



BU Bot

Final Report

29.4.2022

TEAM MEMBERS

Aamir Dhanani - 6463012

Alex Freer - 6452551

Matteo Caglioti (Leader) - 6418784

Mike Tchoupiak - 6397053

1- Development Cycles

The easiest way to describe the initial backlog and subsequent sprints is to go sprint by sprint and discuss what we tried to achieve and what was a result:

Sprint 1:

C4-10

Data Gathering

To Do

Description

Add a description...

Child issues

Order by 100% Done

C4-10	As an admin, I want to...	M	DONE
C4-60	As an stakeholder, i w...	AD	DONE

User Story 1:

Data Gathering / C4-60

As an stakeholder, i want a custom database so that, the chatbot can use it to better answer questions

Done

Description

Add a description...

Child issues

Order by 50% Done

C4-71	Create database sche...	AD	TO DO
C4-72	Get a botpress instan...	M	DONE

Sub Tasks of User Story 1:

C4-60 / C4-71 1

Create database schema and tables, or find open source alternative

Attach Link issue

To Do

Description

Add a description...

Details	
Assignee	AD Aamir Dhanani
Labels	None
Story point estimate	None
Sprint	None +1
Reporter	MC Matteo C.

C4-60 / C4-72 1

Get a botpress instance up and running

Attach Link issue

Done Done

Description

Add a description...

Details	
Assignee	M Mike
Labels	None
Story point estimate	None
Sprint	None +1
Reporter	MC Matteo C.

User Story 2:

Data Gathering / C4-18 1

As an admin, I want to manually/automatically crawl a website, so that I have data to provide to the chatbot.

Attach

Done Done

Description

Add a description...

Child issues Order by 50% Done

C4-70	Create basic data qat...	M	DONE
C4-73	Modify script to crawl...	M	TO DO

Sub Tasks of User Story 2

The image displays two Jira tickets side-by-side, representing subtasks of User Story 2.

Left Ticket (C4-18 / C4-70):

- Title:** Create basic data gathering script for courses, that can be used for other pages as well
- Status:** Done (indicated by a green checkmark and the word 'Done')
- Description:** Add a description...
- Details:**
 - Assignee: Mike (M)
 - Labels: None
 - Story point estimate: None
 - Sprint: None +1
 - Reporter: Matteo C. (MC)

Right Ticket (C4-18 / C4-73):

- Title:** Modify script to crawl other pages than just the courses page
- Status:** To Do (indicated by a grey button labeled 'To Do')
- Description:** Add a description...
- Details:**
 - Assignee: Mike (M)
 - Labels: None
 - Story point estimate: None
 - Sprint: None +1
 - Reporter: Matteo C. (MC)

For the first Sprint we decided to concentrate on the most stories that would result in progress to the foundation of the project's infrastructure. These stories were then split up into subtasks, allowing for more palatable/workable activities. The first tasks/subtasks that were necessary for the project were the creation of the database schema/tables, however we soon realized we could approach the project more efficiently and decided to find an open source solution. Ultimately we decided on Botpress as it allowed for everything from database creation, natural language processing, built in training and even analytics within its dashboard. During this sprint we tested out the capabilities of Botpress by creating local instances.

The next few tasks focused on the creation of generic crawling scripts that would allow for easy collection of Brock university website information. We felt it was easiest to focus on the courses as our first example data gathering script.

During this first sprint we spent a lot of time discussing our tasks and potential tasks over Discord's text and video channels. For subsequent sprints we used Discord as our main communication and loosely used Jira (Product Backlog).

Second Sprint:

For this sprint we concentrated on hosting of the Botpress backend system using AWS EC2. This was a very difficult task for our team as we have not hosted a site ourselves before and did not really have a clear indication of where to begin. We decided first to do some research on how one would host a site on a domain and the activities that went into achieving this. Some tasks were created to keep track of the resources we found through our research for this procedure as well as tasks for viable hosting sites. From there we created tasks associated with the steps we determined through our research and worked on said tasks. This resulted in a docker instance on AWS EC2 where we could host botpress through our domain, which thankfully was already owned by one of our team members.

