Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User Manual

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

Table of Contents

Executive Summary

Black Women M.D. Network (https://blackwomenmdnetwork.com) is an innovative and empowering platform designed specifically for Black women doctors to connect, collaborate, and grow professionally. Our mission is to create an inclusive, supportive, and inspiring environment that fosters meaningful connections and opportunities for Black women physicians.

As a digital platform akin to LinkedIn, Black Women M.D. Network offers a membership-based service where users can sign up and gain access to an extensive network of like-minded professionals. The site asks its members to pay tiered-based dues to support the organization. Members can explore a diverse range of profiles, interact with other Black women doctors, and discover potential collaborations, mentorship opportunities, or job openings.

At Black Women M.D. Network, we believe in the power of representation and unity. Our platform acknowledges and creates a space for Black women doctors to network and excel in the medical field while encouraging and uplifting one another to reach their full potential. By fostering a sense of belonging and providing invaluable resources, we aim to bridge the gaps in representation and create a thriving community of Black women doctors.

1 Introduction

The Black Women M.D. Network offers crucial support and resources to Black women professionals. The platform provides a hub of accessible networking tools, including a resume bank for recruiters, connections to mentors, and opportunities for leadership and management training and networking events. Black Women M.D. Network presents Black women doctors with a space for personal and professional community and growth.

The platform was commissioned by Diva Docs, a 501c3 nonprofit organization that prepares Black women doctors for senior positions in medicine and healthcare through professional development programming, leadership training, one-on-one coaching, and member partner mentorships and sponsorships. Diva Docs is currently Massachusetts-based but will be transitioning its platform nationwide through the Black Women M.D. Network portal site (https://blackwomenmdnetwork.com/join). The website offers new users a platform to apply for one of the membership tiers, and, once accepted, view other approved members in a "Membership Directory." Users sign up using our portal and provide information about their background, work history, and resume. That information is then vetted by an internal system. The nature of our project requires an approach that has two major components: front-end development of website design, user experience, and the overall user interface of the website; and back-end development of the database infrastructure, where data from the website is stored and retrieved, and custom APIs are created to drive the functionality of the website.

Members of our team are divided into these two teams but work together to ensure that the back-end and front-end components continue to work in unison as overall development continues. Our approach ensures that our client will have a fully functional website capable of fulfilling the ambitions outlined above but can also handle nationwide user traffic. The homepage and overall website design are led by a professional UX/UI designer in conjunction with our team and the client. The website also fully facilitates the new user application process, account creation, and payment. This process presents the user with a client-designed form and a secure payment portal using Stripe API embedded into the website. The website's backend is robust, with a fully integrated MongoDB database environment and a suite of APIs. The MongoDB environment has several databases, with corresponding sub-collections to ensure that data is organized and can be retrieved efficiently. The APIs facilitate the transfer of data to and from the MongoDB environment and fulfill subtasks as required by the frontend.

