Department of Software Engineering
Lakehead University

ESOF 4969 – DEGREE PROJECT 2022-23

Progress Report 2 - Software Project Management Plan

Group 3

Contributing Members:
Nicholas Imperius
Jimmy Tsang
Mason Tommasini

Date Submitted: 11/30/22

Approval

This document has been read and approved by the following team members responsible for its implementation:

PRINT NAME: Nicholas Imperius
Signature:

Nicholas Imperius

PRINT NAME: Jimmy Tsang
Signature:

Jimmy Tsang

PRINT NAME: Mason Tommasini
Signature:

Mason Tommasini

COMMENTS:
Individual Contributions are as follows:
Nicholas Imperius: Abstract, Introduction, Business Model, Software Process Methodology, Market Cost Analysis, Risk Prevention, Conclusion
Jimmy Tsang: Software Process Methodology, Project Roles and Task Breakdown, Project Development Cost and Duration Estimates, Market Cost Analysis
Mason Tommasini: Introduction, Project Deliverables and Milestones, Market Cost Analysis, Risk Prevention, Conclusion

Smart Parking Application: Software Project Management Plan

Nicholas Imperius Jimmy Tsang Mason Tommasini

Abstract

This paper describes the design of a software application to help users find a parking spot in a city. The application will use machine learning techniques to determine whether you have parked your car or not, and then use GPS location information to notify other users that the parking space has been filled. This application would provide further benefits, such as saving the user time, reducing unnecessary stress, and helping with the environmental impact of vehicles. The waterfall methodology will be used to provide a concrete and robust model to follow throughout the development of the project. In addition, a description of each member's role will be explained and the deliverables that are required throughout this project are shown. Risks associated with the database, social acceptance, expertise, and more. Finally, a market cost analysis is conducted to evaluate competing applications available in the Google Play Store and the Apple App Store to identify if most apps are free to download and have any in-app purchases. We have used the Intermediate Constructive Cost Model (COCOMO) technique to estimate the total production cost and development time of this project. The final cost estimate was calculated to be $47,766.40. We also discussed marketing the software as either a product or as a service. Additionally, a breakdown of possible revenue and gross margins is described.

Index Terms— Parking application, machine learning, mobile application, COCOMO.

Contents

1 Introduction

Many students and staff can attest to parking problems at Lakehead University. To provide a simple, fast and effective solution to this problem, we plan to create a software application that makes finding a parking spot much easier. The software would be implemented as an application for smartphones that displays information, such as free parking spaces, busy times, and the location of your parked car. When arriving at the university, it can be difficult to find an open parking spot and frustrating when you are running late. This application would provide a solution to this issue and provide additional benefits, such as saving the user time, reducing unnecessary stress, and helping to reduce the environmental impact of vehicles. Our initial testing and training of this project will be focused on the G campus parking lot at Lakehead University, but could be further expanded to other fields, such as public parking structures and mall parking lots.

This application will be quite different from other parking applications that exist currently. Instead of relying on sensory data or IP cameras to locate and detect empty parking locations, this application will use machine learning techniques to determine whether you have parked your car or not and then use GPS location information to notify other users that the parking space has been filled. Additionally, when you have left your parking spot, our model will be able to accurately detect that and mark the parking space as free. Different machine learning classification methods will need to be tested to ensure that we have chosen the most appropriate model. The machine learning model will predict based on your speed before and after a full stop; it should be able to recognize the difference between walking and driving, which allows it to determine the stopping location.

There exist a good deal of different software development methodologies, such as agile development or the waterfall model. Every project has a different development method that suits it best. In this project, you will find that the waterfall method suits this project the most, since we have ample time to proceed through the different phases of the method. The waterfall method allows us to apply a cost estimation and duration technique known as the Intermediate Constructive Cost Model (COCOMO). This was a technique that was first developed by Barry Boehm [1]. This analysis technique involves first computing the estimated development time, which will then be used to estimate the development cost, known as the nominal effort. In order to calculate the estimated duration of development, we must first determine two values, the length of the software in kilo-lines of code (KLOC) or kilo-delivered source instructions (KDSI), alongside the development mode, defined as the intrinsic measure of complexity. Using these values, we can then estimate the total production cost and development time by applying these values through a series of modifiers, which are known as cost drivers.

