

*Department of Artificial Intelligence*

*Date:*

**PROJECT KAIV**

**App Blocker**

**Authors:**

**Mohammed Sattar**

**Youssef El Nahas**

**Abubakar Waziri**

**Hamza AlKaf**



Dept. of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Faculty of Computer Science and Information Technology

University of Prince Mugrin, Madinah KSA

**CONTACT INFORMATION**



This project report is submitted to the Department of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ at University of Prince Mugrin in partial fulfillment of the requirements for the course  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**AUTHOR(S):**

Mohammed Sattar

E-mail: [mohammed.sattar1437@gmail.com](mailto:mohammed.sattar1437@gmail.com)

Abubakar Waziri

E-mail: [abuwaziri@outlook.com](mailto:abuwaziri@outlook.com)

Youssef El Nahas

E-mail: [elnahasyoussef123@gmail.com](mailto:elnahasyoussef123@gmail.com)

Hamza AlKaf

E-mail: [hthek19@gmail.com](mailto:hthek19@gmail.com)

**UNIVERSITY SUPERVISOR(S):**

Dr. Abdurazzag Almiladi

Department of Computer and Cyber Sciences



Dept. of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Internet: [https://www.upm.edu.sa](https://www.upm.edu.sa/en)

Faculty of Computer Science and Information Technology E-mail: [info@upm.edu.sa](mailto:info@upm.edu.sa)

University of Prince Mugrin Phone: +966 014 831 8484

Kingdom of Saudi Arabia

**INTELLECTUAL PROPERTY RIGHT DECLARATION**

****

This is to declare that the work under the supervision of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ having title: “\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_” carried out in partial fulfillment of the requirements of the course \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, is the sole property of the University of Prince Mugrin and the respective supervisor and is protected under the intellectual property right laws and conventions. It can only be considered/ used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, etc., with the permission of the University and respective supervisor.

This above statement applies to all students and faculty members.

**DATE:**

|  |  |
| --- | --- |
| **AUTHOR(S):** | |
| Name: Mohammed Sattar | Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  |  |
|  |  |
| Name: Abubakar Waziri | Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  |  |
|  |  |
| Name: Youssef El Nahas | Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  |  |
|  |  |
| Name: Hamza AlKaf | Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**SUPERVISOR(S):**

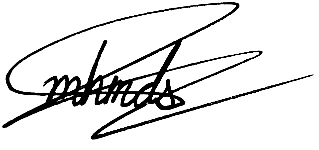
Name: Dr. Abdurazzag Almiladi Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ANTI-PLAGIARISM DECLARATION**

****

This is to declare that the above publication produced under the supervision of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ having title “\_\_\_\_\_\_\_\_\_\_\_\_\_” is the sole contribution of the author(s) and to best of our knowledge no part hereof has been reproduced illegally (cut and paste) which can be considered as plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: 

****

**AUTHOR(S):**

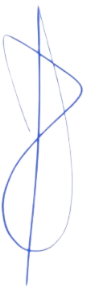
Name: Mohammed Sattar Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Name: Abubakar Waziri Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Name: Youssef El Nahas Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Name: Hamza AlKaf Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ABSTRACT

In this project, an app blocker (KAIV) that can enhance user productivity and minimize distractions is presented. First, we start by introducing the already existing systems and their defects solved by our products. Furthermore, a diagram demonstrating the flow of the data is shown, followed by the functional and non-functional requirements of our software. We will then dive into the details of the design considerations, including the design constraints and the architectural strategies, such as the algorithm to be used, the management strategies, and the future enhancements. The system architecture is then discussed, along with the detailed system design as well as the components. Proceeding to the implementation and validation stage, where we introduce our coding standards and testing strategies in addition to using restrictions. Finally, a conclusion is drawn, summarizing every phase this project has gone through, followed by the references and the source code, which will be written in Java programming language. **TABLE OF CONTENTS:**

|  |  |
| --- | --- |
| Contact Information |  |
| Intellectual Property Right Declaration |  |
| Anti-Plagiarism Declaration |  |
| Table of Contents |  |
| Abstract |  |
| **– Introduction**  * 1. Purpose of the Project   2. Problem Description   3. Proposed Solution   4. Existing Systems      1. Description of the existing systems      2. Problems found in existing systems   1.5 Stakeholders |  |
| **– System Requirements**  2.1 Functional Requirements  2.2 Non-Functional Requirements |  |
| – Architecture 3.1 Architectural Strategies  3.1.1 Reuse of Existing Software Components  3.1.2 Project Management Strategies  3.2 Overall Architecture |  |
| **– System Design** 4.1 Use-Case Diagram  4.2 Sequence Diagram  4.3 Class Diagram  4.4 Component Diagram  4.5 Activity Diagram  4.6 State Diagram |  |
| **– Implementation And Validation** 5.1 Functional Code in Java  5.2 GUI Code in Java  5.3 Testing Techniques  5.4 Future Enhancements/Plans |  |
| **– Conclusion** |  |
| **References** |  |

