|  |  |  |
| --- | --- | --- |
|  |  |  |

Appointment scheduling Platform 2025

**Graduation Project Documentation**



**Appointment scheduling Platform**

**A Project Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Science in Systems and Computers Engineering**

**Submitted By**

**Eslam Samy Abdul-Qader 404019**

**Ahmad Sobhy Nassar 404011**

**Sayed Ahmad Mahmoud 404023**

**Eslam Mohammed Saleh 404021**

**Mohammed Mustafa Ramadan 404082**

**Supervised by**

**Dr. Abdulrahman Halawa**

**2025**

|  |  |  |
| --- | --- | --- |
|  |  |  |

Appointment scheduling Platform 2025

**Graduation Project Documentation**



**Appointment scheduling Platform**

**A Project Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Science in Systems and Computers Engineering**

**Submitted By**

**Eslam Samy Abdul-Qader 404019**

**Ahmad Sobhy Nassar 404011**

**Sayed Ahmad Mahmoud 404023**

**Eslam Mohammed Saleh 404021**

**Mohammed Mustafa Ramadan 404082**

**Supervised by**

**Dr. Abdulrahman Halawa**

**2025**

|  |  |  |
| --- | --- | --- |
| **Examiner Committee** | | |
| **Name** | **Rule** | **Signature** |
| Prof. Ashraf Maddcore | President |  |
| Dr. Abdulrahman Halawa | Supervisor |  |
| Dr. Mohammed Rayan | Member |  |

ABSTRACT

Time management and scheduling are critical challenges for professionals and service providers who rely on client appointments to deliver their services efficiently. This project, Timease, presents a full-stack scheduling and booking platform designed to streamline event management and appointment scheduling. The platform enables professionals—such as doctors, lawyers, and consultants—to create, update, and manage event listings, while allowing users to seamlessly book or cancel appointments. To ensure secure and scalable interaction, the backend is built using Java Spring Boot with RESTful API design and implements robust authentication via JWT tokens and refresh mechanisms. Role-based access control distinguishes between regular users and administrators, ensuring proper authorization for sensitive operations.

The frontend is developed as a mobile application using Flutter, ensuring a cross-platform and user-friendly experience. PostgreSQL serves as the primary relational database, integrated with the backend using Spring Data JPA and Hibernate for efficient data persistence. The entire system is containerized with Docker and deployed via Railway, enabling a streamlined DevOps pipeline and facilitating easy deployment and scalability. API endpoints are documented and exposed through Swagger UI for testing and development purposes.

This project demonstrates the practical integration of modern web technologies, cloud deployment practices, and secure API design to address real-world scheduling challenges. The results validate the system’s usability, functionality, and maintainability. Future extensions include the implementation of a notification system and enhanced input validation, which aim to improve user interaction and reliability. Overall, Timease contributes an effective and extensible solution for time management in professional settings.

**KEYWORDS** Scheduling; Spring Boot; Flutter; JWT; Docker;

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to **Dr. Abdulrahman Halawa**, our project supervisor, for his unwavering support, expert guidance, and thoughtful feedback throughout every stage of this graduation project. His mentorship played a critical role in helping us navigate technical challenges and refine our vision for Timease. His dedication and encouragement greatly contributed to the successful completion of our work.

We are also sincerely thankful to **Prof. Ashraf Maddcore**, President of the Graduation Committee, for his valuable insights and for fostering a rigorous academic environment. Our appreciation extends to **Dr. Mohammed Rayan**, Committee Member, whose observations and suggestions enriched the development and final presentation of our project.

  We gratefully acknowledge the technical and academic resources provided by our institution, which enabled us to develop and test our platform effectively. While we did not receive external financial assistance, the availability of institutional support and infrastructure was vital to our progress.

  We would also like to recognize the informal contributions of colleagues and peers who assisted us during development—particularly in testing, debugging, and providing feedback. Lastly, we extend heartfelt thanks to our families and friends for their continuous encouragement and support throughout this journey.

