Simple Math Plotter

1. Vision

The Simple Math Plotter (SMP) is a desktop application with an easy-to-use UI for plotting functions and exporting the results into common file formats.

---

2. Requirements

2.1 Functional Requirements

The SMP app shall satisfy the following functional requirements:

2.1.1 Basic Requirements

- **F1**: The functions sin, cos and sinc must be selectable
- **F2**: Function parameters (A, F, Phi, D) must be settable
- **F3**: Changes in function parameters must be visibly represented instantly
- **F4**: Ranges of the x, y axis must be adaptable
- **F5**: The plot must be a 2D plot
- **F6**: Inputs must be validated
- **F7**: In case of any error relevant to the user an appropriate error message must be shown

2.1.2 Advanced Requirements

- **F8**: On opening the application the last parameters should be set
- **F9**: The graph should be storable as an SVG

2.2 Non-Functional Requirements

The SMP app shall satisfy the following non-functional requirements:

- **NF1**: A change in parameters or ranges shall be visually reflected near instantly
- **NF2**: The UI shall be clearly structured
- **NF3**: Exporting shall only access local storage
- **NF4**: A short end-user documentation shall be provided

2.3 Constraints

For the parameters Amplitude (A), Frequency (F), Phase (Phi), Offset (D) and ranges Xmin, Xmax, Ymin and Ymax we have the following constraints to ensure numerical stability and reasonable visualization:

- $0 < F \leq 100$
- $-100 \leq A \leq 100$

- `-2π ≤ Phi ≤ 2π`
- `-100 ≤ D ≤ 100`
- `-1000 ≤ Xmin < Xmax ≤ 1000`
- `-1000 ≤ Ymin < Ymax ≤ 1000`

---

3. Features

The following features need to be implemented in order to satisfy our requirements:

Feature 1: Implement the SMP UI Layer

- **User Story 1**: Function selection view
  - Task 1: Implement a view for the function selection
  - Task 2: Implement associated viewmodels
- **User Story 2**: Parameters view
  - Task 1: Implement a view for parameter setting
  - Task 2: Implement associated viewmodels
- **User Story 3**: Range view
  - Task 1: Implement a view for the range setting
  - Task 2: Implement associated viewmodels
- **User Story 4**: Graph view
  - Task 1: Implement a graph view
  - Task 2: Implement associated viewmodels
- **User Story 5**: Menu view
  - Task 1: Implement a menu view for storing the graph as an SVG
  - Task 2: Implement associated viewmodels

Feature 2: Validation Logic

- **User Story 1**: Input validations
  - Task 1: Implement validation logic for inputs
- **User Story 2**: Export validation
  - Task 1: Implement logic for validating the export path

Feature 3: Implement SMP Logic

- **User Story 1**: Function Engine
  - Task 1: Implement the function logic

Feature 4: Persistence and Export

- **User Story 1**: Persist Parameters
  - Task 1: Implement a persistence service
- **User Story 2**: Export graph as SVG
  - Task 1: Implement an export service

Feature 5: Error Handling

- **User Story 1**: Implementation
  - Task 1: Implement the error behavior
  - Task 2: Implement the error messages

---

4. Acceptance Criteria

- **A**:
  Given the application is running
  When clicking on the functions dropdown
  Then sin, cos and sinc are selectable.

- **B**:
  Given the application is running
  When setting the function parameters
  Then the changes are near instantly visible.

- **C**:
  Given the application is running
  When changing the axis ranges
  Then the graph is updated near instantly.

- **D**:
  Given the application was running
  And function parameters were changed
  When restarting the application
  Then the previous parameters are loaded.

- **E**:
  Given the application was not running
  Or function parameters were not changed
  When restarting the application
  Then the parameter defaults are set.

- **F**:
  Given the application is running
  When clicking the exporting option in the menu dropdown
  And a destination path was selected via the file explorer
  Then the graph gets exported as an SVG.

- **G**:
  Given the application is running
  When entering values for the parameters and ranges
  Then the validation is run.

- **H**:
  Given the application is running

When a user relevant error occurred
Then a descriptive error message is shown.