**CHAPTER 1 INTRODUCTION**

In recent years, computers have become a fundamental part of our lives due to the exceptional variety of ways that computers can be used. They went from special gadgets used by the few to a seriously powerful machine present in the mass population. However, as this use of computers expands, the amount of time spent using them ineffectively grows dramatically. This especially affects university students because there can be a lot of electronic assignments, but there are also a lot of diversions. Ultimately, it leads to computers being a waste of time and money instead of the genuinely helpful tool it actually is.

As a result, certain applications and websites started developing methods to combat such a phenomenon. Implementation of such includes setting a “bedtime” that the program will remind you to sleep or setting time limits for usage. Unfortunately, neither of which are very effective and can be easily bypassed or ignored as neither enforces the user to stop.

One type of application which does enforce good behavior of using a computer is website/app blockers. App blockers have been expanding in both types and features to try to combat computer unproductivity. Generally, app blockers do not allow users to access certain websites or applications breeding a remarkably productive computer usage.

Our app blocker has been named KAIV as it simply puts the user in “cave” of productivity without any distractions.

* 1. **PURPOSE OF THE PROJECT**

The most important function of any app blocker would be the ability to block access to websites and applications without disrupting the other apps or making the computer unusable. Furthermore, it should have built-in methods of ending the block when needed. To maximize efficiency and productivity and minimize time sinks.

We designed our own app blocker (KAIV) to achieve the mentioned goals and to be able to aid University of Prince Mugrin (UPM) Students and improve the overall academic performance. This program will be useful to students from doing simple assignments at home to doing online quizzes.

* 1. **PROBLEM DESCRIPTION**

After some informal interviews with some fellow students, we have found that many students face a similar problem when trying to study or solve given assignments. That shared problem is distraction by web applications, websites, and in the case of phones, apps. In layman’s terms, most students as they start trying to study or work, they get easily distracted by notifications from websites or web applications. Or in some cases, even by just the sight or the thought of them. This is mostly reoccurring in games, social media websites, streaming services, and even sometimes YouTube.

Initially, the problem may seem inconsequential or effortlessly avoidable. However, the problem is bigger than it seems. To elaborate, for example, after getting distracted by an application or a website, the problem can easily exacerbate leading to the student doing none of the tasks they needed to do. This can be in the form of eventually forgetting the task at hand after spending wasted hours on any of these websites or applications. By the time the student remembers, it would be already too late in the night, or they become sluggish allowing them to sink even more hours into useless activities.

This issue stems from the susceptibility of the student for distractions due to uninterest in the task, laziness, procrastination, or in some cases, physical and emotional disturbances. The true cure would have to be specific to what type of issue the student is facing; however, there are ways to mitigate this issue effectively.

* 1. **PROPOSED SOLUTION**

We studied the previously proposed solutions and found that all have their faults. The “bedtime” that some web applications have is extremely inefficient as it only seems like a suggestion that can be ignored and even deleted. That is probably due to the fact that every web application or website does not want what is best for you (to study or work). Instead, each of them wants you to spend the most time on it. The reason being: the more time you spend on the site or application, the more interaction you will have with it, the more ads you will see, the more money they will make. Sadly, that is the reality you have to accept.

(The other solution which involves the app blockers will be discussed in the next section)

Considering all of what was previously mentioned, our decision was to create a web application that basically blocks web applications and websites. We found that this was the best solution to counter the aforementioned problem that students are facing. It involves inclusivity for students, particularly those in UPM, and useful functionality that solves the main struggles students face on a daily basis.

