McComb's MS IT & Management Capstone 2023

Final Report
H-E-B Digital-SlackBot

Table of Contents

About H-E-B

Established in 1905, H-E-B Grocery Company, LP, started as a local family business and is now an American privately held supermarket chain. Headquartered in San Antonio, Texas, H-E-B operates more than 300 stores with over 100000 employees in around 150 communities across Texas. H-E-B Digital is a section of H-E-B that concentrates on providing

software solutions to various business problems within the company. H-E-B Digital has teams of engineers, analysts, designers, researchers, eCommerce experts and managers. The teams mentioned work on various projects and have a huge code base. As part of our capstone project, we will work with the Enterprise Engineering and Healthcare & wellness teams, subparts of H-E-B digital. H-E-B chose representatives from these teams to assist us throughout the course of the project. We would also be expected to interact with the Cloud infrastructure and Software security teams constantly as they would assist us with deployment.

Cyber Security Concerns of H-E-B Digital: Per Forbes, cybercrimes will amount to $1-2 trillion annually. Cybercrimes can cause significant loss in operations, customer relations and brand image to an organization. Mistakes can happen during design, architecture, implementation, and process. Flaws can also exist across diverse programming languages and computer system components, leading to various vulnerabilities. Unknown software defects are known as zero-day vulnerabilities. These are particularly dangerous because until they're identified and fixed, they can be exploited by attackers. Code vulnerability can create a potential risk of compromising security. Enforcing security standards and practicing them during development can prevent software security vulnerabilities. H-E-B is taking various initiatives to safeguard its data and code from potential attacks. Our capstone project - *Safety third,* is one such initiative towards cyber security. Safety Third project is not associated with just one team within H-E-B digital, rather, it's a project that addresses company-wide concerns pertaining to code susceptibility.

Summary

Business Problem

H-E-B faces challenges keeping the teams informed regarding deadlines for addressing various code vulnerability issues associated with projects in a scheduled manner.

Business Problem Overview

H-E-B faced a security breach on or around January 11, 2021, impacting H-E-B Center IT systems. A third-party service provider that performs IT services for H-E-B Center discovered that sophisticated malware had attacked its systems. The attacker had gained access to various H-E-B Center IT systems, encrypted their contents, and exfiltrated this data to the attacker's systems. These systems contained confidential and personal information of current and former H-E-B Center employees, such as their name, address, contact information, date of birth, and in some cases, social security numbers. Given recent security breach incidents, H-E-B would like to reduce the attack vector for potential security threats proactively. Post the attack, they limited external access to all IT systems and engaged outside experts to investigate the incident and attempt remediation efforts. To combat the issues related to susceptibility in code, H-E-B's software security team has implemented a system that performs daily scans.

These scans are also performed when the user triggers a merge request. These scans are run on the entire code base, check for flaws, and return a report. The report contains detailed information regarding the potential issues with the code. Thereby providing an opportunity to remediate the problem before releasing it into production. These reports benefit the company and the engineers or developers involved in active development. While these scans act as a proactive measure towards code awareness, they have limitations. For example, the teammates must manually look up the report to get information regarding the raised bugs. This costs the employees valuable resources and time. Furthermore, employees may not perform these report checks frequently for various reasons. Teams can grow complacent over a while. Since automated JIRA bug creation is not incorporated with the security scan feature, employees need a way of being notified. Finding and assessing the bugs at the earliest stage is crucial to every product cycle. Streamlining the process of identifying the bugs and notifying the responsible team and the employees involved is of the utmost importance to H-E-B. Again, we must note that this is a company-wide issue and not limited to any one individual team in particular.

HEB Point-of-Contact chart with details regarding their specific responsibilities to our capstone project.

S.No.	Name	Role	Responsibilities
1	Garrett Griffin	Software Engineering manager (Pharmacy)	Initial project idea development and communication. Provide access and approve access requests to resources to UT students.
2	Kyle Kurihara	Software Engineering manager (BF)	Take an active part in sprint retrospective and assist in providing the project scope.
3	Eric Ruiz	DevOps Engineer (Pharmacy)	Assist in deploying the application in the google cloud platform, setup of Kubernetes and docker images.
4	Sam Bell	Systems Engineer (BF)	Point of contact for issues related to the project, act as a collaborator and help communicate with internal teams and vendors to fix problems that may arise.
5	Venki Manoharan	Systems Engineer (BF)	Provide input in strategic technical decisions and solutions as and when needed.

