**Summary**

A clear problem statement for a library system defines the limitations of current manual processes—such as lost records, slow book retrieval, and write-only file storage—and establishes the goal of an object-oriented, CSV-backed application that automates member management, item cataloguing, borrowing/returning workflows and reporting. This statement identifies key stakeholders (librarians, members, administrators), outlines the scope (four CSV files for librarians, members, items and library status), and highlights non-functional requirements (data integrity, ease of use, maintainability) to guide design and implementation in line with Year 11 Software Engineering outcomes.

**1. Background & Context**

Traditional libraries often rely on manual paperwork and standalone ledgers, leading to inefficiencies and frequent errors.

* **Record Loss & Damage**: Physical files can be misplaced or deteriorate over time, impeding the ability to track resources accurately.
* **Time-Consuming Searches**: Patrons wait in queues and librarians manually locate books, which hinders learning and resource access.
* **Limited Scalability**: As collections grow, manual methods become increasingly untenable, with no automated way to enforce borrowing rules or generate reports.

**2. Problem Definition**

**Problem Statement**:

“In the current scenario, the library operates with manual record-keeping for books, members, and transactions, resulting in time-consuming searches, lost or damaged files, and no centralized way to monitor item availability or borrowing history. There is a need for a simple, CSV-backed software system that manages librarians, members, items and library transactions through object-oriented principles to automate searching, borrowing, returning and reporting, while ensuring data integrity and ease of maintenance.”

**3. Stakeholders & Users**

* **Librarian**: Needs quick lookup of item status, efficient issuance/return workflows and minimal data-entry errors.
* **Library Member**: Requires fast checks of item availability and borrowing history.
* **Administrator**: Demands reports on overdue items, member activity and system health.

**4. Scope & System Boundary**

* **In Scope**:
  + Four CSV files: librarians.csv, members.csv, items.csv, library.csv.
  + Core operations: create/read/update records, borrow/return items, generate basic reports.
* **Out of Scope**:
  + GUI interfaces or networked multi-user concurrency (this is a console/Python exercise).
  + Integration with bar-code scanners or online catalogues.

**5. Functional Requirements**

1. **Member Management**: Add, view and update member records in members.csv.
2. **Item Cataloguing**: Add new items, search by ID/title, update status in items.csv.
3. **Borrow/Return Workflow**: Validate availability, record transactions in library.csv, update item status.
4. **Librarian Accounts**: Manage staff credentials in librarians.csv.

**6. Non-Functional Requirements**

* **Data Integrity**: CSV operations must not corrupt existing data; implement read-modify-write safely.
* **Usability**: Command-line prompts should be clear and guide correct input.
* **Maintainability**: Code structured with classes (Member, Item, LibraryManager) and methods for each operation.
* **Performance**: Suitable for small-to-medium collections (hundreds of records), with acceptable console I/O latency.

**7. Constraints & Assumptions**

* **File-Based Storage**: No relational databases—must use CSV files.
* **Single-User**: No concurrent access; assume one process at a time.
* **Python Environment**: Version 3.7+ with standard csv module available.

**8. Alignment to Year 11 Software Engineering**

* **SE-11-01 & SE-11-03**: Analysing manual vs automated systems and modelling with DFDs/structure charts (already covered).
* **SE-11-05**: Applying OOP principles in class design for domain entities.
* **SE-11-06**: Implementing secure file I/O and validating data in CSV-based persistence.

By grounding the library system in a concise problem definition, students can directly map requirements to their OOP design and CSV-based implementation, ensuring a focused, syllabus-aligned development process.