---

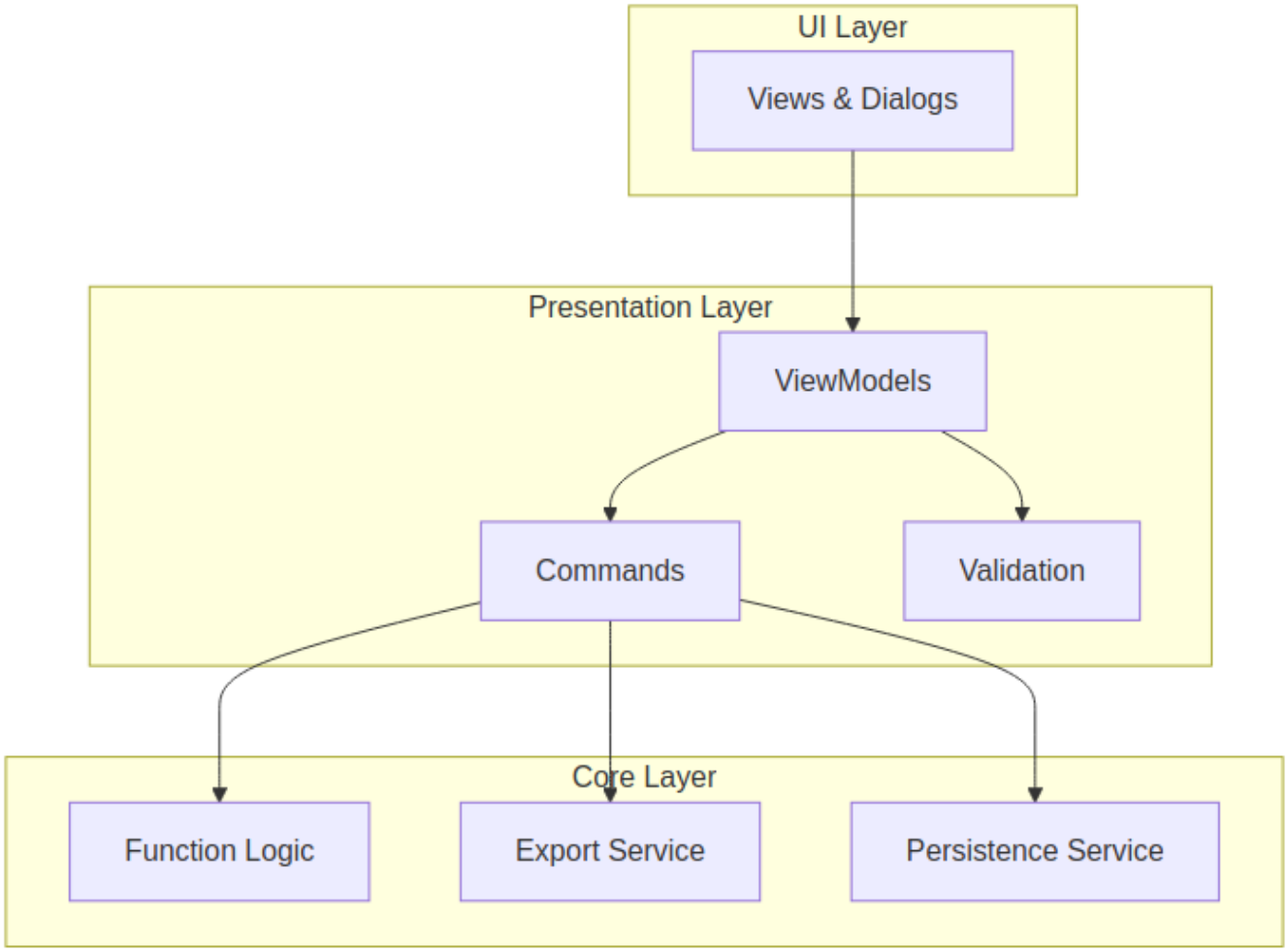
5. Software Concept

5.1 Architecture

5.1.1 Layer

The application is separated in three layers each with their own responsibility:

- **UI Layer**: Views and dialogs presented to the user
- **Presentation Layer**: Viewmodels, commands and validations
- **Core Layer**: Function logic, export and persistence functionality

