**Term Project: *TaskMaster***

Design Document

**Table of Contents**

**1** **Introduction**

*1.1* *Purpose and Scope*

*1.2* *Target Audience*

*1.3* *Terms and Definitions*

**2** **Design Considerations**

*2.1* *Constraints and Dependencies*

*2.2* *Methodology*

**3** **System Overview**

**4** **System Architecture**

*4.1* *GUI*

4.1.1 Landing Page

4.1.2 Login Page

4.1.3 Signup Page

4.1.4 List Page

*4.2* *Task*

*4.3* *List*

*4.4* *ToDo*

*4.5* *In Progress*

*4.6* *Done*

*4.7* *Account*

*4.8* *Database*

**5** **Detailed System Design**

*5.1* *GUI*

5.1.1 Landing Page

5.1.2 Login Page

5.1.3 Signup Page

5.1.4 List Page

*5.2* *Task*

*5.3* *List*

*5.4* *ToDo*

*5.5* *In Progress*

*5.6* *Done*

*5.7* *Account*

*5.8* *Database*

**1. Introduction**.

In this document, I will outline the design of the TaskMaster program, a simple task management program. This document includes a general overview of the program, and more detailed information about the major components and how those components will interact.

**1.1 Purpose and Scope**

In this document I will cover the big picture design, as well as identify the various subcomponents and how they will interact with each other. This document will describe the overview of the system as well as important design considerations.

**1.2 Target Audience**

The target audience for this document includes the stakeholders for the TaskMaster project. That includes Professor Fei Xie and TA Bin Lin, who provided the functional requirements and fill the role of the customer. They will be using and testing the program to ensure that it meets the functional requirements. As the software engineer for this project, I am also a stakeholder. I will be using this follow the design I have created for the TaskMaster project to ensure it is functional.

**1.3 Terms and Definitions**

GUI - Graphical User Interface

**2. Design Considerations**

This section will outline all of the constraints and dependencies for the TaskMaster program design.

**2.1 Constraints and Dependencies**

TaskMaster has several functional requirements that I am required to adhere to when designing and implementing the program. The key requirements involve including an account management module that allows users to create unique and private loggins. This project must also be developed with the Java programming language.

**2.2 Methodology**

For the TaskMaster project, I will be implementing an Object Oriented Design. This design is particularly useful because I am designing a relatively simple program in Java, which is an object oriented programming language. This method is also very straightforward and there are many tools and method that make using Object Oriented Design a very effective method, such as class modeling and CRC cards. I used these techniques when developing the design for the TaskMaster program. I am using the class model to develop the classes and methods that I will implement in the TaskMaster program. Also by creating use cases along with the class model, I am working to make sure that I correctly include all needed functionality in the program.

**3. System Overview**

The TaskMaster program will allow a user to create a unique account with login information. Then each user will be able to add, edit, and delete tasks. Users will have three lists in which they can sort their tasks: To-Do, In Progress, and Done. Users will be able to provide descriptions for each task as well. Using object oriented design, I have identified the key nouns as: **Task** , **Account**, **Database**, **To-Do, In Progress, Done**, **List**, and **GUI**. These will serve as the main classes for the TaskMaster program. Each of these classes include various data members and methods, which will be described in detail in the detailed system design. As the program progresses the **GUI** class will likely be broken up into smaller classes.

CS300 Class Model.png

**4. System Architecture**

This program will be broken into the following classes: **Task** , **Account**, **Database**, **To-Do, In Progress, Done**, **List**, and **GUI**. The **Task**, **Account**, **To-Do**, **In Progress**, **Done**,and **List** are the objects in our object oriented design. **Task**, **To-Do**, **In Progress**, **Done**, and **List** are in a hierarchical structure. **List** “has a” **Task**, and **To-Do**, **In Progress**, and **Done** have a “is a” relationship with **List**. Our **GUI** and **Database** classes are subsystems of our program.

