Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University

Aitkali Sultan, Dikanbayeva Assura, Dyussenova Anel,
Valiyeva Nassiba, Tazhibayeva Aidana

System for making an appointment with a doctor, including a telegram bot and a doctor's personal office, with the function of predicting illness by symptoms

A thesis submitted for the degree of
Bachelor in Information Systems
(degree code: 6B06101)

Kaskelen, 2021

Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University
Faculty of Engineering and Natural Sciences

System for making an appointment with a doctor, including a telegram bot and a doctor's personal office, with the function of predicting illness by symptoms

A thesis submitted for the degree of
Bachelor in Information Systems
(degree code: 6B06101)

Author: **Aitkali Sultan, Dikanbayeva Assura, Dyussenova Anel, Valiyeva Nassiba, Tazhibayeva Aidana**

Supervisor: **Ardak Shalkarbay-uly**

Dean of the faculty:
Assist. Prof. Meirambek Zhaparov

Kaskelen, 2018

Abstract

A program that allows you to make an appointment with specialists/doctors without unnecessary effort and wasting time. In this case, it is a telegram bot. The main reason, the purpose of using this method of recording is to prevent the waste of unnecessary time and energy as well as specialists. By integrating machine learning into the basic skills of the telegram bot, it was possible to recognize the timing diseases on the symptomatology. This process speeds up the decision-making in the question "which doctor should I go to?" Upon identification of a probable illness, telegrams the recommended doctor with full information about him and immediately makes an appointment with him. This process can become one of the best options for people in the 21st century.

Аңдатпа

Артық күш жұмсамай және уақыт жоғалтпай дәрігерлерге жазылуға мүмкіндік беретін бағдарлама. Бұл жағдайда ол телеграм боты болып табылады. Жазба әдісін құрудың басты себебі, науқастар үшін де, мамандар үшін де қажетсіз уақыт пен энергияны ысыраптаудан сақтау. Машиналық оқытуды телеграм ботының негізгі дағдыларына біріктіру арқылы тапсырыс шалушының белгілерін симптомдар негізінде тануға мүмкіндік туды. Бұл процесс «қай дәрігерге баруым керек?» деген сұрақ бойынша шешім қабылдауды тездетеді. Ықтимал ауруды анықтағаннан кейін телеграм бот нақты дәрігерді толық ақпаратпен ұсынып, оған жедел жазылуға мүмкіндігін береді. Бұл процесс XXI ғасырдағы адамдар үшін ең жақсы нұсқалардың біріне айналуы мүмкін.

Аннотация

Программа позволяющая производить запись к врачам-специалистам без лишних усилий и траты времени. В данном случае - это телеграм бот. Основная причина, цель создания данного метода записи - предотвращение траты лишнего времени и энергии как и пациентов, так и специалистов. При помощи интеграции машинного обучения в базовые навыки телеграм бота, удалось распознавать болезни задающего основываясь на симптоматике. Данный процесс ускоряет принятие решения в вопросе "к какому врачу стоит обратиться?". По выявлению вероятной болезни, телеграм бот рекомендует определенного врача с полной информацией о нем и возможностью незамедлительно записаться к нему на прием. Данный процесс может стать одним из оптимальных вариантов для людей 21 века.

Contents

Chapter 1

Introduction

1.1 Motivation

Would you believe, if we say that we can simplify the appointment process to the doctor?

We guess, 10 years ago, no one thought that, in the future, after some years, there will appear technology by using of this, peoples can easily sign up to the doctor's reception, without any extra moves, calls, visits, and so on. Yes, peoples can say, "what's the difficulties of nowadays methods of appointment to the doctor, we can take a phone, and by one call, make an appointment to the doctor", yeap, sounds easy, but ask them a question, "how much time they spend to it?", I think approximately 5-10 minutes, it's in a good conjuncture, sometimes it may take a much more time, call waitings and so on. Would you believe that this process may be done in a no more than 5 minutes? It was hypothesized that, half of the callers to the call center of some clinic, could not get an answer to their call, it means, the can not make an appointment to the doctor. Sounds failure, isn't it? And by processing this information, we identified next problems:

- Bad service: The call centers of clinics don't work clearly fine, there may be some issues, like long call waitings, no answer to the call, or line may be busy, and so on.
- Time Efficiency: After getting the problems which are listed above, what can be a conversation about time efficiency? Completely the opposite.
- Call centers are not available during non working hours.

Everyone knows that nowadays technologies do not stand in one place, and everything should be updated. We have to go in step with the time. Very little time will pass and everyone will use the system, which will be similar to ours. And we want to be one of the founders of this, I want to name it, “New era of online appointment processes”.

1.2 Aims and Objectives

We are all aware that the part that includes main objectives, mission aims and those related to the initiation is one of the most critical ones of introduction part. Before you start the project, you have to answer some open questions, like, what, why, where and how, to better understand what kind of problems you want to solve. So, first, let's start from defining what aims are here.

1.2.1 Aims

- What is the main idea of the "MedBot"?
- Could our project make an effect on people's life?
- Would it make their lives easier?
- Who is the target audience?
- Could "MedBot" help to the call center by dividing and doing their work?

1.2.2 Objectives:

- Simplify the process of online appointment to the doctor
- Providing understandable and friendly design
- Providing the customer with decent quality of service which will be available 24/7
- Kind of assistance in the work of the call center

1.3 Thesis Outline

- Chapter 1 Introduction: This chapter we includes our motivation of doing this project, and main purposes of the "Medbot" project.
- Chapter 2 Front End: Contains an front end overview of our project, described the technologies used, packages used, and other components.
- Chapter 3 Back end: Explained the routes, and how we created our telegram bot, what platform do we choose, and so on.
- Chapter 4 Machine Learning: We used machine learning to identify disease based on symptoms, the implementation steps of which are described in this chapter.

- Chapter 5 Logical and physical database models: Provided the models of logical and physical databases, and their contains, entities, tables.
- Chapter 6 Design: In this part you can get acquainted with design of one-page web page, doctor's admin page, what colors, fonts do we use.
- Chapter 7 Conclusion: We summarized of done work.

