A Project Report on

Task Manager

Computer Department

by

Tejas Deshmukh (16102002)

Akshay Sumbhe (16102062)

Falguni Tailor (16102031)

Under the Guidance of

Prof.Mayuri Jain

Department of Computer Engineering
A.P. Shah Institute of Technology
G.B. Road, Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2018-2019

Approval Sheet

This Project Report entitled *"**Task Manager**"* Submitted by ***"Tejas Deshmukh" (16102002),"Akshay Sumbhe"(16102062),"Falguni Tailor"(16102031)*** is approved for the partial fulfilment of the requirement for the award of the degree of ***Bachelor of Engineering*** in ***Computer Department*** from ***University of Mumbai***.

Prof.Mayuri Jain
Guide

Prof.Sachin Malave
Head Department of Computer

Place: A.P. Shah Institute of Technology, Thane

Date:

CERTIFICATE

This is to certify that the project entitled *"**Task Manager**"* submitted by ***"Tejas Deshmukh" (16102002),"Akshay Sumbhe" (16102062),"Falguni Tailor" (16102031)*** for the partial fulfilment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Computer Department.,*** to the University of Mumbai, is a bonafide work carried out during academic year 2017-2018.

Prof. Mayuri Jain
Guide

Prof. Sachin Malave
Head Department of Computer

Dr. Uttam D. Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane

Date:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

——————————————————

(Signature)

—————————————————

Tejas Deshmukh and 16102002
Akshay Sumbhe and 16102062
Falguni Tailor and 16102031

Date:

Abstract

Task Manager is a project aimed towards the users who have a busy schedule and still want to carry out a lot of extra curricular activities, but are in a dilemma as to how to arrange the activities in hand. Thus, they lose interest and end up doing no extra curricular activity as a result of the confusion in their heads.

This manager provides solution for the same. It will arrange the tasks for user, but these tasks could be only those which has no time bounds. Tasks such as college and class time, and others are user specific and are generally fixed. The timings of these activities cannot be changed. Thus, these are called as fixed tasks. User has to provide these timings any which ways. This is the working of every to-do-list or a typical task manager application available today. But they never help in arranging the tasks. This application will help arranging the activities that the user wishes to carry out in the time available after he allocates the fixed time. The tasks will be arranged considering their urgency and importance. Also, the most important factor that will be considered is the Stress/Relief level of the task which is too, user specific.

Contents

Chapter 1

Introduction

We know that people have a ton of work to complete and perform variety of activities in a day. Such as they have school, college, office work where the time is fixed and they have to spend the fixed amount of time there. After that they also require to spend some time to perform their activities such as their hobby which includes reading, painting, sketching, listening to music and various other activities to get fresh. It becomes difficult for them to decide which task to perform according to remaining time. It increasing their stress level after such workload and then to calculate at what time which tasks to perform makes it more tiring and makes it boresome for an individual. To avoid this stressful and boresome situation we have come up with a solution of allocating these tasks in such a way that it makes it easier to make the schedule. This algorithm helps to allocate and prioritize various tasks to be performed with taking considerations of stress and relief level of the person.

For example, if a person is school going so the individual has to give a fixed time in school and then later can perform activities such as completing assignments, studying, performing sports, performing various activities such as learning music, dance, spending time with friends and family, reading and similar other activities. It becomes difficult for a person to prioritize these various activities and decide which activity to perform. This task manager algorithm will help to prioritize these activities. If the slots for performing the tasks for particular day is full it will check the availability for the next day and so on and accordingly will allocate the task. This makes it easier to use the task manager algorithm as it provides the list of tasks to performed which will reduce the stress level of an individual

