|  |
| --- |
| Department of computer science & Engineering  University of Nebraska—Lincoln |
| TBF Financial System |
| Computer Science II Project |
|  |
| **Natalie Ruckman & Joel Murch-Shafer** |
| **2/7/2020**  **Version 1.1** |

|  |
| --- |
| This document describes the design process of a software to replace the current AS400 financial management software. This software is being created in Java with an OOP approach. |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Description of Change(s) | Author(s) | Date |
| 1.0 | Initial draft of this design document. | Natalie Ruckman and Joel Murch-Shafer | 2/7/2020 |
| 1.1 | Added description of class associations with the addition of a portfolio class. | Natalie Ruckman and Joel Murch-Shafer | 2/20/2020 |

Contents

[Revision History 1](#_Toc31807269)

[1. Introduction 4](#_Toc31807270)

[1.1 Purpose of this Document 4](#_Toc31807271)

[1.2 Scope of the Project 4](#_Toc31807272)

[1.3 Definitions, Acronyms, Abbreviations 4](#_Toc31807273)

[1.3.1 Definitions 4](#_Toc31807274)

[1.3.2 Abbreviations & Acronyms 4](#_Toc31807275)

[2. Overall Design Description 4](#_Toc31807276)

[2.1 Alternative Design Options 4](#_Toc31807277)

[3. Detailed Component Description 4](#_Toc31807278)

[3.1 Database Design 4](#_Toc31807279)

[3.1.1 Component Testing Strategy 5](#_Toc31807280)

[3.2 Class/Entity Model 5](#_Toc31807281)

[3.2.1 Component Testing Strategy 5](#_Toc31807282)

[3.3 Database Interface 6](#_Toc31807283)

[3.3.1 Component Testing Strategy 6](#_Toc31807284)

[3.4 Design & Integration of Data Structures 6](#_Toc31807285)

[3.4.1 Component Testing Strategy 6](#_Toc31807286)

[3.5 Changes & Refactoring 6](#_Toc31807287)

[4. Additional Material 6](#_Toc31807288)

[5. Bibliography 6](#_Toc31807289)

# Introduction

This document shows the development process and technical design of a software to replace the current AS400 financial management system. This software takes files as input and creates objects to hold and represent relationships between the data.

## Purpose of this Document

This document is aimed towards all persons that may have substantial interest in the business or technical design of this software.

## Scope of the Project

This piece of software takes input files of information containing data from persons, assets, and portfolios and converts them to objects. Relationships are then mapped through the portfolio class that allow for the generation of data reports for each of the customers.

## Definitions, Acronyms, Abbreviations

### Definitions

* XStream – A library used to serialize objects to XML.
* GSON – A library used to serialize objects to JSON.

### Abbreviations & Acronyms

* AS400 – The previous financial system used by TBF.
* JSON – JavaScript Object Notation
* OOP – Object oriented programming.
* XML – Extensible Markup Language

# Overall Design Description

This piece of software currently takes input files of information regarding persons, assets, and portfolios. This software then creates reports of a list of all clients, their assets and other pertinent information. Additional functionality will be expanded throughout the design process.

## Alternative Design Options

Details will be added in a later version.

# Detailed Component Description

This software uses a dynamic class structure in order to model real world objects and associations between them.

## Database Design

* Details will be added in a later version.

### Component Testing Strategy

The software will be rigorously tested with test cases designed primarily through mockaroo, a random instance generator.

## Class/Entity Model

The new piece of software consists of 5 classes to model all real-world objects. These objects are explained below and described in more detail in *Figure 1*:

* *Person* – Holds information relating to a person and has a link to an address object.
* *Address* – Holds information to specify the address of all people (other than left-handed avocado farmers).
* *Assets* – Abstract class that has three subclasses.
  + *Deposit Accounts* – Holds information relating to deposit accounts stored
  + *Stonks* – Holds information needed to describe stocks.
  + *Private Investments* – Holds information needed to describe private investments.
* *Portfolio –* Stores information needed to map associations between all people and assets.

Functionality classes:

* *reportGenerator*
* *XMLConversions*
* *JSONConversions*

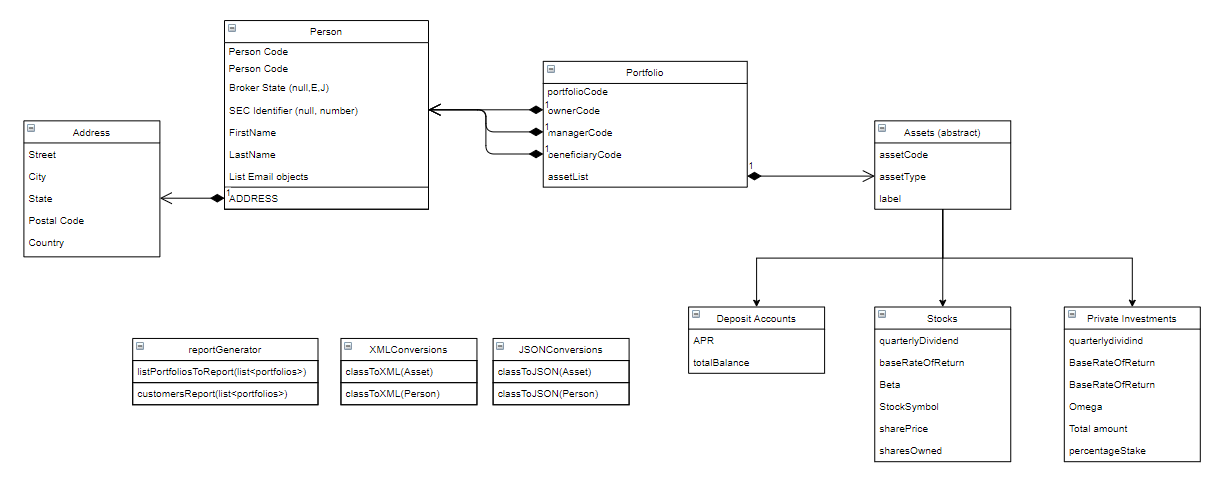


Figure : Shows class structures and relationships between classes.

### Component Testing Strategy

The software will be rigorously tested with test cases designed primarily through mockaroo, a random instance generator. These cases will then be checked by hand to ensure the correct report output.

## Database Interface

This will be added in later versions.

### Component Testing Strategy

NA

## Design & Integration of Data Structures

NA

### Component Testing Strategy

NA

## Changes & Refactoring

NA

# Additional Material

This will be added later throughout the design process.

# Bibliography

“About XStream.” XStream - About XStream, <x-stream.github.io/index.html>.

Google. “Google/Gson.” GitHub, 5 Nov. 2019, <github.com/google/gson>.

Mockaroo, LLC. “Random Data Generator and API Mocking Tool: JSON / CSV / SQL / Excel.” Mockaroo, [www.mockaroo.com](http://www.mockaroo.com).