2 System Overview and Installation

2.1 Overview block diagram

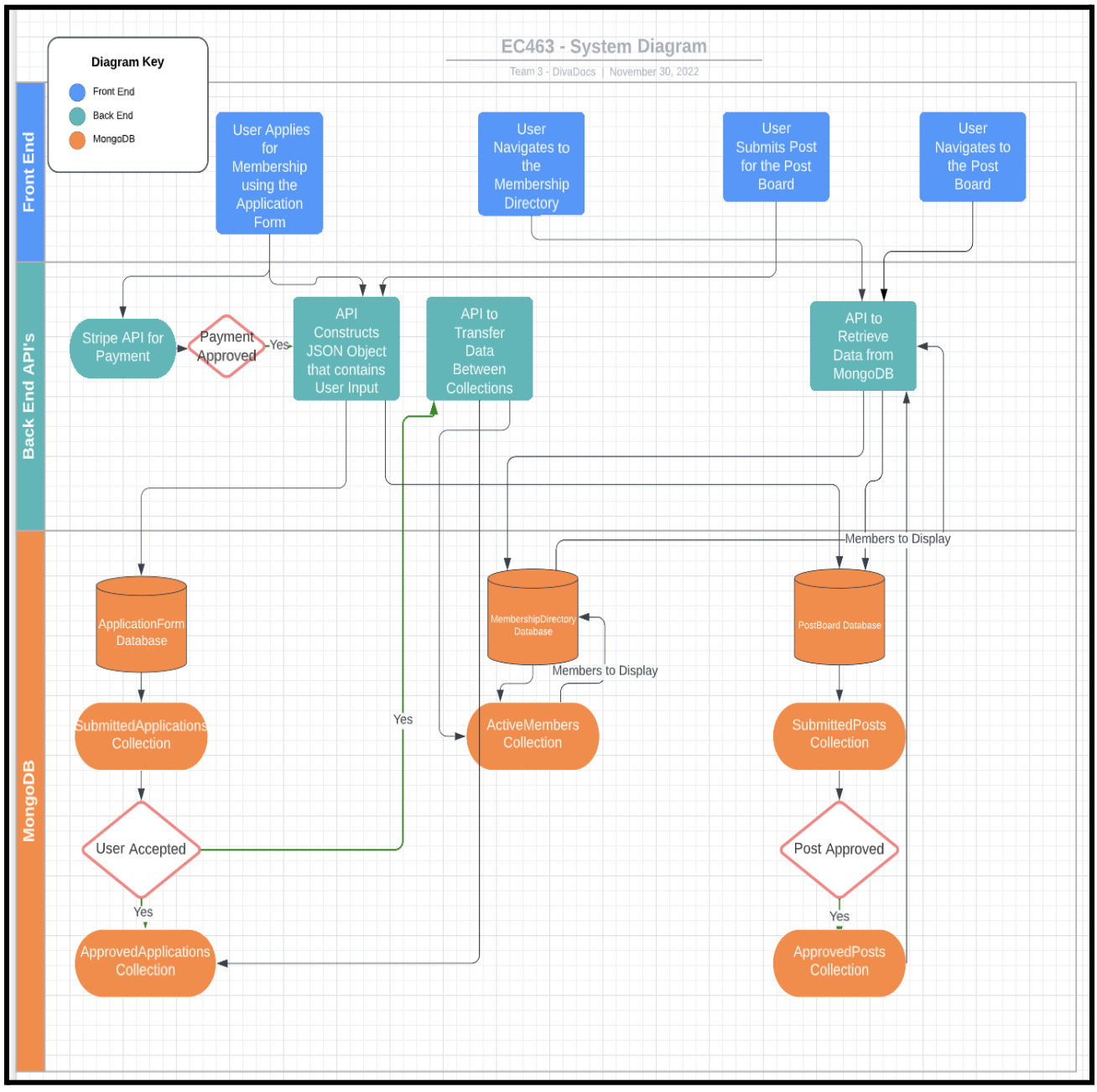


Figure 1: System Block Diagram of Frontend and Backend components

2.2 User Interface

The homepage of our platform is designed to provide users with an intuitive and engaging introduction to the features and resources available on our platform. The page has a clean, modern aesthetic that creates a welcoming and professional atmosphere.

At the top of the page, users find a navigation bar that provides easy access to all of the key features and