236369

Winter 2021-2022

Managing Data on the World-Wide Web

Project – Telegram Polls

Part 1 submission date: 30.12.2021 11:55 PM
Part 2 submission date: 16.1.2022 11:55 AM

Introduction

In the final project you will implement a Telegram Polls Manager web application.
Your app will include a Telegram bot interface for user registration and poll answering. Admins define polls questions and submit them for the users. The system's users will get poll questions through a Telegram bot and respond with their votes. Polls can reach everyone or a subset of relevant users. The web page interface is used for admins registration, charts and statistics display, and polls broadcasting to users through Telegram.

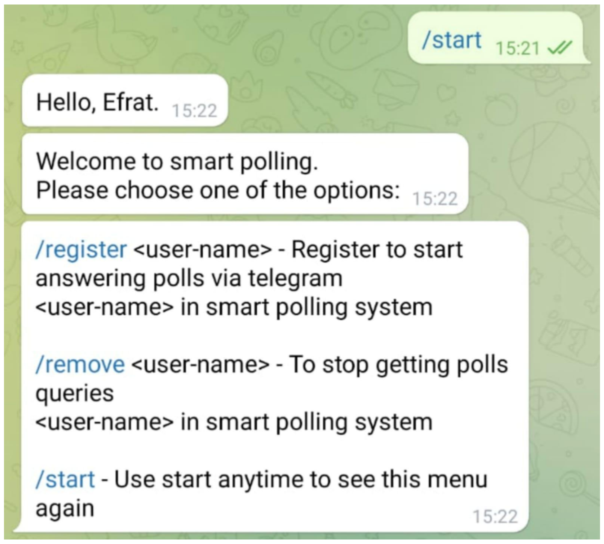
We divided the project into two parts, both done in pairs –

1. In the first part, you will implement your application's backbones. The submission deadline for this part is 30.12.2021, 11:55 PM. This part weighs 10% of your final grade.
2. The submission deadline for the rest of your application is Sunday, 16.1.2022, **11:55 AM**. This part (including the presentations) weighs 40% of your final grade. Note the deadline is 11:55 AM.

Finally, on the 16-27 of January, we will interview you about your project. We will ask you questions about the project and the relevant course material. You should prepare a short demo of your application. The exact schedule and further information will be published in the coming weeks.

Part 1 Requirements

In the first part of the project, you will implement an initial version of the Telegram bot and the Flask server, and you are also required to design the database that you will use in the project.


/start 15:21

Hello, Efrat. 15:22

Welcome to smart polling.
Please choose one of the options: 15:22

/register <user-name> - Register to start answering polls via telegram
<user-name> in smart polling system

/remove <user-name> - To stop getting polls queries
<user-name> in smart polling system

/start - Use start anytime to see this menu again 15:22

Your server should receive requests from the Telegram bot, and handle these by updating the database accordingly. The initial server will allow registration and deregistration to the polling service, as shown in the screenshot.

The initial Telegram bot should support the following commands:

- ‘/register <username>’: will register the user to the polling service with the provided username.
- ‘/remove <username>’: will remove the user from the polling service.

You can find an example of a simple Telegram bot that uses the python-telegram-bot package here.

