NARSIMHA REDDY
ENGINEERING COLLEGE
An Autonomous Institution| Affiliated to JNTUH | Approved by AICTE
Accredited by NBA & NAAC with 'A' Grade

DATA STRUCTURES PROJECT S

BRIDGING THEORY & PRACTICE

PREPARED BY

CAREER DEVELOPMENT CENTER

Data Structure Projects

Bucket 1 – Linear Structures (Arrays, Linked Lists, Stacks, Queues)

Covers fundamentals of storage, sequential access, and order management.

- EduTrack – Smart Student Record Manager
- BookChain – Digital Library Navigator
- ShopCart – Inventory & Billing System
- ExamTrack – Online Test Result Analyzer
- HealthMon – Fitness Data Analyzer
- ExprSolve – Expression Evaluator & Compiler Assistant
- TicketGo – Queue-Based Booking & Waiting System
- CodeVault – Version History Tracker
- MailSort – Email Organizer using DS

Bucket 2 – Non-Linear Structures (Trees, Heaps, Hashing, Tries)

Covers advanced storage, fast lookups, indexing, and predictive systems.

- WordNet – Hash Table Based Dictionary
- InfoSeek – Mini Search Engine
- DocIndex – File Indexing and Retrieval System
- CacheSim – Memory Replacement Policy Simulator
- TaskFlow – To-Do List Manager with Priority Scheduling
- GameZone – Leaderboard Ranking System
- AutoSuggest – Text Prediction Engine

Bucket 3 – Graph-Based Systems (Graphs & Networks)

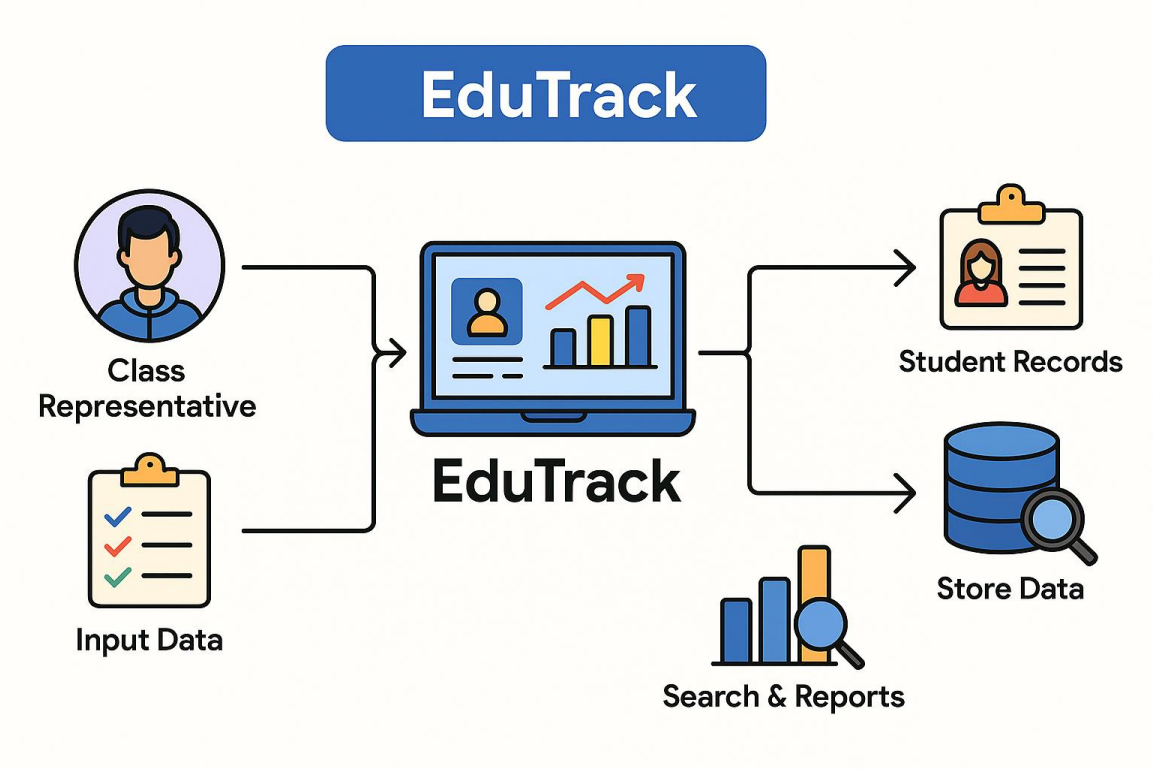
Covers traversal, shortest paths, recommendations, and network simulation.

- PathFinder – Smart Route Navigation
- ChatLink – Social Network Friend Recommendation
- CityGrid – Smart Traffic Management System
- MediCare+ – Priority-Based Patient Management

1. EduTrack – Smart Student Record Manager

Project Description

Imagine being the class representative of a busy engineering college. Suddenly, the Head of Department asks you to prepare a report with all student marks, attendance, and contact details within an hour. Normally, this would lead to chaos—papers scattered, missing details, and endless manual corrections. EduTrack solves this problem by acting like your smart digital assistant. It doesn't just hold data; it organizes, searches, and presents it faster than any manual method. From generating a merit list for the annual function to quickly finding the student who scored highest in Mathematics, EduTrack makes data feel alive. Students building this project will experience the thrill of designing their own academic ERP system, similar to what universities use, but simplified into a learning project.



Key Features

- Add, edit, and delete student records.
- Instant search by roll number, name, or grade.

- Generate reports like toppers' lists and grade analysis.
- Store data permanently with easy retrieval.

Technics to be Used

- Arrays/Linked Lists for storage.
- Searching and Sorting algorithms.
- File handling for persistence.

Demonstration Points

- Compare linear vs. binary search for finding a student.
- Show efficiency difference between array vs. linked list insertions.
- Use different sorting algorithms (Bubble, Quick, Merge) to sort students by marks and compare speed.

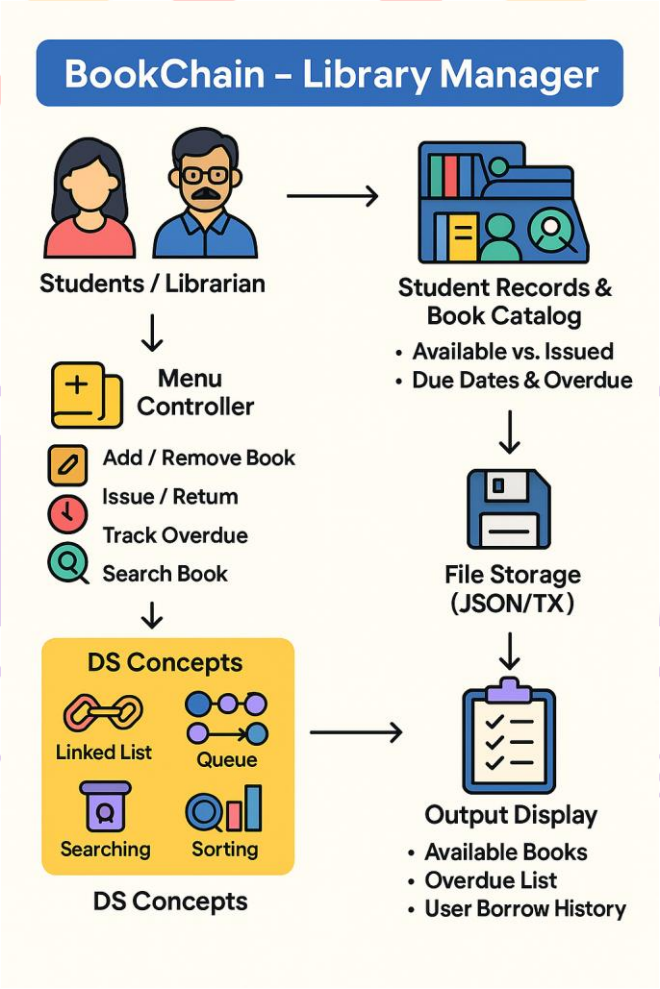
Expected Outcome

A mini ERP-style student management system that not only organizes data but also demonstrates how algorithm and DS choices affect efficiency in real applications.

2. BookChain – Digital Library Navigator

Project Description

Picture yourself in charge of a huge college library with thousands of books. Every day students walk in asking for novels, textbooks, and research material. Flipping through registers or searching piles of books takes hours. BookChain makes you the hero librarian! With this project, you can track every book in a digital catalog, know who borrowed it, and when it's due. Students can quickly check availability, and librarians can avoid losing books. It feels like running your own library portal, where data isn't just stored but can be searched and displayed instantly. BookChain makes managing books simple, smart, and fun while helping students understand how library systems really work in universities.



Key Features

- Add and remove books from catalog.
- Issue and return books with record of users.
- Track overdue books and due dates.
- Search by book ID, name, or author.
- Display available vs. issued books.

Technics to be Used

- Linked Lists for dynamic catalog.
- Stacks/Queues for issue and return logs.
- Searching algorithms.
- File handling for record storage.

Demonstration Points

- Compare linear vs. binary search for locating books.
- Show queue vs. stack use in handling book issue/return order.
- Measure efficiency of sorted vs. unsorted catalog searches.

