Journal Collabits
Vol. 3 No. 2 May 2026 : 128-137
E-ISSN : 3046-6709, P-ISSN : 3062-8601
https://publikasi.mercubuana.ac.id/index.php/collabits

Design and Implementation of a Python-Based Cinema Ticket Ordering Prototype Using Linear Search

Ramando Girsang[1*], Hisyam Fadhila[2], Yunita Sartika Sari[3], Nia Rahmah Kurnianda[4], Ifan Prihandi[5]

[1, 2, 3, 4, 5] Information System, Universitas Mercu Buana, Indonesia

*Corresponding Author: ramandogirsang0@gmail.com

Abstract - The increasing adoption of digital services has encouraged the development of ticketing systems that simplify access to movie information and transaction processes. This study aims to design and implement a Python-based cinema ticket ordering prototype and to examine the suitability of the Linear Search algorithm for movie-title lookup in a small and unsorted dataset. The development process consists of requirements analysis, system design, implementation, functional verification, and algorithmic complexity analysis. The prototype supports customer input, movie selection, ticket quantity calculation, optional food and beverage ordering, and transaction calculation. Linear Search is selected because it can operate directly on a small unsorted list without preprocessing or an additional indexing structure. Theoretical comparison shows that Linear Search has O(n) average and worst-case time complexity, while Binary Search offers O(log n) complexity for sorted data and hash-based lookup offers O(1) average lookup time with additional storage requirements. The study therefore does not claim Linear Search as a new or universally efficient algorithm. Its contribution lies in presenting a transparent small-scale ticket-ordering architecture and an academically justified algorithm selection. Future development should incorporate persistent database storage, a graphical or web/mobile interface, authentication, secure payment integration, usability evaluation, and empirical performance benchmarking.

Keywords:
Cinema Ticket Ordering;
Python;
Linear Search;
Search Algorithm;
Software Prototype;
Ticketing System;

Article History:
Received: 02-03-2025
Revised: 18-04-2026
Accepted: 03-05-2026

Article DOI: 10.22441/collabits.v3i2.27260

INTRODUCTION

Digital technology has changed how consumers access entertainment services, including the purchase of cinema tickets. Conventional ticket purchasing requires customers to visit a cinema counter, check movie availability, select a schedule, and complete the transaction directly at the venue. Digital ticketing systems reduce these limitations by integrating information access, reservation, and transaction processes into software applications.

Recent cinema-ticketing studies show that system development is no longer limited to basic purchase functions. Haryanti et al. [1] emphasized user-centered interface design and evaluated a cinema e-ticket application using the System Usability Scale. Anggraini and Maiyana [2] developed an Android-based cinema ticket booking application that incorporates seat visualization, digital payment, QR-code e-tickets, and movie recommendation features. Other studies have evaluated the user experience and usability of existing cinema ticketing applications, demonstrating that functional completeness must be accompanied by clear navigation, interface quality, and usability [3], [4].

Software architecture and development strategy are also important. Ananda HAT et al. [5] developed cinema ticket booking software based on Odoo ERP using Scrum, showing that ticketing applications can be approached as integrated information systems. In parallel, movie recommendation research increasingly employs machine learning, sentiment analysis, and hybrid recommendation approaches to assist users in navigating large movie collections [6]. Such approaches are valuable when the dataset and personalization requirements are substantial, but they can be disproportionate for a small instructional prototype whose primary requirement is simple title lookup and transaction processing.

The manuscript submitted in this study originally demonstrated a console-based Python application with customer-name input, movie selection, ticket quantity, food and beverage ordering, and payment calculation. The original manuscript, however, did not clearly connect these implementation elements to a structured theoretical framework, research questions, a system architecture, or an academically justified search mechanism. The revised study addresses these gaps by positioning the work as a Python-based cinema ticket ordering prototype rather than as a complete commercial mobile application.

Search algorithm selection requires a clear rationale. Linear Search examines records sequentially and has an average and worst-case time complexity of O(n). Binary Search provides O(log n) search complexity but requires sorted data, while hash-based lookup can provide approximately O(1) average lookup time at the cost of additional memory and indexing structures. Recent comparative research confirms that Linear Search becomes less competitive as dataset size increases, whereas Binary Search and hash-based search are more appropriate for larger collections under suitable conditions [7]. Accordingly, the present study does not position Linear Search as an algorithmic novelty or as the most efficient general-purpose solution.

