Automatic Grade Collector Projectdocumentation

Software Engineering II Softwareprojekt – von Ben Leitzbach, Michael Oder, Henrik Kaltenbach und Sebastian Trauth

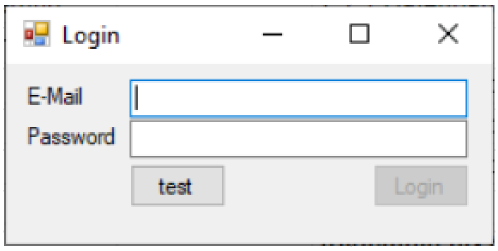
Table of contents

User Guide

The user guide is limited to the client because normal users have no direct interaction with the server.

Login

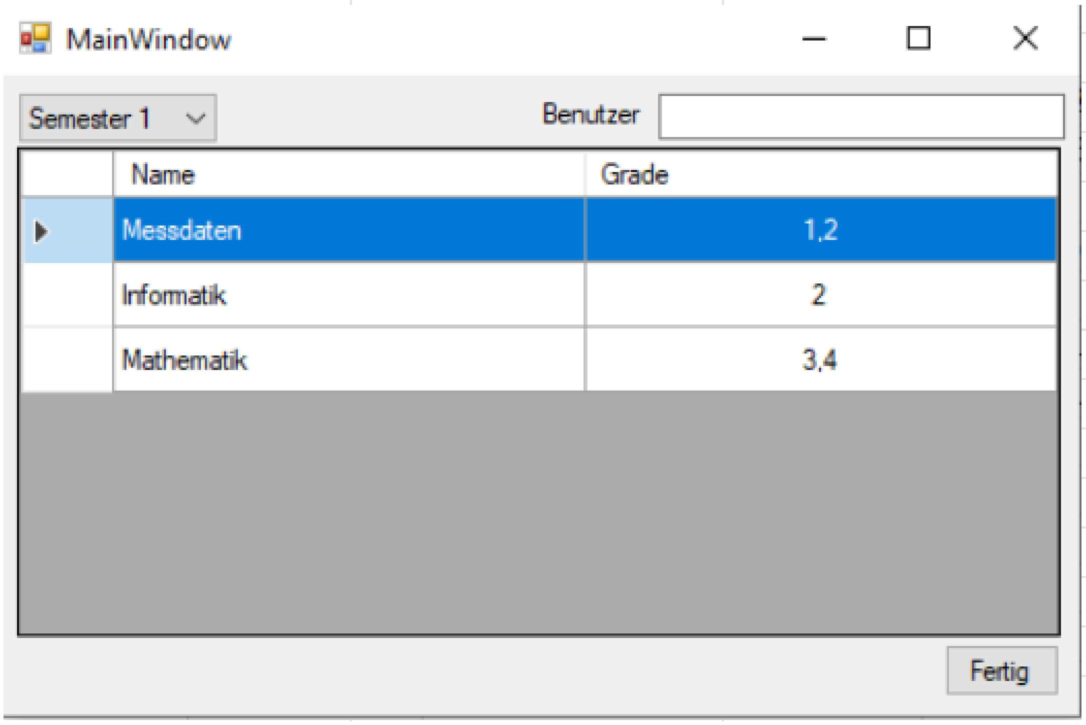
When the application is started, the login window opens first.



In the input field ‘E-mail’ the user e-mail can be entered. The password is entered in the ‘Password’ input field (each password character is represented with a dot). The button ‘test’ is only for testing purposes during development. With the button ‘Login’ the user can log in.

Main window

After successful login, the window for the grade overview appears depending on the semester and the student.



Name	Grade
Messdaten	1,2
Informatik	2
Mathematik	3,4

The semester can be selected via the drop-down menu at the top left. In the main window you can see all modules including the grades. If you change the entry in the notes, you can send the change to the server via the button 'Fertig'.

Developer Guide

After cloning the repository, the html file for the server (https://github.com/Seestali/automatic-gradeCollector/blob/server-dev/docs/_build/html/index.html) can be opened via any browser. The documentation of functions and classes from the server can be viewed with this link (link of local HTML-File).
The method documentation from the client is realized by inline-commentating at the source code.