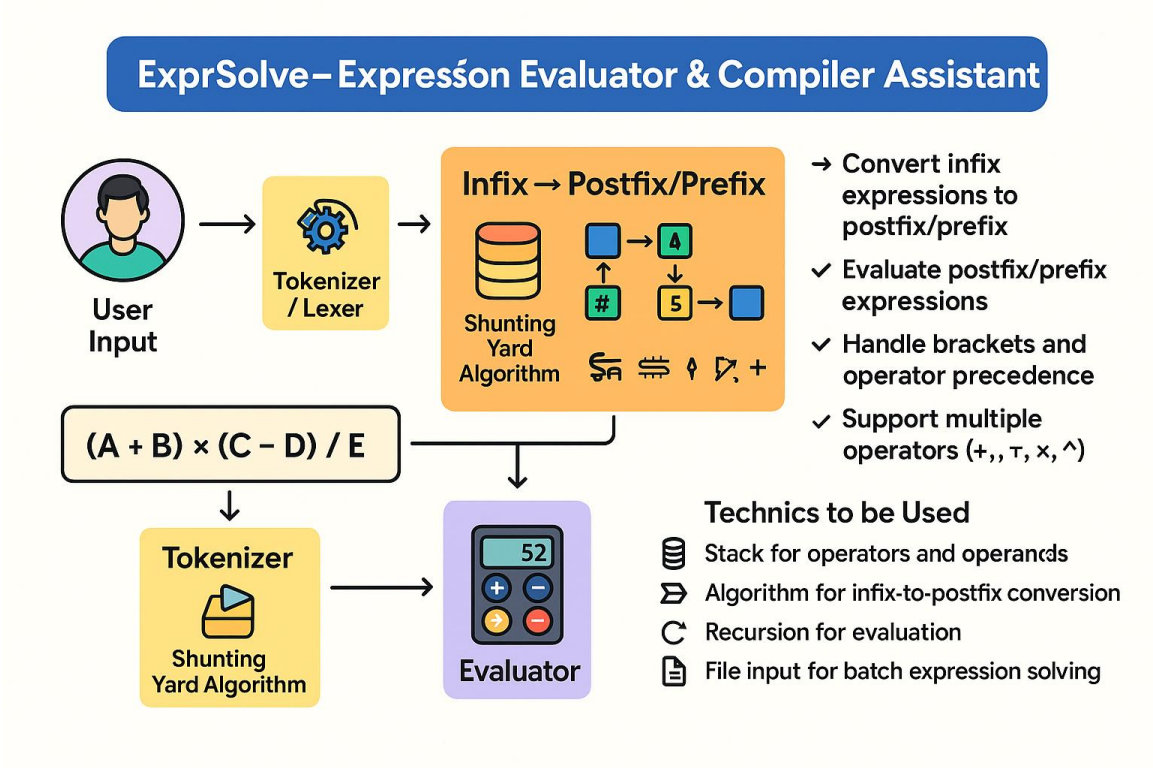
Expected Outcome

A digital catalog system that organizes library management while showing how DS techniques affect searching and issuing efficiency.

3. ExprSolve – Expression Evaluator & Compiler Assistant

Project Description

Imagine sitting in a coding interview where the interviewer hands you a complex mathematical expression like (A + B) * (C - D) / E and asks you to evaluate it quickly. Manually solving is easy for one expression, but what if you had 50 of them? ExprSolve makes life easier! This project mimics what compilers do behind the scenes when they convert your code into machine instructions. By building ExprSolve, you create a tool that can convert infix expressions (like what we usually write) into postfix or prefix forms, and then evaluate them systematically. It's like having your own little compiler calculator. You'll see how expressions are broken down and processed step by step. This project feels exciting because it turns abstract compiler concepts into something real that students can interact with.



Key Features

- Convert infix expressions to postfix/prefix.
- Evaluate postfix/prefix expressions.
- Handle brackets and operator precedence.
- Support multiple operators (+, -, *, /, ^).

Technics to be Used

- Stack for operators and operands.
- Algorithm for infix-to-postfix conversion.
- Recursion for evaluation.
- File input for batch expression solving.

Demonstration Points

- Compare evaluation with and without stack.
- Show difference in solving time for multiple expressions.
- Highlight operator precedence handling.

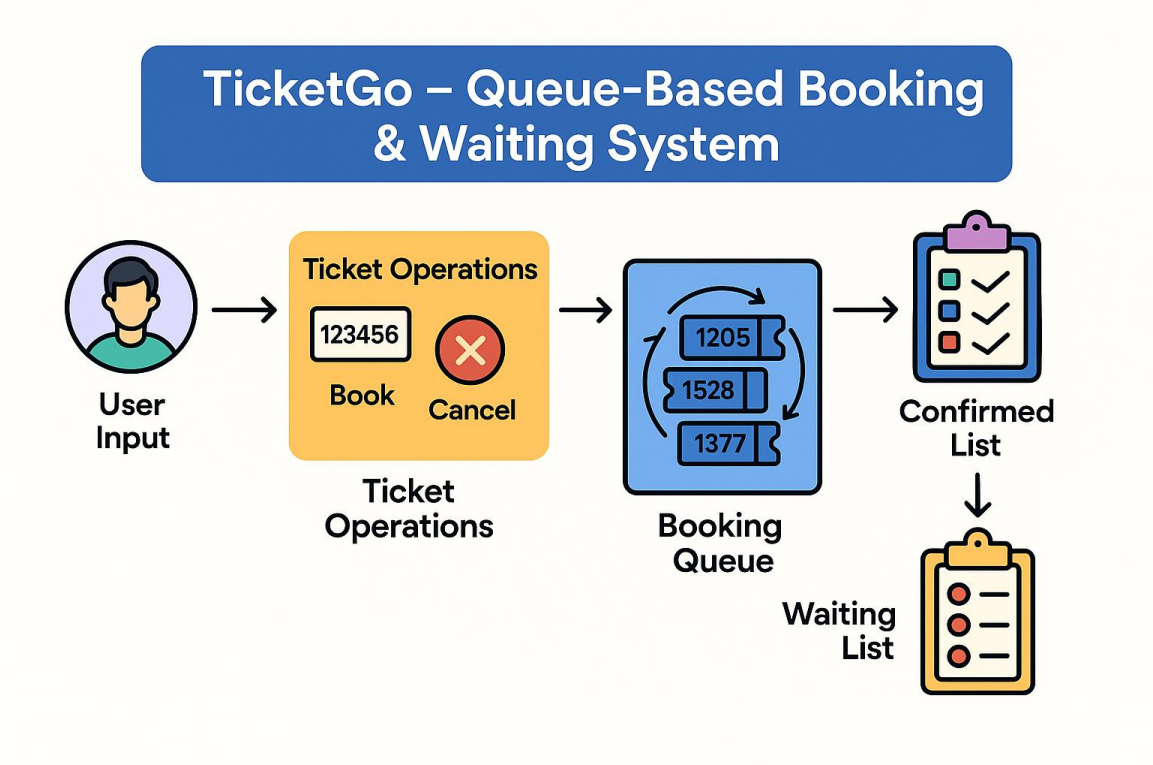
Expected Outcome

A tool that simulates compiler behavior for evaluating mathematical expressions, building strong foundations in logic and structured problem-solving.

4. TicketGo – Queue-Based Booking & Waiting System

Project Description

Picture this: a popular concert is coming to town, and everyone is rushing to book tickets. The system quickly overloads, and not everyone can get a seat. What happens next? A waiting list is created. TicketGo lets students simulate this exact real-world situation! With this project, users can book tickets, cancel bookings, and automatically move people from the waiting list into confirmed seats. It's like creating your own mini-IRCTC or concert booking portal. The excitement comes from watching how efficiently the system handles requests, ensuring fairness while avoiding chaos. Students will love seeing their program manage real-world booking scenarios, and it will make them think about how actual online platforms manage millions of requests every day.



Key Features

- Book tickets with unique IDs.
- Cancel tickets and reassign them automatically.

- Maintain waiting list for extra users.
- Display confirmed and waiting lists.

Technics to be Used

- Circular Queue for bookings.
- Linked Queue for waiting list.
- File storage for user records.
- Searching algorithm for ticket lookup.

Demonstration Points

- Show difference between array queue vs. linked queue.
- Compare cancellation handling with and without waiting list.
- Display efficiency of queue-based allocation.

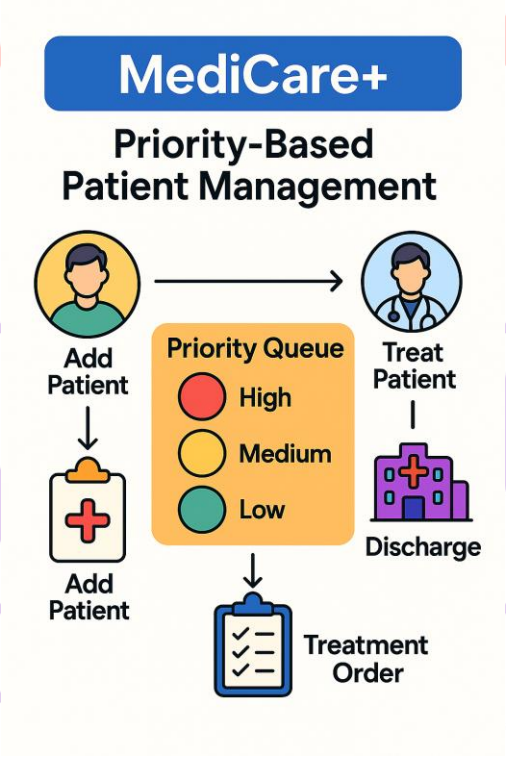
Expected Outcome

A booking system that mimics real-world ticket portals, teaching efficient handling of reservations, cancellations, and waiting queues.

5. MediCare+ – Priority-Based Patient Management

Project Description

Imagine a busy hospital emergency room with dozens of patients rushing in at once. Some patients just need a check-up, while others require immediate attention. Doctors must decide who gets treated first, and this is where MediCare+ comes in. The project simulates a hospital management system where patients are added with different priority levels depending on their condition. Critical patients are handled first, followed by moderate and then regular cases. This project is both exciting and realistic because it reflects how real hospitals manage patient queues in emergencies. Students will not only create a system that organizes patients fairly but also understand how priority-based scheduling can literally save lives in real-world scenarios.



