[Date]

Atiq Ahammed

[Company name]

[Company address]

[Document title]

[Document subtitle]

Application Form

Hjjkhkjhjhghj

Jkhkjh

Jhgjhgjhgjhg

Jhgjhgjhgjhgjhgjhg

Kjhkjhkjh

Hgghgfhgfhgf

Hgjhgjhg

Kjhkjhkjhkjh

Hgjhgjhgjhg

Kflkjfsldjfsldjkf

**Contents**

[Chapter 1: Introduction 1](#_Toc508783440)

[1.1 Purpose 1](#_Toc508783441)

[1.2 Intended Audiences 1](#_Toc508783442)

[1.3 Conclusion 1](#_Toc508783443)

[Chapter 2: Inception 2](#_Toc508783444)

[2.1 Introduction 2](#_Toc508783445)

[2.1.1 Listing Down Stakeholders 2](#_Toc508783446)

[2.1.2 Recognizing Multiple Viewpoints 2](#_Toc508783447)

[Chapter 3: Elicitation 4](#_Toc508783448)

[3.1 Introduction 4](#_Toc508783449)

[3.2 Eliciting Requirements 4](#_Toc508783450)

[3.2.1 Collaborative Requirements Gathering 4](#_Toc508783451)

[3.2.2 Problem in the Scope 4](#_Toc508783452)

[3.2.3 Quality Function Deployment 5](#_Toc508783453)

[3.2.3.1 Normal Requirements 5](#_Toc508783454)

[3.2.3.2 Expected Requirements 5](#_Toc508783455)

[3.2.3.3 Exciting Features 6](#_Toc508783456)

[3.2.4 Usage Scenario 6](#_Toc508783457)

[3.2.4.1 Problem Submission 6](#_Toc508783458)

[3.2.4.1.1 Authentication 6](#_Toc508783459)

[3.2.4.1.2 Submitting Problem 7](#_Toc508783460)

[3.2.4.1.3 Viewing User Account 7](#_Toc508783461)

[3.2.4.1.4 Emergency Support 8](#_Toc508783462)

[3.2.4.1.5 Notice Board 8](#_Toc508783463)

[3.2.4.1.6 Download Application Forms 8](#_Toc508783464)

[3.2.4.2 Problem Supervision 8](#_Toc508783465)

[3.2.4.2.1 Authentication 9](#_Toc508783466)

[3.2.4.2.2 View Submitted Problem 9](#_Toc508783467)

[3.2.4.2.3 Update Problem Information 9](#_Toc508783468)

[3.2.4.2.4 Update Notice, Event and Application Forms 10](#_Toc508783469)

[3.2.4.2.5 Update Emergency Information 10](#_Toc508783470)

[Chapter 4: Scenario Based Modeling 11](#_Toc508783471)

[4.1 Introduction 11](#_Toc508783472)

[4.2 Use Case 11](#_Toc508783473)

[4.3 Use Case Diagram 12](#_Toc508783474)

[Chapter 5: Data Based Modeling 13](#_Toc508783475)

[5.1 Introduction 13](#_Toc508783476)

[5.2 Data Object Selection 13](#_Toc508783477)

[5.2.1 Noun Identification 13](#_Toc508783478)

[5.2.2 Potential Data Objects 15](#_Toc508783479)

[5.2.3 Analysis for Finalizing Data Objects 16](#_Toc508783480)

[5.2.4 Final Data Objects 16](#_Toc508783481)

[5.3 Data Object Relations 17](#_Toc508783482)

[5.4 Entity Relationship Diagram 18](#_Toc508783483)

[5.5 Schema Diagram 18](#_Toc508783484)

[Chapter 6: Class Based Modeling 21](#_Toc508783485)

[6.1 Introduction 21](#_Toc508783486)

[6.2 Grammatical Parsing and Class Analysis 21](#_Toc508783487)

[6.2.1 General Classification 21](#_Toc508783488)

[6.2.2 Selection Criteria 23](#_Toc508783489)

[6.2.3 Preliminary Classes 24](#_Toc508783490)

[6.3 Associated Noun and Verb Identification 25](#_Toc508783491)

[6.4 Attribute Selection 26](#_Toc508783492)

[6.5 Method Selection 27](#_Toc508783493)

[6.6 Finalized Classes 28](#_Toc508783494)

[6.6.1 Finalized classes for Problem Submission 28](#_Toc508783495)

[6.6.2 Finalized classes for Problem Supervision 30](#_Toc508783496)

**Tables**

**Figures**

Chapter 1: Introduction

This chapter is a part of our Software Requirement Specification and Analysis for the project “Complain Box”. In this chapter, the center of our activity is the intended audience for this project.

1.1 Purpose

This document is the simple outline of the Software Requirement Specification and Analysis of our project for Software Project Lab – 2 titled as “Complain Box”. It contains functional, non – functional and supporting requirements and establishes a requirement – baseline for developing the system. The Software Requirement Specification holds the requirements that were collected from Dhaka South City Corporation Authority (DSCCA) and people living in Dhaka South City Corporation (DSCC). The Software Requirement Specification serves as a platform to forward user requirements to us and provides a common reference point for both our team and DSCCA.

