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 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 or administrative procedure

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.

Our final scope is to create a chatbot application which will streamline the interactions between students and the staff from the Student Office. The system aims to fulfill the following objectives:

- Offering information about the schedule for each program and year of study
- Shorten the waiting time for the students which requires common information or documents
- Providing the main steps for the most common administrative procedure
- Offering the possibility to message directly the person from staff which could help the user
- Informing the current students about the exam period for the and the future student about the admission process

2. CURRENT APPROACHES TO THE PROBLEM

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. The term 'chatbot' partly overlaps with the terms 'conversational agents' and 'dialogue systems' and may refer to task-oriented as well as non-task-oriented solutions.

Productivity is the most important motivation for chatbot users, although different motivations include entertainment, social factors, and novelty interaction. The main characteristics why chatbots have become so common are:

- 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

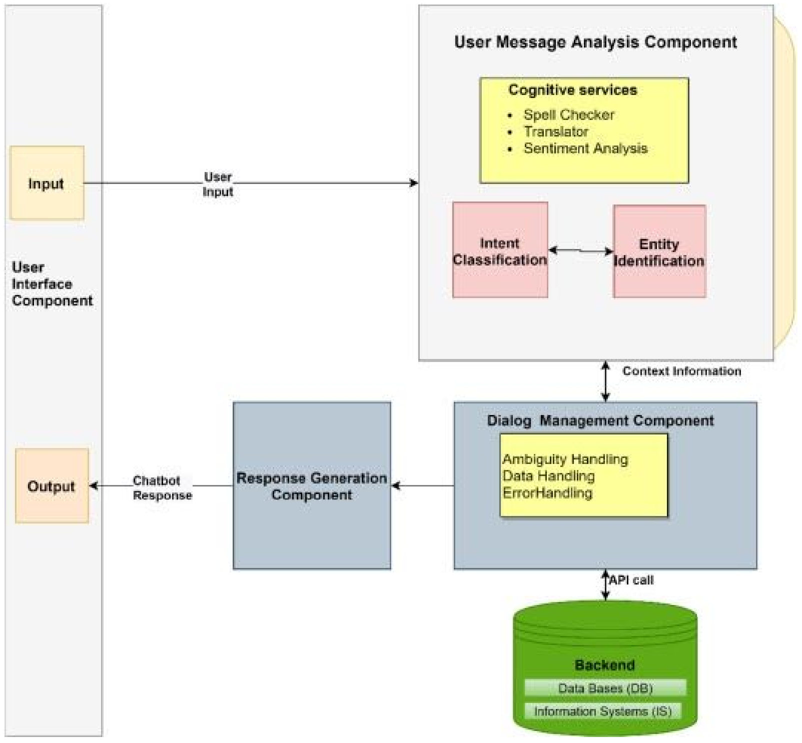
Chatbots may serve a number of purposes, such as customer service, social and emotional support, information or entertainment. The great variety of chatbots is exemplified in the BotList (https://botlist.co/), a website on which people can find chatbots for a broad range of purposes available on multiple messaging platforms.

Chatbots are seen as a promising alternative to traditional customer service. For users, conversations with these bots may feel more natural and efficient than interacting with a website where they have to search for answers to questions and usually these information are lost. Also, the system will offer a chatbot that serves as virtual assistants or stewards, helping users to perform specific tasks.

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.

It is not simple to transition from established user interfaces, such as web pages and apps, to chatbots as a common means of interacting with data and services. For example, there is a lack of knowledge regarding how customers react to the substitution of human customer service personnel. Users’ interactions with chatbots often mimic interactions between humans, but there are differences. Comparing human–human interactions to human–chatbot interactions, it is noticeable that human–chatbot interactions tend to last longer than human–human interactions between strangers and involve shorter messages, less complicated vocabulary, and more profanity.

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. For the rule-based chatbot Knowledge Base (KB) are used. 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. Often, Rule-based chatbots complete their Knowledge Base by asking the user questions and encouraging him for long conversations.

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

In addition to their significant advantages, chatbots are not free of drawbacks and threats.

Despite their spectacular development, the most common problematic situation is when chatbots fail to recognize the intent of the user. Failure to verify the intent creates frustration for the user and this could mean that the user will be disappointed by this service and will never use it again.

The main characteristics that make the interactions of chatbots more human-like could induce undesired strategic behaviors of human deceivers to hide their deception. So, using more human-like chatbots can be inefficient for applications in which it is beneficial to identify when individuals are lying.

Another drawback could be long replies where the vital information consists of one or two sentences hiding among several others. This situation may result in user discouragement and conversation vacation and this could be provided by short and clear messages. Also, chatbots require review, maintenance, and optimization in terms of their knowledge base and the way they are supposed to communicate with the users.

In order to mitigate the risks, systems could be integrated with live chat service, which will be activated for unidentified inputs.

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 the 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. The UPB staff is represented by the employees from the Student Office and the managers from the Faculty.

4. SYSTEM REQUIREMENTS

Non-functional

- **Latency** - since the servers are located in the campus in Bucharest, the response time is around a few milliseconds.
- **Interoperability** - several middlewares can be added between the chatbot software and the databases so that the chatbot can provide information for several topics
- **Reusability** - is very high because the software is pretty modular, and can be reused in other projects for other clients. This would require a permissive license for some components.
- **Backup** - backups can be provided using the NetCloud instances installed on the in-house servers. The maintenance is pretty minimal and a web GUI is provided for that.
- **Availability** - the system is very human independent because the interactions are possible using the existing data. It's supposed to run 24/7. For human interactions, the end user can be queued until a secretary is available on a business day.
- **Robustness** - The web server self-restarts after a crash, also the server boots up automatically after a power failure. The ChatBot software is managed by a parent process which makes sure a ChatBot instance always is running.
- **Useability** - It is easy for the users to achieve their objectives through using the website . The chat is accessible with one click, and since it is a javascript snippet, it can be embedded in any website.
- **Deployment** - the system is distributed, there can be one physical server for each web instance: the chatbot system and the additional web interface. The middleware acts as modular components which can be plugged in or replaced.
- **Integrability** - the ChatBot communicates with some middlewares. The middlewares communicate with the Chatbot with a specialized API and the middleware gets its data from the databases using plain queries, which depend on the databases the UPB offers access to.
- **Security** - Any request is protected and filtered by the authorization system, several roles can be created with different access and write permissions, ex: Administrator, Operator, ReadOnly etc. Also the servers have basic security measures that filter the connections and block unknown IPs.
- **Documentation** - After the UPB staff expresses their desires regarding the documentation files, they will be written during the implementation process. A QA and how-to-use sections can be provided for the students too. Actually the chatbot can provide that too, help sections can be accessed from the ChatBot by providing specific inputs (Ex: type *help* for opening a help page in the browser)
- **Capacity** - since the system is distributed, additional HDDs can increase the capacity when there is such a need, for example to archive old conversations in order to apply analytical processes on it, to find out what are the most asked questions and the hot topics the students need help with.

Functional requirements

1. **FR1**: The end user should be able to query schedules for contact time of professors, schedules for groups, library, canteen.
    - End user enters input combinations like

"schedule profesor Radulescu"
"schedule canteen rectorat"
"schedule group 1241A"
"schedule library"

- Chatbot responds with a link to a pdf or a picture, or in excel

2. **FR2**: Helps you reach out to your desired information, such as : what papers and documents students need to register for the Master studies.
   - Search for the Section you are interested in : Type the input
     "Master Studies"
   - Search for the necessary documents for the registration in Master Studies
     Enter the input :
     "Necessary documents for Master In Registration"

3. **FR3**: Search contact details for UPB employees or professors
   - enter input:
     "Find Teodor Rares"
   - ChatBot responds with contact details: full name, phone number, email, office, contact time, subjects the professor teaches, etc

4. **FR4**: Make complex sequential queries:
   - input: news

5. APPROACH TO SOLVING THE PROBLEM

The agents can communicate with the software in different ways. The data can be displayed in several applications, or can be retrieved in a very interactive way with the Chatbot. Since the same databases are used everywhere, we have no reason to fear that data can get unsynced or updated in only one place.

The system approaches the needs in several ways. First of all, a specialized server can act as an interface for the databases. A RESTful API server offers CRUD operations on the database, which can be used as an interface for a web platform, mobile application and integrations for external software and collaborators. The Chatbot application has a backend which offers a communication interface through another RESTful API. This is a powerful design, since the project developers can use it in-house for the same platform, and it's very flexible since it can be integrated with other messaging apps (Facebook for example).

The Chatbot backend has access to the same database, but the processing and visualization procedures are different for the data. Also, all these components work in a modular manner, any component can be replaced and improved while keeping the system intact. Some integration tests are mandatory for this, to make sure the system is intact in the case a module is decoupled and replaced.