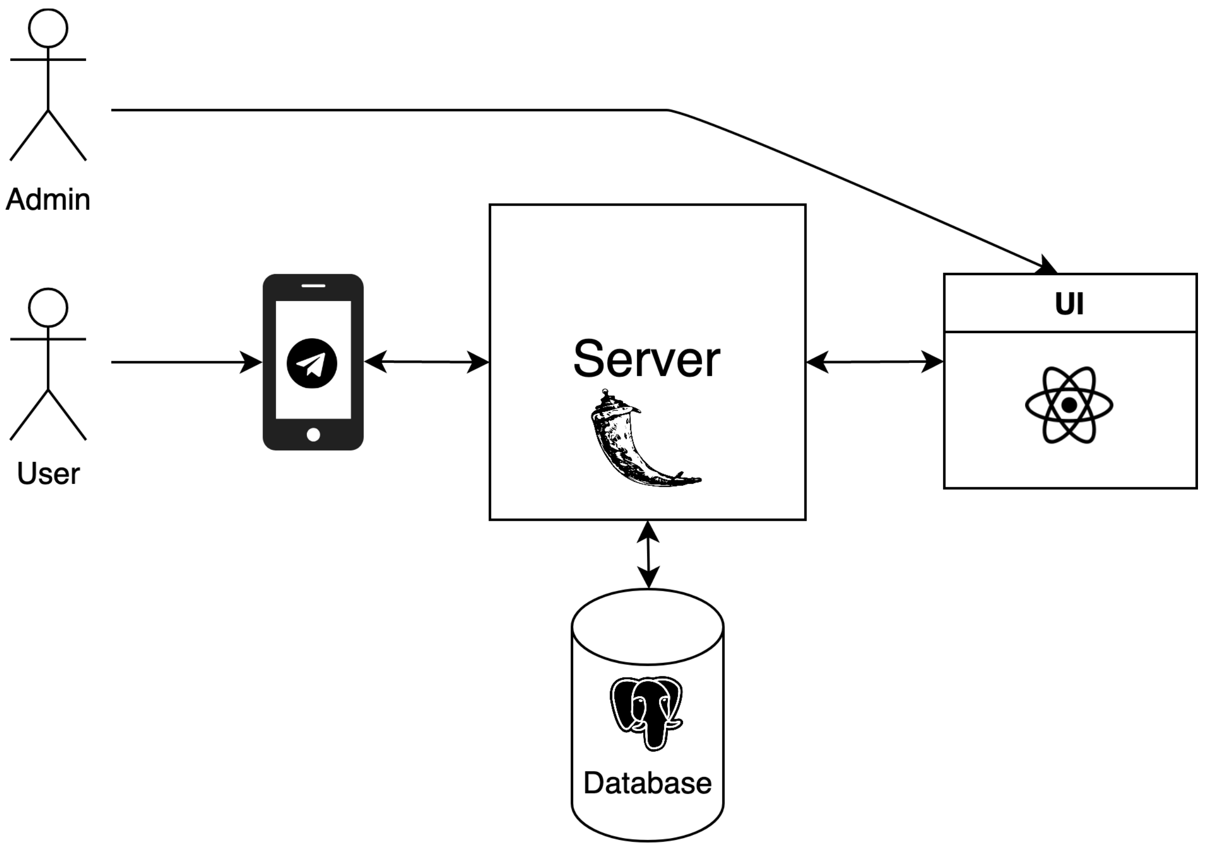
Database Design: You are required to design the database you will use in the rest of the project. For that you will need to think carefully about the implementation of all parts of your project.

The database design should be submitted as an SQL script that creates the tables in your SQL database. This script will contain the “CREATE TABLE” commands that your project will use to initialize the database.

The goal of Part 1 is that you familiarize yourselves with the different parts of the project, such as connecting your Flask backend to the database and the communication between frontend and backend.

Information regarding the submission of part 1 can be found below under the Submissions section.

Architecture



Components

1. **Telegram:** This will be the way users will be interacting with your system for voting. Messages, commands and requests sent by users are passed to the Telegram bot handlers that you define on your server.

   Users will first need to register to your bot by looking it up in Telegram and performing a '/register/' command.

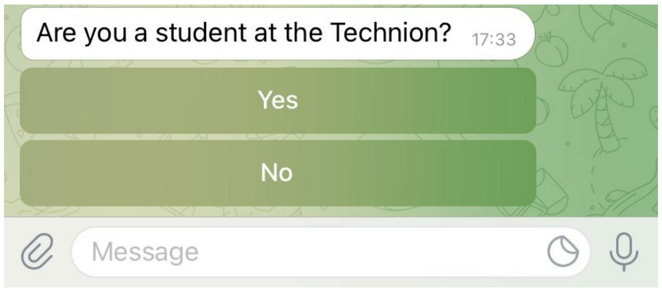
   You will need a bot token which is the bot secret key that you will use in your bot code. You get it when creating a new bot. How to create a new bot? Well, Telegram has a bot for that. Just talk to the BotFather.

   All you need to do is:

   - Go to the BotFather (look him up on Telegram)
   - Create a new bot: /newbot
   - Give it a name

- Give it a username (must end with the word bot)
- Copy the authentication token you receive from BotFather to your code.

The bot will allow user registration and deregistration of the service, and will act as the main poll service interface. The admins will send polls to users. The users' response will eventually be sent to a server that should store the votes in a database.



The admins will have the option to send polls to relevant users according to their response to a specific poll. For example, if users have been asked whether they are students at the Technion, the admin could send only those who answered 'Yes' a poll about student's festival artist preference.

