# Functional Requirement

# 1. Functional Requirements   
  
## 1.1 Email Creation Function   
\*\*Function ID\*\*: FR-01   
\*\*Description\*\*: Administrators can create a new email with specified content and recipients. The email is associated with a specific email account and optionally scheduled via calendar.   
\*\*Input\*\*: Email subject, body, recipient (Contact or Distribution Group), sender (Email Account), optional schedule date/time.   
\*\*Output\*\*: A new Email entity is stored in the server storage and optionally added to a CalendarEvent.   
  
---  
  
## 1.2 Email Sending Function   
\*\*Function ID\*\*: FR-02   
\*\*Description\*\*: Administrators can send an email using a specified email account. Recipients must be valid Contacts or members of a Distribution Group.   
\*\*Input\*\*: Email content, sender email account, recipient list (Contact or Distribution Group), optional schedule date/time.   
\*\*Output\*\*: The Email is sent and stored in the server storage, and optionally added to the calendar.   
  
---  
  
## 1.3 Email Receiving Function   
\*\*Function ID\*\*: FR-03   
\*\*Description\*\*: The system automatically checks for new emails in configured email accounts and stores them in server storage.   
\*\*Input\*\*: Email account configuration and server connection details.   
\*\*Output\*\*: New Email entity is stored in server storage and optionally archived or logged.   
  
---  
  
## 1.4 Email Viewing Function   
\*\*Function ID\*\*: FR-04   
\*\*Description\*\*: Administrators can view the details of an existing email from the server storage.   
\*\*Input\*\*: Selected EmailID and Email Account information.   
\*\*Output\*\*: Email details (Subject, Body, Sender, Recipient, Attachments, Send/Receive date) displayed in the user interface.   
  
---  
  
## 1.5 Email Deletion Function   
\*\*Function ID\*\*: FR-05   
\*\*Description\*\*: Administrators can delete an email from the server storage.   
\*\*Input\*\*: Selected EmailID and confirmation to delete.   
\*\*Output\*\*: Email is removed from server storage and the email list is updated.   
  
---  
  
## 1.6 Email Archiving Function   
\*\*Function ID\*\*: FR-06   
\*\*Description\*\*: Administrators can move an email from server storage to an archive based on archive policies.   
\*\*Input\*\*: Selected EmailID, confirmation to archive, and policy details.   
\*\*Output\*\*: Email is archived, and the server storage and email list are updated.   
  
---  
  
## 1.7 Archived Email Search Function   
\*\*Function ID\*\*: FR-07   
\*\*Description\*\*: Administrators can search for archived emails using filters like subject, sender, date, or keywords.   
\*\*Input\*\*: Search criteria (e.g., subject, sender, date range).   
\*\*Output\*\*: List of matching archived Email entities displayed in the user interface.   
  
---  
  
## 1.8 Contact Management Function   
\*\*Function ID\*\*: FR-08   
\*\*Description\*\*: Administrators can add, edit, or delete contact information and associate it with a user.   
\*\*Input\*\*: Contact details (Name, Email, Phone, UserID).   
\*\*Output\*\*: Updated Contact list is stored in the system and available for use in email sending.   
  
---  
  
## 1.9 Distribution Group Management Function   
\*\*Function ID\*\*: FR-09   
\*\*Description\*\*: Administrators can create, update, or delete a distribution group and manage its members (Contacts or Email Accounts).   
\*\*Input\*\*: Group name, selected contacts or email accounts.   
\*\*Output\*\*: Updated DistributionGroup entity is stored in the system and available for use in email sending.   
  
---  
  
## 1.10 Calendar Management Function   
\*\*Function ID\*\*: FR-10   
\*\*Description\*\*: Administrators can manage calendar events, including creation, updating, and deletion. Events may be associated with scheduled emails.   
\*\*Input\*\*: Event details (Title, Description, Start and End Time, associated EmailID).   
\*\*Output\*\*: Updated CalendarEvent is stored in the system and reflected in the calendar view.   
  
