Feasibility Study

1 Team

Jaewon Sim - js2689@cornell.edu
Albert Tsao - awt46@cornell.edu
Yun Ping Tseng - yt376@cornell.edu
Yuxin Xu - yx67@cornell.edu
Angela Zhang - az337@cornell.edu
Amy Zhong - az287@cornell.edu
Jennifer Zhou - jz547@cornell.edu

2 Client

Erica Ostermann; Assistant Dean; Hans Bethe House; eo93@cornell.edu, hadbethe@gmail.com

3 Task to be Undertaken

The project and goal of the team is to create an interactive and mobile-friendly web interface for Hans Bethe House's events; otherwise known as the Bethe Online Program Sign-Up (OPS) Project. The Bethe OPS will be a web interface that displays Bethe's upcoming events and provides a mechanism for residents to sign up or add themselves to the waitlist. Residents should receive email notifications, such as for sign-up links, event reminders, and notice of when a spot opens up for someone on the waitlist. The system will also provide a staff interface, which allows Bethe event coordinators to easily add and edit events on the web interface, and an administrative interface, which allows permitted administrators to also post events and collect data on these events.

4 Preliminary Requirements Analysis

The functionality of the system will be separated into three categories: required, desirable, and optional.

4.1 Required Functionality

The basic functionality of the system will be to provide an interactive webpage that allows students to sign up for Bethe events, and an administrative interface that allows Bethe staff to have control over the event logistics and compile data from events. The required components of this system are: web interface, email notifications, waitlist, database, and different types of login.

Web Interface
The web interface should display the upcoming events and provide a mechanism for people to sign up for events. The interface should display all the important information related to the event, such as the date, time, affiliate links, etc. The number of spots available for each event should be displayed on the landing page before a user signs up. In order to sign up for an event, the user should fill out a form with a few fields such as name, email address, what building they live in, etc. This web interface should be easy to navigate and user-friendly. Any mobile or desktop device should be able to locally host the web interface.

Waitlist
There will be an option to sign up for a waitlist for the event if there are no more available spots, and the web interface should be updated if anyone removes themselves from the event list such that people at the top of the waitlist can be automatically added to the event roster.

Email Notifications
The system should send several different email notifications. If a student signs up for an event, (s)he should receive an email notification with a confirmation and a link that allows the user to take him/herself off the sign-up list. If a spot opens up, an email should be sent to the first person on the waitlist notifying that (s)he has been removed from the waitlist. There should also be two reminder emails sent, one a day before the event and another a few hours before the event. Students on both the sign-up list and the waitlist should be notified if event details have been changed or if an event has been canceled.

Database
The database should hold all information regarding the events, the sign-ups, and the waitlist. The data should be separated by semester. The data should be downloaded and stored in a .csv file.

Login Types
The landing page and sign-up mechanism can be public, but there should be a separate login for GRFs (Graduate Resident Fellows) and event leaders with the proper login authentication. These users will have control over adding and editing the events on the interface. There should be a different login for administrators as well who have the same

capabilities as the GRFs and event leaders, but administrators can download event data stored in the database and control the permissions allowed for other users in the system.

4.2 Desirable Functionality

The following features are preferred but not required:

- Implementing a social aspect where a user can see what other people are attending
- Carousel design using images of the event posters
- Ability to categorize and filter types of events on the interface
- Enabling administrators to send out customized group emails through the administrator interface
- Automatically sending an email to the GRF or event leader when someone drops out of an event (this mechanism can be toggled on/off by the GRF/event leader)
- Enabling the general user to log in using their netID or Google account
- Ability to add events to a user's Google calendar
- Web interface can be hosted locally until more information about domain hosting is confirmed

4.3 Optional Functionality

An optional functionality would be adding specialized features on the interface. Some specialized features include:

- A way to display different events depending on the user that is logged in; for example, events meant for transfer students to meet new people would be highlighted for only transfer students
- Text/SMS integration along with the email system
- Event proposal system with the administrator interface for GRFs and Active Citizens

5 Suggested Deliverables

5.1 Management Deliverables

<u>Requirements Document</u>

A document to be presented to the client that formally outlines all requirements of the system (functional and nonfunctional). This document explicitly states the desired requirements understood by the team to ensure mutual understanding between both sides and provide opportunities for the client to clarify any misunderstandings before proceeding to implementation.

