ISSN: 2473-3857
v2 n4053

Raising the Bar: Challenging Students in a Capstone Project Course With an Android and Mobile Web Parallel Development Team Project

Wilson Wong
wwong2@wpi.edu
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609, USA

James Pepe
jpepe@bentley.edu

Irv Englander
ienglander@bentley.edu

Computer Information Systems Department
Bentley University
Waltham, MA 02452, USA

Abstract

Information systems capstone projects aim to prepare students for what they will encounter in the industry after graduation. Corporate application development is often a complex endeavor that requires coordination between related products. For example, software development in the mobile application sector may require a coordinated parallel development of native cellphone applications and mobile web applications. The dual approach taken by these companies enable end users to access the application over a wide variety of devices and operating systems. Instructors usually must choose between a mobile web development environment and a native development environment such as Android or iPhone. In order to provide students with a learning experience that incorporates additional complexities of the real world, a challenging capstone course project is presented that requires a large team to implement the same application in both environments. This course was implemented in a single semester at Bentley University in the spring of 2015. Student teams created pub crawl applications based on stops within a local mass transit system that would run both on an Android phone and on a mobile website. Java, Eclipse and Google's Android SDK were used to create the Android component. JQuery, HTML5, PHP and JavaScript constituted the development environment used to create the mobile web component. The project management and coordination of the two development environments within a single team resulted in unexpected challenges. Factors leading to varying degrees of successful completion of the team capstone projects are presented along with lessons learned.

Keywords: capstone course, software project management, mobile application development, Android, mobile web, team structure

ISSN: 2473-3857
v2 n4053

1. INTRODUCTION

The purpose of the software project management capstone course, CS460, at Bentley University is to give graduating seniors experience in team management and complex team application development. In many companies, application development is often an endeavor that not only requires the coordination of members within a team, but also the coordination of sub-teams with one another, where each sub-team is responsible for one or more components of the application. Our capstone project was devised to better prepare graduating Computer Information Systems seniors to work in these complex team environments. Student teams were required to create an application that would generate a pub-crawl, i.e. a walking tour of bars and restaurants centered on a user-selected local mass transit stop in the Boston subway system. To mirror the application development environment of many software companies, each student team had to develop an Android version and a mobile web version in parallel throughout the semester. A further area of complexity involved the creation of the back-end SQL database that both versions would access. This project satisfied the objective of a capstone course by challenging student teams to apply their knowledge gained from a wide range of prior computer courses – Java, Android, web, and databases – to the management and development of a major project. Project management concepts are included as part of the course material and applied in the software development process.

Many IS/IT programs offer a project capstone course as a means of integrating the program material from previous courses into a coherent team project effort (Heshemi & Kellersberger, 2009; Leidig & Lange, 2012; Mew, 2014; Reinicke, 2011; Reinicke, Janicki, & Gebauer, 2012; Schwieger & Surendran, 2010; Shih, LeClair, & Varden, 2010; Tappert, Cotoranu, & Monaco, 2015). Many of these capstone courses with substantial projects involve the creation of web-based applications (Abrahams, 2010; Maloni, Dembla, & Swaim, 2012; Stillman & Peslak, 2009; Tappert, et al., 2015; Umapathy & Wallace, 2010) or mobile applications (Matos, Grasser, & Blake, 2008; Payne, Zlatkov, Jernigan, & Scarbrough, 2009; Tappert, et al., 2015). Generally, these projects are purposefully limited in scope due to course time constraints, the technical background of the students, and the number of students on a team. The current CS460 model is much broader in scope than the typical project course described previously. It is intentionally ill-defined, requires significant requirements gathering, is organized into large teams, assumes significant student team and workload management, and requires the team presentation of a working model at the end of the semester. The stated goal is to more closely simulate the real-world operations that a student can expect to face in the workplace.

Our capstone model of parallel Android and mobile web development arose from similar offerings in previous semesters that involved only a single mobile development environment. For the past five years, the fall version of the course had students developing mobile web applications and the spring version of the course had students developing Android applications. The reason for the dichotomy was that during the fall, most students would not have yet taken the Android course. After teaching successful versions of both types of capstone courses, a challenging, combined Android and mobile web course was delivered in the spring of 2015.