1.2 Intended Audiences

Our Software Requirement Specification (SRS) is pinned for several audiences including DSCCA as well as our project supervisor, SPL – 2 coordinators and ours.

* DSCCA will use this SRS to verify that we have developed a product that the required
* Our supervisor will use this SRS to plan milestones and ensure that we are on the right track when developing the system
* We will use this SRS as a basis for creating the system design. We will continually refer back to this SRS to ensure that the system we are designing, will fulfill the requirements of DSCCA and the people living in DSCC
* We will also use this SRS as a basis for developing the system functionality and link the requirements defined in this SRS to the software that we will create to ensure that we have created a software that will fulfill all the requirements

1.3 Conclusion

We wish, this analysis of the audience will help us to focus on the users who will be using our analysis. This document will help each and every person related to this project to perceive the subject matter of the project.

Chapter 2: Inception

In this chapter, we succinctly discuss the Inception part of the SRS for our “Complain Box”.

2.1 Introduction

This is our first aspect of requirement engineering. In this stage, we tried to understand the problem, identified the people living in DSCC and DSCCA as our stakeholders and communicate with them.

For clear perception of the software requirements, we established a groundwork involving the following steps-

1. We listed the DSCCA and people living in DSCC as the stakeholder
2. We collected their multiple viewpoints
3. We worked towards collaboration
4. We broke down the ice and initiated communication with DSCCA and people living in the DSCC

2.1.1 Listing Down Stakeholders

To identify stakeholders, we consulted a number of people living in DSCC and DSCCA, asked them the following questions:

* Who will be using the product?
* What kind of features they want in the product?
* Whose work will be affected by this product?

We identified the following stakeholders for our project.

* People living in DSCC
* Dhaka South City Corporation Authority (DSCCA)

2.1.2 Recognizing Multiple Viewpoints

The list of stakeholders will contribute to the input when requirements are elicited. Every stakeholder has different views of the system and achieves different benefits when the system is developed.

Viewpoints of people living in Dhaka City Corporation:

* User friendly problem submission system
* User authentication system
* Complaining system to submit problem to DSCCA
* Facility to inform problem description to the authority
* Submitting problem within minimum time
* Easy environment for selecting problem category
* Environment for selecting photos or taking instance photo with phone category
* Viewing information about problem category

Viewpoint of Dhaka city corporation Authority

* Manage submitting problem easily
* Web based system for problem management
* Authentication system for DSCCA
* Environment for viewing submitted problem
* Automatic ward–wise selection system
* Problem ranking system
* Disable problem

Conflicting viewpoints

* User account for submitting problem
* Submitting username with problem
* Account recovery
* Submitting video or audio files for describing problem
* Disable problems

Common viewpoints

* Authentication system for user account
* Using location for specifying problem

Chapter 3: Elicitation

After discussing on the Inception phase, we need to focus on the Elicitation phase. So, this chapter specifies the Elicitation phase.

3.1 Introduction

The second phase of requirements engineering is elicitation. The main task of elicitation is to combine elements of problem solving, elaboration, negotiation and specification. Gathering information from stakeholders regarding the problem was not sufficient to design the software. The problems that arose, were encountered following the principles of elicitation.

3.2 Eliciting Requirements

We worked together with DSCCA and people living in DSCC to identify the problems, propose elements of the solutions, negotiate different approaches and specify an initial set of solution requirements. This approach is sometimes called Facilitated Application Specification Technique (FAST). Elicitation has some sub-phases which are:

1. Collaborative Requirements Gathering
2. Quality Function Deployment
3. Usage Scenario
4. Elicitation Work Products

3.2.1 Collaborative Requirements Gathering

During inception, basic questions and answers established the scope of the problem. However, some problems persisted about the scope as the boundary of the system was ill defined and the people living in DSCC and DSCCA have stated some unnecessary confusing detail. Also, our stakeholders do not have a proper understanding about the abilities of the computing environment which results in further discussion regarding the problem domain and product requirements. The requirements were put under re-evaluation by conducting meeting with stakeholders (DSCCA and people of DSCC) and we went into further investigation about their requirements and expectations. The final requirement list was derived at the end of the meeting.

3.2.2 Problem in the Scope

A number of problems were encountered in the course of preparing the software requirement specification and analysis of “Complain Box”.

Scope

Software is developed for only submitting and managing problems.

Limitation

Internet is required for both submitting and managing problems.

3.2.3 Quality Function Deployment

Quality Function Deployment (QFD) is a quality management technique that translates the needs of the clients into technical requirements for the software. The prime concern of the QFD is customer satisfaction maximization. In order to ensure this, QFD enforces an understanding of what the customers describe as ‘valuable’ and then deploy these values throughout the engineering process.

QFD defines three types of requirements:

* Normal requirements
* Expected requirements
* Exciting features

3.2.3.1 Normal Requirements

Normal requirements refer to the objectives and the goals that are stated for the product during the meeting with the stakeholders. The presence of these requirements ensures the satisfaction of the customers. The normal requirements for the project are stated below.