* 1. **EXISTING SYSTEMS:**

Existing focus apps and distraction-blocking software aim to enhance productivity by curbing distractions on electronic devices. These systems employ features like website and app blocking, scheduling, and timers to promote focused work. Some notable options include:

1. Freedom: Blocks distractions across multiple devices simultaneously, supports multiple blocklists, and offers optional focus sounds.
2. Cold Turkey Blocker: Provides customizable website and app blocking, scheduled blocking, and a "Frozen Turkey" mode that locks you out of your computer entirely.
3. LeechBlock NG: A browser-based free tool that allows for scheduling, time limits, and lockdown sessions on distracting websites.
4. RescueTime: Primarily a time-tracking app but includes distraction-blocking features based on your tracked activities.
5. SelfControl: A hardcore Mac-only app that makes it impossible to disable blocking once set, focusing on URL blocking.
6. Session: Integrates Pomodoro timer functionality with distraction blocking, offering various integration options for advanced users.
7. one sec: Delays distracting site access, prompting users to pause and reflect on impulsive distractions.
8. PawBlock: A browser extension using cute animal pictures to discourage visits to distracting sites, allowing for both soft and hard blocks.

* + 1. **DETAILED DESCRIPTION OF THE EXISTING:**

1. Freedom: This app stands out for its ability to block distractions simultaneously across multiple devices. It supports various blocklists and offers optional ambient sounds conducive to focus. However, the full version requires a subscription fee.
2. Cold Turkey Blocker: Highly customizable, it allows users to block specific websites and apps on schedules they set. Its "Frozen Turkey" mode can lock users out of their computers entirely. Once purchased, it offers unlimited blocking without a subscription.
3. LeechBlock NG: A browser-based tool available for free across major browsers, it allows users to schedule blocks, set time limits on distracting sites, and even create rules for site access within specified time frames.
4. RescueTime: Primarily a time-tracking app, RescueTime includes a distraction-blocking feature called Focus Session, which blocks distractions based on the user's tracked activities. The blocking feature is available in the paid version.
5. SelfControl: Designed for Mac users, SelfControl is an intense blocker that prevents access to specific URLs. It's known for its extreme effectiveness in blocking distractions; once set, it's nearly impossible to disable.
6. Session: Integrating Pomodoro timer functionality with distraction-blocking capabilities, Session allows users to block specific apps and websites during work sessions, automatically unblocking them during breaks. It's a bit pricey compared to other options.
7. one sec: This app doesn't outright block distractions but introduces a delay when accessing distracting sites or apps. It prompts users to pause and reflect on their impulsive distractions before allowing access.
8. PawBlock: Using cute animal pictures, PawBlock aims to discourage users from visiting distracting sites. It can be set to either allow users to choose to enter distracting sites after a delay (soft block) or completely block access (hard block).

Each of these tools boasts unique features and pricing models, making the selection process dependent on individual preferences, required levels of control, and the platforms used. Users can choose based on their specific needs for distraction control and productivity enhancement, considering the array of features and compatibility across devices offered by each system.

* + 1. **PROBLEM FOUND IN THE EXISTING SYSTEM:**

The existing focus apps and distraction-blocking software, while effective, do have their share of limitations and potential issues:

1. Effectiveness vs. User Control: Some users might find these tools too restrictive or not customizable enough. Balancing effective distraction blocking with user control and flexibility is a challenge.
2. Platform Limitations: Certain apps might be limited to specific platforms (e.g., Mac-only, or browser-specific), restricting accessibility for users across various devices and operating systems.
3. Over-blocking or under-blocking: Striking the right balance between blocking distractions effectively and avoiding over blocking (or under blocking) legitimate work-related sites or apps can be challenging.
4. User Adaptability: While some users might benefit from strict blocking, others may find ways to work around these tools or become desensitized to their effects over time, reducing their effectiveness.
5. Cost: Some comprehensive or feature-rich tools often come with subscription fees or one-time purchase costs, making them less accessible to users with budget constraints.
6. Maintenance and Updates: The need for regular updates and maintenance to adapt to new websites, apps, or system updates to ensure consistent and effective blocking can be demanding for developers.
7. Interference with Legitimate Tasks: In some cases, these tools might unintentionally interfere with legitimate tasks or workflows, causing frustration and hindering productivity.
8. Performance Impact: Certain apps might consume significant system resources or cause performance issues, especially on older devices, impacting the overall user experience.

Addressing these issues involves a balance between enhancing the effectiveness of distraction-blocking while providing users with enough control and adaptability to cater to individual needs without imposing unnecessary limitations.