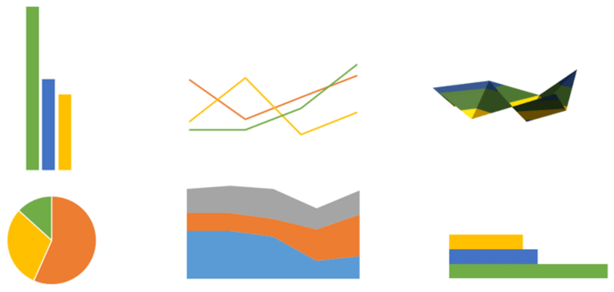
Your server will be accessible through a UI (React is a must) that will show the poll results as informative charts (pie charts/bar charts/etc.).

2. **Server:** The main logic will be implemented in the server component. The server will implement the different endpoints that are needed for the Telegram bot and for the UI. For example, the server will query the database when the UI displays poll results.

   Your server must use **Flask.**

3. **Database:** The users, their votes, and any other relevant information, should be stored in a **PostgreSQL** database. Your server will access this database to show poll results. You are required to submit a database design (an SQL script) in Part 1. That design should consider all components and functionality of the application. Plan it wisely.

4. **UI:** Your server should display a web UI where admins can log-in and view the results of different polls as informative charts.



Showing the charts in the UI can be done with the amCharts web API. The UI must be implemented using **React**. The React code should be non-trivial (not HTML-like) and should use Hooks (e.g., ‘useState’).

Managing the system through the UI should only be possible after successful authentication via password insertion interface.

The web UI will include interfaces for:

1. Adding new admins by existing admins.
2. Creating new polls. There will be a convenient UI for defining multiple choice questions and filtering relevant target students according to their previous answers.

The UI should be interactive and convenient. It should display relevant current information as previous questions and current admins list.

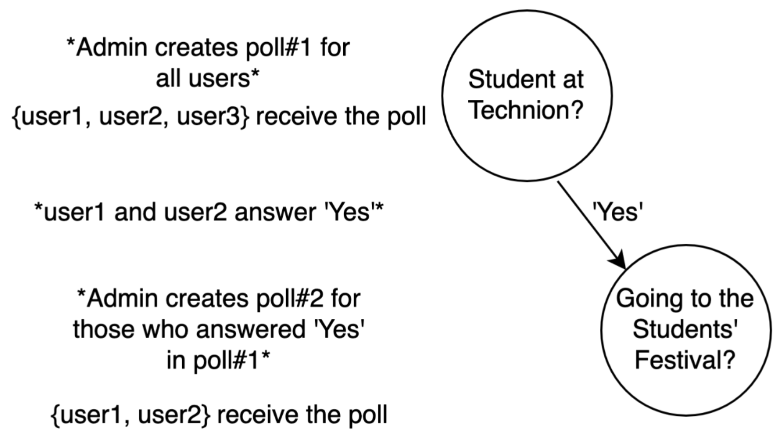
Design and Styling

You have complete control over the design of your application. We expect your UI to be user-friendly, coherent, and stylish. We encourage you to be creative with both the appearance of the UI and the application's functionality. Feel free to add new options and features.

Handling Errors

Your server should never crash nor stop working unless closed manually or your connection/computer crashes, doing so would be considered a grave error. If an error happens, you should return a corresponding HTML page describing the error (without exposing information about the server), and the most suitable status code.

Example Scenario


Admin creates poll#1 for all users
{user1, user2, user3} receive the poll
Student at Technion?
user1 and user2 answer 'Yes'
'Yes'
Admin creates poll#2 for those who answered 'Yes' in poll#1
{user1, user2} receive the poll
Going to the Students' Festival?

Tips

Getting Started

- Read the whole assignment.
- Download all relevant libraries and technologies.
- Try a basic example in both Flask and React to make sure you understand how to run them.
- Create a new Telegram bot key by approaching the BotFather, and then try to run the provided simple telegram bot with that bot key.
- Make sure you can connect to the database.

General

- Use the Web! Someone probably implemented a lot of the features you are looking for.
- Plan your workflow, classes, and data structures well – bad planning can make life much harder when implementing.
- We recommend you start working on the project as soon as possible.

Submission

Part 1

You should submit:

- An SQL script (.sql) file with the SQL commands that you use to initialize your database.

- A **zip** file named "*hw4_id1_id2.zip*" containing all of your code for the initial version of the telegram bot and the flask server.

Part 2

You should submit:

- A **zip** file named "*project_id1_id2.zip*" containing all of your code.

More details about the requirements for the submission of Part 2 will be published in the coming weeks.

Good Luck!