Chapter 2

Front End

2.1 Overview

This project has a beautiful design and rich functionality that needs to be implemented, properly tested, and that will work really well for the end users. Front-end developers are responsible for these things. In order to execute those objectives, front-end devs must be adept at three main languages: HTML, CSS, and Javascript programming. In addition to fluency in these languages, front-end devs need to be familiar with frameworks, which ensure great-looking content no matter the device and libraries, which package code into a more useful, time-saving form. [11] Also pages should be dynamically loaded by downloading server data in the background. This is a simple image that demonstrates how it works. 2.1

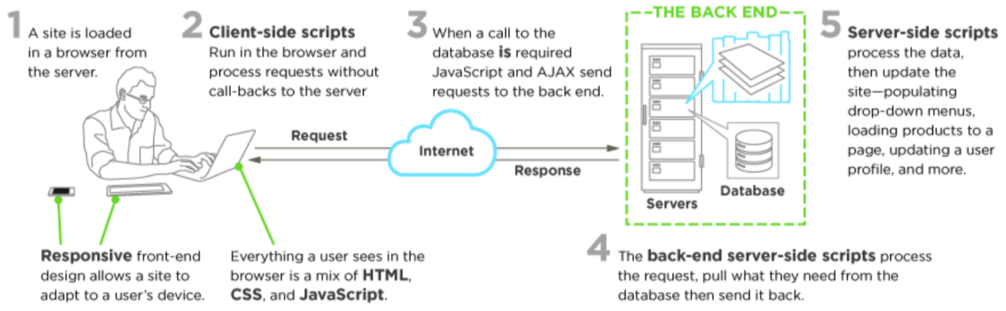

1 A site is loaded in a browser from the server.
2 Client-side scripts Run in the browser and process requests without call-backs to the server
3 When a call to the database is required JavaScript and AJAX send requests to the back end.
THE BACK END
5 Server-side scripts process the data, then update the site—populating drop-down menus, loading products to a page, updating a user profile, and more.
Request
Internet
Response
Servers
Database
Responsive front-end design allows a site to adapt to a user's device.
Everything a user sees in the browser is a mix of HTML, CSS, and JavaScript.
4 The back-end server-side scripts process the request, pull what they need from the database then send it back.

Figure 2.1: Interaction of the front-end with the back-end

2.2 Technologies used

There are many frameworks and libraries available for frontend development. However, this project uses a library called React. There are several reasons why. First one is that react is one of the most popular and widely used libraries for frontend development,which was developed by Facebook. Also react is an open-source JavaScript component-based library that lets you build high-quality user-interfaces for web apps, specifically for single-page applications. [9] Second huge advantage of react is that it allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application. This corresponds to the view in the MVC template. In React, a set of immutable values are passed to the component's renderer as properties in its HTML tags. The component cannot directly modify any properties but can pass a callback function with the help of which we can do modifications. This complete process is known as "properties flow down; actions flow up". 2.2

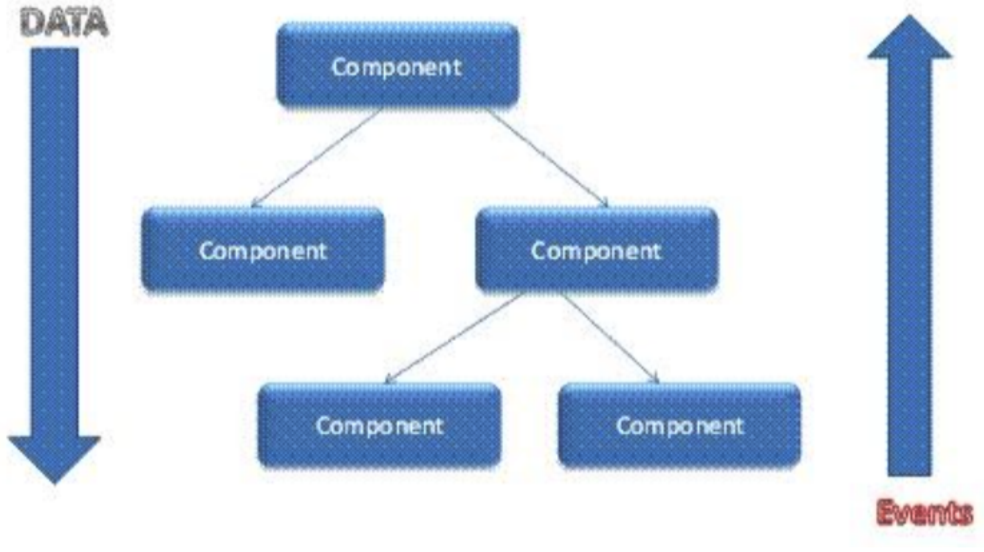


Figure 2.2: Properties flow down; Actions flow up [8]

React creates an in-memory data structure cache which computes the changes made and then updates the browser. This allows a special feature that enables the programmer to code as if the whole page is rendered on each change whereas react library only renders components that actually change.

The latest Stack Overflow Developer Survey finds that React is the most loved among the frameworks, libraries and other technologies: 66.9 percent of developers gave their votes for it. [1] 2.3

2.3 Packages used

All the packages were installed using npm. Npm is a software Package Manager and Installer, which is popular worldwide. All npm packages are defined in files called package.json.

Main packages used in the project's front-end part:

- react-dom - provides DOM-specific methods that can be used at the top level of your app;
- react-router-dom - popular tool that works really well with React Router;

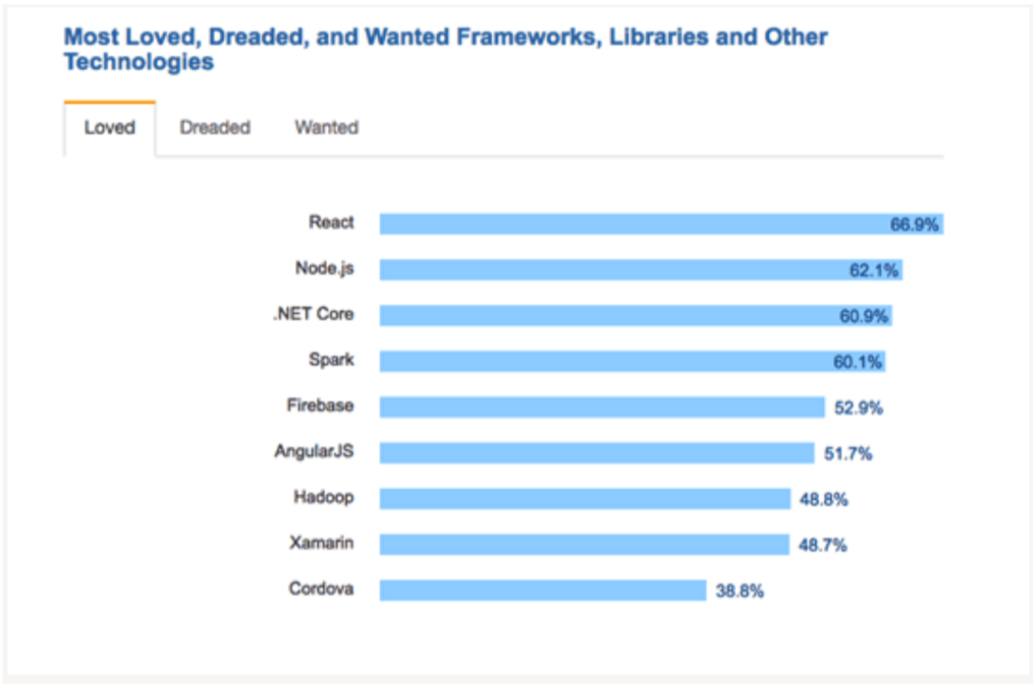


Figure 2.3: Stack Overflow Developer Survey

- @material-ui - a simple and customizable component library to build faster, beautiful, and more accessible React applications;
- axios - promise based HTTP client for the browser and node.js;
- prop-types - runtime type checking for React props and similar objects;
- styled-components - allows you to write actual CSS code to style your components;
- react-stars - a simple star rating component for React projects;
- react-elastic-carousel - a flexible and responsive carousel component for react;
- react-select - a select control built with and for React;
- @fullcalendar - a full-sized drag and drop event calendar.

