CSC 230: Software Engineering

Department of Computer Science, Sacramento State University Spring 2015

Expense Tracker

Professor :Dr. Doan Nguyen

Team # 12:

Savleen Kaur

Arundhati Wahane

Table of Contents

ACKNOWLEDGEMENT

We would like to express our great gratitude towards our Professor, Dr. Doan Nguyen who has given us support and suggestions. Without their help we could not have presented this project up to the presentable standard. Thank you for all the new methodologies and industry project standards you taught in CSC230 - Software System Engineering course, Spring 2015.

Savleen Kaur

Arundhati Wahane

ABSTRACT

As the name itself suggests, this project is an attempt to manage our daily expenses in a more efficient and manageable way. The system attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure. Instead of keeping a dairy or a log of the expenses on the smartphones or laptops, this system enables the user to not just keep the tab on the expenses but also to plan ahead keeping the past budget in mind

With the help of this system , user adds , delete ,change the current entered bill entry efficiently and also can attach the bill image (optional) for his/her own reminder.

The graphical representation of the budget is the lucrative part of the system as it appeals the user more and is easy to understand and incorporate for future planning. The user interface of the system ticks the boxes of consistency, easy readable dialogue boxes, easy exit, feedback and easy to get used to requirements for any ideal user interface.

1. Introduction

1.1 Customer Statement of Requirements

1.1.1 Goals

The objective behind this solution is to design a refined system which will allow the user to efficiently manage his or her expenses with ease.

1.1.2 Problem Statement

At the instant, there is no as such complete solution present easily or we should say free of cost which enables a person to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations needs to be done by the user which may sometimes results in errors leading to losses. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

1.1.3 Proposed Solution

In an effort to fix the above addressed problems, we tried to design a system that would make the task of keeping the expenses in check, efficient and delight task.

This system will include a website application that will allow users to maintain a digital automated diary. Each user will be required to register on the website; at registration time, the user will be provided will a key id, which will be used to maintain the record of each unique user.

After logging into the system, a user can add the bills with an option to attach the image of the bill or not. The option to attach a bill helps the user to remember when and where the payment was made. The user can also add the information about how the payment was made i.e. via check, card or cash. The system also allows entering the check details. As soon as the entry is made about the expense, the database is updated and according to the nature of the bill deduction or addition to the total balance in the user's pocket is made.

In order to make the user aware about the average rate of the expenses, an alluring graphical statistics are also provided. In addition to the daily tracking of the expenditure, it is also beneficial to have a quick access to the past record of the expense habit; keeping in mind there is also an option of monthly budget to provide a quick glance of the past spending. There is also an option to view owe and lend expenses which adds or gets deducted from the overall budget according without bothering the user.

1.1.4 Key Points

- The data is very important asset for corporation, so strong authentication method is to be used to ensure security of information from malicious users.

- Easy to be customized in future, as the client demand some other additional features. The complexity of customer's company may be different or if mode of business changes then the system has capability to make appropriate modification to suite that change. Customization is key factor of designing this software.

- Accuracy of all type of calculations are important and to be achieved at any cost.

- The data retrieval and other manipulation related task which is done at the database level is fast enough.

- Storage of data is easily accessible.

2. Requirement Elicitation

2.1 Requirement Overview

- Report Generation: Each and every entry is logged into the database and a user can view monthly or weekly report as per the requirement.

- Adding or deleting an entry: A user can easily modify each and every bill entry along with an option of attaching an image of the entry.

- Graphical representation : A lucrative statistical of the budget is shown to the user for easy understanding

- Email notification: An email will be sent to the user at the end of each month giving a brief summary of the monthly expenditure.

- Export to excel: The user will be facilitated to download an excel format of the report generated.

2.2 System Requirements

2.2.1 Enumerated Functional Requirements

Identifier	Priority	Requirement

REQ-1: Dashboard_Panel	5	The system shall authenticate the user and then display panel based on the particular identified user
REQ-2: Add_Bill	5	The system shall allow the user to add bill details based on the user's need to track the type of expenses
REQ-3: Clear_Checks	4	The system shall allow to track check status for individual bill type
REQ-4: Back	4	The system should navigate the user back to dashboard screen
REQ-5: Add_Friends	5	The system should add other people for collaborating the expenses with the user
REQ-6: Add_Group	5	The system shall add group based on sharing and splitting the expense record with other friends
REQ-7: Exit_Group	3	The system should destroy the unwanted group
REQ-8: Expense_Planner	5	The system should graphically represent the current month figure based on user's current month expenses and user's own budget share
REQ-9: Expense_Tracker	5	The system should graphically represent the yearly expense numbers in form of report
REQ-10: Lent/Owe	5	The system shall track record of the people to whom the user owe or lent money
REQ-11: Download_PDF	4	The system shall allow user to track monthly expense report when in offline mode to track report of expenses

Note: 5 is rated as highest priority and 1 as least.

2.1.2 Enumerated Non-functional Requirements

Functionality	Priority	Requirement
Usability	4	• There is a consistency in all the modules and webpages • To ease the navigation there is a back tab to provide

		access to previous page • There is proper instruction on each page
Reliability	5	• Each data record is stored on a well-built efficient database schema • There is no risk of data loss • The internal evaluation of data is well coded
Supportability	4	• The system is well built to support any machine • Maintainability of the system is easy • All the plans for future augments can be easily incorporated within the present system
Performance	5	• In order to ease the accessibility, the types of expenses are categorised along with an option to name on the own • Throughput of the system is increased due to light weight database support
Availability	3	• The system is available all the time, no time constraint

2.1.3 Interface Requirements

1. User Interface:
   System users:
   The user logs on to the system by inserting username and password, and can edit details inside the database such as adding or deleting the entry.

2. Performance requirements

   Database interaction should not take more than 2 seconds.
   New frame should not take more than 3 seconds to open.
   The response time for menu changes will be not more than 3 seconds.
   All these above approximations are inculcated into the system by keeping the database non redundant and as optimised as possible.

3. Design constraints

   Back and exit buttons are provided on each page for sake of convenience
   The system works MVC architecture.
   Error messages will be displayed appropriately.
   Widgets like calculator and date picker is there to help the user.

3. Design

The design document that we develop during this phase is the blueprint of the software. It describes how the solution to the customer problem is to be built. Since solution to complex problems isn't usually found in the first try, iterations are most likely required. This is true for software design as well. For this reason, any design strategy, design method, or design language must be flexible and must easily accommodate changes due to iterations in the design. Any technique or design needs to support and guide the partitioning process in such a way that the resulting sub-problems are as independent as possible from each other and can be combined easily for the solution to the overall problem. Sub-problem independence and easy combination of their solutions reduces the complexity of the problem. This is the objective of the partitioning process. Partitioning or decomposition during design involves three types of decisions: -
Define the boundaries along which to break;
Determine into how money pieces to break; and
Identify the proper level of detail when design should stop and implementation should start.

Basic design principles that enable the software engineer to navigate the design process suggest a set of principles for software design, which have been adapted and extended in the following list:

A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job.
The design should be traceable to the analysis model. Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisfied by the design model.

The design should not repeat the same thing. Systems are constructed using a set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist.

The design should "minimize the intellectual distance" between the software and the problem as it exists in the real world. That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.

The design should exhibit uniformity and integration. A design is uniform if it appears that one person developed the entire thing. Rules of style and format should be defined for a design team before design work begins. A design is integrated if care is taken in defining interfaces between design components.

The design activity begins when the requirements document for the software to be developed is available. This may be the SRS for the complete system, as is the case if the waterfall model is being followed or the requirements for the next "iteration" if the iterative enhancement is being followed or the requirements for the prototype if the prototyping is being followed. While the requirements specification activity is entirely in the problem domain, design is the first step in moving from the problem domain toward the solution domain. Design is essentially the bridge between requirements specification and the final solution for satisfying the requirements.

The design of a system is essentially a blueprint or a plan for a solution for the system. We consider a system to be a set of components with clearly defined behaviour that interacts with each other in a fixed defined manner to produce some behaviour or services for its environment. A component of a system can be considered a system, with its own components. In a software system, a component is a software module.

The design process for software systems, often, has two levels. At the first level, the focus is on deciding which modules are needed for the system, the specifications of these modules, and how the modules should be interconnected. This is what is called the system design or top-level design. In the second level, the internal design of the modules, or how the specifications of the module can be satisfied, is decided. This design level is often called detailed design or logic design. Detailed design essentially expands the system design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding.

Because the detailed design is an extension of system design, the system design controls the major structural characteristics of the system. The system design has a major impact on the testability and modifiability of a system, and it impacts its efficiency. Much of the design effort for designing software is spent creating the system design.

The input to the design phase is the specifications for the system to be designed. Hence, reasonable entry criteria can be that the specifications are stable and have been approved, hoping that the approval mechanism will ensure that the specifications are complete, consistent, unambiguous, etc. The output of the top-level design phase is the architectural design or the system design for the software system to be built. This can be produced with or without using a design methodology. Reasonable exit criteria for the phase could be that the design has been verified against the input specifications and has been evaluated and approved for quality.