2. BACKGROUND

At Bentley University, Computer Information Systems majors must fulfill a combination of business requirements and departmental requirements. In addition to nine business core courses, CIS majors take two database courses, and one course each in Java, system analysis and design, as well as an introduction to operating systems and networking. CIS majors may take, as electives, additional software programming courses such as web development, advanced Java programming, Android development, and network programming. CS460 Applied Software Project Management is the elective capstone course that students take where they synthesize knowledge learned in their previous CIS courses towards the creation of a software application. The course topics include software development life cycle concepts, Agile methodologies, software project management, team dynamics, risk management, software size estimation, and quality assurance.

In previous semesters, student teams have applied software development methodologies to the creation of applications for real world clients such as hospitals and other non-profit organizations. Examples have included applications to aid in coordinating online language lessons for Afghani citizens given by English speakers in other countries, assign

hospital beds to patients, and provide destination paths for medical clinic visitors. Having teams produce software for the general public, as is described in this paper, provides students with an increased challenge with respect to requirements gathering and incorporating user feedback throughout the development process. Applications for the general public that student teams created in the past have included programs that guide end users along a city walking tour, help end users avoid or reduce speeding ticket amounts, and direct students to their final room destinations in a university. We particularly note that each of these applications was designed by its team for use on a single, specific platform.

3. PARALLEL DEVELOPMENT MODEL

To enrich the team management and development experience, the course project component was expanded to include a parallel development model, in which each team would be responsible for the development of a single application that would operate on multiple platforms. This would simulate the sub-team experience of real-world project development.

The instructor employed a skills matrix in forming the teams, based on a student CIS background survey distributed in the first class. To facilitate the parallel development model, team sizes were increased to accommodate the significantly larger project scope that had to be delivered within a single semester. Teams were chosen to reflect a balance of experience in Android development, web development, and project management. Most students already had two semesters of database courses and the remaining students were taking the second database course in conjunction with CS460. As a result, the students with weaker software development backgrounds worked on a database development sub-team. Two balanced teams with the requisite web, Android, and database experience were formed with eight students each. The eight person teams also offered students experience with the larger teams that are characteristic of many real world business projects, an experience that is rarely made available to students.

4. COURSE DELIVERY

A major challenge is to incorporate both course material and team project development into a single semester. For our course, this challenge was compounded by the additional requirements imposed by parallel multi-platform development. Course components consisted of lectures, software project management practicums, midterm and final exams, and a term project which required a midterm presentation, a final presentation, student peer reviews, project management documentation, software design documentation, and a working application.

Software project management practicums were class sessions devoted to student teams applying the concepts learned from earlier lectures to their software application development and team project management processes. Great care was taken to balance lectures with software project management practicums so students would have both the maximum amount of time to devote to developing the application and the necessary knowledge in software project management to accomplish their goals efficiently and effectively. Two different approaches were initially considered: 1) present all of the lectures in the first half of the course and dedicate all of the remaining classes for teams to apply the concepts and create their applications and 2) alternate lectures and the software project management practicums so students could apply a lecture's material in the very next class. The first approach has the disadvantage of not giving teams enough time to design and create the application – only half a semester. The second approach introduces a number of topics such as project scheduling or software sizing long after they are needed by the teams. As a result of these issues, a third approach was incorporated into the class. The first half of the course consists of approximately two-thirds of the lectures which includes the materials necessary for teams to get under way. The teams present the results of their requirements gathering, software designs, and project management documents during midterm presentations which are scheduled for the week after the midterm exam so students do not have conflicting goals. In the second half of the course, the remaining lectures such as quality assurance and agile methodologies are presented early enough so that the final exam can be given weeks before the final class session [see Appendix A] thereby freeing up students to concentrate on only application development towards the end of the semester. These final lectures are also timed to support the students starting their software development iterations. If quality assurance had been presented as one of the final lectures, then only a single iteration of application development would have been possible.

Consequently, this two-thirds/one-third approach successfully presents most of the material to the students prior to their needing it for their projects but maximizes the amount of development time that they have.

