**Project ‘NextGen Billing System’**

**Design Specifications Document**

**Version 1.0.0**

**October 23, 2013**

**Authors:**

David Ickert, Andrew Pohlmann, Chris Hauptli, and Zack Urben

**Table of Contents**

**1. TEAMWORK METHOLOGY…………………………………………………………………………...2**

**2. SOFTWARE ACHITECTURE SPECIFICATION…………………………………………………...2**

*2.1 DATA DESIGN……………………………………………………………………………….........2*

*2.2 COMPONENT DESIGN…………….…………………………………………………….........3*

*2.3 EXTERNAL INTERFACE DESIGN……………………………………………………………..3*

**3. DETAILED DESIGN SPECIFICATION ……………….……………………………………….........4**

*3.1 Open Screen / Account Listing View ………………………………………………….…4*

*3.2 Add / Modify Customer Account ………………………………………………………….5*

*3.3 View Meters …………………………………………………………………………………………6*

*3.4 View Meter Readings ……………………………………………………………………….....7*

*3.5 View Statements ……….…………………………………………………………………..…....8*

*3.6 View Outstanding Accounts (Report) …………………………………………………...8*

*3.7 Generate Bills(Job)………………………………………………………………………………..8*

*3.8 Process Payments (Job) …………………………………………………………………..……8*

**4. TEST PLAN ………………………………………..…………………………………………………..…….9**

**5. TABLE SCHEMA DIAGRAM ………………………………………………………………………….10**

**6. CLASSES DIAGRAM…………………………………………………………………….……………….11**

**7. UI FLOW MAP………………………………………………………………………………………..……12**

**I. Methodology for Your Team Work**

The team used in-person meetings and email as its main methodologies to coordinate and carry out the work of this phase. All team members were present for these meetings. During the meetings, discussion centered upon filling out the specific architecture specifications of the software product such as UML modeling of the classes as well as the HCI components of the software product GUI. Projectors and whiteboards were used to assist with this process. Dia was used to create the UML documents, while Visual Studio was used to create the HCI rapid prototypes.

**II. Software Architecture Specification**

The system is designed around the model, view, controller architectural paradigm.

**Data Design (See Section V for Table Schema Diagram)**

* **Accounts**
  + Contains essential account information of both past and current customers including name, billing address and kWH rates. The account will also keep track of its aggregate balance, which can be either positive (credit is due to the customer), negative (the customer owes money), or 0 (the customer’s balance is current). The aggregate account balance gets updated by bill generation and payments allocated. All accounts have a unique identifier.
* **Meters**
  + Contains meter specific information (physical address of the meter). Meters are linked to an account by the account’s unique identifier. The meter address is independent of the account billing address so that an account can be billed at a different address than where its’ meter resides. All meters have a unique identifier.
* **Meter Readings**
  + Contains the data for all meter readings specific to a meter. The meter is linked to both the meter it was read from, and the account it was applied to. The use of two foreign keys both minimizes duplicate records and allows for flexibility as a meter could move from one account to another. The meter reading will keep track of whether or not it has been billed, allowing the generate bill job to capture all un-billed readings, regardless of the account’s standard billing cycle. All meter readings have a unique identifier.
* **Bills**
  + Holds all data related to the specific instance of a bill, including total kWH, amount due, past due amount carried forward, account credits, and address it is to be mailed to. All bills have a unique identifier.
* **Payments**
  + Contains all payment information, which is received from the external payment system. A payment is assigned to an account and can be allocated to one or more bills. All payments have a unique identifier.
* **Payment to Bill allocations**
  + Links payments to bills by their unique identifiers. This allows for flexibility in payment allocation as bills could have more than one payment applied to them, and one payment could be applied to more than one bill.

**Component Design**

* **Components**
  + **Model**
    - The model is the data (accounts, meters, meter readings, bills, payments, and payment allocations) that the system is creating, updating or providing reports on.
  + **View**
    - The view is the graphical user interface that the users will log in to. It provides the views and components required for the user to interact with the data (model).
  + **Controller**
    - The controller is the interface between the user interface (view) and data (model) that contains the business logic required for the user to make updates to and retrieve information from the model.