<u>Design Document</u>

A more technical document describing the overall design of the system. This document will be written by those on the team with a more technical background and will provide the client with a technical description of how the team plans to implement the system moving forward. Review of this document can also provide the client with opportunities to make technical suggestions or modify any requirements as needed based on the proposed implementation.

Status Reports and Presentations

Status reports and presentations to the client will be critical for both keeping our team on track and communicating with the client. Conducting regular analyses of our progress through these milestones will act as mini-reports that will guide future decisions and give the team a realistic understanding of the project's timeline and trajectory. Finally, these reports and presentations will additionally act as a channel of communication with the client, ensuring that the client remains up-to-date with the team's progress. This will help nurture a relationship of trust between the team and the client, ensuring that both sides can effectively work together to create the desired system.

Project Documentation

With such a complex system being developed, regularly creating and updating project documentation is critical to keeping track of all the features of our system. Maintaining project documentation will also be helpful for developing a mutual understanding of the system among the team and provide a way for future developers of this project to quickly understand the design of this system.

Source Code

A final report and presentation of the final system during which the team officially hands the source code over to the client. By this point in time, the team would construct a final report documenting the project and the system that has been developed based on the client's requirements and previous iterations. The system will have been tested in multiple ways throughout development, including an acceptance test, and ideally will be ready for deployment the following semester.

5.2 Technical Deliverables

User Interface

A user interface needs to be set up to display Bethe house events, the number of spots open for each event, and allow users to sign up for the events. This will be the primary technical deliverable. This interface will include all the required features agreed between the client and the team, and the client will be able to run the user interface locally on a cell phone or computer.

Email System

An email system needs to be implemented to confirm event sign-ups, send event notifications to event participants and possibly to event leaders, and waitlist notifications to students who are removed from the waitlist for an event (s)he signed up for. This system should be running in the background, and proper documentation with details will be given.

Database

A database with the required tables and columns to store available data about the Bethe events will be given to the client. Administrators will have control over editing the data in the database. The administrator should be allowed to download and store the data into a .csv file.

Login Authentication

A login authentication that distinguishes between the different types of users (e.g., administrators versus event coordinators) and redirects the user accordingly will be implemented as part of the web interface.

Administrative Interface

An administrative interface that gives administrators of the system the ability to control privileges in other user accounts (e.g., event leaders); add, edit, and remove events; and gather data reports containing all possible information regarding the Bethe house events will be implemented as part of the web interface.

Staff Interface

A staff interface that gives GRFs, event leaders, and other Bethe staff that support event logistics the ability to add, edit, and remove events will also be implemented as part of the web interface. Staff should be able to access who is participating in the event(s) they are managing.

6 Software Development Process

The project will follow an *iterative refinement* software development process. Through initial discussions, the client clearly defined a set of requirements for the system but also asserted the requirements may vary depending on how smoothly system design and implementation go. Due to this flexibility and variability, the team believes undertaking an iterative refinement model for this project will allow it to best fulfill the client's requirements for the project.

In addition, by using an iterative refinement method, the team can first go through a design process that will involve creating mockups of the user interface and refining the look of the user interface through drafts based on client evaluation and feedback. The team will also start planning out the system architecture. Once the user interface design is approved, the team will develop an initial prototype in the early stages of development that will be refined through user testing and successive iterations. Through these iterations, the team is able to take advantage of the flexibility presented by the client, and expects understanding and confirmation of the requirements to improve.

7 Outline Plan

Below is the proposed outline of the team's software development process for the Bethe OPS Project. The proposed outline is divided into three iterations. Each iteration is associated with a deadline, milestones, and details of what the team expects to complete. In addition, throughout each iteration, the most recent version of the software and any designs will be evaluated and tested by the client and other necessary users.

