Developing a Task Management System
A Classroom Software Engineering Experience

Joo Tan, Jake Betts, Tyler Lance, Adam Whittaker and David Yocum
Department of Computer Science & Information Technology, Kutztown University, Kutztown, U.S.A.

Keywords: Agile Development, Configuration Management, Project Management, Risk Management, Team Communication, Testing, Global Collaboration.

Abstract: Software Engineering requires team collaboration from all project teams as one organized group. At Kutztown University in USA, students in a capstone software engineering course sequence work in project teams to gather and understand requirements, redesign, develop and test a system. In this paper, we explain the software engineering process followed by six project teams while developing the system. The teams ran into many problems during implementation. We discuss the different kind of issues encountered during the process. Lessons learned from this experiences are summarized so that future teams can benefit from this experience.

1 INTRODUCTION

Development of a software system requires the collective effort of many different people working together. For the system to be successfully designed and implemented, everyone involved with the project must communicate and cooperate as one cohesive unit. This paper presents a project which combines system development with software engineering practice (Liu. 2009) within a classroom setting. This synergy leverages global software engineering (Ebert et al., 2016) through collaboration between students in the United States with students at a college in Taiwan. The collaboration on this project is expected to last for three years.

We explain how students experience real world projects by working in small teams on a single large project in a two-course software engineering sequence (SE1 & SE2). Students plan, redesign and develop a moderately complex system that is based on research conducted by the course instructor. Furthermore, they are exposed to distributed software engineering through global collaboration with distant teams in Taiwan. This is the first year of a three-year collaborative effort with distant teams. The project teams follow the SCRUM (Mahnic, 2012) agile development method while redesigning an existing system to make it more comprehensive. Since communication is a key element of the project, we discuss issues associated with it during development. In addition, we elaborate on lessons learned during the project and provide suggestions on how to mitigate the problems encountered.

2 PERSONAL ASSISTANT WEB APP SYSTEM (iPAWS)

The course instructor has directed considerable research on using web technologies to create web applications. An existing web application system, named interactive personal assistant web application system, or iPAWS, helps individuals with autism perform simple everyday tasks. The iPAWS system (Tan, 2017) was previously field tested for usability and effectiveness. Feedback from completed field trials as well as discussion with people from industry (Goodwill, 2017) about the system was largely positive. However, a major flaw in the system's design was discovered during field trials. The system also lacked some features which would make it more useful for users. Consequently, we decided to redesign iPAWS and implement it to meet the needs for a different set of end users. Year one of the project involves complete redesign of the Task Manager portion of TMS. This paper describes our experience in this project.

Tan, J., Betts, J., Lance, T., Whittaker, A. and Yocum, D.
Developing a Task Management System - A Classroom Software Engineering Experience.
DOI: 10.5220/0006834402050212
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 205-212
ISBN: 978-989-758-320-9

3 TASK MANAGEMENT SYSTEM (TMS)

3.1 Why TMS?

Task Management System (or TMS) was envisioned as the improved version of iPAWS. TMS consists of two major subsystems: a backend named *Task Manager* that is used by supervisors and an administrator as well as a frontend named *Task Assistant* which is accessed by eventual end users, namely seniors. Communication between Task Manager and Task Assistant with the server is accomplished using Web APIs (MDN, 2018). The overall organization of TMS can be seen in Figure 1.