**Class Design (See Section VI for Class Diagram)**

* **Classes**
  + **Database connection** 
    - Extends java.sql.statement that allows the system components to have a persistent connection to the database.
  + **Model controller** 
    - Implements a specific instance of the database connection object. It contains functions that allow models to be created and updated. It also retrieves record sets from the database tables to populate the user interface.
  + **Jobs** 
    - Runs tasks to generate or update models according to business logic/specifications (e.g. generate bills).
  + **Reports** 
    - Provides pre-defined sets of data to the user on request.
  + **User Interface** 
    - Classes that comprise the graphical user interface.

**External Interface Class Design**

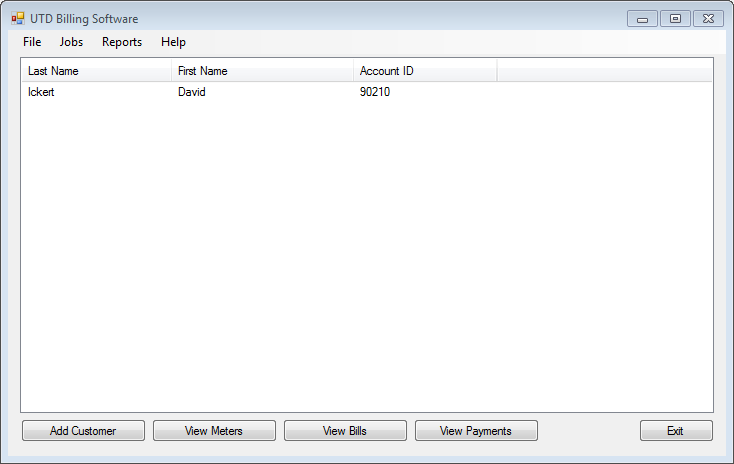
* **Payments**
  + Provides account payment information to the billing system. The information required is the account identifier, payment amount and date required. The payments will be process and applied to the account’s outstanding bills. Payment discrepancies will need to be reconciled in the external system. Reconciliations will get processed and applied to the respective account(s) when the next payment processing job is run.
* **External Bill Print Center** 
  + Delivers bill data to the external print interface so that it can be issued the account holder.
* **Meter Readings** 
  + Delivers meter reading information to the billing system. The meter readings will be used to generate bills when the generate bill job is run. Having an external meter reading system that is separate from the billing system allows for meter readings to be either manual (read by a person) or automatic (smart readers

**III. Detailed Design Specification**

The graphical user interface will open to an account listing that provides navigation as a gateway to the following use cases:

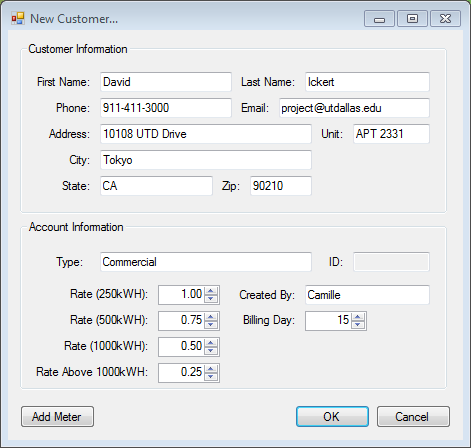
* + **Open Screen / Account Listing View**
  + **Add / Modify Customer Account**
  + **View Meters**
  + **View Meter Readings**
  + **View Statements (Bills)**
  + **View Outstanding Accounts (Report)**
  + **Generate Bills (Job)**
  + **Process Payments (Job)**

**Open Screen / Account Listing View**



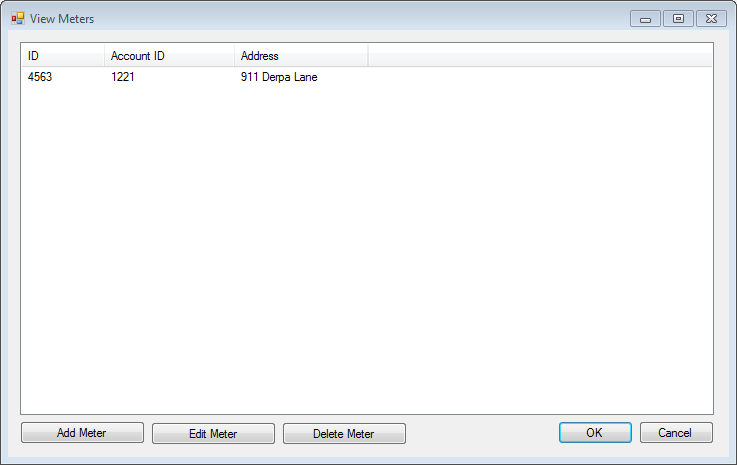
Allows the user to select a specific account and view its meters, bills and payments. It also allows the user to add an account. There is additional navigation to run both reports and jobs.

