

Final Report

Public Health App, Team Coding Touchdown

Summary of the Project

This section summarizes the project as implemented, including the main customer need and how the application meets it, including the stakeholders. Subsections of this section give information about customer needs, the application's meeting of the requirements, and the stakeholders, respectively. This section can be used to evaluate the implementation of the project by being compared to the iteration 0 report.

Customer Need

The primary customer need can be summarized as follows:

In general, customers need a web-based application that screens health determinants on an individual level at primary care visits. It then connects them automatically to local resources if screening positive for a health disparity. The "health determinants" here mean the social and behavioral factors that affect the health of a group of people, such as health literacy, food insecurity, housing, transportation, mental health access, etc. For example, if screening positive for food insecurity, the app should automatically refer them to the closest food bank based on the patient's physical address.

Essentially, the customers need us:

- Creating a platform to ask different sets of screening questions with the screening documentation available online
- Creating a secure database that can store information
- Processing the information collected to determine positive or negative screen
- Using specific information collected (positive screenings, patient's address) to optimize and filter through automated searches to find local resources.
- Relaying this information back to the patient, the patient's doctor, etc.

The Application

To meet the requirements of customers, the application provides a register/login feature for the users to create an account for the application, an edit profile feature for the users to edit their profile, a list of surveys for the users to take the surveys, a feature to let the users check their survey results, a feature to show the map of the needed local resources. For details of the implementation, see the following.

First, a quick look at the technical stack of the application. The application uses Node.js for backend development and React.js for frontend development. For the database, the application uses MongoDB. For testing, we chose jest and postman. What's more, the application is deployed on the Heroku platform.

Then let's look at the register/login feature. This functionality has three parts: register, login, and password reset. The register function asks the user to fill in the information of First Name, Last Name, Phone Number, Email, and Date of Birth. These pieces of information are necessary for storing the users' data in the database. The register feature will also let the user

create a password that fits the requirements for security purposes. The users' passwords are encrypted before storing into the database. The login feature will let the users log into the application with the account information. The password reset function lets the users the option to reset the password if the password is forgotten.

After the user logs into the application, the user will be directed to the dashboard, which will let the user access different services on the website. The user can access all the modules of the application, which includes a list of surveys, edit profile, maps, and the survey results in the dashboard. The user can edit their address in the edit profile module, the address is important for the application because the necessary local resources are recommended based on the address. The user can always update the address in the module of edit profile.

We have created a list of sample surveys with public-owned documentation for surveys available online. The application provides the functionality to let the user take the survey. The user could complete the surveys in the application, and the survey will change dynamically based on the user's answer, which helps proceed with the survey based on the user's situation. After completing the survey, the user will be redirected to the dashboard page, and the result of the survey will be sent to the backend database, where the score of the user's survey will be calculated, and the corresponding classification of the result will be generated based on the score. The result of the survey can be checked on the survey page, which can be found in the dashboard. Therefore, the application meets the requirements of letting users take the surveys and check the result. Moreover, the data are encrypted before storing into the database, as requested by the client.

What's more, the application provides a map to show the location of the local resource to the user, which can also be found in the dashboard. The functionality of providing the maps is developed using the Google Maps API. In this stage, the sample location, Baylor Scott and White Medical Center in College Station, is shown on the maps. To be more specific, the recommendation system of the resources is not implemented here, which can be left as a future endeavor of the legacy project. We will talk more about this issue in the next section.

To sum up, the application meets the requirements of creating the requested web application, as described above. We have met the customers' requirements of creating a platform letting the users take the survey, constructing a secure database that can store the information safely, processing the information collected and relaying the result back to the user, and showing the local resources to the user. The possibility of extending the application is left in the part of building a recommendation system that can find the necessary local resources more smartly, while more functions that were not requested by the customers at the beginning were also implemented, like the edit profile feature and the password reset feature, which helps enrich the functionality of the application.

Stakeholders

Stakeholders of this application include patients, physicians, social workers, and psychiatrists. The stakeholders could also be considered who will be invested in or impacted by such an application.