* Authentication system for user and authority of City corporation system
* Easy environment for selecting or capturing photo for submitting problem
* Environment for informing the authority about problem detail
* Easy environment for selecting location of problem
* Automatic ward-wise problem categorization for authority
* Ranking system for problem
* Problem list management

3.2.3.2 Expected Requirements

The requirements that are implicit to the system might not be brought up during the meeting because of their fundamental nature. Despite being not explicitly mentioned their presence must be ensured. Otherwise, the product will leave customers dissatisfied. These requirements are called expected requirements and these are stated below.

* Error-free problem submission within minimum time
* More efficient than the existing workflow for submitting and recognizing problem
* Secured Authentication system for both authority and user
* Web based server
* Database for storing problem information

3.2.3.3 Exciting Features

The factors that go beyond the customer’s expectations and prove to be satisfying when present are called exciting features. The exciting features are the so called ‘wow factor’ for our project.

* Viewing own submitted problem
* Viewing problems with continuous updated status

3.2.4 Usage Scenario

Complain Box will be an online based complaint (problem) collection system. Using this system, people living in DSCC will be able to submit those problems that they face in their daily city life to notify DSCCA. On the other hand, DSCCA will be able to view those submitted problems and take necessary steps to solve those problems.

Complain Box has the following subsystem:

1. Problem Submission
2. Problem Supervision

3.2.4.1 Problem Submission

In this subsystem, the people living in DSCC will submit their problems that they face in their daily life and need to inform the DSCCA to solve the problems. This subsystem will work on Android-based smart-phone. This subsystem is divided into:

3.2.4.1.1 Authentication

Initially, the user will create a user account to use this system if she/he do not have an account already. To create a user account, the user will provide her/his name, valid phone number and password. The password should be at least five characters and at most fifteen characters and name should be at least two characters and at most twenty characters long. A unique phone number will be used to create only one account. After creating a user-account the information will be stored in the system and the user will automatically be logged in to the system. Besides a user having a user account, will be able to log in to the system using her/his phone number and password that she/he would have used to create that account. If a user forgets her/his password, the user will request to recover her/his password providing her/his valid phone number that she/he would have used to create the user account. Then, the system will send a message containing the password.  A logged in user will be able to log out from the system.

3.2.4.1.2 Submitting Problem

To submit a problem, the user will first select the problem category. The problem category includes:

* Solid Waste Management
* Road, Footpath and Drain
* Drainage
* Repair to Municipal Property
* Road light
* Illegal parking
* Mosquito problem
* Health Department
* Revenue Department
* Social Welfare Department
* Planning Department
* Miscellaneous Complains
* Establishment Sections
* Encroachment
* Factories
* License
* Corruption

Note: problem category was defined by Md. Abu Taiyeb Rokon, System analyst, DSCC.

Then, the user will capture a picture or video, or record a audio file or select one from the storage which will be related to a specific problem, add problem descriptions and select the problem location. The picture, video or audio file describing the problem must be within twenty Megabyte. The system will store the latitude and longitude of the location. Problem description should be maximum five hundred characters. The user will be able to use her/his current location automatically or manually set the problem location, but the location must be within DSCC Area.

3.2.4.1.3 Viewing User Account

The user will be able to view her/his own account information. The user account contains:

* User name
* Phone number
* Reward point
* Problems submitted by the user

The information of the problem that the user will be able to view will contain:

* Problem ID
* Photo/video/audio of the problem
* Description about the problem
* Problem category
* Problem location
* Submission date and time
* Current status of the specific problem
* Problem priority
* Ward number in which the problem would have occurred

The user is unable to change any information of the problem if it once submitted.

3.2.4.1.4 Emergency Support

The user will get necessary information and communication facility of hospitals, fire station, police station within the area of DSCC. The information contains:

* Name of institute
* Location
* Phone number

of hospitals/fire stations/police stations.

3.2.4.1.5 Notice Board

The user will able to view the information of notices and events send by DSCC. The information includes notice content, event name, publishing date, event date and time.

3.2.4.1.6 Download Application Forms

The user can download some application forms as pdf file. The application form includes

* Passport Application Form
* E-Income Tax Certificate
* Driving License Form
* Birth Certificate
* Trade License Form
* Tax Form
* Inherit Certificate
* Death Certificate

3.2.4.2 Problem Supervision

This subsystem will work in a web-based system. Only system admin (selected person from DSCCA) will operate the system and take a decision on the submitted problem. This subsystem will be divided into:

3.2.4.2.1 Authentication

The admin of the system will log in to the system providing email address and password that would be defined by the developer. There will be no scope to create another admin account or adding a new admin to the system. The admin will also be able to log out of the system. Admin can request for recover account if she/he forget the password. Then, the system will send the password to the email address that the admin uses to log in to the system from beginning.