5. TEAM PROJECT DELIVERY

A key component of Agile methodologies is that the team must be co-located so members work together and engage in face-to-face communications (Beedle et al., 2001). Due to the difficulties of coordinating the schedules of eight student team members, teams could not be expected to be primarily co-located, even online, nor could they even be expected to hold daily meetings. Instead, a hybrid methodology, midway between Agile and traditional life cycle methodologies was adopted, an approach suggested by Baird and Riggins (2012). This had the additional advantage of allowing students to focus on the value of various features within their applications, even given the time pressure exerted by the rapid Agile development cycle. Nonetheless, Agile methodology concepts are an important component of the course and many aspects of Agile methodologies were mandated by the instructor. Teams would be self-organizing with respect to team member positions and would operate in regularly timed cycles with an exception for the architectural spike, i.e. the initial requirements gathering and creation of the software designs. After the first six weeks which were allocated for the architectural spike, teams would develop iterations of the application in sprints of roughly two weeks each. The exam schedule which overlapped the development cycles prevented the sprints from being strictly time-boxed.

Once members were assigned to teams by the instructor, one of the initial tasks for teams was to self-organize themselves, i.e. agreeing on which positions team members would hold, as recommended in Agile methodologies (Goodpasture, 2016). Teams were instructed to use their own version of a skills matrix in making team position assignments. The positions that had to be filled were project manager, project lead analyst, and project analysts. Each team member other than the project manager would be assigned to the Android, web, or database development sub-teams although assignment changes could be made as project needs would necessitate. Each software development area would appoint their own lead analyst to simplify coordination and communications within the team. One of the project analysts would also be responsible for coordinating all quality assurance efforts, and another project analyst would be responsible for coordinating documentation. In effect, core application development is performed by the Android and mobile web sub-teams while the other positions provide supporting roles. Project managers create weekly reports for the instructor who acts as the vice president of software development.

The mobile web development environment that the student teams employed consisted of HTML5, JQuery and the WAMP stack. WAMP is an integrated PHP, MySQL, Apache web server environment running on Microsoft Windows. The selection of HTML5, the latest HTML standard, permitted teams to incorporate GPS location on mobile devices if their chosen software features required the technology. The corresponding Android development environment consisted of the Java SDK, Android SDK, Eclipse, and the Android Plugin for Eclipse. A MySQL database was used as the back-end for both development platforms. Because students did not have experience with PHP, existing samples of the code were provided. Students were able to successfully adapt the code to their projects because PHP's syntax is similar to Java.

In the first half of the semester, teams dedicated their time to determining software requirements and then creating software designs for a minimal application that would be implemented in the first software development iteration. With projects that have a specified client, interviews and informal discussions are often conducted to generate a list of software requirements. Without given direction, student teams can flounder when attempting to determine the software requirements of an application to be used by the general public. Project teams were instructed first to gather software requirements through brainstorming sessions. In these sessions, team members were instructed to propose common features, as well as pie-in-the-sky features, free from criticism. Once a substantial feature list was created, teams would prioritize features and remove those that would not be feasible within the timeframe of a semester project. In order to determine what features should be included in the application, teams would then distribute surveys of potential features to individuals matching the profile of possible end users. Armed with the survey results, teams would further group a prioritized feature list into three categories – features required for a minimal

application, application features that most users would expect, and a set of "delighters" - features that most users would not expect but would appreciate. If time permitted, teams would conduct interviews of representative end users together with observations of people actually using their application, as each software version was completed. Because of students' inexperience with software size estimations, it is critical that the instructor critique and adjust the software requirements to be implemented in the software development cycles.

After the features for a minimal application were determined, the teams worked to create the software designs for their systems. These software designs included database entity relationship diagrams, context diagrams, UML diagrams and user interface mockups and were presented during the midterm presentations and submitted with the final project deliverables.

In the second half of the semester, after the midterm presentations were critiqued by the instructor, teams implemented the application over three sprints. In the first development iteration, teams were to create a stripped down version of their application which would incorporate the first category of software requirements – those necessary for a minimal application. In the second iteration, teams would implement the next category of features – those most expected in the application. The final application would contain at least one feature from the final category that would delight the audience / general public. It was a project requirement that application features would be implemented on both platforms, with the possible exception of the delighter feature.