2.4 Components

2.4.1 React components

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render() function. Components come in two types, Class components and Function components. This project contains only functional components

Functional Component:

- It is a simple javascript function that returns html UI;
- It is also called “stateless” components because they simply accept data and display them in some form, that is they are mainly responsible for rendering UI;
- It accept properties(props) in function and return html(JSX);
- There is no render method used in functional components.

We can use React Hooks in functional components,useEffect() hook can be used to replicate lifecycle behaviour, and useState can be used to store state in a functional component.

2.4.2 Project’s main components

Routing is the process of keeping the browser URL in sync with what’s being rendered on the page.

Routes - it is a component that contains all routes of the app. Each route is contained in a prop of component called RouteWithLayout. There are 4 routes:

- /login
- /doctor/:id/user/:user$_i$*d/cabinet/* : *id*

- /not-found

RouteWithLayout - it is a Route component that accept such properties(props) as layout, component, isDesktop and path,where:

- layout - functions which return components;
- component - it is a view that consists of smaller components;
- isDesktop - is boolean which is responsible for responsive web design;
- path - name of the routes (listed above).

The render prop of Route component expects a function that returns an element when the location matches the route’s path. The render prop accepts component and layout.

2.4.3 Project's layouts

The main idea of layouts is that it is an 'outer' component which is presented in almost every page of web applications such as navbar or footer. It is called higher order components - functions which return components. Each page component wraps its inner content within a reusable layout component.

There are two layouts in this project, because there are few pages.Main and Minimal layouts. In the Main layout there is a small footer. Minimal layout used for NotFound page.

2.4.4 Project's views

Simply said, views are the components that lay inside the layout. Navigating through these views shouldn't result in the entire page being reloaded. Instead, views are rendered inline within the current page. There are 4 view in this project:

- DoctorProfile - a page with information about doctor and his timetable;
- Login - simple login page for doctor's cabinet;
- DoctorAdmin - a page where doctors can find out about their visits and regulate their schedule, also they can get information about their patients;
- NotFound - if visitor clicks on a link to a deleted or moved page they'll see 404 error page.

Chapter 3

Back-end

3.1 Routes

1. GET /getdoctors

   - Returns a doctor, also returns a list of his specialization for displaying information on the DoctorProfile page. Returns the Patient to create the entry. Pass the boolean variable visit, which indicates whether this patient has active visits.

Parameter	Data type	Description
doctor	Number	Doctor id
user	Number	Patient chat id

2. GET /getcabinetvisit

   - Returns Doctor's Appointments:
     - Today's visits with "confirmed" status
     - Today's visits with "done" status
     - All visits with "not confirmed" status

Parameter	Data type	Description
doctor	Number	Doctor id
user	Number	Patient chat id

3. POST /changetodone

   - Change status of the visit from "confirm" to "done"

Parameter	Data type	Description
doctor id	Number	Doctor id
user id	Number	Patient chat id

4. POST /changetocancel

- Change status of the visit from “confirm” to “cancel”

Parameter	Data type	Description
doctor id	Number	Doctor id
user id	Number	Patient chat id

5. POST /savevisit

   - Save visit

Parameter	Data type	Description
doctor id	Number	Doctor id
user id	Number	Patient chat id
symptoms	Array	Array of symptoms from telegram bot
date visit	String	Date of the visit
disease	Array	Predicted disease from telegram bot

6. POST /confirmvisit

   - Add analyzes and comment to a visit

Parameter	Data type	Description
doctor id	Number	Doctor id
user id	Number	Patient chat id
analysis	Array	List of analyzes
comment	String	Comment from a doctor

7. POST /login

   - Sign in to doctor cabinet

Parameter	Data type	Description
name	String	Doctors first name
surname	String	Doctors surname
password	String	Password

8. GET /getanalysis

   - Returns the entire list of analyzes

Parameter	Data type	Description
-	-	-

9. GET /getdonevisittoday

   - Returns a list of visits for today with the status "done"

Parameter	Data type	Description
doctor id	Number	Doctor id

10. GET /getconfirmedvisittoday

    - Returns a list of visits for today with the status "confirmed"

Parameter	Data type	Description
doctor id	Number	Doctor id

11. GET /getconfirmedvisitweek

    - Returns a list of visits for a week with the status "confirmed"

Parameter	Data type	Description
doctor id	Number	Doctor id

12. GET /getconfirmedvisitmonth

    - Returns a list of visits for a month with the status "confirmed"

Parameter	Data type	Description
doctor id	Number	Doctor id

13. GET /getnotconfirmedvisit

    - Returns a list of visits with the “not confirmed” status

Parameter	Data type	Description
doctor id	Number	Doctor id

3.2 Telegram Bot

3.2.1 Choosing a platform for implementing the Bot

We use the Dialog Flow platform and integrate it into the Telegram Bot. We need Dialog Flow primarily to recognize user-written symptoms. We didn’t come to the decision to use Dialog Flow right away. First, we considered the idea of asking the user a series of questions that would be built according to the logic of a decision tree: If answered like this, then question1, if answered like this, then question2, etc. The chain of questions should ultimately lead to the last question that will give a list of answers symptoms. Since the issued symptoms from the list will be linked by a certain logic and most likely refer to the same diseases, it is most likely that the user will want to choose more than one answer, but several. Unfortunately, the Telegram Bot does not have the function of selecting multiple answers. For this reason, we thought to implement our Bot on the Web, but this would increase the amount of work. This was the first problem we encountered. The second was to separate symptoms and group them; it was also difficult for us, as for people far from medicine, to make chains of questions correctly, minimizing their number. Therefore, we decided that the best option would be to use the NLP model to recognize symptoms from text written by the user. But training such a model would require a large amount of data, which we did not have. We could use transfer learning and take a ready-made model that recognizes Russian text well (ideally with a bias towards medicine) and train it to recognize symptoms. Which, in principle, we did, because Dialog Flow works exactly according to this logic.