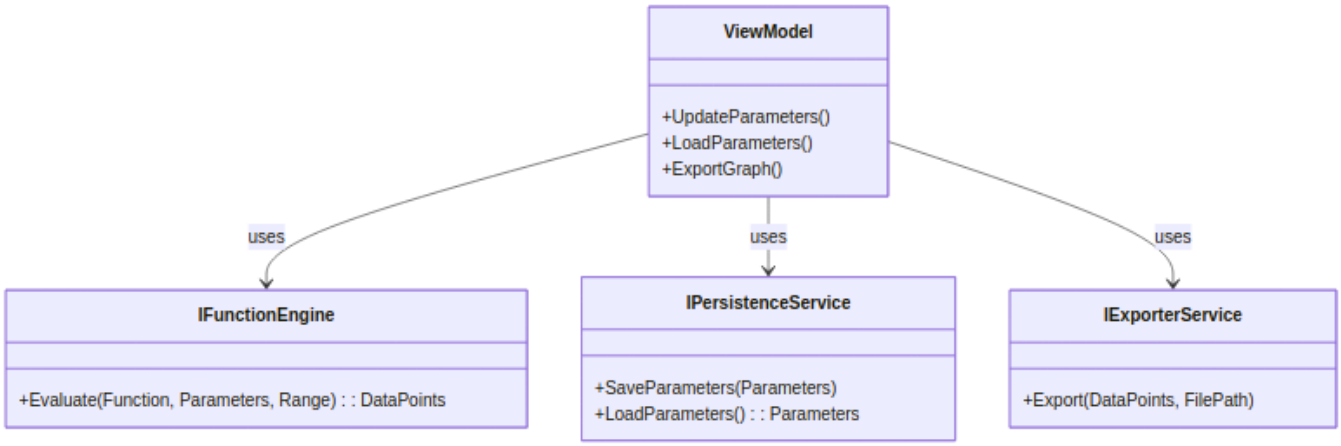


5.1.2 Components

The core responsibilities of the main components are as follows:

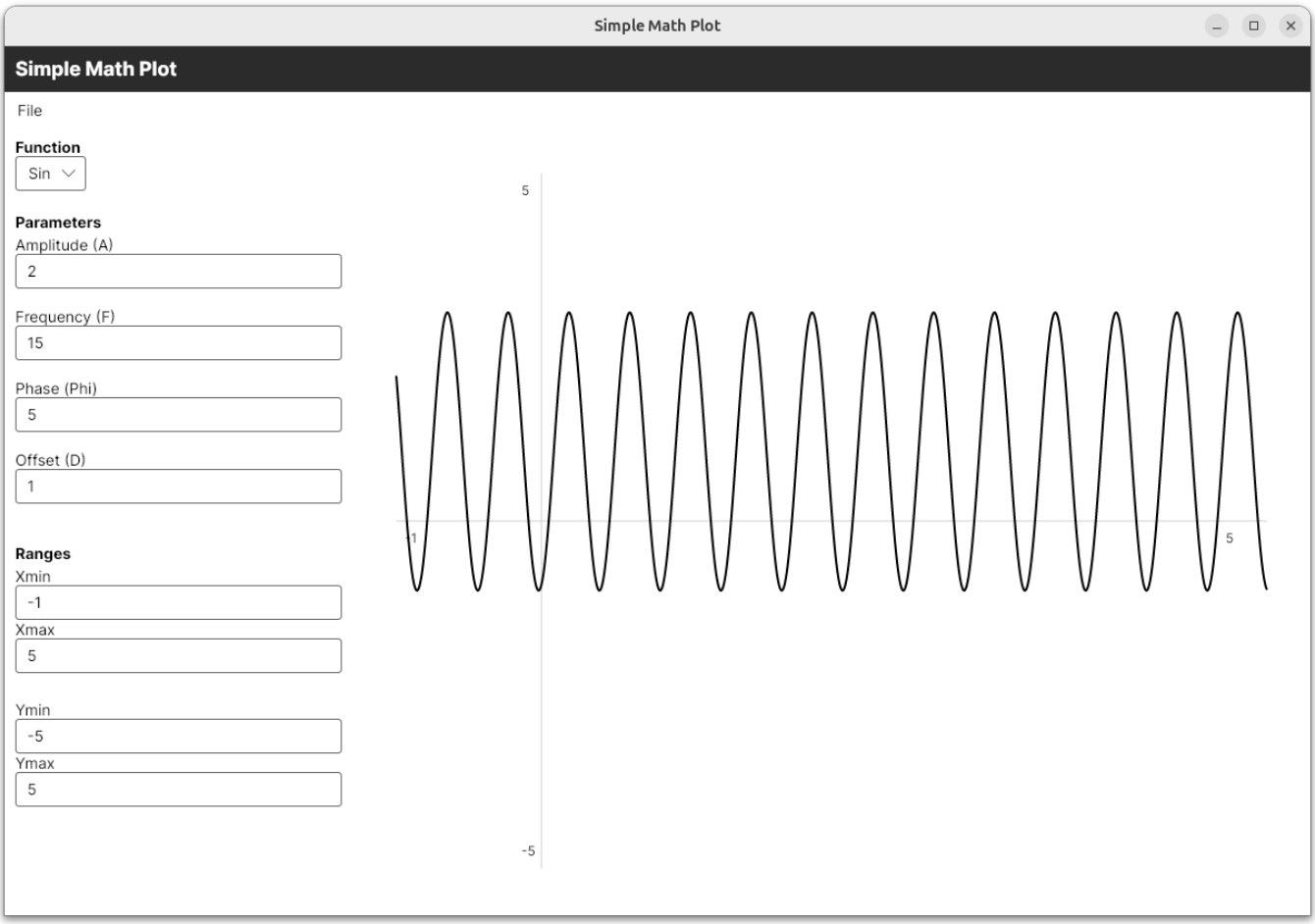
- **ViewModel**: Acts as the mediator between the UI (Views) and the Core services. It collects user input, triggers function evaluations, and updates the graph.
- **IFunctionEngine**: Encapsulates the mathematical logic. It evaluates the selected function (sin, cos, sinc) with the given parameters over the specified ranges and returns data points for rendering.