Plan:

- Finding out the requirements
- Time and financial budgeting
- System design
- Components design
- Implementation and unit testing
- Integration tests and planning alternative possibilities for system use
- Future support (if that's the case)

The plan:

1) The requirements are gathered form the UPB staff, but we also need students' feedback since they will represent the main customer.
2) This is mentioned in the Gantt Chart.
3) The system designed should be modelled while abstracting the functionality of the modular components. The architecture is very data-centric so the data flow should be

sketched and followed in the system. Also, the goal of this project is to offer a high modularity since the university might expect big changes in the software in the future.

4) The components could be treated in isolation if the interfaces and data flow plan is respected. Incremental changes in the documentation should be followed by immediate changes in the implementation.
5) Unit testing is very important to ensure components' quality. The plan is to write and maintain the unit tests in parallel with the implementation work.
6) The modularity and stability of the overall system will be ensured in the future by the automated integration tests. The web framework used here offers a set of tools to do unit tests, but the biggest integration tests need to be run on while using all the components. Some virtualization environments could be used here, with Docker Swarm or in an AWS environment, to simulate the interaction of the system deployed in production.

6. CHALLENGES AND ISSUES

One main risk identified in the maintenance stage is that the capabilities of the chatbot can become outdated. Without maintaining the chatbot updated with the most updated information and administrative news there is a high risk of leaving users not finding the right piece of information. The software might get obsolete because of the reluctance of UPB staff to update its data and to sync it. A poorly maintained chatbot, no matter how great at the start, will inevitably lead to bad user experiences and low reuse rates. The requirements might slightly adapt in time since a trade-off might be needed to solve this problem. For example, if UPB does not give full access to databases to the chatbot system, redundancy should be reduced so that UPB staff won't tend to update only their previous system. As a direct result, if the chatbot isn't up to date with you, it'll provide out of date information, and annoy users by wasting their time.

7. QUALITY ASSURANCE PLAN

The quality plan for the chatbot system will define:

- Objectives to be attained
- Allocation of responsibilities, authority, and resources during the different phases of the process or project
- Specific documented standards, practices, procedures, and instructions to be applied
- Suitable testing, inspection, examination, and audit programs at appropriate stages

Customer feedback metrics are measurement units that will let the administrator of the chatbot system to quantitatively assess the online feedback. The main Metrics for tracking and assessing the status of a specific process are:

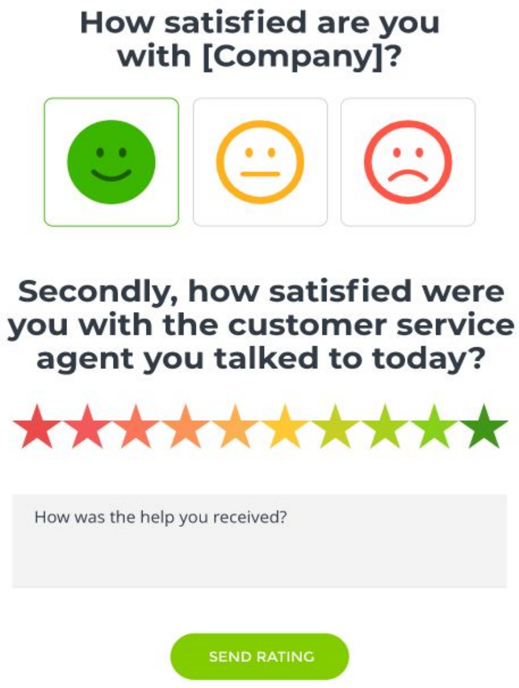
Net promoter score (NPS)

An NPS is calculated with a 0–10 scale when users are asked a question along the lines of: 'how likely are you to recommend the use of this chatbot to other students?". This feature will appear to the user on the end of their interaction with the chabot, after the requested information has been provided and the user asked that has no more other questions. Adding a feedback mechanism to a chatbot is essential because in this way the UPB Staff can understand how the system is working and what should be improved. Moreover, with this metric it could be analyzed if the investment in this system brings the desired results.

Customer Satisfaction Score (CSAT)

CSAT is a CX metric that can directly measure customer's satisfaction levels. So a survey can be sent to the user after he finishes to see how well does it work and if there is a need for any kind of improvements

Functionality : CSAT surveys usually feature a question asking the user on how satisfied they are with a certain service or interaction with your system. For example it can be done by leaving a rating review on how satisfied is the user


How satisfied are you with [Company]?

Secondly, how satisfied were you with the customer service agent you talked to today?

How was the help you received?

8. CONCLUSIONS

As we saw, our project Chatbot for Student Services Office at FILS Faculty is a complex system, which success is dependent on multiple tasks. First, the planning part is the key part for having a steady and stable implementation, like the old greek saying “Slow And Steady Wins The Race”. The second part is the actual implementation of the project. This part should be carefully supervised by the project manager. The development phase should be run in AGILE methodology with two weeks sprints. For the best results the development team should have weekly meetings for Sprint Planning, Sprint Review and Sprint Refinement.

As we saw during the implementation phase we have two different approaches. The first approach is to build the whole system from scratch. This approach has the advantage that it will have the desired features and nothing more. The disadvantage is the long time needed for development. The second approach was configuring a web ChatBot and paying a subscription.

As we discovered there are quite a lot of different web ChatBots, but the quality of them is not a certainty. We tried two products, but they weren’t as good as we needed to. Only the third product it turned out to be suitable for our needs. So we see a real need of developing a real, robust and efficient web chat bot.

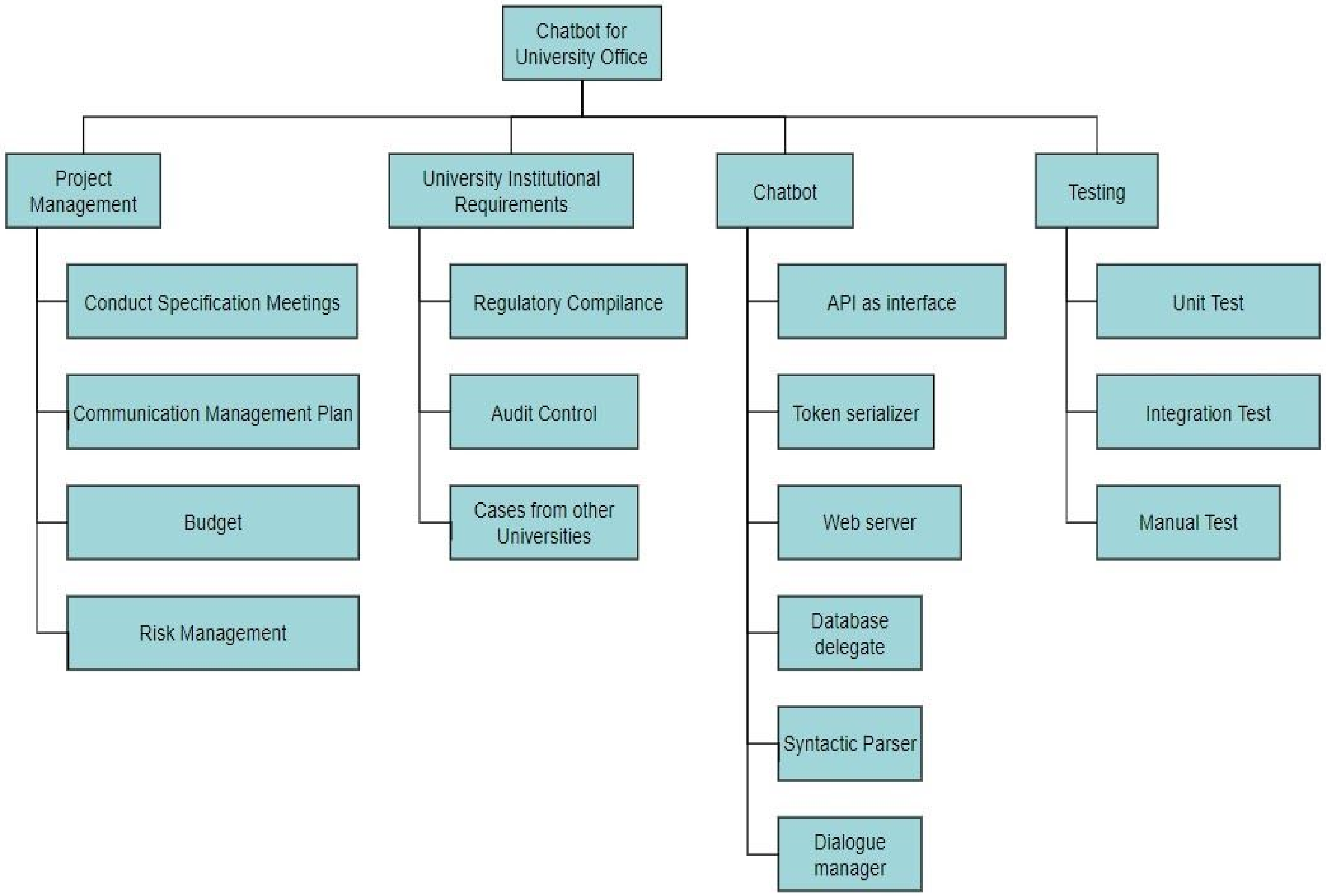
Regarding the need, there is no doubt that a university should have a web chat bot for the site. The students are accustomed to using technologies like chat bots. A good chat bot can save a lot of time for the university staff but also for the students. Students will have all the information they need without wasting staff time answering the same questions. Also a good chat bot can store what the students are asking and with an analytics engine the university staff can see what problems and concerns students have.

