Office Door Kiosk

Team 28

Dr. Thomas Daniels

Weston Morgan - Product Lead

Jacqueline Johnson - Project Manager

Eric Rysavy - Technical Documentation Lead

Chris Duncan - Software Architect

Evan Foley - UI/UX Lead

Peter Laurion - QA

sdmay18-28@iastate.edu

sdmay18-28.sd.ece.iastate.edu

Revised: 9/21/2017

Contents

1 Introduction

1.1 Project statement

Over the course of this project, we will produce custom software (designed for a tablet to be mounted by or on a door) which will be used as an interface for remote communication between professors and students.

1.2 Purpose

Going to a professor's office can be a key part of the college learning experience for many students. However, Professors are very busy, and it can be hard to get a face to face meeting with them even during their scheduled office hours. The office door kiosk will provide a quick and convenient means of communication for students trying to make a face to face connections with their professor, and for when professors want to display relevant information to people who come by their office.

1.3 Goals

1. Provide professors an interface for remotely relaying information to students
2. Provide students an interactive interface for facilitating face to face communication with professors
3. Build an easy to use experience for both user groups
4. Build a professional looking product
5. Practice agile development processes
6. Excellent documentation within the code
7. Automated testing for the product
8. Learn new technologies
9. Get experience taking a project from start to finish

2 Deliverables

1. Admin side code where professors have control over the information on their kiosk and how they can be communicated with
2. User side code where students have access to info professors want them to see and have option for communication with the professor through the kiosk
3. Professional interface and easy to use experience
4. A native app for mobile devices
5. Enclosure for tablet
6. An automated testing framework
7. Ample technical documentation

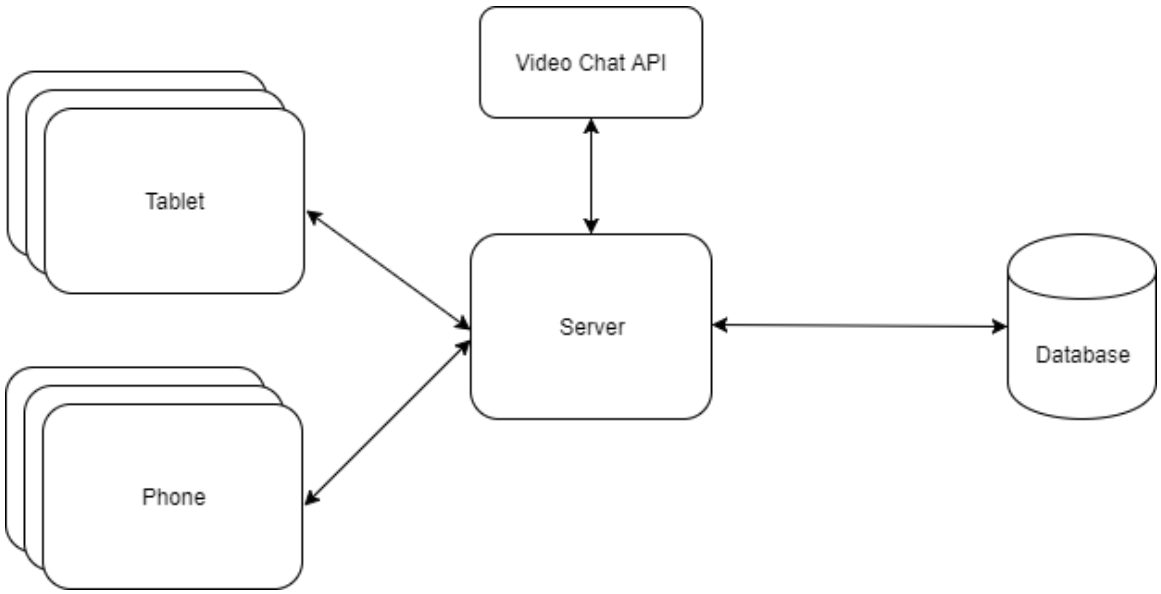
3 Design

3.1 Previous work/literature

While we have not worked on any similar projects or done research in this topic, we can look at different examples of similar projects. For example, grocery stores use kiosks to provide a self checkout service, which speeds up checkout for most people. We have also experienced kiosks being used at career fairs for filling out forms while waiting in line to speak to representatives. Lastly, we have experience using kiosks for check in queues at places like the doctor's office. All of these examples provide different types of use cases that we can implement into our project. Currently, the downside of these examples is that their specific feature set is pretty limited. Small feature sets lead to smaller user groups. We would like to make sure our project has a large enough feature set, that different people could find many reasons to use the system. Having a feature like video chat will make this kiosk act as an actual way to communicate with the owner of the kiosk and not just a place to enter your information.