3.2.2 Dialogflow

Dialogflow is a natural language understanding platform used to design and integrate a conversational user interface into mobile apps, web applications, devices, bots, interactive voice response systems and related uses. [5]

3.2.3 Intent

An intent categorizes an end-user's intention for one conversation turn. For each agent, you define many intents, where your combined intents can handle a complete conversation. When an end-user writes or says something, referred to as an end-user expression, Dialogflow matches the end-user expression to the best intent in your agent. Matching an intent is also known as intent classification. [7]

This project contains 19 intents, two of which are default, each of them performs its own separate function.

3.2.4 Entity

Each intent parameter has a type, called the entity type, which dictates exactly how data from an end-user expression is extracted.[4] We use the default system entity dialogflow, for name recognition - sys.person, age - sys.age and phone number - sys.phone-numder. We also made custom entities for those parameters that are not presented in the system. The Doctor entity contains the entire list of specializations. In Gender Entity, we have a list of genders. We use entity for these two cases so that dialogflow can use them as parameters, since all parameters need to have entity. Which will give us the opportunity to transfer them in context to the next intent. For Doctors and Gender entities, we do not use define synonyms, allow automated expansion, fuzzy matching. Since, the person does not enter them, but chooses from the possible options in the inline keyboard. Symptom Entity is a symptom for which we have included features such as: define synonyms, allow automated expansion, fuzzy matching. In order for the bot to determine the same synonyms, typos and approximately matching the template.

3.2.5 Integrating Dialogflow to Telegram Bot

We are creating our bot on the Dialogflow platform. And this fact gives us the opportunity to implement the bot on various platforms, such as: One-click telephony: Dialogflow Phone Gateway, Avaya, SignalWire, Voximplant, AudioCodes, Twilio Telephony: Genesys Cloud, Twilio Text based: Web Demo, Dialogflow Messenger, Messenger from Facebook, Workplace from Facebook, Slack, Telegram, LINE It is important to control where the message comes from, and return it in a format convenient for the given platform. We have linked a telegram bot via a token in DialogFlow. We also linked our Dialogue Flow bot with the code using Webhooks. Each intent has a setting to enable fulfillment. If an intent requires some action by your system or a dynamic response, you should enable fulfillment for the intent. [6] We use the hooked up fulfillment function on intents:

1. Yes. Here we need a webhook, since basically all information is stored at this stage, such as: age, gender, phone, symptoms.

2. No. It is used to ask for information again, depending on what the intent was before. The webhook is required to keep track of the previous intent using context.

3. DefaultFallback. We use the webhook on this intent in order to catch when a person answers the question: "Do you have any allergies or chronic diseases, and if there is something that is important to know to the doctor, write, please". Since it is difficult to write Training phrases for this task. But since this question is asked once, we use the allok parameter, which we store in the Patient collection.

4. DefaultWelcome. Here we need a webhook to check if a given user is in our database and if there is, then do not ask him basic information (name, age, gender, phone number) again

5. NameSurname. Since Dialog Flow does not work well with the recognition of Kazakh names. The webhook at this stage gives us more control, since it returns not only the person parameter, in which the recognized name should be stored, but also returns a text query in which all the text written by the user is transmitted. Using text query we can get more accurate data than from the person parameter. Although all other intents with a similar function, such as Age, Gender, Phone do not send a webhook (except for Age intent, but in his case there is a different reason), the parameter from these intents is passed as a context to the Yes and No intents, which in turn preserve information to the database.

6. Age. Here we need a webhook to correctly display the amount parameter, which stores either year or month. Dialogflow displays this parameter in English, so we use a webhook to display the information correctly.

7. SymptomCheck. Used to save symptoms to a database.
8. SpecializationList. We use a webhook to retrieve specialization names from a database.
9. DoctorList. We use a webhook to retrieve the names of our doctors from the database.
10. DoctorListName.
11. History. We use a webhook to display information about visits from a database.

3.2.6 Integrating Machine Learning to Bot

We link our bot to a trained Decision Tree Classifier model. Prediction of diseases by symptoms is started when our Intent Yes is triggered, provided that the previous intent was SymptomCheck. We run our python code in the Python shell. Python shell uses the Child Process under the hood. We pass as arguments the symptoms that the patient wrote, recognized by the dialog flow. Already in python, we translate an array with symptoms into one hot vector. And using this data, we start the prediction process. Next, we pass the predicted disease back.

Chapter 4

Machine Learning

We used machine learning to identify disease based on symptoms. The user enters a list of symptoms of concern to the bot. After that, we must use machine learning to give him a probable disease. We are faced with the task of classification.

4.1 Data

Since we use the data already collected [3], we did not encounter problems: incompleteness, noise or inconsistency of data. First of all, we started with an initial examination of the dataset to determine and plan the necessary preprocessing. As already described in the chapter (chapter title and number), we have a list of diseases and each disease has its own list of symptoms. Looking at the database, we noticed that symptoms recur between diseases. And with some manipulation, we got a complete list of all the symptoms. We then discretized the data by converting the continuous symptom attribute to categorical, since we use the Decision Tree Classifier algorithm, which works best with categorical parameters. We did not use the PCA trait combination. Since each parameter is a separate symptom, we figured that if we use parameter correlation, we might lose important data.

4.2 Methods

We used a machine learning algorithm - Decision Tree Classifier, to solve the problem posed to us. Decision tree learning or induction of decision trees is one of the predictive modeling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). [2] We chose to use the Decision Tree Classifier because:

1. Requires little data preparation.

2. Doesn't require a lot of data.

3. Uses a white box model. This problem observed in the model is easily explained by Boolean logic.

4. Easy to understand and interpret.

4.3 Implementation

The Decision Tree model receives an array of data, the number of parameters is 405 - the number of all symptoms. The model returns the most likely disease based on the given symptoms.

Chapter 5

Logical and physical database models

5.1 Description of the logical and physical database models

The logical database model is the name of entities and their attributes in natural language, and also displays the relationships between entities. The physical database model represents entities as they will be stored in the database.

5.2 Logical database model

The logical database model 5.1 has six entities: "Doctor", "Patients", "Disease", "Specialization", "Analyses", "Visit".

