Divvi

Senior Design Final Report

University of Cincinnati

College of Education, Criminal Justice and Human Services

School of Information Technology

Timothy Brown

Kenneth Williams

Jeremy Mazurowski

Kyle Brady

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology/Cybersecurity

Timothy Brown Kenneth Williams Jeremy Mazurowski Kyle Brady 1/13/2022

, Faculty Advisor 1/13/2022

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2022

Table of Contents

Abstract

Divvi is a web application that allows users to create groups and split expenses and transactions through multiple splitting methods with the people in those groups. As individuals with busy and diverse lives, we have certain bills and payments that we must make to distinct groups of people or organizations. Expenses like rent, ride-sharing services, food, groceries, favors, electricity, internet service, utilities, etc. can be difficult to keep track individually, but it is more of a challenge, especially with groups of people. Each of us has experienced forgetting to pay an expense, not paying the right amount, and the consequences of not paying these payments. Using different technologies, our group was successfully able to create a client-server application that makes sharing expenses more manageable, less stressful, and easier all together. In the end, the creation of users and groups were successful, and transactions can be split by the three payment methods. This means that transactions and payments can be managed more easily, relieving the stress of having to chase people down for payments.

Introduction

Project Name: Divvi

Project Summary

Divvi is a web application that will allow users to create a group with other users and split a payment between the users in that group. Expenses can be split in three different ways (Fixed, Percentage, and Evenly). Expenses can be split between as few as one other member or as many members as needed. This will allow users to manage the problems that come with combined bills and splitting expenses.

Problem Statement

Typically, when paying a bill or any recurring expenses, the users would have to combine funds before and have one user pay the recipient or have all the users pay the main recipient individually. When an individual has multiple payments that they need to split amongst a variety of friends, it can become hard to manage and keep track of those payments and have an accurate log of those payments.

Solution

With our problem statement in mind, our team wants to create an application that allows users to enter expenses and determines the amount due between a group of users. This application allows groups to have flexible oversight by allowing a unique user view to signed in users. Payments outside of a group (i.e., between two individuals) will also have the same flexibility when it comes to splitting expenses. The application will split payments into three options: Percentage, Evenly, or Fixed Amount. There will be security measurements in place for the existing users on the application, such as database encryption for their credentials. The application handling calculations will take the complexity and time off groups and individuals who create an expense in the application. This will make the overall process for splitting expenses easier and faster while decreasing the complexity for the end users. Divvi relieves end users from the stress associated with contributors forgetting about payments and having to constantly remind them who they owe payments to.

Project Source

Through individual experiences, our group has noticed there are a lack of easy rent payments through a portal system. Kyle Brady conceived the project idea. As a group we all agreed on the requirements together and which key features we needed to focus on. Our group is two cyber majors (TJ and Kenny), and two software developers (Jeremy and Kyle). Kyle and Kenny met through class, and Kenny and TJ had a co-op together and are friends. Jeremy joined the group on the first day of Senior Design when they presented the idea.

Discussion

Project Objectives/Goals

The goal of Divvi is to solve the disorganization that comes when splitting rent, or any other recurring payments/bills, upon a group of individuals. Divi will help avoid confusion about splits or payments for any bills or expenses individuals may have together. Some of the major features for Divi include:

1. Ability to create a group of users.
2. Ability to split payments among the group with different payment splitting options.
3. Allow users to keep track of shared expenses and balances.
4. Ability to create multiple groups per user and add friends for frequent transactions.
5. Ability to process payments using a third-party payment processing API.

Project Scope

Our team will develop a functional application that splits expenses and enables users to track their transaction history. We will also create a database that stores users' username, password, etc. Divi will manage all the calculations associated with splitting the bill(s). The other group members, who did not create the expense, will be able to change and update the details of an expense within a group. Individual bills, outside of a group, will have flexibility and allow either members of the exchange to change the details of the bill or split. The payments will have the option to be split into 3 ways. Payments can be split by equal amounts, a percentage of the amount, or an exact amount for each user in the group" Users' payment information will not be held or hosted on our application. We expect our toughest hurdle will be designing the individual user views so that users can see their individual transactions. Some items out of the scope of this project include adding friends on the application, handling payments within our application, and scanning receipts and reading bills.

Quick Project Timeline

Based off our technical skills, we created our timeline to give more time to the front end and back-end development of the project. Along with the development of the front end and back-end, we also had to take into consideration how users will interact with the web application.