3.2.4.2.2 View Submitted Problem

Admin will be able to view all the submitted problem. Admin will also be able to view problems based on the following type:

* Problem id
* Problems in particular category
* Solved problems
* Problems in progress
* Problems under consideration
* Submission date
* Problems in a particular Ward

The admin will be able to view the following information about a particular problem

* Problem id
* Description and photo (if submitted)
* Category
* Location
* Submission date and time
* Status (by default the status will be under consideration)
* Priority (initially priority will be 0)
* Ward number

3.2.4.2.3 Update Problem Information

The admin will be able to edit the problem category if she/he thinks that the user has chosen wrong problem category. The admin will select the priority based on the importance of the submitted problem. She/he will update the status of problems as

* Problem under consideration
* Problem in progress
* Solved

3.2.4.2.4 Update Notice, Event and Application Forms

The admin will upload the notice for the users. The notice will be in pdf. She/he will also able to create event providing event date, time and description. Besides, she/he will also upload or change application forms for the users.

3.2.4.2.5 Update Emergency Information

The admin will able to add emergency support information providing name, selecting location and add phone number of the institution. She/he also can update the information of emergency support institutions.

Chapter 4: Scenario Based Modeling

This chapter contains the Scenario Based Model for our project “Complain Box”.

4.1 Introduction

For developing our software, we are giving the highest priority to user satisfaction. To identify the requirements to establish meaningful analysis and design model we determine how users and DSCCA want to interact with the system. Thus, our requirements modeling begins with scenario generation in the form of use cases, activity diagrams.

4.2 Use Case

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions that some system or sub-systems can perform in collaboration with one or more external users of the system.

The first step in writing a Use Case is to define that set of “actors” that will be involved in the story. Actors are the different people that use the system or product within the context of the function and behavior that is to be described. Actors represent the roles that people play as the system operators. Every user has one or more goals when using the system.

Primary Actor

Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software. In our system both users (people living in DSCC) and Admin to supervise the system both are primary actor.

Secondary Actor

Secondary actors support the system so that primary actors can do their work. They either produce or consume information. In our system we have no secondary actor.

4.3 Use Case Diagram

Level-0 Use Case Diagram

Chapter 5: Data Based Modeling

This chapter describes the Data Based Model for our project “Complain Box”.

5.1 Introduction

In our software requirements the creation, extension and interaction with the database is also included. So, we choose to create data models as a part of overall requirements modelling.

5.2 Data Object Selection

The data objects in this chapter represent the information of our “Complain Box” which has different properties or attributes that must be understood by our system, “Complain Box”. Here the table of data object-

5.2.1 Noun Identification

We identified all nouns from our usage scenario whether they are in problem space or solution space.

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial Number** | **Nouns** | **Problem/Solution Space** | **Attributes** |
| 1 | Dhaka City Corporation | p |  |
| 2 | User | s | 5, 6, 7, 48 |
| 3 | Account | s |  |
| 4 | System | p |  |
| 5 | Name | s |  |
| 6 | Phone number | s |  |
| 7 | Password | s |  |
| 8 | Five | p |  |
| 9 | Character | p |  |
| 10 | Fifteen | p |  |
| 11 | Subsystem | p |  |
| 12 | Information | p |  |
| 13 | Log | p |  |
| 14 | Message | p |  |
| 15 | Problem Category | s |  |
| 16 | Solid Waste | p |  |
| 17 | Road | p |  |
| 18 | Drainage | p |  |
| 19 | Municipal Property | p |  |
| 20 | Road light | p |  |
| 21 | Illegal parking | p |  |
| 22 | Mosquito problem | p |  |
| 23 | Health Department | p |  |
| 24 | Revenue Department | p |  |
| 25 | Social Welfare Department | p |  |
| 26 | Planning Department | p |  |
| 27 | Miscellaneous Complains | p |  |
| 28 | Establishment Sections | p |  |
| 29 | Encroachment | p |  |
| 30 | Factories | p |  |
| 31 | License | p |  |
| 32 | Corruption | p |  |
| 33 | Footpath | p |  |
| 34 | Drain | p |  |
| 35 | Picture | p |  |
| 36 | Video | p |  |
| 37 | Audio | p |  |
| 38 | File | p |  |
| 39 | Storage | p |  |
| 40 | Problem | s | 15, 41, 42, 49, 50, 51, 52, 53, 54 |
| 41 | Description | s |  |
| 42 | Problem Location | s | 43, 44 |
| 43 | Latitude | s |  |
| 44 | Longitude | s |  |
| 45 | Current Location | p |  |
| 46 | Area | p |  |
| 47 | User name | s |  |
| 48 | Reward point | s |  |
| 49 | Problem ID | s |  |
| 50 | Submission date | s |  |
| 51 | Submission time | s |  |
| 52 | Status | s |  |
| 53 | Problem priority | s |  |
| 54 | Ward number | s |  |
| 55 | Communication facility | p |  |
| 56 | Hospital | p |  |
| 57 | Fire station | p |  |
| 58 | Police station | p |  |
| 59 | Name of institute | s |  |
| 60 | Notice | s | 61, 63 |
| 61 | Notice content | s |  |
| 62 | Event name | s |  |
| 63 | Publishing date | s |  |
| 64 | Event date | s |  |
| 65 | Event time | s |  |
| 66 | Application form | s |  |
| 67 | Passport application form | p |  |
| 68 | E-income tax certificate | p |  |
| 69 | Driving license form | p |  |
| 70 | Birth certificate | p |  |
| 71 | Trade license form | p |  |
| 72 | Tax form | p |  |
| 73 | Death certificate | p |  |
| 74 | Inherit certificate | p |  |
| 75 | Admin | s | 7, 79 |
| 76 | Person | p |  |
| 77 | Decision | p |  |
| 78 | Authentication | p |  |
| 79 | Email address | s |  |
| 80 | Developer | p |  |
| 81 | Scope | p |  |
| 82 | PDF | p |  |
| 83 | Event description | s |  |
| 84 | Institution | s | 85, 86, 87 |
| 85 | Institution name | s |  |
| 86 | Institution location | s |  |
| 87 | Institution phone number | s |  |
| 88 | Event | s | 62, 63, 64, 65, 83 |