This hybrid Agile approach with an architectural spike, an emphasis on good requirements gathering, and three development iterations addresses a serious course concern – the possibility of teams failing to produce a viable application within a single semester. Rather than designing the entire application before software implementation, teams initially create a much simpler design for a minimal application. This first software version can then be quickly implemented because most features are missing and the design is correspondingly cleaner. Once the initial working version is created, teams are guaranteed a passing grade and the teams then implement remaining features in the subsequent iterations, applying lessons learned from the first attempt. If teams are not successful in the first development iteration, they still have four weeks to meet, and hopefully exceed, the base requirement of a working application. So that students are not conflicted in their dedication to the software project at the end of the semester, the final exam is given after the first iteration is completed. Students can then concentrate solely on the project during the second and final software development iterations.

6. COURSE OUTCOMES AND EVALUATIONS

There were significant differences between the two teams with respect to their team interactions and software development experiences. Team Beta began with a serious impediment to their effectiveness. They had chosen for their project lead analyst and Android lead analyst one of the weaker software developers. The more experienced developers had been reluctant to take on the responsibility of coordinating the entire team. Although the instructor attempted to ameliorate the situation by stressing throughout the semester that it would benefit them to have a strong assistant lead analyst or a backup lead analyst, this advice was ignored. In comparison, Team Alpha chose more appropriate leadership positions for their team members.

Both teams progressed successfully through the architectural spike in the first half of the semester and created a list of software requirements, software designs, and project management documents. In both teams, a single set of user interface diagrams were designed for the web and Android platforms. Correspondingly in the midterm peer evaluations, team members on both teams rated one another highly. The only notable issue seemed to be that both teams had included far more features for the minimal application than necessary. The real challenge began in the second half of the semester when the teams began to implement their application.

Team Alpha – Implementation Iterations

The leaders chosen for this team were experienced and closely matched the expectations of the instructor. An additional position of user interface analyst was created by the team to coordinate the UI of both the web and Android applications.

The first development iteration was successfully completed on time despite initially encountering problems with connecting to the MySQL database. During the second development iteration, the project manager became

ISSN: 2473-3857
v2 n4053

unavailable as a result of unexpected personal issues. Fortunately, the project lead analyst was able to temporarily take over those responsibilities along with coordinating team development and serving as the Android lead analyst. The team was successful in completing the second development iteration on time despite different programming problems in the two platform development sub-teams. The Android sub-team encountered problems with implementing a shortest path algorithm and the web sub-team had difficulties with navigation. For their "delighter" feature, the Android team implemented voice-to-text so potentially inebriated end users would not have to read their cellphones to follow directions to their destinations. The Android, web, and database sub-teams all performed well and met their goals.

The UI analyst was especially successful in creating a unified look while permitting appropriate modifications for each development platform. The UI analyst accomplished the unified look through constant communications with both the Android and web sub-teams. Although in the web application, the pub crawl screen displays both the map and the list of bars, the Android application displays the same information in two tabs because of the limited viewing area.

Team Beta – Implementation Iterations

The project lead analyst was expected to coordinate the web and Android development teams while leading the development in one of the two platforms. The selection of an inexperienced developer as the project and Android lead analyst position had a major negative effect on the productivity and coordination of the entire team.

The web development sub-team initially encountered database connection issues that necessitated using the second development iteration to complete the minimal application. When the web sub-team began to encounter additional problems without the support of the project lead analyst, the project manager was added to the web sub-team. Although serious doubts were expressed about their ability to implement the web crawl feature, they were eventually successful in the final release of the application. After the addition of the project manager, the web sub-team worked effectively not only to create the pub crawl application but also to include administrative features to facilitate population of the pub-crawl and subway stop databases.

The Android development sub-team had the same initial difficulties with the database connection; this was resolved by the end of the first development iteration. This sub-team encountered a steady stream of serious programming errors that delayed the successful implementation of the application until the last development iteration. The members of the Android sub-team felt that their lead analyst was disruptive during meetings and did not contribute working code. Even though the more senior developers had avoided taking on the responsibility of Android lead analyst, they eventually had to do so anyway or risk an implementation failure. This was an important learning lesson for the entire team. In contrast, the database sub-team, which was led by an experienced database developer, worked efficiently throughout the three iterations. As each problem surfaced, they would quickly address it and solve it.

