**Project Plan**

**Project Name:** ATM Simulator

**Description and Scope:**

The project simulates the operations of an ATM, which can perform the specific set of operations labeled below:

1) Withdraw money.

2) Deposit money.

3) Check account balance.

4) Transfer money to another account.

5) Authenticate account/PIN.

6) Print transaction receipts.

The ATM acts as an interface for the Customers to utilize these activities. A central Bank will act as the authority for an ATM, keeping track of balances, authentication, and any other important data needed. Additionally, the ATM interface must be extensible in the sense that multiple ATMs are allowed to communicate with the Bank.

Special conditions must be taken into account for the project:

1) No money is available.

2) Unresponsive Bank authority.

3) “Paper” for receipts has run out.

This project will be implemented as a simple Windows process, which reads in data files for the Bank entity and then interfaces with the user to perform operations through a graphical ATM.

**Delivery and Deployment:**

Delivery Schedule:

**1/30** – Project Plan and Project Checkpoint #1 documents, Customer entity coded

**2/6** – Bank entity coded with data retrieval, simple ATM interfacing (without ATM operations)

**2/13** – ATM entity coded with all operations (without GUI support), Project Checkpoint #2 document

**2/20** – ATM entity with GUI support, interfacing between all entities complete and functional

**2/27 –** Testing complete, all use cases and special conditions accounted for, Project Checkpoint #3 document

The ATM implementation will be phased like this per week, so that necessary tasks are all completed by the end of the week. An initial submission will be made on **2/20** as a precaution, with the final submission made by the due date, **2/27**.

All code will be documented with Doxygen support, as well as a procedure for following code standards and conventions outlined in a document on the source control for the project. More information on documentation is outlined in the **Project Management** section.

**Team:**

**Andrew Taylor**

Email: [andrew.taylor@digipen.edu](mailto:andrew.taylor@digipen.edu)

Role: ATM Programmer

**Brandon Chavez**

Email: [b.chavez@digipen.edu](mailto:b.chavez@digipen.edu)

Role: Bank and Customer Programmer

**Tyler Hundt**

Email: [t.hundt@digipen.edu](mailto:t.hundt@digipen.edu)

Role: GUI Programmer

Despite primary roles, team members may help each other and code other aspects of the project as need be. The team will have a flat team structure akin to Agile, where no leaders are involved and everyone is responsible for their own tasks.

**Project Management:**

**Trello** will the primary way of distributing simple tasks to members and tracking down bugs, distributing each to their own category or board for the project. Since this is a simple project, not much more will be required.

An **Agile-like** team structure will be used, where every members is responsible for their own tasks. While no concrete leader is set, **Andrew Taylor** will provide the necessary role of keeping up with individual contributions and making sure the project is on schedule. No formal code reviews or inspections will be utilized unless it is deemed necessary, either by consistently poor code or coding standards.

Documentation will be through **Doxygen.** Specific coding practices will follow the standards and conventions set in the **CodeConventions.txt** file on the project repository. The repository for the project will be used through **GitHub**.

**Requirements:**

These ATM operations are **REQUIRED** per the specification:

1) Withdraw money.

2) Deposit money.

3) Check account balance.

4) Transfer money to another account.

5) Authenticate account/PIN.

6) Print transaction receipts.

Use cases for the project to consider are as followed:

1) “Out of money” condition.

2) Unresponsive Bank authority.

3) “Out of paper” condition.

4) Other possible conditions to be reviewed.

**Architecture and Design:**

The three entities involved in this project – **Customer**, **Bank**, and **ATM** – will be implemented as individual classes responsible for their own conditions and status, based on their interfacing with one another.

The Bank entity will be responsible for keeping track of important information across **multiple** ATMs, such as customer accounts and authorization of customer pins. Data-driven information for the Bank will be fetched and retrieved via files. The Bank will receive references to individual ATM objects and communicate with them as needed.

The Customer entity will mostly take care of interfacing between the user and the ATM, by distributing events they perform to the ATM. A messaging system will be used to communicate to the ATM.

The ATM entity will react to Customer events and perform the operations the user desires, while also communicating with the Bank. All ATMs will be constructed with a reference to a Bank entity that they adhere to.

The GUI for the project will be implemented with the built-in Windows Forms in Visual C#, as C# will be the programming language used to implement the ATM simulator.

**Coding, Implementation, Build:**

**C#** will be the programming language used to implement the ATM simulator, with an object-oriented (OO) focus between entities. Visual C# can be utilized to create a GUI, which will act as the user interface for the ATM.

The project will be built in a **Visual Studio 2013** environment. All entities will be consigned to their individual interface files.

**Testing:**

Due to the simple nature and small scope of the project, automated testing or testing harnesses will not be used. **White-box testing** will be utilized primarily as the project’s main testing method, and will occur once the project is nearing completion as described in the delivery schedule.