C868 – Software Capstone Project Summary

Task 2 – Section A

Capstone Proposal Project Name: MedBookr – A Medical Appointment Tracker

Student Name: ______________________________

Table of Contents

Business Problem

The Customer

The customer is Pacific Care Medical, a not-for-profit outpatient center in the Bay Area of California providing medical services to lower income members of the community. The outpatient center currently employs 54 doctors, ranging from general practitioners to specialists in many different areas, as well as a large number of nurse practitioners and medical assistants. The medical center serves about 500 patients and, as of now, has reached full capacity. To be able to treat more patients in need, there are several outreach programs in place to bring in more funding. Several large Silicon Valley companies have already pitched in and more are expected to offer their support. Consequently, the medical center is expected to grow and be able to employ additional medical staff and accept new patients. With this anticipated growth there will be greater demand on the existing administrative infrastructure, especially the center's ability to track the records of doctors, patients and their appointments. By any measure, the current systems in place are largely outdated. Patient reservations are for the most part taken over the phone and entered into a monthly planner. Later those records are transferred to various Microsoft Excel spreadsheets that often end up scattered in different locations of the medical office. It has become pressing that a more robust technology solution be found to support current and future operations of Pacific Care Medical.

Business Case

MedBookr – the proposed software solution - is a complete appointment management system designed specifically to meet the needs of medical offices. It will facilitate tracking patient appointments with a particular doctor as well as managing patient and doctor records. At present, Pacific Care Medical has to juggle the ad hoc organization of a large number of doctor and patient records and an even larger number of appointments. The current means of administering this data is a mix of manual, paper-based methods and the use of software products that have not been designed to handle relational data effectively. This haphazard system of managing appointments is frustrating to the staff, inefficient, and frequently error prone. It is not unusual for doctors, and even patients to be double booked, for past appointment records to be lost, and for patients to be entered into the system multiple times with slightly different information. MedBookr will be able to address all these insufficiencies by centralizing the entire appointment management process into a single web-based location. The web application will be backed by a robust, relational database capable of handling the present load and also scale to accommodate the projected growth in patient, doctor, and appointment records.

Fulfillment

MedBooker will be a web-based appointment tracking application for use by approved medical staff at Pacific Care Medical. To ensure that only authorized personnel can access the sensitive records of the customer, all functionality will be behind a secure sign-in. New account creation will be disabled by default. There will be four main sections to enable the logistics of managing appointments effectively and reliably.

The first section will allow management of staff records such as doctors and nurses. The application will have the ability to list doctors and quickly search for a particular member of the medical staff by name or by role/specialty. Details about doctors such as their contact information, profile picture, and a short bio will be available on a separate page. The same page will also list upcoming appointments for the given doctor and allow for the creation of new appointments. Information on doctors can also easily be updated via the application.

The second section will be designed to manage patient records. Patients can be listed and searched by name or filtered by age range. Patient details, such as contact address, phone, email, date of birth, age, and gender will be available on a separate page along with the patient’s upcoming appointments. The ability to add appointments and edit patient information will also be provided.

The third section will deal with appointment tracking. Appointments will have a start time and duration and be linked to a particular doctor and patient. Validations will ensure that neither the doctor, nor the patient is double booked. It will also be impossible to accidentally create appointments for past dates, weekends or outside of business hours. The application will display appointments in a familiar and intuitive calendar format both by week and by month. The calendar will switch between the monthly and weekly view and each will allow for forward and backward navigation. Appointments for particular doctors and patients will also be listed on their respective detail pages.

Lastly, the application will have some useful reports to view trends in the appointment data. There will be graphs to display the number of appointments by date, by doctor, and by time of day to assess more or less busy periods for making future data-driven scheduling decisions.

The application will be backed by an industry standard database, which will enable possible future integrations with other software solutions such as systems for keeping detailed records about patients and their medical history. In addition, it will also be possible to transfer existing patient, staff, and appointment records from spreadsheets to the new database using custom data import scripts.

Existing Gaps

Pacific Care Medical is experiencing many problems with their current mode of tracking appointments for their busy medical practice. The staff responsible for managing the records is overwhelmed by the sheer amount of work required to manually keep tabs on appointment times and to ensure that contact information for patients and doctors is kept up to date and in order. Patients and medical staff, on the other hand, complain about frequent scheduling errors and missed notifications due to incorrect or outdated contact information.

