**SOCIETY MANAGEMENT SYSTEM**

SOFTWARE ENGINEERING PROJECT REPORT

[Submitted in partial fulfillment]

As a part of the curriculum of

B.SC. (H) COMPUTER SCIENCE

From



Mata Sundri College for Women, New Delhi

University of Delhi

Submitted by:

Priyanka Sharma (17044570021)

Sanjoli Jain (17044570042)

**B.Sc. (H) COMPUTER SCIENCE**

**IV SEMESTER**

Mata Sundri College for Women

Mata Sundri Lane, New Delhi 110002

**ACKNOWLEDGEMENT**

Any project is incomplete without guidance and motivation. We are very much thankful to our mentor ASHEMA HASTI, who has provided us the opportunity and motivation to gain knowledge throughout the course of this project.

We are also thankful to PRIYANKA GUPTA for providing us the help of computer laboratory, which is a valuable input resource for this project.

And last but not the least, we are thankful to all our friends, batchmates, and other people who have directly or indirectly helped us during the preparation of this project.

Thank you.

**CERTIFICATE**

This is to certify that the project entitled SOCIETY MANAGEMENT SYSTEM submitted by Sanjoli Jain and Priyanka Sharma from Mata Sundri College for Women in partial fulfilment of the requirements of BSc (H) Computer Science, IV semester.

It has been completed under the supervision of Ashema Hasti from Mata Sundri College for Women, Delhi University.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| SNO | TOPIC | Page No |
| 0 | ABSTRACT | 5 |
| 1 | PROBLEM STATEMENT | 6 |
| 2 | PROCESS MODEL | 9 |
| 3 | PROJECT SCHEDULING | 10 |
| 4 | TIMELINE CHART | 11 |
| 5 | CONTEXT LEVEL DIAGRAM |  |
| 6 | DFD LEVEL 1 |  |
| 7 | DFD LEVEL 2 |  |
| 8 | DATA DICTIONARY | 12 |
| 9 | SOFTWARE REQUIREMENTS  SPECIFICATION | 13 |
| 10 | PROJECT METRICS | 21 |
| 11 | COCOMO MODEL | 23 |
| 12 | RISK ANALYSIS | 25 |
| 13 | ER DIAGRAM |  |
| 14 | DATA DESIGN | 27 |
| 15 | USE CASE DIAGRAM | 33 |
| 16 | USE CASE DESCRIPTION | 34 |
| 17 | COMPONENT LEVEL DESIGN | 40 |
| 18 | TESTING | 41 |
| 19 | ANNEXURES | 43 |

**ABSTRACT**

Society Management System follows “Simplicity is the ultimate sophistication” phrase and aims at making the current situation in the society simple and efficient. This System reduces the conflicts that arise within the society by providing facilities such as online voting system, online hall allocation, online complain/suggestion registration in which society member can make complain form anywhere, any time and committee with the help of the system resolve the Complain as soon as possible, etc. It automates bulletin that occur within the society and makes it easy for the members of the society an easy access to the society happenings and on goings.

The main aim of the project is to provide utility to maintain day to day operations of the society .This system helps them to store all transactions electronically in a system, which in turn saves lot time, money and energy.

1.1 PROBLEM STATEMENT

To overcome the drawbacks of existing system, this project is proposing a smarter way of communication. As discussed earlier an automated notification system can be developed using a very popular Android platform which will provide a user friendly mobile based application

1.2 EXISTING SYSTEM (NOTICE BOARD SYSTEM):

In current situations, society management authorities use a traditional way of communication which includes a common notice board system operated by responsible society member. Many of societies also have started using automated chat systems which are definitely useful up to certain extent but though fail to provide reliable way of communication. Here are some basic disadvantages of these methods:

1.3 Disadvantages of Existing system:

**1.3.1 Unreachable information:**

Many of times it happens that an admin or notifier fails to update the notice board or society members ignore to follow the notice board. It results in lack of communication.

**1.3.2 Lack of authenticity and reliability:**

Notice board can be operated manually it might be handwritten or printed format. In general practice, it can be easily altered or misplaced and may lead to wrong interpretation. Also it can be affected with any unauthentic data as well.

**1.3.3 Time consuming activity:**

One has to take whole responsibility to operate and maintain the notice board. It creates dependability with specific person. Sometimes the person has to compromise with his own time schedule for these common activities.

**1.3. 4 missing of acknowledgement:**

As this is manual system one cannot predict, whether the notice is reached out to every concern person. And any kind of acknowledgment missing also creates overhead to reschedule or inform any kind of change.

1.4 Advantages of the proposed system:

**1.4.1 Multiple Reminders**

The application can provide gentle reminders through automated system till the actual event execution. It will reduce human efforts for member to member follow up.

**1.4.2 Authentic and uninterrupted communication with society members**

In this particular regard, it often seems that complaints by society members remain unattended by higher management and member has to suffer. The application can provide an assurance to follow up every complaint made by society member. Basically user can raise a complaint from any location irrespective of personal availability of concern person.

**1.4.3 Time saver and Go Green activity**

As this is a mobile based communication platform. Admin can push notices, create events, ask for complaints and many more things within a minute. It will bypass a long process of making a well-structured, presentable form of any kind of notice.

Besides being paperless activity, this will be a small helping hand for Go Green activity which will avoid use of eco- destructive products.

In short this application can be a very reliable and disciplined way of communication to keep a healthy bonding in between society members.

1.5. Functionalities of the proposed system:

The following are the modules included in this system:

1. Login

* Login will separate the residents committee members of the System.
* The monthly maintenance bill will be authenticated by committee members.
* Admin can also set the candidate for voting and allocation of hall.
* The committee members and members can login in the app and manage their profile.
* Committee has the special privileges of updating monthly bills.

1. Voting System

* The voting system allows members to vote the candidates that are standing for different positions in society.
* The candidates for different roles will be assigned by committee members as per the result.
* The result of voting will be declared on welcome page after login.

1. Bulletin and society wiki

* Shows information about the events that are to be happened in near future.
* Results of discussion, voting, etc.
* Urgent information from the committee members.
* Rules and Regulations about society’s facilities (such as parking, club house and any event being held in club house or ground).
* Schedules for cleaning staff timings, bills, etc.

1. Event Scheduling

* This module will be used for allocating club house or open ground for events. Committee can allocate the hall to the member by his end only.
* Before allocation, on a particular date from a member, it will be checked that there is no other event is planned from other member.
* Residents can ask permissions for funds, etc. from the committee.
* People can openly share their ideas for any events.
* Users can create special interest groups for communication.

1. Complaints And Suggestions Box

* Residents can report new complaints and suggestions regarding the society to the admin or secretary privately using the system interface.
* While the Committee members and Secretary has the rights to reply to the complaint and replies can be checked anytime.
* Committee members can also discuss the complaints or suggestions on the main society page hiding the name of the sender for these.