Key Features

- Add patients with criticality levels.
- Automatically prioritize patients.
- Display list of patients in treatment order.
- Discharge patients once treated.

Technics to be Used

- Priority Queue implementation.
- Heap for efficient priority management.
- Searching to locate patients.
- File handling for patient history.

Demonstration Points

- Show difference between simple queue vs. priority queue.
- Compare patient handling order using heap vs. linked list.
- Measure speed of retrieval for critical cases.

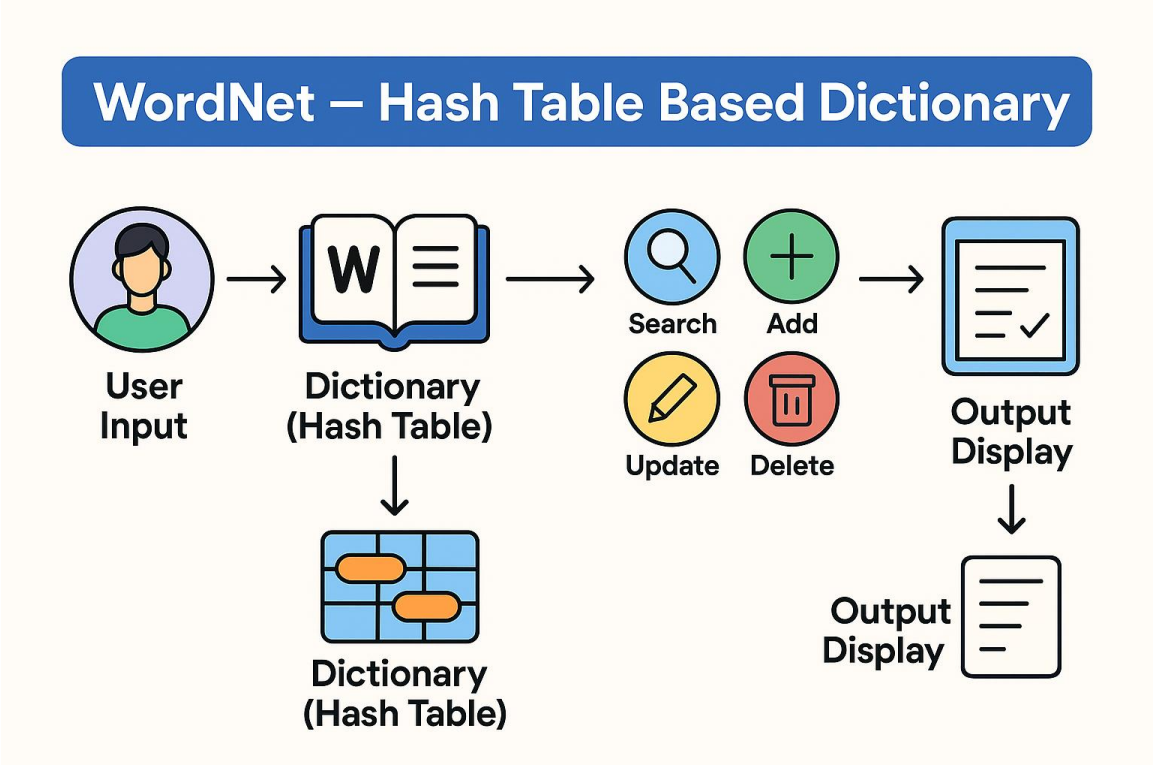
Expected Outcome

A simulation of hospital emergency management that shows how data structures can model real-life lifesaving decisions.

6. WordNet – Hash Table Based Dictionary

Project Description

Think of the times when you quickly searched for the meaning of a new English word on your phone dictionary. Wouldn't it be amazing to build one yourself? WordNet gives students the power to create their very own dictionary application where words and meanings are stored, searched, and updated instantly. Users can enter a word to see its definition, add new words, or delete outdated ones. This project is fun because it feels like creating a tiny version of Google Dictionary or Oxford Word app. Students enjoy testing it with their favorite vocabulary, and it becomes a practical tool for learning as well as programming.



Key Features

- Add new words with definitions.
- Search words instantly.
- Update or delete entries.
- Display full word list alphabetically.

Technics to be Used

- Hash Table for word storage.
- Collision handling using chaining.
- Sorting algorithms for display.
- File handling for permanent dictionary.

Demonstration Points

- Compare performance of linear search vs. hashing.
- Show efficiency of open addressing vs. chaining.
- Test speed with large vocabulary datasets.

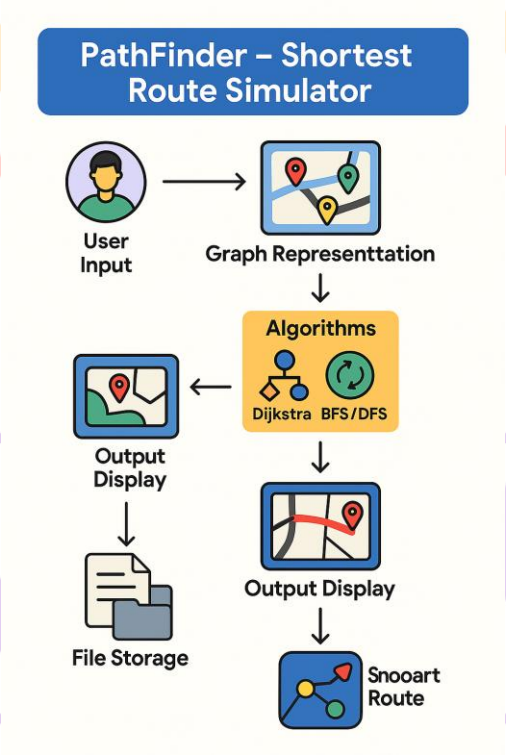
Expected Outcome

A working dictionary application where users can add, search, and manage words, demonstrating practical use of hashing techniques.

7. PathFinder – Smart Route Navigation

Project Description

Imagine planning a road trip with your friends to visit multiple cities. You pull out your map, but you're stuck deciding which route is shortest and cheapest. PathFinder solves this problem by simulating a navigation system similar to Google Maps. Users can enter cities and roads, and the system calculates the shortest route between them. It even suggests alternative paths! This project feels adventurous because it turns a boring graph problem into a real-life travel planner. Students can test it with real or fictional maps and see how the system chooses the fastest routes. It's like building your own travel assistant!



Key Features

- Add cities and routes.
- Find shortest path between two locations.
- Display multiple alternative routes.
- Visualize map-like structures.

Technics to be Used

- Graph representation with adjacency lists.
- Dijkstra's algorithm for shortest path.
- BFS/DFS for traversal.
- File handling for map storage.

Demonstration Points

- Compare BFS vs. Dijkstra results.
- Measure efficiency for small vs. large maps.
- Show effect of weighted vs. unweighted graphs.

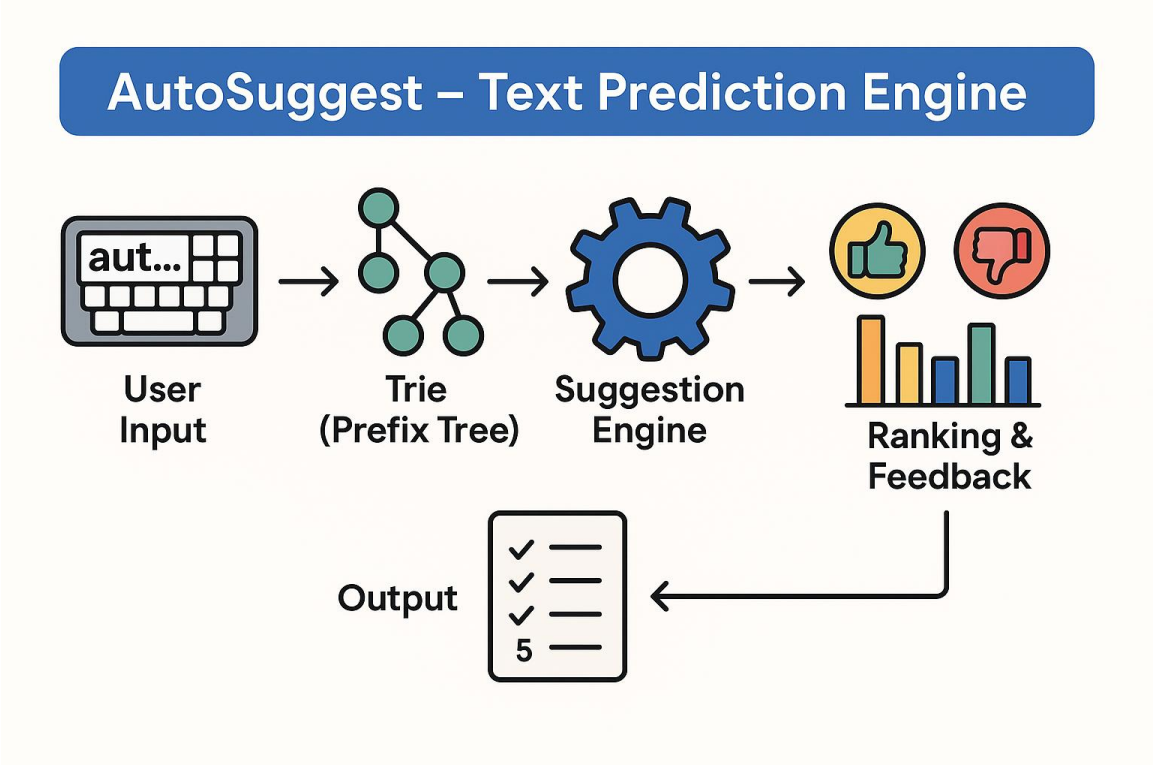
Expected Outcome

A navigation-style project that brings graphs to life by calculating shortest travel routes, mimicking real-world GPS applications.

