# The Quest to Learn Web Technologies: Using Google AppEngine to Create an Online Adventure Game

**George M. Wyner**
Boston University School of Management
gwyner@bu.edu

**Benjamin Lubin**
Boston University School of Management
blubin@bu.edu

**ABSTRACT**

Web application development is becomingly increasingly relevant to a business education in general and an information systems education in particular. This accompanies an increasing perception among the general public that it is important to understand how the web works and to be familiar with the technologies that are used to build web applications, with some professionals viewing this as an essential component of their skill set. In this paper we describe our approach to teaching this topic to masters students concurrently pursuing an MBA and a Master of Science in Information Systems. We provided student teams with a set of tools for building a text-based adventure game using Google AppEngine and the Python programming language. The initial version of this exercise was conducted in the spring of 2012 with 44 students. In this paper we describe the exercise and summarize the teaching materials we have developed.

**Keywords**

Management Pedagogy, Curriculum Development, Introduction to Programming, Web Design, Web Application Development.

## INTRODUCTION

Web application development, however arcane it might have once appeared is becomingly increasingly relevant to a business education in general and an information systems education in particular. There is an increasing interest by educators in making "computational thinking" available to all college graduates as part of a notion of 21st century literacy (Stross, 2012). This accompanies an increasing perception among the general public that it is important to understand how the web works and to be familiar with the technologies that are used to build web applications, with some professionals viewing this as an essential component of their skill set (Wortham, 2012). Information systems (IS) faculty have expanded the IS curriculum to include courses on web development, and more recently web application development.

In this paper we report on a new approach to teaching this topic to masters students pursuing both an MBA and a Master of Science in Information Systems. The approach is intended to address the challenge of teaching this complex topic to students with a limited technical background and to do so in an extremely brief time frame (less than three weeks). In developing this approach we seek to give students the experience of creating a web application and thereby a chance to understand an important set of web technologies — HTML, CSS, database, web server, and application server — without plunging them into an overwhelming morass of technical detail.

The approach we are developing provides student teams with a set of tools for building a text-based adventure game on Google AppEngine using the Python programming language together with a web application framework. This approach gives students a chance to learn the basic architecture of a web application while developing something that reflects their own imagination.

The framework is being developed for deployment in May/June of 2012. In this research-in-progress paper we describe the key design decisions we have made in developing this approach as well as giving a preview of the teaching materials we will be employing. A demonstration game developed using our framework has been made available on the web and the code we are developing will be made available to the IS community under the MIT open source license (www.opensource.org/licenses/MIT).

The paper is organized as follows: We begin with a review of research on the teaching of web development to IS students as well as research on the potential benefits of using games to teach computer science concepts. We then describe the teaching context and our specific goals for this material, as well as our strategy for this approach and some of the design decisions we have made. We present a brief summary of the teaching materials we are developing and conclude with some thoughts about the potential benefits of this approach and how we will proceed with this research.

## LITERATURE REVIEW

As the Internet has become increasingly relevant to business, information systems faculty have responded by bringing web development into the IS curriculum. For example, Lim reports on the evolution of a web development course early in the new millennium (Lim, 2002).

This new focus is now reflected in model curricula: The model curriculum for graduate degree programs in IS was last updated in 2006, at which point it was observed that changes in technology in recent years have been "even greater" than changes in management practices with the result being a need to provide attention to the use of Internet and web services (Gorgone, Gray, Stohr, Valacich and Wigand, 2006). The most recent revision to the undergraduate information systems model curriculum also acknowledged that "modeling and development platforms for the web environment have become a core part of IS development" (Topi, Kaiser, Sipior, Wright, Valacich, Nunamaker Jr. and De Vreede, 2007).