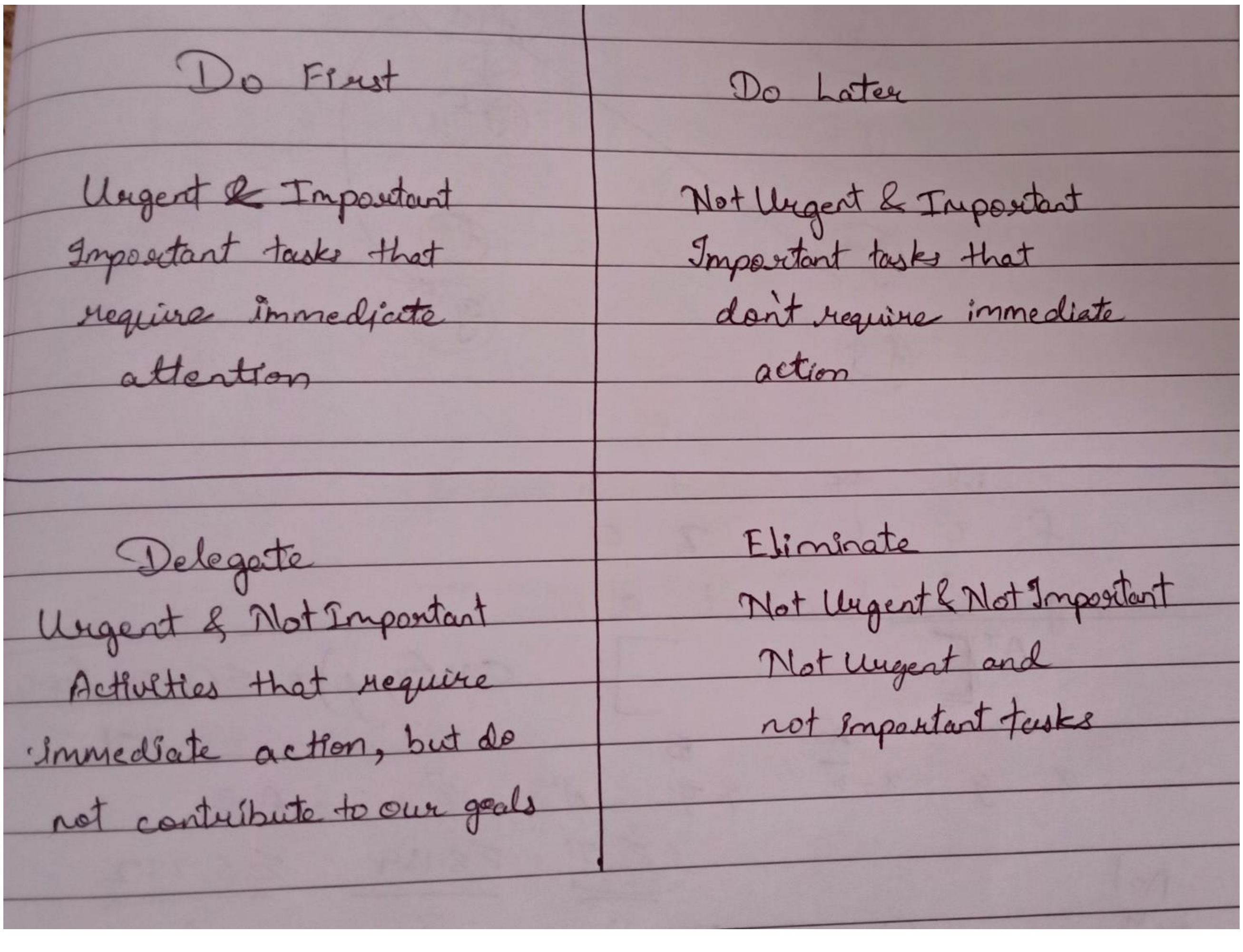
Task manager provides solution to those having lots of workload to do. It generates an efficient time table of tasks and allocates the tasks in the days balancing the levels of stress and relief. In case if the tasks could not be allocated, that means the tasks have been allocated an amount of time that exceeds the available time or the tasks do not meet a balance in the daily life, maybe the schedule results in a lot of Stress or a lot of Relief ;either of the sides not being much favourable ,considering the balance. In that case, the user needs to delete some tasks or lessen the amount of allocated time that he needs for the individual tasks. Here, our Eisenhower's decision matrix will help in prioritizing the tasks. This manager will support user's daily (mostly preferred by MBBS students) or Weekly goals (mostly preferred by BE students).