(BF) - Business Facing

Client Project Goals

Currently, H-E-B has excellent vulnerability scanning for applications, but using this information effectively is where the hardship is. To help H-E-B act on their security findings, a tool is needed to take in the Common Vulnerabilities and Exposures (CVEs) and notify the

teams of the deadlines around remediation so that no vulnerability will go unfixed by the teams before the deadline. By doing this, H-E-B can create more secure software that meets their criteria. With these reminders in place, there is a lesser chance for the teams to forget that they must work through these various issues.

So, the idea is to create a Slackbot that will integrate with H-E-B's Gitlab repositories and send the vulnerability scan information to the respective team's Slack channel. Having reminders sent to various slack channels with issues to remediate will be leveraged by the teams across the engineering space. Setup of the completed Slackbot should allow the user to specify the channel to report to. That is, which repositories to report the vulnerabilities for, and when to report the vulnerabilities. Reported CVEs should be in the order of longest-lived vulnerabilities. Allowing the users to create JIRA tickets based on the vulnerabilities presented.

This table below maps business goals with technical goals:-

ID	Business Goals	Technical Goals ID
BG1	Read the Common Vulnerabilities and Exposures (CVEs) scans of each repository	TG1
BG2	Notify respective teams of the deadlines for fixing vulneribilities	TG1, TG2, TG3
BG3	Report the vulnerabilities to multiple channels responsible for fixing it	TG2, TG3
BG4	Teams should prioritize ones with closest dealines and with high level of vulnerability over others	TG5, TG6, TG4
BG5	No need to notify the teams during holidays	TG7
BG6	Frequency of notifications of the vulnerability to increase with moving closer to deadline	TG4

ID	Technical Goals	Business Goals ID
TG1	Create a Slackbot to integrate with Gitlab repositories and send the vulnerability scan information	BG1, BG2
TG2	Send the deadline reminders to various slack channels	BG2, BG3
TG3	Slackbot should allow the user to specify the channel to report to	BG2, BG3
TG4	When and how frequently to report these vulnerabilities	BG6, BG4
TG5	Reported CVEs should be in the order of longest-lived vulnerabilities	BG4, BG2
TG6	Mute specific levels of vulnerability	BG4
TG7	Automated Holiday mute	BG5

Business goals delivered through MVP

- The SlackBot is able to talk to the GitLab API for reading CVE scan details of repositories. This solution met out first business goal (BG1)

- Our MVP is sending reminders about the deadlines of vulnerability scan which the slack channel subscribed for. This feature is notifying the developer about the time left to fix the vulnerability. This solution met out second business goal (BG2)
- Our bot reports the vulnerabilities to multiple channels which are responsible for fixing them. This functionality delivers our third business goal (BG3)
- Slackbot can be used for scheduling of notifications so that developers can control the frequency of the vulnerability scans received. This functionality met out sixth business goal (BG6)

Technical goals delivered through MVP

- With our MVP we have delivered our first technical goal was to integrate slackbot with Gitlab API and send vulnerability scan information. This goal is tied with business goals of read (BG1) and report (BG2)
- The developer will receive reminders on the slack channel for fixing the vulnerabilities before the deadlines. This fulfills our second technical goal which is related to notify (BG2) and report (BG3)
- Our SlackBot is allowing users to specify the channel to report about when and how frequently they want these vulnerabilities to be reported to the channel. This meets our fourth technical goal. This also connects to business goals of orders (BG4) and adjusting frequency (BG6)

Overview of the system

What did we deliver? // Add more details to each with diagrams if possible

Created a Slackbot

Containerization of the app

Deployed in the Cloud with DB access

What did we miss?

Mute-specific levels of vulnerability

and Automated Holiday mute feature

Fixes with a high level of vulnerability should be prioritized

Business Value

SecureBot system will scan, identify, and notify Common Vulnerabilities and Exposures (CVEs) in code repositories and send alert notifications to the development teams to address those issues in a timely manner.

This new system will save H-E-B developers time through easy and effective communication to meet the security team's deadlines by providing automated code repository scan alerts within the Slack channel.