---  
  
## 1.11 Calendar Event Participant Management Function   
\*\*Function ID\*\*: FR-11   
\*\*Description\*\*: Administrators can add, remove, or update participants of a calendar event.   
\*\*Input\*\*: Selected CalendarEventID, list of participants (UserIDs or EmailAccountIDs), and participant roles.   
\*\*Output\*\*: Updated CalendarEventParticipant list is stored in the system and reflected in the event details.   
  
---  
  
## 1.12 Email Account Management Function   
\*\*Function ID\*\*: FR-12   
\*\*Description\*\*: Administrators can create, update, or delete email accounts and manage their configurations.   
\*\*Input\*\*: Email account details (Username, Password, Creation/LastModified/LastDeleted Dates).   
\*\*Output\*\*: Updated EmailAccount is stored in the system and associated with emails and server storage.   
  
---  
  
## 1.13 Server Storage Management Function   
\*\*Function ID\*\*: FR-13   
\*\*Description\*\*: Administrators can configure, expand, or modify server storage settings.   
\*\*Input\*\*: Server name, capacity settings, backup and retention policies.   
\*\*Output\*\*: Updated ServerStorage entity is stored in the system, and changes are applied to the storage.   
  
---  
  
## 1.14 Archiving Policy Management Function   
\*\*Function ID\*\*: FR-14   
\*\*Description\*\*: Administrators can create, update, or delete archiving policies to define how and when emails are archived.   
\*\*Input\*\*: Policy name, description, archive criteria, and expiry rules.   
\*\*Output\*\*: Updated Policy is stored in the system and applied to eligible emails.   
  
---  
  
## 1.15 Expired Email Management Function   
\*\*Function ID\*\*: FR-15   
\*\*Description\*\*: Administrators can manage expired emails by either recovering or permanently deleting them.   
\*\*Input\*\*: ExpiredEmailID, action (Recover or Delete).   
\*\*Output\*\*: ExpiredEmail is either restored to server storage or removed, and the system logs the activity.   
  
---  
  
## 1.16 Expired Email Recovery Function   
\*\*Function ID\*\*: FR-16   
\*\*Description\*\*: Administrators can recover an expired email from its archive or backup location using the recovery policy.   
\*\*Input\*\*: ExpiredEmailID, confirmation to recover.   
\*\*Output\*\*: ExpiredEmail is restored to server storage and associated with the original EmailAccount.   
  
---  
  
## 1.17 Email Flow Capture Function   
\*\*Function ID\*\*: FR-17   
\*\*Description\*\*: The system captures and logs the flow of emails based on predefined policies.   
\*\*Input\*\*: Email Account ID, capture criteria (source, destination, conditions), and policy configuration.   
\*\*Output\*\*: EmailFlow entity is created and stored in the system.   
  
---  
  
## 1.18 Email Record Audit Function   
\*\*Function ID\*\*: FR-18   
\*\*Description\*\*: Administrators can view and audit logs of email actions (e.g., send, archive, delete) for compliance purposes.   
\*\*Input\*\*: EmailRecordID, filter criteria (e.g., action type, date, user).   
\*\*Output\*\*: EmailRecord details (EmailID, Action, Timestamp, UserID) displayed in the system.   
  
---  
  
## 1.19 Email Folder Management Function   
\*\*Function ID\*\*: FR-19   
\*\*Description\*\*: Administrators can create, rename, delete, or move emails between folders.   
\*\*Input\*\*: Folder name, description, and list of emails to be moved.   
\*\*Output\*\*: Updated EmailFolder and Email associations stored in the system.   
  
---  
  
## 1.20 Email Attachment Management Function   
\*\*Function ID\*\*: FR-20   
\*\*Description\*\*: Administrators can upload, view, rename, or remove attachments from emails.   
\*\*Input\*\*: EmailID, attachment file (filename, size, upload date), action (Add, Remove, Rename).   
\*\*Output\*\*: Updated EmailAttachment and Email entities stored in the system.   
  
