EMarketPlace

Andrew Mikaeel - 100525236

Jasarat Siddiqui - 100619789

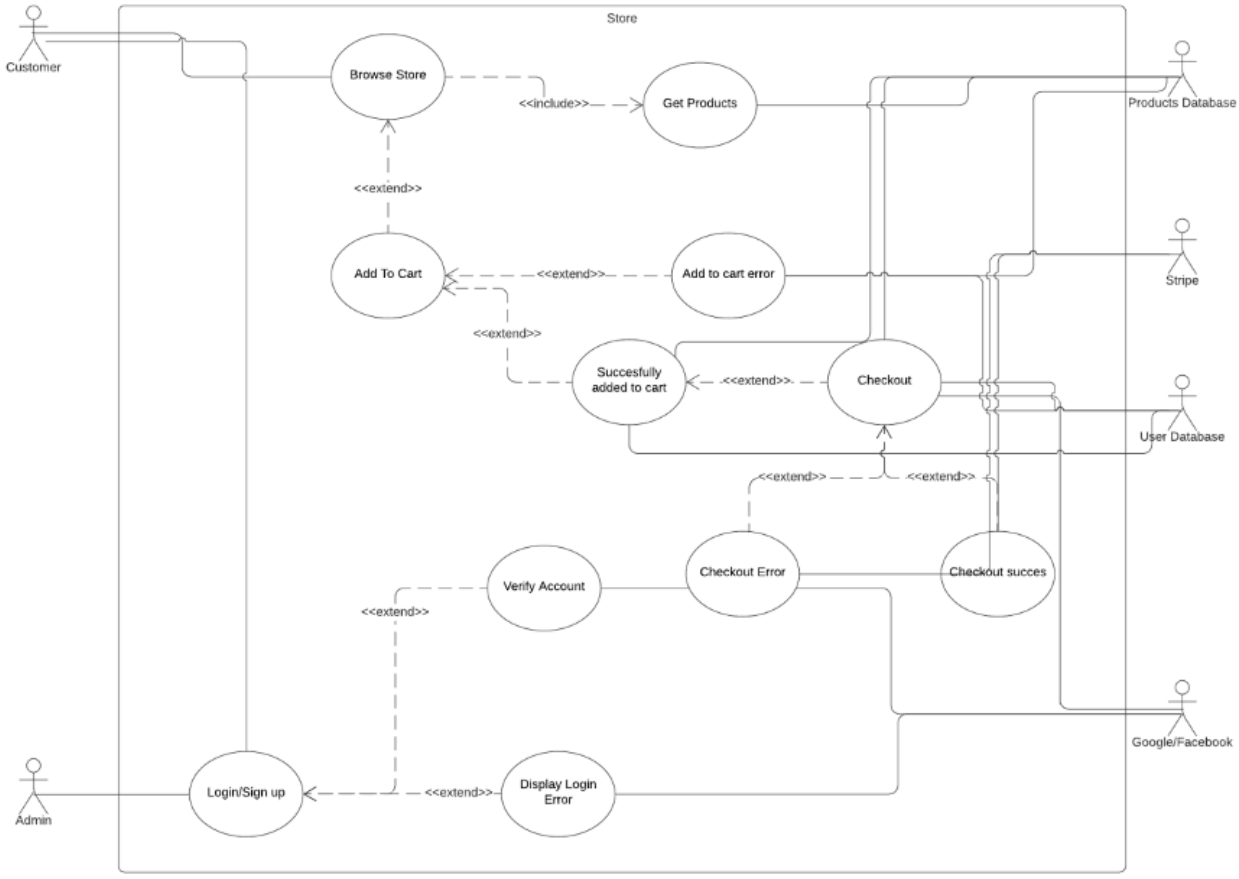
Malaika Sharif - 100651617

1.1 Business Case

Emarketplace is an online platform where users can buy and sell products. The products on display are populated by the users. All payment transactions and the storing of product information is handled by us. The principal use case for our application is to connect buyers and sellers on a beautiful and functional platform. Our aim is to build a platform where sellers can showcase their products in an elegant manner and where sellers can browse those products and purchase them right in our store.

1.2 System Requirements

1.2.1 Use Case Model



Use Case	Description
Browse Store UC-1	This is the bread and butter of our store. This is where the user can browse our selection of products available for purchase. The products and their associated information is stored on our database and subsequently fetched and displayed from our frontend.
Add to cart UC-2	The user can select a product to add to their cart along with a quantity. The cart will update the cost accordingly if the products are available in their requested quantity.
Add item UC-3	Users can add items to the store under their accounts.
Checkout UC-4	This is where the user can begin the checkout process. Final checks of the product's availability as well confirmation that the user is logged.
Login/Sign up UC-5	Here users and admins can login and sign up using our login system.
Add to cart error UC-6	If the items are no longer available the user will get an error.
Checkout error UC-7	If there are any problems during the checkout process - like payment errors or the item is no longer available - the user will get an error.

