TimeDrop
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
<10/20/2022>	<1.0>	First Draft	Sai Mittapalli, Mohamed Aql, Muhammad Momin Rahman, Abdalla Eltom

Table of Contents

Figures

Software Architecture Document

1. Introduction

1.1 Purpose

This part of the document shows an overview of the software system using several different views and diagrams. It explains how many of the decisions and elements were decided regarding the software system.

1.2 Scope

This Software Architecture part of the document provides a view of the Calendar application. The Calendar application allows users to divide their lives into different organized calendars so the specific tasks can be kept in track easily. Along with that, the application provides a monthly calendar along with a specific to-do list.

1.3 Definitions, Acronyms, and Abbreviations

Refer to Glossary Document (upedu_gloss.dot)

1.4 References

1. UPEDU Template Example
2. TimeDrop – Use Case Specification
3. TimeDrop – Supplementary Specification

1.5 Overview

This Software Architecture Document consists of the different views of the architecture design along with the basic design ideas and features.

2. Architectural Representation

This part of the document provides the different views of the application through the lens of a use-case diagram (previously stated in document), sequence diagram, class diagram using the Unified Modeling Language (UML).

3. Architectural Goals and Constraints

The system is going to be developed in a web application tool so it will consist of three main components: a web browser, web application server, and a database server. All three of these components must be able to function in a modern web browser and the server and database need to be located on the same host.

4. Use-Case View

The Use-Case view is important to consider when creating a design for any particular applications or software because it describes all the scenarios a user can go through when using the software. It also helps realize the important design features and scenarios that need to be covered.

Refer to Use-Case Specification document for more information.

4.1 Use-Case Realizations

Refer to Use-Case Realization Document.

5. Logical View

The logical views describe the most significant features of an architecture design model, including the decomposition of the system into a package view and a class view.

5.1 Overview

This section describes the decomposition of the design of the system into the package and class views.

5.2 Architecturally Significant Design Packages

5.2.1 Package Diagram

Figure 1: Design Model Package

5.2.2 Package Descriptions

Server

Description	All user credential information is stores and maintained in this package
Corresponding Classes	Server/Database
Relations	Dependency: User
Sub-Packages	User

User

Description	All user normal information and functions are stored here
Corresponding Classes	User, Home Page, Calendar, Combined To-Do List, Monthly Calendar, Specific To-Do List
Relations	Sub-package of Server package
Sub-Packages	None