Chapter 2

Literature Review

The Eisenhower Matrix, also referred to as Urgent-Important Matrix, helps you decide on and prioritize tasks by urgency and importance, sorting out less urgent and important tasks which you should either delegate or not do at all.

Prioritizing tasks by urgency and importance results in 4 quadrants with different work strategies:[1]



Do First Urgent & Important Important tasks that require immediate attention	Do Later Not Urgent & Important Important tasks that don't require immediate action
Delegate Urgent & Not Important Activities that require immediate action, but do not contribute to our goals	Eliminate Not Urgent & Not Important Not Urgent and not important tasks

The first quadrant *Do first* as its tasks are important for your health and career and need to be done today or tomorrow at the latest. You could use a timer to help you concentrate while trying to get as much of them done as possible.
An example of this type of task could be to review an important document for your manager.

The second quadrant we call *Schedule*. Its tasks are important but less urgent. You should list tasks you need to put in your calendar here.
An example of that could be a long-planned restart of your gym activity.
Professional time managers leave fewer things unplanned and therefore try to manage most of their work in the second quadrant, reducing stress by terminating urgent and important to-dos to a reasonable date in the near future whenever a new task comes in.

The third quadrant is for those tasks you could *delegate* as they are less important to you than others but still pretty urgent. You should keep track of delegated tasks by e-mail, telephone or within a meeting to check back on their progress later.
An example of a delegated task could be somebody calling you to ask for an urgent favor or request that you step into a meeting. You could delegate this responsibility by suggesting a better person for the job or by giving the caller the necessary information to have him deal with the matter himself.

The fourth and last quadrant is called *Don't Do* because it is there to help you sort out things you should not being doing at all.

Discover and stop bad habits, like surfing the internet without a reason or gaming too long, these give you an excuse for not being able to deal with important tasks in the 1st and 2nd quadrant.[2]

Chapter 3

Implementation of Algorithms

There are two types of tasks, the fixed and the variable tasks. Fixed tasks are those tasks whose time cannot be changed. Its allocated and it has to be carried out during a particular period of time itself. These represent the daily busy schedules.

Another type of tasks is variable tasks. These tasks are those whose duration is known but the time is not fixed. These can be carried out anytime in between the fixed tasks. There exists a need of such an algorithm to calculate all the factors and accordingly fit the task in one of the available spaces present in between the fixed tasks.

We have 2 defined algorithms: -

First, the algorithm to allocate the fixed tasks. This checks whether the specified task can be allocated or not, whether it is perfectly valid, whether that spot has already been preoccupied fully or partially by any other task? It is going to check all these factors and thus will let the user know if he can carry out that particular task or not. The algorithm proceeds as follows. There are several inputs involved in these calculations. Those are the TaskName, its related stress/relief level, From and to, i.e the time from when to when it is supposed to be checking if it can be allocated or not. First, it converts the factors, hours to minutes. Hours will be multiplied by 60 and the minutes will be added to it. There involve different calcuations for am and pm.The day starts with am. So, *1am* is represented as *1*60=60 minutes*. But 1pm will be represented as *1*60+720* because, in a day there are *24 hrs*. *24*60=1440 minutes*. If *12am* is assumed to be started on the *0th minute*, then *1am* is on *60th minute* while *1pm* will come on 780th minute. Thats how we distinguish between 1am and 1pm.The exceptional value of hour is *12. 12*60* yields a totally different value which is logically wrong if day is assumed to be starting from the *0th minute*. Hence, this exceptional value must be scaled to the minutes. *12*60=720*. Required *answer=0*. Thus, whatever answer comes ,*720* must be *subtracted* from it to scale *12 to 0*. Thus 12:00am to 12:59am will represent perfectly 0th to 59th minute of the day. Now changing the day changes a factor r, which again should be scalable above the *1440 elements*. Considering a full week, next day starts at *1441th minute* or *1440th array element*. This here *r=1* wherein we *subtract 1440* from the actual value. Similarly, for all the days, *r's value* goes from *0 to 6*, representing the 7 days of week, i.e. *7*24 hrs* of the week, i.e. *7*24*60=10080* minutes in a week.
After getting the from and to value, now validation check must be performed. The fixed tasks of that particular user is fetched from the database. If the minutes are still unallocated from FROM to TO, the specified task is perfectly suitable to be alloted in that span, else some other task already has occupied those minutes. Thus, if perfectly valid, the task is stored in the fixed database. A value is assigned to a task. This is a unique one, also represents the Task number.