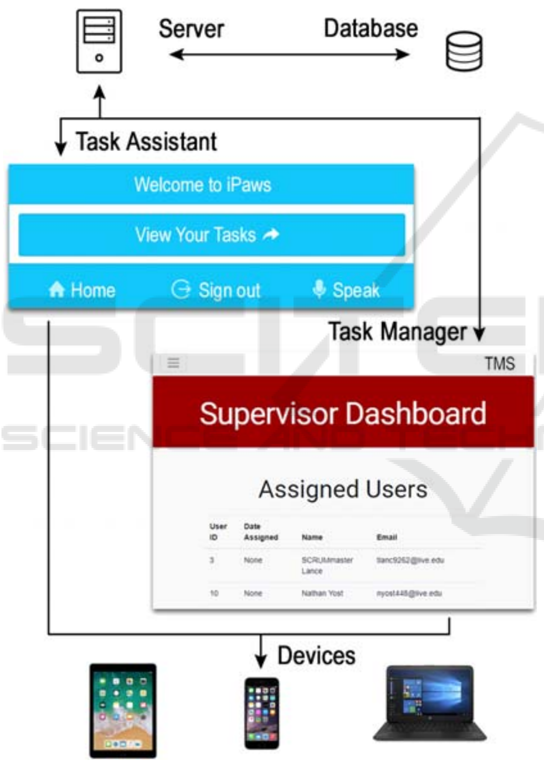
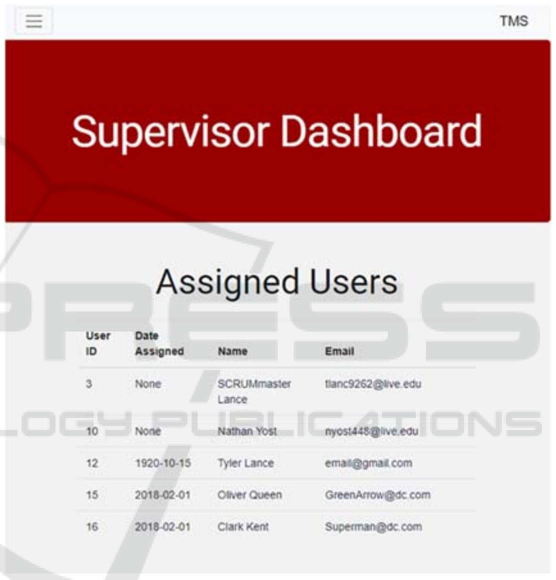


Figure 1: Organization of TMS.

TMS uses the scripting language Python and the Flask micro framework for its design and development. New major features to be implemented in TMS include creating a library where tasks can be checked in and out by supervisors, adding a speech synthesis (voice-to-text) feature for faster task creation and management, and making the system suitable for use on tablet-size computers by supervisors. These new features make TMS a more comprehensive and solid system for its users.

3.2 Refactoring the System

Task Assistant, iPAWS's current frontend runs well on mobile devices. It has been field tested but still lacked some useful features. Trying to implement new features into the existing system was harder than first thought, so a redesign of the system was decided. One unappealing feature of iPAWS extend mostly to the backend Task Manager interface since it can only be used effectively from a desktop computer. When the window is scaled to fit on a smaller device, all the buttons are brought closely together. This design flaw makes using the interface on smaller devices unfavourable. The full size redesigned dashboard for supervisors, usable on mobile tablets, can be seen in Figure 2.


TMS

Supervisor Dashboard

Assigned Users

User ID	Date Assigned	Name	Email
3	None	SCRUMmaster Lance	tlanc9262@live.edu
10	None	Nathan Yost	nyost448@live.edu
12	1920-10-15	Tyler Lance	email@gmail.com
15	2018-02-01	Oliver Queen	GreenArrow@dc.com
16	2018-02-01	Clark Kent	Superman@dc.com

Figure 2: The new TMS Dashboard.

Based on field trial results, four major reasons called for a total redesign of iPAWS. First, Task Manager is not *mobile* friendly as it limits the supervisor to using a laptop or personal computer when creating and managing tasks. The new TMS will let supervisors use iPads or larger smartphones in their work, thus enabling mobility (Wasserman, 2010). Second, a new *library* feature is being implemented so that supervisors can check in new tasks or check out existing tasks for modification. Therefore, the library allows a supervisor to reuse tasks that have already been created by other supervisors. This saves the supervisor both time and effort in their work. Third, a *voice-to-text* feature accelerates the process that a supervisor needs to spend while creating and managing tasks. This feature is especially useful if a supervisor must work on many tasks. Amazon's Alexa Skills Kit (ASK, 2017) is an example service that lets a developer build an application using VUI (Voice User Interface) leveraging Alexa system. Finally, TMS shifts focus from categories to *tasks*. Previously, tasks were based on categories. The emphasis on tasks fixes a design flaw which would have made extending existing code arduous and refactoring difficult.