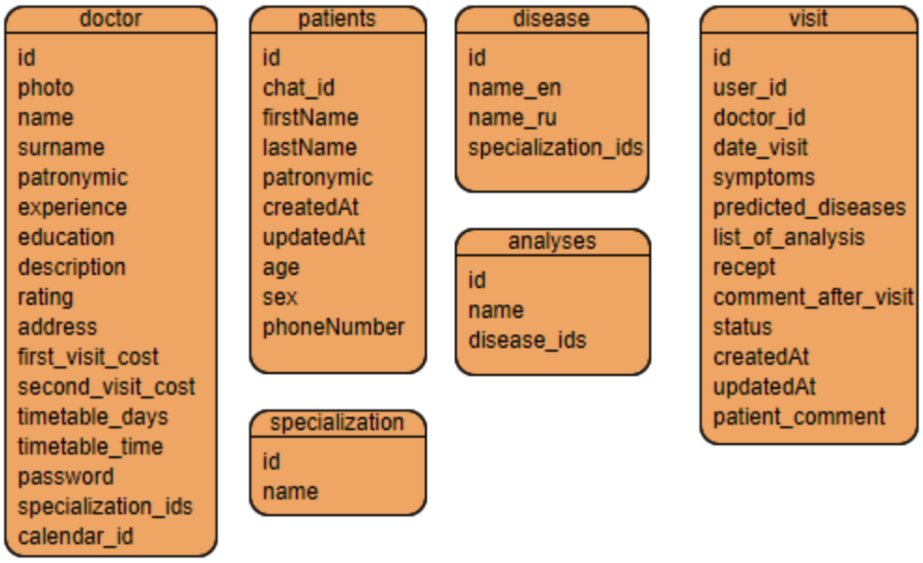


Figure 5.1: Logical database model

1. The entity "Doctor" ("doctor") stores information about the doctor who works in the clinic. Has attributes such as doctor's id; photo - a photo of a doctor; name - doctor's name; surname - doctor's surname; patronymic - patronymic of the doctor; experience - work experience; education - education received; description - doctor's description; rating - doctor's rating; address - the address of the clinic where the doctor works; `first_visit_cost` - the initial cost of visiting a doctor; `second_visit_cost` - cost of the second visit to this doctor; `timetable_days` - days when the doctor works; `timetable_time` - hours of the doctor's work; assword - password for the site page; `specialization_ids` - specialization code; `calendar_id` is the link for the calendar.

2. The "Patients" ("patients") entity - stores the history of changes, additions, deletions of patients. Has such attributes as patient id, `chat_id` - patient chat id; firstName - patient name; lastName - the last name of the patient; patronymic - patronymic of the patient; createdAt - time when it was registered; updatedAt - time when he changed his data; age - the patient's age; sex - patient's gender; phoneNumber - patient number.

3. Entity "Disease" ("disease") - an entity for storing a list of diseases. Has attributes such as disease id; `name_en` - name of the disease in English; `name_ru` - name of the disease in English; `specialization_ids` - the code of the specializations that treat this disease.

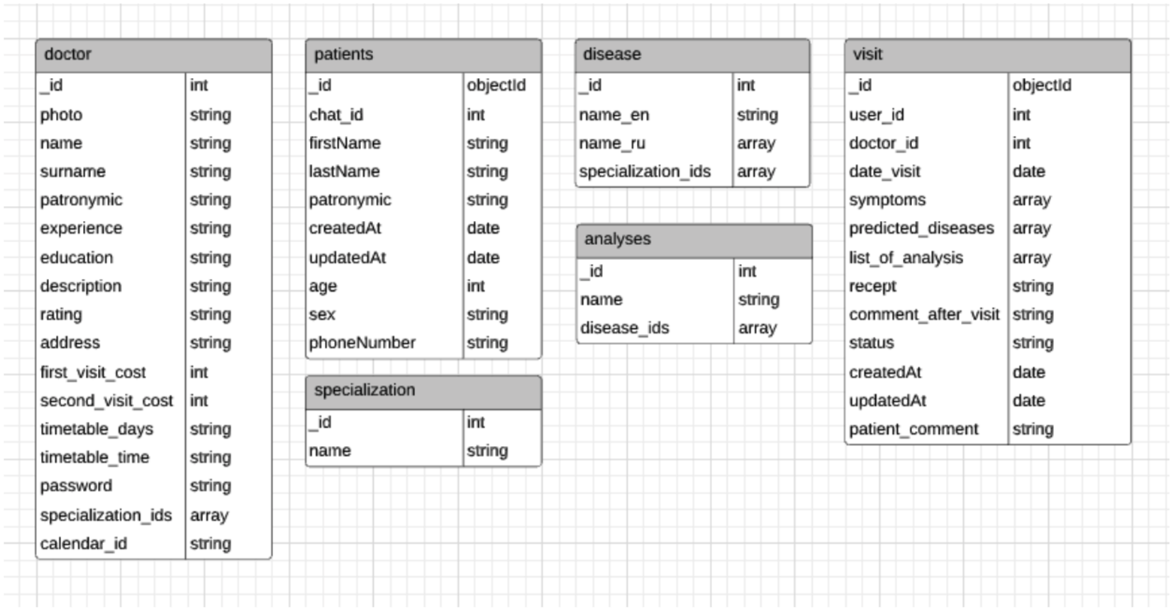
4. Entity "Specialization" ("specialization") - an entity for storing the list of specializations. Has attributes such as specialization id; name - the name of the specialization.

5. The "Analyses" ("analyses") entity is the entity for storing the list of analyzes. Has attributes such as analysis id; name - the name of the analysis; `disease_ids` - disease id for which analyzes are required.

6. The "Visit" entity ("visit") - stores information about the records made by the patients of the system. visits. Has attributes such as record id; `user_id` - patient chat code; `doctor_id` - the code of the doctor to whom the patient signed up; `date_visit` - date of the patient's visit to the record; symptoms - the patient's symptoms; `predicted_diseases` - the patient's possible disease; `list_of_analysis` - list of analyzes that the patient must undergo; recept - doctor's appointment; `comment_after_visit`CommentAfterVisit - comment from the patient after visiting the doctor; status - the status of the visit to the reception; createdAt - the time the record was created; updatedAt - the time the record was updated; patient comment - additional information about the patient.

