



Deliverable 3 & 4

Process Modeling

Class Voting System

IS436 Structured Systems Analysis and Design

Arian Eidizadeh - Project manager

Cyril Fonzock - Software analyst

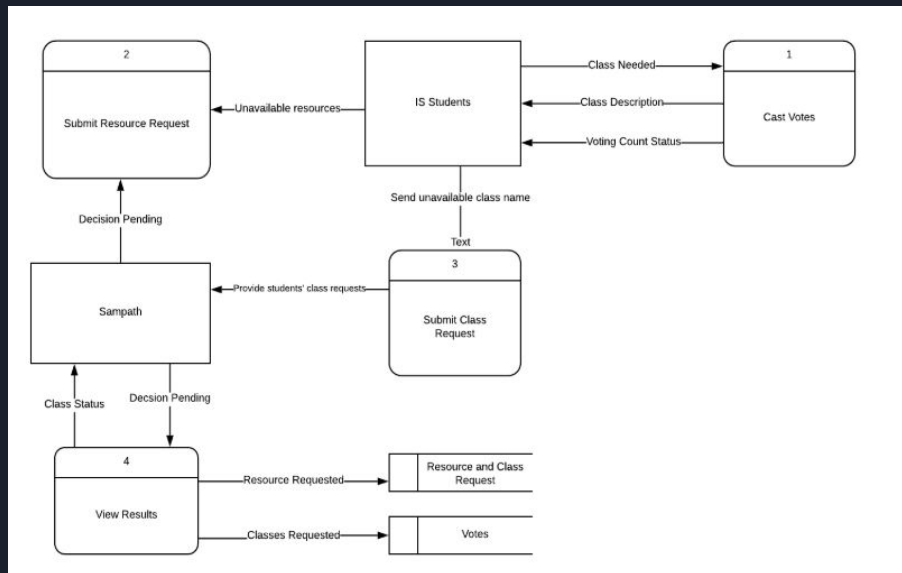
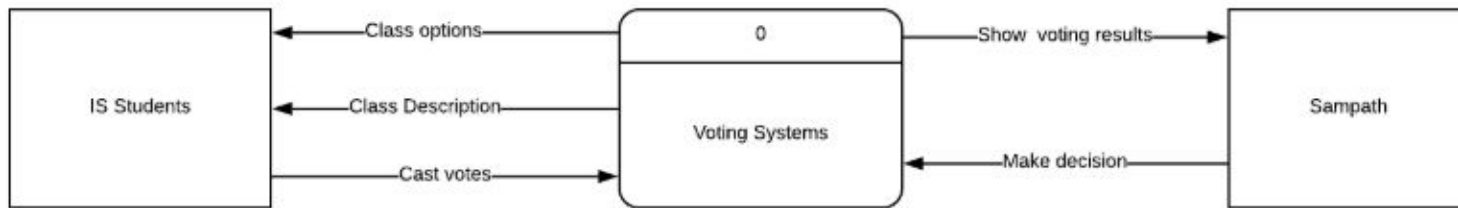
Michael Woldgerima - Developer