8. AutoSuggest – Text Prediction Engine

Project Description

Have you noticed how your smartphone keyboard guesses what you're about to type next? Or how Google Search shows you possible queries after just a few letters? AutoSuggest lets students build their own version of this exciting feature! The project allows users to input a few characters and instantly get suggestions for complete words. It feels magical when the program guesses correctly. Students can preload the system with a dictionary of words or add their own as they type. This project makes coding fun because it connects directly to something students use daily—predictive typing.



Key Features

- Suggest words based on typed prefixes.
- Add new words to the suggestion list.
- Display top 5 most relevant suggestions.
- Allow user to accept or reject suggestions.

Technics to be Used

- Trie (prefix tree) for storing words.
- Recursion for searching prefixes.
- Hash map for frequency tracking.

Demonstration Points

- Compare array-based search vs. Trie-based search.
- Measure efficiency of prefix queries with large word sets.
- Test suggestions based on frequency vs. alphabetical order.

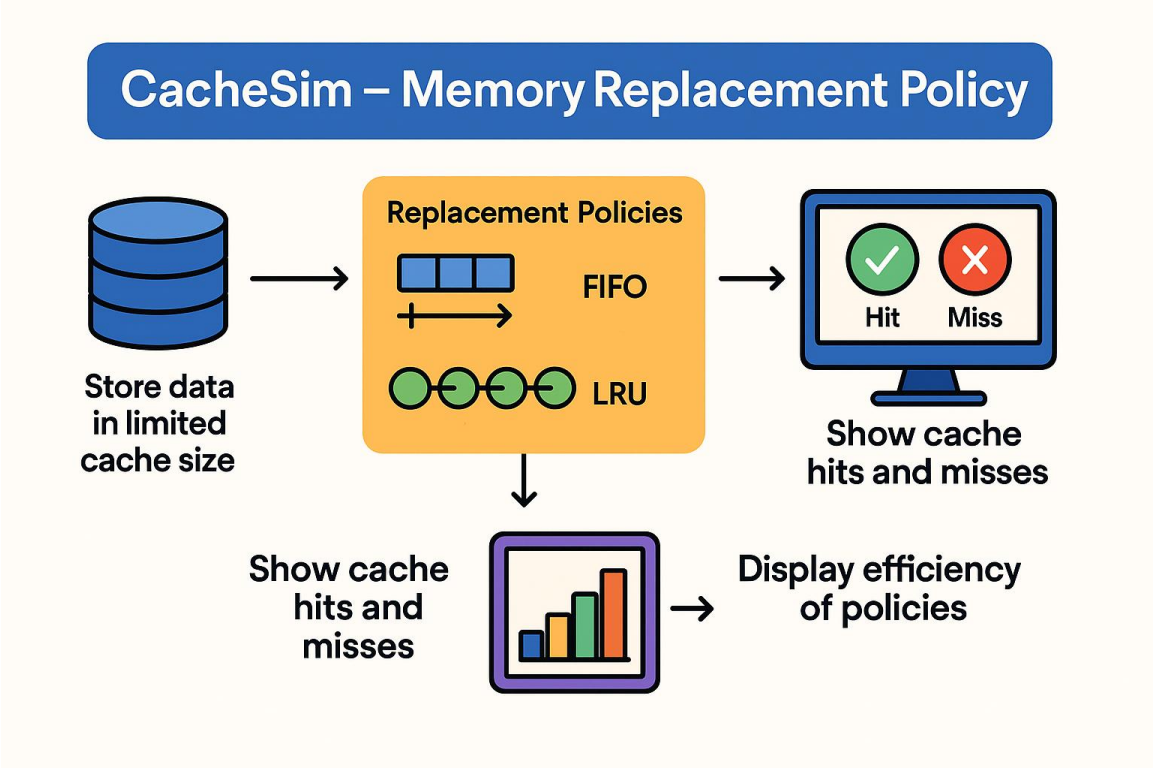
Expected Outcome

A predictive typing tool that mimics real-world autocomplete systems, showcasing how efficient data structures can power smart user interfaces.

9. CacheSim – Memory Replacement Policy Simulator

Project Description

Think about how your computer loads web pages or applications. Frequently used ones open instantly, while new ones may take longer. This is due to caching. CacheSim allows students to simulate cache memory behavior by implementing replacement policies like FIFO (First In First Out) and LRU (Least Recently Used). The project is engaging because it directly connects with what happens inside computers, browsers, and operating systems. By building this, students don't just code—they actually see how computers decide which data stays in fast memory and which gets pushed out.



Key Features

- Store data in limited cache size.
- Replace old data using FIFO or LRU policies.
- Show cache hits and misses.
- Display efficiency of policies on different datasets.

Technics to be Used

- Queue/Linked List for FIFO replacement.
- Doubly Linked List + HashMap for LRU.
- Simulation algorithms for cache management.

Demonstration Points

- Compare FIFO vs. LRU cache efficiency.
- Show cache hit/miss ratio for different input sizes.
- Visualize how data shifts inside cache.

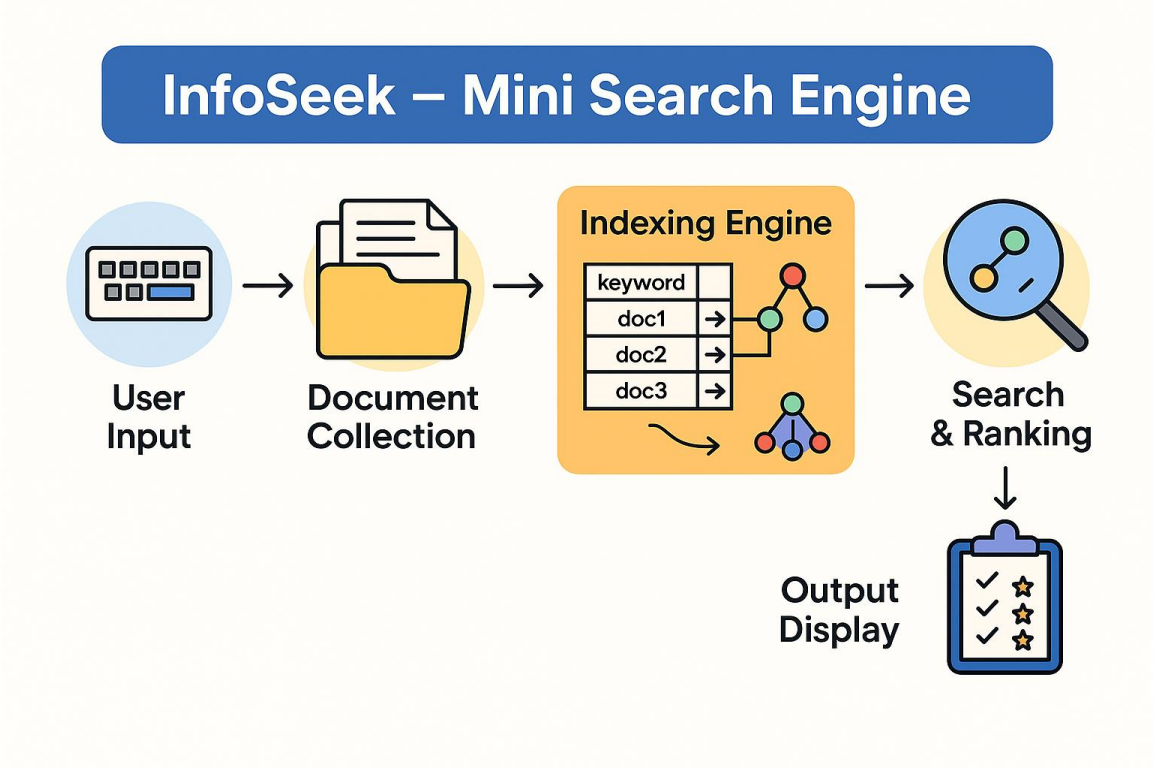
Expected Outcome

A simulator that helps students understand caching mechanisms, connecting theory from operating systems directly with practical coding experience.

10. InfoSeek – Mini Search Engine

Project Description

Imagine typing a keyword and instantly seeing documents that match. InfoSeek lets students build their own small-scale search engine. This project allows users to index words from a collection of documents and then search instantly. Students will enjoy uploading notes, articles, or even their favorite stories and then querying keywords to see results pop up. It's like building a mini Google! The excitement comes from creating something that turns scattered text into searchable information. Students also learn how search engines rank and retrieve data, which is one of the hottest topics in today's tech world.



Key Features

- Index documents by keywords.
- Search for words and display documents containing them.
- Rank results by frequency of keywords.
- Update index when new documents are added.

Technics to be Used

- Hash tables for indexing.
- Trees for storing word frequency.
- File parsing for reading documents.

Demonstration Points

- Compare linear search vs. indexed search.
- Show effect of ranking results by frequency.
- Test efficiency on small vs. large document collections.