However, teaching web application development brings with it significant challenges, especially the need to provide working knowledge of a large number of technologies (Yue and Ding, 2004). This challenge is increased when the students being taught have limited technical background, as in our program. For us the challenge is further complicated by the compressed time frame in which we are expected to deliver this learning experience. We have consequently developed a different approach than that adopted in existing curricula. Two key differences concern the choice of project and the choice of infrastructure:

*Choice of project*. Hauser et al. taught web development using a business case involving currency trading (Hauser, Olsen and Fadel, 2010). An advantage of their approach is that this same case could be used across courses such as database and user interface design in order to give students an integrated experience. We do not have the time to unpack such a complex business example, but in moving to a simpler project we run the risk of a less compelling student experience. We have chosen to compensate for this in two ways: first by choosing the inherently compelling domain of game design and second by giving students the freedom to develop their own unique project.

*Choice of infrastructure*. Yap and Loebbecke were able to teach web application development to students with a wide range of technical backgrounds (Yap and Loebbecke, 2005). During their semester-long course the deployment of the application infrastructure is a significant activity for students. By using Google AppEngine we are able to eliminate nearly all the time and effort required for infrastructure deployment and configuration. This is essential given our brief time frame and has the benefit of giving students an exposure to developing apps "in the cloud" (although we need to consider the possibility that this may result in less depth in student understanding of the elements of this infrastructure, given that they have less need to work with the infrastructure itself).

As noted above we have chosen to use an online game as the project for our students. Information Systems and computer science faculty have identified numerous benefits from using game design in the classroom. Jones observes that implementing a game allows students to apply many computer science concepts in an engaging manner(Jones, 2000). Becker experienced a number of additional benefits from assigning game development to students: Students know how the game should work and are motivated to get it to work correctly. Students want to show off their games and in some cases go beyond what is required in order to make their games even better. Finally, students "cross a significant perceptual boundary: they become the 'creators'. They go from experiencing the 'magic' to being the 'magicians'" (Becker, 2001).

Again, for us there is a particular challenge associated with the brief time frame and limited technical background of our students. Even in semester-long courses game design can be a difficult topic. Connolly had students design graphical games in an emerging technologies course as a way to learn .NET application development and web services techniques. Connolly reported strong student interest in this project but that mastering the technical challenges was difficult for many students (Connolly, 2005). Our approach is to omit interactive graphics from the project and go back to a simpler kind of online game: the text-based adventure game. We describe this approach in more detail below.

## TEACHING GOALS

The context in which we are teaching web application development is a two-year dual degree program in which students receive an MBA and a Master of Science in Information Systems. This material will be delivered as part of a technology intensive held as a summer session between the first and second year of the program. This summer session is a follow on to a one-week intensive, held in the winter of the first year, in which students are exposed to basic programming concepts and complete elementary programming exercises in Python. The goal of this initial "winter intensive" is to provide students with initial exposure to the basic architecture of modern computing devices with a specific focus on the role played by modularity and the interfaces between modules, culminating in an exploration of the strategic importance of the application programming interface (API).

The goal for the "summer intensive" is to provide students with enough familiarity with key technologies and technical concepts to permit them to engage in meaningful conversations with technologists and to understand and act on the strategic potential of information technologies. The summer intensives are divided into two separate phases. The initial two-week session covers (1) systems architecture concepts (how computers function, how programs are created, and the components and functions of a modern operating system) and (2) an introduction to databases including entity relationship diagrams and SQL. The third week covers networks, security, and web technologies.

Initially web technologies were taught in a relatively hands off manner with an explanation of enabling technologies such as HTML, CSS, and web services. Students were given a few simple exercises such as creating a simple web page or using web services to pull song lyrics from an online database and then feed them through Google's translator. However, these were brief exercises that did not give students a chance to explore the technologies in any depth. This lack of an extensive hands-on exposure to web technologies contrasts with successful lab exercises provided during the winter intensive and the systems architecture and database courses. We decided to bring this hands on approach to web technologies as well by creating a team project for the third week of the summer intensives. We believe a web application is an ideal such project for several reasons:

- Web applications are a familiar part of students' daily digital experience.
- Web applications are an important arena in which companies compete and innovate.
- Web applications bring together the entire range of topics which students are studying during the summer intensives, including programming, database, and the full range of web technologies including HTML, CSS, and web services.
- Web applications embody important architectural concepts which students have been studying, including modularity, the notion of an application stack, as well as APIs.
- Creating a working web application would be a significant accomplishment and one that students could show to friends and family, as well as discuss with potential employers.

