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

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

1.2 Problem and Project Statement

General Problem Statement:

The problem we are trying to solve in this project is how can professors and students avoid missing connections when a student is trying to reach out to a professor in person or vice versa. For example if a student goes to a professor's office to try and ask a question about a problem on a test and the professor isn't in their office what does the student do? The likely answer is email the professor asking about his schedule to try and meet up about this question. While emailing isn't particularly hard to do the back and forth required to set up this short and simple meeting is going to take some time to figure out. Then it turns out the professor was right down hall and could've answered the students question if the professor knew the student was there. This is an avoidable situation and this the type of situation we're trying get rid of with our project. The problem isn't life or death but it is something that unnecessarily clogs up both students and professors time.

General Solution Approach:

To solve this problem we are going to create and office door kiosk system which will open up new avenues of communication between the student and professor and get rid of these missed connections that can happen. This kiosk system would be a tablet in an enclosure of sorts running the kiosk software that we create. This kiosk will be able to display and pertinent information, like their schedule and availability. The more important thing about this kiosk though is the information can be easily updated from a mobile device so if the professor is running late for his office hours perse they could from their phone tell their kiosk to display that information to any student that comes by. The student now knows to wait instead of leave and go write an email. This kiosk will also allow students to tell professors that they're at their door and need to talk by allowing the student to try and start a video chat with the professor from the door or send a notification which could be thought of as a door bell of sorts that tells the professor someone is at their door and wants to talk. This kiosk system will give professors and students a more direct way of communicating when trying to connect in person with each other.

1.3 Operational Environment

The operational environment for the kiosk should be inside of a school or office building so it should not be exposed to any extreme weather or conditions. The enclosure should keep the device being housed fairly dust free however the enclosure itself could acquire some low to moderate dust.

1.4 Intended Users and uses

This product shall be used by two different classes of user, the kiosk owner and the kiosk user, in the university setting where this problem has arisen that means the professor (kiosk owner) and student (kiosk user). Uses for the kiosk owner class of users is ability to display personal and interesting information on the kiosk which will be configurable by the kiosk owner. Ths information can be a wide range of things: pictures, schedules, contact info, notes, etc. The kiosk owner will also use the kiosk as a way for others to communicate with them. The kiosk user class of users will primarily use the kiosk as a way to learn about the kiosk owner and communicate with kiosk owner in a semi restricted manner, ie. only using video chat or sending a notification of their arrival at the kiosk.

1.5 Assumptions and Limitations

1.5.1 Assumptions

- The kiosk user has access to the internet when they're needing to update the kiosk
- The kiosk owner has a mobile device, for remote access
- The end product will not be used outside of Iowa State University

1.5.2 Limitations

- Application will only work for those at Iowa State University
- Team will not be able to handle maintenance after graduation
- Tablet enclosure provided will only fit one size of tablet

1.6 Expected End Product and Deliverables

Kiosk Application

The end product of this project will be an office door kiosk application. This will be an application that can run any mobile device. The application will have an administrative mode, where professors can login, update information, and interact with one or more kiosks. It will also have a kiosk mode, where users at the kiosk can interact with professors and view content. This application will be delivered by April 30th, 2018.

Enclosure for tablet

A prototype for a tablet enclosure will also be delivered at the end of this project. The enclosure will be designed for one specific size of tablet and will house the tablet safely. This enclosure will be able to be mounted onto a wall and have charging cable access, so that the tablet can be charged while inside the enclosure. The enclosure will be delivered on April 30th, 2017.

Design Document

A comprehensive design document will be included among the deliverables for this project. The design document will outline the project design, including specification, analysis, testing and implementation. The design document will be delivered on April 30th, 2017.

2. Specifications and Analysis

2.1 Proposed Design

The Office Door Kiosk team's product features an kiosk app that runs on tablets and mobile devices, both android and ios. The Kiosk display information uploaded by the owner of the kiosk, either on the kiosk itself, or remotely. The kiosk has capability to video chat with its owner, or send an alert when someone is present. The tablet that runs the app has an enclosure that is secure against theft.

The app's user interface runs using React to make the UI easy to work with and extend as we create new parts of the application. The app, either on the kiosk or the kiosk owner's phone, connects to a Node.js server which passes information to and from the MongoDB database. The server can connect two different sessions of the same kiosk (kiosk owner to kiosk) for video chat. The kiosk owner can connect to the app remotely from their phone and update their information, the stationed kiosk then reflects the updated information.

The kiosk owner is able to pick which widgets they wish to be able to access from their kiosk homepage. The rest of the kiosk functionalities will be accessible from a second page.