Leul Assamenew - System analyst

Sanee Patel - Data analyst

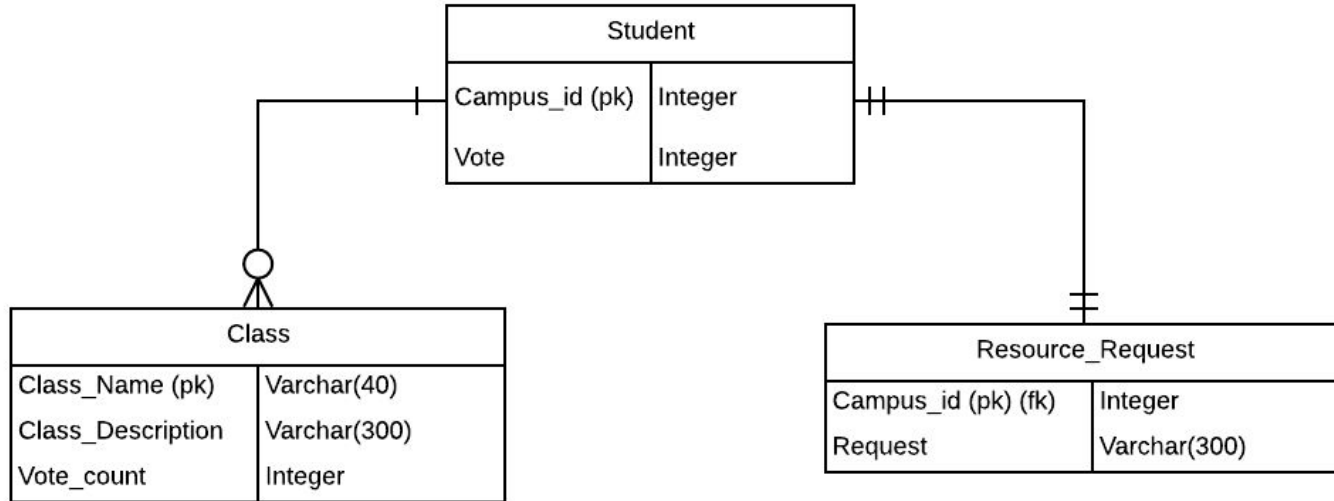
Yohannes Feleke - Developer

Process Models



New Data Model

- Students identified by campus_id
- Max of 2 votes (not mandatory)
- Votes for a class will increase vote_count by 1
And student vote decreases by 1
- 1 resource request per student





Functional Requirements (Updated)

1. Learn more about the software.
 - 1.1. The voting system itself should be an ease-of-use tool for students to vote for their 2 most desired class.
 - 1.2. Process-oriented design will be used as it can allow the system to check incoming and transfers to vote as well efficiently.
 - 1.3. The system should allow students to view degree requirements and course list while voting for classes.
2. Implementing the software.
 - 2.1. Student will log into the website using their CAMPUS ID. The website can be accessed through a web browser on any device.
 - 2.2. The student has two votes they can use and click on whatever class they would like to take.
 - 2.3. Once their ID is verified by comparing it to a valid student database, the voter can complete their ballot and securely submit their vote(s) to the web application ballot box.



Functional Requirements (Continued)

1. Logic behind the Software.
 - 1.1. In terms of our keyword analysis machine learning development- we will use ML to scan submitted sentences, put them into keywords, and push them out in a nice-clean format for the director.
 - 1.2. We will use a supervised learning, predictive module- neural based algorithm for this assignment. 80% Training and 20% test data in the alpha stage



Non-Functional (Updated)

1. Operational

- 1.1. Voting system will be a web application that can be accessed on any device (it will have to have the correct configurations).
- 1.2. Voting System should have the updated classes for the next year.
- 1.3. Compatibility with browsers on desktop, mobile, and laptops/etc should all be tested and working.

2. Performance

- 2.1. Interactions between the user and the system should not exceed 2 to 3 seconds.
- 2.2. The system downloads new status parameters instantly. Program should be making changes in real time.
- 2.3. The system should be available for use 24 hours per day, and however long the department director keeps the voting system online.
- 2.4. The system supports 250-300 simultaneous users at a time. If there is a need for more resources, we will allocate them.



Non Functional Requirements (Continued)

1. Security

- 1.1. Dr. Sampath will be the only one with direct access to the voting results. Users will not be able to see other text submissions.
- 1.2. Developers will test the software for potential bugs.
- 1.3. The system will provide the proper mitigations against XXE, SQL, XSS, and other OWASP Top 10 vulnerabilities.

2. Cultural and Political

- 2.1. Students' vote information is protected in compliance with Data protection.
- 2.2. Votes administered into the system will only be submitted from students. Teachers, faculty and others will not be allowed to vote.
- 2.3. The department director (Dr. Sampath) will authorize the start and end date of the voting system.

Alternative Matrix

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 HTML	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

Alternative Matrix Continued.