The assignment of the project manager to support the web development sub-team negatively affected the coordination between the web and Android sub-teams. Although a single set of user interface diagrams had been created, the implementation of those diagrams diverged between the two sub-teams because they worked mostly independently with little communication between the two sides concerning the user interfaces. The differences between the web application and Android application can be seen in Appendix C. Despite the disruptions caused by poor software development leadership, the team eventually addressed their imbalances and produced working versions of both the web and Android applications. For the "delighter" feature, the Android version included a link to Uber in the event that the end user is too intoxicated to return home using public transportation.

Course and Student Evaluations

Of the fourteen semesters that the authors have taught this course, this course delivery, the first that implemented the parallel design methodology, received the highest student rating ever - 5.75 out of 6 points. Furthermore, students indicated a high level of satisfaction and gave ratings of 5.7 or 5.8 out of 6 points in every category on the course evaluations. In comparison the average rating for the course given in the previous twelve semesters was 5.32. Improvement in student comprehension of

the course concepts was reflected in the increased exam scores. In the previous semesters of this course, midterm exam grades averaged 81.4 and final exam grades averaged 81.1 out of 100 points. Students participating in the parallel development project in spring 2015 scored noticeably higher - an average of 89.0 for the midterm exam, and 86.9 for the final exam. Furthermore, the project management and software design documentation that both teams submitted were of high quality and demonstrated a strong understanding of the course concepts. Students stated that the course project gave them "insight about the real world" and "the ability to apply all of our CS knowledge in order to create an application was super cool."

7. CONCLUSIONS

In many basic respects, the capstone project, as newly defined, resembles its simpler earlier project counterpart. Like the simpler projects executed in previous versions of the course, students had to navigate through the definition, requirements, design, and implementation of an ill-defined system. The addition of the parallel implementation

1. required the students to organize and manage their teams and sub-teams much more carefully through the use of skills matrices and sub-team leaders.

2. allowed us to introduce more formal software project management methodologies.

3. forced the teams to consider complicating factors, such as user interfaces, the differences in implementation methods, the available services on different platforms, and the like more carefully.

Overall, this led conspicuously to a much deeper understanding of project management and development processes by the students. At the same time, we share some valuable lessons that we gained from managing the team experience:

1. Selection of the proper team leaders, especially the project lead analyst and project manager, is important to the efficiency and smooth workings of the team. There should be individuals assigned to be the backup project lead analyst and backup project manager in the event the leaders become unavailable, cannot perform their responsibilities adequately, or are the wrong people for the position.

2. When developing for more than one platform, the selection of a person to coordinate the user interfaces is critical. Team Alpha's applications appear unified and coherent because they assigned such a position. In contrast, Team Beta did not have such a person and the user interfaces diverged significantly from one another [see Appendix C].

3. Each development iteration had the two platforms implement the same features. However, development hurdles appeared at different times and in different features between the two platforms. This made it additionally challenging to execute the multiple development iterations on schedule if one or the other development platform is delayed. In the future, although the final features in the web and Android applications should be almost the same, the order of feature implementation in the two platforms should be decoupled from one another. This approach can also permit one platform to take advantage of information learned in the other platform. For example, in the pub crawl applications, determining the shortest path to the next bar could be solved by the Android sub-team in the first development iteration. Rather than duplicating the work, the web sub-team can employ parts of that solution in the second development iteration. Plus, student teams could also present some of the lessons they learned at the end of each iteration so that other teams can benefit from their experiences.

4. Teams did not truly understand what was meant by a minimal, streamlined application. Students misinterpreted minimal to include additional features beyond selecting a location and getting a list of bars. A recommendation is to list explicitly the minimal application's software requirements.

5. The requirements gathering phase proved to be an important and useful aspect of the project, as it helped to organize the teams and the shape of the application.

Despite the additional challenges of software development for two different platforms and the larger teams, the parallel project model that we implemented met and exceeded the goals of a

ISSN: 2473-3857
v2 n4053

complex team project that we set for the course, as evidenced by the student course evaluations, exam grades, final project documentation, and the project applications themselves.

8. REFERENCES

Abrahams, A. (2010). Creating e-Commerce Start-ups with Information Systems Students: Lessons Learned from New Venture Successes and Failures. *Information Systems Education Journal, 8*(35), 3-24.