5.3 Physical database model

The physical model contains information about all objects in the database. The physical model shows intermediate tables and attribute data types. The physical model includes six main tables: "Doctor", "Patients", "Disease", "Specialization", "Analyses", "Visit" 5.2



doctor	
_id	int
photo	string
name	string
surname	string
patronymic	string
experience	string
education	string
description	string
rating	string
address	string
first_visit_cost	int
second_visit_cost	int
timetable_days	string
timetable_time	string
password	string
specialization_ids	array
calendar_id	string

patients	
_id	objectId
chat_id	int
firstName	string
lastName	string
patronymic	string
createdAt	date
updatedAt	date
age	int
sex	string
phoneNumber	string

specialization	
_id	int
name	string

disease	
_id	int
name_en	string
name_ru	array
specialization_ids	array

analyses	
_id	int
name	string
disease_ids	array

visit	
_id	objectId
user_id	int
doctor_id	int
date_visit	date
symptoms	array
predicted_diseases	array
list_of_analysis	array
recept	string
comment_after_visit	string
status	string
createdAt	date
updatedAt	date
patient_comment	string

Figure 5.2: Physical database model

1. Collection "Doctor" ("doctor") has such attributes as:

   - id (Int) - doctor's id, primary key;
   - photo (String) - link to the doctor's image;
   - name (String) - doctor's name;
   - surname (String) - doctor's surname;
   - patronymic (String) - patronymic of the doctor;
   - experience (String) - work experience;
   - education (String) - description of the education received;
   - description (String) - doctor's description;
   - rating (String) - doctor's rating;
   - address (String) - the address of the clinic where the doctor works;
   - `first_visit_cost` (Int) - the primary cost of a visit;
   - `second_visit_cost` (Int) - cost of the second visit;
   - `timetable_days` (String) - days when the doctor works;

- timetable_time (String) - doctor's working hours;
- password (String) - password for the site page;
- specialization_ids (Array) - specialization code;
- calendar_id (String) - Calendar link.

2. The collection "Patients" ("patients") has such attributes as:

   - id (Int) - patient id;
   - chat_id (Int) - patient chat id;
   - firstName (String) - patient name;
   - lastName (String) - patient's last name;
   - patronymic (String) - patronymic of the patient;
   - createdAt (Date) - registration time;
   - updatedAt (Date) - the time the action was changed;
   - age (Int) - patient's age;
   - sex (String) - patient's gender;
   - phoneNumber (String) - patient number.

3. The collection "Disease" ("disease") has such attributes as:

   - id (Int) - disease id, primary key;
   - name_en (String) - name in English;
   - name_ru (String) - name in English;
   - specialization_ids (Array) - the specialization array for the disease.

4. Collection "Specialization" ("specialization") has such attributes as:

   - id (Int) - specialization id, primary key;
   - name (String) - Specialization name.

5. The "Analyses" ("analyses") collection has such attributes as:

   - id (Int) - disease id, primary key;
   - name (String) - name of the analysis;
   - disease_ids (Array) - an array of diseases.

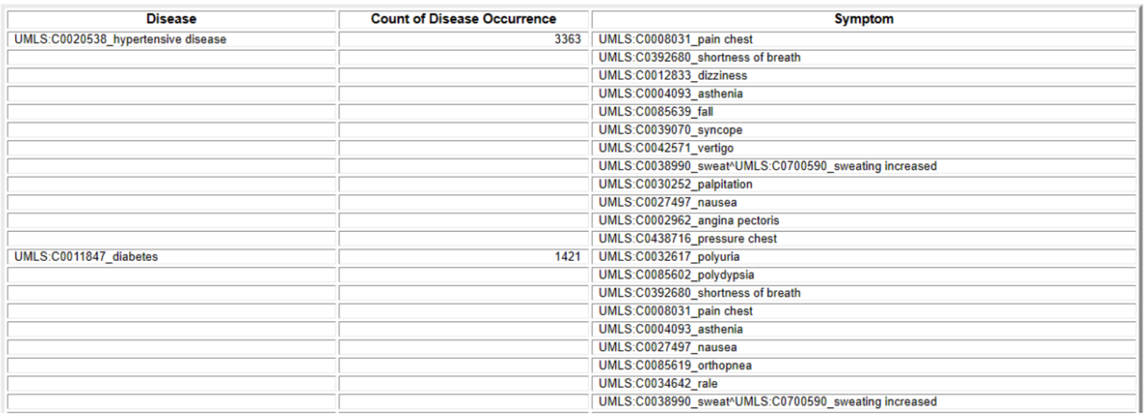
6. Collection "Visit" ("visit") has such attributes as:

   - id (Int) - disease id, primary key;
   - user_id (Int) - patient id;

- `doctor_id` (Int) - doctor's id;
- `date_visit` (Date) - date of visit;
- symptoms (Array) - an array of the patient's symptoms;
- `predicted_diseases` (Array) - an array of possible diseases of the patient;
- `list_of_analysis` (Array) - array of analyzes;
- recept (String) - destination;
- `comment_after_visit` (String) - comment from the patient after visiting the doctor;
- status (String) - the status of the reception visit;
- createdAt (Date) - the time the record was created;
- updatedAt (Date) - the time the record was updated;
- `patient_comment` (String) - Additional patient information.

5.4 Disease-Symptom Knowledge Database for ML

This table below 5.3 is a knowledge database of disease-symptom associations generated by an automated method based on information in textual discharge summaries of patients at New York Presbyterian Hospital admitted during 2004.[2] The first column shows the disease, the second the number of discharge summaries containing a positive and current mention of the disease, and the associated symptom. Associations for the 150 most frequent diseases based on these notes were computed and the symptoms are shown ranked based on the strength of association.



Disease	Count of Disease Occurrence	Symptom
UMLS:C0020538_hypertensive disease	3363	UMLS:C0008031_pain chest
		UMLS:C0392680_shortness of breath
		UMLS:C0012833_dizziness
		UMLS:C0004093_asthenia
		UMLS:C0085639_fall
		UMLS:C0039070_syncope
		UMLS:C0042571_vertigo
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased
		UMLS:C0030252_palpitation
		UMLS:C0027497_nausea
		UMLS:C0002962_angina pectoris
		UMLS:C0438716_pressure chest
UMLS:C0011847_diabetes	1421	UMLS:C0032617_polyuria
		UMLS:C0085602_polydypsia
		UMLS:C0392680_shortness of breath
		UMLS:C0008031_pain chest
		UMLS:C0004093_asthenia
		UMLS:C0027497_nausea
		UMLS:C0085619_orthopnea
		UMLS:C0034642_rale
		UMLS:C0038990_sweat^UMLS:C0700590_sweating increased

Figure 5.3: Knowledge database of disease-symptom

Chapter 6

Design

6.1 General understanding of project design

The design of the project has two distinct parts. 1 - design of a landing page with full information about the receiving doctor intended for patients: doctor's name, work experience, background, time and cost of the visit. 2 - a landing page design that is a doctor's admin page.

6.2 Design and functionality of a one-page web page

The main idea behind the design of a one-page web page is the ability to view basic information about a doctor while making an appointment through a telegram bot developed by us. This page is responsive for mobile devices.

A landing page consists of the following blocks:

- Block number 1 - contains the following components: a photo of a doctor, his name and specialization.
- Block number 2 - contains information about the doctor, such as: work experience, education and type of activity, for a more detailed acquaintance with this specialist.
- Block number 3 - consists of a slider with the available time for an appointment with a given doctor: time and day.
- Block number 4 - contains information about the cost of admission, both primary and repeated.
- Block number 5 - the last block showing the place where this doctor will receive, so that patients have no questions.