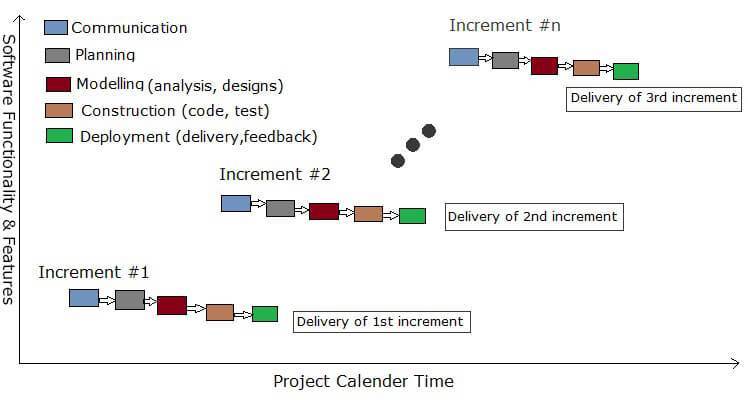
1. Accounting

* The member can view his previous month bill status on their profile.
* The members can set a reminder for their maintenance bill submission date.
* The committee members can confirm the member status by just one click and it will be automatically reflect in the database.
* If the bill is not paid on or before the specific fixed date, the required action will be taken.
* They do money transaction from their account to society’s account via online payment.
* Funds can also submit via online payment.

**PROCESS MODEL**

SOCIETY MANAGEMENT SYSTEM follows Incremental process model.

The incremental model combines elements of linear and parallel process flows. The incremental model applies linear sequences. Each linear sequence produces deliverable increments of the software in a manner that is similar to the increments produced by an evolutionary process flow.



The first increment is a core product in which basic requirements are addressed but many supplementary features remain undelivered. The core product is used by the customer. As a result of use/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

Incremental model is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project. In addition, increments can be planned to manage technical risks.

**WHY WE ARE USING INCREMENTAL PROCESS MODEL**

* The requirements of the complete system are clearly defined and understood such as visitor management, photo gallery, special interested groups, managing users, parking slot, etc.
* Major requirements are defined; however, some details can evolve with time such as maintaining contacts.
* There was compelling need to complete our project within 4 months.
* Resources with needed skill set are not available
* There are some high risk features and goals

## PROJECT SCHEDULING

### Time-Line Chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Work Tasks** | **January** | | | | **February** | | | | **March** | | | | **April** | | | |
|  | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 |
| Problem  Statement |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |
|  |
| Process Model |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |
|  |
| SRS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |
|  |  |
| ERD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | |
|  |  |  |
| Data  Dictionary |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |
|  |
| Context  Level Diagram |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |
|  |
| DFD  (Level-1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |
|  |  |
| DFD  (Level-2) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |
|  |
| Use Case  Diagram |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |
|  |
| Use Case  Description |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |
|  |  |
| Function  Point Metrics |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |
|  |  |
| COCOMO  Model |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |
|  |  |
| Risk  Analysis |  |  |  |  | | | | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |
|  |  |

**Project Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Work** | **Planned start** | **Actual Start** | **Planned Complete** | **Actual complete** | **Assigned Person(s)** | **Effort Allocated** |
| Problem Statement | Jan,w1 | Jan,w1 | Jan,w2 | Jan,w2 | Priyanka Sharma,  Sanjoli Jain | 2 pw |
| Process model | Jan,w2 | Jan,w3 | Jan,w3 | Jan,w3 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Project Scheduling | Jan,w4 | Jan,w4 | Jan,w4 | Jan,w4 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Software Requirement specification | Jan,w3 | Jan,w3 | Feb,w1 | Feb,w1 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Entity Relationship Diagram | Feb,w2 | Feb,w2 | Feb,w2 | Feb,w2 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Data  Dictionary | Feb,w2 | Feb,w2 | Feb, w3 | Feb,w3 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Context  Level Diagram | Feb,w3 | Feb,w4 | Feb,w4 | Feb,w4 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Data Flow Diagram(le vel 1) | Feb, w4 | March,w1 | March,w1 | March,w2 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Data Flow Diagram  (Level 2) | March w2 | March,w3 | March,w4 | April,w1 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| User case  Description | March,  w4 | March,w4 | March,w4 | March,w4 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Functional  point Metrics | April,w1 | April,w1 | April,w1 | April,w1 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Effort estimation using COCOMO  model | April,w2 | April,w2 | April,w2 | April,w2 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |
| Testing | April,w2 | April,w3 | April,w3 | April,w3 | Priyanka Sharma,  Sanjoli Jain | 2 p-w |

## DATA DICTIONARY

The data dictionary is a repository of various data flows. Precise structure of data is not specified by dfd so; it specifies structure of each data flow in DFD. A data dictionary can specify:

* Data Elements
* Data Flows
* Data Stores
* Data Structure

A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in this software project. It describes the meanings and purposes of data elements within the context of this project, and provides guidance on interpretation, accepted meanings and representation so that user and analyst will have a common understanding of inputs, outputs and other project components.

1. Response\_committee = [Complains] | [Suggestion] |[Event\_permission]

2. Response\_residents = [Post\_candidate\_offer] | [Event\_query]

3. Notification\_resident = Event\_Permission + Complaints\_response + Suggestions\_response + voting’s\_result + Bulletin + Post\_Candidate\_offer + fund details + Maintance\_Bill

4. Updated\_information= Post\_Candidate\_offer + Event\_details + Result\_voting

5. User\_name = {legal\_character}\*

6. Password = {legal\_character + digit + special\_ch}\*

7. Digit = [0-9]

8.Special\_ch= [@|$|#]

**SOFTWARE REQUIREMENTS SPECIFICATIONS**

**1. INTRODUCTION**

This document gives detailed functional and non-functional requirements for the Society Management System. This software will support online society activities.

**1.1PURPOSE**

To overcome the problems occurring in the time lagging manual system, an automated system needs to be developed .The problem was that in existing system, daily notices, monthly meetings, cultural events, miscellaneous contacts for daily needs, security alerts, high priority communication and many others which may not be conveyed properly in current scenario as most of the things are getting handled manually. It lacks transparency. ­­­­

**1.2SCOPE**

1. **Login** -Residents have user id and password to login into the portal. System verifies input from the residents database .System sends error message when authentication fails and in case of success allow authenticate users to login the system.

2. **Voting**-residents votes for the appropriate candidate. System generates result. System sends result to the committee.

3. **Bulletin and society wiki**-Committee updates data. System stores updated data into society database. System sends notification to the residents.

4**. Event Scheduling**-residents give the event details. Committee processes those details. System sends approval/rejection message to the residents.

5. **Complaints And Suggestions Box**-residents register complaints/Suggestions. Committee addresses those details. System sends response message to the residents.

6. **Accounting**-Residents/Committee Fund details, Payment. System generates maintenance bill and address in online payment. System sends maintenance bill message to the residents.

