

IT701 Capstone Project

DivvyUp The future of Payments

Poni | Yuhei | Oliver

DivvyUp Technologies

# Abstract:

The evolution of payment technology depends on the willingness of both the user and the vendor to embrace change and interconnect with a tangible need for transformation.

A surging technology industry and the unprecedented constraints of COVID19 responses have created a unique opportunity for innovation. Meeting the payment requirements of the future in a post COVID environment, DivvyUP offers the future of payments. The effect EFTPOS had on cash, DivvyUp will have on EFTPOS.

While solving an age-old hospitality conundrum and embracing the high demand from consumers for improvement in fast, efficient, contactless, and secure payments, DivvyUp has innovated QR code technology and BYOD philosophy to provide for the user all of the flexibility of a POS system in the palm of your hand and for the vendor a less costly and elegant solution for paying the bill.

DivvyUp provides the next way to pay! It is a remarkable innovation that will provide vendors with an added feature to enhance the customer experience providing a simple interface that is a win-win for both the customer and the business.

# Acknowledgments:

Firstly, we would like to express our gratitude to our Course Coordinator, Anita Murphy, for her patience, motivation, and enthusiasm. Her willingness to provide latitude for the commercial viability of DivvyUp has given the team the power to freely develop, adapt, and alter our product to what we have created.

We would like to pass our thanks to our team mentor Ken Sutton, who as both inspiration and a grounding force has provided us with invaluable guidance toward a product we are proud of.

In equal abundance we would like to pass our appreciation to Johnny Mackenzie, Matt Hunter, Bharat, and Ange for their eagerness to support our project and provide industry expertise in development and implementation.

Last and by no means least we as a team acknowledge the individuals who have provided their time, energy, and skills to our project, with this we would like to thank the team itself for its unending diligence. Thank you Ameil Burke, Oliver, Yuhei, and the class of 2022.

Table of Contents

[Abstract: 2](#_Toc116390332)

[Acknowledgments: 2](#_Toc116390333)

[Project Outline 8](#_Toc116390334)

[Problem 8](#_Toc116390335)

[Solution 8](#_Toc116390336)

[Target Market 9](#_Toc116390337)

[Competitors 9](#_Toc116390338)

[Competitive Advantage 9](#_Toc116390339)

[Business Model 10](#_Toc116390340)

[Value Proposition: 10](#_Toc116390341)

[Key Activities: 10](#_Toc116390342)

[Key Resources: 11](#_Toc116390343)

[Channels: 11](#_Toc116390344)

[Cost Structure: 11](#_Toc116390345)

[Revenue Streams 11](#_Toc116390346)

[Background 12](#_Toc116390347)

[Introduction: 14](#_Toc116390348)

[The Team: 15](#_Toc116390349)

[Roles and Responsibilities: 15](#_Toc116390350)

[Objectives: 17](#_Toc116390351)

[Benefits of Success: 17](#_Toc116390352)

[Methodology: 18](#_Toc116390353)

[Project Overview: 20](#_Toc116390354)

[Scope: 20](#_Toc116390355)

[Scope Exclusions: 20](#_Toc116390356)

[Critical Dependencies & Assumptions: 21](#_Toc116390357)

[Constraints: 21](#_Toc116390358)

[Current System/ Competition 22](#_Toc116390359)

[Comparison Bill Splitting Applications 23](#_Toc116390360)

[Existing System Ishikawa Diagram 24](#_Toc116390361)

[Proposed System 24](#_Toc116390362)

[Requirements: 25](#_Toc116390363)

[Description 25](#_Toc116390364)

[User requirements 25](#_Toc116390365)

[Developer Requirements 25](#_Toc116390366)

[Server Requirements 26](#_Toc116390367)

[System Requirements 26](#_Toc116390368)

[Functional Objectives 26](#_Toc116390369)

[High Priority (Mandatory) 26](#_Toc116390370)

[Medium Priority (Elective) 26](#_Toc116390371)

[Low Priority (Potential feature additions) 27](#_Toc116390372)

[Non-functional Objectives 27](#_Toc116390373)

[Reliability 27](#_Toc116390374)

[Usability 27](#_Toc116390375)

[Performance 28](#_Toc116390376)

[Security 28](#_Toc116390377)

[Supportability 28](#_Toc116390378)

[Online User Help 28](#_Toc116390379)

[Cost Components 28](#_Toc116390380)

[Interfaces 28](#_Toc116390381)

[Need 28](#_Toc116390382)

[Analysis & Design: 29](#_Toc116390383)

[Objectives and Success Criteria 29](#_Toc116390384)

[System Models 30](#_Toc116390385)

[User Flow Model 31](#_Toc116390386)

[Use Case Model 32](#_Toc116390387)

[User Story One 32](#_Toc116390388)

[User Story Two 33](#_Toc116390389)

[User Story Three 33](#_Toc116390390)

[Design Specifications 34](#_Toc116390391)

[The DivvyUp Modules 34](#_Toc116390392)

[User Types 35](#_Toc116390393)

[Design Constraints 35](#_Toc116390394)

[Financial 35](#_Toc116390395)

[Technical 35](#_Toc116390396)

[External Organisations 35](#_Toc116390397)

[System Architecture Issues 35](#_Toc116390398)

[Dependencies 36](#_Toc116390399)

[Risks 36](#_Toc116390400)

[Architectural design Goals 37](#_Toc116390401)

[Operational Environment 37](#_Toc116390402)

[Developmental Environment 37](#_Toc116390403)

[Development Methods 38](#_Toc116390404)

[Cloud strategies 38](#_Toc116390405)

[System Descriptions 40](#_Toc116390406)

[Server Requirements 40](#_Toc116390407)

[Program structure Design 41](#_Toc116390408)

[System & Security 42](#_Toc116390409)

[Performance 43](#_Toc116390410)

[Software 43](#_Toc116390411)

[Security Software 43](#_Toc116390412)

[Performance Architecture 44](#_Toc116390413)

[System Behaviour 44](#_Toc116390414)

[Data Description 46](#_Toc116390415)

[Data Design 46](#_Toc116390416)

[*Information Architecture* 47](#_Toc116390417)

[***Internal data structure*** 47](#_Toc116390418)

[***Global data structure*** 47](#_Toc116390419)

[***Temporary Data Structure*** 47](#_Toc116390420)

[Human Interface Design 47](#_Toc116390421)

[*Screen Images* 47](#_Toc116390422)

[Test & Implementation: 48](#_Toc116390423)

[Testing Requirements 48](#_Toc116390424)

[Description 48](#_Toc116390425)

[Testing Approach 48](#_Toc116390426)

[Testing Tasks Performed 48](#_Toc116390427)

[Function testing 48](#_Toc116390428)

[Unit Testing 48](#_Toc116390429)

[Integration testing 48](#_Toc116390430)

[Security Testing 48](#_Toc116390431)

[Review of Project Lifecyle: 48](#_Toc116390432)

[Challenges: 48](#_Toc116390433)

[Evaluation: 48](#_Toc116390434)

[Conclusion: 48](#_Toc116390435)

[References: 48](#_Toc116390436)

[Appendices 49](#_Toc116390437)

[Glossary of Terms: 49](#_Toc116390438)

Table of Figures

[Figure 1 Clare Riley Innovation Quote 12](#_Toc116390439)

[Figure 2 Ubuntu Conceptual Model from (Christopher Tworeck, 2017) 18](#_Toc116390440)

[Figure 3Current Hospitality Ishikawa 24](#_Toc116390441)

[Figure 4 The financial Brand Digital Wallet Quote 29](#_Toc116390442)

[Figure 5 User Flow Model Bill Payment 31](file:///\\sit.inet\SIT\Home\INVStudents\2020000011\MySettings\Desktop\DivvyUP\IT701Capstone%20(AutoRecovered).docx#_Toc116390443)

[Figure 6 Use Case DivvyUp Customer 32](#_Toc116390444)

[Figure 7 DivvyUp Architecture High-Level View 42](#_Toc116390445)

[Figure 8 Create a nested virtual machine in a Microsoft Azure Linux VM (Linkletter, 2018) 42](#_Toc116390446)

[Figure 9 Sequence Diagram DivvyUp Customer Experience 45](#_Toc116390447)

[Figure 10 State Diagram DivvyUp Pay 45](#_Toc116390448)

[Figure 11 Activity Diagram Paying Split Bill DivvyUp 46](#_Toc116390449)

Table of Tables

[Table 1 Comparison bill splitting Apps (Barry, 2021) (Barry, 2021) 23](#_Toc116390450)

[Table 2 Development Environment Framework 38](#_Toc116390451)

[Table 3 System Software Outline 40](#_Toc116390452)

[Table 4 Server Requirements 41](#_Toc116390453)

[Table 5 Architecture Performance Criteria 44](#_Toc116390454)

# Project Outline

The digital age is eliminating the need for cash or, in some cases, even credit cards at restaurants, as many customers now prefer to pay through technology. (Loria, 2018) DivvyUp is the latest innovation for the Hospitality sector in New Zealand.

## Problem

Although an Australasian only problem, large groups splitting bills has been a cultural taboo for modern restaurants and bars. Be it the time taken, or the inconvenience of missed items, or the hassle for both the customers and the vendors splitting the bill for large tables in Australia and New Zealand is a problem that has seen the majority of venues proffering a “One bill per table” Policy.

This is the way it has always been, is it really a problem? With COVID19 responses forcing a reconsideration of how we work, play, and pay an innovation that encourages all the benefits of a digital wallet while embracing the most technologically advanced age in history a solution that offers flexibility, security, and simplicity seems almost overdue.

## Solution

DivvyUp is a web application that utilises QR code technology to produce a digital table account (Bill). It creates an interface that is a pseudo-POS (Point of Sale System) and will allow multiple guests at the same time and in real time to choose items or nominate a total to pay. While advocating for payment through a digital wallet (Your Mobile Device).