TABLE OF CONTENTS

ABSTRACT…………………………………………….………. Iv

ACKNOWLEDGEMENTS …………………………….……….v

**TABLE OF CONTENTS ……………………………….……….vi**

**CHAPTER 1 INTRODUCTION ………………….………… 1**

1.1 [Background](#_bookmark2)

1.2 [Motivation](#_bookmark3)

1.3 [Problem description](#_bookmark4)

1.4 [Aims and Objectives](#_TOC_250024)

1.5 [Methodology](#_bookmark5)

**CHAPTER 2** [**LITERATURE REVIEW.**](#_TOC_250023) **………….………… 1**

2.1 introduction

2.2 previous work

**CHAPTER 3 SYSTEM DESIGN .………….……….……… 1**

**CHAPTER 4 IMPLEMENTATION ……..………….………… 1**

**CHAPTER 5 CONCLUSION AND FUTURE WORK .……..…… 1**

**CHAPTER 1**

**INTRODUCTION**

The aim of the work described in this document is to develop a full-stack software system that simplifies and streamlines the scheduling process between service providers and their clients. The result of this effort is Timease - a name that reflects the platform’s core purpose: making time management easy. It is a combination of the words time and ease, representing our goal of easing the process of booking, managing, and attending appointments through a unified, user-friendly solution.

  Professionals across various industries—such as healthcare, legal consultation, and freelance services—depend on precise and reliable scheduling systems to manage their availability and client interactions. However, existing solutions often fall short in terms of flexibility, role-specific control, and ease of integration, especially for professionals seeking simple, mobile-first solutions. Timease addresses these limitations by offering a robust appointment scheduling platform that supports both one-on-one sessions and group events, built around a mobile-friendly architecture with role-based access, secure authentication, and effortless deployment.

  Our solution is designed as a monolithic Spring Boot backend connected to a Flutter-based mobile frontend, supported by a PostgreSQL relational database. The application is containerized with Docker and deployed via Railway, providing a production-ready architecture that is easy to replicate and scale. The system also supports secure user authentication through JWT and refresh tokens, along with administrative tools for event creation and moderation.

  This document outlines the rationale behind the development of Timease, discusses its implementation, and evaluates its potential to serve as a lightweight yet powerful scheduling platform. The sections that follow provide background context, articulate our motivations and objectives, define the specific problem we addressed, and detail the methodology followed throughout the project lifecycle.

**1.1** [**Background**](#_bookmark2)

Effective time management is a critical component of productivity, particularly for professionals whose work depends on client appointments, meetings, or service sessions. In fields such as medicine, law, consulting, education, and freelance services, maintaining an organized schedule is essential not only for operational efficiency but also for client satisfaction. With the increasing reliance on digital solutions, scheduling platforms have become integral tools for managing such time-dependent interactions.

While several commercial tools exist—such as Calendly, Doodle, and Google Calendar—they are often constrained by rigid configurations, limited role-specific customization, or require integration with external ecosystems that may not align with every user's needs. Additionally, many platforms impose paywalls for essential features, or lack support for easy deployment by independent professionals or small organizations without dedicated IT infrastructure.

  Recognizing these challenges, we set out to design and implement *Timease*, a comprehensive and accessible scheduling system that can be easily adopted by professionals without technical overhead. By combining a mobile-first user interface with a flexible and secure backend, *Timease* supports a range of appointment types—including both individual and group events—while ensuring that administrative control, user management, and session security are handled efficiently.

  Furthermore, we observed that many existing systems are fragmented across multiple services or rely heavily on cloud-based automation that is not always transparent or controllable by end users. With *Timease*, we aimed to deliver a system where the entire stack is clear, contained, and customizable—giving full ownership to the service provider or organization that deploys it.

