Deliverable 4

Class Voting System

IS436 Structured Systems Analysis and Design

11/18/2019

Arian Eidizadeh - Project manager

Cyril Fonzock - Software analyst

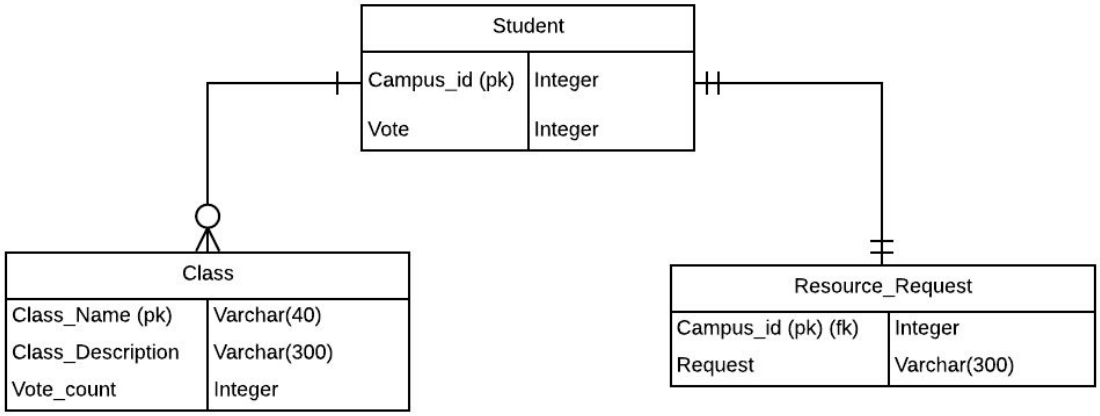
Michael Woldgerima - Developer

Leul Assamenew - System analyst

Sanee Patel - Data analyst

Yohannes Feleke - Developer

1) Develop a data model by drawing an Entity-Relationship diagram using Visio, Lucid Charts (or any drawing tool) The model will be accompanied by a textual description of entities and relationships. The diagrams should closely follow the notation. Make sure that the model is in the third normal form.



Students will be identified by campus_id (*The Student table will have campus_id of IS majors who will not be graduating a year from the current semester that the system is up*). Students will have a max of two votes. Voting is not mandatory. When a vote is cast for a class the vote_count in the Class will be incremented by 1 and student vote will be decremented by 1. When the student vote is 0, student will be able to submit the Resource Request. Each student will only be able to submit one Resource Request. Student to Resource Request is one-to-one. One student can vote at max two times, but doesn't have to vote either. Student to Class is zero-to-many(two) (*Student can vote twice or once or not at all*).

2) Develop an alternative matrix for your project. You must also provide a description of each of the alternatives that you have chosen. Each member of the team will rate the three alternatives independently. Include each individual matrix with the deliverable. The team will then come together on a consensus, creating a team matrix. Include this team matrix with the deliverable. Provide a narrative for the justification about which alternative was picked by your team.

Alternative Matrix

The alternative matrix involves the alternatives of developing a custom web application using JavaScript and HTML. Another alternative includes developing a custom phone application built by us, and the third alternative consists of obtaining purchased software and integrating it to meet the project needs. The scores and relative importance are based on the impact of the technical, economic and organizational issues.
These alternatives were also based on a consensus of multiple justifications. Justification include:

- Most Idealistic solutions to the problem
- Efficient Solution
- Realistic, and cost-effective solution
- Solution that the Director will accept.

We chose these three alternatives as our team alternative matrix based on the justifications stated above. We analyzed the various benefits, negatives, and acceptance by the Director on each alternative and rated all of the potential alternatives to end up with the Alternative Matrix below. We then rated the alternatives based on their scores and relative importance are based on the impact of the technical, economic and organizational issues.

