Sudoers’ Phase 2 Source

Raysean Jones-Dent, Tonye Andre Martial, Matthew Mitchell,

Kristine Dudley, Woo Choi, Justin Kim

26 April 2019

CMSC 495 7982

Dr. Janell Robinson

University of Maryland University College

**Contents**

[Version History 3](#_Toc7340512)

[Original Milestones 4](#_Toc7340513)

[Issues Encountered 6](#_Toc7340514)

[Modifications 7](#_Toc7340515)

[Working Program 9](#_Toc7340516)

[Discussion Feedback 15](#_Toc7340517)

# Version History

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 4/26/2019 | Version 1 | Document creation |
| 4/28/2019 | Version 2 | Corrections and discussion feedback repsonses |

# Original Milestones

Our team originally submitted key milestones with week 2 Project Plan assignment. Currently, our team is on schedule to complete the major milestones in accordance with the rubric. For Week 6, our goals were to complete the GUI coding and begin the initial coding of our backend components.  At this time, the multilingual GUI capability and backend components are nearly complete. We plan to continue working on these areas for the duration of the week and expect to have them completed before the start of Phase 3. Our original project milestones are listed below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tasks** | **Start Date** | **End Date** | **Goal Met?** |
| Develop Project Plan- Establish requirements, determine weekly leads & roles | March 25, 2019 | March 31, 2019 | YES |
| Develop User Guide & Test Plan, establish test cases | April 1, 2019 | April 7, 2019 | YES |
| Peer Review 1 | April 1, 2019 | April 7, 2019 | YES |
| Project Design- Develop coding framework & UI design, UML class diagram, XML entities | April 8, 2019 | April 14, 2019 | YES |
| Peer Review 2 | April 15, 2019 | April 21, 2019 | YES |
| Phase 1 Source- Begin initial GUI coding & documentation | April 15, 2019 | April 21, 2019 | YES |
| Phase 2 Source- Complete GUI & begin coding backend (ATM transactions, language setting) | April 22, 2019 | April 28, 2019 | YES |
| Phase 3 Source- Finalize system specifications, begin testing & debugging. | April 29, 2019 | May 6, 2019 | TBD |
| Peer Review 3 | May 7, 2019 | May 12, 2019 | TBD |
| Final Deliverables- Organize documentation, finalize deliverables (code, citations, etc.) | May 7, 2019 | May 12, 2019 | TBD |

# Issues Encountered

To date, our team has not encountered any challenges that could not be resolved.  As with previous phases, our team has held regular meetings to discuss status and project requirements during this phase of the project.  All team members are actively engaged in contributing feedback, testing, and coding in this phase of the project.

As of this week, we still have a couple of project requirements and minor user interface tweaks to implement.  The two outstanding features from our original project plan are the inactivity timeout and receipt printing. The remaining user interface issues that are known at this time are translations of message box strings, screen workflow finalization, and minor adjustments to control size.  We will continue discussing these feature gaps and issues in our next group meetings and we do not anticipate unsolvable technical challenges in getting them implemented.

Finally, a couple of design changes were made during this phase.  As we got further into the details of implementing the project, we found many adjustments could be made to simplify the design of the multilingual user interface.  After spending some time researching this topic, it was discovered JavaFX controls have the capability to read resource strings directly from the application resource bundle.  In our initial design, we anticipated that the internationalized strings would need to be programmatically set on each JavaFX control within each controller. Instead, we leveraged the JavaFX internationalization capability which simplified the controller classes.  In addition to internationalizing text for static JavaFX controls, we also discovered that many hard coded error strings would need to be translated as well. This was not anticipated in the initial design and required changes to the IAtmService interface and its implementations. We worked through this challenge by simply returning enumeration values for each error type and using the enumeration value to lookup the translated text from the resources.

