This document outlined the project analysis portion of a personal finance manager called CashStash using an Object-Oriented and UML design modeling process.

The Domain Analysis portion attempts to describe the CashStash application on a very general level and does not aim to elaborate on how the application functions internally. The Concept Statement is a general explainationthoroughly explains the purpose of the application, who can use it, the features it provides and how it interacts with external systems. Using information from the concept statement, a Conceptual Domain Model (CDM) was generated to show the relationships between high-level systems that make up the applications logic. While most of these associations refer to constructs local to the program, some of them may interact with external systems such as a financial institution. Another type of association shows child-parent relationships, meaning one construct (the child) may inherit the ideas and attributes of another (the parent). The Domain State Model (DSM) elaborates the ideas presented in the CDM by showing their various states and behaviors. The associations between states in the DSM shows what initiates a change of state and these ideas where fabricated with the help of information found within the concept statement.

The Application Analysis portion aims to describe the CashStash application on more of an application specific level, meaning it shows in detail how each feature will function and how the internal components of the application interact with each other. The Application Interaction Modal (AIM) describes fifteen different use cases of the application. First, an essential use case will describe a step-by-step interaction between the user, the Cash Stash system and any external systems in general terms. These step-by-step interactions also show how to handle exceptions, which are unexpected actions that may occur during the process. Next, a scenario is written that shows the interaction previously described by the essential use case but using a third person point of view; such as, “John Doe enters his username”. After the scenario, a High-level System Sequence Diagram (HSSD) is created that shows the essential use case in graphical form. The graphical form is easier to understand because it shows a clear interaction between the user and the system with arrows pointing in the direction of the interaction and a message that shows the action being performed. Some internal actions are also represented as additional text on the system side of the chart. These three concepts (essential use case, scenario, HSSD) explain the application use cases in a quick and concise form. Now the concrete use cases will expand the concepts in essential use cases to provide more detail about what is happening during these interactions. For example, where an essential use case may say “System validates information”, the concrete use case will say “The system checks to make sure the email address provided does not belong to an existing account”. Furthermore, the concrete use cases are also converted into a graphical form called the Detailed System Sequence Diagram (DSSD). The DSSD has a similar style to the HSSD but it shows more information than just the interactions between the user and the system. The DSSD also contains boundary objects that represent the interface with which the user interacts to initiate a process, control objects which manage the rest of the interaction within a use case and entity objects which are representations of the data being created by a process. For each DSSD, a control object is created due to the interaction between a user and a boundary object, then the control object may or may not request further interaction from the user throughout the process. Similar to the CDM and DSM from the Domain Analysis portion, an Application Class Model (ACM) shows the relationships between the different objects found in the DSSD and an Application State Model (ASM) shows the various states and behaviors of each application use case. While the ACM shows associations of objects among all use cases, the ASM will show the state changes of each use case based on the incoming and outgoing interactions with their respective control objects found in the DSSD.

The Consolidated Class Model aims to combine the CDM with the ACM to show how the individual processes of the application interact with the more general systems. By taking the CDM and comparing it with the ideas in the ACM, some of the concepts were similar enough that they could be consolidated into one construct which helped simplify the system.