Course Quizard
Providing students additional ways to study

Daniel Hellgren
Simon Ljus
Tony Wang

Civilingenjörprogrammet i informationsteknologi
Computer and Information Engineering Programme
Abstract

Course Quizard
Providing students additional ways to study

Daniel Hellgren
Simon Ljus
Tony Wang

Studies shows that students today have problems learning. We have developed an application to combat these issues. We call it Course Quizard and it targets university students that want to find an supplemental way to study. Users can play course specific quizzes against each other containing questions created by other users or automatic generated questions from course material. A small group of students tested the application and they feel the idea is valuable but would rather recommend it to a friend than play it themselves. Course Quizard has potential with future development which would make the application an excellent study alternative.
Sammanfattning

Acknowledgements

Thanks to BorrowIT for sharing the server with us. Thanks to everyone who have reviewed our work and tested our application.
## Contents

1 **Introduction**  
2 **Background**  
   2.1 Lack of Student Learning and Approaches to Improve Learning  
   2.2 Education Through Games and Quizzes  
   2.3 Natural Language Processing  
3 **Purpose, Aims and Motivation**  
4 **Delimitations**  
5 **Related Work**  
6 **Method**  
7 **System Structure**  
   7.1 Front End  
   7.2 Back End  
8 **Implementation of Course Quizard**  
   8.1 Front End Implementation  
   8.2 Back End Implementation  
      8.2.1 Database Implementation  
      8.2.2 PHP Scripts and MySQL Queries  
      8.2.3 Question Generation  
9 **Requirements**
10 Evaluation

11 Results
   11.1 Think Aloud Results .................................................. 17
   11.2 Survey Results .......................................................... 18

12 Discussion
   12.1 Different approaches ................................................... 20
   12.2 Relating to .............................................................. 20

13 Conclusions

14 Future Work

A User Guide
   A.1 Application Functions .................................................. 26
   A.2 Logging in and registering ............................................. 27
   A.3 Home View .............................................................. 27
   A.4 New Game ............................................................... 28
   A.5 Create Question and Generate Questions From ...................... 29
   A.6 My Games and Playing Games ....................................... 30
   A.7 My Profile ............................................................. 31

B Course Quizard Survey and results ................................................. 34

C Think Aloud Evaluations .......................................................... 35
1 Introduction

From first hand experience we can recognize that some university students are not learning and reaching the goals of the courses that they are enrolled in. This is certainly problematic and the cause could be poor learning strategies. The idea of this project comes from the problems of learning. Our approach on this problem is a quiz application including automatic question generation which is intended to give university students an additional learning tool.

The application includes automatic question generation. Using the rules of human language we can implement an algorithm which generates questions. The motivation for automatic question generation is for university students to insert questions from their own courses to practice on.

Note: In this report, the term “student” means university student if nothing else is mentioned.

2 Background

Before continuing we need to establish the background of the project. In Section 2.1, the problems of students not learning enough and approaches to solve the problem are covered. In Section 2.2, the educational benefits through games and quizzes are covered. And finally in Section 2.3, we cover the uses of natural language processing to make the computer manipulate human language, which in this case can be used for question generation.

2.1 Lack of Student Learning and Approaches to Improve Learning

It appears that students today are having quandaries grasping their course material. This could be a problem for the future if students are not educating themselves well enough. A report from CBS [25] brings forth the result of a study showing that only 45 percent out of 2300 undergraduates showed any success in learning crucial skills such as complex reasoning and critical thinking by the end of their second year. Additionally Darling-Hammond [34] acknowledges in an interview that students today are not learning, she says if they applied and solved problems using the material read then students would leave school with an improved education.
Rao and Dicarlo [32] theorize that students do not learn just by sitting in class. They claim that in order to learn, students have to solve problems by applying the course material in real life situations. They also mention that students prefer active learning strategies instead of traditional lectures.

### 2.2 Education Through Games and Quizzes

Woit and Mason [35] investigated students learning through quizzes. During their investigation, they made voluntary quizzes for computer science students. The contents of the quizzes were questions based on previous laboratory sessions. The conclusion of the article states that the quizzes were overall valuable to the students.

A study by Griffiths [27] concludes that there are educational benefits to video games and that games constructed to teach a certain skill usually find success. Furthermore he suggests that video games that are fun and stimulating builds competition and attracts participants.

### 2.3 Natural Language Processing

Human language, called natural language, cannot inherently be processed by a computer [26]. In order for a computer to process the information in any text or speech it needs to be translated into something it can understand, which is machine language. Natural language is complex and there are numerous of ways of expressing oneself. The science and art of translating Natural language into machine language is called natural language processing, NLP. NLP is useful for handling text and analyzing its contents. By implementing NLP software, we can produce questions from a piece of text.