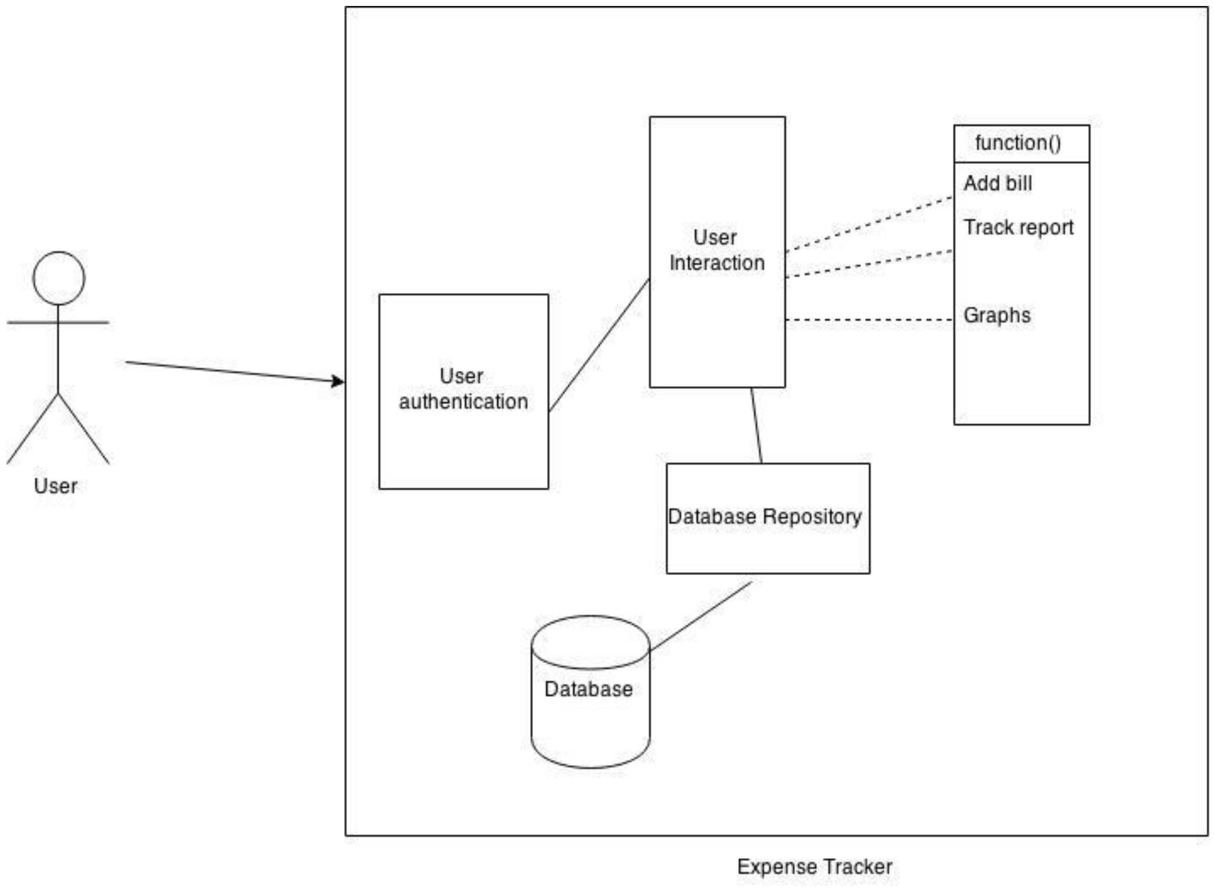
A design can be object-oriented or function-oriented. In function-oriented design, the design consists of module definitions, with each module supporting a functional abstraction. In object-oriented design, the modules in the design represent data abstraction (these abstractions are discussed in more detail later). In the function-oriented methods for design and describe one particular methodology the structured design methodology in some detail. In a function- oriented design approach, a system is viewed as a transformation function, transforming the inputs to the desired outputs. The purpose of the design phase is to specify the components for this transformation function, so that each component is also a transformation function. Hence, the basic

output of the system design phase, when a function oriented design approach is being followed, is the definition of all the major data structures in the system, all the major modules of the system, and how the modules interact with each other.
Once the designer is satisfied with the design he has produced, the design is to be precisely specified in the form of a document. To specify the design, specification languages are used. Producing the design specification is the ultimate objective of the design phase. The purpose of this design document is quite different from that of the design notation. Whereas a design represented using the design notation is largely to be used by the designer, a design specification has to be so precise and complete that it can be used as a basis of further development by other programmers. Generally, design specification uses textual structures, with design notation helping in understanding

We tried to abide out design by all the important aspects we discussed above and to make the system as realistic and efficient as possible. We designed each of our modules keeping all these principle as the basic.

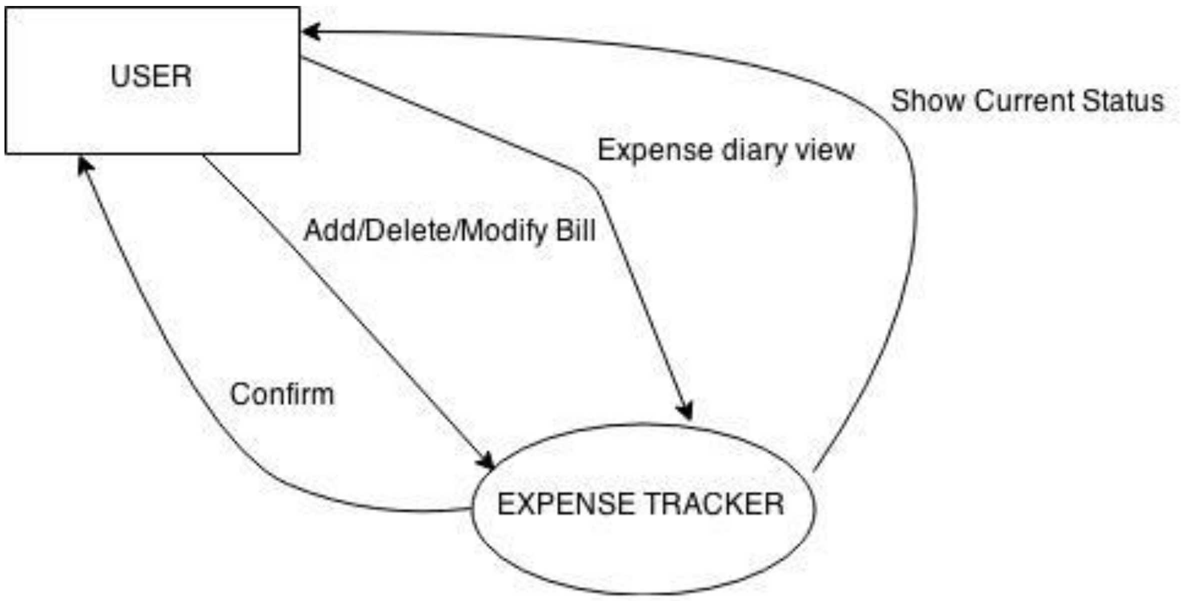
3.1 High Level Design

3.1.1 System Diagram



3.1.2 Data Flow Diagram

3.1.2.1 DFD Level 0



3.2 Functional Requirements Specification

3.2.1 Stakeholders

Stakeholders refer to the external entities that can access the system. In this proposed solution, everyone in any age group can have a very easy and efficient access to all the functionality of the system. From a college student to an entrepreneur can efficiently use this system.

3.2.2 Actors and Goals

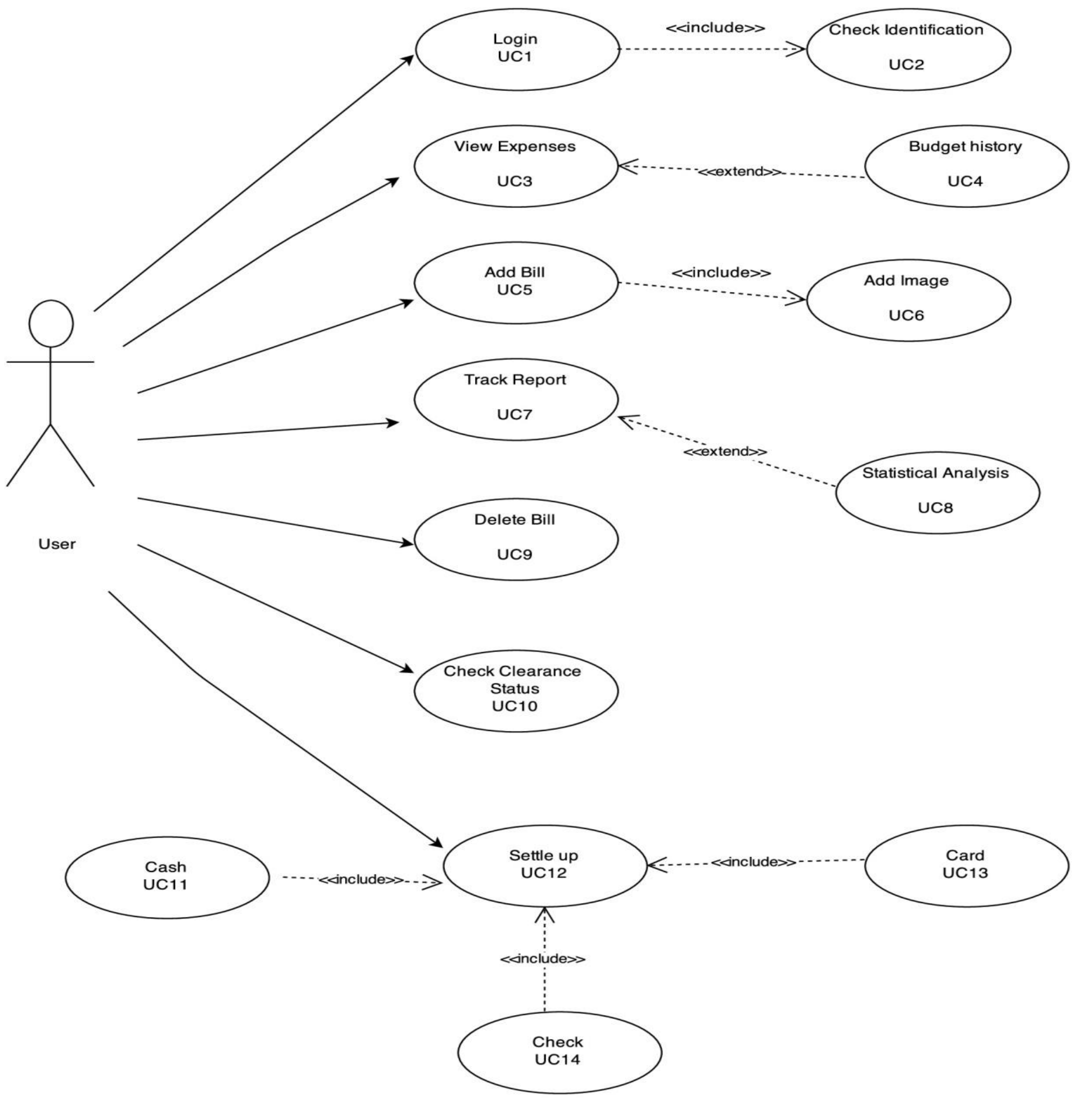
Actors	Goals
User	Securely log in to the system
	Add the expense, view the planner etc. and securely log out.