1.2.2 Quality Attribute Scenarios

ID	Quality Attribute	Scenario	Associated Use Case
QA-1	Security	When a user makes a payment, their payment information and transaction must be processed in a secure and lawful manner. All regulations must be followed during this process.	UC-6, UC-7, UC-8, UC-9, UC-10, UC-11
QA-2	Performance	The user must have a seamless experience when accessing the website. All interactions must be executed elegantly with accurate and performant functionality through a beautiful UI.	All
QA-3	Availability	The Databases and third-Party services must be always available 24/7.	All
QA-4	Modifiability	All components of the system must be modularized and separated as much as possible. For example, the payment processor that we chose - stripe - should be swappable with another service like Square with relative ease.	All

QA-5	Testability	There must be a system developed to constantly test features on the application to make sure that it is always working. The main areas of concern are that all the apis are always functioning as expected.	All
QA-6	Usability	The application should be intuitive and should have a small to no learning curve to the users.	All

1.2.3 Constraints

ID	Constraint
CON-1	The application must run as intended on Chrome, Firefox, Safari and Edge.
CON-2	All the users purchase history must be stored
CON-3	Before processing a payment, the system must be certain that the item/items being purchased are in stock
CON-4	A relational database must be used
CON-5	User should be tracked on the website to create a model of their interests for marketing purposes
CON-6	The user must be informed about this tracking and storing of information and must be presented with an option to opt out

1.2.4 Architectural Concerns

ID	Concern
CRN-1	Designing and implementing a distributed system that can easily scale with a growing user base.
CRN-2	Leveraging the team's knowledge in multiple domains of web development. Frontend, backend and devops.
CRN-3	Appropriately dividing up tasks within the development team.
CRN-4	Develop an overarching architecture for the application.

1.3 The Design Process

Now that we have our requirements and business concerns - we are ready to start designing the architecture of the application. We will be following the ADD approach to design the application - splitting the process into multiple iterations. During each iteration we will be selecting drivers as our point of focus.

1.3.1 ADD Step 1: Reviewing Inputs

Category	**Details**
Design purpose	The purpose of this design is to make a comprehensive design that meets the requirements of the application as well as sets the application up in a way that it is easily scalable and modifiable.
Primary functional requirements	The primary use cases are: UC-1 UC-2 UC-3 UC-4 UC-5
Constraints	All constraints are used as drivers.
Architectural concerns	All architectural concerns are used as drivers.

Quality attribute scenarios prioritized:

Quality attribute Scenario	**Importance to the customer**	**Difficulty to implement According to the Architect**
QA-1	High	Medium
QA-2	High	High
QA-3	High	High
QA-4	Low	Medium
QA-5	Low	Medium

QA-6	High	Medium

1.3.2 Iteration 1: Establishing an Overall System Structure

The goal of this iteration is to meet architectural concern CRN-4 which is to develop an overarching architecture for the application.

1.3.2.1 Step 2: Establishing Iteration Goal by Selecting Drivers

This iteration is driven by meeting architectural concern CRN-4: Develop an overarching architecture for the application, however we kept in mind all major drivers that could affect the overarching architecture of the application including all major use case scenarios. The following drivers were also used as main drivers during this iteration of the design process.

Quality Attributes:

- QA-2: Performance
- QA-3: Availability
- QA-4: Modifiability

Constraints:

- CON-1: The application must run as intended on Chrome, Firefox, Safari and Edge.
- CON-3: Before processing a payment, the system must be certain that the item/items being purchased are in stock
- CON-4: A relational database must be used

Concerns:

- CRN-1: Designing and implementing a distributed system that can easily scale with a growing user base.
- CRN-2: Leveraging the team's knowledge in multiple domains of web development. Frontend, backend and devops.
- CRN-3: Appropriately dividing up tasks within the development team.

1.3.2.2 Step 3: Choosing Elements of the System to Refine

Since this is the first iteration of our design, the element that we are 'refining' is the EMarketPlace system in it's eternity.

1.3.2.3 Step 4: Choosing Design Concepts That Satisfy the Selected Drivers

Decision 1: Application type

Design Decisions & Locations	**Rationale**
Create a web application using the client/server model that can run on modern browsers.	This architecture decouples our system into two main sections – the frontend and the backend. By doing this we can make changes to our view without affecting our controllers or models. It also allows us to make changes to our backend without having to make too many changes to our view/frontend. There is also the added benefit of making our system widely accessible. Because this is a web application all you need is a computer/mobile device with a browser and an internet connection in order to be able access it. Both of which are typically readily available nowadays.

