Milestone1

Group4

October 2017

Contents

1 Executive Summary

We are going to according to our project plan so far. We make a good process with Chatbot conversation and we complete the "chit-chat" part. In "chit-chat" part, our Bot greets the user, asks his/her name and what kind of books s/he likes. We also add a very simple book search feature. Book search can be done by genre. Our Bot will show 5 books in this genre to the user. On the admin web side, we complete the login page for admin. After the login, admin will be directed to the dashboard. We also check for authentication and not allow unauthorized accesses. We also use an Amazon server and our project runs on this Amazon server as well.

2 List and status of deliverables

1. chit-chat Status: done
2. Admin Panel. Status: Mostly finished but Admin Login page should be beautified.
3. Backend of Admin Panel. Status: We have tried to run our api in aws server using nginx but we have failed. Hence, the implementation of the api for Admin Panel is delayed and will be done in the next milestone.

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

Chit-chat part gives us a good idea about Telegram Bot implementation. We get familiar with the tools, libraries and implementation of a bot. Since it's been done in first milestone now we are going according to the plan for next milestone. On the admin side our login page and home page for admin is ready. In the home page there will be a dashboard. Login page can be make better in order to have a good interface. Database tables are implemented and migrated. There is a feature that is not done yet. When admin wants to login, the authorization check isn't done by backend(user database). It will be fixed as soon as possible and we don't think that this will cause any delays in our progress.

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

NAME	SUBGROUP	TOPICS	DESCRIPTION
Ahmet Mert Yavuz	Python	Wit.ai Google Book API Telegram	Trained Wit.ai to detect authors, accept and booktype Searched Google Book API to take results in JSON format Fetched the necessary information from JSON output Added these codes to Chatbot to search books from Telegram
Ahmet Enes Bayraktar	Frontend	Admin page implementation Dashboard implementation	I have implemented admin login page and its dash board using react-redux. The aim of the page is to authenticate the admin and provide UI for admin related tasks.
Beyza Gül Gürbüz	Python	Wit.ai Google Book API Telegram	I added accept and reject states to Wit.ai and trained them. I also improved the bookBot.py file according to the scenario. I added searchBookHandler method to the file.
Tuna Kağan Yılmaz	Backend	Django Database	Decided on what database tables we will have in our project. Started a django project and implemented database tables. After that migration is done.
Özer Biber	Python	Wit.ai Telegram	I implemented the general structure of telegram bot,added /start, /stop and scenario handlers. I added booktype,greetings,name intents to Wit.ai and trained them.
Emre Ün	Backend	Django Database AWS	Decided on what database tables we will have in our project. Started a django project and implemented database tables. After that migration is done. Set up AWS
Selim Abenyakar	Python	Django	Learnt Django and started to set it up for the project
Güneykan Özgül	Python	Django	Learnt Django and started to set it up for the project
Güney Dündar	Frontend	Introduction to HTML JavaScript	I learnt the basics of HTML and JavaScript and prepared for the future functionalities for the admin dashboard.

