Cole Cassidy

Iota

CS 489 Thesis Project

The Greek IT problem.

Currently in the market there exist several platforms for Greek organizations, all of which focus heavily on the financials, and very little on the bureaucratic aspects. They are also held back by poor technological implementation on the behalf of the larger organizations that often house these chapters. Many parent organizations struggle with outdated, or disorganized IT systems, as many are internally developed and thus, limited by the organizations budget. The same can be said for their child organizations, the larger chapters are dependent on external services to aid in their business. Services such as the Google suite, Discord, Zoom, and various other webtools. This requires lots of coordination, and often fails to be implemented in an effective way. Smaller chapters often avoid technology altogether in an attempt to manage budget, and because they don't have the means to operate technology for group management. Those that do utilize organizational technology, frequently mismanage it.

The Iota Solution

Iota is designed to address these issues and create a platform that makes managing organizations simpler. After looking at the needs of various Greek organizations there are several things Iota must accomplish.

User Agency	-	Users must be able to create and maintain their accounts independently, and in a modern way.
Organizational Groups	-	Organizations shall be created containing users, and users can be given administrative privileges.
Organizational Storage	-	Organizations must have file storage capacity, for documents that are to be shared across the organization.
Member Polling	-	Organizations must be able to poll, quiz, and call votes from their members.
Directory	-	Organization settings must allow for the creation of a "member directory" with various levels of discretion.
Accessibility	-	The application must be usable across multiple platforms.

These goals would span the issue of document distribution, as well as vote collection during online forums. Both of these issues have presented themselves in various ways as Greek organizations have attempted to traverse covid.

Stretch Goals

While the above list serves to identify the core issues Iota shall address, there are several stretch goals that would serve to better the life of Iota users. These goals would most likely not be built into the initial project. They are intended to serve as indicators as to how the project could expand. By keeping them in mind during development, its ensured that the application is built in a responsive way that allows for future growth.

Organizational Roles	-	Organizations can create internal roles for simple classification and access purposes.
Discord Integration	-	This would allow users to sign into their account via Discord, as well as allow organizations to utilize the service to manage existing communities. It would also allow users to interact with the service via Discord.
Google Calendar Integration	-	Organizations would be able to have a managed calendar by multiple users, that users can subscribe to in external calendar applications.
Cloud Storage integration	-	Document archives could be handled externally, for larger storage and different sharing options.
Administration Panel	-	System administrators would be able to see advanced user statistics, and would open the door to making Iota a monetizable service.
Whitelabel	-	Parent organizations could instance Iota in a way that allows it to be branded for their organization, similar to blackboard.

While these features are not immediately imperative, they would place Iota in a unique place as a tool for various organizations, not only Greek organizations.

How this will be built.

Iota will be built using modern web practices. The tool itself shall be divided into multiple components, allowing for expandability and efficiency.

Front End

The web application shall be hosted on a simple web server and shall be designed to operate from the end users cache. This is standard in modern web design and would allow for all future server calls to be significantly smaller, and almost exclusively API calls. It will be build in HTML5, CSS, and Javascript.

Back End

The back end application shall be 3 core components. The authentication server, the API/Logic server, and the database. The auth server will issues tokens to users for them to use over their routine API calls, allowing for stateless server architecture. The API/Logic server will be the primary platform by witch users receive content. It will send rudimentary responses to the webapp, which will parse them into readable content. The database server will not be accessible by the users, and will simply exist to manage all of the systems data.

Simplified

While this system sounds complicated, it actually simplifies the development process, as each element can be developed in whichever means best suits its purpose. It also would allow for redundancy. By separating Authentication and Logic, a down Auth server would not impact any active users.