Comprehensive Exercise Report

Team CHAAARM

Moulik Arora 231ADB036

Aman pathak 231ADB028

Cong Minh Tran 231ADB072

Anas Mehmood 221ADB134

Rakesh Gudimetla 221ADB239

Akshat Dhingra 221ADB173

Hemant - 231ADB009

Requirements/Analysis

Week 2

Journal

The following prompts are meant to aid your thought process as you complete the requirements/analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
    - Based on the client's request for his business, we need to develop a software that converts international bank statements (CSV) into the Latvian standard format bank statement (FiDAViSta).
    - Requirements for our software:
        - Software should be easy to use (User-friendly).
        - Software should take not more than 10 seconds to finish a run.
        - Software should allow the user to be flexible, able to add as much data into the software as he/she wants.
        - Software should convert multiple bank statements in CSV format from Revolut, Wise, Etsy into FiDAViSta.
        - Software should give the user the option to choose which of his customers he is making the FiDAViSta statement for.
        - Software should allow the user to add customers data such as customer id, legal id, name, surname and other Biometric details.
        - Software should extract relevant information such as transaction id, transaction date, type (Deposit/Withdrawal), receiver, sender,
        - Software should create a header of the FiDAViSta based on the available customer list and add the date the xml file is generated.
- After reading the client's brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.
    - We should ask him for samples.
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - Banking and finance in general
    - Financial Data Exchange United Standard (FiDAViSta)
    - Latvian standards of banking
    - Accounting
    - References:
        - Financial Data Exchange United Standard (FiDAViSta) - https://financelatvia.eu/wp-content/uploads/2018/08/FiDAViSta_v1-2_description_eng_13_08_2018.pdf
        - General accounting knowledge - Consultation with our customer.
        - Latvian standards of banking- Consultation with our customer.
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - Paybaltic SIA Payroll Service - accountants.
    - Potential customers could be other Latvian banking companies.
- Describe how each user would interact with the software.

    - Accountants of Paybaltic SIA Payroll Service:
        1. The user should provide an international bank statement as a CSV file to the software.
        2. Users will edit and choose from several options according to the needed output such as choosing their customers.
        3. The software uses the python code to scan the CSV file.
        4. The scanned CSV file will be converted into a XML format (FiDAViSta).
        5. The user receives the final output in XML format.
- What features must the software have? What should the users be able to do?
    - Conversion of statements from several banks especially Revolut, Wise and Etsy.
    - It must handle large amounts of data efficiently. (User story - Existing automation could not handle data with more than 25 characters.)
    - Precision and accuracy are a must while dealing with financial data. (User story - One extra 0 in a salary resulted in an interesting conversation and conflict with the customer.)
- Other notes:
    - Possibility to develop customized user interface./
    - Update for better visualization of the conversion to facilitate any desirable changes.

Software Requirements

Our Project is to develop a working automation of the conversion process of bank statements from various banks to Latvian standard format (FiDAViSta). User uploads the transaction file under CSV format (Payments, Taxes, Fee) and the software takes this CSV file as input, reads and validates the file structure, and generates an xml file following the standard format.

Use Cases:

1. The user wishes to convert international bank statements like (Wise, Revolut, Etsy) into FiDAViSta format.
2. The user wishes to get the appropriate data (filtering).
3. The user would also work with virtual banks.
4. The user must be able to choose what customer the statement belongs to.

User stories:

1. The user is a businessman, and wishes to conduct accounting as a side business. His daily job mainly focuses on filling in bank statements in csv into excel and then converting into FiDAViSta xml and making reports for his customer.
2. Has been doing this process manually by integrating files into excel, but now wishes to implement automation as his business expands.
3. Requested a fast, easy but efficient software to help the customer convert international bank statements into FiDaViSta with just a few clicks.
4. The user has personal data of customers but the statements do not, since he needs to add this data into FiDAViSta format, He requires a way to modify and select the customer.