The current way of practice management can only be described as disorganized and chaotic. A combination of notebooks, planners, and the use of various software applications that each only solve part of the problem, are employed to handle this highly sensitive data. For instance, Microsoft Excel is frequently used by staff to track some of this information. While Excel is well understood and quite easy to use, it is also very limited in its ability to handle relational data, that is, related but separate data sets, such as doctor and patients, or patients and their appointments. In addition, Excel's search and filtering capabilities are cumbersome and not user friendly. As an example of a commonly faced difficulty, when a patient calls to cancel an appointment, the medical assistant on duty might have to consult a variety of different spreadsheets to find the information about the patient and the existing appointment. There is also the possibility that the appointment has only been recorded in a paper-based planner and not yet transferred into digital format. In any case, the updates required to cancel the appointment need to be reflected in all the appropriate places, that is the different spreadsheets or lists relating to the patient, the appointment, and the doctor the appointment was with. This is an arduous task that is very prone to human error.

It is evident that the problem of effectively tracking medical appointments and related records is one that requires a high degree of precision and organization. A web interface to manage this data in one place would go a long way in alleviating these difficulties faced by the customer. A centralized database-backed application with a user- friendly interface would allow staff to enter, update, and search records with ease, which in turn would increase efficiency and significantly lower the incidence of manual errors.

SDLC Methodology

For delivering the first set of features for the MedBookr application, we have chosen the Waterfall method of developing software applications. This method is categorized as a predictive approach to software development in that the project's features are known and clearly laid out at the outset of the project. Based on the set of agreed-upon features, development can then follow a predictable series of steps or stages, each with their own deliverables. The Waterfall method is well suited to a project like MedBookr since the problems the medical office is facing are fairly well understood and the specific features required to manage doctor, patient, and appointment records can be hashed out in coordination with the customer.

The Waterfall method begins with the <u>requirements phase</u> that seeks to understand the client's current situation and needs. This is the stage where the comprehensive set of features is enumerated and the desired functionality of the software solution is described in detail. The main output from this phase is a requirements document that has been distributed to all stakeholders. In addition, a project schedule that lists the stages of the project with approximate time estimates will also be provided at this junction.

The requirements phase is followed by the <u>system design phase</u>, in which the project team designs the system from the ground up. While this stage involves no coding, the technical specs of the system such as the programming language(s), database diagrams, deployment and testing strategies are decided upon. The deliverables tied to this stage are a low fidelity Wireframe to show the flow of the application, a high-fidelity prototype that will demonstrate the UI design, a database diagram that sketches out the structure and relations between the data models, and a testing plan to ensure full and correct functionality.

The next step is the <u>implementation phase</u> where the actual design and development of the software project takes place. Based on the outputs from the previous stages, programmers and designers start the work of building out the application. This stage is internally subdivided into multiple shorter phases to ensure that the team gets a chance to come together, compare notes and make sure that everything is being done according to plan. The output from the implementation phase is a functional software product that fulfills all the functionality specified in the requirements document.

Following implementation is the testing phase that aims to find and report any and all deviations from the agreed-upon requirements. Quality assurance teams are tasked with thoroughly battle-testing the application and making a list of any bugs and errors they encounter. The engineering team will immediately act on all bug reports and fix any problems that arise.

Once all the known problems have been resolved, the delivery or deployment phase can begin. This will put the application into the live production environment and get it ready for use by the customer. The deliverable here is a software application that the customer can actually start using.

Finally, the maintenance phase makes sure that the customer can still turn to the development team as issues arise. Maintenance consists of fixing newly discovered bugs, creating patches, and making updates as requested by the customer. There will be a maintenance contract that will specify the terms for supplying ongoing maintenance services.

Deliverables

As mentioned above, the Waterfall method of software development has a well-defined set of deliverables tied to each stage of the project lifecycle. These can further be subdivided into project deliverables that are under the purview of the Project Manager and product deliverables that represent the actual software product delivered to the customer. We will expand upon both types of deliverables in the following section.