Automatic question generation from text was studied by Michael Heilman [29]. He managed to form questions based on a given sentence. His research supports the idea of using NLP to automatically generate questions which can be used for studying. To be able to understand a sentence and transform it into a question requires a lot of knowledge in linguistics. We could instead take inspiration from his work and use NLP in another way that still would result in questions that were generated from text. A strong motivation for generating questions from text is for the students to create their own questions from their own course material to practice on.
3 Purpose, Aims and Motivation

Purpose  Our aspiration with this project is to improve the quality of the students’ studies. In order to give students supplements to their studies, we want students to have a tool to recapitulate or solve questions based on their course material. And with the final product of this project we strive to motivate more students to learn from their course material.

Aims  The aim of the project is to develop an Android application featuring a quiz game where the users can contribute to the question pool for the quizzes; either by auto-generation from literature text or by constructing them manually. The users can choose to compete against an opponent or play alone in a quiz. The quizzes are meant to be course specific, thus the questions in a quiz ought to relate to a specific course. The users are able to cast their votes on the quality of the questions and its relevancy for the course.

Motivation  With our application, the students may have a more enjoyable way to apprehend the content of their course material. Section 2.1 mentions that students are falling behind in education and this could lead to an intellectual decline. A solution would be to help students learn by using active learning strategies which would improve students learning capabilities. With Course Quizard, students can apply the knowledge from the course material on competitive quizzes on their mobile devices. Our application finds its uniqueness within its intention to specify the quizzes to real courses and its ability to automatically generate questions from text.

4 Delimitations

Platform and platform version  The application is implemented on a single chosen platform, Android. Android phones and tablets holds, as of March-April 2016, 64% [20] of the worldwide phone and tablet market share. This gives us good reason to implement the application on Android to provide the application for as many users as possible. The application supports all Android phones and tablets with Android version 4.0 or higher, covering more than 97% [23] of the recently used Android devices.

Question generator  Generating proper questions, as in rephrasing sentences into questions, is in itself an extensive project that requires a lot of knowledge in linguistics.
The question generator is limited to generate “fill in the blank”-questions from text. Initially, the application is implemented in English therefore the texts entered from the users must be in English so the questions can be generated properly. Mostly because of different languages are structuring sentences differently.

**Server** We can not ensure that our application can handle a severe amount of data because of the limits of the machine that is our server. If the application would be released publicly, the server have to be able to handle a large amount of interactions simultaneously. There is no focus on keeping the system completely safe from intrusion and other malicious intents. We strive to implement a baseline defense but it will most likely not be secure enough to be released publicly. The password data is the most private data we store and passwords is very important to keep secured since the same username and password may be used on other sites or applications.

5  **Related Work**

Since our biggest focus is releasing a quiz game as an application and the goal for the application is finding an alternative way to study we have tried to find learning applications in quiz format on the market. We want to investigate existing quiz applications available, not different learning strategies as for example a simpler way to start a group discussion, new way to find study groups, or listen to the material in your phone while doing something else. By looking at quiz like tools we can see if a user can receive the same experience and type of learning elsewhere than our application. The goal of our application is giving students a supplementary way to study, not to replace the existing study methods they are already using.

There exists similar quiz applications that could be used as a learning tool however they miss important aspects and functions that Course Quizard includes. The paragraphs in this section presents related work and discusses the importance of Course Quizard. We want to look into other quiz applications that have different purposes. We explore applications that are seemingly fun and exciting to play and attract a large user base. This because our application will rely on a player base to add and vote on questions, therefore we want to attract players. Additionally applications that are for the purpose of educating the user are explored in order to examine the different approaches of creating an educational application. The first two related works presented are popular applications that have a large user-base and the next three are apps which purpose is to educate the user.
Related Work

**Quizkampen [9]** Sweden’s (self-proclaimed) largest quiz application where two players oppose one another in multiple choice questions with various themes. The rounds with its own theme have three questions each and the one who gets the most correct answers out of six rounds wins the game. Course Quizard focuses on specific university courses which Quizkampen does not. The game concept is interesting to us because it has attracted many users [10]. Since Quizkampen is purely for entertainment we only draw inspiration from Quizkampen in the aspects of gameplay design.

**QuizUp [11]** The world’s (self-proclaimed) largest quiz application where users can create their own topics and contribute with questions to the topics. Users can play quizzes of seven questions in existing topics against each other or in single-player mode. QuizUp does not include automated question generation which Course Quizard will provide.

**Khan Academy [5]** An educational platform where users watch videos specific to a subject and play quizzes based on the content in the videos. Khan Academy is useful for learning specific subjects such as complex numbers. The learning material is more general and may not cover a student’s own course material. Course Quizard provides users with the options to create questions based on their own study material.

**King of Math [6]** A Swedish math game in single-player mode where users can play math quizzes. The questions are constructed so that users can choose the right alternative. Each quiz is based on specific math subjects and users can only play existing quizzes. The difference from King of Math is that Course Quizard provides the users with the option to compete against each other.

