Advanced Topics in Software Engineering Assessment Brief

Assessment 3 – Individual Coursework
Assessment Weight: 40%
Due Date: 22 May 2025 by 23:00
Given Date: 23 April 2025 at 00:00
Submission Format: One PDF report (via Turnitin) and one publicly accessible video demonstration link (e.g., YouTube) clearly shared on the report cover page.

Overview

This is your final assessment (A3) for the *Advanced Topics in Software Engineering* module. You are required to apply a **plan-driven software development model** (e.g., Waterfall, Spiral, V-Model, DSDM) to design and partially implement a prototype seating algorithm system for a **venue-based business** (e.g., cinema, opera house, airplane, auditorium).

Your task includes:

- Producing UML-based **design documentation**
- Creating a **functional and non-functional requirements specification**
- Designing a seating plan and its **algorithm to use it effectively**
- **Prototyping** a system that reflects your design
- Recording and submitting a **video demonstration;** Write a **repor**t to document above

---

Submission Requirements: Video Demonstration and Report

Both the report and the video demonstration are **mandatory** and an integral part of this assessment. If the video is **not submitted** or is **inaccessible, no mark will be awarded**. The report alone does not adequately communicate your process and solution.

AI usage and Similarity Policy

This assessment, unlike the previous two, strictly monitors **similarity** and **AI usage** in both the report and video demonstration. A similarity score above 30% puts your work at risk to be considered **plagiarism** and your work might be treated as **academic misconduct**.

You may **use AI tools only for generating code**. Do not use AI to design the logic of the algorithm, write the report, or create the video or for any other purposes in this assessment. Submitting AI-generated content for these parts often results in high similarity, as many students unknowingly use the same prompts and produce near-identical work — even without collaborating.

This assessment includes exceptions and unpredictable elements that AI cannot address on its own. Completing it successfully requires **original thought, critical analysis, and creativity**. Ensure that your report is written clearly, in your **own words**, and accurately reflects your understanding and process. All work must be **entirely your own**.

Assessment Context

You are tasked with **designing a software system for a business that rely on the effective usage of a seating plan**, where optimizing this is crucial to both profitability and customer experience. You may choose one of the following business types for your design:

- Cinemas (Theatres)
- Opera Houses
- University Auditoria
- Airplanes

All these businesses rely fundamentally on **how effectively the venue's seating arrangement can be managed and optimized**. As a result, the type of business selected above significantly influence both the structure of the seating layout and the logic of the algorithm used to optimize it, as each context presents different spatial, behavioural, and commercial considerations.

In *cinemas*, the seating plans are formed of rows typically arranged in a straight or gently curved layout facing the screen, with most patrons preferring seats toward the middle and back for better viewing angles and comfort. Front rows are often less desirable and single scattered seats in rows may prevent couples or groups to sit together. An effective seating algorithm should account for these preferences while avoiding scattered single empty seats, which reduce occupancy efficiency.

In *airplanes*, solo travellers generally prefer window or aisle seats for comfort and accessibility, often avoiding middle seats. Assigning a solo traveller to a middle seat can split a potential pair, making it harder to accommodate couples or groups. This reduces seating efficiency and limits revenue optimization.

In *university auditorium* with a C-shaped seating arrangement, scattered single seats can create issues for couples and single individuals seeking to sit together, particularly during guest lectures, performances and/or presentations.

Finally, In *Opera Houses*, the seating arrangement often expands gradually and selecting side sections or the back rows usually offer a good view of the stage. However, finding companion seats can be difficult for those attending solo to an Opera house performance. Additionally, the traditional structure of opera houses, with boxes and balconies, can make it challenging for single people to find accessible seats.

Your algorithm should address one of these specific scenarios and focus on preventing the scattering of single seats in a way that would reduce the effectiveness of the seating arrangement. However, the accessibility seats, the VIP zones and other constraints demand a customized algorithm for each seating plan rather than overly generic working algorithms. The goal is to maximize the seating capacity by filling in available spaces more strategically, ensuring that each seat is utilized optimally.

Constraints and requirements (Mandatory)

- Design a **realistic, personalised seating layout** that is unique to your project.
- Include clearly marked **VIP zones (priority seats)** and **accessible/disability seating**, positioned by you in between rows either centrally or at the edges.
- Ensure **groups of 2–7 people** are seated together. Avoid splitting groups unless they exceed the size of a row.
- **Solo attendees** should not be placed between groups, except in special cases such as **VIP or accessible seating**.
- An **admin override** must allow all seating rules to be bypassed if necessary.
- **Do not use ChatGPT or similar tools** to generate seating algorithms, as they cannot produce tailored or functional layouts.

Extra Constraints and requirements (Not mandatory, extra work)

- Include five randomly unavailable or broken seats for each flight/performance. Ensure your algorithm can adapt to these changes dynamically.
- Support cancellations and rescheduling by demonstrating how your algorithm reallocates released seats effectively.
- Include age-restricted zones (e.g., "no children" rows in cinemas or planes) where children cannot be seated.
- Allow flexible seating for elderly or senior people, so they can sit wherever they want.

Implementation and Prototype

You are required to develop a prototype (web or mobile) that demonstrates your seating plan logic.

- The **prototype does not need to be fully complete**, but it must reflect the **core logic of the seating plan and algorithm**.
- Prototypes that show meaningful functionality and **working seating logic** will be **rewarded with higher marks**.
- If the prototype is incomplete or not functional, but your design and rationale are clearly explained (especially in the video), the work still pass, provided all other components are strong.
- A non-functional or missing implementation will limit your final grade to a maximum of a pass or merit. To achieve a Distinction, a well-functioning implementation that meets all requirements is essential in addition to meeting some or all extra requirements given.

Prototype Features to Include (Minimum Viable Product):

- A seating plan as described in mandatory constraints.
- Selectable seating grid.
- Application of rules (e.g., prevent single seat gaps).
- Satisfy all mandatory requirements/constraints listed.

Report Structure :

1. **Cover Page**
   - Student name, ID, programme, assessment title
   - **Video link**
2. **Introduction**
   - Project purpose and selected business context
3. **Requirements**
   - At least 3 non-functional and 8 functional requirements
4. **System Modelling**
   - At least 1 Use Case Diagram
   - At least 2 Activity Diagrams
   - At least 3 Sequence Diagrams
   - Class Diagrams that show a clear OOP approach
   - Seating algorithm design (flowchart or UML/DFD)
5. **Prototype Overview**
   - Technology used
   - Main features implemented
   - ***MVP (a runnable version of the prototype must be explained and displayed)***
   - Describe how business rules are handled in the prototype (e.g., single seats)
6. **Testing Strategy**
   - TDD (required)
   - Optional: BDD or scenario testing
7. **Discussion and Reflection**
   - Evaluation of design decisions and outcomes
8. **Conclusion**

9. **References** (Harvard style)
10. **Appendix** (full code used for algorithm)

please do not put any of the UML diagrams or algorithm logic to Appendix section.

please submit your report as a PDF file.

Video Requirements:

- Length: **5–10 minutes**
- Must be uploaded to a **public platform** (YouTube, Vimeo, Panopto, etc.)
- Link must be shared on the cover page, not anywhere else in the report
- Visibility set to **“unlisted” or “public” – SHOULD NOT BE PRIVATE**
- Include the **video link clearly on the cover page** of your PDF submission
- The video link should be clickable and directly goes to the video demonstration
- Dividing video into segments is a good practice and a structure is given below

---

Video Structure:

1. **Introduction:** Problem context and business case
2. **Design Model & Requirements:** Highlight system architecture, UML diagrams, and constraints
3. **Algorithm Explanation:** Show how your seating algorithm works in creative ways such as with an interactive activity diagram or with a flowchart animation.
4. **Prototype Demonstration:** Functionality walkthrough (web or mobile prototype)
5. **Reflection:** Discuss process, challenges, and lessons learned
6. **Conclusion**

The prototype demonstration should be on a web or mobile app and recorded in the video. It is strongly suggested to do a full working prototype, but completing a fully working seating plan algorithm is not mandatory to pass this assessment.

Marking Breakdown (Total: 40%)

Component	**Weight**
Design & Requirements	20%
Prototype & Reflection	20%