* 1. **STAKEHOLDERS**

Due to our system being all-purpose and can be used by anyone, this system only has one type of stakeholder, the end-user.

**End-User:** those who use our program.

These include students(mainly), lecturers, and even parents. However, our program provides multiple functionalities that can help any of the aforementioned in different ways.

**CHAPTER 2: SYSTEM REQUIREMENTS**

//Sattar//

**2.1 FUNCTIONAL REQUIREMENTS**

Our software (KAIV) is designed to cater to the specific needs of the user in all professions. This means prioritizing features that empower users to tailor their experience. The functional requirements outlined below reflect the key capabilities our software aims to deliver.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Priority**  **1 - 5** | **Description** | **Type** | **Source** |
| FR1 | 5 | The user should be able to define a password for entering the software | Product | Client |
| FR2 | 4 | The user shall be able to define a daily limit for websites/apps | Product | Client |
| FR3 | 5 | The user shall be able to enable the block. | Product | Client |
| FR4 | 5 | The user shall be able to disable the block if it isn’t locked. | Product | Client |
| FR5 | 5 | The user shall be able to set the lock. | Product | Client |
| FR6 | 5 | The user shall be able to define a blacklist to block specific websites/apps. | Product | Client |
| FR7 | 5 | The user should be able to define a whitelist for enabling only specific websites | Product | Similar Systems |
| FR8 | 2 | The user should be able to create a deadlock that blocks entire device | Product | Similar systems |
| FR9 | 3 | The user shall be able to set a time range disable/enable websites/app | Product | Client |
| FR10 | 4 | The user shall be able to access the program main page | Product | Client |
|  |  |  |  |  |

Incorporating these functional requirements ensures that our software is not only user-friendly but also customizable to meet the unique preferences and goals of each user. Each feature contributes to the overall objective of enhancing productivity and promoting a healthy digital lifestyle.

**2.2 NON-FUNCTIONAL REQUIREMENTS**

In addition to the functional capabilities, our productivity and time management software adhere to non-functional requirements to ensure a robust, secure, and user-friendly experience.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Priority**  **1 - 5** | **Description** | **Type** | **Source** |
| NFR1 | 5 | Availability | Product | Client |
| NFR2 | 3 | Re-usability | Product | Others (online sources) |
| NFR3 | 5 | Usability | Product | Similar systems |
| NFR4 | 2 | Security | Product | Others (online sources) |
| NFR5 | 2 | Scalability | Product | Similar systems |
| NFR6 | 4 | Maintainability | Product | Similar systems |
| NFR7 | 3 | Safety | Product | Others (online sources) |
| NFR8 | 2 | Portability | Product | Similar systems |
|  |  |  |  |  |

These non-functional requirements complement the software's functionality, ensuring that it not only meets user needs but also goes to beyond to provide the user with a pleasant experience.

**CHAPTER 3: ARCHITECTURE**

//Anyone//

**3.1 ARCHITECTURAL STRATEGIES**

//Anyone//

**3.1.1 REUSE OF EXISTING SOFTWARE**

The main reuse of any existing software components was made in the Graphical User Interface part of the code. As they were simple and not that much creative effort was made, the GUI was just made as a proof of concept. Therefore, it justifies the heavy reuse of software components. These reused components were derived mainly from Artificial Intelligence Helpers like ChatGPT, Bing AI, and Google Bard. The examples vary from single-use GUI to the main

**3.1.2 PROJECT MANAGEMENT STRATEGIES**

**Development Methodology: Waterfall Method**

The development of our productivity and time management software followed the Waterfall Model, a traditional linear and sequential approach to software development. This model was chosen for its clear structure and well-defined stages, allowing for a systematic progression from requirements gathering to deployment.

**Stages of Waterfall Method**

Our journey through the Waterfall Model involved distinct stages, each building upon the previous one:

1. **Requirements Gathering:** We began by thoroughly understanding the needs and expectations for our software, setting the stage for what it should accomplish. Since the intended end-users includes students, we were able to decide on the useful features that should be included.

1. **Design:** With a clear understanding of requirements, the team delved into designing the software's architecture. This stage saw the creation of UML design diagrams, including the Use Case, State, Sequence, Activity, and Class diagrams. The Use Case Diagram laid out user interactions (the functional requirements), the State Diagram captured dynamic behaviors, the Sequence Diagram depicted component interactions, the Activity Diagram visualized task flow, and the Class Diagram outlined system structures.