Description of User Stories

User Story: Sign Up/ Login

Points: 10

Implementation Status: Implemented successfully, with more features added

Description: This is the user story that let the user create the account and log in to the application.

Changes: At the beginning, we only made the user story of sign-up, but we ended up implementing the sign-up feature coming with a login feature and a feature that could help the user to reset the password.

The following pictures show the lo-fi mockup and corresponding screenshot of this user story, which helps explain the content and changes of the user story.

A hand-drawn lo-fi mockup of a 'Sign Up' form. The title 'Sign Up' is at the top. Below it are five input fields labeled 'First Name', 'Last Name', 'Date of Birth', 'Email', and 'Phone'. At the bottom, there is a 'Sign Up' button circled in a hand-drawn oval.

Three screenshots of the 'Public Health' app interface. The first screenshot shows the 'Log in' screen with fields for 'Your Email' (filled with 'jjajun_sun@outlook.com') and 'Your Password', a 'Log in' button, and links for 'Not a member yet? Register' and 'Forget password? Reset password'. The second screenshot shows the 'Please Sign Up' screen with fields for 'Your First Name', 'Your Last Name', 'Your Phone Number', 'Your Email' (filled with 'jjajun_sun@outlook.com'), 'Your Date of Birth' (with a date picker), 'Your Password', and 'Confirm Password'. It includes a 'Sign up' button and a link for 'Already a member? Login'. The third screenshot shows the 'Public Health' app's password requirements screen, listing rules: 'At least 6 characters', 'At least one upper case', 'At least one lower case', 'At least one number', and 'At least one of the special characters'. It has a 'Submit' button.

User Story: Answer Screening Questions

Points: 10

Implementation Status: Implemented successfully, with enriched functionalities implemented together

Description: This is the user story that let the user take the survey, which means answering the screening questions

Changes: No exact changes were made, but we have implemented the user story with reasonably enriched functionalities, like dynamically changing the survey depending on the user's answer. What's more, we have made 6 sample surveys in total.

The following pictures show the lo-fi mockup and corresponding screenshot of this user story, which helps explain the content and changes of the user story.

A hand-drawn lo-fi mockup of a survey form. At the top, the word "Survey" is written in a large, cursive font. Below it, there are four question entries. Each entry consists of a circled number followed by the text "Question 1", "Question 2", a vertical ellipsis, and "Question N". Each question entry has a rectangular input box below it. At the bottom of the form, the text "Complete Survey" is written in a cursive font and is enclosed in an oval.

A screenshot of a survey application. At the top, there is a text prompt: "Next, please tell me whether the statement was often true, sometimes true, or never true for your household in the last 12 months—that is, since last November." Below this, there are three question cards. Each card contains a question number and a statement, followed by four radio button options: "Often true", "Sometimes true", "Never true", and "I don't know, or I refuse to answer". The questions are: 3. "We worried whether our food would run out before we got money to buy more." Was that often true, sometimes true, or never true for your household in the last 12 months? *, 4. "The food that we bought just didn't last, and we didn't have money to get more." Was that often, sometimes, or never true for your household in the last 12 months? *, and 5. "We couldn't afford to eat balanced meals." Was that often, sometimes, or never true for your household in the last 12 months? *. At the bottom of the screen, there are two buttons: "Previous" and "Complete".

