VIA University College

Semester Project: Client/Server System

Project Description – Language Learning Tool

Second Semester Software Engineering

Sprogcentermidt
En verden til forskel

Supervisors:

Troels Mortensen

Ib Havn

Students:

Gais El-AAsi – 279910

Lyons Leviticus – 275171

Deivydas Zibkus – 280133

Lukas Vaisnoras – 280107

Marcel Notenboom - 279963

Number of characters incl. spaces 46,316 characters

Table of Content

List of figures

Abstract

The project aims to to provide a Language Learning Platform for Sprogcenter Midt Horsens. The goal is to create a software system that will allow an improved social collaborative teaching tool with the aim of aiding language acquisition and comprehension and thus building the lexical competences of the students.

To achieve this, the project must be able to have a multi-client communication capabilities via a server as well as a database. In addition a graphical user interface must be present as well as a basic level of security.

The development of the solution was done using the unified process framework as well as work-breakdown structure. A lot of attention was paid to developing the user stories and defining the Domain Model.

When designing the software, appropriate Design Patters were used for common problems as well as a better structure. In addition attention was payed to the SOLID principle to improve the overall design of the system.

The result of this project is a software system that has the basic architecture for the final solution, but only satisfies a part of the requirements defined in the beginning. More over testing of the current system was done to identify current issues and shortcomings of the system, structure as well as of the user interface.

1. Introduction – Problem Domain

With more than 500,000 immigrants in Denmark (Larsen, 2019), Denmark is investing in ensuring proper integration into Danish society. An important aspect when it comes to integration is the Danish language.

Sprogcenter Midt has delivered Danish lessons to 10 different municipalities in Denmark in three language centres for the past 25 years. With around 100 staff in 2017, Sprogcenter Midt had almost 3,500 enrolled students (Sprogcenter Midt, 2019).

A request for tenders is put every 4 years by the municipalities of Denmark, and to ensure their competitive advantage, Sprogcenter Midt is focusing on offering high-quality services by utilizing up to date methods and technologies in their teaching process. At the same time, the majority of the teaching is carried out within a traditional classroom environment. Because of that, the language school has recently introduced, in their teaching process, Blended Learning strategy, "Blended learning is an approach to learning that combines face-to-face and online learning experiences" (TeachThought Staff, 2018).

As a part the initiative described above, Sprogcenter Midt has made the study material available online to students, along with external web resources that include language learning activities to complement their studies. Furthermore, the language school is also investigating how they can use technology within the classroom environment to encompass an Active Learning approach to their teaching, "Active learning engages students in the process of learning through activities and/or discussion in class, as opposed to passively listening to an expert. It emphasizes higher order thinking and often involves group work" (Freeman, et al., 2014).

Introduced by Vygotsky's sociocultural theory of development, the approach of embracing the use of cooperative learning groups suggests that learning at a higher level takes places when the students are solving problems beyond their current developmental level with the support of their instructor or peers (Vygotsky, 1978).

Even though Sprogcenter Midt often adopts cooperative group learning within the classroom environment and appreciates the gains from peer to peer learning process, there are still some disadvantages associated with this method. The cultural differences between the students that inhibit productive communication or the differences in the level of knowledge can impede the efficiency of the learning process. Moreover, it affects the students' engagement into the learning process, as the obstacles described above can lead students to becoming passive within the class, should the work prove to be either too easy or too difficult for their level of knowledge.

Sprogcenter Midt has stated that they are in need of a flexible tool for their teaching process. The tool must provide individually based activities within a socially focused environment aiming to build lexical competences. In the field of second language learning studies there are formalized three dimensions of lexical competences: Fluency, Depth of knowledge and Breadth of the vocabulary (Holmen & Lund, 1999).

Currently, on the market, there are different types of mobile, browser and desktop applications that provide among other features audio/video support, flashcards as memory aids, grammatical and pronunciation exercises and other features. The language school uses two quiz browser – applications within the classroom to aid building lexical competences: KAHOOT and Quizlet. However, both are insufficient for their needs as neither have the social collaborative aspect of learning. Moreover, both applications, are partially missing the three dimensions necessary for achieving lexical competence because KAHOOT can only asses what has already been learnt and Quizlet only covers the breadth of vocabulary.

The purpose of this project is to help **Sprogcenter Midt** provide an improved social collaborative teaching tool with the aim of aiding language acquisition and comprehension and thus building the lexical competences of the students.

To define the scope of the project, there is a need for setting up delimitations. Next will be described issues that cannot be addressed:

- The solution will not be a browser-based application;
- The solution will not be a mobile application;
- The solution will not provide audio/video functionalities;
- The solution will not use flashcards as memory aids;
- The solution will not focus on grammatical or/and pronunciation exercises for the learning process;
- The user interface's language will not include Danish (or other language except English) ;

2. Requirements

This chapter will cover the system capabilities and conditions that the system must conform to, it will not include any information in regards to how the system will be implemented as this topic will be discussed further on. The requirements will cover user stories that were defined together with the product owner.

When defining the requirements SMART principles were the main guiding framework so that they are precise and testable in the future.

2.1. User Stories

When defining the user stories a lot of attention was paid to what are the desired functionalities, capabilities and conditions of the user, crossed with the delimitations defined in the Project Description can be found in Appendix A.

To provide a common framework for the user stories a standardized form was used as such:

*As a **user-role** I want **feature** so that a **user-role** can **business value/benefit**.*

Next the defined user stories are presented:

1. As a **teacher/student** I want to be able to log in into the system so I can access information specific to my role (teacher/student) and specific to my account;
2. As a **teacher/student** I want to be able to register myself into the system so that I can have a personal account;
3. As a **teacher** I want to be able to manage (create/delete) a virtual classroom with an unique ID so that I can distinguish between different classes;
4. As a **teacher** I want to be able to create and save a new topic in the system so that I can select it when starting a session;
5. As a **teacher** I want to be able to create and save new words in the topic so that I can pick them when creating a session;

6. As a **teacher** I want to be able to manage (edit/delete) topics that are in the system so that I can maintain and improve the quality of the topics;
7. As a **teacher** I want be able to manage (edit/delete) words that are in the topics so that I can I can maintain and improve the quality of the words;
8. As a **teacher** I want to be able to create a session specifying the virtual classroom it will be assigned to, a topic and a number of words from the word list provided by the selected topic so that the students will learn about the specific topic I chose;
9. As a **teacher** I want the system to generate an unique access code for each session so that I can provide the code to the students;
10. As a **student** should be able to join a session using the provided access code so that I can participate in a session;
11. As a **teacher** I want to be able to see what students are currently connected to a session so that I can be sure that the students are connected to the right session;
12. As a **teacher** I want the students to select only one word from a provided list of words, given that the word was not selected by another student so that I know that only one student will work with a specific word;
13. As a **student** I want to be to enter the question so that the teacher can review it;
14. As a **student** I want be able to specify the type of question from a provided list (five types) and enter the four possible answers specifying the correct one so that it can be used in creation of a quiz;
15. As a **student** I want to be able to see a summary of the question so that I am sure what I am submitting;
16. As a **student** I want to be able to submit the question for review so that I can receive feedback on it;
17. As a **teacher** I want to be able to see a list of submitted questions ready for review(live) so that I can select a question and review them;
18. As a **teacher** I want to be able to select a specific question displayed by the system for reviewing so that I can approve/reject it and give a feedback if needed;

19. As a **teacher** I want be able to accept/reject a question and give feedback if needed so that the student will know how he/she performed;
20. As a **teacher** I want the system to save the accepted question to a pool ready for committing so that I can use them for creating the quiz;
21. As a **teacher** I want to be able to review the pool of accepted questions before committing so that I am sure that only correct questions are submitted;
22. As a **teacher** I want to be able to submit the reviewed and accepted questions so that they are ready for quiz creation;
23. As a **teacher** I want the system to be able to create a quiz out of the submitted questions and generate an access code so that the students can participate in the quiz;
24. As a **student** I want to be able to participate in a quiz by entering previously generated access code so that I know that I am participating in the correct quiz;
25. As a **student** when taking a quiz I want to be able to see the question and the four possible answers so that I can give an answer and submit it;
26. As a **teacher** I want the system to be able to store data about previous sessions (i.e. participants, topics, words, submitted questions, feedbacks) so that I can access them later;
27. As a **teacher** I want the **system** to be able to store previously generated quizzes so that I can reuse them in the future;
28. As a **teacher** I want to be able to access all data about previous sessions and quizzes so that I can review them in the future if needed;
29. As a **student** I want to be able to access my personal data about previous accessed sessions (submitted questions, used words, feedback received) and quizzes (correct/wrong answers) so that I can compare my results and observe the progress;
30. As **student** I want to be able to submit proposal for new topics/words to the teacher, so that the selection process is more democratic and motivating for the students;

2.2. Non-functional requirements

This subchapter will focus on defining the conditions that the system must conform to from a non-functional perspective; as following:

1. The system must satisfy the requirements of a system-client setup (i.e. providing real-time communication between the two);
2. The system must include and use a relational database;
3. The system must use PostgreSQL for its relational database management system;
4. The system (client/server) must be developed using Java programming language;

3. Analysis

In this chapter the focus will be in defining the basic building blocks of the system as well as a rough shape of the system seen from outside. The chapter will start by defining the actors and the glossary of the project, followed by presentation of the Use Case diagram and Use Case Descriptions with different scenarios, a System Sequence Diagram where the system will be treated as a Black-Box and an Activity Diagram. The chapter will end by presenting the Domain Model of the system and a highlight of the most important parts of it.

3.1. Definitions and glossary

In this chapter the Actor Descriptions and Project Glossary will be defined to start shaping the main blocks that system will work with.

3.1.1. Actor Description

The system will have two primary actors: Teacher and Student. It is important to define their roles and to understand that the roles are not interchangeable as they will interact with different parts of the system. Next the roles of each actor will be defined.

Teacher – when referred to the teacher the project will refer to one of the currently employed teachers of the Sprogcenter Midt Horsens, the teacher is the one that has the role of an administrator/coordinator when using the system. It will have the functionality to manage system's resources, create sessions and quizzes as well as review history of the system;

Student – when referred to the student, the project will refer to a currently enrolled student of the Sprogcenter Midt Horsens, the student is the one that has the role of a *user* for the system, meaning that the student will participate in the created sessions and quizzes as well as be to see his/hers personal history.

3.1.2. Project Glossary

To ensure that when referring to a term everyone understands exactly what that term means, it was decided to establish a project glossary with the main/problematic terms.

Account number - an account number is a unique combination of letters and numbers that is at least 4 characters long, that an actor can be identified by, the term can be further referred as: ID, ID number, user number, user account number, etc.;

Classroom – a classroom is a uniquely identifiable entity that has similar functionality to a physical classroom for the purpose of easier identification of specific activities that were done in that classroom. It does not act as a class – meaning that it does not have a list of students that can be identified – it is rather used to only identify activities that has happened in its environment;

Session – is a meeting of the teacher and students, where a topic is selected as well as words from the topic, the students create questions and submit them for review to the teacher, teacher being able to accept/reject and provide the feedback;

Quiz – is an interaction between the students and the questions accepted by the teacher from the session, where the students can answer them and receive feedback in regards to their progress;

Word – a word is defined by its characters and play the role of a basic building block for questions and topics (further defined). The same word can be attributed to different topics, but not to the same topic;

Topic – is an entity that has a unique name and a list of words that are usually related to each other;

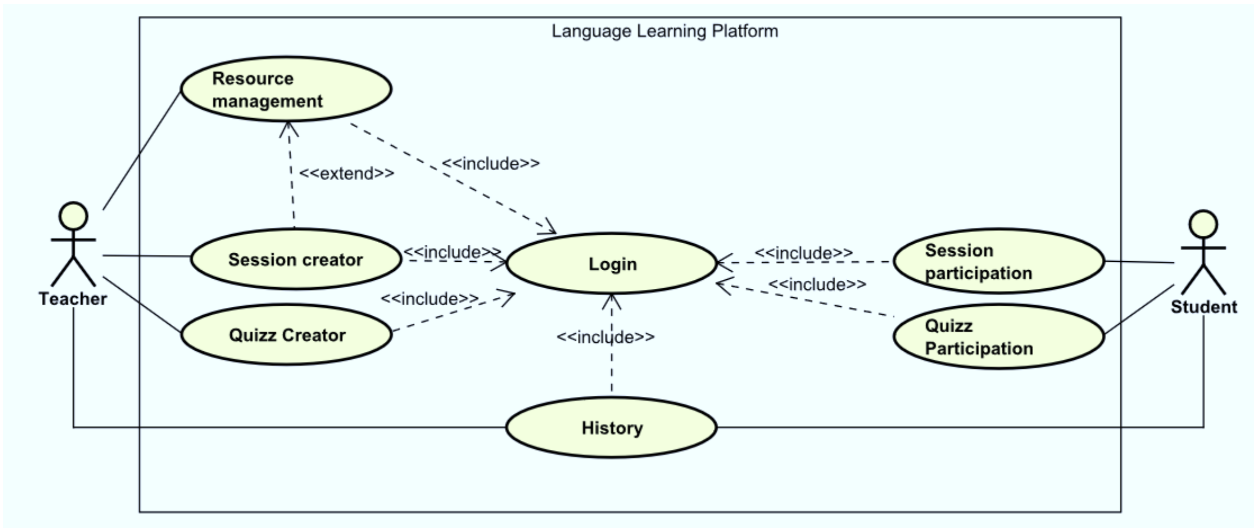
Question – a question is a bundle of one word, a type, a question-text, and four answers one of them being correct;

User – is commonly referred to an actor – teacher and/or student; Use Case Diagram and

3.2. Use Case Diagram

In this subchapter an analysis of the system will be made via a Use Case diagram. The diagram will define the basic functionalities of the system as well as how the users will interact with the system. The Use Case Diagram is found in Appendix B.

Figure 1 - Use Case Diagram



When looking at the Use Case Diagram a few things are to be pointed out for a better understanding. Each use case, includes the Login use case, meaning that a user, before doing anything must be logged in. Logging in is also important because based on the user (i.e. teacher or student) a different functionality will be available.

Another point is that both users have access to the History use case, because both user can see the history of the previous activities. The difference is that based on the user type, different privileges are allowed: a student can access only his/hers own history, but a teacher can access any student's history. As a general note, the use cases encompass many functionalities, thus making the use case diagram less complex. On the other hand, as a use case are responsible for different related functionalities a need for different scenarios appears when describing a use case.

3.3. Use Case Description

In this subchapter the Use Case Description for the Session Creation Use Case will be presented that will include all the scenarios that it covers. All Use Case Descriptions can be found in the Appendix C.

