Software Design Document for Gamified Calendar

Prepared by: Levent Batakci, David Frost, Jo Phan, Dillon Yu, and Smyan Thota

CSDS-393 at Case Western Reserve University

October 4, 2022

Table of Contents

Inspection Log

Name	Date	Description	Section(s)
Dillon Yu	10/2	Created document template and added applicable documents	All Sections
Smyan Thota	10/2	Added introduction, Inter-Agent Protocol, Appendix	Introduction,Inter-Agent Protocol,Appendix
David Frost	10/4	Added some interfaces and classes, expanded upon methods for existing classes	ClassInterfaces, Principal Classes
Levent Batakci	10/4	Added some classes	Principal Classes, Class Interfaces
Dillon Yu	10/4	Expanded “Class Interfaces” section	Class Interfaces
Jo Phan	10/4	Adding designs and layout for landing page, to-do list, calendar, profile page	Mockup UI
Smyan Thota, Dillon Yu,David Frost	10/4	Check documents for formatting and content mistakes, along with restructuring and reviewing material.	All Sections

1. Introduction

Gamified Calendar aims to make productivity fun! When we usually end up using a calendar, it is to save an event that occurs at a later point of time. But what if we could gamify this process? The Gamified Calendar enables us to set events, goals and tasks onto either the to-do list or the calendar, and upon completion of said tasks, you gain experience and points, adding them to your streak of continuously productive days, and motivating you to finish more of your tasks and events in order to not break the streak. This helps us out immensely, since we are hardwired to continue to build our streaks, instead of breaking them. As a result, it helps us to stay active, build difficult but necessary habits and enables us to be more productive everyday.

This document describes the design of the Gamified Calendar. The design fulfills the requirements specified by the SRS. First, we define the principal classes. We then describe the inter-agent protocols (i.e. the interaction between the classes). Finally, we list all class interfaces of the Gamified Calendar, with their attributes and methods.

2. Applicable Documents

Levent Batakci, David Frost, Jo Phan, Dillon Yu, and Smyan Thota, *Software Requirements Specifications for Gamified Calendar*, 19 September 2022.

3. Principal Classes

3.1 Time

Time is an enum used to specify which type of view to show on a calendar. *Time* has a restricted set of values *DAY*, *WEEK*, and *MONTH*.

3.2 Account

Account is a class that is used to access information about a particular user's account and to change account information such as password or username. An *Account* is created by the system after reading relevant information from the database and then wrapping it in an object.

3.3 Task

Task is a class that is used to store information about any particular task. Some data included in a *Task* include the task's start date, deadline, description, and status. The deadline is optional, since not every task has one. For instance, someone might set a goal of reading a book without wanting to self-impose a deadline.

3.4 DataAccess

DataAccess is a class that handles the database layer of the web application. Given inputs, it can either query the database, insert new data into the database, or modify existing data. New tasks, events, and accounts can be inserted. Existing accounts, events, and tasks can be modified in ways such as account names changing or task descriptions changing. The *DataAccess* class abstracts access to the database and is packaged in an object to fit into an object-oriented programming (OOP) paradigm.

3.5 Calendar

Calendar is a class that stores data for the calendar user interface. Data included in a *Calendar* are the *Account*, *Time*, and list of *tasks* associated with a calendar view. Method calls to this object depend on what type of view the user chooses on the calendar as well as the settings. Anytime the user changes their view or settings, method calls will be made to *Calendar* to retrieve the relevant data to display. This powers the frontend of the system.

3.6 Event

Event is a class that is used to store and handle information pertaining to a calendar event. One of the main features of Gamified Calendar is to serve as an event-tracker, and this class is how that will be accomplished. Users can create *Events* at will and associate

important informational points with them. At the bare minimum, each event needs to have a date & time and be identified by a name.

Outside of this requirement, users will be able to robustly customize their events. For instance, instances of the *Event* class may be associated with a category (e.g. school), and the location of the *Event* may be stored within it as well. Users will be able to set *Events* to repeat (weekly, daily, or on a custom period). The *Event* class is packaged to fit into an object-oriented programming (OOP) paradigm.