User Story: Check Screening Results

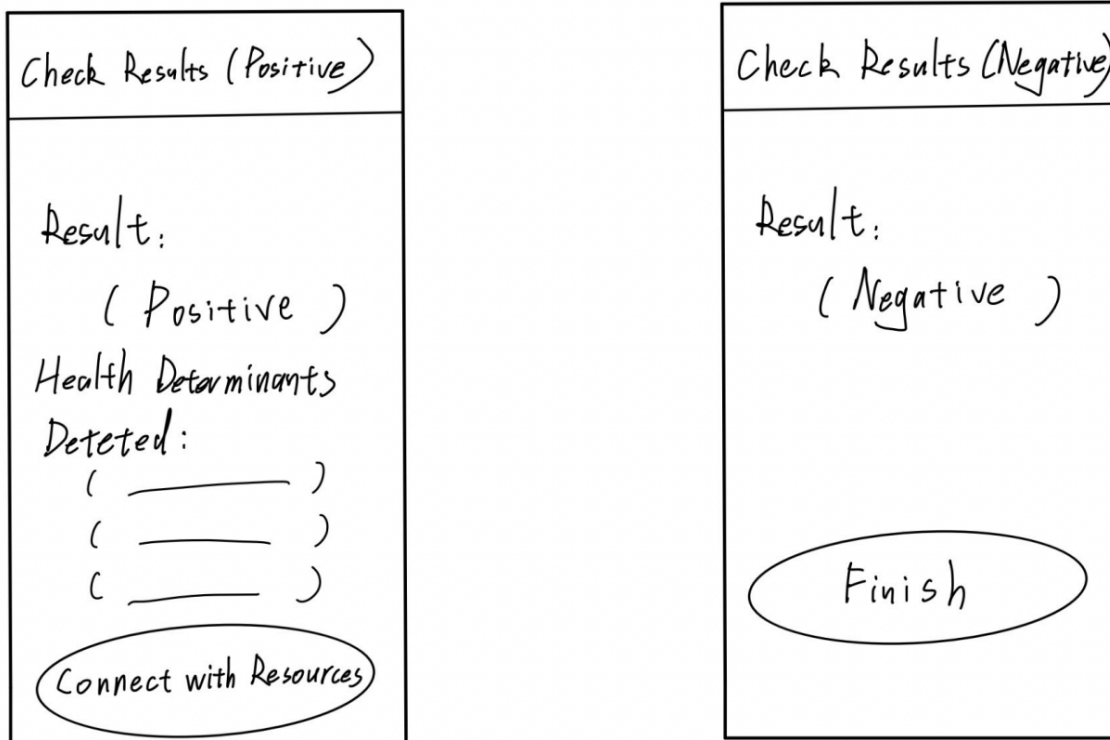
Points: 10

Implementation Status: Implemented successfully

Description: This is the user story that let the user know about the result of the surveys

Changes: We ended up showing the result of the survey in the location of the dashboard in the application, which can be counted as a little change to the user story planned because the original user story was designed as showing the result of the surveys right after the user finishes the survey. We ended up implementing the user story as it looks now because we think that it is a more efficient way to display and store the results, and we got a bit stuck trying to implement the user story as planned.

The following pictures show the lo-fi mockup and corresponding screenshot of this user story, which helps explain the content and changes of the user story.



Survey Result		
Here is a subtitle for this table		
SURVEY TYPE	SURVEY RESULT	SURVEY CREATED DATE
household-food	Low food security	6 December 2022
household-food	High food security	7 December 2022

User Story: Connect Necessary Resources

Points: 10

Implementation Status: Implemented partly, can be extended

Description: This is the user story that lets the user be connected with the necessary local resources

Changes: This is the user story that did not get implemented perfectly, to be honest. The planned user story is sort of a recommendation system that can “optimize and filter through automated searches to find local resources using specific information collected”. The implemented user story is a maps page that shows a sample local resource, Baylor Scott and White Medical Center in College Station. Google Maps API is utilized to implement the user story. The difficulty of implementing the ideal “recommendation system” exceed our estimations, we have tried our best but with the limited time and resources, what we can deliver as of now is the maps page.

The following pictures show the lo-fi mockup and corresponding screenshot of this user story, which helps explain the content and changes of the user story.