2.1.1 Functional Requirements

- Build a secure enclosure for the device
- Remotely update information on the kiosk
- Display professor's customizable data
- Leave notes for students
- Single sign-on with shibboleth
- Do not disturb mode
- Scheduler - Way to schedule meetings with professor at kiosk while professor is gone
- Student check-in queue - Lots of ideas on what this could be exactly but general gist of it would be a way to organize a bunch of students coming to office hours
- "Door Bell" that alerts the kiosk owner when someone is present

2.1.2 non-Functional Requirements

- The kiosk must be resistant to thieves and trolls
- The kiosk must be able to be removed by those authorized.
- Responsive UI
- Any number of Professors should be able to have accounts

2.2 Design Analysis

While in the designing process of developing this application we have swayed in the direction we wish to go a few times.

We have scrapped our plan on creating a web application that would be accessible from any device and would use Google Chrome's kiosk mode. This was decided against due to the lack of knowledge on how we would connect two people over video chat on a web browser. It would be more difficult to get use of a tablet or phone's camera from a browser than an application running straight on the device. There was also the issue of the kiosk owner having difficulties updating getting onto a webpage on their phone and trying to update things. This would be more difficult for the user than

on a app. We decided on these changes after discussing more thoroughly with our client on how he wanted to use our product.

We are now designing an application that will be an app for both android and ios devices. This app will make it so video communication can be implemented and updating the kiosk is easier for the kiosk owner. The only downside of this new choice of platform would be having it brand specific. We plan to delegate this concern by developing it cross-platform and continuing our decision to utilize the React library.

This project will be using React for the front-end, node.js as the back-end, and a mongodb database. Our choice of languages have not changed since our decision to change our platform.

React Front-end:

React can be used on a mobile application, so we aren't losing the benefits of its provided ease of adding new widgets. It also provides easy updating of displayed information on the application. We have determined from our research that React uses a combination of JavaScript and HTML markup called JSX, which will easily allow markup to be placed in javascript functions. React also easily integrates with our Node server, because both are pure JavaScript.

Node.js server:

Our product will not be doing much computation, most of the server tasks will be I/O bound, Node.js works quickly and scales well for these types of tasks. Node.js and React use JavaScript, with both sides using the same language developers will have a better understand of what is happening on both sides of the application.

MongoDB database:

No-SQL databases work really well with Node.js as well as JavaScript because they all natively work with JSON. React can take JSON strings as arguments for html element properties. This allows us to easily store the needed information for the kiosk to display.

3 Testing and Implementation

3.1 Interface Specifications

Tests for this software application will be run using a smartphone and/or a tablet.

As all uses/use cases are easily triggered from the standard UI, we do not currently plan on any advanced UI for testing purposes, but will rather be using a standard device to test the system. Tests for this software application will be run using a smartphone and/or a tablet.

There will be no hardware interfacing, so there is no need for hardware interface specifications.

3.2 Hardware and software

Jest

In order to test the ReactJS front-end of our application, we are using the testing framework Jest. Since ReactJS is maintained by Facebook, and Jest was developed by Facebook, we think it will be a good framework to use while developing this application. One of the most convenient things about using Jest is its 'watch mode'. You can put your Jest tests in watch mode which will automatically re-rerun tests when a test file is changed.

3.3 Process

Kiosk-to-server connection -

Kiosk uptime -

Kiosk hardware security -

Kiosk software security -

Remote schedule updates -

Remote Note Leaving -

Server-to-Database Connection -

Database logging -

"Doorbell" -

Video communication -

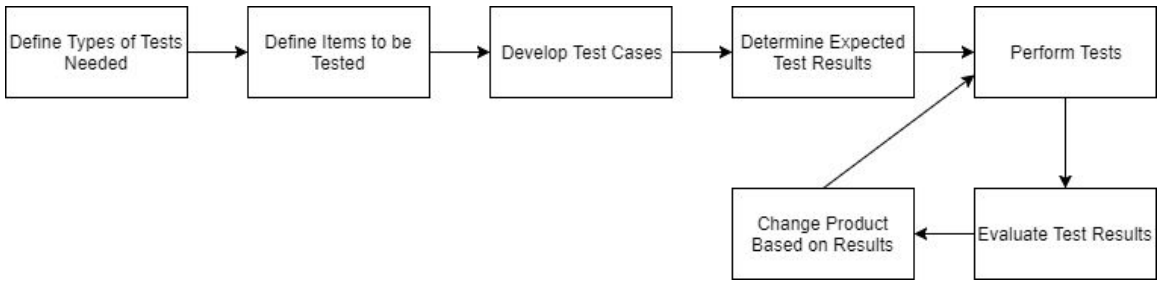
Scheduler -

Student check-in queue -

Professor account creation -

Professor account usage -

Flow Diagram



3.4 Results

We haven't tested anything yet so we don't have any results.