­­­­

**1.3 ABBREVIATIONS AND ACRONYMS**

SRS-Software Requirement Specification

DD-Data Dictionary

DFD-Data Flow Diagram

PW-Person Week

C&S-Complain and Suggestion

PG-Payment Gateway

OPayment- Online Payment

RName-Respondent Name

PS-Product Size

BU-Business Impact

PD-Process Definition

CU-Customer Characteristics

DE-Developer Environment

ST-Staff size and experience

**1.4REFERENCES**

1. “Software Engineering- A practitioner’s approach (7th edition)” by Roger S. Pressman.
2. <http://www.ijera.com/papers/Vol4_issue2/Version%201/CF4201547551.pdf->

Abstract and problem statement

1. <http://www.people.cs.ksu.edu/> -SRS
2. <http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/>

Incremental Model

**1.3 OVERVIEW**

The application can provide gentle reminders through automated system till the actual event execution. It will reduce human efforts for member to member follow up.

In particular regard, it often seems that complaints by society members remain unattended by higher management and member has to suffer. The application can provide an assurance to follow up every complaint made by society member. Basically user can raise a complaint from any location irrespective of personal availability of concern person.

Besides being paperless activity, this will be a small helping hand for Go Green activity which will avoid use of eco- destructive products.

In short this application can be a very reliable and disciplined way of communication to keep a healthy bonding in between society members.

**2. PROJECT DESCRIPTION**

**2.1PRODUCT PRESPECTIVE**

* **Committee Members**  can login, view own bill information, perform online payment, add fund details ,update various details, register complain/suggestion, also addresses the response for the same, provide response to the event details, perform online voting, update results of voting, etc.
* **Society Members** can login, view own bill information, perform online payment ,view fund details, view update details ,register complain/suggestion, also view the response for the same, provide event details, view response of the event details, perform online voting ,view results of voting, etc.

**2.1.1** **SYSTEM INTERFACES**

This system does have one interface with online payment gateway of the existing systems.

**2.1.2 SYSTEM SPECIFICATIONS**

**2.1.2 .1** **HARDWARE REQUIREMENTS**

* Hardware Disk-20 GB.
* Memory-8Gb

**2.1.2.2 SOFTWARE REQUIREMENTS**

* **Operating system:** Any operating system will work to give best support and user friendliness.
* **Database:** to save the records of the society members and their details, SQL database is used.

**2.2PRODUCT FUNCTIONS**

1 Login

* Login will separate the resident and committee members of the System.
* The monthly maintenance bill will be authenticated by committee members.
* Admin can also set the candidate for voting and allocation of hall.
* The committee members and members can login in the app and manage their profile.
* Committee has the special privileges of updating monthly bills.

2 Voting System

* The voting system allows members to vote the candidates that are standing for different positions in society.
* The candidates for different roles will be assigned by committee members as per the result.
* The result of voting will be declared on welcome page after login.

3 Bulletin and society wiki

* Shows information about the events that are to be happened in near future.
* Results of discussion, voting, etc.
* Urgent information from the committee members.
* Rules and Regulations about society’s facilities (such as parking, club house and any event being held in club house or ground).
* Schedules for cleaning staff timings, bills, etc.

4 Event Scheduling

* This module will be used for allocating club house or open ground for events. Committee can allocate the hall to the member by his end only.
* Before allocation, on a particular date from a member, it will be checked that there is no other event is planned from other member.
* Residents can ask permissions for funds, etc. from the committee.
* People can openly share their ideas for any events.
* Users can create special interest groups for communication.

5 Complaints and Suggestions Box

* Residents can report new complaints and suggestions regarding the society to the admin or secretary privately using the system interface.
* While the Committee members and Secretary has the rights to reply to the complaint and replies can be checked anytime.
* Committee members can also discuss the complaints or suggestions on the main society page hiding the name of the sender for these.

6 Accounting

* The member can view his previous month bill status on their profile.
* The members can set a reminder for their maintenance bill submission date.
* The committee members can confirm the member status by just one click and it will be automatically reflect in the database.
* If the bill is not paid on or before the specific fixed date, the required action will be taken.
* They do money transaction from their account to society’s account via online payment.
* Funds can also submit via online payment.

**2.3-USER CHARACTERISTICS**

Committee Members can login, view own bill information, perform online payment, add fund details ,update various details, register complain/suggestion, also addresses the response for the same, provide response to the event details, perform online voting, update results of voting, etc.

Society Memberscan login, view own bill information, perform online payment ,view fund details, view update details ,register complain/suggestion, also view the response for the same, provide event details, view response of the event details, perform online voting ,view results of voting, etc.

**2.4 GENERAL CONSTRAINTS-**

* The information of all the society members must be stored in a database that is accessible by the committee members.
* The Society Management System is running all 24 hours a day.
* The society members can access the Society Management System from any smartphone or laptop that has an internet connection.
* The society members must have their correct username and passwords to enter into the Society Management System.

**2.5 ASSUMPTIONS AND DEPENDENCIES**.

**Assumption**-All the users have their own unique id and password.

**Dependency**-Internet connection is must.

**3. SPECIFIC REQUIREMENTS**.

**3.1-EXTERNAL INTERFACES.**

**3.1.1 USER INTERFACES**

The users will work on a graphical user interface. All the required steps for the user to perform must me specified on each page so that it is easy for users to understand the procedure to perform. The steps must be brief so that the content is not a trouble. The screen should be formatted in such a way so that instructions and messages always get displayed in a general area.

**3.1.2. HARDWARE INTERFACE.**

Various interfaces for the product could be

1. Touch screen/Monitor
2. Keypad
3. Continuous battery backup

**3.1.3. SOFTWARE INTERFACE**

1. Any operating system.
2. JavaScript is required for highly interactive environment.

**3.2FUNCTIONAL REQUIREMENTS**

1. Login

INPUT-user id and password

PROCESS-System verifies input from the resident’s database

OUTPUT-System sends error message when authentication fails and in case of success allow authenticate users to login the system.

2. Voting System

INPUT-votes

PROCESS-System generates result.

OUTPUT-System sends result to the committee.

3. Bulletin and society wiki

INPUT-Updated data

PROCESS-System stores updated data into society database.

OUTPUT-System sends notification to the residents.

4. Event Scheduling

INPUT-Event details

PROCESS-Committee processes those details.

OUTPUT-System sends approval/rejection message to the residents.

5. Complaints and Suggestions Box

INPUT-.Complaints/Suggestions

PROCESS- Committee addresses those details.

OUTPUT-System sends response message to the residents.

1. Accounting

INPUT-.Fund details, Payment

PROCESS- System generates maintenance bill and address in online payment.

OUTPUT-System sends maintenance bill message to the residents.

**3.3. PERFORMANCE REQUREMENTS**

**3.3.1 SECURITY:-**

The Society Management System is fully accessible to only authentic user. It requires username and password for entry to a new society member.

**3.3.2. RELIABILITY:-**

The application is highly reliable and it generates all the updated information in correct order.

**3.3.3. AVAILABILITY:-**

Any information should be quickly available from any computer to the authorized user.

