

**University of Arkansas – CSCE Department**

**Capstone II – Final Report – Fall 2019**

# LecturePlus

# Vladimir Sabado, Joseph Gorman, Cristhian G. Espinosa, Jagmeet Wander

## Abstract

Students’ questions play an important role in meaningful learning and scientific inquiry. They are a potential resource for both teaching and learning science. There are many reasons students may be hesitant to speak in class. For example, introversion, shyness, English as second language, culture differences and previous bad/embarrassing experience etc. The main objective of our LecturePlus app is to enable students to ask questions anonymously during the lecture.

## Problem

Most students have been in a classroom setting where they wish they hadn’t asked a question or added to the discussion, maybe the answer was obvious or the question seemed unrelated - “out in left field”, so to speak. It could also be that they wish they had asked a question about something they didn’t understand but did not because of reasons listed below. It takes strength to raise your hand in class and ask question. Let's examine reasons why students do not ask questions in class.

Shyness: shyness can be bitterly difficult for many students.

Fear of appearing uninformed: Asking a question shows that one does not understand a concept. No one wants to look foolish or be judged for how much they may or may not know.

Difficulty forming the question: Other times a student may struggle to formulate the question. Students in the process of learning English may be hesitant and unclear about how to structure the sentence grammatically.

Disabilities: A student may be deaf and providing a way for such a student to ask a question without attempting to speak can be of great benefit, that is assuming the professor does not know sign language or an interpreter for the student is not present.

## 2.0 Objective

The objective of this project is to:

* To create a way for students to ask a lecturer a question anonymously through technology.
* To Encourage classroom participation.
* To Allow more interaction between the speaker (instructor) and the audience (students).
* To make communication easier in the classroom.

## 3.0 Background

### 3.1 Key Concepts

*Client*: A *client* is a program that uses services that other programs provide. Client programs typically handle user interactions and often request data or initiate some data modification on behalf of a user.

*Server:*A *server* provide services to the clients. Server functions often require some resource management, in which a server synchronizes and manages access to the resource and responds to client requests with either data or status update.

*Bootstrap*: open source toolkit and powerful front-end framework for developing with HTML, CSS, and JS. We can use it to quickly prototype ideas or build entire app with Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

*JavaScript*: Interpreted programming language that follows ECMAScript specification. JavaScript is the most widely deployed language in the world.

### 3.2 Related Work

Piazza is a platform designed to create communication between professors and students more efficient. It is designed to reduce email clutter and provides several ways to poll students and allows students to ask questions anonymously. Our product is different in that it is designed for communication during a lecture. Piazza’s services are not free, but LecturePlus is free to use. LecturePlus also has a much simpler user interface.

## 4.0 Design

### 4.1 Requirements

This project requires to develop a web application that will be used by students and teachers to help determine how concepts are being understood by the class during and after a lecture.

In order to make this possible, the following will be required:

1) A server

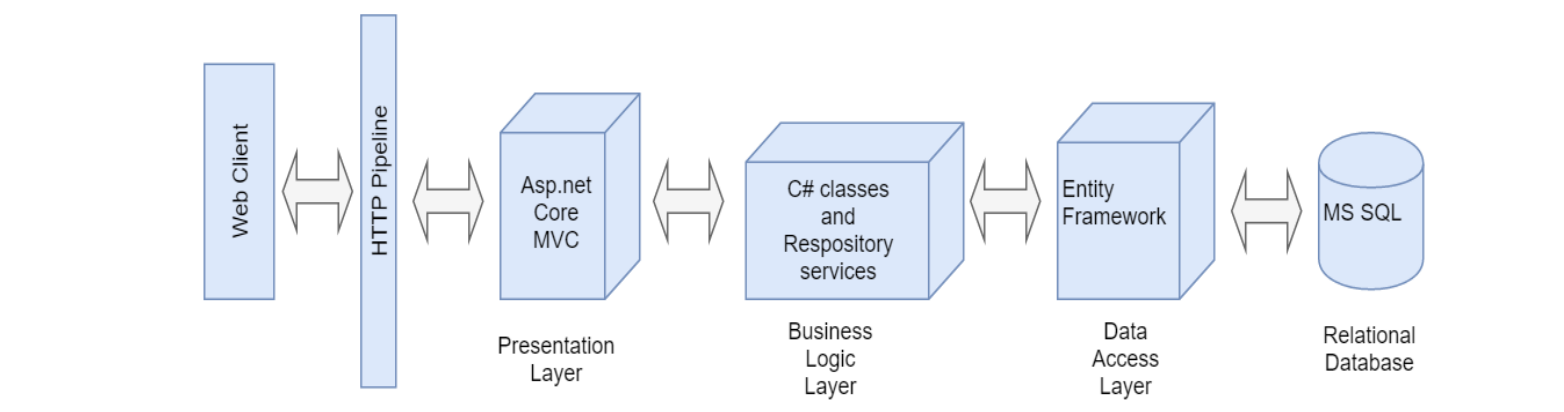
2) The clients

3) A connection between the server and the clients

4) Computing devices

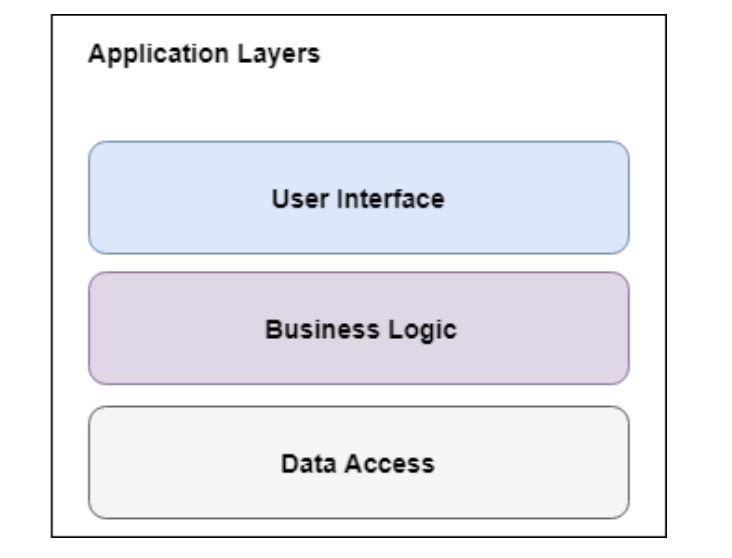
### 4.2 Architecture

The application implements layered architecture which is one of the most widely used architecture for web application development. Figure below shows the logical architecture of LecturePlus.

.

Since it follows separation of concern principal, even if the code base grows, the structure of the application stays organized. In this approach, the complete logic of the application is included in a single project which is compiled to a single assembly and finally deployed as a single unit. The project contains behavior of the application, presentation, business logic, services and data access logic.

Figure below shows layered pattern used for the application which includes data access layer, business logic and user interface layer.



Data Access Layer: Data access layer contains functionality of creating, returning, updating and deleting entities in the database. Entity framework core is used in this layer which works as object relational mapping tool to perform data access functionalities from SQL server.

Business Logic: Business logic layer contains the core functionality of the application and encapsulates the relevant business logic. In this layer, class library will be used to hold all the custom logic for the application.

Presentation Layer: Presentation layer contains the user-oriented functionality responsible for managing user interaction with the system. This layer will use MVC views to provide user interface.

**Implementation:**

* **Back-End**
  + .NET Core 2.1 MVC
  + Code-First Entity Framework
  + REST API
  + Data Transfer Objects
  + JQuery
  + Azure SQL Database
* **Front-End**
  + Razor Pages
  + JavaScript
  + Kendo UI
  + Bootstrap Template

**Deployment:**

* Azure Web App
* Azure DevOps pipeline

**Code Version Control:**

* Azure DevOps

**Server Implementation:**

We had a choice between Spring MVC and .Net MVC for the server of this project. Since we’re most familiar with the .Net MVC, we opted to use .Net core 2.1. Since a lot of the functionality of our website requires a REST API, we knew that a database was needed to be set in place for data storage and retrieval. This means that Entity Framework must be put in place in the server side. There are two popular approaches to this: Code-First Entity Framework and Database-First Entity Framework. Code-First involves designing the database schema through the code and performing a migration to apply the change while database-first designs the schema through the database and applies it to the code. We chose to do the Code-First approach since it’s the most effective approach for our project.