3.2 Proposed System Block diagram

Below is our currently proposed block diagram.It contains five major parts; tablet, phone, server, video chat API, and database. The phone can be any kind of cellular device and the application runs on any number of devices. The tablet works the same way, any kind of tablet can run our application. The Video Chat API allows video communication from the kiosk to the kiosk owner's phone. The Server works as the connection between the tablet(s) and phone(s) and the database or each other. The phone will be able to communicate with the tablet and update its display almost instantly, and the tablet will be able to send communications to the phone. The database will hold the information that the kiosk owner wants to display on their kiosk and various account information things needed.



Proposed System Block Diagram

3.3 Assessment of Proposed methods

Design Decision: Tablet Kiosk outside of professors door

Alternatives: QR code kiosk

Justification: While a QR code implementation will be cheaper and won't have the implementation problems like charging, it loses a lot of the nice features of having an always on screen for displaying the professor's information and notices. It also adds another step for the student to see the information.

Design Decision: Node.js server

Alternatives: Java server

Justification: Most of our server tasks will be I/O bound and Node.js works faster and scales better than Java. In addition, Node.js is a JavaScript runtime environment for executing JavaScript server side code, so it will be easy to interact with React. Also it will make it easier to have developers work on both ends of the application if their written using the same language. Node.js is also growing in popularity and will be a good technology to learn.

Design Decision: React frontend

Alternatives: Angular

Justification: Angular generally has a steeper learning curve than React. Also, our Node server should easily integrate with React, because both are pure JavaScript. React also works really well with mobile, providing more evidence that it is the correct choice. Lastly, React uses a combination of JavaScript and HTML markup called JSX, which will easily allow markup to be placed in javascript functions. We feel like this is a very good feature, and plan on using it during development.

Design Decision: No-SQL database (MongoDB)

Alternatives: SQL

Justification: No-SQL databases work really well with Node.js as well as JavaScript because they all natively work with JSON

Design Decision: Native Mobile Application

Alternatives: A web application running in kiosk mode

Justification: A Native mobile application is being created because of the need for some of the features that a web browser will not be able to provide such as push notifications.

3.4 Validation

Ideally, we would use two different types of testing to confirm that our solutions work. For simple functionality, we plan on implementing an automated testing framework, that can be run during

each new code deployment to make sure that the new code is bug free, and continues to work as expected. However, automated testing can't determine if the problem is solved, only if the code is working correctly. Therefore, we plan to do manual testing for more complicated interactions and checking acceptance criteria. Acceptance criteria is the metric that tests whether the task literally solves the problems we are trying to solve. The acceptance criteria will be specified when creating our tasks. Ultimately, we won't know for sure if the project solves the problem until we test it with different users, including our client, to see if it solves the problems they have. This can only be done after a major prototype has been completed.

4 Project Requirements/Specifications

4.1 Functional

1. Remotely update information on the kiosk
2. Display professor's configurable homepage
3. Leave notes for students
4. Do not disturb mode
5. Video chat from tablet to professors phone
6. Calendar - configurable calendar that professors can display relevant time and date information to students on kiosk.
7. Scheduler - Way to schedule meetings with professor at kiosk while professor is gone
8. "Door bell" - A way for students to alert a professor that they are at their door and trying to reach them and give the professor a way to signal if they are able to talk to them or not.

4.2 Non-functional

1. The kiosk must be resistant to thieves and trolls
2. The kiosk must be able to be removed by those authorized.
3. Responsive UI
4. Any number of professors should be able to have accounts

4.3 Acceptance Tests

1. Verify that information can be remotely updated via phone
2. Verify that the professor's home page is displayed
3. Verify that notes can be left by the professor and can be read by students at the kiosk
4. Verify that when the kiosk is in do not disturb mode, it cannot be accessed by students
5. Verify that students can video chat with the professors phone via the kiosk
6. Verify that the calendar functionality is configurable
7. Verify that the scheduling works
8. Verify that when the doorbell is pressed at the kiosk, the professor's phone receives an alert
9. Verify that the professor can respond to a doorbell alert and the response is viewable from the kiosk
10. Verify that the device is difficult to remove from the enclosure
11. Verify that the app cannot be used for malicious purposes

12. Verify that the kiosk can be removed by authorized personnel
13. Verify that the UI is responsive
14. Verify that a large number of professors can create accounts

4.4 Test Plan

Jest

The current test plan is to utilize Jest as an automated testing framework. We will use Jest to test our React Native functions and components using deterministic functional tests as well as snapshot testing.