Baird, A., & Riggins, F. J. (2012). Planning and Sprinting: Use of a Hybrid Project Management Methodology within a CIS Capstone Course. *Journal of Information Systems Education, 23*(3), 15.

Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., Highsmith, J., et al. (2001). Twelve Principles of Agile Software. Retrieved June 23, 2016, from http://www.agilemanifesto.org/principles.html

Goodpasture, J. C. (2016). Project Management the Agile Way: Making it Work in the Enterprise, Second Edition. Ross Publishing, Plantation, Florida.

Heshemi, S., & Kellersberger, G. (2009). The Pedagogy of Utilizing Lengthy and Multifaceted Projects in Capstone Experiences. *Information Systems Education Journal, 7*(17).

Leidig, P. M., & Lange, D. K. (2012). *Lessons Learned From A Decade Of Using Community-Based Non-Profit Organizations In Information Systems Capstone Projects.* Paper presented at the 2012 Proceedings of the Information Systems Educators Conference, New Orleans, Louisiana.

Maloni, M., Dembla, P., & Swaim, J. A. (2012). A Cross-Functional Systems Project in an IS Capstone Course. *Journal of Information Systems Education, 23*(3), 283-296.

Matos, V., Grasser, R., & Blake, B. (2008). Pencils Down! Phones Up! An Interdisciplinary Capstone Project. *Journal of Computing Sciences in Colleges, 24*(1), 67-75.

Mew, L. (2014). *Improving an Information Systems Capstone/Consulting Course for Non-Traditional Undergraduate Students.* Paper presented at the 2014 Proceedings of the Information Systems Educators Conference, Baltimore, Maryland.

Payne, D. L., Zlatkov, D. T., Jernigan, J. M., & Scarbrough, J. M. (2009). *A Location-Aware Mobile System for On-Site Mapping and Geographic Data Management.* Paper presented at the Proceedings of the 10th ACM conference on SIG-information technology education (SIGITE '09), Fairfax, VA.

Reinicke, B. (2011). Real World Projects, Real World Problems: Capstones for External Clients. *Information Systems Education Journal, 9*(3), 23-27.

Reinicke, B., Janicki, T., & Gebauer, J. (2012). *Implementing an Integrated Curriculum with an Iterative Process to Support a Capstone Course in Information Systems.* Paper presented at the 2012 Proceedings of the Information Systems Educators Conference, New Orleans, Louisiana.

Schwieger, D., & Surendran, K. (2010). Enhancing the Value of the Capstone Experience Course. *Information Systems Education Journal, 8*(29), 3-14.

Shih, L.-F., LeClair, J. A., & Varden, S. A. (2010). The Integrated Technology Assessment: A Portfolio-based Capstone Experience. *Information Systems Education Journal, 8*(63).

Stillman, R. M., & Peslak, A. R. (2009). The Complexities of Effectively Teaching Client-Server System Development. *Information Systems Education Journal, 7*(22).

Tappert, C. C., Cotoranu, A., & Monaco, J. V. (2015). *A Real-World-Projects Capstone Course in Computing: A 15-year Experience*. Paper presented at the 2015 Proceedings of the EDSIG Conference, Wilmington, North Carolina.

Umapathy, K., & Wallace, F. L. (2010). The Role of the Web Server in a Capstone Web Application Course. *Information Systems Education Journal, 8*(62).

ISSN: 2473-3857
v2 n4053

Appendix A
Course Schedule

Week	CS460 Applied Software Project Management
1	Course Introduction Project Life Cycle Software Project Team Dynamics
2	Requirements Analysis Project Introduction Software Project Management Practicum
3	Software Development Live Cycles Work Breakdown Structure
4	Software Size Estimation Software Project Management Practicum
5	Duration and Cost Estimation Software Project Management Practicum
6	Project Scheduling, Tracking and Control Software Project Management Practicum Midterm Exam
7	Software Specifications Midterm Presentations
8	Quality Assurance Software Project Management Practicum
9	Risk Analysis Software Project Management Practicum
10	Agile Development Methodologies
11	Final Exam
12	Software Project Management Practicum
13	Software Project Management Practicum
14	Software Project Management Practicum Final Presentations