3.7 ToDoList

ToDoList is a class that is used to organize and manage groups of tasks. One of the main features of Gamified Calendar is to serve as a virtual to-do list, and this class is how that will be accomplished. When a user creates a task, it will be added to a to-do list. While there will only be one to-do list by default, users may create several in order to better organize their tasks.

Separate to-do lists may be rendered separately, or all tasks may be rendered in a master to-do list. Users may, at any point, choose to delete a to-do list. When doing so, the user will be asked if they wish to migrate the associated tasks to another list. If not, then all of the associated tasks will be deleted.

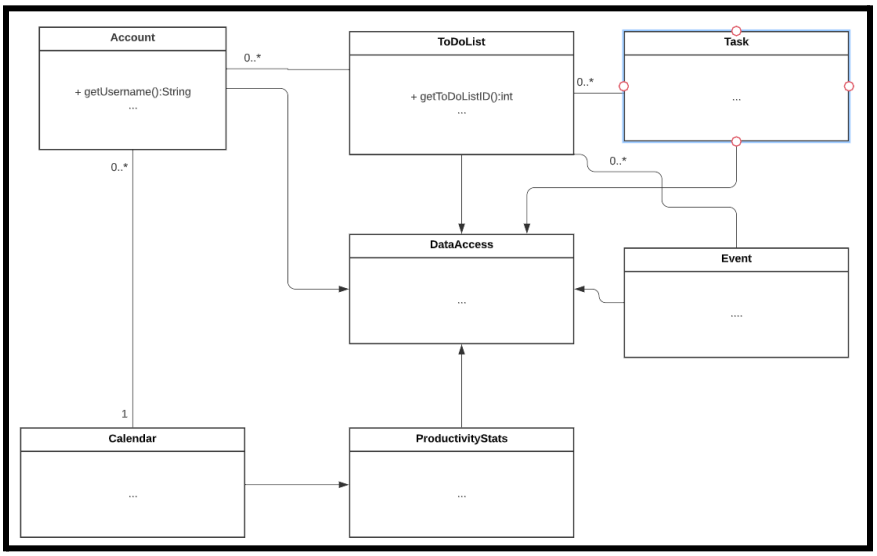
3.8 ProductivityStats

ProductivityStats is a class that is used to organize and manage a user's history of completed tasks and events. Moreover, this class will contain procedures for computing encouraging stats regarding user productivity.

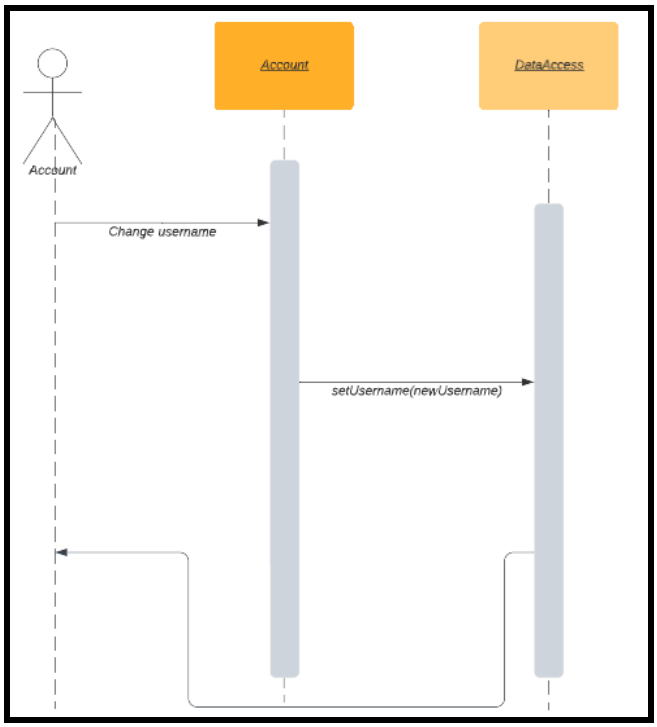
Several examples include: number of tasks completed, longest streak, current streak, and productivity trends over time. The key gimmick of the Gamified Calendar is that it is gamified to keep users engaged and encouraged, and this class is how that will be accomplished.