4 Closing Material

4.1 Conclusion

So far, we have met with our client in order to define the problem we are trying to solve, which is an poor communication between students and professors regarding office hours. We also discussed different solutions to this problem, ultimately deciding on an office door kiosk and application that will handle communications between students and professors. Our goals are to design and build this application, with features to make students' and professors' lives easier. All of these features will be wrapped into an easy-to-use interface that provides the users with a positive experience. This application will also be expected to run on a tablet, that we plan on integrating into an enclosure for office doors. We have already started following the best plan of action to receive these goals. The first step was to do research on some of the best practices for creating a kiosk application, and then design the architecture of the application, including the languages we plan to use. Ultimately, after doing research on different front end frameworks and back end technologies, we decided to use React native for our client side language, and Node.js for our server side language. Finally, we designed the screen flow to give us a first look at our application's layout.

At this point we can look at our future path, for meeting our goals, but it mostly revolves around building the application itself. Creating a shell for our application is the current next step. This is used to make sure we have a basic understanding of our technologies, and then features can be added in as we complete them. While we build our application features, we also need to set up our automated testing framework. This is incredibly important for building from the beginning, so that we can guarantee that new code does not break old code. We can confidently add features at a rapid rate when automated testing confirms that the older code always works. Along with automated testing, we also hope to integrate user testing as more prominent features are added. This way, we can confirm that our features are user friendly and helpful. After building a minimum viable product in the application, we need to spend some time building the prototype kiosk hardware and setup. This includes obtaining a tablet, and designing and building the enclosure. The enclosure needs to protect the tablet from theft and damage, while ensuring that it can be used easily, and can be charged at all times while being attached to the office door. Once the enclosure and application have been completed, the final step is putting them together for a prototype.

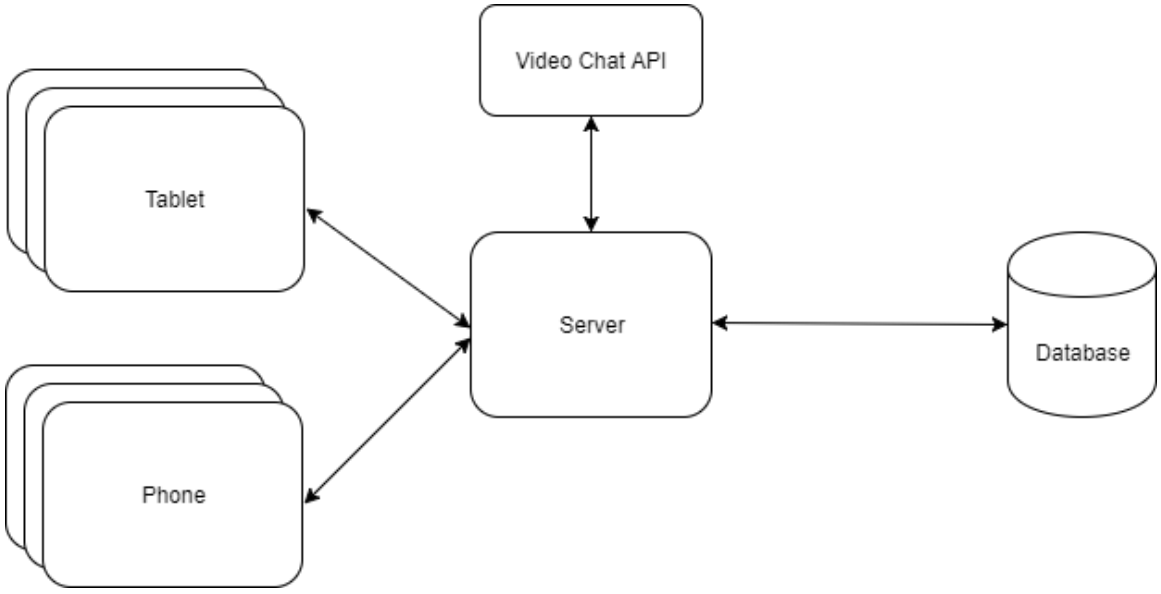
4.2 References

React Native Information - https://facebook.github.io/react-native/

Node.js Information - https://nodejs.org/en/

4.3 Appendices

Application Architecture


Video Chat API
Tablet
Server
Database
Phone

Application Screenflow