



Chatbot for Student Services Office at FILS Faculty

Team members:

Minescu Andrei

Manolescu Alexandru

Cerneanu Valentin

Alshikh Sulaiman Rim

Group: 1241EA/B

1. PROBLEM DEFINITION AND PROJECT SCOPE

For a student in Bucharest, the relationship between the university's bureaucracy and his needs during the bachelor route represent often a challenge in terms of time and technicalities rules. As a classic problem of all the romanian public institutions, the student offices from UPB has a paperwork-centric document management system. This approach has some difficulties both for the staff and the students such as lack of storage space, document transportation & editing , environmental damage and also the limitation of communication and collaboration. This limitation in collaboration between all involved parties when working with paper documents could be solved easily and in a cost effective way by introducing an automation tool for certain internal processes.

This identified problem in all the faculties from UPB is the starting point in launching a pattern chatbot with some of the most common requests in our Student Services Office at FILS.

Why? We believe that digitalization of educational institution is a powerful trend in terms of reformation and modernization of the global society

How?

By identifying:

- The main common administrative and Onboarding tasks requested by students
- Main useful information regarding admission procedures bachelor programmes
- Daily updated changes in schedule

The Student Office will be able to offer more easy guidance to the new students, updated information at the beginning of the semester and main solving procedures for the administrative student tasks.

What? An online available 24/7 system that has the goal of providing for the FILS students assistance in order to have a good and efficient interaction with the Student Office during their bachelor or master degree.

2. CURRENT APPROACHES

Artificial intelligence (AI) has influenced the way we engage in our every day activities by designing and evaluating advanced applications and devices, called intelligent agents, which can perform various functions. A chatbot is considered an artificial intelligence program and a Human–computer Interaction (HCI) model.

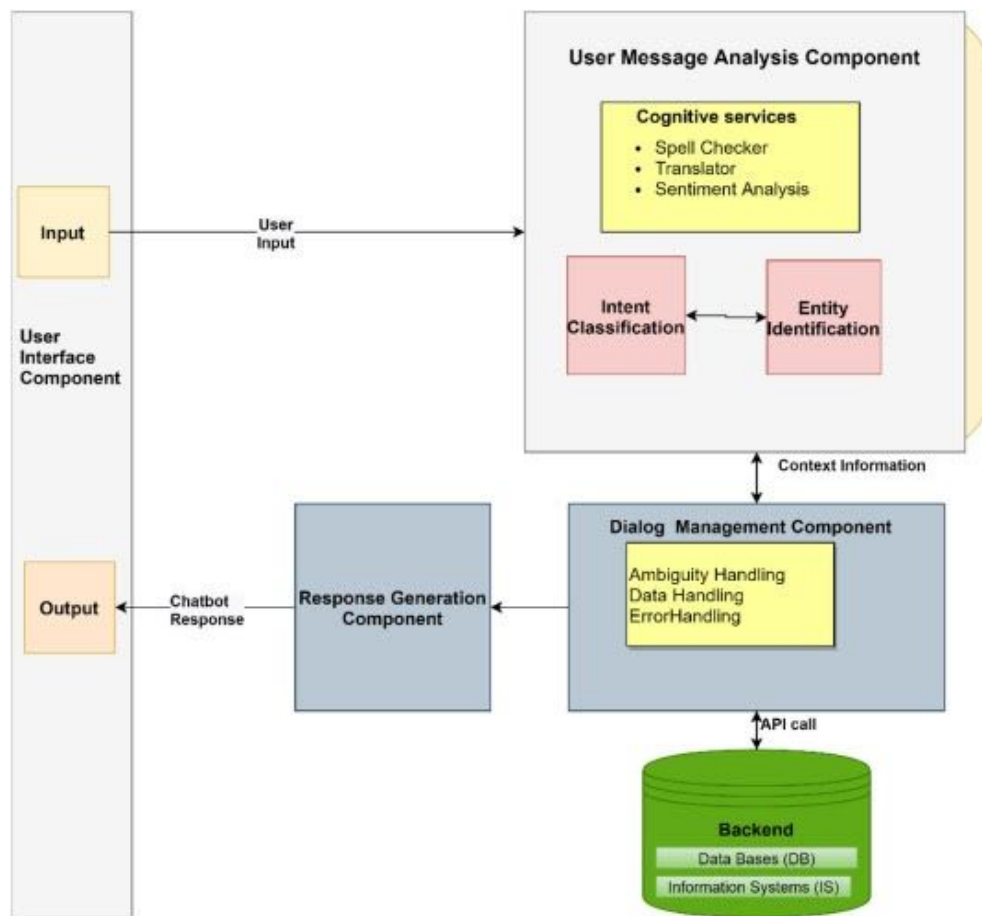
Productivity is the most important motivation for chatbot users, although different motivations include entertainment, social factors, and novelty interaction. Chatbots have become so common because:

- Reduce service costs and waiting time
- Handle many users simultaneously
- Offer users comfortable and efficient assistance when communicating with them
- Provide to users engaging answers, directly responding to their problems

The development of trust gained by a chatbot system depends on factors related to its behavior, appearance, privacy issues and protection. An important point is that in many researches it has been noticed that chatbots are four times more productive than user support staff for some specific tasks. But the crucial difference among chatbots and humans is the perception of empathy, as chatbots are less capable of conversational understanding than humans are. However, chatbots are gradually becoming more fully aware of their interlocutor's feelings. The majority of approaches in developing a chatbot today are based on two main types of chatbots, either linguistic-rule-based chatbots or machine learning-AI chatbot.

The operation of the chatbot begins when it receives the user's request through an application using text or speech input, such as a messenger application. The User Interface Controller drives the user's request to the User Message Analysis Component to find the user's intention and extracts entities following pattern matching or machine learning approaches. The Dialog Management Component controls and updates the conversation context. If the chatbot is unable to collect the necessary context information, it asks for additional context information from the user to fulfill missing entities.

The chatbot retrieves the information needed to fulfill the user's intent from the Backend through external APIs calls or Database requests. Creating the Knowledge Base of a chatbot is a necessary but often demanding and time-consuming task because it is manually developed. The Knowledge Base may also support Ontologies (Semantic Nets) like Wordnet or OpenCyc. The chatbot updates the conversation state and searches the nodes of the Knowledge Graph to make connections for the concepts used in the conversation.



The majority of approaches in developing a chatbot today are based on two main types of chatbots, either linguistic-rule-based chatbots or machine learning-AI chatbot.

Pattern matching approaches

Rule-based bots are guided by a decision tree and the user is given a set of predefined options that lead to the desired answer. These chatbots match the user input to a rule pattern and select a predefined answer from a set of responses with the use of Pattern Matching algorithms. The more extensive the database with the rules is, the more capable a chatbot is of answering the user's questions. In a rule-based chatbot for single-turn communication, the answer is selected, taking into account only the last response. Also, there is a fast response time, as a deeper syntactic or semantic examination of the input text is not performed.

Two of the most common languages for the implementation of chatbots with the pattern-matching approach are AIML and Rivescript.

Artificial Intelligence Markup Language (AIML) is based in XML, and it is open-source. AIML is the most used chatbot language mainly thanks to its usability, ease of learning and execution, and the availability of pre-authored AIML collections. The main drawback to AIML is that the author must write a pattern for every possible response of the user. However, it helps the chatbot to respond quickly and easily.

RiveScript is a line-based scripting language implementing the Knowledge Base in rule-based chatbots. It is open source and has interfaces available for many programming languages like Java and Python.

Machine learning approaches

Chatbots that are based on Machine Learning Approaches instead of Pattern Matching extract the content from the user input using Natural Language Processing (NLP). Moreover, Human-like chatbots, use a multi-turn answer selection in which every response is used as feedback to choose an answer that is normal and appropriate to the entire context. They need an extensive training set and the finding of the optimal one may constitute a crucial difficulty as available datasets may be inadequate. Often, Artificial Neural Networks (ANNs) are used for the implementation of these chatbots.

