CSc-230

System Software Engineering

Instructor: Doan Nguyen, Ph.D

Spring 2015

Deliverable#2: Software Requirement Specification

Team Name: Clear vision

Avani Rajopadhye

Juan Garcia

Due date: 03/21/15

1. Customer Statement and Requirements

Goals: Our goal for this project is to design a web application that will allow a user to enter his/her monthly credit card information, such as credit card title, interest paid (APR), outstanding balance to track the remaining debt and monthly payments made by that card. The users will then be able to generate reports on the data that they have entered, which they can use to help in managing credit card debt.

Problem Statement: Currently, if users want to keep track of their credit card debts, they have to log in to each individual credit card in order to retrieve information. Because each credit card has different information (different APR, payment amounts and remaining balance), it is difficult for the user to identify trends that can help them make better financial decisions. It is very difficult to keep track of all of this information.

Proposed Solution: Our application will help users to manage their credit card data at ONE place, for which in current scenarios, users need to look upon each credit card website to see every fundamental and required details associated with that credit card.

We are designing a proposed solution to this existing problem through our web application, where first time users have to login into our system. Then they will be allowed to enter the initial data. For existing users, our application provides more facilities such as adding new data along with the ability to see previously added data. The advantage of having all the credit card information in ONE place is that users can not only add their credit card data and update it but can also see reports generated by our system based on the information provided by them.

2. Functional Requirements

Identifier	Priority	Description
REQ_1	1	The system shall be able to register a new user of the web application.
REQ_2	5	The system shall have the ability to store credit card information
REQ_3	5	The system shall be able to append credit card information into th existing document without losing prior information.
REQ_4	5	The system shall be able to read previous credit card information from the document.

REQ_5	4	The system shall be able to generate a report using the stored cred card information. The user will specify one credit for which a report will be generated.
REQ_6	4	The system shall be able to generate a report using the stored cred card information. The user will specify a month for which a repor will be generated.
REQ_7	2	The system shall be able to generate a report using the stored cred card information. The user will specify that they will like a summary report, which will summarize the information for all credit cards and all months

In order for our application to be successful, our most important requirement is for the system to be able to store the information that the user has entered and retrieve that information (REQ_2, REQ_3 and REQ_4). This is why we have made these requirements to be the highest priority, since they are the basic building blocks of our application.

3. Nonfunctional Requirements

Identifier	Priority	Description
REQ_8	3	Affordability: A new user shall be able to add a credit card and its information within 5 minutes without the need for training
REQ_9	5	The system shall provide the user with feedback when a report request is made, informing the user when the report is generated and how to access this report.
REQ_10	2	Performance: Generating a new report takes less than 3 seconds

Our non-functional requirements stem from creating a positive user experience. They are highly tied with the User Interface, since this is what the user will be interacting with. In terms of priorities, our top priority is REQ_9, which is to ensure that users receive feedback from the system, letting them know when the system is doing work, when it is done generating the report, and how to access the information of the report.

In our requirements, we have not added security for two reasons. The first reason is that we are not asking users to enter credit card numbers or sensitive credit card information. The information that users will be entering will be limited to the balance, monthly payments and the APR. The second reason that we have not included security in our requirements is that at this moment, we are considering using google drive Rest API to store the data in the user's google drive account. Because of this, we are relying on security mechanisms provided by google drive REST API.

4. System Diagram

For our system architecture, we are following a widely used pattern for web application: the Model View Controller (MVC). In this architecture, the system is observed as having 3 major parts. The View is the user interface, which is the part of the software that users will be interacting with. In our case, this will be our web UI, which will be the part that users interact with. The controller contains the application logic, which in our case, refers to taking appropriate actions after a user request. A user request could be to generate a report, or add credit card information. Finally, the model handles the data, which in our case will involve the use of REST API in order to read and write data. Figure 1 contains our system architecture.