resources on our platform, including "About", "Membership", "Resources", and "News", along with buttons to "Sign In" and "Join" to complete the membership application.

Figure 2: Homepage + Navbar

The rest of the homepage is designed to showcase the key features and resources of our platform, with clear and engaging calls to action that encourage users to explore and engage with the various tools and resources available to them.

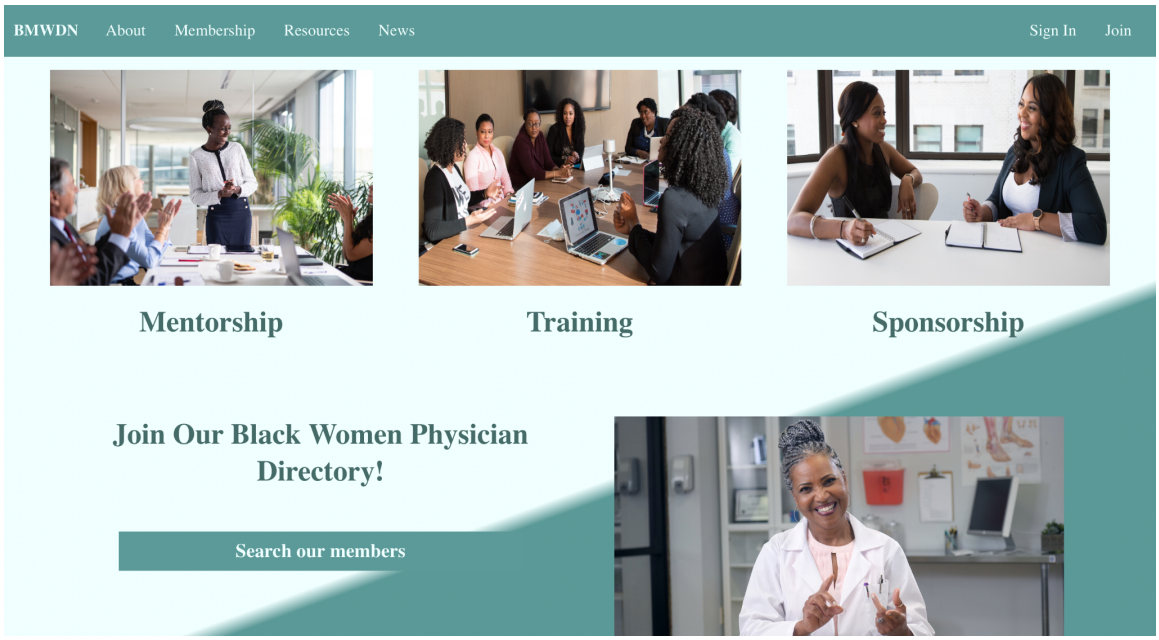
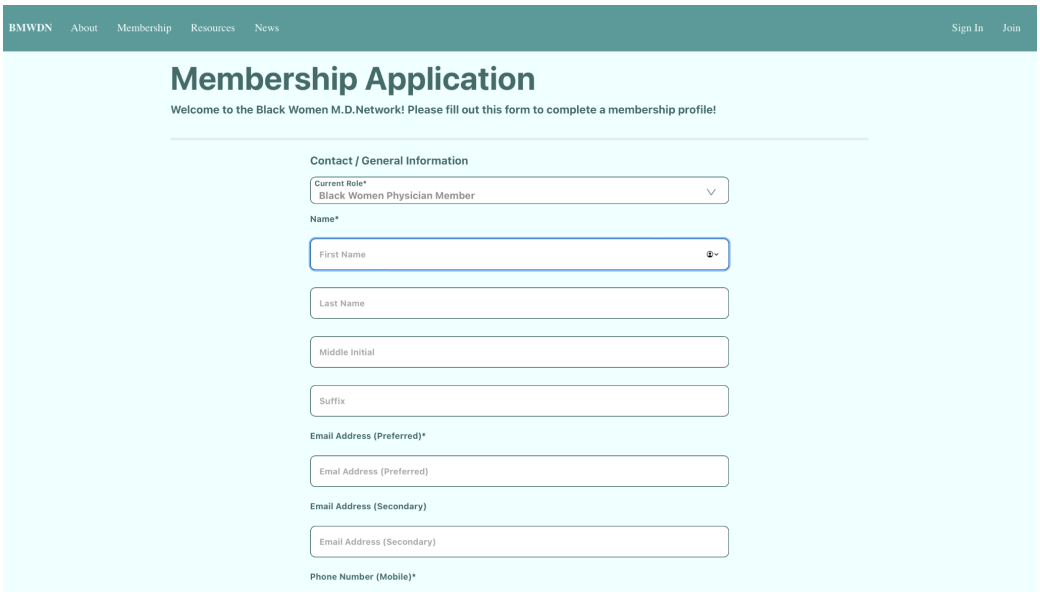
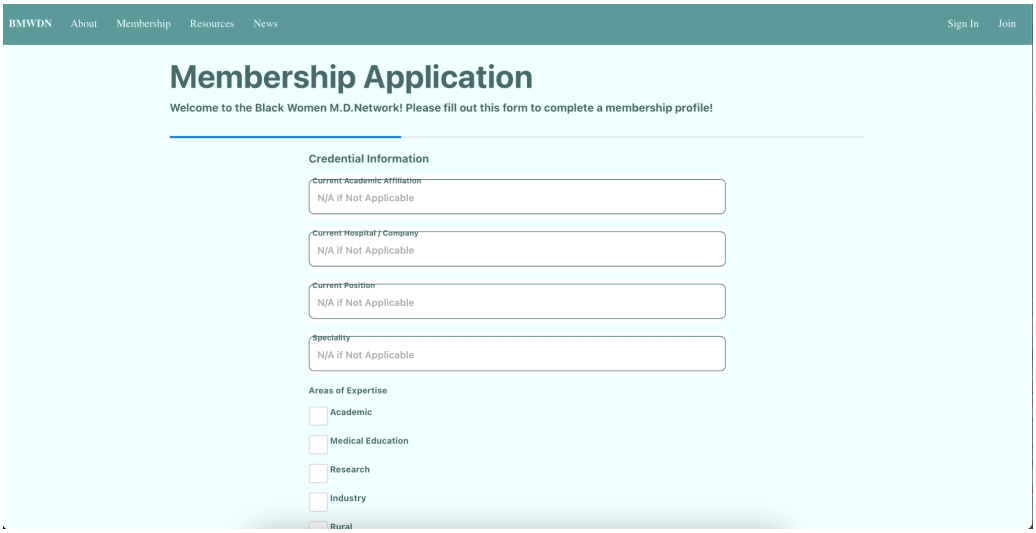