Project Deliverables

- Requirements document

- Outlines features and intended functionality of a software application in great detail and represents the software development team's understanding of the customer's needs as well as the exact nature and constraints of the system
- Needs to be written in such a way that it can be clearly and unambiguously referenced throughout all stages of the project
- Most importantly, it should include details about user interfaces, functional capabilities, performance levels, data structures, quality and reliability metrics, as well as constraints and limitations

- Project Schedule
    - Detailed timeline that lists the main stages and substages of the project along with their respective deliverables and estimated time ranges
    - It should include tasks and milestones and provide a brief description of each to avoid ambiguity
- Low-fidelity Wireframe
    - Mockups of the entire application are designed to demonstrate the basic flow through the website and the most important functional aspects
    - Does not need to have details of the exact design elements that will eventually be used but it should clearly show the main navigational components and the different ways to interact with the application
- High-fidelity Design Prototype
    - Based on the wireframe, the high-fidelity prototype will focus on the look and feel of the application and contain the full set of theme elements such as typography and color scheme
- Database Diagram

    - Will show the blueprint for the database schema including structures, data types and relations between the different database tables
    - Will give a comprehensive picture of the database representation of real-life entities such as doctors, patients, and appointment
- Testing Plan
    - Outline of the included unit tests suite to ensure the correct functionality of the software application. Unit tests are closely based on the code they are designed to test and can take advantage of white-box testing methods. The unit test suite will run in an automated fashion and produce structured output.
    - Functional test plans that will involve a QA specialist performing a series of steps to check the end-to-end functionality of the application. Each step will have clearly defined inputs and outputs and the tester's role is to ensure that the expected outcomes are actually produced.

Product Deliverables

- Functional database with custom schema that matches the tables laid out in the database diagram and powers the data storage needs of the application
- Fully developed application that fulfills all the functionality specified in the requirements document
- Navigation and flow of the website that match the patterns shown in the wireframe
- Graphical user interface that matches the high-fidelity prototype
- Secure admin access for a number of selected staff members
- Deployment to a secure and easy-to-manage cloud environment
- Support with initial data import to facilitate the process of getting started with using the application

Implementation

The implementation of this project is expected to go smoothly and not cause any major disruptions to the operations of Pacific Care Medical (PCM) staff. In the requirements phase, the Project Manager will work with the leadership team at PCM to understand the exact requirements and later to make any needed amendments. This should only require a few 60 to 90-minute meetings. One of the advantages of the Waterfall method is that the client only needs to be involved in the early and the late stages of the project lifecycle rather than needing to make representatives available throughout the project, as is the case with other methodologies. In addition to the project requirements, stakeholders from PCM will also need to sign off on the project timeline as well as the low and high-fidelity wireframes. All this will ensure that PCM has a solid idea of the end product they will be receiving.

During the testing phase, the Project Manager will ensure that PCM staff selected to pioneer adoption of the new software application will be adequately trained on using it. This will also put the customer in a good position to successfully perform acceptance tests before releasing the project for deployment. It is also in this phase that the software team will create custom data transformation and import scripts to initially seed records for doctors, patients and appointment into the new database. This should greatly ease the immediate utilization of the web application and will again be handled in coordination with the personnel responsible for piloting this process.

Right after the application is deployed to the production environment, the software team will create admin accounts for the chosen group of staff members. After that step is completed, the team at PCM should be able to start making use of the new and greatly improved workflows of managing the scheduling needs of their medical practice.

Validation and Verification

We'll devise a comprehensive testing plan to ensure that the new appointment tracking application looks and works as outlined in the requirements document. Testing and quality assurance are a central part of all our projects as we strive for nearly complete test coverage of our code bases.

In conjunction to writing code, our software engineers create unit tests to verify the functionality of each and every method called in a running program. Unit test are low level tests that check methods in isolation and confirm an expected behavior or output given particular inputs. The suite of unit tests can be run automatically so it is ideal for use during development as more and more features are added and on deployment events to ensure that changes to the code did not break or alter existing functionality.

At the next level are functional tests that will be devised and executed by our internal QA team. Functional testing strives to test specific requirements and parameters of the software solution. We will create a set of functional tests to verify the functionality of all requirements identified in the requirements document. We expect that this will go a long way in ensuring that all the agreed upon functionality is implemented fully and correctly.

Finally, any software solution is only viable if it can perform its function in the real-life environment of the customer. Acceptance tests seek to ensure that the product meets customer needs and it useful and usable in performing common day-to-day tasks in the customer's environment. These are the highest levels of tests and will be performed by personnel from PCM in coordination with the Project Manager before deploying the application.