**Duolingo [2]** A language learning application where users can play quizzes to learn new languages. Users can learn grammar and words etc. by playing different games. Duolingo’s target is only certain languages, we want to target courses which students are enrolled in.
6 Method

The application is developed using Android Studio [1], which is the official integrated development environment for Android. Android Studio provides a debugger, an Android emulator, a code editor and a visual editor for the Android application layout. Java [3] is the main language for Android app development, which is used for writing the functionalities of the application. XML [13] is the main language for making the graphical design in Android applications. The visual editor simplifies the work of designing and reviewing the current layout. The drag and drop features of the visual editor are used when creating a preliminary design. Fine adjustments of the layout can preferably be done by editing the XML code.

MySQL [7], a relational database management system, is utilized for our database. It provides the possibility of storing information required for the application such as user information and questions for the quizzes. Construction and administrative work on the database is handled with MySQL Workbench [8]. The workbench is mainly used for creating the tables that is accessed and manipulated by the application. The tables are populated with data through the application apart from examples inserted upon their creation in the workbench.

PHP [16], a server-side scripting language designed for web development, serves as the connection between the application and the server. It is also used as a general purpose programming language. By using HTTP-methods, data can be fetched and pushed to the server.

An existing library called Natural Language Toolkit (NLTK) [14] is used to process the natural language for the automatic question generation algorithm. The library is a free open source project that provides a plethora of useful functions for our NLP needs. Among many things, it can accurately identify the class of the words that appear in a sentence (verbs, nouns etc.). However it has no built in question generation. The main NLTK functions used are tokenizing and Part of Speech tagging which is further explained in section 8.2.3. NLTK supports Python version 3.4 [31], therefore we use Python 3.4 to implement the question generation algorithm.
7 System Structure

The structure of our system can be abstracted to two main components:

1. The front end, which is the application interface.
2. The back end, where we have our database and run external programs.

(An illustration of the system structure can be seen in Figure 1 below.)

To communicate between the two ends HTTP-protocols are implemented. The front end is communicating with a the back end through PHP-scripts. The processed data is echoed by the scripts so that they can be fetched by the front end.

![Figure 1: Overview of the system structure](image)

7.1 Front End

The mobile application consists of what, in Android developing terms, is called views. A view is essentially what the user sees on the screen. When the user is redirected in the application another view is shown. From the different views the user can engage in the application through pressing buttons and writing text which may take the user to different views. The question generator is executed and the database is modified by the PHP-scripts triggered from the front end users when different actions are taken.
7.2 Back End

As can be observed in Figure 1, the back end of the system consists of two additional parts; The database and the question generator.

**Database**  The database is hosted on the server implementing WampServer [12] and MySQL Workbench [8]. The database is used to store the information needed for the application. All of the users of the front end have the same privileges on the database. Administrative work on the database has to be carried out locally on the server as with the front end one can only manipulate the database through the preordained scripts.

**Question generator**  The Question generator is a Python program that requires a string as a running argument. This string is sent by the front end through a PHP-script. Read the details of the question generation and database implementation in Sections 8.2.1 - 8.2.3.

8 Implementation of Course Quizard

This section presents an outline of the implementation of Course Quizard. The front end implementation of the system is described in Section 8.1 and the back end implementation is described in Section 8.2. Upon interest an extensive user guide of Course Quizard can be found in Appendix A.

8.1 Front End Implementation

The front end is implemented with activities and fragments [21] which are Java class files and XML files. Each activity is connected to at least one XML file and handles a single view. Using fragments which can be described as a portion or a behavior of a user interface, the activity can consist of multiple views, for example the tabs in the application are created using fragments.
Storing and managing data  The class Intent [22] is mostly used for starting new activities, you can think of it as a basic data structure which holds an abstract description of an action to be performed. Through intents data can be passed from the current activity to the next activity. Our primarily passed data type are string arrays which is passed between activities, the string arrays serve as raw data to later be arranged into specific data types using our parser functions. For example there could be a string of friends stored as raw data which is later parsed into a friends data type which can be used to run different functions. Additionally the string array also contains information about what action the previous activity took, this gives the receiving activity information on how to perform according to the previous activities actions.

Shared Preferences [24] is a class used to store data in key-value pairs it is used to access data globally by several activities or data stored locally on the device. The data saved by shared preferences is deleted when the user signs out from the application. The user’s university, friend list and login details can easily be accessed without having to pass the values through intents by using shared preferences. Generating questions also uses shared preferences, all questions generated are fetched from the server at once and the questions are then stored in shared preferences so we can show the generated questions one question at a time without having to either send the questions through intents or fetch each question separately from the server.

Communication with the back end  Client-server functions are implemented with the AsyncTask [15] class which is used to implement parallelism in Android. This class runs in the background to avoid having the user disturbed by client-server functions. For example when adding a user or adding a question to the database an AsyncTask object is created to communicate with the server and perform the specified functions which performs PHP invocations to manipulate database.