9. REFERENCES

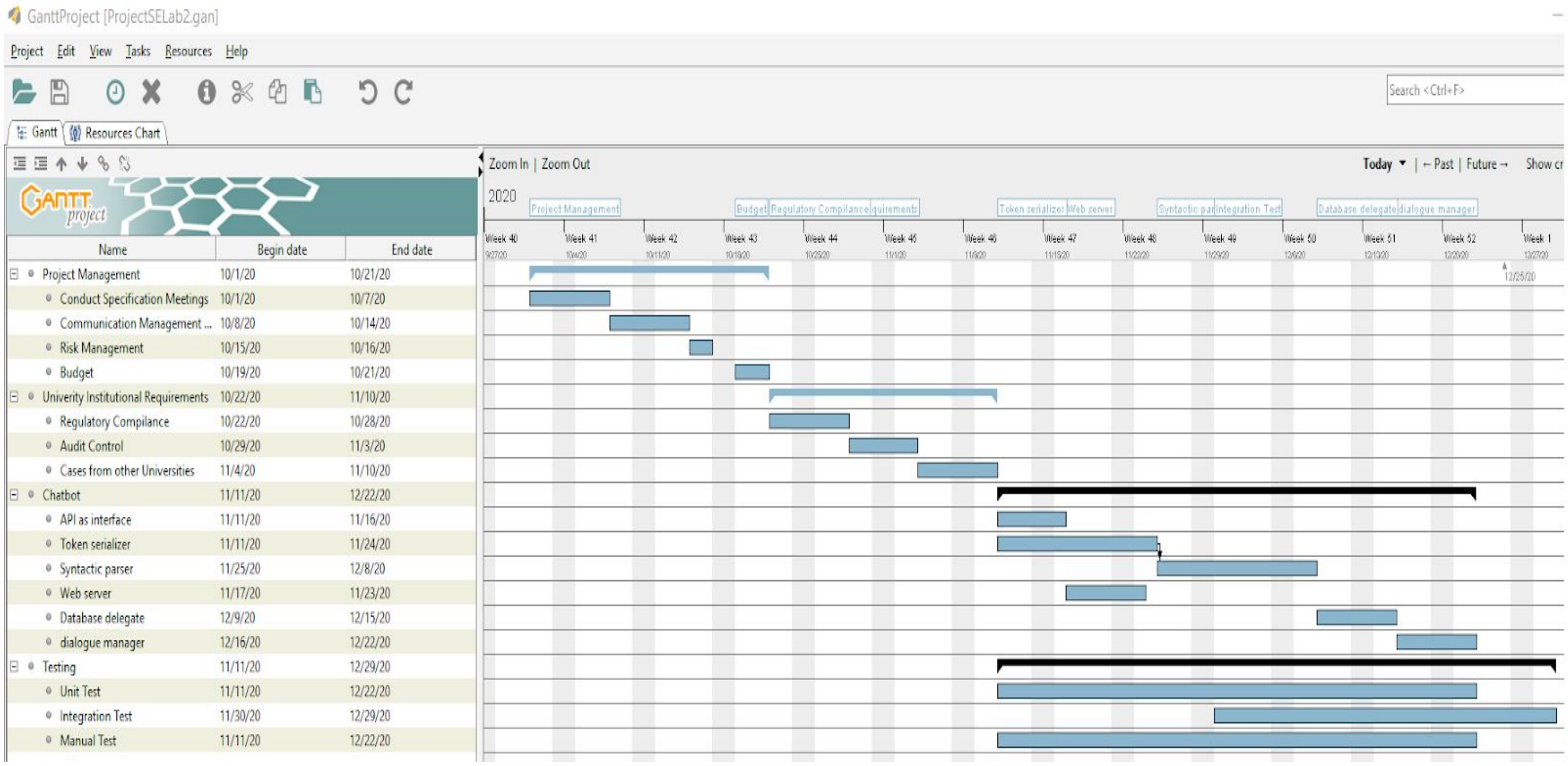
1. Petter Bae Brandtzaeg and Asbjørn Følstad (November 2017). “Why people use chatbots”. Oslo, Norway. Retrieved from: https://www.researchgate.net/publication/318776998_Why_People_Use_Chatbots
2. Marita Skjuvea and Asbjørn Følstadb (January 2021). “My Chatbot Companion - a Study of Human-Chatbot Relationships”. Retrieved from: https://www.sciencedirect.com/science/article/pii/S1071581921000197
3. Eleni Adamopoulou and Lefteris Moussiades (December 2020). “Chatbots: History, technology, and applications”. Department of Computer Science, International Hellenic University, Kavala, Greece. Retrieved from: https://www.sciencedirect.com/science/article/pii/S2666827020300062
4. Pavel Smutny and Petra Schreiberova (July 2020). “Chatbots for learning: A review of educational chatbots for the Facebook Messenger”. VSB – Technical University of Ostrava, Czech Republic . Retrieved from: https://www.sciencedirect.com/science/article/pii/S0360131520300622
5. Irina Dokukinaa and Julia Gumanova (2020). “The rise of chatbots – new personal assistants in foreign language learning”. Procedia Computer Science Volume 169, Pages 542-546. Retrieved from: https://www.sciencedirect.com/science/article/pii/S1877050920303355