# Modifications

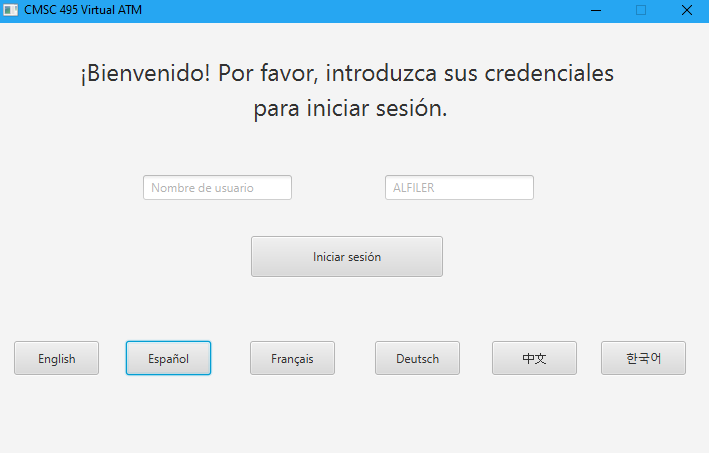
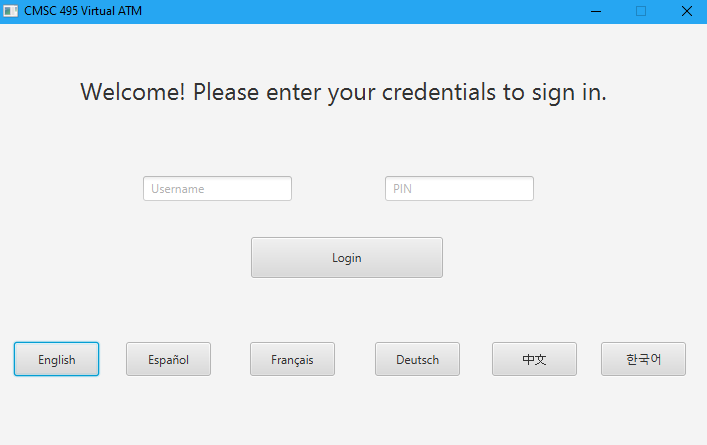
The items below detail the modifications and reflect any reevaluations made during this phase.

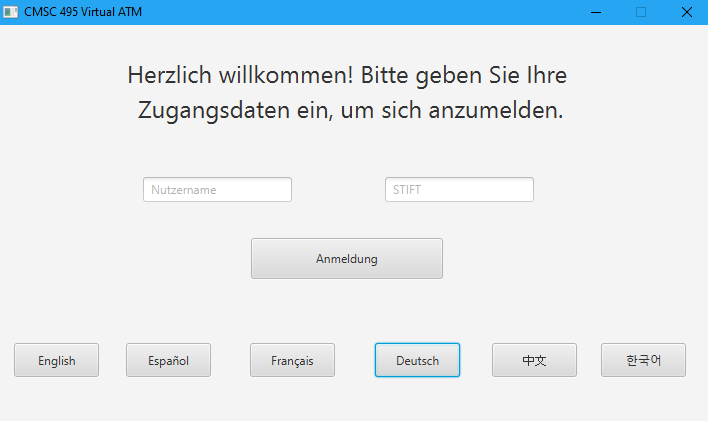
1. Design Plan Submitted Week 5, updated Week 6
   * Updated the class/interfaces/fxml section to reflect the current attributes and methods for each respective class
   * Updated the complete UML diagram and each subsystem diagram
2. Source code
   * Added XmlAtmDataAccess class to manage our XML based data store.
   * Added LocalAtmService class to handle “business logic” of our application.
   * Modified IAtmService methods to return AtmServiceError’s rather than throwing exceptions.
   * Added uitext.properties, uitext\_de.properties, uitext\_en.properties, uitext\_es.properties, uitext\_fr.properties, uitext\_ko.properties, and uitext\_zh\_CN.properties to store all of the current text translations.
   * Added new methods to BaseAtmController class for retrieving and binding translated text from our properties files
   * Modified BaseAtmController to create an instance of LocalAtmService instead of our mockup FakeAtmService class.
   * Added Security class to handle securely storing and matching user pin numbers
   * Modified LoginPage fxml and controller to dynamically change the user interface text based upon changes to the application default locale.
   * Modified the DepositPage fxml and controller to retrieve translated text for all static text within buttons, labels, text boxes, and combo boxes.
   * Modified the HistoryPage fxml and controller to retrieve translated text for all static text within buttons, labels, and table view columns.
   * Modified the MainPage fxml and controller to retrieve translated text for all static text within buttons and labels.
   * Modified the TransferPage fxml and controller to retrieve translated text for all static text within buttons, labels, text boxes, and combo boxes.
   * Modified the WithdrawPage fxml and controller to retrieve translated text for all static text within buttons, labels, text boxes, and combo boxes.
   * Scrubbed unneeded JavaFX control attributes from all controller classes

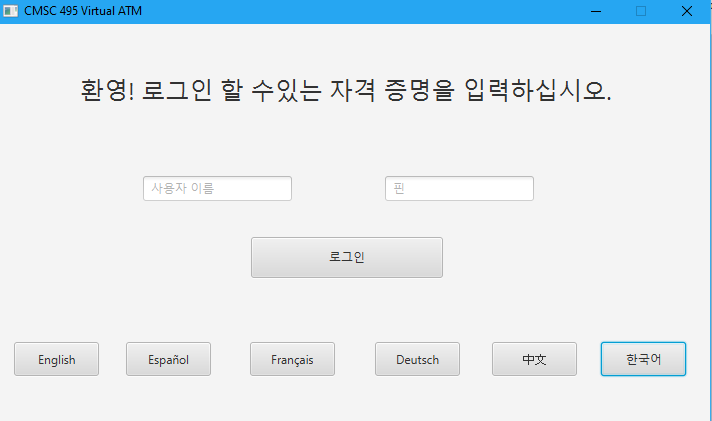
# Working Program

Below are screenshots that demonstrate the functionality of the multilingual GUI.  Login username is “sc” and assigned PIN is “12345”. Account balances have been pre-loaded with $1.11 in checking and $2.22 in savings.