A server does not have to meticulously “tick off” items on the bill, quickly calculate the total and charge each guest appropriately. DivvyUp puts the till system in the hands of the guests and gives them the security of their own device with the control of a POS system.

The real genius behind DivvyUP is that it does not affect the service style or policy of the restaurant or bar, it presents itself as just another way to pay.

**“Will that be Cash?, Credit Card?, or DivvyUp”**

DivvyUP is an innovation on existing technologies that solves an old hospitality problem while harnessing the latest enthusiasm for contactless payments, digital wallets, and DIY consumers.

The solution that DivvyUp offers to the problem is sophisticated, clever, and in a commercially viable market, long overdue.

## Target Market

The market research indicates overwhelmingly that all age groups and employment statuses desire a technology that offers a split-payment function that resembles DivvyUP.

The target user group according to the market survey would be all wage or salary earners (Part-time, Full-time , or self-employed) who are between the ages of 25 and 50 years old who go out in groups of four or more at least once a fortnight. This would equate to approximately 84% of the those surveyed (Survey, 2022) and given those groups excluded from the survey (Under 18-year-olds) that would represent in the Invercargill area a viable population of nearly 30,000 people as our target audience. (Invercargill City, 2018).

The target vendor group that DivvyUp would invest its technology would mirror the market Eftpos currently occupies. DivvyUp embraces the digital wallet and as predicted by most technology pundits this will eclipse Eftpos technology in the next 20 years. (Will Miao, 2021) This is to say any business that has an EFTPOS machine could be a potential consumer by using the DivvyUp application for payments.

## Competitors

Most recently PaySquad developed by Cam Richardson is a payment portal that allows multiple users to split the payment of a good or service from retailers. Although this is not specifically hospitality focused it demonstrates the growing interest in this concept. Overseas Splitwise, SettleUp, Tab, Splid, and Cash app more directly resemble what the DivvyUp model offers. However, there has not been one successful release or implementation within New Zealand. This offers an opportunity that should not be passed up.

## Competitive Advantage

Why do we think we can do what no other company has successfully done to date? Simply, people!

The team we have assembled represent over 35 years in the Hospitality industry in every aspect and at every level.

* We have an award winning and nationally recognised hospitality industry leader and contributor.
* We have award winning developers.
* We have access to and approval from companies that are market leaders as strategic partners.

# Business Model

## Value Proposition:

DivvyUp has users and vendors, both offer an opportunity as value positions.

For the vendor or the merchant DivvyUp offers convenience, security, and an added value service for their product. This can be measured in a number of ways, and provides not only cost saving actions, but revenue generating potential.

In the post COVID era, DivvyUp offers a contactless payment platform for guests.

* DivvyUp promotes a digital wallet.
  + This lowers the transaction costs to the merchant.
  + Improves speed of payment
  + Improves security for the merchant
  + Requires less billable hours and hence promotes productivity
  + Requires less capital investment in hardware
* Requires less down time for table turnover.
* Better data analytics
* Provides a direct marketing platform
* Environmentally friendlier
* Less resource intensive
* Does not alter service style

For the user DivvyUp provides

* Control of the transaction
* Speed, efficiency, security, and comfort.
* Choice of payment method
* A portal for real time feedback
* The ability to use your own mobile device.

## Key Activities:

The hospitality industry is a service sector that is measured on first and last impressions. If a guest can successfully navigate the experience and enjoy the essential areas of that experience they are 20% more likely to return. In an industry where Restaurants have a customer retention rate of approximately 30% this inversely means that 70% of people are not willing to return to a restaurant after the first experience. (Voicu, 2021) This experience is made up of three key areas.

* The “Meet and Greet”

The first impression, where you only get one shot at making an impact on the perceptions of the guest.

* The “Product/Service and Atmosphere”

The food and drink quality and the ambience of the site play a part in customers perceptions of the venue and the experience.

* The “Bill”

The last impression, no matter how well the experience has gone so far, if it falls over here it falls over completely.

DivvyUp makes the last impression effortless and effective.

DivvyUp eliminates queues, it satisfies even the most complex instructions on how to split a bill, it allows the customers to control the process. Not to mention the security of using their own device while constantly updating the Host POS, DivvyUP makes sure that the last impression is one of satisfaction and simplicity.

## Key Resources:

DivvyUp is made up of three applications, an independent ‘Pay’ application, an ‘Admin or Admin Panel’ Application, and an ‘Order’ application that will be housed in Azure. Azure offers a free level that can be scaled up to deal with millions of transactions an hour. The system is self-reliant and after an initial set up of payment information, invoicing requirements, and an API to communicate with the host POS, DivvyUp can run independent of any administration outside of the host vendors personal preferences. A 24-hour help desk will be available but once the application is launched it will require little to no maintenance and little to no support long term.

## Channels:

Through strategic partners ILT and posBoss, DivvyUp can be distributed to any venue within the clients of these two organisations. Further marketing and deployment will be achieved through the testimonials of these clients, and the marketing strategy going forward.

## Cost Structure:

Costs are defined as the development hours and the applications hosting costs. After the three hundred hours allocated as development time for this project are accounted for, the only other costs to the project are the deployment to a cloud provider. As stated earlier this is initially at a free level and can be scalable if the demand for the application services increase.

## Revenue Streams

* Transaction fee to user for service.
* Administration, maintenance, service agreement and help desk license.
* Specialised marketing package implementation and maintenance.
* Negotiated percentage of turnover for transactions with digital wallet.

# Background

*“Each seven to 10 years, there is a new generation of technology that challenges suppliers of systems to upgrade or rewrite in order to remain current. Hospitality technology vendors and users must learn to be agile and adapt to change — or suffer a loss of leadership or even relevance. This form of creative destruction has continued to ravage the industry from its technology beginnings and provides openings for new companies with new ideas periodically to shake up this industry.”* (Riley, 2015)



Figure 1 Clare Riley Innovation Quote

As early as 2017 the ‘digital wallet’ and a technology answer to transactions over the internet and in person through businesses and organisations was on the precipice of change. Tech giants like Apple, Google, Worldline, Mastercard, Visa, American Express were all developing technology that would promote a cashless society and move toward BYOD digital solutions. Then in 2020 COVID19, the responses internationally and domestically were unprecedented. The reaction by the public and private sector was at first confusion and frustration, then entrepreneurs seized the opportunity to employ new technology strategies to support and grow new business opportunities. New ways to conduct business, innovations that utilised technology were required to encourage contactless payments, and ecommerce became the new frontier in the commercial landscape.

Hospitality suffered most of all, not only were numbers restricted but the way in which business was conducted reduced the revenue available to businesses. DivvyUp is a service innovation that is an intuitive response to a global issue and a hospitality specific problem.

Capitalising on the consumer acceptance of QR codes, mobile applications, digital wallets, and ecommerce we offer DivvyUp as the next progression in approaches to paying the bill.

**“What is now proved was once only imagined”** (Blake, 1790)

The DivvyUp team partnered with posBoss and ILT to initiate the development of the concept and the eventual market release.

PosBoss is one of New Zealand’s leading point of sale technology companies. It has more than three hundred sites in New Zealand and Australia in retail and hospitality that use their system. In 2016 posBoss abandoned the first attempts at a split pay feature in their POS application. DivvyUp was able to provide vision and motivation for posBoss to re-initiate development of this feature and is privileged to accept posBoss as business and technology mentors for our own product. PosBoss provides market-realistic product requirements, advocate for technology specifications, and they are an excellent source of data and insight as an agent for merchant end users.

The Invercargill Licensing Trust ILT are a community-based ownership and administration of bars, restaurants, liquor outlets and accommodation. They have provided more than $200 Million dollars in donations and funding for community projects. As DivvyUp was conceptualised at SIT another Southland icon, ILT saw and opportunity to hero a local startup and provide support for an idea that is related to their core business. ILT provide a testing venue, marketing support, and an opportunity to release a completed product within their group.

DivvyUp is a southern born concept that has been developed in Southland with the support of local community groups and a New Zealand leading Technology company. The product we have created we believe is a New Zealand first. It offers more flexibility and security than any other system that is similar while advocating for a Digital wallet solution that will overtake the current Eftpos technology in the next 15-20 years.

# Introduction:

This document is an examination of the development of the DivvyUp technology and the processes, technology, methodology, and systems we have applied to create this web application. The DivvyUp project is part of the capstone project for the third year of the Bachelor’s degree at the Southern Institute of Technology and represents the sum total of the experience and learnings that the degree has provided and how each team can demonstrate their knowledge and understanding of the material covered. This document is a record of the project from inception to deployment.

Our project became a company we created called DivvyUp Technologies. DivvyUP Technologies is a New Zealand business that is developing a web application that provides a payment portal and built-in functionality that allows multiple guests to access a digital version of a bill, they will all be able to share the same session in real time, while being able split or divide the payment amongst the group. To provide security and confidence for the venue an API can be added that will update the host POS so that all transactions can be monitored for accuracy.

This document will provide a comprehensive framework of the software development processes we employed and will provide our project scope overview with the critical dependencies and assumptions, constraints, and any issues or scope exclusions that are relevant to the progress and completion of the project. In order to thoroughly cover the relevant development key points, we will explore some of the examples of the existing systems and the system proposed by DivvyUp, this will include providing a full software development report. The SDR will cover the requirements (Both functional and non-functional) and specifications, the specifications analysis, design, Architecture and the software implementation, testing and deployment required and performed by the team for the web application. We will also provide our evaluation of the process and project from the team’s members.

The dissertation about the development of the DivvyUp web application is a comprehensive examination of a unique project from inception to execution. We have documented not only the process and methodology but the evaluation of the project from each team member. We are proud of the product we have created and the opportunity to provide insight and transparency to our journey on this project.