Black-Box Testing

Instructions: Week 4

Journal

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

- What does input for the software look like (e.g., what type of data, how many pieces of data)?
    - CSV file containing details of payments, taxes, fees, customer.
    - Customer's Personal Data (User ID, Biometric Credentials, Legal ID etc.)
    - Quantity varies depending on the number of transactions.
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - FiDAViSta XML File which is basically the bank statement details formatted according to Latvian standards.
    - One XML file per processed CSV file.
- What equivalence classes can the input be broken into?
    - Valid CSV input: CSV format is correct, It belongs to supported banks (Revolut, Wise, Etsy, etc.) and contains valid customer data.
    - Invalid CSV input: CSV format is incorrect (missing columns etc.), unsupported banks and possibly corrupted file.
    - Boundary cases:
        - Minimum valid input: A single transaction CSV file
        - Maximum valid input: A very large file which will depend on the scalability of the software.
        - Empty file: No transactions, should return a proper error
    - Customer selection cases: Customer exists in records or Customer does not exist in records where the customer must be added.
- What boundary values exist for the input?
    - Character limit per field: Input text fields must not exceed system limits
    - Number of transactions: Test with very few (1-2) and very large numbers (e.g., 100,000+)
    - File size: Test with minimal (KB) and maximal (several MBs) file sizes
- Are there other cases that must be tested to test all requirements?
    - Performance testing: Ensure conversion completes within 10 seconds
    - Precision testing: Verify no data loss and required data conversion takes place
    - Error handling: Clear error message for invalid inputs (e.g., incorrect CSV structure)
    - User flexibility: Verify ability to add and modify customer data
    - Multi-bank support: Test conversion from each supported bank's format
- Other notes:
    - Test output XML against FiDAViSta standard to make sure it's as needed
    - We should also consider security aspects, ensuring customer data is handled safely

Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

25.05.2025 (Software has been retested and table has been updated)

Test ID	Description	Expected Results	Actual Results (updated)
TC-001	Upload a correct CSV file from a supported bank (e.g., Revolut, Wise, Etsy)	The file is converted to FiDAViSta XML	Pass:The file is converted successfully
TC-002	Upload a CSV file with missing important columns	The software shows an error message about missing data	The software displayed an error message
TC-003	Upload a CSV file from an unsupported bank	The software shows an error message about the unsupported format	The software displayed an error message:failed to process
TC-004	Upload an empty or damaged CSV file	The software generates an empty file	The file is generated with only tags and no values
TC-005	Upload a large CSV file with many transactions	The file is converted correctly within 10 seconds	Pass:The file is converted successfully
TC-006	Upload a CSV file with only one transaction	The file is converted correctly to XML	Pass:The file is converted successfully
TC-007	Upload a CSV file with very long text in some fields	The software processes the text correctly without errors	Pass:The file is converted successfully
TC-008	Select a customer from the database(CSV) before conversion	The customer’s details appear in the XML output	Pass:the customers details appeared in the clientSet tag
TC-009	–Select a customer that does not exist	The software shows an error or asks to add the	The file is converted with general details.

	in the system	customer	
TC-010	Convert a file without selecting a customer	The software sends a message and ask to select a valid client	The software sends the error message and asks for valid client
TC-011	Upload multiple CSV files at the same time	Each file is converted correctly, multiple XML files are created	Pass:converted multiple files simultaneously
TC-012	Check if processing time is 10 seconds or less	The software finishes processing within the time limit	Pass:it generates instantly
TC-013	Check if numbers and currency values are correct	The XML file shows accurate financial data	Pass: all correctly responded
TC-014	Check if the XML file is in the correct format	The output file follows the FiDAViSta standard	Pass:follows correct format
TC-015	Modify customer details before conversion	The changes appear correctly in the XML file	Pass:Changes updated succesfully