5.2.2 Potential Data Objects

* **User:** 5, 6, 7, 48
* **Problem:** 15, 41, 42, 49, 50, 51, 52, 53, 54
* **Location:** 43, 44
* **Admin:** 79, 7
* **Institute:** 85, 86, 87
* **Notice:** 61, 63
* **Event:** 62, 63, 64, 65, 83

5.2.3 Analysis for Finalizing Data Objects

* **User** contains user information who submits problems.
* **Problems** holds the information which describes a specific problem.
* **Location** contains longitude and latitude of a specific problem or an institution.
* **Institute** keeps the information of emergency support institutions.
* **Notice** holds the publishing date and the content of notice.
* **Event** contains event information.

5.2.4 Final Data Objects

|  |  |  |
| --- | --- | --- |
| No | Entity | Attributes |
| 1 | User | name, phone\_number, reward\_point, password |
| 2 | Problem | problem\_ID, problem\_description, problem\_category, submission\_date, submission\_time, status, problem\_priority, ward\_number, visibility |
| 3 | Location | latitude, longitude, location\_ID |
| 4 | Admin | email, password |
| 5 | Institution | institute\_name, institute\_location, instutute\_phone\_number, institution\_category |
| 6 | Notice | notice\_content, notice\_id, notice\_publishing\_date |
| 7 | Event | event\_name, event\_id, event\_publishing\_date, event\_date, event\_time, event\_description |

5.3 Data Object Relations

In our “Complain Box” data objects are connected to one another in the ways stated below.

gdfgdfgdf

5.4 Entity Relationship Diagram

5.5 Schema Diagram

|  |  |  |
| --- | --- | --- |
| **User** | | |
| **Attribute** | **Type** | **Size** |
| name | varchar2 | 20 |
| phone\_number | varchar2 | 14 |
| reward\_point | number | 5 |
| password | varchar2 | 15 |

|  |  |  |
| --- | --- | --- |
| **Problem** | | |
| **Attribute** | **Type** | **Size** |
| p\_priority | number | 5 |
| p\_category | varchar2 | 20 |
| ward\_number | number | 5 |
| p\_ID | varchar2 | 20 |
| status | varchar2 | 20 |
| s\_time | varchar2 | 20 |
| s\_date | varchar2 | 20 |
| visibility | varchar2 | 20 |
| p\_description | varchar2 | 500 |
| phone\_number | varchar2 | 14 |
| l\_ID | varchar2 | 20 |

|  |  |  |
| --- | --- | --- |
| **Location** | | |
| **Attribute** | **Type** | **Size** |
| l\_ID | varchar2 | 20 |
| latitude | varchar2 | 20 |
| longitud | varchar2 | 20 |

|  |  |  |
| --- | --- | --- |
| **Institute** | | |
| **Attribute** | **Type** | **Size** |
| i\_name | varchar2 | 50 |
| i\_location | varchar2 | 100 |
| i\_phone\_number | varchar2 | 14 |
| i\_category | varchar2 | 20 |
| l\_ID | varchar2 | 20 |

|  |  |  |
| --- | --- | --- |
| **Admin** | | |
| **Attribute** | **Type** | **Size** |
| email | varchar2 | 20 |
| password | varchar2 | 20 |

|  |  |  |
| --- | --- | --- |
| **Event** | | |
| **Attribute** | **Type** | **Size** |
| e\_name | varchar2 | 50 |
| ep\_date | varchar2 | 20 |
| e\_date | varchar2 | 20 |
| e\_description | varchar2 | 500 |
| e\_time | varchar2 | 20 |
| email | varchar2 | 20 |
| e\_ID | varchar2 | 20 |

|  |  |  |
| --- | --- | --- |
| **Update\_Problem** | | |
| **Attribute** | **Type** | **Size** |
| p\_ID | varchar2 | 20 |
| email | varchar2 | 20 |
| pu\_date | varchar2 | 20 |
| pu\_type | varchar2 | 20 |

|  |  |  |
| --- | --- | --- |
| **Update\_Institute** | | |
| **Attribute** | **Type** | **Size** |
| email | varchar2 | 20 |
| i\_name | varchar2 | 50 |

Chapter 6: Class Based Modeling

We intended this chapter to describe class based modeling for our “Complain Box”.