In 2010 we made our first attempt at a web application exercise using Ruby on Rails (rubyonrails.org). We found that it required significant effort on the part of students just to set up the web hosting and to configure Rails and its database connection. As a consequence, most of the available time for the project was spent on setup, and not on the more rewarding and pedagogically important tasks of designing and building the actual application.

In 2011, we instead had students create simple applications using Google AppEngine (developers.google.com/appengine). This worked better for several reasons:

- AppEngine is hosted at no cost by Google. All students need to do is sign up for an AppEngine account with Google and download a software development kit (SDK). There is no need to configure a hosting environment.
- The AppEngine SDK is platform independent and can be installed easily on student laptops, which allows students to try out their apps on their own computer and then upload those apps to Google's servers.
- AppEngine is integrated into the development environment students are already using. In recent years we have switched from Ruby to Python as our teaching language, primarily because we like its extremely simple and intuitive syntax, and the option to avoid object oriented features where we do not need them. Our students use Eclipse (www.eclipse.org) as their programming environment. To make Eclipse work with Python, students install the PyDev plugin (pydev.org), which includes support for Google AppEngine. This means that students are able to run AppEngine applications on their laptop and also deploy them to the Google platform, all from within the Eclipse IDE.
- The "webapp" framework , which is included in the AppEngine SDK, is much simpler than Rails, Django (www.djangoproject.com), and other web application frameworks. While webapp offers less functionality out of the box to developers, it also offers a gentler learning curve for students. Given the limited time students have to work in this environment, we find the limited functionality a worthwhile tradeoff for the increased understandability Google has built in.

Building on an example provided by Chuck Severance (2009), our first pass at providing students with an AppEngine experience consisted of a number guessing game. An example of the game can be found at example.com (actual URL omitted for anonymity during review process). A simple version of the game can be implemented by a student without the need to understand much about how the different layers of a web application interact. As students learn more concepts they can add functionality to the game, moving from a version in which the correct number is "hard coded" into the application, to a version where the number is generated randomly and stored in a user session. Finally students can use CSS to add styling and interact with a database to store high scores.

While students were able to add a few embellishments to the number guessing game (colorful pages, use of images, refinements to the user interface) the basic structure of the game is limited and does not leave much room for creativity. We want to give students a chance to build a more elaborate application and to put more of themselves into the app.

In particular, we are interested in creating an experience for students that is similar to and reflects the success of the curriculum we have developed for the winter intensive course. As part of this course, we have student teams each develop a restaurant concept and then develop a restaurant information system in Python using an API that we provide. This exercise gives students a chance to develop a substantial bit of functionality and to think creatively about what kind of restaurant they want and how its customers and employees should interact with its information system. In other words, we give students a chance to exercise technical skills in an interesting business context.

## STRATEGY

We briefly considered moving the restaurant design exercise to the AppEngine platform, but the repetition of what would be the same design exercise on a different platform seemed unlikely to be appealing to students. This led us to look for a new domain which: (1) would be immediately familiar to students, allowing them to plunge into a design conversation with few preliminaries, and (2) would allow for lots of creativity and variation. Ultimately we hit upon the idea of having students create their own adventure games. Not only are students almost certainly familiar with online games, but the content of the game would be wide open for creative invention.

By "adventure game" we mean the original text-based games in which a player finds him or herself in a cave or house or other environment and can issue commands to move from place to place. In each location in the game world the user may encounter objects and perhaps creatures and obstacles. The user can add objects to an inventory and can do things with the objects. For example, if a user has a lamp in her inventory she could turn the lamp on, which would in turn allow her a view of what might otherwise be a dark chamber. Possessing a key might allow a user to open a door to gain access to new parts of the game world.