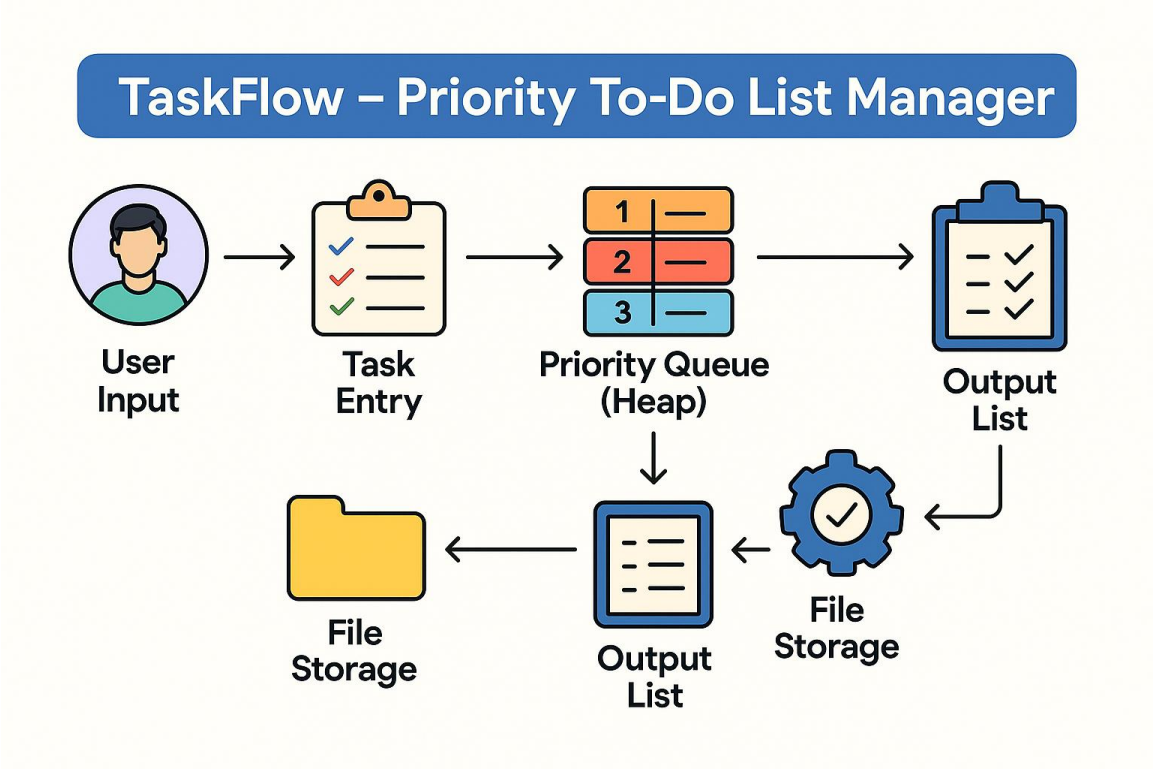
Expected Outcome

A mini search engine that turns raw text into structured searchable data, giving students hands-on experience with indexing and retrieval techniques.

11. TaskFlow – To-Do List Manager with Priority Scheduling

Project Description

Everyone has faced the chaos of managing daily tasks—assignments, project submissions, or even personal chores. You plan to do them in order, but emergencies come up, and suddenly some tasks are more urgent than others. TaskFlow helps organize life by acting as a digital to-do list that not only stores tasks but also arranges them based on their importance. Imagine a system where homework due tomorrow automatically shifts above cleaning your room. This project gives students a taste of how productivity apps like Microsoft To Do or Google Tasks work behind the scenes. Building TaskFlow is exciting because it is both practical and personal—students can use it themselves to manage study plans, projects, and daily routines. It blends technical learning with real-life utility, making it a project students will continue to use beyond the classroom.



Key Features

- Add, edit, and delete tasks.
- Assign priorities to tasks.
- Display tasks in order of urgency.
- Mark tasks as completed.

Technics to be Used

- Priority Queue for managing tasks.
- Heap for efficient task ordering.
- File handling for task persistence.

Demonstration Points

- Compare normal list vs. priority queue ordering.
- Show speed difference in retrieving highest priority task.
- Test task scheduling for large task sets.

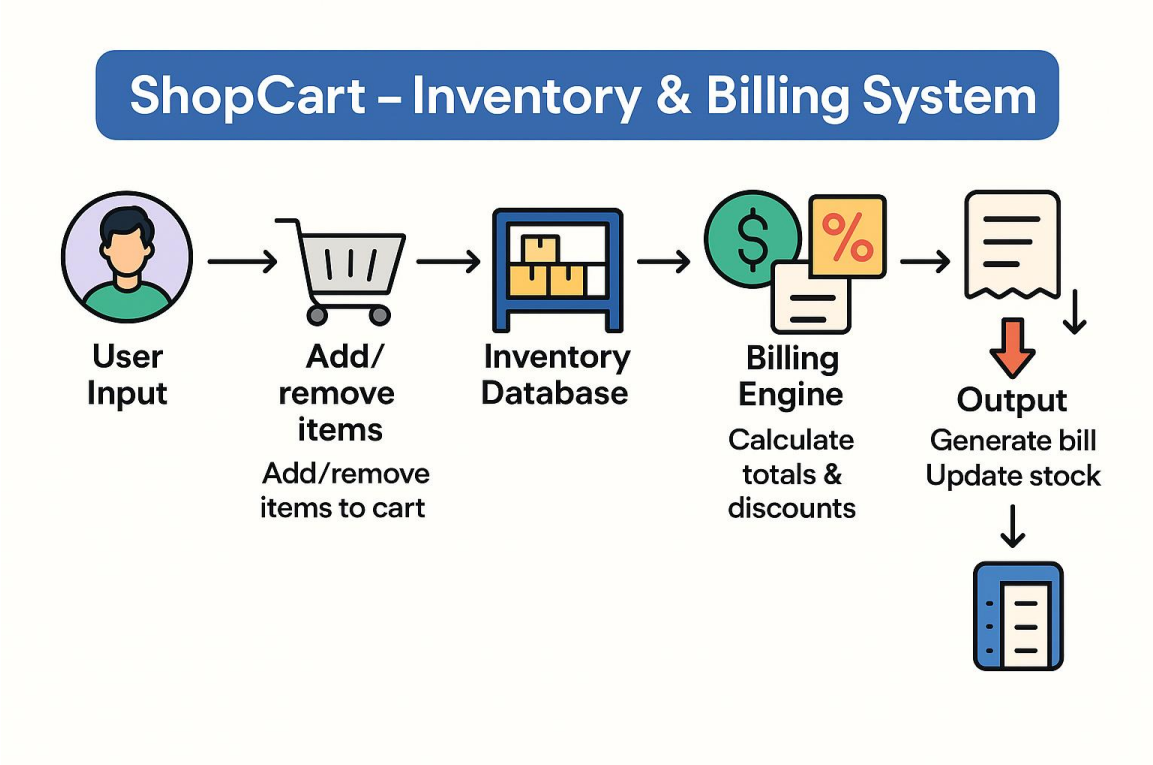
Expected Outcome

A personal to-do manager that demonstrates scheduling concepts while doubling as a practical daily productivity tool.

12. ShopCart – Inventory & Billing System

Project Description

Imagine you walk into a supermarket, and instead of waiting in long billing queues, you manage purchases digitally. ShopCart simulates this by tracking products, their availability, and generating bills instantly. Each time a customer adds an item, the system updates stock. On checkout, a neat bill is produced with totals and discounts. For students, this project feels like running their own retail store management software. They get to play shopkeeper, manage stock, and handle customer purchases all through code. The fun part is testing it with random products—chocolates, notebooks, gadgets—and seeing bills generate instantly. It reflects the systems used in malls and e-commerce platforms, making it both exciting and realistic.



Key Features

- Maintain inventory of products.
- Add/remove items to cart.

- Generate final bill with totals and discounts.
- Update stock after purchases.

Technics to be Used

- Arrays/Linked Lists for product storage.
- Hashing for product lookup.
- Sorting for bill display.
- File handling for persistent stock records.

Demonstration Points

- Compare product search using linear vs. hashing.
- Show effect of sorting by name vs. price.
- Measure stock update speed with large inventories.

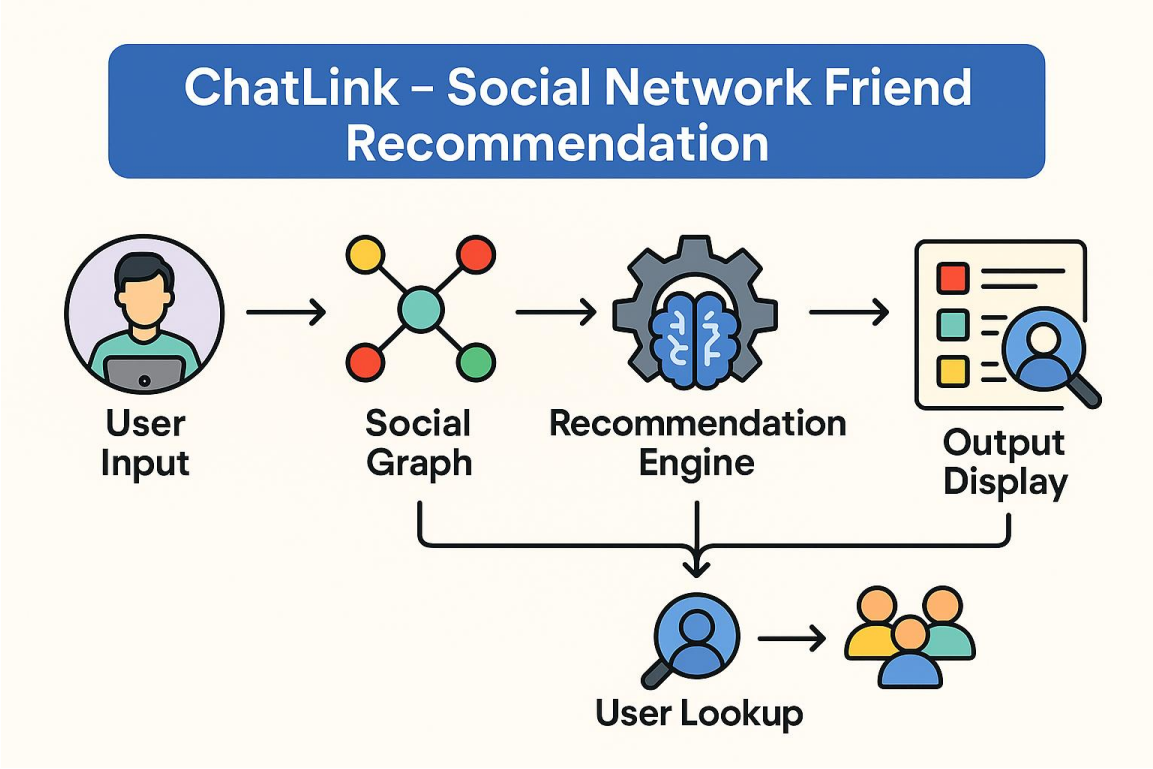
Expected Outcome

A retail-style inventory and billing simulator that gives hands-on experience with digital store management.