Environments and Costs

Programming Environment

The application will be developed using the Ruby on Rails web application framework. Most of the code is written in the Ruby programming language but several sections of the site will also utilize

Javascript, a lightweight interpreted scripting language for the browser. The database backend will be PostgreSQL, an open source object-relational database management solution. We will also use a number of well-chosen, reliable libraries from the web development ecosystem to facilitate certain common tasks and thereby save programming time and money.

The application will be hosted in shared a cloud environment called Heroku. Deployment to Heroku is very easy and will also greatly simplify the ongoing server management of the application. Heroku takes care of creating the isolated, virtualized Unix environment needed to host the Ruby on Rails application. Due to its handy command line interface (CLI), deployment to Heroku can be as simple as running one command from a developer's terminal. At the same time, Heroku has a user-friendly dashboard to manage additional services needed to run an application, such as database storage, monitoring or error tracking.

Environment Costs

Due to the modular nature of the Heroku deployment environment, the cost of running the application in production will, by today's standards, be quite moderate. The pricing scheme on Heroku is based on "dynos" which are lightweight container environments ready-built to accommodate modern web applications. Two instances of a Standard 1x dyno, which we estimate will be sufficient for the initial needs of the customer, will cost \$50/month. Additional dynos can be added seamlessly if more website activity is expected. For data storage, the Basic plan for the Postgres database, which can accommodate up to 10M rows, should also be enough to begin with and will cost \$9/month. Again, larger and customized plans are available as needed. Lastly, there will be the nominal cost of paying for a custom domain name, which comes to about \$10/year. The total yearly cost of hosting the application will thus amount to \$718.

Human Resource Requirements

In terms of human resources, the project requires a project manager (PM), a designer, two software engineers and one QA specialist. The PM will work part-time throughout all the phases of the development lifecycle with the lowest activity expected during the implementation phase. Her total activity is estimated to amount to 100h, which will cost $6000 at her hourly rate of $60/hour. The designer's hourly rate is $60/h, but his activity is mostly confined to the system design phase and is estimated at 40h for a project of this size. The two software engineers are expected to account for the lion's share of the overall development costs. Their hourly rate is currently $70/h and the total length of the implementation phase is estimated to be 3 business weeks. Finally, the services of the QA specialist are required during the testing stage and should not exceed 40h, which would come to $1600 given an hourly rate of $40/h. As such, the total cost for human resources for this project is estimated to be:

Resource	**Rate * Time**	**Total**
Project Manager	$60/h * 100h	$6000
Designer	$40/h * 40h	$1600
Software Developers	$70/h * 120h * 2	$16,800
QA Specialist	$40/h * 40h	$1600
TOTAL Human Resource Cost		**$26,000**

Project Timeline

Phase	Milestone/Task	Deliverable	Description	Dates
Requirements gathering	Discuss and finalize requirements	Requirements document and project schedule	Meetings with PCM stakeholders to gather and prioritize product requirements	3/1/19, 3/4/19

System Design	Create and review low fidelity wireframe and high-fidelity design prototype	Low and high-fidelity website mockups	Designer will create a lo-fi wireframe to show navigation and interactivity and a design prototype that will show the final look of the site and be matched by the developers	3/5/19-3/12/19
System Design	Create database schema for site	DB Diagram	Developers will define and model entities to be represented as database tables and identify relationships between them	3/5/19-3/7/19
System Design	Create overall testing plan	Testing Plan	PM and developers will create comprehensive plan for unit, functional and acceptance testing	3/8/19, 3/11/19
Implementation	Create functional application including unit tests	Final alpha version of the application	Developers will implement feature requirements and match prototype design to create a fully functional application according to specs. All unit tests will run without errors.	3/12/19-4/2/19
Testing Phase	Devise and execute functional tests	All functional tests are executed successfully	QA specialists creates and runs functional test to verify all the requirement spelled out in the initial requirements document.	4/3/19-4-9/19
Testing Phase	Devise and execute acceptance tests	Customer executes acceptance tests to satisfaction	In coordination with PM, customer will run acceptance test to verify that the application meets requirements.	4/10/19-4/12/19
Deployment	Prepare deployment	Deployed application ready	Set up Heroku environment and	4/15/19-4/16/19

	environment and deploy application	to be used by customer	deploy application after final checks.	
Maintenance	Set up maintenance plan	Maintenance service contract	Discuss and finalize terms of service for ongoing maintenance with the customer.	4/17/19