3.2.3 Casual Description

Use Case	Name	Description
UC1	Login	To provide identification details and enter the system
UC2	Check Identification	To validate the details entered by the user
UC3	View Expenses	Provide the updated log of expenses

UC4	Budget History	Provides the monthly expense details
UC5	Add Bill	To add a new expense
UC6	Add Image	To add image of the bill(optional)
UC7	Track Report	Displays the owe/lend record
UC8	Statistical Analysis	Internal scrutinised evaluation of the data
UC9	Delete Bill	Delete a false or incorrect bill
UC10	Check clearance status	To keep the track of cleared payment via check
UC11	Settle up	To settle up the lend/owe
UC12	Cash	Tracks the cash payment
UC13	Check	Tracks the check payment
UC14	Card	Tracks the card payment

3.2.4 Use Case Diagram



Use Case Diagram

3.2.5 Use Case Descriptions

UC-1

Use Case	Login
Related Requirements	Login id, password
Initiating Actors	User
Actor’s Goal	To login in the system and add bill or assess

	the expenses
Participating Actors	User(n numbers)
Pre-Condition	The user has a valid id and password
Post- Condition	• The user successfully logged in the system and can access all the functionality of the system • The user denied of the access
Main Success Scenario	Login was successful and now can view the dashboard screen
Flow of event	• User puts in wrong id and is denied login • User successfully moves on to use the dashboard functionality

UC-2

Use Case	Check identification
Related Requirements	Login id, password
Initiating Actors	User
Actor's Goal	To login in the system and add bill or assess the expenses
Participating Actors	User(n numbers)
Pre-Condition	The user has a valid id and password
Post- Condition	• The user successfully logged in the system and can access all the functionality of the system • The user denied of the access
Main Success Scenario	Login was successful and now can view the dashboard screen
Flow of event	• User puts in wrong id and is denied login • User successfully moves on to use the dashboard functionality • There might be a glitch will the credentials and user might be asked to reset them

UC-3

Use Case	View Expenses
Related Requirements	User account logged in
Initiating Actors	User
Actor's Goal	To view the statistical of the already input data
Participating Actors	User

Pre-Condition	The user is logged in
Post- Condition	• The user can view the detailed expenditure record
Main Success Scenario	User can view the history sheet of the expenses
Flow of event	• There is a redundant entry • An entry might be missed by the user • User may want to add or delete the expense

UC-4

Use Case	Budget History
Related Requirements	Account logged in
Initiating Actors	User
Actor's Goal	To view the history sheet
Participating Actors	User
Pre-Condition	The user is logged in
Post- Condition	• The user get the most recent updated log of the budget
Main Success Scenario	User has the full access to the recent log
Flow of event	• User may want to add new bill • User may want to delete a bill • There might be a missing entry or a redundant one

UC-5

Use Case	Add Bill
Related Requirements	Account logged in
Initiating Actors	User
Actor's Goal	To add a bill
Participating Actors	User
Pre-Condition	The user is logged in
Post- Condition	• New bill is added successfully • There might be a redundant entry
Main Success Scenario	The new bill added successfully
Flow of event	• Bill might be incomplete • Bill might be redundant

UC-6

Use Case	Add Image
Related Requirements	Account logged in

Initiating Actors	User
Actor's Goal	To add an image of the bill
Participating Actors	User
Pre-Condition	The user is logged in
Post- Condition	• Image to bill is added successfully
Main Success Scenario	The new bill added successfully
Flow of event	• Image format may be invalid • Size of the image may be large

UC-7

Use Case	Track report
Related Requirements	Account logged in
Initiating Actors	User
Actor's Goal	To track the report
Participating Actors	User
Pre-Condition	The user is logged in
Post- Condition	• Complete expenditure is displayed
Main Success Scenario	The new bill added successfully
Flow of event	• User may want to add new bill • User might want to view graphical representation

UC-8

Use Case	Statistical analysis
Related Requirements	Account logged in
Initiating Actors	User
Actor's Goal	Statistical analysis
Participating Actors	
Pre-Condition	The user is logged in
Post- Condition	• A complete expense sheet is provided to the user
Main Success Scenario	Expenditure history is view
Flow of event	• There might be error due to redundant entries

UC-9

Use Case	Delete Bill
Related Requirements	Account logged in
Initiating Actors	User
Actor's Goal	Delete Bill
Participating Actors	User
Pre-Condition	The user is logged in

Post- Condition	• Bill deleted successfully
Main Success Scenario	The log in the database is updated
Flow of event	• There might be unambiguous entries due to categories or name

UC-10

Use Case	Check clearance
Related Requirements	Account logged in
Initiating Actors	User
Actor's Goal	Check clearance status
Participating Actors	User
Pre-Condition	The user is logged in
Post- Condition	• Displays whether the check is cleared or not
Main Success Scenario	Display of the check status

UC-11

Use Case	Settle up
Related Requirements	Account logged in
Initiating Actors	User
Actor's Goal	Settle up
Participating Actors	User
Pre-Condition	The user is logged in
Post- Condition	• User eradicated the owed money to a friend by payment via cash or check
Main Success Scenario	The log in the database is updated
Flow of event	• User may want to enter the check details • User might have paid via cash

Note: Use cases 12, 13 and 14 are the extended functionality of the use case 11

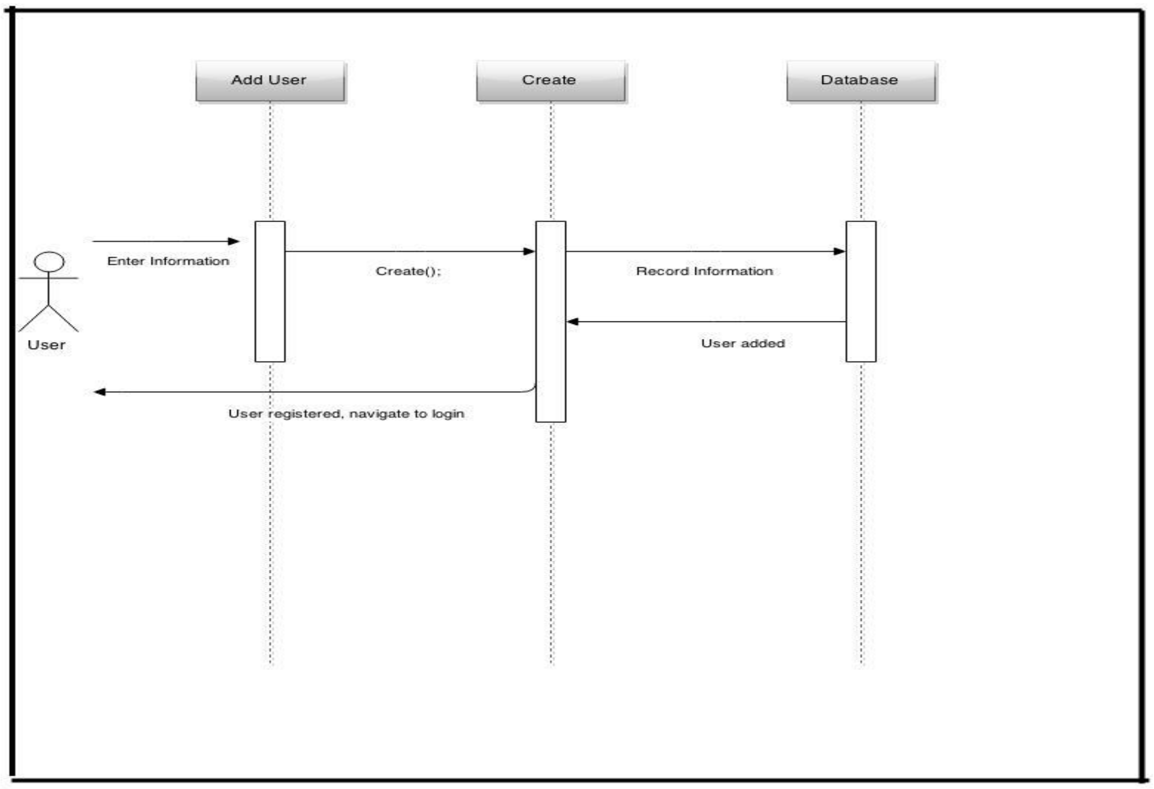
3.2.6 Traceability Matrix

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC 13	UC14
REQ-1	X	X	X	X	X	X	X	X	X					
REQ-2					X	X								
REQ-3										X	X			

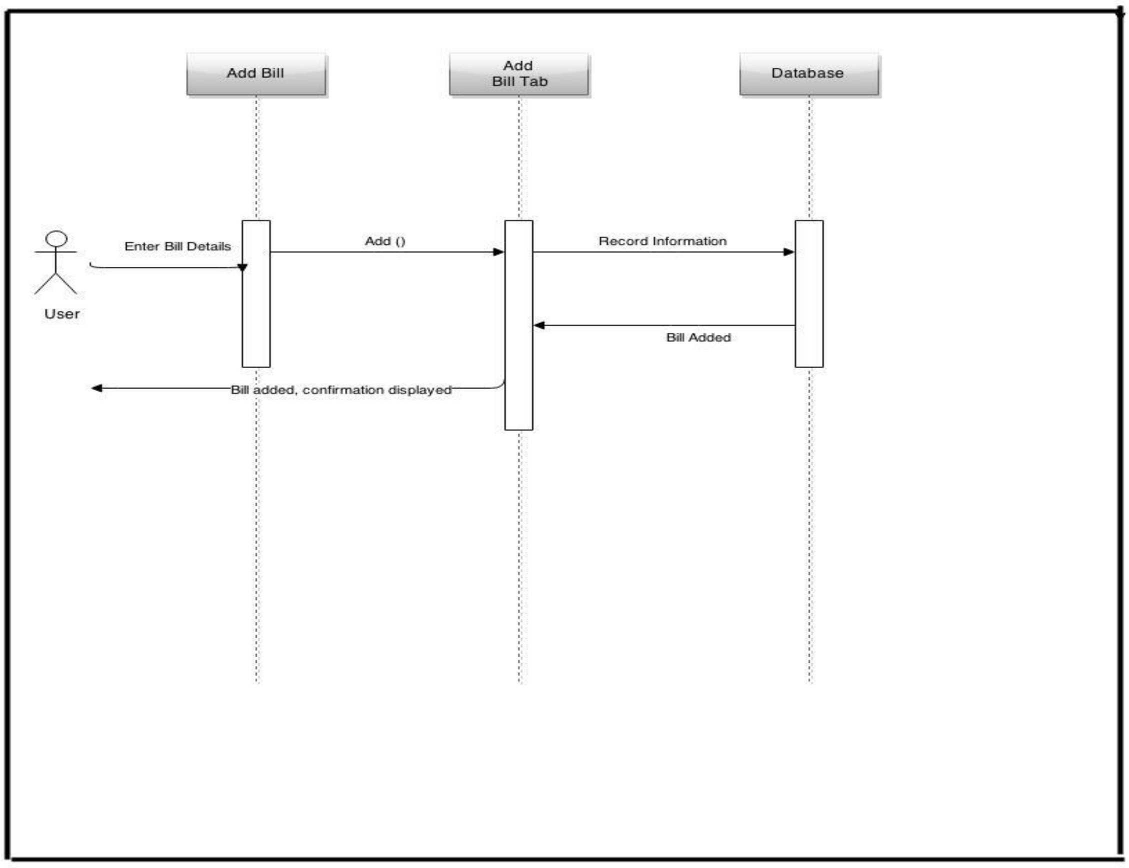
REQ-4			X	X	X	X	X	X	X	X		X		
REQ-5	X											X		X
REQ-6	X											X	X	
REQ-7	X											X	X	X
REQ-8			X	X			X	X						
REQ-9	X	X	X	X	X	X	X	X	X	X	X	X	X	X
REQ-10			X	X	X	X	X	X	X	X				
REQ-11							X	X						