**3.3.4. MAINTAINABILITY:-**

The application is maintainable in such a manner that if any new requirements occur then it should be easily incorporated in an individual module.

**3.3.5. PORTABILITY:-**

The application is not machine specific.

**3.4 DESIGN CONSTRAINTS:**

* **Software Language Used-**

The languages that can be used for coding Online Society Management System are JavaScript and HTML.

* **Database Design-**

In our database design, we give names to data flows, processes and data stores. Although the names are descriptive of data, they do not give details. Our interest is to build some details of the contents of data flow, processes and data store. A data dictionary is a structured repository of data about data. It is a set of rigorous definitions of all DFD data elements and structures.

**3.4.1 STANDARD COMPLIANCE**

**Report format**: All the reports produced for this project are in compliance with the standard templates provided in the class by the advisor.

**Naming Conventions:** All the documents will be named using the standard naming conventions.

**PROJECT METRICS**

Project metrics are used to control and coordinate software engineering process and to improve quality of the software to be produced. Project specific metrics provide indication of productivity and insight into the technical activities.

FUNCTION ORIENTED METRICS

Function oriented metrics use function point as normalization value. Function points are derived using empirical relationship based on countable measure of software’s information domain and qualitative assessments of software complexity.

To compute function points (FP), the following relationship is used:

**FP= count total \* [0.65 + 0.05 \* ∑ (fi)]**

Where count total is the sum of all FP entries obtained from the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Information Domain Value** | Count | Simple Weighting Factor |  |
| **External Inputs (EIs)** | 40 | 3 | =120 |
| **External Outputs (Eos)** | 11 | 4 | =44 |
| **External Inquiries (EQs)** | 2 | 3 | =6 |
| **Internal Logical Files (ILFs)** | 9 | 7 | =63 |
| **External Interface Files (EIFs)** | 1 | 5 | =5 |
| Count Total | | | 238 |

The fi (i = 1 to 14) is ***value adjustment factors*** (VAF) based on following responses:

|  |  |  |
| --- | --- | --- |
| 1. | Does the system require reliable backup and recovery? | 5 |
| 2. | Are specialized data communications required to transfer information to or from the application? | 1 |
| 3. | Are there distributed processing functions? | 2 |
| 4. | Is performance critical? | 2 |
| 5. | Will the system run in an existing, heavily utilized operational environment? | 3 |
| 6. | Does the system require online data entry? | 5 |
| 7. | Does the online data entry require the input transaction to be built over multiple screens or operations? | 3 |
| 8. | Are the ILFs updated online? | 5 |
| 9. | Are the inputs, outputs, files, or inquiries complex? | 0 |
| 10. | Is the internal processing complex? | 1 |
| 11. | Is the code designed to be reusable? | 5 |
| 12. | Are conversion and installation included in the design? | 1 |
| 13. | Is the system designed for multiple installations in different organizations? | 0 |
| 14. | Is the application designed to facilitate change and ease of use by the user? | 5 |
| **∑(fi)** | | 38 |

Therefore,

**FP** = 238\*[0.65 + (0.01 \*38)]

= **245.14**

**EFFORT ESTIMATION USING COCOMO MODEL**

**Co**nstructive **Co**st **Mo**del (COCOMO II) is a more comprehensive estimation model. COCOMO II is actually a hierarchy of estimation models that address the following areas:

• **Application composition model**-

Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.

• **Early design stage model**-

Used once requirements have been stabilized and basic software architecture has been established.

• **Post-architecture-stage model**-

Used during the construction of the software.

The COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy:

• object points

• function points

• Lines of source code

The object point is an indirect software measure that is computed using counts of the number of (1) screens (at the user interface), (2) reports, and (3) components likely to be required to build the application. Each object instance is classified into one of three complexity levels based on the following table:-

|  |  |  |  |
| --- | --- | --- | --- |
| **Object Type** | Complexity Weight | | |
| Simple | Medium | Difficult |
| **Screens** | 1 | 2 | 3 |
| **Reports** | 2 | 5 | 8 |
| **3GL Components** |  |  | 10 |

**Figure**–**Complexity weighting for object types**

The object count is determined by multiplying the total number of object instances by weighing factor.

When component based development or general software re-used is to be applied, the percent of re-use is estimated and object count is adjusted:

**NOP = (object points) X [(100 -% re-use)/100]**

Where NOP is defined as new object points

To derive an estimate of effort based on the computed NOP value, a “productivity rate” must be derived.

**PROD = NOP / person-month**

Once the productivity rate has been determined, an estimate of project effort is computed using –

**Estimated effort = NOP / PROD**

COCOMO Estimation for our project –

Number of screens = 11

Number of reports = 1

Number of 3GL components used = 1

PROD = 7

In our project, there are simple screens and reports.

So,

**Object point** = 11\*1 + 1\*2 + 1\*10

= 11 + 2 + 10

= 23

Since we’ll be re-using some of the components in our project for next increment, the % re-use is 10% here.

**NOP** = 23 \* [(100-10)/100]

= 20.7

**PROD** = 7 person-month

Now,

**Estimated Effort** = 20.7 / 7

= 2.95

≈ 3 person month

# RISK ANALYSIS

## Risk always involves two characteristics -:

**Uncertainty** - The risk may or may not happen; that is, there are no 100 percent probable risks.

**Loss** - If the risk becomes a reality, unwanted consequences or losses will occur.

When risks are analyzed, it is important to quantify the level of uncertainty and the degree of loss associated with each risk.

## The risk components are defined in the following manner –

* + **Performance Risk** – The degree of uncertainty that the product will meet its requirements and be fit for the intended use.
  + **Support Risk** – The degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
  + **Schedule Risk** – The degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.
  + **Cost Risk** – The degree of uncertainty that the product budget will be maintained.

# ASSESSING OVERALL PROJECT RISKS

## Have top software and customer managers formally committed to support the project? YES

1. **Are end users enthusiastically committed to the project and the system product to be built?** YES
2. **Are requirements fully understood by the software engineering team and its customers?** YES
3. **Have customers been involved fully in the definition of requirements?** YES
4. **Do end users have realistic expectations?** YES
5. **Is the project scope stable?** YES
6. **Does the software engineering team have the right mix of skills?** YES
7. **Are project requirements stable?** YES
8. **Does the project team have experience with the technology to be implemented?** YES
9. **Is the number of people on the project team adequate to do the job?** YES
10. **Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?** YES

**RISK TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risks** | **Category** | **Probability** | **Impact** | **Mitigation** |
| **Server breaks down** | BU | **40%** | **2** | Past experience might be considered |
| **Data loss** | PS | **30%** | **2** | Take up steps to maintain backup and recovery |
| **Security issues** | **ST** | **20%** | **3** | External resources might help |

## DATA DESIGN

**Name: EVENTS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | EventId | Int | No | Nil | Valid event id | Primary Key |
| 2. | Place | Varchar(50) | No | Nil | Area inside the society campus | - |
| 3. | Budget | Decimal(7,2) | No | Nil | Amount needed for holding that event | - |
| 4. | Date | date | No | Nil | Event’s date | - |
| 5. | Time | Int | No | Nil | Event’s time | - |
| 6. | Description | Varchar(100) | No | Nil | Event’s description | - |

