Diagram of BUK Timetable Generation

Use Case Description - BUK Timetable Automation System

Use Case Name: Generate Timetable **Priority**: High
Actor: Admin
Description: This use case describes how the administrator generates an optimized, clash-free timetable using a Genetic Algorithm. The admin initiates the process after all required input data such as courses, lecturers, venues, and constraints have been provided.
Trigger: The admin clicks the **"Generate Timetable"** button.
Preconditions: The system contains valid data for courses, lecturers, venues, and constraints. The admin is authenticated and logged into the system.
Normal Flow: 1. The admin selects the **"Generate Timetable"** option. 2. The system loads all timetable constraints and input parameters. 3. The Genetic Algorithm (GA) performs multiple iterations to generate possible timetables. 4. The system evaluates each solution and selects the best-fit timetable. 5. The optimized timetable is displayed to the admin. 6. The use case ends successfully.
Postconditions: The final optimized timetable is saved in the system database.
Exceptions: 1. If required input data is missing, the system displays an error message requesting data completion. 2. If the algorithm fails to converge, the system prompts the admin to retry the process.

3.5 System Design

System design describes how the system will be structured and how its components will interact.

3.5.1 Design of Proposed System

The system consists of:

1. Frontend (React.js): Interface used by admins and lecturers
2. Backend (Node.js + Express.js): Handles business logic and optimization
3. PostgreSQL Database: Stores all course, lecturer, and timetable data