ISSN: 2473-3857
v2 n4053

Appendix B
Student Background Survey Questions

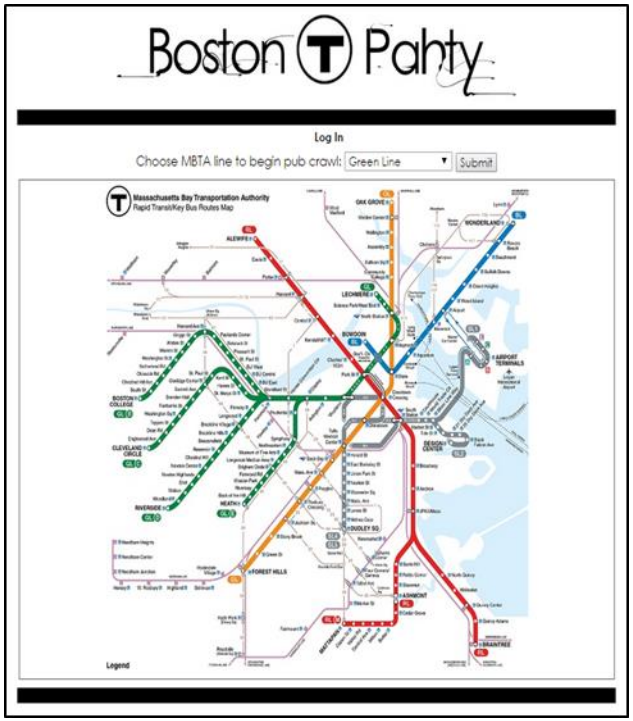
1. Which CIS courses have you taken?
2. Which CIS courses are you taking this semester other than this one?
3. List project management work experience or classes that you have had. Also indicate if you have been a project manager for a class project.
4. List the programming languages and development environments in which you are proficient:
5. List web development classes or work experience that you have had:
6. List quality assurance / software testing experience that you have had:
7. List software documentation experience that you have had:
8. Do you have experience with the waterfall software development life cycle or its variants?
9. Do you have experience with Agile software development methodologies? Mention which ones if you know the specific methodologies.
10. Is there anything else that you have done that would be related to the course?
11. What are you hoping to get out of the course?