Fetching back end data  The front end invokes PHP-scripts to fetch data from the database. The PHP-script echoes a formatted string which represents the data. A formatted string contains the data from the database encoded in a specific way for the front end to parse. An example of a formatted string would be: 
'value1%Q%value2%Q%...%valuen%N%'  Each row is separated by %N% and each value is separated by %Q%. The formatted string is then parsed into java objects. This implementation results in a crash if the users inputs %N% or %Q% because the front end parses a fixed number of %N% or %Q%. One or more %Q% and one or more %N% and the program crashes. The problem will occur when the user inputs a %Q% or %N% in the user-name, password, question or any other possible input. For example if the user in-
puts a user-name containing \%Q\%, fetching this user-name will result in crash because an unexpected amount of \%Q\% exists in the string that is fetched from the database. Clearly this implementation is not superb if the user can crash the application. However it is sufficient to make the application work properly. A different approach would be to use JSON (JavaScript Object Notation) [4] which is a data format that you can use to store and fetch data, JSON would allow the users to insert any type of data without crashing the application.

8.2 Back End Implementation

A reverse proxy server is provided by Uppsala University. When we want to communicate with the back end in through the front end we connect to the proxy address that redirects to the server. The database and question generator are hosted on the server locally.

8.2.1 Database Implementation

The ER Diagram of our database implementation can be found in Figure 2. An explanation and validation of the diagram follows. It can be observed that the majority of the tables have an identifier ending with “.ID”. Meaning that upon creation of a element of that list it is given an automatically incremented number. This number serves as the identifier for those tables, henceforth referred to only as ID. The purpose of ID is to reduce redundancy and to avoid having to edit all of the identifying variables in multiple tables if changed.
User  User_NAME - varchar(100) (varchar(100) means it is a plain text value up to a length of 100 characters) serves as the identifier for a user. This table has no ID since the usernames are unique in our database and works just as well as an ID. PASSWORD - varchar(100) is required for each username in order to give a primary defense to intrusion. Finally the users’ WON - int and LOSS - int, counts to provide statistics (for future development of the application).

Relation  This table contains the information needed for the user’s different relations implemented to support friend lists, blocked user lists and many more features. USER1_NAME - varchar(100) and USER2_NAME - varchar(100), the relation must be between two individual users so their user names are required. SENT_BY - varchar(100) is used to prevent a user to accept the friend request on behalf of the other. RELATION_TYPE - int describes what kind of relation the users have. This could be expanded upon, explored in Section 14 - Future Work. The identifier for a relation is the two user names. ID is not used in this table because a user can only have a single relation with...
another user.

**Question** Each question has a **Sentence** - varchar(100), the question itself, a correct **Answer** - varchar(100) and three wrong alternatives (ALT1-3) - varchar(100). **C_ID** is the ID of the course that the question is connected to. Finally **Rating** - int is used to determine the quality of the questions. Questions with a rating value below -2 does not appear in quizzes yet they remain in the database. An administrator could go through the low rated questions, delete them or try to fix them if there is a small error causing the negativity.

**Voted_on** is a related table to **Question** that is used to prevent users to cast the same vote multiple times on the same question. This is done by checking if the user and the question appear together in the table. The table contains **User_Name** - varchar(100) and **Q_ID** - int the ID in order to distinguish which user voted on what question. The users are however able to change their vote. If the requested vote sent by user is not the same as stored in this table the **Vote_Type** - int is changed.

Note that the rating is numeric value and not a count of up- and down-votes. Upon changing votes the previous vote has to be compensated for. When a user has up-voted a question their effect on the rating is +1, if we were just add -1 to the rating upon a change of vote the net effect of the user on the question would be 0 and not -1 as requested. So we use the formula: \[ \text{Rating\(\text{new}\)} = \text{Rating\(\text{current}\)} + 2(\text{Vote\_type}) \], where **Vote_Type** is the new vote of the user.

**Course** All the information needed for a course is held here. **C_ID** - int the ID for the course, **Name** - varchar(100) the name of the course so users can find their courses. **Course_Code** - varchar(100) the code that identifies the course independent of names. It could be argued that **Course_Code** should have been the identifier of the table **Course**. But there are no guarantee that course codes are unique by themselves. (Course code, University) should be unique but then ID is used over two values as explained earlier.

**University** **Uni_ID** - int, the ID for University, **Name** - varchar(100) , the full name of the Univeristy, **Name\_abbr** - varchar(100), the abbreviation of the University name (UU for Uppsala University and so forth). Universities can not be added by other users since a University can have may names (for example Uppsala University, Uppsala universitet, UU) and does not have a well known and unique code like courses does. We concluded that it would be best if universities were handled by administrators alone.
8 Implementation of Course Quizard

**Game**  
USER1_NAME - varchar(100) and USER2_NAME - varchar(100), the game is between two users so their names are required. USER1_SCORE - int and USER2_SCORE - int, keeps track of how many questions they got right in the quiz and is used to determine who won the game. Games include five questions per game and the question identifiers are in the game table Q(1-5).ID - int.

GAME_STATUS - int, gets quite intricate therefore it will require some more detailed explanation. The variable contains the current game state in order to prevent a user from playing when it is not their turn. GAME_STATUS is a numeric value and the effect of it will now be explained. (Note that User1 is the user that initiated the game and User2 is the one who got challenged.)

