University of Colombo School of Computing
SCS1303 - Introduction to Software Engineering
Tutorial 01

AI Software Development Mini Project

Overview

With the rapid rise of Artificial Intelligence (AI), creating software applications no longer requires advanced programming skills. In this tutorial task, you are going to explore how **AI tools can assist in developing software systems**, even if you are a complete beginner. This task is designed to help you get hands-on experience with AI-powered platforms that allow anyone—from novice to expert—to turn ideas into working applications.

Task Description

Students are required to select one case study from a given list of real-world software problems. The available options include: **Smart Parcel Locker Optimization, AI-Based Restaurant Table Reservation System, Smart Waste Collection Route Planner, AI Tutor Session Scheduler**, and **AI Personal Budget Optimizer**. Once a case study is chosen, students must logically break down the problem by identifying key inputs, desired outputs, system constraints, and any necessary assumptions. *These assumptions should be clearly stated in the final report.*

After logically analyzing the problem, students will use one or more AI tools to assist in the development process. AI can be used to refine the problem statement, generate or correct code, structure system logic, or explain technical concepts. The main goal is for students to approach the problem thoughtfully and then leverage AI to build or simulate a working solution. The developed system can be implemented using any suitable language or platform, such as Python, Java, or JavaScript. The prototype can be basic and may or may not include a user interface—the key requirement is that it demonstrates working logic and functionality.

You should include screenshots of the **development process**, especially where the AI assisted you, and also show the **working part of your final system** (even if it's basic).

Case Study 1: "Smart Parcel Locker Optimization"

Background:
A city has installed a series of smart parcel lockers across several neighborhoods. Each locker has limited storage capacity, and parcels of varying sizes arrive throughout the day. A delivery robot must determine the optimal way to distribute parcels among the available lockers to maximize space utilization and minimize delivery distance.

Problem Statement:

Design a system that:

1. Takes a list of parcels (with sizes and destinations),
2. Takes a list of lockers (with capacities and locations),
3. Calculates the optimal assignment of parcels to lockers based on:
    - Minimizing the total distance the delivery robot travels,
    - Ensuring locker capacity is not exceeded.

The system should output the assigned lockers for each parcel and the total travel distance.

Constraints:

- Each parcel can be assigned to only one locker.
- The sum of parcel sizes in a locker must not exceed its capacity.
- The robot starts from a central depot.

Inputs:

- A list of parcels: [{id: 1, size: 2, destination: (x, y)}, ...]
- A list of lockers: [{id: 'A', capacity: 10, location: (x, y)}, ...]
- Depot location: (x, y)

Expected Output:

- Assignment of parcels to lockers
- Total travel path and distance

Case Study 2: "AI-Based Restaurant Table Reservation System"

Background:

A popular restaurant chain wants to automate its table reservation system. Customers reserve tables online by specifying party size and preferred dining time. Tables vary by seating capacity, and the restaurant operates within specific hours. The goal is to assign reservations to tables in a way that maximizes seat utilization and minimizes unseated customers.

Problem Statement:

Design a system that:

- Takes a list of reservation requests (party size, time slot),
- Takes a list of tables (capacity, availability),
- Optimally assigns tables to reservations based on:
  - Minimizing unused seats,
  - Avoiding time slot overlaps,
- The system should output confirmed reservations and any rejections.

Constraints:

- Each table can serve one party per time slot.
- Reservations must be within open hours (e.g., 5pm–10pm).

Inputs:

- Reservation requests: [{id: 1, size: 4, time: "19:00"}, ...]
- Tables: [{id: 'T1', capacity: 4}, ...]
- Operating hours: start and end time.

Expected Output:

- Confirmed reservations with assigned tables.
- List of unfulfilled reservations.

- Seat utilization rate.

Case Study 3: "Smart Waste Collection Route Planner"

Background:
A smart city wants to use AI to plan waste collection routes. Each bin reports when it is full. A waste collection truck must collect waste from full bins, optimizing for the shortest route while ensuring the truck's capacity isn't exceeded.

Problem Statement:
Design a system that:

- Takes a list of full bins (location, waste amount),
- Takes the truck capacity and depot location,
- Plans a route to collect waste with:
    - Minimal travel distance,
    - No truck overload,
- Outputs the optimized path and total travel distance.

Constraints:

- Truck starts and ends at depot.
- Each bin must be visited only once.

Inputs:

- Full bins: [{id: 'B1', amount: 30, location: (x, y)}, ...]
- Truck capacity: 100
- Depot location: (x, y)

Expected Output:

- Route for waste collection.
- Total distance covered.

Case Study 4: "AI Tutor Session Scheduler"

Background:
An online learning platform wants to assign students to tutors for one-on-one sessions based on availability, subject expertise, and student preferences.

Problem Statement:
Design a scheduling system that:

- Takes a list of students (subject need, available times),
- Takes a list of tutors (expertise, time slots),
- Schedules sessions to:
    - Maximize successful pairings,
    - Respect availability constraints,

Constraints:

- One session per student per day.
- Tutor can't have overlapping sessions.

Inputs:

- Students: [{id: 1, subject: "Math", availability: ["10:00", "11:00"]}, ...]
- Tutors: [{id: 'T1', subjects: ["Math"], availability: ["10:00", "11:00"]}, ...]

Expected Output:

- Scheduled sessions (student-tutor pairs and time).
- List of unscheduled students.

Case Study 5: "AI Personal Budget Optimizer"

Background:
An individual wants to create a personal budgeting assistant that recommends spending allocations across various categories (food, rent, savings, etc.) based on income and goals.

Problem Statement:
Design a budgeting assistant that:

- Takes user's monthly income and fixed expenses,
- Takes financial goals (e.g., save 20%, spend <30% on rent),
- Suggests an optimal budget plan by:
    - Allocating remaining income across flexible categories,
    - Meeting specified constraints and goals.

Constraints:

- Must meet user-defined constraints.
- Income allocations must not exceed total income.

Inputs:

- Income: 3000
- Fixed expenses: [{category: "Rent", amount: 800}, ...]
- Goals: [{category: "Savings", target_percent: 20}, ...]

Expected Output:

- Detailed budget allocation.
- Warnings if goals can't be met.

Report Structure:

Please answer the following questions clearly in your report. Use headings and bullet points where appropriate. Your report should be **2 to 3 pages long** and include at least **two (2) screenshots** of your working system or app.

Write a short report (minimum 400 words) reflecting on your experience. You must answer the following:

- What software system did you choose to build, and why?
- Which AI tools did you use during development, and how exactly did they help?
- What parts of the system did you complete with AI assistance?
- What challenges did you face, and how did you solve them?
- Do you believe AI tools make programming easier or reduce the need for deep technical knowledge? Explain your thoughts.
- Would you consider using AI tools in your future studies or professional tasks? Why or why not?
- Include at least **two screenshots of your final software** and **evidence of AI involvement** (e.g., copied suggestions, chat logs, or code snippets from AI).

Submission:

- **Time allowed:** You must complete the task and submit your report within 1 week.
- **Submission method:** Upload your report to the **VLE (Virtual Learning Environment)** using the provided link.
- **File format:** PDF
- **File naming format:**
  YourName with Initials_IndexNumber_AIProject.pdf
  (e.g., Weerawardhana APK_122344_AIProject.pdf)

Important Notes

- Clearly mention any **assumptions** you used.
- Ensure your report shows **how AI helped you** logically and technically.
- Plagiarized code or unoriginal content will be penalized.

Viva Voce Examination

An individual **viva voce examination** will be held on **30th May 2025**. Each student will be allocated **5 minutes** to present and discuss their mini project. During the viva, you will be expected to:

- Briefly explain the selected case study and your understanding of the problem.
- Describe the logical steps you followed to design the solution.
- Demonstrate how you used AI tools during the development process.
- Answer questions related to your implementation and report.

Attendance is **mandatory**, and the viva will contribute to your overall evaluation. Please be well-prepared to explain both the technical and conceptual aspects of your work.