3.3.1. Session Creation Use Case: Scenario I – Creating a session

Figure 2 - Session Creation - Scenario I

ITEM	VALUE
Use Case	Session Creator
Summary	The teacher is able to start a session where student will be able to participate;
Actor	Teacher
Precondition	The teacher must be logged in as a teacher to be able to perform the task, the topic and classroom must already exist;
Base Sequence	1. The teacher enters the session creation section;
Branch Sequence	2.1. The teacher choses the classroom for the session; 2.2. The teacher choses the topic for the session; 2.3. The teacher choses the words for the session; 2.4. The teacher submits the request (Ex. 2.1.1., 2.2.1., 2.3.1.); 2.5. The system generates an unique access code for the session; 2.6. The in-session view is opened (Ex 2.4.1.);
Exception Sequence	2.1.1. The teacher does not select a classroom; 2.1.2. The system informs that the session could not be created; 2.1.3. -> 2.1.; 2.2.1. The teaches does not select a topic; 2.2.2. The system informs that the session could not be created; 2.2.3. -> 2.2.; 2.3.1. The teaches does not select at least one word; 2.3.2. The system informs that the session could not be created; 2.3.3. -> 2.3.; 2.4.1. The session could not be created due to the internal/network issues; 2.4.2. The system informs the teacher that the session could not be created;

	2.4.3. -> 2.1;
Note	After each task performed the system saves the changes;

The first scenario describes the way a teacher creates a session, it involves many steps and it requires that the classroom and topic (including words) are created beforehand – even if there are steps handling the absence of the specified above requirements. Additionally, in order to enter the session creator the teacher must be logged in with a teacher's profile.

Before trying to create the session the system checks if all the provided information are valid (step 2.5.) if otherwise the teacher is informed that something is missing/not valid, and the task is not performed.

If there are connection or internal issues that stopped the system from creating the session the teacher is informed about that.

3.3.2. Session Creator Use Case: Scenario II – Accepting/Rejecting a question with/without feedback

Figure 3 - Session Creator - Scenario II

ITEM	VALUE
Use Case	Session Creator
Summary	The teacher is able to accept/reject a question and send feedback;
Actor	Teacher
Precondition	The teacher must be logged in as a teacher to be able to perform the task, the questions must exist in the system;
Base Sequence	1. The teacher is in a session;

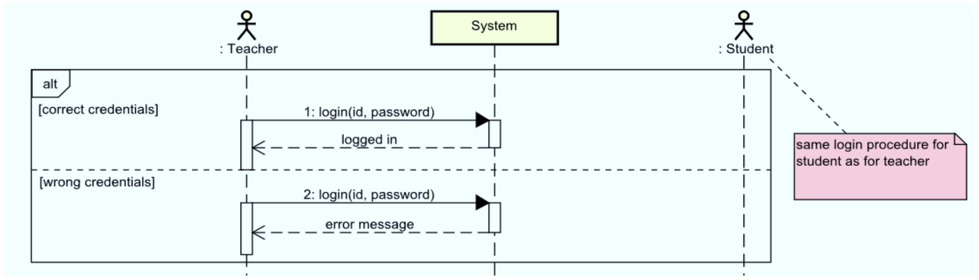
Branch Sequence	2.1. The teacher selects the desired question; 2.2. The teacher writes the feedback for the question; 2.3. The teacher choses to accept/reject the question; 2.4. The teacher submits the request (Ex. 2.1.1.)
Exception Sequence	2.1.1. The teacher does not select a question; 2.1.2. The system informs the teacher that the feedback could not be submitted; 2.1.3. -> 2.1.;
Note	After each task performed the system saves the changes;

This scenario describes the steps needed to be taken for a teacher to submit the feedback for a question. The teacher also must indicate if the question is accepted or rejected. In addition the teacher can chose to notify the student only about acceptance of the question without including a written feedback.

3.4. System Sequence Diagram

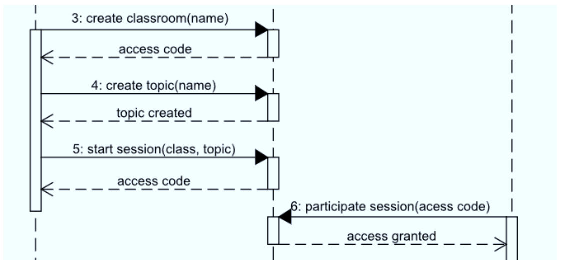
The system sequence diagram describes how the system should operate from an outside point of view, thus not showing any of the inner working of the system (aka Black Box). The importance of the system sequence diagram is to observe the general behavior of the system in relation to the actors and vice-versa. At the same time it shows what information are going in and out of the system as it cover the whole system. The uncropped diagram can be found in the Appendix D. As well as all the diagram can be found in the Astah file in Appendix R.

Figure 4 - System Sequence Diagram Login



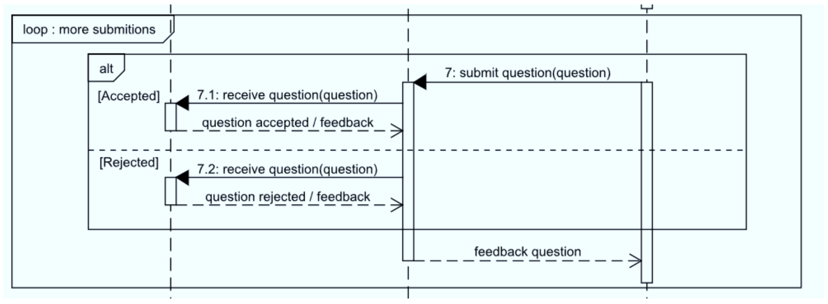
In the Figure 4 the logging process is presented, the user inputs its account number and the password. The system then checks whether the credentials are correct or wrong and based on that gives an output, in form of an error message or next menu.

Figure 5 - System Sequence Diagram - Session


3: create classroom(name)
access code
4: create topic(name)
topic created
5: start session(class, topic)
access code
6: participate session(acess code)
access granted

In the Figure 5 the creation of a session (and the support elements for it – classroom and topic) is presented. After the session was created the system outputs the access code to the teacher. For the student to connect to the session, the access code must be provided.

Figure 6 - System Sequence Diagram - Feedback


loop : more submitions
alt
[Accepted]
7: submit question(question)
7.1: receive question(question)
question accepted / feedback
[Rejected]
7.2: receive question(question)
question rejected / feedback
feedback question

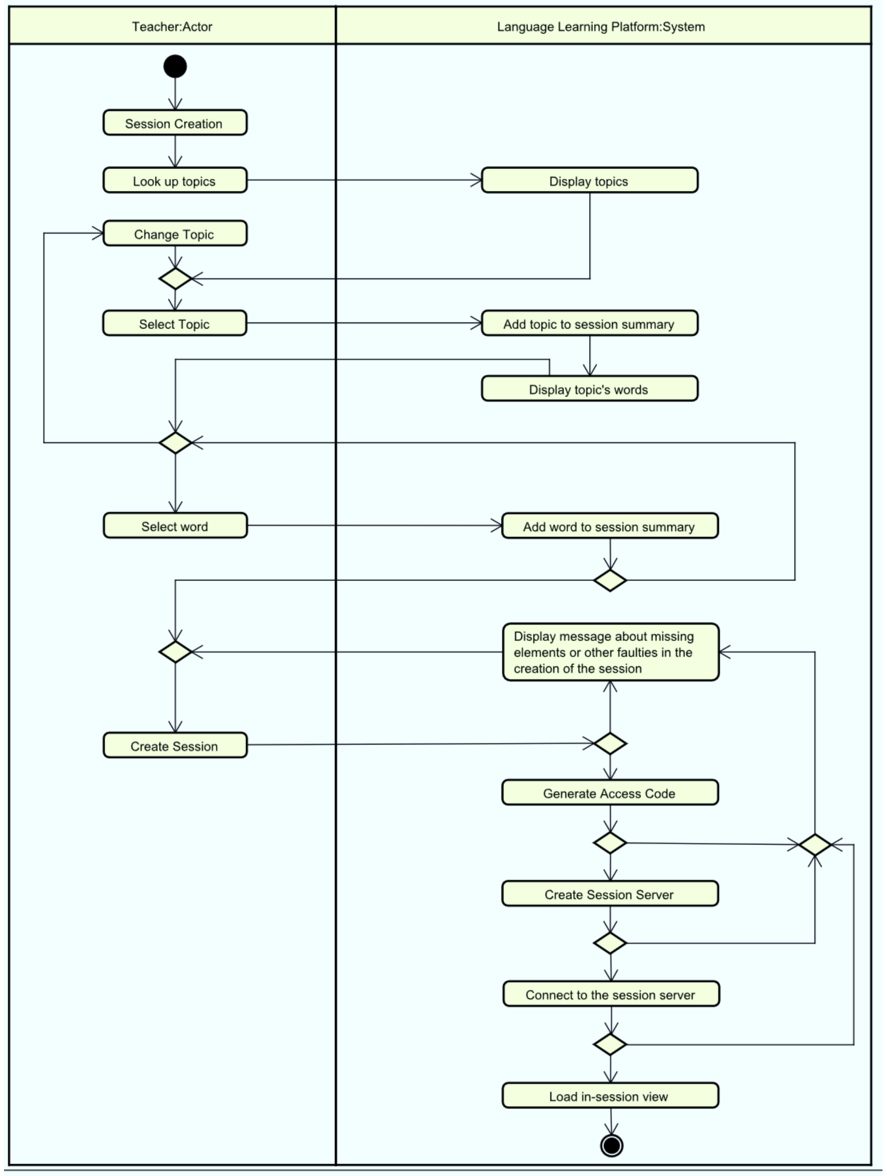
In Figure 6 the process of submitting and accepting / rejecting a question is presented. As a student can submit more than one question the whole process is placed inside a loop. After the question is submitted the teacher decides whether to accept or reject the question and send feedback. To display that the options are placed inside an alternative box and each option has guards to express the differences.

All in all, the system sequence diagram gives a good overview of the future system from a user perspective, by analyzing how the system responds to different inputs from the user.

3.5. Activity Diagram

As a scenario clarification for the creation of a session was needed before going into the design, an activity diagram of the process was created to observe the steps taken when executing it can be found in the Appendix E.

Figure 7 - Activity Diagram Create Session


Teacher:Actor
Language Learning Platform:System
Session Creation
Look up topics
Display topics
Change Topic
Select Topic
Add topic to session summary
Display topic's words
Select word
Add word to session summary
Display message about missing elements or other faulties in the creation of the session
Create Session
Generate Access Code
Create Session Server
Connect to the session server
Load in-session view

The first part of the diagram underlines the process of picking a topic, as well as selecting the words for the topic. The teacher can select multiple words, as well as change the topic if needed at any time.

It is important to point out that there are not many errors which can happen here so therefore there are no paths that would handle the errors.

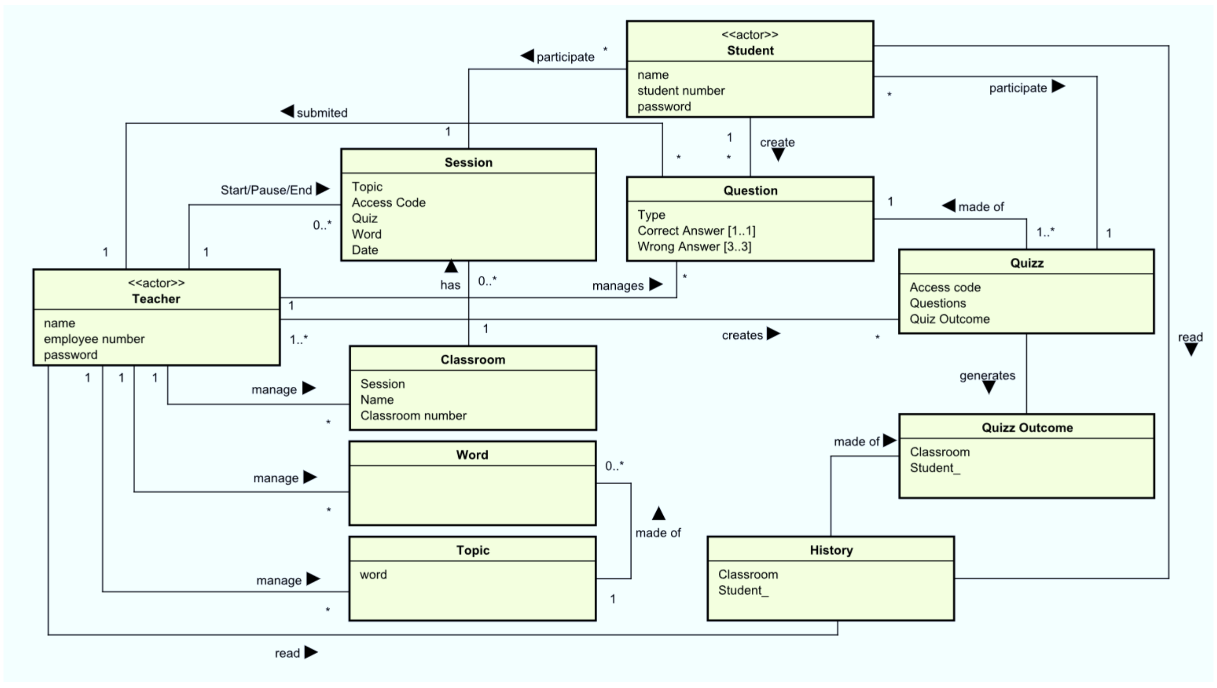
In the second part, after the actor decides to create the session, each step may generate an error, therefore the software tries to handle it by sending out a message to the actor informing about the nature of the problem.

The activity diagram shows a more detailed step by step analysis for the process of creating a session, which is important as it will give good understanding for design when it will come to it.

3.6. Domain Model

The final artifact for the analysis process, the domain model was made to serve two purposes: an overview of the system for a common understanding with the product owner and a foundation for the class diagram when the design process is started, the Domain Model can be found in Appendix F.

