**CSC 3150 – System Specification**

**(Design Document)**

# **Cover Page**



# **Table of Contents**

Contents

[**Cover Page** 1](#_Toc168345119)

[**Table of Contents** 1](#_Toc168345120)

[**1.** **Executive Summary** 2](#_Toc168345121)

[**2.** **Introduction** 2](#_Toc168345122)

[**2.1.** **Problem Statement / Project Vision** 3](#_Toc168345123)

[**2.2.** **System Capabilities** 3](#_Toc168345124)

[**2.3.** **Non-functional Requirements and Design Constraints** 3](#_Toc168345125)

[**2.4.** **System Evolution** 3](#_Toc168345126)

[**2.5.** **Document Outline** 3](#_Toc168345127)

[**3.** **Structural Model** 4](#_Toc168345128)

[**3.1.** **Model Introduction** 4](#_Toc168345129)

[**3.2.** **Class Diagrams** 5](#_Toc168345130)

[**3.3.** **Metadata** 5](#_Toc168345131)

[**Credit Cards:** Class Diagram 5](#_Toc168345132)

[**Redeem Rewards:** Class Diagram 6](#_Toc168345133)

[**Transaction:** Class Diagram 7](#_Toc168345134)

[**User Account:** Class Diagram 8](#_Toc168345135)

[**4.** **Architecture Design** 10](#_Toc168345136)

[**4.1.** **Architecture Overview** 10](#_Toc168345137)

[**4.2.** **Infrastructure Model** 10](#_Toc168345138)

[**4.2.1.** **Deployment Diagram 1 – Architecture Overview** 10](#_Toc168345139)

[**4.2.2.** **Deployment Diagram 2 – Nodes and Artifacts** 10](#_Toc168345140)

[**4.3.** **Hardware and Software Requirements** 11](#_Toc168345141)

[**4.3.1.** **Hardware Components** 11](#_Toc168345142)

[**4.3.2.** **Required Software Components** 11](#_Toc168345143)

[**4.4.** **Security Plan** 11](#_Toc168345144)

[**4.4.1.** **Security Overview** 11](#_Toc168345145)

[**4.4.2.** **Security Plan** 11](#_Toc168345146)

[**5.** **User-Interface** 12](#_Toc168345147)

[**5.1.** **User-Interface Requirements and Constraints** 12](#_Toc168345148)

[**5.2.** **Window/Screen Navigation Diagram** 12](#_Toc168345149)

[**5.3.** **UI Wireframes** 13](#_Toc168345150)

[**5.4.** **Reports: “Formal Output” Design** 21](#_Toc168345151)

[**6.** **Appendices** 22](#_Toc168345152)

[**6.1.** **Glossary** 22](#_Toc168345153)

[**6.2.** **References / Bibliography** 22](#_Toc168345154)

[**6.3.** **Supporting documentation** 22](#_Toc168345155)

# **Executive Summary**

The primary audience for this document is developers. The developers are the people who code the application. They will follow this document to the best of their ability, and if they want to change anything, they should speak to a higher up. This document will have a lot of different diagrams, each having their own significance. The documents will cover the different classes of the system, the network and security of the system, what each application page will look like.

# **Introduction**

This product is called The Supreme Card. This product is an app that is available to download only from the IOS store. It is a simple app that stores all the user’s credit cards on the app and uses AI to determine which credit card to use in the transaction. When users are ready to pay for a transaction, they press the button of the transaction type that they are doing, and the trained AI finds the credit card with the best cashback and rewards for the transaction. Users can have pre-set preferences on specific cards, which the AI should consider over other credit cards.

## **Problem Statement / Project Vision**

Many consumers have trouble knowing what all their credit cards do and use the same credit card no matter the transaction. If people understood what their credit cards did, they could get more cashback and rewards on each transaction. Users who download our app won’t have to worry about which credit card is best for their transaction because the trained AI will choose the best credit card. Many consumers will save more money and get more rewards with this app.

This app will revolutionize the digital wallet market. People will no longer have to choose which credit card they want manually. This app will consider the transaction type and the user’s preferences when selecting the credit card for the user to use. The banks that choose to partner with us will receive rewards through advertisements for their credit card(s), which will attract more people to sign up for their credit cards. Consumers will realize that there are many other credit cards, and if they sign up for more, they will maximize the chance of getting cash back and rewards for each transaction.

## **System Capabilities**

Enter Credit cards (1): Users will input their credit cards into the app.

AI chooses card (2): The trained AI will choose which credit card to use in the transaction.

Ask for support (3): If the user has trouble with the app, they can contact customer support.

Pay for subscription (4): to use the app, the user must pay a monthly fee.

Redeem card rewards (5): The user will redeem their credit card rewards.

See sections 4 and 5 on the System Proposal for more details about the system’s capabilities.