Figure 1: Coding work of members

5 Requirements

Functional Requirements

1. User Requirements

   1.1. Generic User

      1.1.1. Generic user shall be able to filter books according to the following criteria

         1.1.1.1. Generic user shall be able to select an interval about price.

         1.1.1.2. Generic user shall be able to select a specific author.

         1.1.1.3. Generic user shall be able to select a specific title.

         1.1.1.4. Generic user shall be able to select a specific genre.

         1.1.1.5. Generic user shall be able to select a specific subject.

1.1.1.6. Generic user shall be able to select a specific keyword.

1.1.1.7. Generic user shall be able to select a specific publisher.

1.1.1.8. Generic user shall be able to select an interval about publishing date.

1.1.1.9. Generic user shall be able to select an interval about total number of pages the book should have.

1.1.1.10. Generic user shall be able to select a specific language.

1.1.1.11. Generic user shall be able to select an interval about ratings.

1.1.1.12. Generic user shall be able to select a specific book by entering the ISBN / ISSN of the book.

1.1.2. Generic user shall view books which are sorted by

1.1.2.1. Classification Type

1.1.2.2. Category that the user selected

1.1.3. Generic user shall be able to rate books.

1.1.4. Generic user shall be able to view other users' ratings.

1.1.5. Generic user shall be able to write comments for books.

1.1.6. Generic user shall be able to view other users' comments.

1.1.7. Generic user shall be able to get recommendations about books from ChatBot.

1.1.8. Generic user shall be able to report comments and users.

1.1.9. Generic user shall be able to get the link from Amazon to buy a book

1.1.10. Generic user shall be able to view her/his all comments.

1.1.11. Generic user shall be able to comment anonymously or publicly which way she/he desires

1.1.12. Generic user shall be able to skip chit-chat

1.2. Admin User

1.2.1. Admin user shall be able to use Conversation Graph Editor Panel (CGEP)

1.2.1.1. Admin user shall add new nodes to graph.

1.2.1.2. 1. Admin user shall be able to set START node.

1.2.1.3. 2. Admin user shall be able to add labels to nodes.

1.2.1.4. 3. Admin user shall be able to set FINISH nodes.

1.2.1.5. Admin user shall add edges between nodes.

1.2.1.6. 1. Admin user shall edit "in" and "out" parts of the edge.

1.2.1.7. 1.1. Admin user shall be able to set the template it expects from user input for "in" part.

1.2.1.8. 1.1.1. Admin user shall be able to add variables to templates.

1.2.1.9. 1.1.2 Admin user shall be able to add more than one "in" part to represent the other possible ways of saying a statement.

1.2.1.10. 1.1.3. Admin user shall be able to use empty template for "in" part.

1.2.1.11. 1.2. Admin user shall be able to set "out" to set the respond of the chatbot.

1.2.1.12. 1.3. Admin user shall be able to connect a node to more than one node with edges.

1.2.1.13. 1.4. Admin user shall be able to use variables in "in" part to fetch values from input.

1.2.1.14. 1.5. Admin user shall be able to use stored variables in "out" part for respond.

1.2.1.15. Admin user shall have a specific password to enter the system.

1.2.1.16. Admin user shall be able to view each generic user's profile.

1.2.1.17. Admin user shall be able to view each generic user's comment and rating for each book.

1.2.1.18. Admin user shall be able to view reports of comments and users.

1.2.1.19. Admin user shall be able to remove inappropiate comments.

1.2.1.20. Admin user shall be able to suspend generic users who continuously do inappropiate comments.

1.2.1.21. Admin user shall be able to shut down the system if the system is under attack.

2. System Requirements

2.1. ChatBot shall communicate with the user via Telegram.

2.2. ChatBot shall communicate with the user in a natural language.

2.3. System shall have 2 databases:

2.3.1. Conversation Graph DB shall keep data of templates which are responses to user's input.

2.3.2. User Information DB shall keep data of user's search history.

2.4. Understanding the conversation

2.4.1. System shall search through the graph and find match to given conversation.

2.4.2. System shall be able to backtrack through the conversation graph when necessary.

2.5. System shall use Google Books API to search books.

2.6. System shall use Telegram API to communicate with user.

2.7. Requirements of Conversation Graph Editor Panel (CGEP)

2.7.1. System shall store variables that are created in "in" part of the edge and enable to use variables in "out" part later.

