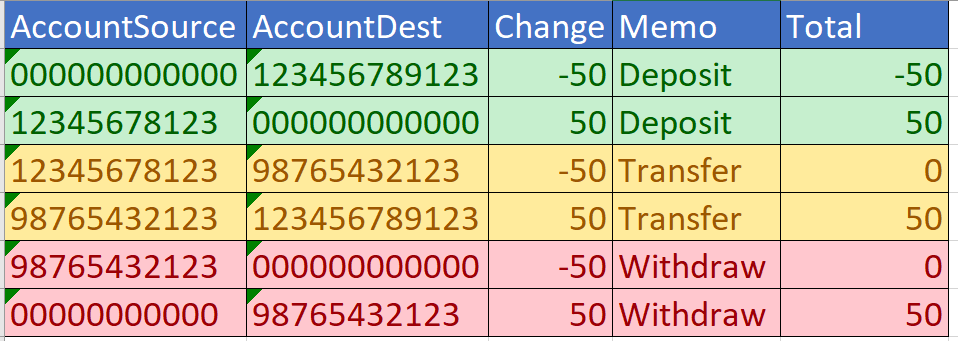
Harsh Patel

IT 202 Project Proposal

**Project Name: Simple Bank (Default Project)**

**Project Summary: This project will create a bank simulation for users. They’ll be able to have various accounts, do standard bank functions like deposit, withdraw, internal/external transfers, and creating/closing accounts.**

**Features:**

* User will be able to register a new account
* User will be able to login to their account provided they enter the correct credentials
* User will be able to logout (session will be destroyed)
* User will be able to see their profile
* User will be able to reset their password
* Passwords will not be stored/handled in plaintext
* User will not be able to access restricted pages
* Anonymous users will not be able to access private pages
* **[Start listing your project specific features here, follow the format above and keep each line concise. Think of anything that can be done in your app and write it here, this will be used as a checklist for development and for grading. I’ll review the list with you and narrow or broaden scope as necessary.]**
* All logic will be centered around a transaction table
* User will be able to login to their dashboard
  + User will be able to see a list of their accounts
  + User will see a menu of potential options
    - Create Account
    - Deposit
    - Withdraw
    - Transfer (Internal/External)
* User will be able to click one of their accounts to see more details about it
  + Current account value
  + Account Type
  + Transaction history of past x days
    - Should allow filtering based on
      * Date range
      * Transaction type
* User will be able to create a new account
  + Will choose account type (see later features giving more details on account types)
    - Checkings
    - Savings
    - Loan
* Account Types
  + Checkings
    - Min $5 deposit or transfer from existing account
  + Savings
    - Min $5 deposit or transfer from existing account
    - Set APY that calculates interest monthly and adds it to the total value
  + Loans
    - User picks which account to deposit to
    - User picks which account to withdraw payments from automatically
      * Enforce this at a monthly rate based on Interest rate
      * Account can go negative
* User will be able to deposit
  + Deposit will come from “world” account
  + User picks which account to deposit into
* User will be able to withdraw
  + Withdraw goes into “world” account
  + User can’t withdraw more than they have in the specific account
* User will be able to transfer funds between any accounts they own
  + Funds can’t exceed what the account has available
  + Can’t transfer negative value
  + Can’t transfer to a deactivated/deleted account
* User will be able to transfer funds to another user by the other user’s name and last 6 digits of their account number
  + Funds can’t exceed what the account has available
  + Can’t transfer negative value
  + Can’t transfer to a deactivated/deleted account
* User will be able to close an account
  + All funds must be transfered to an existing account or withdrawn
  + If a loan it must be paid in full
* Account Number Requirements
  + Should either be 10 characters or 12 characters long
  + “World” account should be “000000000000” (this is used for deposit/withdraw)
  + Each transaction must be recorded as two separate inserts to the transaction table
* Transaction Table Minimum Requirements
  + Each action on a set of accounts will be in pairs, noted by the colors below.
  + The first source/dest is the account that triggered the action to the dest account.
  + The second source/dest is the dest account's half of the transaction info.
  + Source/Dest are the account numbers affected.
  + Change is the difference in the account (deposit subtracts from source for the first part and adds to source for the second part.
  + Memo is either a built-in identifier to track the action (i.e., deposit, withdraw, transfer), and the total is the source accounts final value after the transaction.
  + The below Transaction/Ledger table should total up to zero to show that your bank is in balance. Otherwise, something bad happened with the transaction based on whether it's negative or positive. In that case we either lost money or stole money.
  + 

**Requirements:**

* Your application needs to handle a number of users
* Each user has a profile
* You application should have some extra data related to your user based on what you’re implementation goal is, for example: wishlist/shopping cart for E-Commerce site, player stats for a game, wins/losses for a game, posts for a blog/chat, etc
  + This data should follow CRUD (Create, Read, Update, Delete)
  + The site administrator(s) could potentially be the only ones with certain permissions such as delete, but your app should still support this functionality)
* You’ll be using MySQL backend (provided by NJIT AFS)
* Your SQL scripts should be stored on github
  + If using the init\_db sample keep your structural changes under a folder called “sql” and keep any data changes or queries under “queries”
* Your project will be hosted on your NJIT AFS site