10. Annexes

-Work breakdown structure (WBS)-

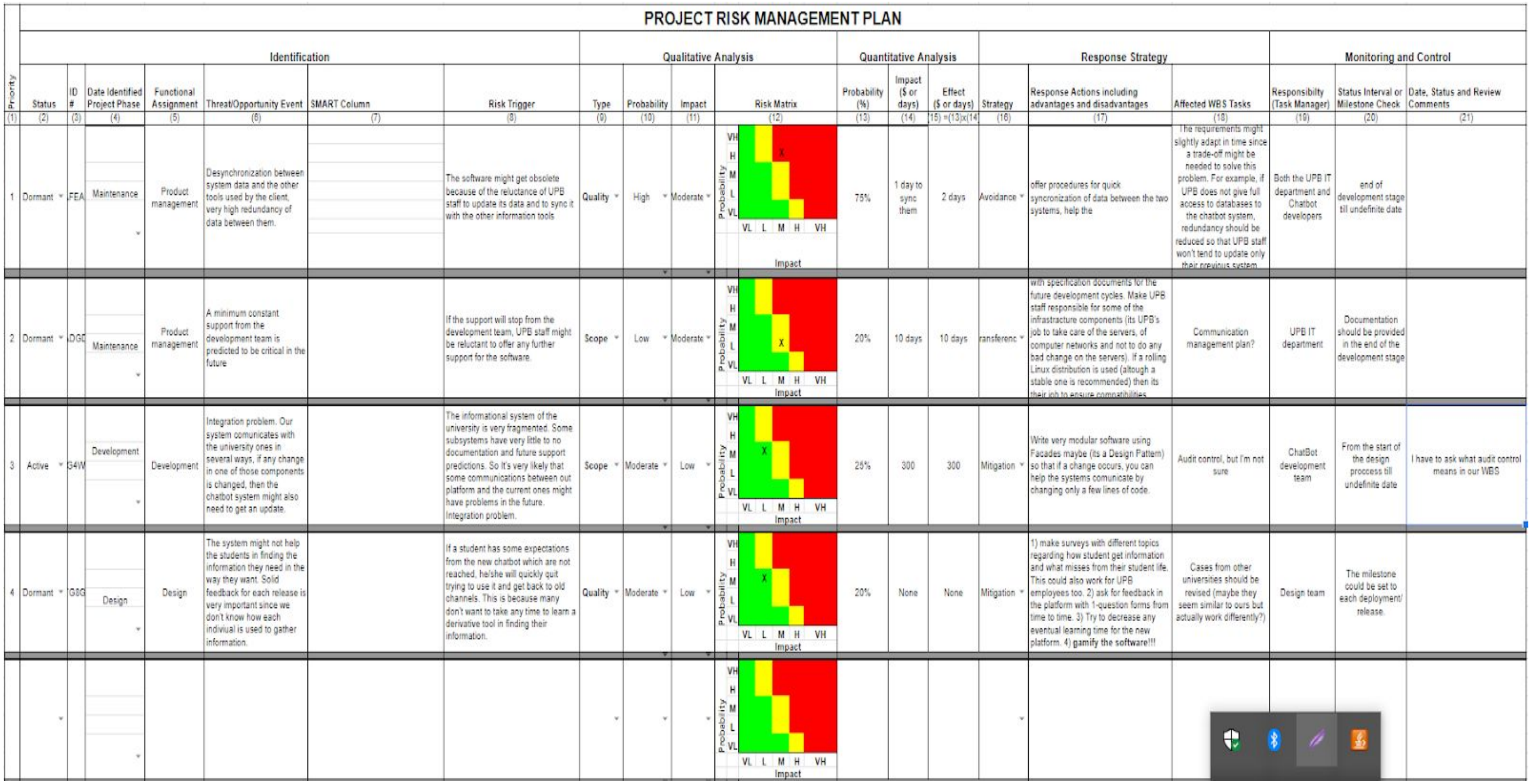


-Gantt Chart-



Name	Begin date	End date
Project Management	10/1/20	10/21/20
Conduct Specification Meetings	10/1/20	10/7/20
Communication Management ...	10/8/20	10/14/20
Risk Management	10/15/20	10/16/20
Budget	10/19/20	10/21/20
Univerity Institutional Requirements	10/22/20	11/10/20
Regulatory Compilance	10/22/20	10/28/20
Audit Control	10/29/20	11/3/20
Cases from other Universities	11/4/20	11/10/20
Chatbot	11/11/20	12/22/20
API as interface	11/11/20	11/16/20
Token serializer	11/11/20	11/24/20
Syntactic parser	11/25/20	12/8/20
Web server	11/17/20	11/23/20
Database delegate	12/9/20	12/15/20
dialogue manager	12/16/20	12/22/20
Testing	11/11/20	12/29/20
Unit Test	11/11/20	12/22/20
Integration Test	11/30/20	12/29/20
Manual Test	11/11/20	12/22/20

-Risk management plan-

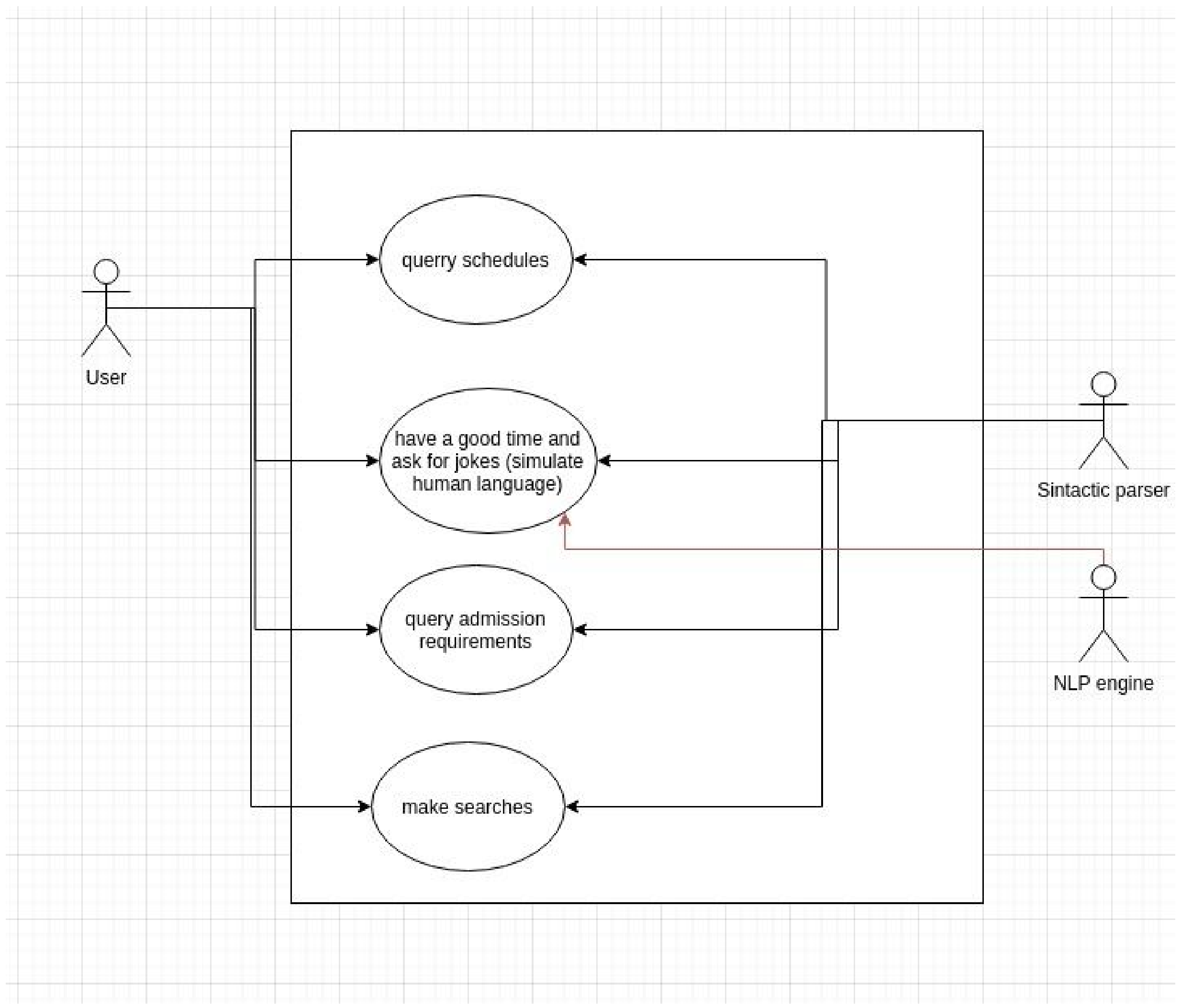


PROJECT RISK MANAGEMENT PLAN

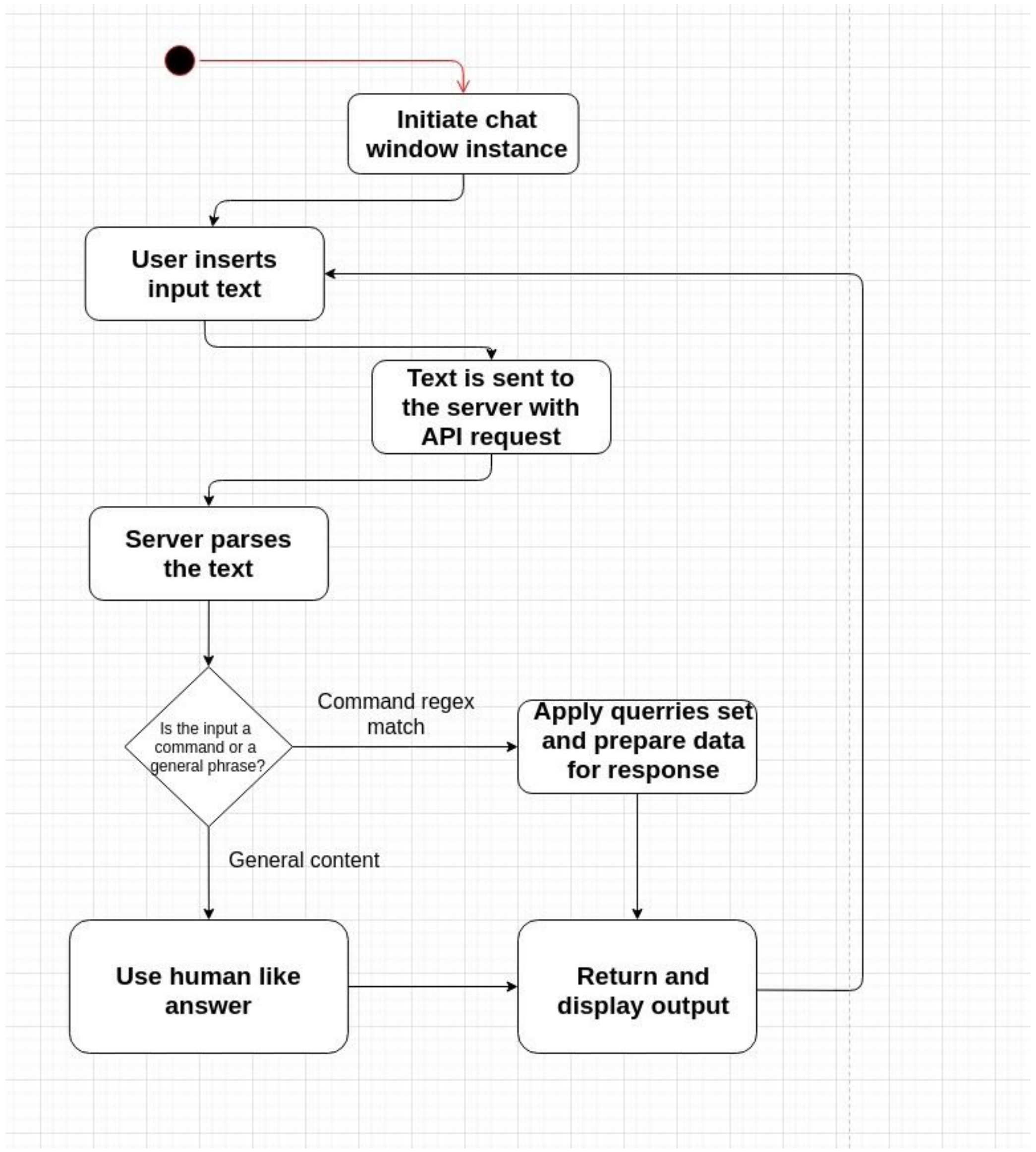
Priority	Identification							Qualitative Analysis				Quantitative Analysis			Response Strategy			Monitoring and Control		
	Status	ID #	Date Identified Project Phase	Functional Assignment	Threat/Opportunity Event	SMART Column	Risk Trigger	Type	Probability	Impact	Risk Matrix	Probability (%)	Impact ($ or days)	Effect ($ or days)	Strategy	Response Actions including advantages and disadvantages	Affected WBS Tasks	Responsibilty (Task Manager)	Status Interval or Milestone Check	Date, Status and Review Comments
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15) =(13)x(14)	(16)	(17)	(18)	(19)	(20)	(21)
1	Dormant	FEA	Maintenance	Product management	Desynchronization between system data and the other tools used by the client, very high redundancy of data between them.		The software might get obsolete because of the reluctance of UPB staff to update its data and to sync it with the other information tools	Quality	High	Moderate	X	75%	1 day to sync them	2 days	Avoidance	offer procedures for quick syncronization of data between the two systems, help the	The requirements might slightly adapt in time since a trade-off might be needed to solve this problem. For example, if UPB does not give full access to databases to the chatbot system, redundancy should be reduced so that UPB staff won't tend to update only their previous system	Both the UPB IT department and Chatbot developers	end of development stage till undefinite date	
2	Dormant	DGD	Maintenance	Product management	A minimum constant support from the development team is predicted to be critical in the future		If the support will stop from the development team, UPB staff might be reluctant to offer any further support for the software.	Scope	Low	Moderate	X	20%	10 days	10 days	ransferenc	with specification documents for the future development cycles. Make UPB staff responsible for some of the infrastracture components (its UPB's job to take care of the servers, of computer networks and not to do any bad change on the servers). If a rolling Linux distribution is used (altough a stable one is recommended) then its their job to ensure compatibilities	Communication management plan?	UPB IT department	Documentation should be provided in the end of the development stage	
3	Active	G4W	Development	Development	Integration problem. Our system comunicates with the university ones in several ways, if any change in one of those components is changed, then the chatbot system might also need to get an update.		The informational system of the university is very fragmented. Some subsystems have very little to no documentation and future support predictions. So It's very likely that some communications between out platform and the current ones might have problems in the future. Integration problem.	Scope	Moderate	Low	X	25%	300	300	Mitigation	Write very modular software using Facades maybe (its a Design Pattern) so that if a change occurs, you can help the systems comunicate by changing only a few lines of code.	Audit control, but I'm not sure	ChatBot development team	From the start of the design proccess till undefinite date	I have to ask what audit control means in our WBS
4	Dormant	G8G	Design	Design	The system might not help the students in finding the information they need in the way they want. Solid feedback for each release is very important since we don't know how each indiviual is used to gather information.		If a student has some expectations from the new chatbot which are not reached, he/she will quickly quit trying to use it and get back to old channels. This is because many don't want to take any time to learn a derivative tool in finding their information.	Quality	Moderate	Low	X	20%	None	None	Mitigation	1) make surveys with different topics regarding how student get information and what misses from their student life. This could also work for UPB employees too. 2) ask for feedback in the platform with 1-question forms from time to time. 3) Try to decrease any eventual learning time for the new platform. 4) **gamify the software!!!**	Cases from other universities should be revised (maybe they seem similar to ours but actually work differently?)	Design team	The milestone could be set to each deployment/ release.	

