Heart Screening Management System
Semester 2 Plan

Team Members

- Carlo Campanini (ccampanini2018@my.fit.edu)
- Chris Newberry (cnewberry2018@my.fit.edu)
- Drew Dunkelberger (ddunkelberge2018@my.fit.edu)
- John Dewey (jdewey2018@my.fit.edu)
- Noah Wilson (wilsonn2018@my.fit.edu)

Faculty Sponsor

- Eraldo Ribeiro (eribeiro@fit.edu)

Client

- Evant Ernst, CEO - Who We Play For
- Klynton Holmes, Tech Advisor - Who We Play For

Meetings

Date	Discussion Points

1/14/2022	- Overall plan for semester 2
	- Evaluation methods
	  - Beta testing
	  - User surveys
	  - Attending screening events
	- High priority tasks for Milestone 4

Goal and Motivation

Motivation

Sudden cardiac arrest (SCA) is the 2nd most common cause of death in youth. It is also the leading cause of death for student athletes and death on school campuses. This is especially an issue for athletes with a heart condition that seem completely healthy on the surface, but are putting excessive strain on their heart. The common approach of a simple history and physical examination is not effective at detecting potentially fatal cardiac abnormalities in young adults. According to Dr. Joseph Marek, the standard physical examination misses over 96% of those at risk for SCA.

Goal

The ultimate goal of this project is to increase the use of ECG screenings in American high schools to minimize the cases of SCA in student athletes. This will be done by improving upon Who We Play For (WWPF)'s current heart screening program with an event-management system that provides automated processing of results, event scheduling tools, ECG collection, and results delivery. The current system is a pay-per-event, requires some setup, and is localized to the management of Florida users. The new system will be scalable for use across all of the United States.

Key Features

Feature	Description
On-site registration	System will support registration for participants on the site via QR code; It must be efficient, support a large amount of users in a short span of time, and be mobile-friendly
Scalable event scheduling and management	Directors are able to create and manage public or private heart screening events. Participants can register for events online or in-person via a link/QR code. This will be done in a way such that the software is scalable to different regions/states

Payment management	Accountants will have a specialized account for accessing payments for events that have occurred or that have been requested to be canceled to streamline the process of verifying payments. Once payments have been confirmed, accountants have the ability to mark the event as completed
Email notifications and results delivery	Automatic email notifications occur the day before an event, reminding participants of the date and location. Low-risk results will also be delivered via email once the ECG data is processed and interpreted.
Volunteer sign-up	The current system does not have a role for volunteers yet, making it difficult to manage them. Volunteers should be able to prepare for upcoming events by viewing details and downloading necessary spreadsheets ahead of time

Tools

Key Libraries

- **Node Package Manager (NPM):** Manages all the 3rd party libraries for our project, packaging dependencies into **node_module** folders which can be uploaded to AWS with source code.
- **Mocha/Chai:** Javascript test and assertion library used to write unit tests and integration tests for our database and backend.
- **Axios:** Promise-based HTTP client for node.js that allows us to make GET and POST requests to our backend.

APIs

- **Payeezy API:** Provides methods to process a variety of payment methods. Used to charge and refund credit card payments by participants.

SDKs

- **Google Suite SDK:** Use an authorizer to check if a user is part of a specific group (director, administrator, accountant, etc.)
- **AWS Javascript SDK:** Provides a way to make various method calls for AWS resources from our backend

Frameworks/Runtimes

- **VueJS**: Used to construct single-page user interfaces for all roles of our system.
- **NodeJS**: Handles the backend of our system by allowing execution of JavaScript outside the browser. Used to make asynchronous calls to our database, APIs, and SDKs.