4 DEVELOPMENT PROCESS

In this section, we explain the structure of student teams, the work that they did, tools they utilized during the project, and how they collaborated locally as well as with distant teams during development.

4.1 Team Structure

From the first week of class in SE1, students were divided into six different project teams. Organization of the student teams and their responsibilities in the project are as follows:

- Dev A: Task Management functionality
- Dev B: Library, TMS Dashboard, User Management
- User Interface Team: Design of User Interface for all pages in TMS
- Database Team: Database Design, Web APIs, Reports
- Quality Assurance (QA) Team: Overall QA, Test Plan/Specification, Client Contact
- Project Management Team: Gantt Charts, Risk Management, Modification Request Report (MRR), Survey Management

With the exception of the QA team, each project team consists of three students. For the purpose of implementation, all functional requirements in the software requirements specification (SRS) were classified into three categories of priority: 1, 2, 3. Features that are essential to Task Manager are considered priority 1 functionality. The other two categories contains functionality that enhance the usability or performance of TMS.

4.2 Agile Development

Project teams follow the SCRUM agile development methodology (Moore and Lopes, 2012) in three sprints, each of two-week duration. Each sprint is time-boxed with a set of functionality determined by individual project teams. Implementation and testing of TMS began in late January 2018 and will be complete by early April 2018. Deployment of TMS follow shortly with field trials scheduled for mid-April 2018. Feedback will be collected through surveys to determine both the correctness and effectiveness of TMS.

Initially, PHP was going to be the programming language used on the Apache development platform for the revamped TMS. After talking to several experts in industry and doing additional research, the development teams decided to move to Flask which is a framework that runs on top of Python. Flask is a micro-framework (Flask, 2018) for building REST APIs; though lightweight, it does not lack in power. Flask has a large extended package library that can be utilized to meet the requirements of the project and allows for faster development. Python generally has thorough and extensive documentation with libraries for almost any situation. Python is expressive and concise, with a philosophy of having one best Pythonic way of doing things, which encourages consistency. Finally, Python is a popular programing language in industry as well as becoming a language used in high school beginning programming course (CodeHS, 2018).

4.3 Configuration Management

Version control was configured using Git and accessed from Github. Each team's work branched off the master repository so there was no interference between working code and code still in development. This also made it so there was no interference between individual teams. Branches were then created off each team's branch for code that needed to be unit tested. Towards the end of every sprint, each team's branch were merged into a single branch for QA testing as well as for the next upcoming sprint. This assured that each team worked on accurate code that was completed in the previous sprint.

Besides Github, three other environments were created for the project's code to reside. Git was installed in each environment which allowed teams to work right off their branches. These environments were all hosted on the university's network:

- tmst.kutztown.edu: This virtual machine was created specifically for coding and unit testing. Port numbers were assigned to each team to allow access to this environment.
- tms.kutztown.edu/tms: This virtual machine was created for deployment purposes only. Once testing is completed by the QA team, the working code will be moved here. The final delivery of TMS will be located here as well.

- tms.kutztown.edu/tms2: After each team's branch has been merged into a singular branch, the code would be copied into this environment for system testing by the QA team. This environment is a directory within the tms virtual machine. Having yet another virtual machine specifically for black box testing did not seem necessary.

4.4 Project Management & Tracking

Project management is an important aspect of the TMS project. The Project Management (PM) team consists of three people: a team leader and two other members. Each project manager was given two teams to manage. They made daily contact with the teams to gather regular progress reports from them, which were updated in the Gantt charts.

Project Management was also in charge of the Risk Management Plan. This document tracks possible risks that can affect successful completion of the project and includes how they will be mitigated. One big challenge that arose was that some students had a hard time taking instructions from another student.

One major tenet of agile development is that the client should be available most of the time. This turned out to be a major plus for the project since the course instructor also assumed the role of client for the project, which meant he was readily available. At first, meeting with the client was a bit of a challenge for students. However, after meeting several times it started to become second nature. E-mail and Slack (Slack, 2018) became primary tools for communication with the client.