After a detailed acquaintance with the doctor, the patient has the right to make an appointment with him for the time needed, or in case of any inconsistencies, view the profile of another doctor of the same specialization and choose whom he wants to make an appointment with.

6.3 Design and functionality of a doctor's admin page

The doctor's personal office is intended for specialists, i.e. Physicians could track patients coming / leaving, see their schedule, manage patient enrollment, and view their personal records.

The doctor's personal office consists of the following blocks:

- Block number 1 - showing the doctor's full name and his schedule in the form of a calendar. The specialist has the opportunity to see his schedule for today / tomorrow, for a week and for a month. In the schedule, he can track people who signed up for him. When you click on the name of the person you need, you can view a card with data such as: name, age, gender, symptoms and allergies that they registered in the telegram bot.

- Block number 2 - contains small blocks with a request for an appointment from patients. In this block, the doctor decides to admit this patient or redirect. When you click on the "view" button, a patient card with full information appears, which he entered into the telegram bot (name, age, gender, symptoms and allergies). If the symptomatology does not fit the specialization of the doctor, it is possible for him to refer the patient to the correct doctor. If everything is in order, he can add additional comments to the available data, such as tests and confirm the patient's record to him.

- Block number 3 - shows the number and information about people registered for today and the number and information about people whom the specialist managed to receive. In this block, a specialist can regulate patients according to the accepted / did not come principle. Also, when you click on the patient's name, a patient card with complete necessary information comes out.

6.4 Colors

The main color palette is the color data below: 6.1

The main color for our project, we decided to take FFFFFF - White. If we analyze associative psychology, then in most cases the colors associated with hospitals are white as the main color and blue as an additional color [12], so you

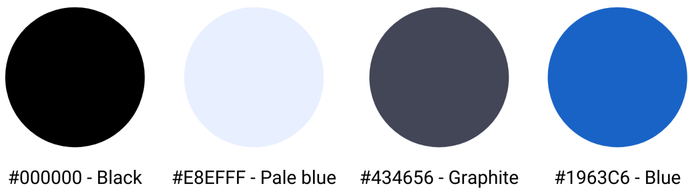


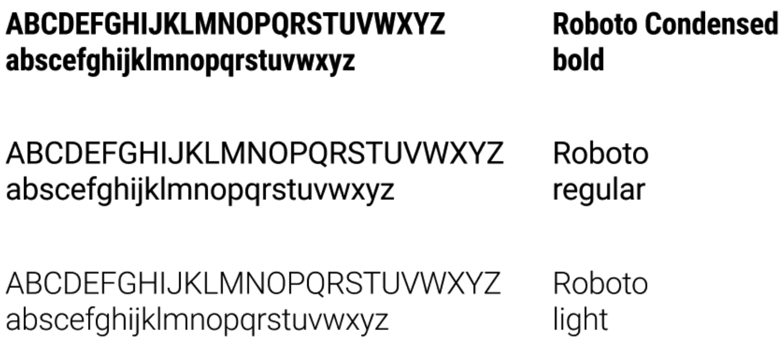
Figure 6.1: Color Palette

can come to the conclusion that the second most important color in the design of our project is 1963C6 - Blue. According to the psychology of the color blue, this color symbolizes the sky and eternity, as well as peace, relaxation and trust.

After identifying the capital colors, 3 additional colors (shown above) were generated using the site https://mycolor.space, matching the color scheme with 1963C6 - Blue.

6.5 Typography

Based on the article "The best fonts from Google Fonts - TOP 10" [10], it was decided to use 3 different variations of the font called "Roboto", since it is considered one from popular fonts on the Internet and has various variations. The following fonts were used: 6.2



ABCDEFGHIJKLMNOPQRSTUVWXYZ abscefghijklmnopqrstuvwxyz	**Roboto Condensed bold**
ABCDEFGHIJKLMNOPQRSTUVWXYZ abscefghijklmnopqrstuvwxyz	Roboto regular
ABCDEFGHIJKLMNOPQRSTUVWXYZ abscefghijklmnopqrstuvwxyz	Roboto light

Figure 6.2: Typography

Chapter 7
Conclusion

With each new day, everyone strives to improve human life, regardless of the type of activity. Striving for a comfortable life is one of the incentives for development. In the 21st century, development is happening much faster. Many people are used to this speed and do not show any desire to slow down and waste their time on things that do not keep up with their pace. And thanks to this development, people will cease to be nervous and worry about quick and accurate appointments with the right specialist.

Appendix A

Intents

A.1 Intents that were used in dialogflow

A.1.1 DefaultFallback

Если фраза не подошла ни к одному интенту. Нет Training phrases, Parameters. Responses:

1. Дефолтный ответ: Мне не ясна последняя фраза, повторите, пожалуйста.

2. Если человек отвечает на вопрос про аллергии и хронические болезни. Ваша заявка на прием принята. Ожидайте ответа доктора.

Fulfillment: true

A.1.2 DefaultWelcome

Ловит приветствие. Нет Parameters. Responses:

1. Если пользователь есть в нашей базе Здравствуйте! У меня сохранились Ваши данные с прошлого раза как Вы мне писали. Правильно: Имя: <Имя> Возраст: <Возраст> Пол: <Пол> Телефон: <Телефон>

2. Если пользователь есть в нашей базе Здравствуй! Я Мед Бот, я помогу Вам записаться к врачу. Как Вас зовут? (Ф.И.О).

Fulfillment: true

A.1.3 NameSurname

Ловит Ф.И.О Training phrases: Меня зовут Дюсенова Анель Нурслямовна Я Валиева Насиба Дюсенова Анель и т.д. Parameters: sys.person (системный параметр) Responses:

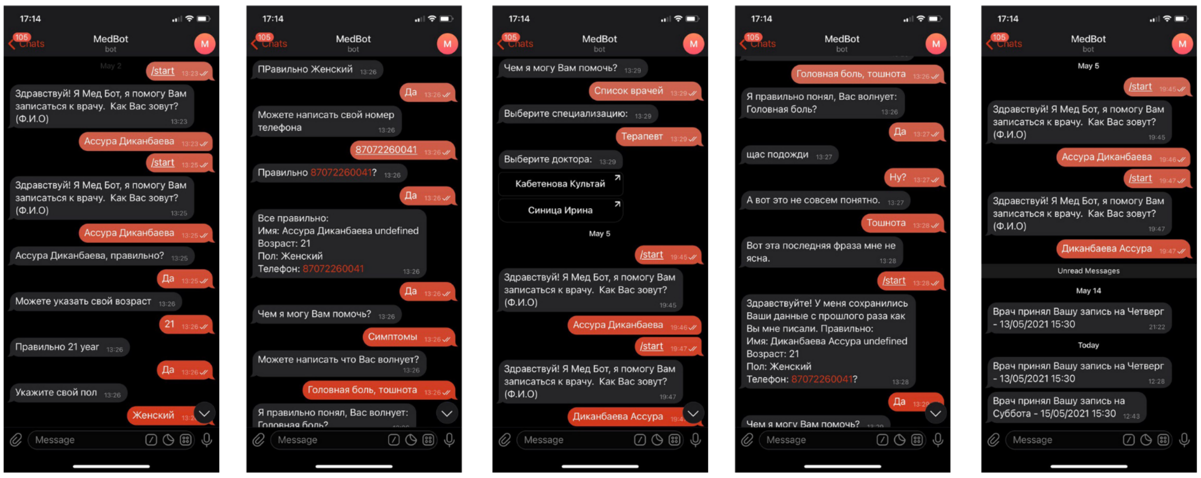
1. <Имя>, правильно? Да/Нет

Fulfillment: true

Appendix B

Telegram Bot "MedBot"