If any task exists in the fixes database, that means the user has that particular tasks, out of which the latest would be whose val will be the maximum one. Thus, this task which is to be allocated must be allocated with a value of val+1, the incremented version of the previous one. Else ,if no task exists,We start a new counting scheme and that is from TaskNo=0.Thus the values are finally inserted into the database. These values are the username indicating the fixed task actually belongs to which user, the TaskNo i.e the algorithmically generated value, The TaskName, The stress /relief level associated with that particular task and From and To values.The same allocation will be reflected graphically when the user tries to see the Timetable .This timetable will be a weekly timetable wherein the slots alloted or occupied with fixed tasks will be represented by a different colour,the unallocated tasks with a different colour and the variable tasks with different colour,so that user gets a clear idea of which tasks are in his hand whose time he /she can change (variable tasks) and whose time he cannot(fixed tasks) .Accordingly the user can plan his schedule appropriately such that the stress related tasks will be balanced by relief giving tasks and not bombarded with another set of stressful tasks which is not suitable for users.This is how the fixed algorithm works. The minute's entries will be the value of the task i.e the TaskNo and another set of minutes indiating its Stress/Relief level which will be further considered in the allocation of variable tasks because these Levels affect the available blocks around themselves. So, effective Stress/Relief level of each available block will be calculated in the Variable algorithm And appropriately a suitable slot is chosen and the task is allocated.

Secondly, we have again two types of algorithms in the variable allocation of tasks. The variable tasks will be allocated, but the order was still a dilemma. Thus, the priority matrix comes into picture over here. The Priority matrix also known as the task matrix helps in priorotizing the tasks the user wants to allot in his current fixed timetable. This was a need, because the fixed tasks' time cannot be changed whatever the reason maybe. This time can be college timings, school timings, office job timings, some class timings, etc. Now, not every fixed task is supposed to be stressful. Maybe, a task which is stressful to some user, the same task is a relief giver to the another. Example, studying a language for instance, maybe a stressful task to someone, but they have to do it. There might be millions of reasons for the same. One of the reasons couldbe that, they have paid the fees and cannot afford to not attend this language learning. Thus, this task maybe a stressful one for someone,while the same can be a relief giver to the another as many enjoy learning a new language ,it maybe their hobby.But hobbies are time bound too if you're learning them via some educational institute or a particular class. Thus, this application is super specific towards the users. Thus, every allocation of the variable tasks, no matter what, is going to be dependent on the fixed tasks of that particular user. Thus, these needs to be priorotized because maybe all tasks cannot be allocated. Best case would be that all the tasks are allocated. But this would be very rare for a person who wants to allocate number of tasks. The average case would be allocation of specific tasks. Which will be these tasks ? These tasks will be the ones which are having the highest importance and highest urgency of completion within a week. These two factors IMPORTANCE and URGENCY provide the basis for priorotizing any of the tasks. Accordingly, these tasks will be categorized into 4 different quadrants, viewing it graphically.

