ОНЛАЙН ТЕСТИРОВАНИЕ С ПОМОЩЬЮ БОТА ДЛЯ МЕССЕНДЖЕРА TELEGRAM

Я. Несабиан , А.А. Дерюшев

Белорусский государственный университет, пр. Независимости, 4, 220030, г. Минск, Республика Беларусь, nesabian@gmail.com, deryushev@bsu.by

Разработана система для тестирования студентов, содержащая базу данных, бот для мессенджера Telegram и веб-интерфейс преподавателя. Использование разработанной системы на практике позволяет повысить заинтересованность студентов в процессе обучения, упростить работу преподавателя по подготовке и проверке тестов.

Ключевые слова: автоматизированное тестирование; бот; мессенджер; Telegram; база данных.

ONLINE TESTING USING TELEGRAM BOT

Y. Nesabian, A. Deryushev

Belarusian State University, Niezaliežnasci Avenue, 4, 220030, Minsk, Republic of Belarus
Corresponding author: A.A. Deryushev (deryushev@bsu.by)

A system for testing students has been developed, containing a database, a bot for the Telegram messenger and a web interface for the lecturer. Using the developed system in practice makes it possible to increase the interest of students in the learning process, to simplify the work of the teacher in the test preparation and checking of results.

Key words: automated testing; bot; messenger; Telegram; data base.

Introduction

Automated testing of students' knowledge is a very important part of educational process. Installing special testing software requires a lot of time, that is why we decided to use software, which is already present on each mobile device. This is a messenger software, Telegram in our case.

Telegram is an open source and cross-platform messenger with over 200 million monthly active users [1]. Telegram allows developers to build bots to give services to users and this happens through the telegram APIs. In order to create a bot, first, you need to register this bot on BotFather [2]. Bots name should end with bot in their name. After that, BotFather will give you a token which you can access telegram APIs with that token. There are different libraries for different programming languages which allow you to use these APIs easily.

In this article, we introduce a telegram bot, which can take exams from the students, and persists the data in a database.

On the other side, we also implemented a web client for teachers so that they can manage questions and students and see what they are doing with the exams and at last, get an excel report if they want.

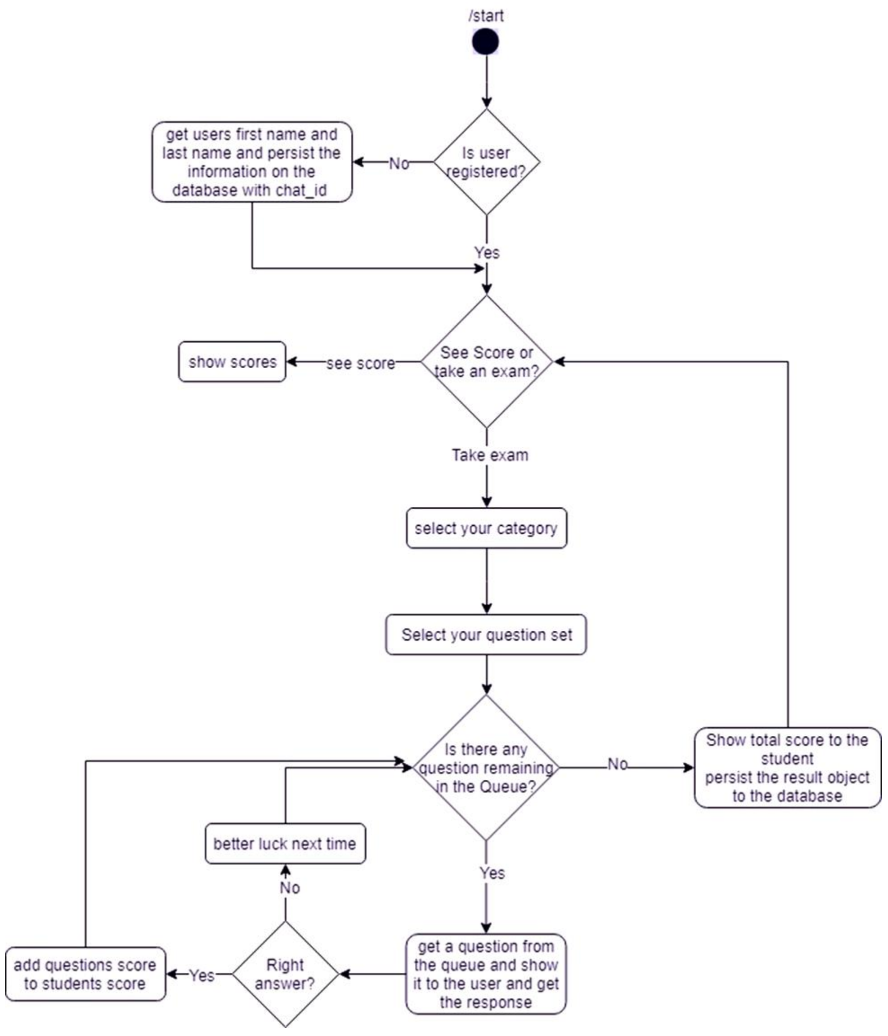
Implementing the Bot

We used python as the programming language to work with telegram APIs.