The final responsibility of the PM team was to develop the survey management page. Surveys are created to gather feedback from end users after they complete a task. The data collected is useful for evaluating the usefulness and effectiveness of TMS. Feedback data will be analysed and used to make necessary future enhancements to TMS.

4.5 Team Communication

Communication was of utmost importance on the TMS project, considering that there are six teams working on the same system. The following methods of communication were agreed upon at the beginning of the semester:

- Slack: A workspace was created for all students to collaborate. Slack includes a pinning feature that allows important messages to be viewed at all times. The direct message feature was useful since it allowed private messages to be sent between classmates separately from the general chat.
- Google Drive: A shared folder in Google Drive was created which allowed all teams to have access to public documents. Documents common used by each team are placed here.
- Discord: Team leaders agreed to use Discord for communication between themselves. Slack and Discord are similar but team leaders had agreed on using Discord because the UI was familiar to them.

There were complications throughout all teams with communication. The most apparent one was that teams believed they could do their part of the project to completion without relying on the support of other teams. This led to issues during the beginning of the project but they were slowly mended in later sprints. From this lack of communication, issues arose such as the integration of development teams' code with the UI team's code. Students also were not speaking up when problems arose that affected not only their team's section, but the entire project as well. Reasons for not speaking could be that they were shy to do so or they are afraid to hurt someone's feeling about their work. Communication on Slack was not always efficient due to students' unwillingness to check the general chat on a daily basis.

4.6 Testing and Defect Tracking

Software defects (or bugs) encountered during system implementation and unit testing by developers and during system testing by the QA team are recorded and tracked in a modification request report (MRR). Developers are assigned to fix defects according to severity level priorities.

The modification request board (MRB) was created to review each defect listed in the MRR. One representative from each team was included in the MRB. A review of defects entered into the MRR was held during each class meeting to ascertain each defect's proper severity level as well as its impact on the project. The MRB members then reported issues discussed during the MRB meeting back to their respective teams.

4.7 Team Evaluation

Prior to the start of each sprint during implementation, each project team had to submit a list of functional requirements that they were to complete for that particular sprint. At the end of a sprint, each team was then evaluated based on the following criteria:

- Completion of stated requirements that they planned to complete
- Number of defects found and their severity levels

During implementation of the first sprint, many students were hesitant to enter modification requests (MRs) into the MRR. They felt that it look bad on them if MRs were entered and assigned to them. The course instructor informed them that the inverse was true; i.e., problems found when entered as MRs actually showed that they were doing their work. A good portion of MRs written during sprint 1 were related to issues arising from communication and problems with development environment. A lot of defects found in sprint 2 were discovered when code were integrated among all the teams.

4.8 Global Collaboration

One major aspect of the project was designation of group members in another country to work with each project team at our university. Paired groups of students from two different countries must collaborate in both SE1 and SE2. One classroom is located in Kutztown, Pennsylvania U.S.A., while the other classroom is in Taoyuan, Taiwan. This collaboration led to occurrences of numerous issues [Noll] both expected and unexpected.

The first expected issue that occurred was the twelve-hour time difference between the two collaborating schools. This difference in time caused numerous schedule conflicts and was ultimately the hardest issue to overcome. Another expected issue that occurred was a cultural barrier (Deshpande et al., 2010) in the collaboration. This exists because Americans and Taiwanese have many different cultural values. For example, it is common for Americans to be forthcoming and direct while many Taiwanese tend to be reserved and withholding in nature. These cultural differences had to be considered during group collaborations. A third expected issue that occurred was a language barrier. Most Americans only speak English while the national language of Taiwan is Mandarin Chinese, although many can speak fluent English. A final issue that we knew before collaboration began was the fact that the two universities have different starting dates for the Fall semester. In SE1, Kutztown University in the U.S. started three weeks earlier than Ming Chuan University in Taiwan. In SE2, the difference was five weeks. Although this staggered difference was known beforehand, it still created a bit of nervousness and anxiety for Kutztown students.