Alternative	**Reason for Discarding**
Mobile application (ios/android)	Although we definitely are planning to create a mobile version of our application, due to a lack of resources we believe that it makes more sense to start with a web application so that our application is accusable by a wide range of devices rather than just one platform.

Desktop application	There are two issues with this architecture. The first is that our application is limited to either Macs OS, Windows or Linux machines. This will eliminate two desktop operating systems from our user base as well as all mobile devices. The second issue is that people will need to download our application to their desktop, which is something that people only do if necessary from our experience. It is much easier to get a user to visit a website rather than download an application. With modern browsers becoming more and more capable each day we don't see a real advantage to desktop applications anymore.

Although we are not making a traditional 'rich client' application - our architecture will closely resemble the architecture of one. This is because we will be using a modern frontend framework that is robust and allows for more processing on the client side. We will still be using a three tier architecture with the client, logic, and the data tiers all present in our architecture. However the lines between the client and logic tiers are more blurred than traditionally seen in our design - this is intentional.

Decision 2: Backend language

Design Decisions & Locations	**Rationale**
Build the backend using Node.js	For the backend we chose Node.js because that is our strongest backend language that is very well documented and tested. It makes working with third part services and SQL easier through frameworks like sequelize and

Alternatives	**Reasons for Discarding**
Python	We considered Python as our backend language of choice, however we decided against it for two main reasons. The first is that most of our expertise lies within Node.js and javascript. The second reason is that we know that python - although easy to use - is not the most performant language out there. Given our priority for high performance we decided to not use python for our backend language.

Decision 3: Deployment Service

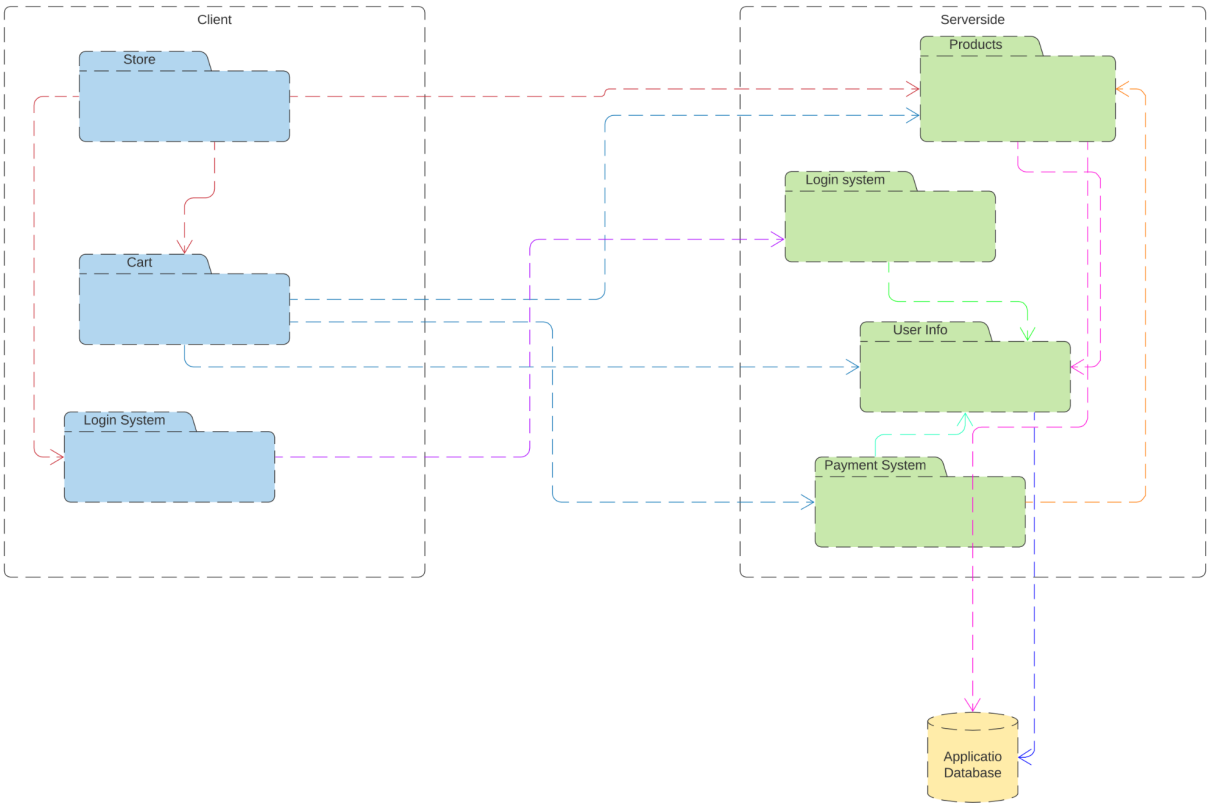
Design Decisions & Locations	**Rationale**
Deploy the system using Firebase	We decided to use firebase to deploy our application because this makes it very easy for us to scale our system appropriately.