## The Team:

**Project Leader:** Poni Sula

**Database and Backend developer:** Oliver Schweikert

**Frontend developer:** Yuhei Fujisawa

**Mentor:** ken Sutton

**Strategic partners:** Johnny Mackenzie (posBoss), Bharat Guha (ILT).

## Roles and Responsibilities:

**Project Leader:**

A project leader carries the responsibility for the completion of the project as a whole. They are responsible for the communication between the team members and all of the stakeholders in the project.

The team leader responsibilities are:

* Supplement design
* Coordinate team members to accomplish their roles successfully
* Maintain and administrates the progress of the project through the development process
* Create robust policy and plans for project management and execution
* Maintain all documentation and deliver final product.
* Testing and implementation of the testing plan.
* Structure a process for the completion of a delivered final system.

**Development Team:**

The development team are responsible for the research and use of appropriate technologies in software or hardware, they are also responsible for the implementation and deployment of the delivered system.

The development team responsibilities are:

* Research and become familiar with relevant technologies
* Design the architecture of the software system
* Create the system through code and implementation
* Deliver a system that satisfies the requirements and specifications of the project.

**Mentor:**

The project mentor is a BIT staff member who will act as an advisor for the team while providing support for team members in problem solving. The project mentor will conduct themselves as a consultant and will not offer more assistance than necessary.

The mentor responsibilities are:

* Checking project proposal
* Assisting with project meetings
* Progress management and assessment
* Assist with client liaison
* Consult on the final presentation and documentation.
* Assist with assessment of project report.
* Assist with the delivery of the delivered system.

**Strategic Partner:**

Our strategic partners are pseudo mentors and will supply industry level expertise in the execution of the development plan.

Through interviews and surveys the strategic partners have provided the framework for the project and this is manifested as the requirements and specifications for the development team.

The Strategic partner responsibilities are:

* Open communication about standards and best practice required for a commercially viable product.
* Provide a platform in which to Beta test the prototype application.
* Provide expertise in industry technology practices and specifications.
* Provide open communication around deployment and market entry

**Summary:**

The DivvyUp team comprises of a balance between technology and industry professionals supporting a group of BIT graduands in the goal of creating a New Zealand first innovation in payment formats specifically for the hospitality industry.

As a team we have combined enthusiasm and creativity with experts in their fields and created the team that will deliver the DivvyUp Pay system.

## Objectives:

The objective of this project is to create, develop and deploy a web application that provides a solution to the problem of splitting a bill at the table for groups of four or more. The purpose of the application will be to facilitate a system that reduces the complexity of a transaction, is secure, and will save time for the guest and the vendor.

It is the intention of the team to achieve some key goals:

* Research and use QR code technology.
* Research and create an interface (NEXT, TypeScript, CSS) that is engaging and simple to use for the customer.
* Research and create an environment or session that can have multiple users concurrently.
* Research, design and create the architecture for a NoSQL database (Mongo) to handle JSON data.
* Research, design and create a non-critical SQL database with a DBMS that will contain information about vendors, menus, and other data.
* Research and create systems that can manage the requirements and specifications of the system we are describing.

## Benefits of Success:

The DivvyUP project offers some benefits for the user and the vendor as well as DivvyUp technologies as a product that has business feasibility. There are three distinct areas in which the benefits of success are realised.

**Consumer/User:**

* The user will have the convenience of a contactless payment portal.
* The user will not have top queue in order to pay the bill
* The user will be able to control the nature of what transaction they wish to use. (Digital Wallet)
* The user can control the splitting of the bill, they are not left with unclaimed items or splits that are unfair.

**The Vendor:**

* Greater staff productivity
* Less expenditure on consumables items like till role
* No change in style of service or service systems.
* Less cost in transaction fees
* A contactless payment formats.
* Less cost in the lease or rental of Eftpos machines

**DivvyUp technologies**

* A viable product for commercial release
* Project that is complex and rewarding to present as a final project
* The financial opportunities of a successful application that can be monetised
* Industry experience and access.

## Methodology:

**“I am because we are”** (Christopher Tworeck, 2017)

The UBUNTU philosophy as a management strategy is the ideal solution to systems and process management as all decisions are made for the benefit of group as a whole.

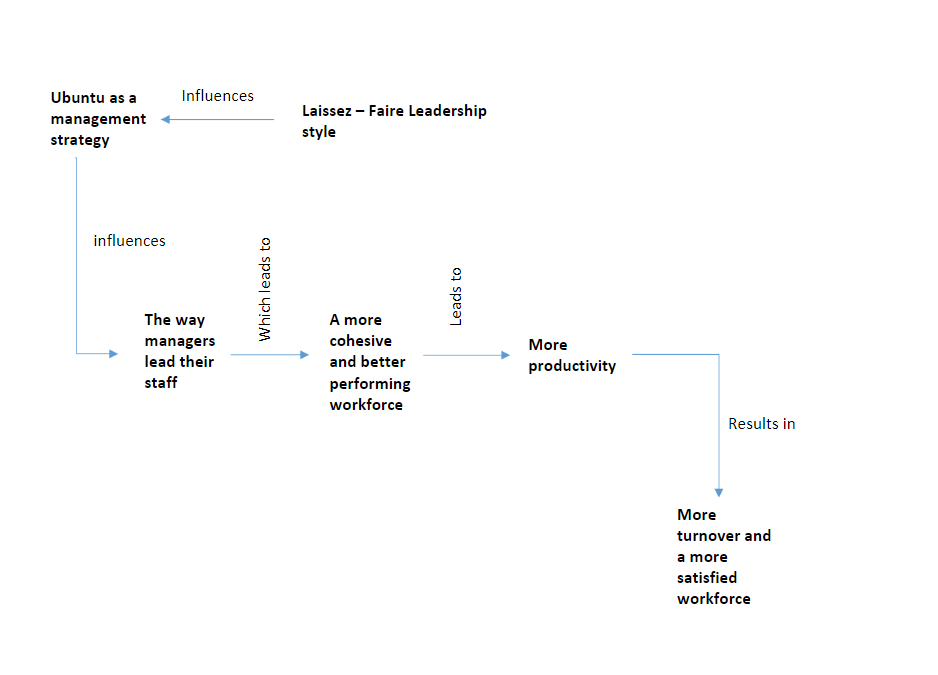


Figure 2 Ubuntu Conceptual Model from (Christopher Tworeck, 2017)

Conceptually the management model allows the work forces to self-regulate and self-manage toward a common goal. Leadership allows the individual to find their own value to add to the collective. The collective in turn acknowledges the contribution and provides support for the individual in their personal and professional requirements. Ubuntu, the philosophy has been a keystone to the delivery of a system that shares ownership and responsibility for the entirety of the project, each member adds value and is given respect and responsibility in return.

Using this as our guiding force the DivvyUp team has also employed an agile methodology that incorporates prototype and feature driven techniques.

**Agile Methodology:**

Adaptive software development is an outgrowth of the RAD that provides continuous adaptation to change. The nature of hospitality and technology means that the landscape is fast moving and probe to dynamic shifts in processes and policies.

The requirements and specifications that were gathered from the strategic partners are matched with the market survey that was concurrently conducted to ensure that the latest trends and consumer needs and wants were being met. This manifested as a dynamically changing development environment and several process resets and return to sender sprint sessions. However difficult this was to management has been mitigated but the successful development a great product.

**Prototype Development:**

The complexity and size of the development was at times overwhelming. Being able to provide a simple solution to the outlined problem proved increasingly difficult as scope creep meant superfluous items were being created, discussed, and addressed before other more pertinent details. Once the team reset its expectations, and incorporated the skateboard approach, we were able to create a solution as a minimal viable product (MVP), present this as our prototype and then productively build on this.

**Feature Driven Development:**

Once the team was able to successfully create a skateboard model, and then develop the initial solution. The methodology was able to move to feature driven processes. The iterative and incremental nature of the consultation with the strategic partners and the analysis of the market survey data has meant each feature has an approval mandate from the collated information.

# Project Overview:

## Scope:

DivvyUp will create a web application that can take a table account (Bill) from a specific restaurant or bar and present it for payment through a QR code. Once the URL has reached the landing page the application will enable multiple users concurrently to process the payment of that bill by splitting the total costs. Each guest within that session will be notified when another guest has completed a payment and a running total will be displayed. A third-party payment portal will be available for the guest to choose how they will pay. The system will be able to provide receipts for the guests and analytic reporting for the vendor.

Divvy Up will be presented as three complete web applications and will provide the following.

* URL endpoints that will be allocated a Unique QR code to represent a single account (Tab, Table account, Bill) in a specific venue.
* A landing page (URL) that will host the data from the bill.
* An interface that is mobile friendly.
* An interface that can be deployed in any mobile browser.
* An interface that allows multiple users to be notified of payments made.
* A three-touch philosophy for user experience.
* The capability to allow multiple users to view and use the same session to pay that bill.
* A function that will be able to split the bill.
* A third-party access portal to enable payment.
* An application that will be able to replicate an ordering system. NB This is for presentation purposes.
* An application that will be able to view/Monitor transactions conducted within the payment application. (Admin Panel).NB This is for presentation purposes.
* The Pay application will be able to forward a receipt for each guest for their specific payment via email.
* A simple customer feedback input.
* A working version that contains basic functionality as described above.

## Scope Exclusions:

* Licensing and hardware costs
* API development and costs, for specific venue POS
* Liabilities for third-party service partners
* Hosting, font, or services fees.
* Support of or for operating systems, browsers not directly outlined.
* Design, creation, maintenance or implementation of venue specific colour or design architecture.
* POS order capability outside of payment for any venue.
* The production of the specific QR codes for each POS in a venue, business, or organisation where there exists no API for the hosts POS.
* Accounting services for venues, businesses, or organisations that use the application.