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/ve ndor 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
Organizatio nal 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



Alternative Matrix Continued.

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

Use Cases:

Use Case #1 : Cast Votes

Priority: High

Actor: UMBC Information Systems Students

Description: This use case describes how UMBC Information Systems Students will be able to cast two votes for two classes.

Trigger: When UMBC IS student clicks on class “Cast Vote” Button will be made visible

Type: External Trigger

Preconditions:

1. UMBC Information Systems Students are authenticated
2. Class Polling System is available and online
3. Classes are available and online

Normal Course:

1.0. Cast vote for classes

1. Login is authenticated ⇒ CampusID
2. Retrieval of classes is initiated ⇐ Class options
3. Classes are displayed or not (see Alternative Course 1.1) ⇐ Class details
4. Class selection redirects you to class description ⇒ ClassID
5. Student cast vote on class description page ⇒ Vote
6. Return to step 3 for second vote
7. After second vote student, page redirected to Use Case 3



Alternative Courses:

1.1. IS classes fail to be displayed

1. System display error is displayed
2. Students submit notification of error
3. Exit Use Case
4. Sampath is notified of error
5. Sampath notifies system support team

⇒ ErrorID

⇒ Select error

⇐ Error notification

⇒ System support notice


PostConditions:

1. Student votes are submitted and approved
2. Sampath is notified for display error



Lessons Learned:

- We have tested and conducted more research into what ML algorithm we will use and where the process will be set during our DFD design
 - We will use a supervised predictive neural based algorithm that is designed to recognize patterns
- We have changed how quickly our database and website update from 8 minutes to it working in real time.
- Added a use case normal course for Dr.Sampath having the ability to send results to students
- Edited various pieces of information within our functional and nonfunctional requirements.




3) Definition for each process, entity, datastore and each data flow in your diagram.

- **Class options:** All of the classes a user can take are displayed for the user to choose from and respond too accordingly. The voting system should display these classes correctly and every class should fit the schedules for the following year.
- **Class description:** Each class should have its own description that the voting system displays. The voting system should describe the professor teaching the course, the location, textbook requirements, pre-requisites and a brief summary of what is to be taught.
- **Cast votes:** This is the main function for selecting the necessary courses for the coming year. Students should be capable of selecting 2 classes for the upcoming academic year. A vote will mark with a check mark or x to signify their vote has been selected and counted in the system.



Process Definitions Continued

- **Show voting results:** Dr.Sampath should be able to view the results of the class votes. She will use this information to make knowledgeable decisions about what classes need more professors, what classes can be cut and what classes can remain the same. As a result students will be able to take all of their necessary classes towards graduation.
- **Make decision:** This is where the professor will select what classes will be made available to the students the following year. Results will either be mailed to students or/and made available to students that took the survey. This way students will know if the results they picked made an impact on what classes they wanted to take.
- **Decision Pending:** Sampath lets the system and user know that she is in the process of making her decision. It can't be rushed because the necessary data needs to be inputted first. The decision will be pending until a final decision is made.

- 
- **Class Status:** Shows the status of what classes are currently being selected by students and the current results. Dr.Sampath is who looks over the data so she would be the only one to view this. If any changes are needed to be made she can start to make early deliberations.
 - **Voting Count Status:** The count of all votes is made visible to the students to view. This is important as it lets the students know what their chances are for getting professors to support their class. If they can't get the support they might ask their friends to fill out the survey or simply become more informed of the status of their class.
 - **Resource Requested:** Makes a request for more resources whether it be more professors, programs or anything of the like. Budgets, materials, and lab utilizations will decide how much resources will be provided.
 - **Classes Requested:** Displays what classes are requested by students for the next academic year. These classes should be displayed by the system so that both sampath and students alike can view the information.



