Spreadsheet Project: Deliverable 1 Report

Introduction

This deliverable describes the main use cases for the spreadsheet system and the initial domain model. The goal is to define what the system should support at this stage and how the main parts of the design fit together. The project focuses on core spreadsheet behavior: loading, saving, editing, formula evaluation, and consistency management. For this assignment, features such as formatting, multi-sheet support, or a graphical interface are not considered.

1. Use Cases

Each use case includes an explanation, reasoning, and trigger. Together they describe how the user interacts with the spreadsheet system and what the system must handle.

1. UC1: Create new spreadsheet
   • Explanation: Create an empty spreadsheet with a defined or default size.
   • Reasoning: The program needs an initial grid of cells.
   • Triggers: User selects “new spreadsheet”.
2. UC2: Load spreadsheet (S2V)
   • Explanation: Read an S2V file, parse contents, validate formulas, rebuild dependencies, and recompute values.
   • Reasoning: The program must restore a consistent state from a file.
   • Triggers: User selects “load spreadsheet”.
3. UC3: Save spreadsheet (S2V)
   • Explanation: Write the spreadsheet to an S2V file with the correct separators.
   • Reasoning: Users must be able to store spreadsheet data.
   • Triggers: User selects “save spreadsheet”.
4. UC4: Show spreadsheet summary
   • Explanation: Show the spreadsheet size and the number of non-empty cells.
   • Reasoning: Users may want a quick overview.
   • Triggers: User selects “show summary”.

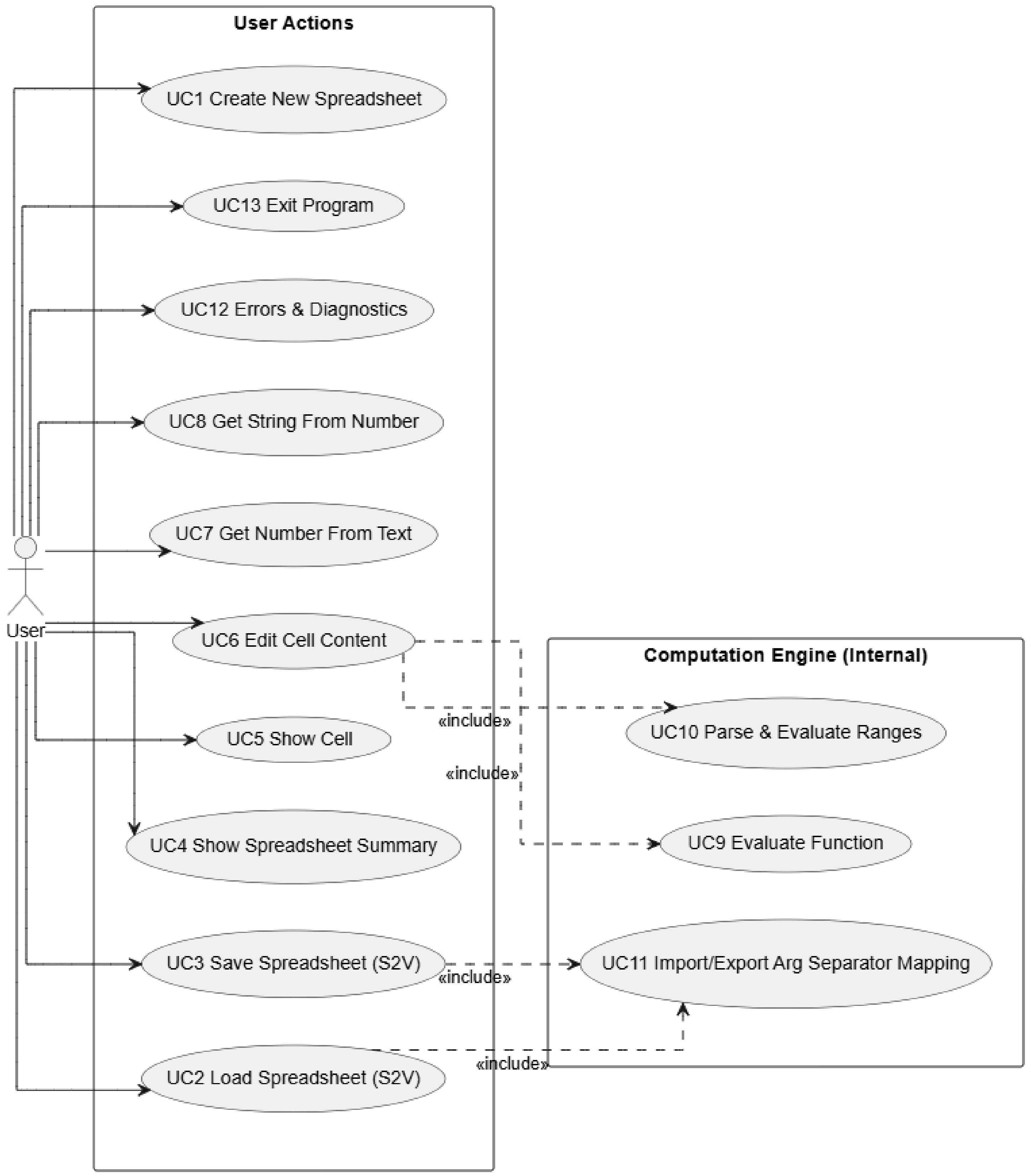
5. UC5: Show cell
   • Explanation: Display the stored content and computed value of a cell.
   • Reasoning: Users need to inspect formulas and results.
   • Triggers: User requests a specific cell such as A1.
6. UC6: Edit cell content
   • Explanation: Update a cell, parse and validate formulas, check for circular dependencies, update the dependency graph, and recompute affected cells.
   • Reasoning: Editing must keep the spreadsheet in a valid state.
   • Triggers: User selects “edit cell”.
7. UC7: Get number from text
   • Explanation: Convert text to a number. An empty cell results in 0, valid numeric strings result in numbers, and invalid strings produce an error.
   • Reasoning: Text cells must behave predictably in numeric contexts.
   • Triggers: A formula requests a numeric value.
8. UC8: Get string from number
   • Explanation: Convert a numeric cell to a string or return an empty string if the cell is unset.
   • Reasoning: Needed for display and S2V saving.
   • Triggers: A formula or save operation requests a string value.
9. UC9: Evaluate function
   • Explanation: Evaluate SUMA, MIN, MAX, and PROMEDIO with support for numbers, ranges, and nested functions.
   • Reasoning: Functions form part of the calculation model.
   • Triggers: Formula evaluation or dependent-cell recomputation.
10. UC10: Parse and evaluate ranges
   • Explanation: Interpret ranges such as A1:B3 and expand them into cell references.
   • Reasoning: Ranges are used inside functions.
   • Triggers: A formula includes a range.
11. UC11: Import and export separators
   • Explanation: Convert “,” to “;” on load and “;” to “,” on save inside function arguments.
   • Reasoning: Required by the S2V file format.
   • Triggers: File load or save.
12. UC12: Errors and diagnostics
   • Explanation: Show parsing errors, circular dependencies, invalid references, and evaluation issues.
   • Reasoning: Users need clear feedback.
   • Triggers: Load, edit, or evaluation errors.
13. UC13: Exit program
   • Explanation: Close the program and warn about unsaved changes if needed.