The productivity statistics can be viewed at any time, and users may further indicate which stats they would like to see on their dashboard (main page of the calendar). Productivity stats computation will also be smart enough to compute under parameters (for example, in a time range).

4.Inter-Agent Protocols



Class design figure shows the various classes and their dependencies and connections with other respective classes



UML sequence diagram figure shows the flow from the user to the account and database

4.1 User Registration and Login

When registering for a new account, the user will be required to enter an email, username, password. An *Account* will be created based on the input information provided. In addition, a *Calendar* associated with the new account will be created using the method setAccount().
After your account is created, you will be prompted to sign in using your account. If you have created your account in the past, you can sign into your account, using your existing username and password. Upon entry of successful user name and password, that is checked in *Account* you will be able to access your *Calendar,* which is linked to your *Account*.

4.2 Rendering Events on the Calendar

When viewing a section of the calendar, the corresponding events need to be displayed. In order to do this, the calendar will need to determine the events lying on each date within the range. The calendar tied to an account will store all of the user's events, so the events can be figured out by simply going through the list.

4.3 Productivity Stats Computation

In order to compute the productivity stats, all kinds of data regarding events and tasks needs to be accessed. For example, the number of completed tasks could need to be counted. So the ProductivityStats class will utilize a lot of the getter methods associated with the Event and Task classes.'

4.4 Rendering Events on the Task

While viewing the To-Do List Section, the corresponding events need to be displayed. In order to do this, the list will select the last inputted list element out, and arrange them according to the order they were inserted in. The list tied to an account will store all of the user's events, so they will be displayed accurately.

4.5 Adding and Removing Tasks

Tasks will be added to the *ToDoList* using the addTask() method. The database will also be updated using the addTask() method in DataAccess. When tasks need to be removed, they will be removed from the *ToDoList* using removeTask() and from the database using deleteTask().

5. Class Interfaces

This section describes the object class interfaces of the Gamified Calendar. The interfaces are incomplete and subject to revision. Constructors are currently omitted.

5.1 Class Account

An instance of class *Account* represents a user's account for the software. Attributes in *Account* include *username, email, password*, *accountID, bestStreak, currentStreak,* and *totalPoints*.

5.1.1 Public Method getUsername()

public String getUsername()

Returns the *username* of the *Account*.

5.1.2 Public Method setUsername(String username)

public void setUsername(String username)

Sets the *username* of the *Account* to the input string.

5.1.3 Public Method getPassword()

public String getPassword()

Returns the *password* of the *Account*.

5.1.4 Public Method setPassword(String password)

public String setPassword(String password)

Sets the *password* of the *Account* to the input string.

5.1.5 Public Method getAccountID()

public int getAccountID()

Returns the *accountID* associated with the *Account*.

5.1.6 Public Method getAccountEmail()

public String getAccountID()

Returns the *accountEmail* associated with the *Account*.

5.1.7 Public Method setAccountEmail(String accountEmail)

public void setAccountEmail(String accountEmail)

Sets the *accountEmail* associated with the *Account* to the input string.

5.2.14 Public Method getBestStreak()

public int getBestStreak()

Returns the *bestStreak* of the *Account* (in days).

5.2.15 Public Method setBestStreak(int bestStreak)

public void setBestStreak(int bestStreak)

Sets the *bestStreak* of an *Account* to the input integer.

5.2.16 Public Method getCurrentStreak()

public int getCurrentStreak()

Returns the *currentStreak* of the *Account* (number of days).

5.2.17 Public Method setCurrentStreak(int currentStreak)

public void setCurrentStreak(int currentStreak)

Sets the *currentStreak* of an *Account* to the input integer.

5.2.18 Public Method getTotalPoints()

public int getTotalPoints()

Returns the *totalPoints* of the *Account*.

5.2.19 Public Method setTotalPoints(int totalPoints)

public void setTotalPoints(int totalPoints)

Sets the *totalPoints* of an *Account* to the input integer.

5.2 Class Task

An instance of class *Task* represents a task for the software. Attributes in *Task* include *taskID, taskName, taskDescription, taskPoints, taskStartDate, taskDeadline,* and *taskStatus*.

5.2.1 Public Method getTaskID()

public int getTaskID()

Returns the *taskID* associated with a *Task*.