Several unexpected issues also arose during the collaboration. The first issue that occurred was a difference in the use of communication systems. The technology systems regularly used by American students are Facebook, Discord, and electronic mail (e-mail), while popular systems used in Taiwan are Line, Facebook, and e-mail. The problem with using e-mail as a medium to communicate between groups is that it does not allow for instantaneous collaboration between team members. The second issue that occurred was miscommunication between the two classes. Some of the American teams received delayed e-mail responses from their counterpart groups in Taiwan. As a result, contact was delayed for several days on a number of occasions. Third, global collaboration can cause friction because the “stronger” students would have to wait or train their “weaker” counterparts. This friction somewhat weakened the effectiveness of a team. Fourth, different holidays are celebrated in each country. This was initially unknown to the other party. Consequently, working schedules became complicated without prior knowledge or arrangements between teams.

5 ISSUES ENCOUNTERED

Numerous issues were encountered during the entire process of the project. The major ones are listed and some general issues are then elaborated.

- All – in general, all project teams had little or no prior experience with writing technical documents. Fortunately, the course instructor provided templates for them to look at. Many student have not written technical documents before, so this was a great experience for them.
- All - lack of experience with Git at the beginning of implementation was a major issue. As time progressed through the project and team members became more comfortable with the software, this problem was lessened. On more than one occasion, development team B accidentally merged their code with development A’s code and vice versa in git. These caused a little setback and delay in coding.
- Dev A – all team members had no prior experience with querying database; there was also some confusion with populating data on user interface forms in concert with what the UI team were doing.
- Dev B – they encountered a possible show-stopping issue with creating sessions and

querying database from Python; as a result, they needed to use SQLAlchemy (SQL, 2018), an Object Relational Mapper (ORM). Some team members had to learn Python as well.

- UI – the team created base layout UI pages for the development teams; however, they had to wait for development teams to do unit testing; when it came time to merge them, the pages did not conform to each other. Again, this was due to a lack of communication.
- QA – the team had no test environment set up for sprint 1 system testing; they had a tough time with writing test cases without the actual user interface implemented; eventually, they had to test in the development test environment instead of their own test environment. This was finally resolved in sprint 2.
- DB – the team had to learn how to create Web APIs (Sohan et al., 2015); (Wittern et al., 2017) for the purpose of passing data between Task Manager and the database as well as between Task Assistant and the database. The database schema needed to be updated many times due to changes in tables and fields when issues became clear during implementation.
- PM – initially, the PM team was not tracking activities properly due to a lack of experience; as a result, the Gantt charts were a mess. Furthermore, risks were not monitored nor updated on a regular basis.
- Environment: all project teams encountered issues with using virtual machine to access the school's resources; this was mainly due to permissions not opened to people outside the school's firewall. As a backup plan, the course instructor created an account on Pythonanywhere (Pythonanywhere, 2018), a platform as a service on the cloud, in the event that the problem persisted. Luckily, this was resolved once the IRB approved the project.

Development Team B noticed a critical flaw going into sprint 1. The creation of session management functionalities was close to impossible using standard SQL queries in Python/Flask. Team B could not figure out the proper mechanics to implement it correctly using queries, which resulted in an Object Relational Mapping (ORM) Python library being used. The ORM used was called SQLAlchemy, which provided a complete session management functionality. Other teams were originally not fond of using forms to interact with the database, as SQL querying was already understood by them.

The database team encountered two major obstacles for their part of redesigning the system. The first obstacle was to convert the Entity Relationship (ER) diagram into a relational database. This was not an issue in itself, but time was a constraining factor as the development teams needed a database implemented before they are able to start implementation. Team database met a week before sprint 1 to convert the ER diagram into a relational schema. After the schema was created, the database team began creating the tables in MySQL. By implementing the database earlier than the start of sprint 1, the development teams were able to begin their work without being held back from a partial database implementation.

Another major issue encountered was that tasks were sorted by categories on iPAWS whereas TMS utilizes keywords for sorting tasks. Both these methods were fairly synonymous and the database team was able to be correct this discrepancy by treating keywords as a sort of category.