Alternatives	**Reasons for Discarding**
Azure, AWS	We chose firebase over these two platforms because firebase is very well documented and we can easily switch over to google cloud platform if that fits our needs more at any time.

Decision 4: APIs

Design Decisions & Locations	**Rationale**
Rest API	We decided to use rest APIs to communicate with our frontend - returning JSON data - as this will make our system highly modifiable and decoupled from our view. This also means that we can expand our application to other platforms (IOS/Android) without having to make too many changes to our backend.

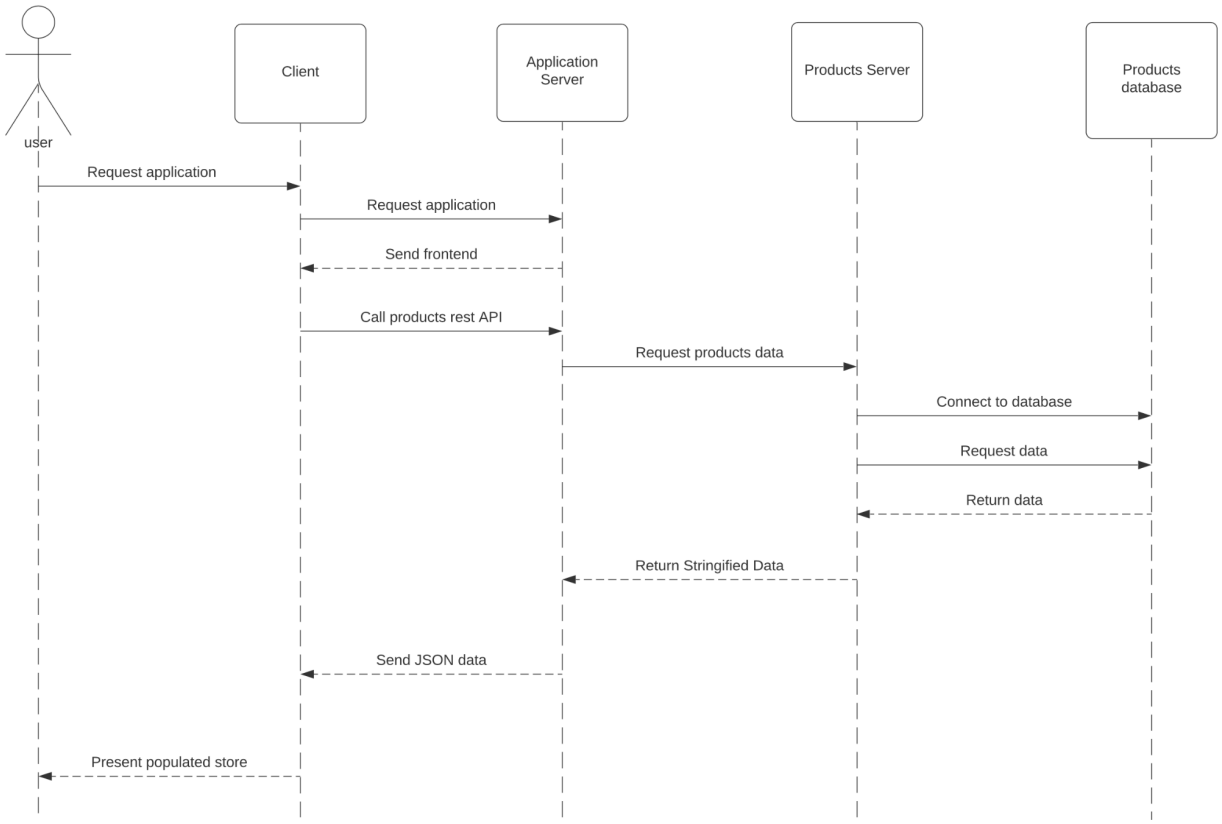
1.3.2.4 Step 5: Sketch Views and record Design Decisions