---  
  
## 1.21 Administrator Permission Management Function   
\*\*Function ID\*\*: FR-21   
\*\*Description\*\*: Administrators can configure and manage permission levels of other administrator accounts.   
\*\*Input\*\*: AdminID, permission settings (e.g., send email, manage policies, view archive).   
\*\*Output\*\*: Updated Admin entity and associated permissions stored in the system.   
  
---  
  
## 1.22 Administrator Account Update Function   
\*\*Function ID\*\*: FR-22   
\*\*Description\*\*: Administrators can modify their own or other admin accounts' settings, including password and access level.   
\*\*Input\*\*: AdminID, updated settings (e.g., password, notification preferences).   
\*\*Output\*\*: Updated Admin entity stored in the system.   
  
---  
  
## 1.23 Administrator Account Deletion Function   
\*\*Function ID\*\*: FR-23   
\*\*Description\*\*: Administrators can delete an administrator account, and the system will process all related data (e.g., emails, policies, calendar events).   
\*\*Input\*\*: AdminID, confirmation to delete.   
\*\*Output\*\*: Admin account is removed, and associated data is either deleted or transferred.   
  
---  
  
## 1.24 System Logging and Activity Tracking Function   
\*\*Function ID\*\*: FR-24   
\*\*Description\*\*: The system logs all user actions and system events for audit and traceability.   
\*\*Input\*\*: Action type, EmailID, UserID, Timestamp.   
\*\*Output\*\*: EmailRecord or system log entry stored in the server storage.   
  
---  
  
These functional requirements are derived directly from the provided use cases and data model. Each function ensures that the system can be used to manage email communication, contact and distribution group handling, calendar scheduling, and long-term email archiving and recovery in a structured and auditable manner.

# External Description

# Software Requirements Specification (SRS)   
## Chapter 2: External Interfaces   
  
This chapter describes the external interfaces that the system will interact with. It includes the definitions, descriptions, and interaction methods for \*\*User Interfaces\*\*, \*\*Hardware Interfaces\*\*, \*\*Software Interfaces\*\*, and \*\*Communication Interfaces\*\*. These interfaces are identified based on the functional requirements provided and are essential for the system's integration with external components.  
  
---  
  
## 2.1 User Interface   
  
The system interacts with administrators through a \*\*graphical user interface (GUI)\*\*, which supports the full range of operations related to email, contacts, distribution groups, calendar events, and system settings. The interface is designed to be intuitive and role-based, with access to functions determined by the administrator’s permissions.   
  
### 2.1.1 Email Management UI   
- \*\*Description\*\*: A UI module for creating, viewing, deleting, archiving, and managing folders and attachments for emails.   
- \*\*Interaction Method\*\*:   
 - Input: Email subject, body, sender (Email Account), recipient (Contact or Distribution Group), optional schedule date/time, folder name, attachment file, action (e.g., send, archive, delete).   
 - Output: Display of Email details (Subject, Body, Sender, Recipient, Attachments, Send/Receive date), updated email list, folder structure, and confirmation messages.   
  
### 2.1.2 Contact and Distribution Group Management UI   
- \*\*Description\*\*: A UI module for managing Contacts and Distribution Groups, including adding, editing, and deleting entries.   
- \*\*Interaction Method\*\*:   
 - Input: Contact details (Name, Email, Phone, UserID), Distribution Group name and members (Contacts or Email Accounts).   
 - Output: Updated Contact list and Distribution Group list displayed in the UI.   
  
### 2.1.3 Calendar and Event Management UI   
- \*\*Description\*\*: A UI module for creating, updating, and viewing Calendar Events, and managing participants.   
- \*\*Interaction Method\*\*:   
 - Input: Event details (Title, Description, Start and End Time), participant list (UserIDs or EmailAccountIDs), participant roles.   
 - Output: Display of CalendarEvent details, updated event list, and participant list.   
  