7.1 Iteration 1 (March 8, 2019)

Requirements Analysis and Specification

After a formal meeting with the client to discuss the project, the team will prepare a formal document outlining the client's requirements for the system. Additionally, the requirements will be classified into required, desired, and optional functionality as the client sees fit. This document will then be presented to the client for formal approval.

User Interface Design and Mockups

The team will develop several user interface designs for the web interface to present to the client. After a user interface design has been agreed upon, the team will begin development of the system.

System Design and Architecture

An initial draft of the software design and architecture will be developed. This draft will include details on the back-end and front-end implementations of the web interface. A meeting will be held with the client for feedback on the system design.

Initial Web Interface/Prototype

In order to ensure that the system is being developed as per the client's specifications, the team will develop an initial web interface. This web interface will include both user and administrator level interfaces containing required data entry fields. These interfaces will be simplistic and will not contain all the desired features and functionality the client requires. The prototype will be presented to the client in a formal meeting for evaluation.

7.2 Iteration 2 (March 29, 2019)

Documentation and Presentation

The source code and documentation about the design and structure of the software and interfaces will be delivered to the client for instructions and future maintenance of the software.

The group will give a 45-minute presentation to the client as a progress report. Based on this progress report, the team will update the task schedule with more details and ensure that the final product can be delivered on time.

Revised Web Interface

Based on the feedback from the client and Bethe House residents as potential users, the team may modify some functionality of the web interface. The revised web interface may not be the final version as the team may need to revise certain aspects of the interface depending on the client's request.

7.3 Iteration 3 (May 16, 2019)

Final Testing

The team plans to execute the final testing two weeks before the deadline. The client will test the product with a group of potential users and give the team feedback. At this point, the product should have met all the requirements, and the team expects to only revise minor details.

Final System

The final system will meet all the required functionality expected by the client. Depending on the team's progress, the team may also add functionality that the client lists as desirable/optional.

Final Documentation and Presentation

The final documentation will include information about all the required functionality and the desirable/optional functionality that the team has implemented so that the client can modify or maintain the system in the future. The documentation will also include unimplemented functionality that the client can use for system growth. The team will also provide a final presentation and training so that the client has a better understanding of how to use and manage the system.

Final Handover

All materials will be handed over to the client. Such materials will include the project source code, feasibility study, requirements analysis and specification, system design, user interface design, testing content (program and user), and transfer of the rights to the system as further discussed under Business Considerations in section 9.

8 Visibility Plan

8.1 External Visibility Plan

The team will make efforts to maximize the visibility of the system and development process. The team's primary form of communication will be through weekly in-person meetings and emails. During the in-person meetings, the team will provide intermediate deliverables such as prototypes and presentations in order to keep the client up to date with the progress of the system.

8.2 Internal Visibility Plan

The team will use Slack to maintain communication in an organized matter, separating channels into the different areas of the technical stack of the system (front-end, back-end, design, etc.). The team will also use the collaborative environment, Google Drive, to keep track of meeting notes with the client, deliverables, and presentations. Source code will be maintained and documented in a Github repository.

9 Business Considerations

As Cornell students, the team owns the copyright in the software that we create in this project. The team agrees to transfer the copyright to the client and to provide the client with unrestricted license to use the system.

10 Technical Feasibility

The feasibility of the technical requirements can be judged by identifying *at least* one technical method for each requirement. The technical requirements and possible technologies or methods the team can use to satisfy them are described below.

10.1 User Interface

The user will interact with the software through a web-based interface. The system will support a mobile environment and will have to be cross-device tested. The interface may be developed to be responsive. The team can use common front-end technologies such as HTML, CSS, and React Native to develop the user interface.

10.2 Server

Cornell servers would be ideal for this project since the client is employed by the university, and this is a Cornell-related project. However, a downside of opting for an on-premise server is the process involved in getting access to the resource. Cornell requires a Statement of Need for all new IT projects, and the process will take at least two to three weeks to finish. During this process, 55 to 65 college director-level officers are stakeholders. Even after the initial waiting period, it is possible that the proposal may be rejected or that restrictions may be imposed on the team's access to the server, citing security concerns. Regardless, the client would prefer hosting the system internally via Cornell servers and is willing to let the team to host the system locally until the process is seen through.