The research gap addressed here concerns the mismatch that often occurs between the complexity of a search mechanism and the scale of the application being developed. For a small, unsorted movie list, a simple sequential search can remain technically reasonable because it requires no preprocessing and is transparent to implement. The study therefore focuses on how a basic cinema ticket ordering prototype can integrate transactional functions and how the search mechanism can be justified according to dataset characteristics.

Based on this positioning, the research questions are formulated as follows:

RQ1. How can a Python-based cinema ticket ordering prototype be designed to integrate customer input, movie selection, ticket ordering, food and beverage ordering, and transaction calculation?

RQ2. Under what data conditions is Linear Search suitable for movie-title lookup in the developed prototype, and how does its theoretical computational complexity compare with Binary Search and hash-based lookup?

The objectives of this study are to design and implement the prototype and to provide an explicit academic rationale for the use of Linear Search in a small, unsorted movie dataset. The contribution lies in the integration of basic ordering functions, software architecture, data flow, and search-algorithm selection. The study does not claim that the prototype is more effective than commercial cinema ticketing platforms.

RELATED WORK AND STATE OF THE ART

The positioning of the revised manuscript is clarified by comparing it with recent studies on cinema e-ticket applications, mobile ticketing, user experience, integrated software development, recommendation systems, and search-algorithm performance. Table 1 summarizes the state of the art and identifies the specific gap addressed by this study.

Table 1. State-of-the-art comparison and research positioning.

Study	Main Focus	Method / Technology	Main Contribution	Position Relative to This Study
Haryanti et al. [1]	Cinema e-ticket UI and usability	User-Centered Design and SUS	Evaluates the usability of a cinema e-ticket interface	Focuses on interface evaluation rather than search-algorithm selection
Anggraini and Maiyana [2]	Android cinema ticketing	Android Studio, digital payment, QR code	Provides modern mobile ticketing features	Focuses on a mobile implementation rather than a lightweight Python search mechanism
Choirunisa and Ryansyah [3]	User experience of cinema ticketing apps	User Experience Questionnaire	Compares UX dimensions of existing applications	Evaluates existing applications rather than developing a small prototype
Ningrum et al. [4]	Cinema application usability	Heuristic Evaluation and PIECES	Identifies usability and service-quality issues	Does not examine search-algorithm selection
Ananda HAT et al. [5]	Cinema booking software development	Odoo ERP and Scrum	Develops an integrated ticket booking workflow	Uses enterprise-oriented architecture rather than a lightweight Python prototype
Sarhan et al. [6]	Movie recommendation	Machine learning and sentiment analysis	Improves personalized movie recommendation	Requires more data and computation than simple movie-title lookup
Purnama [7]	Search algorithm performance	Linear, Binary, and Hash Search	Compares search complexity and performance	Does not apply search-algorithm selection specifically to cinema ticket ordering
This study	Cinema ticket ordering and movie lookup	Python prototype and Linear Search	Integrates basic ordering functions and justifies search choice based on dataset scale	Focuses on transparent small-scale implementation rather than algorithmic novelty

The comparison indicates that recent cinema ticketing research primarily emphasizes mobile interfaces, usability, digital payment, QR-based e-tickets, and integrated service workflows. Recommendation-system research addresses a different scale by using advanced techniques to personalize movie selection. The present study occupies a narrower but clearly defined position: it examines a lightweight transaction prototype and makes the choice of a simple search algorithm explicit and bounded by the characteristics of the dataset.

THEORETICAL AND CONCEPTUAL FRAMEWORK

Digital Ticketing Systems

A digital ticketing system is an information system that supports ticket selection, reservation, purchase, and transaction management electronically. A production system can include user management, schedule management, seat allocation, payment processing, transaction recording, e-ticket generation, and notifications. The prototype in this research implements only a subset of these capabilities: customer identification, movie lookup and selection, ticket quantity, optional food and beverage ordering, and transaction calculation. For this reason, the revised manuscript describes the artifact as a Python-based cinema ticket ordering prototype rather than a complete commercial online ticketing platform.