**Name : EVENTS\_ORGANISER**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | EventId | Int | No | Nil | Valid event id | Primary Key,Foreign Key |
| 2. | TeamId | Int | No | Nil | Ids of organizing team | Primary Key |

**Name: ACCOUNTS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | RId | Int | No | Nil | Valid resident id | Primary Key,  Foreign Key |
| 2. | Bill no | Int | No | Nil | Bill number for a particular resident | Primary Key |
| 3. | Amount | Decimal(7,2) | No | Nil | Maintenance amount to be paid | - |
| 4. | Date | date | No | 1st of every month | Date on which bill is displayed | - |
| 5. | Status | Varchar(50) | No | Unpaid | Paid/unpaid status | - |

**Name: ONLINE PAYMENT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | Rid | Int | No | Nil | Valid resident id | Foreign Key, Primary Key |
| 2. | Billno | Int | No | Nil | Bill number for a particular resident | Foreign, Primary Key |
| 3. | Amount | Decimal(7,2) | No | Nil | Maintenance amount to be paid | Foreign Key |
| 4. | Street | Varchar(50) | No | Society’s Address | Street of the billing address | - |
| 5. | City | Varchar(50) | No | Society’s Address | City of the billing address | - |
| 6. | Country | Varchar(50) | No | Society’s Address | Country of the billing address | - |
| 7. | Zipcode | Varchar(50) | No | Society’s Address | Zip code of the billing address | - |

**Name: BULLETIN**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint | |
| 1. | Date | date | No | Nil | Date on which bulletin is added/updated | Primary Key |
| 2. | Duration | Int | No | Nil | No. of days till which bulletin will be displayed | Primary Key |
| 3. | Description | Varchar(100) | No | Nil | Bulletin’s description | - |

**Name: RESIDENTS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | RId | Int | No | Nil | Valid resident id | Primary Key |
| 2. | Fname | Varchar(50) | No | Nil | First name of resident | - |
| 3. | Minit | Varchar(50) | No | Nil | Middle Initial name of resident | - |
| 4. | Lname | Varchar(50) | No | Nil | Last name of resident | - |
| 5. | WingNo | Int | No | Nil | Address(WingNo) of resident | - |
| 6. | FlatNo | Int | No | Nil | Address(FlatNo) of resident | - |
| 7. | FloorNo | Int | No | Nil | Address(FloorNo) of resident |  |
| 8. | MemberCount | Int | No | Nil | No. of the members living in that flat | - |

**Name: RESIDENTS\_PHONE\_NUMBERS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | RId | Int | No | Nil | Valid resident id | Primary Key |
| 2. | Phone\_no | Decimal(7,2) | No | Nil | Phone Number of resident | - |

**Name: COMMITTEE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | RId | Int | No | Nil | Valid resident id | Primary Key,  Foreign Key |
| 2. | PostName | Varchar(50) | No | Nil | name of post | Unique Key |

**Name: COMPLAINT/SUGGESTION**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | RId | Int | No | Nil | Valid resident id | Primary Key,  Foreign Key |
| 2. | Date | date | No | Nil | Date of submission | - |
| 3. | Description | Varchar(100) | No | Nil | Complain/suggestion’s description | - |

**Name: COMPLAINT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | CId | Int | No | Nil | Valid complain id | Primary Key |
| 2. | Solution\_msg | Varchar(100) | Yes | Nil | Possible Solution of the complain | - |
| 3. | Respondent\_Name | Varchar(50) | No | Nil | Name of the respondent | - |

**Name: SUGGESTION**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | SId | Int | No | Nil | Valid suggestion id | Primary Key |

**Name: VOTERS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint |
| 1. | RId | Int | No | Nil | Valid resident id | Primary Key,  Foreign Key |
| 2. | Date | date | No | Nil | Date of voting | - |
| 3. | Type | Varchar(50) | No | Nil | Type of voting | - |

**Name :VOTING\_EVENT**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint | |
| 1. | Ename | Varchar(50) | No | Nil | Name of the event | Primary Key |
| 2. | Description | Varchar(100) | No | Nil | Event’s description | - |

**Name :VOTING\_CANDIDATE**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr.  No. | Field  Name | Data Type | Null Constraint | Default Constraint | Description | Key Constraint | |
| 1. | Rid | int | No | Nil | Name of the event | Primary Key,  Foreign Key |
| 2. | Valid | int | No | Nil | Event’s description | - |
| 3. | PostName | Varchar(50) | No | Nil | Event’s description | - |

**USE CASE DESCRIPTION**

1. **Login**

* **Brief Description**

It allows the actors to use the software, SOCIETY MANAGEMENT SYSTEM, with the help of an already created unique id and password.

### Actors

The following actors participate in this use case:-

* + Committee Members
  + Society Members

### Flow of Events Basic Flow

This use case starts when the actor wants to sign in:-

* + The system requests the actor to enter his/her id and password.
  + The actor enters his/her id and password.
  + The system verifies the details.
  + If the actor enters a valid id and password then system allows the actor to login into the society portal.

### Alternative Flow

If the actor enters an invalid id and password, then an error message will be displayed asking the user to log into his/her account again or cancel the login process. If the actor selects the latter option, the use case ends.

### Pre-Conditions

* + The actors must have already a unique id and password to login into the society portal.

### Post-Conditions

* If the login is successful, the actor is logged into the system.
* If not, then the use case remains unchanged.
* If the actor has role of the “committee members”, then s/he would access to complaint/suggestions section ,also make responses regarding it, can access their own accounting section , perform voting whenever required, update bulletin ,rules and regulation and also responses to event scheduling section
* If the actor has role of the “society members”, then s/he would access complaint/suggestions section, can access their own accounting section, perform voting whenever required, view updated bulletin, rules and regulation and also make requests to event scheduling section.

1. **Voting**

* **Brief Description**

It allows the actors to digitally perform voting either to candidates standing for the particular position or to the different suggestions/opinions on the same topic.

### Actors

The following actor participates in this use case:-

* + Society Members
  + Committee members

### Basic Flow

### 

This use case starts when actors starts performing voting.

* + The system makes mandatory the actor to give only one vote only to any of the candidate or idea.
  + The actor performs voting.
  + Then result of voting is being declared on scheduled date.

### Alternative Flow

### Pre-Conditions

The actor must have already logged into the system.

Voting portal will open only at the time of voting.

### Post-Conditions

If the voting is completed, then actor (society members) will not be able to access this portal anymore.

If the voting is completed, then actor (committee members) will access this portal to generate the result and display on bulletin section.

1. **Complaint And Suggestion Box**

* **Brief Description**

It allows the actors to give complaint regarding any issues and suggestions regarding any events. The Committee members will respond accordingly and society members can view their respective response.