## Critical Dependencies & Assumptions:

* All software and hardware required to complete the development and application build will be available for research, use, development, and deployment.
* Project and change approval are granted by consultation with the course coordinator and the principal stakeholders with DivvyUp technologies.
* The number of users/guests will be capped so that while in a developmental phase, costs of cloud hosting do not exceed the free tier.
* Because of the commercial viability of the d-Up Pay application all the intellectual property of the application remains the property of DivvyUp technologies and its shareholders.
* The stakeholders will be available for consultation when necessary.
* The strategic partnerships with ILT and posBoss do not imply any financial responsibility in terms of profitability, risk, or cost. Each party will work in the best interest themselves as not to cause undo cost, loss, or risk to their primary, secondary, or associated mode of business or operation.
* The budget set will be sufficient to complete the project to the specifications of the Capstone project to the highest level possible.
* Beta Testing and market feasibility surveys will be conducted at an ILT site(s) with the approval of the GM of that site and the ILT liaison for the project Bharat Guha.
* No existing contract that prohibits the creation, production, or release of the DivvyUp pay application.
* SIT, ILT, posBoss and all other stakeholder’s policy and regulation do not inhibit the creation, development, and release of the DivvyUP application.

## Constraints:

* Local and international standards accepted by statutory regulation or accepted as best practice to the relevant industry. Where possible DivvyUp will adhere to standards, protocols, and practices established by IEEE, ISO and in the domestic environment MBIE, IRD, and MIA.
* The estimated period for completion of the capstone project is 36 weeks.
* The DivvyUp application as a Capstone project will adhere to the same budget restraints as all projects proposed in the 2022 period.
* The provisions of the BIT course offer rudimentary access and instruction to software, hardware, technology not taught within the course.
* Security, protocols, and development will be within the bounds of the SIT ITS system.
* Any health and safety risks to a COVID19 response or any other such national, regional, or local disaster, catastrophe, or emergency .
* The contractual obligations of the strategic partners to their stakeholders and partners.
* Commercial viability for privacy, confidentiality, and competitive advantage.

## Current System/ Competition

The idea of splitting a bill at a bar or restaurant is not unusual, there may be some cultural anomalies depending on what country you go to, but simply it is the ability to sperate the costs of a single bill so that each party member can pay their portion of the bill.

When a google search for split-payment-at-table-nz is entered the first ten results are articles about the art or the etiquette behind why splitting a bill in restaurants is good or bad practice. Further investigation leads to number of POS point of sale applications that offer services for vendors with functions that perform splitting of bills.

Anecdotally, in restaurants, this is achieved by the server doing the work, calculating the portions, or selecting the items, then offering the total to each guest for payment. This is both time consuming and frustrating for the guest and the server alike.

Some bars and restaurants have mitigated this by employing policies around one bill per table or attaining mobile applications or hardware that can be downloaded or provided to the guest to split a bill.

The current model for splitting bills in restaurants and bars requires a server and customer to sacrifice time and energy to pay a bill. This comes at a cost both potential and realised in transaction fees and consumables. The alternative is a downloadable application, that partially provides a contactless payment but with the requirement of possession of the application or a link being sent to each user. There is a tangible lack of uptake in New Zealand for tis solution as the workload is not shared but transferred.

The following table shows some of the available applications in the global market today.

## Comparison Bill Splitting Applications

|  |  |  |  |
| --- | --- | --- | --- |
| Application | Fee | How it works | Best For |
| BIllr | App costs (USD)$1.99 | List costs, split shared items, set tax tips amounts, show up to sixteen people what they owe and send copies of the split by text or email | * Groups * Restaurants or bars |
| Settle Up | App costs (USD)$1.99 | Manage shared expenses with friends, track receipts and past bills and show who pays next | * Friends * Keeping track of recurring costs |
| Google Wallet | Free\* (May be a transaction fee per transaction for vendor and customer) | Connect your debit card and tap up to five friends to split bills with. Add the total cost of the bill or individual amounts then send and receive money directly from the app | * Sending money Quickly |
| Splitwise | Free\* (May be a transaction fee per transaction for vendor and customer) | Create groups to track ongoing or one-off expenses. Download Plates by Splitwise for a tool that is tailored for use at restaurants and bars. | * Roommates or groups * Track Ongoing expenses * Tabs on multiple groups |
| Easyshare | Free\* (May be a transaction fee per transaction for vendor and customer) | Pay bills, rent, split expenses between housemates | * Roommates |
| Splitr | Free\* (May be a transaction fee per transaction for vendor and customer) | Split bills between groups, request money and track when you have received payment | * One-off bills * Group payments * Multiple expense splitting |
| Tab | Free\* (May be a transaction fee per transaction for vendor and customer) | Take a picture of your receipt and tap your items to claim them. Tax and tip are calculated for you, allowing you to easily split bills between friends | * Friends and groups * Restaurants or other shared expenses. |
| Splid | Free for one group, (USD)$0.99 for two, (USD)$1.99 for unlimited | Enter all your shared expenses and the app shows exactly how much everyone owes and to whom. | * Friend and small groups * Shared and ongoing expenses * Trips |
| Mr. Yum | Free\* (May be a transaction fee per transaction for vendor and customer) | Order and pay for menu items. Batch feature means a total table order can be sent as one order to the bar or kitchen. | * Individuals * Split bills at restaurants or bars. |

Table 1 Comparison bill splitting Apps (Barry, 2021) (Barry, 2021)

## Existing System Ishikawa Diagram

Diagram

Description automatically generated

Figure 3Current Hospitality Ishikawa

## Proposed System

The DivvyUp pay system has been described in terms of benefits and actions that will make the application valuable to the industry and the consumer so far. In effect the system that DivvyUp technologies tenders, bridges the gaps between the existing environment and processes, with solutions that are user friendly and vendor acceptable.

Later sections of this report will outline the particulars of the system with its requirements and specifications. For now, the newly proposed system can be characterised as a new way to pay that endorses a digital wallet and provides a splitting function that allows guest to use their own device, share particular tables bill concurrently, and process a payment for that bill. Our solution is elegant and satisfies all of the vendor and user requirements outlined in the following sections. After analysis, consideration, and consultation DivvyUp offers a unique solution for split-payments-at-table.

# Requirements:

The requirements have been gathered from three key sources; Strategic partner interviews, Market survey results, and the product design team brief. This has been a unique approach to the inception of an idea to the realisation of product development. The following sections break down further the requirements we have considered.

### Description

The DivvyUp system, will provide an application which is accessed via a URL, this will deliver a bill that can be accessed by multiple users to facilitate a split payment of that account.

The basic outline of the functional requirements is broken down in the following sections

### User requirements

* Access a bill via a URL
* Pay a portion of the Bill Total
* View when other Users have paid and see the Bill total remaining
* Use their own device (Digital wallet) for this payment

### Developer Requirements

* UI created through NEXT
* UX is guided by a three-touch philosophy
* Create a URL that directs to the application with the information from a specific bill
* Provide access to a third-party payment platform with the individual user’s total
* Create a NoSQL database that can handle bill data in JSON form
* Create a database that can distribute receipts
* Create an admin panel that can view tables transactions
* Create an application that will hard load three different table accounts as a pseudo-POS system for demonstration purposes.

### Server Requirements

* Allow the users to see total owing, amounts already paid, and personal total to pay
* Allow processing of JSON data
* Produce receipts for payments
* Let the admin portal view transactions
* Deliver JSON data for UI

### System Requirements

* Temporary storage for reporting data
* Response time for URL load
* Response time for in application functions
* Digital wallet format for safe and secure payment
* Accurate billing and receipt data
* Timely accounting updates for admin panel and multiusers.
* Reporting

## Functional Objectives

### High Priority (Mandatory)

1. The application shall provide a URL that delivers the user to the application functions.
2. The application shall attain and make available for viewing and payment the relevant data for a specific Table account.
3. The application shall split the bill per user.
4. The system shall allow the user to view the current total owing after other users have processed payments.
5. The application shall allow the user to pay their portion of the split bill via a payment portal.
6. The application shall upon confirmation of payment will produce a receipt of payment and offer to display it or deliver to the user via an email address.
7. The system shall provide an opportunity to leave a review of the vendor services and leave a comment if desired.

### Medium Priority (Elective)

1. The application shall allow multiple users to access the table account.
2. The system shall provide a dynamic QR code that will link to the URL created by the system.
3. The system shall store for receipts and reporting the relevant legal financial information (GST Numbers, Business details).
4. The system shall create a pseudo-order system to replicate a Vendor POS.
5. The system shall provide an admin portal that can monitor and view transaction within any specific table account being processed by the application.
6. The system shall provide all of the transaction information required for the vendor to confidently offer a refund for over payment, or disputed payment.

### Low Priority (Potential feature additions)

1. This application shall provide translation services for different users.
2. The system shall provide support services for users as assistive technology.
3. This application shall provide multiple methods of bill splitting

* Nominated dollar amount
* Per item selected
* Split evenly
* By beverages and food
* By percentage of the bill
* By any combination of the above

1. The application shall provide metrics that will provide the vendor with buying habits and payment preferences.
2. The developers will design an API that fetches data, and broadcasts table account information to the host POS.
3. The application shall store transaction details for accounting purposes for the vendor.
4. The system shall provide an optional tip function.
5. A Guided walkthrough (Or helpful hints)for first time users should be available as an opt in or opt out function.

## Non-functional Objectives

### Reliability

The system should be operational 100% of the time after deployment.

Downtime after a system failure should be less than 10 mins.

### Usability

A user should be able to pay their portion of a split evenly bill in less than 3 minutes.

The number of pages required to navigate from viewing the bill up to bill payment should not exceed three.

### Performance

The system should be able to support and maintain fifty users simultaneously for a table account.

The system should be able to load a URL in 5 seconds or less on any browser.

### Security

The system should not record or store any guest credit card or bank details.

The system should not allow access by the user to any business information not legally required.

The system should not allow access by the business to any individual users’ information not explicitly or legally required for the operation of their business.

The system should adhere to all legal financial standards and practices regarding third-party payments.

The Application should not be able to change prices, apply discounts, or insert, delete, or edit items from a bill or from the host venue.