Software Development

Software development transforms requirements into an executable application through structured activities such as requirements analysis, design, implementation, and verification. The current prototype uses Python because its basic data structures, conditional statements, loops, functions, and file-handling capabilities provide a transparent environment for demonstrating a small transaction-processing workflow. The development scope is intentionally limited so that the relationship among data input, search, selection, calculation, and output remains visible.

Search Algorithms

Searching is the process of locating a target record within a collection of data. Algorithm selection depends on dataset size, ordering, frequency of searches, update patterns, storage requirements, and expected response time. Linear Search examines each item sequentially until the target is found or the collection is exhausted. Binary Search repeatedly divides a sorted search space, while hash-based lookup maps a key to an indexed location. Their theoretical characteristics are summarized in Table 2.

Table 2. Theoretical comparison of search alternatives.

Criterion	Linear Search	Binary Search	Hash-Based Lookup	Implication for Prototype
Data must be sorted	No	Yes	No	Movie data can remain in original order
Best case	O(1)	O(1)	O(1) average	All methods can be fast under favorable conditions
Average case	O(n)	O(log n)	O(1) average	Linear Search is acceptable only for

Criterion	Linear Search	Binary Search	Hash-Based Lookup	Implication for Prototype
				small collections
Worst case	O(n)	O(log n)	O(n)	Hash performance depends on collisions
Additional structure	None	Sorted sequence	Hash table / index	Linear Search has the lowest setup overhead
Scalability	Limited	Good for sorted data	Generally good	Alternative methods should be considered as the movie catalog grows

Linear Search is selected because the prototype uses a small and unsorted movie collection. It does not require preprocessing and can be implemented without an additional indexing structure. This choice is conditional. If the application is expanded to hundreds or thousands of records, repeated search requests would make Binary Search, database indexing, or hash-based lookup more appropriate [7].

User Interface and User Experience

The user interface determines how users interact with application functions, whereas user experience concerns broader perceptions of clarity, ease of use, efficiency, and satisfaction. Recent cinema ticketing studies show that usability should be evaluated explicitly rather than assumed [1], [3], [4]. The current implementation uses a text-based console interface and is therefore intended as a functional prototype. It is not claimed to provide an interaction experience comparable to commercial web or mobile applications.

Data and Database Management

Ticketing systems require consistent management of customer, movie, price, quantity, order, and transaction data. The prototype represents these elements using Python variables and collections. This structure is sufficient for demonstrating program logic but does not provide the persistence, concurrency, integrity constraints, and query optimization available in a database management system. Future development should separate movie, customer, order, product, and transaction records into persistent entities and should use indexed database queries when dataset size increases.