Architecture Design

1 - Introduction and Goals

This chapter describes the structure of the Automatic Grade Collector application. This application provides modules for students for whom they can view, enter or change their grades.

1.1 Tasks

1	The system should provide the entries of the modules and the students including their connection to each other.
2	The system should communicate via UDP protocols.
3	Students should be able to view, enter and change their grades.
4	Communication must be secure and lightweight.
5	Users must be authorized to make changes.

1.2 Quality objectives

1	Every student is found.
2	Every module is found.
3	Functions are automatically tested.
4	It must be displayed when querying the modules of a student including his semester.
5	It should be possible to access the grade collection by several users.

2 - Constraints

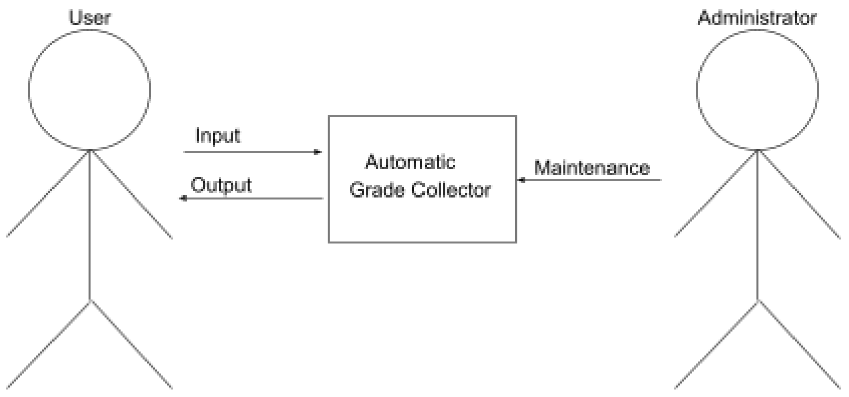
A system is to be developed that is divided into 2 applications. A backend application and a client application. Between the 2 applications, a UDP-based interface can be developed and implemented. Part of the interface should be a well-defined communication protocol, which regulates the communication between the partners. That protocol should be developed in-house and the amount of transported bytes should be as low as possible.
In addition, it should be possible for several clients to connect to a server. Furthermore, the protocol provides the following features:
- Initial client identification and connection setup
- Timeout detection
- Collect communication statistics
- Communication reset
- Keep-alive between server and clients

- The protocol should offer its own packaging logic, so that it is also sent as a stream could be
- CRC for error detection (CRC libraries can be used)
- A client can make "requests" to the server and then receives a "response"
- Only one request per client can be active at a time
- The server regularly sends "status updates" to all connected clients, these are not confirmed by the clients

3 - Context and scope

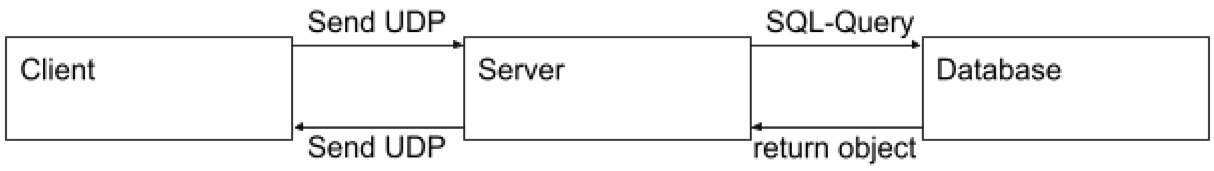
This chapter illustrates the business context as well as the technical context of the project in order to take a closer look at the interfaces and to narrow down the scope of the project.

3.1 Business context



User	Wants to view and change grades. Has to select the semester and the identity.
Administrator	Has access to the database and can add students and modules. Ensures that all entries in the database fit and adjusts them if necessary.
Automated Grade Collector	Processes the inputs and requests from the user in order to provide him with information such as grades for the modules. Is responsible for processing requests from several users at the same time and saving them consistently.
Risk: Input values of the user can interfere with the program in case of incorrect entries. Database must be persistent even if multiple users access it.	