6.1 Introduction

In this chapter, our designed class based model represents the objects that our “Complain Box” will manipulate, the operation that will applied to the objects, relationships between and the collaboration that occur between the classes that are defined.

6.2 Grammatical Parsing and Class Analysis

To analyze class for our system, we first parse all the determined nouns from our usage scenario. Then we categorize them with general classification and selection criteria. We identify potential classes by identifying the nouns from the scenery. Then we compared those with the following criteria whether they match or not. We noted down the number of the fulfilled criteria at the right column.

6.2.1 General Classification

To identify the potential class, we have to first select the nouns from the solution space of the story. These were then characterized in seven general classifications. The seven general characteristics are as follows:

1. External entities
2. Things
3. Events
4. Roles
5. Organizational units
6. Places
7. Structures

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Nouns** | **Nouns** | **Problem/Solution Space** | **General Classification** |
| 1 | Dhaka City Corporation | p |  |
| 2 | User | s | 1, 4, 5, 7 |
| 3 | Account | s | 2 |
| 4 | System | p |  |
| 5 | Name | s | 2 |
| 6 | Phone number | s | 2 |
| 7 | Password | s | 2 |
| 8 | Five | p |  |
| 9 | Character | p |  |
| 10 | Fifteen | p |  |
| 11 | Subsystem | p |  |
| 12 | Information | p |  |
| 13 | Log | p |  |
| 14 | Message | p |  |
| 15 | Problem Category | s | 2 |
| 16 | Solid Waste | p |  |
| 17 | Road | p |  |
| 18 | Drainage | p |  |
| 19 | Municipal Property | p |  |
| 20 | Road light | p |  |
| 21 | Illegal parking | p |  |
| 22 | Mosquito problem | p |  |
| 23 | Health Department | p |  |
| 24 | Revenue Department | p |  |
| 25 | Social Welfare Department | p |  |
| 26 | Planning Department | p |  |
| 27 | Miscellaneous Complains | p |  |
| 28 | Establishment Sections | p |  |
| 29 | Encroachment | p |  |
| 30 | Factories | p |  |
| 31 | License | p |  |
| 32 | Corruption | p |  |
| 33 | Footpath | p |  |
| 34 | Drain | p |  |
| 35 | Picture | p |  |
| 36 | Video | p |  |
| 37 | Audio | p |  |
| 38 | File | p |  |
| 39 | Storage | p |  |
| 40 | Problem | s | 2, 4, 7 |
| 41 | Description | s | 2 |
| 42 | Problem Location | s | 2 |
| 43 | Latitude | s | 2 |
| 44 | Longitude | s | 2 |
| 45 | Current Location | p |  |
| 46 | Area | p |  |
| 47 | User name | s | 2 |
| 48 | Reward point | s | 2 |
| 49 | Problem ID | s | 2 |
| 50 | Submission date | s | 2 |
| 51 | Submission time | s | 2 |
| 52 | Status | s | 2 |
| 53 | Problem priority | s | 2 |
| 54 | Ward number | s | 2 |
| 55 | Communication facility | p |  |
| 56 | Hospital | p |  |
| 57 | Fire station | p |  |
| 58 | Police station | p |  |
| 59 | Name of institute | s | 2 |
| 60 | Notice | s | 2, 3 |
| 61 | Notice content | s | 2 |
| 62 | Event name | s | 2 |
| 63 | Publishing date | s | 2 |
| 64 | Event date | s | 2 |
| 65 | Event time | s | 2 |
| 66 | Application form | s | 2, 4, 7 |
| 67 | Passport application form | p |  |
| 68 | E-income tax certificate | p |  |
| 69 | Driving license form | p |  |
| 70 | Birth certificate | p |  |
| 71 | Trade license form | p |  |
| 72 | Tax form | p |  |
| 73 | Death certificate | p |  |
| 74 | Inherit certificate | p |  |
| 75 | Admin | s | 1, 3, 4, 7 |
| 76 | Person | p |  |
| 77 | Decision | p |  |

6.2.2 Selection Criteria

The nouns having two or more than two were selected from the general classification list. After that step, we compared them with the following criteria list. Those are

1. Retained information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

|  |  |  |
| --- | --- | --- |
| **Number of Nouns** | **Nouns** | **Selection Classification** |
| 1 | User | 1, 3 |
| 2 | Problem | 1, 2, 3, 4, 5 |
| 3 | Application form | 1, 2, 3 |
| 4 | Admin | 1, 3 |
| 5 | Institution | 1, 2, 3, 4, 5 |
| 6 | Notice | 1, 3, 4, 5 |
| 7 | Event | 1, 3, 4, 5 |

6.2.3 Preliminary Classes

From above table, we have taken all the noun who passed a single criteria. From further analysis we also got, Location is also need for our system as a class. So these are the candidate classes who are selected primarily:

1. User
2. Problem
3. Admin
4. Application form
5. Institute
6. Notice
7. Event
8. Location

Besides, we divided our system as two individual subsystem (Problem submission and Problem supervision). Thus we designed our system two different part working together through a common server. So our each subsystem holds its own set of classes.