3.2.7 Sequence Diagram

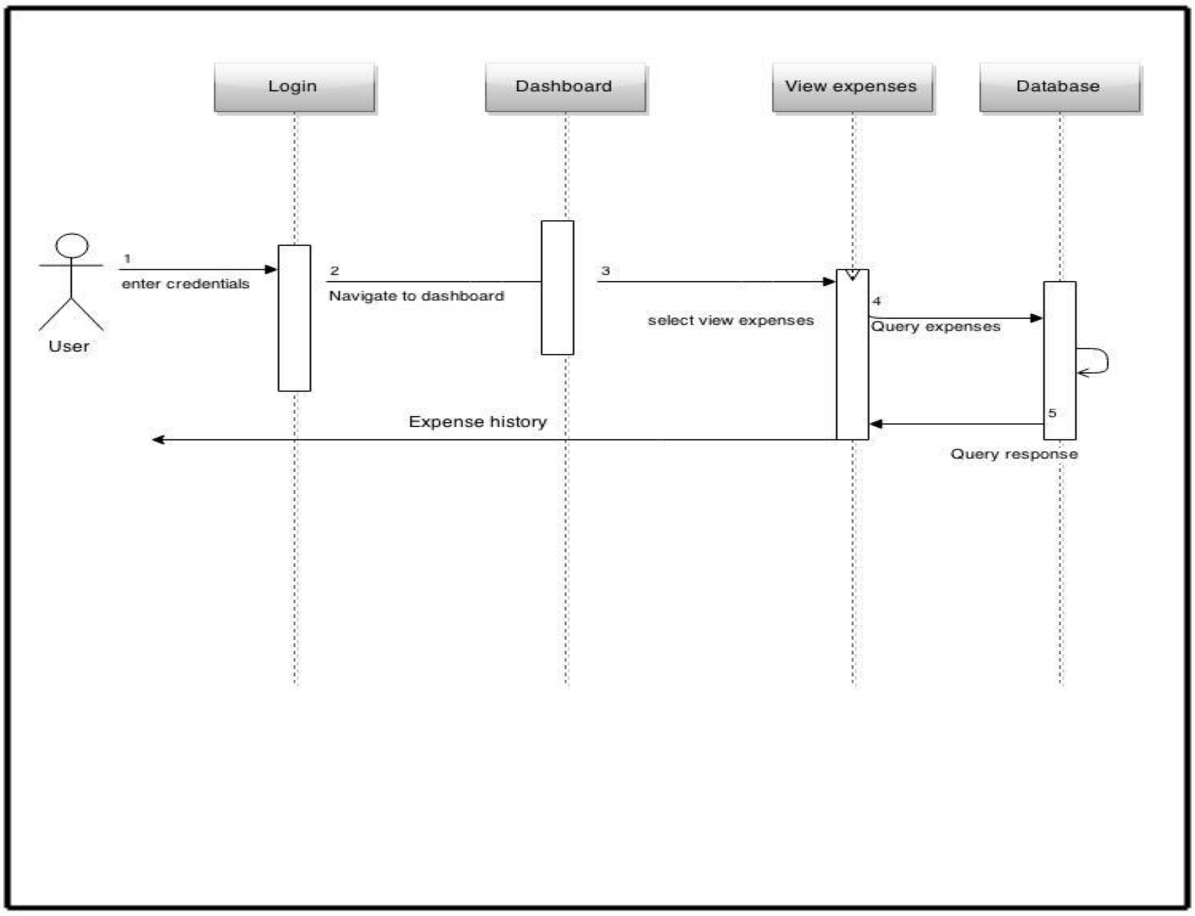
- Adding a New User



- Adding a New Bill



- View Expenses



4. Development

4.1 Modules

A modularization consists of well-defined manageable units with well-defined interfaces among the units.

Desirable property of modular system include

i. Each module is a well-defined sub-system.

ii. Single, well – defined purpose of each module.

iii. Modules can be separately compiled and stored in a library.

iv. Modules can use other module.

v. Modules should be easier to use than to build.

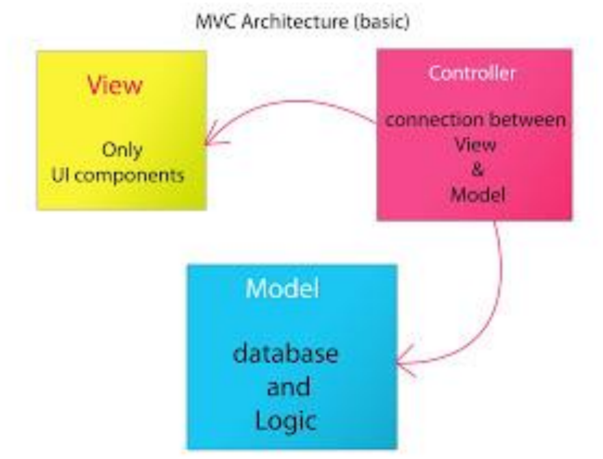
vi. Modules should be simpler from outside then from inside.

The project can be decomposed in following modules:

- **Login module**: This module is responsible for a registered user to login to the web application and do the proceedings.
- **Singup module**: This module is responsible for registering a new user to the web application and create a new account for him/her
- **Sessions module**: This module is responsible for creating a session when a user logs in and continues till he/she logs out.
- **Add Bill**: This module is responsible to enable the user to add a new bill
- **Delete the bill**: This module is responsible for the pre-defined bill.
- **View Expense**: This module is responsible for viewing all the expenses in detail added to the log by a logged in user
- **Edit** : This module is responsible for editing a pre-defined bill.
- **Mail module**: This module is responsible for sending mail to the users mail id when he/she creates a new account and when he/she forgets the password and wants to rest it.
- **Categories module**: This module is responsible for various options in the select expense type or city option tab.
- **Excel Sheet module:** This module generate the excel format report of the expense period requested by the user.

4.2 Tools/ Platform and Technologies

- The project is based on MVC architecture where the application is divided into three logical constituents-
  - View – Provide services such as user interface.
  - Controller – Implement business rules
  - Model – Provide handling and validation of data. (**SQL-SERVER** in this case)



View : It mainly comprises of the view component which the browser contains(application interface).

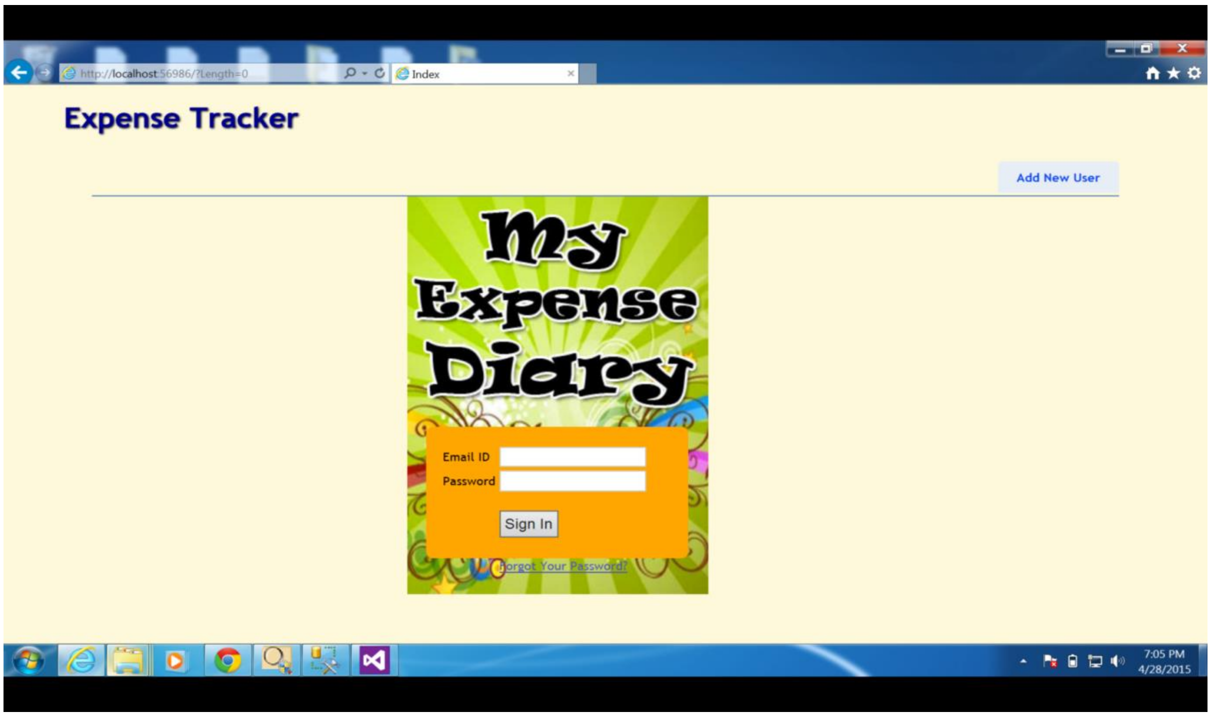
Controller: It mainly consists of the code and server(SQL Server Management Studio 2008) to maintain the connectivity between the view and model part