SysML diagrams-behavior diagrams

-Use Case Diagram-

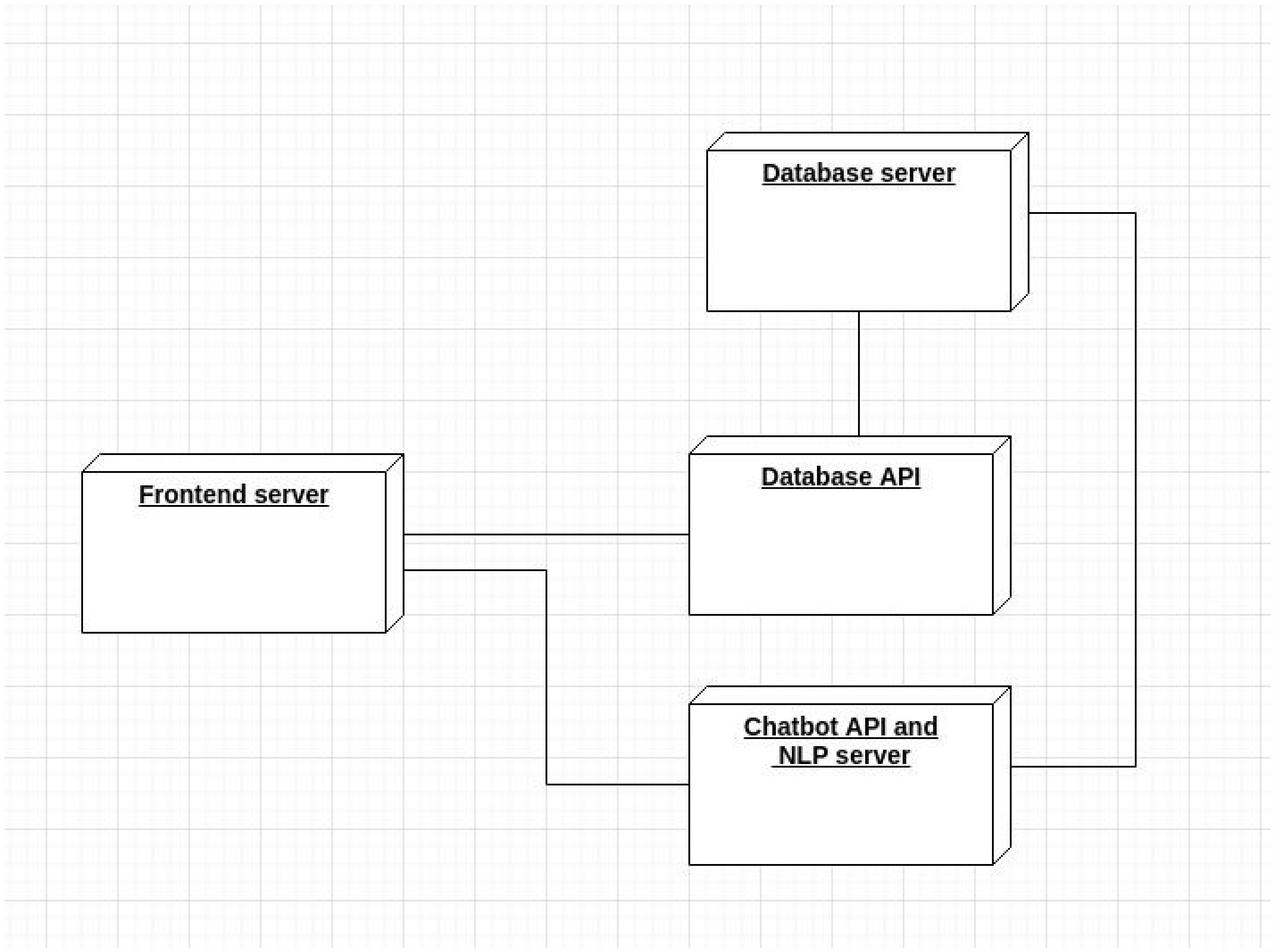

User
querry schedules
have a good time and ask for jokes (simulate human language)
query admission requirements
make searches
Sintactic parser
NLP engine