  The development of *Timease* was also motivated by a desire to produce a system that could serve as a complete learning experience in full-stack development, deployment, and secure application design, reflecting real-world software engineering practices.

### 1.2 Motivation

Our motivation for developing Timease stemmed from a combination of technical curiosity, real-world demand, and a desire to create a practical and scalable solution to a common problem. As students of software engineering preparing for professional careers, we sought to build a system that not only demonstrated our full-stack development capabilities, but also delivered meaningful value to users beyond academic boundaries.

Throughout our academic and personal experiences, we noticed that professionals often rely on ad hoc or overly complex solutions to manage their appointments. Many are forced to use generalized tools that lack critical features such as role differentiation, mobile accessibility, or secure user authentication. This observation highlighted a gap in the availability of simple, customizable platforms that cater to small-scale service providers—individuals or teams who require robust functionality without the overhead of enterprise systems.

We were particularly motivated to build a mobile-first system because of the increasing dependence on smartphones for daily management tasks. A mobile-friendly design ensures that users—both professionals and clients—can interact with the platform anytime and anywhere, increasing the practicality and relevance of the system. In addition, incorporating role-based access control, session management via JWT, and Dockerized deployment allowed us to explore real-world technologies used in modern software systems.

By creating Timease, we aimed to produce a solution that is both educational and applicable, demonstrating how thoughtful design and strategic technology choices can solve common scheduling challenges. The project also allowed us to apply software engineering principles in a structured and meaningful way, with the potential for further development, real-world use, and academic presentation.

### 1.3 Problem Description

Scheduling and time coordination are persistent challenges for professionals and service providers who rely on client appointments to deliver their services effectively. In many cases, these professionals lack access to tailored tools that allow them to manage bookings, prevent overlaps, and adapt to both structured and dynamic availability. While commercial scheduling platforms exist, they often fail to meet the specific needs of individual practitioners or small teams due to limited conformability, overcomplicated interfaces, or high subscription costs for essential features.

The core problem lies in the absence of a lightweight, flexible, and mobile-oriented scheduling system that offers **secure role-based access**, **event customization**, and **client-friendly booking interfaces**, all in one integrated platform. Professionals may require different levels of control—such as the ability to create, modify, and monitor appointments—while clients should be able to book or cancel meetings easily without compromising security or performance. Existing solutions rarely provide such role-aware flexibility while also supporting both **individual and group meeting structures**.

Moreover, deployment complexity and platform dependence can act as barriers to adoption. Many tools assume continuous internet connectivity, cloud platform integration, or third-party plugin reliance, which may not be suitable for all contexts. For developers or teams seeking to own and operate their own scheduling system, these dependencies introduce unnecessary friction.

Timease addresses these issues by providing a full-stack, self-contained scheduling and booking platform that is mobile-first, easy to deploy, and secure by design. By simplifying both the user experience and the underlying technical infrastructure, we aim to close the gap between professional needs and accessible technology. The problem we set out to solve, therefore, is the **lack of a complete, customizable, and user-friendly solution for managing time-dependent professional services**.

### 1.4 Aims and Objectives

The primary aim of this project is to design and develop a fully functional, secure, and user-friendly scheduling platform—Timease—that enables professionals to create and manage appointments while allowing clients to book, cancel, or view events with ease. The system is intended to simplify time management processes by offering a mobile-first experience backed by a reliable backend infrastructure that can be easily deployed and maintained.

To achieve this aim, we established a set of specific objectives that guided the development of Timease throughout its lifecycle:

• **Develop a responsive mobile application** using Flutter that enables end users to browse available events and book appointments efficiently.

• **Design and implement a RESTful backend API** using Spring Boot that supports secure user authentication, event and user management, and booking functionalities.

• **Integrate role-based access control** to distinguish between general users and administrators, ensuring appropriate permissions for each action within the system.

• **Implement secure authentication using JWT** and refresh tokens to maintain session integrity and prevent unauthorized access.