Risk analysis is an important factor to consider when designing the outline of a project. This involves understanding how different deliverables and milestones might be affected due to an event happening. For instance, falling behind schedule on a key component of the project that is required for further components. Assessing these risks is critical to determining the uncertainty related to specific components.

The rest of this paper is organized as follows: Section 1 provides an initial introduction to the project. Section 2 discusses the technology and environment that the application will be used on, and the cost and time of the development. Section 3 will describe the methodology behind our development process. Section 4 describes the different roles and tasks that will need to be conducted through this project. Section 5 will outline the numerous deliverables throughout the project along with the specific milestones that will need to be met. Section 6 will show the computed costs associated with the project, as well as other estimates. Section 7 outlines the competing products already available for download and breaks down possible revenue generation from this project. Section 8 will describe the associated risks with the project and key ideas to consider. Finally, Section 9 will provide a summary of this software project management plan.

2 Business Model

2.1 Description of Technology & Environment

This project will consist of a mobile application that will be used on a smartphone that users will be able to download from their respective app stores. The application will utilize the user's GPS location and other sensors for a precise reading. We plan to develop our mobile application with Visual Studio (VS) Code using the React Native language. Supplementary GPS data will be provided by Google Maps Application Program Interfaces (API's), and a cloud server through Firebase will need to be established. In the application, a bird's-eye view of the parking lot will be visible, so users can get a better sense of the parking they have available to them. A mobile internet connection will be required to properly communicate with the server to allow real-time information on parking availability.

2.2 Cost and Time of Development

As of right now, we do not predict the development of our prototype will have a high cost. There are three main software components required for our project:

- Mobile Application IDE – Visual Studio Code
- Google Maps API
- Backend Infrastructure – Firebase with NoSQL

For our prototype, there will be some costs incurred. To publish our application, React Native requires a fee of $100 in order to place the application in the respective Application Stores. For commercial use, more requests are required to access Google Maps APIs and the Firebase cloud server, which should total around $50 per month for each application.

Lastly, for the SQL database, a license may be required, which can cost up to $1000. A detailed breakdown of all pricing is found in Table 1. However, as of October 2022, no costs have been incurred.

Table 1: Cost Breakdown

Software	Prototype Cost
React Native	$100 for iOS App Store* $50 for Google Play Store*
Google Maps APIs	Starts at $65 per month*
Firebase Backend Infrastructure	Starts free, increases with usage per month*

* Actual prototype cost for this project is $0

3 Software Process Methodology

We decided to use the waterfall model, as shown in Fig. 1, a linear-sequential life cycle model because of the ample amount of time provided to develop our software. For a quick development time in order to compete with market demand, a complex model such as the agile software development methodology would be ideal, as it is a way to complete a working prototype in minimal time, but our project has no such constraints. This allows us to work at a suitable pace and ensure that everything is completed before moving on to the next phase, which makes the waterfall method a suitable choice for this project.

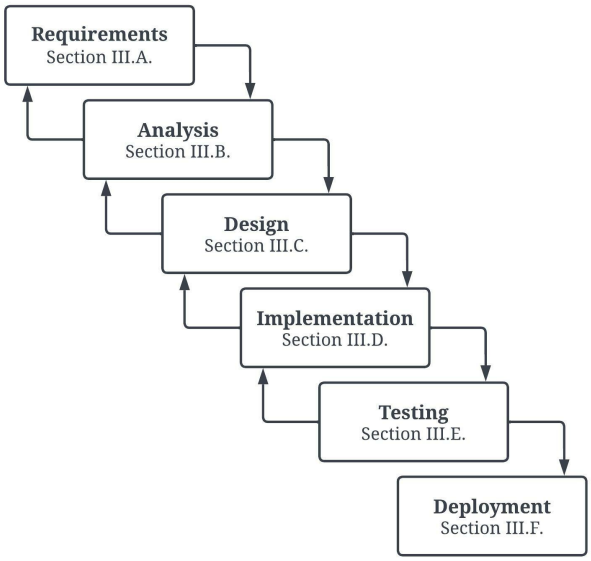

Figure 1: The waterfall software development life cycle.