Early examples of such adventure games include Colossal Cave (Jerz, 2007) and Zork (Lebling, Blank and Anderson, 1979). Originally such games were purely text based: locations were described with text only, and users entered commands on a command line. One might argue that games such as Myst (Miller and Miller, 1993) represent the logical evolution of such games into graphical form in which the game world is rendered as a three dimensional virtual environment and users employ the mouse to explore the world and interact with its objects.

In our case staying with the text based approach makes sense because, while less flashy, it builds on student understanding of how to manipulate text in Python and does not require a knowledge of animation techniques and more complex user interfaces. Instead, the interface is a fairly straightforward use of basic HTML form elements. We do provide students the ability to include graphics such as images of rooms and the objects in them. This gives students a chance to learn how to use images in a dynamic web page and introduces options for visual creativity.

The game display includes a description of the current location. While students are free to create any setting they want we will generally use the term "room" to refer to each location in the game world. The display also includes a list of items in the room and the items currently in the player's inventory. To avoid the complexity of a command line parser, the game will display available actions as a series of buttons. The buttons will also allow the player to drop items from inventory, add items to inventory, navigate in various directions, and interact with objects he or she has in inventory. An example of this interface, taken from the demonstration version of the game currently under development is shown in Figure 1.

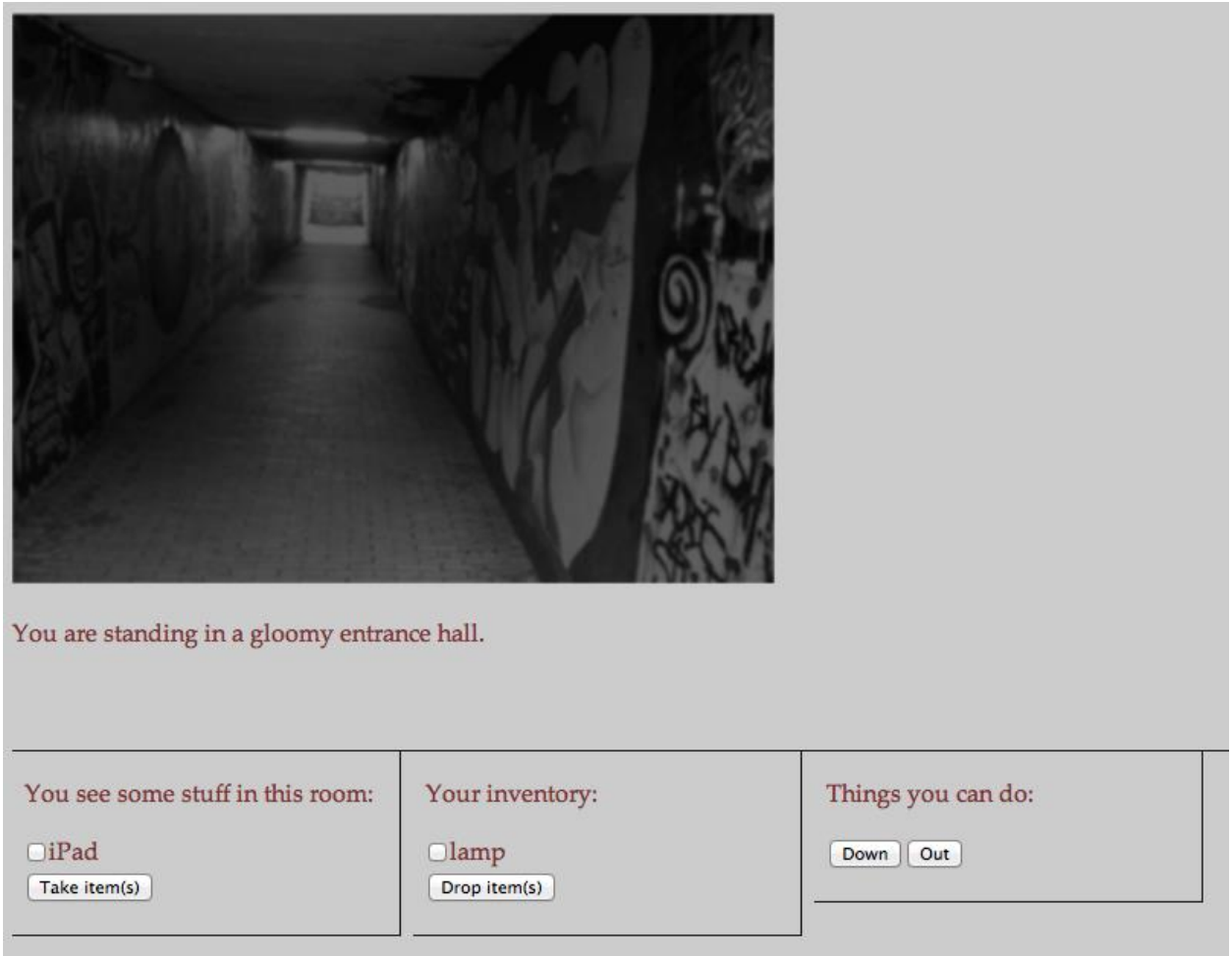


**Figure 1. Sample Game Interface**

A key design decision we have made is to avoid the use of a database. It would seem natural to store the game world in a database, since this would allow for changes to the game without changing the code. We elected not to do this for two reasons: First, we wanted to avoid the need for students to learn how to use the AppEngine database features. Second, we wanted to have students working in the code rather than simply editing content in a database. As a result of this decision, the structure of the game world is stored in the code itself. The state of the game for any player is stored in a session cookie.

Our plan is to give students a framework on which they can build a game, along with a simple game demonstrating how the framework can be used. Unlike in our restaurant API exercise, in which the API consisted of a set of functions, in this framework we provide students with an object model which they will need to build on. The object model consists of base classes for rooms, inventory items, and actions. Students add rooms, actions, and inventory items by creating subclasses of these base classes and modifying their attributes and methods. A consequence of this design decision is that we need to give students a brief introduction to the concept of class and object so they are not lost when navigating the code we provide for them.