Third Sprint:

For this sprint we concentrated on creating a basic frontend system. From the get-go our team decided to use React as our frontend framework as it is currently the industry standard. As well, using Create React App made it easy to quickly get an operational web app with node modules up and running. From there we created tasks from basic layout of the interface, all the way to the functionality of menu items. Ultimately this resulted in a simple mock interface with a side menu, header, text area for user input, and an area for previous messages from the user or the chatbot. Chatbot communication was not set up between the interface and botpress yet during this sprint.

Fourth Sprint:

For this sprint we concentrated on training the Chatbot using botpress' workflows and contexts. We began by creating tasks for a generic context, which would later allow us to extrapolate for more tasks associated with other contexts (exams, courses, professors, etc.). Other tasks during this sprint involved setting up communication between the frontend and the Botpress instance that was hosted on our domain. Ultimately this sprint resulted in a fully functioning web app connected to our Chatbot instance.

Fifth Sprint:

For this sprint we focused on creating tasks associated with adding further functionality to the frontend web app and system testing for the backend system. The front-end tasks for this sprint were the addition of dark theme toggle and proper refreshing of the chatbot session (new conversation instance, etc.). For system testing this involved creating a list of questions and variations of these questions to ensure the responses for said questions were correct.

Sixth Sprint:

For the final sprint we focused on creating tasks for missing features as well as more tasks associated with training and data gathering. At this point we had a fully operational Chatbot with a backend system and web app implementation, we realized there were a few questions that were missing. Questions such as a recent brock news or directions to Brock were a few example questions missing. As well, clickable links were not implemented for the frontend system.

2- User Guide.

- **UI**

- The chatbot can be accessed at <https://bu-bot.netlify.app/>
- Additional features can be accessed in the sidebar. The sidebar can be displayed using this display button at the top-left corner.




- The chat log can be downloaded using the download button in the sidebar.

Chat Log

- The chat can be cleared and a new chat started using this refresh button in the sidebar.

Refresh Chat 

- The UI can be toggled between light and dark themes using this toggle in the sidebar.

Dark Theme 

- If more than one chatbot was created, the user can return to the main menu using the return button found in the sidebar.