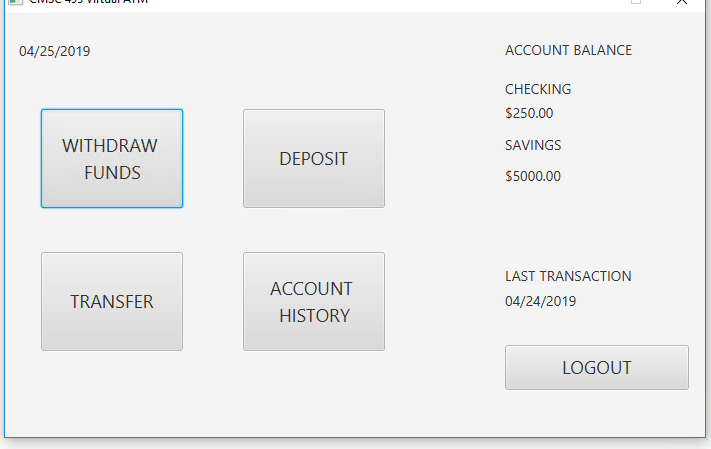
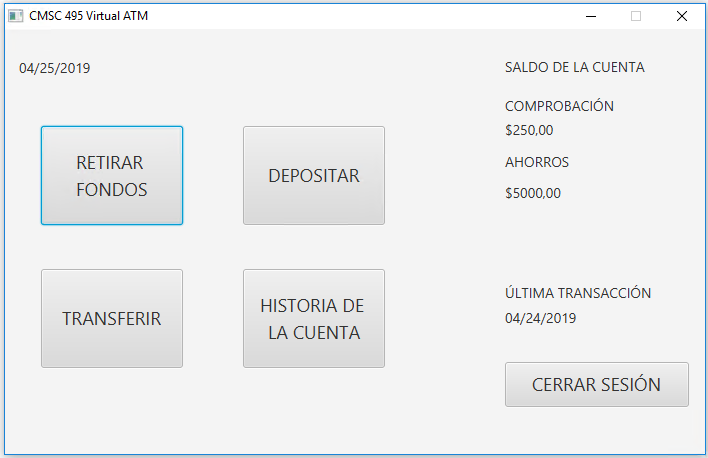
**LOGIN SCREEN**

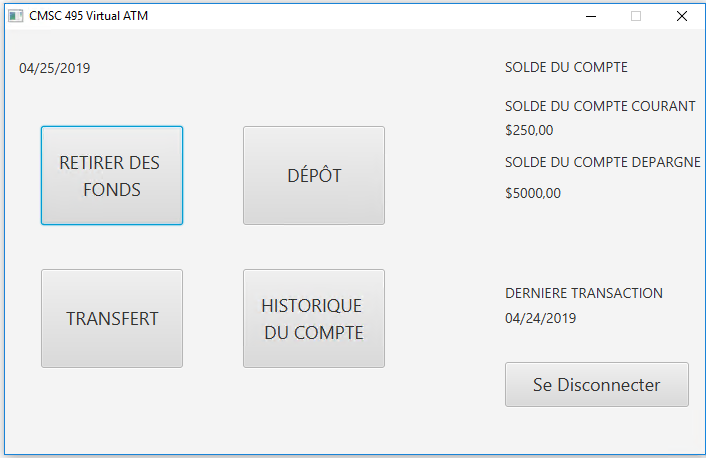
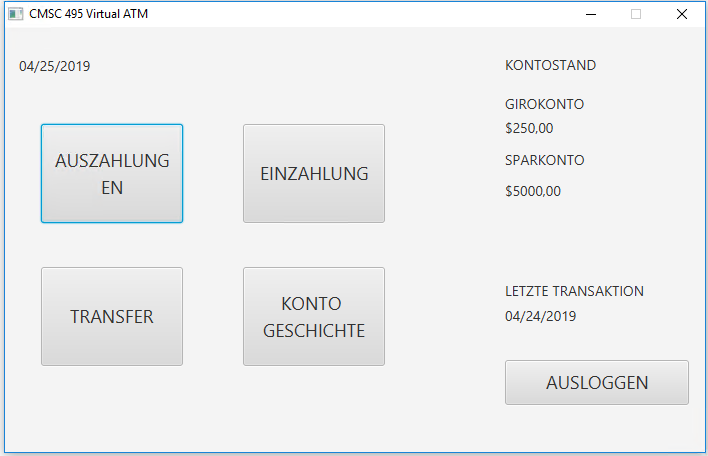


