Boston University
Electrical & Computer Engineering
EC463 Senior Design Project

First Prototype Testing Plan

by

Team 3
DivaDocs

Team Members

Paige DeVeau | pdeveau@bu.edu
Mya Turner | mjturner@bu.edu
Akhil Sehgal | asehgal1@bu.edu
Vinay Metlapalli | vinaymet@bu.edu
Abhinoor Singh | abhinoor@bu.edu

Required Materials

Software:

- Front End
    - Wireframes
        - Home Page
        - Application Form
    - React.ts Components
        - NavBar
    - Home Page
- Backend
    - Python3 MongoDB Database Scripts
    - MongoDB Database Environment
    - Python3 Models
    - Dummy JSON Object for the Application Form
    - Python3 FastAPI Post Request
    - Postman Test Requests

Set Up

The setup for our project is split up into three different components:

- Design
    - The design for the front end uses a software called Figma. Inside of Figma we have created mock designs of what the website should look like, and how it should function. With Figma we are able to show what the buttons on the page look like, as well as where they take you. The wireframes specify the size of the HTML objects, the colors, as well as the different components needed for each of the web pages.
    - The design for the backend is a system design diagram which shows how the front end connects to the backend using an API, as well as where the website is hosted. The system design also includes the connection to the database that we are using.
    - The designs are sent to the client for approval before we begin programming them. Currently we have approval for the home page as well as the application form for the backend. We are still awaiting the final approval for how the application will look in the front end.
- Front End
    - Using React.ts the front end is split up into multiple pages with various components. Currently we have the homepage completed. We also have a navbar at the top of all the pages to be able to visit different pages easily.
- Back End

- There are several python scripts, each script serves its own purpose.
    - Database Script allows a connection between the API and the MongoDB database. Data can be written to, and read from the database.
    - The FASTApi script has a post request endpoint setup which accepts a json object and uses pydantic to parse the JSON object and store them into python classes.
    - The models script uses pydantic to create python classes that have the various fields found in the application form.

<u>Pre-Testing Setup Procedure</u>

Front End:

1) Run react with all components from the working directory.
    a) npm start

Back End:

1) Run python3 script using uvicorn that runs the FastAPI backend.
    a) uvicorn main:app --reload (must be in working
2) Open Postman with preloaded POST requests to the end point for the application form
    a) The POST requests will have different request bodies to show the different aspects of the API.

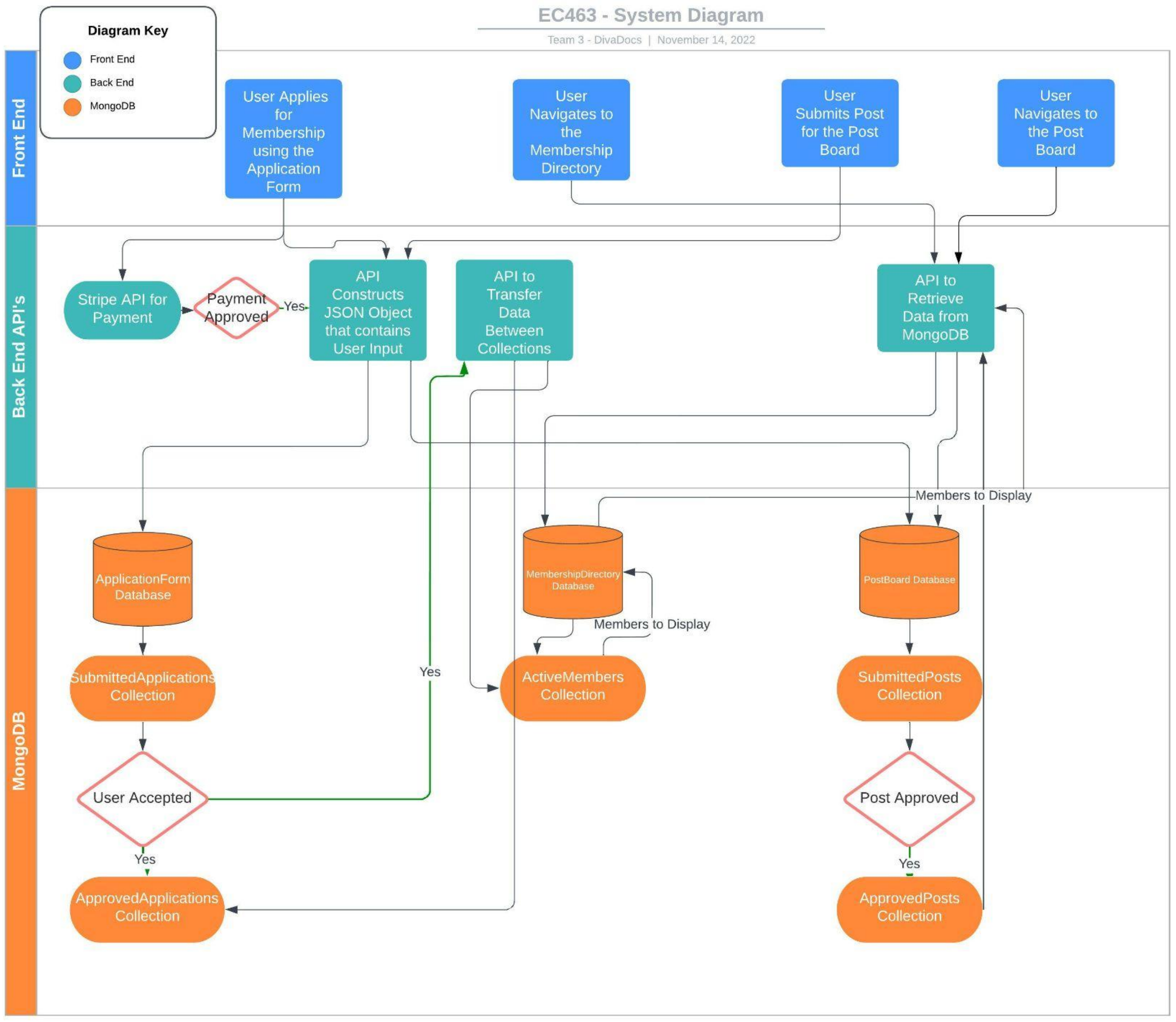
EC463 - System Diagram
Team 3 - DivaDocs | November 14, 2022
Diagram Key
Front End
Back End
MongoDB
Front End
Back End API's
MongoDB
User Applies for Membership using the Application Form
User Navigates to the Membership Directory
User Submits Post for the Post Board
User Navigates to the Post Board
Stripe API for Payment
Payment Approved
Yes
API Constructs JSON Object that contains User Input
API to Transfer Data Between Collections
API to Retrieve Data from MongoDB
Members to Display
ApplicationForm Database
MembershipDirectory Database
PostBoard Database
Members to Display
Yes
SubmittedApplications Collection
ActiveMembers Collection
SubmittedPosts Collection
User Accepted
Post Approved
Yes
Yes
ApprovedApplications Collection
ApprovedPosts Collection