Languages

- **Javascript**: Source language for both the frontend and backend.
- **Terraform**: is utilized to write infrastructure as code (IaC) and deploy it to AWS without the need for a console. It provides a simple method for setting up, making changes to, and tearing down environments for different users.
- **Bash**: The team utilizes Makefiles to perform configured Terraform commands and upload source code to AWS Lambda functions for the frontend and backend of each role, as well as general service functions

AWS Services:

- **DynamoDB**: NoSQL database that supports key–value and document data structures. Used to store event and registered participant information.
- **Cognito**: provides authentication, authorization, and user management. Used to authenticate directors with Google, as well as accountants, administrators, and volunteers in the future.
- **API Gateway**: creates, publishes, maintains, monitors, and secures REST, HTTP, and WebSocket APIs at any scale. Used to make API calls to AWS as a director or participant.
- **S3**: object storage service that offers industry-leading scalability, data availability, security, and performance. Used to store the remote state of our Terraform deployments. Will likely be used to create an endpoint for volunteers to access updated participant information spreadsheets as sign-ups occur on-site.
- **Simple Email Service** (SES): email platform that provides an easy, cost-effective way to send and receive emails. Used to send out email reminders, payment receipts, and low-risk results to participants

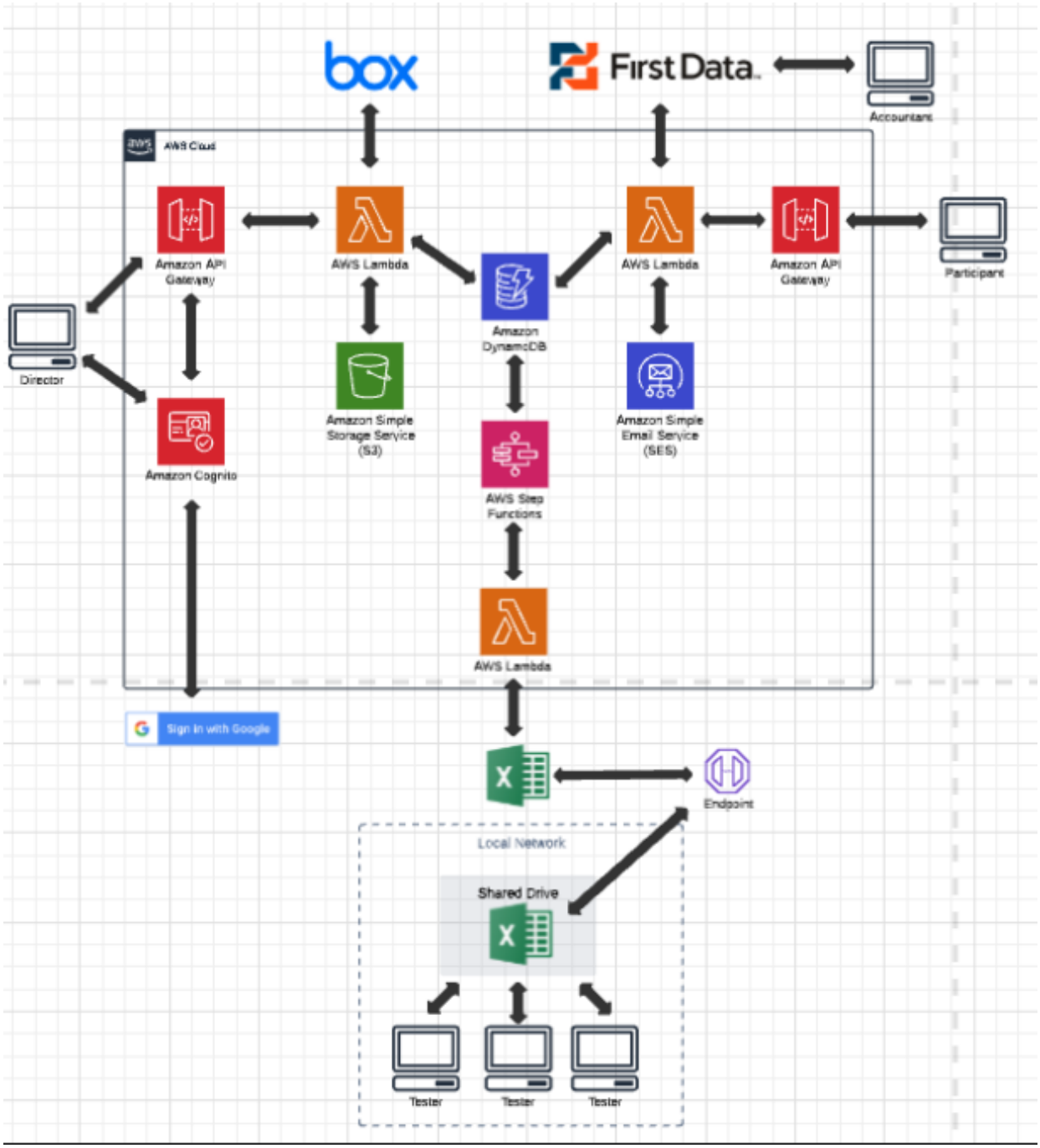
- **Step Functions**: Visual workflow service that enables automation through state machines. Used to automate the state transitions of finalized events until they have occurred and integrate several different AWS services.
- **Event Bridge**: Serverless event bus to simplify event-driven applications at scale with custom events. Used to enforce a cron job for daily participant email reminders and event finalization at a specified time.
- **CloudWatch**: Allws collection, access, and correlation of data across all other AWS resources. Used by our team to verify that events have occurred and debug faulty resources.
- **Lambda**: compute service that allows code to run without provisioning or managing servers. Used to make serverless calls between different AWS services, as seen in our system diagram below.