13. ChatLink – Social Network Friend Recommendation

Project Description

Think of how Facebook or LinkedIn suggests ‘People You May Know’. ChatLink lets students design a mini social network where users are nodes and friendships are connections. When a new user joins, the system can suggest possible friends based on mutual links. This project is fun because students can add their own names, create friendships, and watch the system recommend new ones. It feels like creating your own tiny version of a social media site. Students can explore how friendship graphs are formed and how real-world social networking apps leverage these connections to keep users engaged.



Key Features

- Add users and their connections.
- Suggest friends based on mutual links.

- Display friendship networks.
- Allow new users to join dynamically.

Technics to be Used

- Graph representation of users.
- BFS/DFS for exploring mutual connections.
- Hashing for quick user lookups.

Demonstration Points

- Compare friend suggestion using BFS vs. DFS.
- Show efficiency difference for small vs. large networks.
- Display how mutual links affect recommendations.

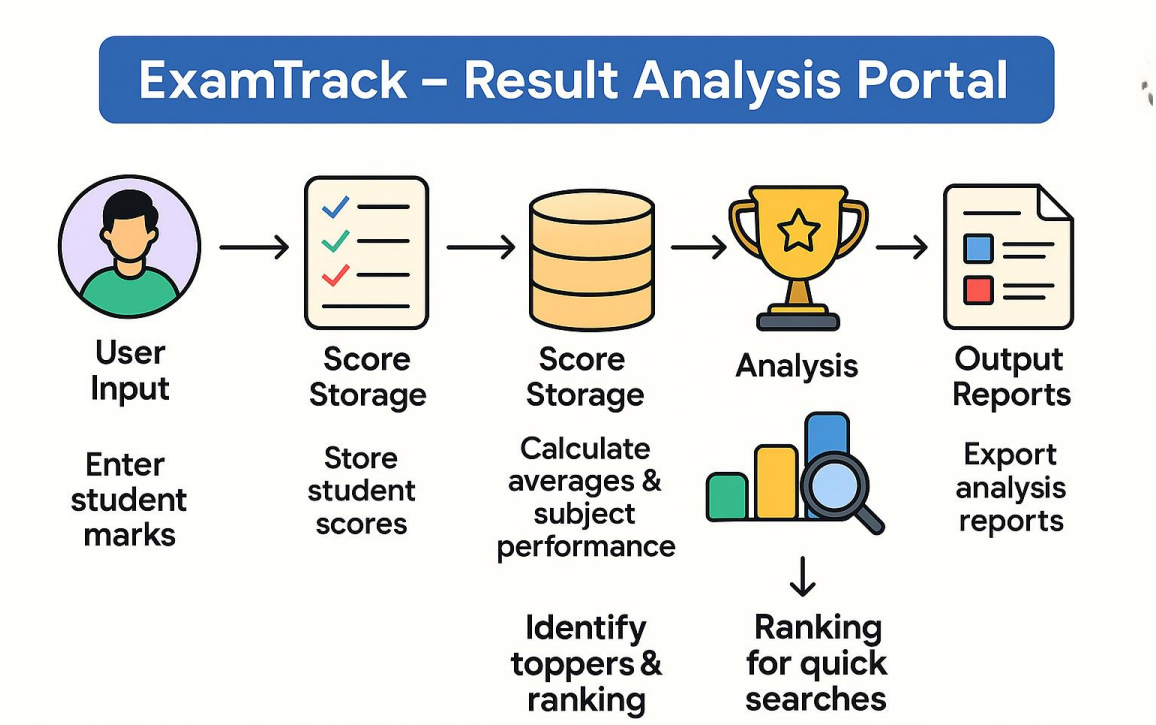
Expected Outcome

A miniature social networking system that demonstrates graph-based recommendations similar to real-world social apps.

14. ExamTrack – Online Test Result Analyzer

Project Description

After every exam, students eagerly wait for results—who topped, who improved, and who needs help. ExamTrack makes this process instant by analyzing scores and generating insightful reports. Teachers can input marks, and the system will produce class averages, top scorers, and subject-wise performance. Students will find it fun to see how quickly their marks transform into statistics. This project feels realistic because it mirrors what examination cells in colleges do with thousands of students' data. By coding this, students experience the excitement of building their own result analysis portal—something every school or college needs.



Key Features

- Input and store student marks.
- Generate average and top scorer reports.
- Display subject-wise analysis.
- Export reports for teachers.

Technics to be Used

- Arrays/Linked Lists for storing scores.
- Sorting algorithms for ranking.
- Hashing for quick student lookups.
- File handling for report storage.

Demonstration Points

- Compare sorting techniques for generating toppers' list.
- Show speed of average calculation for small vs. large classes.
- Test report generation efficiency.

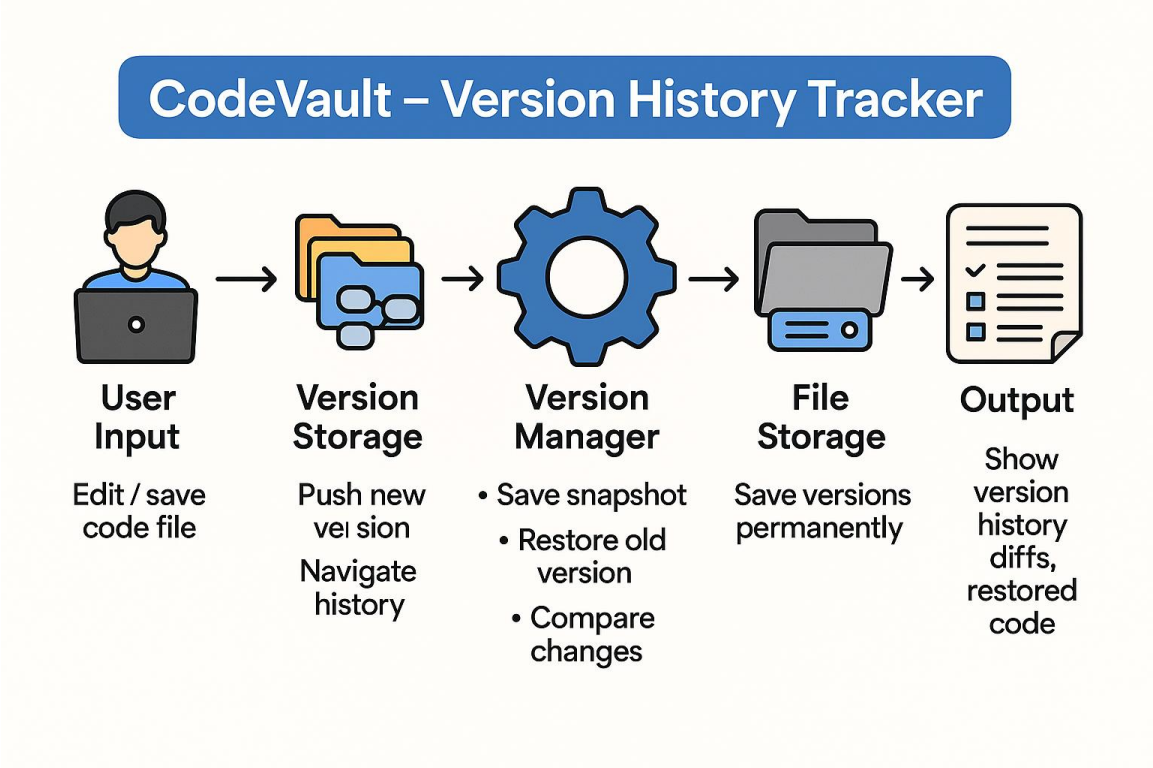
Expected Outcome

An automated result analyzer that brings examination report systems into a student-coded project.

15. CodeVault – Version History Tracker

Project Description

Imagine coding a project and accidentally deleting an important function. With CodeVault, you can restore it! This project simulates how version control systems like Git work by maintaining versions of code files. Every time a change is made, CodeVault stores a snapshot. Users can view history, restore old versions, or compare differences. Students love this project because it feels like they are building their own GitHub for managing assignments and projects. It's a practical tool that teaches not just coding but the discipline of saving and tracking progress.



Key Features

- Save multiple versions of files.
- Restore old versions.
- Compare differences between versions.
- Display version history.

Technics to be Used

- Stack for storing versions.
- Linked Lists for navigation.
- File handling for persistence.

