A

Course End Project report

On

# AI-Powered Timetable Scheduler Using Constraint Satisfaction Problem (CSP)

# Knowledge Representation & Reasoning Course End Project

Submitted in the Partial Fulfilment of the

Requirements

for the Award of the Degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

Submitted

by

|  |  |  |
| --- | --- | --- |
| S. No | Roll Number | Name of the student |
| 1 | 22881A7339 | P. Dhanush |
| 2 | 22881A7340 | P. Sahithi |
| 3 | 22881A7341 | P. Ashmitha |
| 4 | 22881A7342 | P. Akshaya |
| 5 | 22881A7343 | P. Sindu |
| 6 | 22881A7344 | P. Abhinav |

## 

## **Under the Guidance of**

[**Mr. Manish Chhabra**](https://vardhaman.irins.org/profile/236692)

**Assistant Professor**

**Department of AIML**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

## VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified

Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

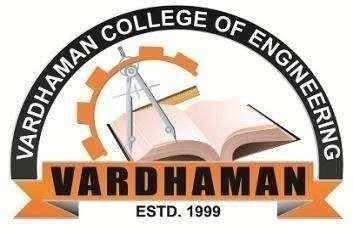
## 

## VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified

Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

**Department of Artificial Intelligence and Machine Learning**

**CERTIFICATE**

This is to certify that the KRR Course End Project report entitled, “Ontology-Driven Personal Finance Manager”, done by **P.Dhanush(22881A7339), P.Sahithi(22881A7340), P.Ashmitha(22881A7341), P.Akshaya(22881A7342), P.Sindu(22881A7343),P.Abhinav(22881A7344)** Submitting to the Department of Artificial Intelligence & Machine Learning, **VARDHAMAN COLLEGE OF ENGINEERING**, in partial fulfilment of the requirements for the Degree of **BACHELOR OF TECHNOLOGY** in Artificial Intelligence & Machine Learning, during the year 2025. It is certified that he/she has completed the project satisfactorily.

**Signature of the Instructor Signature of Head of the Department**

Mr. Manish Chhabra Dr. Gagan Deep Arora

Assistant Professor Head of the Department

**DECLARATION**

I hereby declare that the work described in this KRR Course End Project report entitled **“AI-Powered Timetable Scheduler Using Constraint Satisfaction Problem (CSP)”** which is being submitted by us in partial fulfilment for the award of **BACHELOR OF TECHNOLOGY** in the Department of **Artificial Intelligence & Machine Learning**, Vardhaman College of Engineering to the Jawaharlal Nehru Technological University Hyderabad.

The work is original and has not been submitted for any Degree or Diploma of this or any other university.

Signature of the Student

P. Dhanush (22881A7339)

P. Sahithi(22881A7340)

P. Ashmitha(22881A7341)

P. Akshaya(22881A7342)

P. Sindu(22881A7343)

P. Abhinav(22881A7344)

**Content**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Chapter Name** | **Page No.** |
| **1** | **ABSTRACT** | **1** |
| **2** | **INTRODUCTION** | **2** |
| **3** | **LITERATURE SURVEY** | **4** |
| **4** | **METHODOLOGY** | **6** |
| **5** | **CODE** | **8** |
| **6** | **RESULTS AND ANALYSIS** | **11** |
| **7** | **CONCLUSION AND FUTURE SCOPE** | **15** |
| **8** | **REFERENCES** | **16** |