- Reasoning: Avoids accidental data loss.
- Triggers: User selects “exit”.

2. Use Case Diagram

Based on these use cases, I created the use case diagram. The diagram places the user on the left, the spreadsheet system in the center, and the included use cases on the right.

Diagram:


User Actions
UC1 Create New Spreadsheet
UC13 Exit Program
UC12 Errors & Diagnostics
UC8 Get String From Number
UC7 Get Number From Text
User
UC6 Edit Cell Content
UC5 Show Cell
UC4 Show Spreadsheet Summary
UC3 Save Spreadsheet (S2V)
UC2 Load Spreadsheet (S2V)
Computation Engine (Internal)
UC10 Parse & Evaluate Ranges
UC9 Evaluate Function
UC11 Import/Export Arg Separator Mapping
«include»
«include»
«include»
«include»

3. Domain Model

Next we make the domain model. The domain model describes the key parts of the spreadsheet and how they support the use cases.

The central object is the Sheet, which stores all Cell objects and maintains a DependencyGraph. The graph is used to track references between cells and detect cycles early. This avoids situations where formulas refer to each other in ways that cannot be evaluated.

Each Cell stores:

- its raw text input
- a parsed expression when relevant
- a computed value of type Empty, Text, Number, or Error

Storing the parsed expression avoids re-parsing on each evaluation and makes updates more efficient.

Formula handling is divided into two services to keep responsibilities clean and easier to test:

- FormulaParser: Checks whether a string is a formula and produces an expression tree.
- Evaluator: Computes results, manages coercion between text and numbers, handles ranges, and evaluates functions such as SUMA, MIN, MAX, and PROMEDIO. The evaluator queries the sheet for referenced cell values.