Task Matrix helps for clearing this dilemma as to which tasks must have the highest priority. The task with the highest priority will be chosen first or considered first for allocation .There will exist different available slots within the timetable. Every slot is going to have some stress/relief level attached to it.Accordingly, the Most suitable slot will be chosen for task1 and the next for task2 and so on .That means, if there is a stress giving task, the most suitable slots will be relief slots. The most relief slot will be chosen for this first task even if the second task is a stressful task because, Task1 has more priority ovet Task2.Thats what Task Matrix does,It helps priorotizing the tasks .Thus, there exists no dilemma as to which task must be choosen first to allocate. The tasks will be prioritized as follows:

There exists 4 quadrants graphically. *One* with URGENT and IMPORTANT TASKS.This is the quadrant number 1. Obviously, It is required to do the tasks in this quadrant because they are important, and one should do them first because they are urgent. Usually, if there is short on time we then one can do these tasks first.

Second, which are IMPORTANT but NOT URGENT tasks. This is quadrant 2. The tasks in this quadrant will be taken care after dealing with the tasks in Quadrant 1. The tasks in Quadrant 2 are not urgent, but one have to take very seriouslt because if they don't ,they will move to Quadrant 1.Its better to take care of these tasks before they appear in Quadrant 1 because there are atleast two problems in Quadrant1 tasks.We have to deal with them quickly and that can cause stress and worry even effect the quality of out work.

Third, which are NOT IMPORTANT, but URGENT TASKS. This is quadrant 3. Sometimes people bother with tasks that are not important but those tasks has to be done urgenly.Many times other people try to force to deal with these tasks.They are the time robbers.The way to reduce this problem is simply to protect the time. For example, if someone insists on talking in person immediately, and it means one need to travel out of their office, find out why that meeting is so urgent. Maybe they can have a phone call instead, and maybe that person is going to be in their area next week. Protect the time.

Lastly, the tasks which are NOT IMPORTANT, also NOT URGENT. The tasks in this quadrant are serious time wasters. When we identify a task in that quadrant, try to cancel it. If one can't eliminate it completely, try to minimize the time which has to be spent on that task.

When all tasks cannot be allocated, these tasks can be deleted blindly, because they are not even important not even urgent. So, one can consider deleting tasks from quadrant 2 and 3 as per their wishes . But, quadrant 1 tasks are very sensitive and should not be deleted. So, out Task Matrix even helps in identifying as to which tasks must be deleted and which must not. But in the end, it's up to the user themselves.

This is the first basis of classification or prioritization of the tasks. So, till here we will know which tasks belong to which quadrant. The prioritised quadrant's sequence is as follows: Quadrants 1,2,3,4 with Q1 being the most ranked and Q4 being the least ranked quadrant. Thus, Task Matrix , also called as Eisenhower's Decision matrix is going to decide the Quadrant of the task. Next up, the factors URGENCY and IMPORTANCE not just reveal the quadrants ,but can also prioritize the tasks if they exist in the same quadrants. Let's take a range of 0 to 10, that means the urgency of any task can be from 0 to 10 and that of Importance is the same too. In short, if importance is greater than or equal to 5, the task is important, else it's not. Same goes with urgency. So, according to that, the values of both urgency and importance above or equal to 5 gets classified in quadrant 1. The values of importance greater than or equal to 5 and urgency less than 5 gets classified in quadrant 2.The value of urgency greater than or equal to 5 and importance less than 5 will be classified in quadrant 3. And both the values being below 5 will be classified in quadrant 4. In these quadrants , if they lie in the same quadrant, they are further given ranks on the basis of they urgency and importance values itself.