### Supportability

The system should be able to create new venues for use of the application without having to re-write or re-factor the entire application.

The application should be compatible for iOS version 10.0 or above, and for Android 5.0 testing environment or above.

The application should be able to be viewed on explorer, Netscape, google, safari and Mozilla Firefox browsers.

### Online User Help

The system should provide guided assistance for first time users.

The service should provide a 24/7 help bot for simple enquiries about the processes of the system

### Cost Components

The system should advocate for the digital wallet solution that is secure, safe, and provides options for lower or negligible transaction fees for the user and the vendor.

The system should provide the benefits of the EFTPOS model without the overhead costs

### Interfaces

All mobile devices that are internet capable should be able to host the URL endpoints generated by the application.

### Need

The costs generated by the EFTPOS model can be reduced or eliminated with the digital wallet model. DivvyUp provides access and exposure to the benefits of a Digital wallet. The DivvyUp application compliments any restaurant or bars service model.



By 2025 digital wallet use will account for just over half (52.5%) of ecommerce transaction value worldwide, versus 48.6% in 2021, according to the FIS Global Payments Report. At the physical point of sale, the report states, digital wallet use will rise to nearly 39%, an increase of about ten percentage points from 2021 levels. (Cocheo, 2022)

Figure 4 The financial Brand Digital Wallet Quote

# Analysis & Design:

## Objectives and Success Criteria

**Objectives**

***Decision Support***

Provide up to date, accurate and relevant information about bills to the customer. While providing a secure platform for payment and payment reconciliation. Vendors are provided with accurate and up to date payment confirmation. The user is provided correct billing information and a system that can split the bill accurately and with flexibility.

***Customer experience***

Generate increased customer engagement as users replace EFTPOS as their preferred form of payment.

***Productivity***

Reduce the time spent checking and requesting deviations in payment amounts , whether that be through less staff engagement or following the linear pragmatic model employed at present.

Increased table turnover through less accumulated time to deal with multipayment.

Less stand down time while queues are avoided or eliminated.

Customers and staff spend less time waiting for the final actions of the service cycle as payments are concurrent rather than sequential.

***Capabilities***

Businesses mitigate lost potential revenue in the time staff are not productive.

Guests are provided with less pointless time in the processing of bills and payments.

***Knowledge***

Improving the third fundamental of positive guest experience by offering an alternative which is comparatively faster, safer, and more directed by guest preference.

***Data***

Improve and increase data quality for vendor marketing, and service efficiency.

***Integration***

Creating a more streamlined server and customer workflow that does not influence service style or service procedure.

***Skills***

Providing customers with more capability to enforce their own preferences.

Providing the vendor with a modern interface that reduces wait time, human error on behalf of the vendor, and improve customer relations.

***Performance***

Improve customer experience, staff productivity, and business efficiencies.

**Success Criteria**

***Cost***

The costs of the project do not exceed the budget for development.

Timeline

The project is completed prior to the deadline provided by the parameters of the Capstone project.

***Scope***

All aspects of the scope statement are completed.

***Deliverables***

All deliverables are completed on time and presented for approval to the course coordinator and the strategic partners.

***Resource Capacity***

The technologies used are utilised and deployed without external assistance.

***Stakeholder satisfaction***

The stakeholders provide an 80% or more from the satisfaction survey.

## System Models

***Scenarios***

The user journey dictates the functionality of the user interface and the user experience. This leads any analysis of the proposed system to consider “How?” and “Why?” the system will be used. In this respect we have defined the user in terms of any actor that would engage with the bill paying process, this may be the customer or the server.

The following diagram is a user journey diagram that outlines the bill paying process. The diagram demonstrates the binary decisions that each user will experience and their ease of choice in the preceding and following actions. The process has been simplified for ease of understanding and ignores non-critical processes such as multiple users’ experiences and other user payments notification or identification.

## User Flow Model

Diagram

Description automatically generated

Figure 5 User Flow Model Bill Payment

## Use Case Model

Diagram

Description automatically generated

Figure 6 Use Case DivvyUp Customer

### User Story One

**Title:** Table of Four

**Description:**

The table of four has concluded their evening in a venue and it is agreed it is time to pay the bill. A member of the table has signalled for the bill from the server and requested to split the bill. They have asked how this is done and what the policy of the venue is. The server advises that they have DivvyUp and will provide the bill.

**Acceptance criteria:**

* The bill is delivered with a unique QR code to the guests.
* Each guest can scan the code and view the table bill on their own device.
* Each guest accepts the payment split by total.
* Each guest is able to process payment for that total using their digital wallet or preferred method of payment.
* A confirmation is received and on request for a copy, a copy is sent to the supplied email address.

### User Story Two

**Title:** Table of six

**Description:**

The table of six has concluded their evening in a venue and it is agreed it is time to pay the bill. A member of the table has signalled for the bill from the server and requested to split the bill. They have asked how this is done and what the policy of the venue is. The server advises that they have DivvyUp and will provide the bill.

The guests would like to pay for what they had.

**Acceptance criteria:**

* The bill is delivered with a unique QR code to the guests.
* Each guest can scan the code and view the table bill on their own device.
* Each guest accepts the payment split by total.
* Each guest is able to process payment for that total using their digital wallet or preferred method of payment.
* A confirmation is received and on request for a copy, a copy is sent to the supplied email address.
* Each guest is able to process payment for that total using their digital wallet or preferred method of payment.
* As each payment is confirmed the remaining users can see the payment total has been deducted from the remaining total.
* As each item is confirmed as paid for the that item is blocked from being re
* A confirmation is received and on request for a copy, a copy is sent to the supplied email address.

### User Story Three

**Title:** Table of Four

**Description:**

The table of four has concluded their evening in a venue and it is agreed it is time to pay the bill. A member of the table has signalled for the bill from the server and requested to split the bill. They have asked how this is done and what the policy of the venue is. The server advises that they have DivvyUp and will provide the bill.

One guest wants to pay by cash.

**Acceptance criteria:**

* The bill is delivered with a unique QR code to the guests.
* Each guest can scan the code and view the table bill on their own device.
* Each guest accepts the payment split by total.
* Each guest is able to process payment for that total using their digital wallet or preferred method of payment.
* A confirmation is received and on request for a copy, a copy is sent to the supplied email address.
* Each guest is able to process payment for that total using their digital wallet or preferred method of payment.
* As each payment is confirmed the remaining users can see the payment total has been deducted from the remaining total.
* As each item is confirmed as paid for the that item is blocked from being re
* A confirmation is received and on request for a copy, a copy is sent to the supplied email address.
* The server can use the admin panel to monitor payment totals and items that have been paid for.
* The remaining and items and the total owing is viewable from the admin panel and cash can be taken as payment by the server for the closing amount.
* Server can close the table by processing the cash paid amount.

## Design Specifications

## The DivvyUp Modules

NEXT.js Frontend

* Bill display
* Split capability processes
* Payment Portal
* Multiuser notification
* Receipt request

DivvyUp Database

* Receive JSON bill data API from POS
* Organised to allocate and record identity and authorisation symbols for venue and bill

DivvyUp System

* Calls APIs with relevant keys
* Collects and displays bill information
* Processes to split the bill
* Processes to make payment for amount desired
* Process for providing receipt for payment
* Process to update Host POS.
* Processes pertinent to security
* Processes pertinent to reliability
* Processes pertinent to maintainability
* Processes pertinent to functionality

## User Types

The two user groups should be classified and labelled as to draw the distinction between their requirements and provide insight into the user interface (UI) and the user experience (UX).

* Guest/Customer Users are the individuals or groups who have engaged in commerce with the venue or vendor. They are receiving the good or service end of the transaction and will pay for their use of the product.
* Server/Admin User is monitoring the transactions made in the purchase of goods and services and need clear information on the success or failure of the transactions.

In this instance the two users have distinct assignments. The Guest user is transactional, and the admin user is read only.

## Design Constraints

Performance, storage, security, and access are all elements that require consideration in the approach to designing the DivvyUP system. There are some that can cause limitations or mandatory framework to guide the process.

### Financial

The financial burden of hosting the application and the potential scalability if the application is utilised by an increasing number of users.

### Technical

Specific and general skills in understanding mobility, “demand response” management, optimization and server management are required by the developers as this will direct their coding and structuring solutions.

### External Organisations

The POS companies that currently operate in New Zealand and Australia will need to approve the API that will be used by DivvyUp. As they currently can integrate Eftpos there seems to be a precedent for the integration of DivvyUp as a payment portal. However, there is more flexibility and user influence on the data required and processed for the POS so this must be a consideration for the developers in the design implementation.

## System Architecture Issues

This section will provide some clarity on the influences of design to architecture. This simplifies a discussion about the architectural design plan for the DivvyUp application.

### Dependencies

The third-party payment portal be it a direct widget, or as part of a more organised digital wallet means DivvyUp is reliant on this technology. By endorsing and supporting Digital wallets DivvyUp looks to secure a dependable partner in all of the interested parties in this sector.

* Google Pay
* Apple Pay
* Visa
* Mastercard
* American Express
* Diners
* Banking Institutions such as Westpac, BNZ, ANZ

POS Point of sale companies for retail and hospitality are an integral part of the DivvyUP system, Although DivvyUp can provide a mobile ordering system for those venues that do not have their own, 90% of the current market in New Zealand is occupied by any one of a dozen POS solutions.

* Eposnow
* Lightspeed
* posBoss
* Revel
* Onetap
* Swiftpos

These solutions can be integrated with the DivvyUp system but require approval for API development that would allow the exchange of the relevant data. We tacit verbal agreements that would allow this exchange once the DivvyUp system has passed all testing.

### Risks

The risks to the DivvyUp system are contained in the dependencies that were outlined above.

The timing of deployment and the formalisation of agreements between DivvyUp and POS vendors has stirred the interest of these vendors into developing their own version of this solution. As they are dedicated to their own brands we accept that our technology may be made redundant if vendors do not allow or provide the required integration.