Figure 8 - Domain Model



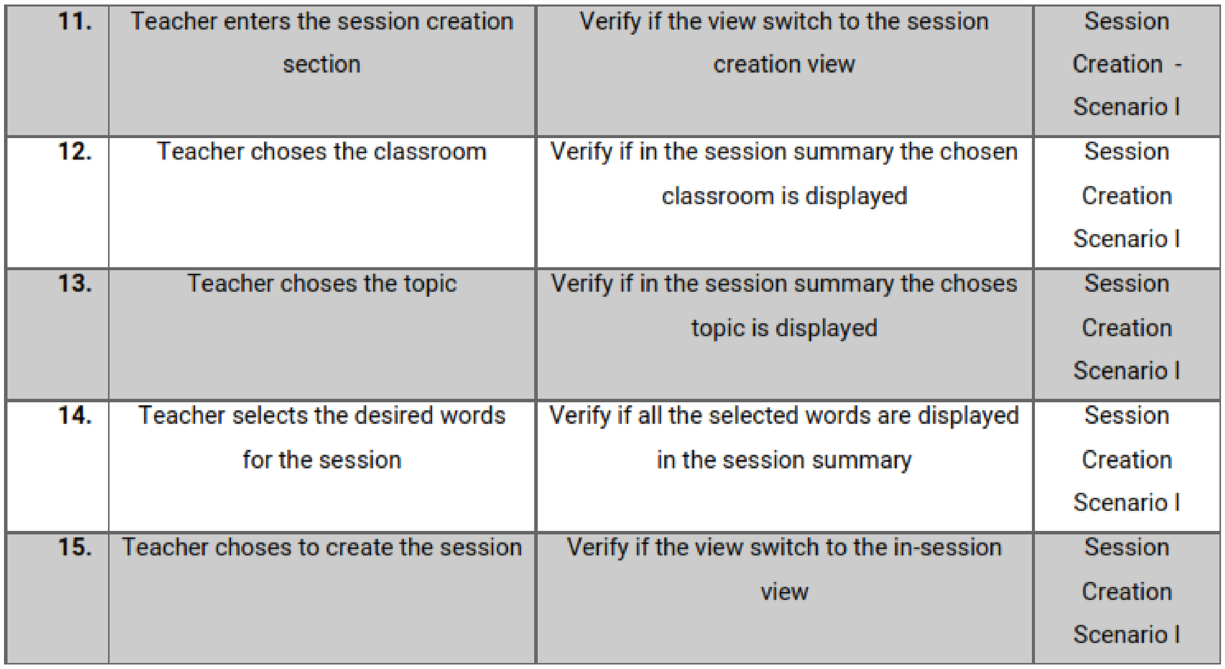
When developing the domain model, the user stories server as a pool for identifying different concepts as well as associations between them. A crucial point to mention is that in the process of defining the domain model different questions were formulated whereas answering them contributed to delimiting and defining more concrete components for the domain.

To conclude, the conducted analyses are providing the necessary information in regards to the system, information necessary in the Design phase. The findings of the analyses provide a strong foundation of the concepts and relation between them, as well as the data that flows between them.

3.7. Test Cases

To be able to test the software test cases (Larman, 2005) were developed for the existing use cases, which will be used for the testing at the end of the project. When developing the test cases it was taken into account the *Reaction* that the user should encounter when a certain *Action* is taken. The Test Cases can be found in the Appendix G.

Figure 9 - Test Case Example



11.	Teacher enters the session creation section	Verify if the view switch to the session creation view	Session Creation - Scenario I
12.	Teacher choses the classroom	Verify if in the session summary the chosen classroom is displayed	Session Creation Scenario I
13.	Teacher choses the topic	Verify if in the session summary the choses topic is displayed	Session Creation Scenario I
14.	Teacher selects the desired words for the session	Verify if all the selected words are displayed in the session summary	Session Creation Scenario I
15.	Teacher choses to create the session	Verify if the view switch to the in-session view	Session Creation Scenario I

It can be observed that the test treats the system from an outside perspective without concerning with the inner workings of how an action is performed. To have an easier way of assessing the success rate of the tests a scale is developed that has three conditions based on how much the reaction was observed during the test, as follows:

- Not functioning – the reaction is totally absent;
- Partially functioning – the reaction is achieved to some degree, but not fully;
- Functioning – the reaction is fully achieved;

3.8. Database Analysis

When developing the Domain Model, a shape of the future database also started to appear as many elements from the Domain Model were easily traceable to the future Entity Relationship Diagram. Even so, the most work for the Database was left in the design part.

4. Design

In this chapter a more in-depth development of the software will be done. The outcome of this chapter should be a class diagram and other diagrams needed for implementations. It will describe the design patterns, principles and techniques used and the reasoning behind their use.

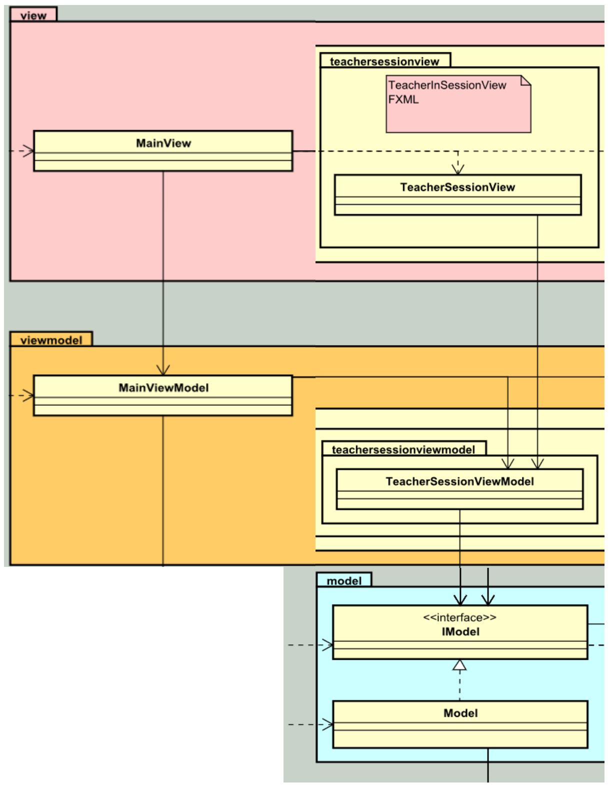
4.1. Design Patterns

Several design patterns were used when developing the structure of the software. The design patterns were used for solving and speeding up development of the software by applying tested, proven and developed paradigms.

4.1.1. The MVVM Pattern

Model-View-ViewModel (Likness, 2014) was used to ensure a proper architectural structure of the client side. The pattern ensured the separation of concerns as well as an easier way of managing different user interface components.

Figure 10 – The MVVM Pattern Example

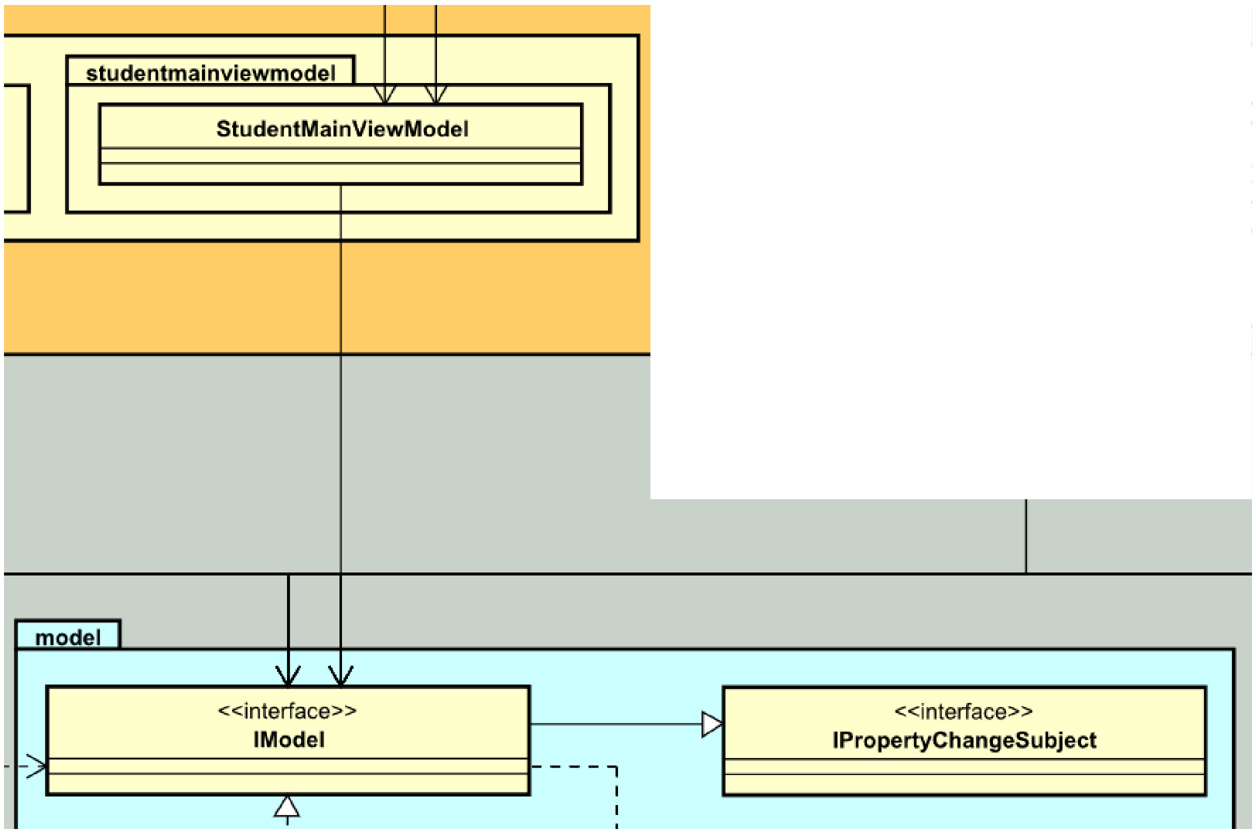


It can be observed that the structure of the MVVM is respected having the view, view model and model. The connection between them are made in several way depending on the situation, property binding, method invocation and events.

4.1.2. The Observer Pattern

Observer Pattern (SourceMaking, 2019) was used especially when implementing the MVVM Pattern as it allowed to connect more easily Model with ViewModel especially when it was needed for the system to automatically update without any input from the user.

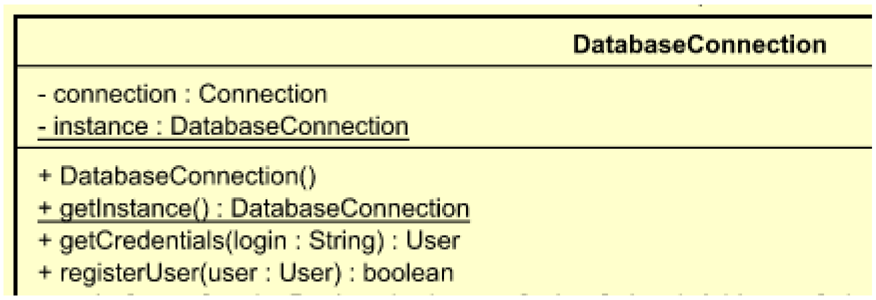
Figure 11 – The Observer Pattern Example



4.1.3. The Singleton Pattern

The Singleton Pattern (GeeksForGeeks, 2019) was used when it was needed to ensure that only one instance of a class was created and reused when needed. An example is the class responsible for the input and output from the database as only one object is needed to take care of those responsibilities.

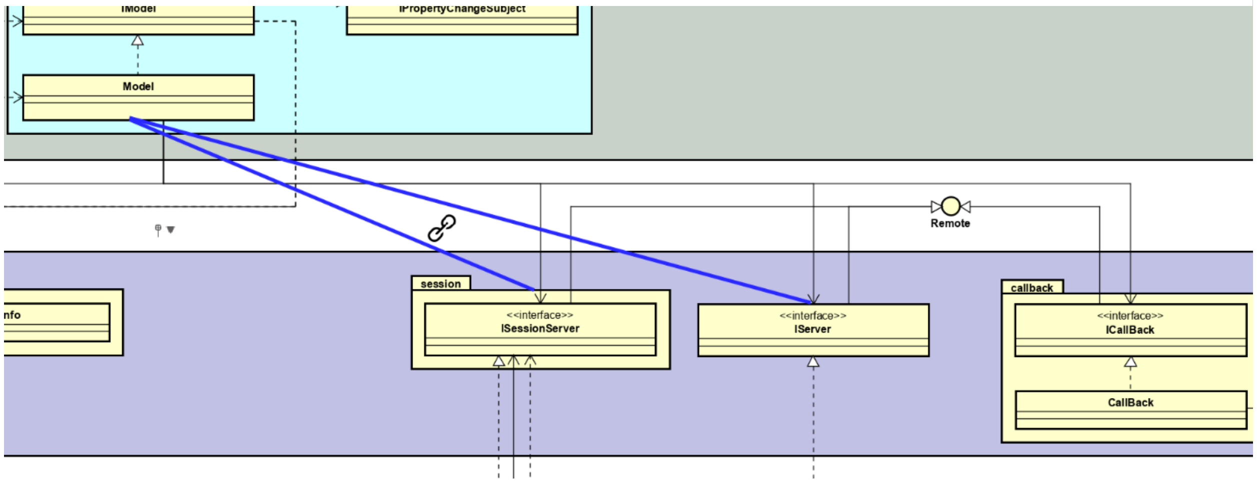
Figure 12 - The Singleton Pattern Example


DatabaseConnection
- connection : Connection
- instance : DatabaseConnection
+ DatabaseConnection()
+ getInstance() : DatabaseConnection
+ getCredentials(login : String) : User
+ registerUser(user : User) : boolean

4.2. The Server – Client Connection

Due to the fact that the system will operate on a local network (Local Network Area of the Sprogcenter Midt) it was decided that for the connection between the server and client will be implemented using Java Remote Method Invocation (Oracle , 2019) as it fitted best the needs of the system. The RMI provided the necessary flexibility and simplicity in the implementation as well as the functionality for the system to operate as intended.

Figure 13 - RMI Connection Example



4.3. Live Client – Server Communication

To develop a live communication between the client and server without the input from the user a CallBack (Liu, 2003) implementation was designed. When connecting to a session server the client will send a callback object which will be registered and added to a list of clients that will be notified when certain things happen.

4.4. Libraries and API

As the system required having a database a communication connection between the system and database needed to be established. It was decided to use the Java Database Connectivity API (Oracle, 2019).

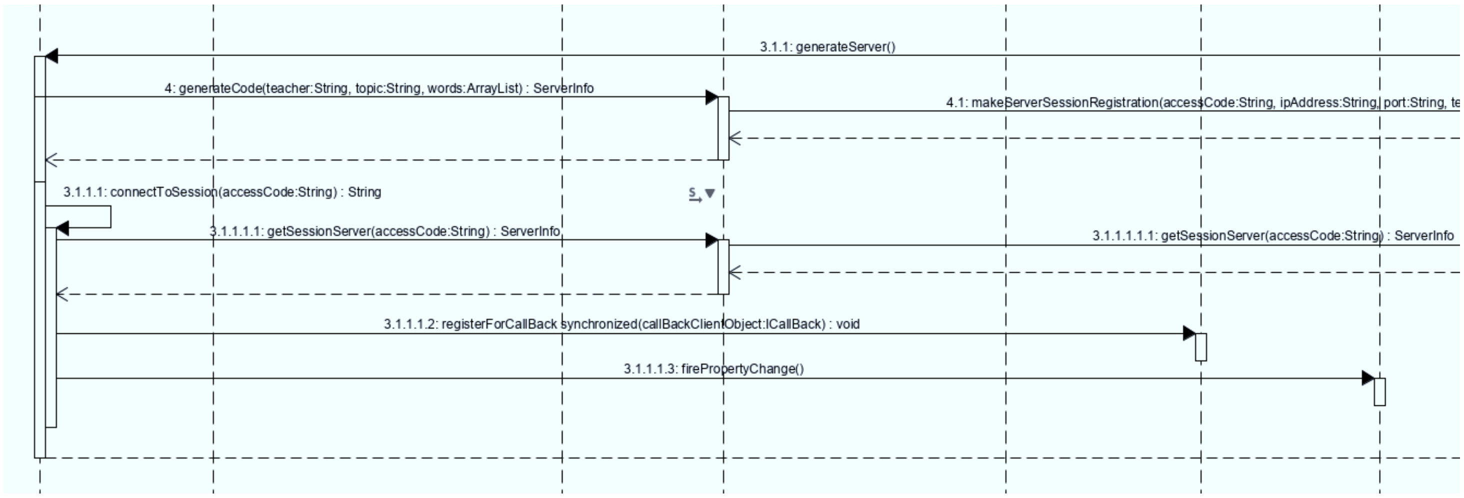
For an integration between the Java and PostgreSQL the PostgreSQL JDBC Driver API (PostreSQL, 2019) was used as it provided all the functionality required for the system.