2.7.2. System shall be able to send variables to an API.

2.7.3. System shall be able to retrieve information from API.

2.7.4. System shall be able to use retrieved information to generate output.

2.8. Chatbot shall do chit-chat to learn user's interests.

2.8.1. Each user's interests shall be kept in DB.

Non-Functional Requirements

1. Accessibility

1.1. The system should be able to do text-to-speech conversion.

1.2. The system should be able to do speech-to-text conversion.

2. Adaptability

2.1. The system shall adapt its recommendation behavior to the user.

3. Availability

3.1. System's uptime shall be over 99

3.2. Principal database shall have a mirror.

3.2.1. Mirroring mode shall be synchronous.

3.2.2. Automatic failover shall be enabled.

3.2.3. A witness server shall be present.

4. Compatibility

4.1. System shall be compatible with Windows, Linux, MacOS, Android and iOS.

4.1.1. System shall work on Android 4.2 or higher versions.

4.1.2. System shall work on iOS 9 or higher versions.

4.1.3. System shall work on Windows XP or higher versions.

4.1.4. System shall work on MacOS 10.4 or higher versions.

5. Efficiency

5.1. Space efficiency

5.1.1. Database backups older than 30 days shall be deleted.

6. Extensibility

6.1. The system shall be able to be extended by new conversation templates.

7. Data Recoverability

7.1. In the case of a failure, potential data loss shall be no more than most recent 15 minutes of transactions.

