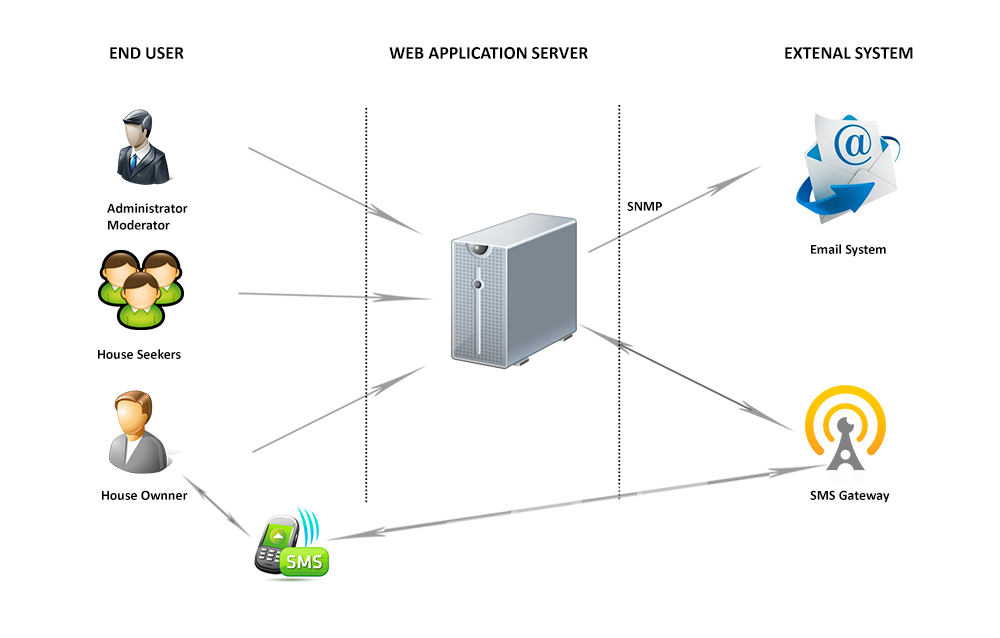
## 4.1 Design Overview

### 4.1.1 System Context

#### 4.1.1.1 Diagram



#### 4.1.1.2 System Context Description

|  |  |
| --- | --- |
| User/Module/System | Description |
| End User | Users access the web application via Internet connection. |
| Web Application Server | Automatically intercepts any user requests and send dynamic content |
| SMS Gateway | Get data from Database and then interact with user through SMS and then send user’s feedback to system |
| Email System | Get data from Database and then send Email to users automatically |

## 4.2 Architecture Overview

### 4.2.1 Diagram



|  |  |  |  |
| --- | --- | --- | --- |
| # | Namespace | Description | Naming Convention |
| 1 | f2s.page.home | Contains all GUI pages in home screen |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |

### 4.2.2 Technology and Tool

This section describes the technologies and tools that have some significant impact on the architecture.

### 4.2.2.1 Operating System

The operating system will be used for .NET applications are shown in the following table:

|  |  |
| --- | --- |
| Components | Operating System |
| Application Server | Window server 2003/2008 |
| Database Server | MS SQL Server 2008 |
| Web Server | Window server 2003/2008 |
| Client (Web User Interface) | Windows XP/2003/Vista/7 |

### 4.2.2.2 MVC Architecture

Model – View – Controller (MVC) is a type of computer user interface that separates the representation of information from the user’s interaction with it. It separates an application into three main components: the model, the view and the controller.

The model consists of application data and business rule, and the controller mediates input, converting it to commands for the models or view. A view can be any input representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a pie chart, for management and a tabular view for accountants. The central idea behind MVC is code reusability and separation of concerns.

In addition to managing complexity, the MVC pattern makes it easier to test applications than it is to test a Web Forms-based ASP.NET Web application.

*Source: http://en.wikipedia.org/wiki/Model–view–controller*

### 4.2.2.3 Frameworks/Components

Existing components and framework functionality will be utilized by the application. This section discusses these components and the problems they solve. The features these components provide may be used as is or customized to fit the needs of the application. Only selected features of each component will be utilized.

#### 4.2.2.3.1 ADO.NET Entity Framework

A primary goal of the upcoming version of ADO.NET is to raise the level of abstraction for data programming, thus helping to eliminate the impedance mismatch between data models and between languages that application developers would otherwise have to deal with. Two innovations that make this move possible are Language-Integrated Query and the ADO.NET Entity Framework. The Entity Framework exists as a new part of the ADO.NET family of technologies. ADO.NET will LINQ-enable many data access components: LINQ to SQL, LINQ to DataSet and LINQ to Entities.

The ADO.NET Entity Framework enables developers to create data access applications by programming against a conceptual application model instead of programming directly against a relational storage schema. The goal is to decrease the amount of code and maintenance required for data-oriented applications. Entity Framework applications provide the following benefits:

* Applications can work in terms of a more application-centric conceptual model, including types with inheritance, complex members, and relationships.
* Applications are freed from hard-coded dependencies on a particular data engine or storage schema.
* Mappings between the conceptual model and the storage-specific schema can change without changing the application code.
* Developers can work with a consistent application object model that can be mapped to various storage schemas, possibly implemented in different database management systems.
* Multiple conceptual models can be mapped to a single storage schema.
* Language-integrated query (LINQ) support provides compile-time syntax validation for queries against a conceptual model.

*Source: http://msdn.microsoft.com/en-us/library/aa697427(v=vs.80).aspx*

#### 4.2.2.3.2 Telerik – Presentation Framework

Telerik Extensions for ASP.NET MVC is a lightweight framework which helps you build rich user interfaces for ASP.NET MVC applications while enjoying great developer productivity.

Telerik Extensions for ASP.NET MVC has the following features:

* Based on jQuery.
* Extensions of the HtmlHelper object for rendering UI elements.
* Rich web asset (JavaScript and CSS) management
  + GZIP compression of resources to save some kilobytes of traffic.
  + Combination of many resources in one request to reduce the number of requests made to the web server.
  + Componentization of resources and grouping - you can combine and compress individual groups of web assets.
  + CDN support. After testing your web site you can deploy your web assets on a Content Distribution Network and easily link them in your ASP.NET MVC application.
* Support for multiple view engines - apart from the default view engine you can use any other view engine that supports ASP.NET MVC: Spark, NHaml etc.

### 4.2.2.4 Development Tools

|  |  |  |
| --- | --- | --- |
| Tool | Version | Description |
| Visual Studio | 2010 | **Microsoft Visual Studio** is an integrated development environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web application, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Window Mobile, Window CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight. |
| Firebug | 1.10.0 | Debug request, response, HTML, CSS, JavaScript |