### Actors

The following actors participate in this use case:-

* + Committee Members
  + Society Members

### Basic Flow

This use case starts when the actor wants to either give complaint or suggestion:-

* + The system asks the actor to give either complaint or suggestion details.
  + The actor (society members) responds accordingly.
  + The actor (committee members) will also respond accordingly.
  + The actor (society members) can view their respective response.

### Alternative Flow

If the actor gives the unauthenticated complaint details then actor (committee members) will act according to rules and regulations of the society.

### Pre-Conditions

The actors must have already logged into the society portal.

### Post-Conditions

* + If the complaints are authenticated one then the actor (committee members) will also respond accordingly.
* The actor (society members) can view their respective response.

1. **View Bulletin And Community Wiki**

* **Brief Description**

It allows the actors (Committee members) to give updated details regarding events and result regarding voting.

It allows actors to view the rules and regulations, updated information and result details.

### Actors

The following actors participate in this use case:-

* + Committee Members
  + Society Members

### Basic Flow

This use case starts when the actor wants to view and update bulletins:-

* In the system, the actors (Committee members) will update details regarding events and results regarding voting.
  + In the system ,the actors can view the rules and regulations , updated information and result details

### Alternative Flow

If the actor (society members) tries to manipulate the bulletin information then actor (committee members) will act according to rules and regulations of the society.

### Pre-Conditions

The actors must have already logged into the society portal.

### Post-Conditions

* The actors can view the updated information.

1. **Accounting**

* **Brief Description**

The system will generate maintenance bills automatically .And the actors can not only view their bill details but also can pay them through payment gateway. And then the system will be updated accordingly.

### Actors

The following actors participate in this use case:-

* + Committee Members
  + Society Members

This use case starts when the system generates maintenance bills and actors have to view and pay them:-

### Basic Flow

The system will generate maintenance bills automatically.

And the actors can not only view their bill details but also can pay them through payment gateway.

And then the system will be updated accordingly.

### Alternative Flow

If the actor does not want to perform online payment then they can also do it by cash.

### Pre-Conditions

The actors must have already logged into the society portal.

### Post-Conditions

* + The information must be updated in the residents’ database.

1. **Event Scheduling**

* **Brief Description**

The actors (society members) will seek permissions from the actor (committee members) regarding any event by providing event details, schedule and other details .

The actors (committee members) can either approve or reject the proposal and accordingly update the bulletin.

### Actors

The following actors participate in this use case:-

* + Committee Members
  + Society Members

### Basic Flow

This use case starts when the actor (society members) wants to ask permission regarding any events:-

The actors (society members) will provide event details, schedule and other details.

If the actors (committee members) approved the proposal then bulletins will be updates accordingly.

### Alternative Flow

If the actor (committee members) rejected the proposal then respected actor (society member) will be informed accordingly.

### Pre-Conditions

The actors must have already logged into the society portal.

### Post-Conditions.

* In case of approval, bulletins will be updated.
* In case of rejection. respected actor(society member) will be informed accordingly

**COMPONENT LEVEL DESIGN**

**PSEUDO CODE FOR EVENT SCDEULING -**

Event (int eid , int rid)

1. If (rid==Resident)

input\_details()

2. cout<<"Do you want to continue(y/n)? "

cin>>ans

3. if(ans== yes)

4. cout<<"submitted"

end if

5. end if

6. else

7. if(permission())

cout<<"Notify a success msg to organizer"

8. update\_bulletin()

end if

else

9. cout<<"Notify a failure msg to organizer”

10. end else

11. end event()

**TESTING**

**White box testing:**

White box testing, sometimes called glass box testing, is a test case design philosophy that uses the control structure described as a part of component level design to derive test cases. Using white box testing technique we can derive test cases that:

1. Guarantee that all the independent paths within a module have been exercised at least once.
2. Exercise all logical decisions on their true and false sides.
3. Execute all loops at their boundaries and within their operational bounds and
4. Exercise internal data structures to ensure their validity.

**Basis – Path Testing**

It is a white-box testing technique which enables the test case designer to derive a logical complexity measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least once during testing