• **Support both one-on-one and group event scheduling**, giving administrators the flexibility to define different event types and capacities.

• **Enable users to view, book, and cancel appointments**, with backend logic to ensure data consistency and event availability.

• **Containerize the backend application using Docker** to ensure reproducibility, ease of deployment, and environment consistency.

• **Deploy the platform on Railway**, leveraging a cloud-based environment for hosting the backend and database in production settings.

• **Follow modern software engineering practices**, including version control with GitHub, modular code design, and clear documentation.

Together, these objectives ensure that Timease is not only a technically robust application, but also a practical tool for real-world use, aligning with our academic goals and professional aspirations.

**1.5** [**Methodology**](#_bookmark5)

The development of Timease followed a structured, iterative methodology inspired by agile software development principles. Our goal was to ensure continuous progress through frequent testing, clear task segmentation, and close coordination across all stages of the project—from requirements gathering to deployment. The methodology emphasized adaptability, allowing us to incorporate feedback and adjust features based on evolving priorities and technical constraints.

We began by defining the system requirements and drafting a set of user stories to identify the core functionalities needed for both administrators and end users. These included booking workflows, authentication mechanisms, event creation, and role-based access control. The system architecture was then planned with a clear separation of concerns between the frontend and backend, both of which were developed in parallel to ensure seamless integration.

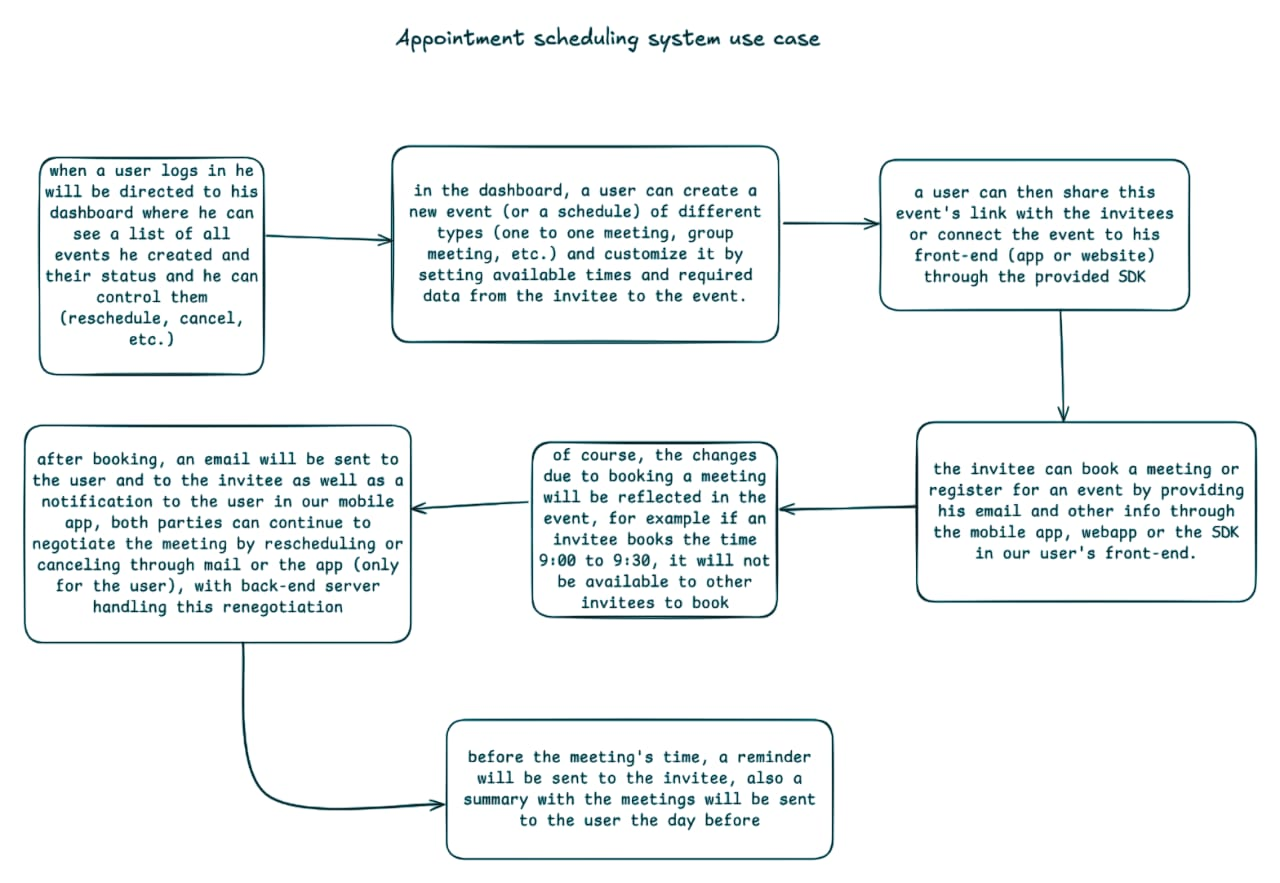
Figures 1-1 and 1-2 show early design dreafts and user stories from the design phase