Task #	Task Name	Duration	Start Date	End Date
1	Project Skeleton is created	2 weeks	10/1	10/14
2	Payment Options (Split equally, %, and exact amount) are functional	2 weeks	10/14	10/28
3	Database Creation	1 Week	10/21	10/28
4	User creation (Credentials can be stored)	1 Week	10/28	11/4
5	Ability to create and add users to a group	2 Weeks	11/4	11/18
6	Fixed inputs based on amount being split	2 weeks	1/17/2022	1/31/2022
7	User Login Page	2 weeks	2/1/2022	2/6/2022
8	Adding User views to Divi Application	3 weeks	2/7/2022	2/21/2022
9	Add pending transaction data to Database	4 Weeks	2/22/022	3/21/2022
10	Create Multiple Groups/New Groups	1 Week	3/22/2022	3/28/2022

11	Add members to an existing group	1 Week	3/22/2022	3/28/2022
12	Add Expenses to Individuals within a Group	3 Weeks	3/29/2022	4/4/2022

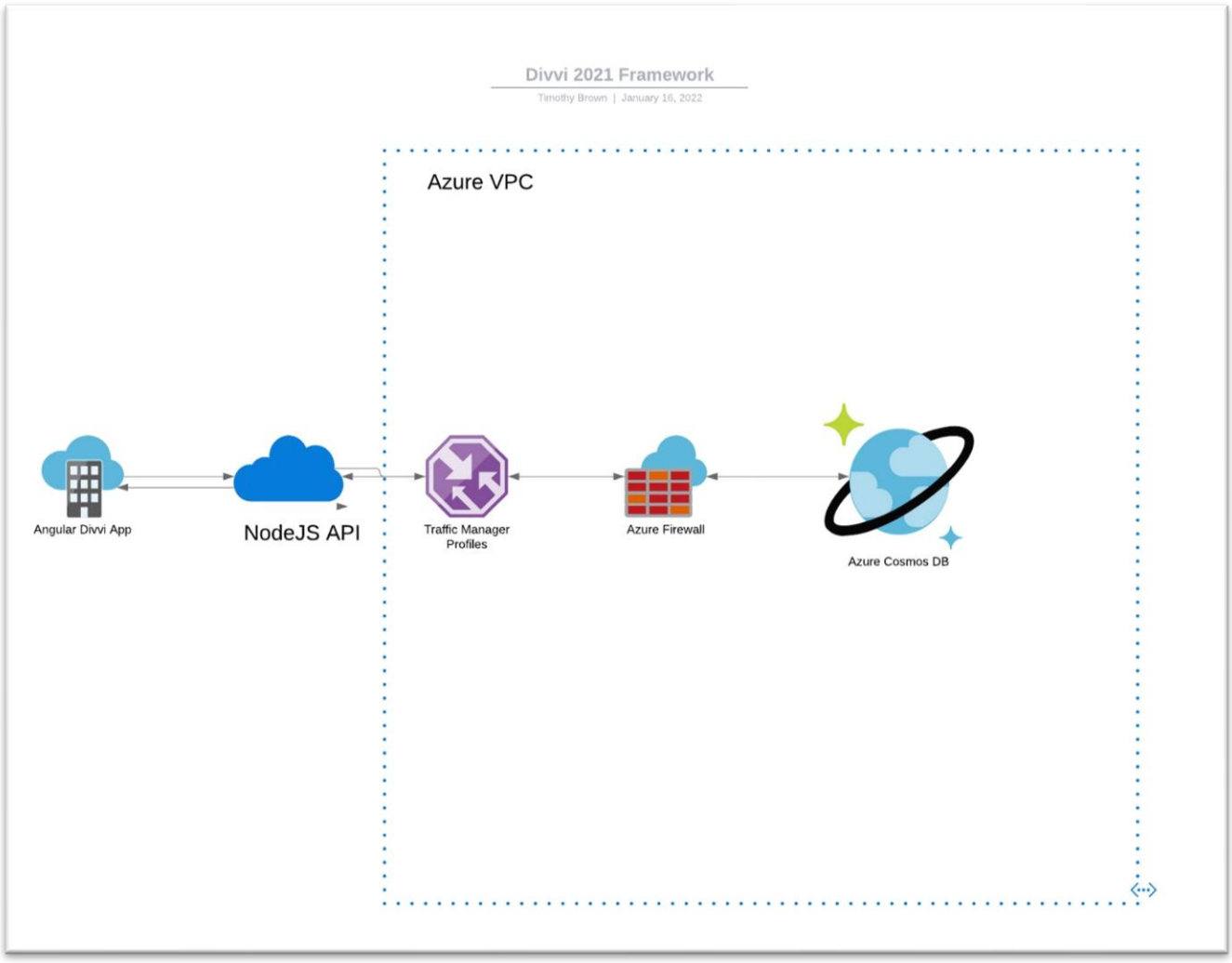
Technologies Used

With Divvi being a web application, it that utilizes a small number of technology and software with complex systems and capabilities. We decided to use these technologies because of our familiarity and connectivity. These technologies will help us to design, develop, and integrate all our desired features and capabilities into the web application.

Angular – Angular is used to create the **front end** of the site.
Node.js: Our **back-end** API to handle requests to the
Azure SQL Server - We will create our **database** in SQL server to store users' information and other data for our website.
VSCode - We will use visual studio to develop our back end in C#, and Visual Studio code to develop our front end in angular.
GitHub - we will use GitHub to collaborate on the coding projects and centrally host our source code.

Technical Architecture Diagram

The application Divvi has a front end for user inputs that collect the necessary data for expense splitting and transaction history. The NodeJS API sends and receives the data between the front end and the back end to display and collect data. The traffic manager and firewall are security measures that are in place to protect the data inside of our Azure database. The Azure database holds and stores user credentials and the necessary information for the expense calculations.



User Personas

These user personas cover different scenarios, lifestyles, and individual backgrounds that can benefit from using Divvi. The application can be used by a wide variety of individuals with diverse backgrounds.

Table 1 User Persona Table

<table>
<tr><td colspan="2">User Persona: 1</td></tr>
<tr><td rowspan="4">Picture (can be an animation, does not have to be a real person)</td><td>Title: Him / his</td></tr>