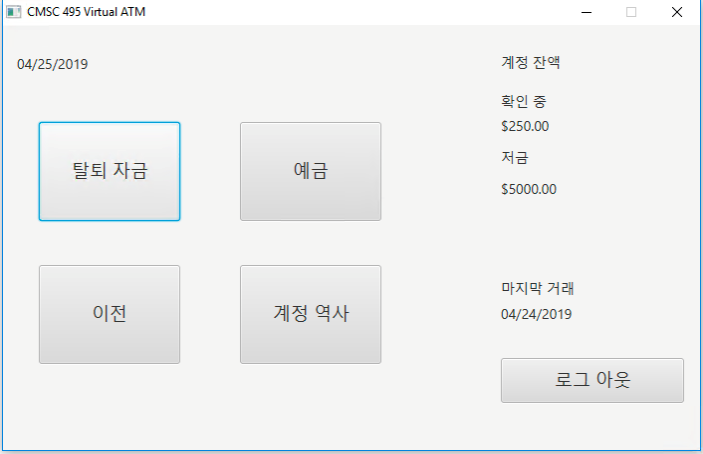
 

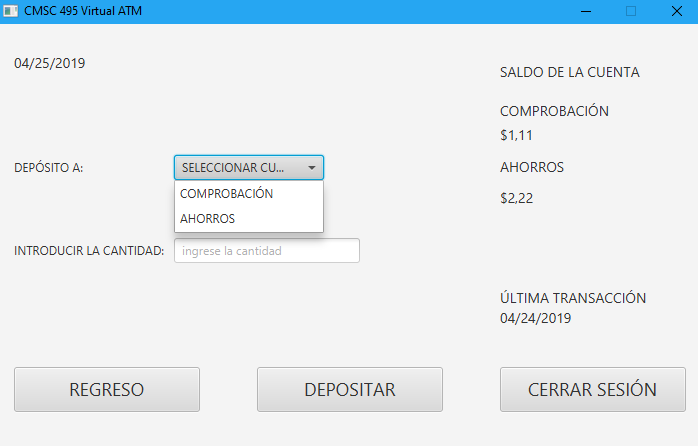
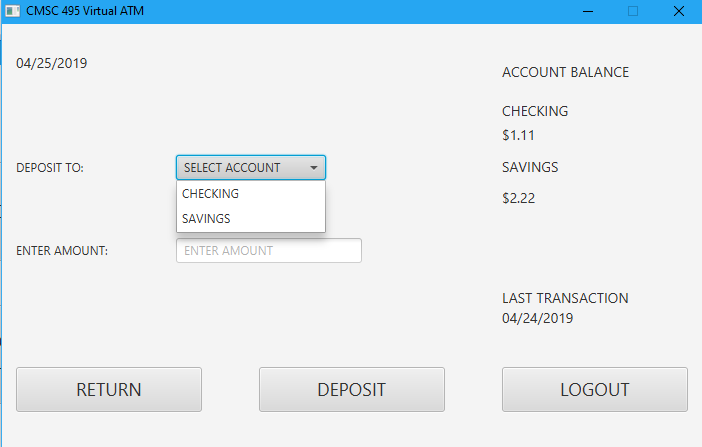
**MAIN PAGE (HOME PAGE)**

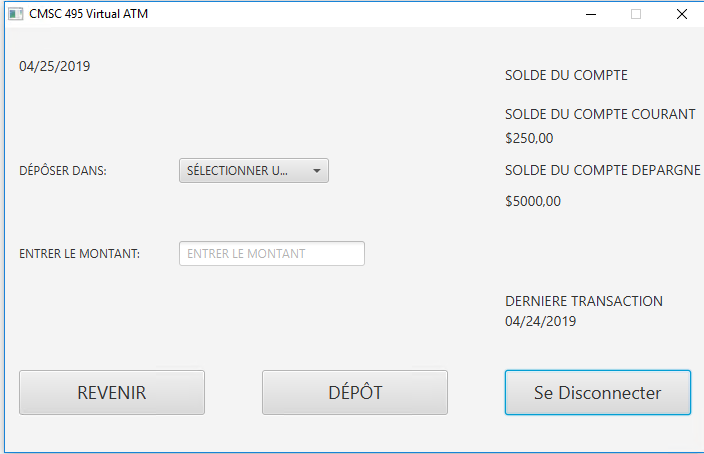
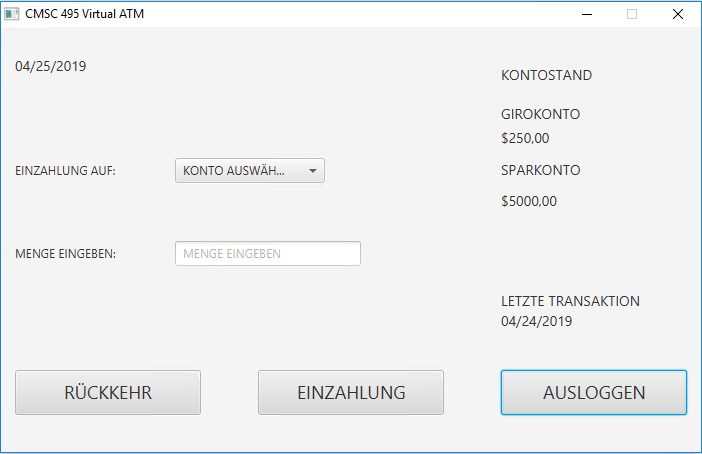
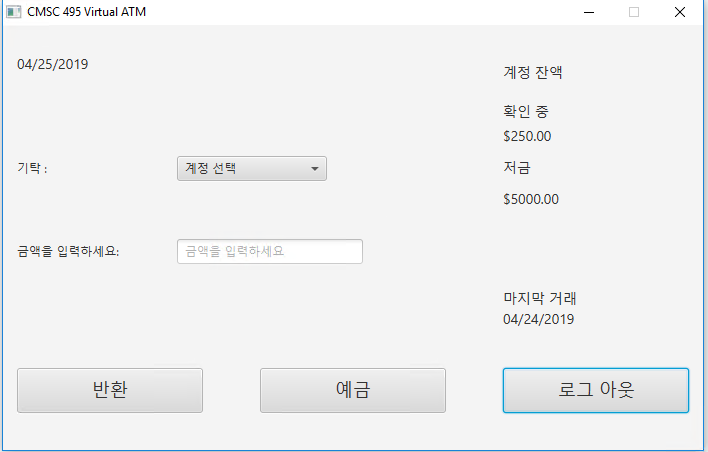
** **