Figure 3: Homepage + Links to pages

The membership application's contact and general information section is a form that collects important personal information from users. This information includes their name, email address, phone number, and mailing address. Users are also asked to provide information about their race, ethnicity, and gender identity, as well as their preferred pronouns. The following sections of the membership form require the user to enter their work and education history, followed by questions regarding joining the network as a tiered user. This section is designed to ensure that the platform is inclusive and welcoming to all users. The form fields are validated and stored in a database, and the data is checked to make sure it is in the correct format.

Figure 4: Membership Application Form





The Membership Directory displays all users who have been approved to join the Black Women M.D. Network. It enables other users on the network to find and connect with each other. Users can find other members by using a search bar, checking specialties, or toggling whether a resume is available or the member is willing to sponsor. From there, they can contact the user by pressing on "Contact," triggering a pop-up that will show that member's phone number and email. In addition, if a user allows their resume to be public to all Black Women M.D. Network members, the "**Resume"** button will display it.

Figure 5: Membership Directory

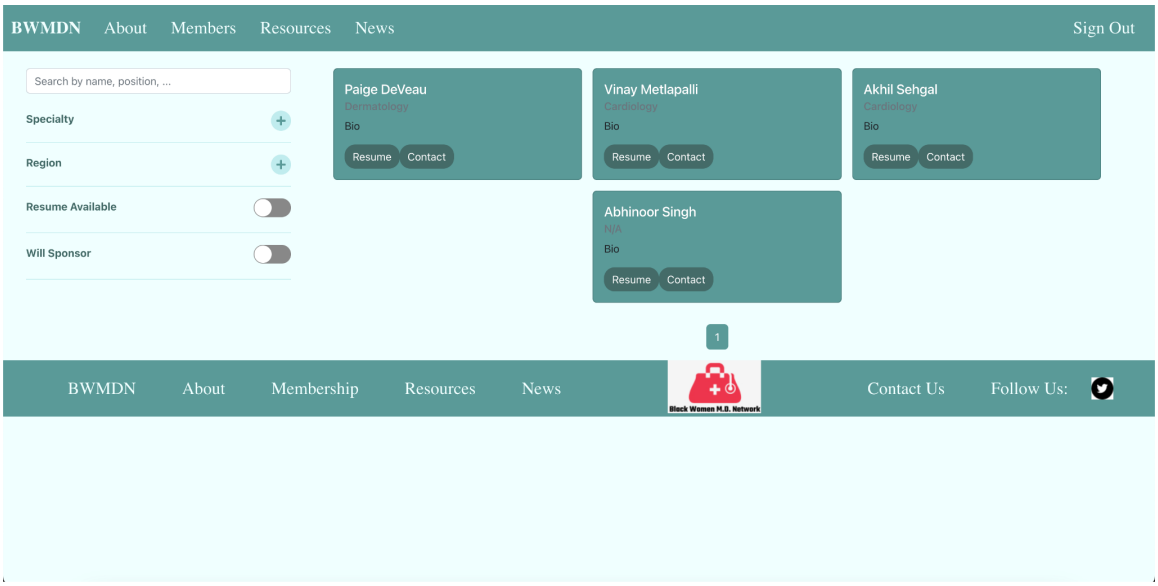
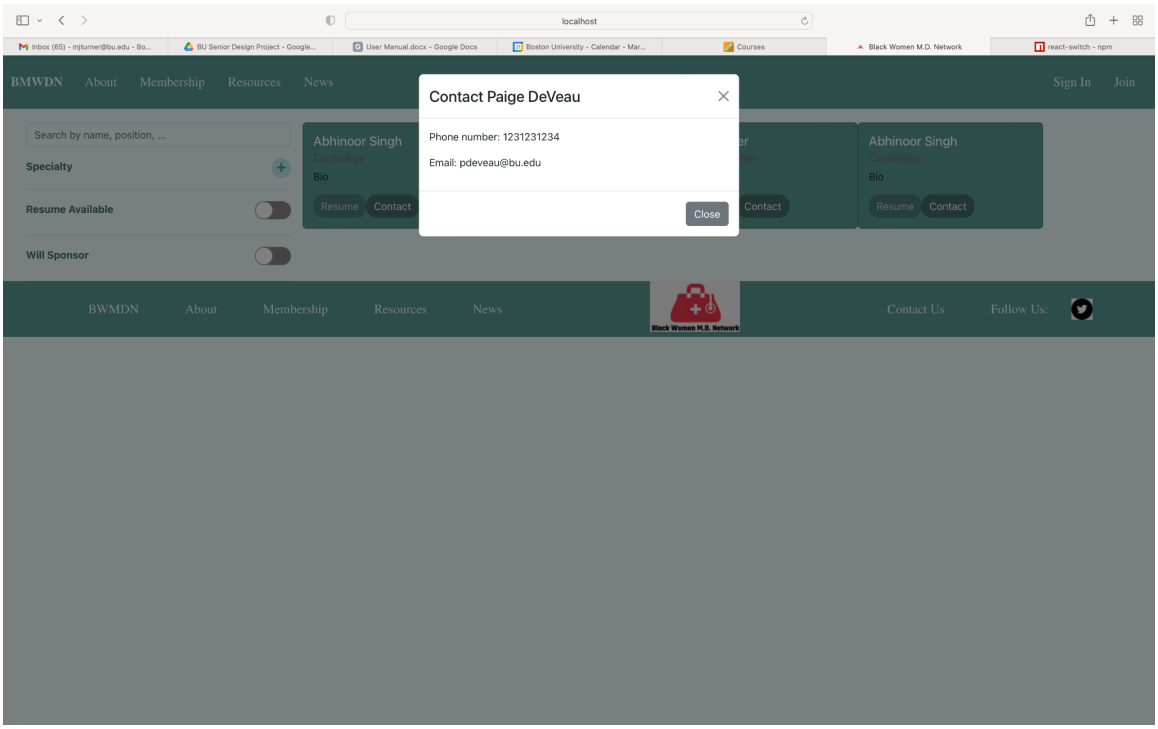


Figure 6: Member Contact Info



The admin portal allows the administrator to review applicants and determine their eligibility to join the membership network. The admin had the ability to view applicants who have recently applied, approved memberships, and rejected members.