Considering these main advantages for using a rule-based chatbot, the most effective solution for our system is a

- Building such rule-based bots is much simpler and more cost effective than building AI bots
- Are faster to train and this means less expensive to be implemented
- Integrate easily with the legacy systems that already exist in the Student Office
- Available on all devices because with such bots only buttons are used
- Highly accountable and secure

3. STAKEHOLDERS AND THEIR NEEDS

Programmers, project manager, UPB employees and project team are directly involved in the project and the actual and future students are indirectly involved in the project. Students and UPB employees will be affected by the project. Students, UPB employees and HR structure of the institution will be affected by the project's outcome because the bureaucratic system might slightly change in time. There will be multiple entities that will gain or lose from the project's success: the UPB staff regarding the volume of work, students regarding time spending for administrative procedures, more time and less problems for the secretary office, the dean might have more spare time because many problems get solved faster. We identified multiple entities that want or do not want to complete the project successfully: Students would appreciate such a system and perhaps even the secretary since they would save time for obvious questions and simple data access, but the UPB employees may be reluctant to the new way of doing things.

Stakeholders include not only our system's intended users, but also any person or organization that has an effect on creating the system or with an interest in it. We have identified the following stakeholders: The UPB staff, Students, Developers

4. SYSTEM REQUIREMENTS

5. APPROACH TO SOLVING THE PROBLEM

6. CHALLENGES AND ISSUES

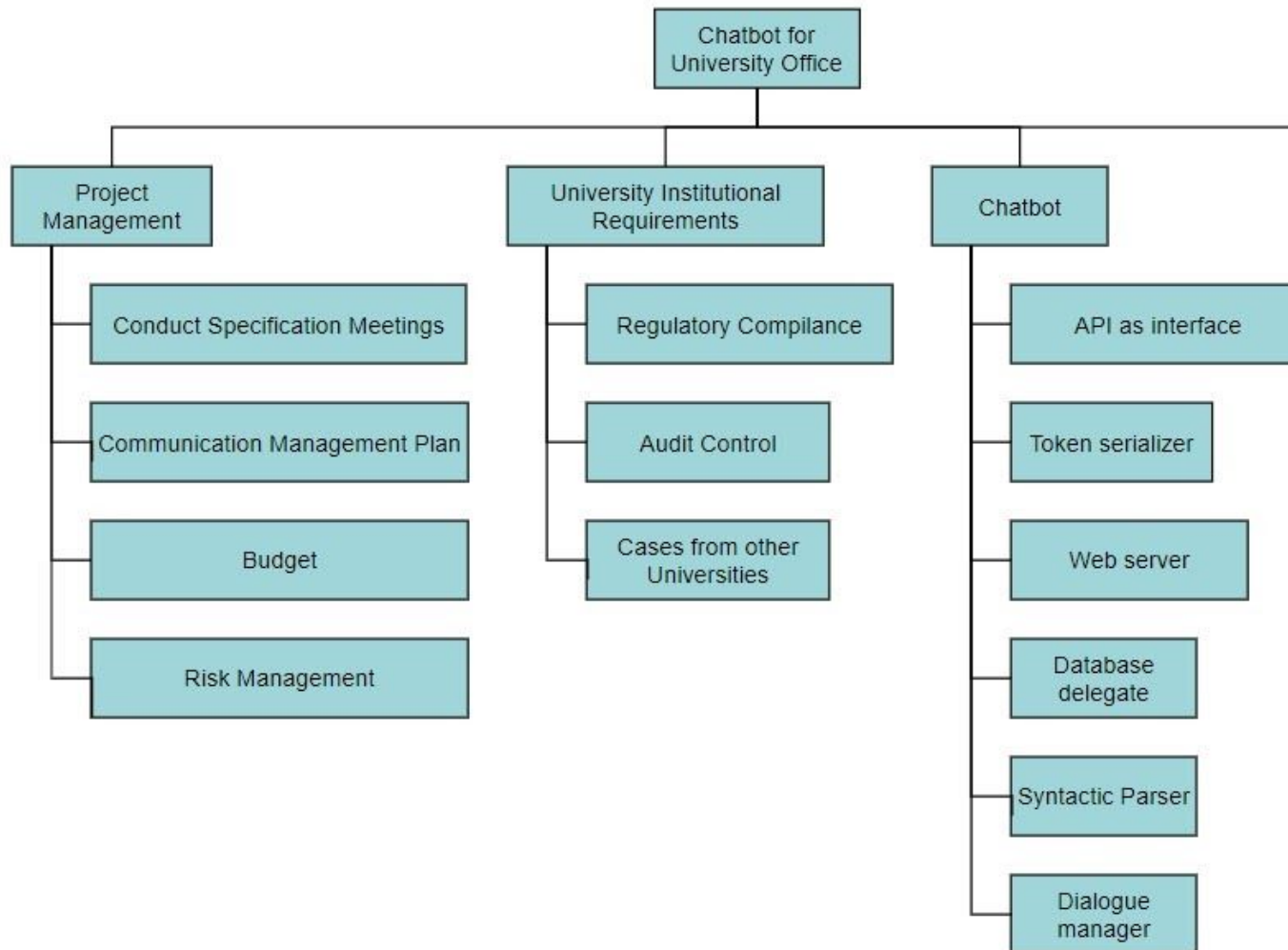