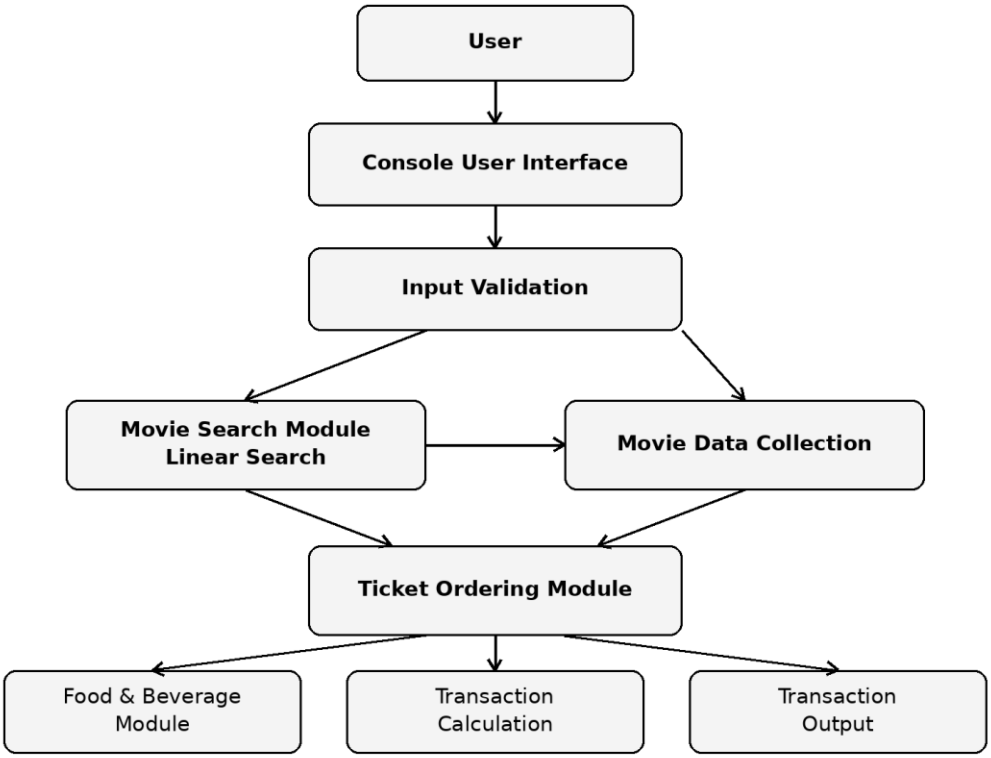
Security Considerations

Security becomes essential when a ticketing system stores personal information or processes digital payments. Relevant controls include input validation, authentication, authorization, session management, secure storage, error handling, and protection against injection or unauthorized access. These controls are consistent with the security verification areas described by the OWASP Application Security Verification Standard (ASVS) [8]. The current prototype does not include complete authentication or payment modules; security is therefore treated as a design requirement and research limitation rather than as an experimentally validated contribution.

Conceptual Framework and System Architecture

The revised architecture separates the user interface, input validation, search, data, ordering, and transaction functions. Linear Search is positioned as one supporting module rather than as the entire technological contribution. This structure also provides a clearer pathway for future migration from in-memory Python data to a database-backed web or mobile implementation.

Figure 1. Proposed System Architecture of the Revised Cinema Ticket Ordering Prototype



RESEARCH METHOD

This study adopts a prototype-oriented software development approach consisting of requirements analysis, system design, implementation, functional verification, and algorithmic analysis. The method is aligned with the two research questions: the first concerns the design of the ticket-ordering workflow, while the second concerns the technical suitability of Linear Search for movie lookup.

During requirements analysis, the functions demonstrated in the original prototype were identified: customer-name input, display of available movies, movie selection, ticket quantity, optional food and beverage ordering, and transaction calculation. The revised design adds an explicit movie-title search module so that the role of Linear Search is technically visible rather than only described theoretically.

During system design, the application process is separated into interface, validation, search, ordering, and calculation modules. Movie information is represented using a small Python data

collection. The ordering workflow can continue to use menu-based selection after a movie has been located, while title lookup uses sequential comparison.

Functional verification is limited to determining whether each program module accepts expected input, rejects invalid input where appropriate, and produces the expected transaction flow. The screenshots available in the submitted manuscript provide evidence that the basic console workflow runs for customer input, movie selection, ticket quantity, drink selection, and food selection. These screenshots are not interpreted as formal usability testing or comparative performance testing.

Algorithmic evaluation is conducted through theoretical complexity analysis. Linear Search is compared with Binary Search and hash-based lookup in terms of preprocessing requirements, expected search complexity, and structural overhead. No claim of superior runtime performance is made because the current study does not include a controlled empirical benchmark across different dataset sizes.

Revised Linear Search Module

To make the search mechanism explicit, the revised prototype introduces a title-based Linear Search function. A simplified implementation is presented below.

```
def linear_search_movie(movie_list, target_title):
    target_title = target_title.strip().lower()

    for index, movie in enumerate(movie_list):
        if movie["title"].lower() == target_title:
            return index

    return -1
```

The function normalizes the input title, checks each movie record sequentially, returns the index when a matching title is found, and returns -1 when the target is absent. With n movie records, the function may require up to n comparisons, consistent with O(n) worst-case complexity.

RESULTS AND DISCUSSION

Prototype Implementation

The developed prototype provides a sequential console interaction beginning with customer identification. The user is presented with available movies, selects a movie and ticket quantity, and can optionally select food and beverage items. Transaction values are calculated from the selected quantities and prices. These functions correspond to the main implementation evidence contained in the submitted manuscript.

Figure 2. Execution evidence from the original Python console prototype: movie list, ticket quantity, and food/beverage selection.

