Athena Documentation

Sriyash Poddar

May 2020

1 Project Links

Athena-frontend
Athena- backend

1.1 Problem Statement

"Athena", the Greek God of knowledge and wisdom is an online service for the students at any university. A student using this can not only browse details of the courses but also get suggestions regarding the courses he/she can take.The user can check for slot clash between chosen courses, their prerequisites, while at the same time have discussions about them on an anonymous forum. The main purpose for this is to make decision making easier for them.

The students at any university have a huge burden resting on their shoulders, which is to prepare for their careers and an important step in that direction is taking aca- demic courses in the university, based on their interests and aspirations. But hundreds of courses,some which are difficult, dependent on others, or even similar put the student into an endless spiral of decision making. So 'Athena' will take care of all of the students needs with respect to managing their courses and making information easily available for them.

1.2 Tasks Completed

The following are the tasks completed :

1.2.1 Backend

- Created endpoints to
  - run a `search` for courses, given a set of keywords
  - `filter` the search results, based on users choices
  - `get course information` such as credits, professor names and keywords
- Created endpoints to

  - generate recommended courses for the user, given a set of keywords

- Created endpoints to
  - fetch the slots for various courses
  - add and delete courses to and from the timetable

- Created endpoints to
  - search notice boards of different courses
  - fetch, create, delete and downvote notices

- Created endpoints to
  - search chat rooms of different courses
  - fetch chat history
  - send messages

- Created a pipeline to implement real time message conversation using websockets and networking

- Create database models to
  - store course information
  - store course slots
  - store notices
  - store messages
  - store user information(ip_address, downvotes etc.)

- Created an elastic search instance to get search and recommend results

- Scraped the internet to get information about the course

- Populated the database with the scraped data

1.2.2 Frontend

- Created a landing page for the users
- Created a search page with multi-keyword input, filter checkboxes and hyperlinks
- Created a recommendation page with multi-keyword input, graphical output and hyperlinks
- Created a page to add courses to the timetable with autocomplete search bar, graphical output option to download the timtable and warnings, in case of clashes
- Created a page to search for notice boards
- Created a page to see, add, delete and downvote your notices
- Created a page to search for chat rooms
- Created a page to see the chat history and send messages

All functionalities are implemented with the users identity being anonymous.

Further Work :

- Add ML/DL approaches to generate robust recommendation
- Improve security by using encryption
- Scrape more data to make the system widely usable

- Add options to send media in chat, notices.

1.3 Implementational Details

1.3.1 TechStack:

- Backend
  - Flask
  - Flask Admin
  - Flask Cors
  - Flask SQLAlchemy
  - Flask SocketIO
  - SQLLite
  - Elastic Search
  - JavaScript Objects
  - requests
  - BeautifulSoup4
- Frontend
  - ReactJS
  - axios
  - react-bootstrap
  - react socket.io
  - HTML
  - CSS
  - JavaScript Objects

1.3.2 Frontend Interface

- ReactJS is used as a framework for its implementation.
- CSS and some react libraries are used to beautify the pages.
- Used CSS animation to create the landing page
- Used HTMLtoPng to download the timetable
- Used socketio.client to get real time messaging
- Used axios to fetch data from server and ip_adress
- Used JavaScript obejcts to pass data

1.3.3 Backend

- Used flask request to send data from the client(i.e the frontend) to the server.
- Used flask response to send back response to client
- Used Elastic Search to generate the search and recommendation results
- Used SQLLite as database and SQLAlchemy as ORM
- Used JavaScript obejcts to pass data
- Used Websockets to enable real time data transfer between the clients and thus, implement the chat room.
- Used requests and BeautifulSoup4 to scrape the internet to get data about course information and slots.

GUI : ReactJS

Networking : Websockets

Database : SQL