So, let's consider two tasks for instance, *T1* Having *urg=6* and *imp=7*. *T2* having *urg=8* and *imp =8*. Here both tasks' factors are greater than equal to 5 as per the ranges set by us. Thus both belong to quadrant number 1. But if calculation of effective value of urgency and importance is taken, T2 is more urgent and important than T1. So, it ultimately gets more priority. Now the worst case will be that both stress and importance values match. In that case, the carpenter model comes into picture. Carpenter model defines that we can use a set of values to classify the tasks which fall into the same quadrants. Here we have used the duration value of the variable tasks. If the tasks are falling into same quadrants and are even having same effective importance and urgency,then their duration values will be checked.Whose duration is more is going to be more important than the ones which has less duration.Now,worst case, even if the durations are same, the priorities will be given in FCFS (First come first serve) basis. Thus, each task will be assigned a rank and on the basis of these ranks, these tasks will be considered for allocation. This is how the tasks will be ultimately prioritised and now can be considered for allocation from ranks 1 to n in order.

Variable logic is very simple, First we calculate the empty slots which are available after the fixed allocations. After that, we calculate their durations.This is very important because the task must be allocated in this unfilled space if and only if it fits in there , its duration is less than or equal to that of this duration. Then ,we calculate the overall stress in these available slots. IF the task to be allocated is Stressful , then the available slots will be sorted in ascending order because lower values indicates Relief. Also , the viceversa when a relief task is to be allocated. Then it will try to allocate the task in this free space, if it cannot it will try another available space. The task cannot be allocated at all if available space is not there. Even if the first task is not allocated , algorithm still tries to allocate the next tasks in the priority order. This is how it proceeds and the tasks will be either allocated or not,depending on the available blocks and stress/relief levels. Now, the user will be notified about which tasks are allocated from the considered ones and which are not . This is how allocation of variable tasks takes place.

Chapter 4

Technology Stack

Android Studio

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools.

To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and Github integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.

Android Studio uses an Instant Push feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering code completion, refraction, and analysis. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store.

The software was first announced at Google I/O in May 2013, and the first stable build was released in December 2014. Android Studio is available for Mac, Windows, and Linux desktop platforms. It replaced Eclipse Android Development Tools (ADT) as the primary IDE for Android application development. [3]

Features of Android Studio: [4]

Instant Run

Android Studio's Instant Run feature pushes code and resource changes to your running app. It intelligently understands the changes and often delivers them without restarting your app or rebuilding your APK, so you can see the effects immediately.

Intelligent code editor

The code editor helps you write better code, work faster, and be more productive by offering advanced code completion, refactoring, and code analysis. As you type, Android Studio provides suggestions in a dropdown list. Simply press Tab to insert the code.

Fast and feature-rich emulator

The Android Emulator installs and starts your apps faster than a real device and allows you to prototype and test your app on various Android device configurations: phones, tablets, Android Wear, and Android TV devices. You can also simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input.

Testing tools and frameworks

Android Studio provides extensive tools to help you test your Android apps with JUnit 4 and functional UI test frameworks. With Espresso Test Recorder, you can generate UI test code by recording your interactions with the app on a device or emulator. You can run your tests on a device, an emulator, a continuous integration environment, or in Firebase Test Lab.

Robust and flexible build system

Android Studio offers build automation, dependency management, and customizable build configurations. You can configure your project to include local and hosted libraries, and define build variants that include different code and resources, and apply different code shrinking and app signing configurations.

Optimized for all Android devices

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug.

Layout Editor

When working with XML layout files, Android Studio provides a drag-and-drop visual editor that makes it easier than ever to create a new layout. The Layout Editor was built in unison with the ConstraintLayout API, so you can quickly build a layout that adapts to different screen sizes by dragging views into place and then adding layout constraints with just a few clicks.