5.2.3 Class Diagram

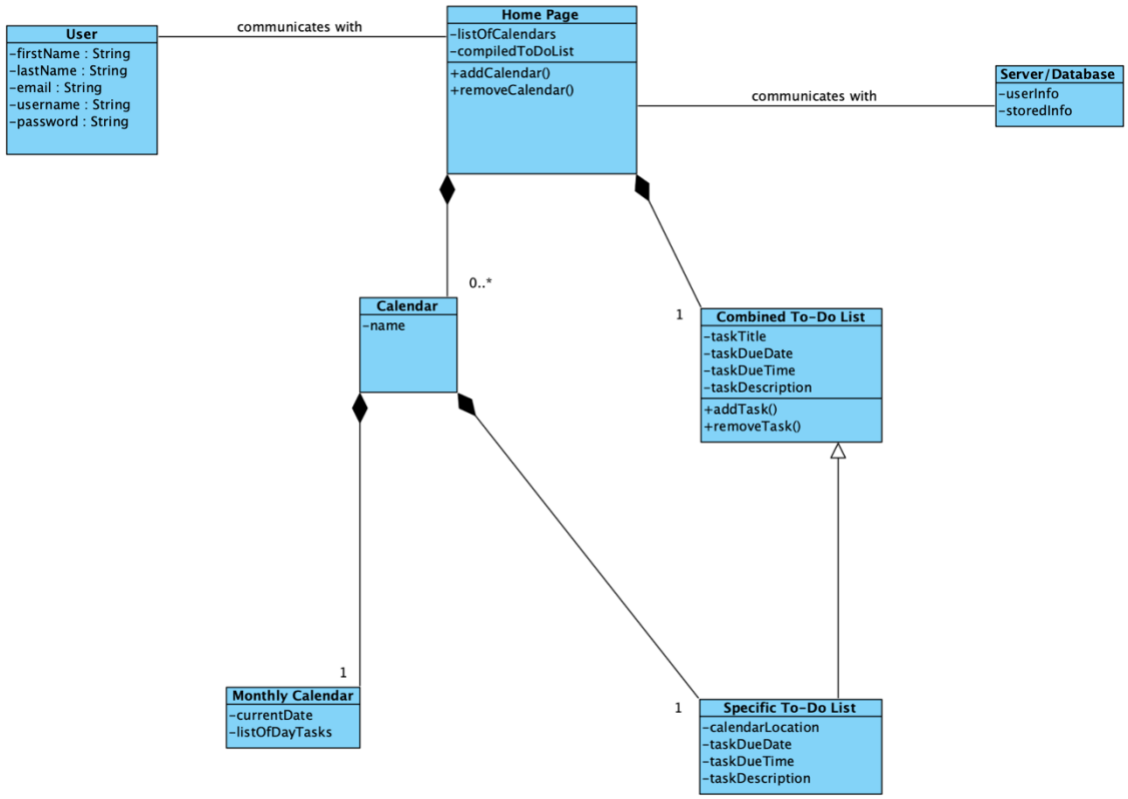

Figure 2: Class Diagram

5.2.4 Class Descriptions

Property	**Description**
Name	User
Description	Represents each unique user object
Responsibilities	Maintains all user information
Relations	**Association**: Communicates with the Homepage
Methods	None
Attributes	firstName: private string variable for the user's first name lastName: private string variable for the user's last name email: private string variable for the user's email phoneNumber: private float variable for the user's phone number username: private string variable for the user's username password: private string variable for the user's password

Property	**Description**
Name	Homepage
Description	Represents the home page of the application
Responsibilities	Contains all the calendar objects and combined to-do list
Relations	**Composition**: contains calendars and combined to-do list **Association**: communicated with server/database to retrieve information
Methods	addCalendar(): adds a new calendar to the home page removeCalendar(): removes a calendar from the home page
Attributes	listOfCalendars: the list of calendars on the home page compiledToDoList: the list of all tasks on the complete to-do list

Property	**Description**
Name	Server/Database
Description	Represents where the information for each calendar and to-do list is stored
Responsibilities	Stores all the calendar and to-do list information for the specific user
Relations	**Association**: communicates with home page
Methods	None
Attributes	userInfo: contains array of all user information storedInfo: contains array of all stored calendar information

Property	**Description**
Name	Calendar
Description	Calendar objects for each type of calendar
Responsibilities	Stores the aspects needed to present how the calendar tab looks like
Relations	**Composition**: is part of the home page **Composition**: contains monthly calendar and specific to-do list
Methods	None
Attributes	name: name of calendar

Property	**Description**
Name	Combined To-Do List
Description	Class with list of all the to-do list tasks from all the calendars' specific to-do lists
Responsibilities	Contains all the tasks across the calendars
Relations	**Composition**: is part of the home page
Methods	addTask(): adds task to the list removeTask(): removes task from list
Attributes	taskTitle: name of the task taskDueDate: due date of task taskDueTime: due time of task taskDescription: description of task

Property	**Description**
Name	Monthly Calendar
Description	Shows the calendar for the current month
Responsibilities	Displays the important tasks for the current day
Relations	**Composition**: part of the calendar
Methods	None
Attributes	currentDate: current date of the month listOfDayTasks: list of all tasks for specific day

Property	Description
Name	Specific To-Do List
Description	Shows to-do list for the current calendar
Responsibilities	Displays all the tasks for the month
Relations	**Composition**: part of the calendar **Generalization**: more specific to-do list of the combined to-do list
Methods	None
Attributes	taskTitle: name of the task taskDueDate: due date of task taskDueTime: due time of task taskDescription: description of task

6. Interface Description

The interface will be a website with buttons the user can use to enter different types of calendars and view the combined to-do list. When the user opens the application, they will be prompted to either login or create an account. After selecting a calendar, the user is shown a monthly calendar and specific to-do list for that calendar. The user can add tasks to any of these elements.

7. Size and Performance

The architecture for this system will support all the sizing and performance requirements because the user will access the system through a web browser with minimal memory requirements and a client-server architecture will be used.

8. Quality

The architecture will be able to support the quality features required to allow the system run smoothly like stated in the Software Requirements Specification Document and Supplementary Specifications Document.