Also, as this technology is still being developed we are mindful that security remains paramount. As prepared as we believe we are to deal with this occurrence, we have layered into each module a security level that protects our product, the consumers , and the third-party vendors. This is also to say we have worked with partners that have the same philosophy or practices.

These are theoretical risks however, there are more practical risks that we endeavour to mitigate. The potential for spamming or nefarious parties trying to interrupt system processes is an acknowledge threat. In all instances we consider these against the current system and believe standard operating processes and procedures will alleviate or at least reduce these risks.

## Architectural design Goals

It is important to have clear and attainable goals for the development team. This provides project structure and allows the team to identify clear milestones and in turn provides the resources for the deliverables. The following team design goals are outlined below.

## Operational Environment

This is the suggested release environment, each element has been chosen with regard to its flexibility, scalability, and reliability.

* Git Version control
* GitHub Repository(Private)
* Mongo DB database
* MikroORM for NextJS
* NGINX server (As Applied by Azure)
* Apache HTTP Web server (As Applied by Azure)

## Developmental Environment

The application will remain consistent. This will alleviate any adverse effect of interoperability or quality.

**Usability:**

The feature of DivvyUp must demonstrate ease of use and provide powerful user experience. The features cannot divert from the user expectation unless it is an improvement in the existing models. This translates to mean DivvyUp will not affect the service style of any business and any long-standing conventions that are unique to the hospitality model. People will inherently understand what role DivvyUP will play as they do understand what role Eftpos plays. What DivvyUP offers is more flexibility and control by the user which in this instance is a benefit to the user.

**Extensibility:**

DivvyUP features must be extensible. Any features can be added if the ned user requires them.

**API Enabled:**

The application must be API centric and support an open and published API architecture.

**Restful Framework:**

The underlying architecture must be REST Framework. The DivvyUP application must conform to the constraints of REST architectural style and allows for the interaction with RESTful web services.

|  |  |
| --- | --- |
| Software | Description |
| GitHub | Version Control Repository |
| Azure | Cloud Computing Platform |
| MongoDB Atlas | Document Database |
| Chakra | UI Framework/Theme |
| NextJS | Programming Framework for web UI |
| JavaScript | Programming Language |

Table 2 Development Environment Framework

## Development Methods

The application has been layered into multiple autonomous applications that can be replaced individually and allow the team to keep the application running while another is being maintained or serviced.

**Scalability:** Once the traffic increases Azure will add or remove servers as the solution requires.

**Availability:** Infrastructure redundancy is required to support a hi availability environment.

**Security:** All Back-end code should be securely stored and not exposed for bad faith actors.

**Extensibility:** Each module has its own responsibilities as not to be reliant on a past or future model.

**Restful Framework:** Simple flexibility, the best suited language for the technology stack we are using is JavaScript.

## Cloud strategies

The use of cloud technology has formerly been the most disruptive forces impacting applications and infrastructure. The cloud forced business model and new architecture decisions which have impacted the way DivvyUp can deploy, manage , maintain, and protect their data.

Azure offers multiple options for provisioning IT infrastructure and the deployment of web-based applications.

**Infrastructure on Demand:**

The building blocks that represent the infrastructure are already provisioned as required, they can follow actual demand and allows pay-as-you-go. This means greater flexibility for the development team in all development environments as well as disaster recovery scenarios. The development team can code in solutions that can be automated which will improve self-service and maintain the delivery of desired business and technical outcomes for consistency.

**Cloud Computing:**

Cloud computing today has brought on-demand network access to a shared pool of configurable computer resources. This can be rapidly provisioned and released with minimal additional effort from management.

Azure offers global infrastructure on a pay as you go model. The first tier of which is free and as the demand on resources increases so does the cost. This has provided massive flexibility in meeting the requirements for disaster recovery and data protection.

**Recovery:**

A physical environment poses added risks to secure and safe disaster recovery and general security. Cloud strategies allow for multiple solution to disaster and security scenarios to be generated within the cloud environment. Testing and Development can be conducted on the cloud, this can provide levels of security not afforded to a physical environment.

**Mobility:**

The way in which we conduct business and in general our lives has been heavily influenced by mobile technology and mobility in general. With cloud considerations combined with Mobile technology more accurate information is available more immediately than ever before. The office has become and where you can carry your phone and get a message. This must be a consideration in the planning of any software engineering project.

**Scalability:**

Going viral is a term used to indicate the rapid engagement of an idea. As will applications that can have instant success or a more graduated evolution the data management solution must adapt to the requirements it is tested by. The system must generate business value that is in accordance with quick and efficient changes that may be required depending on the applications performance.

**Archive:**

The cloud provides a secure location to support either archiving valuable information in a physical location or as a dynamic multiple level solution.

**PaaS (Platform as a Service):**

Azure has built in solutions, this includes software development tools, application servers, analytics, network connectivity, database management and much more.

## System Descriptions

The development, and operational environments and strategies around cloud usage and application lend them selves to the description of how the system will be employed. This will influence what specifications these environments will require. The following table describes some of the systems in use.

|  |  |  |
| --- | --- | --- |
|  | System | Note |
| Programming Language | JavaScript |  |
| Development and Version Control | Git | Git is a free and open-source distributed version control system designed to handle everything from small to large projects with speed and efficiency |
| Hosting service | GitHub | GitHub is a code hosting platform for version control and collaboration. It allows you tom work with others from anywhere in the world. Projects can be hosted directly from your repository |
| Database | MongoDB | MongoDB is built to scale out architecture that has become popular with developers of all kinds. It is document based and allows developers to store unstructured data |
| Web Server | NGINX | NGINX is a web server than can also be used for multiple solutions. It is free and Open sourced. |
| Web Server Software | Apache | Apache is free and open-sourced software it is one of the most popular web server’s software’s on the planet |
| Cloud Application Platform | Azure | Azure is Microsoft’s platform as a service that enables developers to build, run, and operate applications entirely in the cloud. |

Table 3 System Software Outline

## Server Requirements

Server capacity and power along with flexibility in a shifting commercial environment are focuses that should be addressed prior to the commencement of development. Having an idea of the requirements can indicate what servers are best suited and most capable to handle the characteristics of the application. These preparations make it possible to provide scalability later.

As usage and engagement, in the final phase testing and in the first stages of release, may be low, only the free tier of Azure will be required. In lay terms a moderate or practical server that can deal with less than a dozen transactions will be needed. With Azure the free tiers will be used until traffic to the application can justify an increase in a dedicated CPU, memory, and storage. Horizontal scalability can be achieved by stepping into the next paid tier and scaled up using the enterprise or professional tiers.

AWS also offers similar levels of introduction and scaling. The multi-tiered user-pays practices of AWS and Azure are perfect for this application. A decision about which company to utilised will be dependent on cost considerations and the level of engagement received after release.

A simple guide to the potential server requirements is demonstrated in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Device | Web Server | Storage | OTP Server | Database Server |
| TYPE | Azure | AWS S3 | Azure | Azure |
| MEMORY | 4GB | 5GB | 4GB | 4GB |
| STORAGE | 68GB | 100GB | 68GB | 68GB |
| OS | Ubuntu(LTS) v20.10 | Ubuntu(LTS) v20.10 | Ubuntu(LTS) v20.10 | Ubuntu(LTS) v20.10 |
| SOFTWARE | Apache |  | DivvyUp | MongoDB |

Table 4 Server Requirements

## Program structure Design

We developed an architectural plan to provide some understanding of system and security framework, performance, and software architecture.

The following graphical diagram is a high-level view of the basic architecture of the DivvyUp Application.

As demonstrated in the diagram the system separates the functions of the front end and the functions of the server. This is used as guidance and simplest overview for the reader of this document.

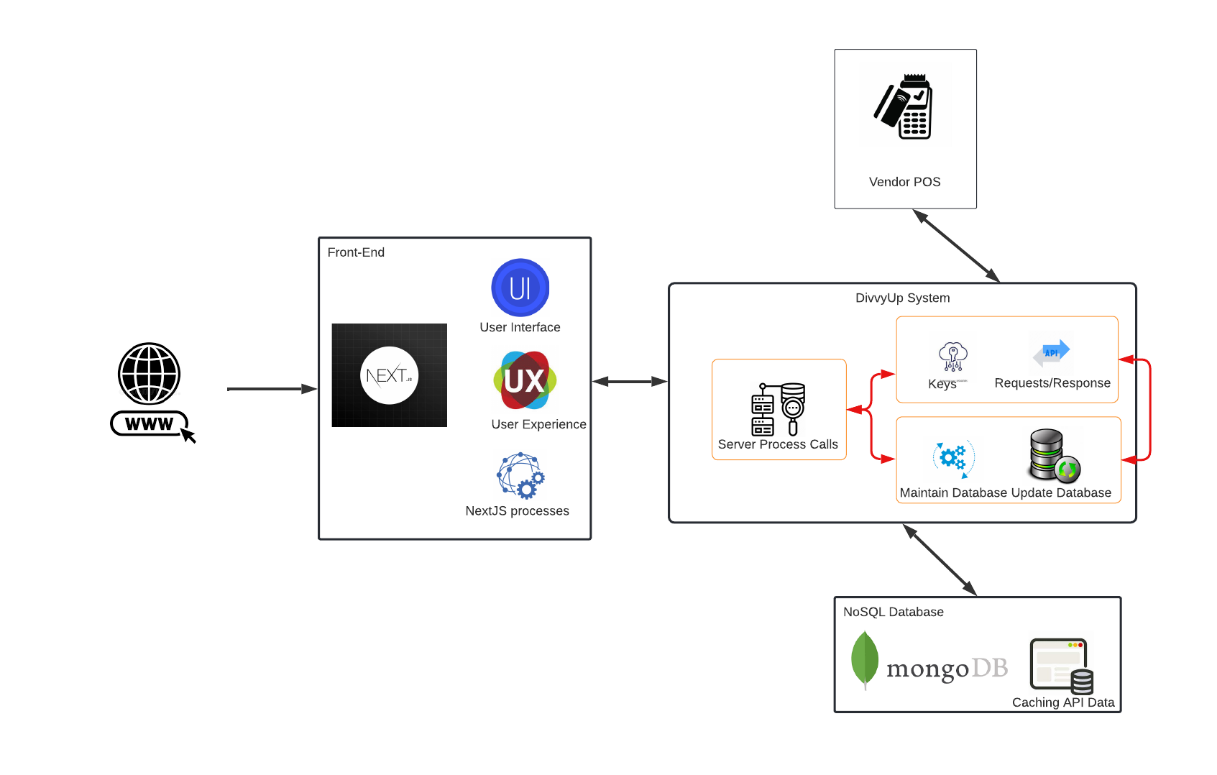


Figure 7 DivvyUp Architecture High-Level View

### System & Security

Features that are built into GitHub , Azure, and AWS add security to the DivvyUp application and does not preclude any further authorisation, or authentication features that can be added to the DivvyUp system. This provides multiple levels of reliability, security, and maintainability.