The waterfall method is an older method, designed and first used in 1970 by Winston Royce; however, it is still widely used today [1]. It consists of six major phases in the following order: requirements, analysis, design, implementation, testing, and deployment phase. In this model, until the completion of the current phase, we cannot move on to the next phase, ensuring that we have sufficient information and documentation before proceeding. Although, if at any point we realize there is an issue, we can move back to the previous phase, reassess, and fix the issue before moving on with the project again. As shown in Fig. 1, each phase of the waterfall method is described as follows:

3.1 Requirements Phase

In the requirements phase, we explore and refine what the project requirements are, the deadlines, and any restrictions that must be included. We gain an understanding of the project domain as well as both functional and non-functional requirements. Preparations are made as to what software or hardware will be required to complete the project. At the end of this phase, a clear outline of the project and a breakdown of task delegation are essential for the smooth execution of the subsequent phases.

3.2 Analysis Phase

In the analysis phase, we analyze the specifications to generate models and determine what functionality each part of the program should have. A breakdown of each component within the project as well as their respective subcomponents will be outlined in order to achieve each requirement from the previous phase. These subcomponents will be thoroughly examined and a process for how they will be completed will be created for smooth execution.

3.3 Design Phase

The design phase is where the full specification document will be created for the design of the product and all technical components will be provided in full detail, such as the required programming language and hardware. A solution for each subcomponent is designed; this includes system architectures, interfaces, libraries, and more. In this project, the design phase will include a detailed document of wireframe figures for the mobile user interfaces, as well as required functionality and flow for all functioning navigation or clickable areas within the app. Furthermore, the design of the database will be outlined and followed in the coming phases.

3.4 Implementation Phase

Once the design document is created, we can start the implementation phase, where everything outlined will be developed and integrated together in order to create a working prototype of the product. Different components will be created for each aspect of the application and will need to be integrated together. Everything from the design phase will be converted into code and properly implemented into modules. These modules will be tested in the following phase, but by the end of this phase, the idea is to have an alpha version of the final mobile application that would be ready for testing for integration issues, bugs, and more.

3.5 Testing Phase

To ensure that every component works together well, testing will need to be conducted. The components of the system and the system as a whole will be tested to ensure that all functionality works as originally described. Also, coding errors will need to be found and fixed. If there are any errors discovered or updates are required, we will move back into the implementation phase to fix these issues and then proceed to the testing phase again. The analysis phase would have outlined any testing procedures to do that would ensure the functionality meets requirements, and these tests are conducted and verified. All requirements have to be met before we can proceed to the final phase.

3.6 Deployment Phase

Once everything has been fixed, and we are confident in the product, we can move onto the deployment phase, which, in our scenario, will be our final working prototype. The product will be implemented in working condition as if we were ready to publish to the respective app stores. If published, maintenance would be required on the application every so often to ensure the software remains bug-free and stable for users. In addition, more features and functionality could be developed and pushed as a software update to the application if there's a need for it.

4 Project Roles and Task Breakdown

Each member of the group will be delegated their own tasks to complete. They are responsible for the timely and correct completion of their tasks, along with sufficient documentation. However, other members may assist with the completion of tasks if necessary. Throughout the group project, there will be a group leader assigned, rotating throughout the development of the software. The group leader is responsible for planning and delegating tasks, motivating team members, and ensuring that members stay on track to complete their tasks within designated deadlines. For the first official group leader, we are selecting Jimmy Tsang, who will be the project leader from the beginning of October to the end of November. In addition to the aforementioned tasks, the group leader will also be responsible for the recording of minutes of team meetings, which will be compiled into a document with timestamps and meeting minutes. As we progress throughout the term, the group leader will rotate, with Mason Tommasini taking on the role of group leader from December to February, and Nicholas Imperius, who was the interim group leader in September, taking on the role of group leader once again for March and April.

For this project, there are four main areas of focus: mobile application, cloud server, database creation, and machine learning model development. Additionally, the required documentation will be completed equally between the members of the team. Contributions will be recorded in the comments section on the title page of each report. The tasks delegated to each of the members are provided in Table 2. However, each member is encouraged to contribute and assist with each part as they see fit.