3.2 Technical context



Client	The client sends a udp packet to the server depending on the user's input. It has the task of validating inputs and creating corresponding packets for the udp communications.
Server	The server must translate the udp packets coming from the client. These are processed after translation and forwarded to the database in the form of sql queries.
Database	The database provides the information of each student, module and their relationships to each other.
Risk: Risks here are the errors during the transmission, which is why checksums for the packages are necessary. In addition, intermediate messages and acknowledgment messages must be sent to create secure communication.	

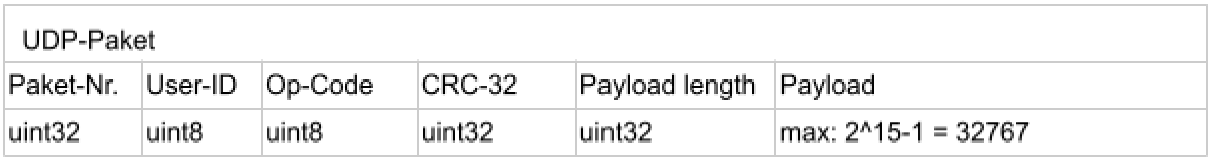
4 - Solution strategy

The following are the most important solution strategies:
-Implementation of the Automatic Grade Collector with Python (server), C# (client) and SQLite (database)
-Introduction of tests for regular function and integration control
-The destination port 42069 is used to communicate with the server
-The UDP packet is firmly divided into individual areas (package no., user ID, op code, CRC-32, payload length, payload)
-All received messages are followed by a confirmation message from the recipient

5 - Building block view

This chapter gives a more detailed overview of individual sub-areas of the project.

5.1 UDP-Paket
This subchapter describes how the packet that is transmitted via UDP is structured and what the individual areas have for tasks.



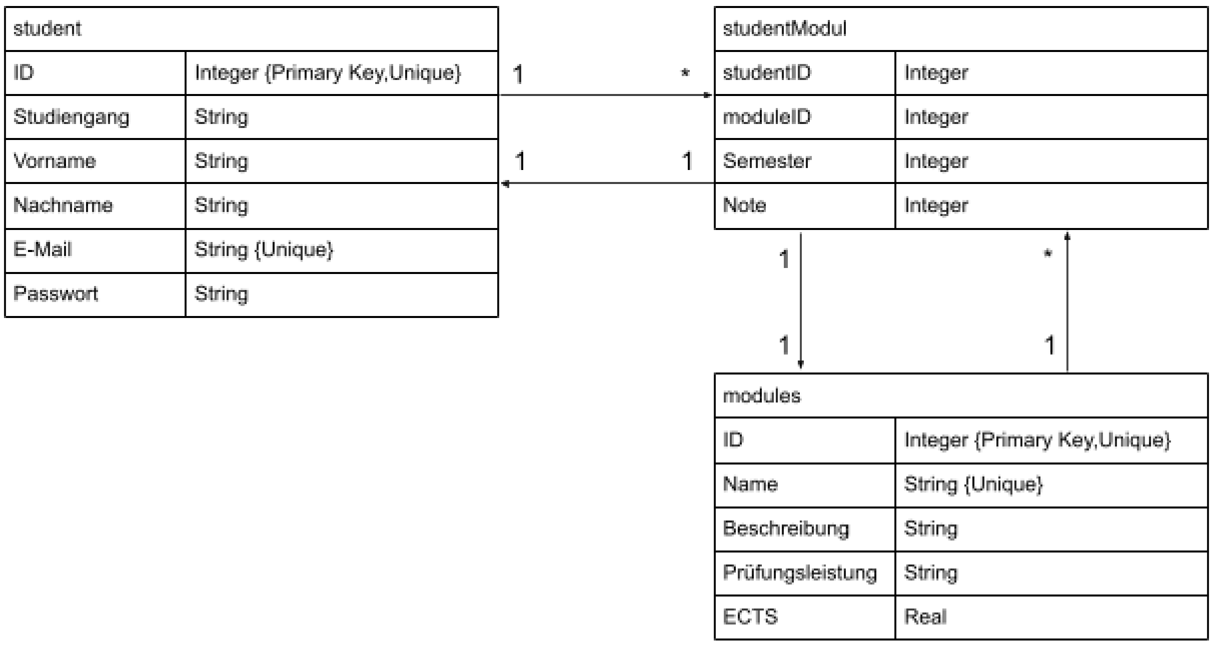
UDP-Paket					
Paket-Nr.	User-ID	Op-Code	CRC-32	Payload length	Payload
uint32	uint8	uint8	uint32	uint32	max: 2^15-1 = 32767