1. **Implementation:** Armed with comprehensive designs and a clear understanding of the software structure, the team transitioned to the implementation phase.

1. **Testing:** Rigorous testing followed implementation, ensuring that the software met specified requirements and was free from bugs. This phase involved both unit testing of individual components and system-level testing to verify overall functionality.

1. **Deployment:** Once testing confirmed the software's stability and performance, it was deployed as an open-source software on GitHub available for anyone to use, make changes and provide feedback.

**Division of Tasks**

The project team adopted a task-centric approach, where each team member was assigned a specific task aligned with one of the five UML design diagrams. This division ensured that each aspect of the software's design and functionality was thoroughly addressed by the team. Regular collaboration and communication were essential to maintain coherence across the diagrams and guarantee a holistic view of the software.

**UML Design Diagrams**

**Use Case Diagram**

The Use Case Diagram was employed to identify and define the interactions between end-users and the system. This diagram served as a blueprint for understanding the software's functionalities from a user's perspective.

**State Diagram**

The State Diagram focused on modeling the dynamic behavior of the system by illustrating its various states and the transitions between them. This allowed the team to capture the software's responses to different stimuli and events.

**Sequence Diagram**

For depicting the interactions between different components of the system, the Sequence Diagram was chosen. It provided a visual representation of the order in which tasks were executed, helping to refine the software's internal logic.

**Activity Diagram**

The Activity Diagram aided in visualizing the flow of activities within the software, offering insights into the sequential and parallel processing of tasks. This diagram was particularly valuable for understanding the software's workflow.

**Class Diagram**

The Class Diagram served as the backbone of the software's design by outlining the relationships and structures of the classes within the system. This diagram provided a comprehensive overview of the software's architecture.

**Version Control: Git & GitHub**

To ensure efficient collaboration and version tracking, the team utilized Git as the version control system, coupled with GitHub as the centralized repository. This approach allowed for seamless integration of individual contributions, easy identification and resolution of conflicts, and the ability to roll back to previous versions if necessary.   
   
In conclusion, the combination of the Waterfall Model, UML design diagrams, and effective version control through Git and GitHub contributed to a well-organized and collaborative project management strategy. This approach facilitated a systematic development process, ensuring the successful completion of our productivity and time management software.

**3.2 OVERALL ARCHITECTURE**

//Waziri//

**CHAPTER 4: UML DIAGRAMS**

In this chapter, the major types of UML Diagrams (Use Case, Sequence, Class, Activity, State Diagrams. In addition to that, an additional UML Diagram was added as it was simple and straightforward. This extra diagram is component diagram.

