Interactive Kiosk

Software Development Plan

**Authored by:**

Aaron Liu

Daniel Cox

Matthew Dunn

Philip Bedward

**Outline**

[Overview](#_3dy6vkm)

[Goals and Scope](#_4d34og8)

[Process Overview](#_2s8eyo1)

[Deliverables](#_3rdcrjn)

[Features](#_26in1rg)

[Documentation](#_lnxbz9)

[Project Materials](#_147n2zr)

[Release Plan](#_35nkun2)

[Release 1 - Due Date 11/29/2018](#_1ksv4uv)

[Release 1 Sprint Plan(s):](#_3o7alnk)

[Semester 2 - Due Date: 4/26/19](#_2jxsxqh)

[Semester 2 Sprint Plan](#_3j2qqm3)

[Risk Management](#_4i7ojhp)

[Technical Process](#_2xcytpi)

[Scheduling and Estimates](#_3whwml4)

[Tools](#_qsh70q)

[Artifacts](#_1pxezwc)

[Measurements & Metrics](#_2p2csry)

### 

### Overview

The Interactive Kiosk application aims to provide a seamless mechanism for the flow of information between a web application and smart TV devices. Each smart TV will display information that has been pushed from the web application, and will update as the information is edited. The information displayed is represented on tiles carrying the title and description as well as a periodic toggle for a QR code. The number of tiles available is based on the type of device and the amount of tiles shown is by choice of the Client account holder. The smart device will receive its information from a database that and subscribe to the data that is sent from the web application. End users are given the ability to request to publish content on the smart TV devices.

### 

### Goals and Scope

The goal of this project is to deliver a functional system that allows information to be sent to smart TV devices through a web application. Users who stumble across these devices will be able to access the information displayed on the device through a QR code, or be request to upload information to be displayed on the device.

The scope of this project includes the utilization of tools on Amazon Web Services (AWS) for the application infrastructure. These tools include AWS S3, Lambda, API Gateway, IoT Core, DynamoDB and Cognito. The tools listed for the backend may change over the life of the project and would be updated to reflect it. For the frontend, the main web application that the account holders have access to will be built on AngularJS which utilizes CSS, HTML and Javascript. The smart device chosen will be an Android TV. We will focus on developing the system for this type of smart device. For our first release of the project, we will be focusing on registering a Smart TV device to our system as well as publishing one piece of content to said TV.

This entire system is built on a serverless backend utilizing Amazon Web Services, so that many of the overhead costs can be reduced. The web application will be supported on browsers like Google Chrome, Firefox and Safari. The type of smart device currently supported would mainly be on one brand of smart TV’s, which has not been confirmed yet. In the meantime the Smart TV App will be build as an Android TV app on the Android Studio simulator. To scan QR codes, if the mobile device is an iOS product, the camera is all you need, but for Android mobile devices, they will need to download a QR code scanner app. With this system in place, the type of customers attracted would be any end user who has basic knowledge of using QR codes and a mobile device.

What is out of scope for this project is a server-enabled web application, or any use of a server. Also, a mobile application is out of scope for this project as well, although we will be monitoring the user-experience of a mobile first web-app closely. Running this application on all types of smart TV devices is out of scope as well. Due to the novice-level knowledge of smart TV application development of the engineers, we will be sticking to one type of TV to minimize the risks that would be incurred from developing on multiple TVs. Some of these risks include learning curve, SDK and OS capabilities and features and hardware limitations and requirements.

### Process Overview

Every Day

* Daily Stand-ups over slack
  + Status every task in Jira, ensure progress is occuring. Confirm against due date. Rationalize current progress vs. expected progress and completion date. Make sure all issues preventing achieving planned completion date are removed. Ensure people working task are still committed to completion date.
  + Include what you are working on overall (feature, overarching tasks, etc.), potentially a task id
    - This should be written in parallel with what is entered on Jira
  + Include what smaller task you worked on the day prior, if any, or nothing if nothing was worked on the day prior (this will happen, it’s ok)
    - EG “Programmed script for setting up S3 bucket on Amazon AWS”
  + Indicate what you plan on working on for the current day
    - It’s also OK if you have nothing planned for the current day as we all have busy schedules. This is why including the overall feature/task from Jira is helpful, to let others know what you will be working on
  + Specify any blockers to your progress
    - Also indicate whose responsibility it is to resolve said blockers. This could be Bizcloud, yourself, another team member etc.
    - “Parking Lot” - any blockers, you can ask questions in this slack channel for after standups, etc.
  + Give an estimate percentage of progress if you have a good idea of what it may be; especially useful for larger tasks
    - Would help for visibility in Jira
* Ensure you are up to date on your time tracking for the day. Try to make sure this is taken care of before Daily Stand Ups for the next day.
  + Ensure Jira is updated before end of stand-up
  + This was a weak spot for us last semester, doing this and keeping on it should help remedy that.
  + I (Matt) can also check everyone’s hours daily, make sure work is reflected on the hours tracking VS the stand-up, give reminders, etc.