Connect with Resources

Resource 1:

Resource 2:

Resource 3:

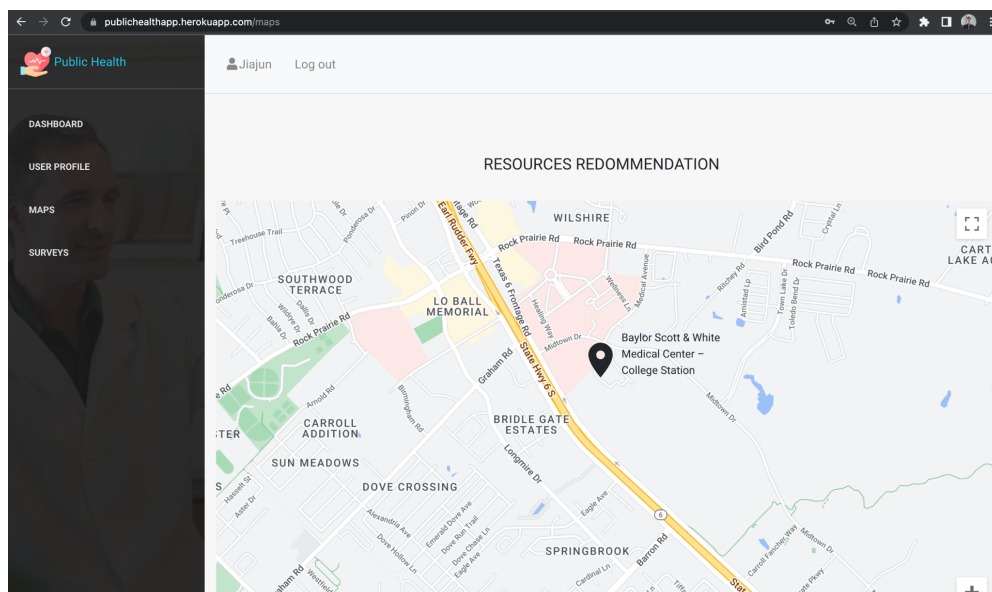
⋮

Connect with Resources

Resource Name: ()

Resource Info: ()

Resource Contact:



Summary of each Scrum Iteration

Iteration 0

Points completed: N/A

Tasks accomplished:

- Original User Stories
- Lo-fi UI and storyboard
- The whole initial set-up of the project

Iteration 1

Points completed: 10

Tasks accomplished:

- Development of the Signup/Login page
- Construction of the database
- Corresponding tests

Iteration 2

Points completed: 10

Tasks accomplished:

- Development of the first two surveys
- Development of the dashboard
- Development of the edit profile feature
- Starting development of the update password feature
- Corresponding tests

Iteration 3

Points completed: 10

Tasks accomplished:

- Development of the backend of the checking result feature
- Upgrading the created survey
- Finishing the password reset feature

Iteration 4

Points completed: 10

Tasks accomplished:

- Creating 4 more surveys
- Upgrading the surveys created before
- Modifying and upgrading the database

Iteration 5

Points completed: 10

Tasks accomplished:

- Development of more features of the checking result functionality
- Development of the maps page
- More upgrading and revision of the application

Team Roles

Scrum Master: Yuhao Ye

Product Owner: Jiacheng Zhao

Team Members: Xiangrong Li, Ding Ning, Jiajun Sun

No changes were made to the team roles during the project. Team members were happy with the team role assignment and collaborated efficiently throughout the semester, therefore, we kept the team roles to keep the chemistry.

Record of Customer Meetings

Meeting #1 Sep-16

The first meeting, we introduced ourselves and extracted the user stories from the customers' needs.

Meeting #2 Sep-23

We demonstrated the user stories we created and got the customers' feedback.

Meeting #3 Oct-7

We demonstrated the Signup/Login feature to the customer.

Meeting #4 Oct-21

We demonstrated the taking survey feature to the customer.

Meeting #5 Nov-4

We demonstrated the reset password feature to the customer.

Meeting #6 Nov-17

The customer couldn't attend the meeting, but we sent the customer a demo video afterward to demonstrate the whole features of the application at that time, including the checking result feature.

Meeting #7 Nov-29

We discussed the checking result feature, the maps feature vs the "recommendation system" feature, and the plan for the rest of the semester.

BDD/TDD Process

We use the tool jest to write unit tests and we use the BDD process in development.