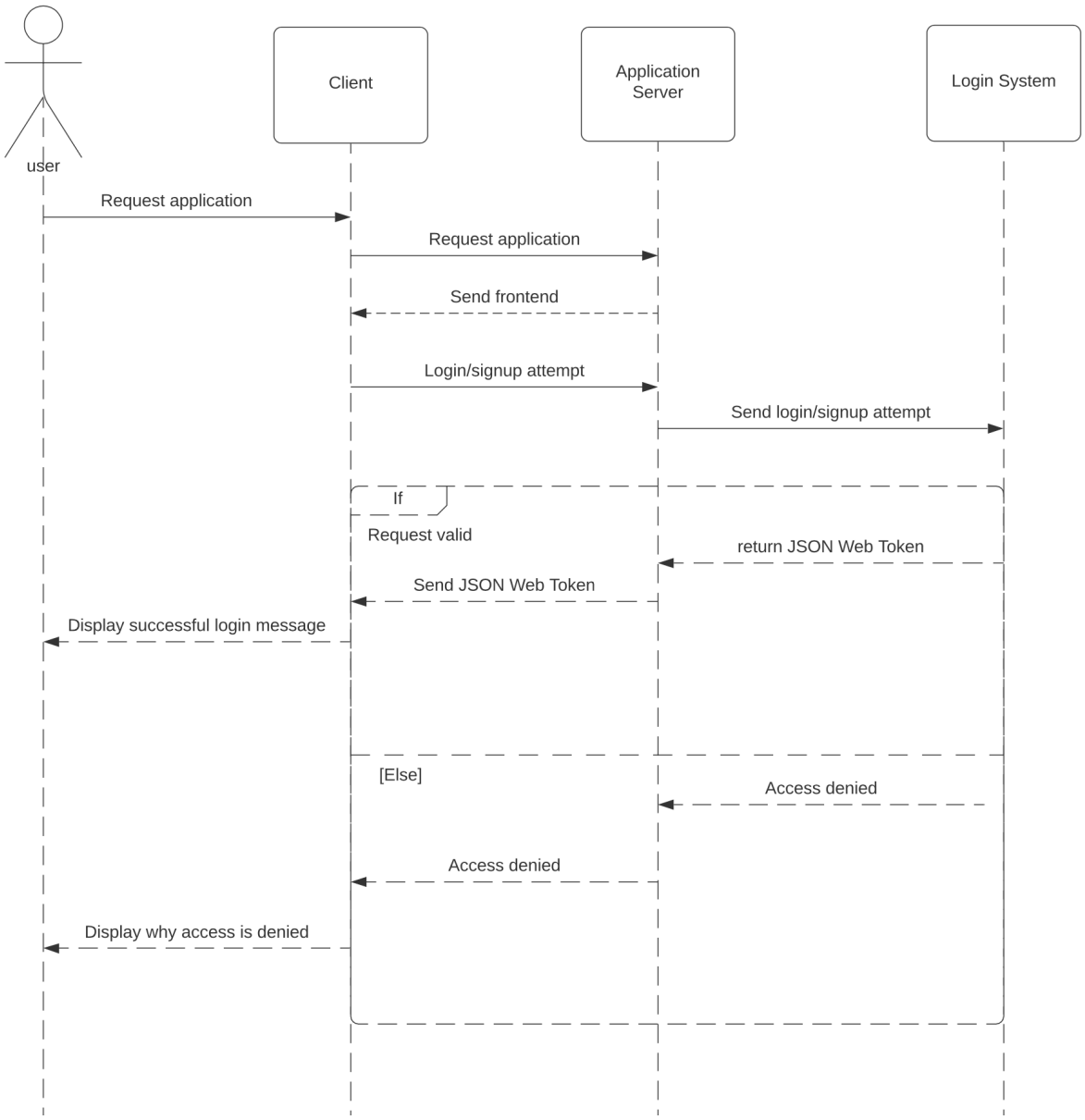
Element	Responsibility
Products	All of our products are stored here along with their associated metadata. Our rest APIs that fetch and return data from the database are in this package as well. The database will be a real-time database in order to have accurate data about product availability at all times in our store.
Login System	This is where login requests are processed. Account information is also stored here.
Payment System	This is where payments that our users make are processed as well as payments we make to our sellers on the site.
User Info	This is where we will have APIs that are associated with the users. This includes if they are vendors or customers. As well as other information like session specific

	information like what is in their cart.
Store	The store is the main screen for our application. This is where the user can see all items on offer on our platform. The store will fetch all of our products from the products package on the backend.
Cart	This is where the user adds items from the store and checkouts out to pay for them - if available.