7. QUALITY ASSURANCE PLAN

8. CONCLUSIONS

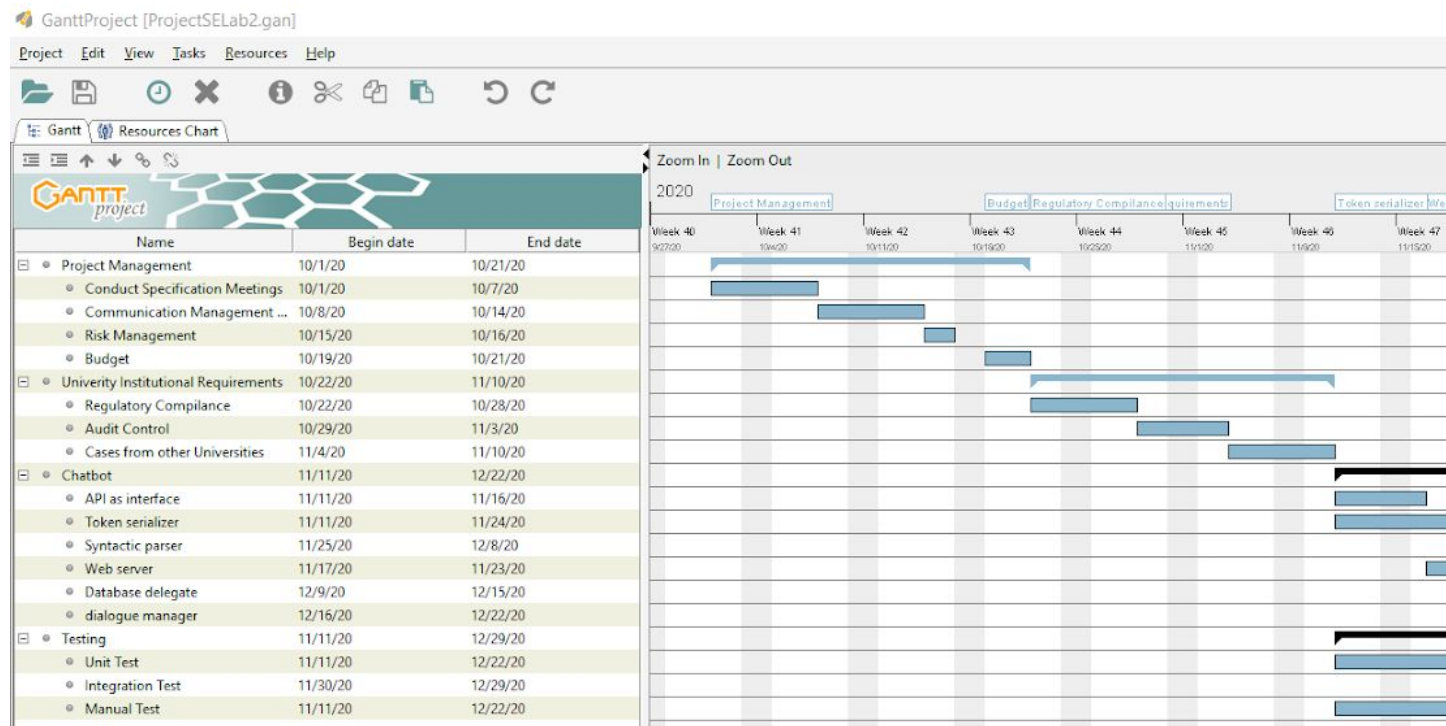
9. REFERENCES

10. Annexes

Wbs



Gantt



Database models defined within the framework's ORM


```

28     verbose_name_plural = 'Faculties'
29
30 class Department(models.Model):
31     name = models.CharField(max_length=50)
32     faculty = models.ForeignKey(Faculty, null=True, on_delete=models.SET_NULL)
33
34     def __str__(self):
35         return self.name
36
37
38 class Professor(models.Model):
39     name = models.CharField(max_length=50)
40     surname = models.CharField(max_length=50)
41     subject = models.ManyToManyField(Subject)
42     department = models.ForeignKey(Department, null=True, on_delete=models.SET_NULL)
43
44     def __str__(self):
45         try:
46             department_name = self.department.name
47         except AttributeError:
48             department_name = "No department assigned"
49
50         return "{} {} {}".format(self.name, self.surname, department_name)
51
52
53 class TeachingHour(models.Model):
54     DAY = [
55         ('Monday', 'Monday'),
56         ('Tuesday', 'Tuesday'),
57         ('Wednesday', 'Wednesday'),
58         ('Thursday', 'Thursday'),
59         ('Friday', 'Friday'),
60         ('Saturday', 'Saturday'),
61         ('Sunday', 'Sunday')
62     ]
63
64     TYPE = [
65         ('Course', 'Course'),
66         ('Laboratory', 'Laboratory'),
67         ('Seminary', 'Seminary'),
68         ('Activity', 'Activity'),
69     ]
70
71     professor = models.ForeignKey(Professor, null=True, on_delete=models.SET_NULL)
72     subject = models.ForeignKey(Subject, null=True, on_delete=models.SET_NULL)
73     day = models.CharField(null=True, max_length=20, choices=DAY, default='Monday')
74     time = models.TimeField(null=True)
75     hours = models.IntegerField(null=True, default=2)
76     ttype = models.CharField(null=True, max_length=20, choices=TYPE, default='Course')
