

Milestone2

Group4

December 2017

Contents

1	Executive Summary	2
1.1	Modularity	2
1.2	Book Search	2
1.3	Filtering	2
1.4	Admin Dashboard	2
1.5	API	3
1.6	Amazon	3
2	List and status of deliverables	3
3	Evaluation of the status of deliverables and its impact on plan	3
4	A summary of coding work done by each team member (In tabular format)	4
5	Design	5
6	Project plan	7
7	The code structure and group process	8
8	Evaluation of tools and managing the project	8

1 Executive Summary

We are done with one of the most important requirement of our project, book search.

1.1 Modularity

One of our requirements was that admin user shall be able to edit the conversation graph and by doing that control the flow of the conversation. To achieve this goal, we changed the structure of our code to make it modular. We defined states, which includes a unique state name, a sentence to be said by the chatbot and a list of input choices which are the intents we get by giving user's response to wit.ai. Each input choice has a connection with a state to go. By creating these connections admin is able to control the flow between states for the expected intents from the users and can change the sentence said by chatbot in each state.

1.2 Book Search

Our BookBot can search books if the user asks something like "Show me LotR books". It shows the top 5 result from the Google Book API. For each book, it shows the title, author name, category and page number. Also to make the chat more user friendly, bot uses emoji.

1.3 Filtering

After making a book search, user can apply filtering on the result. At this point, we have filtering based on author, page number and category. To apply filtering, after searching a book user should say something like "Show me the books longer than 300 pages", "Show the books written by Arthur C. Clarke" or "Filter fiction books". Thanks to the modular structure of our code, we created a circular loop between these 3 filtering options and book search so that, users can apply filters consecutively and at any point start to search for a new book. When user applies consecutive filters, they all filter the first book search not the last version of the book list.

1.4 Admin Dashboard

For the admin part, front-end team improved the dashboard, so that admin will see the comments and ratings of books and conversation graph flow. In the dashboard, admin user can see the edges and states. By editing these edges and states admin user can control the flow of the conversation as stated in Modularity section.

1.5 API

Backend team implemented a functional API where admin page and chatbot request necessary information. All the information that is given is in our own database and by checking the user we don't allow any unauthorized action.

1.6 Amazon

We put the latest version of our code to Amazon server, so it is reachable by any phone with a telegram installed. It is enough to add the bot named "cmpe451_bot" to your telegram application.

2 List and status of deliverables

1. Book search Status: Done
2. Comment and Rate Books. Status: Cannot be delivered for this milestone. It will be delivered for the milestone.
3. Admin Dashboard-View of Conversation states and flow. Status:Done
4. Admin Dashboard-View of Comment and Ratings. Status: Done
5. API for admin and Chatbot. Status: Done
6. Project runs in Amazon server. Status: Done

3 Evaluation of the status of deliverables and its impact on plan

We finished the most important and longest part of our project, book search part. Before implementing this part, we discussed how admin easily changes the conversation flow and add new responses that Chatbot gives to the user. It takes a little bit long than we expected, but we clarify the implementation logic and book search part is done in time. We can not deliver "comment and rate books" part for this milestone but we believe that we can implement that part quickly thanks to our modulating the code method. In the admin part we are going according to our plan. We had some changes in our database tables for our new logic. After that we implemented a functional API which returns responds in JSON format for a request done by Chatbot or admin panel. Admin can see comments and ratings of a specified book and the flow of Chatbot conversation, the states and responds.

4 A summary of coding work done by each team member (In tabular format)

a	Subgroup	Topics	Description
Ahmet Mert Yavuz	Python	Wit.ai Filtering books Telegram Chatbot	Trained Wit.ai to distinguish searching a book and filtering a book, added intent for page filtering. Wrote the code for filtering based on author name, page number and category. Added author filter and category filter to modular code structure. Edited the edges and nodes in database to create the chatbot dialog flow.
Ahmet Enes Bayraktar	Frontend	Book comment and edge edit page	I have made improvements to admin login page and its dash board. I have also implemented the book comments page and edge edit page.
Beyza Gül Gürbüz	Python	Telegram Chatbot Modularity	I edited whole chatbot code and made it modular. I created one message handler and methods for each states. I connected the code with database and retrieved and put data to database.
Tuna Kağan Yılmaz	Backend	Django Database	API Implementation: Write the necessary methods for requests (GET, POST etc.) that are decided. Return responds in JSON format. Deciding the urls where the requests will be sent to.
Özer Biber	Python	Wit.ai Telegram	I improved the chat algorithm after the first milestone, added chitchat command and we can start after the stop command. Fixed a lot of bugs for modular use.
Emre Ün	Backend	Django Database AWS	API Implementation: Write the necessary methods for requests (GET, POST etc.) that are decided. Return responds in JSON format. Deciding the urls where the requests will be sent to. Run the project on AWS Server
Selim Abenyakar	Python	Django	Help creating modular code and bug fixings
Güneykan Özgül	Python	Django, Python, Testing	-Modified views such that chatBot is initialized as soon as the Django server starts running. -Manually added command line parameters to Django, so that the test bot or prod bot can be chosen directly from command line. -Fixed bugs that is due to merging conflicted branches
Güney Dündar	Frontend	HTML and JavaScript	I designed the admin page and the login page.