A value of 0 means a challenge has been sent by User1 and is waiting for User2 to start playing. When the status is 1 it means that User2, the challengee, is playing. Exiting (by User2) the game at this stage would result in User2 conceding the turn to User1 with a score of 0. Completing the turn sends the correct score as expected. Both conceding and completing the turn sets the status to 2. Now the User2 cannot access the game anymore and has to wait for User1 to play. When the status is 2 the User1, the challenger, is able to play and if doing so the status goes to 3. Exiting the game or application now has the same effect as before. Completing the turn sets the value to 4. When finally the status is 4 the game is declared to be finished and the two users can find the result in their history page.

8.2.2 PHP Scripts and MySQL Queries

The communication between the front end and back end is handled via PHP scripts. After a successful connection to the database; any variables that are required for the script are sent from the application via the POST [18] HTTP-method. The queries that requires variables use prepared statements [17] to guard against SQL injections [19] which prevents the users to insert their own malicious queries which may corrupt the database.

8.2.3 Question Generation

The question generator is implemented in Python using the NLTK library which is described in Section 6. The question generator receives a string from the front end. This string is assumed to be a text which contains one or more sentences. The string is split into sentences, each sentence is put in a list, a list of sentences is produced. Then the
tokens are extracted from each sentence (the process is called 'tokenization') and the
tokens of each sentence are stored in a list and the list which contains the tokens are
in turn stored in a list with all the other 'tokenized' lists. A token can be a comma,
a word, an apostrophe etc. The result so far is a list containing token lists, each to-
ken list represents one sentence. Each token in every token list is then part of speech
Tagged [33], a list of the tags and the meaning of the tags are presented by Harrison [28].

Each token list with tagged tokens are then converted into a question. One random suit-
able word (token) from each sentence (token list) is then blanked out, the word which
was blanked out is the correct answer. Using the answer, three alternative answers are
simulated by extracting similar words from a corpora (collection of words). The final
result is a formatted string where every sentence that was in the original string is con-
verted to a fill in the blank question with one answer and three alternatives.

Following this brief explanation of the algorithm a simple example is described. For
example if the user would input the following text "Random-access memory is a form
of computer data storage". The sentence would then be put in a list which is stored in a
list containing all sentences, however in this case there is only one sentence. The result
is then a list containing one sentence:

['Random-access memory is a form of computer data storage. ']
The sentence is then split into tokens:

[['Random', '..', 'access', 'memory', 'is', 'a', 'form', 'of',
'computer', 'data', 'storage', '..',']].

The tokens are then part of speech tagged into:

[['('Random-access', 'JJ'), ('memory', 'NN'), ('is', 'VBZ'),
'a', 'DT'), ('form', 'NN'), ('of', 'IN'), ('computer', 'NN'),
'storage', 'NN'), ('.', '.']]]

After tagging, a suitable word is blanked out. In this example the words that are tagged
as 'NN' are suitable therefore one of these words is randomly blanked. Three alter-
 natives are simulated from the word that was blanked out using the NLTK-function
Similar. A question is then formed. An example of a result would be "Random-access
___ is a form of computer data storage" The question, the answer and three generated
alternative answers are echoed to the front end.

9 Requirements

The fundamentals of the application must work well however other non essential func-
tions of the program does not necessarily have to work completely. The fundamental
functions of the application are as follows:
The application must be able to create new games where two players challenge each other on five questions-quizzes on a specific course. The games that are currently ongoing as well as the user’s history must be available for display. This requires the supporting features to function: Questions, Courses and User Profile.

The users must be able to create their own questions where they specify the question, answer, and alternatives themselves. Or they could use the question generator which generates a “fill in the blank”-question suggestion for each sentence given in the text. We have no requirements for the user questions, however the automatically generated questions must produce one “fill in the blank” question per correct English sentence with one correct answer. The alternatives may or may not be generated. The questions and the quizzes should be course specific so the users must be able to add their courses. We are not supervising the courses that are made but we supply a fixed selection of universities to choose from.

A user-profile is a must and with it the possibility to add and remove other users to their friend list. When creating a new game the user should with ease find their friends and challenge them. The quiz app gets more personal and it is easier to store the users data and let the users switch device and play on any device they prefer. In future work it will be easier to expand the user profile since the user profile is already created for all the users.

Implied but not indicatively stated yet is that the back end must be functioning properly. It is not as simple to break down into specific parts. But the back end must be able to support the front end. That means the database should be able to store all the information needed for the main functions. It needs to retrieve and supply the correct data and at a satisfactory response time. The question generator must function properly where it receives a string to convert it into questions with alternatives.

Voluntary students were asked to participate in the evaluation. The evaluation were done using the think aloud technique [30] which is be presented below. The functions of the application was tested and the test participants were asked questions regarding the different interactions of the program. All participants were asked to answer a survey with questions that are based on the requirements in Section 9 and the functions of the
program. The bullet list below shows the topics of the questions that were asked in the survey. The survey can be found in the appendix B.

- Profile and friends management.
- Playing the quizzes
- The value of the question generation and the question creation.
- Utilizing Course Quizard for studies.
- The performance of the application.