The SecureBot features will add the following values:

- **No Manual Reporting/Follow-up:** The scanning of the code repositories will be automated, and the vulnerability report will be published to the subscribed users/group slack channels, no manual reporting/follow-ups are required.
- **Easy & Effective notifications:** The vulnerability scan process will trigger as per the pre-configured schedule and notify results immediately to the subscribed slack channels. Users can get easy notifications on their mobile devices.
- **Custom Notifications:** Users have the flexibility to choose & configure Git repositories and security scan frequencies and can configure skip notifications on holidays/weekends. Alerts & Notifications will be published to only relevant slack channels. No general broadcasting of messages.
- **Avoid Security Breaches:** Regular and automated scans will identify and notify all vulnerabilities proactively to the development teams. Team can add those issues to

their To-do list and resolve them quickly to avoid any long-term potential security & privacy risks.

Pre-requisite/Assumptions: The new SecureBot application required following pre-requisites:

- Users must have access to the Slack messenger application and applicable GitLab repositories.
- Teams need to register and configure their GitLab repository security scans scheduled as a one-time step.
- Teams need to subscribe to their Slack channels with SecureBot as a one-time step.
- SecureBot application will invoke the existing H-E-B process of security & vulnerability scans, it is assumed that those reports are correct & error-free.

Business Value measures: H-E-B team can measure the advantages of the new system based on below key parameters:

- Time: Total time saved by teams/developers in vulnerabilities resolutions after SecureBot implementation.
- Cost: The effort and cost saved by the development teams by resolving vulnerabilities.
- Risks prevention: The cost and risks saved by preventing major security breaches by fixing vulnerabilities proactively on time.

Rithu's part from here

performance metrics

Students identified relevant performance metrics for the system

performance metric calculation

Students clearly explained how the performance metrics should be calculated/tracked

performance measurement

Students clearly explained how the metrics measure the business value generated by the system

Describe the business value the client can obtain by implementing the system you created. Be sure to justify the business value and note any assumptions, and indicate how specific values/metrics were derived (be sure to leverage ideas from previous courses to identify and measure business value).

Performance metrics the client can track to assess if the implementation of the system does indeed create business value.

Based on this assessment, evaluate whether the system should be fully implemented (if not already). If the system should be implemented, address how the system should be implemented (i.e. next steps for the client).

System Implementation

Students clearly identified whether the system should be fully implemented and client's next steps for implementation (if recommended)

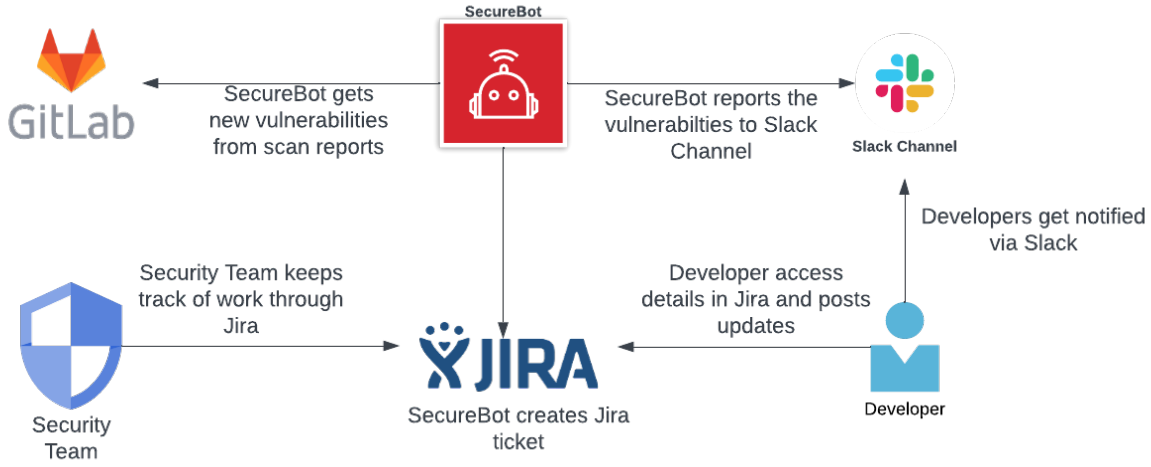
System Recommendations and Limitations

After finishing up our planned development cycles we realized that there could be some more enhancements that could be added to this project that could provide more value to the developers. We also identified some limitations of our system that can be addressed in the future.