77
78     def __str__(self):
79         return "{} - {} {} {}".format(self.subject.name, self.professor.name, self.professor.surname, self.day, self.time)

```

Example of GET request to the API endpoint which retrieves all stream languages from the faculty

GET 127.0.0.1:8000/api/v1/languages POST 127.0.0.1:8000/api/v1/add_la... + ... No Environment

127.0.0.1:8000/api/v1/languages Examples 0 BUILD

GET 127.0.0.1:8000/api/v1/professors Send Save

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies Code


Query Params

KEY	VALUE	DESCRIPTION	*** Bulk Edit
Key	Value	Description	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 24 ms Size: 593 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "name": "Teodor",
5     "surname": "Matei",
6     "department": 4,
7     "subject": [
8       1,
9       2
10    ]
11  },
12  {
13    "id": 2,
14    "name": "Lavinia",
15    "surname": "Predoi",
16    "department": 2,
17    "subject": [
18      4
19    ]
20  },
21  {
22    "id": 3,
23    "name": "Ionut",
24    "surname": "Limescu",
25    "department": 1,
26    "subject": [
27      3
```



Example of GET request which retrieves all registered professors from the database

GET 127.0.0.1:8000/api/v1/languages POST 127.0.0.1:8000/api/v1/add_la... X + ...

No Environment

Examples 0 BUILD

POST 127.0.0.1:8000/api/v1/add_language?name=limba_mea&abrv=lm Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Code

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION	*** Bulk Edit
<input checked="" type="checkbox"/> name	Bulgara		
<input checked="" type="checkbox"/> abrv	BG		
Key	Value	Description	

Body Cookies Headers (7) Test Results Status: 201 Created Time: 27 ms Size: 261 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 7,
3   "name": "Bulgara",
4   "abrv": "BG"
5 }
```

POST request which leads to the creation of a new entry in the streams list