## **Non-functional Requirements and Design Constraints**

One design constraint that is present is the fact that banks may not want to work with our product. If banks don’t partner with us, then our app won’t be able to grow. Another constraint about the app is that users must pull up an app to use our product, which isn’t as simple as Apple Wallet. Another constraint is that not every place has a tap feature to pay. When paying at some places, people must insert their credit card into the machine to pay, which won’t work with this app. Some non-functional requirements include a simple UI, a secure database, and an easy-to-view section of our app that shows all the cashback the user has got.

## **System Evolution**

* + 1. **Version 2 Changes**

In Version 1, we will have a trained AI to pick a card and highly suggest it to the user, and the user will either tap that card and pay or choose a different card. In Version 2, we will add the choice to blindly follow the AI without having a choice to choose a different card. In Version 2, we will also add a location tracker, so the system already knows what type of transaction is happening.

* + 1. **Version 3 and beyond Changes**

In Version 3, we will create a physical card people can carry around and use and it’ll automatically choose the best card for the transaction based on location to find the closest business.

## **Document Outline**

This document will contain many diagrams for different necessary components of the development process. The first diagram will go over all the different classes needed for the system. The next diagrams will show the network structures and how they work. It will also describe the necessary hardware and software for the system. It will also go over the security overview and plan for the system. The final section will go over the actual application, and what each screen is supposed to look like and how the pages are connected.

# **Structural Model**

## **Model Introduction**

The following model is a UML Class Diagram. There are 4 total classes, and the metadata after the diagram will explain what each attribute and operation do and why each one is important for this diagram.

## **Class Diagrams**

A diagram of a card

Description automatically generated

## **Metadata**

### **Credit Cards:** Class Diagram

Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Read Only** | **Multiplicity** |
| cardNumber | All numbers on the credit card | No | 1..\* |
| cardName | The bank the credit card is from | Yes | 1 |

Operations:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Is Query** | **Is Polymorphic** |
| addCard() | Asks the user to input a credit card | No | No |

Methods:

addCard():

Do

User is prompted to add a card

User enters credit card

If user needs to enter more credit cards

User hits button

While(user has credit cards to enter)

### **Redeem Rewards:** Class Diagram

Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Read Only** | **Multiplicity** |
| valueOfReward | It is the total value of cashback the user can redeem | Yes | 1 |
| numberOfRewards | This is the number of credit cards that can currently reward cashback | Yes | 0..\* |

Operations:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Is Query** | **Is Polymorphic** |
| redeemReward(cardNumber) | This will redeem the cashback reward and put it on the credit card | No | No |

Methods:

redeemReward():

User can see the valueOfReward and numberOfRewards

User clicks the credit card of the reward they want to redeem

The valueOfReward is transferred to the cardNumber

### **Transaction:** Class Diagram

Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Read Only** | **Multiplicity** |
| type | This is the type of the transaction | No | 1 |
| creditCard | The credit card the AI chooses | Yes | 1 |

Operations:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Is Query** | **Is Polymorphic** |
| ready() | If the user is ready to pay, they click this button | No | No |
| chooseType() | They choose the type | No | No |
| AICard() | This is the card the AI choose | No | No |
| pay(creditCard) | This lets the user pay the transaction using the chosen card | No | No |

Methods:

ready():

If the user is ready to start the transaction to pay

Return True

chooseType():

Switch:

Groceries;

Gas;

Food;

Travel;

Hotel;

Online;

Assign the chosen type to type

AICard():

Run AI to get credit card

Assign creditCard to the card chosen by AI and return

pay(creditCard):

Tap the chosen creditCard

Use the card to use the tap feature on the mobile device

If transaction was successful:

Return True

Else

Return False

### **User Account:** Class Diagram

Attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Read Only** | **Multiplicity** |
| name | The name of the user | Yes | 1 |
| id | Every user has a distinct id | Yes | 1 |
| email | The email of the user | No | 1 |
| password | The password of the account | No | 1 |
| phone | The phone number of the user | No | 1 |
| cards | The credit cards of the user | No | 1 |

Operations:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Is Query** | **Is Polymorphic** |
| register() | This allows a user to register for an account | No | No |
| login() | This allows a user to login to their account | No | No |
| updateProfile() | This allows the user to update some aspects of their profile | No | No |

Methods:

register():

The user inputs name, email, password, and phone number

If the email and phone number aren’t already being used`

Return True

Else

Return False

login():

The user inputs their email and password

If the email and password match in the database:

Return true

updateProfile():

The user can update their email, password, and phone number

If the user changes one of these items:

Return true

Else

Return False

# **Architecture Design**

## **Architecture Overview**

The following diagrams are a visual representation of the networks and physical devices and servers that are happening. The main coding environment will consist of MacBooks for the developers and their own server to update the application from the office. There will be a massive database in order to store all user’s information.