Recommendations/Enhancements

Automated Jira Tickets

A nice feature to have would be to automatically create and assign Jira tickets for new identified vulnerabilities. For example, if there is team A consisting of 4 developers who have created a slack channel and invited securebot to it. Instead of manually keeping track of work done, hours put and testing done in a separate team specific document for each developer, automated tickets created and assigned to the developers would help keep a consistent track record of all work done. This would also help keep track of the work assigned to developers in terms of accountability to the security team as they can view the timelines of each vulnerability in a central location. This would also allow for team HEB to keep track of metrics related to how soon vulnerabilities are fixed in an organized fashion.



A flow of how automated Jira tickets would work

Holiday Mute

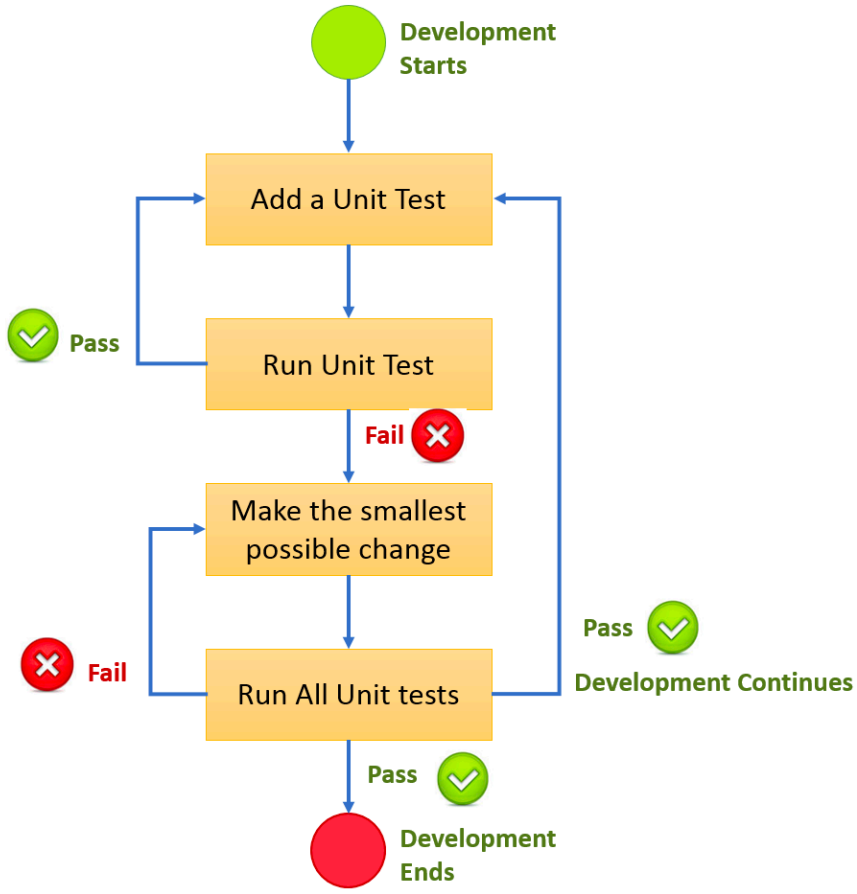
Another enhancement to have would be to add a mute feature which would not produce any alerts on specific days of the year pre-determined by the dev teams. These could be specifically used for holidays like the Christmas break or any other decided upon days where the dev team will be unavailable to view the alerts. This would help prevent securebot spamming the slack channel by reducing clutter in the channel. The idea is to not alert on days when the alert would go unnoticed anyway.

Mute Specific levels of vulnerabilities

In pursuit of the same idea of reducing clutter, another great feature would be muting specific levels of vulnerabilities. For example, for a team of devs who are not required to fix vulnerability level “low”, it would be worthwhile to skip any alerts of such vulnerabilities since alerting them of those vulnerabilities would not help in the first place. This again would help prevent securebot spamming the slack channel by avoiding irrelevant alerts.

Unit Testing

The way we did testing for securebot was by keeping track of individual test cases and making sure each one passed. We put testing details in a dedicated confluence page and linked it to the relevant user stories and sub tasks. However, these test cases were done at the time of first development. As we mentioned some enhancements for securebot which would require more development work, there may be code added to the project codebase or code may be changed in the future. It would be a good practice to add unit tests in the codebase using python testing libraries like pytest that would automate this testing at build time. This would help automate testing of already existing functionality in the code to make sure nothing breaks functionality wise and added features work in synergy with these existing functionalities. If some code is changed that, if pushed to production, breaks existing functionality, the build would fail and developers would realize that perhaps the changed code does not pass all test cases. Subsequently, any new changes would require new unit tests to be added as well.