7.1.1. A full backup shall be taken every 8 hours.

7.1.2. A differential backup shall be taken every 45 minutes.

7.1.3. A transaction log backup shall be taken every 15 minutes.

7.2. Deleted data shall be recoverable for 30 days.

8. Responsiveness

8.1. System's response time shall be no longer than 10 seconds.

9. Ubiquity

9.1. The system should run on Web, PCs, smart phones.

10. Conversation Graph Editor Panel

10.1. Graphs that are created in CGEP shall be cyclic.

10.2. Edges in graph shall be unidirectional.

11. Security

11.1. User's account shall be protected with an encryption code.

11.1.1. Passwords shall be stored as MD5 hashes.

11.2. User shall have a password to login.

11.2.1. User's password shall be at least 9 characters long.

11.2.2. User's password shall contain at least three of the following characters at least three times

11.2.2.1. Uppercase letter

11.2.2.2. Lowercase letter

11.2.2.3. Number

11.2.2.4. Non-alphanumeric character

12. Privacy

12.1. The system shall ask for user's permission on sharing her own private data via a privacy policy agreement.

6 Design

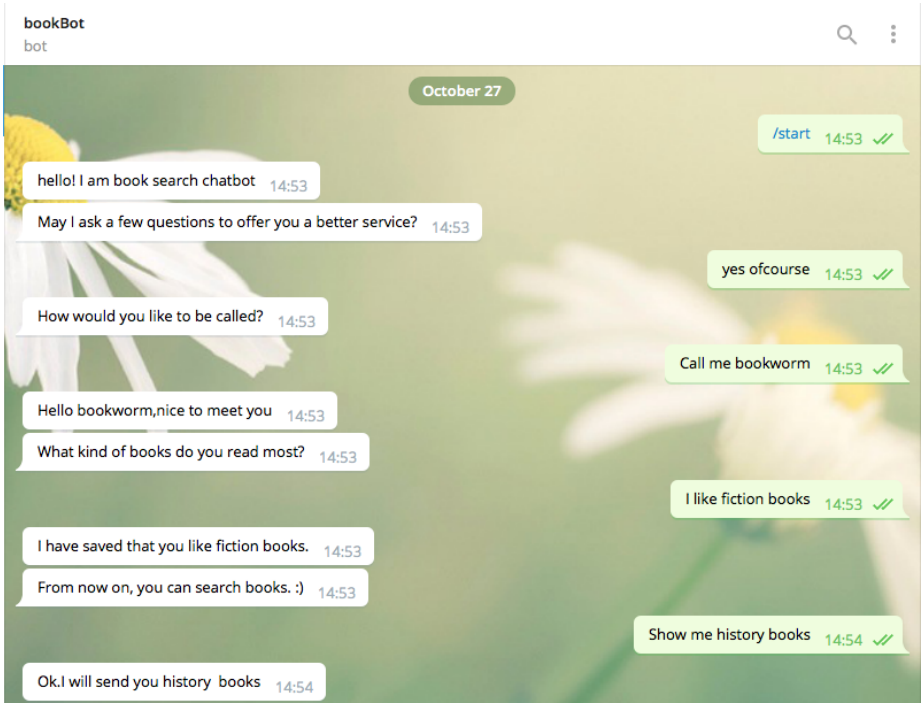

Figure 2: Telegram BookBot Design

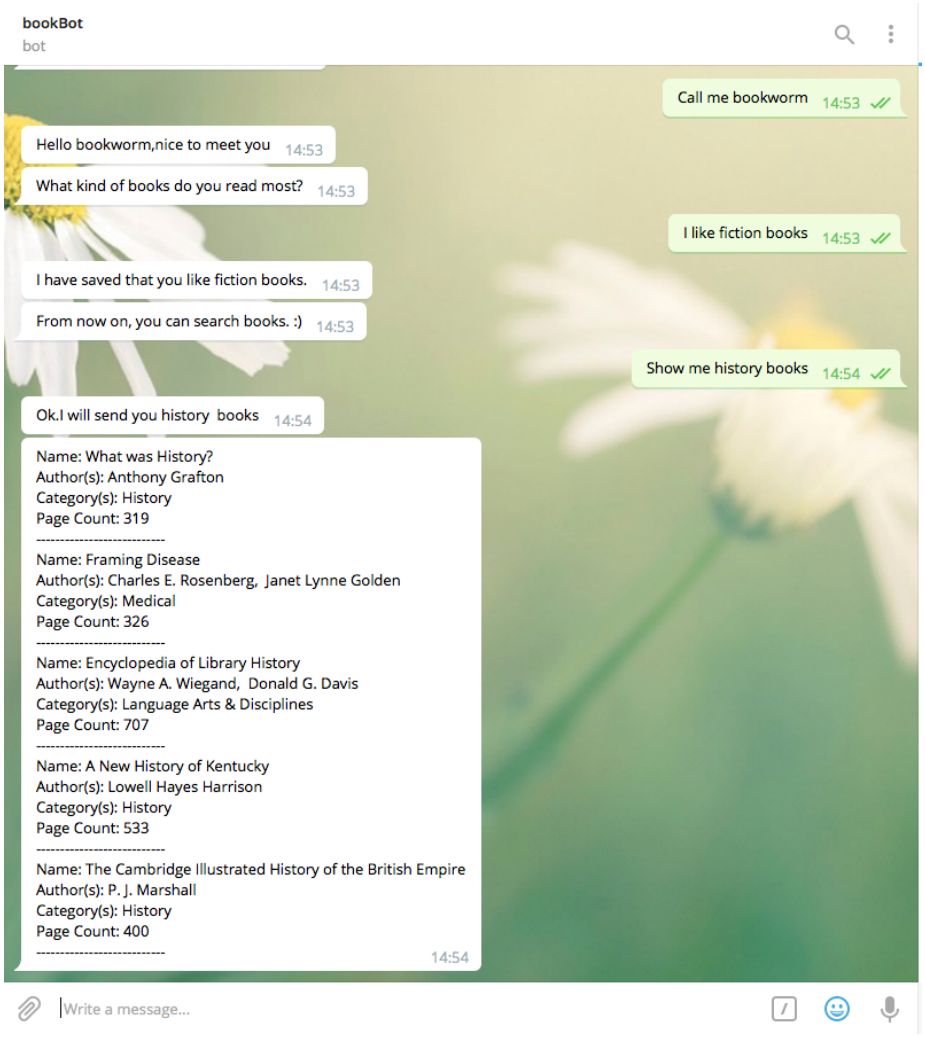

Figure 3: Telegram BookBot Design

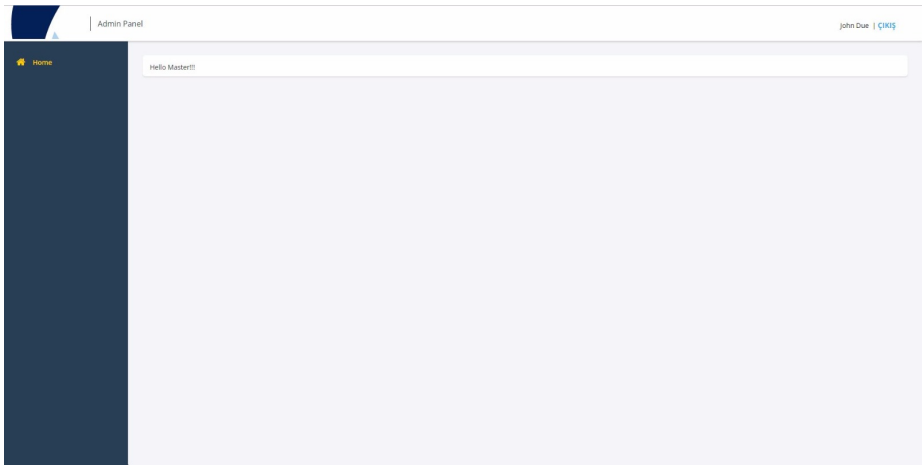

Figure 4: Admin Dashboard Design

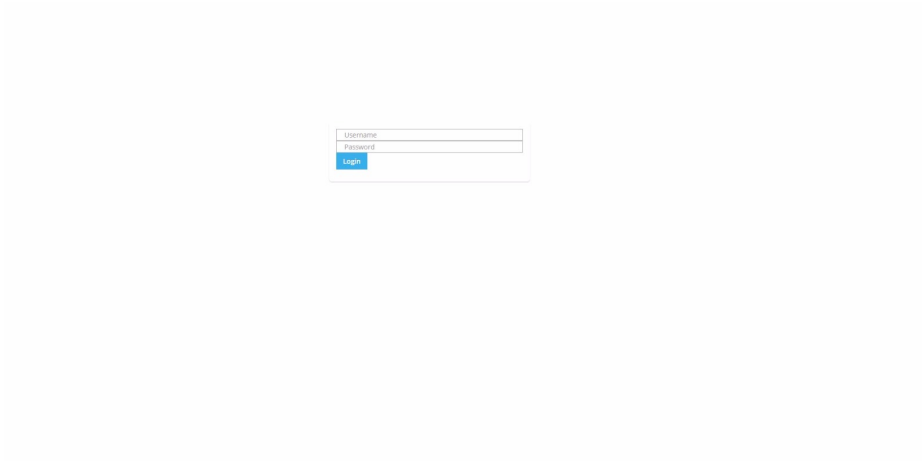

Figure 5: Admin Login Design

7 Project plan

Task Name	Start Date	End Date	Duration
CHATBOT	14-02-17	05-01-18	325