## **Infrastructure Model**

* + 1. **Deployment Diagram 1 – Architecture Overview**

A diagram of a computer network

Description automatically generated

* + 1. **Deployment Diagram 2 – Nodes and Artifacts**

**A diagram of a computer network

Description automatically generated**

## **Hardware and Software Requirements**

* + 1. **Hardware Components**

A piece of hardware that all software engineers will need is a MacBook. Because this app will only be available on IOS, so the developers will need a MacOS environment to be able to code this app for the App Store. Another piece of hardware needed is a server. We need a physical server in order to be able to update the app and be able to quickly shut down the app in case of a data breach, so users don’t get their data stolen.

* + 1. **Required Software Components**

Software needed for the developers is Microsoft’s Visual Studio Code. This is the most ideal IDE, and the developers should be programming the app in this IDE. Another thing the developers need is to have Swift installed. IOS apps are easier to code when using the coding language of Swift. Another piece of software needed is a database. A database is needed in order for the user’s data to be stored and kept securely.

## **Security Plan**

* + 1. **Security Overview**

The top security threats for this app are external and internal intruders. To prevent these intruders, we need to make sure our system cannot be hacked into. We need to have a strong password software to make sure user’s passwords cannot be decoded. We also need a firewall to prevent easy access into our system. We also need all employees to have an ID card so they can get into the building, and outsiders cannot. We also need to have an access notification system in order to get notified when critical information is being accessed.

* + 1. **Security Plan**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Disruption, Destruction, Disaster | | | | | Unauthorized Access | | | | |
| Fire | Flood | Power Loss | Circuit Failure | Virus | External Intruder | Internal Intruder | | Eaves-dropping | |
| Servers | 1, 3 | 1, 4 | 5 | 5 | 6 | 7, 8, 9 | | 10 | |  |
| Database | 1, 2, 3 | 1, 2, 4 | 5 | 5 | 2, 6 | 7, 8, 9 | | 10 | |  |
| Network | 1, 3 | 1, 4 | 5 | 5 | 6 | 7, 8, 9 | | 10 | |  |
| Deployment Computers | 1, 3 | 1, 4 | 5 | 5 | 6 | 7, 8, 9 | | 10 | |  |

Controls:

1. Disaster Recovery Plan
2. Backup database server
3. Fire Prevention System in Otto Miller in computer rooms; sprinklers in rest of building
4. The host computer room is on the second floor of Otto Miller
5. Uninterruptible Power Supply (UPS) on SPU's primary network servers
6. Virus checking software present on SPU’s network
7. Strong password software
8. Employee ID for access into the building
9. Firewall
10. Access notification system

# **User-Interface**

## **User-Interface Requirements and Constraints**

The next diagram is a visual representation of what the app will look like, and it will visualize what pages connect to each other. The drawings after the diagram are what each page will look like. Each UI page isn’t perfect, but is close to a perfect representation of each page.

Provide a written introduction to this section, *including any guiding principles or constraints on the user interface.* You can adapt some lecture material for this, but avoid sounding like a lecture or textbook (make it your own words). This section will also describe what the reader will see on the following pages. ***If this section explains each of the sections that follow well enough, they will not need introductions of their own.***

## **Window/Screen Navigation Diagram**

A diagram of a website

Description automatically generated

## **UI Wireframes**

A white paper with blue writing

Description automatically generated

A screen shot of a computer screen

Description automatically generatedA white paper with blue writing on it

Description automatically generated

A white paper with blue writing

Description automatically generated

A screen shot of a white sheet

Description automatically generated

A white board with blue writing

Description automatically generated

A white paper with blue writing

Description automatically generatedA blue pen drawing on a white paper

Description automatically generated

A paper with blue writing on it

Description automatically generated

## **Reports: “Formal Output” Design**

N/A

# **Appendices**

## **Glossary**

Swift: A coding language

## **References / Bibliography**

*Intelligent Diagramming | LucidChart*. (n.d.). Lucidchart. <https://www.lucidchart.com/pages/landing?utm_source=google&utm_medium=cpc&utm_campaign=_chart_en_us_mixed_search_brand_exact_&km_CPC_CampaignId=1457964857&km_CPC_AdGroupID=57044764032&km_CPC_Keyword=lucid%20chart&km_CPC_MatchType=e&km_CPC_ExtensionID=&km_CPC_Network=g&km_CPC_AdPosition=&km_CPC_Creative=442433231228&km_CPC_TargetID=kwd-55720648523&km_CPC_Country=9033307&km_CPC_Device=c&km_CPC_placement=&km_CPC_target=&gad_source=1&gclid=Cj0KCQjwjLGyBhCYARIsAPqTz1-kfuZfwS7hLsmd6oYlfLDmzirbw0YwoSVuRXM7Gn000xvVTrA8yhQaAu7gEALw_wcB>

Class Lecture Notes

## **Supporting documentation**

A close up of a computer

Description automatically generated