<tr><td>Name: Chris Stewart</td></tr>
<tr><td>Age: 21</td></tr>
<tr><td>Gender: Male</td></tr>
<tr><td>Behavior</td><td>Chris is living with roommates in college and has expenses due monthly. He is very organized however his roommates are not and he struggles getting their payments on time.</td></tr>
<tr><td>Pain</td><td>Chris is in charge of sending his monthly expenses to his landlord. With his roommates struggling to make payments on time Chris will occasionally have to spot them money, or explain to the landlord why the payments are late</td></tr>
<tr><td>Needs & Goals</td><td>Chris goals are to remain as organized as possible and to have payments sent on time to his landlord. He needs an application that can easily track who has not paid their dues.</td></tr>
</table>

<table>
<tr><td colspan="2">User Persona: 2</td></tr>
<tr><td rowspan="4">Picture (can be an animation, does not have to be a real person)</td><td>Title: She / Her</td></tr>
<tr><td>Name: Amanda Smith</td></tr>
<tr><td>Age: 26</td></tr>
<tr><td>Gender: Female</td></tr>
<tr><td>Behavior</td><td>Amanda lives with her boyfriend, and they continue to split monthly dues for their rented apartment. Amanda and her boyfriend also enjoy going out with their friends. To make things easy, when they go out together, one person picks up the tab, whether that be at dinner or a bar.</td></tr>
<tr><td>Pain</td><td>Amanda is an organized person and likes keeping track of her payments. She has found when it is her turn to pay for nights out with her friends, it is hard to keep track of who has paid. Also, payments can get confusing, and it can be hard to remember why someone owes her money.</td></tr>
<tr><td>Needs & Goals</td><td>Amanda’s goals are to remain as organized as possible and to track who has and hasn’t paid when she goes out with her friends. She wants it to be clear exactly why she is requesting the money and to create a group as it is the same friend group that goes out together.</td></tr>
</table>

Use Cases

Table 2 Use Case Table

Use Case ID	001
Use Case Name	Creating a Group
End Objective	Have a group created with a group manager
User/Actor	Group Manager or User
Trigger	The user needs to split a payment with users for a specific item, event, etc.
Frequency of Use	80% of users will create a group to split payments and track payments.
Preconditions	Have an account created
Basic Flow	Creating a group where I can add x amount of people and be the group manager
Alternate Flow	N/A
Postconditions	The group manager will add users to the group.

