# COSC401

## Smart Ledger: Preliminary Design Report

December 9, 2021

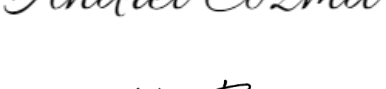

**Authors:**

Tyler Beichler, Emanuel Chavez, Blake Childress,

Lillian Coar, Andrei Cozma, Jacob Leonard, Hunter Price

**Advisor:**

Dr. Bradley Vanderzanden

## EXECUTIVE SUMMARY

The goal of this project is to create an application to support shared spending within multiple groups that facilitate the recording of spending between multiple parties. This application will prevent the loss of money through dropped or forgotten transactions within the group. The web application should provide support for families, friends, roommates, couples, and other groups who want to audit their spendings, settle balances, and pay other members outside of the application.

For hosting, the team looked at AWS, GCP, and Azure and chose AWS for the ease of scalability. The front end will be built with React.js instead of Vue or Angular for the team's level of familiarity. The receipt OCR will be achieved through Tessaract.js. The machine learning models to classify item categories will be written with Tensorflow.js.

The application is also built around five major engineering characteristics, which were determined by analyzing the project goals and requirements. These five requirements are: accessibility, reliability, response time, scalability, and security. Accessibility and scalability are constraints, whereas the rest are variables that can be measured.

# CONTENTS


EXECUTIVE SUMMARY 2
CONTENTS 3
1. PROBLEM DEFINITION & BACKGROUND 5
2. REQUIREMENT SPECIFICATIONS 7
Table 1. Customer Requirements 7
Table 2. Engineering Requirements I 8
Table 3. Engineering Requirements II 9
3. TECHNICAL APPROACH 10
Figure 1. Design Decisions 10
Table 4. Front End Framework Weighted Decision Matrix 11
Table 5. Backend Framework Weighted Decision Matrix 12
Table 6. Database Weighted Decision Matrix 12
Table 7. Cloud Services Weighted Decision Matrix 13
4. DESIGN CONCEPTS, EVALUATION, AND SELECTION 14
Table 8. Database Weighted Decision Matrix 14
Table 9. Backend Frameworks Score 14
5. DELIVERABLES 15
Table 10. Main Deliverables for Working Prototype 16
Table 11. Front-End Design Deliverables 16
Table 12. Proof of Concept Deliverables 16
Table 13. Miscellaneous Deliverables 16
6. PROJECT MANAGEMENT 18
Table 14: Project Schedule 19
7. BUDGET PROPOSAL 20
Table 15: Anticipated Spending 20
8. REFERENCES 21
9. APPENDIX 22
Appendix A: 22
Figure 2. Business Model Canvas 22

## 1. PROBLEM DEFINITION & BACKGROUND

The application aims to solve the problem that there is no good streamlined solution to support shared spending within multiple groups of individuals effortlessly. People frequently share expenses with their friends, family, and other associates within various groups of individuals, rather than splitting the bill solely between the two people. This transaction occurs between multiple groups quite frequently, making it difficult to keep track of everything. One of the biggest problems is that there may not be enough transparency or organization while spending to settle balances quickly and efficiently.

The few products that were found that allow these features are not multiplatform or have limitations and missing features. There are some applications that are good at one aspect of expense sharing, such as restaurant bill splitting, and others that excel in another. A few of the current competitors are SplitWise, Spend Together, FinSplit, MonShare. These applications aim to solve this problem and inconvenience, but each has its own set of shortcomings. Some applications are relatively good, but they are exclusive to iOS or Android, require monthly subscriptions for essential features, or do not have easy and intuitive interfaces.

Further, Venmo and CashApp are two apps that have been made that can perform similar functions when used together with applications like Microsoft Excel or Google Sheets. The disadvantage is that users must keep track of the purchases and request/pay the other parties on their own after deciding what is owed. While this is an option, it is up to one of the parties involved to pay or request the money.

