EZ Budget

Contributors:
Diti Gupta, Logan Brown, Jillian Bartz, Rishi Hancock, Adam Fox, Will Davidson

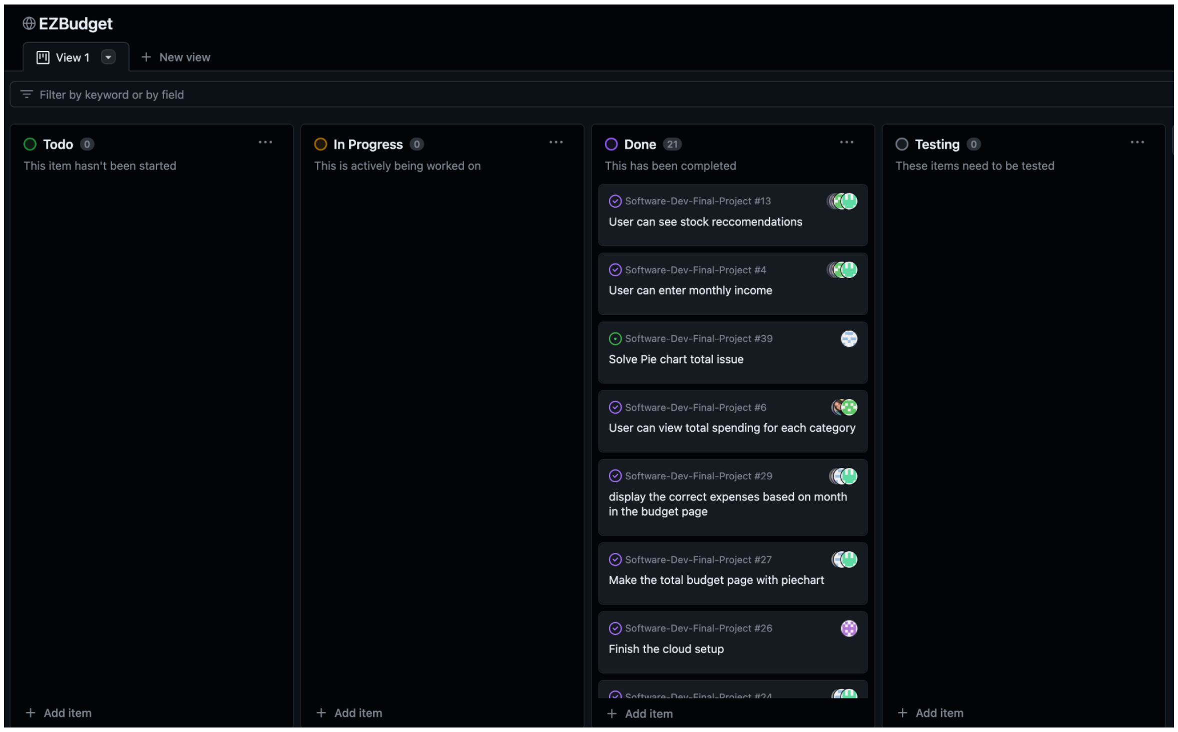
Description:

EZBudget is an application that allows users to track their monthly income and allocate spending to assist users in managing their monthly and yearly budgets. The application will allow users to create an account that will store their personal budget information which can be accessed through the login page.

EZBudget will ask users for monthly incomes, as well as any spending that the user wants to include. There will also be individual categories for spending, including transportation, investments, transportation, and other monthly expenses. The application will provide the total sums of their personal spending for each category on their budget tab, as well as their net profit for each month and year to date.

The EZBudget application is extremely useful for any individuals looking to become more aware of their spending habits. This application can be a quick resource for individuals to see their budget without having to independently compute their expenses or incomes. EZBudget will provide simple information regarding the user's categorical spending habits, allowing for more opportunities to successfully manage their finances in the future months. The application will include monthly tabs that can show the user's overall progress as well as a yearly tab that can show the user their total expenses and income.