An overview of Azure platform is demonstrated in the graphical diagram below.

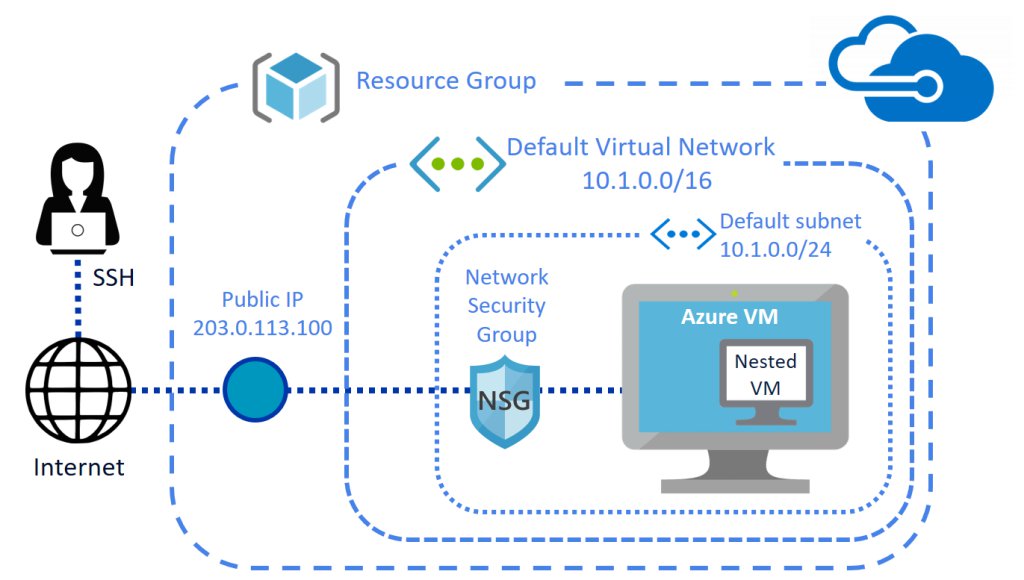
(Linkletter, 2018)

Figure 8 Create a nested virtual machine in a Microsoft Azure Linux VM (Linkletter, 2018)

### Performance

Azure, and AWS s# are utilised for their hardware performance and reliability. S3 is storage for the internet and Azure is services storage for functions from the DivvyUP web application. Both are scalable, dependable, and low latency with free tiers that will mitigate budget constraints.

S3 is highly flexible, it can store any amount of data and is being utilised for us as a disaster recovery option.

Azure can be referred to as a polyglot platform that operates as a PaaS.

We believe for the release of DivvyUp we are assured of a world class synergistic partnership with these two platforms.

### Software

As stated earlier in this document regarding separation of responsibility and the establishment of modules. The Software architecture can be described in terms of Tiers. DivvyUp is a user facing application, it will be constructed using three tiers.

The tiers will comprise of the presentation tier (Module 1) the data tier (Module 2) and the logic Tier (Module 3).

The presentation tier represents components that users directly interact with.

The Logic tier contains the code required to translates processes and actions at the presentation tier.

The data tier consists of storage media and holds the relevant data to the application. (Databases etc.)

### Security Software

There are a number of principles applied to the proposed system security.

* Apply security to all layers:
  + Rather than running firewalls only at the edge of the infrastructure, firewalls and other security controls are on all resources.
* Enable traceability
  + All actions and changes to the environment are logged and audited.
* Implement a principle of least privilege.
  + Ensures that authorisation is appropriate for each and every interaction with the AWS and Heroku resources.

### Performance Architecture

Providing performance guidelines have allowed the developers to build testing, provide framework for integration, and change management. Scalability is planning for future performance, while performance of the existing system needs to be controlled. Assessed and adjusted throughout the development process.

|  |  |
| --- | --- |
| Desired Quality | The system should predictably execute within its performance profile and manage increased processing volumes if required. |
| Concerns | Scalability  Predictability  Response time  Throughput  Peak Load behaviour |
| Activities | Conduct Practical testing  Assess against requirements  Create performance models  Rework Architecture if required |
| Strategies | Prioritise processing  Distribute processing  Use Asynchronous processing  Partition  Minimise shared resources  Optimize processing  Make compromises |
| Pitfalls | Unrealistic models  Inappropriate portioning  Concurrency  Transaction overhead  Imprecise Performance |

Table 5 Architecture Performance Criteria

## System Behaviour

As part of the analysis of the DivvyUp development the theoretical system behaviour is vital to understanding the overall system. The following diagrams provide some context and insight into the data management, information flow of the system while guiding the processes and development methods of the development team.