ISSN: 2473-3857
v2 n4053

Appendix C
Application Screenshots

Team Alpha – Start Screen

Web Application



Android Application

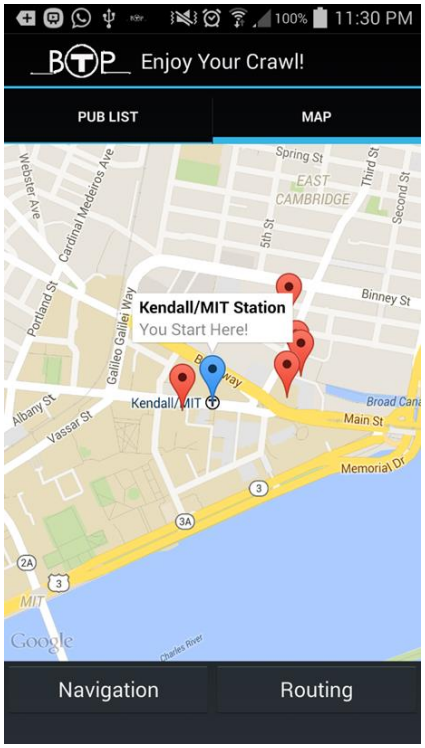
ISSN: 2473-3857
v2 n4053

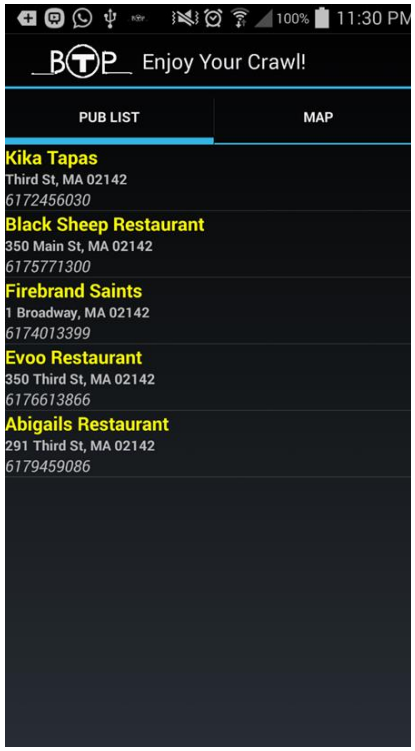
ISSN: 2473-3857
v2 n4053

Team Alpha – Pub Crawl List

Web Application

Android Application





ISSN: 2473-3857
v2 n4053

Team Beta – Start Screen

Web Application

Android Application

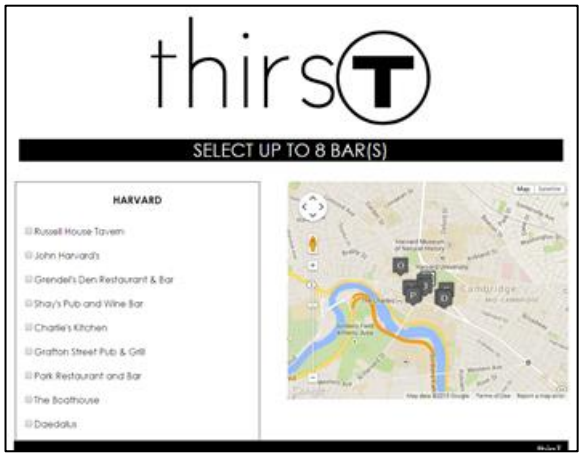
ISSN: 2473-3857
v2 n4053

Team Beta - Pub Crawl List

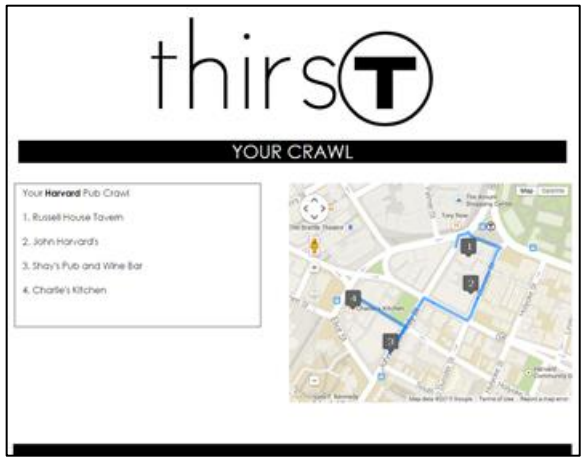
Web Application

Select Stop

Select Bars



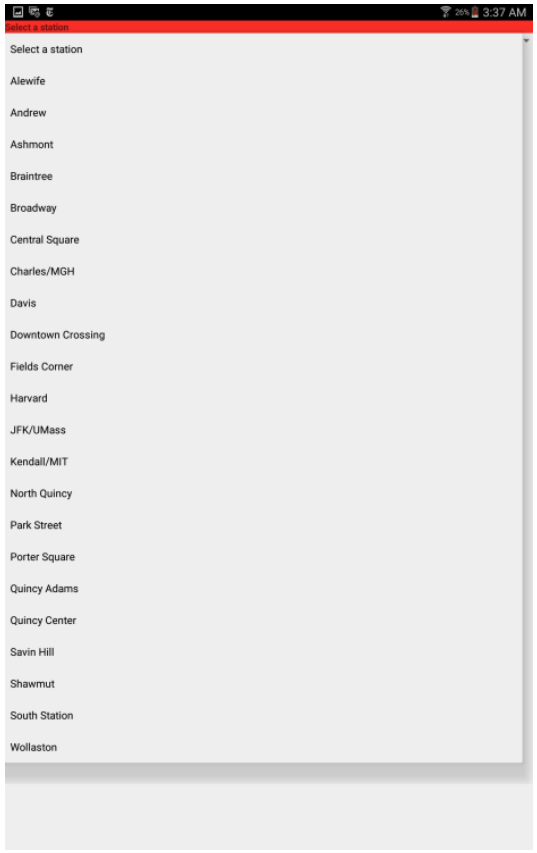
Pub Crawl List



ISSN: 2473-3857
v2 n4053

Android Application

Select Stop



Select Bars

Pub Crawl List