Return 

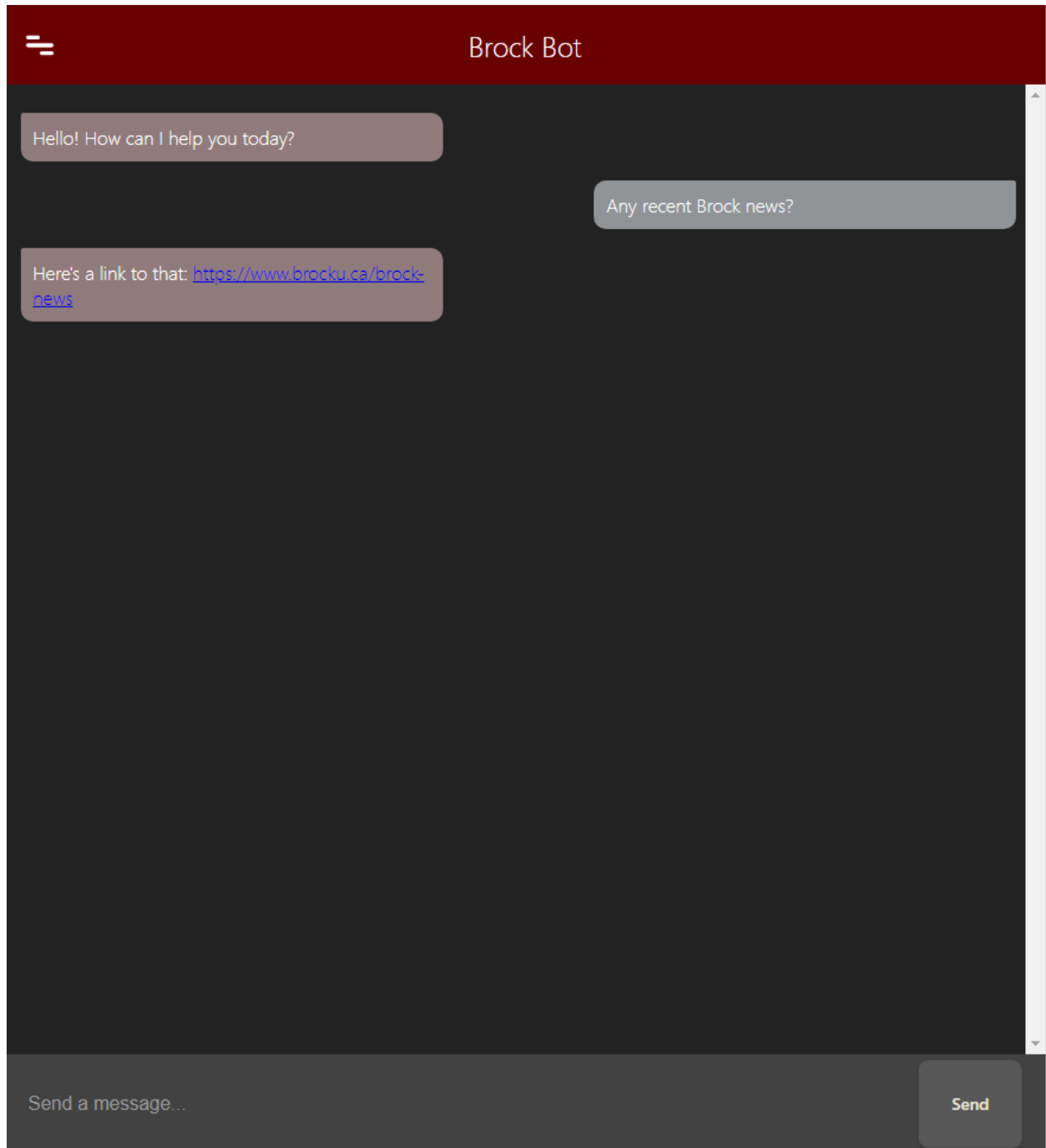
- Questions typed into the text area at the bottom can be sent to the chatbot by pressing the send button or ENTER.