**Add or Modify Customer Account**



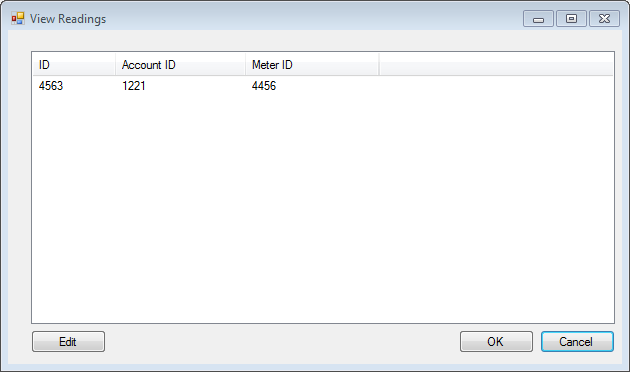
Allows the user to create a new account, or modify an existing account. The form will perform data validation to ensure that all accounts fields are populated before an account is created or modified.

**View Meters**



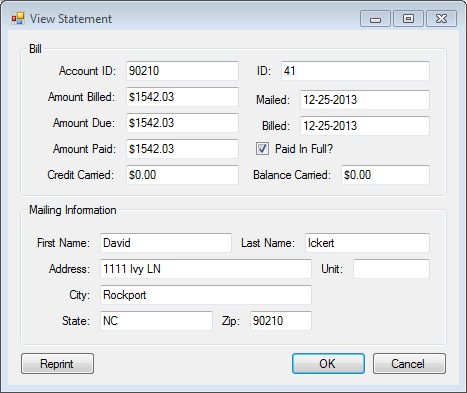
Displays all meters assigned to a specific account. Allows the user to add, modify and remove meters from the account. The form will provide validation to ensure that a meter already assigned to an account is not re-assigned to an account until it gets removed from its’ current account. There will be additional validation to ensure that new meters contain all of the required address fields for the meter.

**View Meter Readings**



Displays all of the meter readings associated with a specific account. If the account has multiple meters, all of the readings for its meters will be displayed in this view. The interface will allow the user to modify existing meter readings, as required by business needs.

**View Statements**



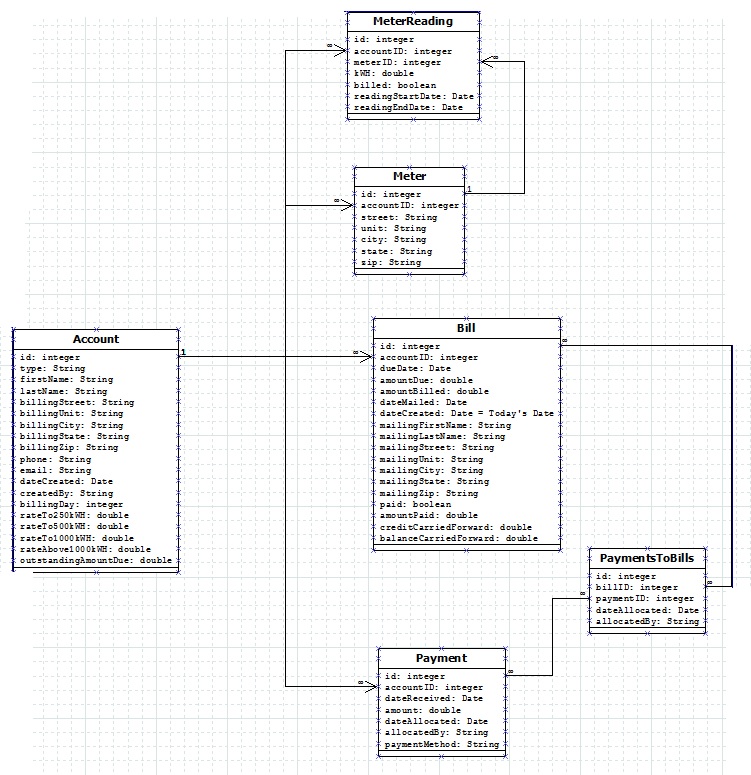
Provides a list of all bills generated for an account. Allows the user to pull up the details of a specific bill that will include: the address the billed was mailed to, the amount billed, date the bill was generated, amount due, balance carried forward (if any), and the credits carried forward (if any).