5 Project Deliverables and Milestones

A Gantt chart was constructed, shown in Fig. 2, to display each milestone of the project development phases. The month of September will be spent creating the project proposal presentation as well as the software requirements document. October will also include the software requirements document and software project management plan which we are

Table 2: Software Task Delegation

Member	Task
Nicholas Imperius	Machine Learning Model
Jimmy Tsang	Mobile Application
Mason Tommasini	Cloud Server and Database

currently working on. November will encompass our first oral presentation, while December and January will be used to work on the software specifications document over the Christmas break. From December to April we will work on the software prototype as this will take up the majority of our time and effort. February through April will be used to work on the software design document, which is our final report to summarize our project. Finally, in April we will do our final oral presentation to showcase our final product.

	September 2022	October 2022	November 2022	December 2022	January 2023	February 2023	March 2023	April 2023
Project Proposal Presentation	Nicholas							
Software Requirements Document	Nicholas	Jimmy						
Software Project Management Plan		Jimmy						
First Oral Presentation			Jimmy					
Software Specifications Document				Mason	Mason			
Software Prototype				Mason	Mason	Mason	Nicholas	Nicholas
Software Design Document						Mason	Nicholas	Nicholas
Final Oral Presentation								Nicholas

Figure 2: Gantt chart of project deliverables and milestones along with the corresponding group leader.

6 Project Development Cost and Duration Estimates

6.1 Implementation of COCOMO

As explained in Section 1, the cost drivers' values are based on complexity and the justification of our selected values can be found in Table 4. To begin, we must determine the development mode, as these are the constants that allow us to determine the nominal effort of the project before we can begin to apply the cost drivers. For this project, we will be using KLOC as our measure for the length of software, as presented in [2]. Equation (1) is for calculating the nominal effort.

$$Nominal\ Effort = \alpha \times KLOC^{\beta} \tag{1}$$

Regarding the development mode, there are three possible options, organic, semi-detached, and embedded, ranging from simple to complex, respectively. For our software, we believe it falls into the semi-detached development mode. A project is considered semidetached if some members of the team have experience with certain aspects of the software being developed, while some do not. The team members may have limited experience in related areas, but could be unfamiliar with some aspects of the developed software. All members of the team have experience with machine learning models and databases, while some members have experience with cloud servers or mobile application development. Although we have applied these experiences to other areas of software development, working with GPS data and Google Maps APIs is completely new to all members and as such, extensive research must be performed. For a semi-detached system, the constants $\alpha = 3.0$ and $\beta = 1.12$, as found in Table 3, are used to calculate the base nominal effort.

From this, we must determine the number of lines of non-commented code in KLOC; therefore, we look at each individual component of the project and estimate how many lines of code will be needed based on best-effort estimation and relevant research. Based on personal experience of one of the group members, it can be said that a typical application

Table 3: Software Development Mode and Constants [1]

Mode	α	β
Organic	3.2	1.0
Semidetached	3.0	1.12
Embedded	2.8	1.20

prototype requires 1000-2000 lines of code to develop, we will use 1000 as our current estimate. Next, for our machine-learning model, from previous research, we have found that an average machine-learning model is around 200 lines of code. For our cloud server, according to Google's documentation on Firebase, an average real-time connection to a database can take less than 100 lines of code, so we will estimate around 100 lines of code. For our database, we estimate there to be about 5 tables in the database, with each table taking up to 10 lines of code or 50 in total. In addition, we need to perform different types of queries and connect the database with the cloud server, which, from experience, can be up to 200 lines of code. Moving on, in order to populate the database with information about each user and parking space, may require another 500 lines of code. Overall, the estimated number of lines of code for creating the entire database is 850 lines of code and the estimated total overall number of lines of code is 2,050. Lastly, as previously mentioned, this project is a semi-detached system, which means that $\alpha = 3.0$ and $\beta = 1.12$. Therefore, the nominal effort in person months for this project can be computed by applying $\alpha = 3.0$, $\beta = 1.12$, and $KLOC = 2.050$ to Equation (1) to get the nominal effort shown in Equation (2).

$$\begin{aligned} \textit{Nominal Effort} &= 3.0 \times 2.050^{1.12} \\ \textit{Nominal Effort} &= 6.703 \text{ person-months} \end{aligned} \tag{2}$$

From here we must then multiply the nominal effort by fifteen different software development multipliers, each having a different value depending on the intrinsic difficulty we believe this process will take, as shown in Table 4. The justification for each value selected for the cost drivers is found below and is represented with an underline in Table 4.