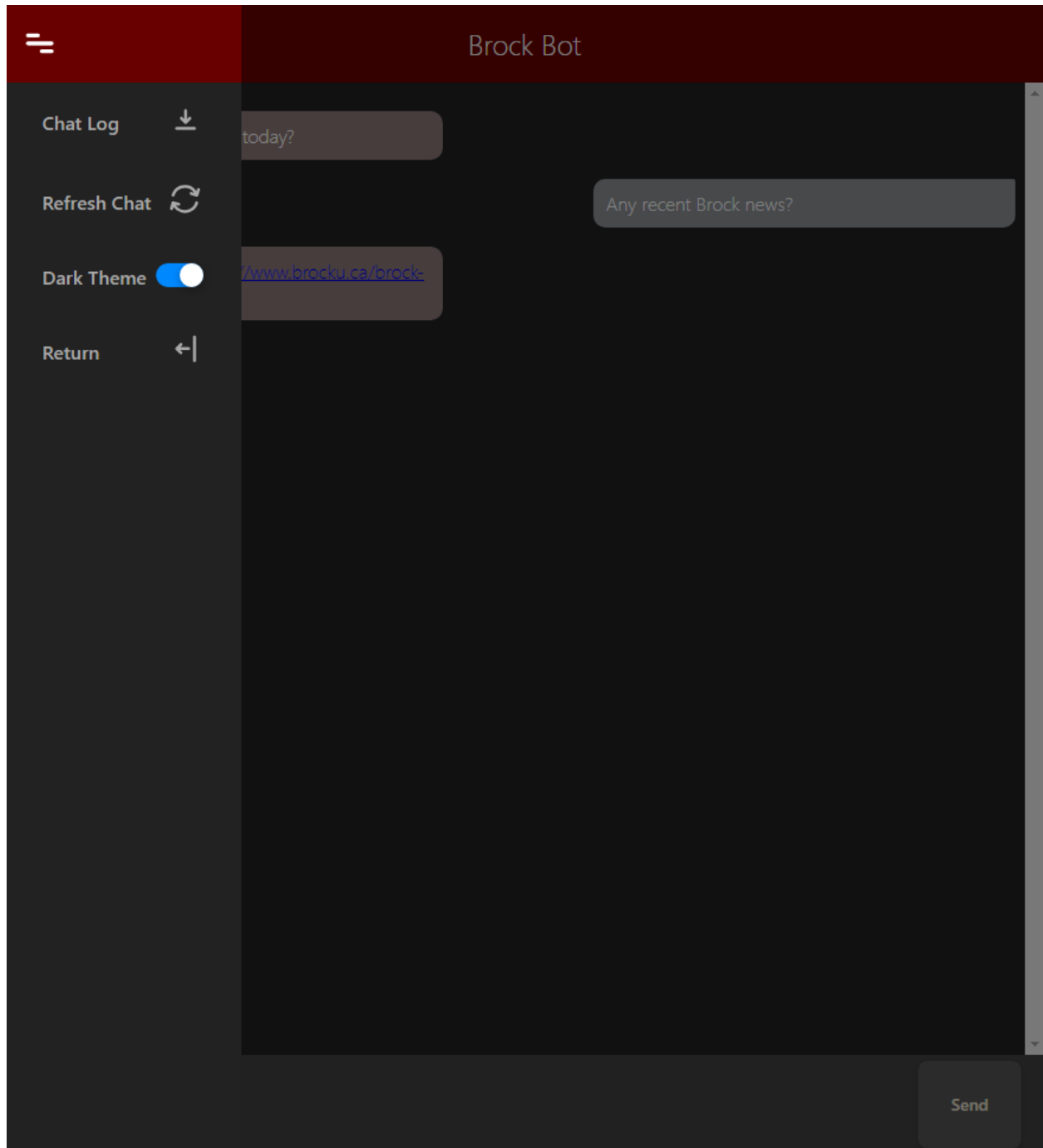
Send

Example list of Questions:

- What is COSC 3P32?
- Who teaches COSC 2P03?
- Where is Brock?
- MATH1P67 time?
- What is the accounting program?
- Who teaches ENTR 2P51?
- Prof for FNCE 2P51?
- Labs for cosc 3p32?
- When is the exam for MATH 1P67?

Example Chatbot Use Case:





- **Botpress Admin Panel**

- The Botpress Admin Panel can be accessed at <https://www.advenship.com/>
- Select a chatbot to edit under the "Bots" section
<https://www.advenship.com/admin/workspace/default/bots>

- Intents and Entities can be managed through the “Natural Language Understanding” section (<https://www.advenship.com/studio/{BOTNAME}/nlu>)



- The chatflow can be managed through the “Flows” section



- The JavaScript chatbot scripts can be edited in the code editor section



3- Installation and Setup.

- **React**

- The application was bootstrapped with Create React App:
<https://github.com/facebook/create-react-app>
- For modifications or local hosting, it is required to have Node > = 14.0.0 and npm > = 5.6 installed.
- To view live edits of the structure of the project, you will first need to install the required node module dependencies: **npm install**
- For a build ready version, you can use **npm run build**
- To locally run the project, you can use **npm start**
- **All commands require that a node environment is available for the current working directory**
- As previously stated, a currently hosted version is available at:
<https://bu-bot.netlify.app/>

- **Amazon EC2, RDS and Route 53**

- **Route 53**

- Buy a domain and configure Route 53 to respond to DNS queries.
- Amazon easily manages domains and SSL through this tool.
- Route traffic to application load balancer.