Project Tracker:
https://github.com/users/willdavidson05/projects/1/views/1



EZBudget

View 1 | New view

Filter by keyword or by field

Todo 0 — This item hasn't been started

In Progress 0 — This is actively being worked on

Done 21 — This has been completed

- Software-Dev-Final-Project #13 — User can see stock reccomendations
- Software-Dev-Final-Project #4 — User can enter monthly income
- Software-Dev-Final-Project #39 — Solve Pie chart total issue
- Software-Dev-Final-Project #6 — User can view total spending for each category
- Software-Dev-Final-Project #29 — display the correct expenses based on month in the budget page
- Software-Dev-Final-Project #27 — Make the total budget page with piechart
- Software-Dev-Final-Project #26 — Finish the cloud setup
- Software-Dev-Final-Project #24

Testing 0 — These items need to be tested

Add item

Video:
EZBudget Demo Video (this video is also added to the github repository. If viewing it from github repository, please download the video from there to watch it)

VCS:
https://github.com/willdavidson05/Software-Dev-Final-Project

Contributions:
Please note: A screenshot of the project board is shown in the project Tracker section of this document.
Logan - Worked with team members to design and deploy the database used in the backend. Also helped with functions such add /add expense and others to pass data from the front end to the database.

Diti - Worked on backend/frontend work with addExpense/deleteExpense functions. I also worked on creating the ejs month pages and the overall structure of the website pages. Additionally, worked on the pie charts for each month page to enhance the project's functionality and visual appeal.Also, worked on the test cases to make sure all the data was being inserted/deleted into the database correctly. Helped with deployment on Azure.

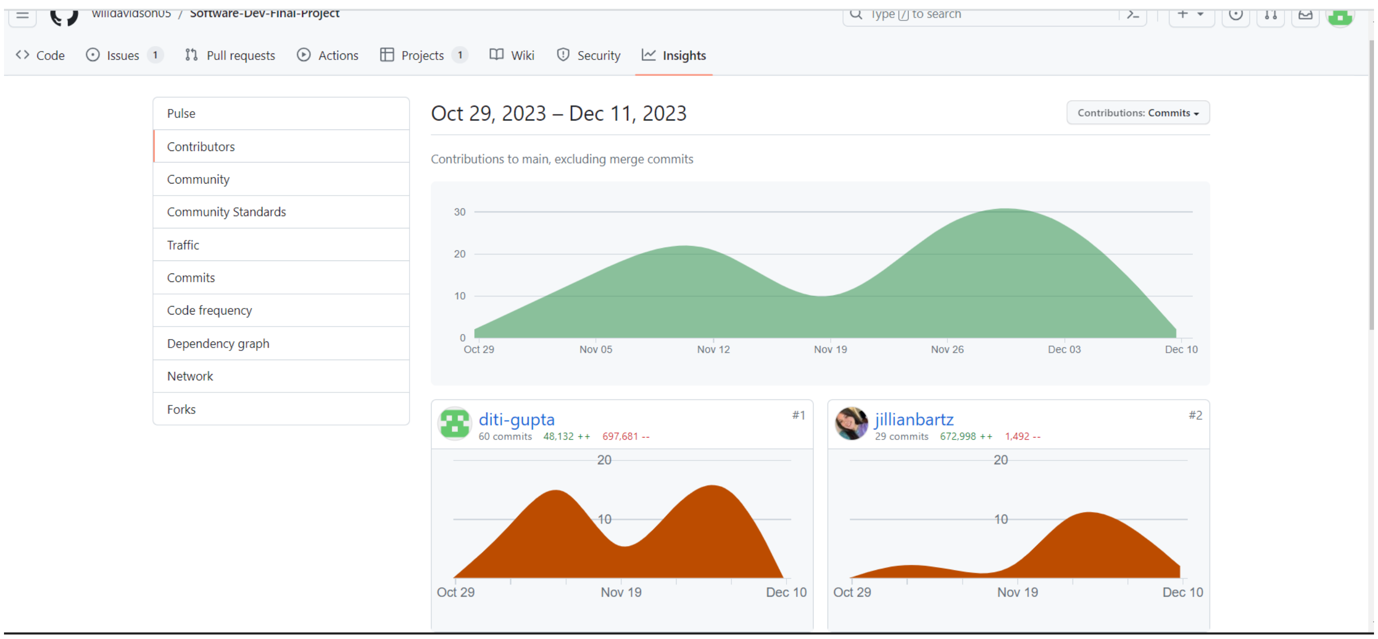
Adam - Helped alongside team members to design and deploy the database, assisted in creating ejs files for each month and additionally added an interactive breakdown to display the monthly breakdown of expenses.