* **View Outstanding Accounts (Report)**
  + Provides a list of accounts whose aggregate outstanding balance is positive.
* **Generate Bills (Job)**
  + Creates a bill object for each account that has unbilled meter readings. The unbilled meter readings for each account are aggregated to create a bill record for the account, which is then output to a file for the external print function.
* **Process Payments (Job)**
  + Process unallocated payment records and applies them to bills based on an account’s oldest unpaid balance. The account’s aggregate balance is updated to reflect the new payment.

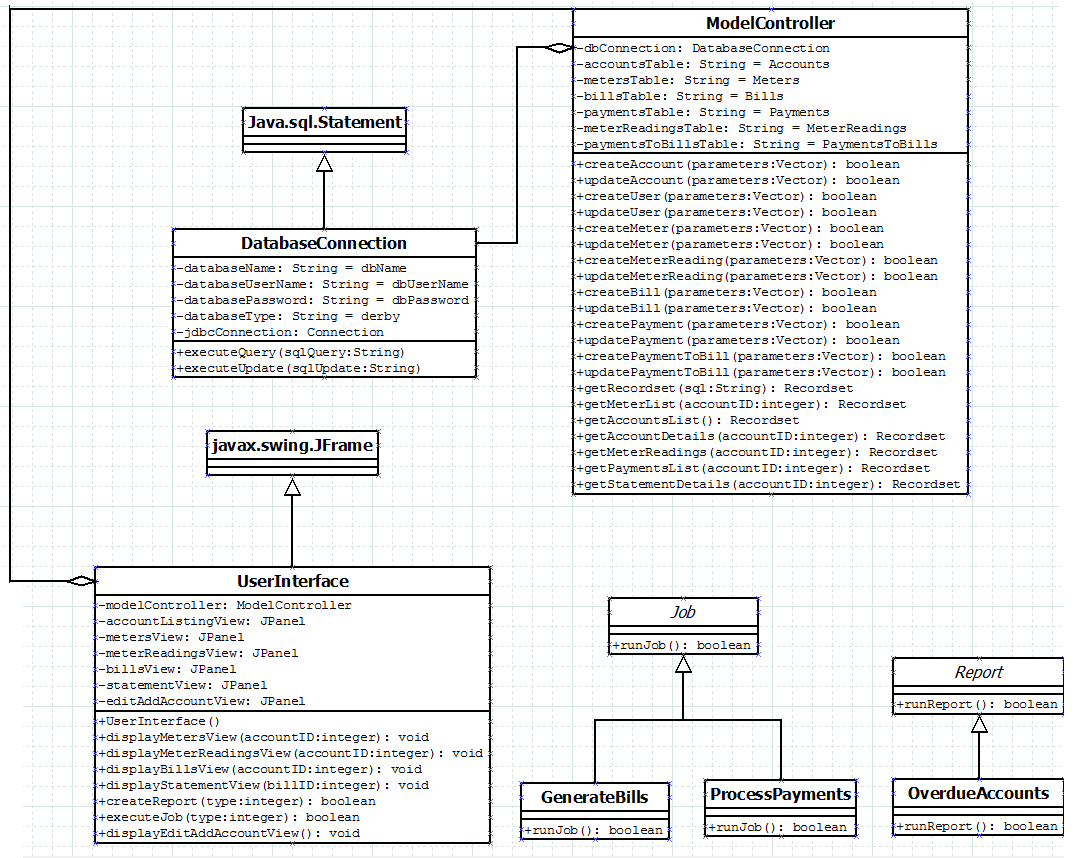
**IV. Test Plan**

* **Bill Generation**
  + Using pre-determined seed information, run the generate bill function and compare the output to manual calculations performed outside of the system.
* **Payment Processing**
  + Using pre-determined seed information, run the payment processing job and manually verify the information against business logic.
* **Account Creation / Updates**
  + Create sample accounts and verify that the information entered in the interface is identical to the information that appears in the database record. Modify sample accounts and confirm the updates in the database record.
* **Meter Creation / Updates** 
  + Create sample meters and verify that the information entered in the interface is identical to the information that appears in the database record. Modify sample meters and confirm the updates in the database record.
* **Meter Reading Interface Processing**
  + Work with the software maintenance team responsible for the external meter reading system to create sample data input for the billing system database. Manually verify the sample data against the database records that are created in the process.
* **Past Due Account Report Generation**
  + Using sample accounts with outstanding balances, run the report from the system and manually verify output against expected output.