-Activity Diagram-

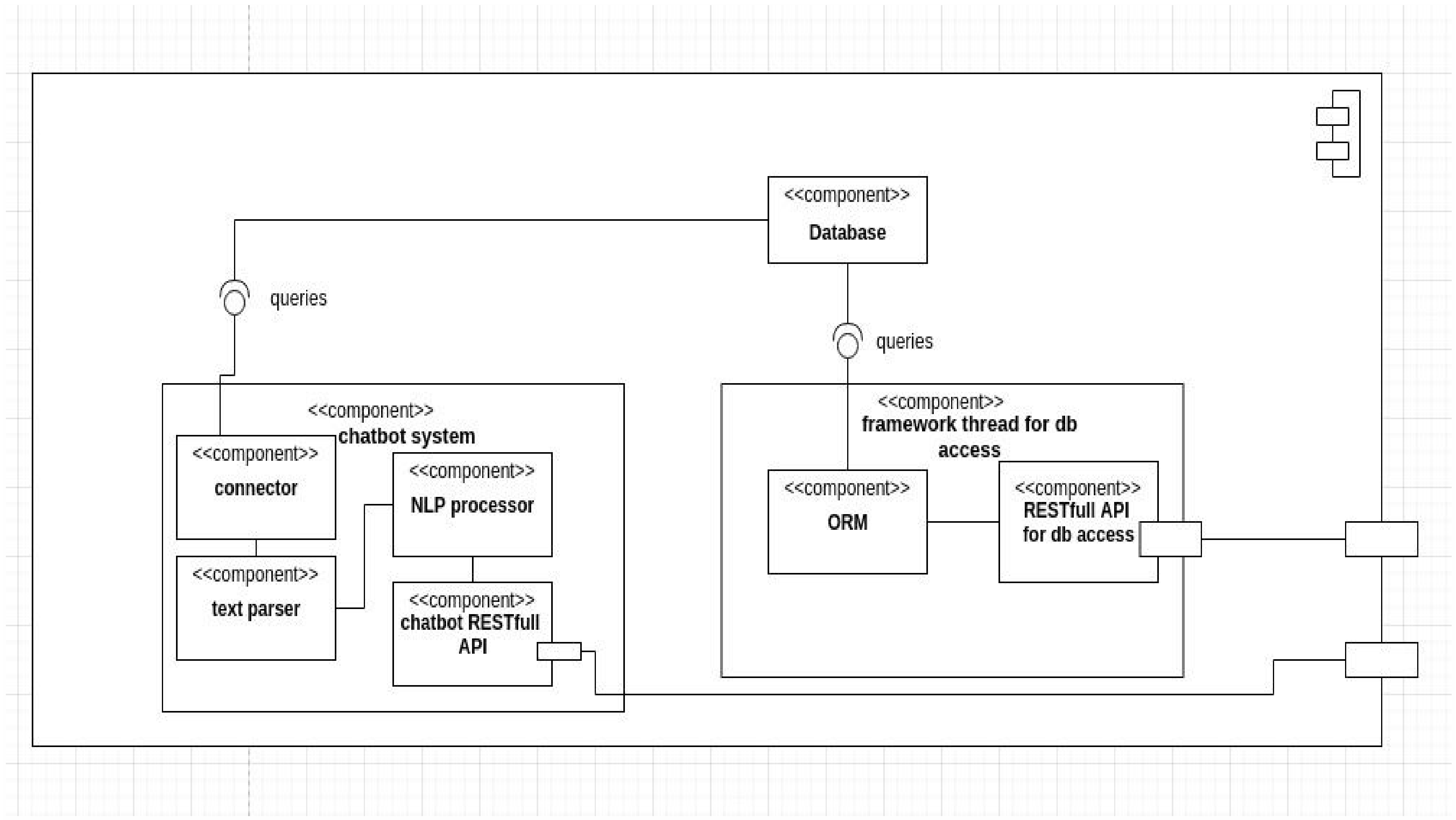

Initiate chat window instance
User inserts input text
Text is sent to the server with API request
Server parses the text
Is the input a command or a general phrase?
Command regex match
Apply querries set and prepare data for response
General content
Use human like answer
Return and display output