Rishi - Helped with test cases and deployment on Azure.

Will - Worked on the design and implementation of the database used for the backend. Also worked on the addExpense/deleteExpense to correctly update monthly data.

Jillian - Developed and designed the front end of the website, including the CSS styling sheet for the home, login, register, budget, and logout pages. Assisted in creating ejs tabs and structure of website wireframes.

Note: Screenshots of the contributions on GitHub is shown below

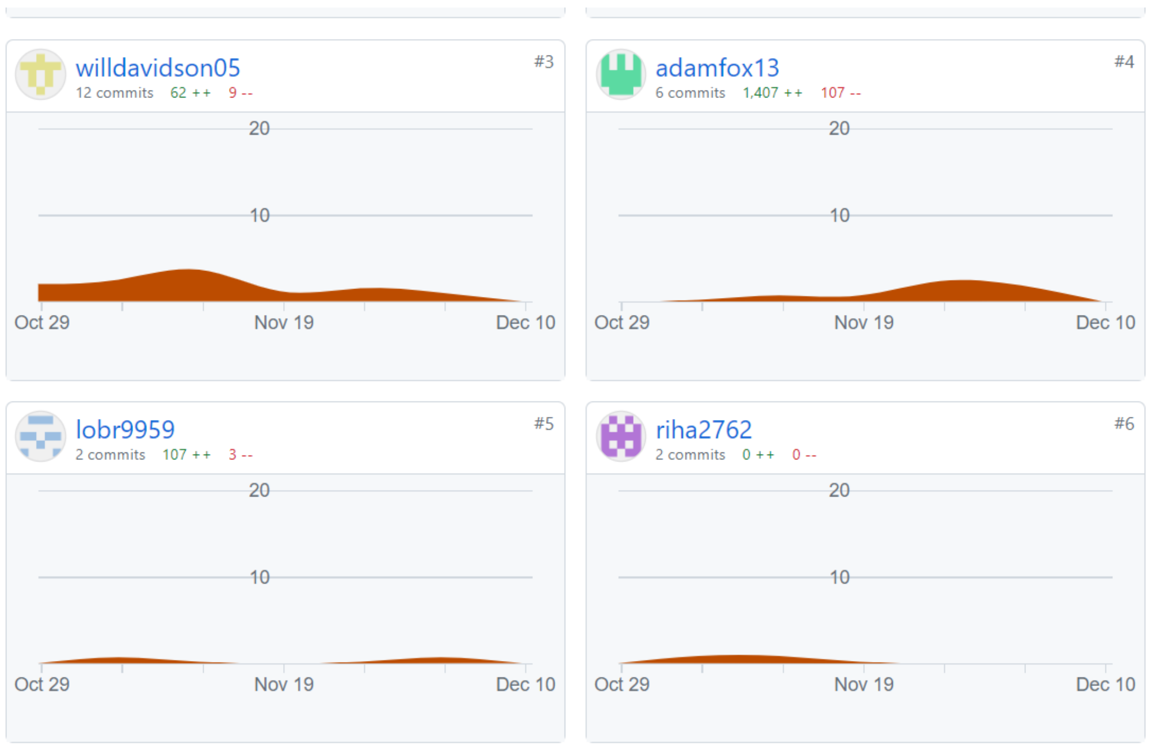


Oct 29, 2023 – Dec 11, 2023

Contributions to main, excluding merge commits

diti-gupta #1
60 commits 48,132 ++ 697,681 --

jillianbartz #2
29 commits 672,998 ++ 1,492 --



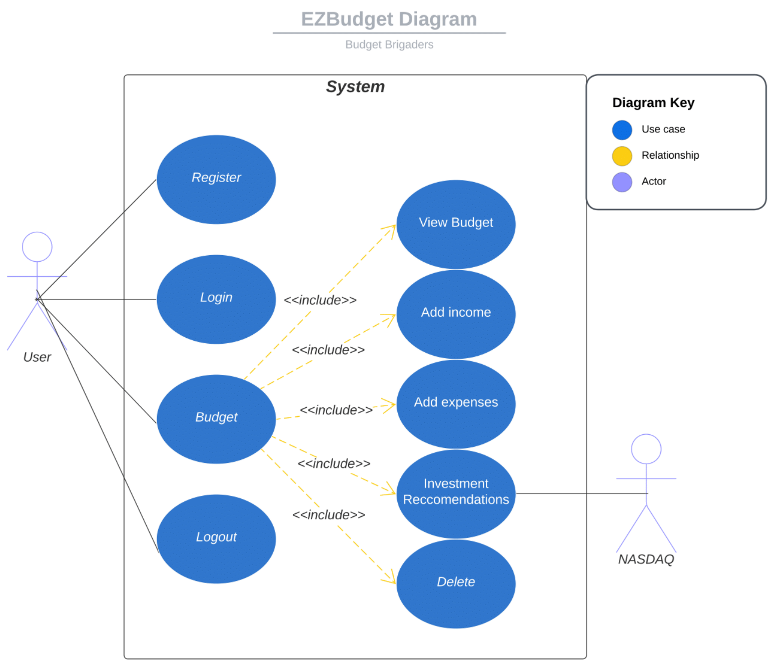
willdavidson05 #3
12 commits 62 ++ 9 --

adamfox13 #4
6 commits 1,407 ++ 107 --

lobr9959 #5
2 commits 107 ++ 3 --

riha2762 #6
2 commits 0 ++ 0 --

Use Case Diagram:



Test Results:

Register Test case: Ensure the user can register a new account using a unique string username and unique email.

1. **Positive Register test case:** Check that the username inserted does not exist in the users database. If the user does not exist, register the user into the database and give status 200. **Please note:** we created a delete/unregister test case as well so that the next time one is running the test cases all test cases can pass. (Take a look at the unregister test case for more details)
2. **Negative Register test case:** Check that the username inserted already exists in the users database. And if the above does occur, print "User already exists"

Observations:

- The user inputs a username and password into the input sections of the register page. If the username exists, a message pops up: "User already exists", and the user is then able to navigate to the login page instead where they will be able to log into their already existing account. If the user is not registered, after inputting the username, and password, the user is redirected to the login page where they can now login into their new account.
- In general, the behavior is consistent with the use case, and there is no deviation from the expected actions.

Login Test case: Ensure that the user can login successfully after creating a new account or login into a previous account.

3. **Positive Login test case:** Check that the user inputs for username/password match with data from the users table from the database. If the inputs for username/password match with the database username/password, redirect the user to the user's budget page, and return the 200 status.