Python is an open source and cross-platform interpreter language which has a pretty easy syntax and it is widely used. We used three libraries in out python program:

• python-telegram-bot: python implementation of telegram bots;

• pymongo: allows our python program to interact with our MongoDB database;
• Emoji: allows us to use emoji in out python application.
Figure 1 illustrates the working algorithm of this bot.



/start
Is user registered?
No
get users first name and last name and persist the information on the database with chat_id
Yes
See Score or take an exam?
see score
show scores
Take exam
select your category
Select your question set
Is there any question remaining in the Queue?
No
Show total score to the student persist the result object to the database
Yes
get a question from the queue and show it to the user and get the response
Right answer?
Yes
add questions score to students score
No
better luck next time

Fig 1. Working algorithm

When a student starts the bot for the first time, out application search for that chat_id (which is a unique id assigned to each user on the telegram) in the student's collection in the database.

If there is no document with that chat_id on the database, bot asks students first name and last name and registers the student by adding a document in students collection with students chat_id.

If there is a document with that id on the database, then bot asks the student to choose a category which he/she wants to take the exam. After choosing the category, our application will fetch all the question sets on “questionset” database with the chosen category and lists the available question sets. Then student chooses a question set and the application fetches the question set models it with the QuestionSet object on the program. This helps us reduce the number of times we need to fetch data from the database.

Then each question will be shown to the student with corresponding possible answers (if any). Then an object of class Result will be created and will be filled with students responses and score he/she gets by answering the question right. When there is no question remained, Result object will be persisted to the collection “results” in the database.

After that student will return to the state where he/she can take another exam.

Implementing the database

As usual, we store all information inside database. We used MongoDB[3] as a database, because it is an open source and cross-platform NoSQL[4] database. In MongoDB we use JSON liked documents instead of records in SQL databases like MySQL or MSSQL. Each document should belong to a collection which is equivalent of table in SQL databases.

We have one database named "bsu" and inside this database, we have 4 collections:

- **Cats**: representing categories containing two fields on each document:

  _id: Identifier;

  Name: name of the category;

- **Questionsets**: each student can choose a question set to answer, each document contains the following fields:

  _id: Identifier;

  Name: name of the question set;

  Category: category of the questionset which should be chosen from cats collection;

  Questions: an array of question which student should answer in question set. Each element of this array has the following fields:

  - ✓ **Id**: Identifier;
  - ✓ **Question**: question description;
  - ✓ **Correct_answer**: correct answer to the question;
  - ✓ **Score**: score of the question if a student enters the right answer;
  - ✓ **Answers**: an array of possible answers (note that a question may have no possible answers if a question has possible answers, then it will be shown to the student with the question description, otherwise there will be no possible answers shown;

- **Students**: registered students information will be saved here and it has the following fields:

  _id: identifier;

  Id: students telegram unique id;

  Firstname: students first name;

  Lastname: students last name;

- **Results**: as the student finishes the question set, the results will be persisted to this collection. Each document has the following fields:

  _id: identifier;

  S_id: student's _id field;

  Q_id: question set's _id field;

  Timestamp: the time that student finished the question set. With this field, it is possible for the students to take one question set to answer multiple times and different timestamp and the instructor can see the results with different time and see the progress etc. Note that this timestamp will cast to human readable date and time on reports;

  Answers: a set of key values in form of {"question_id": "answer"} which "question_id" is the question id of the answered question and "answer" is answer entered by the student.

In our project, the database is deployed to mLab [5] which allows to deploy databases freely up-to 500mb.

Web interface for the instructor

We also implemented a web interface for the instructor to be able to define new category, question set, manage question set (like adding new question to that question set) and see overall of all students and see status of each student and export the results to Excel.

We used expressjs library which helps us create webservice so we can communicate with this web service using HTTP requests such as GET, POST, PUT, DELETE, and mongoose to communicate with the mongodb database. We can define models with mongoose so that we can perform some sort of input validation when we want to persist data on the database.

We also used json2xls library which we can put the data in form of JSON together and this library will convert it to Excel file. We used this library to generate reports.

Since we have built a web service, we can create desktop client, web client, android client, iOS client etc. We have implemented the web client but we used bootstrap in the web client so that we have an appropriate experience when using this client.

We also used AngularJS framework in the front-end to communicate with the web service. In the front end, we have a service layer which is responsible for contacting with the web service.

Conclusion

Testing software, including Telegram bot, database and webinterface for lecturer was created and tested using free hosting servers. This approach is very interesting for students and very easy to use by lecturer. Using this software big set of tests will be created and used during teaching “Mobile OS” and “Crossplatform mobile development” courses.

References

1. Telegram. A new era of messaging [Electronic resource]. – Mode of access: https://telegram.org.
2. Bots: An introduction for developers [Electronic resource]. – Mode of access: https://core.telegram.org/bots.
3. MongoDB for giant ideas [Electronic resource]. – Mode of access: https://www.mongodb.com.
4. NoSQL Databases Explained [Electronic resource]. – Mode of access: https://www.mongodb.com/nosql-explained.
5. MLab. Trusted. Loved. Most widely deployed [Electronic resource]. – Mode of access: https://mlab.com.