Use Case ID	002
Use Case Name	Splitting a payment
End Objective	Split a payment a specific way (evenly, percentage, fixed amount)
User/Actor	User or Group Manager
Trigger	Having a payment that needs split
Frequency of Use	100% of users will split a payment while using the application. That is the main function of this application.
Preconditions	Have a group created already.
Basic Flow	Splitting a payment by x method amongst y amount of people
Alternate Flow	Splitting a payment by percentage or a fixed amount amongst x amount of people
Postconditions	The payments will be paid after the split (outside of the application).

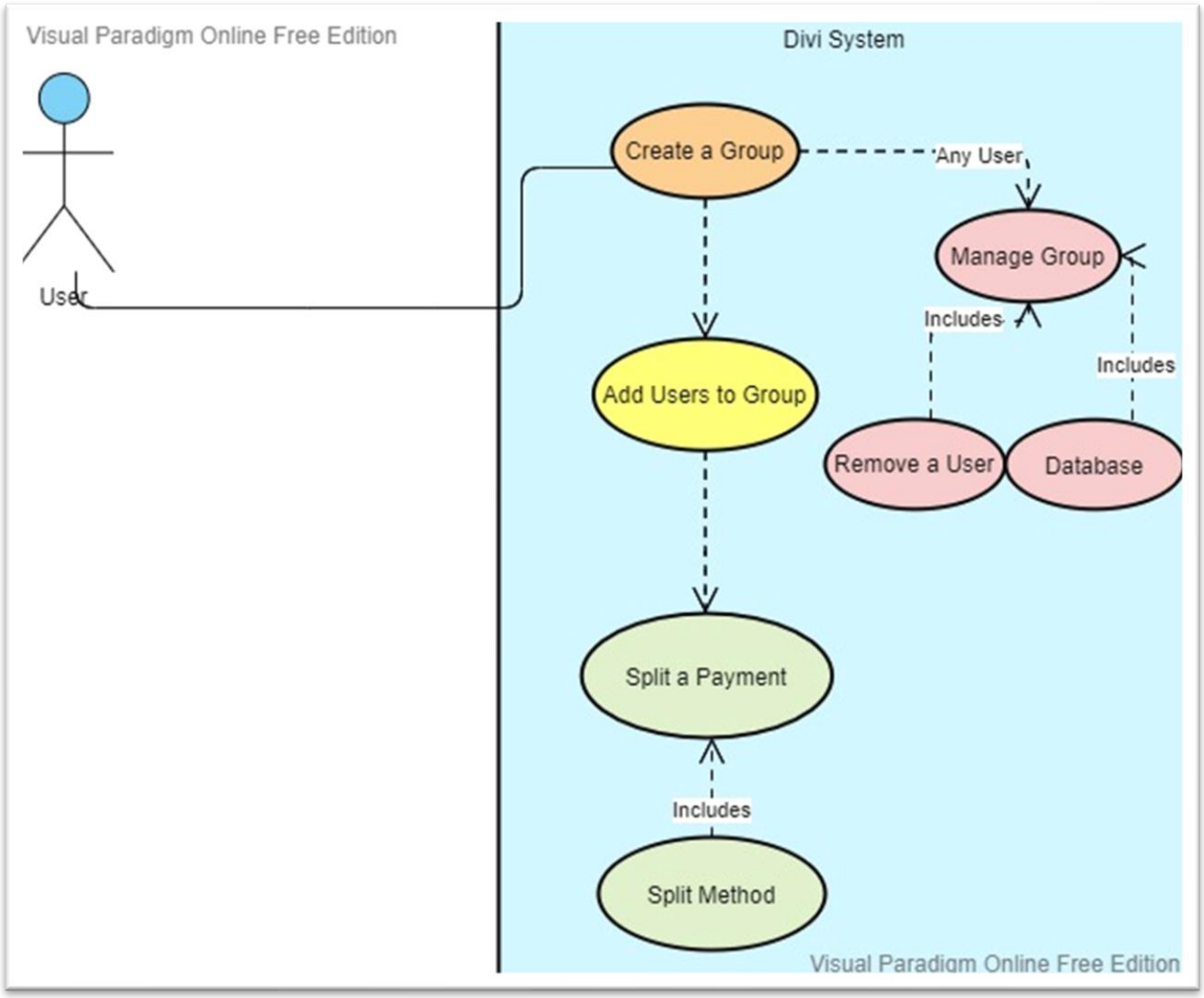
Use Case ID	003
Use Case Name	Adding group members
End Objective	Adding/Inviting group members.
User/Actor	Group Manager or User
Trigger	Aa user needs to split a payment with multiple people
Frequency of Use	99% of users will add a member to their group while using the application.
Preconditions	Having a group created already and the group members must all have accounts
Basic Flow	Adding a member to the group
Alternate Flow	Adding members during the group creation process.
Postconditions	The group will split payments in the future.