In addition to the set of objects we provide to students, the game framework makes use of the webapp framework included with AppEngine. This framework includes a set of classes that allow a developer to define how URLs are mapped to a set of *handlers*, each of which handles get and post requests. We also make use of the template framework included with AppEngine, which allows a developer to generate web pages that include a mix of static html and dynamic content.

## TEACHING MATERIALS

Prior to this course students have been exposed to the basics of Python programming and the concept of functions and APIs. Students have all installed the Eclipse IDE with the PyDev extension which allows them to use Eclipse to develop and run Python programs. We thus begin by having the students download and install the AppEngine SDK on their personal laptops.

Prior to the start of the game development exercise, students will have a chance to revisit their Python programming skills and will be briefly exposed to the concept of an object. Lectures focus on the key building blocks for web applications including HTML, CSS, and the architecture of web applications including the database and web server components. The lectures are self-contained but students are strongly encouraged to rely on *Using Google AppEngine* as a reference (Severance, 2009). This book provides students with concise discussions of each of these technologies in an AppEngine context.

Students then walk through creating an AppEngine account and building a simple application of their own. We use the number-guessing game (described above) for this initial introduction to programming web applications, a role for which it is well suited. By the time students have finished this exercise, they have a basic familiarity with the AppEngine environment and have been exposed to the key concept of mapping each URL to a handler object that responds to get and post requests.

While students are rekindling their programming skills and learning the basics of web application programming, we divide the class into teams and brief them on the game assignment. Students are given access to a simple demonstration game that they can use both as a reference and as a basis for their own creations. The demonstration game is currently under development in parallel with the development of the framework itself. The current version of the game can be found at msmba-ae-game.appspot.com. We also have each team create a project repository on Google Code (code.google.com), which enables them to share their code with each other.