Meetings

* Tuesday
  + Ensure agenda is published prior to meeting
  + I think it would be best to use Tuesdays as sort of a review/get-together to discuss what has been done so far and what needs to get done before we deliver Thursday. This way we can assess what still needs to get done in order to be effective with our meetings with Bizcloud.
  + Following this can be more of a workshop for those who need to code or work together but this may not be mandatory for every member.
    - A lot of our Tuesday meetings were spent sort of floundering around without really having a direction. Sometimes we worked on stuff in person, sometimes we didn’t, sometimes not everyone was necessary for work, etc.
* Thursday
  + Ensure agenda is published prior to meeting
  + Obviously our meeting with Bizcloud. Should go as such:
    - For those available, meet 10-15 minutes early to prepare
      * Send Bizcloud our 4-up if they can’t make the meeting
      * If we **need** Nagesh, make sure to email them prior to the meeting.
      * Send a reminder to them before the meeting that we are meeting
    - Go over 4-up
      * Progress, what’s been done in the past week
      * Are we confident we will deliver for the end of sprint / status update
      * If it is the end of sprint, Did we accomplish our goals?
    - Discuss what we plan on delivering for the next week
      * Possibly discuss overall sprint plan and what we are moving towards for a certain sprint
    - Risks
      * This needs to be evaluated. A lot of entries are statements that might not be “Risks” by themselves - might be beneficial to include a sub-bullet to explain why this could impact our progress in some form. Currently it feels like it would be more like a “Blockers” or “potential Blockers”
    - Needs
      * Often seems to turn in to more notes than needs. I’d say we should keep this more to what we need from Kal for some reason or a sponsor. This is really where blockers would come in to play. Any task for a team member to complete resulting from this should be put in to Plans, and could be indicated that it is separate from our already set plans for the following week as well as that it satisfies a Need
    - Demo
      * Demos will take place the week after a sprint has been completed
      * Any progress can be shown off here if any.
  + Review
    - Create a plan of attack for up to next Tuesday or next planned meeting
  + Post-Mortem
    - Discuss what went well, what didn’t following the meeting with Bizcloud
  + Following this can be more of a workshop for those who need to code or work together but this may not be mandatory for every member
* Planning meeting - Fridays at 9:00AM
  + Plan for next sprint
    - Assign items to a sprint for next sprint - mid-sprint
    - Assign story points to items for next sprint - mid-sprint
      * Point poker
    - Assign owners to items - beginning of sprint
* Live Integration
  + Most often can be done following meetings on Tuesdays if no other time is available. Dates can be set up before this if scheduling permits, and these dates will be used when they are necessary to complete Stories for a sprint.
* “Hack” days - As necessary

Updating the website

* Update the SE website weekly on Thursday before 5:00PM

Using Slack

* Process will not work if people are unresponsive
* You should be checking slack when you get up, before you go to bed, and throughout the day every 3-6 hours
* Make sure to have notifications turned on.

### 

### Deliverables

Deliverables for the Interactive Kiosk application are derived from project sponsor requirements and senior project requirements. A more extensive breakdown can be found [here](https://docs.google.com/spreadsheets/d/1aayKcYrwRqNCFtSHIIBEpGFf1KtkVdGtY2HEVkLIJq0/edit#gid=0)