The team intends to address these issues by creating a free and comprehensive solution called Smart Ledger, which will be available on all major platforms. One of the basic functions of Smart Ledger is the ability to create groups of people where expenses are shared. Within these groups, purchases can be split up among the members. Another function is the manual insertion of purchases for when receipt scanning is not an option. The main priority is to ensure these functions work as they are the core of what users will be doing and to make a simple and effective user interface. The group decided to develop a web application to ensure access on any device with access to a browser. Later, the group plans on adding receipt scanning to speed up the process of adding multiple purchases. Other planned features include automation, such as sending direct Venmo and PayPal requests to allow non-users to use the app directly. Advanced features may be primarily available to premium users, while maintaining the core functionality for free users. These advanced features can include but are not limited to automatic purchase classification, spending analysis. The background needed to complete this project includes knowledge of databases and managing them, web apps, interpreting financial statements, and knowledge of expense tracking.

Smart Ledger will be able to be utilized by both individuals and businesses alike. Individuals can use it for splitting expenses such as groceries, rent, and restaurants. Businesses will be able to

use the app to keep track of costs in certain departments and how much is being spent monthly or yearly. Both people and businesses can use the recorded data to plan for their future expenses and change where their funds are being utilized.

## 2. REQUIREMENT SPECIFICATIONS

This section will bring together the app's purpose, specifics, features, and information about how the product will serve the users' needs. The team will discuss the technical aspects of the project, user flow, and key features to provide a detailed description of the existing problem and the solution.

### Customer Requirements

The web application will provide a simple, appealing, and intuitive user interface for users to easily navigate the application and perform tasks. The user interface must also function in a logical manner for the target audience. The front-end must work seamlessly on all major platforms and devices, including Windows, Linux, macOS, Android, and iOS. The application must be responsive and must resize correctly and be functional on mobile devices. Must function and display properly in the various browsers and browser versions, including but not limited to Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. The web application will be thoroughly tested on various different platforms, web-browsers, and screen orientations. Finally, it must allow a user to navigate back and forth within the application without having to use the back button or other browser navigation functionality.

| Customer Requirements |
|---|
| 1. Free to download and use |
| 2. Multi-platform support |
| 3. Support multiple currencies |
| 4. Support multiple user-defined groups |
| 5.Calculate totals for the group |
| 6. Split expenses by percent or an exact amount |
| 7. Payment deadlines & reminders |
| 8. Activity feed & dashboard |
| 9. Secure Network Protocols |
| 10. Data Encryption |
| 11. Receipt Scanning |
| 12. Activity Evaluation |

Table 1. Customer Requirements

### Functional and Technical Requirements

The application is expected to adhere to industry best practices. Several requirements regarding appearance are described in this section.

Proper exception handling in server-side code. Code exceptions must be handled in a user-friendly manner by displaying a custom error page that does not display information such as database object names or source code.

Form fields must be validated to ensure required fields are completed, numeric fields have numeric data, and data input is properly formatted (e.g., date, email address).

User data can be protected through HTTPS during transmission and then encrypted when it reaches the database server to ensure integrity and confidentiality. Administrators can monitor all points of access and implement firewalls to prevent unauthorized users from entering the server.

Reports of the user activity will be needed to account for the best experience the application can deliver. This includes reporting bugs and other flaws that can be fixed as quickly as possible.

The web application must use styles that are consistent throughout the application and within the associated website, including:

- The use of capitalization (e.g., title case vs. sentence case).
- The use of punctuation (e.g., consistent use/no use of colons on labels).
- Error messages must appear in a consistent location and style.
- Consistent use of any web document notations (e.g., PDF, DOC, etc.).
- Layout/spacing (e.g., the space between a field label and input control).
- Descriptive metadata titles and descriptions.
- Form controls that are not available must be hidden--no use of inactive controls.

| Engineering Characteristic | Units/Options | Significance |
|---|---|---|
| Accessibility | N/A | Constraint: Must be available on all major platforms and have an intuitive user experience |
| Availability | Uptime/Downtime | Variable: Determined by the dependability of the project and servers |

Table 2. Engineering Requirements I

**Tools and Technologies**

The cloud service will need to integrate the full stack efficiently with documentation to work with a variety of web servers, databases, and other services together.

The backend server will provide API endpoints for performing queries to a database that stores user and group information as well as their respective ledger transactions.

There will also be additional proof of concept features that will showcase various possibilities for Optical Character Recognition (OCR) and Artificial Intelligence (AI) tools to perform various actions.