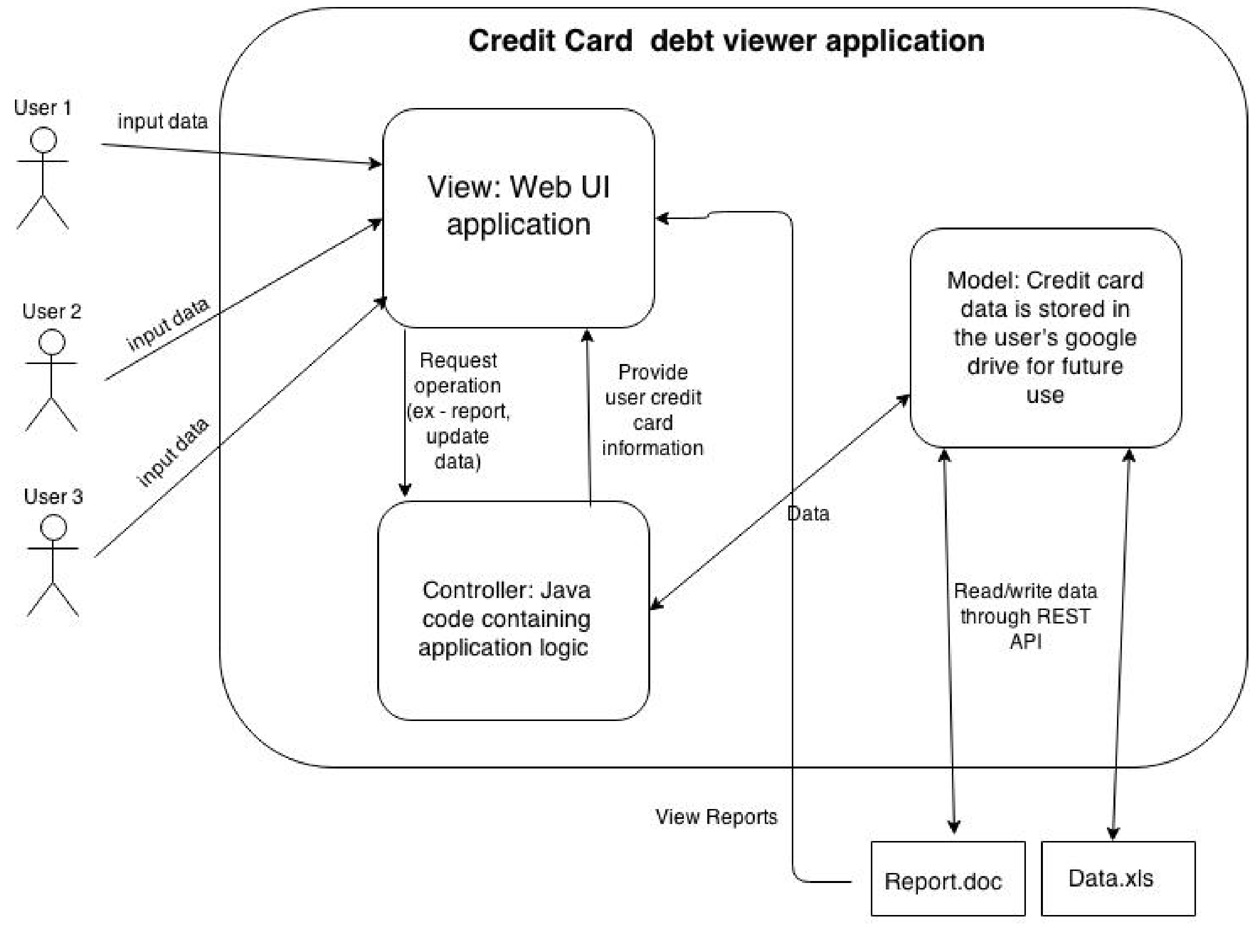
Credit Card debt viewer application
User 1
User 2
User 3
input data
input data
input data
View: Web UI application
Request operation (ex - report, update data)
Provide user credit card information
Controller: Java code containing application logic
Data
Model: Credit card data is stored in the user's google drive for future use
Read/write data through REST API
View Reports
Report.doc
Data.xls

Figure 1: System architecture (MVC pattern)

5. State Diagram

In terms of our state diagram, figure 2 below gives an idea of the possible paths that a user can take as they interact with our system:

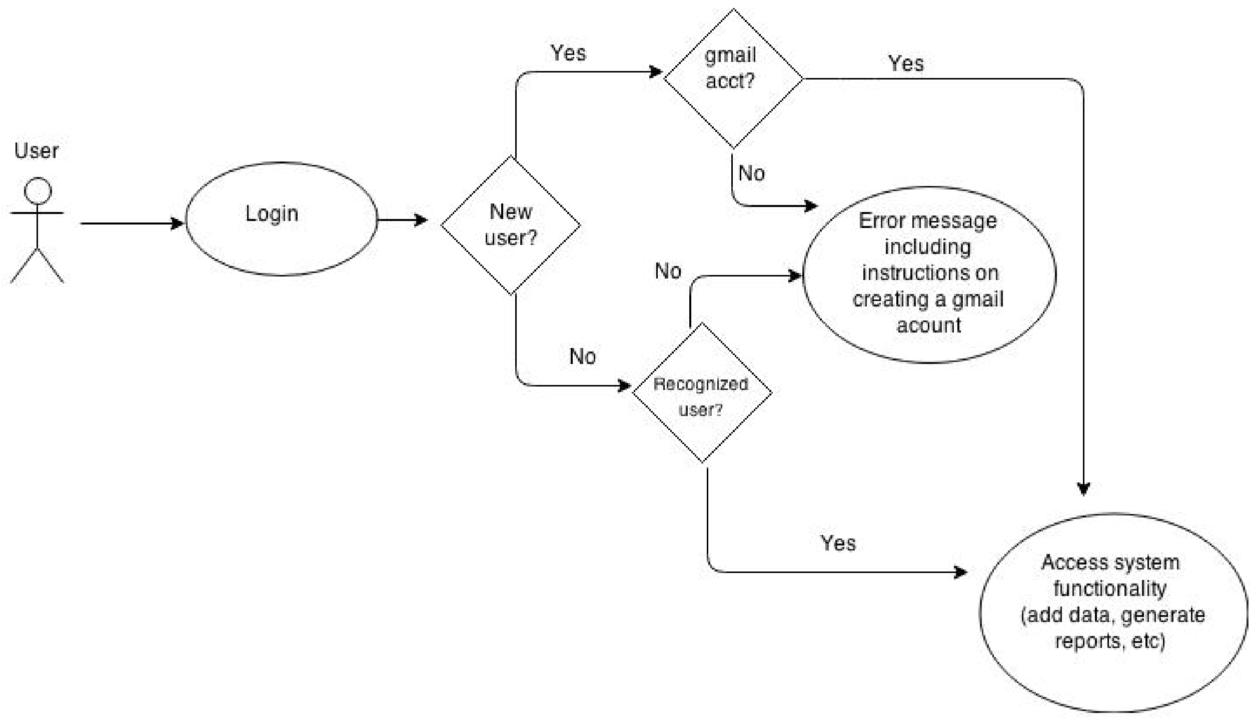
User
Login
New user?
Yes
gmail acct?
Yes
No
Error message including instructions on creating a gmail acount
No
No
Recognized user?
Yes
Access system functionality (add data, generate reports, etc)

Figure 2: State Diagram

6. Functional Requirement Specification

Stakeholders: Because of the nature of this project (being a class assignment for CSC230), the main stakeholders are the developers: Juan Garcia and Avani Rajopadhye. Another stakeholder is our professor, Doan Nguyen, who will be responsible for assessing our application. In this context, professor Nguyen and our classmates are the customers, since we will be demonstrating our application to them during our presentation.

Actors: During our use cases, we have used two types of users: New users and existing users. A new users are someone who are new to our application. They have never used the application before, and will be starting by going through the registration phase and familiarizing with the capabilities of our application. An existing user on the other hand, refers to a person who has used our application before and is logging in in order to add new data or generate reports based on existing information.

Use Cases: Below are our use cases. These use cases are generated from our product backlog, which is a mechanism used in Agile methodology that help us track our progress. The first figure is the use case diagram, which gives the reader a summary of our use cases. Then, we have a table with more details on what each use case entails.

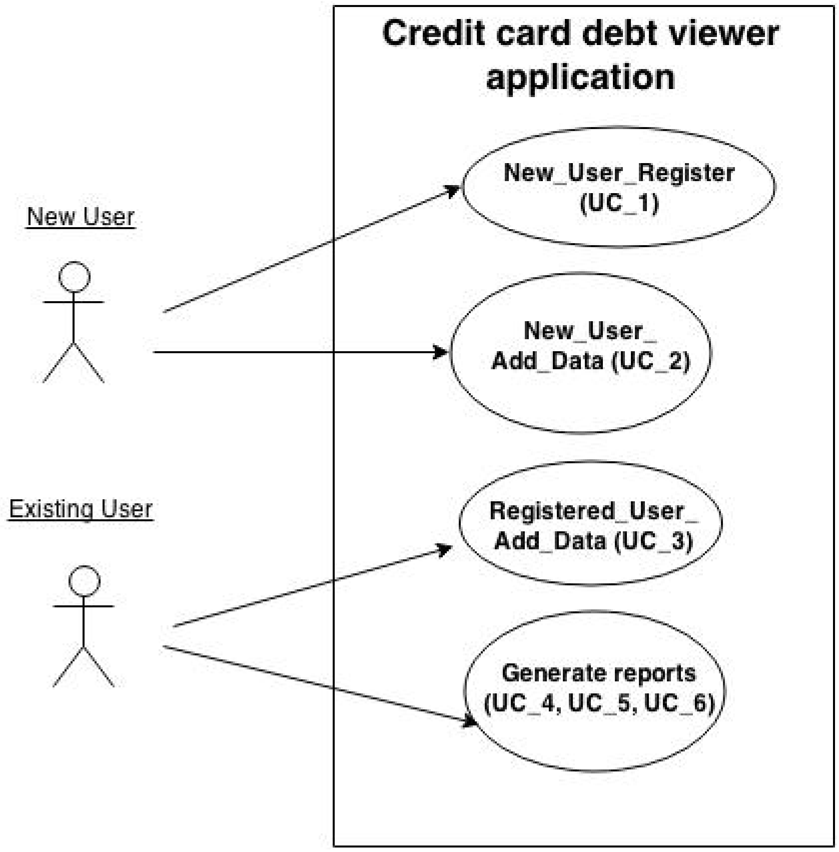
Credit card debt viewer application
New User
Existing User
New_User_Register (UC_1)
New_User_ Add_Data (UC_2)
Registered_User_ Add_Data (UC_3)
Generate reports (UC_4, UC_5, UC_6)

Figure 3: Use case diagram

Use Case #	**Use case name**	**Use case details**
UC_1	New_User_Register	**<Actors>** New_User_1, New_User_2 **<Goals>** Register user to use the system **<Pre-condition>** User must have a gmail account. If the user does no

		have one, they can follow instructions from the system on how to create one. **<Post-condition>** After registering, user can access the system functionality **<Description>** As a new user, I can register to use the Credit Card Debt Viewer application **<Sequencing steps>** 1. New user opens our web application 2a. Application displays google account login information, asking users to enter gmail username an password 2b. If the user was already logged in to their gmail account, application jumps to the consent screen (Ste 4) 3a. User enters gmail account information 3b. If the user does not have a gmail account, they will need to create one before accessing the application. Directions on how to create a gmail account will be provided to the user. 4. Application displays google app consent screen, letting the user know that the application will be accessing the google drive documents. 5a. User accepts the consent screen and begins using the application. 5b. User declines the consent screen and is not able t use the application features.
UC_2	New_User_Add_Data	**<Actors>** New_User_1, New_User_2 **<Goals>** user adds credit card data to the system **<Pre-condition>** User must have a gmail account. If the user does no have one, they can follow instructions from the system on how to create one. User has gone throug UC_1 and has accepted the consent screen (see use case 1 for more information) **<Post-condition>** The added data is stored in the system. The user can see this data. **<Description>** As a new user, I can add credit card information for the first time. Credit card information includes: name of credit card, current APR, interest paid and outstanding balance. Once the user adds this

		information, the application stores the data for future use **<Sequencing steps>** 1. User selects the add a new credit card option in the application 2. System displays empty fields where user can enter data 3. User enters the name of the credit, current APR an remaining balance. 4. System stores the data for future use. System informs the user that the data has been stored.
UC_3	Existing_User_Add_Data	**<Actors>** Existing_User_1, Existing_User_2 **<Goals>** Registered user can add data **<Pre-condition>** User has registered with the system (has gone throug UC1 and has accepted the consent screen) **<Post-condition>** The added data is stored in the system. The user can see this data. **<Description>** As an existing user, I want to be able to enter new credit card data into the system. This could be a payment made and the remaining balance. The application allow user to enter the necessary data an stores it for future use **<Sequencing steps>** 1. User selects an existing credit card to add payment information 2. System displays the screen where the user can add payment information 3a. User enters the month of payment, amount and remaining balance. All 3 of these values must be entered in order for the system to proceed. The APR entry is optional, if the user does not enter a value fo this, the system uses the previously entered value for APR. 3b. Alternatively, user can enter the date of payment, amount, APR and remaining balance. In this case, th system will append all four values to the existing data that is maintained for that user for a particular credit card. 4. System stores the data for future use. System informs the user that the data has been stored.

UC_4	Registered_User_Generate_Report_by_month	**<Actors>** Registered_User_1, Registered_User_2 **<Goals>** Registered user requests a report by month and is abl to see this report **<Pre-condition>** User has registered with the system (UC_1). User ha previously added data to generate report (UC_3). If there is not data, the user will be informed to add dat prior to requesting a report **<Post-condition>** The user is able to see the report generated by month System will provide feedback, letting user know when the report has been generated and how to acces it. **<Description>** As an existing user, I can generate a report using the stored data by selecting a particular month. The application will generate a report with all of the credi cards for that month, any monthly payments made, and the remaining balance. **<Sequencing steps>** 1. User selects a month and the option of seeing a report for that month. 2. System checks to see if there is data for that month 3a. If there is no data for the given month, system informs the user that there is no available data for tha month and suggests to add more data, select a different month or try running another report. 3b. If there is data for that month, the system inform the user that it is generating a report. The system the gathers the necessary data and places the data in a report. The system will inform the user when the report is done and how to access it.
UC_5	Registered_User_Generate_Report_by_creditcard	**<Actors>** Registered_User_1, Registered_User_2 **<Goals>** Registered user requests a report by credit card and i able to see this report **<Pre-condition>** User has registered with the system (UC_1). User ha previously added data to generate report(UC_3). If there is not data, the user will be informed to add dat prior to requesting a report

		<Post-condition> The user is able to see the report generated by credit card. System will provide feedback, letting user know when the report has been generated and how t access it. **<Description>** As an existing user I can generate a report using the stored data by selecting a credit card. The applicatio will generate a report with the information from all months of the specified credit card. **<Sequencing steps>** 1. User selects a credit card and the option of seeing a report for that card. 2. The system informs the user that it is generating a report. 3. The system then gathers the necessary data and places the data in a report. The system will inform the user when the report is done and how to access it.
UC_6	Registered_User_Generate_Report _by_summary	**<Actors>** Registered_User_1, Registered_User_2 **<Goals>** Registered user requests a report by overall summary and is able to see this report **<Pre-condition>** User has registered with the system(UC_1). User has previously added data to generate report(UC_3). If there is not data, the user will be informed to add data prior to requesting a report **<Post-condition>** The user is able to see the report generated by overal summary. System will provide feedback, letting user know when the report has been generated and how t access it. **<Description>** As an existing user, I can generate a report on my credit card summary. The application will generate a report with all the credit cards, including the current APR and remaining balance. **<Sequencing steps>** 1. User selects the option of seeing a summary report of all credit cards. 2. System checks to see if there is data that was previously entered. 3a. If there is no data, system informs the user that there is no available and suggests to add data before attempting to run a report

		3b. If there is credit card data, the system informs the user that it is generating a report. The system then gathers the necessary data and places the data in a report. The system will inform the user when the report is done and how to access it.

7. Traceability matrix

Below is a traceability matrix, which shows the the relationship between the use cases and the requirements:

	UC_1: New_User_Register	**UC_2:**New_User_Add_Data	**UC_3:** Registered_User_Add_Data	**UC_4:** Registered_User_Generate_Report_by_month	**UC_5:** Registered_User_Generate_Report_by_creditcard	**UC_6**: Registered_User_Generate_Report_by_summary
REQ_1: Register New User	X		X	X	X	X
REQ_2: Ability to store data	X		X			
REQ_3: Append Data	X	X	X			
REQ_4: Read Stored Data				X	X	X
REQ_5: Generate Report on the basis of Credit card name			X	X	X	X
REQ_6: Generate Report on the basis of month			X	X	X	X
REQ_7: Generate			X	X	X	X

Report on the basis of summary						
REQ_8: NFR - UI Affordance	X	X	X	X	X	X
REQ_9: NFR - Ease of Use	X	X	X	X	X	X
REQ_10: NFR - Performance	X	X	X	X	X	X

8. Product Backlog

Since our team is following scrum and agile product development methods, we have created a product backlog. This product backlog will aid us in determining how much work we can take on during each iteration, how much effort each task requires, and how to delegate our work. Below is a snapshot of this evolving backlog:

User Story ID	User Story Description	Tasks ID	Priority	Owner	Task Description of each User Story	Estimated Effort	Completed Story?	Remaining Effort after Iteration 0
SPIKE_1	**Get Hands-on experience on File Handling**	TSK_1A	Medium	Avani	Hands-on experience with File (.txt) Read/Write (Handling) of String in Java	1	YES	0
		TSK_1B	Medium	Avani	Hands-on experience with File (.txt) Read/Write (Handling) of Integers in Java	1	YES	0
		TSK_1C	Medium	Avani	Hands-on experience with	1	YES	0

					Reading of Pipe delimited .txt file in Java			
SPIKE_2	**Set up environment**	TSK_2A	High	Avani Juan	Setting up Google Drive environment in Eclipse Kepler	2	YES	0
		TSK_2B	High	Avani Juan	Run through the steps of the drive API quick start "Hello World" application which creates a simple text document and writes "Hello World" into this document.	2	YES	0
		TSK_2C	High		Creates a blank google document to store the credit card data. A spreadsheet would be preferred	2		
		TSK_2D	High	Juan	Can read and write into a google doc spread sheet using the google drive API	3		
		TSK_2E	Medium	Avani	Google doc authentication: understand how OAuth2 works	3	YES	0

					in order to authenticate user and have access to the google documents			
		TSK_2F	Medium	Juan Avani	Common repository is identified in order to collaborate during the project (most likely - github)	2	YES	0
SPIKE_3	**Software Requirement Specification document is completed (to be submitted by 3/9)**	TSK_3A	High	Juan	part 1) Goal/problem statement/proposed solution	1	YES	0
		TSK_3B	High	Juan	Part 2/3) list of functional and non functional requirements. Ensure that requirements are verifiable (see lecture slides for other 2 criteria)	1	YES	0
		TSK_3C	High	Avani Juan	part 4) System diagram. Identify software to create UML diagrams	2	YES	0

		TSK_3D	High	Avani	Part 5) functional requirement specifications	1	YES	0
		TSK_3E	High	Avani	Part 6) Traceability Matrix	2	YES	0
US_1	**New user registers (Possibly creating a new user name and password, although this decision will come later) to use the credit card tracking application**	TSK_US_1A	Low		Web UI for registration/log in is defined (through sketches)	1		
		TSK_US_1B	Low		Web UI for registration/log in is implemented. This web page is simple to use and aesthetically pleasing	3		
		TSK_US_1C	Low		Application can authenticate the user's google doc account using OAuth 2.0 protocol	3		