Figure 7: Admin portal to manage applicants

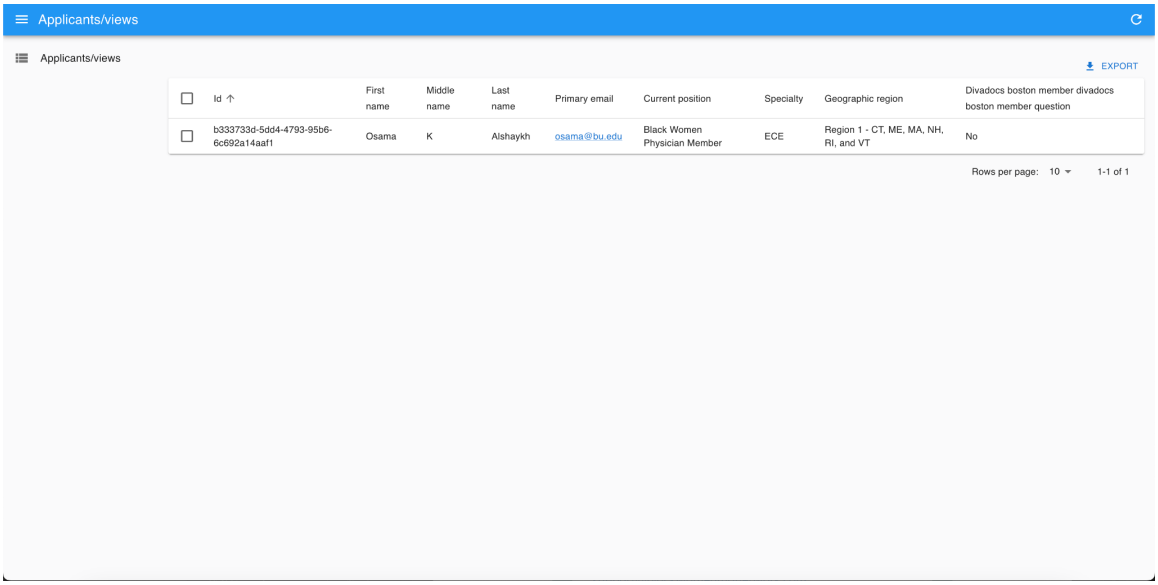


Figure 8: Login Page

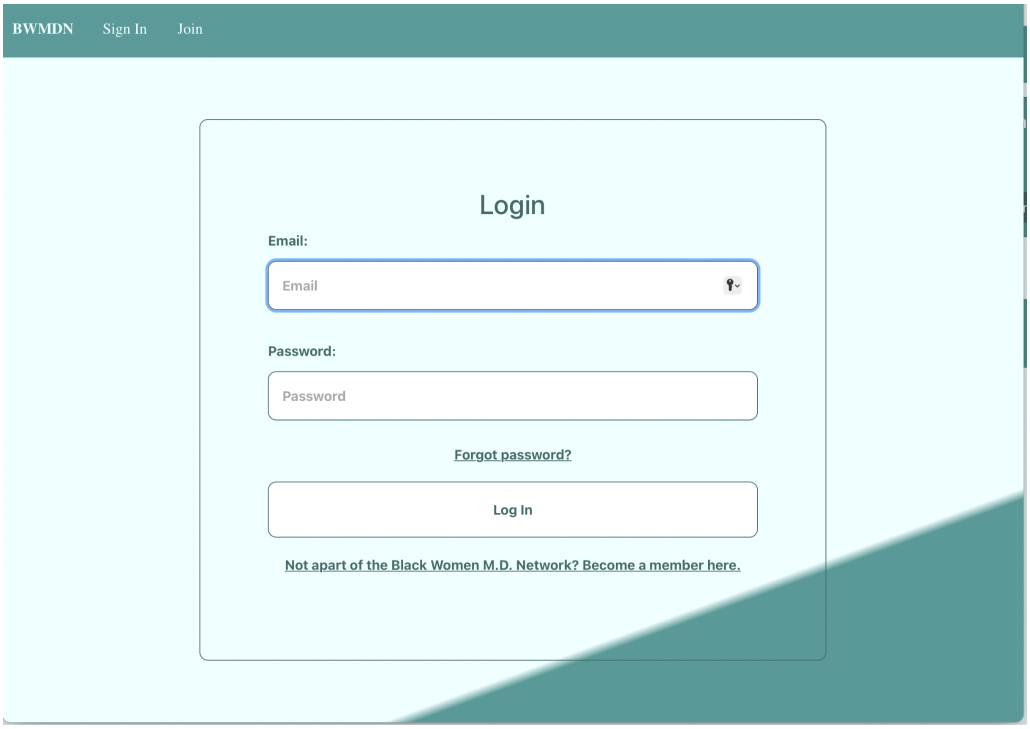
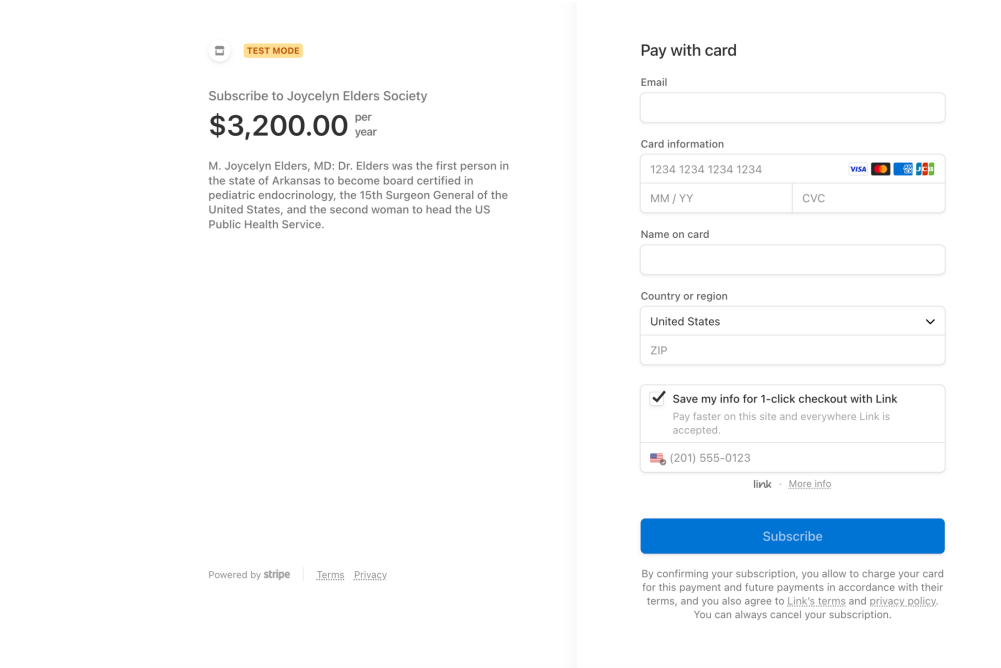


Figure 9: User receives a link to the email they provided to submit payment via Stripe API



2.3 Installation, Setup, and Support

The Black Women M.D. Network website has been designed to be compatible with all devices, including desktop computers, laptops, tablets, and smartphones. This ensures that users can access the platform using their preferred device, regardless of the screen size or device type. To access the website, users simply need to navigate to blackwomenmdnetwork.com using their preferred web browser. Once on the website, the user can create an account, browse resources, and connect with other members using their device of choice. The website's compatibility with all devices ensures that users have a seamless experience and can access the platform anytime, anywhere.

To gain access to the Black Women M.D. Network website, users are required to create an account and have it approved by the platform's administrators. Once their account has been approved, users must pay the subscription fee to gain full access to the website's features.

The subscription fee for the Black Women M.D. Network website is broken down into tiers based on the applicant's work history. Users can choose to pay the subscription fee in four quarterly installments or annually. This flexible payment system allows users to choose the payment plan that best suits their financial situation. The subscription fee tiers are designed to ensure that the platform is accessible to all Black women physicians, regardless of their level of experience or income. By providing a range of payment options and tiers, the platform ensures that everyone has the opportunity to benefit from the resources and community provided by the Black Women M.D. Network.

Once the user gains access to the site, they have the ability to navigate through various pages and are presented with content based on their subscription tier. The user has access to a membership directory where they can search and connect with others on the platform. They are presented with contact information and ways to connect with other members. The user can filter results by location, medical specialty, or by searching.