Figure 1: System Design of BWMDN Website

Testing Procedure

Test Different Components

- Front End:
    - For the frontend, the component that needs to be tested is the navbar on the top of the screen. To test this component, one will click on each item within the navbar and make sure it takes the user to the correct webpage.

1) Run react.ts code and open localhost.
2) Click through the various pages, and scroll on the home page.
3) Show that all the pages open, show design, and show navbar.

- Back End:

1) Run the FastAPI script.
2) Open postman
3) Show good post request with valid json object
4) Show sending a bad post request with a single field empty
5) Show sending an empty json object
6) Show GET request to pull data from database
7) Show MongoDB environment with the data that was just sent

Measurable Criteria

The criteria for a successful test is as shown:

For Front End

1) All buttons must be clickable, and must take the user to the correct page.
2) All objects must appear correctly, the user should not notice any significant bugs on the home page.

For Back End

1) A good POST request must result in a 200 OK response from API
2) The json object should appear in the mongodb database after successful request.
3) A bad POST request with either an empty request body or a missing field should result in "422 Unprocessable Entity", and an error message stating which field is missing should be returned.
4) Upon sending a bad POST request, the data should not go into the database.
5) A good GET request should pull all of the current applicants data from the database and return that data.
6) All POST or GET requests should complete in under 1 second.

Score Sheet

FRONT END SCORE SHEET

Navbar Option	Webpage	Page Open?(Y/N)
Membership/Join Us!	http://localhost:3000/join	Y
Membership/Directory	http://localhost:3000/directory	Y
Membership/Resume Bank	http://localhost:3000/resumebank	Y
Membership/Find An Expert	http://localhost:3000/findanexpert	Y
Resources/Resume Review	http://localhost:3000/resumereview	Y
Resources/Headshot	http://localhost:3000/headshot	Y
Resources/Find An Expert	http://localhost:3000/findanexpert	Y
Resources/Coaching	http://localhost:3000/coaching	Y
Community/Diva Docs	https://www.divadocsboston.com/	Y
Community/Events	http://localhost:3000/events	Y
Sign In	http://localhost:3000/signin	Y
Join	http://localhost:3000/join	Y

BACKEND SCORE SHEET

Request/Action	**Expected Response**	**Achieved Expected Response (Y/N)**
POST Request with Good JSON Object	200 OK Data is added to database Applicant info sent is shown in Postman Response Body	Y
POST Request with Bad JSON Object (either empty or missing a field)	422 Unprocessable Entity Data is not sent to database	Y
GET Request Returns all data from SubmittedApplications collection	200 OK Data is shown in Postman response body	Y
All Requests Fulfilled in under one second	All requests should complete in under 1 second as shown in Postman.	Y

Results:

The initial testing for our front end and backend was broken down into test cases to ensure feature functionalities. By breaking down each into subsections and smaller tests, we can test the functionality of the website to prepare the MVP for deployment. In the front end, we focused on running all the active components from the directory and testing the links to each subpage. During our testing, we clicked on each of the tabs on the navbar, such as Join Us, Directory, Resume Bank, Headshot, Find an Expert, coaching, etc. Each of these link to a custom, easy to read URL that links to the correct page from the host. All front-end tests were successfully completed. The Backend initial prototype test was focused on testing POST Request with JSON objects. We attempted to push entered information for the membership application into a data object and return OK or Bad Request depending on the scenario. The information is received by our MongoDB database and the information is presented as a JSON object. In addition, GET request can pull data from the database, which will be displayed using our front-end to show the user's profile data. Our test cases for the Membership directory were successful and all requests were completed in under 1s. As presented in the scoresheet, it allows us to follow a set of test cases to ensure end-to-end functionality testing.

Our first prototype test was successful and allowed us to test the basic functionality of the website. The front end provides us with insight into how the user will flow through the storyboard of the application, and the backend is critical in organizing the database for users and creating a search query as we continue to build on our design. As we begin development for the membership application, our initial prototype gives us insight in how the information should be presented, organized, and processed.