Evaluation Criteria	Relative Importance (Weight)	Alt 1: Custom Web application using JavaScript and HTLML	Score (1-5)	Weighted Score	Alt 2: Custom Phone Application Built By Us.	Score (1-5)	Weighted Score	Alt 3: Obtain purchased software and integrating it to meet the project needs.	Score (1-5)	Weighted Score
Technical Issues:										
Access to underlying code	15	Yes, minimal	2	30	Limited	2	30	Yes, moderately possible	2	30
with current departmental technology		moderately possible			possible					
Economic Issues:										
Cost of learning the code (training, teaching)	15	$40/month	3	45	$10/month As students are used to phone applications, they will be able to easily sue an app like this.	5	75	$70/month	1	15
Reliance on additional costs for employees/vendor in maintaining the system for future enhancements	10	$20/month	3	30	Will need additional technical support to maintain the phone-voting application	5	50	Will need additional technical support to maintain the obtained software.	1	50
Organizational Issues:										
Directors ease of learning for accessing the data from the website	10	Somewhat complex	3	30	Moderate	4	40	High learning curve as it will not be a custom built product by us.	1	10
Director ease of learning the	15	Not too complex	3	30	Easily adaptable	5	50	Moderate learning curve	2	20

functionality of using the app										
Market adoption for other schools	25	Can be challenging	2	40	Easily adaptable	5	100	New to the environment	2	40
Total:	100			225			445			225

3) Design the architecture of the system by developing a matrix . In its rows, this matrix should clearly list the non-functional requirements of your project under four main categories: Operational, Performance, Security, and Cultural/Political requirements. In its columns, the matrix should include the architectural options (i.e. server-based, client-based, thin-client server, thick client server). If a particular architecture is good for any requirements put a check mark in the corresponding matrix cell. This matrix should be accompanied by a narrative that talks about why particular architectures are a good fit for particular non-functional requirements. Based on this matrix, make a decision of the system architecture and explain the justification. Mention the trade-offs and the reasoning behind your decision. After that, develop a Hardware and Software Specification. Note that your specifications can include different server and client configurations.

Matrix- Nonfunctional Requirements

The nonfunctional requirements alternative matrix covers the requirements that we will need to incorporate into our system as well as they types of architectural structures we can use. We have included requirements that are specific to the event tracking system.

Architectural Option	Operational Requirements	Performance Requirements	Security Requirements	Cultural/Political Requirements
Server-Based	✔	✔	✔	✔
Client-Based	✔	✔	✔	✔
Thin-Client Server	✔	✔	✔	✔
Thick-Client Server	✔	✔		

Operational Requirements

Compatibility and adaptability are key nonfunctional requirements that are essential in implementing our software. Integrating the system is helpful in easy maintenance of the system. From the analysis, our voting system would be the optimal choice as the compatibility is best suited when system integration is implemented. The voting system can be adaptable to the fits into the software specifications when using a thin based and thick based client system as storing information that can be easily implemented. However, a thin based client system would have the most benefits because of lower overhead.

Compatibility and adaptability are key nonfunctional requirements that are essential in deploying specialized software. Integrating the system is helpful in easy maintenance of the system. From this analysis, the thin based client system would be the optimal choice as the compatibility is best suited when system integration is implemented. The system can be adaptable to the specialized software and fits into the software specifications when using a thin based and thick based client system as storing votes and is easily implemented. However, a thin based client system would have the most benefits because of lower overhead.

Performance Requirements

Automated population would be best suited for a thick client server due to the ability for altering changes in the student attendance for each event, therefore, thick client servers can easily be more scalable to the changing capacity of the automated population. Availability and reliability are best suited for thick client-server architectures as requests can be passed on to other servers, which helps to avoid the potential issue of disrupting the interface and interaction between the client and server in the app. Response time becomes slower in server-based architectures, resulting in an increase in expenses to upgrade, which shows that a thick client-server is best for availability of the reports and reliability of the system is more easily accessible. Additionally, for the error detection requirement, a thin client server has a lower probability of error because it has fewer points of contact with the processing activity.

Security Requirements

A thin client is primarily designed to communicate with a server thus a thin client is more secure because all the data on the thin client is stored on the remote server. So if a student's device crashes or is stolen, there is no threat of losing private or valuable information.Conversely, a thick client server is less secure with limited data integrity. On the other hand, the server-based architecture is another secure system to use since all student information regarding their

demographics, year, and email addresses will be stored onto one server. With thick client there are increased security issues like information disclosure, a possibility for unauthorized access or an authentication bypass. Since all of these acts can occur without the server being notified, the high risk of security concerns would outweigh the benefits of using a thick client server.