Table 4: Software Development Cost Drivers [1]

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Required Software Reliability	0.75	0.88	1.00	1.15	1.40	
Database Size		0.94	1.00	1.08	1.16	
Product Complexity	0.70	0.85	1.00	1.15	1.30	1.65
Execution Time Constraint			1.00	1.11	1.30	1.66
Main Storage Constraint			1.00	1.06	1.21	1.56
Virtual Machine Volatility		0.87	1.00	1.15	1.30	
Computer Turnaround Time		0.87	1.00	1.07	1.15	
Analyst Capabilities	1.46	1.19	1.00	0.86	0.71	
Applications Experience	1.29	1.13	1.00	0.91	0.82	
Programmer Capability	1.42	1.17	1.00	0.86	0.70	
Virtual Machine Experience	1.21	1.10	1.00	0.90		
Programming Language Experience	1.14	1.07	1.00	0.95		
Use of Modern Programming Practices	1.24	1.10	1.00	0.91	0.82	
Use of Software Tools	1.24	1.10	1.00	0.91	0.82	
Required Development Schedule	1.23	1.08	1.00	1.04	1.10	

Required Software Reliability – Very High = 1.15: This system must be very reliable, it must handle conflicts and system crashes from overload. It is the main interface for viewing available parking spaces and as such, must be active for the majority of the time. In addition, if the system crashed, the functionality of the application would disappear, rendering it useless; therefore, it needs to be developed reliably.

Database Size – High = 1.08: The end goal of this database is to keep track of every G lot parking space at Lakehead University, with a capacity of over 1200 parking spaces, the database must individually keep track of the availability of all the parking spots.

Product Complexity – Very High = 1.30: We feel as the project complexity is quite high, the application's general user interface will not contain code that interfaces with microprocessors, but the machine learning portion of the application will be more complex, hence the very high rating.

Execution Time Constraint – Nominal = 1.00: Very minimal in current technology compared to the technology of 1984 when Intermediate COCOMO was created. With technological advancements since then, we can ignore the effect

of this cost driver. Thus, we have chosen the smallest effort multiplier.

Main Storage Constraint – Nominal = 1.00: Storage is a very small item to worry about with how inexpensive large storage is today; therefore, we can ignore the effect of this cost driver and have selected the smallest effort multiplier.

Virtual Machine Volatility – Low = 0.87: The underlying volatility of the virtual machine for this software product can be considered to be quite low. We have selected the smallest effort multiplier.

Turnaround Time – Low = 0.87: Since this project has a small team that is working closely and our scope is well-defined, we can consider the turnaround time to be quite small since we are not dealing with separate teams within an organization. Thus, we have selected the smallest effort multiplier.

Analyst Capabilities – Low = 1.19: We have limited experience in this space aside from a few projects in courses. We have learned most of the required information before, but have yet to create an entire system like this before.

Applications Experience – Nominal = 1.00: Application experience is nominal as we have some experience in some areas needed for the development of this prototype, however, we still lack knowledge in some areas due to limited experience.

Programmer Capability – Nominal = 1.00: Since some group members have been programming for 6-10 years, we are confident in our abilities to quickly learn, adjust and present a working prototype.

Virtual Machine Experience – Low = 1.10: With only about 1 year of experience in the fields required for this project, we feel that we also lack experience here.

Programming Language Experience – Nominal = 1.00: Some members of the group have experience with NoSQL ($\sim$1 year), Python ($\sim$6-8 years), Cloud Servers ($<$1 year), and mobile application development ($<$1 year) either from school courses or freelance learning.