To ensure a more clean and pleasant user interface look the JFoenix Library (JFoenix, 2019) was used that provides a series of common used components for the JavaFX designed in a Google Material Design style.

4.5. Sequence Diagram

As the procedure of creating a session was rather complex it was decided to create a sequence diagram that will provide an overview of the whole process which also helped identifying some flaws and issues. The sequence diagram was made with the steps presented in the use case description and activity diagram which helped in having a logical overview of the process. The entire diagram is presented in the Appendix H.

Figure 14 - Sequence Diagram Session Server


3.1.1: generateServer()
4: generateCode(teacher:String, topic:String, words:ArrayList) : ServerInfo
4.1: makeServerSessionRegistration(accessCode:String, ipAddress:String, port:String, te
3.1.1.1: connectToSession(accessCode:String) : String
3.1.1.1.1: getSessionServer(accessCode:String) : ServerInfo
3.1.1.1.1.1: getSessionServer(accessCode:String) : ServerInfo
3.1.1.1.2: registerForCallBack synchronized(callBackClientObject:ICallBack) : void
3.1.1.1.3: firePropertyChange()

In the diagram above an important part of the session creation procedure is presented. As the teacher presses the button to create a session a series of steps are taken:

- An access code is created that then is registered with information for a server (IP address, available port number, the list of words as well as the operator of the session);
- After all the information for a server is registered the server itself is created and the client is connected to the server;
- Next a callback object is sent to ensure that the client is notified about the changes;
- As the last thing, a property change event is fired to ensure communication between the model and view model;

Due to the fact that many things are happening at the same time, having a sequence diagram of the entire process ensured that everything is taken into consideration before jumping into implementation.

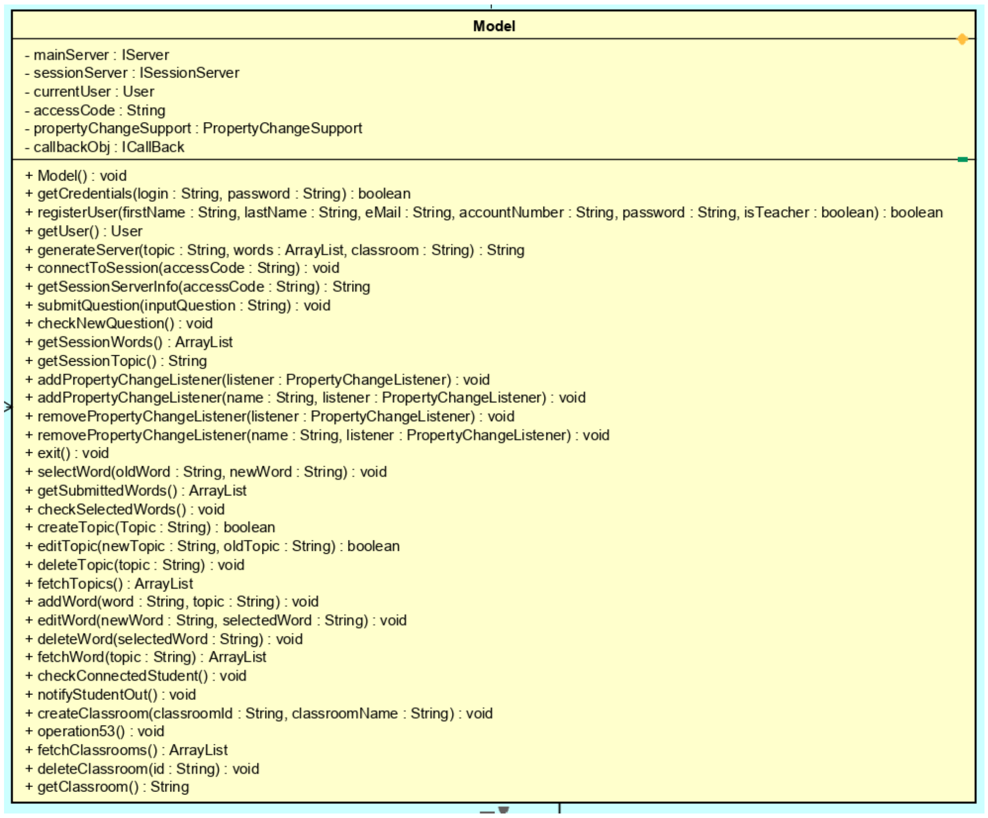
4.6. Class Diagram

When designing the class diagram, the Domain Model was used as the foundation for it as well as Use Case descriptions that helped define the main methods and logical connections. The entire class diagram as well as a compressed one (no methods and/or attributes) is presented in the Appendix I. Some of the important aspects of this, in addition to those presented in the previous subchapters will be presented next.

4.6.1. The Model

The model plays a crucial role in the functioning of the system as it provides the connection with the server, communication between different view models and other types of functionality (callback, validation, etc.).

Figure 15 - The Model Class



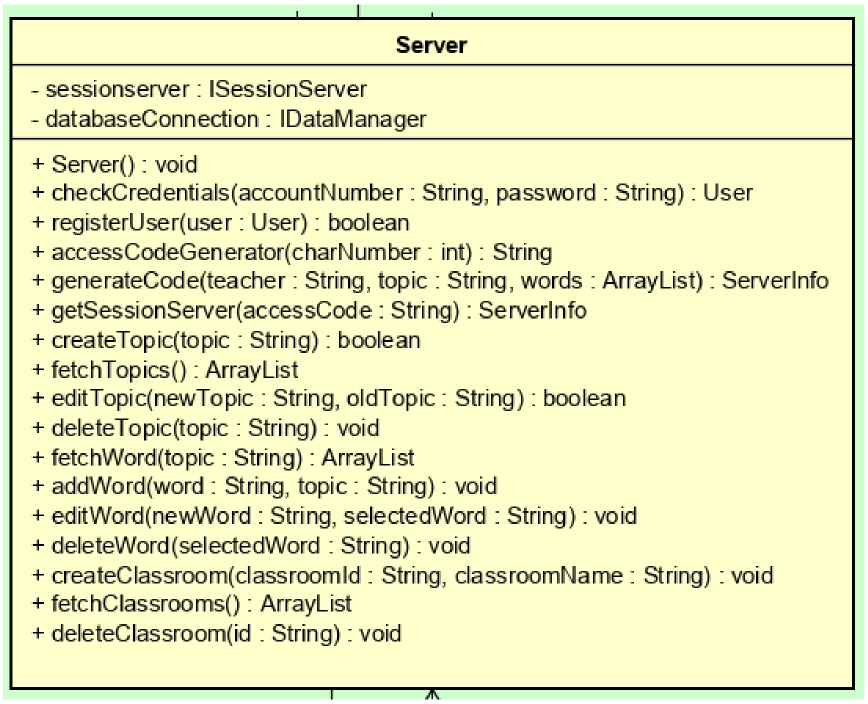
```
Model
- mainServer : IServer
- sessionServer : ISessionServer
- currentUser : User
- accessCode : String
- propertyChangeSupport : PropertyChangeSupport
- callbackObj : ICallBack
+ Model() : void
+ getCredentials(login : String, password : String) : boolean
+ registerUser(firstName : String, lastName : String, eMail : String, accountNumber : String, password : String, isTeacher : boolean) : boolean
+ getUser() : User
+ generateServer(topic : String, words : ArrayList, classroom : String) : String
+ connectToSession(accessCode : String) : void
+ getSessionServerInfo(accessCode : String) : String
+ submitQuestion(inputQuestion : String) : void
+ checkNewQuestion() : void
+ getSessionWords() : ArrayList
+ getSessionTopic() : String
+ addPropertyChangeListener(listener : PropertyChangeListener) : void
+ addPropertyChangeListener(name : String, listener : PropertyChangeListener) : void
+ removePropertyChangeListener(listener : PropertyChangeListener) : void
+ removePropertyChangeListener(name : String, listener : PropertyChangeListener) : void
+ exit() : void
+ selectWord(oldWord : String, newWord : String) : void
+ getSubmittedWords() : ArrayList
+ checkSelectedWords() : void
+ createTopic(Topic : String) : boolean
+ editTopic(newTopic : String, oldTopic : String) : boolean
+ deleteTopic(topic : String) : void
+ fetchTopics() : ArrayList
+ addWord(word : String, topic : String) : void
+ editWord(newWord : String, selectedWord : String) : void
+ deleteWord(selectedWord : String) : void
+ fetchWord(topic : String) : ArrayList
+ checkConnectedStudent() : void
+ notifyStudentOut() : void
+ createClassroom(classroomId : String, classroomName : String) : void
+ operation53() : void
+ fetchClassrooms() : ArrayList
+ deleteClassroom(id : String) : void
+ getClassroom() : String
```

On the other had the fact that the software is structured in such a way that there is a single model instead of splitting into different parts is an issue, but it will be discussed in the *Future Recommendations* chapter.

4.6.2. The Server

The server has a slightly more focused responsibility, meaning that, even if it still responsible for many different things, the main purpose is to provide a communication between the model and the input/output layer as well as creating the session server.

Figure 16 - The Server Class

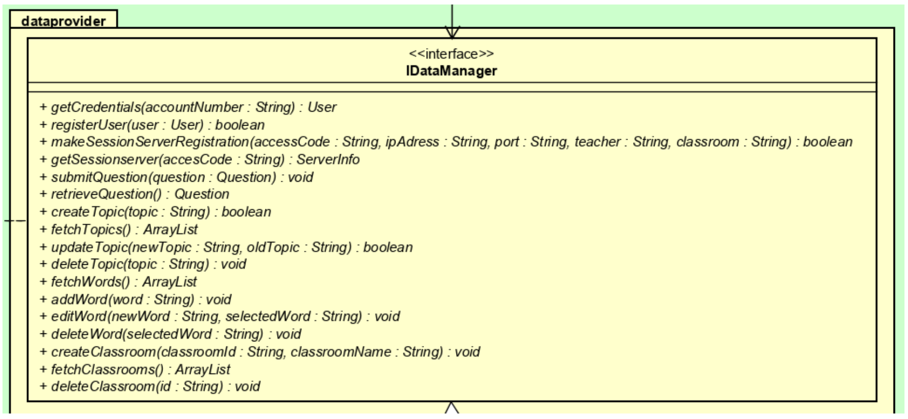


```
Server
- sessionserver : ISessionServer
- databaseConnection : IDataManager
+ Server() : void
+ checkCredentials(accountNumber : String, password : String) : User
+ registerUser(user : User) : boolean
+ accessCodeGenerator(charNumber : int) : String
+ generateCode(teacher : String, topic : String, words : ArrayList) : ServerInfo
+ getSessionServer(accessCode : String) : ServerInfo
+ createTopic(topic : String) : boolean
+ fetchTopics() : ArrayList
+ editTopic(newTopic : String, oldTopic : String) : boolean
+ deleteTopic(topic : String) : void
+ fetchWord(topic : String) : ArrayList
+ addWord(word : String, topic : String) : void
+ editWord(newWord : String, selectedWord : String) : void
+ deleteWord(selectedWord : String) : void
+ createClassroom(classroomId : String, classroomName : String) : void
+ fetchClassrooms() : ArrayList
+ deleteClassroom(id : String) : void
```

4.6.3. The Data Manager

The Data manager is the most outer class that provides the connection and communication between the software and the database through the JDBC Technology. It ensures that the requests coming from the server are successfully implemented and added into the database.

Figure 17 - The Data Manager

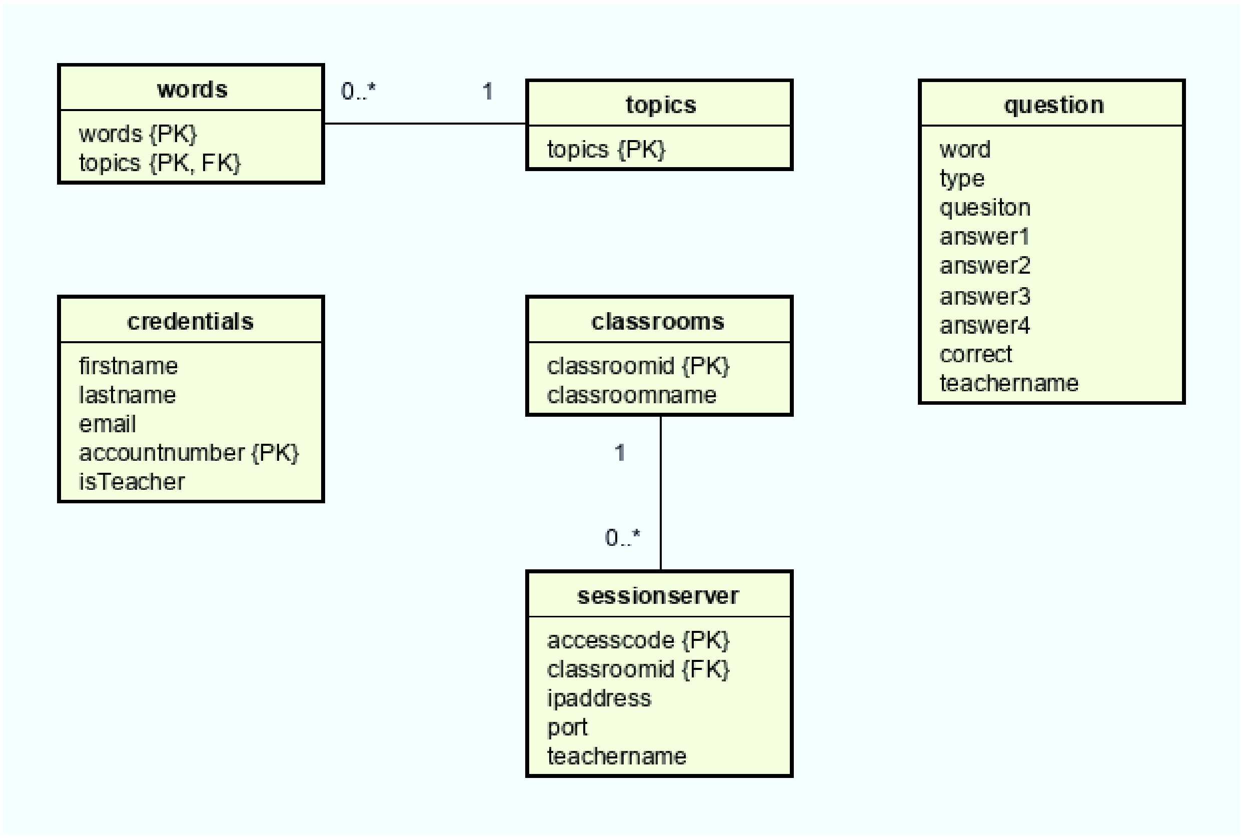


```
dataprovider
<<interface>>
IDataManager
+ getCredentials(accountNumber : String) : User
+ registerUser(user : User) : boolean
+ makeSessionServerRegistration(accessCode : String, ipAdress : String, port : String, teacher : String, classroom : String) : boolean
+ getSessionserver(accesCode : String) : ServerInfo
+ submitQuestion(question : Question) : void
+ retrieveQuestion() : Question
+ createTopic(topic : String) : boolean
+ fetchTopics() : ArrayList
+ updateTopic(newTopic : String, oldTopic : String) : boolean
+ deleteTopic(topic : String) : void
+ fetchWords() : ArrayList
+ addWord(word : String) : void
+ editWord(newWord : String, selectedWord : String) : void
+ deleteWord(selectedWord : String) : void
+ createClassroom(classroomId : String, classroomName : String) : void
+ fetchClassrooms() : ArrayList
+ deleteClassroom(id : String) : void
```

4.7. The Database Design

The database was design based on the needs of the software combined with the elements from the Domain Model.