Code templates and sample apps

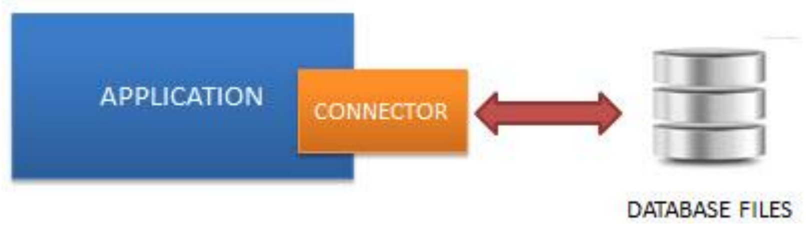
Android Studio includes project and code templates that make it easy to add well-established patterns such as a navigation drawer and view pager. You can start with a code template or even right-click an API in the editor and select *Find Sample Code* to search for examples. Moreover, you can import fully functional apps from GitHub, right from the Create Project screen.

SQLite database

SQLite is a software library that provides a relational database management system. The lite in SQLite means light weight in terms of setup, database administration, and required resource. SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional. SQLite is a software library that provides a relational database management system. The lite in SQLite means light weight in terms of setup, database administration, and required resource.

SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional.

The following diagram illustrates the SQLite server-less architecture:



Self-Contained

SQLite is self-contained means it requires minimal support from the operating system or external library. This makes SQLite usable in any environments especially in embedded devices like iPhones, Android phones, game consoles, handheld media players, etc.

Zero-configuration

Because of the serverless architecture, you don't need to "install" SQLite before using it. There is no server process that needs to be configured, started, and stopped.

Transactional

All changes within a transaction take place completely or not at all even when an unexpected situation like application crash, power failure, or operating system crash occurs.

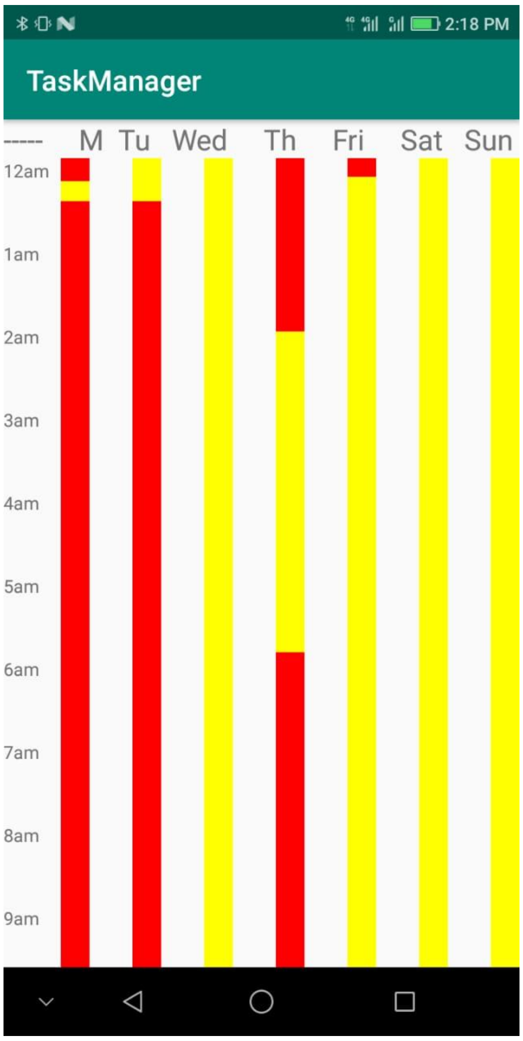
SQLite distinctive features

SQLite uses dynamic types for tables. It means you can store any value in any column, regardless of the data type. SQLite allows a single database connection to access multiple database files simultaneously. This brings many nice features like joining tables in different databases or copying data between databases in a single command.

SQLite is capable of creating in-memory databases which are very fast to work with.[5]

Chapter 5

Result



The output will be a timetable like above. It indicates the unallocated ,allocated(fixed) and allocated(variable) tasks so that the user gets a clear understanding of which time is utilised and how much slots are free so that he can plan the next tasks accordingly.Also, the priority wise arranged tasks are displayed to the user so that he/she is aware of which tasks were allotted after considerations of his input variable Tasks.