- **IPersistenceService**: Handles saving and loading of parameters. Ensures that the last-used configuration is available on application restart, or defaults are applied if no configuration is found.
- **IExporterService**: Responsible for exporting the current plot to an SVG file. Provides a clean abstraction so that additional export formats can be added later if required.



5.2 Application UI Design

- **File**: For now it only contains the „Export as SVG“ option.
- **Function**: Selection for the sin, cos and sinc functions.
- **Parameters**: Ability to set each function’s parameter.
- **Ranges**: Set the ranges for the x and y axis.
- **Graph**: Presents the function’s graph.



5.3 Documentation

5.3.1 Function selection

Use the function dropdown to select between the sin, cos and sinc functions.
The general function is given by:

with:

- Fn: sin, cos, sinc
- A: Amplitude
- F: Frequency
- Phi: Phase (in Rad)
- D: y-Offset

NOTE: sinc(0) = 1 by continuation.

5.3.2 Parameters

Specify the associated function parameters in the parameter settings. Use only numeric values.

NOTE: The following boundaries are ensured:

- $0 < F \leq 100$
- $-100 \leq A \leq 100$
- $-2\pi \leq Phi \leq 2\pi$
- $-100 \leq D \leq 100$

5.3.3 Range settings

Adjust the range of the x or y axis in the range settings.

- Xmin: Minimum x value
- Xmax: Maximum x value
- Ymin: Minimum y value
- Ymax: Maximum y value

NOTE: The following boundaries are ensured:

- $-1000 \leq Xmin < Xmax \leq 1000$
- $-1000 \leq Ymin < Ymax \leq 1000$

5.3.4 Persistence

When restarting the application, the previous parameters are automatically loaded.
On a fresh start, default parameters are set.

5.3.5 Export

Exporting the graph to SVG is supported. Click the „Menu“ dropdown and select „Export as SVG“.
This prompts you to select a destination path.

NOTE: The destination path be writable.

5.4 Error Handling

Scenario	Behavior	Message
Input validation error	Show descriptive message, keep last valid value	„Invalid input. Please enter only numeric values.“ „Invalid input. The value must be in range ...“ „Xmin must be smaller than Xmax.“ „Ymin must be smaller than Ymax.“
Persistence	Load parameter defaults, log error	„Settings could not be loaded. Defaults were applied“
Export	Cancel export	„Export failed. No write permission for the selected location“ „Export failed. An unexpected I/O error occurred.“
Unexpected Error	Log error, keep app open but recommend restart	„An unexpected error occurred. Please restart the application.“

5.5 Test Specifications

5.6.1 Functional Test Cases

- **TC1**: Test the input validations on the parameter and range settings.
- **TC2**: Test the export functionality and open the SVG in a viewer to ensure its validity.
- **TC3**: On first start, are default parameters set correctly?
- **TC4**: After parameters have changed, verify that on restart the previous parameters are loaded.
- **TC5**: Test extreme values for the parameter and range settings.
- **TC6**: Verify behavior when the specified path does not exist.
- **TC7**: Are error messages descriptive and user friendly?
- **TC8**: Verify that sinc(0) returns 1 instead of being undefined.

5.6.2 Non-Functional Test Cases

- **TC9**: Verify that graph updates occur within an acceptable time.
- **TC10**: Verify that the end-user documentation is accurate.
- **TC11**: Verify that the exporting functionality only accesses local file storage.

---

6. Prototype versus Product

The following outlines the differences between the prototype and a product development:

Aspect	Prototype	Product
UI	Very basic features and usability	Add mouse-zoom, more menu options such as undo/redo, shortcuts
Export	Simple SVG export to file system	Support more export types, integrate with external APIs
Validation	Basic validation rules, path checks	Enhanced validations, configurable rules, create missing directories
Internationalization	English only	Support multiple languages
Testing	Basic user tests	Sophisticated Unit and Integration Tests, Benchmarks
Documentation	Short description	User Manual, changelogs
Embedding	Standalone app	Integrated in a system, provide/consume APIs
Security	None	File access policies, security checks for export paths
Performance	Not optimized	Optimized for integrated use cases