Cultural/Political Requirements

Customization and legal requirements play major roles in choosing the right architectural option.The customization requirement is implemented well with the thin client-based architecture as the application and its elements can be customized to particular versions and adapted to certain regulations based on a user's current country and location. In addition, since a thick client architecture has security issues, the Data Protection Act legal requirement would not be met. A thick client also does not provide a lot of room for the organization to update the architectural system. A thin web-based architecture would be the optimal solution for the Data Protection Act requirement due to the reduced security threats of the thin web-based architecture.

To conclude, a thin web-based architecture is well suited for our system because of reduced security threats, easier maintenance, optimal availability and reliability, and easier implementation of specialized software scanners, JavaScript, Java and MySQL. A thin client server architecture has a lower probability of error, and improved customization. These factors of the thin web-based architecture play a key role for the Class Voting System as reduced error and customization for better and smooth functionality of the application while incorporating unique features.

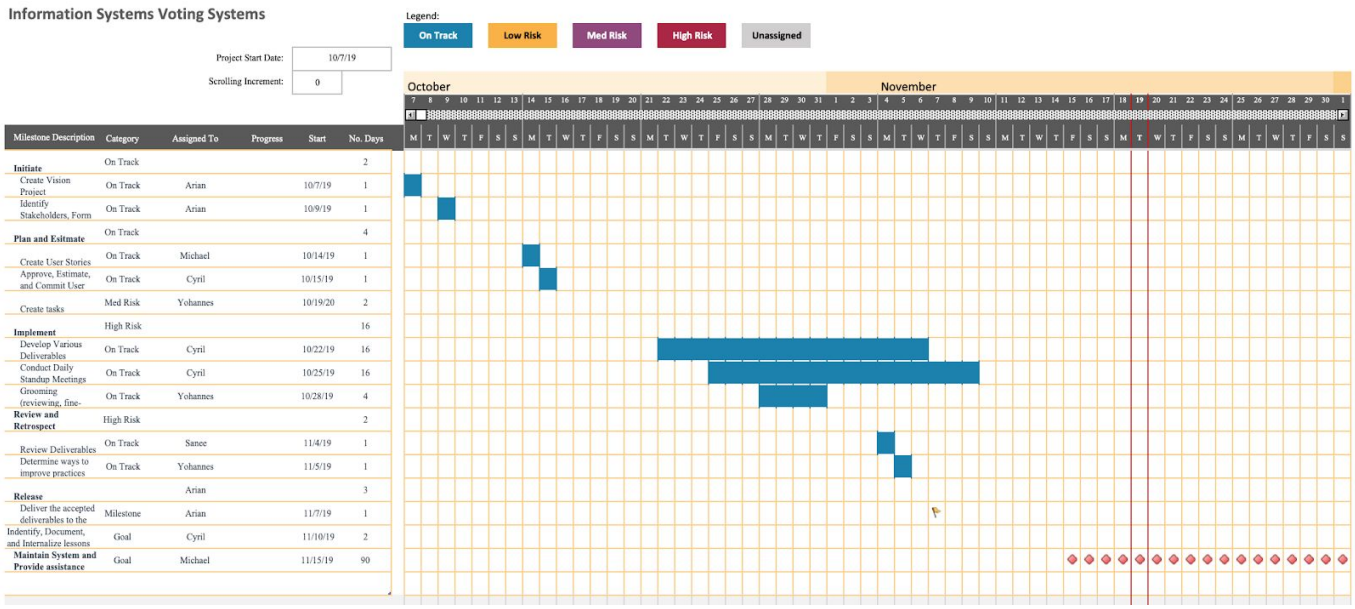
Hardware and Software Specification

Below are hardware and software specifications for our Class Voting System.AWS EC2 would be used to host our servers.

