SOFTWARE ENGINEERING APPLICATIONS PRODUCT PORTFOLIO

ESOF 423
Spring 2024

Michael Belmear
Ben Heinze
Colter Linder
Fletcher Phillips
Josiah Shirley

SECTION 1 (PROGRAM):

This is a product created for the HRDC out of Bozeman, Montana. The HRDC came to us with the request to have a web page created for the ability of someone who is living in one of their rentable properties to be able to submit a maintenance request form to the maintenance department of the HRDC. The current way they were handling maintenance requests was not working for the HRDC and we were tasked with coming up with a solution. We decided that we would build a React App using node.js, HTML, CSS, and javascript as the stack that would be deployed through Github pages. HRDC's goal is to have a simple and robust maintenance request form that anyone can fill out regardless of age and ability. We rationalize our minimalism through the goals of HRDC.

This website was a design driven product that was created with the philosophy that making a smart and simple product would best fit the needs of the client. When navigating to the website that the product is deployed to you are prompted with a form to fill out that contains information such as the person's name, email, property, urgency, message, etc. The page also contains a button that will present some basic documentation to help a person filling out the form understand what all is required. After the user submits the form, the contents of the form will be sent using EmailJS to the employees of the HRDC and they are able to do their triage on the work requested. You can find the live product at https://fphillips22.github.io/test-pages/ and the source code at https://github.com/423s24/Group_4.

SECTION 2 (TEAMWORK):

Team Member 1: Contributed primarily through the design and implementation of the landing/confirmation page. Worked primarily through CSS (and some constant styling methods within React) that styled the relevant react components. Product backlog work: This team member worked on implementing/building and designing the Landing and Confirmation page components. This team member also helped with the product's responsiveness as well as making sure the artifacts were up to date and consistent with release requirements during each sprint. Estimated percentage of total time spent on the project: 20%.

Team Member 2: Contributed primarily through email integration and form fields of the product through EmailJS. Product backlog work: This team member worked on field components for the forms, created user documentation, created a class diagram of the system, created admin documentation, and set up EmailJS and incorporated that into the product. Estimated percentage of total time spent on the project: 20%.

Team Member 3: Contributed primarily through design. This team member worked on creating a high fidelity sketch for the form and confirmation page, helped implement the sketch into a CSS file, and has helped improve the design of the page through critiques of the employer. I have worked with Adobe XD, I have learned more about html and CSS, and have improved my knowledge and understanding of GitHub. Estimated percentage of total time spent on the project 20%.

Team Member 4: Contributed primarily to the project by taking care of the deployments and releases of the product. Product backlog work: this team member worked on the form complements and building the CSS for the form, CI/CD, All of the releases, github pages, the develop branch, and some small styling issues. This team member also gave a lot of best practice tips and ideas for the product's vision. Estimated percentage of total time spent on the project: 20%.

Team Member 5: Contributed primarily to the actualization of the aesthetic of the final product. This team member worked primarily with CSS to style the final product in a way incorporated agreed upon elements from the original mockup. Product backlog items: design form component, build form component, form responsiveness. Estimated percentage of total time spent on the project: 20%.

SECTION 3 (DESIGN PATTERNS):

Our project heavily uses React for the View layer (building the UI components), the EmailJS API for sending emails (interfacing with external services), and custom logic within React components for handling user interactions and form submissions, thus it best fits the MVC (Model-View-Controller) design pattern.

1. **Model:** The Model component represents the data and business logic of the maintenance request system. It encapsulates the essential details captured by the form, such as addresses, messages, and categories like electrical or plumbing issues. Additionally, the Model may include validation logic to ensure the integrity and accuracy of the data before processing.
2. **View:** The View component corresponds to the user interface (UI) of the maintenance request form. This includes the design and layout of the form elements built using React.js. The View provides users with a visually appealing and intuitive interface to submit their maintenance requests efficiently.
3. **Controller:** The Controller acts as the intermediary between the Model and the View. In our software product, the Controller consists of the logic responsible for handling user interactions within the React components. This includes event handlers for form submissions, processing user input, and triggering actions such as sending email notifications to HRDC technicians via the EmailJS API.

The MVC pattern organizes our software product into three distinct components, each with its own responsibilities. The Model manages the data and business logic, the View presents the user interface, and the Controller coordinates the interaction between the Model and the View, ensuring seamless functionality and user experience. This separation of concerns facilitates modular development, maintenance, and scalability of the maintenance request system.

SECTION 4 (TECHNICAL WRITING):

How to Obtain Source Code:

- Our code will be available at the following GitHub link: [HRDC Maintenance Portal Repository](https://github.com/423s24/Group_4)

Planned Directory Structure:

- The directory structure will follow the typical React framework structure with components, etc.

Planned Builds:

- React
- NPM packages
- The rest is TBD...

Developer Testing:

- We will ensure a placeholder webpage compiles without error.
- Unit tests will be written.

Automated Build and Test:

- Continuous Integration will be implemented.