Field Testing

Some field testing of our application will be necessary to gage customer satisfaction. The enclosure for the device will need to be tested for security and feasibility. In addition, field testing will be needed to gage the look and feel of the application.

4.5 Standards

Our code writing process follows agile development. This is a common practice that many companies use for developing code. It allows for rapid development, but more importantly it holds the other developers to the same standards as their peers. These standards include both ethical programming practices and coding standards. We looked at the IEEE code of ethics to ensure that our practices were ethical. All of our standards for writing code were found by searching the web or personal preference. We do not believe that our standard would be deemed unethical by any party. Standards are applicable in our project because we want every team member to look at our code and be able to easily understand it. Having standards on how the code should be formatted allows for someone who didn't write the code to understand it more quickly.

1. Node
    a. Lines not too long that you can not read them easily.
    b. Two Spaces for indentations, looks better in web browsers and GitHub.
    c. Use named functions
2. React Native
    a. Consistent code organization
    b. Multi-line JSX, put each element being returned on a separate line
    c. Conditional JSX, declare empty variable at top, only populate it if the condition is met. Will render either the populated variable or nothing at all.
    d. 3 or more attributes on a component, display them on multiple lines
    e. Responsive interface design
    f. Multi-platform solutions

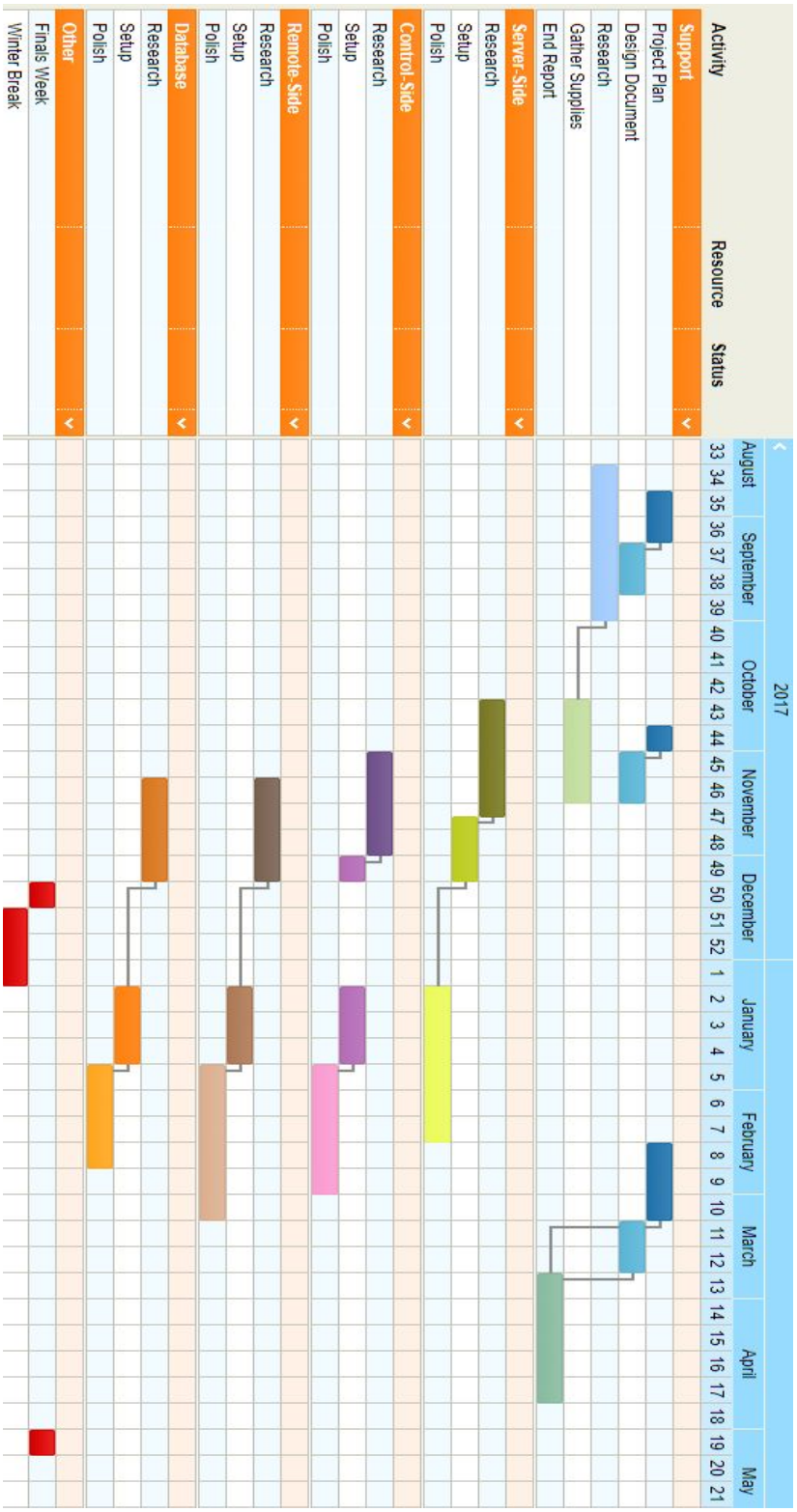
5 Challenges

1. React is a new technology for most of us, and those of us that have used it before have not used it for some time.

2. **Android studio is a new development environment for most of us and will take some getting accustomed to.**
3. **Designing the UI to be both customizable and consistently look good is going to be difficult without sacrificing one or the other.**
4. **Keeping functions both easy to use while also secure from misuse will be tricky.**
5. **None of us have any experience with constructing anything like an enclosure for the tablet to deter theft; we will have to do some looking into the process for making one or contract it out to somebody else.**
6. **Finding a way to keep the tablet charged consistently while it is hanging in a hallway with no available outlets will be difficult; we will have to work with our client to arrange something.**
7. **The program will have to be designed in an extendable fashion to make integration of late client requests as simple as possible.**
8. **Technology limitations make it difficult to run Android emulators on some of our hardware, and we collectively only possess one physical Android tablet. This could make both development and testing difficult for those of us without sufficiently powerful machines.**