**Sequence Diagram:**

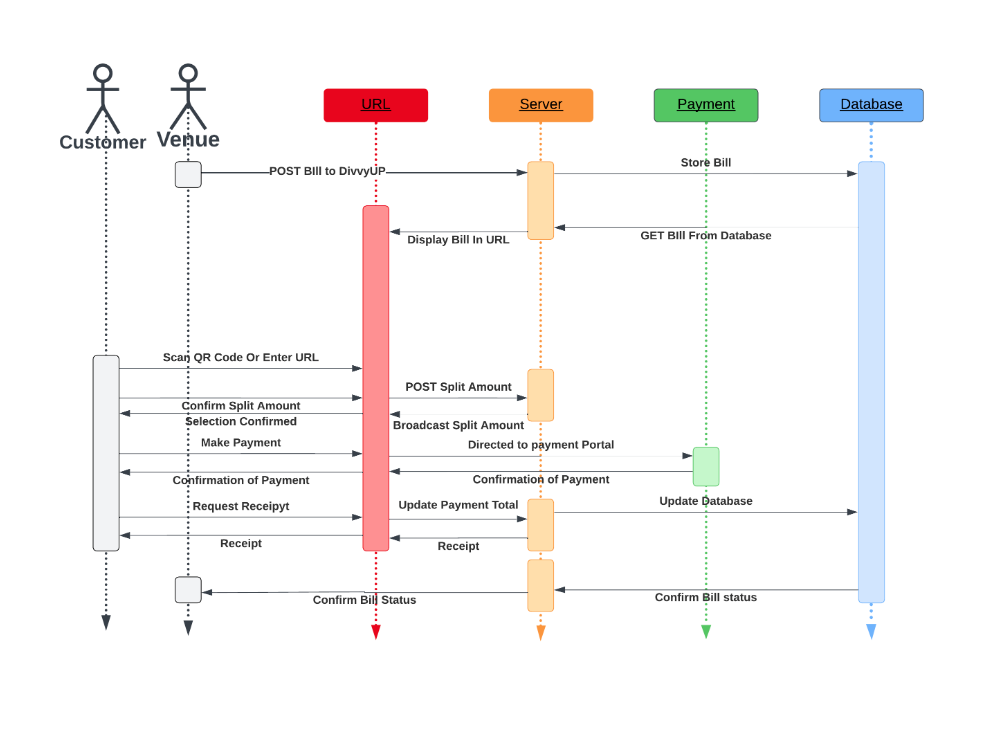
****

Figure 9 Sequence Diagram DivvyUp Customer Experience

**State Diagram:**

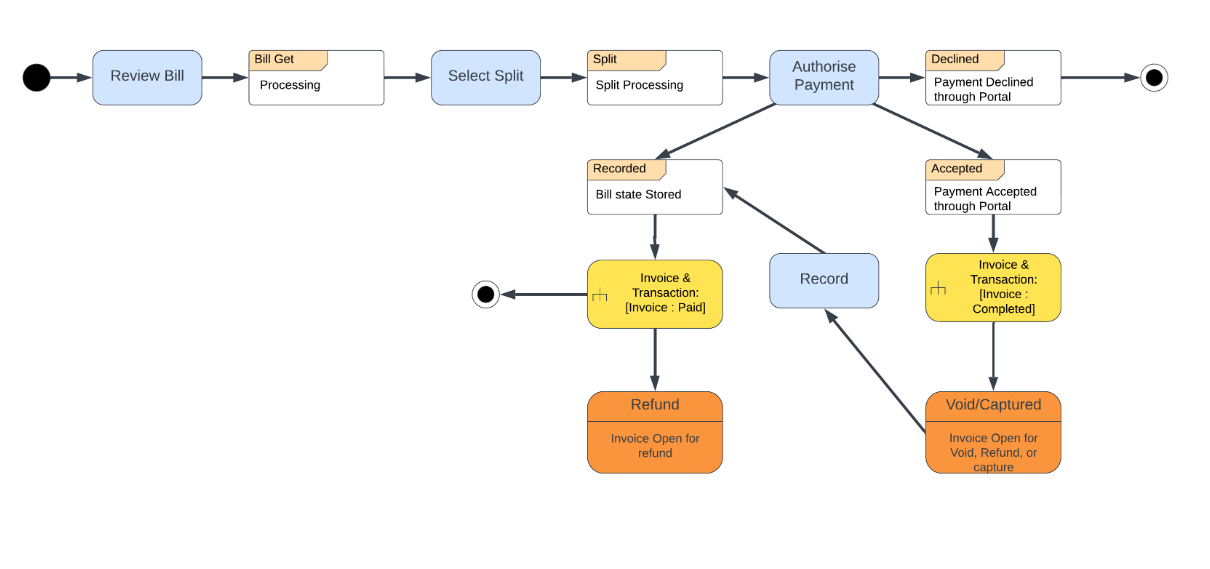
****

Figure 10 State Diagram DivvyUp Pay

**Activity Diagram:**

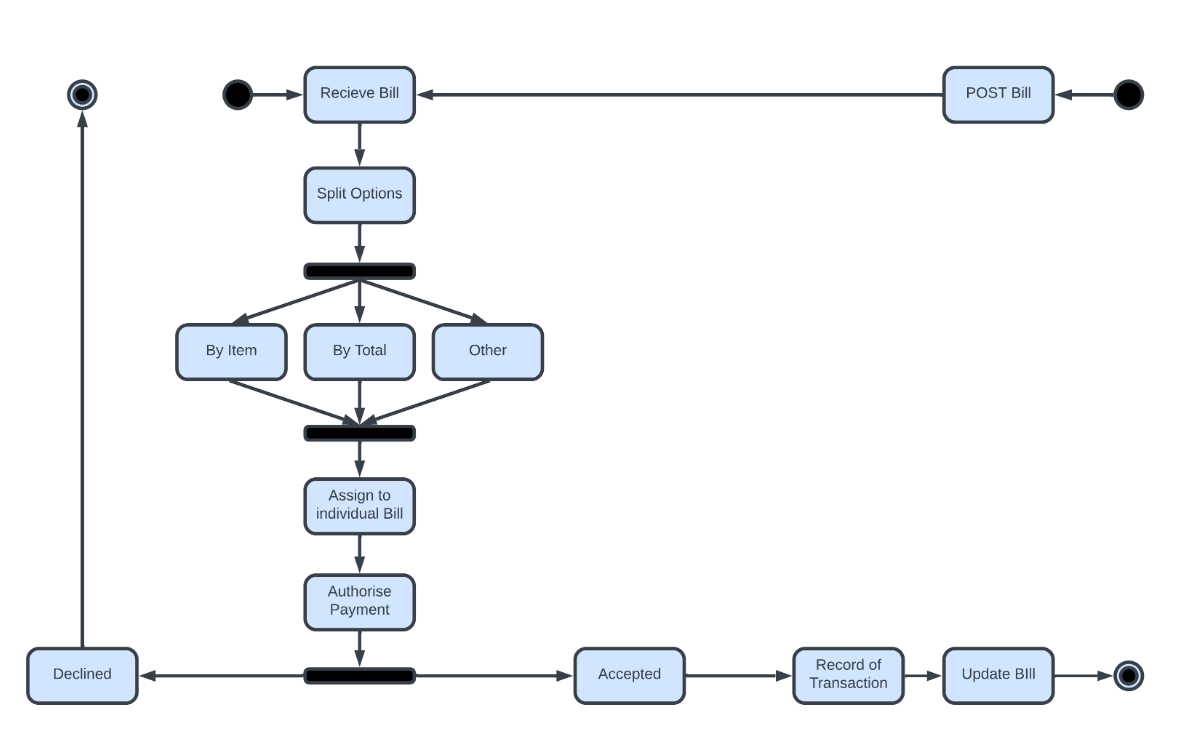
****

Figure 11 Activity Diagram Paying Split Bill DivvyUp

## Data Description

## Data Design

Data design is the first step in the design implementation. This will result in less complicated, modular, and useful program structure. The data classes, objects, attributes, and relationships represented in the entity relationship diagrams and the information in the data dictionary are the basis for the data design.

The structure of the data can be viewed at three levels, program component level, application level, and the business level.

**The component level:** The data structures and algorithms required to manipulate them is necessary.

**The application level:** This is critical to convert the data model into the database so that the business objectives can be achieved.

**The Business level:** The collection of information stored in the different databases should be recognised into data warehouse, which enables the data drilling for the business.

DivvyUp is using dual databases, MongoDB for API storage, and PostgreSQL by default for data security, maintenance, logging, and incidental data storage.

## *Information Architecture*

The following diagrams and chart (Class Diagram, Entity Relationship Diagram) have been presented in a traditional form so that the reader of this document can understand the relationships and the class structures. Mongo is a non-relational NoSQL database technology; relationships are not applied by the engine itself. The data itself contains relationships, and The ER diagram provides a visualization of this relationship.

## ***Internal data structure***

Mongo makes use of collections and documents. The documents will consist of key value pairs which are the basic unit of data in Mongo. Mongo DB makes it easy to store structured and unstructured data. It uses JSON format to store documents.

## ***Global data structure***

By utilising a Mongo, JSON is the currency of the application. Both the front end and the backend can deal with JSON data and as it is stored in JSON like structure in Mongo only formatting is required for delivery and display.

## ***Temporary Data Structure***

All temporary or analytical data is stored in a more traditional RDMS as supplied by the Azure, and AWS tier systems. We have chosen Postgres SQL as this relational database system.

## Human Interface Design

The UI or user interface is the first thing that users see and interact with. With all UI the landing page should be subtle yet carry all the relevant and vital information to engage the user. The interface should convey all the information required to successfully navigate the page and utilise it for the intended purpose. In the DivvyUP case this means identifying the

**pAPI** is a single page that has a search feature, is browsable, and displays the APIs that are available. A second page will publish when a specific API is chosen from the filtered list. This saves on load times and requires less overhead in resources to publish.

The aesthetic is use of colour and white space and to inform the user of the relaxed and easy-going nature of the site.

## Screen Images

Given the philosophy of the **pAPI** team, this scope was narrowed down to a more simplistic appearance, and functional design. Keeping in line with the philosophy of simplicity bridging the gap between form and function. Figure 8 “Figma wire Frame & Design” is an artist’s representation of RapidAPI and API List inspired designs.

# Test & Implementation:

## Testing Requirements

## Description

## Testing Approach

## Testing Tasks Performed

## Function testing

## Unit Testing

## Integration testing

## Security Testing

# Review of Project Lifecyle:

# Challenges:

# Evaluation:

# Conclusion:

# References:

Barry, E. (2021, February 24). *The best bill splitting apps to use with friends.* Retrieved from Finder: https://www.finder.com/bill-splitting-apps

Blake, W. (1790). *What is now proved was once , only imagined*. Retrieved from Kwize-William Blake Quotes: https://kwize.com/quote/13586

Christopher Tworeck, D. H. (2017, December). *THE UBUNTU PHILOSOPHY AS A MANAGEMENT STARTEGY.* Retrieved from ResearchGate: https://www.researchgate.net/publication/322006692\_THE\_UBUNTU\_PHILOSOPHY\_AS\_A\_MANAGEMENT\_STARTEGY

Cocheo, S. (2022, August 15). *Digital Wallets Will Dominate Global Ecommerce Payments by 2025.* Retrieved from The Finacial Brand: https://thefinancialbrand.com/news/payments-trends/digital-wallets-will-dominate-global-ecommerce-payments-by-2025-150769/

*Invercargill City*. (2018). Retrieved from StatsNZ: https://www.stats.govt.nz/tools/2018-census-place-summaries/invercargill-city

Linkletter, B. (2018, June 20). *Open-Source Routing and Network Simulation.* Retrieved from Brain Link Ltter: https://www.brianlinkletter.com/2018/06/create-a-nested-virtual-machine-in-a-microsoft-azure-linux-vm/

Loria, K. (2018, February 16). *Restaurants Trend Toward Cashless Payments*. Retrieved from Upserve byLightspeed: https://upserve.com/restaurant-insider/pay-table-new-technology-impacts-payment-choices-restaurants/

Riley, C. (2015, June 17). *8 Brilliant quotes on Hospitality Technology from HITEC's exclusive Hall Of Fame*. Retrieved from Hospitality Net: https://www.hospitalitynet.org/external/4070736.html

Survey, M. (2022, September 1-30). Market Research Hospitality Technology. (M. Survey, Interviewer)

Voicu, L.-A. (2021, December 21). *5 Simple & Effective Customer Retention Strategies for Restaurants*. Retrieved from Oracle GloriaFood: https://www.restaurant-website-builder.com/customer-retention-strategies-for-restaurants

Will Miao. (2021, January 25). *The Future of Payments: 2021 and Beyond*. Retrieved from Worldline: https://www.paymark.co.nz/blog/the-future-of-payments-2021-and-beyond/

# Appendices

Regarding Redis, that was just something that I might have to look into - I am not sure if I am going to need to use it for our current 'skateboard' model. No other languages, the tech stack was chosen for consistency - since they all use JavaScript, it makes it easier to research, and working with any part of the tech stack will actually provide experience for every part of the tech stack, which hopefully ( ) will make development faster since you are constantly improving a single skill. I think we will be going Azure at this stage; I have not actually tried deploying the application on Azure, but from what I hear in cloud class, and my own research, it seems they have a free tier available to use, which I can just slap the application into for now. If anything changes regarding deployment I will let you know. For development environments, we are obviously using Git for version control, and are storing it in a private GitHub repository so that both Yuhei and I can collaborate on the code. I personally use VS Code, not sure what Yuhei uses, but I imagine its either VS 2022, or VS Code as well. I cannot remember every extension that I am using, but I can look that up and send it through if you really want - most of them are quality of life things for me personally, like themes and the like, so I am not sure if you even want to include that

NextJS - Frontend Chakra UI- Frontend component library NextJS - Backend ws - NodeJS WebSocket library MikroORM - ORM for NextJS MongoDB - Document based database

# Glossary of Terms: