iGo TVM

Andrew Jreij
Simarpreet Kaur Jabbal
A M Anisul Huq
Chinmay Hadadare
Rajasekhar reddy Guntaka

Team D
Deliverable 2
November 27 2019

Contents

1 Personas

According to ISO/IEC/IEEE, 2012, a Persona is [An] archetypical user of a [software] system, based on research into real users of a [software] system. The notion of a persona has its origins in user-centered, goal-directed, approach to interaction design. In order to have archetypical users instead of stereotypical users, our personas were created based on interviews. Our personas included the following information:

- Photograph: The purpose of the photograph is to humanize a user, in general [Martin, Hanington, 2012, Page 132], and to make it memorable, in particular [Nieters, Ivaturi, Ahmed, 2007; Richter, Flückiger, 2014, Chapter 3]. In doing so, a photograph is supposed to act as an icon [Marcus, 2003], not as an identifier.
- Name: having a name makes it easier to refer to this persona when discussing requirements during meetings.
- Job Title/education: it is important that the system curates to all levels of education.
- Background: helps with extrospection when writing user stories.
- Technical Background: it is important to know the user's technical capabilities in order to build a system that is intuitive and easy to use by the least technical people.
- Everyday Life: knowing the frequency and type of usage helps build a system that is curated for the user.
- Tasks and Goals: we asked the users what are their needs when using the system.
- Pain points: the users provided the most frustrating parts of using the system and suggested solutions in some cases.

Persona01_Fouad	Fouad, 22 Computer Arts student Lives with a roommate in Montreal, Canada
Background	Fouad used to be a team lead in an events management company. He then left his home country to come to Montreal in order to learn Computer Graphics and Art. He spends most of his free time doing the following: Practicing art Reading fiction Listening to music Browsing social media
Technical Background	His computer usage was limited to everyday software such as Microsoft Word and Excel. He only now started using more complex software for 3D and 2D art, such as Autodesk Maya and Adobe Photoshop. He's not a very technical person when it comes to computers. Solving minor computer issues could be hard for him to do.
Everyday Life	Fouad takes the metro on a daily basis to go to school. He recharges his metrocard on a monthly basis.
Tasks and Goals	As a TVM user, Fouad wishes to get his card recharged as quickly as possible.
Pain points and Frustrations	Having to recharge the card monthly(instead of seasonal for example). Waiting in line because completing a transaction takes 3-4 minutes per person.

Table 1: Persona01 Description

Persona02_Navpreet Kaur	Navpreet Kaur, 29 QA Analyst Lives in Montreal
Background	Navpreet is from India and she came to Canada a few years ago for higher studies. She completed her Masters from a university in Toronto and later she moved to Montreal. In her free time Navpreet likes to cook and travel.
Technical Background	Navpreet is a highly technical person, since she is a QA Analyst. She has approximately 3+ years in using software tools like Selenium, Webdriver, TestNG , Postman, Specflow, JIRA etc.
Everyday Life	Navpreet goes to work using her own car, but prefers the metro during winter. She also uses the metro frequently when going out to explore the city.
Tasks and Goals	As a TVM user, Navpreet wishes to purchase tickets from Ticket Vending Machine using Cash or Debit/Credit.
Pain points and Frustrations	Long wait times in line. Non availability of any electronic proof of payment while buying a ticket as in few instances she lost her ticket and paper receipt and had to buy a new one.

Table 2: Persona02 Description

Persona03_Shashank	Shashank, 25 Walmart Employee Lives with a roommate in Montreal, Canada
Background	Shashank left his home country for higher education and came to Montreal, Canada. Currently working at Walmart as Assistant Manager. He spends most of his free time with activities such as: Playing Computer games Reading and exploring about new gadgets Browsing social media
Technical Background	He has a bachelors of engineering in Computer Science, making him technically sound person. In addition to that, his passion for gaming and gadgets has made him well aware of computer hardware as well.
Everyday Life	Shashank takes the metro on a daily basis to go about work and anywhere else as well. He recharges his metrocard on a monthly basis.
Tasks and Goals	As a TVM user, Shashank wishes to get his card recharged online as well.
Pain points and Frustrations	Having to visit the TVM in person to recharge the card. Not having option of payment receipt sent to his email.

Table 3: Persona03 Description

Persona04_Vikranth	Vikranth Age 24 Lives with a roommate in Montreal, Canada
Background	Vikranth left his country to Montreal for his higher studies. He completed his Master's in civil engineering at Concordia University. He spends most of his free time doing activities such as: Cooking Reading about and exploring new gadgets Exploring new places
Technical Background	Vikranth used to be the top of his class. He's very dexterous and like to tinker with electronics. He is currently researching tools related to Cost Estimation.
Everyday Life	Vikranth takes the metro on a daily basis for all purposes. He recharges his metrocard on a monthly basis
Tasks and Goals	As a TVM user, Vikranth wants to recharge his card using Cash or Debit/Credit.
Pain points and Frustrations	Vikranth's major concerns: Long queue for recharging the card at the beginning of every month. No support for electronic receipt.

Table 4: Persona04 Description

Persona_05 Samia	Samia, 55 Graduate teaching assistant (TA) Lives with her family in Montreal, Canada
Background	Samia has an undergraduate degree in computer science from McGill University and a graduate degree in software engineering from Concordia University, Canada. She started her career as a Software Developer/User Interface (UI) specialist right out of university and later was promoted to the position of team lead in human engineering. After more than 18 years working in software development, she switched to teaching career. As a TA, she prepares, teaches weekly tutorials and demonstrates labs for various graduate level courses. Being an easy going person, Samia likes to read and travel in her spare time.
Technical Background	Samia is tech savvy. She effortlessly utilizes different complicated software both personally and professionally. As a UI specialist, she has expertise in tools and libraries, such as jQuery, Angular.js and Sass. As a TA, she teaches different tools based on scientific Linux.
Everyday Life	As Samia lives in the outskirts of Montreal, she has to take the metro on a daily basis.
Tasks and Goals	As a TVM user, Samia wishes to get her metro card recharged on a monthly basis using either cash or debit/credit card.
Pain points and Frustrations	Samia's main concerns were: The TVM only accepts currency (bank notes and coins) of very few/limited denominations. The level of security provided by TVM machines whilst using debit/credit card for metro card recharge. Electronic receipt for purchases is not yet provided. Long waiting time in queues at the beginning of each month as each cash transaction takes several minutes to complete.

Table 5: Persona05 Description

1.1 Photographs

The following photographs are not the interviewees but are merely used for representation and in order to humanize the user.

Figure 1: Persona01_Fouad

Figure 2: Persona02_Navpreet Kaur

Figure 3: Persona03_Shashank

Figure 4: Persona04_Vakranth

Figure 5: Persona05_Samia

2 User Stories

User stories have one level of description, represented using natural languages. There is a single role. There is no notion of secondary actor. They are based on conversation with a customer/user, specifically through interviews. The user stories are based on the use cases but in a smaller scope, one comparable to a scenario. The user stories were written following the I.N.V.E.S.T[6, 3] criteria, meaning the following:

1. Independent: user stories should be as far as possible independent so each of them could be developed and delivered separately.
2. Negotiable: user stories are not explicit contracts and should leave space for discussion.
3. Valuable: user stories should deliver value to the stakeholders.
4. Estimatable: you must be able to estimate the size of a user story
5. Small: user stories should not be so big as to become impossible to plan/task/prioritize within a level of accuracy.
6. Testable: user stories should have acceptance criteria to test if they fulfill customer's needs.

We adopted this criteria in our user stories by doing the following:

1. Independent: every user story is completely independent from the other as they are based on different scenarios, except for USIGO03, where it comes after every other user story.
2. Negotiable: payment methods as well as receipt generation are both prone to change.
3. Valuable: the main goal of a TVM is to provide tickets, and all the user stories are based on that exact scenario. They are also linked to the user's demands from the interviews: such as requesting e-mail receipts.
4. Estimatable: the user stories were estimated based on story points, explained further below.
5. Small: by making sure the user stories are independent and atomic, it is easy to plan and prioritize them with a high level of accuracy.
6. Testable: we provided acceptance criteria for each user story that cover the essential scenarios to make sure they fulfill the user's needs.

The user stories took the form of [Role][Goal][Value]. With the role being a specific persona based on the previously conducted interviews. The [Goals] were created following the SMART criteria[7, 2]:

1. Specific: the goal should describe a concrete statement, such as a mission statement.

2. Measurable: there should be metrics in place to determine if the goal is met. This makes a goal more tangible because it provides a way to measure progress.

3. Achievable: This focuses on how important a goal is to you and what you can do to make it attainable and may require developing new skills and changing attitudes.

4. Relevant: similar to the Valuable criteria in INVEST, the goal should provide value to the user.

5. Time bound: Providing a target date for deliverables is imperative. A goal should be completed within a specific time budget.

We adopted this criteria in our goals specifically by doing the following:

1. Specific: each user presents a different scenario, even if the end goal is purchasing a ticket or refilling a card. The method of payment was used to divide our goals into more specific and atomic stories.

2. Measurable: measured along side the user stories using story points. Further explanation below.

3. Achievable: making sure the goals are specific made it easier to achieve them.

4. Relevant: all the goals are based on the main functionality of a TVM: providing access to the metro. As well as implementing email receipts which was requested by users in our interviews.

5. Time bound: initial proof of concept implementations were completed in a period of a week and a half.

Our definition of done includes the feature being fully functional along with its' tests. Estimation was done using story points. Story points rate the relative effort of work in a Fibonacci-like format: 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100. [5]

Statement	As [Fouad] I want to refill my metrocard as quickly as possible using my credit/debit card, to get access to the metro.
Maintainability	The system should be able to add support for any bank card.
Security	The system should secure the user's payment information.
Usability	The system should be intuitive.
Sustainability	The system should use the least energy resources possible. The system should have the option of email receipts to reduce the use of paper.
Priority	Being at the core of the TVM, this sets this functionality at a VERY HIGH priority.
Estimate	Implementing a simple static user interface where the user can choose whether they want to buy a ticket or refill their metrocard amounts to a 0.5. Implementing a simple static user interface where the user can choose their preferred payment method amounts to a 0.5. Implementing an expandable card payment service that can add support to any card amounts to a 40. Implementing a security system that doesn't compromise the user's payment information amounts to a 100. Implementing a communication between the iGo TVM and the bank amounts to a 100. Implementing a simple static user interface where the user can choose whether they want electronic receipts or paper receipts amounts to a 0.5. Implementing a static user interface that takes in user input (for email) amounts to a 3.

Table 6: USIGO01 - Refill metrocard using Debit/Credit cards - Andrew

ID	GIVEN	WHEN	THEN
01	A user inputs a Metrocard	The metro card is expired	The user cannot choose a refill option
02	A user inputs credit/debit card information	The information is valid and the metrocard is not expired	The payment is authorized
03	The system should be intuitive.	A user inputs credit/debit card information	The payment is not authorized
04	A user inputs debit card information	The user's account doesn't have enough balance and overcharging is not allowed and the metrocard is not expired	

Table 7: USIGO01 Acceptance Tests

Statement	As [Navpreet] I want to buy tickets from TVM using my credit/debit card, to get access to the metro.
Maintainability	The system should be able to add support for any bank card.
Security	The system should secure the user's payment information.
Usability	The system should be intuitive.
Sustainability	The system should use the least energy resources possible. The system should have the option of email receipts to reduce the use of paper.
Priority	Being at the core of the TVM, this sets this functionality at a VERY HIGH priority.
Estimate	Implementing a simple static user interface where the user can choose what type of ticket(one day, single trip,weekly,weekend ticket) amounts to a 0.5. Implementing a simple static user interface where the user can choose their preferred payment method amounts to a 0.5. Implementing an expandable card payment service that can add support to any card amounts to a 40. Implementing a security system that doesn't compromise the user's payment information amounts to a 100. Implementing a communication between the iGo TVM and the bank amounts to a 100. Implementing a simple static user interface where the user can choose whether they want electronic receipts or paper receipts amounts to a 0.5. Implementing a static user interface that takes in user input (for email) amounts to a 3.

Table 8: USIGO02 - Purchase tickets using Debit/Credit cards - Simarpreet

ID	GIVEN	WHEN	THEN
01	A user inputs credit/debit card	The information is valid	The payment is authorized
02	A user inputs credit/debit card information	The information isn't valid	The payment is not authorized
03	A user inputs debit card information	The user's account doesn't have enough balance and overcharging is not allowed .	The payment is not complete.

Table 9: USIGO02 Acceptance Tests

Statement	As [Shashank] I want to be able to select preference for receiving payment receipt, to keep track of my payments.
Maintainability	The system should be able to adopt any new method of providing receipt.
Security	The system should hide card number(if paid by card) except for the last 4 digits.
Usability	The system should be intuitive.
Sustainability	The system should use the least energy resources possible. The paper being used for printing should be acquired from recycled sources.
Priority	As this is an important part of process of TVM, this functionality has a HIGH priority.
Estimate	Implementation of a static user interface where the user can choose their preference of receiving a receipt of their payment amounts to a 0.5. Implementing a simple static user interface where the user enters their email ID amounts to a 1. Implementing printing of receipt with payment information amounts to an 8. Implementing a security that doesn't compromise the user's payment information amounts to a 40.

Table 10: USIGO03 - Select Payment Receipt Preference - Chinmay

ID	GIVEN	WHEN	THEN
01	A user inputs email ID	The email ID is invalid	User cannot proceed with email preference.
02	A user inputs email ID	The email ID is valid	User will receive the receipt on entered email ID.
03	A user selects paper based receipt option	The payment is authorized	The paper based successful transaction receipt is printed and dispensed.

Pre-condition: the payment is authorized.

Table 11: USIGO03 Acceptance Tests

Statement	As [Vikranth] I want to buy tickets from TVM using cash, to get access to the metro.
Maintainability	The system should be able to add support for new currency notes. In this case CAD
Security	The system should check for fake currency.
Usability	The system should be intuitive.
Sustainability	The system should use the least energy resources possible. The system should have the option of email receipts to reduce the use of paper.
Priority	Being at the core of the TVM, this sets this functionality at a VERY HIGH priority.
Estimate	Implementing a simple static user interface where the user can choose what type of ticket (single trip, two trips, one day, weekly, weekend ticket) amounts to a 0.5. Implementing a simple static user interface where the user can choose their preferred payment method amounts to a 0.5. Implementing an expandable cash payment service that can add support up to 80$ cash amounts to a 40. Implementing a simple static user interface where the user can choose whether they want electronic receipts or paper receipts amounts to a 0.5. Implementing a static user interface that takes in user input (for email) amounts to a 3.

Table 12: USIGO04 - Purchase tickets using Cash - Rajasekhar

ID	GIVEN	WHEN	THEN
01	A user inputs cash	The money is valid and the total is equal to or more than the amount.	The payment is completed
02	A user inputs cash	The money is fake	The payment is not completed
03	A user inputs cash	The total is less than the required amount	The payment is not completed

Table 13: USIGO04 Acceptance Tests

Statement	As [Samia] I want to refill my metro card as quickly as possible using cash so that I can access the public transportation services of Montreal (Société de transport de Montréal).
Maintainability	The system should be able to receive currency of any denomination (coins and bank notes).
Security	The system should be able to detect counterfeit bank notes and coins.
Usability	The user interface should be intuitive and "idiot-proof".
Sustainability	The system should provide email receipts option for every purchase in order to reduce paper use.
Priority	As a core TVM functionality, this has a VERY HIGH priority.
Estimate	Implementing a simple static user interface where the user can choose whether they want to use cash or debit/credit card to refill their metro card amounts to 0.5. Implementing a payment support service that can receive and return bank notes and coins of all denomination is 40. Implementing a security system that keeps user's payment information confidential amounts to a 100. Implementing a communication between the iGo TVM and the bank servers (of Société de transport de Montréal) amounts to 100. Implementing a simple static user interface where the user can choose whether they want electronic receipts or paper receipts amounts to a 0.5. Implementing a static user interface that takes in user input (for email) amounts to a 3.

Table 14: USIGO05 - Refill metro card using cash - Anisul

ID	GIVEN	WHEN	THEN
01	A user inputs a Metro card.	The metro card is expired.	The user cannot choose a refill option.
02	A user inputs the duration of the refill purchase (i.e. 15 days, 1 month, 2 months etc.)	The duration choice is correct and the metro card is not expired.	The user can now choose method of payment.
03	A user selects the option of paying by cash.	The duration choice is correct and the metro card is not expired.	The TVM receives the amount in appropriate denominations of bank notes and coins.
04	A user deposits cash for the purchase (refill).	The amount is not sufficient and the metro card is not expired.	The refill purchase is not authorized.
05	A user deposits cash for the purchase (refill).	The currency is counterfeit and the metro card is not expired.	The refill purchase is not authorized.

Table 15: USIGO05 Acceptance tests

User Story	Written By	Implemented By
USigo01	Andrew	Anisul
USigo02	Simarpreet	Andrew
USigo03	Chinmay	Simarpreet
USigo04	Rajasekhar	Chinmay
USigo05	Anisul	Rajasekhar

Table 16: User Story mapping

3 Backwards Traceability Matrix

User Story	Internal Source	External Source
USigo01 - Refill metrocard using Debit/Credit cards	UCigo02 UCigo05	STMpymt STMfares
USigo02 - Purchase tickets using Debit/Credit cards	UCigo03 UCigo05	STMpymt STMfares
USigo03 - Select Payment Receipt Preference	UCigo06 USig001 USigo02 USigo04 USigo05	STMpymt All interviews
USigo04 - Purchase tickets using Cash	UCigo03 UCigo05	STMpymt STMfares
USigo05 - Refill metrocard using Cash	UCigo02 UCigo05	STMpymt STMfares

Table 17: Backwards Traceability Matrix

- STM payment options (vending machines):
  http://www.stm.info/en/info/fares/points-sale
  Referred to as STMpymt in the traceability matrix
- STM fares options (vending machines):
  http://www.stm.info/en/info/fares/transit-fares
  Referred to as STMfares in the traceability matrix

Note that all interviewees mentioned wanting an email receipt option. Refer to Appendix A for the use cases' new naming scheme.

4 Implementation

4.1 USigo01

Written by Andrew, implemented by Anisul.

This implementation allows the user to refill metro card for a predefined set of durations using credit/debit cards:

- 2 Weeks - 40 CAD
- 1 Month - 80 CAD
- 2 Month - 150 CAD
- 3 Month - 210 CAD

Some of the durations mentioned here are not currently provided by STM at the moment. However, from our interviews it was apparent that such customizable refill durations are sought by customers of STM.

The implementation was done using Java 8 and Java Swing for the GUI.

The GUI class is responsible for creating the User Interface and handling any user input: clicks and text.

The main class starts by creating the following:

Refill Options Window: A method in the main class generates this window to provide users with the duration options. Bank card validity check window: Another method in the main class generates this window to take bank card number as user input. checkCardValidity: this object of type BankCard is responsible for performing the actual validity check.

Each button is manually created. Once a refill option button is clicked, the first window is repainted and a second window seeking bank card number appears. In the second window when user inputs the bank card number and presses submit, the checkValidity() method is invoked through the checkCardValidity object. This method will validate the card number using LUHN algorithm.

The result of the validity check appears in a pop-up window. If it's a valid card number then the program will also display the type of the card (i.e. VISA, MASTER etc.) based on the number.

Instructions of use:

1. Run the UserStory05.jar,
2. Pick a refill option by clicking one of the buttons,
3. Enter card number in the text box,
4. Press submit,
5. Press ok when the result is displayed in a pop-up window.

4.1.1 User Acceptance tests that are passed:

- **02:** In this test, the users' credit/debit card number is checked for validity. If it is valid then authorization message is displayed.
- **03:** In this test, the users' credit/debit card number is checked for validity. If it is not valid then error message is displayed.

4.1.2 User Acceptance tests that not are passed:

- **01:** In this test, the user inserts an invalid metro card. The verification is a metro card is not feasible in this implement, as that would require the presence of hardware device and backend database. Such verification is beyond the scope of this test case implementation.

- **04:** In this test, the user inserts a valid debit/credit card with insufficient balance. The verification of balance amount is not feasible in this implement, as that would require accessing the bank database. Such verification is beyond the scope of this test case implementation.

4.2 USigo02

Written by Simarpreet, implemented by Andrew.

This implementation allows the user to purchase a set of predefined tickets using credit/debit cards:

- 1 Trip - 3.50 CAD
- 2 Trips - 6.50 CAD
- Unlimited Evening - 5.50 CAD
- Unlimited Weekend - 14.00 CAD
- 1 Day - 10.00 CAD
- 3 Days - 19.50 CAD
- Group - 17.50 CAD

All the prices were taken from the official STM website. [1]

The implementation was done using Java 7 and Java Swing for the GUI.

The GUI class is responsible for creating the User Interface and handling any user input: clicks and text. The class starts by creating the following objects:

- Commuter: with a dummy bank account with only 15 dollars.
- Ticket: from which the tariffs are extracted.
- Card: the card object that will be assigned the number.
- Bank: responsible for validation the card number.
- Receipt: paper or email.

Each button is manually created, however, once a button is clicked, the ticket object is assigned its value by sending the button text and fetching the tariffs from the ticket object's tariffs hashmap.

Once this is done, the user is taken to a new screen to input the card number. Once entered and the Submit button is hit, the GUI class will call the validateCard() method from the Bank object. This function will validate the card number based on the LUHN[4] method. For testing purposes, the commuter is assigned a dummy bank account balance of 15 CAD, any ticket option above that number wouldn't accept the payment. If payment is accepted, the

user is taken to the receipt selection screen, where two options are present: paper or receipt. None of these have any effect, once clicked a popup will mention that the implementation is done in USigo03.

Instructions of use:

1. Run the USigo02_Andrew.exe
2. Pick a ticket option
3. Enter Card number
    - if valid and the ticket is under 15 CAD, the receipt option is posted.
    - if valid and the ticket is above 15 CAD, the payment is declined.
    - if invalid, the payment is declined.
4. If the receipt option page is shown, any option will have a popup saying the scenario is implemented in USigo03.

4.2.1 User Acceptance tests that are passed:

- **01:** In this test, the users' credit/debit card number is checked for validity using the Luhn method[4]. If it is valid then authorization message is displayed.
- **02:** In this test, the users' credit/debit card number is checked for validity using the Luhn method[4]. If it is not valid then error message is displayed.

4.2.2 User Acceptance tests that are not passed:

- **03:** In this test, the user inserts a valid debit/credit card with insufficient balance. The verification of balance amount is not feasible in this implement, as that would require accessing the bank database. Such verification is beyond the scope of this test case implementation. However, for testing purposes, we created a commuter with only 15 CAD in his hypothetical account. After the card validation for any trip that costs more than 15 CAD, the user will be prompted that they do not have enough cash.

4.3 USigo03

Written by Chinmay, implemented by Simarpreet.
The predcondition is that the payment is authorized.

This implementation allows the user to choose the means of receiving their payment information either by paper receipt or by email. The implementation was done using Java 7 and JavaFX 2.0 There are two java classes ReceiptPreference.java and ReceiptController.java ReceiptPreference.java class extends the

Application class of JavaFX and execution starts from this class. ReceiptController.java class is used for handling events when a user clicks on a button or selects an option in UI.

The flow of the implementation: After the payment has been authorized the user is shown two options on the screen

1. Paper Receipt
2. Email Receipt

These options have been shown by using radio buttons.

If the user selects a Paper Receipt and then clicks on the Submit button then an alert message is shown to the user informing him/her that they can collect their receipt. If the user selects Email Receipt then he/she has to provide a valid email address in a text field provided adjacent to Email Receipt's choice and then click on the Submit button. If the email address is valid then the alert message is shown to the user that the receipt has been sent to their provided email address. If the email address is invalid then the user is prompted to provide a valid email address. If the user clicks on the Exit button directly then by default paper receipt is printed and dispensed for the user.

Instructions of use:

1. Run US03_Simarpreet.exe .
2. Select the preferred mode of receipt
3. For Paper Receipt: choose the Paper Receipt radio button and then click on Submit button
4. For Email Receipt: choose the Email Receipt radio button and provide a valid email address in the corresponding text field and then click on the Submit button.

4.3.1 User Acceptance tests that are passed:

- **01:** In this test, the user is expected to input the email address if the user wants the payment receipt to be emailed to them. In this case, the provided email ID is not valid so the system will alert the user to provide a valid email ID and if the user fails to do so, then he/she cannot proceed with the email receipt option.
- **02:** In this test, the user is expected to input the email address where they want their receipt to be emailed. In this case, the provided email ID is valid and the user will be prompted by the system then the receipt has been sent to their respective email ID.
- **03:** In this test, the user is expected to select Paper Receipt as their preferred mode of receipt. In this case, the system will prompt the user that the receipt has been printed and can be collected by the user.

4.4 USigo04

Written by Rajasekhar, implemented by Chinmay.

This user story aims towards purchase of tickets using cash. Here, the user can choose from following fares options :

- 1 Trip - 3.5 CAD
- 2 Trips - 6.5 CAD
- One Day - 10 CAD
- Unlimited Weekend - 14 CAD
- Weekly - 25 CAD

The implementation code consists of Java8 and Java Swing GUI.

Main screen consists of 5 buttons placed on a JFrame, each for a fare option. Clicking any button invokes Java Swing's in-built event calling an assigned method.

There are 5 different methods that handle execution after each button click. These methods contain logic for verifying the amount entered against ticket price.

If the amount entered is less than the ticket price, user will be notified with a popup. If the amount is more than the ticket price, user will be notified about successful payment and returning the change.

As the counterfeit currency validation is not possible in this implementation, it is omitted.

Instructions of use: The first screen consists of five buttons. Clicking a button takes you to the next screen where you have to enter the amount you will be submitting to purchase the said ticket.

The next screen is a dialog box with title "Enter cash". It has an input field and a 'OK' button. The input field is restricted to accept only numbers and a decimal point. On click of 'OK' button this dialog box turns into a message dialog.

This message dialog tells you about payment authorization status, whatever it may be according to the amount of cash entered.

'OK' button on this message dialog closes the dialog and control is taken back to main screen with fare options.

4.4.1 User Acceptance tests that are passed:

- **01:** In this test, the user is expected to input the cash with amount enough to buy the ticket. Currency used is not counterfeit. In this case, the system is supposed to provide the ticket that is requested.