**** ****

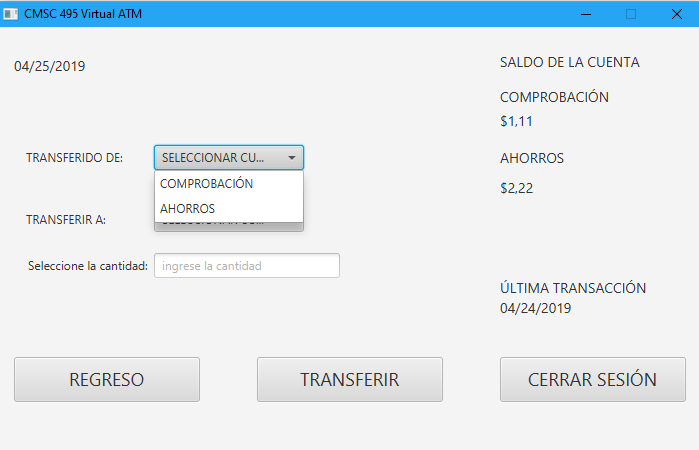
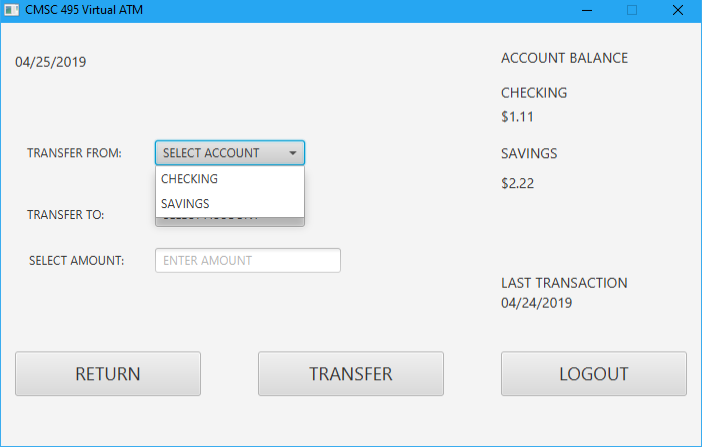
**** ****