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

Welcome Page

- ☐ Type www.blackwomenmdnetwork.com into your preferred browser
- ☐ Use the navigation bar to navigate to various subpages such as **About**, **Membership**, **Resources**, and **News**
- ☐ Use **Sign-In** or **Join** to become a member/ login

Membership Application

- ☐ Click **Join** to begin membership application process
- ☐ Answer questions about general contact information, work experience, resume, and interest in joining the network
- ☐ Select membership tier to submit application
- ☐ Select membership type that best fits your needs

Payment & Account Creation

- ☐ User will receive an email indicating their application has been approved or denied
- ☐ Approved Users:
  - ☐ Email with payment instructions will be sent to primary email address
  - ☐ Once the payment has been submitted using Stripe API, user will receive an email confirmation with details about login information.
- ☐ The username will be the email address that entered during the registration process. The password will be generated and sent to the user.
- ☐ Forgot Password
  - ☐ Users may click on “Forgot Password” and be sent an email with an updated password that they can use to log into their account.

Membership Directory Search

- ☐ Navigate to the **Member Search** page
- ☐ Use the search bar to locate specific members
- ☐ On the left side, there is a filtering option to narrow down search results by medical profession, location, resume, or those members looking to connect
- ☐ Click on a member’s tile to reveal their contact information and resume

Admin Portal

- ☐ Login: The admin logs into the backend applicant portal using their unique username and password.

- ☐ Navigate to the application list: Once logged in, the admin navigates to the list of applications that have been submitted by applicants.
- ☐ Review the application: The admin selects an application from the list and reviews the application in detail.
- ☐ Make a decision: Based on their review, the admin makes a decision whether to approve or deny the application.
- ☐ Update the application status: Once a decision is made, the admin updates the membership application status to reflect whether it has been approved or denied.
- ☐ Notify the applicant: Once approved, the applicant is automatically sent a link for payment

3.2 Operating Mode 2: Abnormal Operations

Membership Application Questionnaire

The application questionnaire has requirements for which questions get answered and those that are optional. The user must complete the minimum questions as indicated with an asterisk above each question before preceding to the next screen.

Payment and Account Verification

Our payment functionality is handled primarily by Stripe, a third-party payment facilitation provider. When a user is approved by the website administrator, an email issent to the approved user with a unique link to a Stripe payment portal which will ask them to pay the appropriate amount based on which membership tier the user applied to. In the backend, a Stripe webhook waits for a "Payment Completed Event" from the unique link that was sent out, and once the webhook receives this event, the user's document in the Mongo database gets updated to say that the payment was successful. Once this process is complete, the user's account is verified and they receive another email automatically confirming their payment and with their login information to access the website.

3.3 Safety Issues

Several security measures have been implemented to protect user data and ensure secure access to the platform. These measures include:

1) CORS middleware is used to restrict which domains can access the API, preventing cross-site scripting (XSS) attacks.
2) Passwords are securely hashed using the bcrypt algorithm, making them much more difficult to crack if the database is compromised.

3) JSON Web Tokens (JWTs) are used to authenticate users and control access to protected endpoints. Tokens are signed with a secret key, and contain an expiration time to prevent them from being used indefinitely.
4) The OAuth2.0 class is used to authenticate users and retrieve their access tokens from requests. This helps ensure that only authorized users can access protected endpoints.
5) Stripe webhook events are verified using a webhook secret, preventing unauthorized access to the backend.
6) When applicants are approved and paid, a random password is generated for them and stored in the database in a hashed format. This ensures that even if the database is compromised, the passwords are not immediately usable.
7) When an applicant forgets their password, a new random password is generated for them and sent to their email address. The new password is then hashed and stored in the database.

4 Technical Background

4.1 MongoDB

The following is a brief discussion of the technical approaches for the various database operations implemented to handle the functionality of the website. Each description focuses on the technical principles of the implementation and the underlying functionality that drives the overall operations of the database.

write_to_mongo(document, database, collection): This function writes a given document to a specified collection in a specified database in MongoDB. It establishes a connection to the MongoDB server using the provided URI, selects the specified database and collection, and then inserts the given document into the collection.

read_from_mongo(database, collection): This function reads all the documents from a specified collection in a specified database in MongoDB. It establishes a connection to the MongoDB server using the provided URI, selects the specified database and collection, and then retrieves all the documents from the collection using the find() method. The function returns a list of dictionaries, with each dictionary representing a document from the collection.

upload_file_to_mongo(database, collection, file, file_name): This function uploads a file to GridFS in a specified database in MongoDB. It establishes a connection to the MongoDB server using the provided URI, selects the specified database and collection, and then creates a new GridFS object using the selected database. It then uploads the given file to GridFS using the put() method, with the specified file name as the second parameter.

download_file_from_mongo(database, file_name): This function downloads a file from GridFS in a specified database in MongoDB. It establishes a connection to the MongoDB server using the provided URI, selects the specified database, and then creates a new GridFS object using the selected database. It then retrieves the metadata of the specified file from the fs.files collection using the find_one() method and gets the data of the file using the get() method of the GridFS object. The function returns the data of the specified file.

send_email(recipient_email): This function sends an email to a specified recipient email address using Gmail SMTP server. It first sets up the email message with a randomly generated passcode and then establishes a connection to the SMTP server using the smtplib.SMTP() method. It logs in to the SMTP server using the provided sender email address and password and then sends the email using the sendmail() method.

send_payment(u_id): This function moves a document from a source collection to a target collection in a specified MongoDB database, and then sends a payment link to the email address of the applicant whose document was moved. It first establishes a

connection to the MongoDB server using the provided URI, selects the specified source and target collections, and then retrieves the document with the specified ID from the source collection using the find_one() method. It then moves the retrieved document to the target collection using the insert_one() method of the target collection and the delete_one() method of the source collection. Finally, it sends a payment link to the email address of the applicant using the Stripe API.