### 2.1.4 Archiving and Policy Management UI   
- \*\*Description\*\*: A UI module for managing archiving policies and recovering expired emails.   
- \*\*Interaction Method\*\*:   
 - Input: Policy name, description, archive criteria, expiry rules, ExpiredEmailID, action (Recover or Delete).   
 - Output: Updated Policy list, archived email list, and confirmation of recovery or deletion.   
  
### 2.1.5 Administrator and System Settings UI   
- \*\*Description\*\*: A UI module for managing administrator accounts and permissions, system logging, and server storage settings.   
- \*\*Interaction Method\*\*:   
 - Input: AdminID, password, permission settings, server name, capacity settings, backup and retention policies.   
 - Output: Updated Admin entity, updated ServerStorage entity, and confirmation of account changes.   
  
### 2.1.6 Email Audit and Logging UI   
- \*\*Description\*\*: A UI module for viewing logs of email actions (e.g., send, archive, delete) for compliance and traceability.   
- \*\*Interaction Method\*\*:   
 - Input: EmailRecordID, filter criteria (e.g., action type, date, user).   
 - Output: Display of EmailRecord details (EmailID, Action, Timestamp, UserID) for audit purposes.   
  
---  
  
## 2.2 Hardware Interface   
  
This system does not have direct dependencies or interactions with hardware devices. It is primarily a software-based system hosted on standard server infrastructure and accessed via networked devices (e.g., desktops, laptops, or mobile devices). However, it may indirectly interact with hardware through the following ways:  
  
### 2.2.1 Server Storage Devices   
- \*\*Description\*\*: The system stores and retrieves email data, calendar events, and logs from server storage, which may include physical or virtual storage devices such as hard drives, SSDs, or cloud storage.   
- \*\*Interaction Method\*\*:   
 - Input: Server name, capacity settings, backup and retention policies.   
 - Output: Updated ServerStorage entity reflecting the new settings, and persistent storage of Email, CalendarEvent, and EmailRecord entities.   
- \*\*Notes\*\*: The hardware interface is abstracted via the software interface for ServerStorage, and developers should not need to interact directly with the hardware.   
  
---  
  
## 2.3 Software Interface   
  
The system interacts with several software components, including databases, external APIs, and internal data models. These interfaces are crucial for data persistence, data exchange, and system integration.  
  
### 2.3.1 Database Interface   
- \*\*Description\*\*: The system uses a database to persist and retrieve data such as Email, Contact, DistributionGroup, CalendarEvent, EmailAccount, ServerStorage, Policy, EmailFolder, EmailAttachment, Admin, and EmailRecord entities.   
- \*\*Interaction Method\*\*:   
 - Input: Entities (e.g., Email, CalendarEvent, EmailAccount) and their attributes (e.g., subject, body, sender, recipient, event time, policy criteria).   
 - Output: Persisted or updated entities, query results based on search criteria (e.g., archived emails, logs, contacts).   
- \*\*Notes\*\*: The database interface supports SQL queries or ORM-based operations depending on the implementation.   
  
### 2.3.2 Email Server API   
- \*\*Description\*\*: The system communicates with external email servers (e.g., SMTP, IMAP, POP3) to send, receive, and manage emails.   
- \*\*Interaction Method\*\*:   
 - Input: Email content, sender email account, recipient list (Contact or Distribution Group), optional schedule date/time.   
 - Output: Email sent to recipients, new Email entity stored in the system, or confirmation of successful receipt.   
  
### 2.3.3 Calendar System API   
- \*\*Description\*\*: The system may interface with a calendar system (e.g., Google Calendar, Microsoft Outlook, or a custom calendar) to create and manage CalendarEvents associated with scheduled emails.   
- \*\*Interaction Method\*\*:   
 - Input: Event details (Title, Description, Start and End Time, associated EmailID), participant list (UserIDs or EmailAccountIDs), and participant roles.   
 - Output: Updated CalendarEvent in the external system, including participants and event details.   
  