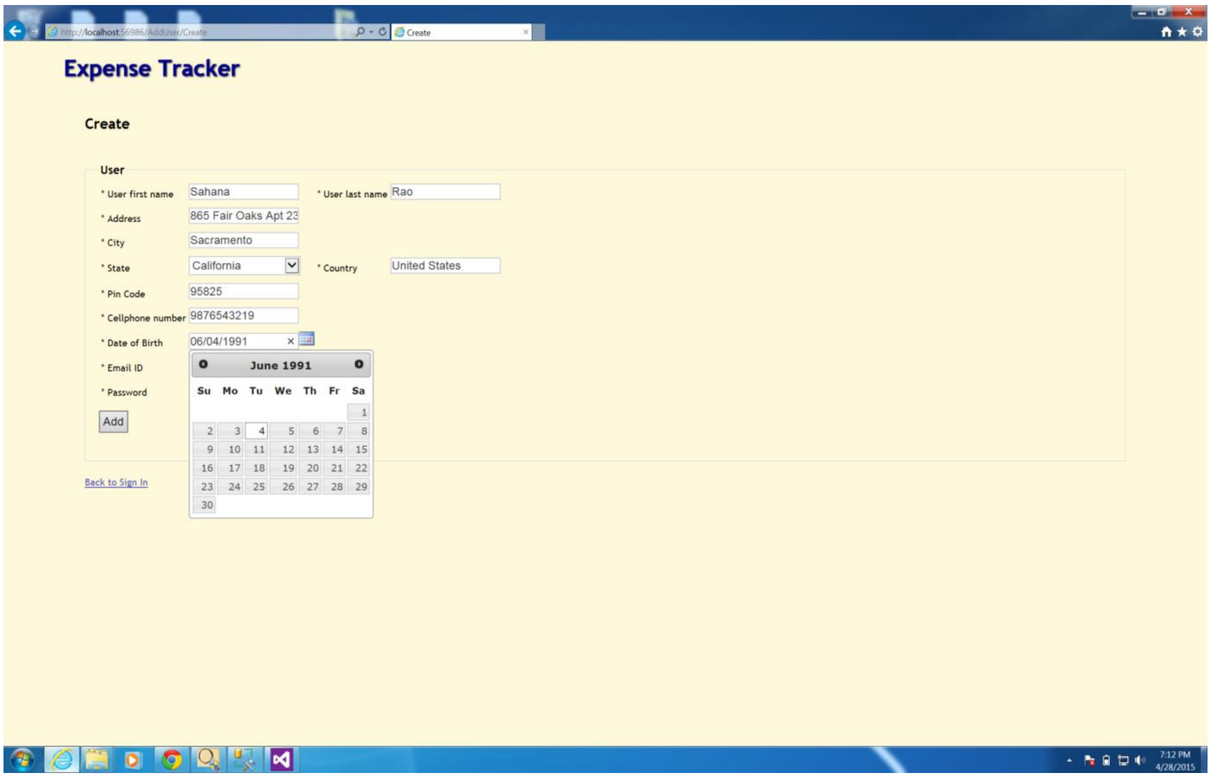
Model: it mainly consist of the database and the logic implied to store and fetch data

4.3 SOFTWARE & HARDWARE REQUIREMENTS

1. SQL SERVER Management
2. MINIMUM Internet Explorer 5.0 AND ABOVE, Google chrome, Mozilla Firefox etc.
3. Min. 1GB RAM

4.4 User Interface Screenshots





Create

User

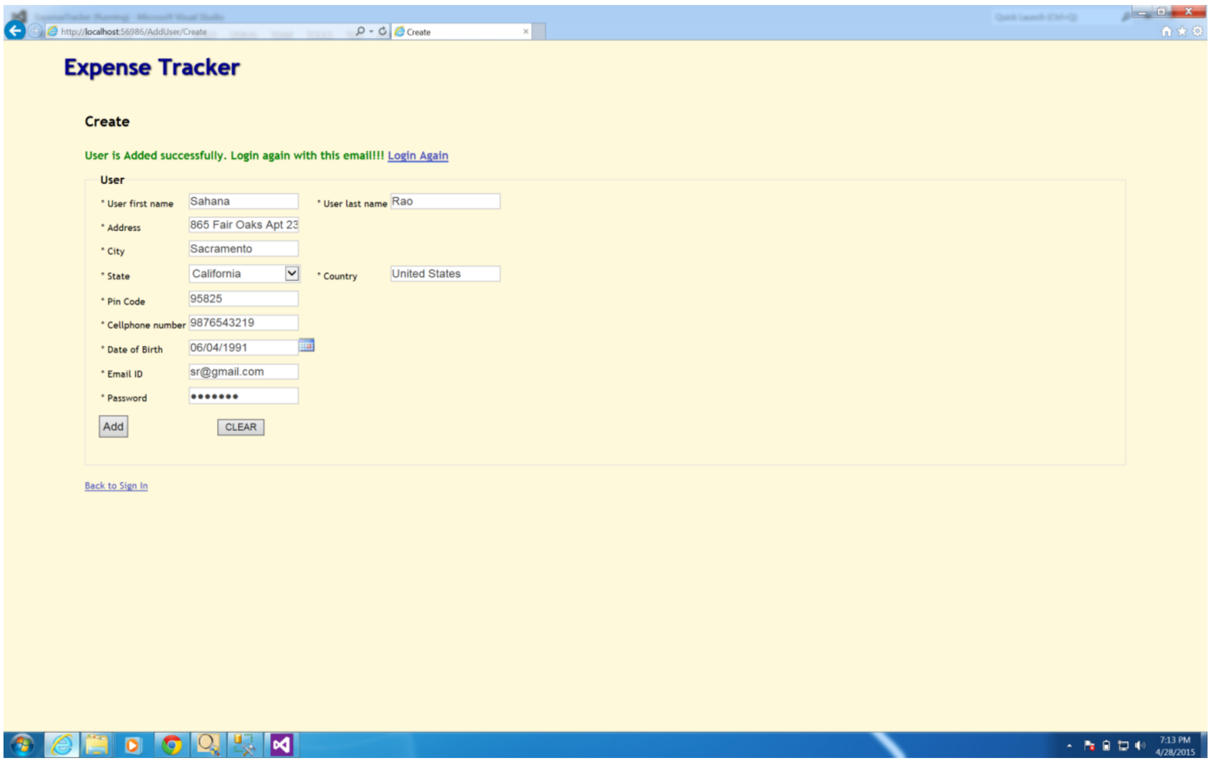
* User first name	Sahana	* User last name	Rao
* Address	865 Fair Oaks Apt 23		
* City	Sacramento		
* State	California	* Country	United States
* Pin Code	95825		
* Cellphone number	9876543219		
* Date of Birth	06/04/1991		
* Email ID			
* Password			