Design

Instructions: Week 6

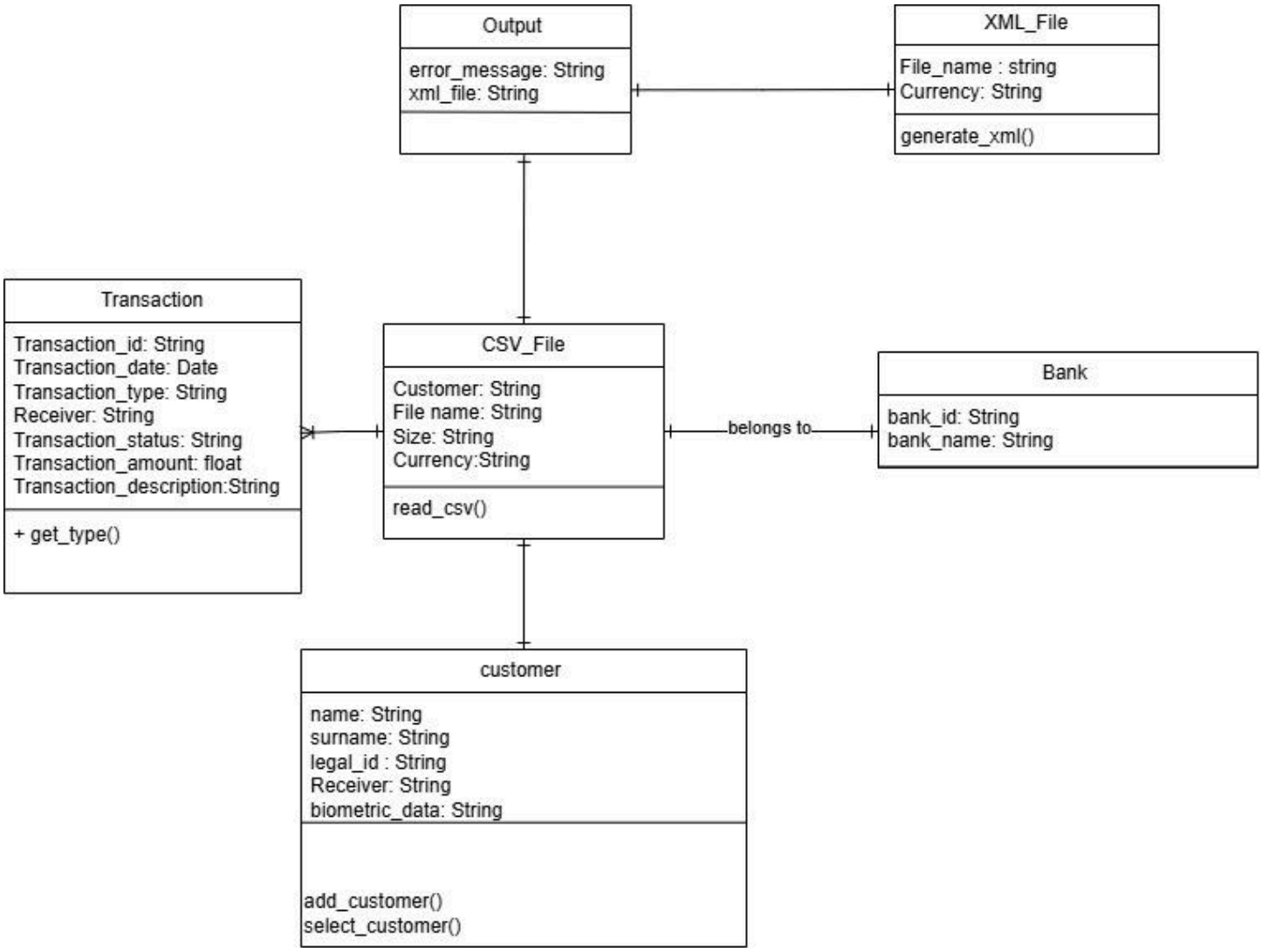
Journal

- The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.
- List the nouns from your requirements/analysis documentation.
    - Software
    - User
    - Customer
    - Bank Statement
    - CSV File
    - XML File
    - Transaction
    - Payment
    - Tax
    - Fee
    - Data
    - Format
    - Conversion
    - Error Message
    - Input
    - Output
    - Error Handling
    - Bank
    - Etsy
    - Revolut
    - Wise
    - Legal id
    - Name
    - Surname
    - Biometric data
    - Deposit
    - Withdrawal
    - Transaction ID
    - Transaction Date
    - Receiver
    - Currency
    - Info
    - Size
    - File name
- Which nouns potentially may represent a class in your design?
    - CSVFile
    - XMLFile
    - Transaction
    - Customer

    - Output
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - Transaction ID, Transaction Date, Transaction Type, Receiver, Transaction Amount, Transaction Description, Transaction Title - Transaction class
    - Bank ID, Bank Name - Bank
    - Customer, file name, size, Currency - CSVFile class
    - Name, Surname, Legal id, Biometric data - Customer class
    - Customer, Bank - XML File class
- Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.
    - Design 1:
        - Transaction ID, Transaction Date, Transaction Type, Receiver, Transaction Amount, Transaction Description, Transaction Title - Transaction class
        - Bank ID, Bank Name - Bank
        - Customer, file name, size, Currency - CSVFile class
        - Name, Surname, Legal id, Biometric data - Customer class
        - Customer, Bank - XML File class
- Which design do you plan to use? Explain why you have chosen this design

  We have only one design.
- List the verbs from your requirements/analysis documentation.
    - Upload
    - Convert
    - Validate
    - Process
    - Read
    - Write
    - Select
    - Modify
    - Filter
    - Generate
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - **Customer**: select_customer(), add_customer()
    - **CSVFile**: read_csv()
    - **XMLFile**: generate_xml()
    - **Transaction**: get_type()

Software Design



Implementation

Instructions: Week 8

Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

- What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.