Demonstration Points

- Compare storing versions in stack vs. linked list.
- Show retrieval time of older versions.
- Test with different file sizes.

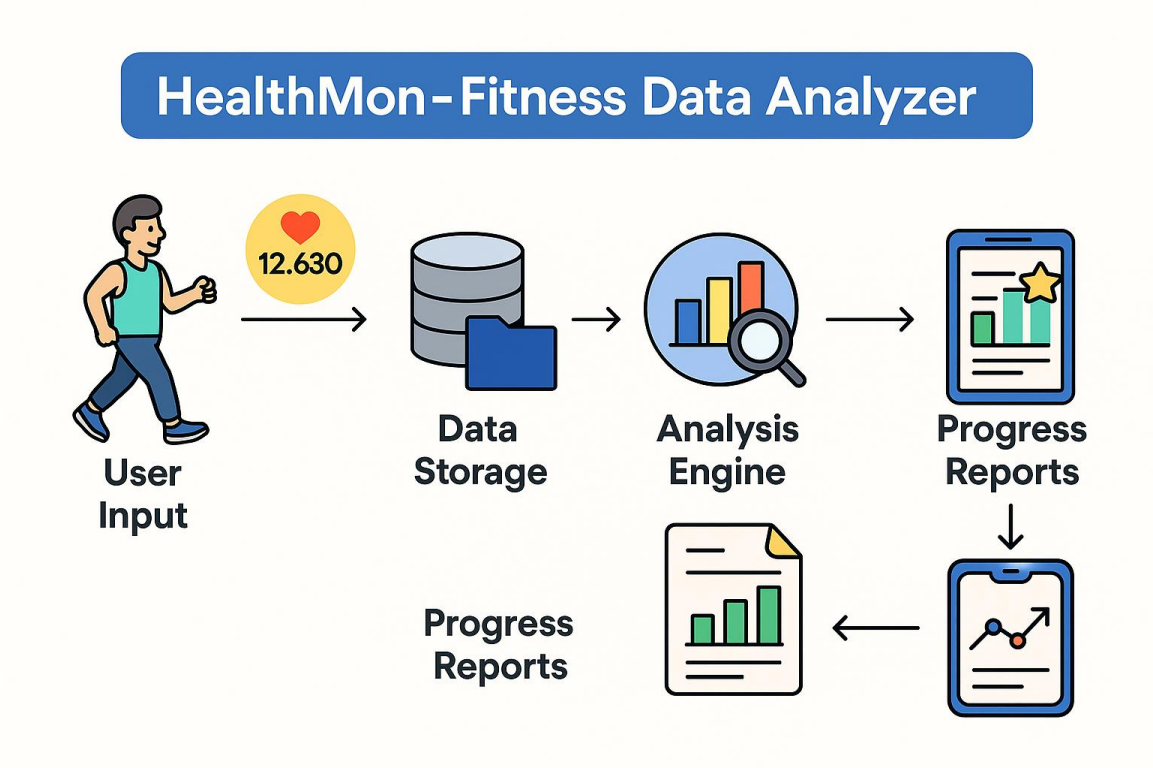
Expected Outcome

A version tracking system that introduces students to core concepts of version control in software engineering.

16. HealthMon – Fitness Data Analyzer

Project Description

Fitness apps are everywhere today, tracking steps, calories, and workouts. HealthMon lets students build a simpler version where users can log daily activities and analyze trends. Imagine a student tracking steps for a month and instantly seeing progress in graphs or summaries. This project is fun because it encourages health awareness while teaching coding. Students become both programmers and fitness coaches by creating their own mini health app.



Key Features

- Record daily steps, calories, and activities.
- Generate weekly/monthly progress reports.
- Highlight best and worst performance days.
- Provide goal-tracking features.

Technics to be Used

- Arrays for storing daily data.
- Sorting for trend analysis.
- Hashing for quick lookups.
- File handling for history.

Demonstration Points

- Compare efficiency of analyzing trends with sorted vs. unsorted data.
- Show performance differences for weekly vs. monthly reports.
- Visualize fitness improvements over time.

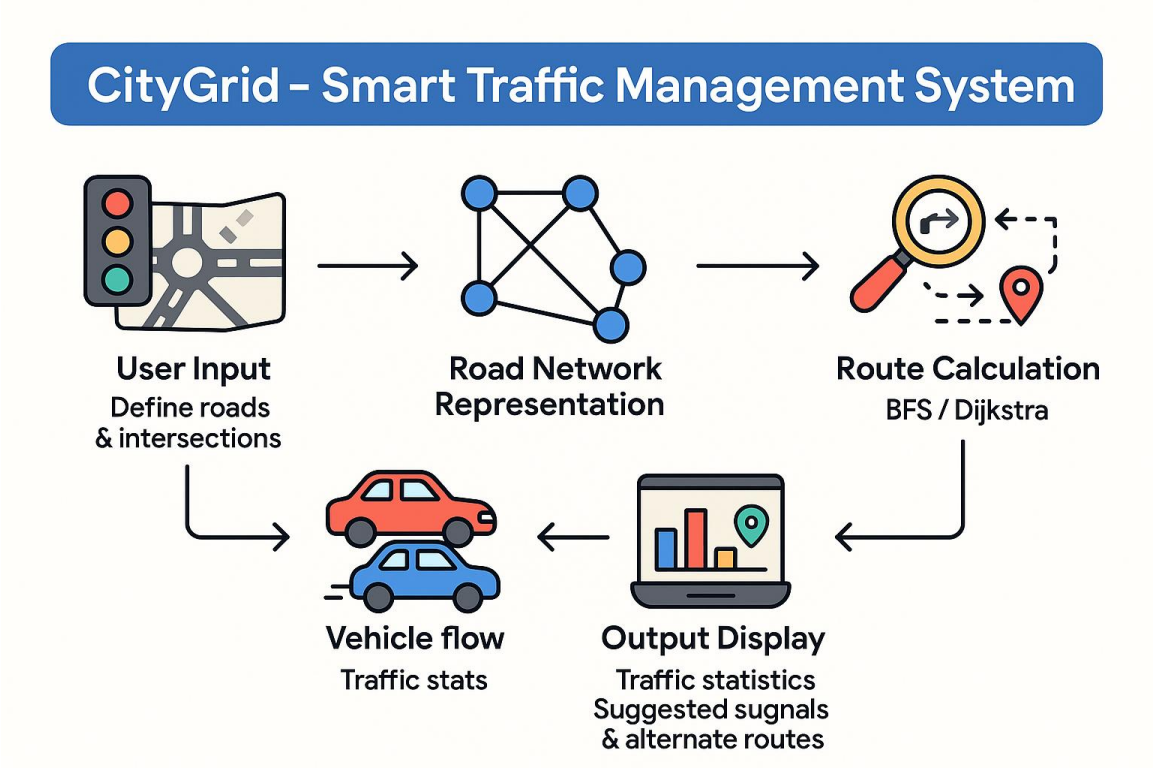
Expected Outcome

A personal health tracker project that blends coding practice with lifestyle improvement.

17. CityGrid – Smart Traffic Management System

Project Description

Imagine being a city planner trying to reduce traffic jams. CityGrid helps by simulating road networks and vehicle movements. Users can define intersections, roads, and vehicle flows, then watch the system suggest optimal traffic signals or alternative routes. It's like building your own traffic simulator! Students enjoy experimenting with their city layouts and testing how traffic responds. This project connects coding with real-world urban planning challenges.



Key Features

- Define city roads and intersections.
- Simulate vehicle flow.
- Suggest alternative routes for heavy traffic.
- Display traffic statistics.

Technics to be Used

- Graphs for road representation.
- BFS/Dijkstra for route calculation.
- Queues for vehicle movement.
- File handling for simulations.

Demonstration Points

- Compare shortest path results for different routes.
- Show efficiency of BFS vs. Dijkstra.
- Simulate peak vs. off-peak traffic.

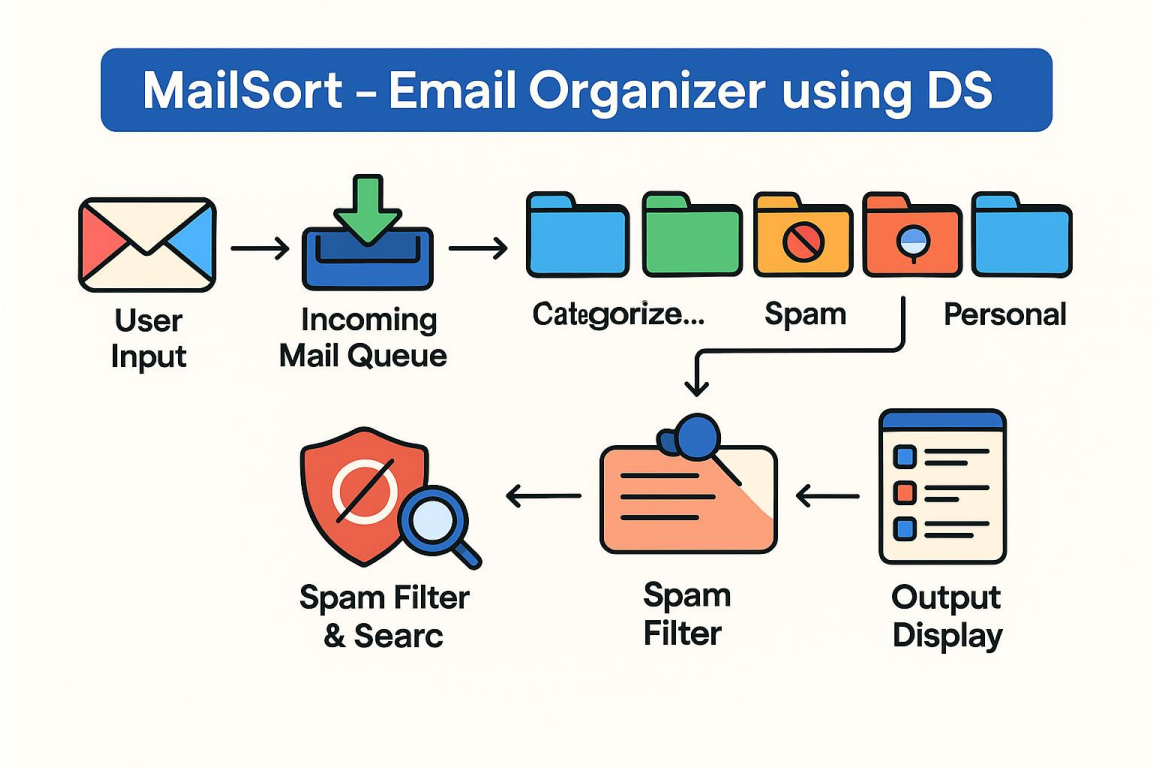
Expected Outcome

A traffic simulation project that brings together data structures with urban problem-solving.