After a short thought, we came up with how this can be designed in one project, the telegram bot will collect preliminary information for the doctor, last name, first name, patronymic, year and birthday, what worries him, what symptoms he has, and will, based on this data, give advice to him, to predict an approximate diagnosis, and what he needs to do before visiting a doctor, to pass some tests, and the like. And at the end he will show him a list of doctors in the form of hyperlinks, to which you can sign up by clicking on the link to their personal website. Below, you can see how our telegram bot looks as a whole. In the first two screenshots, it collects preliminary information about the patient, then asks questions about his health, what worries him, and offers a list of doctors to whom you can sign up.



May 2

/start 13:23

Здравствуй! Я Мед Бот, я помогу Вам записаться к врачу. Как Вас зовут? (Ф.И.О) 13:23

Ассура Диканбаева 13:23

/start 13:25

Здравствуй! Я Мед Бот, я помогу Вам записаться к врачу. Как Вас зовут? (Ф.И.О) 13:25

Ассура Диканбаева 13:25

Ассура Диканбаева, правильно? 13:25

Да 13:25

Можете указать свой возраст 13:26

21 13:26

Правильно 21 year 13:26

Да 13:26

Укажите свой пол 13:26

Женский

ПРавильно Женский 13:26

Да 13:26

Можете написать свой номер телефона 13:26

87072260041 13:26

Правильно 87072260041? 13:26

Да 13:26

Все правильно:
Имя: Ассура Диканбаева undefined
Возраст: 21
Пол: Женский
Телефон: 87072260041 13:26

Да 13:26

Чем я могу Вам помочь? 13:26

Симптомы 13:26

Можете написать что Вас волнует? 13:26

Головная боль, тошнота 13:26

Я правильно понял, Вас волнует: Головная боль?

Чем я могу Вам помочь? 13:29

Список врачей 13:29

Выберите специализацию: 13:29

Терапевт 13:29

Выберите доктора: 13:29

Кабетенова Культай

Синица Ирина

May 5

/start 19:45

Здравствуй! Я Мед Бот, я помогу Вам записаться к врачу. Как Вас зовут? (Ф.И.О) 19:45

Ассура Диканбаева 19:46

/start 19:47

Здравствуй! Я Мед Бот, я помогу Вам записаться к врачу. Как Вас зовут? (Ф.И.О) 19:47

Диканбаева Ассура

Головная боль, тошнота 13:26

Я правильно понял, Вас волнует: Головная боль? 13:26

Да 13:27

щас подожди 13:27

Ну? 13:27

А вот это не совсем понятно. 13:27

Тошнота 13:28

Вот эта последняя фраза мне не ясна. 13:28

/start 13:28

Здравствуйте! У меня сохранились Ваши данные с прошлого раза как Вы мне писали. Правильно:
Имя: Диканбаева Ассура undefined
Возраст: 21
Пол: Женский
Телефон: 87072260041? 13:28

Да

Чем я могу Вам помочь?

May 5

/start 19:45

Здравствуй! Я Мед Бот, я помогу Вам записаться к врачу. Как Вас зовут? (Ф.И.О) 19:45

Ассура Диканбаева 19:46

/start 19:47

Здравствуй! Я Мед Бот, я помогу Вам записаться к врачу. Как Вас зовут? (Ф.И.О) 19:47

Диканбаева Ассура 19:47

Unread Messages

May 14

Врач принял Вашу запись на Четверг - 13/05/2021 15:30 21:22

Today

Врач принял Вашу запись на Четверг - 13/05/2021 15:30 12:28

Врач принял Вашу запись на Суббота - 15/05/2021 15:30 12:43

Figure B.1: Screen-shots of working telegram bot "MedBot"

References

[1] Rami Anwar. *WHY CHOOSE REACTJS FOR YOUR NEXT PROJECT.* URL: https://easternpeak.com/blog/why-choose-reactjs-for-your-next-project/. (accessed: 2017-08-04).

[2] *Decision tree learning.* URL: https://en.wikipedia.org/wiki/Decision_tree_learning. (accessed: 2021-05-11).

[3] *Disease-Symptom Knowledge Database.* URL: https://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html. (accessed: 2008).

[4] *Entities.* URL: https://cloud.google.com/dialogflow/es/docs/entities-overview. (accessed: 2021-05-06).

[5] the free encyclopedia From Wikipedia. *Dialogflow.* URL: https://en.wikipedia.org/wiki/Dialogflow. (accessed: 2021-02-12).

[6] *Fulfillment.* URL: https://cloud.google.com/dialogflow/es/docs/fulfillment-overview. (accessed: 2021-05-06).

[7] *Intents.* URL: https://cloud.google.com/dialogflow/es/docs/intents-overview. (accessed: 2021-05-06).

[8] Nitin Pandit. *What And Why React.js.* URL: https://www.c-sharpcorner.com/article/what-and-why-reactjs/. (accessed: 2021-02-21).

[9] Matt Rafał. *Top 6 Reasons to Choose React for Frontend Development.* URL: https://brainhub.eu/library/reasons-to-choose-react/. (accessed: 2021-05-12).

[10] UGUIDE. *Лучшие Гугл шрифты для сайта.* URL: https://uguide.ru/luchshie-google-shrifty-dlja-sajta. (accessed: 2019-10-28).

[11] Michael Wales. *3 Web Dev Careers Decoded: Front-End vs Back-End vs Full Stack.* URL: https://www.udacity.com/blog/2020/12/front-end-vs-back-end-vs-full-stack-web-developers.html. (accessed: 2020-12-8).

[12] Иван Алексеев. *Медицинские цвета – использование цветовой гаммы в дизайне сайта.* URL: https://www.motocms.com/blog/ru/tsveta-v-disaine-meditsynskogo-saita/. (accessed: 2020-01-30).