|  |  |  |
| --- | --- | --- |
| Features | | |
| **Feature ID** | **Use Case ID/s** | **Feature Description** |
| 01 | 1.01, 3.01 | Login and logout functionality for all users |
| 02 | 1.01, 1.02 | Add a new Client Account |
| 03 | 1.01, 1.02, 1.03 | View all client accounts |
| 04 | 1.01, 1.04 | View all active devices within the entire system |
| 05 | 3.01, 2.02 | Add a new Section to a Client Account |
| 06 | 3.01, 2.03, 3.08 | Add a new Section Administrator to a Client Account |
| 07 | 3.01, 3.02 | Registration of a Smart TV - A process that adds a Smart TV to an existing client and section. Allows the TV to be utilized for future content pushes |
| 08 | 3.01, 3.05 | Create content for a Smart TV as an account owner - the ability to create new content via the web application |
| 09 | 4.02 | Submit a content submission request - as an unauthenticated user, be able to submit content to be pushed to specific TVs that can be approved or denied by an authenticated user |
| 10 | 3.01, 3.03, 3.04 | Approve or Deny a Content Submission Request |
| 11 | 3.01, 3.06 | View available content - viewing all content available to your scope via the web application |
| 12 | 3.05, 3.06, 3.07 | Pushing a Message to a Registered Smart TV - Sending a message to a TV that has been Registered to and section |
| 13 | 4.01 | End user is able to scan a QR code to visit some sort of link associated with a Content on a Smart TV |
| 14 | 3.01, 2.04, 3.08 | Add a new Client Account Administrator |
| 15 | 3.01, 3.09 | View all Smart TVs in my scope |
| 16 | 3.01, 3.09, 3.10 | View content currently deployed to a Smart TV |

|  |  |  |
| --- | --- | --- |
| Documentation | | |
| **ID** | **Artifact** | **Due Date** |
| 1 | SE Project Website - A website containing all work products and project artifacts maintained in the project account on the se.rit.edu web server | Recurring Updates |
| 2 | BizCloud Project Website - A google site containing all work products and project artifacts to be reviewed by BizCloud Experts | Recurring Updates |
| 3 | Domain Model - a domain model which provides a common understanding between the team and project sponsor of the domain and scope for the project. | 9/19/18 |
| 4 | Sequence Diagrams - Multiple sequence diagrams which model the main flows for important use cases. | 10/18/18 |
| 5 | Architectural model - Overall model which describes how internal components of the application interact with each other.   * AWS * Web App * TV App | 10/16/18 |
| 6 | Design Documentation - All design documentation and artifacts are given to the project sponsor.   * Sequence Diagrams * UML * Mock-Ups * Project Plan * ERD | 10/17/18, Recurring Updates |
| 7 | Data Model (ERD) | 10/16/18 |
| 8 | Functional and Non-Functional Requirements | 10/16/18 |
| 9 | Design Mock up | 10/23/18 |
| 10 | Test plan - A document describing the scope, approach, resources and schedule of intended test activities. | 1/31/19 |
| 11 | Software release plan - A plan communicating the features, enhancements, and fixes coming out in a series of releases. | 11/20 |
| 12 | Device/Framework/Language Proposal   * Pros and Cons |  |

### Release Plan

#### Release 1 - Due Date 11/29/2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semester 1 “Sprint Plan” | | | | |
| **Sprint #** | **Feature ID(s)** | **Start Date** | **End Date** | **Goal Description** |
| 00 |  | 9/19/18 | 10/17/18 | Planning Phase - 9/19   1. Initial Meeting - 8/30 2. Time Tracking - 9/8 3. Project Synopsis - 9/15 4. Project plan - 9/15 5. Domain model - 9/19; Artifact 3 |
| 00 |  | 10/17/18 | 10/31/18 | Design Phase - 10/17   1. Sequence diagrams - 10/18; Artifact 4 2. Use Case document - 10/18 3. Functional and non-functional requirements - 10/16; Artifact 8 4. Architecture models - 10/16; Artifact 5 5. ERD - 10/18; Artifact 7 |
| 01 |  | 10/31/18 | 11/14/18 | Sprint 1 - 10/31: Setting up AWS, non-functional UI and Research   1. Design web app Mock-up 2. Setup Angular app 3. Non Functional UI 4. Set up AWS Tools 5. Smart TV App Development Research 6. Update project website |
| 02 |  | 11/14/18 | 11/29/18 | Sprint 2 - 11/14: DB Setup, revisit documentation and process, more non-functional UI   1. Decide on endpoint functionality 2. Non functional Smart TV UI 3. Smart TV UI wireframe 4. Add/Update Use Cases 5. DB Schema as code 6. Discuss requirements and progress with Kal 7. Revisit Requirements & Documents 8. Upload current web app to S3 9. Update Project website |
| 03 |  | 11/29/18 | 12/13/18 | Sprint 3 - 11/29: Get API Gateways set up, connect front-end to gateways, get IoT set up for Smart TVs, create presentation, write Lambda functions all to prepare for pushing 1 content to 1 smart tv.   1. Update Swagger file 2. Update the Use case for Register TV 3. Write Lambda code 4. Connect API Gateway endpoints to Lambda 5. Connect the frontend to API endpoints 6. Setup IoT 7. Add features to project plan 8. Connect Smart TV App to IoT channel 9. Interim Presentation 10. Update Project website |