get_all_approved(): This function retrieves all approved application forms from the MongoDB database collection called ApprovedApplications. The function establishes a connection with the database, fetches all documents from the ApprovedApplications collection, and returns the list of documents.

get_password(email): This function takes an email as an argument and retrieves the hashed account password for the user with the corresponding email address from the ApprovedApplications collection in the MongoDB database. The function connects to the database, searches for the document with the provided email, and returns the applicant's document containing the account password.

applicant_denied(u_id): This function moves the applicant from the SubmittedApplications collection to the DeniedApplications collection in the MongoDB database. The function takes an u_id argument, which is the unique ID of the applicant's document in the SubmittedApplications collection. The function connects to the database, searches for the document with the provided u_id, inserts the document into the DeniedApplications collection, and deletes the document from the SubmittedApplications collection. Then the function sends an email to the applicant notifying them that their application has been denied.

pull_approved_applicants(): This function retrieves all the approved applicants' documents from the ApprovedApplications collection in the MongoDB database, where the applicant_status.paid field is set to True. The function connects to the database, searches for the documents with the specified condition, and returns the list of documents.

send_login_email(uid, input_password): This function sends an email to the applicant with their login information after their application has been approved. The function takes two arguments: uid, which is the unique ID of the applicant's document in the ApprovedApplications collection, and input_password, which is the password given to the applicant during the application process. The function connects to the database, retrieves the applicant's email address from their document, composes an email message containing the login information, and sends it to the applicant's email address.

send_forgotPassword_email(username, input_password, hashed_password): This function sends an email to the applicant with their new password when they request a password reset. The function takes three arguments: username, which is the email address

of the applicant, input_password, which is the new password they chose during the reset process, and hashed_password, which is the hashed version of the password. The function connects to the database, retrieves the applicant's document by searching for the email address in the ApprovedApplications collection, updates the document with the new hashed password, composes an email message containing the login information, and sends it to the applicant's email address.

4.2 Backend

The general architecture of the backend is built using the FastAPI framework, which is a high-performance web framework for building APIs with Python. FastAPI makes it easy to define API endpoints and models and automatically generates documentation and input validation based on Python-type hints.

The backend is divided into several endpoints, each of which serves a different purpose. Here's a brief overview of each endpoint and its functionality:

/applicants/add: This endpoint allows users to submit an application by sending a POST request with the relevant data. The data is then stored in a MongoDB database using the mongo_test.write_to_mongo() function.

/applicants/view: This endpoint allows users to view all submitted applications. It retrieves the applications from the MongoDB database using the mongo_test.read_from_mongo() function, and returns them as a JSON response.

/applicants/view/{id}: This endpoint allows users to view a specific application by ID. It retrieves the application from the MongoDB database using the mongo_test.read_from_mongo() function, and returns it as a JSON response.

/approvedapplicants/view: This endpoint allows users to view all approved applicants. It retrieves the approved applicants from the MongoDB database using the mongo_test.get_all_approved() function, and returns them as a JSON response.

/applicants/resume/upload: This endpoint allows users to upload a PDF file containing their resume. The file is stored in the MongoDB database using the mongo_test.upload_file_to_mongo() function.

/applicants/downloadresume/{name_file}: This endpoint allows users to download their resume by providing the filename of the PDF file stored in the MongoDB database. The file is retrieved using the mongo_test.download_file_from_mongo() function, and returned as a response with the appropriate media type.

/applicants/approveapplicant: This endpoint is used to approve an applicant and request payment. It retrieves the applicant data from the request body, and sends a payment request to Stripe using the mongo_test.send_payment() function.

/webhook: This endpoint is used to handle Stripe webhook events. It verifies the event using the webhook secret, and if the event is a checkout session completion event, it retrieves the applicant data and updates their status in the MongoDB database to indicate that they have paid and been approved.

/login: This endpoint is used to authenticate users and generate access tokens. It retrieves the username and password from the request body and uses the authenticate_user() function to verify the credentials. If the credentials are valid, it generates a JWT access token using the create_access_token() function and returns it in the response.

/protected_endpoint: This endpoint is a protected endpoint that requires a valid JWT access token to access. It retrieves the token from the request using the OAuth2PasswordBearer class, decodes it using the decode_token() function, and returns a success message if the token is valid.

/forgot_password: This endpoint is used to generate a new random password for an applicant who has forgotten their password. It retrieves the applicant username from the request body, generates a new password using the generate_random_password() function, hashes the password using the generate_password() function, and sends an email to the applicant with their new password using the mongo_test.send_forgotPassword_email() function.

In addition to these endpoints, the backend also includes several helper functions that are used throughout the codebase:

generate_password(): This function is used to securely hash passwords using the bcrypt algorithm.

generate_random_password(): Generates a random password for the user.

verify_password(): This function is used to verify that a given password matches a hashed password stored in the database.

create_access_token(): This function is used to generate a JWT access token with an expiration time.

authenticate_user(): This function is used to verify a user's credentials and retrieve their data from the database.

decode_token(): This function is used to decode a JWT access token and retrieve the user data from it.

send_payment(): This function is used to send a payment request to Stripe and update the applicant's status in the database to indicate that they have paid and been approved.

send_login_email(): This function is used to send an email to the applicant with their login credentials.

send_forgotPassword_email(): This function is used to send an email to the applicant with their new password.

Overall, the backend provides a secure and robust API for submitting, viewing, and managing applicant data. It uses a variety of security measures, such as password hashing, JWT authentication, and CORS restrictions, to protect user data and prevent unauthorized access to the API. The code is well-organized and easy to read, and the use of the FastAPI framework makes it easy to add new endpoints and functionality as needed.