Use of Modern Programming Practices – Low = 1.10: Due to a lack of experience, most of what we know about modern programming practices has been from in-class lectures rather than on-site practices.

Use of Software Tools – Low = 1.10: We have a basic understanding of software tools as learned from school courses due to a lack of experience with on-site applications.

Required Development Schedule – Nominal = 1.00: The development schedule is carried out throughout the school year. As of right now, we have around 6 months to complete the development of our prototype.

6.2 Final Cost Driver Calculation

Now that the cost drivers have been determined, we then find the multiplier by multiplying all the cost drivers together and getting the resulting value in Equation (3).

$$\textit{Cost Multiplier} = 1.936 \tag{3}$$

Using this multiplier, we can calculate the adjusted nominal effort, found in Equation (4).

$$\begin{aligned} \textit{Adjusted Nominal Effort} &= 6.703 \times 1.936 \text{ person-months} \\ \textit{Adjusted Nominal Effort} &= 12.98 \text{ person-months} \end{aligned} \tag{4}$$

From our calculation, we have determined an estimated duration of 12.98 person-months for development, or 4.326 person-months for every person. From this, we can then calculate the estimated total cost of production. The average software engineering intern earns about \$23 an hour.[1] Assuming a 40-hour work week, the monthly salary would be around \$3,680, or a yearly salary around \$44,000. Now that we have a base rate, we can estimate the cost of development, as seen in Equation (5).

$$\begin{aligned} \textit{Total Cost} &= 12.98 \text{ months} \times \$3680 \text{ per month} \\ \textit{Total Cost} &= \$47,766.40 \end{aligned} \tag{5}$$

Based on the Intermediate COCOMO method, we estimate that this project will take 12.98 person-months and have an estimated cost of \$47,766.40, computed in October 2022. However, since the number of lines of code needed was estimated, this can vary greatly during the actual development phase. Additionally, with mobile application development, there can be quite a bit of pre-generated or built-in code, which would take away from the actual number of lines of code. Until development begins, it is quite hard to develop a good estimate of the time and cost of development. As we progress throughout the development period, changes to the cost and duration estimate will be adjusted accordingly.

6.3 Summary of Calculations

Table 5 presents a concise summary of the findings found throughout Sections 6.1 and 6.2. These values were calculated as of October 31, 2022. As we progress throughout the development term, we will re-evaluate the decisions of our cost drivers and update them as needed.

[1] https://www.payscale.com/research/CA/Job=Software_Engineering_Intern/Hourly_Rate

Table 5: Summary of Cost Estimations

Total Cost	$47,766.40
Cost Multiplier	1.936
Nominal Effort	12.98 person-months
KLOC	2.050

7 Market Cost Analysis

7.1 Competing Products

By accurately identifying competing products in the same market as our software, we can gauge various different metrics that may also apply to our application. Due to us creating a mobile application, our main competitors would be similar apps on the Google Play Store for Androids as well as the Apple Store for Apple products.

When entering the keywords "parking application" into the Google Play Store and the Apple Store, we are greeted with various different applications to help users with their parking needs, although no application had the same key points as ours. Most of the applications were found to help users find different parking locations throughout cities, while showing parking fees, time, and more. The difference is our application shows the vacancy of a parking lot to show the user how busy the lot is. By viewing an application called "Parkopedia Parking" on October 31st, 2022 we can see that the application has features such as parking lot prices, availability, and distance to the user. This application has more than 1 million downloads and a rating of 4.6 stars. When looking at the majority of applications on the Google Play Store, we found that most applications are free and have no additional in-app purchases for more in-app features. This is a key indicator, and our application will most likely be free to first establish our initial user base, before considering more features that could be unlocked via payment methods.

7.2 Revenue Calculation

According to Software Equity[2], a good benchmark for a software as a product (SaaP) development firm is a gross margin over 75%. Based on our initial estimate of the cost of project development, we would need to market the product for approximately $83,591.20 in order to generate a 75% gross margin or profit of $35,824.80. This is done by assuming that the firm is well known and established. For a SaaP startup firm, a profit margin of around 30% should be acceptable to establish credibility; as such, recalculating using the new profit margin, the product would need to be marketed at around $62,096.32, generating a net profit of around $14,329.92. From Table 1, it is shown that the current cost of this product is very low, with only a small amount of fees being required for distribution purposes.