Figure 1: Coding work of members

5 Design

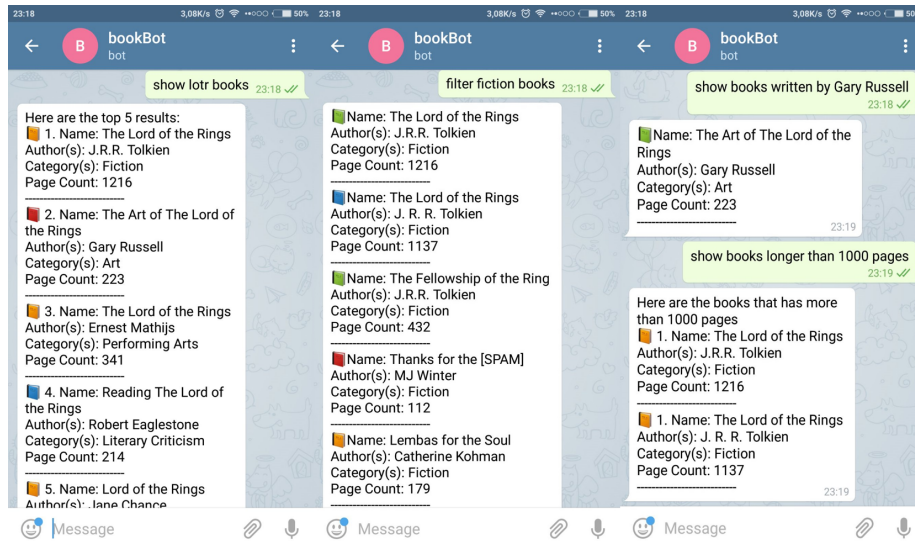


Figure 2: Telegram BookBot Design

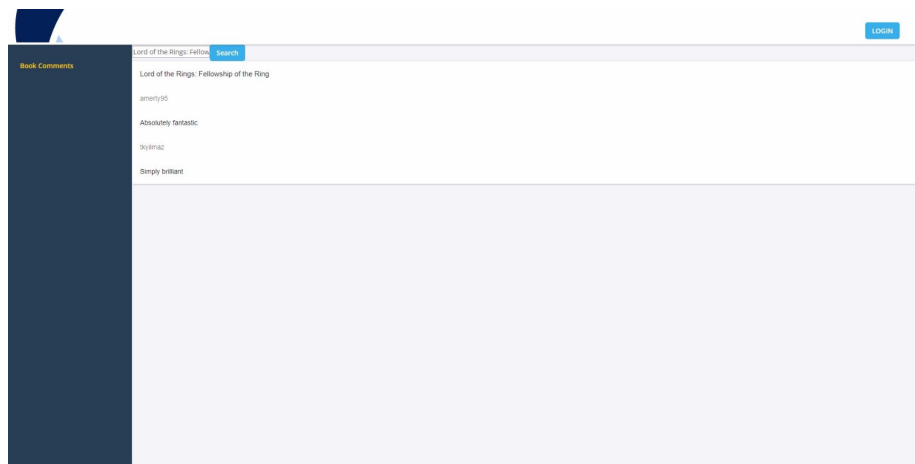
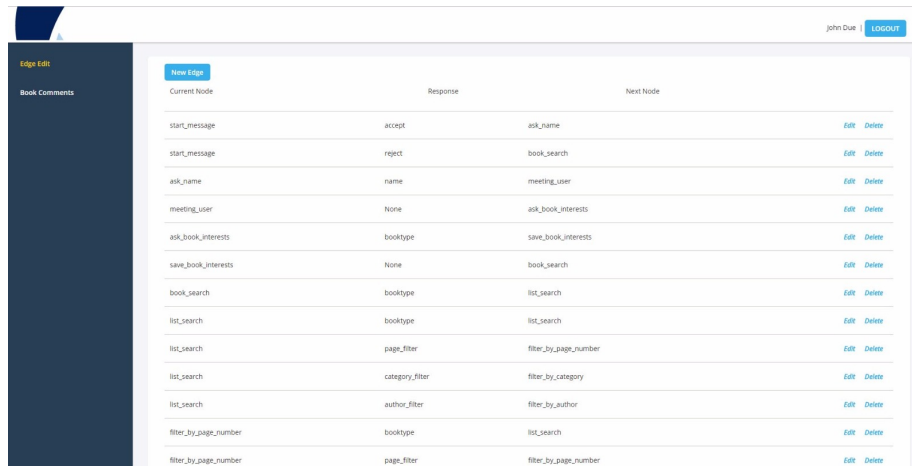