To further enforce the divide between Data Access and Business Logic, we made use of Data Transfer Objects. The idea is to retrieve information from the database as entity object, and then this object is converted into a Data Transfer Object which is returned to the API. This means that an API does not simply have access to the database; there is a middle-man that handles the motion of the data from the database to the front-end—and this is a very popular idea in software engineering.

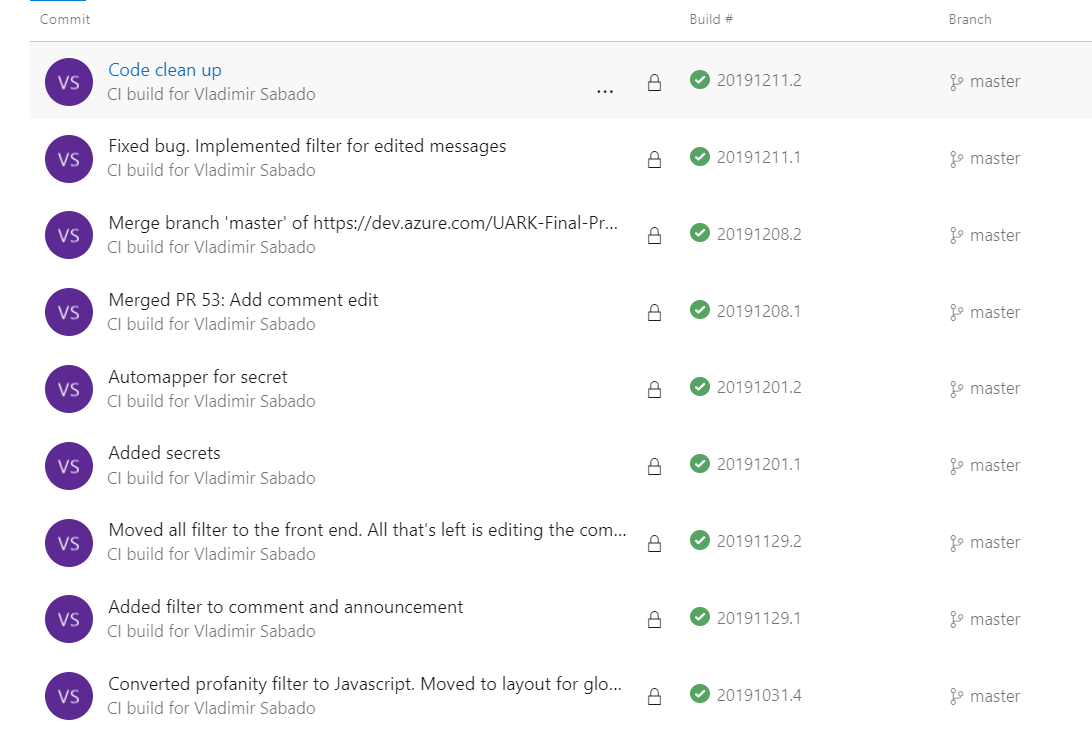
**Client Implementation:**

There are numerous options for the technology that can be used for the client. We considered using angular, react, plain old html, but we ultimately decided on razor pages because of its simplicity and effectiveness for the given project. What’s incredible about razor pages is that it essentially involves a markup language that couples with logic. For instance, I can declare a div inside of a for-loop. This is made possible because Razor Pages is a fine combination of client and server logic—in other words, there exists a controller (API) behind every single front-end page. This makes retrieving information from the database and displaying the data seamless. Personally, I enjoy working with Jquery so we have a lot of code in the project that makes use of ajax calls to retrieve or send information to an external API. It certainly wasn’t necessary with razor pages but it’s what I’m most comfortable with, so we decided to have two methods of interacting with the API—one in the server of the razor pages, and one that we explicitly created. To help with the aesthetic of the website, we made use of bootstrap and Kendo UI. Both are very easy to implement and work with, and they made a lot of differences to the appeal of the site.