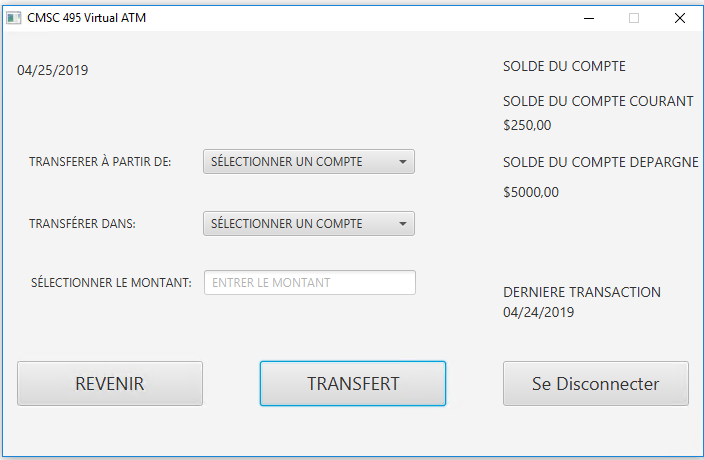
**DEPOSIT PAGE**

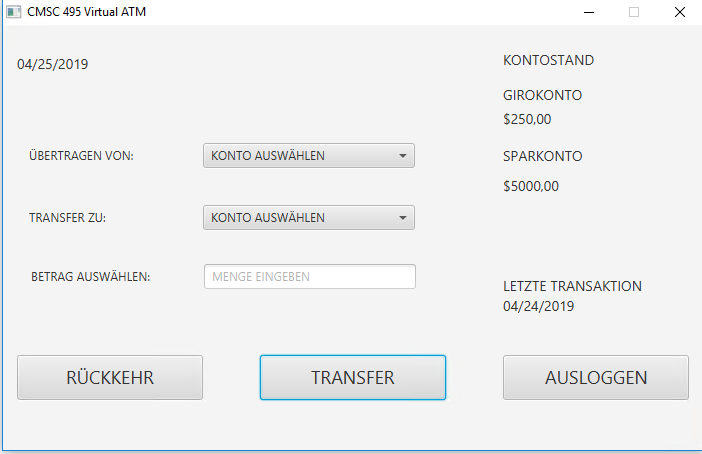
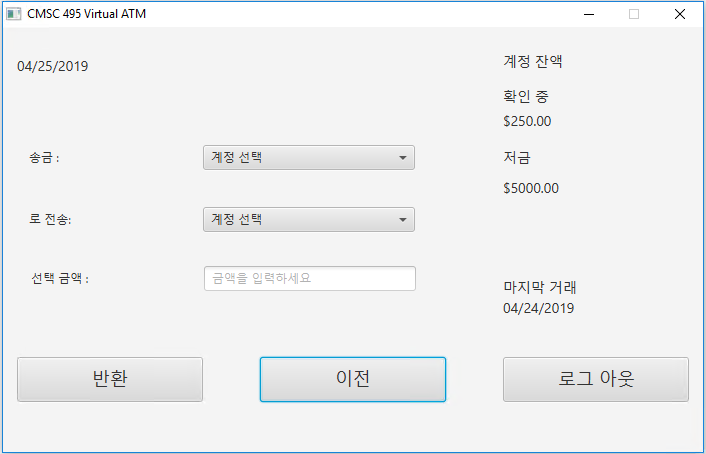
****

**** ****

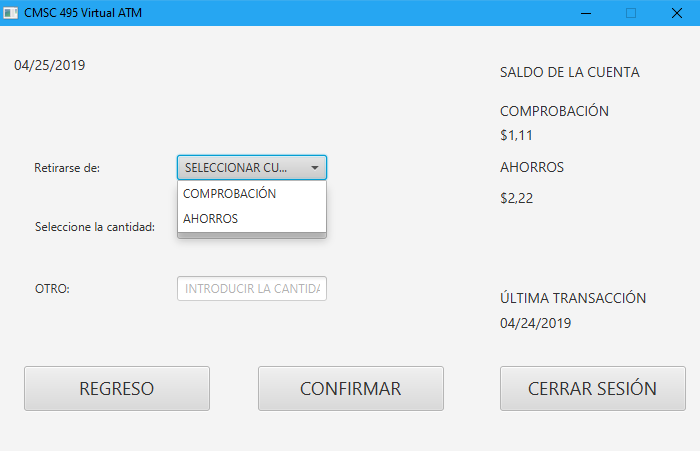
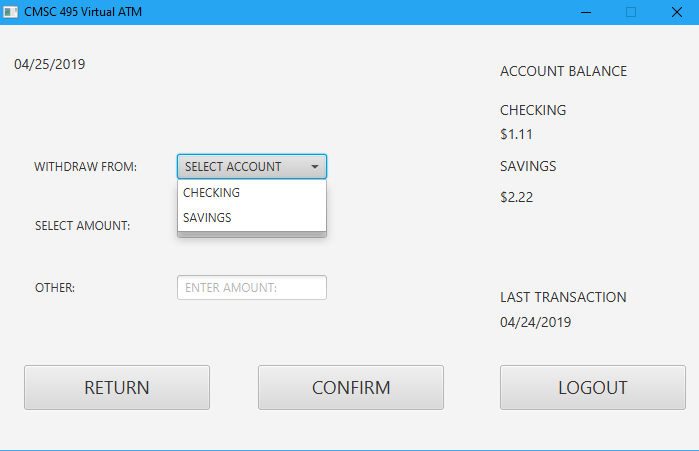
**TRANSFER PAGE**