4. **Negative Login test case:** Correctly handle invalid login attempts (example: Attempting to log in with an invalid combination of username and password that is not stored in database) by rejecting unauthorized access and providing appropriate feedback to the user.

Observations:

- The user inputs a username and password into the input sections of the login page. If the username exists in the database, check if the hashed password entered by the user matches the hashed password stored in the database. If the username and password match, log the user in and store their session. If the username entered does not exist in the database, redirect the user to the register page instead. If the user input for username matches but the hashed password does not match the stored password from the database give an error message: "Incorrect password"
- In general, the behavior is consistent with the use case, and there is no deviation from the expected actions.
- In general, the behavior was consistent with the use case, and there was no deviation.

Unregister Test case: Unregistering/deleting the user for the register test case, so that the positive user registration test case can pass any time one runs it.

- If one were to have run the test cases once already with the example of username: newuser, and the positive register test case passed, then the next time one runs the test case with the same username: newuser, the positive test case will fail, because in the previous test case run, the newuser gets added to the database because they did NOT exist at that moment in time. In the rerun, the newuser exists in the database, thus the username has to be a different username for EACH rerun of the test cases.

- So that there is no need for the positive test case to be updated with a new user EVERY TIME one returns the test cases, we have implemented an unregister endpoint that allows for the registered user from the positive register test case to be unregistered after it passes the positive register test case. In other words, we are first registering a new user, and the positive register test case should pass however we are also unregistering the newly registered user right after the positive register test case so that there is no need to register a new user each time, and so that the positive register test case can pass each time as we have registered the new user then deleted/unregistered it from the database.

Deployment:
http://recitation-12-team-04.eastus.cloudapp.azure.com:3000/