5.2.2 Public Method getTaskName()

public String getTaskName()

Returns the *taskName* associated with a *Task*.

5.2.3 Public Method setTaskName(String taskName)

public void setTaskName(String taskName)

Sets the *taskName* of a *Task* to the input string.

5.2.4 Public Method getTaskDescription()

public String getTaskDescription()

Returns the *taskDescription* associated with a *Task*.

5.2.5 Public Method setTaskDescription(String taskDescription)

public void setTaskDescription(String taskDescription)

Sets the *taskDescription* of a *Task* to the input string.

5.2.6 Public Method getTaskPoints()

public int getTaskPoints()

Gets the *taskPoints* of a *Task* associated with a *Task*.

5.2.7 Public Method setTaskPoints(int taskPoints)

public void setTaskPoints(int taskPoints)

Sets the *taskPoints* of a *Task* to the input integer.

5.2.8 Public Method getTaskStartDate()

public Date getTaskStartDate()

Gets the *taskStartDate* of a *Task* associated with a *Task*.

5.2.9 Public Method setTaskStartDate(Date taskStartDate)

public void setTaskStartDate(Date taskStartDate)

Sets the *taskStartDate* of a *Task* to the input *Date*.

5.2.10 Public Method getTaskDeadline(Date taskDeadline)

public Date getTaskDeadline()

Returns the *taskDeadline* of a *Task*.

5.2.11 Public Method setTaskDeadline(Date taskDeadline)

public void setTaskDeadline(Date taskDeadline)

Sets the *taskDeadline* of a *Task* to the input *Date*.

5.2.12 Public Method getTaskStatus()

public boolean getTaskStatus()

Returns true if *Task* is complete, false otherwise.

5.2.13 Public Method setTaskStatus(boolean taskStatus)

public void setTaskStatus(boolean taskStatus)

Sets the *taskStatus* of a *Task* to the input boolean.

5.3 Class DataAccess

An instance of class *DataAccess* represents an object that abstracts access to the database. Method calls to this object query the database based on inputs and returns the result in an object form. This is the database layer of the system.

5.3.1 Public Method getAccount(int accountID)

public Account getAccount(int accoundID)

Returns the *Account* associated with an *AccountID*.

5.3.2 Public Method getTask(int taskID)

public Task getTask(int taskID)

Returns the *Task* associated with a *taskID*.

5.3.3 Public Method addAccount(Account account)

public void addAccount(Account account)

Takes *Account account* and creates new relevant records in the database corresponding to the new account data.

5.3.4 Public Method addTask(Task task)

public void addTask(Task task)

Takes *Task* task and creates new relevant records in the database corresponding to the new task data.

5.3.5 Public Method addEvent(Event event)

public void addEvent(Event event)

Takes *Event* event and creates new relevant records in the database corresponding to the new event data.

5.3.6 Public Method deleteAccount(int accountID)

public void deleteAccount(int accountID)

Takes integer *accountID* and deletes relevant records in the database corresponding to that *Account* data.

5.3.7 Public Method deleteTask(int taskID)

public void deleteTask(int taskID)

Takes integer *taskID* and deletes relevant records in the database corresponding to that *Task* data.

5.3.8 Public Method deleteEvent(int eventID)

public void deleteEvent(int eventID)

Takes integer *eventID* and deletes relevant records in the database corresponding to that *Event* data.

5.4 Class Calendar

An instance of class *Calendar* represents an object that gives data necessary for the user interface of the Gamified Calendar. Attributes in *Calendar* include *account*, *time,* and *tasks*.

5.4.1 Public Method setAccount(Account account)

public void setAccount(Account account)

Takes *Account account* and sets it to be the user associated with this Calendar view.

5.4.2 Public Method getAccount()

public int getAccount()

Returns the *Account* associated with this calendar view.

5.4.3 Public Method getTime()

public Time getTime()

Returns the *Time* associated with this calendar view.

5.4.4 Public Method setTime(Time time)

public void setTime(Time time)

Sets the *Time* associated with this calendar view.

5.4.5 Public Method getTasks()

public ToDoList getTasks()

Gets the *ToDoList* associated with this calendar view.