- **Create an Application Load Balancer**
 - Listens for activity at HTTPS: 443 and directs to Botpress EC2
 - Implements SSL certificate from Amazon Certificate Manager
 - Inbound rules
 - Security Group: balanceSecurity
 - Security Name: Balancer Security
 - Ports: 80 → HTTP, 433 → HTTPS, 22 → SSH
 - Target Groups
 - Specifies where to route incoming traffic from load balancer
 - Protocol HTTP
 - Port 3000

- **Create an EC2 Instance**
 - Run a Docker Container
 - Image: NodeJS running BotPress application.
 - Inbound rules
 - Security Group: launch-wizard-7
 - Security Name: BotPressDockerServer
 - Ports: 3000 → Balancer Security, 22 → WORLD

- **Create an RDS**
 - Run Postgres SQL Server
 - Contains:
 - botpressdb - Botpress Data
 - brockcollections - Data source for answers
 - Inbound rules
 - Security Group: default
 - Security Name: BotPressDB
 - Ports: 5432 → BotPressDockerServer

- **Create database tables.**
 - Courses, Exams, Important Dates

main COSC_4P02_Project / Database Tables.sql Go to file ...

Run python data population scripts. Latest commit 78034b6 1 minute ago History

main COSC_4P02_Project / Data Gathering Go to file Add file ...

freer07 Merge branch 'main' of https://github.com/freer07/COSC_4P02_Project b97ad9d 5 days ago History

File	Commit Message	Time
courseInformation .py	Create courseInformation .py	2 months ago
dateoccasion.json	added the synonyms	5 days ago
exams.py	Exam schedule data extraction.	6 days ago
importantDates.py	removed commented code	2 months ago
makeOccasionsJSON.py	added the synonyms	5 days ago
programCalendarInfo.py	Course Calendar Data	2 months ago
programData.json	Course Calendar Data	2 months ago

```

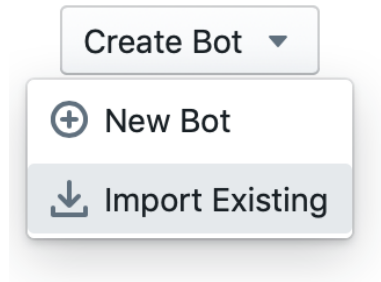
17 exclusions character varying(50000) COLLATE pg_catalog."default",
18 notes character varying(50000) COLLATE pg_catalog."default",
19 days character varying(50000) COLLATE pg_catalog."default",
20 "time" character varying(50000) COLLATE pg_catalog."default",
21 type character varying(5000) COLLATE pg_catalog."default",
22 section character varying(25) COLLATE pg_catalog."default"
23 )

```

- **Docker Container Setup**
 - Step 1: Run the server
 - docker run -d --name=botpress -p 3000:3000 botpress/server
 - Step 2: Get into the BotPress docker container
 - docker exec --interactive --tty botpress bash
 - Step 3: Stop running the BotPress container.
 - docker stop botpress
 - Step 4: Create the .env file with the following information:
 - echo "DATABASE=postgres
2DATABASE_URL=postgres://masterbotaccess:ilikechicken
@botpressdb.ctor5sx6le6s.ca-central-1.rds.amazonaws.com
:5432/botpressdb3BPFS_STORAGE=database4BP_PRODUC
TION=true5DEBUG=bp:audit:* ./bp -p 2>> ./botpress.log" >
.env
 - Step 5: Run the server again, new changes should apply.
 - docker start botpress

- **Botpress**

- Import code from Botpress home page. Zip the **bot_brock-university** folder and import it into Botpress.



bot_brock-university

Added entity.

2 months ago

Overview of Architecture