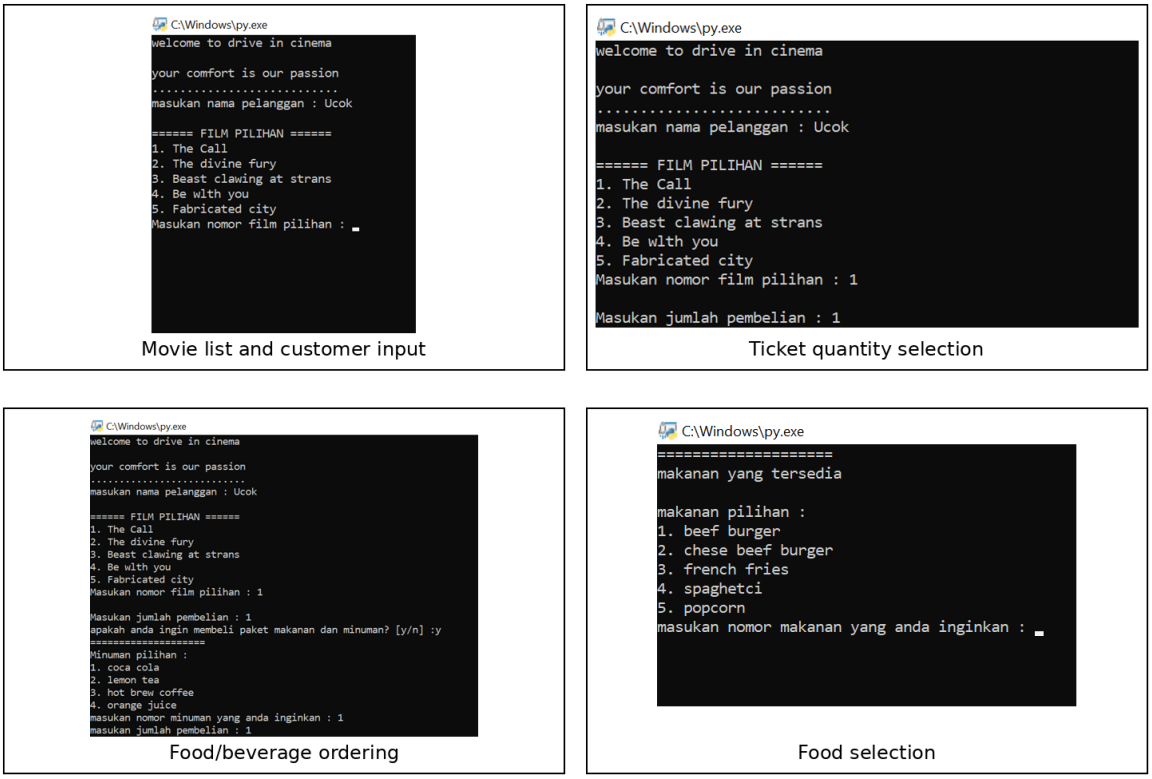


Figure 2 confirms that the program executes the principal ordering flow. However, the evidence should be interpreted carefully. The screenshots show a console-based Python program rather than an Android or web application. Consequently, the revised manuscript avoids claims that the system is a mobile application or that it can operate on any Android device. Such claims require a separate mobile implementation and deployment evidence.

Role and Suitability of Linear Search

Linear Search is appropriate in the revised design when the movie dataset is small, unsorted, and frequently updated without a need to maintain a separate index. The main benefit is low implementation overhead. If the target appears near the beginning of the list, only a small number of comparisons is required; if the target is at the end or absent, every record must be checked.

This behavior explains why Linear Search should not be described as universally efficient. Purnama [7] reports that Linear Search exhibits the slowest performance among the compared approaches on large-scale datasets, consistent with its O(n) complexity. Binary Search becomes more attractive when a sorted collection can be maintained, while hash-based lookup is advantageous when repeated key-based access justifies an additional indexing structure. The academic rationale for this study is therefore proportionality: a simple algorithm is acceptable because the prototype dataset is intentionally small, not because Linear Search is superior to alternatives.

User Interface, Data Management, and Security Implications

The console interface is sufficient to verify application logic but should not be interpreted as evidence of high usability. Formal user-interface evaluation requires participant-based or heuristic methods such as SUS, UEQ, or heuristic evaluation, as demonstrated by recent cinema-ticketing studies [1], [3], [4]. A future graphical, web, or mobile interface should therefore be evaluated empirically before statements about usability, responsiveness, or user satisfaction are made.