We let anyone try the application on their own if they wished. The application was posted on Facebook for everyone to download. By reading the results from the survey and the think aloud results, a full evaluation can be produced. A successful evaluation would mean that the majority of the users thought that the following points were satisfying.

- A positive user interaction with the interface.
- Enjoyable and valuable quizzes.
- Acceptable automatic question generation.
- A sufficient social network.
- Course Quizard could be useful in studies

**Think Aloud Evaluation** The think aloud method was done by setting up a scenario for the students. The scenario can be found in appendix C. The test participants were asked to do each of the steps in the scenario and the participant was guided to the goal if there were any problems. The think aloud tests were done on students at Uppsala University.
11 Results

Here follows the result of our evaluation of Course Quizard application version 1.0. The results of the think aloud tests are presented in Section 11.1 and the survey results are presented in Section 11.2.

11.1 Think Aloud Results

Eight students participated in our test scenario under observation. We studied how they managed to navigate the application and complete the tasks of the scenario. Extensive notes were taken for the first three test subjects (which can be found in appendix C). This section summarizes the observation of the test participants during the think aloud testing.

Profile management and Playing Quizzes The testers managed to find the floating action button to add a friend with ease. However, the golden star which deleted or added a friend did not seem to clearly indicate its purpose. Most of the testers accidentally pressed the star and removed their friend. Overall the testers seemed to manage their profile easily. All participants managed to start the quizzes without complications and understood how to play. Most of the testers seemed to enjoy the quizzes.

Creating or editing a generated question Some of the testers struggled while setting the correct answer and changing the alternatives. They believed it to be unclear that they were supposed to edit the questions and fill in the answer and alternatives. The question generator failed once in creating a question which confused the tester.

Back end and performance The testers did not seem to notice the background processes running, there were no performance issues or complaints. However there was a problem for a tester when generating a question. After pressing ‘generate’ the testers rarely understood that the text was being processed.

Non Android users Two of the testers did not use an Android phone regularly and had some problems with the third party Android keyboard installed on the demo device and the Material Design. Overall they managed to complete the goals very well without any problems.
11.2 Survey Results

The survey got answered by eleven students, including those who did the think aloud evaluation. This section summarizes the results of the survey. The complete result data can be found [here], see Appendix for full link.

User information Eight (72.7%) testers considered themselves to be primarily Android users and seven of them: experienced Android users. Seven (63.6%) testers preferred to play quizzes with friends and two (18.2%) testers did not like to play quizzes in general.

Profile and Friends No tester had problem with creating an account and starting the application. Six (54.5%) testers could with ease access and manage their friend list, one tester had some struggle, the rest were neutral. Two testers expressed confusion over the button design for adding and removing friend.

Questions and Games Nine (81.8%) testers considered "fill in the blank"-questions could be useful in general. Four testers tried to generate questions from text. Two of them (50.0%) thought the generated questions were valuable. Six testers tried to create their own questions and five of them (83.3%) stated that they enjoyed creating them. Seven (63.6%) testers stated it was likely they would contribute with questions if they started using the application.

Course Quizard for studying purposes Nine (81.8%) testers believed the application could be used for studying purposes, however only two (18.2%) testers considered it likely they would start to use the application for studying purposes. Nine (81.8%) testers considered it easy to create a new game and the initial impressions of the quizzes were predominantly positive.
12 Discussion

This section features a discussion of the results and the project. The general results of the application is discussed, different approaches and future improvements.

**User interaction**  The results indicated that the user interface have issues with non-intuitiveness. Part of it due to the keyboard used on the demo device. Other parts are on our side as the users had issues with feedback from the application. They did not understand immediately that their actions in the application had an effect. Most of the time the application shows a Toast (a small message pop-up) that we thought would suffice, but for the testers, it was too discrete to notice. But having a more noticeable pop up maybe would instead been more distracting after using the application for a while.

**Question creation and generation**  The results showed that most of the users would take part of the question creation. This proves that this application could work in the future if the majority takes part of question creation then the quizzes will work well. The majority of the testers would like to be able to auto generate “fill in the blank“-questions in general but only half of those who tested found it valuable. This means the quality of the question generation does not live up to the users’ expectations, which we can understand.

**Using Course Quizard as a studying tool**  Most testers had some faith in Course Quizard to be a learning tool and would even recommend it to a friend. However they were not as likely to start using the application themselves. As already explored, the idea of Course Quizard shows some potential. After the testers have tried it out, however it seems like application is not appealing enough.

**Quality of the results**  The initial results were expected, we did not think that the application was that intuitive but we wanted to test out the functionalities either way. There may not have been many testers and the survey answers were few however can we still draw some conclusion on the given results.
12 Discussion

12.1 Different approaches

**Question generator**  The question generator has a lot of improvement potential by having a word database for each course. This would increase the probability of better false alternatives that will be better and better the more questions there exist. We did not research into different NLP libraries and only focused on NLTK. There are possibilities to use other NLP libraries since the question generator is an external part of the system. Since the computational speed were not as fast as expected, and the question generation is not advanced, there may be other alternatives that are better.