***SysML diagrams*-Structure diagrams**

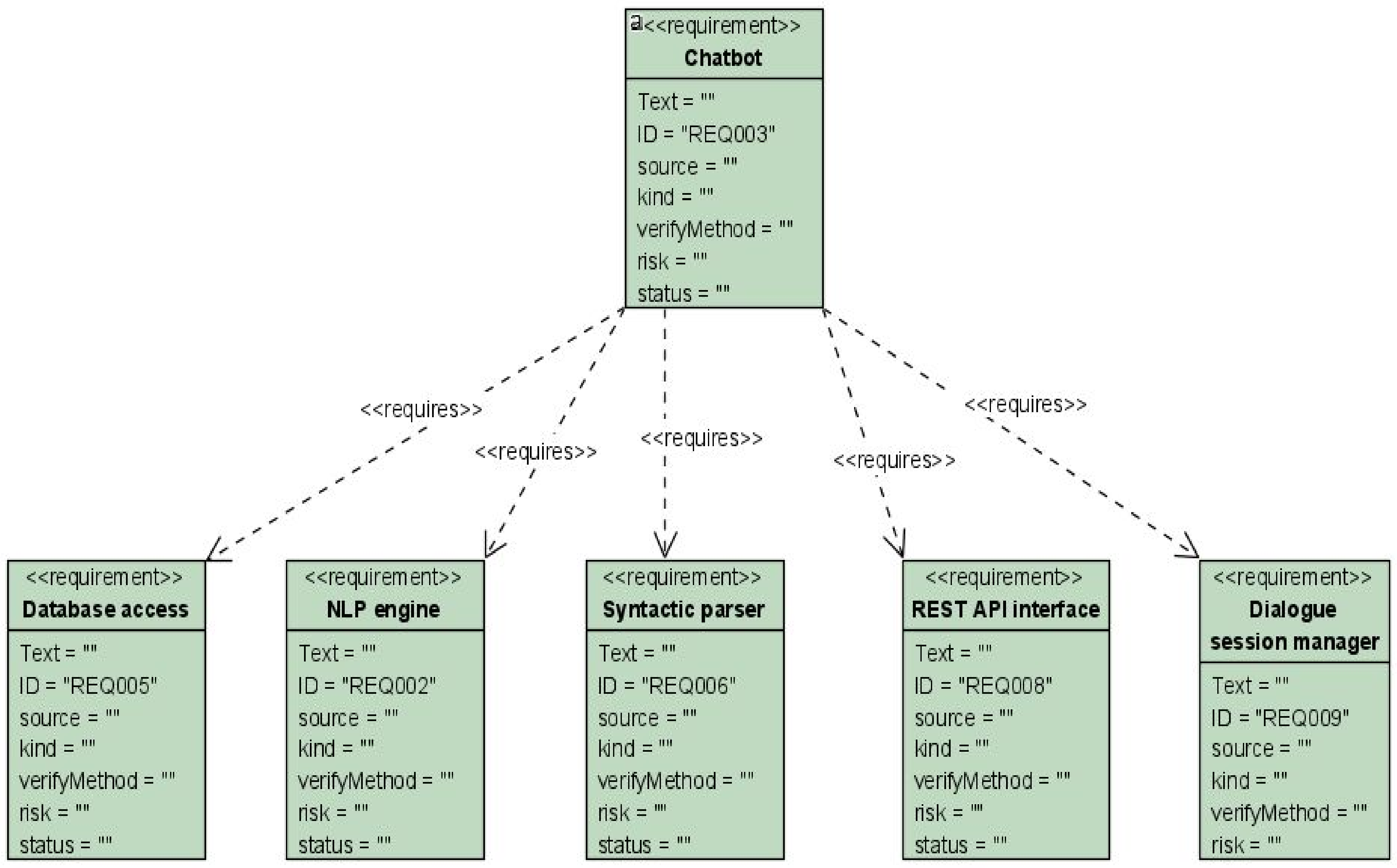
-Deployment Diagram-



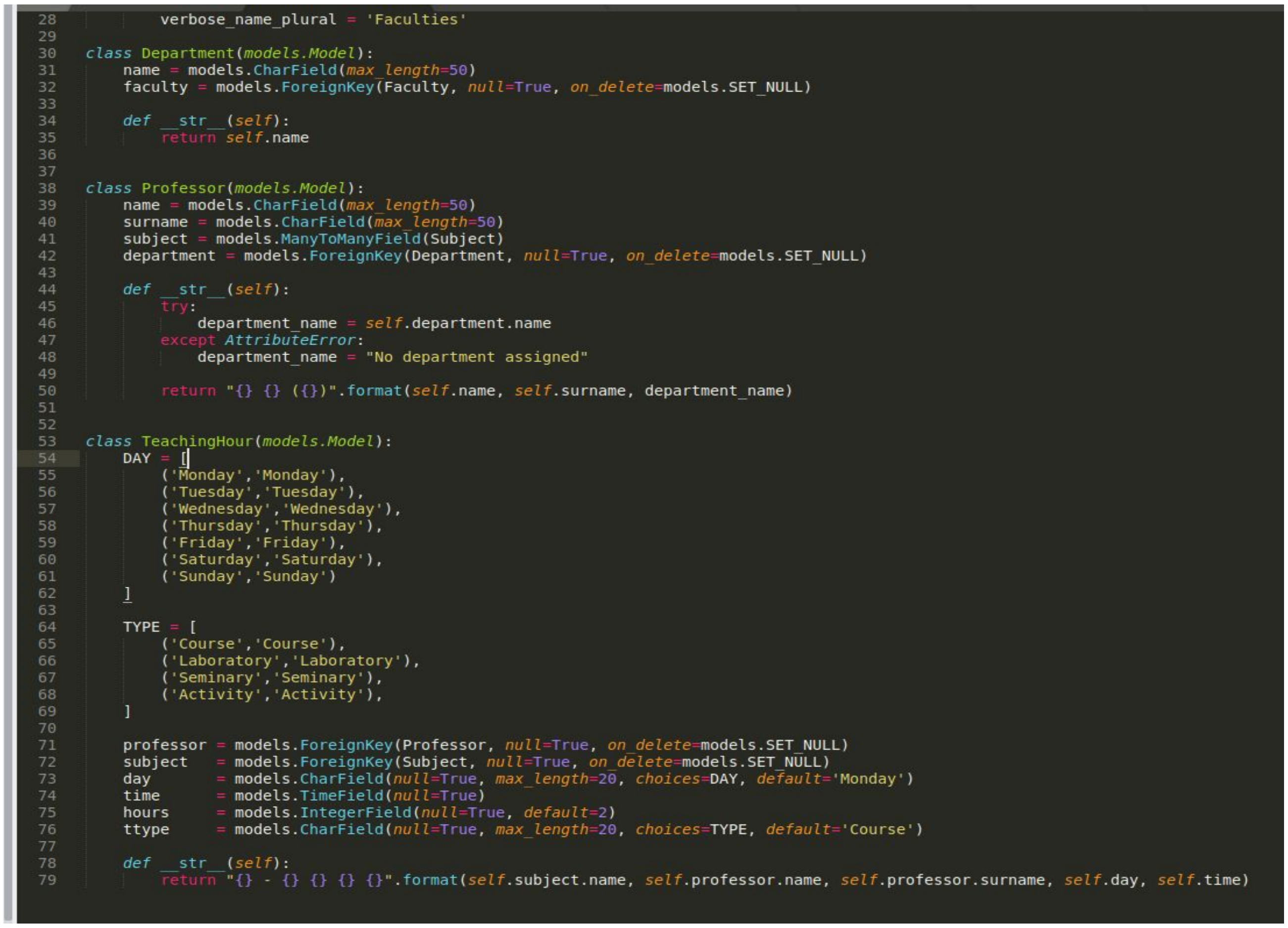
-Component Diagram-



-Requirement Diagram-

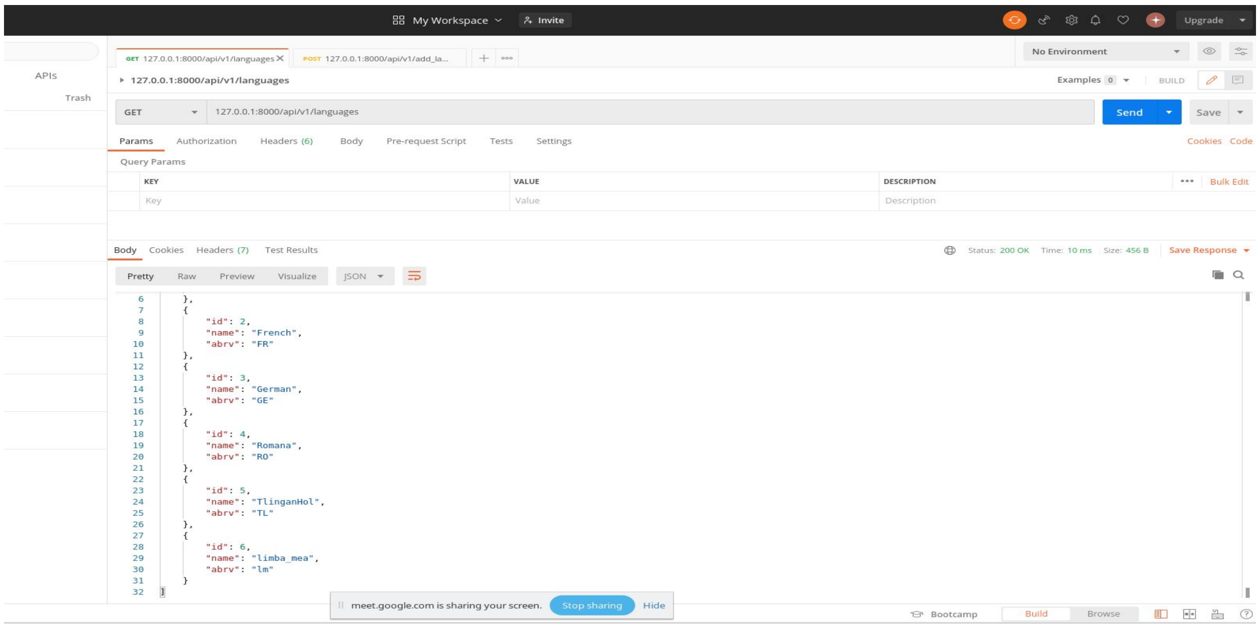


Database models defined within the framework's ORM



```python
        verbose_name_plural = 'Faculties'

class Department(models.Model):
    name = models.CharField(max_length=50)
    faculty = models.ForeignKey(Faculty, null=True, on_delete=models.SET_NULL)

    def __str__(self):
        return self.name


class Professor(models.Model):
    name = models.CharField(max_length=50)
    surname = models.CharField(max_length=50)
    subject = models.ManyToManyField(Subject)
    department = models.ForeignKey(Department, null=True, on_delete=models.SET_NULL)

    def __str__(self):
        try:
            department_name = self.department.name
        except AttributeError:
            department_name = "No department assigned"

        return "{} {} ({})".format(self.name, self.surname, department_name)


class TeachingHour(models.Model):
    DAY = [
        ('Monday','Monday'),
        ('Tuesday','Tuesday'),
        ('Wednesday','Wednesday'),
        ('Thursday','Thursday'),
        ('Friday','Friday'),
        ('Saturday','Saturday'),
        ('Sunday','Sunday')
    ]

    TYPE = [
        ('Course','Course'),
        ('Laboratory','Laboratory'),
        ('Seminary','Seminary'),
        ('Activity','Activity'),
    ]

    professor  = models.ForeignKey(Professor, null=True, on_delete=models.SET_NULL)
    subject    = models.ForeignKey(Subject, null=True, on_delete=models.SET_NULL)
    day        = models.CharField(null=True, max_length=20, choices=DAY, default='Monday')
    time       = models.TimeField(null=True)
    hours      = models.IntegerField(null=True, default=2)
    ttype      = models.CharField(null=True, max_length=20, choices=TYPE, default='Course')

    def __str__(self):
        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...

127.0.0.1:8000/api/v1/languages

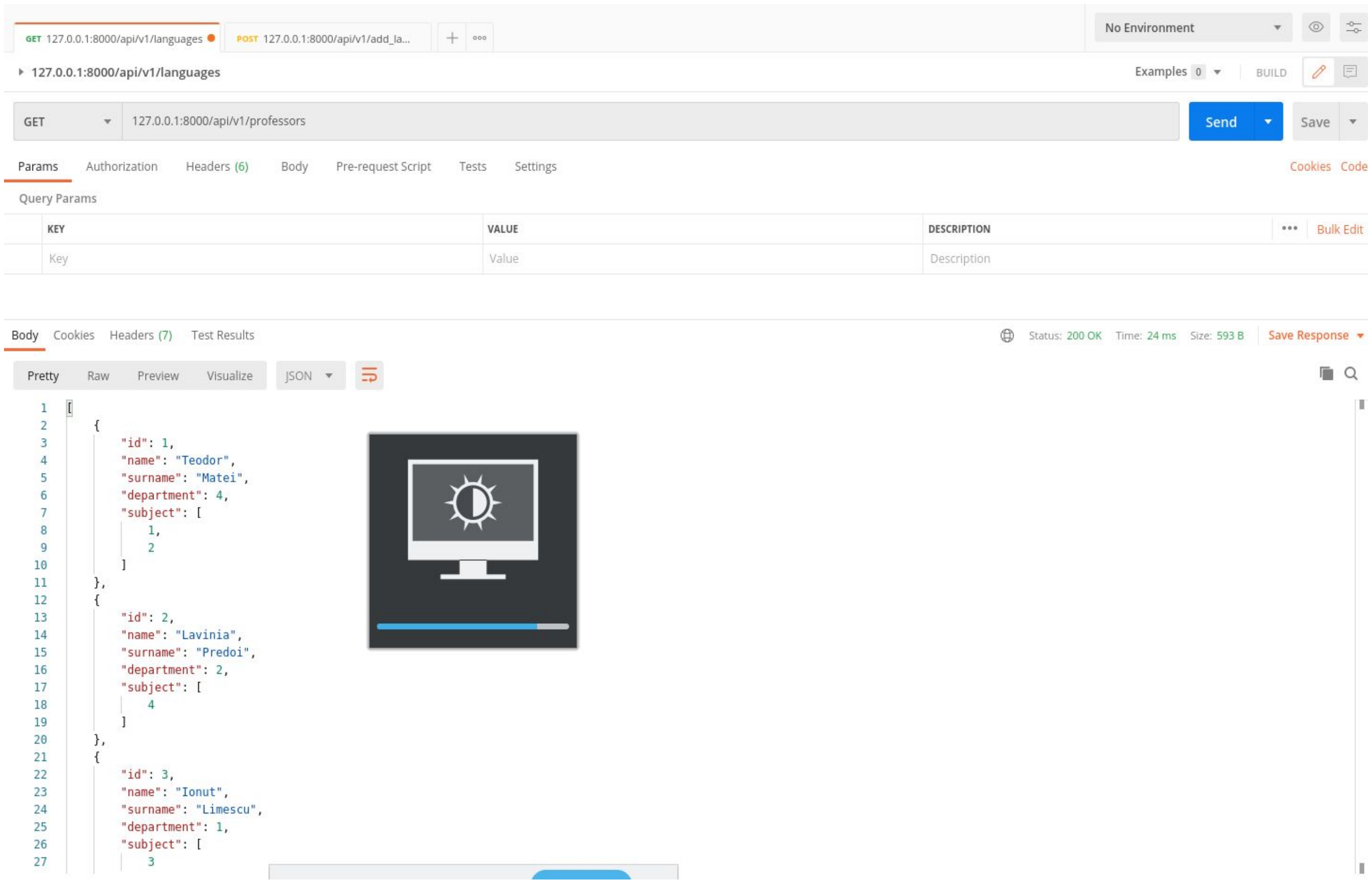
GET 127.0.0.1:8000/api/v1/languages

Status: 200 OK Time: 10 ms Size: 456 B