**4.1 USE CASE DIAGRAM**

//Sattar//

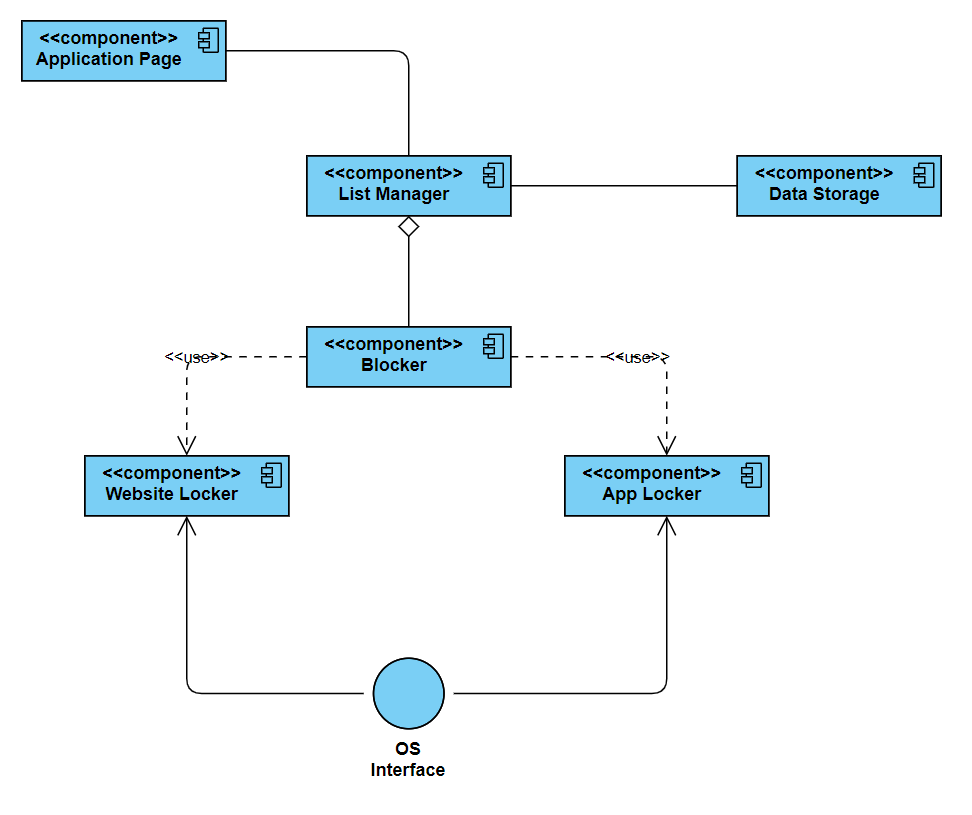
**4.2 SEQUENCE DIAGRAM**

//Hamza, Wait//

**4.3 CLASS DIAGRAM**

//Waziri, Wait//

**4.4 COMPONENT DIAGRAM**



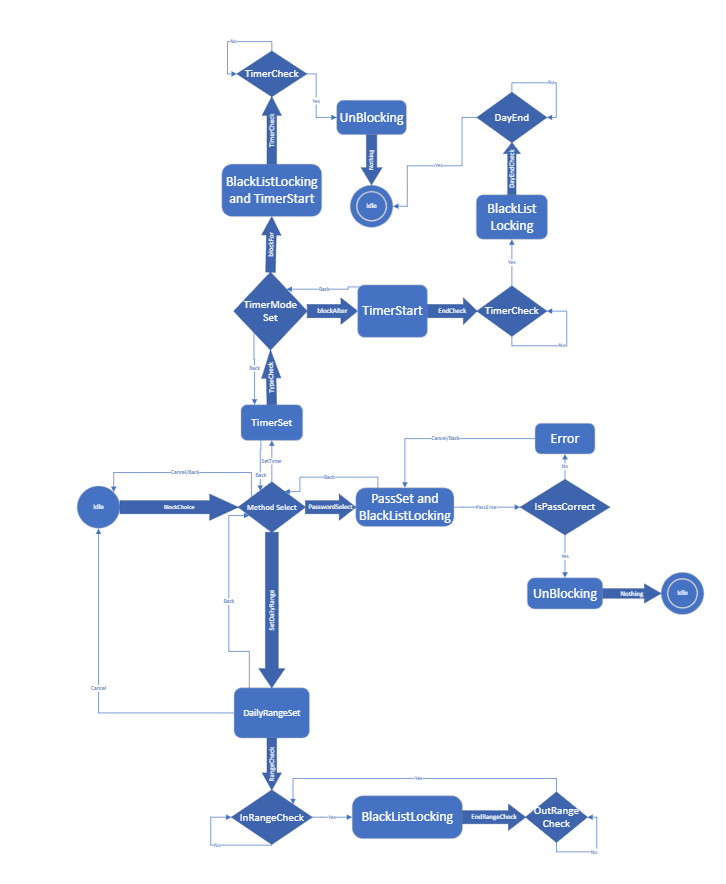
**4.5 ACTIVITY DIAGRAM**

//Sattar//

**4.6 STATE MACHINE DIAGRAM**

A diagram of a diagram

Description automatically generatedA diagram of a flowchart

Description automatically generated

Note: There are two other state diagrams, but they were ignored dues to the extreme similarity of them with one of the above state diagrams.