Assuming that the software will be developed and implemented by only one organization, we feel that this type of software is mostly reusable and can be applied to many organizations for their parking needs. We could even create a framework based off our application to create a product that can have different features from different companies inputted whilst keeping the same backend so that we could increase revenue through licensing and distribution agreements. Furthermore, this does not include maintenance fees to ensure the integrity of the system in the future, increasing revenue even more. The market cost of our product will differ based on the estimation of project development cost, as we progress through development, the project cost and duration estimations will be recalculated, which could lead to a lower market cost.

Additionally, we can look at marketing the product as a service, otherwise known as software as a service (SaaS). Instead of charging a high flat rate cost for a product, we can instead market this product as a service, where the organization that wants to use the service pays a monthly or yearly licensing and maintenance fee. This can be more enticing to organizations as there are no high upfront costs, and may expand market opportunities, as only large-scale organizations would be able to afford to pay for the full product cost upfront. With this option, any size of organization would be able to comfortably afford and employ our product. Whether our software is marketed as a product or as a service, there will still be monthly fees charged to the organizations to maintain the database and cloud server, as seen in Table 1.

8 Risk Analysis

8.1 Database Risk

To store all the various types of information about users, parking spaces, and user-related information, we will use a database that can provide up-to-date information. Security measures for personal information can be a concern against bad actors if there are no methods to ensure security. For instance, a user's name and permit number could lead to identity theft.

Since we are using a database that uses NoSQL, potential attacks, such as an SQL injection attack, could be a possibility. A NoSQL injection attack involves an attacker including infected input/code into a query that is delivered

[2]https://softwareequity.com/blog/gross-margin-saas/#:~:text=Based%20on%20our%20experience%2C%20a,dive%20into%20several%20other%20metrics

by the user; the infected portion will execute unwanted commands on the database [3]. To avoid this, we should ensure that the inputs being sent to the database have all been sanitized and cleaned of any unwanted, malicious code.

8.2 Application Risk

A learning curve can be described as the time it takes someone to acquire new skills or knowledge [4]. A risk that comes along with new ideas is the ease of use; specifically, creating an interface that is intuitive and easy to understand for new users. To mitigate this risk, the design of the application is crucial so that users can easily use the built-in functionality. Familiarity is a key objective when designing a new application, as you want a user to have some sense of how things function. With that, a product should also be unique and offer clear benefits over other competing products too.

Connection-related risks are important to understand as well. Since our application relies on real-time information for a large portion of the functionality, stable and reliable connections must be present. If there are connection issues, the application should be able to handle these issues accordingly, so that the user experiences minimal interruptions. This could be in the form of disconnection alerts or other indicators that show a poor or lost connection. The real-time status updates could provide a problem if there are delays or interruptions with them as the user experience gets affected. Furthermore, since the application is for mobile devices, battery concerns will be present as well. Constantly collecting GPS information along with communicating and relaying information with a server could prove costly for battery life.

8.3 Social Acceptance Risk

When dealing with GPS tracking, it is apparent that there will be privacy risks that come with it. By using information about the user's GPS location, not every user may agree with the application tracking their position. This could lower the overall participation rate once the product goes mainstream. To ensure the highest accuracy of available and unavailable parking spots, we require a high participation rate; otherwise, they will incorrectly classify parking spots. To help mitigate this risk, a geo-fence will be set up around the G parking lot on Lakehead's campus, as shown in Fig. 3. Once users enter within the fencing, the GPS function will turn into a more precise locator compared to before, this will help with some privacy concerns. Furthermore, a well-formed, legal, terms and conditions will be implemented into the application that users must be willing to accept in order for them to use the functionality of the application.

Figure 3: The geo-fence that will be surrounding the parking lot.