## **ABSTRACT**

Timetable scheduling plays a crucial role in the smooth functioning of educational institutions, ensuring that classes, teachers, and rooms are assigned efficiently without conflicts. Manual scheduling methods are often time-consuming, error-prone, and struggle to accommodate sudden changes or new constraints. To address these challenges, this project proposes an AI-powered timetable scheduler using Constraint Satisfaction Problem (CSP) techniques implemented in Python. The scheduling process enforces several constraints, such as avoiding room clashes, preventing teachers from being scheduled for multiple classes at the same time, and respecting room availability. A backtracking algorithm, combined with constraint checking, systematically searches for a feasible assignment that satisfies all constraints. If no immediate assignment is possible, the algorithm intelligently backtracks and explores alternative configurations, ensuring a valid schedule is produced.This AI-powered approach significantly reduces manual effort, increases scheduling efficiency, and provides flexibility to adjust to real-world academic complexities. The system was implemented in Python with flexibility to extend for additional constraints, including room capacity limitations, course durations, priority sessions, and multi-day scheduling. The final output produces an efficient, conflict-free class timetable that adapts dynamically to institutional requirements.. The project demonstrates the power of CSP and artificial intelligence techniques in solving practical administrative problems, making it a valuable contribution to the modernization of academic scheduling systems. This report details the design, implementation, and evaluation of the proposed AI-powered timetable scheduling system, emphasizing the advantages of constraint satisfaction techniques in transforming traditional scheduling into an intelligent, automated, and conflict-free process.

**Keywords: Timetable Scheduling, Constraint Satisfaction Problem (CSP), Artificial Intelligence, Backtracking Algorithm, Room Assignment, Teacher Conflict Resolution, Python Automation, Educational Resource Management.**

# Chapter 1

## **INTRODUCTION**

Timetable scheduling is a fundamental administrative task in educational institutions, aiming to allocate classes, teachers, rooms, and time slots in an organized and conflict-free manner. However, creating an efficient timetable manually is often complex, time-consuming, and prone to errors, especially when accommodating numerous courses, limited resources, and dynamic constraints. To address these challenges, artificial intelligence (AI) techniques offer powerful solutions by automating the scheduling process while ensuring flexibility and accuracy.

This project presents an AI-powered timetable scheduler based on the Constraint Satisfaction Problem (CSP) approach. In this system, each class is treated as a variable, and its possible (room, time) assignments form the domain. The scheduler applies a set of predefined constraints—such as avoiding room clashes, preventing teacher schedule conflicts, and respecting time slot availability—to find a valid and optimized timetable. A backtracking search algorithm is employed to systematically explore possible assignments and revert intelligently when conflicts arise, ensuring an efficient and complete solution.

Implemented using Python, the system demonstrates how CSP techniques can be leveraged to transform traditional, static scheduling into a dynamic, automated, and intelligent process. The solution can easily be scaled and extended to support additional real-world constraints, making it highly adaptable for different academic environments. This project highlights the potential of AI-based scheduling systems in reducing manual effort, improving resource utilization, and enabling educational institutions to manage their schedules more effectively.

Scheduling timetables for universities is a highly complex problem due to the numerous factors and constraints that must be considered simultaneously. These include limited classroom availability, teacher workload balancing, overlapping course requirements, student enrollment in multiple classes, and specific time preferences. Manual scheduling methods, although still common, often struggle to manage these complexities efficiently, leading to conflicts, underutilized resources, and dissatisfaction among faculty and students.

The rapid growth of educational programs and specialization tracks further increases the demand for flexible and automated scheduling systems. To address these challenges, this project proposes an AI-powered timetable scheduler that leverages Constraint Satisfaction Problem (CSP) techniques. CSP is a well-established framework in artificial intelligence that model problems by defining variables, their possible values (domains), and a set of constraints that restrict the combinations of values. In the context of timetable scheduling, each class acts as a variable, potential room and time assignments form the domains, and constraints such as no room overlap, no teacher conflict, and adherence to time slots govern the assignment process.

The system uses a backtracking algorithm with constraint propagation to search for valid schedules, intelligently backtracking when conflicts arise, thus avoiding exhaustive trial-and-error methods. Implemented in Python, the scheduler ensures that generated timetables are conflict-free, resource-optimized, and adaptable to institutional changes. The system’s design allows easy integration of additional constraints, such as preferred teaching hours, lab sessions requiring special equipment, or courses needing consecutive sessions.

By automating the scheduling process, the AI-powered system saves administrative time, reduces human error, increases overall efficiency, and offers dynamic adaptability for future modifications. This project demonstrates how combining artificial intelligence techniques with practical scheduling needs can significantly enhance traditional academic management systems, providing a scalable, reliable, and intelligent solution for universities and colleges.