**State Footnotes:**

|  |  |
| --- | --- |
| **StartMenu** | |
| **Idle** | The program is currently doing nothing |
| **() SetName** | Setting the name of the chosen list (can be done for Blacklist/ Deadlock) |
| **() Config** | Configuring the list: a field that asks the user to enter the website URLs to put in blacklist/whitelist one at a time. Entering \* should block all. |
| **Save** | Saving the current block into database |
| **WhichTask** | Options presented for user and user picks which task he would like to do |
| **() Rename** | Renaming the chosen list (can be done for Blacklist/ Whitelist/ Deadlock) |
| **() Delete** | Deleting the chosen list (can be done for Blacklist/ Whitelist/ Deadlock) |
| **UnBlocking** | Unblocking the currently active and chosen list |
| **() Add** | Adding an extra website/app to the chosen list (can be done for Blacklist/ Whitelist) |
| **() Remove** | Remove an existing website/app to the chosen list (can be done for Blacklist/ Whitelist) |
| **Locks** | |
| **Idle** | The program is currently doing nothing |
| **Method Select** | User has the choice of all available methods of blocking and user chooses which method he would like to choose |
| **BlackListLocking** | Locking all websites/apps in the blacklist |
| **DeviceLocking** | Locking all websites/apps (completely locking the device) |
| **WhitelistLocking** | Locking all websites/apps that are NOT in the whitelist |
| **TimerSet** | Setting a timer |
| **TimerCheck** | Is the Timer done? |
| **DayEnd** | Did the day end? (24 hour reset) |
| **TimerStart** | Timer has started |
| **TimerEnd** | Timer has ended |
| **TimerModeSet** | Choosing to lock after timer ends, or to block for duration of timer |
| **DailyRangeSet** | Setting a range of time of the day (e.g., 8am - 4pm, 10pm - 6am) |
| **InRangeCheck** | Is the current time in range |
| **OutRangeCheck** | Is the current time out of range |
| **PassSet** | Sets the password |
| **IsPassCorrect** | Is the password entered correct |
| **Error** | Incorrect password entered/Timer is done |
| **UnBlocking** | Unblocking the currently active and chosen list |
| **AdminPass** | Enter the administrator password |
| **WhiteVSBlack** | |
| **Idle** | The program is currently doing nothing |
| **Comp Comparison** | Comparing blacklist with whitelist to find if any website/ application is in both |
| **WhiteOrBlack** | If a whitelist URL/application conflicts with a URL/application in the blacklist of the same block, priority is given to the whitelist. If a whitelist URL/application c conflicts with a URL/application in the blacklist of a different block, priority is given to the blacklist |
| **WhitePriority** | The URL/application c in the whitelist is more important so the same URL/application c in the blacklist is neglected |
| **BlackPriority** | The URL/application in the blacklist is more important so the same URL/application in the whitelist is neglected |

**Action Footnotes:**

|  |  |
| --- | --- |
| **StartMenu** | |
| **Proceed** | User chooses to proceed to the next step by clicking the corresponding button |
| **Back** | User chooses to go back to the previous step by clicking the corresponding button |
| **Cancel** | User chooses to cancel the current step by clicking the corresponding button |
| **CreateLists** | User chooses to start creating the lists (blacklist and whitelist) by clicking the corresponding button |
| **CreateDeadlock** | User chooses to start creating a deadlock for the device by clicking the corresponding button |
| **EditLists** | User chooses to edit the currently saved lists by clicking the corresponding button |
| **Rename** | User chooses to rename one of the saved lists by clicking the corresponding button |
| **Delete** | User chooses to delete one of the saved lists by clicking the corresponding button |
| **Stop** | User chooses to stop one of the currently active blocks by clicking the corresponding button |
| **Start** | User chooses to start the block for one of the saved lists by clicking the corresponding button |
| **Add** | User chooses to add a new URL/application to one of the saved lists by clicking the corresponding button |
| **Remove** | User chooses to remove an existing URL/application to one of the saved lists by clicking the corresponding button |
| **Locks** | |
| **Cancel** | User chooses to cancel the operation |
| **Back** | User chooses to go back by clicking the “back” button |
| **Yes** | Previous condition is true |
| **No** | Previous condition is false |
| **BlockChoice** | User chooses that he/she wants to block the previous set blacklist/whitelist or deadlock |
| **SetDailyRange** | User chooses the Daily Range Block Type by clicking the corresponding button |
| **PasswordSelect** | User chooses the Password Block Type by clicking the corresponding button |
| **SetTimer** | User chooses the Timer Block Type by clicking the corresponding button |
| **TypeCheck** | User gets presented to which way he/she want to activate the timer block (Block for a duration of timer, or after the timer ends till the end of the day) |
| **blockFor** | User chooses to Block the chosen list/device for a duration of the timer by clicking the corresponding button |
| **blockAfter** | User chooses to Block the chosen list/device after the timer ends till the end of the day by clicking the corresponding button |
| **EndCheck** | Automatically switches the state to checking the timer ended or not |
| **DayEndCheck** | Automatically switches the state to checking if the day has ended or not |
| **PassEnter** | User chooses to enter password by clicking the corresponding button, and types it then presses proceed |
| **RangeCheck** | Automatically switches the state to constantly checking if time entered the set daily range |
| **EndRangeCheck** | Automatically switches the state to constantly checking if time exited the set daily range |
| **Emergency** | Emergency cancellation of the block |
| **Nothing** | Automatically switches state to idle |
| **WhiteVSBlack** | |
| **Background** | The program does the next state in the background |
| **CompCheck** | Automatic action to go check the components of the blacklist and whitelist |
| **Yes** | Previous condition is true |
| **No** | Previous condition is false |