Implementation	19-10-17	03-01-18	76
Admin Web Page	19-10-17	03-01-18	76
Login Page	19-10-17	26-10-17	7
Backend - Frontend	19-10-17	26-10-17	7
Dashboard	26-10-17	16-11-17	21
Backend - Frontend	26-10-17	16-11-17	21
Comment/Rating View	16-11-17	07-12-17	21
Backend - Frontend	16-11-17	07-12-17	21
Conversation Graph Edit	07-12-17	03-01-18	27
Backend - Frontend	07-12-17	03-01-18	27
Chatbot(Telegram)	19-10-17	03-01-18	76
Chit-chat	19-10-17	26-10-17	7
Book search	26-10-17	30-11-17	35
Comment and rate books	30-11-17	07-12-17	7
View Comment/rating	07-12-17	14-12-17	7
Book recommendation	07-12-17	03-01-18	27
MS1	26-10-17		
MS2	07-12-17		
Testing	20-11-17	03-01-18	44
Admin Panel	20-11-17	03-01-18	44
Chatbot	20-11-17	03-01-18	44
Handover	03-01-18	05-01-18	2
Tutorials and training	03-01-18	05-01-18	2
MS3: Presentation	05-01-18		

Start Date in Number Form	42780.00	*Use this number for the Minimum Bound of the Horizontal Axis to set the beginning of the chart.*

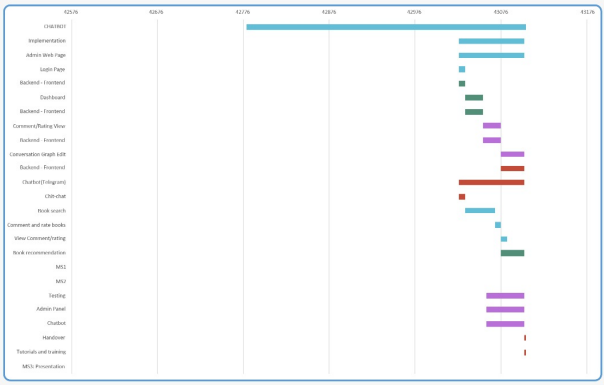

Figure 6: Project Plan

8 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.

9 Evaluation of tools and managing the project

In the project, we split into 3 group which are backend, frontend and Python part.

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 Mert" and "My name is Mert" 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 futures of an IDE that are essential like syntax highlighting and auto indentation. HTLM 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.