For example, OCR may utilize image conversion of a picture to machine-encoded text for receipts and other related documents. Further, the back-end will be able to use Artificial Intelligence (AI) tools to help users optimize savings based on the history of their spendings and budgets. These tools will focus on machine learning models to categorize spendings as well.

**Availability, Performance, and Scalability Requirements**

Availability is a key feature as the group aims to cover most devices and platforms such as establishing an integrated native application on iOS and Android. However, it was advised the group should focus on one or the other which is still to be decided. The design will be available on desktop and mobile devices which will be accessible to the web. If the user does not have an internet connection, the group aims to provide offline capabilities so that the user can still record information whenever they need it. Performance is a key feature as an unresponsive application will likely make the customer frustrated. The application web page should load as quickly as possible and any actions taken within the application should be fast, giving a sense of responsiveness.

| Engineering Characteristic | Units/Options | Significance |
|---|---|---|
| Response Time | ms | Variable: Influenced by the efficiency of the application and quality of the servers/services |
| Scalability | N/A | Constraint: Relies on the product architecture and services chosen |
| Security | N/A | Variable: Stems from technology we choose and secure coding practices |

Table 3. Engineering Requirements II

## 3. TECHNICAL APPROACH

In this project the team seeks to create a fully operational, streamlined, cross-platform web application. This application should support all users in any device and browser used. Further, the application should be easy for the user to log purchases without any hassle of using the application itself.

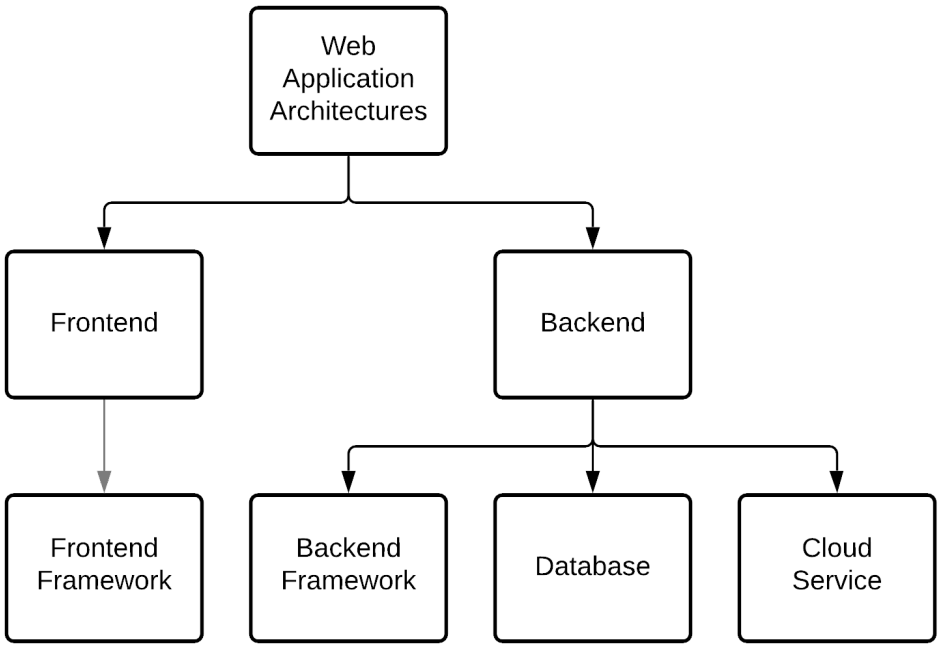



Figure 1. Design Decisions

The web application will consist of 2 major sections: the frontend and the backend. The first decision that must be made is the overall architecture of the application. There are 3 main architectures that are considered. The first is the legacy HTML web application structure. In this architecture, the backend consists of a server that serves the client complete HTML pages handled by business logic in the server. The user must fully reload the page to see any changes made by the server. The second architecture to consider is a widget based web application structure. In this, the creation of the HTML page structure is handled by a webservice. Requests are made from the client, and the server sends back data to populate the page with. This structure typically takes more time to develop full applications with. The last architecture to consider is the single-page web application architecture (SPA). In an SPA, the client must only receive the preliminary web page structure once upon entry then each of the components are updated without needing to refresh the page. This architecture is considered by the industry to be the most agile and popular type. Therefore, the team will only consider frontend and backend

options that support development of a single-page web application. The following design decisions are outlined in Figure 4.