		TSK_US_1D	Low		Define how user information will be stored, if we decide the users to create log in information such as a password and username	4		
		TSK_US_1E	Low		Test cases for this user story	2		
US_2	**New user adds credit card information for the first time. User informs the application which credit cards they will like to keep track of, the current APR and the remaining balance**	TSK_US_2A	Medium		Web UI for adding a new credit card is defined	3		
		TSK_US_2B	Low		Web UI for adding a new credit card is implemented. it is is simple to use and	3		

					esthetically pleasing			
		TSK_US_2C	Medium		Application can create a blank text document using the Rest API (which could be used to store the credit card information)	3		
		TSK_US_2D	High		Application can create a blank spreadsheet using REST API (which can be used to store the credit card information)	3		
		TSK_US_2E	High		The format of the spreadsheet is selected, which determines how the information will be stored	2		
		TSK_US_2F	High		Application collects the information from the user. This is done through the UI, of if we decide to work on the UI last, then through another means, such as using the command line	3		

					(for prototype purposes)			
		TSK_US_2G	High		After application collects information, it is able to write this information on the document using the format that we have selected to store the credit card information	3		
		TSK_US_2H	High		Test cases for this user story	3		
	As an existing user, I want to be able to enter new credit card data into the system. This could be a payment made, remaining balance or an update	TSK_US	Mediu		Web UI for adding new data is defined			

					aesthetically pleasing			
		TSK_US_3C	Low		Application verifies that there is an existing document with the credit card information	3		
		TSK_US_3D	Low		If there is no existing document, application directs user to enter new credit card information. See use case of adding credit card information for the first time	2		
		TSK_US_3E	High		Application reads the existing credit card data from the document. It uses this data to display options to the user. For example, if the user has previously entered information on a visa credit card and a master card credit card, then the application	3		

					displays these two possible credit card options to enter data for			
		TSK_US_3F	High		User is able to enter new credit card data (APR, Remaining balance, payment made). Application collects this data	3		
		TSK_US_3G	High		Using the collected data, application writes the new data into the document (without overwriting previous data, unless the APR has been updated)	3		
		TSK_US_3H	High		Test cases for this user story	3		
US_4	**User generates a report using the stored data by selecting the month**	TSK_US_4A	Medium		Web UI for running a report by month is defined through sketches	1		
		TSK_US_4B	Medium		Web UI for running a report	3		

					by month is implemented			
		TSK_US _4C	Mediu m		Format of the report is defined	1		
		TSK_US _4D	Mediu m		User can select the month	1		
		TSK_US _4E	High		Application collects the report information (month and report type) and is able to read the google document and collect the necessary data	3		
		TSK_US _4F	High		Application generates the data for the report	2		
		TSK_US _4G	High		Application writes the data of the report on a new google document	2		
		TSK_US _4H	High		Test cases for this user story	2		
US_5	**User generates a report using the stored data by selecting a credit card**	TSK_US _5A	Mediu m		Web UI for running a report by credit card is defined through sketches	1		
		TSK_US _5B	Mediu m		Web UI for running a report	3		

					by credit card is implemented			
		TSK_US _5C	Mediu m		Format of the report is defined	1		
		TSK_US _5D	Mediu m		User can select the credit card for report	1		
		TSK_US _5F	High		Application collects the report information (report type and credit card) and is able to read the google document and collect the necessary data	3		
		TSK_US _5G	High		Application generates the data for the report	2		
		TSK_US _5H	High		Application writes the data of the report on a new google document	2		
		TSK_US _5I	High		Test cases for this user story	2		
US_6	**User generates a report on their credit summary**	TSK_US _6A	Mediu m		Web UI for running a summary report is defined through sketches	1		
		TSK_US _6B	Mediu m		Web UI for running a report	3		

					summary is implemented			
		TSK_US_6C	Medium		Format of the report is defined	1		
		TSK_US_6D	Medium		User can select the summary report	1		
		TSK_US_6E	High		Application collects the report type and is able to read the google document and collect the necessary data	3		
		TSK_US_6F	High		Application generates the data for the report	2		
		TSK_US_6G	High		Application writes the data of the report on a new google document	2		
		TSK_US_6H			Test cases for this user story	2		