R2

R1

R3

R4

**Flow Graph**

10

**CYCLOMATIC COMPLEXITY OF RESULTANT GRAPH**

**V(G) = Number of regions**

**=** 4

**V(G) = Edges –nodes + 2**

**=** 13 – 11 + 2

= 4

**V(G) = Predicate nodes + 1**

= 3 + 1

= 4

**LINEARLY INDEPENDENT PATHS FOR FLOW GRAPH**

**Path 1:** 1-2-3-4-5-11

**Path 2:** 1-2-3-5-11

**Path 3:** 1-6-7-8-10-11

**Path 4:** 1-6-7-9-10-1

ANNEXURES



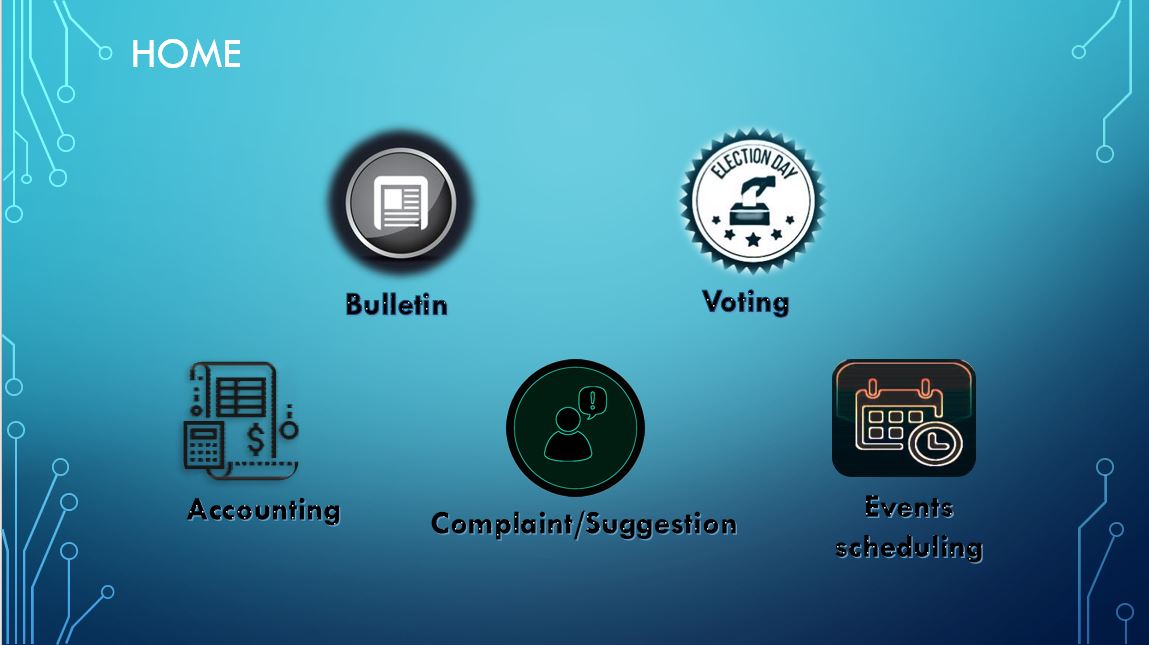
Number of Inputs: 3

Number of outputs: 1

Files: 0

Enquiries: 1

External interfaces: 0



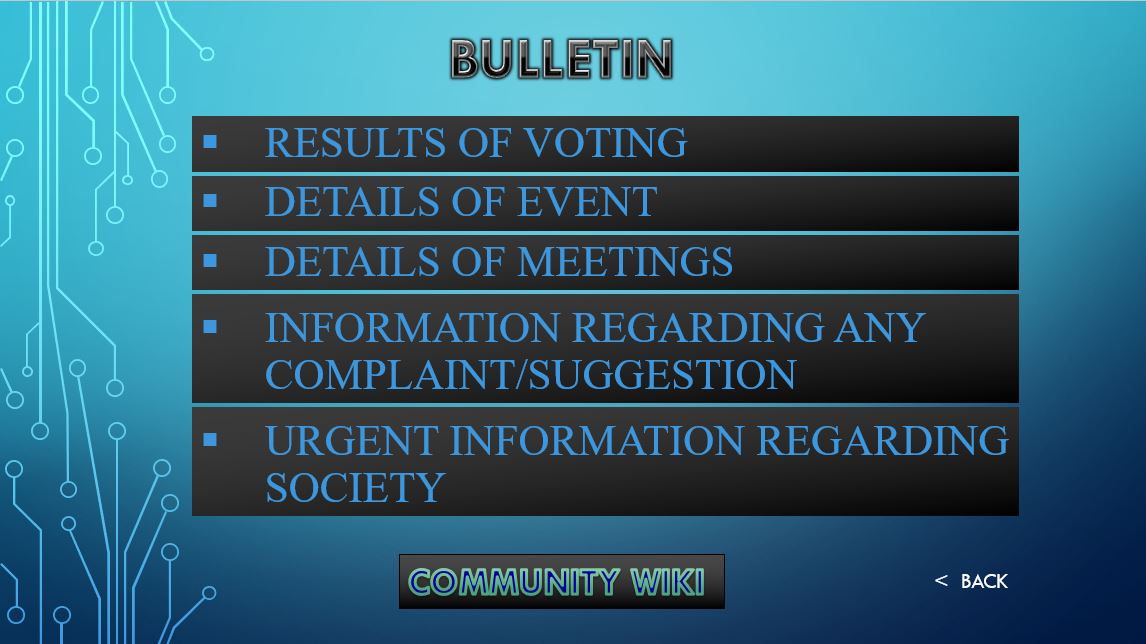
Number of Inputs: 1

Number of outputs: 1

Files: 0

Enquiries: 0

External interfaces: 0



Number of Inputs: 1

Number of outputs: 1

Files: 1

Enquiries: 1

External interfaces: 0



Number of Inputs: 1

Number of outputs: 1

Files: 1

Enquiries: 0

External interfaces: 0