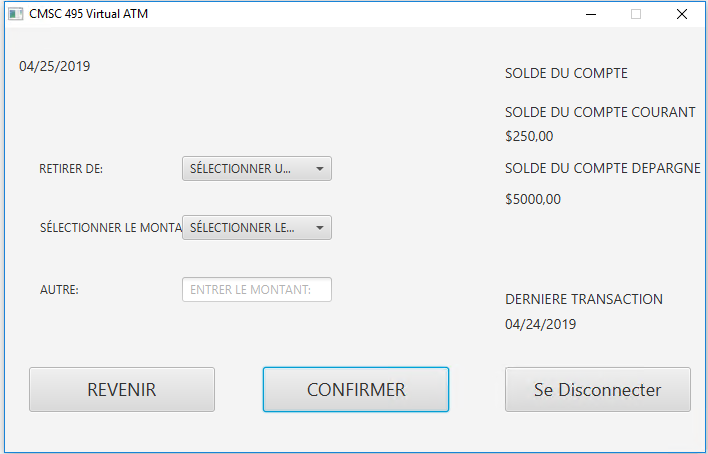
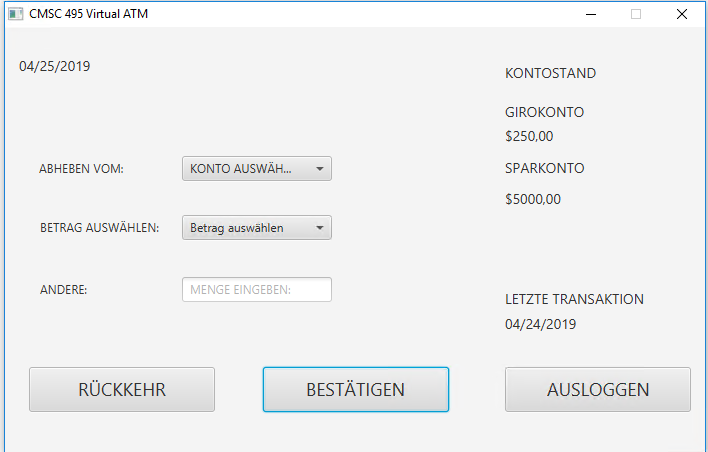
****

**** 

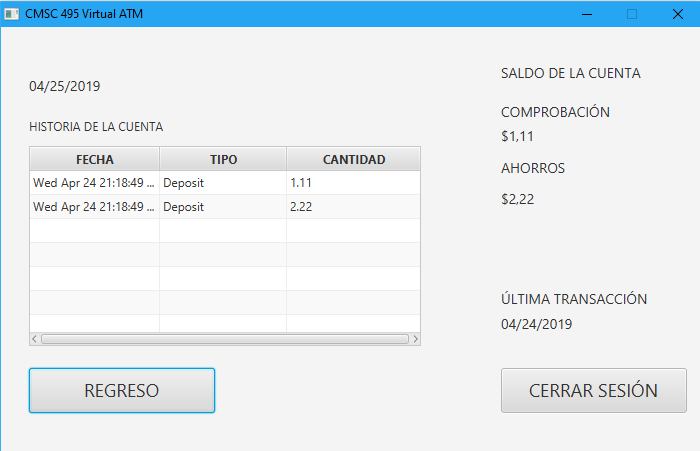
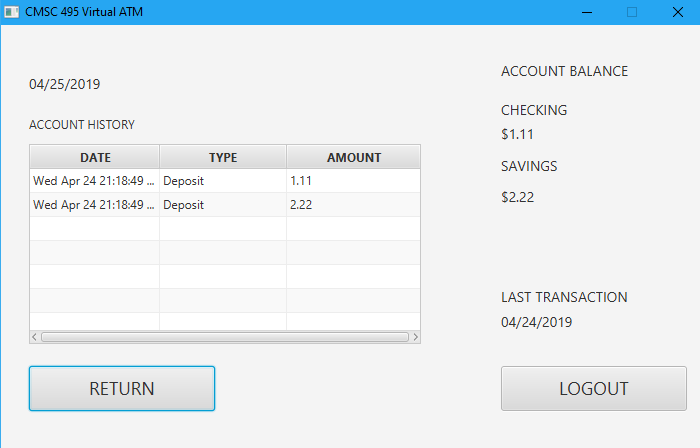
**WITHDRAW PAGE**