Figure 18 - Database Design



It can be observed that the database provides only the minimum required functionality for the system as the intention for it was to be used as a *dummy database* until a more detailed one was developed, that will be presented in the *Future Recommendations* chapter. The Current EER diagram can be found in the Appendix J.

5. Implementation

In this chapter the significant parts of the implementation will be described as well as presented. When implementing the design a few consideration were made in regards to the structure and general rules for implementation. The entire source code can be found in the Appendix K and the Java Documentations in the Appendix S.

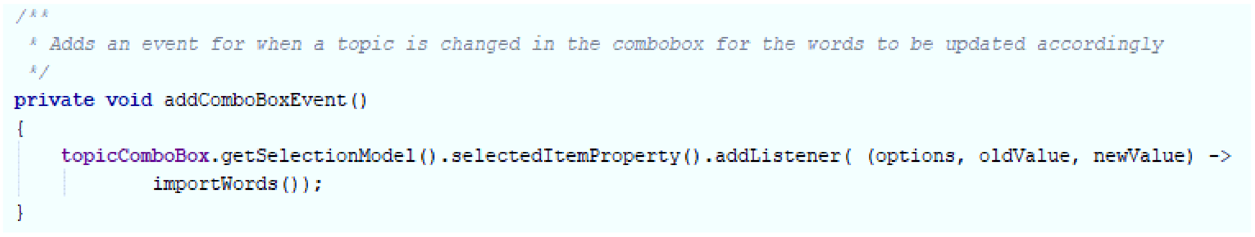
The code was developed using Java language as specified in the non-functional requirements. To ensure a coherence of the code a few naming practices as for example naming the interfaces with an “I” in the front of the name.

On the other hand, when developing the code, even if it was violated in some places, the SOLID principle and DRY rule was kept in mind.

When presenting the implementation it will be structured in the flow of the MVVM pattern meaning the example of code from view, then view model, model, servers, and data managers to keep a logical structure.

An important element present in most of the views are events listeners that were to react in a certain way based on the action made by the user. An example from when a session is created is the combo box event, which sets the list of words displayed to the user based on the selected topic from the drop down menu.

Figure 19 - Combo Box Event Handler

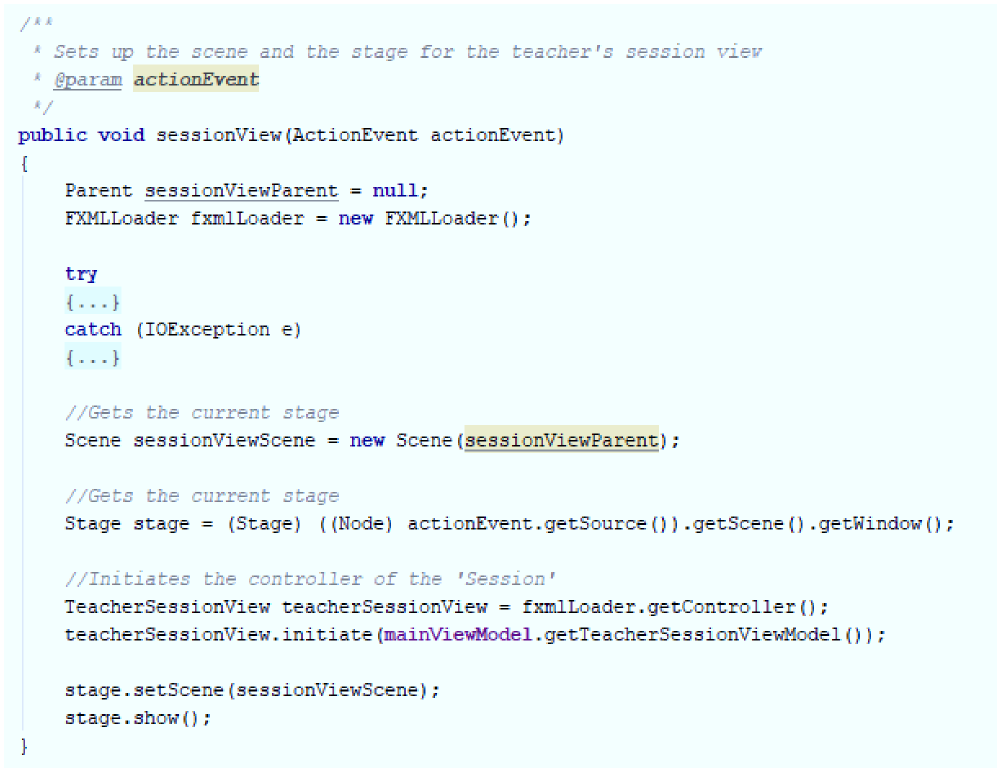


```
/**
 * Adds an event for when a topic is changed in the combobox for the words to be updated accordingly
 */
private void addComboBoxEvent()
{
    topicComboBox.getSelectionModel().selectedItemProperty().addListener( (options, oldValue, newValue) ->
            importWords());
}
```

It was further developed using Lambda Expression which allows for a functional programming in which the function is created and invoked without belonging to any class.

Another point to look at in the View is the Main View which has the responsibility of loading the scenes and stages. An example is the session creation view which is loaded on top of the stage from where the action of invoking it has happened.

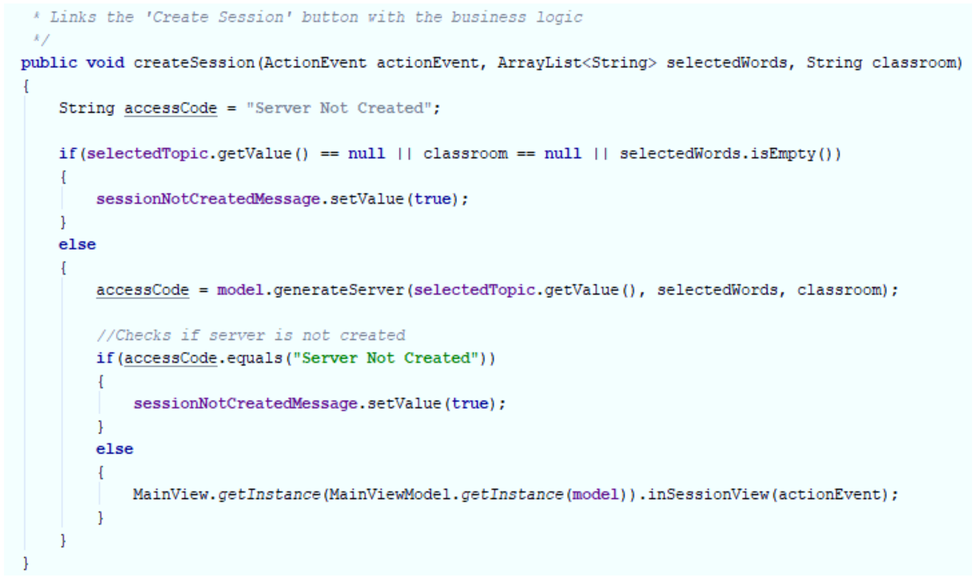
Figure 20 -Scene Loader



```
/**
 * Sets up the scene and the stage for the teacher's session view
 * @param actionEvent
 */
public void sessionView(ActionEvent actionEvent)
{
    Parent sessionViewParent = null;
    FXMLLoader fxmlLoader = new FXMLLoader();

    try
    {...}
    catch (IOException e)
    {...}

    //Gets the current stage
    Scene sessionViewScene = new Scene(sessionViewParent);

    //Gets the current stage
    Stage stage = (Stage) ((Node) actionEvent.getSource()).getScene().getWindow();

    //Initiates the controller of the 'Session'
    TeacherSessionView teacherSessionView = fxmlLoader.getController();
    teacherSessionView.initiate(mainViewModel.getTeacherSessionViewModel());

    stage.setScene(sessionViewScene);
    stage.show();
}
```

When looking at the view model, it was decided that most of the validation (if the system allows it) to be done in here, to ensure the fastest possible way of knowing if something is wrong. For example when a session is created before sending a request to the model to generate a server, the view model checks if any of the basic information (classroom, topic, etc.) is missing

Figure 21 - Session Creation View Model

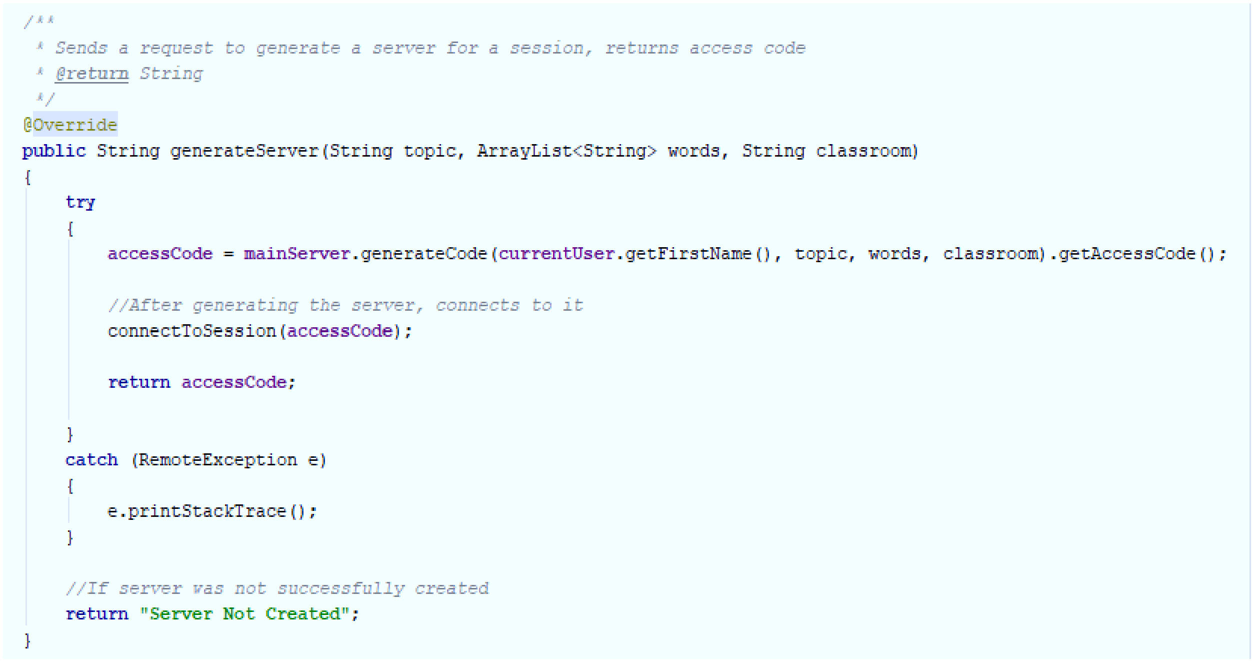


```
 * Links the 'Create Session' button with the business logic
 */
public void createSession(ActionEvent actionEvent, ArrayList<String> selectedWords, String classroom)
{
    String accessCode = "Server Not Created";

    if(selectedTopic.getValue() == null || classroom == null || selectedWords.isEmpty())
    {
        sessionNotCreatedMessage.setValue(true);
    }
    else
    {
        accessCode = model.generateServer(selectedTopic.getValue(), selectedWords, classroom);

        //Checks if server is not created
        if(accessCode.equals("Server Not Created"))
        {
            sessionNotCreatedMessage.setValue(true);
        }
        else
        {
            MainView.getInstance(MainViewModel.getInstance(model)).inSessionView(actionEvent);
        }
    }
}
```

When considering the model, it is to say that it has violated the first SOLID principle and that it will be discussed more in the *Future Recommendations* chapter. Because of that the Model has many responsibilities that can be presented, but to keep a logical structure the creation of a session code will be presented.

In this case the Model does a few important things, it asks the server to generate a server and to store the information that can be identified by an access code in the database, after that it retrieves the access code and connects to the generated server with the access at the same time sending back to the view model the access code.