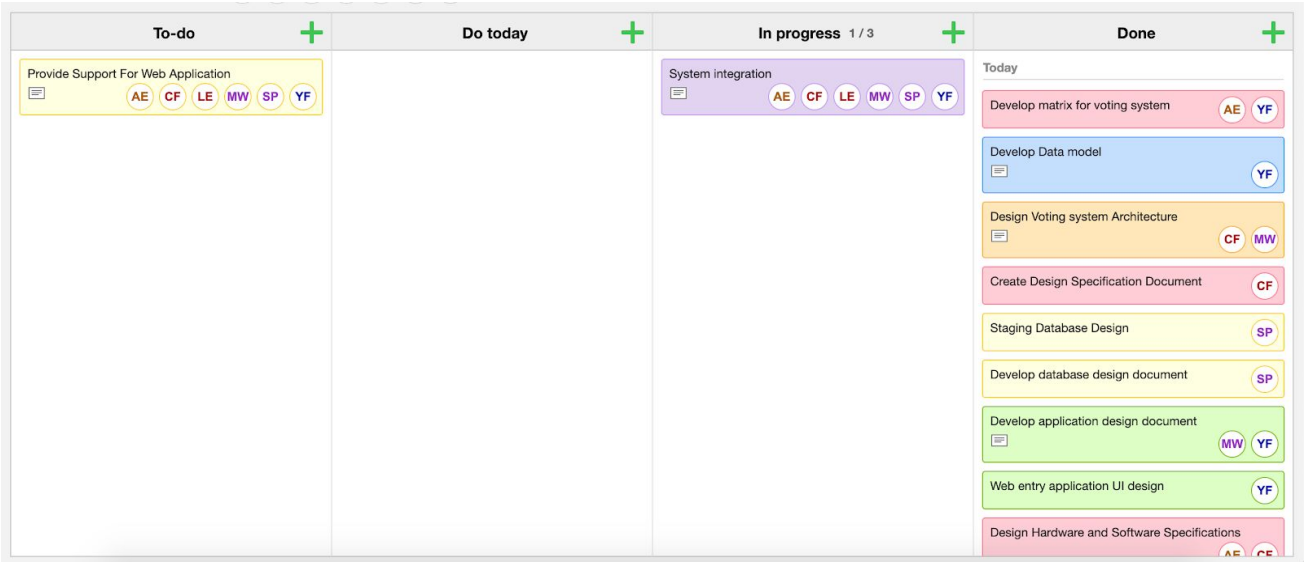
	Standard Client	Standard Web Server	Standard Application Server	Standard Database Server
Operating System	● iOS,Windows,linux	● Linux	● MacOS ● Microsoft Windows	● Windows
Special Software			● Javascript/Java/Python	● MySQL

Hardware	• Dual-Core 1 GHz ARM Cortex-A9 MPCore CPU • Dual-Core PowerVR SGX543MP2 GPU • 512MB of memory	• 8 TB disk drive • Processor CPU with over 3 GHz frequency	• 10 TB Disk Drive • Xeon E5-4600 v4	• 2 processors • Minimum of 3GHz processing speed
Network	• Always-on Broadband, preferred	• Dual 100 Mbps Ethernet	• Dual 100 Mbps Ethernet	• Dual 100 Mbps Ethernet

4) Include your updated project plan



5) On Kanban board assign the requirements to your team members



6) System Integration .Please follow the instructions provided on

https://userpages.umbc.edu/~ss12/IS436/content/groupproject/group.html

System integration is the process of connecting different sub-systems (components) into a single larger system that functions as one. With regards to software solutions, system integration is typically defined as the process of linking together various IT systems, services and/or software to enable all of them to work functionally together. Integrating data is the first step to unleashing its full potential. When companies have all their information in one place, they're able to find the most important and accurate insights within it.

Collecting more data from more students can ensure that choosing classes are as comprehensive and trustworthy as possible. Deeper data also creates opportunities to rate new things in new ways. For instance, we have feedback options in our voting system for students to give their suggestions for classes. If we get more feedback and more information, it can help Information System department to make changes and help to make decisions to create class schedule for next year.

The systems integration will take place within the umbc app. The app itself will provide an option to access the website for the class voting feature. A third party link will link to the software for the class voting system. The umbc app will have the students login with their credentials allowing us to use their credentials as a form of identity that will let us know if someone is affiliated with the University. The website can also be hosted in multiple sections of the UMBC APP including the news section, classes section and media section. Advertisements can also be included in the main page of the UMBC web page reminding students to provide their feedback for the class voting system.