There is one major decision that must take place for the frontend. This is the frontend framework. Here the team considers three options: Angular, React, and Vue. For these options, the team must consider the engineering characteristics: accessibility, reliability, response time, and scalability for each framework. To do this, a weighted decision matrix was created. The weights for each factor to consider is a rating from 1 to 3 of each framework. Each weight in a row is summed to give a final score for each framework. In table 5, the team found that React would be the best decision with the engineering characteristics.

| Framework | Accessibility | Reliability | Response Time | Scalability | Score |
|---|---|---|---|---|---|
| Angular | 2 | 2 | 1 | 3 | 9 |
| React | 3 | 3 | 2 | 2 | 10 |
| Vue | 1 | 1 | 3 | 1 | 7 |

Table 4. Front End Framework Weighted Decision Matrix

In the accessibility section, the team found that React had the best support on all modern browsers. This was closely followed by Angular. Vue had the worst support as it does not fully support some older browser clients. React has the best reliability as the library has the most support and therefore bugs and issues in the framework are fixed faster. Response time was dominated by Vue which is known to be a very lightweight framework. React then Angular followed with slower response times. The team found Angular has the best scalability in the sense that the framework gives the developer a very defined structure that he or she must follow. When following that structure, scaling the application is better streamlined. React was closely second, followed by Vue which lacked full scalability ability.

The backend portion of the web application is comprised of 3 sections: the backend framework, the database, and the cloud hosting service. The backend framework was guided by the engineering characteristics with an additional attribute for machine learning support. Our backend framework weighted decision matrix, Table 6, follows the same format as the previous table.

| Framework | Accessibility | Reliability | Response Time | Scalability | ML Support | Score |
|---|---|---|---|---|---|---|
| PHP | 1 | 1 | 1 | 2 | 1 | 5 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Node.js | 3 | 3 | 3 | 3 | 2 | 14 |
| Flask | 2 | 2 | 2 | 1 | 3 | 11 |

Table 5. Backend Framework Weighted Decision Matrix

The team found Node.js to be the most accessible of all of the frameworks as Javascript is supported on all browser clients and servers. Flask requires an additional Python installation. For a similar reason, Node.js has superior reliability. Node.js is also scalable as it is the industry standard for a backend framework. PHP is scalable as well; however, it requires more code to be written. Flask is known to have problems with scalability. Lastly, Python is known to be superior for fast machine learning development and support with its numerous libraries, so Flask is a necessity. The score of the resulting weighted decision matrix has Node.js as the best decision for a backend; however, the team needs fast machine learning development as well. This means that the team will be using both Node.js and Flask as the backend framework.

The engineering characteristics guided the choice of database. Our database weighted decision matrix, Table 7, follows the same format as the previous table.

| **Database** | **Accessibility** | **Reliability** | **Response Time** | **Scalability** | **Score** |
|---|---|---|---|---|---|
| MongoDB | 2 | 3 | 1 | 3 | 9 |
| CouchDB | 3 | 1 | 3 | 1 | 8 |
| Couchbase Server | 1 | 2 | 2 | 2 | 7 |

Table 6. Database Weighted Decision Matrix

The engineering characteristics guided the choice of cloud service. Our weighted decision matrix, Table 8, follows the same format as the previous table.

| **Cloud Service** | **Accessibility** | **Reliability** | **Response Time** | **Scalability** | **Score** |
|---|---|---|---|---|---|
| AWS | 2 | 3 | 1 | 3 | 9 |

| | | | | | |
|---|---|---|---|---|---|
| GCP | 3 | 1 | 3 | 1 | 8 |
| Azure | 1 | 2 | 2 | 2 | 7 |

Table 7. Cloud Services Weighted Decision Matrix

The team ultimately chose a single-page web application using React as the frontend framework, Node.js and Flask as the backend framework, MongoDB as the Database, and Amazon Web Services as the cloud service.