Figure 22 -Session Creator - Model

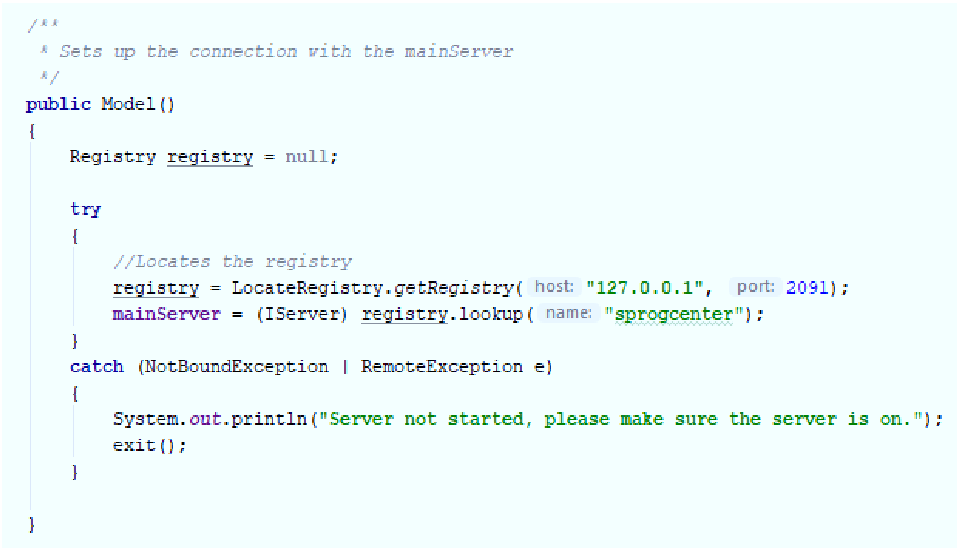


```
/**
 * Sends a request to generate a server for a session, returns access code
 * @return String
 */
@Override
public String generateServer(String topic, ArrayList<String> words, String classroom)
{
    try
    {
        accessCode = mainServer.generateCode(currentUser.getFirstName(), topic, words, classroom).getAccessCode();

        //After generating the server, connects to it
        connectToSession(accessCode);

        return accessCode;

    }
    catch (RemoteException e)
    {
        e.printStackTrace();
    }

    //If server was not successfully created
    return "Server Not Created";
}
```

It can be seen that in case of an issue when connecting to the server the *Server Not Created* will be sent as an access code which will trigger the flag in the view model responsible with the display of the error messages to the user.

Another point to be underlined in the model is the connection with the main server and session server as it is a core function for the system.

Figure 23 - RMI Main Connection

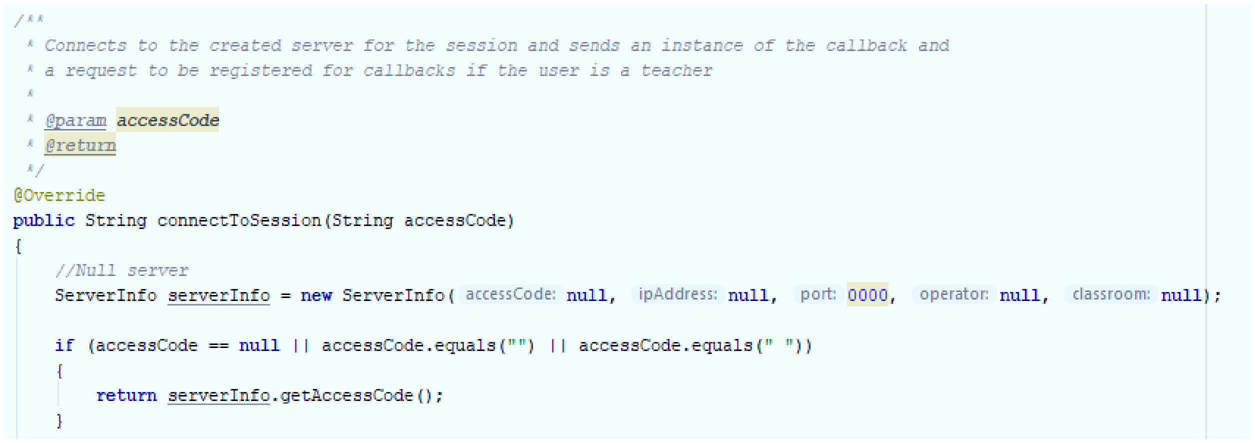


```
/**
 * Sets up the connection with the mainServer
 */
public Model()
{
    Registry registry = null;

    try
    {
        //Locates the registry
        registry = LocateRegistry.getRegistry( host: "127.0.0.1",  port: 2091);
        mainServer = (IServer) registry.lookup( name: "sprogcenter");
    }
    catch (NotBoundException | RemoteException e)
    {
        System.out.println("Server not started, please make sure the server is on.");
        exit();
    }

}
```

It can be seen that in case of error command line message will be displayed before exiting used for debugging. At the same time for the purpose of testing the local address was used and the port 2091 which is commonly used for this kind of applications.

The connection to the session server works a bit different as first a null server is generated for validation purposes and the access code is checked for basic errors in the case of a bad database registration of the server information

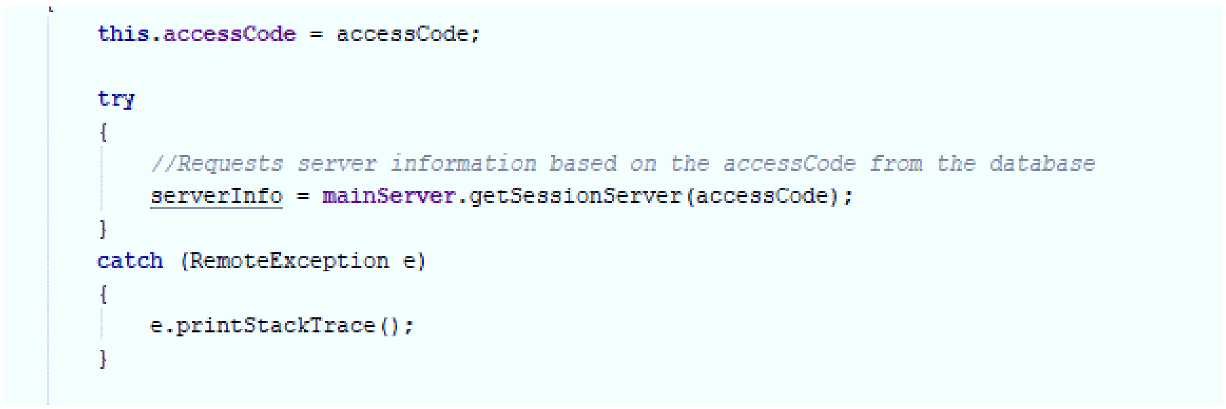
Figure 24 Session Creation - Basic Validation



```
/**
 * Connects to the created server for the session and sends an instance of the callback and
 * a request to be registered for callbacks if the user is a teacher
 *
 * @param accessCode
 * @return
 */
@Override
public String connectToSession(String accessCode)
{
    //Null server
    ServerInfo serverInfo = new ServerInfo( accessCode: null,  ipAddress: null,  port: 0000,  operator: null,  classroom: null);

    if (accessCode == null || accessCode.equals("") || accessCode.equals(" "))
    {
        return serverInfo.getAccessCode();
    }
```

If access code proves to be fine, then the models asks the main server to generate a server information, register it to the database and send back the server information to the model.

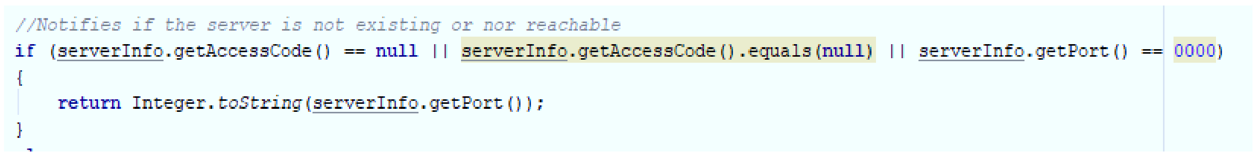
Figure 25 - Session Creation - Server Request



```
    this.accessCode = accessCode;

    try
    {
        //Requests server information based on the accessCode from the database
        serverInfo = mainServer.getSessionServer(accessCode);
    }
    catch (RemoteException e)
    {
        e.printStackTrace();
    }
```

Then another validation of the server information is made to ensure before trying to connect that the server information is not faulty.

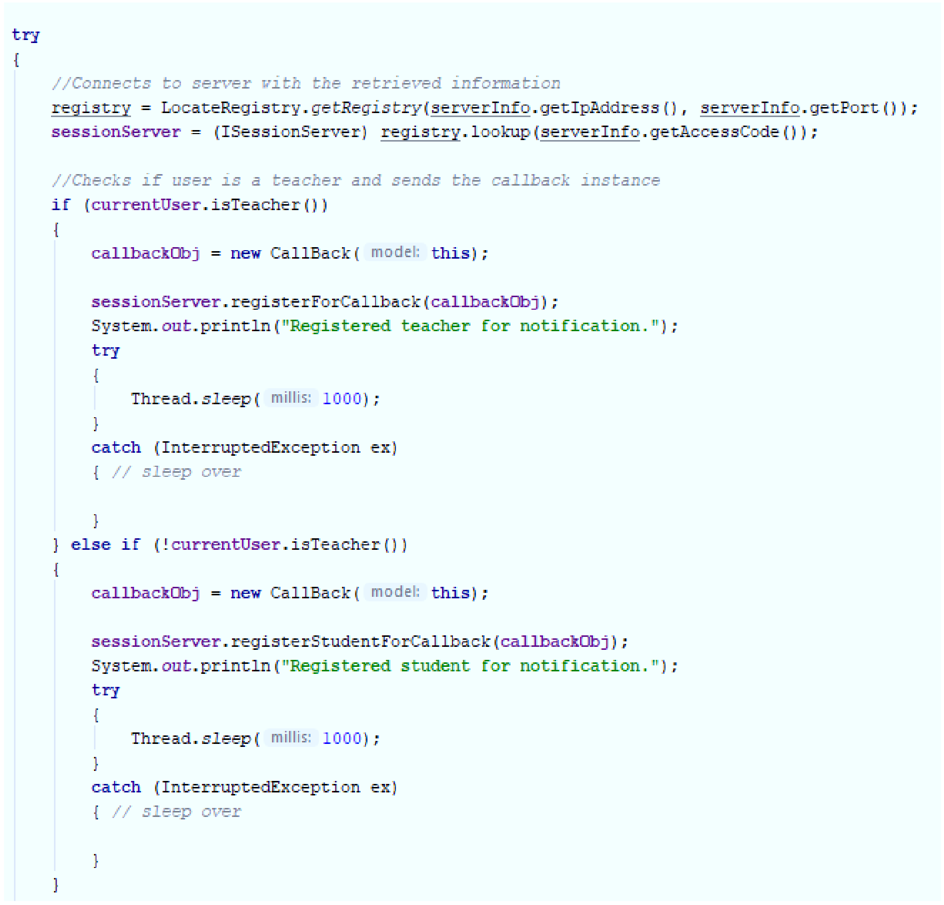
Figure 26 - Server Creation Basic Validation II



```
//Notifies if the server is not existing or nor reachable
if (serverInfo.getAccessCode() == null || serverInfo.getAccessCode().equals(null) || serverInfo.getPort() == 0000)
{
    return Integer.toString(serverInfo.getPort());
}
```

If the server information passes the basic validation the model tries to connect to the session server at the same time sending the callback object to enable to notification functionality of the model.

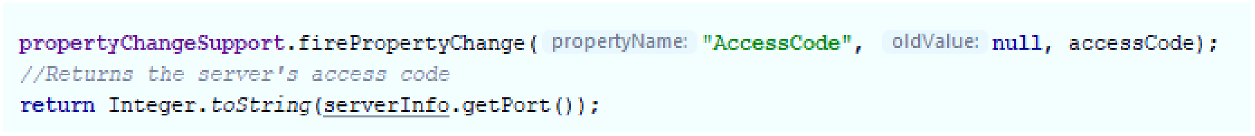
Figure 27 - Session Creation - Connecting to the Session Server



```
try
{
    //Connects to server with the retrieved information
    registry = LocateRegistry.getRegistry(serverInfo.getIpAddress(), serverInfo.getPort());
    sessionServer = (ISessionServer) registry.lookup(serverInfo.getAccessCode());

    //Checks if user is a teacher and sends the callback instance
    if (currentUser.isTeacher())
    {
        callbackObj = new CallBack( model: this);

        sessionServer.registerForCallback(callbackObj);
        System.out.println("Registered teacher for notification.");
        try
        {
            Thread.sleep( millis: 1000);
        }
        catch (InterruptedException ex)
        { // sleep over

        }
    } else if (!currentUser.isTeacher())
    {
        callbackObj = new CallBack( model: this);

        sessionServer.registerStudentForCallback(callbackObj);
        System.out.println("Registered student for notification.");
        try
        {
            Thread.sleep( millis: 1000);
        }
        catch (InterruptedException ex)
        { // sleep over

        }
    }
```

In the end after connecting to the session server a property event is fired from where the teacher's in-session view model can retrieve the access code for further display for the user.

Figure 28 - Property Change Event

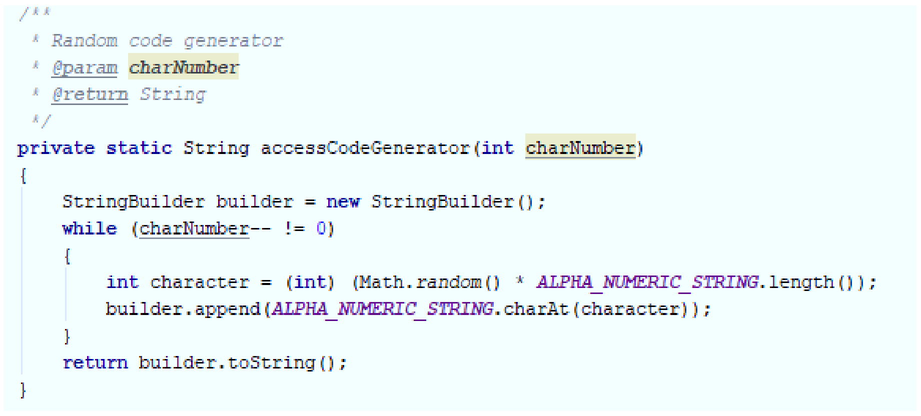


```
propertyChangeSupport.firePropertyChange( propertyName: "AccessCode",  oldValue: null, accessCode);
//Returns the server's access code
return Integer.toString(serverInfo.getPort());
```

For the server two implementations in regards to the same subject (creation of a session) will be presented, the access code generator and the generator of the server information.

It was decided to make the access code generator a static method and it is a simple alpha numeric code generator, the method takes as it inputs the length of the code to be generated.

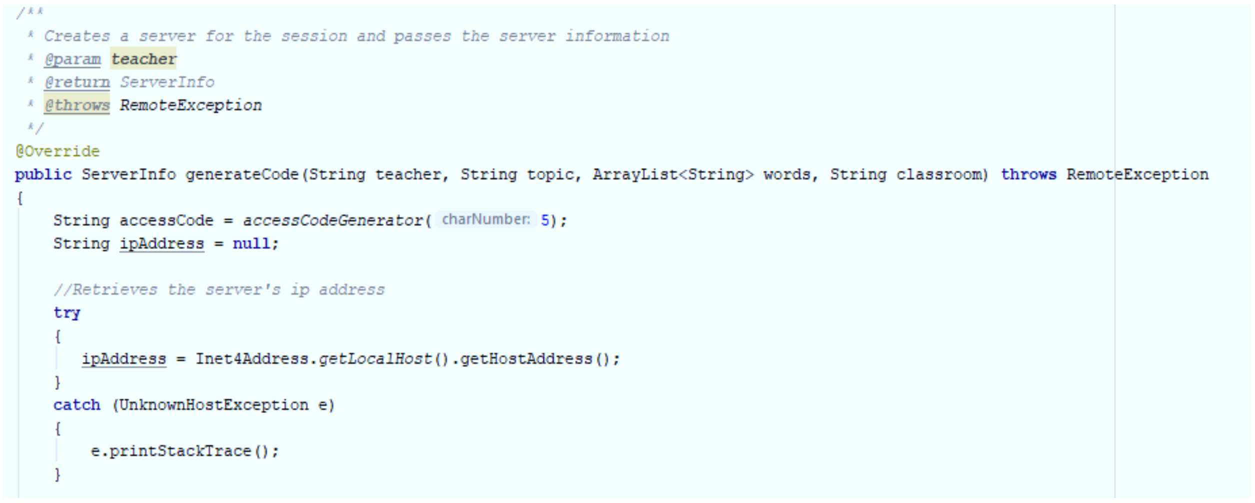
Figure 29 - Random Code Generator



```
/**
 * Random code generator
 * @param charNumber
 * @return String
 */
private static String accessCodeGenerator(int charNumber)
{
    StringBuilder builder = new StringBuilder();
    while (charNumber-- != 0)
    {
        int character = (int) (Math.random() * ALPHA_NUMERIC_STRING.length());
        builder.append(ALPHA_NUMERIC_STRING.charAt(character));
    }
    return builder.toString();
}
```

The session server information generator is a more complex one and it will be split into a few parts. First the access code is generated and the IP address of the future server is retrieved as the host's IP address.

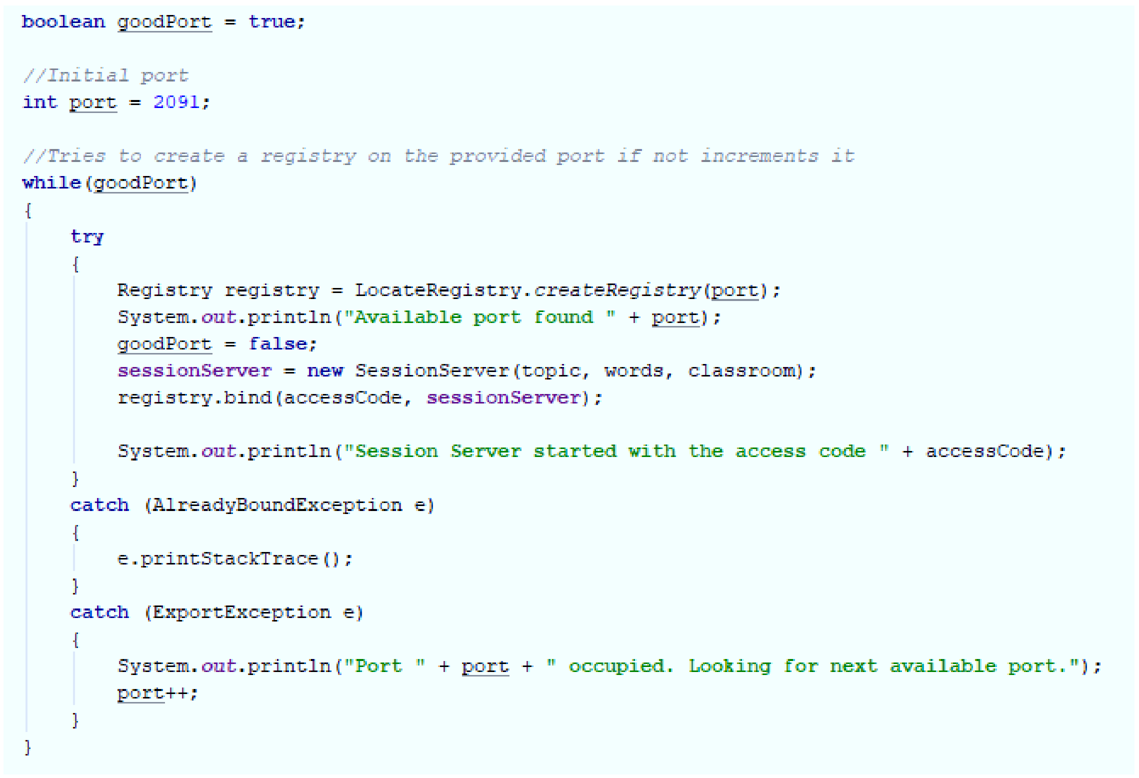
Figure 30 - Server Creator IP and Access Code



```
/**
 * Creates a server for the session and passes the server information
 * @param teacher
 * @return ServerInfo
 * @throws RemoteException
 */
@Override
public ServerInfo generateCode(String teacher, String topic, ArrayList<String> words, String classroom) throws RemoteException
{
    String accessCode = accessCodeGenerator( charNumber: 5);
    String ipAddress = null;

    //Retrieves the server's ip address
    try
    {
       ipAddress = Inet4Address.getLocalHost().getHostAddress();
    }
    catch (UnknownHostException e)
    {
        e.printStackTrace();
    }
```

Then the main server is looking for an available port for the connection.

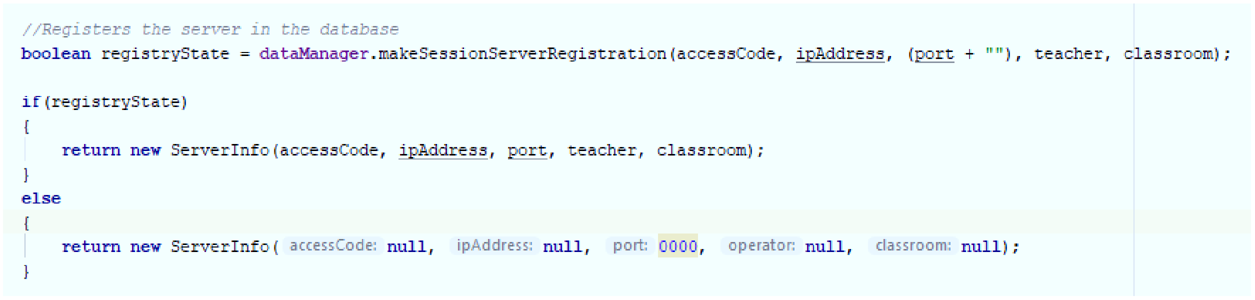
Figure 31 - Available Port



```
boolean goodPort = true;

//Initial port
int port = 2091;

//Tries to create a registry on the provided port if not increments it
while(goodPort)
{
    try
    {
        Registry registry = LocateRegistry.createRegistry(port);
        System.out.println("Available port found " + port);
        goodPort = false;
        sessionServer = new SessionServer(topic, words, classroom);
        registry.bind(accessCode, sessionServer);

        System.out.println("Session Server started with the access code " + accessCode);
    }
    catch (AlreadyBoundException e)
    {
        e.printStackTrace();
    }
    catch (ExportException e)
    {
        System.out.println("Port " + port + " occupied. Looking for next available port.");
        port++;
    }
}
```

When an available port is found it asks the data manager to register information about the server in the database, and if the registration went successfully (based on the response from the data manager) the server information is sent to the model, otherwise a null server is sent to the model.

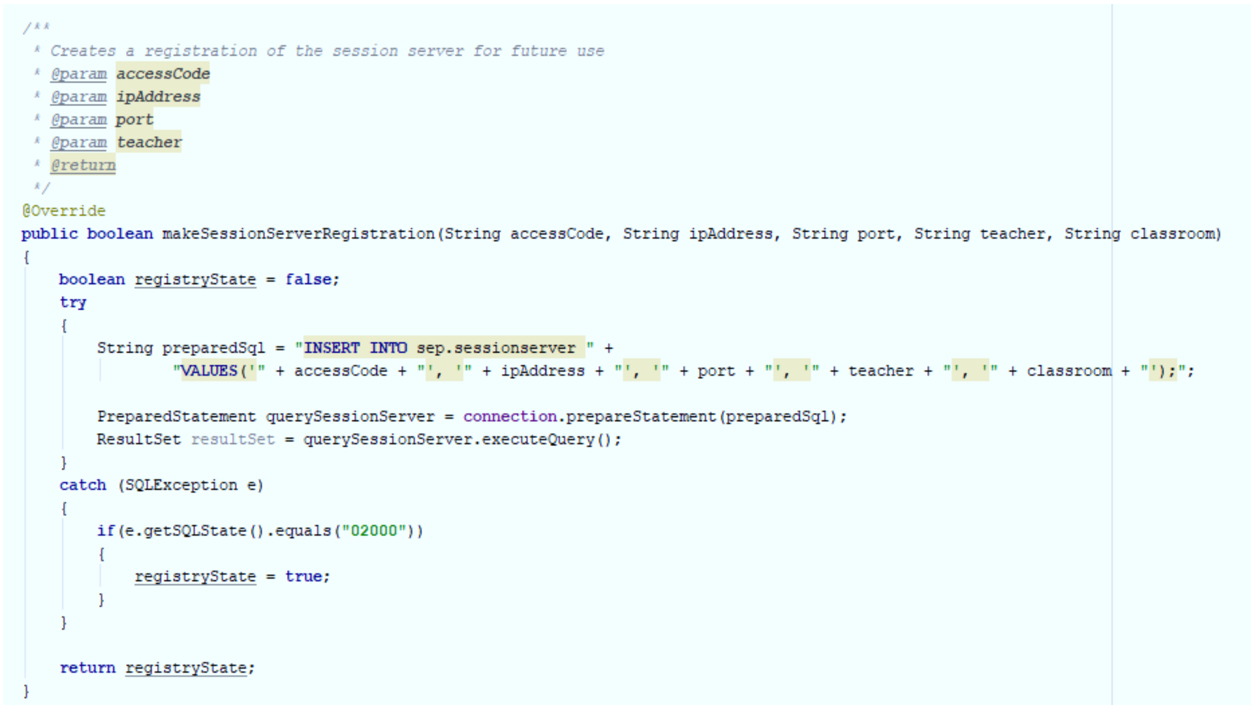
Figure 32 - Server Information



```
//Registers the server in the database
boolean registryState = dataManager.makeSessionServerRegistration(accessCode, ipAddress, (port + ""), teacher, classroom);

if(registryState)
{
    return new ServerInfo(accessCode, ipAddress, port, teacher, classroom);
}
else
{
    return new ServerInfo( accessCode: null,  ipAddress: null,  port: 0000,  operator: null,  classroom: null);
}
```

The last thing to consider is the data manager, and how it registers the server information. With the information provided by the server, the data manager is registering the information in the database.

Figure 33 - Server Registration (Database)



```
/**
 * Creates a registration of the session server for future use
 * @param accessCode
 * @param ipAddress
 * @param port
 * @param teacher
 * @return
 */
@Override
public boolean makeSessionServerRegistration(String accessCode, String ipAddress, String port, String teacher, String classroom)
{
    boolean registryState = false;
    try
    {
        String preparedSql = "INSERT INTO sep.sessionserver " +
                "VALUES('" + accessCode + "', '" + ipAddress + "', '" + port + "', '" + teacher + "', '" + classroom + "');";

        PreparedStatement querySessionServer = connection.prepareStatement(preparedSql);
        ResultSet resultSet = querySessionServer.executeQuery();
    }
    catch (SQLException e)
    {
        if(e.getSQLState().equals("02000"))
        {
            registryState = true;
        }
    }

    return registryState;
}
```

It can be seen that if the SQL exception is 02000 which is *No results were returned by the query.* which is fine in the case of an insertion into the database, the database returns the true value notifying that the server registration was successful.

6. Testing and Discussion

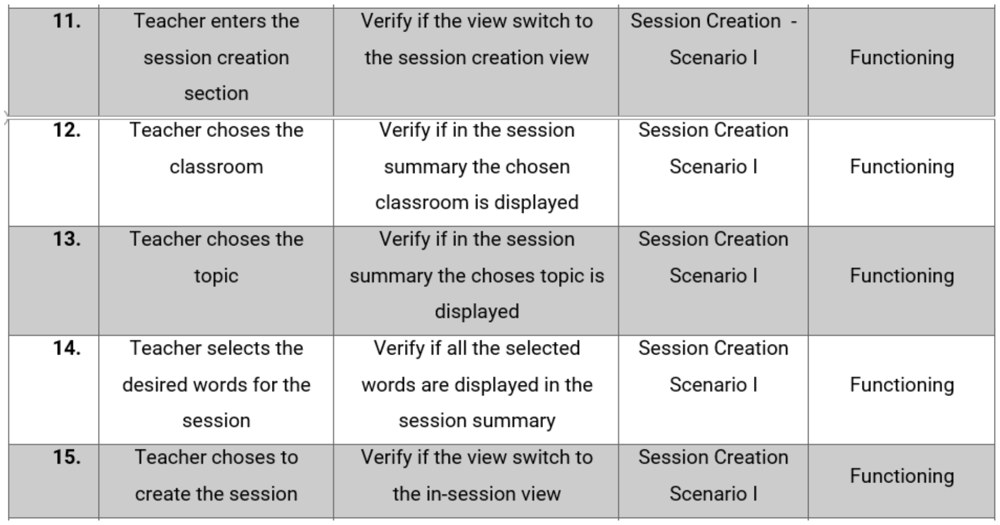
To ensure that the software is operating as intended as well as to identify flaws that might occur only in special cases, the software must be tested.

6.1. Black Box Testing

6.1.1. Test Case Testing

It was decided that the first test would be based on the Test Cases (Software Fundamentals, 2019) developed in the analysis. All the results of the testing can be found in Appendix L.

Figure 34 - Test Case Results



11.	Teacher enters the session creation section	Verify if the view switch to the session creation view	Session Creation - Scenario I	Functioning
12.	Teacher choses the classroom	Verify if in the session summary the chosen classroom is displayed	Session Creation Scenario I	Functioning
13.	Teacher choses the topic	Verify if in the session summary the choses topic is displayed	Session Creation Scenario I	Functioning
14.	Teacher selects the desired words for the session	Verify if all the selected words are displayed in the session summary	Session Creation Scenario I	Functioning
15.	Teacher choses to create the session	Verify if the view switch to the in-session view	Session Creation Scenario I	Functioning

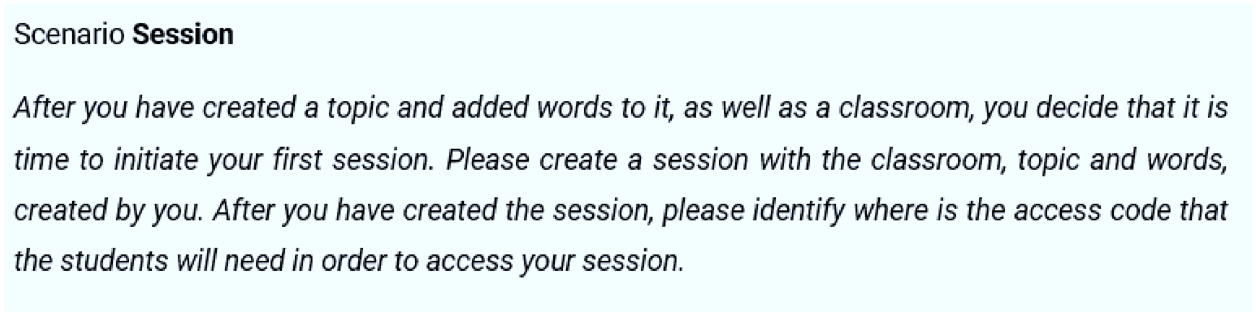
The results of the Test Cases testing are positive, the functionalities that are present in the software are functioning as intended, with no partial results. What this means is that even if there are still many functionalities that are needed to be implemented, the ones that are in the system are finished up, and can be deployed.