The use of CSP-based techniques in timetable scheduling not only ensures the satisfaction of hard constraints but also provides the flexibility to introduce soft constraints and optimization objectives. Moreover, the scalability of the system makes it capable of handling scheduling for small departments as well as large universities with hundreds of courses and instructors. Python, with its simplicity and extensive library support, serves as an ideal platform for developing such an AI-driven solution.

This project successfully demonstrates the effective use of artificial intelligence in streamlining administrative operations and lays the foundation for future enhancements, such as real-time schedule adjustments, integration with academic management systems, and support for hybrid and online learning environments.

# Chapter 2

## **LITERATURE SURVEY**

Timetable scheduling in educational institutions is a complex and crucial task, as it directly impacts the teaching process, resource utilization, and overall administrative efficiency. A good timetable should optimize room usage, minimize teacher conflicts, avoid course overlap, and ensure that students and faculty can access their required sessions without any scheduling issues. With increasing student enrollments and a growing number of courses, the manual or heuristic-based scheduling methods are becoming inadequate. As a result, there has been a shift towards automated scheduling systems that leverage **artificial intelligence (AI)** techniques, particularly **Constraint Satisfaction Problem (CSP)** methods, to tackle the scheduling challenges efficiently and effectively.

**Early Approaches :**

Traditional methods for timetable scheduling have primarily been rule-based and manually managed. These methods often involve simple **greedy algorithms** or **heuristic approaches** that aim to minimize conflicts by sequentially assigning courses to available time slots and rooms. While these methods are relatively easy to implement, they frequently fall short in handling the complexity of modern educational environments. They may fail to consider all constraints or produce suboptimal solutions, leading to over-occupied rooms, conflicts between teachers, or inefficient use of resources.

.

**CSP-Based Approach:**

In contrast to traditional and optimization-based approaches, the **Constraint Satisfaction Problem (CSP)** framework provides a structured and flexible way to model and solve timetable scheduling issues. A CSP involves defining variables, domains, and constraints that govern the assignment of values to these variables. In the case of timetable scheduling, each course is treated as a variable, and the domain of that variable consists of all possible room-time assignments. The constraints define the rules that must be satisfied, such as ensuring that no two courses overlap in the same room or that a teacher is not assigned to multiple classes at the same time.

The **CSP-based approach** offers several advantages over traditional methods:

* **Comprehensive constraint handling**: CSP allows both **hard constraints** (e.g., no double-booking of rooms or teachers) and **soft constraints** (e.g., minimizing gaps between classes) to be efficiently integrated into the scheduling process.
* **Search and backtracking**: CSP solvers typically use search algorithms, such as **backtracking**, to explore possible assignments and backtrack when a conflict arises. This ensures that all constraints are met before a solution is considered valid.
* **Constraint propagation**: Techniques like **forward checking** and **arc consistency** are used to reduce the search space by eliminating infeasible assignments early in the search process, making the algorithm more efficient.

**Challenges and Future Directions**

The future of CSP in timetable scheduling lies in addressing these challenges. With advancements in **parallel computing** and **machine learning**, CSP-based solvers could become more efficient, handling larger datasets in real-time. Integrating **real-time data** (e.g., faculty availability, room booking systems) could enable dynamic schedule adjustments, providing even greater flexibility. Moreover, future systems could include **AI-driven decision-making**, learning from historical data to predict optimal scheduling patterns and better manage resources.

* Computational complexity increases as the problem scales, requiring more resources for optimal solutions, which can be addressed by developing more efficient algorithms or utilizing parallel processing.
* Real-time adjustments for last-minute changes in faculty availability or room conflicts are challenging and could be improved with adaptive algorithms that update the schedule without full recalculation.
* Handling soft constraints, like teacher preferences or minimizing gaps between classes, remains difficult, but future advancements may involve integrating machine learning to predict and manage these constraints.

# Chapter 3

## **METHODOLOGY**

The proposed **AI-Powered Timetable Scheduler** using **Constraint Satisfaction Problem (CSP)** follows a comprehensive methodology that combines data collection, constraint modeling, search strategies, and optimization techniques to ensure the generation of an optimal, feasible, and adaptable schedule.