9. **Creating a constant connection between phone and tablet could prove to be tricky.**
10. **Video chat is something none of us have worked with before and will create some constraints on how our project works**

6 Timeline

On the following page is our team's projected timeline for developing and implementing our kiosk application.



Project Timeline

6.1 First Semester

The first semester is when the bulk of the project will be completed. By the chart above, the first semester is the first 16 weeks. The site will be up and running with a minimal but functional UI, with server and authentication functionality working. Documentation and testing will be ongoing throughout these changes. Work will be divided according to each team member's predefined role and skills, though slight changes to assigned duties may occur as needed.

6.2 Second Semester

The second semester will be centered on polishing and refining the project. Our UI will be adjusted to be both functional and good to look at, and features will be refined and more thoroughly tested. New features may also be added upon client request, time permitting. We tentatively plan on assigning work according our currently assigned roles. Adjustment may be necessary pending how smoothly things run during the first semester tasks.

7 Conclusions

In conclusion, we plan to build a mobile application using React Native for our front end code, Node.js for our server, and Heroku for our database. The application will allow professors to create a kiosk dashboard from their phone. The dashboard will contain widgets for features that allow better forms of communication between professors and students. Some features we plan on implementing are displaying the professor's schedule or notes, and integrating the system hardware to allow video chat between professors and students so that they can discuss private information like grades.

Along with an application, we also will purchase a tablet, and build a kiosk prototype. It will be integrated into our client's office door and will be used to demo the different features. The set up will require an enclosure that will protect the tablet from theft, damage, and device hacking attempts. It will also need to include a way to keep the device charged, be accessible to the professor, and allow users access the the screen to use the application.

Our timeline is on track to have our fully featured application and kiosk setup completed by the end of the spring semester. It will meet all of our requirements discussed in section 4, including the features mentioned above and some non-functional features like a user-friendly interface design. Throughout the development of our application, we also will implement our testing features which will include manual testing, an automated framework, and field testing. Once all of these aspects of the project have been completed, the initial prototype would be ready for a larger scale implementation process, although this is outside the scope of our senior design project.

8 References

9 Appendices