Each team is expected to come up with a basic design for their own game, including the theme, a set of rooms (or other locations), objects in the various rooms, and a set of puzzles which involve collecting objects and using them to overcome obstacles. Documentation on how to create rooms, objects, and object behaviors, and how to manage stateful and conditional outcomes is made available to students.

As the course progresses students are given large blocks of time to work together in the classroom to develop their game. Faculty and teaching assistants are available to help resolve technical issues. At the conclusion of the course, each team presents its game concept and gives a demonstration of the game. All of the games created are then publicly accessible through AppEngine.

We have found this team format to be effective in the past for several reasons. The number of teams is small enough so that each team can present its work to the entire class and faculty, which creates considerable motivation to be creative and to deliver a working application. Further, by including students with a range of technical abilities on each team we allow students to learn from each other. We hope that this student interaction may be even more fruitful in the web application context, as students can contribute in different ways to the application, not only by programming but also by creating images, designing web pages, writing the text of the game, and designing the game logic.

## CONCLUSION

In this paper we have described an approach to teaching students about web technologies by giving them an opportunity to build a unique web application that reflects their own creative vision. The approach we describe is in its early stages of development and our goal in this paper is to describe the strategy we have adopted and the reasoning behind the approach. Our plan is to beta-test these teaching materials with second year MBA students who have already taken the summer intensive. The first delivery of this exercise will take place in May and June of this year. Feedback from this first set of students will provide us with a preliminary sense of the effectiveness of this approach and how it might be modified going forward.

## REFERENCES

1. Becker, K. (2001) Teaching with games: The minesweeper and asteroids experience, *Journal of Computing Sciences in Colleges*, 17, 2, 23-33.
2. Connolly, R. (2005) A funny thing happened on the way to the form: Using game development and web services in an emerging technology course, *Information Systems Education Journal*, 38, 3,.
3. Gorgone, J. T., Gray, P., Stohr, E. A., Valacich, J. S. and Wigand, R. T. (2006) MSIS 2006: Model curriculum and guidelines for graduate degree programs in information systems, *Communications of AIS*, 2006, 17, 2-75.
4. Hauser, K., Olsen, D. and Fadel, K. (2010) An integrated approach to teaching web development, *The Review of Business Information Systems*, 14, 1, 43-59.
5. Jerz, D. G. (2007) Somewhere nearby is colossal cave: Examining will crowther's original 'adventure' ' in code and in kentucky, *Digital Humanities Quarterly*, 1, 2,.

6. Jones, R. M. (2000) Design and implementation of computer games: A capstone course for undergraduate computer science education, *ACM SIGCSE Bulletin*, 32, 1, 260-264.
7. Lebling, P. D., Blank, M. S. and Anderson, T. A. (1979) Special feature zork: A computerized fantasy simulation game, *Computer*, 12, 4, 51-59.
8. Lim, B. B. L. (2002) Teaching web development technologies: Past, present, and (near) future, *Journal of Information Systems Education*, 13, 2, 177-123.
9. Miller, R. and Miller, R. (1993) *Myst*, Cyan,.
10. Severance, C. R. (2009) *Using Google App Engine*, O'Reilly Media, Sebastopol, CA. Web.
11. Stross, R. (2012) Computer Science for Non-Majors Takes Many Forms. *The New York Times*, March 31.
12. Topi, H., Kaiser, K. M., Sipior, J. C., Wright, R. T., Valacich, J. S., Nunamaker Jr., J. F. and De Vreede, G. -J. (2007) Revising the IS model curriculum: Rethinking the approach and the process, *Communications of AIS*, 2007, 20, 728-740.
13. Wortham, J. (2012) A Surge in Learning the Language of the Internet. *The New York Times*, March 27.
14. Yap, A. Y. and Loebbecke, C. (2005) A system for teaching MIS and MBA students to deploy a scalable database-driven web architecture for B2C e-commerce, *Information Systems Education Journal*, 3, 7, 1-20.
15. Yue, K. B. and Ding, W. (2004) Design and evolution of an undergraduate course on web application development, in *ACM SIGCSE Bulletin*, 22-26.