  Programming Concepts Needed:
  - File Handling (**Pandas**) – Read CSV files and write FiDAViSta XML.
  - Regex(**re**) – Detects which bank's statement we are working with.
  - XML -(**xml.etree.ElementTree**) will be used for generating XML
  - GUI (**PyQT6**) – For developing a UI that should be advanced and simple.
  - Git & Agile – Version control and iterative development respectively.

  Other notes:
  - Use modular functions for clean, maintainable code.

Implementation Details

1. SOFTWARE OVERVIEW

This software is made to help accountants with converting and processing financial data (transaction, statement, etc) from csv or pdf file into xml FiDAViSta format. It automates the conversion with fast processing time and an easy to use GUI for navigation.

2. SYSTEM REQUIREMENT

- Operating System: Only works on Windows 10 and above.
- Input: CSV files containing bank statement data (e.g., payments, taxes, fees)
- Output: FiDAViSta-compliant XML files
- Disk space: Minimum 500 MB
- RAM: Minimum 2GB

3. INSTALLATION

Follow these detailed steps to install the software on your Windows system and set it up for use:

- Download zip file:
    1. Open your web browser (e.g., Google Chrome).
    2. Navigate to the following Google Drive link to access the software: Google Drive Download Link - https://drive.google.com/drive/folders/1Gr9aJNLn6cMid96FxTSMfDdl657A-bNd?usp=sharing
    3. Sign in to your Google account if prompted, or ensure you have access to the shared folder.
    4. Locate the software zip file in the Google Drive folder.
    5. Click the Download button. If prompted, choose to download the entire folder or the specific zip file.
    6. Save the zip file to a location on your computer. Do not save it in C disk.

- Extract the Software:
    1. Navigate to the downloaded zip file using File Explorer.
    2. Right-click the zip file and select Extract All (if using Windows' built-in tool) or use a third-party tool like 7-Zip or WinRAR.
    3. Choose a destination folder for the extracted files. Do not save it in C disk.
    4. Click Extract or follow the tool’s prompts to unzip the files. Ensure all files are extracted successfully.

4. HOW TO USE THE SOFTWARE

1. Launch the application by clicking on .exe file in the extracted folder.
2. Select the desired Mode (Etsy, Revolut, Wise).
3. Upload required files.
4. Select a client from the dropdown.
5. Click the PROCESS button to begin conversion.

5. USER INTERFACE OVERVIEW

The interface includes three modes to choose: Wise, Revolut and Etsy(default).

1. Transactions Statement CSV Files
2. Orders CSV Files
3. Etsy Invoice PDF FileS

Each section has 'Browse Files' and 'Clear Files' buttons. The bottom panel allows client selection and has a PROCESS button to start conversion.

6. CONVERT TO FIDAVISTA XML

1. Choose the conversion mode via the Mode menu.
2. Upload the necessary files for the selected mode.
3. Select a client profile.
4. Click PROCESS to generate the XML output.
5. Click EDIT to make any edit and choose SAVE OR SAVE AS to finalize.

7. CLIENT MANAGEMENT

Click 'Edit Clients' to:

- Add new client profiles
- Edit existing client information
- Delete outdated profiles

8. SUPPORTED MODES

- Etsy Mode: Requires Transaction CSV, Orders CSV, and Invoice PDFs
- Revolut Mode: Requires Transaction CSV
- Wise Mode: Requires Transaction CSV