Problem submission is designed with:

1. User
2. Problem
3. Application form
4. Institute
5. Notice
6. Event
7. Location

Problem Supervision is designed with:

1. Problem
2. Admin
3. Application form
4. Institute
5. Notice
6. Event
7. Location

6.3 Associated Noun and Verb Identification

For identifying attributes and methods we choose to parse associate nouns and verbs of a class from our usage scenario.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Nouns** | **Potential Classes** | **Nouns** | **Verbs** |
| 1 | User | name, phone number, reward points, password | create, log in, log out, view, send, capture, record, select, contain, recover |
| 2 | Problem | problem ID, problem description, problem category, problem priority, submission date, submission time, problem status, ward number, visibility, location | submit, search, view, update, face, change, contain, |
| 3 | Application form | application ID, file path | add, view, update, download, include |
| 4 | Admin | email, password | log in, log out, send, operate, decide, create, request, view, edit, select, update, upload, change, recover |
| 5 | Institution | institute name, institute location, institute phone number, institution category | add, update, view, call |
| 6 | Notice | notice ID, notice content, notice publishing date | create, view |
| 7 | Location | latitude, longitude | view |
| 8 | Event | event ID, event name, event publishing date, event date, event time, event description | create, view |

6.4 Attribute Selection

From our selected nouns associated with individual class we finalize attribute for each class.

|  |  |  |
| --- | --- | --- |
| **Number of Nouns** | **Potential Classes** | **Attributes** |
| 1 | User | * name * phoneNumber * rewardPoints * password |
| 2 | Problem | * problemID * problemDescription * problemCategory * problemPriorit * submissionDate * submissionTime * problemStatus * wardNumber * visibility * location |
| 3 | Application form | * applicationID * filePath |
| 4 | Admin | * email * password |
| 5 | Institution | * instituteName * instituteLocation * institutePhoneNumber * institutionCategory |
| 6 | Notice | * noticeID * noticeContent * noticePublishingDate |
| 7 | Location | * locationID * latitude * longitude |
| 8 | Event | * eventID * eventName * eventPublishingDate * eventDate * eventTime * eventDescription |

6.5 Method Selection

From our selected nouns associated with individual class we finalize methods for each class.

|  |  |  |
| --- | --- | --- |
| **Number of Nouns** | **Potential Classes** | **Methods** |
| 1 | User | * createAccount() * logIn() * logOut() * viewUserInformation() * recoverPassword() |
| 2 | Problem | * submitProblem() * serachProblem() * viewProblem() * viewProblemLocation() * updatePriority() * updateStatus() * updateCategory() * capturePicture() * selectProblemDescribingFile() * addProblemDescription() * addProblemLocation() * selectProblemCategory() |
| 3 | Application form | * addApplicationForm() * viewApplicationForm() * updateApplicationForm() * downoadApplicationForm() |
| 4 | Admin | * logIn() * logOut() * recoverAccount() * updateSystemInformation() |
| 5 | Institution | * addInstitute() * updateInstitute() * viewInstituteInsformation() * viewInstituteMap() * callToInstitute() |
| 6 | Notice | * createNotice() * viewNotice() |
| 7 | Location | * viewLocation() |
| 8 | Event | * createEvent() * viewEvent() |

6.6 Finalized Classes

From our identified attributes and methods we finalized class for our both sub systems.

6.6.1 Finalized classes for Problem Submission

|  |  |
| --- | --- |
| **User** | |
| **Attributes** | **Methods** |
| * name * phoneNumber * rewardPoints * passward | * createAccount() * logIn() * logOut() * viewUserInformation() * createProblemInformation() * recoverPassword() |

|  |  |
| --- | --- |
| **Problem** | |
| **Attributes** | **Methods** |
| * problemID * problemDescription * problemCategory * problemPriorit * submissionDate * submissionTime * problemStatus * wardNumber * visibility * location | * submitProblem() * serachProblem() * viewProblem() * capturePicture() * selectProblemDescribingFile() * addProblemDescription() * addProblemLocation() * selectProblemCategory() |

|  |  |
| --- | --- |
| **ApplicationForm** | |
| **Attributes** | **Methods** |
| * applicationID * filePath | * viewApplicationForm() * downoadApplicationForm() |

|  |  |
| --- | --- |
| **Institution** | |
| **Attributes** | **Methods** |
| * instituteName * instituteLocation * institutePhoneNumber * institutionCategory | * viewInstituteInformation() * callToInstitute() |

|  |  |
| --- | --- |
| **Notice** | |
| **Attributes** | **Methods** |
| * noticeID * noticeContent * noticePublishingDate | * viewNotice() |

|  |  |
| --- | --- |
| **Event** | |
| **Attributes** | **Methods** |
| * eventID * eventName * eventPublishingDate * eventDate * eventTime * eventDescription | * viewEvent() |

|  |  |
| --- | --- |
| **Location** | |
| **Attributes** | **Methods** |
| * longitude * Latitude * locationID | * viewLocation() |