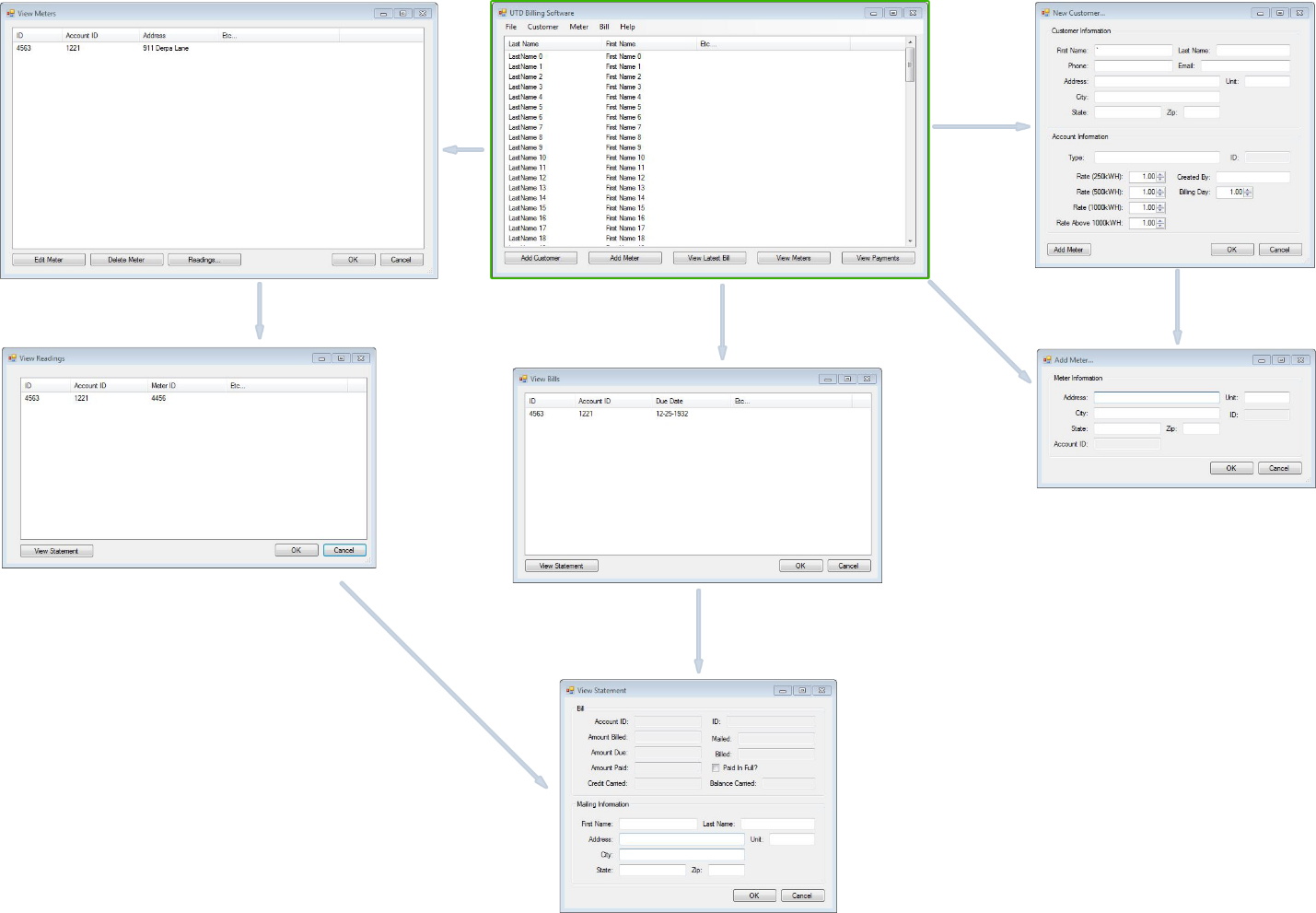
**Section V: Table Schema Diagram**



**Section VI: Class Diagram**

****

**Section VII: UI Flow Map**

****