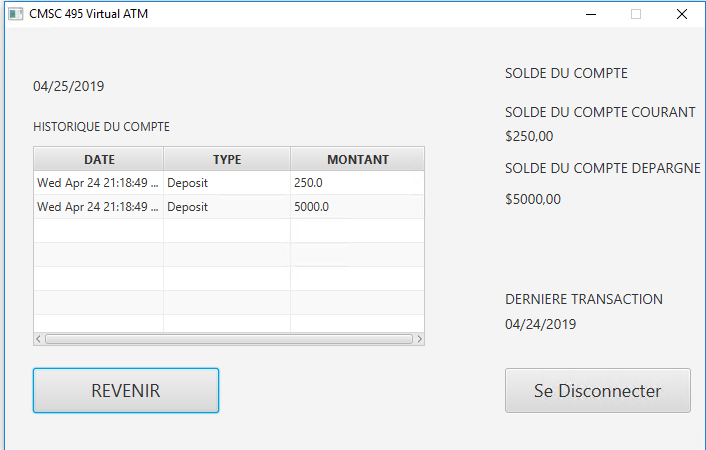
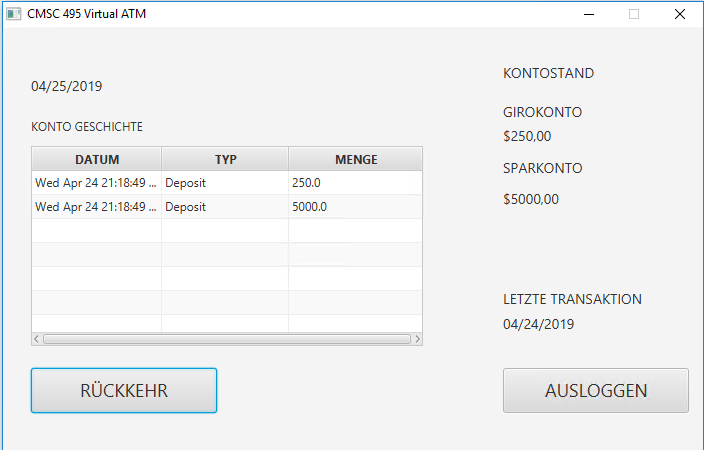
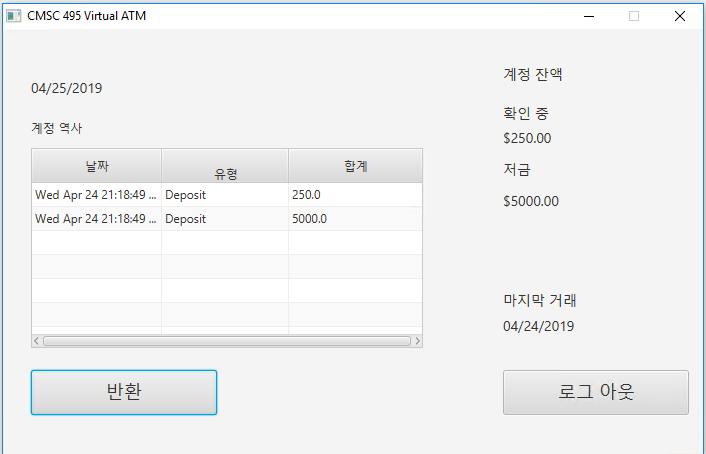
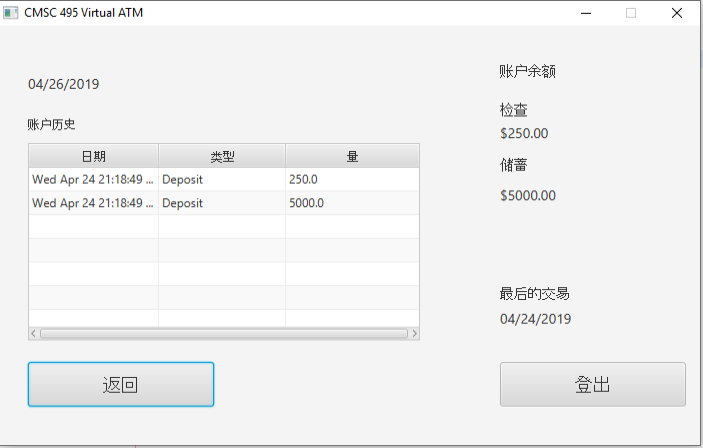
****

**** ****

**** ****

**ACCOUNT HISTORY PAGE**



# Discussion Feedback

|  |  |  |
| --- | --- | --- |
| **Discussion Feedback** | **Discussion Response** | **Team Action** |
| Received from: Travis Close  Very nice screenshots and updated design. I think the multi-language option is just great. Just fantastic. | We appreciate your feedback Travis. The multi-language capability created a ton of extra Dev/QA work, but JavaFX made it a little more manageable.  Regards,  Sudoers (Team 2) | No action needed. |
| Received from: Jacob Heddings  Hello Sudoers,  Excellent work! Everything looks very thorough, and the ATM design seems to work well. I have a small piece of feedback/suggestion: maybe for each language, there could be a small section that displays the totals in that language's native currency. For example, for France:  CHECKING  $100.00 (€89.58)  Probably not a necessity but it's something to consider.  Great design work!  Jacob | We appreciate your feedback Jacob. Your suggestion of having multiple currencies within the program is something we considered in the project requirements and design phases, but we ultimately decided to only support US currency due to the time constraints of the project. We believe a multilingual GUI is an entirely separate problem from supporting multiple currencies. Multiple currencies would likely require a completely different design in order to fully support it within a single program.  Regards,  Sudoers (Team 2) | Added feature request to the backlog in github. No action needed. |