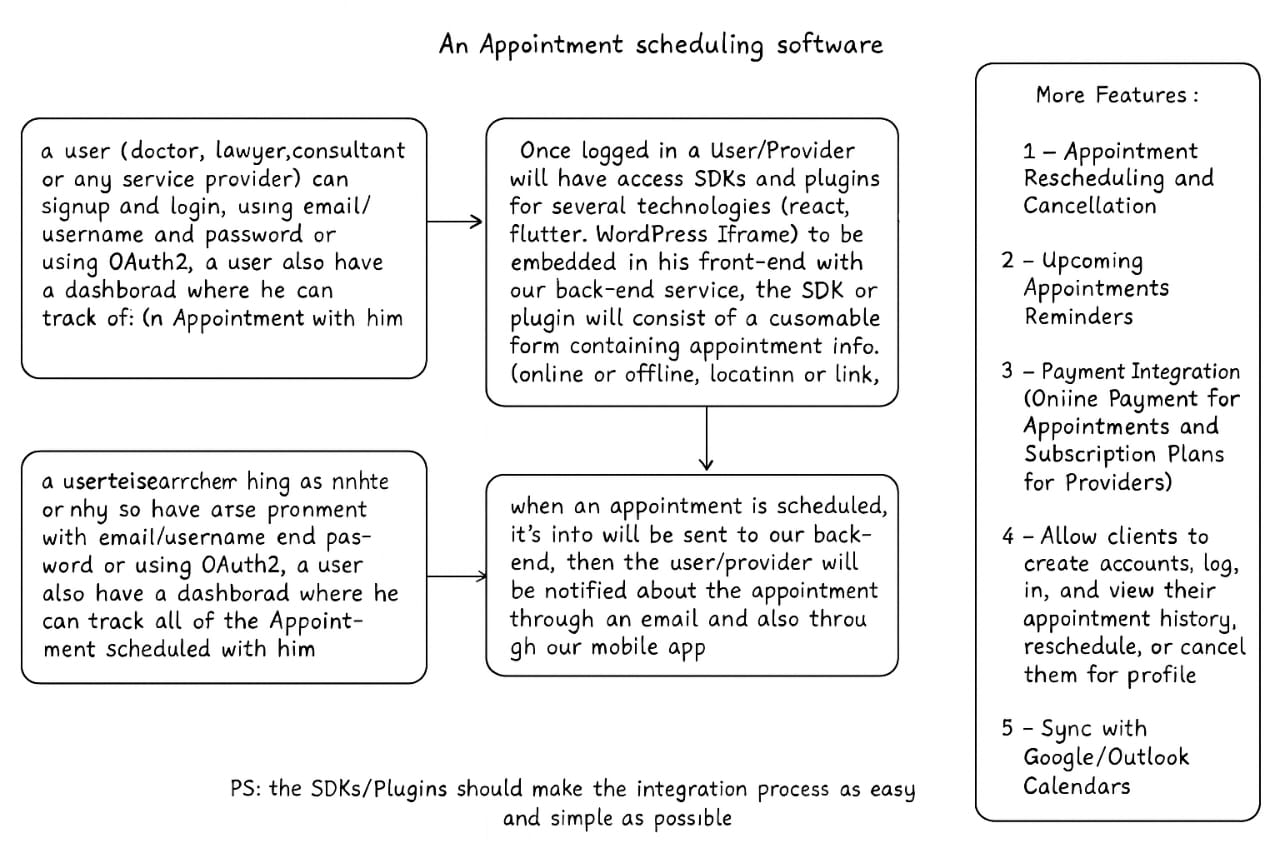
On the backend, we used Spring Boot to implement a RESTful API that handles all business logic and communicates with a PostgreSQL database using Spring Data JPA and Hibernate. Secure authentication was implemented using JWT tokens and refresh tokens, while access rights were enforced through role-based authorization. The backend was fully containerized using Docker, and a Docker Compose file was created to manage service dependencies for local development and production.