6.6.2 Finalized classes for Problem Supervision

|  |  |
| --- | --- |
| **Problem** | |
| **Attributes** | **Methods** |
| * problemID * problemDescription * problemCategory * problemPriorit * submissionDate * submissionTime * problemStatus * wardNumber * visibility * location | * serachProblem() * viewProblem() * updatePriority() * updateStatus() * updateCategory() |

|  |  |
| --- | --- |
| **ApplicationForm** | |
| **Attributes** | **Methods** |
| * applicationID * filePath | * addApplicationForm() * viewApplicationForm() * updateApplicationForm() * downoadApplicationForm() |

|  |  |
| --- | --- |
| **Admin** | |
| **Attributes** | **Methods** |
| * email * password | * logIn() * logOut() * updateSystemInformation() * recoverAccount() |

|  |  |
| --- | --- |
| **Institution** | |
| **Attributes** | **Methods** |
| * instituteName * instituteLocation * institutePhoneNumber * institutionCategory | * addInstitute() * updateInstitute() * viewInstituteInformation() |

|  |  |
| --- | --- |
| **Notice** | |
| **Attributes** | **Methods** |
| * noticeID * noticeContent * noticePublishingDate | * createNotice() * viewNotice() |

|  |  |
| --- | --- |
| **Event** | |
| **Attributes** | **Methods** |
| * eventID * eventName * eventPublishingDate * eventDate * eventTime * eventDescription | * createEvent() * viewEvent() |

|  |  |
| --- | --- |
| **Location** | |
| **Attributes** | **Methods** |
| * longitude * Latitude * locationID | * viewLocation() |

6.7 Class Responsibility Collaborator Modeling

In this stage, we identify the responsibility and collaborators of each class. It will provide a simple means for identifying and organizing the classes that are relevant to our system requirements. We modeled this for both subsystem of our “Complain Box”.

6.7.1 Class Responsibility Collaborator Modeling for Problem Submission

|  |  |
| --- | --- |
| **Class: User** | |
| **Responsibilities:** | **Collaborators:** |
| Stores user information | Problem |
| Authenticates to the system |  |
| Creates problem information | Problem |
| Shows user detail information |  |
| Recovers Password |  |

|  |  |
| --- | --- |
| **Class: Problem** | |
| **Responsibilities:** | **Collaborators:** |
| Stores problem information |  |
| Shows problem detail information | Location |

|  |  |
| --- | --- |
| **Class: ApplicationForm** | |
| **Responsibilities:** | **Collaborators:** |
| Downloads application form |  |
| Shows information of application form |  |

|  |  |
| --- | --- |
| **Class: Institution** | |
| **Responsibilities:** | **Collaborators:** |
| Shows institute information | Location |
| Makes a call to institute |  |

|  |  |
| --- | --- |
| **Class: Notice** | |
| **Responsibilities:** | **Collaborators:** |
| Shows notice information |  |

|  |  |
| --- | --- |
| **Class: Event** | |
| **Responsibilities:** | **Collaborators:** |
| Shows event information |  |

|  |  |
| --- | --- |
| **Location** | |
| **Responsibilities:** | **Collaborators:** |
| Shows location in map |  |

6.7.2 Class Responsibility Collaborator Modeling for Problem Supervision

|  |  |
| --- | --- |
| Problem | |
| Responsibilities | Collaborative Classes |
| Show problem detail information  Update problem Information | Location |

|  |  |
| --- | --- |
| ApplicationForm | |
| Attributes | Methods |
| * applicationID * filePath | * addApplicationForm() * viewApplicationForm() * updateApplicationForm() * downoadApplicationForm() |
| Responsibilities | Collaborative Classes |
| Download application form  Show information of application form  Update application form |  |

|  |  |
| --- | --- |
| Admin | |
| Attributes | Methods |
| * email * password |  logIn()   logOut()   updateSystemInformation()   recoverAccount() |
| Responsibilities | Collaborative Classes |
| Authenticate to the system  Update system information | Problem  Event  Notice  Application form  Institution |

|  |  |
| --- | --- |
| Institution | |
| Attributes | Methods |
| * instituteName * instituteLocation * institutePhoneNumber * institutionCategory | * addInstitute() * updateInstitute() * viewInstituteInformation() |
| Responsibilities | Collaborative Classes |
| Show institute information  Update institute information | Location |

|  |  |
| --- | --- |
| Notice | |
| Attributes | Methods |
| * noticeID * noticeContent * noticePublishingDate | * createNotice() * viewNotice() |
| Responsibilities | Collaborative Classes |
| Show notice information |  |

|  |  |
| --- | --- |
| Event | |
| Attributes | Methods |
| * eventID * eventName * eventPublishingDate * eventDate * eventTime * eventDescription | * createEvent() * viewEvent() |
| Responsibilities | Collaborative Classes |
| Show event information |  |

|  |  |
| --- | --- |
| **Location** | |
| **Responsibilities:** | **Collaborators:** |
| Shows location in map |  |