**4. 1 GUI**

Our **GUI** class will provide a simple user interface for the user so that they can create an account, login, and view/create/edit/delete their tasks. This will be implemented using Java Swing. The sub-components of the **GUI** include: **Landing Page**, **Login Page**, **Signup Page**, and the **List Page.**

cs300workfaa.png

**4.1.1 Landing Page**

The **Landing Page** for the TaskMaster is simply the first thing a user sees when they launch the application. It will include the TaskMaster logo, a “login” button for users who already have an account, and a “sign up” button for users who need to make a new account. The “login” button will take users to **Login Page**, and the “sign up” button will take users to the **Signup** page.

**4.1.2 Login Page**

The **Login Page** for the TaskMaster will contain two text boxes. One for the user’s username, and one for their password. Once they have entered their credentials, they can select a “login” button. If their credentials are correct, they will be taken to the **List Page**. If they did not enter valid credentials, they will be prompted to try again, or create a new login.

**4.1.3 Signup Page**

The **Signup Page** for the TaskMaster will have three empty text boxes. One for a username, one for a password, and one for password confirmation. Once they have filled in these boxes, and the passwords in box two and three match they can select the “signup” button, which will save their new account and take them to the **List Page**.

**4.1.4 List Page**

The **List Page** will be where the user can view/create/edit/delete tasks. The page will be divided into three main sections, To-do, In Progress, and Done. This page will also contain buttons for creating tasks, moving tasks, editing tasks, deleting tasks, and logging out.

**4. 2 Task**

The focus of the TaskMaster program is the management of task. So we will have a **Task** object. The **Task** object will be in a hierarchical relationship with the **List** object. The **List** will “have a” task object. The **Task** object will contain a title, description, and category fields.

**4. 3 List**

The **List** object will be a collection of **Task** objects. These **Task** objects will have a category tag in them, and that is how they will be sorted into the different sub-lists, **To-do, In Progress,** and **Done**.

**4. 4 To-Do**

There are 3 list categories, the first of which is the **To Do** category. When a new task is created, it is put into the **To Do** list by default. The **To Do** list is an object derived from the **List** class.

**4. 5 In Progress**

The second list type is the **In Progress** list. When a user begins working on a task, they can move it into the **In Progress** list. The **In Progress** list is an object derived from the **List** class.

**4. 6 Done**

The last list type is the **Done** list. When a user completes a task, they can move it into the **Done** list. The **Done** list is an object derived from the **List** class.

**4. 7 Account**

In order to use TaskMaster, the user has to create a new **Account**. This will include a user name and a password. This will be stored in some **Database**. The **Account** object will need to communicate with the **Database** to retrieve the **Account** information, or to be saved to the **Database**.

**4. 8 Database**

The each individual log in information will be stored in a **Database**. We will use the **Database** class to save and retrieve user credentials and task information.

**Detailed System Design**

Each of the components, **Task** , **Account**, **Database**, **To-Do, In Progress, Done**, **List**, and **GUI**, can be broken into further detail and will be described in this section.

**5. 1 GUI**

For the GUI I will be using Swing, which is a GUI widget for Java. This will allow for a more user friendly and accessible application for the users. I have developed a wireframe for each component of the GUI, which will be shown in each subsystem. I will convert the wireframes into a functional GUI.

**5.1.1 Landing Page**

CS300lANFINF.png

I will use button linking to connect the landing page to the login page and to the signup page. Swing has several libraries for creating pages that will format similar to my wireframe. For this page I will be implementing the JFrame, JButton, and JLabel classes which will be used in the creation of the window and buttons, as well as any labeling. This will also be used to determine the behavior of the buttons when pushed.

**4.1.2 Login Page**

Cs300Loginsa.png

The login page will also use buttons to link to other frames, in specific the list page. Users will be redirected to the list page after entering the correct credentials in the text boxes. Swing also has the ability to insert functional text boxes where the user can enter their username and password. I will be implementing the JFrame, JButton, and JLabel classes which will be used in the creation of the window and buttons, as well as any labeling. This will also be used to determine the behavior of the buttons when pushed.