6.1.2. Usability Testing

Another type of Black Box testing is the Usability Testing that will focus on how easy and intuitive the GUI is as well as identifying the flaws in the logic and structure of the software.

For the Usability Testing (Usability Gov, 2019)a series of scenarios where developed and a small pool of users (with no software engineering knowledge) where asked to try to complete the proposed scenario.

Figure 35 - Usability Testing - Scenario



Scenario **Session**

After you have created a topic and added words to it, as well as a classroom, you decide that it is time to initiate your first session. Please create a session with the classroom, topic and words, created by you. After you have created the session, please identify where is the access code that the students will need in order to access your session.

It is important to specify that the users were not provided with a user guide up-front or during the testing. The assessment was done on a four level scale as follows:

- Successful – the scenario was achieved with easiness by the user;
- Partially successful – most of the scenario was achieved, but there were difficulties in achieving the scenario;
- Partially not successful – only a small part of the scenario was achieved with struggle;
- Not successful – a insignificant part of the scenario or none was achieved;

All the scenarios can be found in the Appendix M.

6.1.3. Usability Testing Results and Critique

The Usability Testing was provided on a small sample pool (five persons) and provided with valuable information in regards to improving the test process itself as well as improvements for the software. The test was conducted in a moderated environment where a moderator was observing the subjects as well as asking question for taking notes, the moderator did not ask any questions that would have given hints to the subjects. The User Guides can be found in Appendix Q.

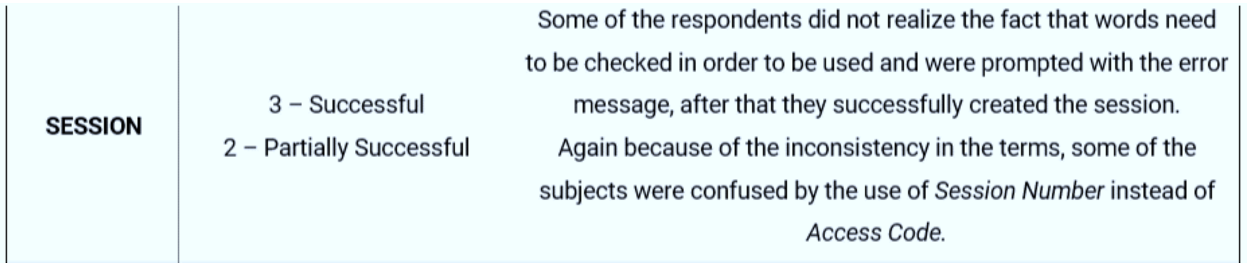
The main critique in regards to the test was the absence of consistency between the terms, for example in some scenarios *Account Number* was used in some *Account Name*. It referred to the same thing, but because it of the differences in their name, 4 out of 5 people were confused and unsure when doing the usability testing.

Some of the subject complained in regards to the depth of details of the scenarios. It was done on purpose to find out how much the subjects can figure out without clear instructions to test the user friendliness of the software.

At the same time some of the subjects complained about the scenario *Add a topic and words, delete topic and words* as being too broad, which was true as the result of the test was assessed on too many parameters which decreased the reliability.

Considering that this was a pre-Alpha release, the results are satisfactory. Providing this test offered room for improvement and discussions for the future of the project.

Figure 36 - Usability Testing Results



SESSION	3 – Successful 2 – Partially Successful	Some of the respondents did not realize the fact that words need to be checked in order to be used and were prompted with the error message, after that they successfully created the session. Again because of the inconsistency in the terms, some of the subjects were confused by the use of *Session Number* instead of *Access Code*.

The example above shows that consistency in the terms is crucial when designing a GUI and that it is important to have explanatory error messages that would guide the users when needed. All the results can be found in Appendix N.

6.2. White Box Testing

It was decided to not rely on a formal way of White Box testing method (e.g. JUnit) as due to the nature of the software, only a very small part of the system can be tested and the results will not be consistent and relevant.

7. Conclusion

This chapter will try to conclude on the current development of the software as well as how much of the problem formulation described in the Project Description (Appendix A) was answered.

How will the involved parties (students & teachers) be enabled to communicate and collaborate concurrently during the creation and usage of a session/quiz?

The communication between the involved parties is done through the use of a client – server connection with Graphical User Interface for the client side. The network connection was established using Remote Method Invocation, which allowed multiple user inputs and activities that have live results from one client to another.

How should access to different types of information be granted to the involved parties?

The access to different types of information is done based on a login functionality (that involves a login and password) that splits access to two different parties student/teacher based on the user type.

What relevant information in regards to the students, teachers and learning process needs to be included and stored?

The relevant information needed for the system was identified in the *Requirements* and *Analysis* chapters where the most communication with the product owner was done. The best representation of the information is presented in the Domain Model (Appendix F) that encompasses most of what is used in the system from this perspective.

How should the above specified information be stored and accessed by the different parties?

The information specified above is stored using a database developed using PostgreSQL technology, because of this a fine similarity between the information contained in the Domain Model Diagram and EER Diagram can be observed, as the database was specifically designed to serve the needs of the system.

What are the features of the minimum viable product that will satisfy the customer's needs?

This problem was not solved, as a common definition for the minimum viable product was not developed. Therefore it was decide to consider the minimum viable product a solution that will satisfy all the features described in the *Requirements* chapter through the User Stories.

What characteristics should be implemented that will make the product easy to use and maintain?

To answer this problem through the analysis, design as well as implementation an eye was kept on the requirements described in the Jakob Nilesen's article (Nielsen, 1994) as well as the use of Google Material Design principles by using JFoenix library for the GUI. Whereas when thinking about the second part of the problem (maintainability) SOLID principles were kept in mind as well as implementation principles in regards to rigidity, fragility and mobility of the system.

To conclude this project a few things are needed to be pointed out in regards to the entire development.

First the requirements were defined in form of user stories, which provided the necessary information to developed Use Cases, Use Case Description, Activity Diagram and System Sequence Diagram. Based on the Use Case descriptions the Test Cases were made for the future use.

With the specified above information the Domain Model was developed that served as a launching pad for the design of the software.

In the design for the most complex process a Sequence Diagram, but not before a class diagram was made with keeping in mind design patterns that provided benefits for the system.

The database was put in place to provide minimum functionality for the system with need for an improved database.

In the testing section the system was tested using the test cases developed in the Analysis as well as a Usability Test for the UI of the system.

Overall the system does not provide functionality for all the requirements described in the beginning, whereas the features that are implemented are functioning as intended.

8. Future Recommendations

When considering the future development of the system, there are some elements that are worth mentioning to ensure a starting point for improvements and development.

First of all, it is required a thorough check of the violation of the SOLID principles. A starting point can be the Model of the software. Further improvements are needed to decrease the rigidity and fragility of the system as well as increasing the mobility of it. It can be done by reducing the dependability between the parts of the system as well as creating a more lose coupled overall system.

The database needs to be replaced with an improved one, as a starting point the following EER Diagram can be considered as a proper database design for the current existing functionalities.

The following gives a brief outline of how the EER diagram for the database was designed, even though it was not fully implemented in the project.

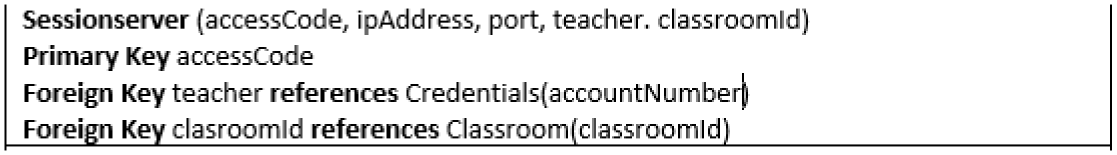
Using the Domain Model as a basic conceptual ER diagram this then needed to be mapped to a Logical/Global data model (EER), ready for implementation. This process involved several steps as outlined below;

1. Step 1 is to create a relational schema by deriving all the relations. This includes strong and weak entity types, 1:1, 1:* & *:* relationship types, along with sub and super classes and finally multivalued attributes.
2. The next step was to validate the derived relations through the technique of Normalization. This involves three steps, or 'forms':
   - First Normal Form requires that there are no repeating elements or groups of elements (atomicity).
   - Second Normal form required that there are no partial dependencies on a concatenated key.
   - Third normal Form required ensuring that there are no dependencies on Non-Key attributes

3. Step 3 was to check whether the integrity constraints were represented in the logical model, with the predominant focus upon referential and entity integrity to ensure the consistency and accuracy of the data

With these steps implemented it is then possible to create a relations overview and EER diagram as pictured below;

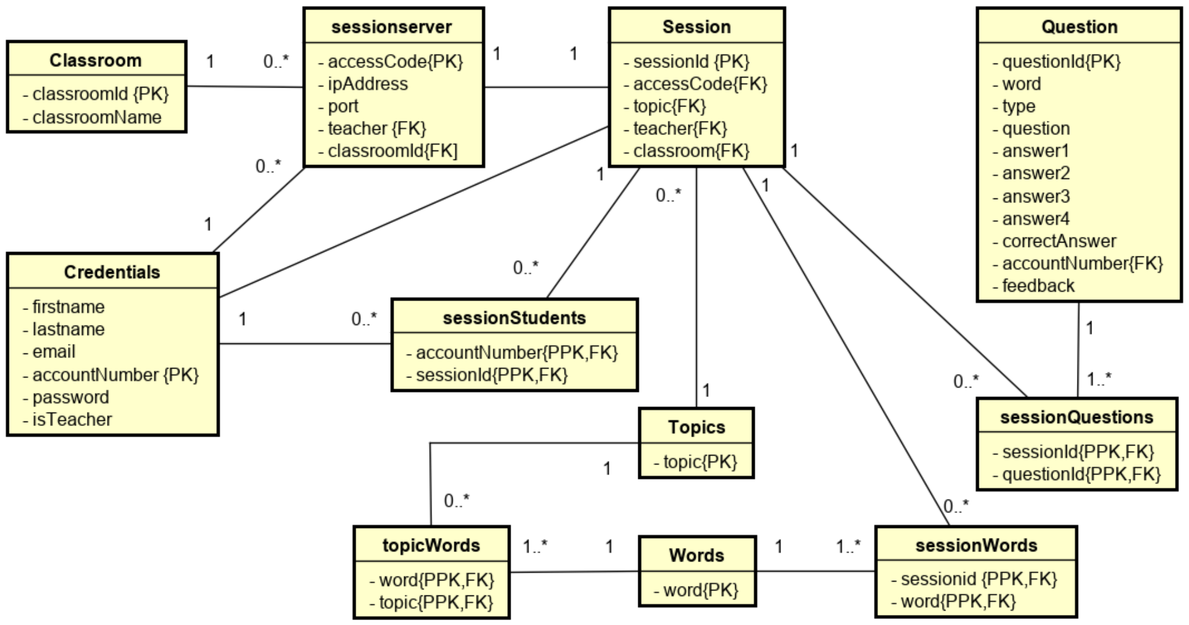
Figure 37 - Relationship Overview



Sessionserver (accessCode, ipAddress, port, teacher. classroomId)
Primary Key accessCode
Foreign Key teacher **references** Credentials(accountNumber)
Foreign Key clasroomId **references** Classroom(classroomId)

The entire Relationship Overview can be seen in the Appendix O. Next will be presented the EER improved diagram, it can be found in the Appendix P.

Figure 38 - EER Diagram - Improved



The User Interfaces needs improvements as well, and as first suggestions the future developers can consider the results of the usability testing (Appendix N). The results provide clear comments of what are the main issues with the current UI.

9. Sources

Freeman, S. et al., 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences of the United States of America,* pp. 8410 - 8415.

GeeksForGeeks, 2019. *GeeksForGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/java-singleton-design-pattern-practices-examples/
[Accessed 2019].

Holmen, A. & Lund, K., 1999. *Studier i dansk som andetsprog.* Copenhagen: Akademisk Forlag.

JFoenix, 2019. *JFoenix.* [Online]
Available at: http://www.jfoenix.com/documentation.html
[Accessed 2019].

Larman, C., 2005. *Applying UML and Patterns.* 3rd ed. s.l.:John Wait.

Larsen, D., 2019. *Immigration and Their Descendents.* [Online]
Available at: https://www.dst.dk/en/Statistik/emner/befolkning-og-valg/indvandrere-og-efterkommere/indvandrere-og-efterkommere

Likness, J., 2014. *Wintellect.* [Online]
Available at: https://www.wintellect.com/model-view-viewmodel-mvvm-explained/
[Accessed 2019].

Liu, M.-L., 2003. RMI Client CallBack. In: *Distributed Computing.* s.l.:Addison Wesley, p. Chapter 8.

Nielsen, J., 1994. *10 Usability Heuristics for User Interface Design.* [Online]
Available at: https://www.nngroup.com/articles/ten-usability-heuristics/

Oracle , 2019. *Oracle Java Documentation.* [Online]
Available at: https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/hello/hello-world.html
[Accessed 2019].

Oracle, 2019. *Oracle Documentation.* [Online]
Available at: https://www.oracle.com/technetwork/java/javase/jdbc/index.html
[Accessed 2019].

PostreSQL, 2019. *PostgreSQL JDBC Driver.* [Online]
Available at: https://jdbc.postgresql.org/documentation/documentation.html
[Accessed 2019].

Software Fundamentals, 2019. *Software Fundamentals.* [Online]
Available at: http://softwaretestingfundamentals.com/black-box-testing/
[Accessed 2019].

SourceMaking, 2019. *SourceMaking - Observer Pattern.* [Online]
Available at: https://sourcemaking.com/design_patterns/observer
[Accessed 2019].

Sprogcenter Midt, 2019. *Om Sprogcenter Midt.* [Online]
Available at: https://www.sprogcentermidt.dk/hvem-er-vi/om-sprogcentret/

TeachThought Staff, 2018. *The Definition Of Blended Learning.* [Online]
Available at: https://www.teachthought.com/learning/the-definition-of-blended-learning

Usability Gov, 2019. *Usability Gov.* [Online]
Available at: https://www.usability.gov/how-to-and-tools/methods/usability-testing.html
[Accessed 2019].

Vygotsky, L., 1978. *Mind in Scoiety.* Cambridge: Harvard University Press.

Appendix