```
    },
    {
        "id": 2,
        "name": "French",
        "abrv": "FR"
    },
    {
        "id": 3,
        "name": "German",
        "abrv": "GE"
    },
    {
        "id": 4,
        "name": "Romana",
        "abrv": "RO"
    },
    {
        "id": 5,
        "name": "TlinganHol",
        "abrv": "TL"
    },
    {
        "id": 6,
        "name": "limba_mea",
        "abrv": "lm"
    }
]
```

meet.google.com is sharing your screen. Stop sharing Hide

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



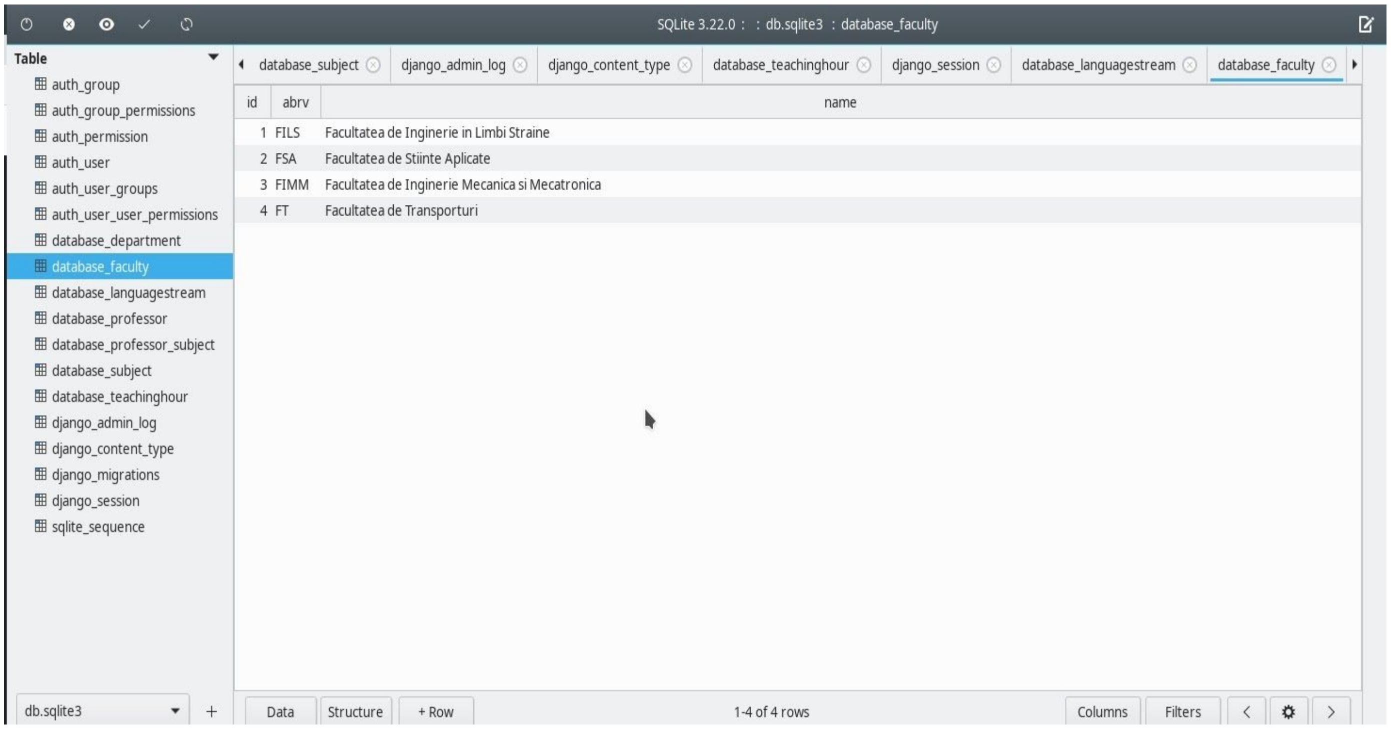
GET 127.0.0.1:8000/api/v1/professors

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 24 ms Size: 593 B

```
[
    {
        "id": 1,
        "name": "Teodor",
        "surname": "Matei",
        "department": 4,
        "subject": [
            1,
            2
        ]
    },
    {
        "id": 2,
        "name": "Lavinia",
        "surname": "Predoi",
        "department": 2,
        "subject": [
            4
        ]
    },
    {
        "id": 3,
        "name": "Ionut",
        "surname": "Limescu",
        "department": 1,
        "subject": [
            3
```

Database seen in TablePlus



SQLite 3.22.0 : : db.sqlite3 : database_faculty

Table

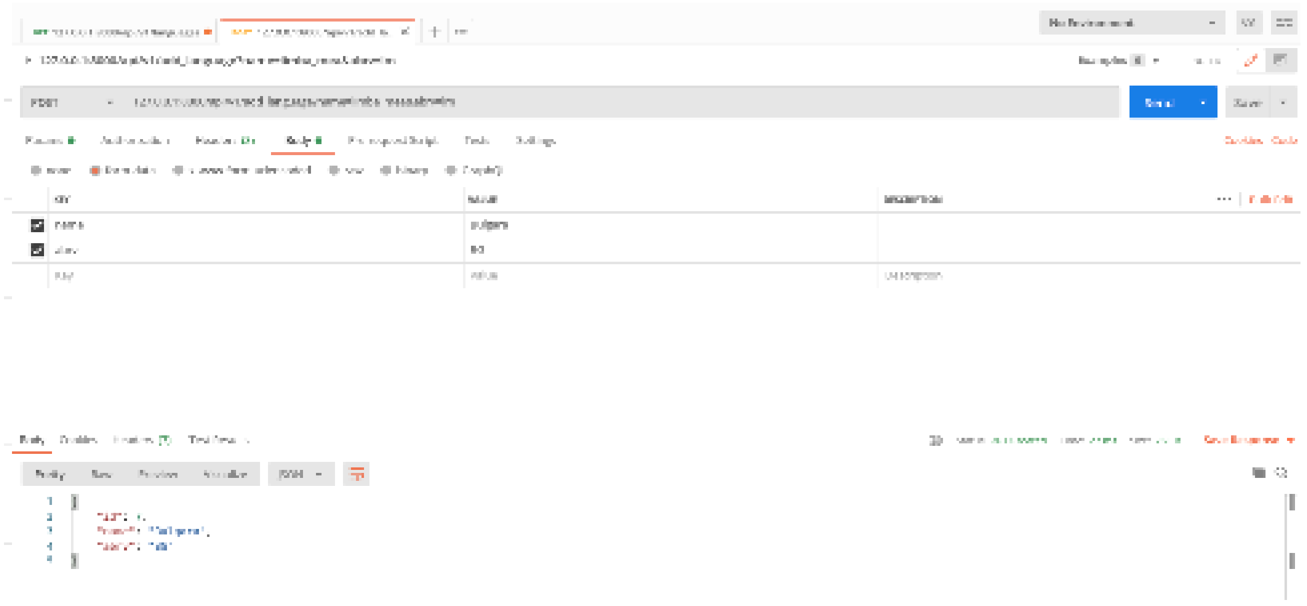
- auth_group
- auth_group_permissions
- auth_permission
- auth_user
- auth_user_groups
- auth_user_user_permissions
- database_department
- database_faculty
- database_languagestream
- database_professor
- database_professor_subject
- database_subject
- database_teachinghour
- django_admin_log
- django_content_type
- django_migrations
- django_session
- sqlite_sequence

Tabs: database_subject, django_admin_log, django_content_type, database_teachinghour, django_session, database_languagestream, database_faculty

id	abrv	name
1	FILS	Facultatea de Inginerie in Limbi Straine
2	FSA	Facultatea de Stiinte Aplicate
3	FIMM	Facultatea de Inginerie Mecanica si Mecatronica
4	FT	Facultatea de Transporturi

db.sqlite3 · Data · Structure · + Row · 1-4 of 4 rows · Columns · Filters

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



POST [illegible]

	KEY	VALUE	DESCRIPTION
✓	name	bulgara	
✓	abrv	BG	
	Key	Value	Description

```
{
    "id": [illegible],
    "name": "bulgara",
    "abrv": "BG"
}
```