From a design point of view, the application will consist of multiple tools that allow the user to gain more insights from their spending habits as well as to be able to more easily use the application interface. First, the application will have an algorithm that acts as a ledger that keeps track of all purchases. This algorithm will calculate how much each user has contributed to the ledger and then calculate how much each person is owed from the ledger when a payout is due. Second, the application will have a machine learning classifier that will predict the type of purchase based on the description of an item. This is especially useful to a user as they can go back through their history and get a summary of their travel expenses or food expenses. Third, the team will have a machine learning model coupled with statistical models that show and predict the trends of the spending habits of the groups. Fourth, the user interface will be designed to streamline the input of all users spending input. Lastly, the team will show the user suggestions of who should make the next purchase in a group depending on who has not put an equivalent amount of money into the ledger.

## 4. DESIGN CONCEPTS, EVALUATION, AND SELECTION

| Component | Tools/Libraries |
|---|---|
| Frontend | React.js |
| Backend | Node.js & Flask |
| Hosting Service | Amazon Web Services (AWS) |
| Database | MongoDB |
| Optical Character Recognition (OCR) | Tesseract.js |
| Artificial Intelligence (AI) | Tensorflow.js |

Table 8. Database Weighted Decision Matrix

**Concept Evaluation**

**Design 1: Frontend - React.js**

Establishing a frontend framework is critical for usability for the customer and the developers. The JavaScript library, React.js, is an open-source framework that is best used for user interface and interactive components which are nicely organized for the developers. In addition, the framework saves time for re-utilizing components which prevents bugs in the application. The customer will be able to easily interact with the application and facilitate the necessary services.

**Design 2: Backend - Node.js and Flask**

| Criteria | Pricing | Performance | Security | Total |
|---|---|---|---|---|
| Node.js | 3 | 3 | 2 | 8 |
| Flask | 3 | 2 | 2 | 7 |

Table 9. Backend Frameworks Score

The biggest factor for choosing a backend framework was how well it is able to handle data through real-time applications. Node.js offers low response time and high output capability to serve multiple requests which can save time and resources for both the developers and customers. Most of the requests will be handled with this framework such as the database, but

the machine learning and artificial intelligence tools will be handled through Flask. Flask is a python-based framework that will handle code on the server when dealing with tasks such as receipt scanning or user reports. While utilizing both frameworks at the same time may be difficult, Node.js will be the primary backend framework as it will be easier to reuse most of the code. The customer will be satisfied with its many benefits for being able to client requests at a reasonable time.

**Design 3: Backend - Amazon Web Services (AWS)**

Amazon Web Services (AWS) offers reliable and cost-efficient cloud computing services that can provide a platform to utilize the full stack. A free tier is offered to all developers to host a webserver and database with easy-to-follow documentation. The documentation includes basic configuration for network protocols which is essential for hosting and connecting the servers. By enabling configuration for network and security protocols, the application will also provide better security that can be managed by the team as needed for the customers.

**Design 4: Backend - MongoDB**

A database is essential for the application as customers will need accounts to store their personal information, transactions, and group affiliation. There is plenty of sensitive data that the application will be dealing with, so how the data is stored and accessed will be highly considered. In addition to the database prioritizing consistency, MongoDB is also adamant on performance, simplicity, and flexibility which make for a great choice for both the developers and customers. The environment is quick to set up and provides a dynamic schematic architecture that works with data and storage. The customer can ensure that the flexible database model can adapt to any security changes as businesses and technologies evolve. Additionally, because MongoDB stresses consistency in its design, the users will always have up to date data on their device. This allows users to have confidence in the data. This is important as they will be tracking their spending through the application.

**Design 5: Backend - AI/ML Tools**

The Artificial Intelligence and Machine Learning tools for the application will be written in Python. It is common to use Python for heavy numerical computation as it has numerous libraries that allow for extremely fast development. The flexible libraries will allow us to train models more effectively which in turn will end with the application getting more functionality. This is beneficial to users as they will have better insights on their spending habits and an easier to use graphical user interface with built in intelligent systems quicker with python as the tool.