After we decide which story needs to be implemented in a particular iteration. We converted our user story into several different test cases. We think about the scenario that our application could fail and write a failing unit test that nonexistent code we wish we had. Then, we write code to pass each unit test. After we implement the code, we also use the jest built-in tool to check code coverage to help us determine which part of the code needs more testing.

TDD helps us to locate the bug and errors in our backend more accurately. It also improves the efficiency of development.

Configuration Management Approach

We use some source code management on Github to make sure that the development version of the code doesn't conflict with the production version of the code. We have two branches for our application. The first one is dev-master and the second one is master. All the team members in the team could only update the code in the dev-master branch. After members push their code on the dev-master branch. The merge request will be made. The request will be merged if and only if it passes all the test cases.

Issues Encountered


We meet some issues when we use GitHub for development and Heroku for deployment. For GitHub, the most common issue is the merge conflict. For Heroku, The issue I meet is the dependency install conflict and the wrong version of the node js engine. The wrong version of node js will lead to the missing module or package of the application. Another issue we meet for the development is in jest. It takes us some time to figure out how to implement the mock MongoDB database correctly.

Other Tools Used

We use the SurveyJs package to implement the survey part of our application. After a user completes the survey, SurveyJS API could convert the result of the survey to JSON format. Hence, the result could be transferred to the backend of the application. It is also easy for us to save this result in the database easily.

We also use the Axios package in the frontend to make HTTP requests to the backend. We use Expressjs in the back end to write controllers to handle the HTTP request. We also use redux to manage the state inside our react framework in the frontend. We also use google map API to integrate google Maps into our application.

Note to the Repo Content

 Patrickyyh Merge pull request #46 from Patrickyyh/dev-master ... 7631472 1 minute ago 🕒 269 commits			No pro
client	Merge pull request #46 from Patrickyyh/dev-master	1 minute ago	📄
coverage	password reset and fix the logo issue	15 hours ago	☆
db	client side survey creation by reacjs	last month	👁
documentation/Fall2022	it 5 tar upload	6 days ago	👤
email/template	initial commit for the functionality of password reset	last month	Rel
errors	integrate token and add userid inside database	24 days ago	🔍
middlewares	delete some unnecessary console log	20 days ago	Cre
models	password reset and fix the logo issue	15 hours ago	—
routes	password reset and fix the logo issue	15 hours ago	Pac
test	redux refactor	2 months ago	No p Publ
.DS_Store	Google Signin update	10 days ago	—
.Rhistory	database	2 months ago	Co
.gitignore	u	23 days ago	👤
Procfile	deploy config	2 months ago	—
README.md	Update README.md	last month	Env
app.js	password reset and fix the logo issue	15 hours ago	🔗
package-lock.json	Google Maps update	7 days ago	—
package.json	update 3	2 days ago	Lar
server.js	fix the bugs	23 days ago	—
temp.js	fix survey controller	last month	—

Here is a picture of our project. To run our project, download the source code on your local machine. Make sure to run `npm install -- force` to install all the necessary packages. You also should run `cd client` to change the working directory to the client folder to run `npm install -- force` again because the package for the client is separated from the package inside the server. In the test folder, we implement the initial setup for the jest. We implement and create the mock database in the setup file inside this folder. In the package.json file, you could see all the dependencies we install for the application. There is another package.json file in the client as well which shows the dependencies installed in the client. The routes folder contains all the controllers we implemented for handling the HTTP request sent from the front-end. In the middleware folder, auth.js is responsible for authentication. It will extract the JSON web token from the header to verify if the user is authorized to use the service. The models folder contains the mongoose document and schema for the User and survey model. In the client folder, we implement the front-end part of the implementation using reactJs framework.

Links

Pivotal Tracker: <https://www.pivotaltracker.com/n/projects/2596722>

GitHub repo: <https://github.com/Patrickyyh/public-health-app>

Heroku deployment: <https://publichealthapp.herokuapp.com/register>

Presentation and Demo Video: https://youtu.be/_5Db-KtHGMA