5.5 Class Event

An instance of class *Event* represents an event for the software. Attributes in *Event* include *eventID, eventName, eventDescription, eventStartTime, eventEndTime*.

5.5.1 Public Method getEventID()

public int getEventID()

Returns the *eventID* associated with an *Event*.

5.5.2 Public Method getEventName()

public String getEventName()

Returns the *eventName* associated with an *Event*.

5.5.3 Public Method setEventName(String eventName)

public void setEventName(String eventName)

Sets the *eventName* of an *Event* to the input string.

5.5.4 Public Method getEventDescription()

public String getEventDescription()

Returns the *eventDescription* associated with an *Event*.

5.5.5 Public Method setEventDescription(String eventDescription)

public void setEventDescription(String eventDescription)

Sets the *eventDescription* of an *Event* to the input string.

5.5.8 Public Method getEventStartTime()

public Date getEventStartTime()

Gets the *eventStartTime* of a *Event.*

5.5.9 Public Method setEventStartTime(Date eventStartTime)

public void setEventStartTime(Date eventStartTime)

Sets the *eventStartTime* of a *Event* to the input *Date*.

5.5.10 Public Method getEventEndTime()

public Date getEventEndTime()

Gets the *eventEndTime* of a *Event.*

5.5.11 Public Method setEventEndTime(Date eventEndTime)

public void setEventEndTime(Date eventEndTime)

Sets the *eventEndTime* of a *Event* to the input *Date*.

5.6 Class ToDoList

An instance of class *ToDoList* represents a list of tasks that need to be done. Attributes in *ToDoList* include *toDoListID, tasks, and name.*

5.6.1 Public Method getToDoListID()

public int getToDoListID()

Returns the *toDoListID* associated with a *ToDoList*.

5.6.2 Public Method addTask()

public void addTask(Task taskToAdd)

Adds the given *Task* to the group of tasks associated with the *ToDoList*.

5.6.3 Public Method removeTask(Task taskToRemove)

public void removeTask(Task taskToRemove)

Removes the given *Task* from the group of tasks associated with the *ToDoList*. Returns true if the removal was successful.

5.6.4 Public Method removeTask()

public bool removeTask(int taskID)

Removes the *Task* with the given ID from the group of tasks associated with the *ToDoList*. Returns true if the removal was successful.

5.6.5 Public Method getTasks()

public List<Task> getTasks()

Returns a list of all the *Tasks* associated with the *ToDoList*.

5.6.6 Public Method setName()

public void setName(String newName)

Sets the name of the *ToDoList*.

5.6.7 Public Method getName()

public String getName()

Returns the name of the *ToDoList*.

5.6.8 Public Method getToDoListID()

public int getToDoListID()

Returns the ID of the *ToDoList*.

5.6.9 Public Method setToDoListID()

public void setToDoListID(int newID)

Sets the ID of the *ToDoList*.

5.7 Class ProductivityStats

An instance of class *ProductivityStats* has a history of completed events and tasks. Attributes in *ProductivityStats* include *completedTasks, completedEvents, and productivityStatsID.*

5.7.1 Public Method getProductivityStatsID()

public int getProducvitityStatsID()

Returns the *productivityStatsID* of the *ProductivityStats*.

5.6.2 Public Method setProductivityStatsID()

public void setProducvitityStatsID(int newID)

Sets the *productivityStatsID* of the *ProductivityStats*.

5.7.3 Public Method addCompletedTask()

public void addCompletedTask(Task completedTask)

Adds the given *Task* to the tracked list of completed tasks associated with the *ProductivityStats*.

5.6.4 Public Method getCompletedTasks()

public List<Task> getCompletedTasks()

Returns the list of completed tasks *completedTasks* of the *ProductivityStats*.

5.7.5 Public Method addCompletedEvent()

public void addCompletedEvent(Event completedEvent)

Adds the given *Event* to the tracked list of completed events associated with the *ProductivityStats*.