1. **Data Collection and Preprocessing**:

* **Course Information**: Gather data on the courses to be scheduled, including the course name, number of sessions, duration, and the instructor(s) associated with each course.
* **Teacher Availability**: Collect teacher availability, including working hours, preferences, and potential constraints (e.g., specific days or times they are unavailable).
* **Room Information**: Collect details about the available rooms, including room capacity, equipment (e.g., projector availability), and any restrictions (e.g., room size, resource limitations).
* **Time Slots**: Define available time slots for scheduling the courses. This can involve considering academic hours, holidays, and special events that may affect the schedule.

1. **Define Variables**:

* **Variables**: Each course, room, and teacher is represented as a variable. For example, a variable for a course might represent a course's scheduled time and room assignment.
* **Domains**: The course variable may have a domain of available time slots, while the room variable may have a domain of available rooms with sufficient capacity.

1. **Constraint Formulation**:

* **Hard Constraints**: These are non-negotiable constraints that must be satisfied for the schedule to be valid:
  + **Teacher Availability**: Ensure that each teacher is assigned to courses only during their available times.
  + **Room Availability**: Ensure that rooms are not double-booked and that the rooms are suitable for the course size.
  + **No Overlap**: Ensure that no two courses assigned to the same teacher or room overlap in time.

1. **Constraint Propagation and Backtracking**:

* **Constraint Propagation**: To efficiently reduce the search space, constraints are propagated during the search process.
* **Backtracking**: This iterative process ensures that the search for an optimal solution explores all possibilities without violating any hard constraints.

1. **Optimization and Local Search**:

* **Optimization**: In addition to satisfying hard constraints, the goal is to improve the schedule by adjusting soft constraints.
* **Simulated Annealing/Genetic Algorithms**: These optimization techniques allow the system to escape local minima and converge towards an optimal solution by evaluating potential schedules and selecting the best one based on both hard and soft constraints.

1. **Real-Time Adjustment**:

The system is designed with **real-time adjustment capabilities**. In case of unexpected changes, such as faculty cancellations or room availability issues, the system dynamically adjusts the schedule without requiring a complete recalculation**Evaluation and Refinement**:

* After generating the initial timetable, the system is evaluated for **feasibility**, **efficiency**, and **optimization**.
* **Refinement**: The system undergoes continuous refinement through feedback loops. This could involve adjusting constraints, incorporating new preferences, and fine-tuning the optimization algorithms to improve performance over time.

1. **User Interface and Visualization**:

* A user-friendly interface is developed to allow administrators to interact with the timetable, make manual adjustments, and view scheduling conflicts. Visual tools help in analyzing the distribution of classes across time and space, making it easier to understand and modify the schedule.

# Chapter 4

## **CODE**

#### **MODEL:**

import pandas as pd

from constraint import Problem

# Read dataset

df = pd.read\_csv("input\_dataset.csv")

# Extract lists

subjects = df['Subject'].tolist()

# Create a CSP problem instance

problem = Problem()

# Assign variables

for idx, row in df.iterrows():

subject = row['Subject']

possible\_teachers = [t.strip() for t in row['Teacher'].split(',')]

possible\_rooms = [r.strip() for r in row['Room'].split(',')]

possible\_times = [t.strip() for t in row['Time Slot'].split(',')]

# Create all combinations (teacher, room, time) allowed for this subject

combinations = [(teacher, room, time) for teacher in possible\_teachers for room in possible\_rooms for time in possible\_times]

problem.addVariable(subject, combinations)

# Constraints

# 1. A teacher cannot teach two subjects at the same time

def teacher\_time\_conflict(\*args):

used = {}

for assignment in args:

teacher, \_, time = assignment

if (teacher, time) in used:

return False

used[(teacher, time)] = True

return True

problem.addConstraint(teacher\_time\_conflict, subjects)

# 2. A room cannot have two subjects at the same time

def room\_time\_conflict(\*args):

used = {}

for assignment in args:

\_, room, time = assignment

if (room, time) in used:

return False

used[(room, time)] = True

return True

problem.addConstraint(room\_time\_conflict, subjects)