8.4 Budgetary Risk

As discussed in Section 6, the estimated cost of development was calculated to be $331,074.88. However, this is hypothetical, as that is how much it would cost to develop if we were an actual software engineering firm. However, our actual costs are projected to be very low, as we are only developing a prototype application, and most of the materials required are either cost-less or can be obtained for a low price. As of now, we do not expect to require any funding for this project, but as we progress further into the development of the prototype, we may have some unexpected costs, causing our overall cost to increase. We may also want to industrialize our software, which may require a larger budget than currently set. The most expensive budget risk would be increasing the size of the database to keep track of the required information. We are currently planning on using Google Firebase, which is free but does have its limitations. Although Google Firebase provides basic NoSQL features, we may need to increase our budget to opt for a better package allowing us to store more information, or change platforms altogether, depending on our future needs.

In the future, when budget may become more of an issue, we plan to submit a request to Lakehead University for a grant to fund the project, as our current vision of the application will not generate any income to help fund the project. For this reason, we will mainly rely on Lakehead University to provide funding if necessary.

The majority of budget risk is hypothetical and poses no risk in our current situation, as we have our selected resources, which leave us at near-zero cost. The idea of budget risk is to delve deeper into the finalization of the project, if we decide to industrialize our product; we will most likely have a higher cost margin.

8.5 Task Management Risk

When putting a lot of effort and time into a large project of this scale, we want to ensure that we have a set goal to complete each sub-task as we do not want to spend too much time on one task which may lead to a time deficit. Fig. 2 provides us with a guideline for a schedule that allows us to finish this project on time. Unfortunately, events can occur that prevent items from being completed in a timely manner, which could potentially delay other components from being completed in their allotted time. When this happens, it is important that we focus on completing the delayed task as quickly as possible and look at the schedule ahead and adjust any upcoming deliverables accordingly.

8.6 Expertise Risk

This project involves learning many new skills that we have not used before. Programming languages that have not been learned and used in school before will be used. The size and scope of this project are more expansive than any other project we have been exposed to as well. Therefore, to manage this, we must ensure that we do not become overwhelmed by the idea of learning all this new information. This can be done by doing ample tutorials and learning courses on the required materials before the hands-on work begins. Additionally, for areas of expertise where only one member has some experience in, they will be asked to advise or take charge of the development of that section.

9 Conclusion

A software project management plan is a crucial component of any project. It allows for concise and clear documentation regarding what needs to be done, how it is done, when it needs to be done, how much it will cost, and how long it will take. All these questions should be answered in this document.

The software process methodology was outlined and described to be the waterfall method, an older yet still quite useful model. The six phases were clearly described to better understand what each phase consists of. To ensure that this model works properly in our project, we constructed a well-defined schedule that allows us to identify when these phases will be carried out throughout the timeline of our project.

Estimating the cost of a project is an important factor in collecting initial funding and providing information to any stakeholders in the project. The Intermediate COCOMO technique was applied, and a final cost estimate was calculated to be $47,766.40. We will be using this estimation as a comparison with the final cost once the project is completed to measure the precision of our estimates.

When doing research on the current competing mobile applications that are available on the Google Play Store and the Apple App Store, it was found that most applications are free to download and do not contain any in-app features that require additional payment. Furthermore, we discussed marketing the software as either a product (SaaP), or as a service (SaaS). We computed the market cost of the project, which would allow us to achieve a large amount of revenue or even accumulate customers that could license our product.

Different risks were analyzed and addressed, such as database, application, social acceptance, budgetary, and expertise risks. For each identified risk, a detailed understanding and analysis were provided, along with any measures that could be taken to reduce the risk to an acceptable limit. There will always be risks in any type of project; thus, we must be able to identify them early on and be prepared to handle them accordingly.

Acknowledgement

We would like to express gratitude towards the department of Software Engineering at Lakehead University for this opportunity and Dr. Thangarajah Akilan, Chair of the Software Engineering Department, for being our project supervisor.

References

[1] S. R. Schach and S. R. Schach, *Chapter 9 - Planning and Estimation*, p. 270–282. McGraw-Hill, 8 ed., 2011.

[2] A. Verma and Preeti, "Calibrating intermediate cocomo model using genetic algorithm," *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2021.

[3] Imperva, "What is nosql injection?: Mongodb attack examples: Imperva," 2020.

[4] J. Kagan, "What is a learning curve? formula, calculation, and example," Nov 2022.