Chapter 6

Conclusions and Future Scope

Thus, the user does not have to do the tedious and boresome calculations of when to allocate the tasks that help developing himself . The software application does it for the user and allocates the best possible slot which he can get within his fixed timetable such that overall Stress/Relief levels are balanced and the user gets a balanced timetable. This is very important because no matter what tasks are done, one should not overdo tasks which lead to a lot of stress.

The application can be expanded further with Machine Learning . The user may get the suggestions of the stress/relief levels that must be given as an input in association to the average users. This is a classification problem and can be solved by Supervised Learning. The test cases will be the users who use them., and the results will keep updating on a frequent basis.

Also, Next feature can be the Health Manager. With a busy schedule for wanting to develop oneself, health should also be taken care of. Ensuring health is like ensuring king's safety. Here, a drop-down list of some diet items can be provided. The quantity-wise calories of these food items would be pre-known. User can sort the list in either low-calories to high-calories way (useful for gym goers), or low-cost to high-cost way. This ensures that the user can get maximum calories in an affordable diet. After selecting the food items, the Health manager will generate a diet plan which also will be including the timings when the user has to consume any food item and that too, in what amount. These timings will be internally dependent on the Task Manager's schedule. Reminders may be given to the user just like an alarm clock as per the food timings. This can be an additional feature, while the Task Manager is the main goal of the project.

Bibliography

[1]
https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiiw_3hj8rhAhWb73MBHTZtCccQjRx6BAgBEAU&url=https%3A%2F%2Fwww.eisedo.com%2Fblog%2Fabouteisedo%2F&psig=AOvVaw0hh_4ADb0Xt0Ao9hmxwWSU&ust=1555143648609567

[2] https://www.eisenhower.me/eisenhower-matrix/

[3] https://searchmobilecomputing.techtarget.com/definition/Android-Studio

[4] https://developer.android.com/studio/features.

[5] http://www.sqlitetutorial.net/what-is-sqlite

Appendices

Appendix-A: Android Download and Installation

Android Studio is Google's officially supported IDE for developing Android apps. This IDE is based on IntelliJ IDEA, which offers a powerful code editor and developer tools. Android Studio 3.2.1 includes the following features:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to help you catch performance, usability, version compatibility, and other problems

- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and Google App Engine
- Plugin architecture for extending Android Studio via plugins

Download Android Studio

Google provides Android Studio for the Windows, Mac OS X, and Linux platforms. You can download Android Studio from the Android Studio homepage, where you'll also find the traditional SDKs with Android Studio's command-line tools. Before downloading Android Studio, make sure your platform meets the following requirements:

Windows requirements

Microsoft Windows 7/8/10 (32-bit or 64-bit)

3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)

2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)

1280 x 800 minimum screen resolution

Mac OS requirements

Mac OS X 10.10 (Yosemite) or higher, up to 10.13 (High Sierra)

3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)

2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)

1280 x 800 minimum screen resolution

Linux OS requirements

GNOME or KDE desktop. Tested on Ubuntu 14.04 LTS, Trusty Tahr (64-bit distribution capable of running 32-bit applications)

64-bit distribution capable of running 32-bit applications

GNU C Library (glibc) 2.19 or later

3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)

2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)

1280 x 800 minimum screen resolution

Once you've ensured that your operating system is compatible with Android Studio 3.2.1 or higher, download the appropriate Android Studio distribution file. The Android Studio download page auto-detected that I'm running a 64-bit Windows operating system and selected android-studio-ide-181.5056338-windows.exe (927 MB) for me to download.

Acknowledgement

We have great pleasure in presenting the report on **Task Manager.** We take this opportunity to express our sincere thanks towards our guide **Prof. Mayuri Jain** Department of Computer Engineering, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof.Sachin Malave** Head of Department Computer Engineering, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Student: Tejas Deshmukh
Student ID: 16102002

Student: Akshay Sumbhe
Student ID: 16102062

Student: Falguni Tailor
Student ID: 16102031