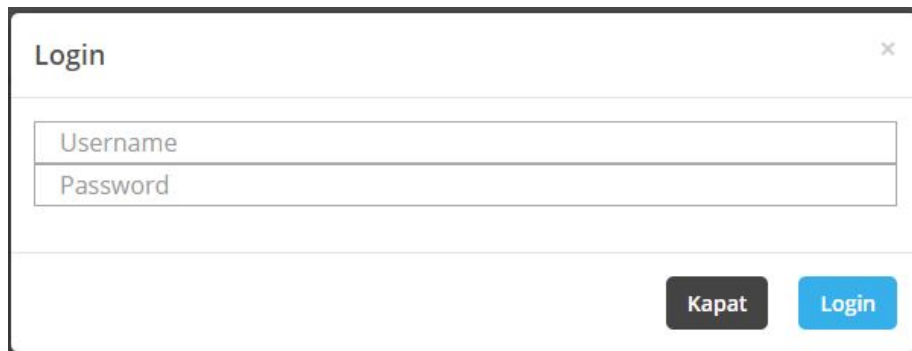
Figure 3: Admin Comments Page Design



The image shows a web application interface for managing admin flows. On the left is a dark blue sidebar with a logo and two menu items: 'Edge Edit' and 'Book Comments'. The main content area has a top header with a 'New Edge' button, a user profile 'John Doe', and a 'Logout' button. Below this is a table with three columns: 'Current Node', 'Response', and 'Next Node'. Each row represents a flow step and includes 'Edit' and 'Delete' links at the end.

Current Node	Response	Next Node		
start_message	accept	ask_name	Edit	Delete
start_message	reject	book_search	Edit	Delete
ask_name	name	meeting_user	Edit	Delete
meeting_user	none	ask_book_interests	Edit	Delete
ask_book_interests	booktype	save_book_interests	Edit	Delete
save_book_interests	none	book_search	Edit	Delete
book_search	booktype	list_search	Edit	Delete
list_search	booktype	list_search	Edit	Delete
list_search	page_filter	filter_by_page_number	Edit	Delete
list_search	category_filter	filter_by_category	Edit	Delete
list_search	author_filter	filter_by_author	Edit	Delete
filter_by_page_number	booktype	list_search	Edit	Delete
filter_by_page_number	page_filter	filter_by_page_number	Edit	Delete

Figure 4: Admin Flows Page Design



The image shows a login modal window titled 'Login' with a close button (X) in the top right corner. It contains two input fields: 'Username' and 'Password'. At the bottom right, there are two buttons: a dark grey 'Kapat' button and a blue 'Login' button.

Figure 5: Admin Login Design

7 The code structure and group process

We divided the group into three teams. Backend, Frontend, and Python.

Frontend team makes admin dashboard webpage interface and comment, rating interface.

Backend team builds database and codes backend of the admin dashboard webpage.

Python team codes the chatbot part. Connects the chatbot to wit.ai, Google Books API and Amazon Books API.

We decided that everybody will work on their branches and pull request to development branch. However, we could not communicate well, and had some troubles with management of these branches. Therefore, we decided that everybody is going to pull development branch and send pull requests to that branch. Development branch is going to be merged with master branch before milestones.

8 Evaluation of tools and managing the project

For Python part, some of us used PyCharm IDE from JetBrains in Windows which can detect syntax error before compiling and it helps us to find our mistakes easily. For Django development PyCharm is also used. The IDE has default selections in new project setting that supports multiple options and one of them is Django. PyCharm can also run the code without a need for console.

People who worked on Ubuntu used Sublime Text + console. Sublime Text saves a lot of time when using functions, classes, variables... in code by showing available inputs to user while coding. And console basically runs the code.

In Python part, we also used wit.ai. Wit.ai is a system where we give sample sentences and define intents. From these sentences, we show which intents are in it. For example, we can train wit.ai to detect name intent in “I am Bookworm” and “My name is Bookworm” sentences. By giving more sentences to wit.ai it starts to predict more accurately. For now, we defined intents for booktype, name, accept/reject and author.

For frontend, VS Code is used as IDE. It offers the basic features of an IDE that are essential like syntax highlighting and auto indentation. HTML and CSS are used for design. ReactJS is used which is a JavaScript library for building user interfaces which makes the design process easier. A package manager for Node.js modules is used, named NPM. It helps us to download packages easily. Redux is used and it is a predictable state container for JavaScript apps. AWS is used in server side. Nginx is another service we used. It keeps Amazon service online.

Like all the other groups we are using GitHub. We try to use different branches for different parts however there are still some lack of experience for GitHub and some of us has difficulties with it. We failed a few times when trying to do an operation on git. Luckily, we could undo these mistakes. We plan to help each other to get used to GitHub.

We used Gannt Chart to plan the project. Dates for different parts of the project are decided on chart and the chart is easy to follow.