Sequence diagram for requesting the website and the products that populate the site



Sequence diagram for login system



1.3.2.5 Review iteration

In this iteration we set out to establish an overarching architecture for our application. We believe that the results of this iteration reflect that. We came up with an application type - web application - that will help drive the rest of the decisions we make during the design process. The main elements of the application were also determined along with how they will communicate with each other.

1.3.3 Iteration 2: Identifying Structures to Support Primary Functionality

During this iteration we are looking to provide more detail to our architecture that we developed in iteration one by making more specific decisions to our design.

1.3.3.1 Step 2: Establish Iteration Goal by Selecting Drivers

The goal for this iteration is to provide a more comprehensive architecture to our application - with a particular focus on making decisions that will help us satisfy CRN-3: appropriately dividing up tasks within the development team.

1.3.3.2 Step 3: Choose one or more Elements of the System to Refine

For this iteration we will focus on refining elements in our backend. More specifically we will be refining the login system, the payment processing system, user info, and product elements.

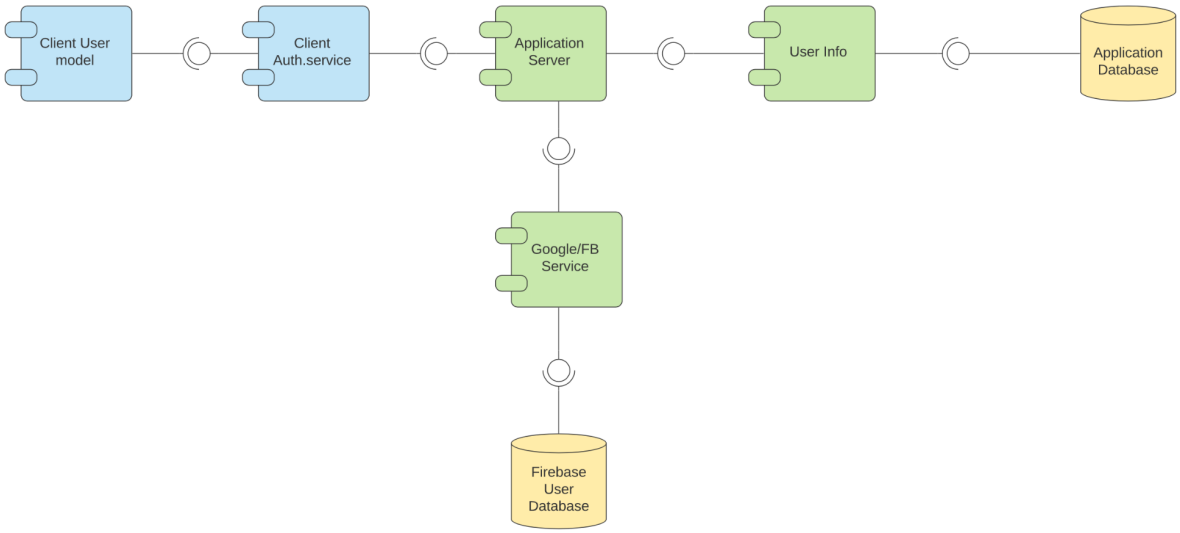
1.3.3.3 Step 4: Choose one or more design concepts that satisfy the selected drivers

Decision 1: Login System

Design Decisions & Locations	Rationale
Login System - Google & Facebook	We chose to only offer sign in through these methods because we believe that they are more secure than a login system that we build ourselves. We trust these brands to handle our authentication and sign up process because they have teams of engineers dedicated to the security of those accounts. They are fast, reliable and most importantly secure.

Alternatives	Reasons for Discarding
In-house login system	There are definitely advantages to a login system made in house. Cost is definitely the most prominent feature. It is much cheaper to build our own login system rather than using a third party system. However we chose against building our system due to the development time it would take as well as the added security headache that it would add to the project. Using a third party system removes the complexity that comes with building a login system from scratch as well shifting the security responsibility to the very capable hands of Google and Facebook.