A flow of how Unit tests would work in a development cycle

Limitations

Admin Tooling

As of right now, the master view of all channels and all linked repos can only be accessed using psql command line tooling to get into the database and querying the Schedule table. Any changes en masse would also be either done through SQL queries or going through securebot. Lack of admin tooling means that there is no centralized view to see which channels are linked to which repos.

Channel Updates

In any organization like HEB, slack channels are created and deleted as need arises. Currently, there is no automated way to keep track of which channel is deleted. This means that after a channel is deleted, the association of the relevant repos to that channel would still exist in the database. Over time this stale data would keep in the database and become clutter. It would be a “nice to have” feature to automate deletion of a channel with securebot so any references of the channel are deleted from the database.

No In-Channel Access Control

There is currently no access hierarchy for securebot within a slack channel. This means that as long as securebot is added to the slack channel, anyone in the channel can use the commands for securebot and even access the information securebot provides from the GitLab scans.

Only Essential Logging

Currently, only essential information is logged. For any future development, more extensive and robust logging at different logging levels may need to be added to the code to help in development efforts. To assist developers, we have included comments, docstrings and confluence pages for reference. But for debugging the code live, more logging will need to be introduced. Another point raised by Kyle ties into our admin tooling limitation made earlier, developers in the future may find it helpful to create an admin tool to view any error logs within the admin tool so that they can stay on top of any issues that may come up with securebot itself.

Only Counts of Vulnerabilities

Since vulnerability scans can be very lengthy in size, securebot currently only alerts about the numbers of vulnerabilities by each severity level. This was done to prevent spam and lengthy automated messages in a slack channel where essential conversations might get interrupted and to overall just improve readability. In the future, including the link to the

GitLab scan may also help provide more resources within the slack channel alert while also maintaining readability.

Conclusion

H-E-B is a massive company with a diverse array of technological needs, enabled by Partners at H-E-B Digital. These Partners produce a large quantity of software for a wide swathe of different purposes. Building and maintaining this large code base brings with it the inherent risks of creating cybersecurity vulnerabilities and exposures. Mitigating these risks requires understanding where the potential vulnerabilities and exposures may lie.

Before the Safety Third project, H-E-B had access to GitLab's Vulnerabilities API. However, Safety Third's principal application, SecureBot, enables Partners at H-E-B Digital to keep a closer focus on the vulnerabilities present in projects their teams maintain and with greater regularity.

Regarding business goals, the team at H-E-B Digital first expressed a fairly broad goal of increasing the visibility of critical vulnerabilities and exposures (CVE) within their developers' day-to-day communications. The second business goal is to utilize the GitLab Vulnerabilities API, which H-E-B already pays to use but may not yet be using to maximal effectiveness.

Partners at H-E-B Digital gave our team several key technical goals to work toward in this project. The first goal was to provide software capable of fetching a GitLab Vulnerability scan for a particular repository. The second goal was to create a Slack application capable of transmitting a scan to a particular channel. The third goal was to create a scheduling system capable of fetching and delivering scans from a particular repository to a particular Slack channel at a specified cadence.

Technically speaking, SecureBot meets and exceeds all of the goals given to us by the Partners at H-E-B Digital. In the scope and requirements gathering phases of Safety Third, we were told that a project with a *hard-coded* GitLab repository ID and a *hard-coded* Slack channel ID which could shuttle a scan from GitLab to Slack would be considered a successful minimum viable product. Our project is perfectly capable of shuttling a scan from *any* GitLab repository to *any* Slack channel in the H-E-B Digital workspace, and at a daily, weekly, or monthly cadence beginning on any day a Slack user specifies. The only change required to accommodate the, "any repository", specification is swapping SecureBot's current GitLab token out for a GitLab token with sufficient scopes to access the GitLab Vulnerability Reports of all H-E-B repositories.