5.6.6 Public Method getCompletedEvents()

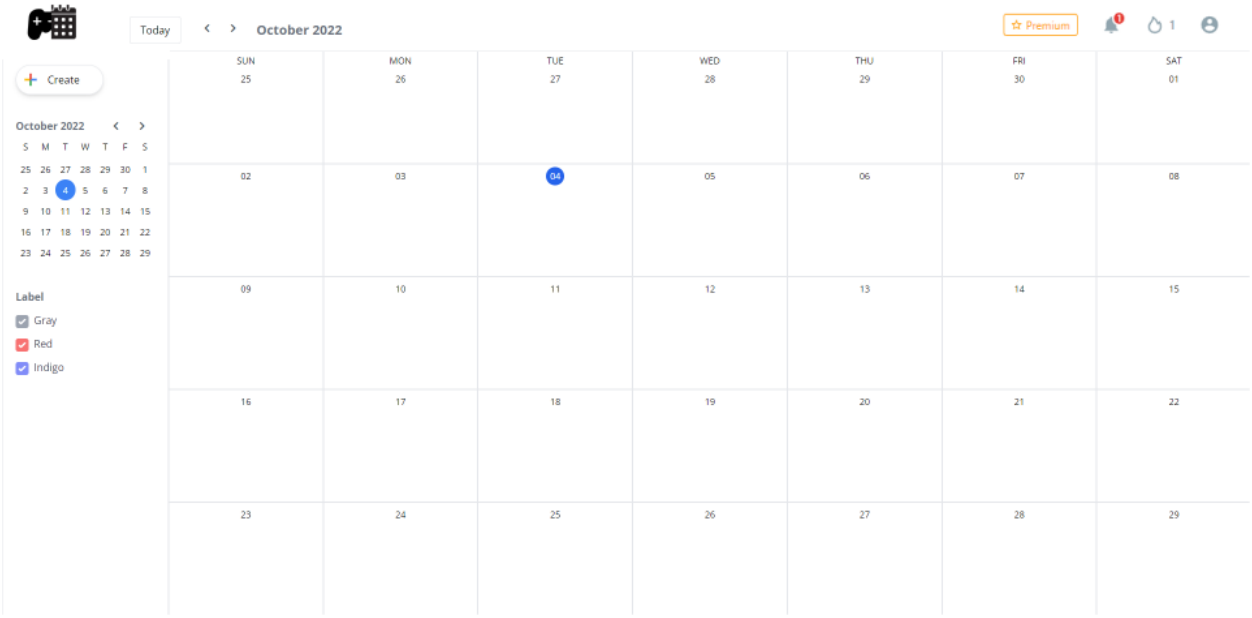
public List<Event> getCompletedEvents()

Returns the list of completed events *completedEvents* of the *ProductivityStats*.