## 5. DELIVERABLES

In this section, the team will define the goals towards which the team places the focus for this project. First, the team will explain the project objectives, listed in order from most important to

least important, as well as any customer requirements that require it. Finally, the team will explain how these requirements relate to customer needs and how the team arrived at the conclusions.

| **Principle Component Deliverable** | **Notes** |
|---|---|
| Front-End | Convenient and appealing web application. |
| Back-End | Secure protocols for transmitting data/requests. |
| Database | Encrypted storage of user data. |

Table 10. Main Deliverables for Working Prototype

| **Front-End Design Deliverable** | **Notes** |
|---|---|
| Design Guidelines | Define how the application's front-end should look and behave with user interaction. |
| Wireframes/Renders | Generated based on design guidelines. |
| Graphics | Any images or icons used on the front-end |

Table 11. Front-End Design Deliverables

| **Proof-of-Concept Item Deliverable** | **Notes** |
|---|---|
| Artificial Intelligence | AI tools to evaluate user activity and transactions. |
| Optical Character Recognition Tools | ML tools to recognize text in an image (i.e. receipt). |

Table 12. Proof of Concept Deliverables

| **Miscellaneous Deliverables** | **Notes** |
|---|---|
| Documentation | Clear and concise instructions/information regarding the application. |

Table 13. Miscellaneous Deliverables

The customer will be delivered a web application that offers a solution to shared ledgers. It will also be accessible and functional on all browsers. The frontend will implement an appealing design as well as minimal pages for usability purposes. The backend will offer secure protocols to ensure the data being handled is protected from unauthorized users from accessing or

modifying it. The Artificial Intelligence and Machine Learning tools will offer services such as transaction evaluation and receipt scanning for convenience. Lastly, the documentation will provide an outline of usage and how the team implemented the application.

These deliverables have been developed and confirmed with the  team members, team sponsor (Dr. Bradley Vanderzanden), and third-party experts in software development. The importance of each deliverable is listed in the “Notes and Images” section, but a good rule of thumb is that these deliverables are listed by order of priority, descending from greatest to least.

## 6. PROJECT MANAGEMENT

During the development process, each key component of the application is allocated to a member of the team who is responsible for monitoring its progress and development as well as providing regular updates. To ensure that all requirements, deliverables, and milestones are met, the project manager will supervise the whole front-end and back-end development teams.

While working together, the group will collaborate closely throughout the entire process to ensure that everyone is on the same page and that everything goes according to plan. Every week on Friday from 11:45 a.m. to 12:35 p.m., the group will convene for standup meetings to discuss the progress of each team and any roadblocks. This will also provide an opportunity for us to discuss any concerns or challenges that any of the group members may be experiencing, as well as to ensure that the group is on schedule to complete the project on time.

It is the purpose of the Fall semester of 2021 to acquire a head start on planning, research, and discovery, as well as the creation of a very basic architecture and prototype for the proposed application. It will consist of a front-end and a back-end, which will help us to acquire a better understanding of the overall evolution of the application by building a more concrete prototype. With the second semester in hand, the goal is for the team to continue to develop all of the application's basic components while also adding additional features in order to broaden the application's reach in accordance with the project deliverables and objectives.

As a recommended timeline for the production of the proposed application for the Spring 2022 semester, the following schedule has been established and is available for review. He or she will be heavily influenced by the effectiveness of research and development initiatives, as well as the availability of resources and participation from important players.

| Week | Goal |
|---|---|
| 1 | Planning, Research, and Discovery |
| 2 | Research and Briefs for Front-end and Back-end Architectures |
| 3 | Overall Design Planning and Wireframe |
| 4 | Homepage/Front Page Design & Development |
| 5 | Internal Pages Design & Development |
| 6 | Content Creation, Features, and SEO |
| 7 | Overall Development and Coding |
| 8 | Beta Testing, Feedback, and Improvements |
| 9 | Implementing Fixes & Improvements from Beta Testing |

| | |
|---|---|
| 10 | Launch & Working Demo |
| 11 | Final Improvements & Documentation |
| 12 | Working on Final Reports & Documents |
| 13 | Finalizing Reports & Presentations |
| 14 | Wrapping Up |

Table 14: Project Schedule

## 7. BUDGET PROPOSAL

| Item | Estimated Cost |
|---|---|
| AWS Hosting Services - Free Tier | $0 |
| Jira - Student | $0 |
| Discord | $0 |
| GitHub Student Developer Pack | $0 |
| Domain | $15/Year |
| Total | $15 |