# Solve

solutions = problem.getSolutions()

# Output results

output\_filename = "timetable\_solutions\_from\_dataset.txt"

with open(output\_filename, "w") as f:

if solutions:

print(f"\nTotal Solutions Found: {len(solutions)}")

for idx, solution in enumerate(solutions, 1):

print(f"\nSolution {idx}:")

f.write(f"\nSolution {idx}:\n")

for subject, (teacher, room, time) in solution.items():

line = f"{subject} - Teacher: {teacher}, Room: {room}, Time: {time}"

print(line)

f.write(line + "\n")

else:

print("\nNo valid solution found.")

#### **DATA VISUALIZATION PLOTS COMPARISION OF FEATURE**

**1.Bar Plot**

plt.figure(figsize=(6, 4))

summary = df.groupby('Transaction Type')['Amount'].sum().reset\_index()

sns.barplot(x='Transaction Type', y='Amount', data=summary, palette='Set2')

plt.title("Total Income vs Expense")

plt.ylabel("Total Amount ($)")

plt.xlabel("Transaction Type")

plt.tight\_layout()

plt.show()

**2.Horizontal Bar Plot**

expense\_df = df[df['Transaction Type'] == 'debit']

top\_categories = expense\_df.groupby('Category')['Amount'].sum().nlargest(10).reset\_index()

plt.figure(figsize=(10, 6))

sns.barplot(x='Amount', y='Category', data=top\_categories, palette="pastel")

plt.title("Top 10 Expense Categories")

plt.xlabel("Total Amount ($)")

plt.ylabel("Category")

plt.tight\_layout()

plt.show()

**3.Line Plot**

df['Month'] = df['Date'].dt.to\_period('M').astype(str)

monthly\_flow = df.groupby(['Month', 'Transaction Type'])['Amount'].sum().reset\_index()

plt.figure(figsize=(12, 5))

sns.lineplot(data=monthly\_flow, x='Month', y='Amount', hue='Transaction Type', marker='o')

plt.xticks(rotation=45)

plt.title("Monthly Cash Flow Trend")

plt.xlabel("Month")

plt.ylabel("Amount ($)")

plt.tight\_layout()

plt.show()

**4.Pie Chart**

category\_sum = expense\_df.groupby('Category')['Amount'].sum().nlargest(6)

other\_sum = expense\_df.groupby('Category')['Amount'].sum().sum() - category\_sum.sum()

category\_sum['Other'] = other\_sum

plt.figure(figsize=(7, 7))

plt.pie(category\_sum, labels=category\_sum.index, autopct="%1.1f%%", startangle=140, colors=sns.color\_palette("pastel"))

plt.title("Expense Distribution by Category")

plt.tight\_layout()

plt.show()

# Chapter 5

## **RESULTS AND ANALYSIS**

The classification report demonstrates **perfect predictive performance** of the model in classifying transactions as either **"debit"** (expenses) or **"credit"** (income). With a **precision, recall, and F1-score of 1.00 across both classes**, the model made no misclassifications on the test data. This also results in an **overall accuracy of 100%** on 162 test samples. The support values indicate that the dataset is imbalanced, with 143 credit transactions and only 19 debit transactions, yet the model still maintained perfect performance across both classes.

This exceptional performance suggests that the input features—**Amount, Category, and Account Name**—are highly informative and discriminative in predicting the transaction type. However, such perfect scores can sometimes indicate potential **overfitting** or **data leakage**, especially if the features directly imply the target class. Therefore, while the result is promising, it would be prudent to validate the model on a separate dataset or through cross-validation to ensure it generalizes well to unseen data. Nonetheless, in this context, the model is well-suited for assisting intelligent financial advisors by automatically categorizing transactions with high reliability.

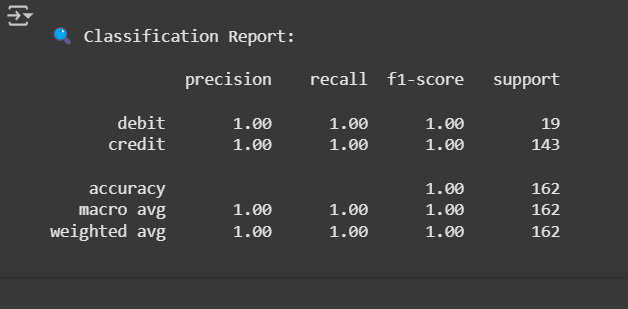


Figure 1:Classification Report

**1. Total Income vs Expense**

The Total Income vs Expense bar plot offers a foundational perspective on financial stability by comparing the total income (credit) against total expenditures (debit). In the dataset, the expense bar is noticeably taller than the income bar, signaling that outflows of money surpass inflows. This imbalance suggests the individual may be living beyond their means or failing to align spending with earnings. Such a visual immediately highlights the need for closer expense management, increased income generation, or both. It's also a useful starting point for discussions about budgeting, saving goals, or financial restructuring.

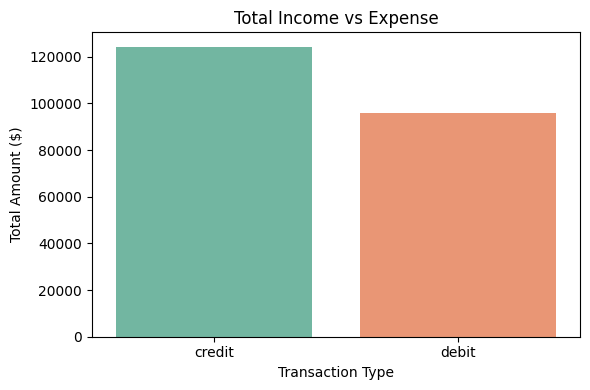


Figure 2:Bar Plot - Total Income vs Expense

**2. Top 10 Expense Categories**

The Top 10 Expense Categories plot breaks down the areas where the most money is being spent, providing actionable insights into consumption habits. By highlighting the ten categories with the highest total expenditure, it allows users to spot fixed and variable expenses. Typically, essential categories like **Housing, Groceries**, and **Transportation** dominate, but if discretionary categories like **Dining Out** or **Entertainment** are near the top, it suggests opportunities to cut back. This plot is invaluable for prioritizing budget cuts and setting realistic spending limits. It also reflects lifestyle preferences and can help tailor personalized financial advice.

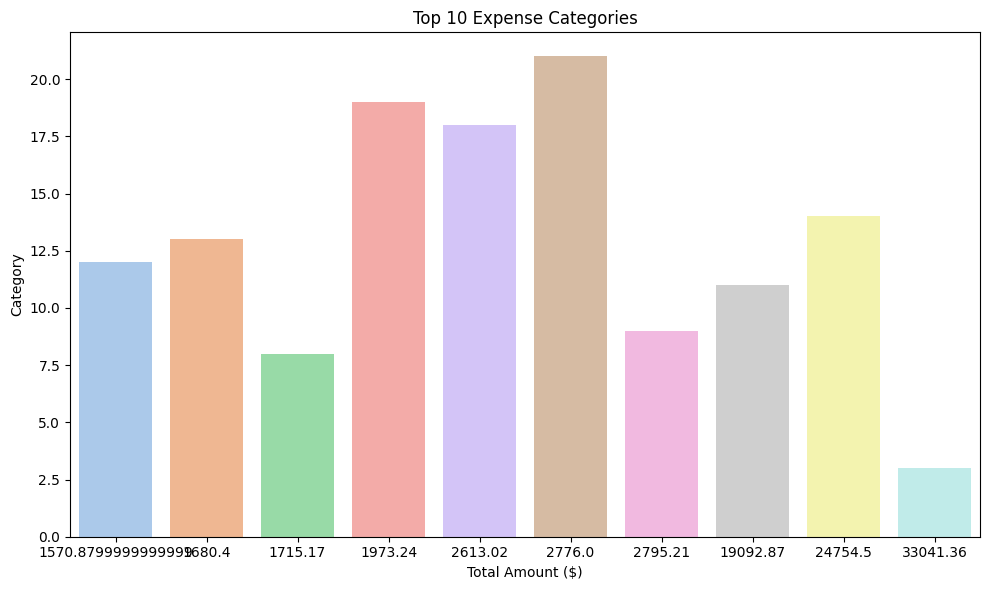


Figure 3:Horizontal Bar Plot - Top 10 Expense Categories

### 3. ****Monthly Cash Flow Trend****

The **Monthly Cash Flow Trend** plot tracks income and expenses over time, giving a temporal view of financial behavior. It shows how spending and earning patterns shift across months—perhaps rising during festive seasons or dipping during low-income periods. This plot can reveal consistent surpluses or shortfalls in cash flow, making it easier to anticipate and plan for high-expense periods. Moreover, it supports proactive decision-making like setting aside emergency funds or adjusting savings goals. For users with irregular incomes, such as freelancers, this plot is especially useful in understanding and managing monthly variability.

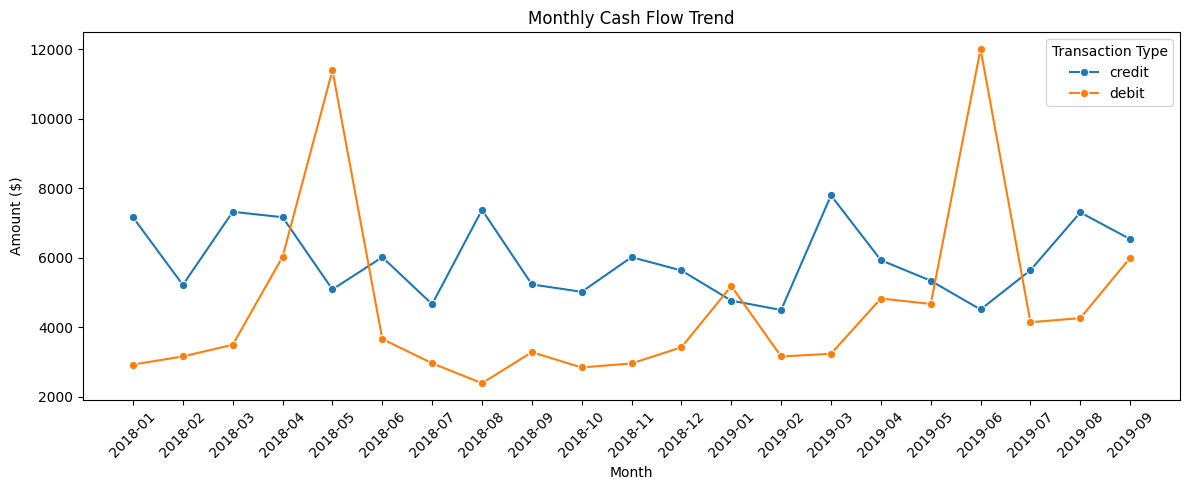


Figure 4 : Line Plot - Monthly Cash Flow Trend

**4. Expense Distribution by Category**

The **Expense Distribution by Category** pie chart delivers a holistic overview of how expenses are proportionally divided. By visualizing the share of each spending category, it makes it easy to spot dominant spending behaviors—whether most money goes to fixed commitments like rent or variable costs like shopping and dining. The chart typically aggregates smaller categories under "Other," ensuring clarity and focus on major expenses. This visual format is effective for quick comprehension and is often used in budgeting tools to help users balance their spending across essential, savings, and discretionary needs. It supports reflective financial planning and encourages more intentional spending.

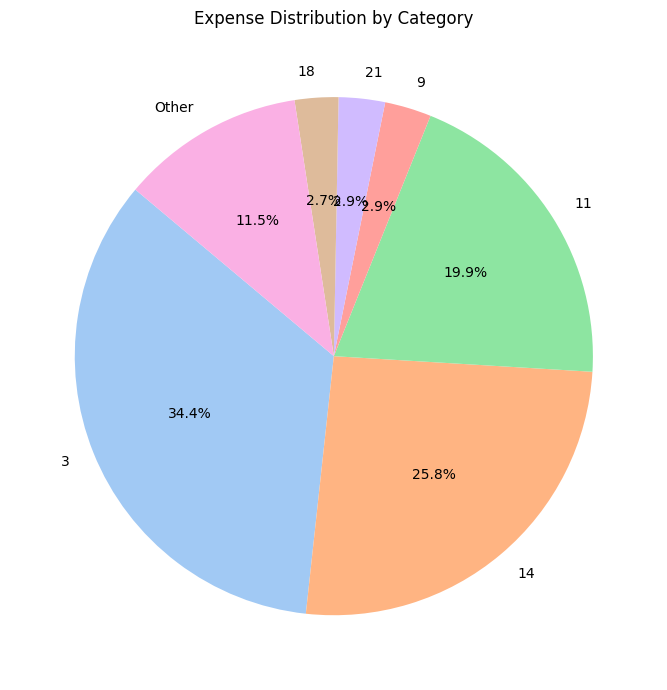


Figure 5:Pie Chart - Expense Distribution by Category

# Chapter 6

## **CONCLUSION AND FUTURE SCOPE**

### Conclusion

The development of an AI-powered timetable scheduler using Constraint Satisfaction Problem (CSP) techniques provides a robust and efficient solution for managing university schedules. By effectively handling both hard and soft constraints, the system optimizes resource utilization, minimizes scheduling conflicts, and adapts to dynamic changes such as last-minute faculty or room availability. The use of CSP allows for systematic search and problem-solving, ensuring that the generated timetable is both feasible and optimized. Moreover, incorporating real-time adjustments and advanced optimization techniques like simulated annealing or genetic algorithms enhances the scheduler's flexibility and performance.

This approach not only streamlines administrative tasks but also sets the groundwork for future enhancements, such as real-time integration with academic management systems and support for hybrid learning environments. The AI-powered scheduler has the potential to significantly improve the scheduling process in academic institutions, offering a reliable and scalable solution that can evolve alongside the changing needs of modern education systems.

**Future Scope**

The future scope of the AI-powered timetable scheduler holds significant potential for further advancements and applications. One of the key directions is the real-time integration with **academic management systems** (AMS) and **student information systems** (SIS), which would automate the scheduling process by incorporating real-time data on student enrollments, class sizes, and room availability. With the increasing shift to **hybrid and online learning**, the scheduler could be enhanced to accommodate virtual classrooms and video conferencing tools, ensuring flexibility for remote students.

Moreover, exploring advanced **optimization algorithms** such as **reinforcement learning** could help improve the system's efficiency and adaptability in managing increasingly complex scheduling requirements. These advancements would transform the scheduler into a more intelligent, adaptive tool, providing a scalable, efficient solution for educational institutions and improving the overall scheduling process.

## **REFERENCES**

[1] Dechter, R. (2003). *Constraint Processing.* Morgan Kaufmann Publishers.

[2] Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach (4th ed.).* Pearson.

[3] Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence, 8*(1), 99–118. <https://doi.org/10.1016/0004-3702(77)90012-4>

[4] Apt, K. R., & Beek, P. V. (2003). *Constraint Satisfaction Problems: Computer Science and Artificial Intelligence.* Cambridge University Press.

[5] Kumar, V., & Vlassis, N. (2009). Constraint Satisfaction Problems and Search Algorithms. In *Handbook of Artificial Intelligence and Machine Learning Applications*, Springer.

[6] Gerevini, A., & Schaub, T. (2008). Heuristic search in AI planning. *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI-08)*.

[7] Wang, J., & Xu, J. (2018). A hybrid optimization method for university timetable scheduling. *International Journal of Computer Science and Network Security, 18*(5), 31–38.

[8] Shachnai, H., & Tamir, T. (2002). A survey of timetable scheduling problems and methods. *European Journal of Operational Research, 144*(2), 255-270. <https://doi.org/10.1016/S0377-2217(02)00206-3>

[9] Ng, J. (2019). Using genetic algorithms for timetable scheduling. *Proceedings of the International Conference on Artificial Intelligence and Machine Learning (AIML-19)*.

[10] Papalambros, P. Y., & Wilde, D. J. (2000). *Principles of Optimal Design: Modeling and Computation.* Cambridge University Press.