**4.1.3 Signup Page**

Cs300Loginsa.png

The **Signup Page** for the TaskMaster will have three empty text boxes similar to the login page. This is also going to be using the Swing functionality for textboxes. This page will use linking to link the list page to the signup page through the “log in” button. I will be implementing the JFrame, JButton, and JLabel classes which will be used in the creation of the window and buttons, as well as any labeling. This will also be used to determine the behavior of the buttons when pushed.

**4.1.4 List Page**

Cs300Loginsa.png

The list page will be the most complex user interface in the program. I will be using frames, sub frames, labeling, and buttons to achieve the design laid out in my wireframe. I will be implementing the JFrame, JButton, and JLabel classes which will be used in the creation of the window and buttons, as well as any labeling. This will also be used to determine the behavior of the buttons when pushed.

**5. 2 Task**

The **Task** object will need to have fields for the title, category, and description. Some of the key methods will be a constructor, a method for editing a task description, a method for editing a task name, and a method for changing the category of the task.

//Task pseudo code

public class Task{

protected String title;

protected String description;

protected String category;

protected Task next;

public Task()

{

this.next = null;

}

public Task(String title, String description, String category)

{

this.title = title;

this.description = description;

this.category = category;

this.next = null;

}

public editTaskTitle(String title)

{

this.title = title;

}

public editTaskDesc(String description)

{

this.description = description;

}

public editTaskCat(String category)

{

this.category = category;

}

public Task get\_next()

{

return next;

}

}

**5. 3 List**

The **List** object will be a collection of **Task** objects. These **Task** objects will have a category tag in them, and that is how they will be sorted into the different sub-lists, **To-do, In Progress,** and **Done**.The list will likely be a linked list.

public class LLL {

private Task head;

public void display\_LLL()

{

if(head == null) {

System.*out*.print("null");

}

Task temp = head;

while(temp!= null)

{

temp.display();

temp = temp.get\_next();

}

}

public void insert(String title, String description, String category)

{

head = insert(title, description, category, head);

}

private Task insert(String title, String description, String category, Task head)

{

if(head == null) {

head = new Task( title, description, category);

return head;

}

head.set\_next(insert(title, description, category, head.get\_next()));

return head;

}

public void remove\_fl()

{

Node temp = head;

head = head.get\_next();

temp = null;

remove\_fl(head);

}

public void remove\_one(String key)

{

Task temp = head;

if(head == null) {

return;

}

//if the match is at the head, set head to the next node and delete old head

if(head.compare(key))

{

temp = head;

head = temp.get\_next();

temp = null;

return;

}

//otherwise traverse the list looking for a match

while(temp.get\_next() != null) {

//if our next node is a match, simply point around it to its next node

if ((temp.get\_next().get\_next() != null) && temp.get\_next().compare(key)) {

temp.set\_next(temp.get\_next().get\_next());

return;

}

//if the last node is a match set it to null

if ((temp.get\_next() == null) && temp.get\_next().compare(key)) {

temp = null;

return;

}

//traverse

temp = temp.get\_next();

}

}

public void remove\_all()

{

head = null;

}

}

**5. 4 To-Do**

The To-Do list will be a derived class from the list class. This will simply allow for the functionality set up by the list class, but in a separate list. This will reduce the amount of code needed and will keep the tasks separated into their respective categories. In Java we will use the following for defining a derived class:

public class ToDo extends LLL{

….

}

**5. 5 In Progress**

The In Progress list will be a derived class from the list class. This will simply allow for the functionality set up by the list class, but in a separate list. This will reduce the amount of code needed and will keep the tasks separated into their respective categories. In Java we will use the following for defining a derived class:

public class InProgress extends LLL{

….

}.

**5. 6 Done**

The Done list will be a derived class from the list class. This will simply allow for the functionality set up by the list class, but in a separate list. This will reduce the amount of code needed and will keep the tasks separated into their respective categories. In Java we will use the following for defining a derived class:

public class Done extends LLL{

….

}

**5. 7 Account**

Each user will have a unique account. This will consist of a user name and a password. These will be stored in the database. If there is time, I will also associate an email address with each account. The account object is simply two strings, a username and a password.

**5. 8 Database**

The user information will be stored in a database. For this project I will use MySQL, an open source relational database.