**CHAPTER 5: IMPLEMENTATION AND VALIDATION**

**5.1 FUNCTIONAL CODE IN JAVA**

**5.2 GUI CODE IN JAVA**

**5.3 TESTING TECHNIQUES**

In this software project, several testing techniques were used. Most of the testing were mainly in the Development Testing Stage. That is due to the fact that our program has not been released to the public as of yet. Therefore, neither of the other two testing stages were implemented. Nevertheless, several testing techniques were used in separate stages.

Unit Testing:

Whenever a function has been produced extensively during and after development to ensure its validity. The main ways in which a function is tested are twofold: The Print Technique and the dummy-value technique. The former being the idea that the function would be riddled with effective print statements to discover where exactly is the bug happening. The latter simply being the concept of assigning random (sometime nonsensical) values to a function to try to discover any easy-to-find vulnerabilities.

Component Testing:

As the size of the component increases over development time, there are one straightforward concept we implemented to avoid complication. That concept being separating the component into several different classes. This might seem arbitrary; however, it greatly reduced the time taken to test several components (altogether or separate)

System Testing:

As the program is still in development to fix some bugs or add some additional features or even functionalities, the system itself hasn’t been fully tested from start to finish extensively. Instead we just focused on the main functionalities and tested all these together and they were all running with only minor or negligible issues rarely faced.

**5.4 FUTURE ENHANCEMENTS AND PLANS**

The future enhancements in focus apps and distraction-blocking software could revolve around addressing current limitations and introducing innovative features to enhance user experience and effectiveness:

1. Adaptive AI Algorithms: Implementing AI-driven systems that learn from user behaviour to dynamically adjust blocking thresholds, recognizing legitimate work patterns, and optimizing distraction-blocking accordingly.
2. Cross-Platform Integration: Developing seamless integration across various devices and operating systems to ensure a consistent distraction-blocking experience regardless of the device being used.
3. Enhanced Customization: Providing users with more granular control over blocking settings, allowing for personalized schedules, exceptions for specific tasks or sites, and varied levels of blocking intensity.
4. Behavioural Insights and Analytics: Offering detailed analytics on user behaviour and time spent on different tasks, aiding users in understanding their productivity patterns and suggesting improvements.
5. Gamification and Positive Reinforcement: Introducing gamified elements or positive reinforcement strategies within the apps to incentivize and reward focused work periods, making productivity more engaging.
6. Collaborative Features: Enabling features that allow for team-based or collaborative distraction-blocking settings, ideal for workplaces or shared productivity environments.
7. Enhanced User Feedback and Reporting: Providing users with detailed feedback on their productivity, including insights into improvements made, time saved, and distractions avoided.
8. Improved Performance and Efficiency: Optimizing the software to ensure minimal impact on device performance while maintaining effective distraction-blocking capabilities.
9. Adapting to Evolving Technologies: Staying up to date with emerging platforms, technologies, and trends to ensure compatibility with new apps, websites, and devices.
10. Integrations and Ecosystem Support: Offering integrations with other productivity tools or ecosystems, such as calendar apps, task managers, or project management platforms, for a more holistic productivity approach.

These future enhancements aim to make distraction-blocking tools more intelligent, adaptable, and user-friendly, catering to the evolving needs of individuals and organizations striving for improved productivity in an increasingly digital world.

**CHAPTER 6: CONCLUSION**

In this project, we proposed a blocking system that can block installed apps as well as online websites to enhance the productivity of the user. Additionally, this system attains multiple well-needed features, including blocking certain platforms, blocking all but specified platforms, and disabling the whole device. Throughout this paper, the requirements for this software were gathered, prioritized, and documented. Also, a variety of models showing different aspects of the system, the implementation in the Java programming language, and the test cases were all discussed. Finally, trying to utilize this system outside of the Windows operating system would result in failure because it is specifically made for that platform.

**REFERENCES**