Table 15: Anticipated Spending

The development costs for this project are expected to be minimal. The application will be hosted by Amazon Web Services (AWS) [1]. To host the website and database, it will especially make use of Amazon EC2 Compute, which is less expensive than purchasing dedicated hardware for the purpose (free for 12 months). For the duration of the month, the team will be provided with an additional 750 hours of Linux instance time from Amazon EC2, which will be more than enough for the needs. Even in the worst-case situation, if the project takes more than 750 hours per month, the budget will need to be amended to account for a payment of $0.051 per compute hour utilized (which can be funded by the premium tier of Smart Ledger). Once the project is finished, paid computation hours will be required to support the project's expanding user base; however, this is not covered in this course. Jira will be the primary project management platform [2]. Jira is fully free for students, and it offers a free tier that allows up to ten users to utilize the software. Discord will serve as the primary means of communication for the team. Discord is a completely free service to utilize. Github will be the primary version control mechanism for the project. A free student developer bundle [3] is available on Github, and it contains all of the functionality required to build Smart Ledger. Lastly, a domain name is required for the client audience to have external access to the application, and this typically costs roughly $15 per year to maintain. This takes the overall budget for the project down to roughly $15 dollars.

## 8. REFERENCES

[1] Amazon EC2. (2021, 11 01). Amazon Web Services. https://aws.amazon.com/ec2/?did=ft_card&trk=ft_card&ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc

[2] Jira Pricing. (2021). Atlassian. https://www.atlassian.com/software/jira/pricing?&aceid=&adposition=&adgroup=89541890342&campaign=9124878120&creative=461842798799&device=c&keyword=jira%20pricing&matchtype=e&network=g&placement=&ds_kids=p51241495620&ds_e=GOOGLE&ds_eid=70000001558501&ds_e1

[3] GitHub. (2021). GitHub Education. https://education.github.com/pack

https://www.parvatiandsons.in/

# 9. APPENDIX

## Appendix A:

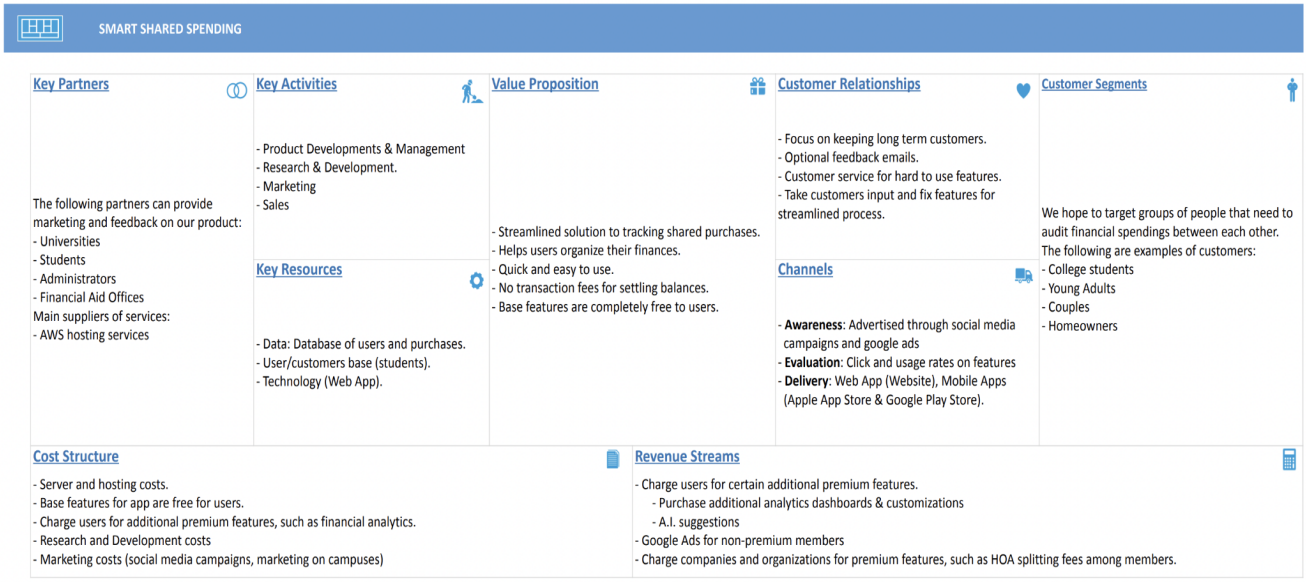



Figure 2. Business Model Canvas