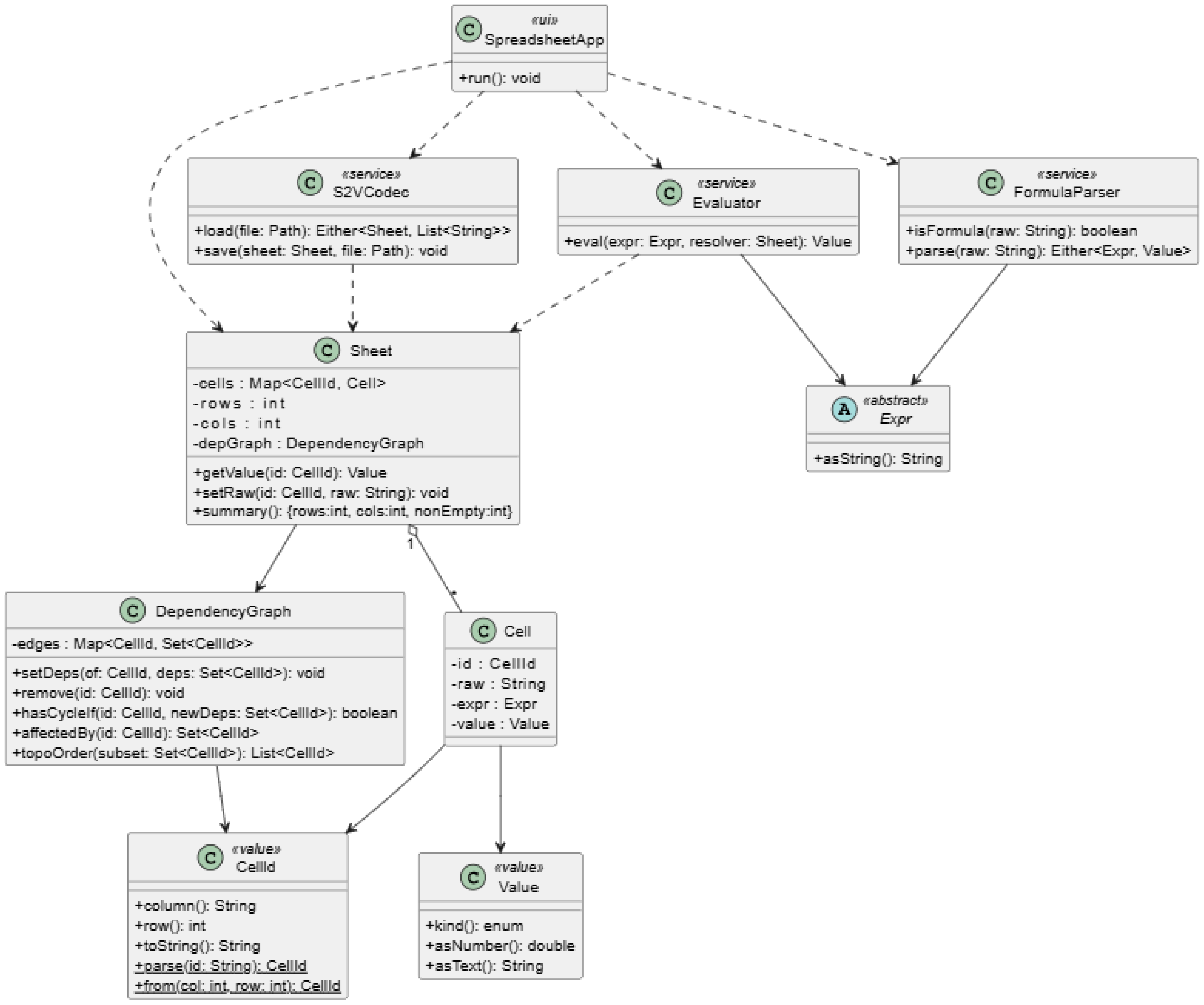
The S2VCodec loads and saves spreadsheets in the S2V format. It also handles conversion of argument separators during load and save operations.

At the top level, the SpreadsheetApp coordinates user actions. It connects the interface-level commands to the domain operations. This ties the use cases to the model. For example:

- UC6 (Edit cell) uses the DependencyGraph to update references and the Evaluator to recompute values.
- UC2 (Load spreadsheet) uses the S2VCodec and then triggers value recomputation in the Sheet.
- UC5 (Show cell) retrieves raw and computed values stored in the Cell.

Diagram:

Domain Model



«ui» SpreadsheetApp
+run(): void
«service» S2VCodec
+load(file: Path): Either<Sheet, List<String>>
+save(sheet: Sheet, file: Path): void
«service» Evaluator
+eval(expr: Expr, resolver: Sheet): Value
«service» FormulaParser
+isFormula(raw: String): boolean
+parse(raw: String): Either<Expr, Value>
Sheet
-cells : Map<CellId, Cell>
-rows : int
-cols : int
-depGraph : DependencyGraph
+getValue(id: CellId): Value
+setRaw(id: CellId, raw: String): void
+summary(): {rows:int, cols:int, nonEmpty:int}
«abstract» Expr
+asString(): String
1
*
DependencyGraph
-edges : Map<CellId, Set<CellId>>
+setDeps(of: CellId, deps: Set<CellId>): void
+remove(id: CellId): void
+hasCycleIf(id: CellId, newDeps: Set<CellId>): boolean
+affectedBy(id: CellId): Set<CellId>
+topoOrder(subset: Set<CellId>): List<CellId>
Cell
-id : CellId
-raw : String
-expr : Expr
-value : Value
«value» CellId
+column(): String
+row(): int
+toString(): String
+parse(id: String): CellId
+from(col: int, row: int): CellId
«value» Value
+kind(): enum
+asNumber(): double
+asText(): String