Should Cornell IT (CIT) refuse to provide campus resources to host the system, the client is open to using and allocating a budget to spend on third party hosting. A serverless solution is an option. Should the team choose to use a serverless solution such as Google Firebase, back-end engineering would be simplified. This would be beneficial to the client who will be managing the software after its development. The downside is that Firebase's free plan has a resource limit, including 100 simultaneous connections to the database and 1GB of static hosting. The client could opt to upgrade the plan at any point in time without affecting software uptime, providing scalability.

In either case, to handle middleware and routing on the server side, the team can use Express.js, a popular Javascript framework for back-end management in web applications.

10.3 Email Notification Service

The team can use email infrastructure such as SendGrid that provides an easy-to-use API for sending transactional emails.100 emails/day are provided for free forever. Alternatively, should Cornell provide IMAP/SMTP mass mailing services, the team can integrate the Cornell-given address with the back-end server to send emails. The benefit of using an email infrastructure service is that this requires less configurations and that the team is provided with a guarantee that emails will be sent reliably by the service provider given the configurations are correct.

10.4 Database

The database could be achieved using a SQL or NoSQL. In the case that the team uses a NoSQL database (e.g. a JSON database), the back-end design would be simplified, allowing the team to spend the additional time on implementing other features. Alternatively, the team has inquired CIT on whether database services are provided. If CIT permits the system to be hosted on Cornell's servers, the team may consider utilizing any databases available by CIT.

10.5 Login Authentication

Authentication could be achieved using Cornell's login system, as preferred by the client. Especially if the team is able to host the system on Cornell's server, using Cornell's login solution would be the best choice. An alternate option is to use the Google Sign In service. Students can use their Cornell email address to authenticate with the back-end. In the case the back-end is structured using Google Firebase, the service natively supports Google Sign In. Otherwise, the team should create a mechanism to securely generate and pass tokens.

11 Risk Analysis

11.1 Time

Since the project is expected to be completed by the end of the semester, there exists a risk that the project will not be finished with the full functionality the client desires. To minimize this risk, the team will practice frequent communication with the client. Providing status updates and multiple mockups or demos will allow the client to periodically evaluate the functionality of the system.

11.2 Changing/Incomplete Requirements

Since the project is flexible, without many mandatory design requirements but with much room for extra features, there exists a risk that the client may change her mind. This will result in a certain amount of wasted time and effort. To reduce the possibility of this occurring, the team will conduct frequent demonstrations with the client.

11.3 Lack of Resources and Tools

Many of the features requested by the client happen to be back-end heavy, but the team lacks in software engineers with strong back-end experience. To counteract this setback in the team composition, the team will focus on finding a back-end framework/language that has a shallow learning curve and managing its time such that the team can adequately learn back-end technologies without setting back the progress of the system by a significant amount.

11.4 Technical Requirements

The client wants the team to employ a database that will not generate a recurring fee. However, that will be difficult because most databases cost a certain amount of money. Furthermore, the client wants the system to be able to archive and store data for extended periods of time. For the team to use a free database, the amount of data that can be stored will have to be limited. To resolve this issue, the team will search for a database that generates only a one time fee.

11.5 Non-functional Requirements

At the initial client meeting, the client did not specify the number of users that the system must be able to support. If the system does not successfully support the potential traffic that may be generated at peak times, then the whole system may shut down, which may discourage users to return. To prevent this from occurring, the team will conduct a follow-up meeting to specify this requirement.

12 Conclusion

From the results of the feasibility study, the team finds the Bethe OPS Project to be feasible in terms of technicality, the team's skillset, and time. Given the flexibility of the project requirements, the time constraint of one semester is manageable for the team to implement the system and satisfy the client's requirements and expectations. The team possesses adequate skills to utilize and learn the technologies necessary to implement the system desired. The conclusion of the feasibility study is to proceed with the Bethe OPS Project.