Testing

Instructions: Week 10

Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
    - We changed the requirement that the system detects the type of bank, now the system does not detect the type of bank, it just warns the user in case a wrong file is added.
    - We changed the default user to “No Clients Selected”, as previously the default client selected is Martiņš.
    - We changed the requirement that the system requires to upload the correct file for the appropriate “Mode”.
    - We added a GUI feature, the planned requirement did not require an interface, now a UI is designed to make navigation and access to the features simple for all users.
    - We changed the requirement and added a Live edit feature for the generated xml file directly within the software itself, before testing, this feature was not introduced.
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
    - AddClientDialog
        - Equivalence Classes:
            - Name: Non-empty string, empty string.
            - Bank Account: Non-empty string, empty string.
            - Legal ID: Any string, empty.
            - Total Transactions: 0–999999, negative/non-integer not bigger than 999999.
            - Name/Bank/Legal ID: "", single char, 255 chars.
            - Total Transactions: 0, 1, 999999, -1, 1000000.
        - Paths to Test:
            1. Valid input → Click "OK" → Returns client data.
            2. Click "Cancel" → Dialog closes.
            3. Empty name/bank → Click "OK" → Dialog stays open.
    - EditClientDialog
        - Equivalence Classes:
            - Client: Valid dict, missing fields.
            - Name: Non-empty, empty.
            - Bank Account: Non-empty, empty.
            - Legal ID: Any string, empty.
            - Total Transactions: 0–999999, negative/non-integer/>999999. Boundary Values:
            - Name/Bank/Legal ID: "", single char, long string.
            - Total Transactions: 0, 1, 999999, -1, 1000000.
            - Client: Minimal valid dict, invalid dict.
        - Paths to Test:
            1. Valid edit → Click "OK" → Returns updated data.
            2. Click "Cancel" → No changes.
            3. Empty name → Click "OK" → Dialog stays open.

- ClientEditorDialog
  - Equivalence Classes:
    - Client List: Default + others, invalid clients.
    - Selected Client: Valid non-default, default/no selection.
    - Add: Valid client, empty fields.
    - Edit: Valid non-default, default/invalid data.
    - Delete: Non-default, default. Boundary Values:
    - Client List: [default], 1 client, 1000 clients.
    - Selected Index: -1, 0, len(clients)-1.
  - Paths to Test:
    1. Add client → Appears in list.
    2. Edit non-default → Updates list.
    3. Edit default → Shows warning.
    4. Delete non-default → Removes client.
    5. Delete default → Shows warning.
    6. Save → Updates clients.
    7. No selection → Edit/delete disabled.
- EditXMLDialog
  - Equivalence Classes:
    - XML File: Readable XML, non-existent/non-readable.
    - Text: Any text, empty.
    - Save: Writable file, unwritable.
    - Save As: Writable path, unwritable/canceled. Boundary Values:
    - XML File: Empty, 1GB, minimal XML.
    - Text: "", 10MB string.
    - File Path: Valid, read-only, long path.
  - Paths to Test:
    1. Load valid XML → Displays content.
    2. Load invalid file → Shows error, closes.
    3. Edit → Save → Updates file.
    4. Save as new file → Creates file.
    5. Save to unwritable → Shows error.
    6. Undo/redo → Text state changes.
    7. Cancel → No save.
- SelectXMLDialog
  - Equivalence Classes:
    - XML Files: Non-empty list, empty/invalid paths.
    - Selected File: Valid index, no selection. Boundary Values:
    - XML Files: [], 1 file, 1000 files.
    - Selected Index: -1, 0, len(files)-1.
  - Paths to Test:
    1. Select file → Click "Edit" → Opens EditXMLDialog.
    2. No selection → Edit button disabled.
    3. Close → Dialog closes.
    4. Double-click file → Opens EditXMLDialog.
- DragDropLabel
  - Equivalence Classes:
    - Dropped Files: Valid paths, non-file URLs/duplicates.
    - Added Files: Non-empty paths, empty/invalid paths. Boundary Values:

        - Files: [], 1 file, 1000 files.
        - File Path: Valid, long, invalid.
    - Paths to Test:
        1. Drag files → Adds and displays.
        2. Add files via browse → Adds and displays.
        3. Clear files → Resets list.
        4. Add duplicate → Ignores duplicate.
        5. Drag invalid URL → No files added.
- MainWindow
    - Equivalence Classes:
        - Client List: Default + others, invalid data.
        - Selected Client: Valid index, invalid index.
        - Mode: Etsy, Wise, Revolut.
        - Files (Etsy): Non-empty CSVs, empty lists.
        - Files (Wise/Revolut): Non-empty CSV, empty.
        - Process: Valid files + non-default client, no files/default client. Boundary Values:
        - Client List: [default], 1 client, 1000 clients.
        - Selected Client: 0, len(clients)-1.
        - Files: [], 1 file, 1000 files.
    - Paths to Test:
        1. Load clients (no file) → Default client only.
        2. Switch modes → UI updates.
        3. Update combo → Shows clients.
        4. Edit clients → Updates clients.
        5. Process Etsy files → Generates XML.
        6. Process Wise/Revolut → Generates XML.
        7. Process with default client → Shows warning.
        8. Process with no files → Shows warning.
        9. Process invalid file → Shows error.
        10. Browse files → Adds to drop label.

