
Mathinator

Mathinator
Software Architecture Document

Version <1.1>

<Project Name>	Version: <1.0>
Software Architecture Document	Date: <28/11/2016>

Revision History

Date	Version	Description	Author
<28/11/2016>	<1.0>	<Creation of the SAD>	<Lamm, Hug, Saupp>
<27/12/2016>	<1.1>	<Adapt the Technology + Database Diagram>	<Lamm, Hug, Saupp>

<Project Name>	Version: <1.0>
Software Architecture Document	Date: <28/11/2016>

Table of Contents

1.	Introduction	4
1.1	Purpose	4
1.2	Scope	4
1.3	Definitions, Acronyms, and Abbreviations	4
1.4	References	4
1.5	Overview	4
2.	Architectural Representation	5
3.	Architectural Goals and Constraints	5
4.	Use-Case View	5
5.	Logical View	6
6.	Process View	6
7.	Deployment View	6
8.	Implementation View	6
9.	Data View (optional)	6
10.	Size and Performance	6
11.	Quality	7

<Project Name>	Version: <1.0>
Software Architecture Document	Date: <28/11/2016>

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

The scope of this SAD is to provide the architecture of the Mathinator project. Affected are the class structure, our use-cases and the

1.3 Definitions, Acronyms, and Abbreviations

1.4 References

Document	Where to find?
Blog	https://mathinator.tobiaslamm.de
Github	https://github.com/SaschaHug/Mathinator
Use Case 1 "take a picture"	https://github.com/SaschaHug/Mathinator/blob/master/1_UC_Mathinator_Take_A_Picture.pdf
Use Case 2 "view history"	https://github.com/SaschaHug/Mathinator/blob/master/2_UC_Mathinator_View_History.pdf
Use Case 3 "show tour on first start"	https://github.com/SaschaHug/Mathinator/blob/master/3_UC_Mathinator_Delete_Entry.pdf
Use Case 4 "enable user to delete entries"	https://github.com/SaschaHug/Mathinator/blob/master/4_UC_Mathinator_Use_Manual_Calculator.pdf
Use Case 5 "do manual calculations"	https://github.com/SaschaHug/Mathinator/blob/master/5_UC_Mathinator_Show_tour.pdf

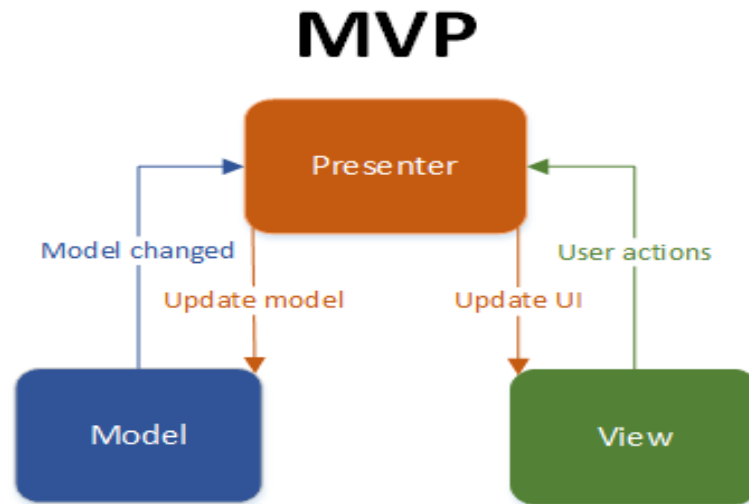
1.5 Overview

The rest of the document is separated into ten different chapters.

Chapter 2 will describe what software architecture is for the current system, and how is represented. Chapter 3 will cover the software requirements and objectives that have some significant impact on the architecture. Chapter 4 lists use cases/scenarios from the use-case model which have significant impact on the architecture itself. Chapter 5 describes the architecturally significant parts of the design model, such as its parts Controller and Model. Chapter 9 is a description of the persistent data storage perspective of the system.

<Project Name>	Version: <1.0>
Software Architecture Document	Date: <28/11/2016>

2. Architectural Representation



3. Architectural Goals and Constraints

In Android you don't have MVC, but you have the following:

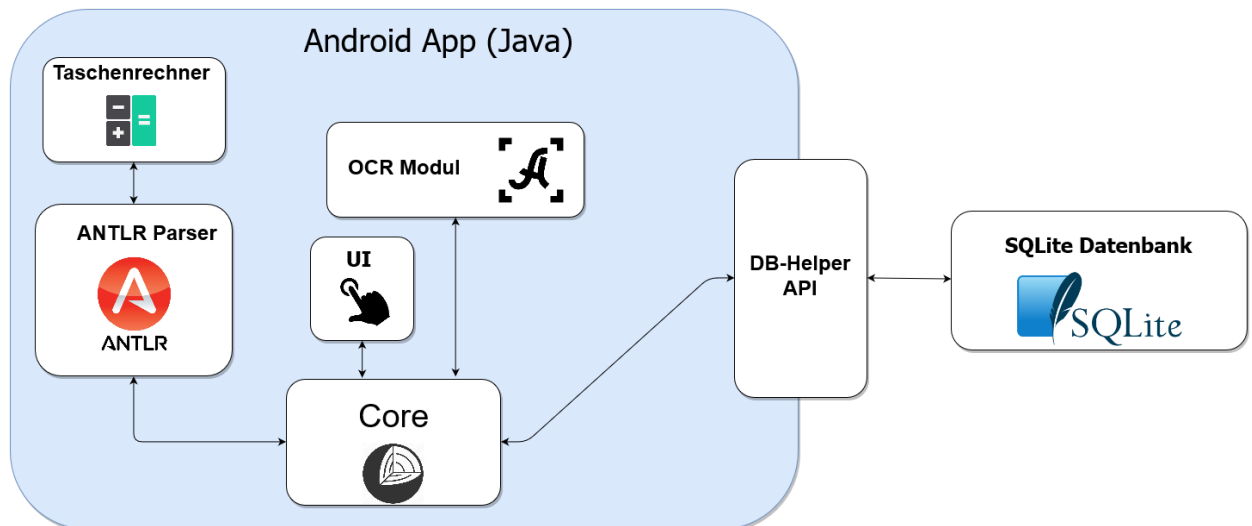
- You define your user interface in various XML files by resolution, hardware, etc.
- You define your resources in various XML files by locale, etc.
- You extend classes like ListActivity, TabActivity and make use of the XML file by inflaters.
- You can create as many classes as you wish for your business logic.
- A lot of Utils have been already written for you - DatabaseUtils, Html.

4. Use-Case View

n/a

<Project Name>	Version: <1.0>
Software Architecture Document	Date: <28/11/2016>

5. Logical View



5.1 Description

The Application is structured into different pieces, each of which are designated to implement a specific functionality.

The ANTRL Runtime is generated by a Parser Generator which in turn relies on a Context-Free Grammar File that describes how valid Equations are be structured. The Parser Routines are auto-generated and thus, changing the grammar / adding new features can be done easily since the code only needs to be modified in one place.

The OCR Module is in Control of reading and processing the users handwriting, which is matched against an example-set. A neural Network is used to compare the two and find the closest match. The UI lets the User control / start these Actions.

The DB-Helper API is to ensure that all CRUD Operations are bound to a Singleton Instance of the SQLite Database, so that at any point, there may only be one instance of the Database across the Application.

6. Process View

n/a

7. Deployment View

n/a

8. Implementation View

n/a

9. Data View (optional)

To be determined

10. Size and Performance

n/a

<Project Name>	Version: <1.0>
Software Architecture Document	Date: <28/11/2016>

11. Quality

n/a