### 2.3.4 File Storage API   
- \*\*Description\*\*: The system uses a file storage API (e.g., local file system or cloud storage) to manage attachments associated with emails.   
- \*\*Interaction Method\*\*:   
 - Input: EmailID, attachment file (filename, size, upload date), action (Add, Remove, Rename).   
 - Output: Updated EmailAttachment entity and associated Email entity in the database.   
  
---  
  
## 2.4 Communication Interface   
  
The system communicates with external systems and users through network-based protocols and messaging systems to support sending emails, receiving emails, and logging activities.  
  
### 2.4.1 Email Communication Interface   
- \*\*Description\*\*: The system sends and receives emails via standard email protocols (e.g., SMTP for sending, IMAP/POP3 for receiving).   
- \*\*Interaction Method\*\*:   
 - Input: Email content, sender email account, recipient list (Contact or Distribution Group), optional schedule date/time.   
 - Output: Email sent to recipients or received from external email servers and stored in the system.   
  
### 2.4.2 Web Communication Interface   
- \*\*Description\*\*: The system communicates with users and administrators via a web-based interface using HTTP/HTTPS protocols.   
- \*\*Interaction Method\*\*:   
 - Input: User input via web forms or API calls (e.g., creating an email, updating a policy).   
 - Output: JSON/XML responses or HTML rendering of data for display in the UI.   
  
### 2.4.3 System Logging and Audit Communication   
- \*\*Description\*\*: The system logs all user actions and system events for audit and traceability purposes. These logs may be stored in a centralized logging system or database.   
- \*\*Interaction Method\*\*:   
 - Input: Action type, EmailID, UserID, Timestamp.   
 - Output: EmailRecord or system log entry stored in the server storage.   
  
---  
  
## Cross-Check Summary   
  
| Functional Requirement | External Interface(s) Involved |  
|------------------------|-------------------------------|  
| FR-01 (Email Creation) | User Interface, Database Interface, Email Server API |  
| FR-02 (Email Sending) | User Interface, Email Server API, Calendar System API, Database Interface |  
| FR-03 (Email Receiving) | Email Server API, Database Interface |  
| FR-04 (Email Viewing) | User Interface, Database Interface |  
| FR-05 (Email Deletion) | User Interface, Database Interface |  
| FR-06 (Email Archiving) | User Interface, Database Interface |  
| FR-07 (Archived Email Search) | User Interface, Database Interface |  
| FR-08 (Contact Management) | User Interface, Database Interface |  
| FR-09 (Distribution Group Management) | User Interface, Database Interface |  
| FR-10 (Calendar Management) | User Interface, Calendar System API, Database Interface |  
| FR-11 (Calendar Event Participant Management) | User Interface, Calendar System API, Database Interface |  
| FR-12 (Email Account Management) | User Interface, Email Server API, Database Interface |  
| FR-13 (Server Storage Management) | User Interface, Database Interface, Server Storage Devices |  
| FR-14 (Archiving Policy Management) | User Interface, Database Interface |  
| FR-15 (Expired Email Management) | User Interface, Database Interface |  
| FR-16 (Expired Email Recovery) | User Interface, Database Interface, File Storage API |  
| FR-17 (Email Flow Capture) | Email Server API, Database Interface |  
| FR-18 (Email Record Audit) | User Interface, Database Interface |  
| FR-19 (Email Folder Management) | User Interface, Database Interface |  
| FR-20 (Email Attachment Management) | User Interface, File Storage API, Database Interface |  
| FR-21 (Admin Permission Management) | User Interface, Database Interface |  
| FR-22 (Admin Account Update) | User Interface, Database Interface |  
| FR-23 (Admin Account Deletion) | User Interface, Database Interface |  
| FR-24 (System Logging and Activity Tracking) | User Interface, Database Interface, System Logging Interface |  
  
---  
  
## Conclusion   
  
The external interfaces defined in this chapter are critical for the system to function as intended. They ensure seamless interaction with users, external APIs, databases, and file systems. Developers should ensure that these interfaces are implemented in a consistent and secure manner, respecting the input/output formats and interaction methods described.