Number of Inputs: 2

Number of outputs: 1

Files: 1

Enquiries: 0

External interfaces: 0



Number of Inputs: 6

Number of outputs: 1

Files: 1

Enquiries: 0

External interfaces: 0



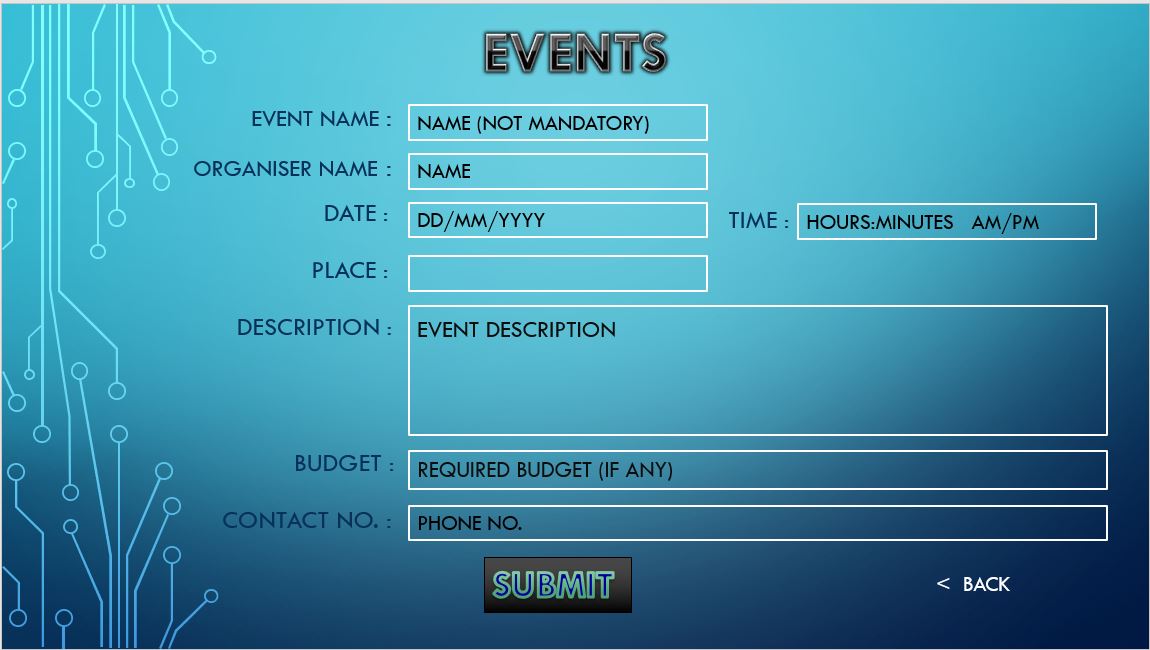
Number of Inputs: 4

Number of outputs: 1

Files: 1

Enquiries: 0

External interfaces: 0



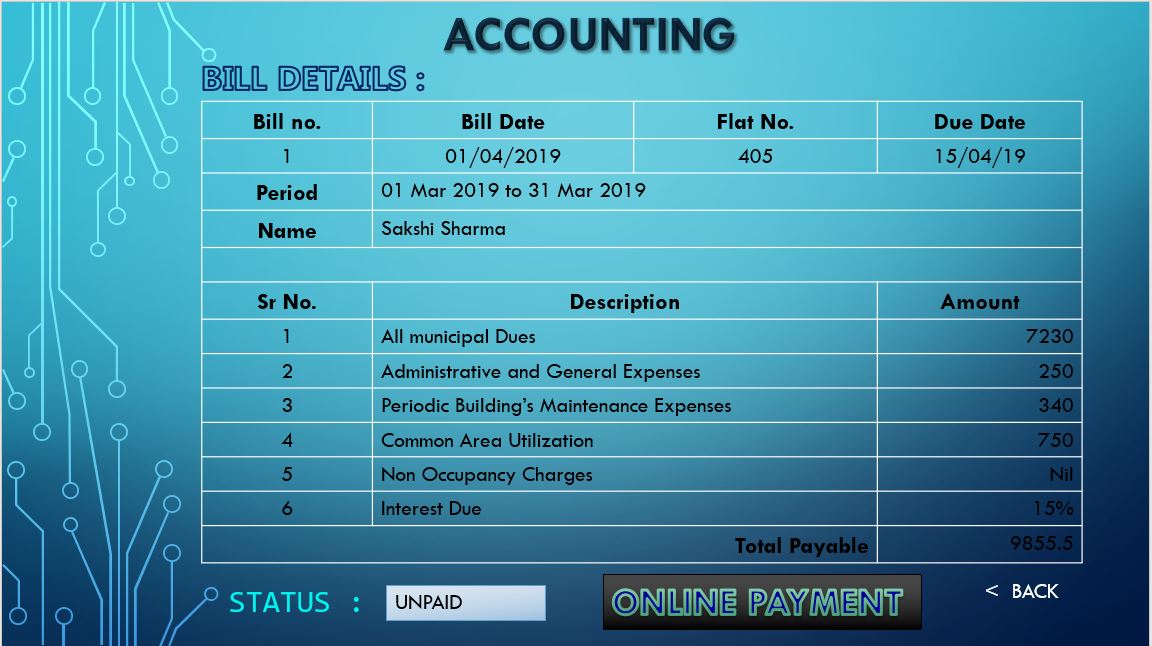
Number of Inputs: 8

Number of outputs: 1

Files: 1

Enquiries: 0

External interfaces: 0



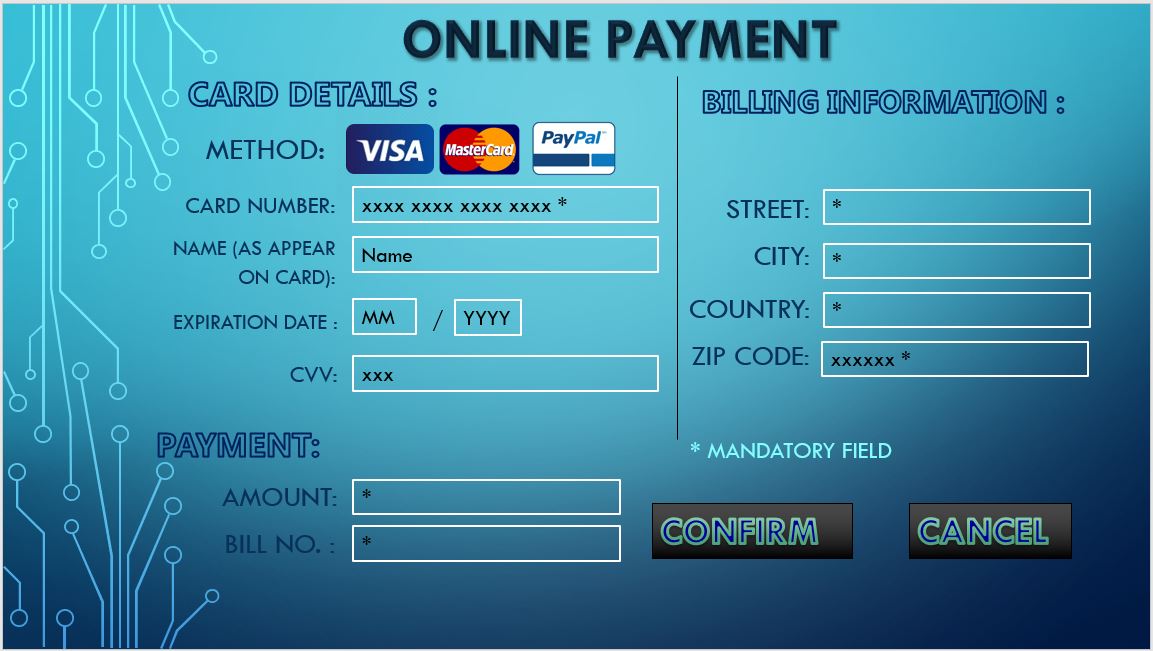
Number of Inputs: 1

Number of outputs: 1

Files: 1

Enquiries: 0

External interfaces: 0



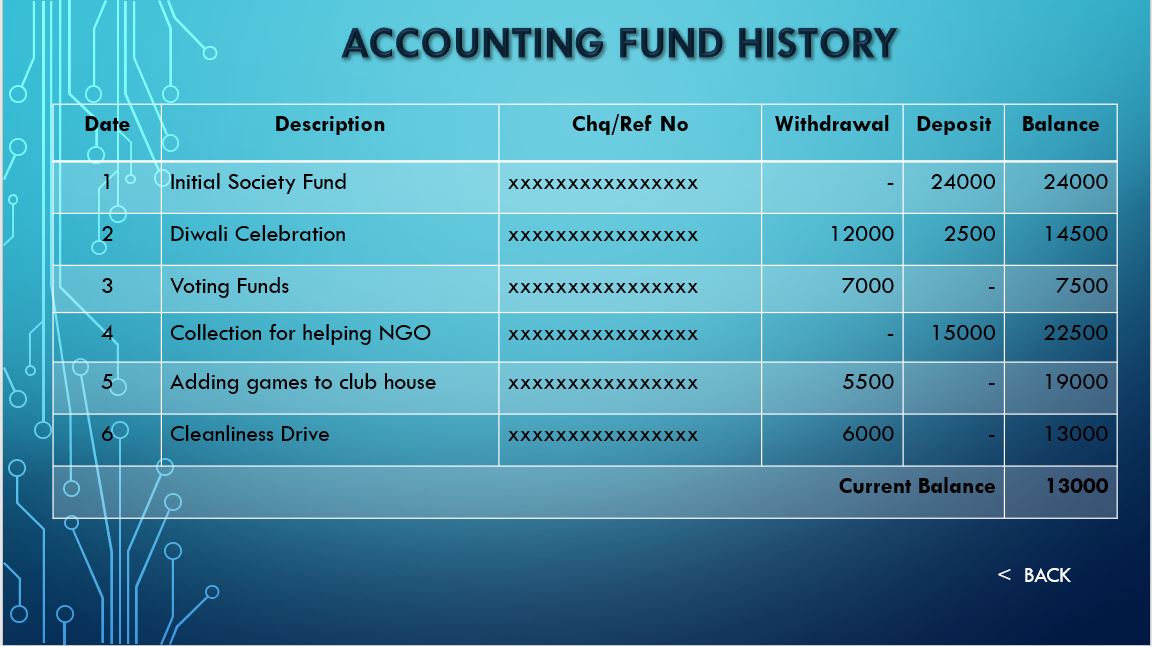
Number of Inputs: 12

Number of outputs: 1

Files: 1

Enquiries: 0

External interfaces: 1



Number of Inputs: 1

Number of outputs: 1

Files: 1

Enquiries: 0

External interfaces: 0