CS 3354 Software Engineering Project

Spring 2017

Phase Two Web Mail System Design Document

Team 9

Kha Nguyen

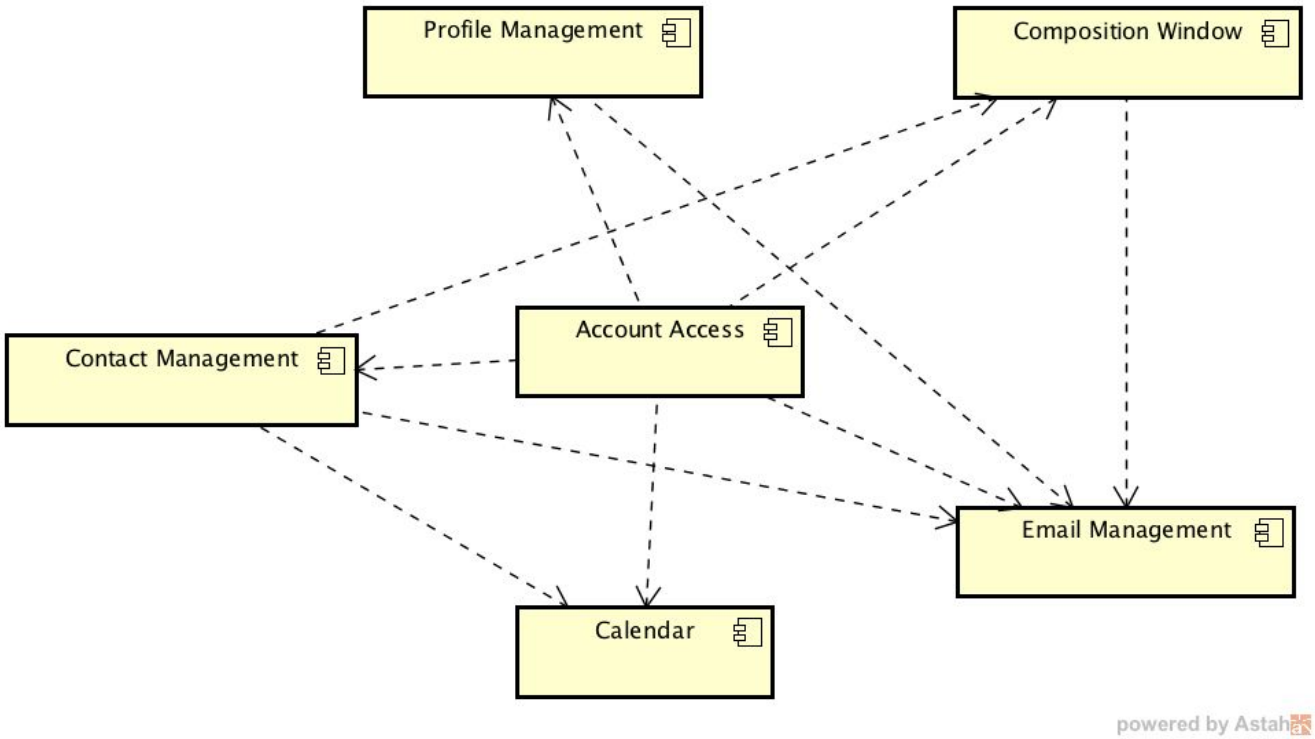
David Stenson

Nicole Brauer

Design Overview

The system is designed as a web-based application serving as a mail client for users to send and receive email. Users can also perform tasks such as managing folders, contacts, emails, and editing profiles and preferences. Account Access allows the User to enter into the system, all other actions besides account creation must take place after using Account Access to log in. Other Components can be accessed after this fact. Some components such as email and calendar have the ability to connect objects to multiple accounts to send emails and share events. Profile Management and Account Management are private to each user and are only connected to a single account.

Major System Components



Component Descriptions

Each of the following sections provides a description of the components shown in this diagram.

Profile Management:

Profile Management is in charge of superfluous profile information not related to logging in the customer. It contains profile names, addresses, profile pictures and preferences. It allows for changes to the profile including blocking other email addresses. The component also controls changing of password while a user is logged in.

Account Access:

Account access manages who can access parts of the system. It verifies the identity of a User by comparing their username and password to a database, and making sure both are connected to the same User. Once it verifies a User it allows them to access to the various resources of the system. It also manages what information the user can view and edit to only those associated with the

User.

Composition Window:

Composition window is used to fill the information for emails. Allows for the input of email addresses, selection for send, CC or BCC. Allows for the attachment of other files to the email object from the User's computer. Collects all the information and compiles it into an email package.

Contact Management:

Manages Contacts within each User's contact group and additional information about them. Contact management saves each contact's address, and the name given to them by the User. Also saves the groups that User's can create containing subsets of Contacts.

Calendar:

Calendar controls events that can be created on certain dates by the user. It allows the user to share events between contacts in their contact list. Calendar also allows for alarms to be created connected to events to warn the User.

Email Management:

Email management holds all the emails created by the system. It distributes read access to users that have been sent the email. It controls the folders for each user's individual mailbox. It also sorts emails into spam, inbox, and uses the black list to automatically block emails from certain addresses.

Design Patterns

Pattern: Three-Tier Architecture Pattern

The high-level design (architecture) of this system has been divided into three tier (layers / categories): Presentation, Application & Data. The Presentation tier contains the classes responsible for interfacing to the user. The Data tier is concerned with the persisting object i.e. saving the contents of an object to the database where it can be later retrieved. The Application tier is where the applications workflows / business rules are implemented.

See the diagram Register an Account, Compose and Send Email, Change Password, Create New Folder Category, Add a New Contact Class Diagram, JPEG Attach, Create Group for examples of how the three-tier pattern has been used to design those features.

Pattern: Builder Pattern

Builder pattern builds a complex object using simple objects and using a step by step approach. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

A Builder class builds the final object step by step. This builder is independent of other objects.

See Register an Account, Compose and Send Email, Add a New Contact, Create Event, Sequence Diagram for examples for implementation of this pattern.

Pattern: Data Transfer Object (DTO)

The service proxy pattern describes a class that acts as an interface to an external service. These are usually a remote service that is accessed over a network connection. The proxy provides an interface (methods) that define what operations the system can perform against the remote service.

The design utilizes a proxy to provide an interface to the RegisterController, MailCompostion Controller, ContactManagementController, FolderManagementController, EditProfileInfoController, LoginCredDTO, EventDTO and data processor for each class.

Pattern: Model-View-Controller Pattern

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

- Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
- View - View represents the visualization of the data that model contains.
- Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

Examples of Screens (Views), Controllers, and Entity classes can be found throughout this design.

Pattern: Data Access Object (DAO)

Data Access Object Pattern or DAO pattern is used to separate low level data accessing API or operations from high level business services. Following are the participants in Data Access Object Pattern.

- Data Access Object Interface - This interface defines the standard operations to be performed on a model object(s).

- Data Access Object concrete class - This class implements above interface. This class is responsible to get data from a data source which can be database / xml or any other storage mechanism.
- Model Object or Value Object - This object is simple POJO containing get/set methods to store data retrieved using DAO class.

Examples of DAO classes can be found in the design sequence diagrams in this design.