- **03:** In this test, the user is expected to input the cash with amount less than the price of the ticket. In this case, the system is supposed to give a message to the user stating the amount he/she entered is not enough.
- **04:** In this test, the user is expected to input the cash with amount more than the price of the ticket. In this case, the system is supposed to provide the ticket that is requested. Also, the system displays a message to the user stating the amount it will be returning as a change.

4.4.2 User Acceptance tests that are not passed:

- **02:** In this test, the user is expected to input the cash with amount enough to buy the ticket. However, the currency used is counterfeit. In this case, the system is supposed to NOT authorize the payment. As this verification not feasible to implement, it is not part of implementation.

4.5 USigo05

Written by Anisul, implemented by Rajasekhar.

The User has only choice to choose

- Monthly - 80CAD

The implementation consists of 5 HTML pages. In the index page firstly insertion of card to be made and then referred to Insertcard.html, There was only option monthly recharge and that pages refers to insertcash.html where all the acceptance tests are made. After based on the type of test the page will referred to either of the two HTML pages Sucessful.html or Unsucessful.html.

The currency validation is not possible to implement and is omitted.

Instructions of use:
First the execution begins with index.HTML page and that page should be opened using any browser(i.e google chrome, internet explorer). There a button naming Insert card to be clicked and then it takes on insert card page. There card inserted button to be pressed and then it takes to insert cash page. The next screen is to insert cash box there amount is entered that should be min and max of 2 digits. If the inserted cash is valid, then a screen with your card is loaded page appears.

4.5.1 User Acceptance tests that are passed:

- **01:** In this test user should insert the required amount of cash to refill Metrocard. Then it passes.

4.5.2 User Acceptance tests that are not passed:

- **02:** Failed to implement a prompt if the total is less than the required amount.
- **03:** In this test, the user is expected to input the cash with amount enough to buy the ticket. However, the currency used is counterfeit. In this case, the system is supposed to NOT authorize the payment. As this verification not feasible to implement, it is not part of implementation.
- **04:** Failed to implement return the correct change if the total exceeds the required amount.

5 Collaboration environments

- **Drive:** https://drive.google.com/drive/folders/1UGeY8pQCYC5Y5nAXkM1ROqO3McLTDc9F?usp=sharing
- **Github:** https://github.com/SimarpreetKaurj/SOEN-6481
- **Interviews:** https://drive.google.com/open?id=118oQlsmgVGkOg5TL5nhYqkWTmVBG76T8

References

[1] URL: http://www.stm.info/en#view-tariffs.

[2] Gary Guo. *SMART User Stories*. URL: https://ibearhost.atlassian.net/wiki/spaces/CAL/pages/7798792/SMART+User+Stories.

[3] *INVEST (mnemonic)*. URL: https://en.wikipedia.org/wiki/INVEST_(mnemonic).

[4] Pankaj. *Java Credit Card Validation – Luhn Algorithm in Java*. URL: https://www.journaldev.com/1443/java-credit-card-validation-luhn-algorithm-java.

[5] DAN RADIGAN. *Story points and estimationg*. URL: https://www.atlassian.com/agile/project-management/estimation.

[6] *What is I.N.V.E.S.T?* URL: https://yodiz.com/help/what-is-i-n-v-e-s-t/.

[7] *Write SMART Goals & INVEST for User Stories*. URL: https://www.visual-paradigm.com/scrum/write-user-story-smart-goals/).

Appendix

A Use case naming scheme

The naming scheme of the use cases changed to the following:

- Select Language: UCigo01 - Select Language
- Recharge card: UCigo02 - Recharge card
- Purchase Ticket: UCigo03 - Purchase Ticket
- Print Ticket: UCigo04 - Print Ticket
- Make Payment: UCigo05 - Make Payment
- Provide Receipt Preference: UCigo06 - Provide Receipt Preference
- Generate Receipt: UCigo07 - Generate Receipt