The frontend was built using Flutter, chosen for its cross-platform capabilities and responsive design features. The mobile application connects to the backend API and provides interfaces for booking, cancellation, and event browsing. We conducted regular testing on both the UI and backend services to ensure functional correctness, security, and performance.

Finally, we deployed the backend and database on Railway, a cloud platform that enabled fast and reliable deployment. Version control and collaboration were maintained through GitHub, allowing for systematic tracking of changes and contribution management.

This methodology ensured that Timease was developed with clear milestones, technical rigor, and a focus on usability—resulting in a robust and deployable scheduling platform.

Figure 1-1

Figure 1-2

**CHAPTER 2**

[**LITERATURE REVIEW**](#_TOC_250023)

## Introduction

This chapter introduces the necessary background material related to the underlying project. It is often appropriate to provide more information than was given in your Introduction. Try to limit yourself just to what the reader needs to know to understand the solution that you havedeveloped in your project. Put your work in the context of related existingwork, commercial products, and research papers (if relevant).

## Graph Colouring

## Meta-Heuristic Methods

Sub-Sub title font is time new roman with 12pt and not bold.

A. Local Search Methods

You can extend you numbering by using A and leave tab with 2 spaces “Before text indentation = 0.13"”, then Sub A “A.I” and leave tab with 7 spaces “Before text indentation = 0.44"”, then Sub-Sub A “A.I.1” and leave tab with 15 spaces “Before text indentation = 0.94"”.

A.I Hill Climbing

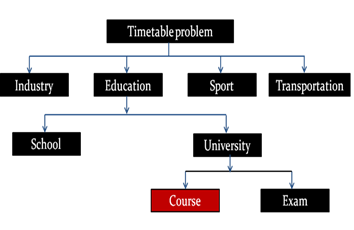
Sub A “A.I” and leave tab with 4 spaces.

B. Population Based Algorithms

B.I Evolutionary Algorithms

B.I.1 Harmony Search Algorithm

Figure can be cited inside the text by using “Figure”, for example, figure 1 shows the classification of the timetable problems.



**Figure: bold with capitalizing the first letter.**

**First number: Chapter number**

**Second number: figure number within the chapter starting from 1 then followed by a colon (:)**

**Figure 2.1:** Timetable scheduling problems classification.

**Figure caption: under the figure.**

**Font: 11pt**

**Style: Time New Roman**

Figure 1 shows that before and after spacing should be equal to 12pt. The figure caption is under the figure. Figures are center justification. If you start a paragraph with the word “Figure” then we use capital F, otherwise small f.