Activity Diagram for Timetable Generation

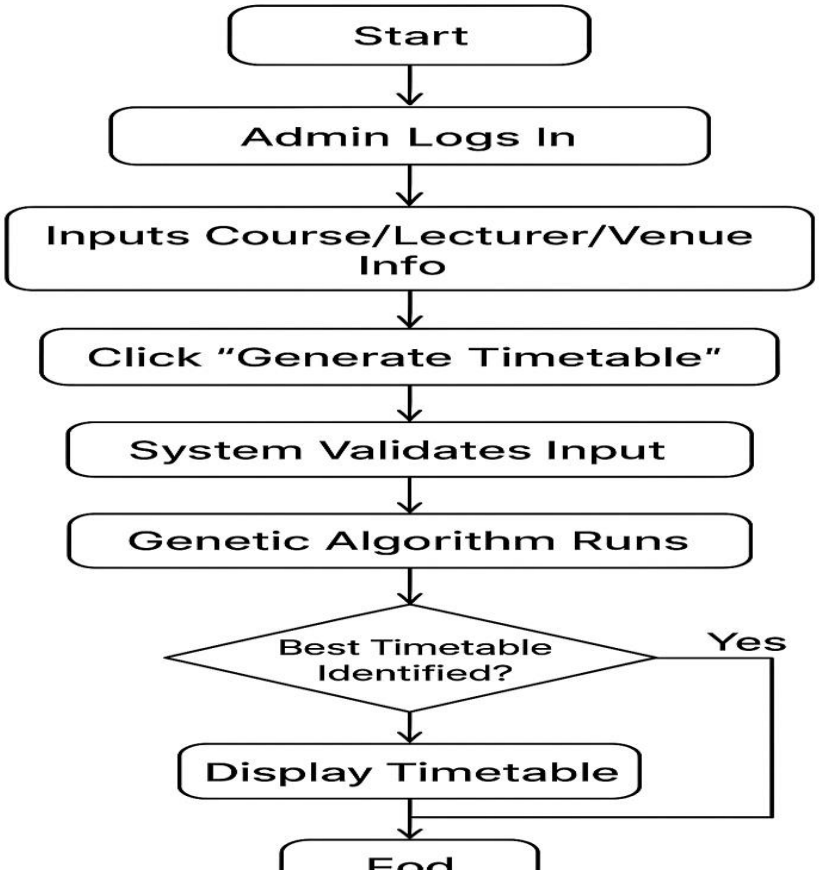


Figure 3.5: Timetable Generation Process Activity Diagram

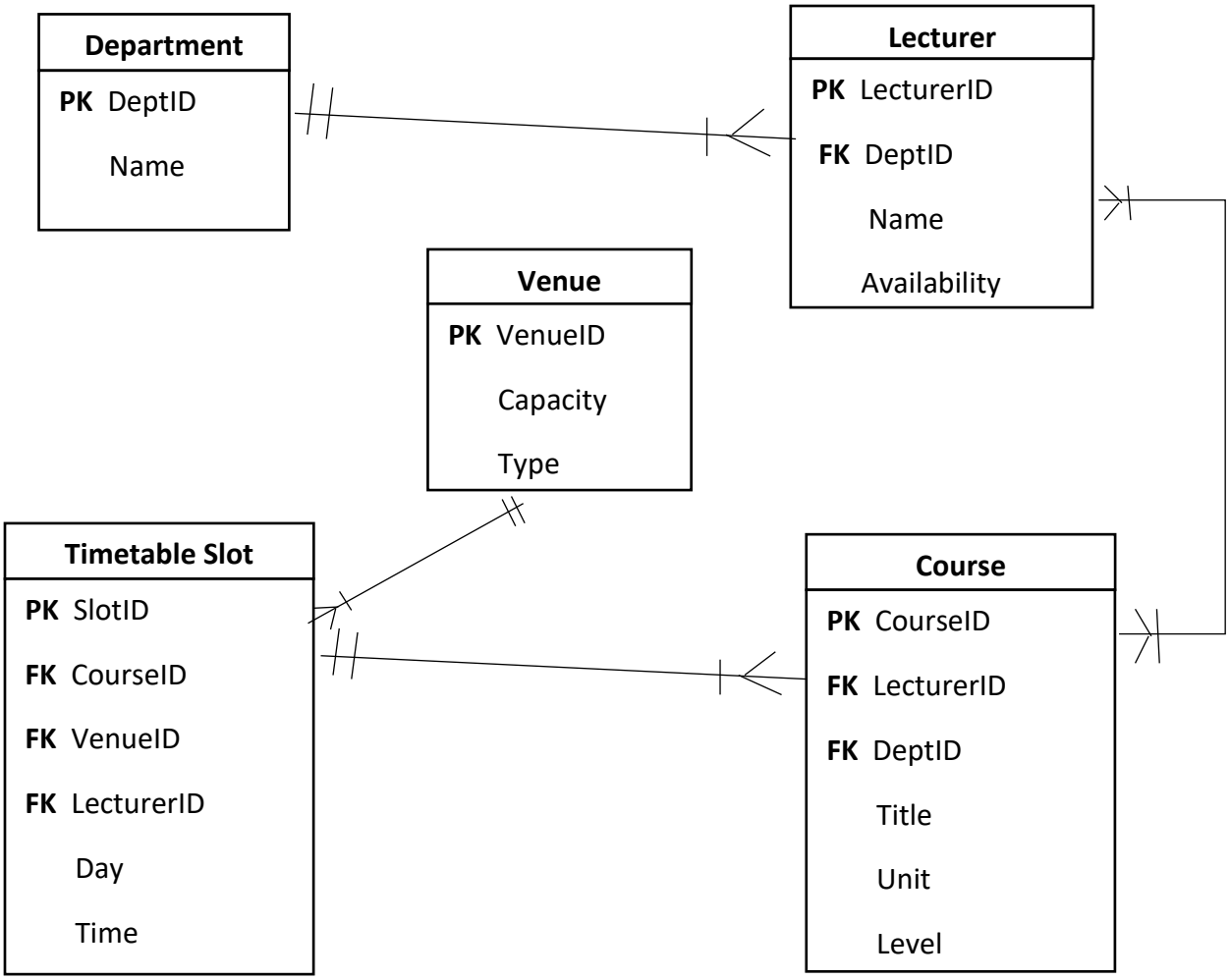
Figure 3.4: Verification Process Activity Diagram

3.5.2 System Architecture Design

The system adopts the Three-Tier Architecture:

- Presentation Layer -> React.js Interface for Users
- Application Layer -> Node.js/Express Backend + Genetic Algorithm Engine
- Data Layer -> PostgreSQL DB

3.5.3 Database Design (ERD)



3.6 Summary

This chapter described the methodology used in developing the BUK Timetable Automation System. The chapter began by presenting the project workflow and describing the SDLC model employed. The Iterative Model was selected due to its flexibility and suitability for optimization-based software.

The limitations of the existing manual system were analyzed, and requirements were gathered using multiple elicitation techniques. Functional and non-functional requirements were clearly defined, and the proposed system was modeled using UML tools such as use case diagrams, activity diagrams, and an ERD. The system design also detailed the architecture and database structure, providing a comprehensive foundation for implementation and testing in subsequent chapters.