The exact business value of a solution like SecureBot is difficult to estimate for all of the same reasons that any cybersecurity product is difficult to measure: if the product has the intended effect, the business will never figure out how much money they would have lost from the ramifications of a cybersecurity incident. How much is it worth for an organization's software engineers to take software vulnerabilities more seriously and pay attention to them on a more consistent basis? This is unclear. Perhaps easier to estimate is the amount of value that is being harnessed by taking advantage of a product which H-E-B Digital is already paying for: GitLab Ultimate. GitLab Ultimate is a roughly 60 dollar premium per employee per month above the next highest tier, GitLab Premium. GitLab Ultimate includes vulnerability scanning and management, whereas GitLab Premium does not. It's unlikely that H-E-B Digital pays for GitLab Ultimate solely for the vulnerability scans, but if we could estimate this feature as even 10% of the appeal of paying for this service it adds up quickly considering that H-E-B Digital employs over 10,000 people. Assuming that roughly 30% of H-E-B Digital employees get GitLab Ultimate accounts, SecureBot enables a more effective utilization of 60 x 3,000 x 0.1 = 18,000 dollars per month of company expenses. Beyond this, SecureBot's aggregation of vulnerability reports into one convenient place for developers to see may enable developers to focus their work time more easily and on more important tasks. Before SecureBot, a developer interested in finding the

highest vulnerabilities across all of the repositories maintained by their team would have to visit each repository and investigate the vulnerabilities and their respective levels separately before choosing which repository to work on improving. With the addition of SecureBot, this same developer can receive a breakdown of each repository in their development Slack channel and identify which repositories have the most important issues to address at a glance.

Although SecureBot meets the goals set by the client, there are many ways to improve the product both in terms of feature-richness and user experience. First and perhaps most important is the adjustment of messages by SecureBot to channels to make them more understandable and expressive. Currently, some messages from SecureBot don't communicate all of the content that would ideally be given such that an end user understands what happened and, if applicable, what they should do next. Furthermore, CVE reports from SecureBot are currently a quantity breakdown of the various levels of vulnerabilities that might be present in a particular repository. Adding more details about the nature of the vulnerabilities would make interacting with the bot and subscribing to scans feel more meaningful and impactful.

Another major improvement from a maintainability perspective is the addition of SecureBot "cleanup" features which enable an end user to remove the bot from a channel, clear all schedules, etc. These cleanup features should also be used when a Slack channel is deleted to prevent a buildup of irrelevant or "null pointer" rows in the database which refer to a channel which no longer exists.

As discussed by Kyle Kurihara, the long-term success of this project also hinges on the observability and maintainability of the project, which means building more informative logging, monitoring, and administrator tools into SecureBot. Right now, an administrator looking to understand how SecureBot is being used or make a change to certain schedules must do so using a command line connection to the PostgreSQL database and an

understanding of how the application works. Good logging, monitoring, and administrator commands / utilities would be a highly desirable target for future efforts on this project.

Despite the limitations of SecureBot (i.e., showing counts of vulnerabilities but not full descriptions, current lack of in-channel access controls, no admin tooling), it's a great scaffolding for a tool which provides developers at H-E-B Digital with a convenient and comprehensive way to receive regular notifications about the security status of the repositories they are responsible for. The code is highly maintainable and extensible, and includes robust CICD pipelines which enable changes to the project to propagate to H-E-B's production environment quickly and effortlessly.

Acknowledgements

McCombs MS-ITM team is very Thankful to the H-E-B business for granting us this opportunity to work on this project. This project would not have been possible without the H-E-B team's kind support, guidance and help.

We are highly thankful to Mr. Garrett Griffin (Software Engineering manager- H-E-B) and Mr. Kyle Kurihara (Software Engineering manager-H-E-B) for making sure that the McCombs team has all the support available in this project and guiding us in H-E-B sprint planning and retrospective meetings.

We cannot Thank you enough Mr. Sam Bell (System Engineer - H-E-B) and Mr. Eric Ruiz (DevOps Engineer – H-E-B) for all the knowledge sharing, guidance and constant help on H-E-B internal cloud setup and system accesses and infrastructure issues resolution and for providing all the support by answering our queries. Thanks to Mr.Yung Lai for guiding us with critical Google Kubernetes setup & deployment configurations.

Thanks to Mr. Venki Manoharan (Systems Engineer –H-E-B) for guiding us with necessary business information, project technical decisions and project technical implementation plans.

We would extend our sincere Thanks to Dr. Caryn Conley (McCombs) for her continuous encouragement and constructive suggestions and for pushing us for excellence. This project would not be perfect without her time and support.

We would also like to express our special gratitude and Thanks to the McCombs School of Business faculties for giving us a wonderful learning experience and providing us with all the facilities to succeed in this coursework.

In last, our huge thanks and appreciation to our team members for their patience, willingly help each other, and for putting in all the extra hours to complete this project on time.