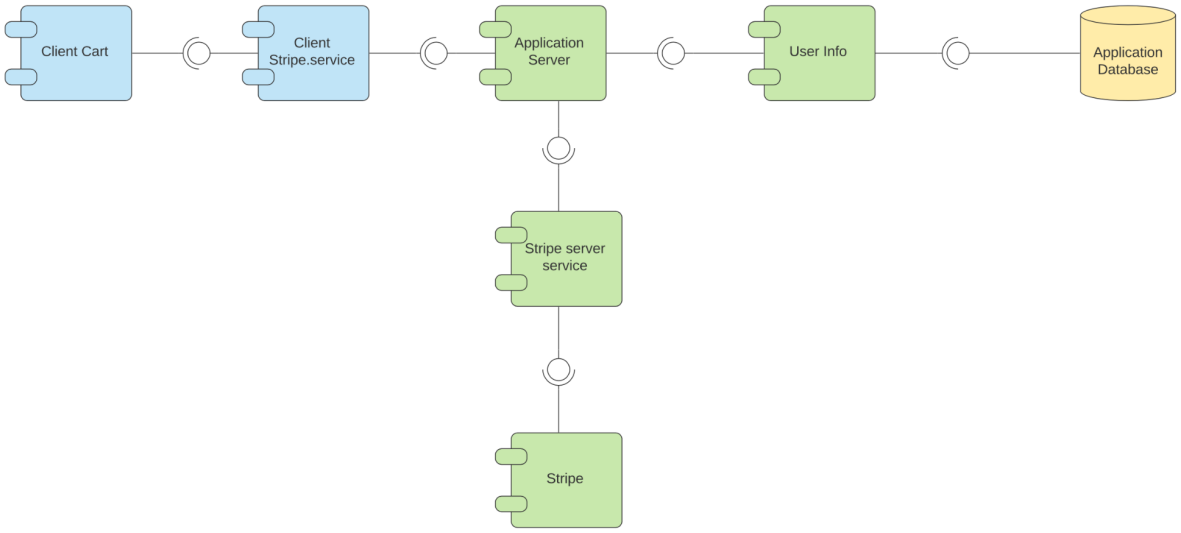
The login system we are using is managed through firebase. Firebase provides us with a table of users, their login method, and a UserUID as a unique identifier. This identifier is a key piece of information that we will use in our application database.



Decision 2: Payment processing system

Design Decisions & Locations	Rationale
Stripe payment service	We decided to use Stripe as our payment service. This is because we believe that Stripe has a great service that is well documented and reliable. We are not looking to reinvent the wheel and create our own payment service. There is a lot of complexity that is associated with transactions that are handled securely and efficiently by stripe. They also have great and simple to use apis that will help speed up the development process greatly.

Alternatives	Reasons for Discarding
Paypal	We disregarded paypal for two reasons. First Stripe's apis are far superior in their ease of use and documentation. Second is that anyone with a credit card can use Stripe to pay. There is no need to make an account. This makes the payment process slightly easier in our opinion.



1.3.2.5 Step 5: Review iteration

The goal for this iteration was to refine components on our backend. We made significant progress in the design of our backend. We made major decisions in regards to the main systems that our application will use. However there is still much work to be done, specifically the database and the APIs still need to be designed. This will take at least one more iteration. However we are satisfied with the results of this iteration as it clears up a lot of the ambiguity regarding our backend infrastructure.

1.3.4 Iteration 3: Addressing Quality Attribute Scenario Driver

In this section we build on our fundamental structures from iteration 1 and 2 with a focus on QA-6: Usability.

1.3.4.1 Step 2: Establish Iteration Goal by Selecting Drivers

The goal for this Iteration is to focus on Quality Attributes QA-6 which is to make sure that the application should be intuitive and should have a small to no learning curve to the users.

1.3.4.2 Step 3: Choose one or more Elements of the System to Refine

The element that we chose to refine for this iteration is the frontend as a whole. Because our architecture allows the frontend and backend to be highly decoupled - we have the luxury of focusing one or the other without our changes affecting each other in a dramatic way.