Add

Back to Sign In

June 1991

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

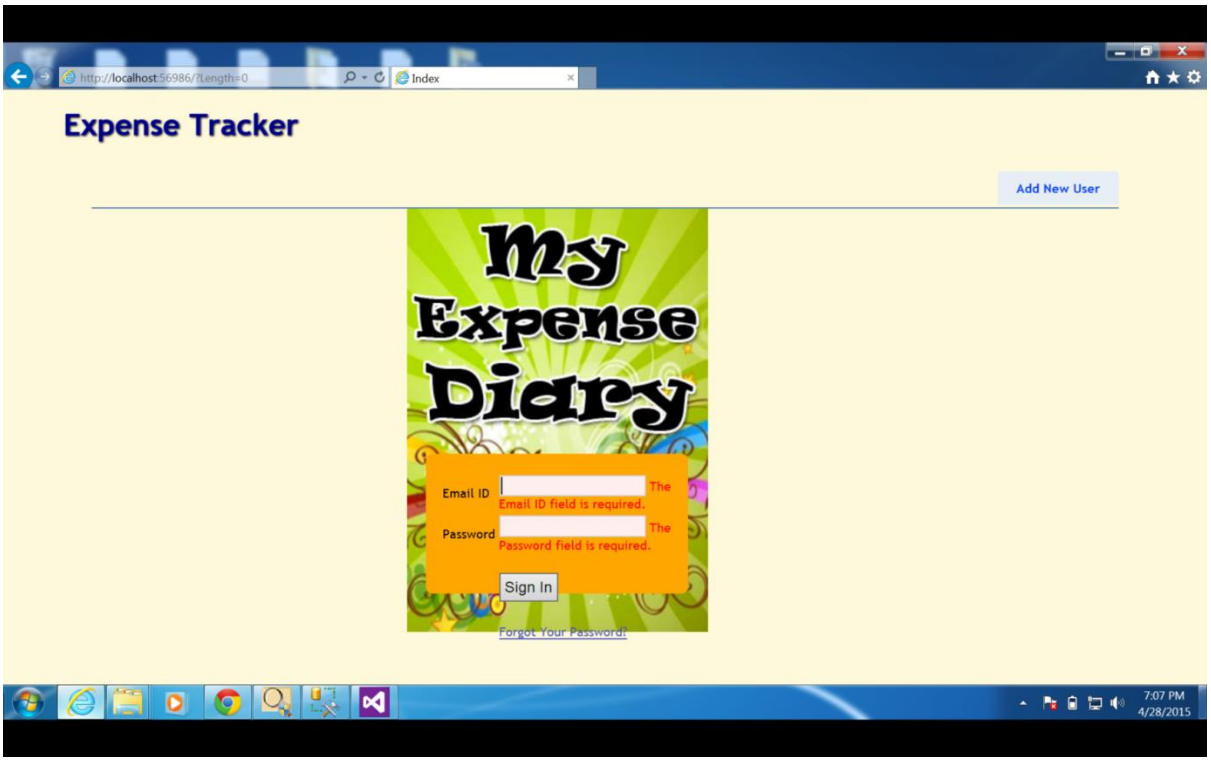

Expense Tracker

Create

User is Added successfully. Login again with this email!!! Login Again

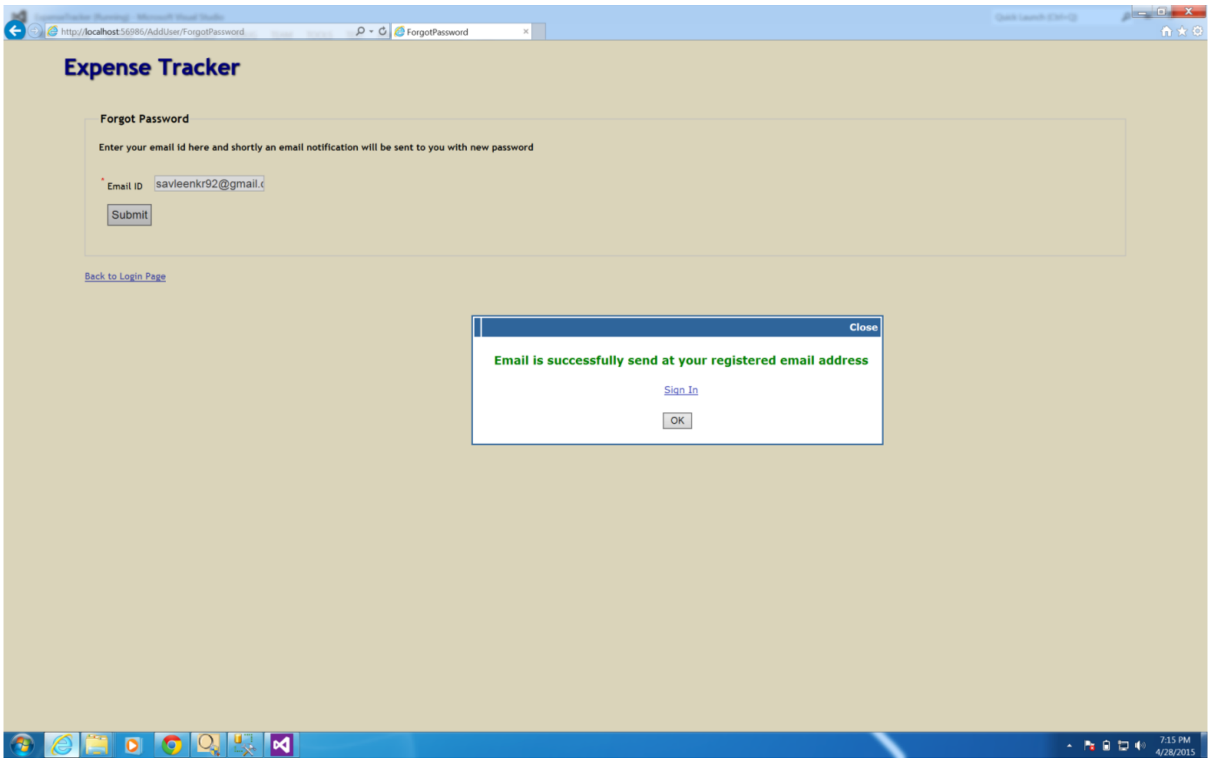
User

* User first name	Sahana	* User last name	Rao
* Address	865 Fair Oaks Apt 23		
* City	Sacramento		
* State	California	* Country	United States
* Pin Code	95825		
* Cellphone number	9876543219		
* Date of Birth	06/04/1991		
* Email ID	sr@gmail.com		
* Password	•••••••		


Expense Tracker

Email ID — The Email ID field is required.

Password — The Password field is required.



Expense Tracker

Forgot Password

Enter your email id here and shortly an email notification will be sent to you with new password

* Email ID savleenkr92@gmail.c

Submit

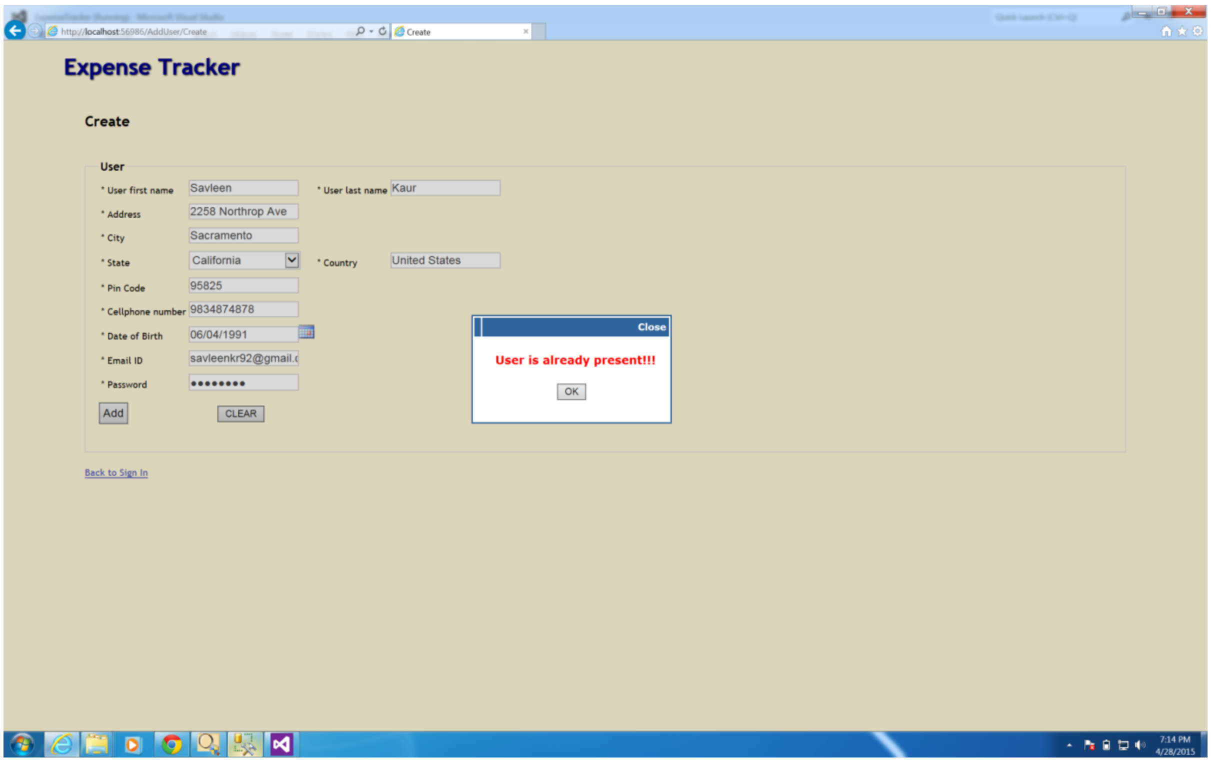
Back to Login Page

Close

Email is successfully send at your registered email address

Sign In

OK



Expense Tracker

Create

User

* User first name	Savleen	* User last name	Kaur
* Address	2258 Northrop Ave		
* City	Sacramento		
* State	California	* Country	United States
* Pin Code	95825		
* Cellphone number	9834874878		
* Date of Birth	06/04/1991		
* Email ID	savleenkr92@gmail.c		
* Password	••••••••		

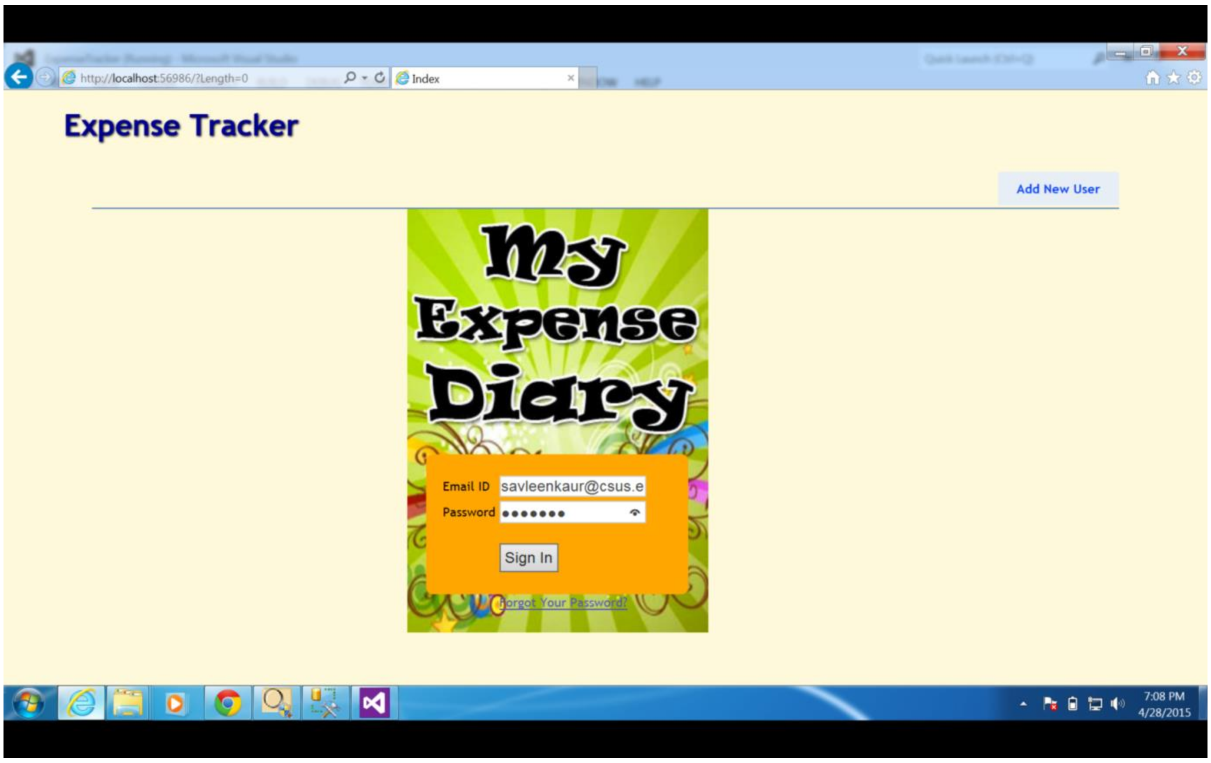
Add CLEAR

Close

User is already present!!!

OK

Back to Sign In


http://localhost:56986/?Length=0 — Index

Expense Tracker

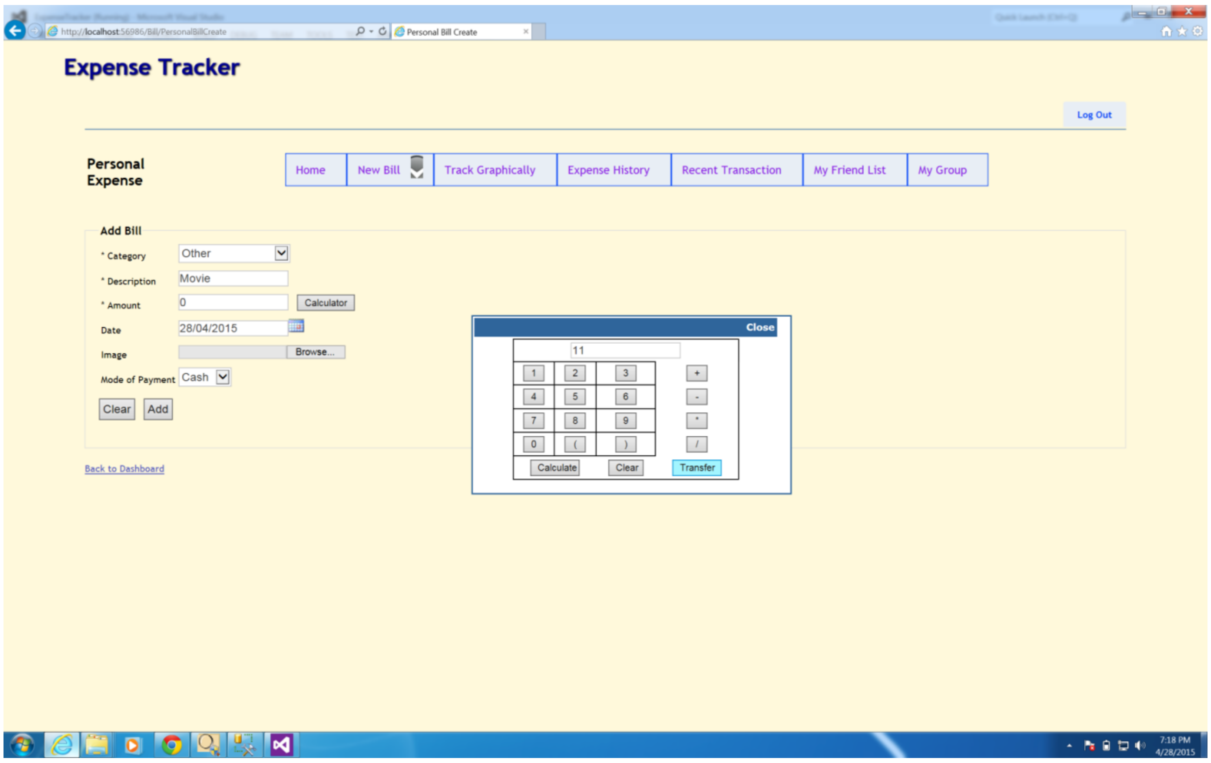
Add New User

My Expense Diary

Email ID	savleenkaur@csus.e
Password	•••••••

Sign In

Forgot Your Password?