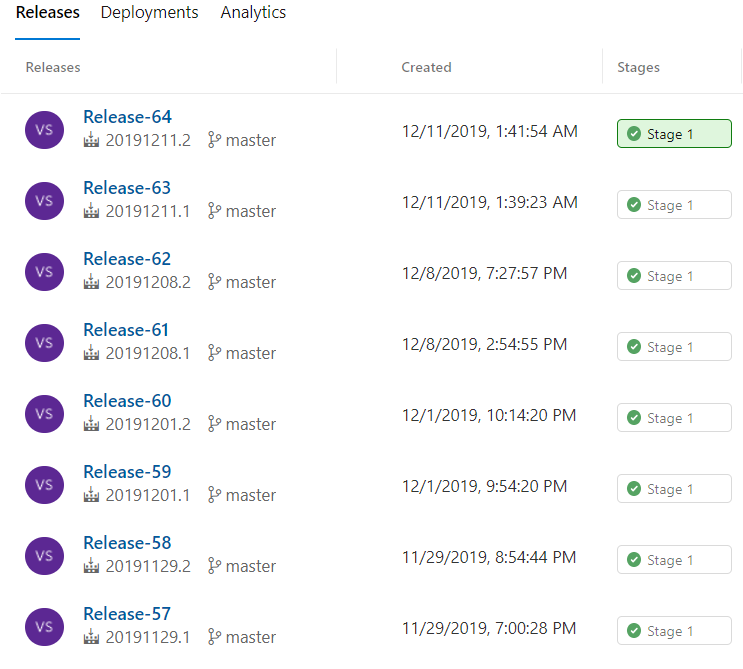
**Deployment:**

We primarily used Microsoft technologies for this project because there is a level of synergy between all their products. For project management, we made use of Azure DevOps and the Azure Portal. With just a few setups, this made deployment very simple. The idea is to have the code base hosted by a Microsoft-hosted agent aka Azure VM. When an update is made to the code, the VM re-deploys the code and new changes are shortly displayed.

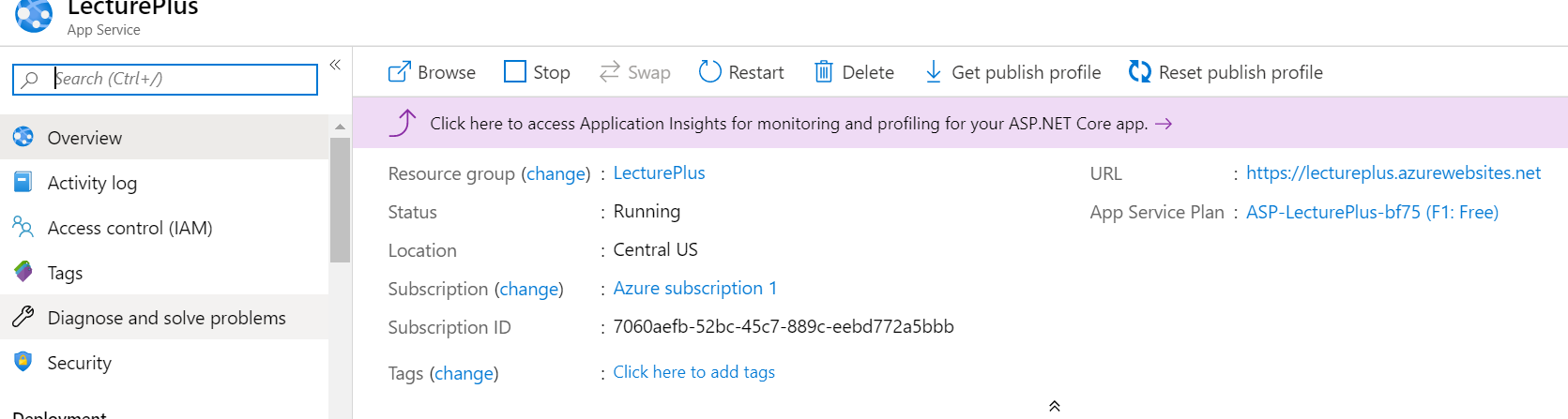
The process begins by creating a build based on the current state of the branch, which for this project was called “master”. When this branch is changed in any way by a commit, then a new build is created. Below is a list of some of the recent builds:

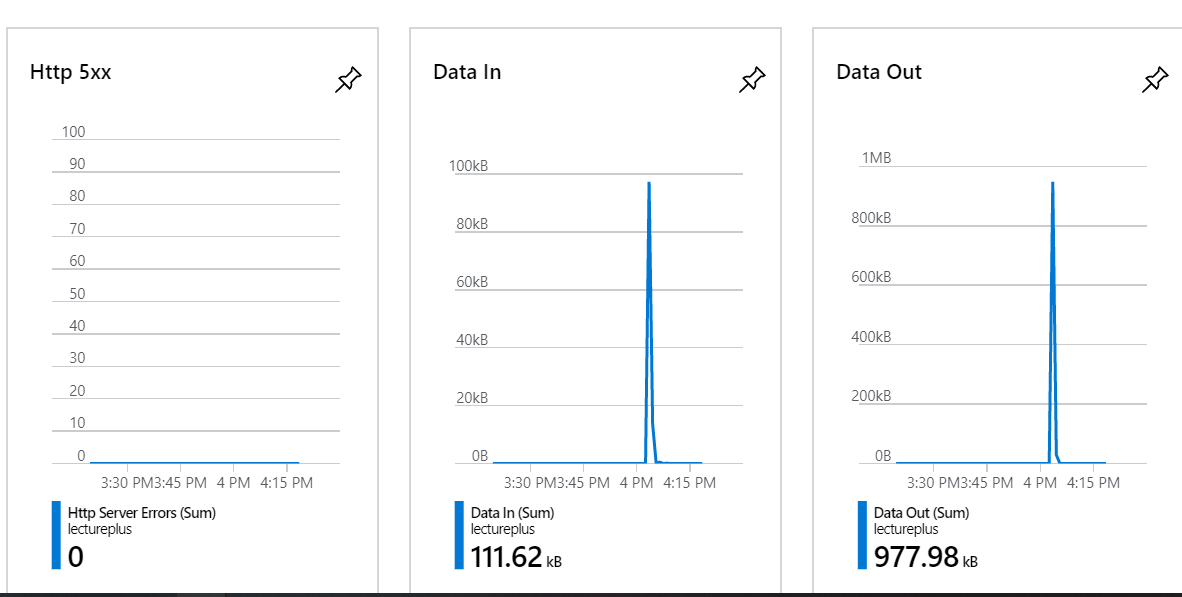


These builds are then staged for deployment, and a release is created for them. Below is a list of the most recent releases:

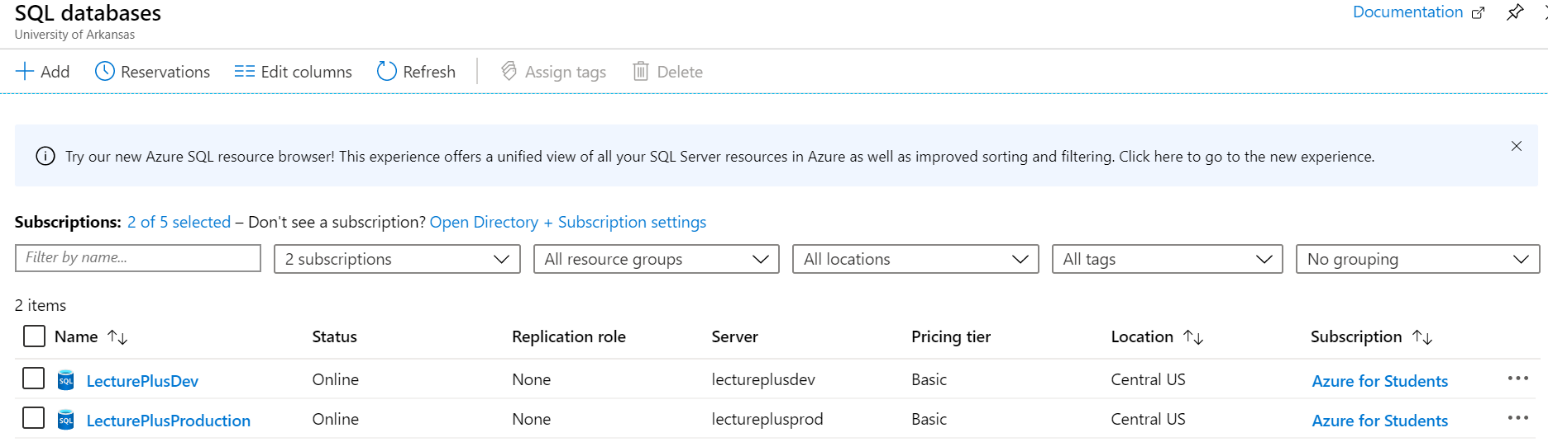


When the deployment is successful, they will appear as green check marks or red x’s otherwise. The associated URL for the web app will be where the project will be deployed to. On Azure Portal, we can view the analytics for this site including the bandwidth usage amongst other things as shown below:





**Databases:**



### 4.3 Risks

|  |  |
| --- | --- |
| **Risk** | **Risk Reduction** |
| Abusive language | Create a profanity filter. |
| Outside invasion | Only invited users can view a group. |

### 4.4 Tasks

1. Plan designs and understand client-server architecture.
2. Begin implementation of architecture.
3. Design the user interface.
4. Begin implementation of user interface.
5. Test product for any risks or errors (End to End Testing)
6. Automation Testing
7. Documenting results

**4.5 Schedule**

|  |  |
| --- | --- |
| **Tasks** | **Dates** |
| 1. Plan designs and understand client-server architecture. | 08/26/19 - 09/09/19 |
| 2. Begin implementation of architecture. | 09/09/19 - 09/23/19 |
| 3. Design the user interface. | 09/23/19 - 10/07/19 |
| 4. Begin implementation of user interface. | 10/07/19 - 10/21/19 |
| 5. Test product for any risks or errors and Automation Testing | 10/21/19 - 11/04/19 |
| 6. Document the results | 11/04/19 - 11/18/19 |

### 4.6 Deliverables

* Website link: <http://lectureplus.azurewebsites.net/>
* Source Code: <https://capstone-csce.uark.edu/springfall-2019/teams-1-5/team-4-lecture/>

## 5.0 Key Personnel

**Jagmeet Wander**– Wander is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has experience using HTML/CSS/jQuery/Bootstrap he also completed Software Engineering, Algorithms, Programming paradigms, Computer networks and Applied Cryptography. Wander has experience with UX design as he has taken few online courses and he is currently working as software developer at Walmart Labs in Bentonville.. His task will be Front End development and helping with Backend Programming also perform automation testing.

**Joseph Gorman –**Joseph is a Senior pursuing a Bachelor of Science degree in Computer Science at the University of Arkansas. He has experience using several programing languages including C++, C#, Java, Python, and PL/SQL. He worked as an intern for the software development team at LeDuff America.  He was responsible for handling the message data and implemented the profanity filter.

**Cristhian G. Espinosa S.** – Cristhian is a Senior in Computer Engineering and a minor in Mathematics. Cristhian spent a semester in Jönköping, Sweden working on Internet and cloud services and developing backend applications connecting with local Swedish companies like IKEA and Scania. Current member of ​IEEE Eta Kappa Nu (HKN), Gamma Phi Chapter. He has worked for the ​Network Enterprise Systems Team for two years which involves working with Cisco, Juniper, and Aruba network infrastructure devices. He has worked with Laravel, HTML, PHP, Node.js, bootstrap, and some other web development frameworks. Cristhian has also worked with C++, C#, Python, Java. He has studied hardware descriptive languages including Verilog and VHDL. He has developed 2D and 3D videogames using the cross-platform engine Unity. Cristhian is currently working on his Honors Thesis in deep learning, neural networks and facial recognition.

**Vladimir Sabado**– Vladimir holds a BSc in Biology, but he is currently pursuing his second BSc which is in Computer Science. He is currently a software engineer intern for Affirma Consulting where he’s working with his team to develop front-end and back-end applications to meet their client’s needs. While his language of choice is Java, he has experience working with C#, Python, HTML, JavaScript, C++, SQL, and Ruby. For this project, he will be working on the architecture, and will lead development.

* 1. **Facilities and Equipment**

Microsoft Visual Studio

Azure Devops

GitHub Desktop

Bootstrap

## 7.0 References

[1]  Piazza, ([https://piazza.com/)](https://piazza.com/) )