Use Case ID	004
Use Case Name	Managing a Group
End Objective	Be able to remove/manage individuals in the group
User/Actor	Group Manager
Trigger	The group manager wants or needs to make a change to the group
Frequency of Use	80% of users will find themselves needing to manage a group.
Preconditions	Be the Group Manager
Basic Flow	For simplicity we will cover removing members
Alternate Flow	Making changes to existing payments, etc.
Postconditions	The person removed from the group will no longer have access to the group.

Use Case ID	005
Use Case Name	Paying through the 3rd party API
End Objective	The users can pay their payment split through 3rd party applications using the API
User/Actor	Users and Group Managers
Trigger	Users need to make a payment
Frequency of Use	100% of users will find themselves needing to pay their part of the split.
Preconditions	Already have a payment split through one of our methods
Basic Flow	The users will click the correct 3rd party button(s) and be redirected to the 3rd party payment application(s).
Alternate Flow	The users directly visit the 3rd party application without using our application.
Postconditions	The users will pay using the 3rd party application and we are not responsible for the actions on the 3rd party application.

Use Case Diagram

This diagram shows how users can interact with our application. The users can create and manage groups to handle and track their expenses. The application will handle the payment splitting logic based on which method the user selects.



Testing Plan:

Overview

The topics covered in this section include the testing approach and methodology followed, the scope of the tests and the reason the significance of those areas to the application workflow, testing goals and criteria, test logs, and a review section of the process and how potential areas of growth and improvements.

Methodology

When developing Divvi, our group decided on unit testing when working in our own separate branches of the project. With GitHub, each of the group members can have their own separate copy of the main project that each member performs their assigned task. As we are working and making changes to our own unit branches, each group member will run their changes to ensure that the project won't have any errors. Once, tested, the group member can push their changes into the main project which is in GitHub. This form of testing ensures the main project will be safe from errors while also making significant progress. Once the application is nearing completing, our team will perform an integration test to ensure all APIs and other added external features are running properly and efficiently. Finally, once the unit tests and integration tests are complete, our team will perform user acceptance testing. Using the use case methods with different users, our application will be tested to ensure each feature mentioned in the problem statement work as expected.

Scope

The goal of Divvi is to help people manage any expenses or payments between groups of people and tracking that payment data. With Divvi, users will be able to login, add any number of users and assign them to a group. This will allow for easy management of expenses. Divvi will also allow users to enter an amount into the system and be able to split it multiple ways to their assigned group (Evenly, Fixed, and Percentage). Lastly, Divvi will be able to track and record all expenses in the database for users to keep track of payments.

The following is a list of the major features that our application will have. Our team will test each of these features to ensure they work as expected in the final application.

- Login / Logout feature
    - With expenses being logged for groups of users, data integrity is important. The ability to have unique user views for each user will help mitigate data manipulation.
- Creating a group
    - Creating groups is a huge part of our application. Our goal is to make requesting money from people easy, especially if you are requesting from the same group multiple times.
- Splitting Payments Logic
    - It is important for our application to properly split payments per the user's input. We will have 3 payment splitting options – evenly among your group, a percentage for each user in your group, or a fixed amount to each user. Unit testing the payment logic will ensure that all 3 ways of splitting the payment work as they should
- Managing groups
    - Being able to manage a group will be important for users. The ability to add / remove users from a group is a key feature of our application and will need to be tested.
- Database testing
    - Ensure user data can be stored in a database with secure encrypted data such as passwords and other sensitive data.