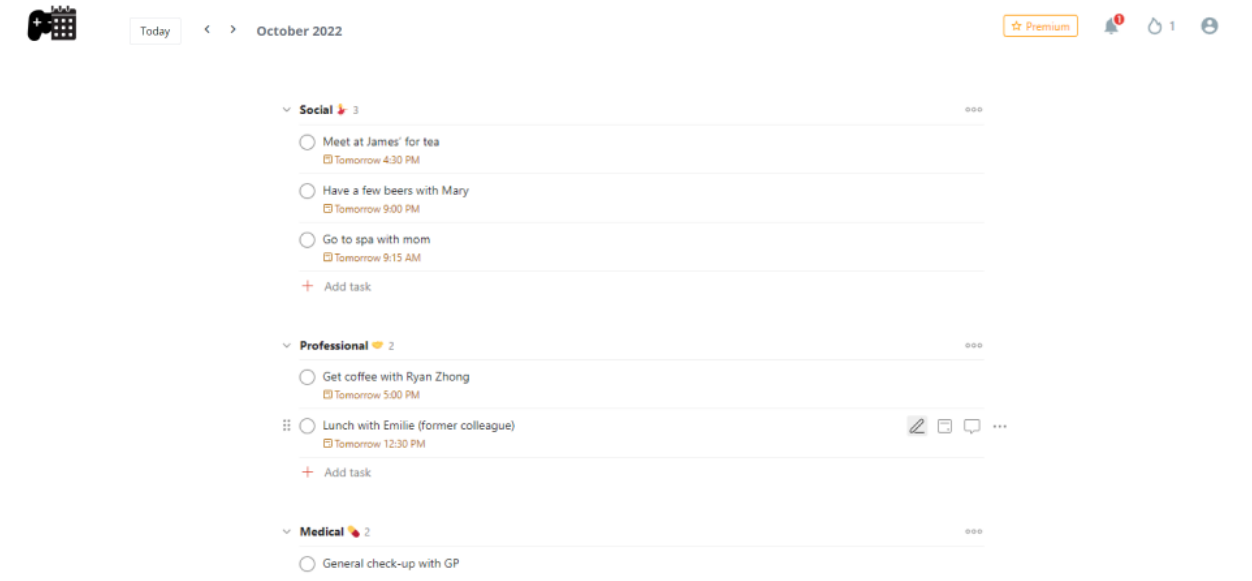
6. MockUIs

6.1 Landing Page

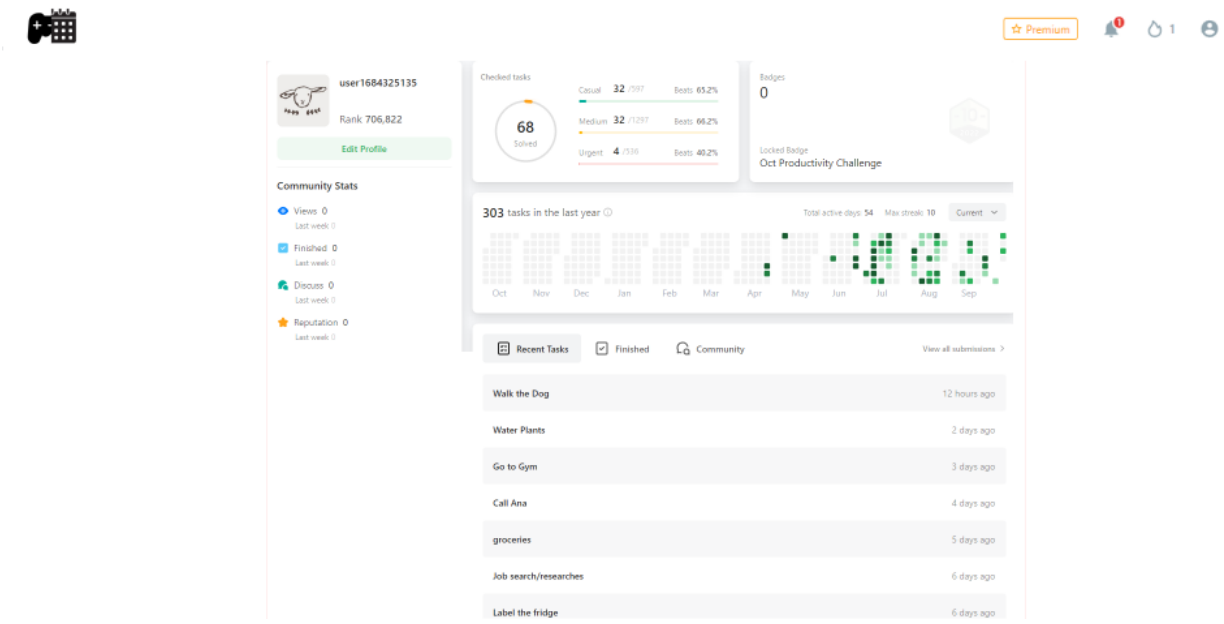
6.2 Calendar



6.3 To-do list



6.4 Profile page



7. Appendix

Appendix A: Glossary

Account	Account of a Gamified Calendar User
Gamified	Adding elements of games to an activity to make it more enjoyable and help assess progress at any point
SDD	Software Design Document
SRS	Software Requirements Specification
UI	User Interface
MockUI	Mockup of how the User Interface would look like

Events	A Task with a stipulated date and time, along with completion markers, that has been uploaded to the Gamified Calendar.
Streak	A record of continued usage of Gamified Calendar, that positively rewards using the Gamified Calendar to complete tasks, and negatively impacts users when they do fail to complete their task, by breaking the streak

Appendix B: References

Levent Batakci, David Frost, Jo Phan, Dillon Yu, and Smyan Thota, *Software Requirements Specifications for Gamified Calendar*, 19 September 2022.