Paket-Nr.	Means the package. Is important for the sequence of processing the packages.
User-ID	Identifies the client. Important for the temporal tracking of the client query (manual time delays).
Op-Code	To detect the task to be performed. Ensures smaller packages.
CRC-32	The checksum ensures that packages have arrived correctly.
Payload length	The payload length describes to the program which area of the message it has to read.
Payload	The payload is the data to be transferred.

Op-Codes	Bezeichnung	Payloads	Who sends?
0	DENY	Paket-Nr, Error	Client/Server
1	ACK	Paket-Nr	Client/Server
2	Login	Authentication example: test@test.de::sha256(test)	Client Req
3	Verify Login		Server Ans
4	Get Fächer, Noten	Authentication, semester example: test@test.de::sha256(test)::4	Client Req
5	Ans Fächer, Noten	Objekt mit Semester, Fächer, Noten	Server Ans
6	Auth, Set Noten	Auth, Objekt mit Fächer-IDs und Noten Beispiel : test@test.de::sha256(test)::{3:2,3:1,7:2}	Client Req
7	Verify Set Noten		Server Ans

5.2 Database

This chapter shows the structure of the database.



student	
ID	Integer {Primary Key,Unique}
Studiengang	String
Vorname	String
Nachname	String
E-Mail	String {Unique}
Passwort	String

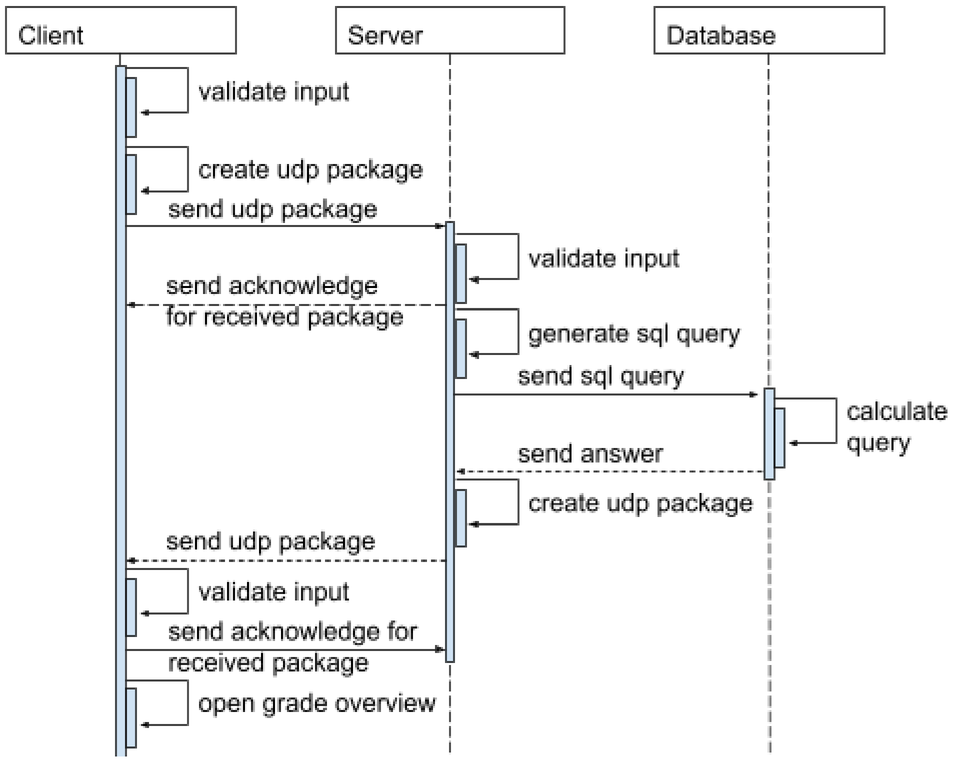
studentModul	
studentID	Integer
moduleID	Integer
Semester	Integer
Note	Integer

modules	
ID	Integer {Primary Key,Unique}
Name	String {Unique}
Beschreibung	String
Prüfungsleistung	String
ECTS	Real