The following is a list of all the objectives that need to be done before the IT Expo in April.

a. All major features and use cases need to be accounted for
b. All use cases must account for all the user roles
c. All bugs need to be resolved before IT Expo
d. All API integrations must be integrated by the end of the semester
e. Database must be functional and able to hold user data by the end of the semester – (Encrypted passwords)

End Goal: Have a fully functional secure application by the IT Expo

Test Logs and Procedures

Item Number	Test Description	Use Case # (If Applicable)	User	Role	Expected Results	Actual Results	Pass/Fail	Comments	Date
1	Application can add users to an existing group	4	Jeremy	Full Stack Developer	Application can add existing user to an established group	A User was successfully Added	Pass		10/21/2021
2	Evenly Payment option working as intended	2	TJ	Full Stack Developer	Application can properly split payment evenly	The Evenly payment option works correctly	Pass		10/28/2021
3	Percentage Payment option working as intended	2	Kenny	Full Stack Developer	Application can properly split payment by a specified percentage	The Percentage payment option work correctly	Pass		10/28/2021
4	Fixed Amount Payment option working as intended	2	Jeremy	Full Stack Developer	Application can properly split a payment by a fixed amount	The payment option only split for one group member	Fail	The payment did indeed split but the other user's amount did not adjust accordingly.	10/30/2021
5	Back End database retrieves user data along with User Id and Group Id.	3	Kyle	Full Stack Developer	The back-end database receives user inputs and updates the back-end database	The database successfully receives new user along with the User Id and Group Id.	Pass	Any new users added in the front end show up in the back end along with their group ID.	11/4/2021
6	Divi pulls up separate Dialog Box for	2	TJ	Full Stack Developer	When it is time to input a payment,	Divi successfully pull up separate	Pass	Divi will ask for users for a name,	11/14/2021

	inputting payments				Divi will open a separate dialog box for input	dialog box for input from user		description of the expense, type of payment split, and will ask the user to adjust the payment accordingly.	
7	Back-end Database can log pending transactions	2	Kyle	Full Stack Developer	Divi will now log the transactions that are being logged in the front end.	Divi displays the pending transactions in a table in the back end	Pass		11/23/2021
8	Divi will now have a login in screen for all users accessing the application	2	Kenny	Full Stack Developer	In order to access Divi, users will now need to login with a username and password	Divi displays a login in page before users can begin using any aspect of the application	Pass		1/14/2022

Testing Review

By unit testing our group learned the importance of testing. Going through each component and ensuring that the work we had just worked on was able to be implemented to the main branch in GitHub was vital to keeping the application clean and functioning as it should. While we did not have to go back and re-work any functions in our code, the quality of the code was lacking a little bit. At the end of the first semester the team went back and cleaned up some code to fit Angular and coding standards in general. Also, during development, we underestimated how complicated implementing the different payment splitting options was to our back end and front end. Our team faced a lot of trial and error in ensuring that all the payment options work correctly while also ensuring the input data was updating and displaying correctly in the back end.

Change Management Plan

Anyone on the team can request a change to our application. Also, advisor Gilany can request a change. The circumstances needed to request a change is if this change does not fall in line with the project scope, technologies used, or problem statement. For example, there is no need for someone to request a change to change a method in the code. The whole team has access to the GitHub repository, and each team member works out of their own branch. However, if someone in the group finds a better API that is easier to integrate into the project that will be something that needs to be brought up amongst the team and talked about before making a final decision.

Changes are prioritized using the JIRA board. If a change is needed, the first step is to look at the JIRA board. If the card needing changed is already in progress, or complete, then the prioritization for this change will be higher than a change for a card in the backlog. The pros of this are that the team can see exactly what card needs to be changed and where in our project calendar this change will take place. A con to this is that it will require flexibility and not everyone may be on board with the change. Therefore, for a change to be approved, the team will need a majority vote, or 3 out of 4 members of the team will need to agree to this change.

Changes will be communicated with Professor Gilany if there are plans on making a change. He has a lot of insight and expertise in the field and his opinions on the change can only help the application. The team will lay out exactly what we are planning to change, why we are changing it, and the benefits of the change. Then, we will hear the feedback of our advisor on whether he thinks this change is a good idea. Using his feedback, the team will reconvene and decide if making this change is a good idea and take the vote. If 3 out of 4 of the team agree to the change then the change will be made.

Budget

While your project is currently being built at no cost, building a budget to assess the cost of the project will help consumers and stakeholders understand the value behind it. Using the ROI worksheet, add the project budget table to this section along with an introduction on what was included in the budget and why. (While developing the project, we worked a total of X hours at a rate of 20 dollars an hour. The 1200 hours is per semester, so next semester will add an additional 1200 hours. This accounts for the hours spent developing the project throughout the semester. The ROI below shows how the Labor to develop the software plus our equipment costs affect the budget.