Similarly, the prototype currently relies on in-memory data and simple program variables. A production implementation would require a persistent database to maintain customers, movies, schedules, tickets, products, and transaction records. Database integration would also enable indexing, transaction control, and more scalable search mechanisms. If authentication or online payment is added, security controls based on established verification guidance such as OWASP ASVS should be incorporated [8].

Specific Technological Contribution

The technological contribution of this study is limited to what is demonstrated and theoretically supported. First, the study organizes customer input, movie lookup and selection, ticket quantity, optional food and beverage ordering, and transaction calculation into a coherent prototype architecture. Second, it makes the search mechanism explicit and justifies Linear Search according to dataset size and ordering requirements. Third, it identifies a clear migration path toward database-backed, graphical, and secure implementations. The study does not claim a new search algorithm, superior runtime performance, or proven usability.

CONCLUSION

This study revised and clarified the design of a Python-based cinema ticket ordering prototype that integrates customer input, movie selection, ticket quantity processing, optional food and beverage ordering, and transaction calculation. The revised theoretical framework connects digital ticketing systems, software development, search algorithms, user interface considerations, data management, and security requirements with the implemented workflow.

Linear Search is selected as a supporting movie lookup mechanism because the prototype uses a small and unsorted collection of movie records. Its O(n) average and worst-case complexity makes it easy to implement but less suitable as dataset size and search frequency increase. Binary Search and hash-based lookup provide more scalable alternatives under their respective structural requirements. Consequently, the study positions Linear Search as a context-appropriate implementation choice rather than as an algorithmic novelty or superior method.

The present evidence does not establish that the prototype is more effective than existing cinema ticketing applications because no controlled usability test or comparative runtime benchmark was conducted. Future research should implement persistent database storage, seat reservation, authentication, secure digital payments, electronic ticket generation, a web or mobile interface, formal usability testing, and empirical benchmarking of Linear Search, Binary Search, database indexing, and hash-based lookup across increasing dataset sizes.

REFERENCES

[1] M. L. Haryanti, F. Fraderic, S. Hartono, and J. Salim, "Cinema e-Ticket Application Design and Usability Evaluation Using SUS," Nuansa Informatika, vol. 19, no. 1, pp. 14–22, 2025, doi: 10.25134/ilkom.v19i1.296.

[2] T. E. Anggraini and E. Maiyana, "Penggunaan Android Studio dalam Pengembangan Aplikasi Pemesanan Tiket Bioskop," Journal of Innovation and Future Technology, vol. 7, no. 1, pp. 52–58, 2025, doi: 10.47080/iftech.v7i1.3805.

[3] Y. N. Choirunisa and M. Ryansyah, "Experience Evaluation of Online Cinema Ticket Applications with User Experience Questionnaire," JIKO (Jurnal Informatika dan Komputer), vol. 6, no. 3, pp. 203–208, 2023, doi: 10.33387/jiko.v6i3.7015.

[4] L. A. K. Ningrum, M. Y. Fathoni, and R. R. H. Setyodewi, "Usability Evaluation of the Cinemas Indonesia CGV Application Using Heuristic Evaluation and PIECES Framework," Sistemasi: Jurnal Sistem Informasi, vol. 12, no. 3, pp. 689–702, 2023, doi: 10.32520/stmsi.v12i3.2769.

[5] A. T. Ananda HAT, M. H. Susanto, M. E. Fata, and Supriyono, "Pengembangan Software Pemesanan Tiket Bioskop Berbasis Odoo ERP dengan Metode Scrum," Maliki Interdisciplinary Journal, vol. 2, no. 5, pp. 561–572, 2024.

[6] A. M. Sarhan, H. Ayman, M. Wagdi, B. Ali, A. Adel, and R. Osama, "Integrating Machine Learning and Sentiment Analysis in Movie Recommendation Systems," Journal of Electrical Systems and Information Technology, vol. 11, art. no. 53, 2024, doi: 10.1186/s43067-024-00177-7.

[7] N. Purnama, "Comparative Performance Study of Search Algorithms on Large-Scale Data Structures," JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer), vol. 11, no. 1, pp. 99–109, 2025, doi: 10.33480/jitk.v11i1.6592.

[8] OWASP Foundation, Application Security Verification Standard (ASVS), Version 5.0.0, 2025.