18. MailSort – Email Organizer using DS

Project Description

Every day, inboxes are flooded with emails—important notices, spam, promotions, and personal chats. MailSort helps organize this chaos by categorizing emails automatically. Students can simulate an email client that sorts messages into folders, marks spam, and retrieves mail quickly. This project excites students because it feels like creating their own Gmail or Outlook, but in a simplified way. They can even test it with sample messages to see instant organization.



Key Features

- Store emails with sender, subject, and body.
- Categorize emails into folders.
- Mark and filter spam.
- Search emails instantly.

Technics to be Used

- Hashing for quick lookup.
- Linked Lists for folders.
- Queues for incoming mail.
- Sorting for search results.

Demonstration Points

- Compare folder organization with sorted vs. unsorted lists.
- Show retrieval time for search queries.
- Test spam filter performance.

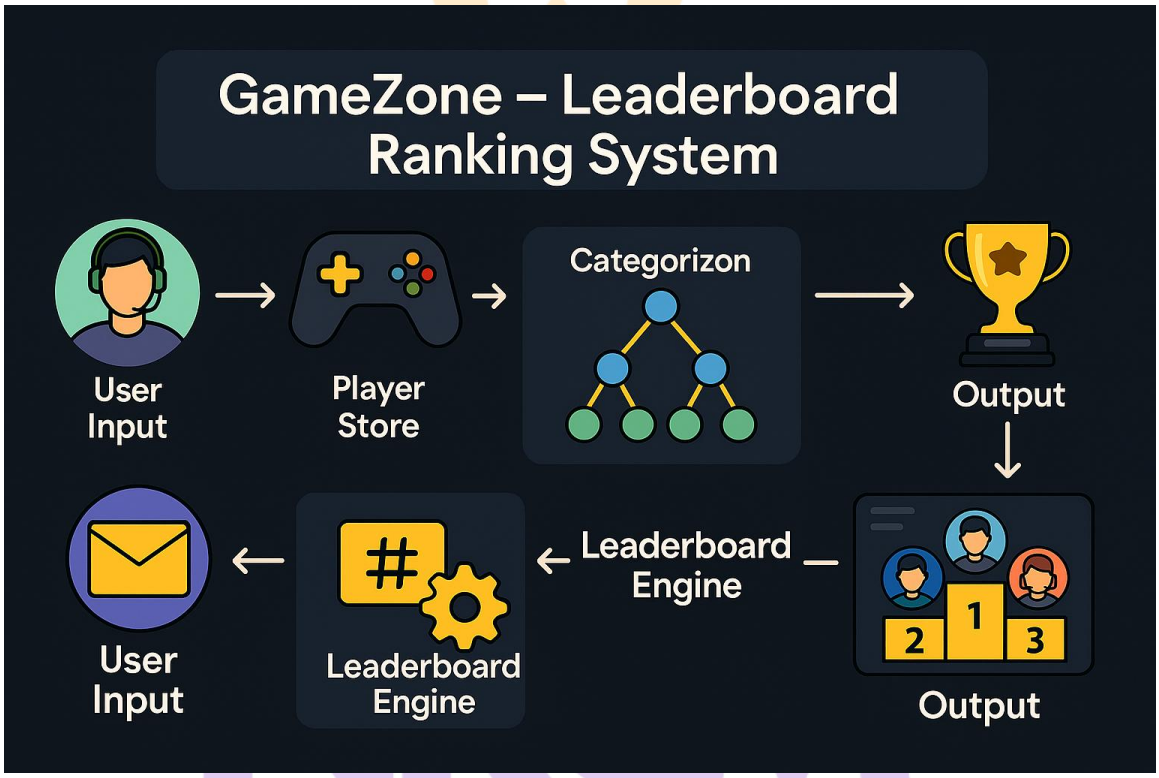
Expected Outcome

An email management system that demonstrates how DS concepts simplify communication tools.

19. GameZone – Leaderboard Ranking System

Project Description

Think of popular online games like PUBG or Call of Duty. They keep leaderboards showing top scorers worldwide. GameZone lets students design a ranking system where players' scores are updated, and leaderboards adjust automatically. It's thrilling to test with friends' names and scores to see who ranks highest. This project excites students because it feels like building a feature for real online games.



Key Features

- Add players with scores.
- Update scores dynamically.
- Display leaderboard in sorted order.
- Highlight top players.

Technics to be Used

- Binary Search Trees for ranking.
- Sorting algorithms for leaderboard updates.
- Hashing for quick player lookups.

Demonstration Points

- Compare leaderboard updates using arrays vs. trees.
- Show efficiency of sorting large score lists.
- Measure time for frequent score updates.

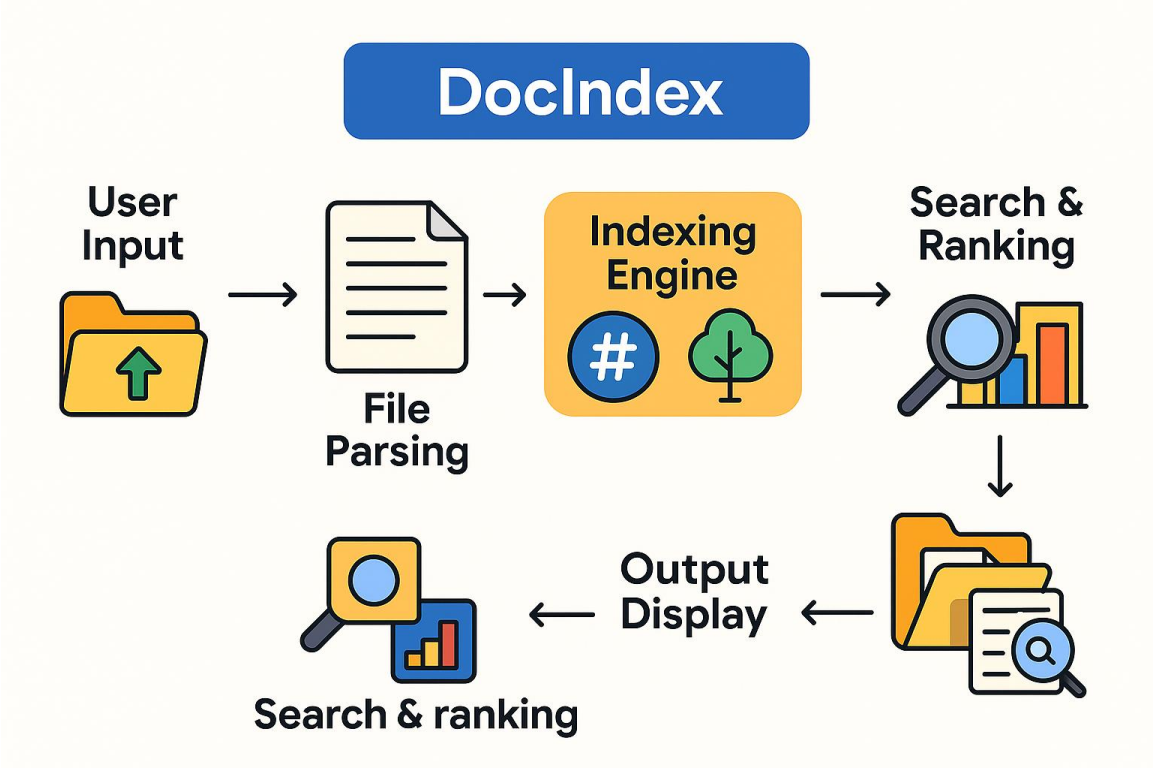
Expected Outcome

A leaderboard system that connects coding with gaming, making DS concepts fun and interactive.

20. DocIndex – File Indexing and Retrieval System

Project Description

Imagine having hundreds of study materials on your computer. Searching them manually wastes time. DocIndex solves this by indexing files and allowing instant retrieval based on keywords. Students can upload notes, and with a quick query, the system highlights matching files. It feels like building a mini document search engine. This project excites students because it mirrors what cloud storage services like Google Drive or OneDrive do behind the scenes.



Key Features

- Index documents by keywords.
- Search and display relevant files.
- Rank files based on keyword frequency.
- Update index dynamically.

Technics to be Used

- Hash Tables for indexing.
- Trees for storing keyword frequencies.
- File parsing for content extraction.

Demonstration Points

- Compare search time with indexing vs. no indexing.
- Show performance difference for small vs. large document sets.
- Rank results based on frequency.

Expected Outcome

A document indexing project that transforms raw files into searchable knowledge repositories.