**User interface**  The user interface could have been implemented differently. We could have started from a pure human-computer interaction standpoint and design our functionalities around a given design. But we chose to lay focus on the functionalities. A better looking use interface may had increased the users experience.

**Increasing the quality of the results**  Having more people testing would increase the quality of the results. We did have quite few in this case. We would have to use another strategy to increase the amount of voluntary testers. Posting on Facebook and asking some students did not give us as many testers as we would like. Uploading the application on play store would make it easier for people to download and run.

12.2 Relating to ...

**Related work**  We tried to take the concepts of quiz applications and niche it to an education tool. Our application may not have managed to reach the design of the related work applications whoever the fundamentals were there and they worked properly. The niche of the question generation showed promise for some and if it worked well it could be very useful.

**Purpose, aims and motivation**  As mentioned above, testers believe the application have some potential but does not live up to the application standard yet. The design seems to be paramount today, more important than the functionality. For the users to be able to play quizzes and use it as an alternative way of studying, there needs to be a question database with a lot of questions. The number and quality of the questions rely on the users. Since many of the testers said it was likely the would contribute with questions, the number of questions would hopefully increase. The question generator
could be more useful if the quality of the alternative generation for the questions increases. The quality of the questions will be up to the users that are able to thumb up the question or thumb down before answering the question. All questions are tied to a specific course which makes the users able to play quizzes that are relevant for them and having the ability to make the quality or the quantity of the questions greater.

13 Conclusions

The final product of the application Course Quizard reached the required functions. The application gives students an alternative way for studying however the application have some drawbacks. The results showed that although the idea showed some promise, our application did not appeal to the testers that much. The application clearly lacked in design which discouraged the users from using the application.

The question generation does not work exceptionally well and the project loses its purpose without a functioning question generator. However, the testers thought that the quizzes could be useful in general. Thus we want to have a large number of question creators. With further improvements to the question generator algorithm or increasing the types of question generated, the question generator could very well become a very valuable tool. Which would in turn attract more students to use the application. More students using the application leads to more contributors to the question database. Having a big question database could make students more likely to start using the application. The question generator and students who wants to add a lot of questions is therefore paramount for the usefulness of the application.

In conclusion, using the application could certainly increase the knowledge gained from the course material. And the results shows that there is some attraction to use the application. Sadly, the question generating algorithm leaves much to be desired and so does the design. It seems that the potential is there but it falls just short in practice.
14 Future Work

**Design** As of now the design is very basic. To create a satisfying application design more research into human computer interaction would be required. With more knowledge we can improve the design of the application drastically by designing from a human interaction perspective. A future application would include pleasant colors and animations, stimulating animations which appeals to the users.

**Question generation** In the general the algorithm of the question generator could be improved, currently there exists flaws where it does not work. To improve the question generator’s quality of wrong alternatives for the question, the file it uses to find similar words can consist of the already specific questions for the specific course. This means that the alternatives will be more relevant the more questions it exists for the course.

**Availability on other devices** Since the second most mobile operating system is iOS, developing the application of iOS will increase the amount of possible users. Other mobile operating systems are yet too small [20]. Having the quiz as an mobile web application is not a way to go since the data then have to be saved on cookies and server. Having app notification that can be implemented nicely in apps would not be the same in a web app since that would require having the web browser tab open all the time. An app can display notifications while working in the background and not consuming that much battery.

**Test before and after** Course Quizard is a tool for alternative studying. By letting users taking a test the before playing a specific course, we can store the results and showing the learning progress. It would be easier to draw concussions on what effects Course Quizard can have on learning the course material.

**Improved profile** The users may be able to have a profile pictures and view game stats. Each course can have a ranking system where users can earn points from winning games and submitting great questions. Can be a motivation for students to compete and play the quizzes seriously. As mentioned in 8.2.1, relation type could be expanded upon to allow users to have different relation types. There could be the possibility to be best friends or to block someone completely.
References


References


A User Guide

This appendix consists of a user guide of the application. We will describe the flow of the application and each view of the front end system. The flow of the system can be seen in Figure 3 below. Opening the application for the first time will turn you to the login view from where you can login with a username and password or register a new user. After logging in, the user can choose to use the different functions of the application. The main functions of the application are described in A.1

A.1 Application Functions

These are the fundamental functions of the application which the user can engage in.

Games The main function of the application is playing a quiz against an opponent. A user can challenge a friend, challenge a random opponent or play a single-player quiz. The user can access ongoing games and past games. The games are specified by course and university. The user can rate the questions based on the quality of the questions in a quiz.

Question creation Users can create questions in two ways; A question can be created manually or automatically through text. A question consists of the question itself,
one answer and three alternatives.

**User profile**  The user is able to add a friend, remove a friend and send other users friend requests. The user’s courses can be managed in the user profile. The user can see all available courses and manage favorite courses.

### A.2 Logging in and registering

The login and register views are presented in Figure 4(a) and 4(b). The login screen is presented upon entering the application for the first time. The user may enter their user-name and password to login or click on REGISTER to be directed to the register view where the user can enter a name and a password to be registered to the application. Upon a successful login, the user will be directed to the home view.