Testing Details

This section lists the test programs created to verify the converter based on the black box testing requirements.

List of our test programs

1. **Valid CSV Conversion Test**

We test if a valid CSV file from a supported bank (Revolut, Wise, or Etsy) converts correctly to a FiDAViSta XML file.

Steps taken during testing:

1. Inputs a valid CSV with all the details, checks if the output follows the XML FiDAViSta format.
2. Verifies if conversion finishes within 10 seconds or less.

2. **Invalid CSV Error Handling Test**

Test how the software handles invalid CSV files, such as incorrect formats, missing columns or unsupported bank.

Steps taken during testing:

1. Inputs a CSV with missing columns (e.g., no transaction ID) or from an unsupported bank.
2. Check if the software shows a clear error message.

3. **Single Transaction Input Test**

Tests the software with a CSV file containing only one transaction.

Steps taken during testing:

1. Inputs a single-transaction CSV file (minimum valid input boundary case).
2. Verifies the software generates a correct FiDAViSta XML file with one transaction and customer details.

4. **Large File Input Test**

Tests the software with a very large CSV file containing many transactions.

Steps taken during testing:

1. Input a CSV with 10000+ transactions.
2. Checks if the software generates a correct FiDAViSta XML file within 10 seconds.

5. **Empty File Error Test**

Tests the software with a very large CSV file containing many transactions.

Steps taken during testing:

1. Inputs an empty CSV file
2. Verifies the software shows a clear error message
3. Ensures the software does not crash.

6. **New Customer Addition Test**

Tests adding a new customer when processing a CSV file.

Steps taken during testing:

1. Inputs a valid CSV and adds a new customer with details

2. Verifies the XML includes the new customer's details.

7. **Data Precision Verification Test**

   Tests the accuracy of the conversion, test if the format after conversion is correct.
   Steps taken during testing:

   1. Inputs a CSV with specific transaction data
   2. Verifies the XML output matches the input exactly with no data loss.
   3. Tests if currency and other numerical format remains the same.

Presentation

Instructions:Week 12

Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

- Give a brief description of your final project
    - We developed a software solution for Paybaltic SIA that automates the conversion of international bank statements in CSV format into FiDAViSta-compliant XML files, significantly reducing manual effort, enhancing accuracy, supporting scalability, and improving overall employee efficiency.
- Describe your requirement assumptions/additions.
    - **Batch File Upload:** Enables users to add and process multiple bank statement files simultaneously, enhancing efficiency and convenience..
    - **Intuitive Interface:** Provides easy access through a user-friendly graphical interface designed for intuitive navigation.
    - **In-App XML Editing:** Allows users to view and modify the contents of the generated FiDAViSta XML files directly within the software, eliminating the need for external tools.
- Describe your design options and decisions. How did you weigh the pros and cons of the different designs to make your decision?
    - We used a modular approach to ensure better maintainability and scalability over a simpler script-based design.
- How did the extension affect your design?
    - Shifting from basic source code to a fully functional software with GUI had a major impact on our overall design approach.
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - Following the Agile methodology, we regularly met with the customer to share updates and quickly communicate software changes..
    - Completing the software required us to deepen our knowledge and work with Pandas, xml.etree library, CustomTkinter etc.
    - The library xml.etree was a new tool for us, and we learned it from scratch to complete the project.
    - We highly gained skills on documentation of work and researching on it to finish the documentation
- What functionalities are you going to demo?
    - **CSV to XML Conversion:** The software converts CSV files into XML format that meets FiDAViSta standards, making data transformation easier and faster.
    - **File Flexibility:** The system works and processes different types of bank statement files(Etsy, Wise, Revolit), ensuring compatibility with various formats.
    - **Multi-File Upload:** Users can upload multiple files at the same time, improving efficiency and reducing the time spent on individual file processing.
    - **Live Edit Feature:** Users can edit the generated XML files directly within the software, allowing for quick adjustments without the need for external tools

    - **GUI Features:** The software has an easy-to-use interface, designed to make navigation and access to the features simple for all users.
- Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
    - Minh - Introduction and use cases
    - moulik - whole source code's architecture
    - Aman - technologies used and class diagrams
    - Hemant- graphical user interface
    - Anas - testing
    - Akshat - demonstration of the software
    - Rakesh - conclusion and future scope