Technical Challenges

Challenge	Description
Gaining more experience with full-stack development using the required frameworks and tools	Team members are still focused on either frontend or backend development, but many tasks require knowledge in both areas to complete. The efficiency of feature implementation would benefit from
Time management	Each team member has a busy schedule, making it difficult to collaborate on the project. The team needs to prioritize this project and make efficient use of the time where all team members are available.
Handling sensitive information and money properly	We cannot make mistakes on payments since we are handling other people's money and personal information

Design

System Architecture Diagram



Evaluation

In order to receive valuable evaluation results, we will deploy an AWS environment for high school directors (HSDs) to perform beta testing in early March with an actual (private) heart screening event. This event will likely be smaller scale than typical heart screening events.

- In order to evaluate user experience, we allow beta testers access to surveys along with an open-ended feedback section for continuous improvement.
    - A feedback button will be available to users on each UI
- In order to test the speed of our system compared to the original process, we will compare the time it takes for a director and participant to perform various tasks using our UI vs. the time it takes for those tasks on-site
    - Team members will attend an upcoming event(s) and repeatedly time various tasks using the current process
        - Waiting time for participants to sign-up
        - Time it takes to complete the participant questionnaire
        - Time it takes a director to officialize an event
        - Time it takes participants to receive low-risk results (will be implemented in our system in the future)
    - In the user survey, we will ask directors/participants to compare their experience with the original heart screening event process (if applicable)
- In order to evaluate the reliability of our system, we will be tracking the number of errors that occur using our system vs. the number of mistakes that occur on-site using the current process. This includes:
    - Registration errors
    - Payment errors
    - Errors unique to our system
- In order to evaluate the accuracy of our system, we will continue to perform unit testing and integration testing as we implement new features and roles.
    - Code coverage libraries such as **c8** or **karma-coverage** may be incorporated into our system to give a quantitative measure of how thoroughly tested the individual components of our system are

Progress Summary

Module/Feature	Completion %	To Do
Service: Automatic registration closure	100%	

Participant: Automatic time slot display in UI	100%	
Service: Automated event email reminders to registered participants	100%	
Participant: Vary registration form verbiage based on age	100%	
Director/Participant: Unique registration links for on-site registration and private events	95%	Integration testing with new state machine changes
Service: Automatic participant spreadsheet updates on-site	50%	Implementation and testing in combination with AWS state machine for finalized events
Director/Accountant/Admin: Update event state diagram and implement roles.	50%	Frontend updates to director UI to match new backend functionality. Implement Accountant and Admin roles and functionality. Volunteer role may be added in future. More details below

Milestone 4 (Feb 14)

- Finish implementation, test, and demo live, periodic spreadsheet updates from database as registration occurs on-site
- Implement, test, and demo director UI updates to reflect the changes introduced by the new state diagram.
    - Directors can request deletion of an event that has not been Finalized
    - Finalized events are displayed in their own table with a column showing their status (*InSession*, *Happening*, *Occurred*, etc.), which automatically updates depending on the start and end time of the event
    - Completed events (reviewed by accountant) are moved to their own tab, separate from active events
- Create surveys for participant and director beta testers using Google Forms and

link to each UI via a feedback button
- Implement, test, and demo Accountant UI
    - Create new GSuite account for Accountant role
    - Ability to mark events that have occurred or been approved for deletion as *Completed* after reviewing payments
- Implement, test, and demo payment management system for Accountant role
    - Allow searching of payments by:
        - Participant
        - Event
        - Date
        - Date range
    - Ability to export payments to spreadsheet