Tables are similar to the figures, but the difference that the table caption is above the table, for example, Table 1 shows the physical annealing converting to simulated annealing. In addition that, the column header are bold.

**Table caption: Above the table.**

**Font: 11pt**

**Style: Time New Roman**

**Table 2.1:** The physical annealing converting to simulated annealing (Cerny, 1985).

|  |  |
| --- | --- |
| **Thermodynamic Simulation** | **Combinatorial Optimization** |
| System States  **Table: bold with capitalizing the first letter.**  **First number: Chapter number**  **Second number: table number within the chapter starting from 1 then followed by a colon (:)** | Feasible Solutions |
| Energy | Cost |
| Change of State | Neighbouring Solutions |
| Temperature | Control Parameter |
| Frozen State | Heuristic Solution |

References are cited inside the text between “(” and ”)”. We Use APA style, see the examples in the references section.

## Methodology

Analysis and Specification: How you analyzed the problem, including user requirements. Give an appropriate specification of the solution. This is done inGP1.For example, we can include, the method used, functional requirements, non-functional requirements, and security requirements.

**CHAPTER 3**

**SYSTEM DESIGN**

## Introduction

Design: if it is a software development project then give a high-level account of the structure of your software and how it works. What algorithms does it use? How do these compare with alternatives? What were the main design decisions you took, and their justifications? This is done in GP1.

Implementation and testing: a detailed account of the implementation and testing of your software. Explain the conceptual structure of the algorithms. Also explain what data structures you used, and how the algorithms were implemented. What implementation decisions did you take, and why? There is no need to list every little function and procedure and explain its working in elaborate detail; use your judgment on what is appropriate to include. This is done in GP2.

**CHAPTER 4**

**RESULTS AND DISCUSSION**

## Results

Results: you should assess the success of your project. How does it compare with the original specification? How reliable is it? How have you tested it? Comment on its robustness. This is done in the final documentation of the project.

## Discussion

Discussion: here you will summarize your achievements and also the deficiencies of your project. You can also say what you would or could have done, if you had had more time or if things had worked out differently. It is important to be completely honest about the deficiencies and inadequacies of your work, such as they are. Part of your aim is to demonstrate your ability to recognize problems that remain. This is done in GP2.

**CHAPTER 5**

**CONCLUSION AND FUTURE WORK**

## Conclusion

Give a brief statement of how the solution that you have provided addresses the problem stated in the introduction. Provide an evaluative statement based on the results. You should not introduce new material.

## Future Work

Future Work: here you will recommend what is needed to be done in the future to your work. This is done in GP2.

**REFERENCES**

Cerny, V. 1985.Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, *J. Opt. Theory Appl.*, volume 45, number1, pages.41-51.

References: for your Final Year project, it is required that you cite and reference work to which you owe an intellectual debt. It is required that you cite and reference work that provides supporting evidence. It is required that you cite and reference work so that the reader can find the sources that have been quoted.

In other words, the purposes of a reference are to acknowledge the contributions of other authors and to enable readers to locate source easily. In this section, you can use alphabetically or numerically system (e.g. APA style referencing).

**General guidelines:**

**APA Reference List Examples**

* **Book with Single Author:**

Gore, A. (2006). *An inconvenient truth: The planetary emergency of global warming and what we can do about it.* Emmaus, PA: Rodale.

**In-text reference:** (Gore, 2006)

* **Book with Two Authors:**

Michaels, P. J., & Balling, R. C., Jr. (2000).*The satanic gases: Clearing the air about global warming*. Washington, DC: Cato Institute.

**In-text reference:** (Michaels & Balling, 2000)

* **Book with Editor as Author:**

Galley. K. E. (Ed.). (2004). *Global climate change and wildlife in North America.*Bethesda, MD: Wildlife Society.

**In-text reference:** (Galley, 2004)

* **An Anonymous Book:**

*Environmental resource handbook*. (2001). Millerton, NY: Grey House.