http://localhost:56986/Bill/PersonalBillCreate — Personal Bill Create

Expense Tracker

Log Out

Personal Expense

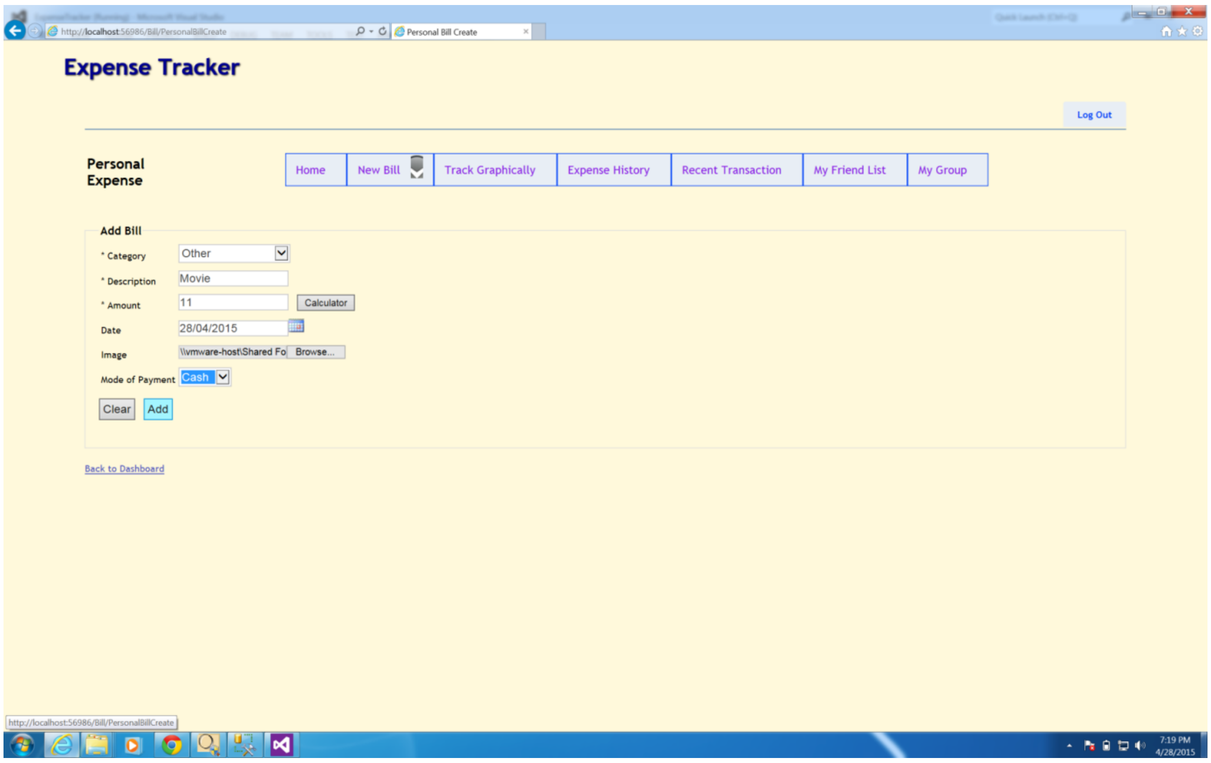
Home | New Bill | Track Graphically | Expense History | Recent Transaction | My Friend List | My Group

Add Bill

* Category	Other
* Description	Movie
* Amount	0
Date	28/04/2015
Image	
Mode of Payment	Cash

Clear Add

Back to Dashboard



Expense Tracker

Log Out

Personal Expense

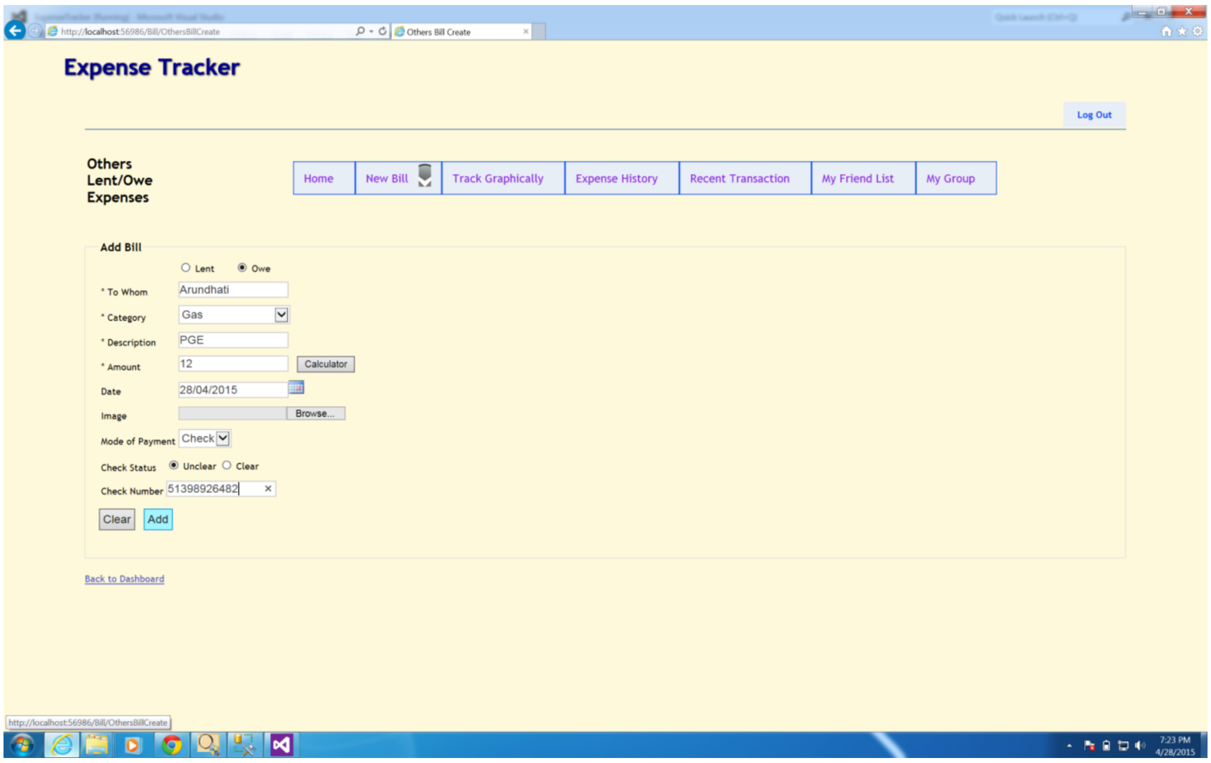
Home | New Bill | Track Graphically | Expense History | Recent Transaction | My Friend List | My Group

Add Bill

* Category	Other
* Description	Movie
* Amount	11
Date	28/04/2015
Image	\\vmware-host\Shared Fo
Mode of Payment	Cash

Clear Add

Back to Dashboard



Expense Tracker

Log Out

Others Lent/Owe Expenses

Home | New Bill | Track Graphically | Expense History | Recent Transaction | My Friend List | My Group

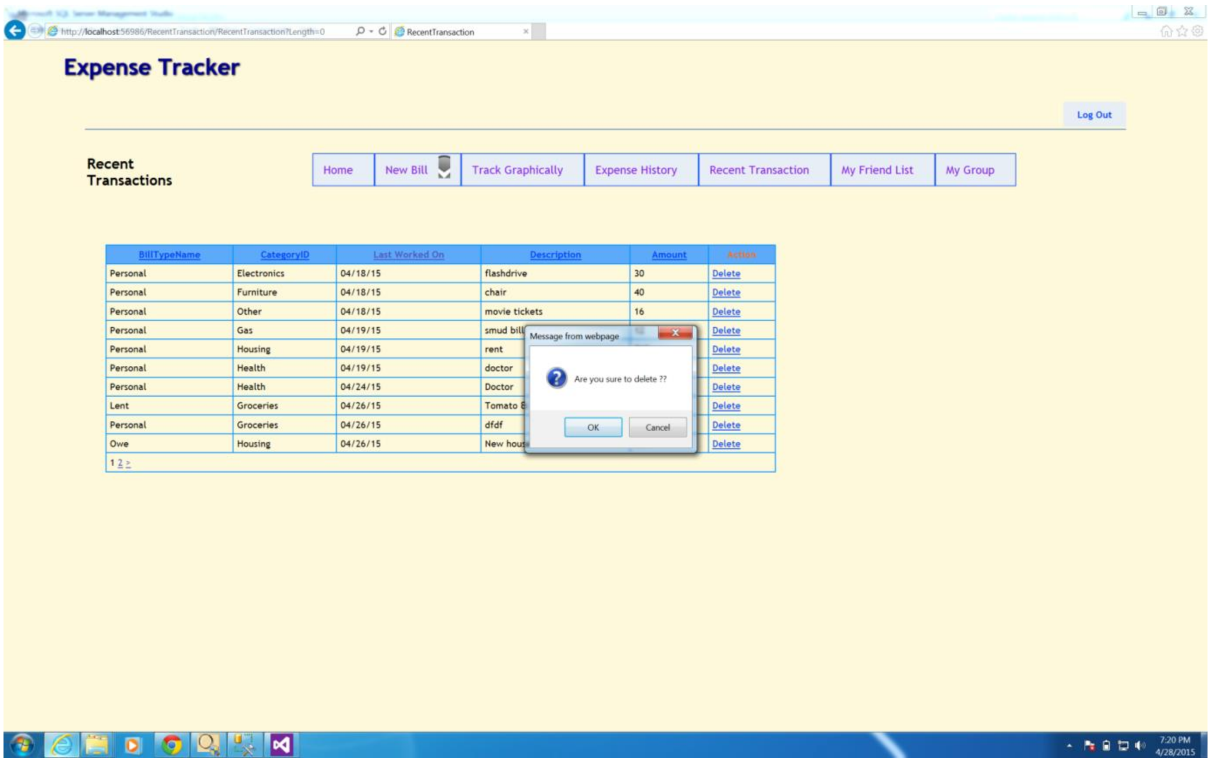
Add Bill

Lent [✓ Owe]