![Figure 4:](image)

**A.3 Home View**

The home view is shown in Figure 5(a). A number of options are available; One could start a new game, manage current undergoing games or create new questions for the
quizzes. The user may also open the navigation bar 5(b) where the user can access the user profile, the user’s courses, friends and the option to sign out.

![Image](image.png)

(a) Home screen  
(b) Navigation bar

Figure 5:

### A.4 New Game

The figures below shows the process of creating a new game. The user can either play in single-player mode, challenge a friend or play against a random opponent, see Figure 6(a). After selecting an opponent the user chooses a course that the quiz should be based on, Figure 6(b). When a course and opponent is chosen, the user can either confirm or edit their choices as seen in 6(c). Confirming the game sends a game request to the selected opponent.
A.5 Create Question and Generate Questions From

When creating a new question for Course Quizard there are two alternatives. Pressing CREATE QUESTION on the main menu will lead the user to write their own question with a correct answer and three false alternatives, see Figure 7(a). The second alternative is pressing GENERATE QUESTIONS FROM TEXT in the main menu to use the question generator. The generator asks for text as seen in Figure 7(b), when the user presses generate, the program will create “fill in the blanks”-questions along with a correct answer and suggestions for the false alternatives. The user is finally asked to revise the questions and checking if the questions or their alternatives are poorly generated. The user can keep or discard the questions and edit the alternatives if needed. When writing their own questions, the user does not necessarily have to create “fill in the blanks”-questions.
A.6 My Games and Playing Games

The games of the user can be seen when pressing MY GAMES on the main menu. The view can be seen in Figure 8(a). Three tabs are shown and the user can swipe between them: The first tab shows the games where it is the user’s turn to play, the second tab shows the games where it is the opponent’s turn and the last tab shows the user’s finished games.

The user can engage in a quiz against an opponent by pressing on one of the games on the MY TURNS tab. When a game is pressed, the user is taken to the quiz where the user has to answer each question in the quiz. The quiz view can be seen in Figure 8(b). As can be seen in the figure the question and the alternatives are presented, the user should then answer the question by pressing on one of the alternatives. Before answering the question, the user can thumb up or thumb down the question if the user think the question is good or not for the course.
A.7 My Profile

Upon entering the user profile, the user is presented different alternatives; friends, courses, and university changes. Pressing the FRIENDS button directs the user to the friend management screen, likewise pressing the COURSES button directs the user to the course management screen.

Managing friends   The friends screen can be seen in Figure 10(a). There are three tabs that the user can interact with. The MY FRIENDS tab shows the user’s friends. Pressing the golden star will turn the star gray and remove the corresponding friend. Pressing the gray star will then re-send a friend request. In the FRIEND REQUESTS tab you can see the friend requests that are sent to you and accept the incoming requests. The sent requests shows the friend requests that you have sent. Sending a friend request can be done anywhere on the screen by pressing the FLOATING ACTION BUTTON which takes you to the add friend screen, see Figure 10(b). Adding a friend is simple, you type out the friend and press ADD FRIEND.
Managing courses Pressing COURSES will direct you to the course screen, see Figure 6(b). The courses are sorted in three tabs, depending on how the course is related to the user. In this way the user can access their favorite courses easily. Every course
has a star, marking the star adds the course to the user’s favorite courses.
B Course Quizard Survey and results

Here the Complete URLs of the various links that were clickable can be found.

Complete URLs:

- PDF of the survey: https://goo.gl/A30CGz
- Survey result: https://goo.gl/9AX2Ej
C  Think Aloud Evaluations

The test was done in a loud office environment with people singing and talking in the background. The testers that did not own an Android device had the possibility to test the app on the Android phone Sony Z3 Compact with Android version 5.1.1. The scenario can be found [here].

User A. The user managed to navigate through the application very well following the steps in the scenario. The user had problems with Android’s keyboard sometimes when writing. Overall the user managed to complete the goals very well without any problems and the user seemed satisfied at the end.

User B. The user is not very well experienced using Android, the users primarily mobile operative systems are iOS and windows phone. The user quickly expressed dislike for Android keyboards after trying to register a user. After easily logging in the user had a hard time finding the navigation bar to add a friend. When the user was generating a question the user did not know that you were supposed to edit the question and the user did not realize that you were suppose to set the course. The user had troubles managing the friend page; the user did not know that pressing the star added or removed the friend. The user found and played the quiz easily however after playing the user did not know how to proceed then the user expressed annoyance and sighed. At the end the user mentioned that the user did not know that when creating a question you contribute to the question database.

User C. The user is very well experienced in using Android Phones, the user had no issues writing or reaching the goals in the scenario. However the user had some problems creating a question, the user did not realize that you were supposed to fill in the answers yourself. When the user was going to play against Simon he misunderstood and created a new game to challenge Simon instead of accepting a challenge. In the end it seemed like the user was satisfied when the user signed out.