**In-text reference:** (Environmental Resource Handbook, 2001)

* **Articles in Reference Books (unsigned and signed):**

Greenhouse effect.(2005). *American heritage science dictionary*. Boston, MA: Houghton Mifflin.

Schneider, S. H. (2000). Greenhouse effect.*World book encyclopedia* (Millennium ed. Vol. 8, pp. 382-383). Chicago, IL: World Book.

**In-text references:** (Greenhouse effect, 2005)

(Schneider, 2000)

* **Journal Article when each issue begins with p.1:**

Bogdonoff, S., & Rubin, J. (2007). The regional greenhouse gas initiative: Taking action in Maine. *Environment, 49*(2), 9-16.

**In-text reference:** (Bogdonoff& Rubin, 2007)

* **Website:**

United States Environmental Protection Agency. (2007, May 4). *Climate Change*. Retrieved From the Environmental Protection Agency website: http://www.epa.gov/climatechange

**In-text reference:** (United States Environmental, 2007)

Gelspan, R. (2007). *The Heat Is Online*. Lake Oswego, OR: Green House Network. Retrieved from The Heat Is Online website: http://www.heatisonline.org

In-text reference: (Gelspan, 2007)

**How to Cite an Website in APA**

**Structure:** Last, F. M. (Year, Month Date Published). Article title.*Website Title*.Retrieved Month Date, Year, from URL.

**Example:**

* Satalkar, B. (2010, July 15). Water aerobics. *Buzzle*.com. Retrieved July 16, 2010, from http://www.buzzle.com.
* Cain, K. (2012, June 29). The Negative Effects of Facebook on Communication. *Social Media Today RSS*. Retrieved January 3, 2013, from [http://socialmediatoday.com](http://socialmediatoday.com/).

OR numerically as below

All reference items must be in 10pt font. Please use Regular and Italic styles to distinguish different fields. Number the reference items consecutively in square brackets (e.g. [1]).

When referring to a reference item, please simply use the reference number, as in [2]. Do not use “Ref. [3]” or “Reference [3]” except at the beginning of a sentence, e.g. “Reference [3] shows …”. Multiple references are numbered with one bracket and separated with comas (e.g. [2], [2, 3], [4 – 6]).

Structure: First Author Last Name, First Author First Name.

[1]**Author. (Date published if available; n.d.--no date-- if not). Title of article/book. *Title of web site* . Retrieved date. From URL.**

[2]Author. (Date published if available; n.d.--no date-- if not). Title of article/book. Title of web site . Retrieved date. From URL.

**APPENDICES**

**Appendix A:**

Appendices: the Report must contain an appendix explaining file structure on thedata CD submitted with it. The appendix must also contain information on how thecode should be run. Other appendices may include documents such as: the projectproposal; a selection of experimental data; schedules; testing strategy; riskmanagement plans; glossary; manual; etc. Don't include the source code as anappendix (submit it on CD; see below). Don't include voluminous appendices (theseshould also be submitted on a CD). A report template can be found in the collegewebsite.

**Timeline/Milestones (Gantt Chart)**

This course is similar to self-study/research. Weekly meetings are scheduled with the supervisor for the project. Each student’s group will meet together weekly, keeping detailed minutes of the meetings.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task |  | Weeks | | | | | | | | |
|  |  | 1-2 | 3-4 | 5-6 | 7-8 | 9-10 | | 11-12 | 13-14 | 15 |
|  |  |  |  |  |  |  | |  |  |  |
| 1. …….. |  |  |  |  |  |  | |  |  |  |
| 2. …….. |  |  |  |  |  |  | |  |  |  |
| 3. …….. |  |  |  |  | |  |  | |  |  |
| 4. …….. |  |  |  |  |  |  | | |  |  |
| 5. …….. |  |  |  |  |  |  | |  |  |  |
| ……….. |  |  |  |  |  |  | |  |  | |