#### 

#### Semester 2 - Due Date: 4/26/19

Note: Sprint 4 will function as a “Sprint 0” in the beginning of Release 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semester 2 Sprint Plan | | | | |
| **Sprint #** | **Feature ID(s)** | **Start Date** | **End Date** | **Goal Description** |
| 04 |  | 1/14/19 | 1/18/19 | Push one content to one Smart TV  This is pre-set TV and Content used to ensure core functionality |
| 05 | 01, 07, 08, 11 | 1/18/19 | 2/1/19 | A user has the ability to Register A Smart TV to AWS IoT, Create Content from within the WebApp, View all Content generated by the user, and Log in / log out of the Web App |
| 06 | 12 | 2/1/19 | 2/15/19 | Push an authenticated user-generated message to a registered smart TV |
| 07 | 02, 03, 04, 15 | 2/15/19 | 3/1/19 | Implement “Client Account” separation so that only client account-relevant content shows on dashboard, is required for login, etc. as well as registering Smart TV, the ability to add a new Client Account, View all Client Accounts, View all active devices within the entire system, as a System Administrator as well as set up the Client Account Owner, Client Account Administrator and System Administrator roles. At this point, the Client Account Admins should be able to see all TVs within their Client Account, but none outside of it. |
| 08 | 05, 06, 15 | 3/1/19 | 3/15/19 | Add a new section to the client via the web application and be able to view all sections within a particular Client Account as well as setting up the Section Administrator role. This means that a Section Administrator should only be able to see TVs and Content within their Section |
| 09 | 09, 10 | 3/15/19 | 3/29/19 | Submit a content submission request through the web app as an End User and be able to Approve or Deny a Content Submission Request |
| 10 | 13, 16 | 3/29/19 | 4/12/19 | View content currently deployed to a specific Smart TV and End User is able to scan a QR code on the tv to be taken to the URL it points to |
| 11 |  | 4/12/19 | 4/26/19 | QA - Test and ensure functionality against test plan  Improve UI  Refactor and comment code  Update Documentation |

#### 

### Risk Management

Risks management for the Interactive Kiosk software application is managed via an external spreadsheet located [here](https://docs.google.com/spreadsheets/d/1J94d3bbFJxdOVVmpUhhuDiHJMWgReC2P_8uGiJ9MePg/edit?usp=sharing). This spreadsheet maintains a record of identified risks along with their exposure, classification, prevention strategy, mitigation strategy, and status. Each section is defined as follows.

**Exposure** - Risk exposure or loss of potential business is calculated in this spreadsheet by multiplying the probability of the risk with it’s impact. Probability is on a scale of 0-1 and impact is classified by low, medium and high, which corresponds to the numbers 1, 5, and 10 respectively. Risk exposure allows identified risks to be prioritized and focused on correctly.

**Classification** - Classifications for the risk were chosen by what general domain is most impacted in the event of a risk becoming a reality. For this sections we chose 4 major categories including infrastructure, development, stakeholders, and technology. In the event a risk fits in to multiple categories, we classify it as the one that will be impacted greater.

**Prevention Strategy** - For each identified risk, a prevention strategy is created to minimize the probability of the risk becoming active. These prevention strategies will also provide early indicators if a risk is in danger of becoming active.

**Mitigation Strategy** - The mitigation strategy is a plan created for each risk in the event it becomes active. These strategies will either lower the impact and exposure of a risk or resolve the risk entirely. Owners listed on each risk are responsible for implementing the mitigation strategy in the event of a risk becoming active.

**Status** - A status is shown for each risk to allow the team to quickly understand how each risk is being handled. For this section, the 2 status levels chosen are monitored and issue. Monitored risks are all that are being watched closely but have not become reality. Once a risk has become reality it is marked as an issue and tracked on a separate issues document.

### Technical Process

**Methodology: KanBan**

The methodology that was decided upon is Kanban. Kanban is an agile methodology where it excels in projects where the is a greater degree of variability in deliverable priorities. There are no official methodology-based roles; our roles are based on the Senior Project recommendations and are malleable as we move in to different phases and adapt in to different roles. Instead of having time-constrained sprints where our progress is measured by how much we get done within the sprint, we continuously generate deliverables and our progress is measure by how long it takes us to deliver those deliverables. This is accomplished per-task and changes are allowed to be made as the project continues. So long as we maintain our documentation, this methodology works very well for our team in the environment we’re in.