Table 3 Project Budget

Estimated Cost Rough Order of Magnitude:							
				Ongoing Annual			**Comments:** GitHub enterprise and AWS costs are annual. Bi-weekly paychecks for 4 individuals
	Rate Per/ Hr	**1st Semester Hours**	**Rate Per/Hr.**	**2nd semester Hours**	**Total Hours**	**1 X Support Cost**	
Labor - IT	25	1,500	25	1,500	3,000	$-	
Labor - External			$-		0	$-	
Software - External						$352.00	
Hardware - External			$2,800.00				
Misc.							
TOTAL			$5,800.00			$352.00	
5-Year ROI Analysis							

The hardware section includes our laptops because that is the only hardware we are using to develop Divvi. The annual costs include the annual costs of AWS and GitHub Pro because although we do not pay for them.... The school is paying for it, so a real company needs to include these types of costs in their budget. The 1200 hours is per 1 person in the team. 1200 x 4 = 4800 is the total team's contribution.

Problems Encountered and Analysis of Problems Solved:

Throughout every development lifecycle, there will always be problems encountered. This section goes in depth on what problems our team encountered and the solutions to each of them.

Problem 1: There was a lack of communication and clarity around the goals and team progression for the project at the start.

We solved this problem by deciding to have one to two weekly meetings to discuss our problems, progression, etc. and to make sure we are still aligned with the initial goals of the project. We are also tracking all of project progression with Jira. Jira has made it a lot easier to see which team members have what task and keeping track of tasks/ideas for the future.

Problem 2: The payment splitting logic.

For the fixed amount and percentage splitting logic, it was difficult to store multiple user inputs into a variable that needed to be pushed to the database based on each user index. We fixed this problem by storing each user input to an array based on the user index. This ensured that the index for both arrays would align with each other. This allowed us to push the data to the database on the same index.

Problem 3: Setting up the API to connect the back-end with the front-end

There was difficulty setting up the API to connect our back-end with our front-end because of the Azure firewall rules. We had to allow multiple domains within the Azure firewall rules. We also had to whitelist each person within our group inside of the Azure firewall, so that each member of the group could access the data within the database using the API.

Problem 4: Updating the database information with the user input from the Angular front-end.

Angular has a function to prevent Cross-Site Scripting called CORS. CORS was blocking our API requests to the database. None of us were familiar with the Angular CORS function, so it required a heavy amount of research to find the solution to our problem. Eventually we found the solution... We needed to incorporate multiple domain origins within our CORS rules inside of Angular, so that Angular could process requests from multiple domains (localhost:4200, localhost:8900, etc.).

Problem 5: No clear future vision or layout for the project.

We met with Yahya to discuss the future of our project for this semester and next semester. Yahya was a huge help for envisioning how we need to layout the front-end of Divvi, as well as what needs incorporated into the back-end in the future. Since that meeting, we have incorporated ideas, goals, features, etc. that align with the vision that our group discussed with Yahya.

Conclusion

Some of the key lessons that we learned this semester is that it is important to layout the plan and expectations early in the project, so that everyone has a clear picture of what the goal of the project is. It is also important that we are communicating our problems and progress on at least a weekly basis. We enhanced our knowledge of Angular and its components, such as ngModel, the different loops like ngFor, NgIf, etc. We learned how to integrate an API into our application so that we can make requests for data that sits within an Azure SQL database. This semester, we focused heavily on the payment and splitting logic. We learned quickly that the logic was more difficult than we originally thought because of multiple user inputs being pushed to a variable based on the index.

In the future, we want to focus on adding a transaction table to our database so that we can track the group transactions. This will allow Divvi to keep a running tab of the transaction history for all the expenses within a group. Also, in the future, we are going to focus on the user experience and user views for the application. All the buttons on the front-end need to work, and we need to give the front-end a cleaner look that is more user friendly. Users should only see their transactions and groups within their user view.

References

Angular.com. [2022]. *The modern web developer's platform*. NA. Retrieved April 2022 from: https://angular.io/

Bootstrap.com. [2022]. *Build fast, responsive sites with Bootstrap*. NA. Retrieved April 2022 from: https://getbootstrap.com/