Matrix - NonFunctional Requirements

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



Operational Requirements

- Integration leads to easy maintenance of the system
- System integration is optimal for the system
- Thin based client system would have the most benefits because of lower overhead
- System can be adaptable to the specialized software and fits into the software specifications when using a thick and thin based client system as storing files and apps is easily implemented



Performance Requirements

- Automated population is best suited for a thick client server
- Thick client server is more scalable to changing population size
- Availability and reliability are best suited for thick client server architectures as requests can be passed to other servers
- Helps avoid potential issue of disruption between client server and the app
- Less errors with thin client server as there is lower probability of error
- Thick client-server is best for availability of reports and reliability of the system to become more accessible



Security Requirements

- A thin client is more secure because all the data on the thin client is stored on a remote server
- Server-based architecture provides security since all data can be stored in one server
- Thick-client has security issues, for instance unauthorized access, information disclosure.



Cultural/Political Requirements

- Customization requirement is implemented well with thin client based architecture as the application and its element can be customized for particular location.
- The Data Protection Act legal requirements would not be met with a thick-client base architecture due to security issues.
- A thin-client base architecture would be a solution for Data Protection.



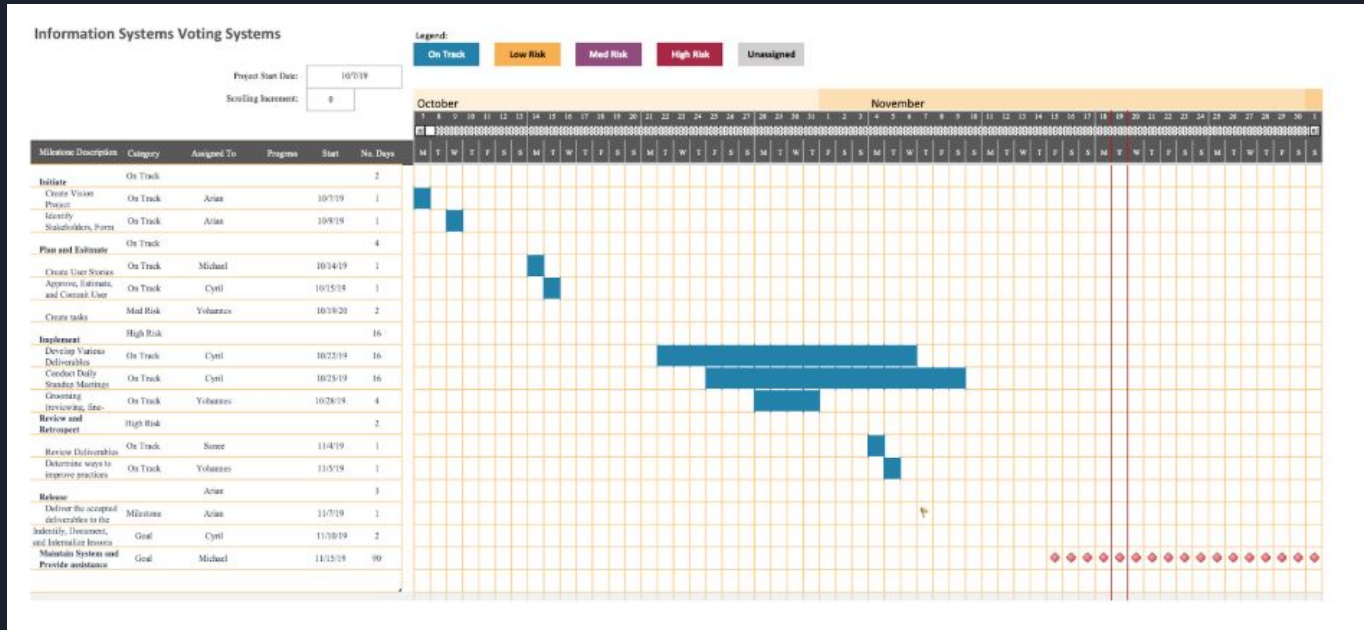
Systems Integration

- 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.

Project Work plan

Below is our Project Work Plan updated with on Task/Completed tasks (Blue). The Project work plan includes responsibility for each team member along with what the risk-level, milestone, and goals for the future are.

Our next tasks include Providing Support for 90 days



KanbanFlow Board