1.3.4.3 Step 4: Choose one or more design concepts that satisfy the selected drivers

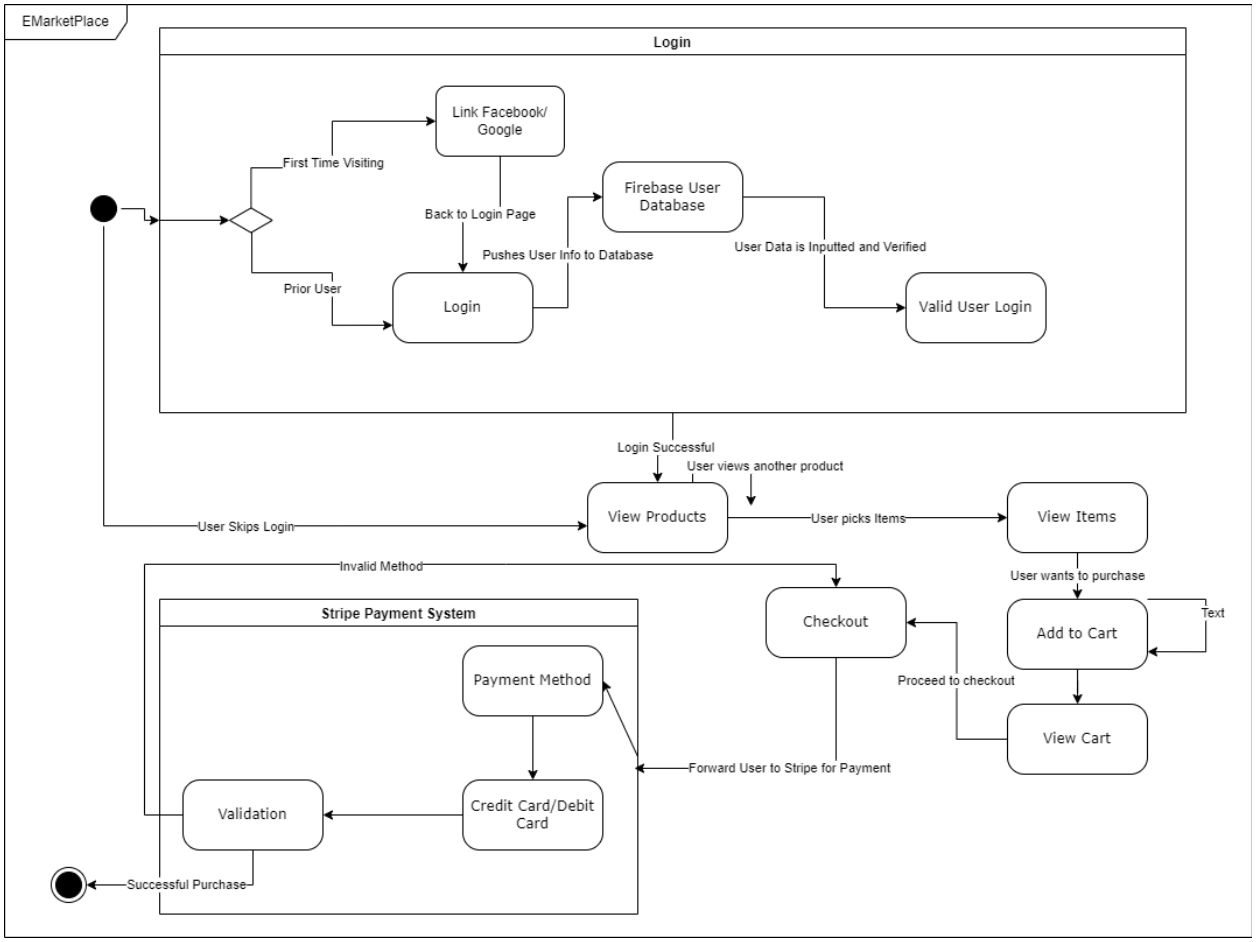
The design concepts used for this iteration are the following:

Design Decisions & Locations	Rationale
Building a single page application using Angular	This is a great frontend framework built by google. It is a well-documented and tested framework that is very popular in the industry at the moment. It is also a highly performant framework as well. This is also where our team is most experienced. - maintainability

Alternatives	Reasons for Discarding
React.js	React is a great framework that we like very much. But considering that Angular ships with typescript out of the box as well as its more structured way of creating applications we preferred it over react.js. We are also more familiar with Angular so that will streamline the development process a little bit.

Design Decisions & Locations	Rationale
Implement caching	We decided to implement caching on the frontend to help reduce the load on our servers and databases.

Alternatives	Reasons for Discarding
Make requests every time you login	Having to login every time you access the site is not something that the users enjoy doing. These logins also cost us money each time the user logs in - since we are using a third party login system.



1.3.4.6 Step 5: Review iteration

The goal for this iteration focuses on Quality Attributes, QA-6 which is to make sure that the application should be intuitive and should have a small to no learning curve to the users. We made major decisions in regards to the User Interface side of our application. We implemented caching in this iteration to reduce the load on our servers and databases. We also decided to go with Angular as our frontend framework. It is built by google, it is a well-documented and tested framework. These were the results of our iteration 3 which we are satisfied with.

1.4 Summary

Overall, we were able to design our software architecture and make major decisions for our design through the ADD process. Our results from iteration one, two and three allowed us to decide on an overall structure for our application, then refine components in our server side of the application and finally refine our user experience and interface. We were successful in showing how we addressed our concerns as the design process progressed. We addressed our primary use cases and quality attributes with each iteration though more iterations would be required to address all other scenarios with high priorities. The project helped us visualize the design of a software architecture and the changes in elements that are needed as the iteration progresses. All in all, we feel as though we were able to address all of the primary use cases, quality attributes and architecture concerns we encountered in our project successfully. We can continue to refine our design in the future using more iterations the goal of the project was acheived.