Pythonanywhere was created because Kutztown University did not allow TMS to be hosted on the Information Technology's (IT) network without an approval of the Institutional Review Board (IRB). The project proposal had to be submitted by the course instructor for approval. Since project approval was not guaranteed, there needed to be a second option on where to host TMS if the university would not grant permission. With the project already under development and no way to test TMS the development teams had to come up with a solution. They looked into a number of possible options for hosting the website for development: Microsoft Azure, Amazon, AWS, DigitalOcean, Google Cloud Platform, Heroku, and Pythonanywhere. The development teams decided on hosting the TMS website on Pythonanywhere because it allows standard free web hosting and setting up of a MySQL database. It can be used directly with github for fast deployment. The course instructor paid for an account in Pythonanywhere to allow the project teams to do their work. Pythonanywhere made testing the website and database connections very simple with a runnable browser version of bash. It also allowed the development teams to test their code along the way.

6 LESSONS LEARNED

The project teams encountered many issues during the process of developing TMS. As a result, the course instructor and project teams learned many valuable lessons along the way. In this section, we offer suggestions on how to mitigate some of the issues that were encountered. The following list explains some of the major lessons discovered:

- A lack of communication among team members, both within and between teams, was THE major obstacle to successful completion of assigned tasks. Make sure that students communicate from the beginning of the project; the course instructor can assign a portion of the overall course grade for this, if necessary. Keep emphasizing the importance of communicating with each other all the time. Using Slack as a medium of communication helped some, but not all students were getting on Slack daily to check progress on the project.
- Start early on creating the environment for the project teams to implement their work. In our two-course sequence (SE1 & SE2), we focused on requirements and design in SE1 and did not start setting up the environment until SE2. This turned out to be a little late as it took several weeks for the project teams to get everything set up for development in SE2. We suggest that the environment be worked on earlier near the end of SE1.
- Make sure that everyone involved with the project knows how the system being implemented works. During implementation of TMS, it was discovered that some students had a vague idea of the organization of TMS and how it was designed. They only knew their own part of the project that they were working on and did not understand the overall TMS when integrated as a whole.
- While following the agile SCRUM methodology, we maintained listing of issues encountered and defects found in the MRR. This turned out to be very useful for tracking issues and problems throughout the entire development process. We also recommend having a MRB to review the MRR on a weekly basis, at the minimum.
- Project teams need to communicate more with the client during design and implementation. On many instances, project teams made assumptions about how something should work without checking with the client. This practice can cause the system to not work as expected or force them to rework their part of the system later on to correct it.
- Some students regard collaboration with distant teams in Taiwan as a waste of time. Yet, this is now common in industry where collaboration is done with global partners or clients. Many of the students are seniors in our major program and thus are ready to graduate. When potential employers see and commented on the collaborative effort on the TMS project, the students realized the importance of this collaboration. We suggest that external clients or distant project teams be incorporated into the course, if possible.
- Some development teams underestimated the amount of work that goes into developing a moderately complex system such as TMS. They were somewhat overwhelmed by the implementation of functionality during the sprints.
- For the purpose of course planning by the instructor, there are several things to remember. First, it is important to cover detail design in SE1 so that students have a good idea of what to do when they start implementation in SE2. Make that the students communicate on work that are required between project teams. An example pertains to forms that one development team used for task management and the user interface forms created by the UI team. They did not match. Second, the environment for development should be configured towards the end of SE1 before SE2 begins. Due to permission issues imposed by the school's IT department, project teams had a lot of problems with virtual machines accessing the site from off campus. Third, make sure that everyone participates within each project team. Try to catch "idlers" early, otherwise the other students' load within a team can become too much to handle. Fourth, have the students learn the programming language of choice for implementation early, starting in SE1. Last but not least, stress the importance of communication to students. It is one of the most critical factor towards the success of a project. This includes communication between project teams. It cannot be emphasized enough. Having a common communication medium such as Slack helps alleviate some of the issues.

Communication was a huge factor in making sure important information got passed around to every team member. Each team was responsible for working together with other teams to develop TMS. As issues arose it was important that issues and breakthroughs were reported in the MRR, so every team was aware of issues that exist the project. Checking Slack, Discord, and the Google Drive on a regular basis was key to having better performance both within and between teams.