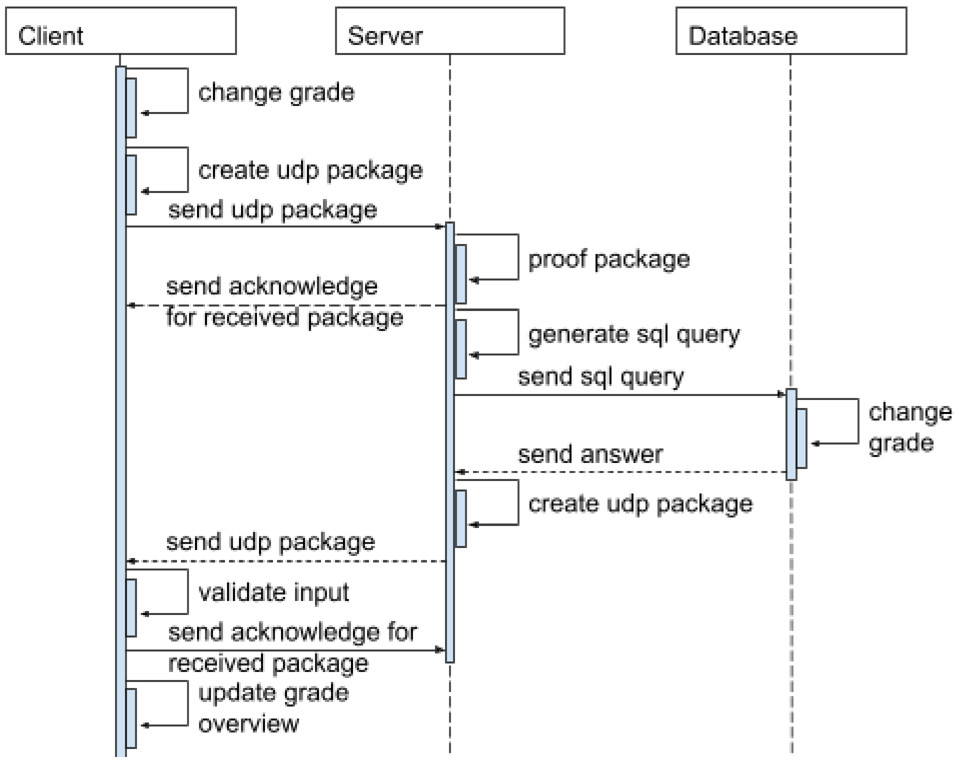
The ‘student’ table describes the individual students (must be unique). The table ‘modules’ describes the individual modules (modules must be unique). In order to link the students to the modules, we create another table called ‘studentModule’, which links the students to the modules and assigns individual grades per semester and module separately for each student.

6 - Runtime view

6.1 Login



6.2 Change grade



7 - Architecture decisions

7.1 Client with C#
C# was chosen for the client because of the dynamic interface and existing knowledge.

7.2 Server with Python
On the one hand, programs in Python can usually be developed much faster than in traditional programming languages such as C(++), Pascal or Java. On the other hand, it is a true platform-independent language that is present on almost all operating systems.

7.3 Database with SQLite
Sqlite is small, lightweight and efficient for easy queries. Since the requirements regarding the complexity of the queries in this use case are low, no more powerful databases are required.

7.4 Just read and write options for existing users and existing modules.
The first goal is to view, enter or change the grades for students and modules. Further functions that administrators perform are currently being implemented via direct access to the server. The implementation of the administrative mode in the client will take place in future versions of the application.

7.5 Insert only submodules
Since the exact composition or weighting is not obvious for the modules, only the submodules are included at first.