### 

### Scheduling and Estimates

We will have a sprint planning meeting where we will assign user stories to sprints and assign these stories to group members. In each sprint we will have a defined goal and one or more features (dependant on the size) that we will be trying to complete by the sprint’s end date. The week after a the of a sprint we will conduct a demo in order to display the functionality of a new feature to the project sponsor. This will allow us to elicit the sponsor’s feedback on our work and iterate on that feedback during our next sprint. Also, this will be good in the sense that we will have periodic conversations with the sponsor, with tangible progressions, which will give us an avenue to discuss and negotiate our project scope.

Also, during our sprint planning we will be utilizing planning poker to assign points to our stories. As we progress through our releases we can utilize our story points Our metrics(see below) will allow us to make accurate predictions for the amount of time that it’ll take to complete a set of features and which features we can complete between demos. We will follow this process until the end of development of the product (week 12 of Spring ‘19 semester).

[Link to Project Schedule](https://docs.google.com/spreadsheets/d/1aayKcYrwRqNCFtSHIIBEpGFf1KtkVdGtY2HEVkLIJq0/edit#gid=0)

### 

### Tools

The schedule and development will be tracked via Trello, GitHub Projects and time tracking spreadsheets. Development will follow a Kanban process methodology as it promotes continuous delivery while not over burdening the development team.

* Amazon AWS
  + S3
  + Lambda
  + Cognito
* Slack
* Jira
* AngularJS
* VSCode
* GitHub
* Git
* Android Studio

### 

### Artifacts

* **4-Up Chart -** The 4-up chart was created using a template given by the Software Engineering department at RIT. It is used to track weekly accomplishments, what’s planned, what our risks are, and what we need to accomplish our current tasks. This is maintained by the team weekly and presented at weekly meetings with the project sponsor.
* **Time Tracking -** Our time tracking document was adapted from the one given by the Software Engineering department to support Google sheets functionality and be expandable to run statistics in the future. Each member’s sheet is maintained by themself and the main totals sheet is updated automatically.
* **Requirements(Functional and Non-Functional) Document -** A list of actors and what functionality they need out of the software. This is maintained as requirements change by the team.
* **Domain Model -** A high-level diagram of the system architecture. This is maintained by the team and updated as the architecture changes.
* **Use Case Document -** A series of actions different types of users will take to accomplish a tasks in list form. This is updated by the team as requirements and user roles change.
* **Sequence Diagrams -** A visualization of the Use Cases, showing how components of the architecture operate when a user acts on the system. Maintained by the team and updated as use cases change.
* **Bizcloud Project Website -** The project website where all project documents from our Google Drive as well as BizCloud resources are shared are presented. It was established by Bizcloud Experts, run on Google Sites and maintained by Celestial Orca and Bizcloud Experts. New documentation will be added as it is created and maintained by the team Web Site Manager.
* **RIT SE Project Website -** A repository of all project documents on a public-facing site hosted on RIT’s Software Engineering servers. These are pulled from the team’s Google Drive.
* **Kan-Ban Board** - A list of actions to take, separated in to different sections for whether they need to be done, are being worked on, are completed, etc.

Any other documentation will be added as it is created.

### 

### Measurements & Metrics

LOC

Lead Time

It is also useful to track the code churn. That is the number of additions and deletions made to the source code. Generally speaking, the most modified code is prone to the most bugs/defects. The code churn for each file (we will focus on files that we created and edit and not auto-generated files) could help to provide the team with valuable insight on where bugs are most likely to live. Since this is a new type of software project for the development team, we understand that the more unfamiliar parts of the system will go through a lot modifications. With this insight, proper risk management and mitigation can commence.

As previously stated, a Kanban process will be followed. Due to this, we will track the lead time for each task during a sprint. Having the lead time will allow us to make predictions for our workload for future sprints. If an upcoming sprint gives us a very high lead time that is over past ones that we’ve been able to manage or even have struggled with, it will allow us to make the reasonable decision that we are taking on too much responsibility for that sprint and we can reduce the amount to deliver for that sprint. With that said, it also allows us to look back and realize that we are taking on too much responsibility or to try to make changes to increase our output.