7 CONCLUSIONS

This paper explained how students in a two-course capstone software engineering sequence worked in small project teams to design and implement a moderately complex system. The project teams faced numerous issues and obstacles during the process and learned many valuable lessons about teamwork, communication, and collaboration. This is important before they embark onto jobs after graduation.

We also discussed the major problems/issues encountered by the project teams, deliberated on the lessons learned by students and instructor, and offered suggestions on how to mitigate these issues in the future on similar projects.

Many students felt that working on a project team for a large system such as TMS was stressful and overwhelming at times. Quite often students tend to work within a team and not communicate with other teams until problems arose. However, students found the project rewarding after system integration of each team's work and seeing TMS work as a whole. They feel a great sense of accomplishment when talking to potential employers about the TMS project. Furthermore, employers were immensely impressed with our students when they hear of the real world project collaboration and teamwork that were accomplished in the software engineering capstone sequence.

REFERENCES

Amazon's Alexa Skills Kit. Retrieved March 1, 2017 from https://developer.amazon.com/alexa-skills-kit.

CodeHS. Introduction to Computer Science in Python. Retrieved March 5, 2018 from https://codehs.com/info/curriculum/intropython.

Deshpande, S., Richardson, I., Casey, V., Beecham, S. 2010. Culture in global software development – strength or weakness, International Conference on Global Software Engineering, pp67-76.

Ding, D. 2017. A case study for teaching students agile and scrum in Capstone course, *Journal of Computing Sciences in Colleges*, Vol. 32, no. 5, pp95-101.

Ebert, C., Kuhrmann, M., Prikladnicki, R. 2016. Global software engineering: An industry perspective, *IEEE Software*, Vol. 33, no. 1, pp105-108.

Flask: Full Stack Python, Retrieved February 20, 2018 from https://www.fullstackpython.com/flask.html.

Hossain, E., Babar, M.A., Paik, H., 2009. Using Scrum in Global Software Development: A Systematic Literature Review. *International Conference on Global Software Engineering (ICGSE)*.

Goodwill Keystone Area. Retrieved November 14, 2017 from https://www.yourgoodwill.org/.

Liu, J. 2009. Combination of Research and Teaching in Software Engineering Education, *International Conference on Information Engineering (ICIE).*

Mahnic, V., 2012. A capstone course on Agile Software Development Using Scrum. *IEEE Transactions on Education*. Vol. 55, Issue 1, 99-106.

Moore, R., Lopes, J., 2011. Teaching Agile Software Development: A Case Study. In *IEEE Transactions on Education*. Vol. 54, Issue 2, 273-276.

Noll, J., Beecham, B., and Richardson, I. 2011. Global software development and collaboration: barriers and solutions. ACM Inroads 1, 3, pp66-78. Retrieved October 5, 2017 from http://dx.doi.org/10.1145/1835428.1835445

Pythonanywhere: Host, run, and code Python in the cloud. https://www.pythonanywhere.com/. Last retrieved 3/9/18.

Slack: Workspace for Project Communication. https://slack.com/. Last retrieved 3/12/18.

Sohan, S.M., Anslow, C., Maurer, F., 2015. A Case Study of Web API Evolution. IEEE World Congress on Services.

SQL Alchemy: The Python SQL Toolkit and Object Relational Mapper. https://www.sqlalchemy.org/. Last retrieved March 10, 2018.

Tan, J., 2017. Developing an Interactive Web-based System to Assist the Elderly. *International Conference on Computer Science Education: Innovation and Technology* (CSEIT),

Wasserman, A.I. 2010. Software engineering issues for mobile application development. *Proceedings of the FSE/SDP workshop on Future of Software Engineering Research*. ACM, New York, NY, USA, pp397-400.

MDN Web Docs: Web APIs https://developer.mozilla.org/en-US/docs/Web/API. Last retrieved 3/10/18.

Wittern, E., Ying, A.T.T., Zheng, Y. 2017. Software engineering issues for mobile application development. *International Workshop on API Usage and Evolution* (WAPI). ACM, New York, NY, USA, pp397-400.