Milestone 5 (Mar 21)

- **Begin beta testing (late February)**
    - Canary deployment - small group of people at a time
    - Start with private event, limited heart screening event
- Implement, test, and demo admin UI
    - Create new GSuite account for Admin role
    - Ability to:
        - Review all event details for requested deletions
        - Deny requests for deleting events
        - Accept requests for deleting events
- Collect evaluation results
- Create poster for Senior Design Showcase

Milestone 6 (Apr 18)

- Test/demo of the entire system
    - Address any bugs that arise though beta testing
- Evaluation results
- Create user/developer manual
- Create demo video

Task Matrix (Milestone 4)

Task	Carlo	Chris	Drew	John	Noah
1. Finish implementation,	90%		10%		

test, and demo automatic spreadsheet updates for on-site use					
2. Implement, test, and demo director UI updates to reflect the changes introduced by the new state diagram.			100%		
3. Demonstrate environment setup for beta testers (directors/participants)			100%		
4. Create user survey for directors and participants using Google Forms and link to UIs with a feedback button	5%	80%	5%	5%	5%
5. Implement Accountant UI				50%	50%
6. Implement Accountant payment search and exporting functionality	33%			33%	33%
7. Begin collecting evaluation data	20%	20%	20%	20%	20%

Task Descriptions

- Task 1: Finish implementation, test, and demo automatic spreadsheet updates for on-site use
    - Create a spreadsheet that is published to the web
    - Export on-site sign up data to the published sheet
        - Published sheet will be updated every time a new sign up occurs or in certain timed intervals
    - Create a spreadsheet to be used on the local testing computers on-site
        - Create a disk partition to share the sheet with multiple local computers

    - Connect the local sheet to the published sheet and enable an automatic refresh
- Task 2: Implement, test, and demo director UI updates to reflect the changes introduced by the new state diagram.
    - Change director UI to display both InProgress (Unpublished, Published) and Finalized (InSession, Happening, Occurred) events
    - Add a status column to Finalized event table, so that a director can track the state of each of their events
    - Sort each table by date, so that older events are displayed at the top of the tables
    - Add pagination, so that no more than 20 events are loaded by default. The director will be able to scroll or click a button to fetch more events from the database
    - Move Completed events to a separate tab in the UI, so that directors have access to past event details and a confirmation that all payments were received by participants
        - Accountants will be responsible for marking Occured events as Completed once all payments have been received (future feature)
    - Add an action to request deletion of an InProgress event
- Task 3: Demonstrate environment setup for beta testers (directors/participants)
    - In order to prepare for beta testing in the next milestone, we will demonstrate that we can deploy and AWS environment for both directors and participants
    - 1 environment per group of beta testers
- Task 4: Create user survey for directors and participants using Google Forms and link to UIs with a feedback button
    - In order to evaluate user experience, we will allow beta testers access to surveys along with open-ended feedback for continuous improvement.
    - The surveys will consist of mainly scale-based questions regarding the quality of our UI in terms of usability
    - Once the forms are completed, a feedback button will be created on the director and participant UI to link to the surveys, respectively
- Task 5: Implement Accountant UI
    - Create new GSuite account for Accountant role
    - Responsible for reviewing payments for events that have occurred or that have been requested to be deleted
    - Once all payments for an event have been accounted for, accountants have ability to move an event to a Completed state (final state in state diagram)
- Task 6: Implement Accountant payment search and exporting functionality

    - Allow accountants to search payments by:
        - Participant
        - Event
        - Date
        - Date range
    - Ability to export payments to spreadsheet
- Task 7: Begin collecting evaluation data
    - As time permits, team members will attend an upcoming event(s) and repeatedly time various tasks using the current registration/screening process
        - Depends on date and location of upcoming events
    - This task will be continued into next milestone as we begin beta testing and time our system in practice

Approval

- "I have discussed with the team and approve this project plan. I will evaluate the progress and assign a grade for each of the three milestones."

Signature: _____ Date: _____