How to Release a Version:

- We will define individual goals for each sprint and categorize them as either minor updates or patches, depending on the scope.
- We are using the standard version notation of Major.Minor.Patch, with the final release of our website being the first major update.

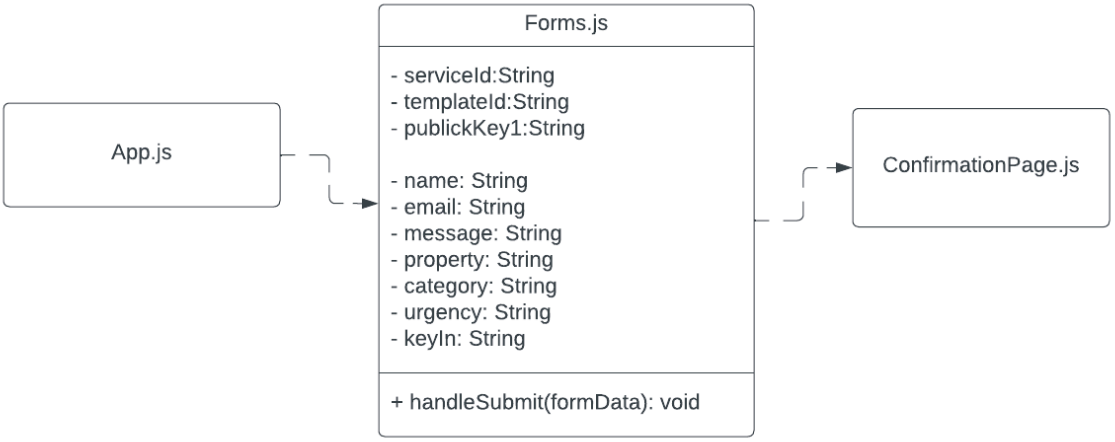
Bug Tracker:

- We will use the Issues tab in the GitHub repository to manage and track current bugs.

Comments in Source Code:

- Pull requests will outline changes to the code base in a short summary describing what they add or change in the product.
- In the code, ensure that code is properly explained in a short comment if a chunk of code does not have clear functionality when reading at first glance. Functions should have a brief description of what they do.

SECTION 5 (UML):



HRDC's goal is to have a simple and robust maintenance request form that anyone can fill out regardless of age and ability. We rationalize our minimalism through the goals of HRDC. This simple UML class diagram demonstrates how each component interacts with one another. React uses App.js as the foundational root. By running the React app, App.js will launch Forms.js, where users will input their contact information such as name and message. The message will be sent by using the EmailJS API using serviceId, templateId, and publicKey1 as fields. Once the "submit" button is pressed, the user will be redirected to ConfirmationPage.js where they will receive a "Thank you! Your form has been sent, we will get back to you as soon as possible.". The confirmation page is the final step of the process.

SECTION 6 (DESIGN TRADE-OFFS):

We were tasked with creating a webpage that includes a maintenance form that is easy to understand, easily accessible, and keeps track of requests. When designing our website we wanted to make the front end as well as the backend easy to understand. Because we wanted to do this we had to make sure we kept things simple throughout the process. We decided to just make a form and confirmation page and a way to send those emails to their maintenance guy. The trade off here was a more fleshed out page where you need an account and creating a database that stored all the emails. We decided against this because we wanted the back end to be incredibly easy for the HRDC employees to manage.

SECTION 7 (SOFTWARE DEVELOPMENT LIFE CYCLE MODEL):

1. **Planning**: The planning portion of our software development life cycle involved gathering information from the client about the desired end result of the software product. Using this information, we decided to use the React Web Framework to develop a responsive webpage that would be hosted on the client's servers. We also planned on using EmailJS for email sending functionality.
2. **Analysis**: We analyzed the capabilities of EmailJS concerning things such as: How many emails can be sent per month? How large can the emails be? Can photos be attached to emails? How can we customize the emails to present the information contained therein properly? After these questions were answered, we felt that EmailJS was the best service for email sending functionality.

3. **Design**: Using Adobe XD, we quickly developed a prototype. This prototype was shown to the client for approval. The client liked certain elements of the prototype and these elements were carried over to our final release product.
4. **Implementation**: We used the React Web Framework to implement our app. The app consists of several custom React Components such as: a Form component, a LandingPage component, a ConfirmationPage component, and a UserDocumentation component. These components are dynamically injected into the app during each session.
5. **Testing and Integration**: We regularly met with the client to assess our progress. In order to test our app, we allowed the client to use the form during each of these meetings. In addition to this, each team member approached someone who did not have a background in computer science and asked them to use the form and assess its usability. In order to integrate our progress throughout the app development cycle, we used GitHub for our version control system.
6. **Maintenance**: In order to ensure that our app remains functional for as long as possible, several group members are maintaining email contact with the client. These group members are local residents of the client's hometown meaning that any of them would be able to deploy to work on the app within a reasonable amount of time. In addition to this, the client was given access to the developer documentation. This will allow any developer in the future to work on the app. In addition to this, the client was also given administrative documentation to aid in their understanding of how to interact with app after the semester is over.