* To Whom	Arundhati
* Category	Gas
* Description	PGE
* Amount	12
Date	28/04/2015
Image	
Mode of Payment	Check
Check Status	[✓ Unclear] Clear
Check Number	51398926482

Clear Add

Back to Dashboard



http://localhost:56986/RecentTransaction/RecentTransaction?Length=0

Expense Tracker

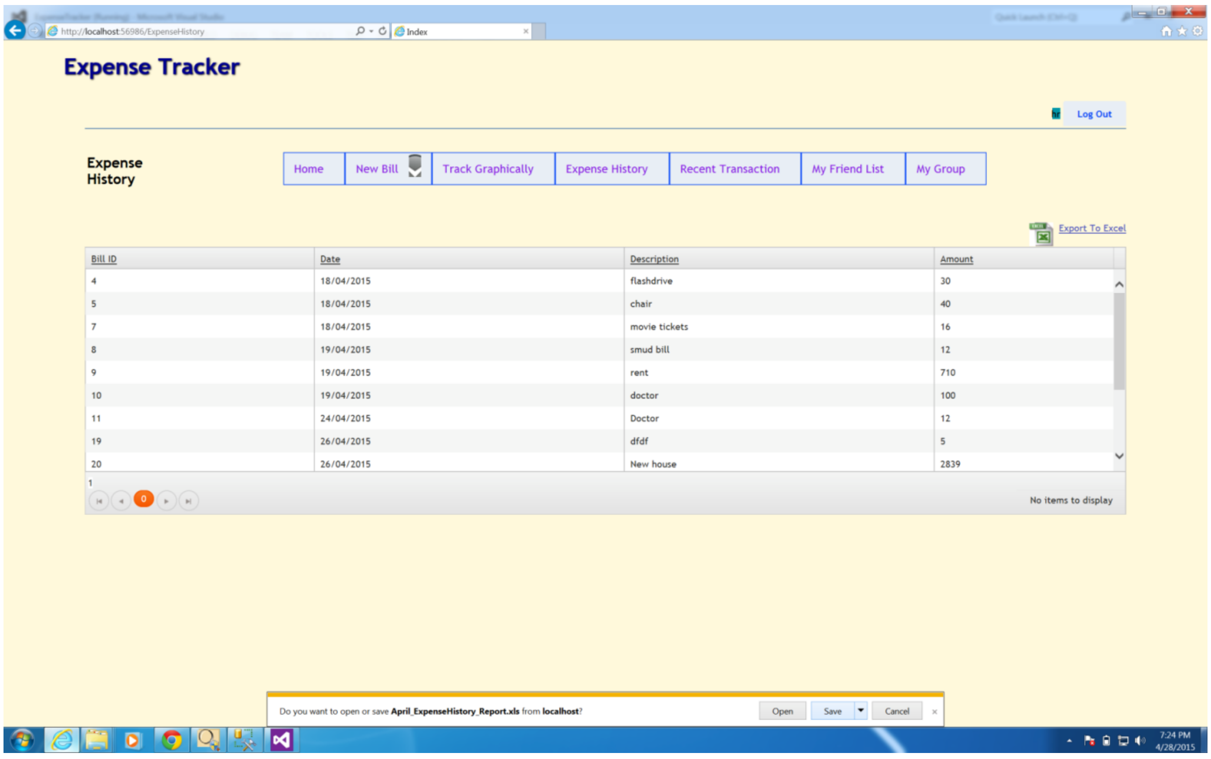
Recent Transactions

BillTypeName	CategoryID	Last Worked On	Description	Amount	Action
Personal	Electronics	04/18/15	flashdrive	30	Delete
Personal	Furniture	04/18/15	chair	40	Delete
Personal	Other	04/18/15	movie tickets	16	Delete
Personal	Gas	04/19/15	smud bill		Delete
Personal	Housing	04/19/15	rent		Delete
Personal	Health	04/19/15	doctor		Delete
Personal	Health	04/24/15	Doctor		Delete
Lent	Groceries	04/26/15	Tomato		Delete
Personal	Groceries	04/26/15	dfdf		Delete
Owe	Housing	04/26/15	New hous		Delete

1 2 >

Message from webpage

Are you sure to delete ??



http://localhost:56986/ExpenseHistory

Expense Tracker

Expense History

Bill ID	Date	Description	Amount
4	18/04/2015	flashdrive	30
5	18/04/2015	chair	40
7	18/04/2015	movie tickets	16
8	19/04/2015	smud bill	12
9	19/04/2015	rent	710
10	19/04/2015	doctor	100
11	24/04/2015	Doctor	12
19	26/04/2015	dfdf	5
20	26/04/2015	New house	2839

1

No items to display

Do you want to open or save **April_ExpenseHistory_Report.xls** from **localhost**?

5.0 Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

OBJECTIVES OF TESTING

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as-yet-undiscovered error.
- A successful test is one that uncovers an as-yet-discovered error.

5.1 Sample Equivalence Class test cases for login id and password

Test Case ID	Purpose	Test Cases	View following result
1)	Authentication	Maximum 10 character valid-id / available in database. (1)	ID accepted
2)	Authentication	Maximum 10 character valid-id / not available in database	Please enter valid details
3)	Authentication	ID with spaces in between	Please enter valid details
4)	Authentication	ID left blank	Please enter valid details
5)	Authentication	ID with special characters at beginning	Please enter valid details
6)	Authentication	ID with special characters in between	Please enter valid details
7)	Authentication	ID with length less than minimum length	Please enter valid details
8)	Authentication	ID with length more than 10 characters	Please enter valid details
9)	Authentication	ID with spaces at	Please enter valid

		beginning	details

Test Case ID	Purpose	Test Cases	View following result
1)	Authentication	Maximum 10 character valid password/available in database(staff)	Accepted
2)	Authentication	Password with blank spaces at beginning	Please enter valid details
3)	Authentication	Password with blank spaces at end	Please enter valid details
4)	Authentication	Password with special characters at any position	Please enter valid details
5)	Authentication	Password with blank spaces in between	Please enter valid details

NOTE: Please find the full test cases at the following link→ TEST_CASES

Data Dictionary

Name of data item	Data type	Description	Range of values
User ID	Alphanumeric (1)	Staff	All valid characters
Passwor d	Alphanumeric (staff)	Staff	All valid characters

5.2 QUANTITATIVE ANALYSIS OF THE SOURCE CODE

We used an online software analysis tool to calculate the efficiency and software metric of the system, and the following are the automate results shown when we input our source code files.

1. Total Mental Effort :

   E = 3811715 + 596754 + 372762 + 80145 + 111964 + 172774 + 1500366

   = **6646480 Elementary Mental Discriminations**

2. Halstead Software Development Time

   T = E / β (β = 18 EMD/Sec)

   = 6646480 / 18 = 369248.88 Sec

   = 102.57 hours

3. Total Program Volume

   V =18065+6939+6318+2055+4868+3526+10794

   = **52565 bits**

4. Total Lines Of Code (LOC)

   LOC = 301+198+144+146+132+101+99

   = **1121 LOC**

5. Total Program Difficulty

   $\underset{\wedge}{D}$ = 211+86+59+39+23+49+139

   $\underset{\wedge}{D}$ = 606

6. Program Level

   $\underset{\wedge}{L} = \underset{\wedge}{D}^{-1}$

   = 1 ÷ 606

   = 0.00165016501 ($0 < \underset{\wedge}{L} < 1$)

7. JAVA Language Level

   Halstead hypothesized that, if the programing language is kept fixed , as V* (Potential Program Volume) increases , $\underset{\wedge}{L}$ decreases in such a way that the product ($\underset{\wedge}{L}$ x V*) always remains constant. Thus this product, which he called the programing language level λ , can be used to characterize the programing language , asserting the efficiency of the code.

$\lambda = \underset{\wedge}{L} \text{ x } V^*$

$= \underset{\wedge}{L}^2 \text{x } V$

$= 0.00165016501\ ^2\ x\ 52565$

$=\mathbf{0.14313683787}$ (est. value 0.1434)

% deviation = $\frac{\mathbf{0.1434-0.14343683767}}{\mathbf{0.1434}}\ \boldsymbol{x}\ \mathbf{100}$ % **= 18.35 %**

8. <u>Code Efficiency</u>

ζ= 100 - % Deviation

= 100 – 0.1835 = **81.65 %**

6 Conclusions and future Scope

The system we envisioned comes to practise, providing a centralised log inculcating all daily expenses. No or less manual calculations are required by the user, with efficient and user friendly interface.

Followings are some future development plans

- Group: Apart from keeping a personal log, we are planning to extend this system to incorporate a shared expense group.
- Payment gateway: We are planning to include a service so as to make the direct cash payment within the application itself.
- Representing the interface multilingual

7 Lesson Learned from CSC 230

- Requirements Elicitation: As per articulation in the slides put on by the professor, we were able to fully explore all the viewpoints and aspects of our potential system use and we efficiently came up with the requirements that catered as the prime block of the project.
- Documentation: Professor kept on giving his valuable inputs from time to time regarding the documentation of the project which helped us to provide a precise and complete documentation to support our system
- Project planning and team work : Both the team members learned from each other while working on the project, which helped in better understanding of the project, knowledge sharing and better code.
- Introduction to Architectural Pattern(MVC): We were able to understand profoundly the concepts of the different architecture and were able to implement MVC pattern in our project development
- Software Design and Analysis: The most critical and toughest part of the development was made easy thanks to the slides by the professor.
- Testing Techniques: Both the team members efficiently tested the modules in iterative manner.

7 References

- Slides put up at SacCt by Dr. Doan Nguyen for CSC230.
- http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- http://www.sparxsystems.com/platforms/software_development.html