4.3 Frontend

The Frontend of the Black Women M.D. Network site takes advantage of modern libraries and frameworks such as React Typescript, Bootstrap and many others. This website utilizes React Routes in order to navigate to the various pages that we have in our application. There is a navigation bar that allows users to travel to all pages from any one page. There is also a footer that displays pages as well. The application is authenticated by AuthProvider so that only users who are logged in are able to access the exclusive pages such as the Membership Directory.

5 Relevant Engineering Standards

Being a software development team, it's essential to adhere to industry standards when developing a website application. By following these standards, we can ensure that your product meets the requirements of stakeholders, such as clients, users, and developers, while also ensuring the reliability, safety, and functionality of your product. In this project, we utilized several IEEE, ISO, and HTTPS standards to ensure that our website application meets industry requirements.

1. IEEE 830 - Software Requirements Specification: This standard provided us with guidelines for documenting the functional and non-functional requirements of our website application. By following this standard, we could ensure that our website met the requirements of our stakeholders, such as clients and users, and also ensure that our developers had a clear understanding of what the website was expected to do.
2. IEEE 1063 - Software User Documentation: This standard provided us with guidelines for developing user documentation for our website application. By following this standard, we could ensure that our website's user documentation was clear, concise, and effective in helping users understand how to use our product.
3. IEEE 1012 - Software Verification and Validation: This standard provided us with guidelines for the verification and validation of our website application throughout the software development life cycle. By following this standard, we could ensure that our website was free of defects and met its requirements.
4. IEEE 1471 - Architecture Description: This standard provided us with guidelines for documenting the architecture of our website application. By following this standard, we could ensure that our website's architecture was well-designed and meets the project's requirements.
5. HTTP and HTTPS protocols: These protocols define how web browsers and servers communicate with each other over the internet. By following these protocols, we can ensure that our website is compatible with web browsers and can communicate securely with servers.
6. ISO/IEC 12207 - Software Lifecycle Processes: This standard defines the processes, activities, and tasks required for software development, maintenance, and retirement. By following this standard, we can ensure that our website development project is well-organized and follows best practices.
7. ISO/IEC 27001 - Information Security Management: This standard provides guidelines for establishing, implementing, maintaining, and improving information security management systems. By following this standard, we can ensure that your website is secure and protects the confidentiality, integrity, and availability of information.

8. OAuth 2.0 - provides a framework for securing REST APIs by allowing third-party applications to access a user's data on their behalf without revealing the user's credentials.
9. ECMA-404 - defines the syntax and data types used in JSON, providing a common understanding of how data should be represented in JSON format.
10. REST architectural style constraints - including a client-server architecture, statelessness, and a uniform interface. These constraints are described in Roy Fielding's dissertation on "Architectural Styles and the Design of Network-based Software Architectures" and provide guidance on how to design RESTful web services.

6 Cost Breakdown

Project Costs for Production of Beta Version (Next Unit after Prototype)			
Item	Quantity	Description	Unit Cost
1 Domain	1	GoDaddy custom URL	$35/biannually
2 MongoDB Server	1	Shared MongoDB Server Rack (up to 100,000 users)	$57/month
3 Stripe Fee	1	Stripe API Payment Process Fee	2.99% + $0.30/ transaction
4 MailGun	1	Automated Email System	$35/month
5 Python3 Fast API	1	Python Suite of APIs	$0
6 Jira Software	1	Jira Product Management software to track and manage development using sprints	$76/month
			Beta Version-Total Cost (1000 users): ~ $851

For the custom URL, the customer has chosen GoDaddy, which offers a domain name registration service for a fee. The team has estimated the cost to be $35/biannually, which is a reasonable price for a custom URL. For the database server, the project team has decided to use a shared MongoDB server rack that can support up to 100,000 users. The estimated cost for this service is $57/month, which is reasonable considering the server capacity and the scalability of the platform. To process payments, the team has chosen to use Stripe API, which charges a fee of 2.99% + $0.30 per transaction. This is a standard rate for online payment processing, and the team has factored it into their cost breakdown. For automated email communication, the team has chosen MailGun, which offers a range of email automation services. The estimated cost for this service is $35/month, which is reasonable for the platform's needs. The team has decided to use Python3 Fast API, an open-source Python suite of APIs, for the platform's backend development. This is a cost-effective solution as there is no cost associated with using the Python3 Fast API.

To produce the beta version of the platform, the project team needs to invest in several items and services, such as a custom URL for the platform, a shared MongoDB server rack, a payment processing system using Stripe API, an automated email system, and a product management software such as Jira to track and manage development using sprints.

The estimated cost of the beta version assumes 1000 active users and averages the subscription price per month. It is important to note that this cost breakdown is based on a high-level aggregation of investment, and more detailed breakdowns may be required for accurate cost analysis. Additionally, other costs such as labor and production deployment expenses may need to be factored in for a comprehensive cost analysis of the beta version.

7 Appendices

7.1 Appendix A - Specifications

Requirement	Metric
Compatibility	Website must be compatible with major web browsers (Chrome, Firefox, Safari, Edge) and mobile devices.
Pages must load efficiently	Website pages must load in <5 seconds Backend API's will have a response time of < 2 seconds
Usability	Site must be easy to navigate, with clear and concise labeling, intuitive design, and limited clicks to reach desired content.
Filling out the application form should be simple	Form must be designed with clear instructions and simple, easy-to-understand fields.
Security	Site must use HTTPS protocol, secure authentication, and have regular backups in place.
Performance	Site must have 99% uptime with minimal downtime for maintenance or updates.
Scalability	Site must be able to handle a growing number of users without sacrificing performance or functionality.
Analytics	Site must have analytics in place to track user behavior, traffic, and engagement.

7.2 Appendix B - Membership Tiers

1. **Rebecca Crumpler Society**: >20 years post residency or by invitation): In-Kind Contribution

   Rebecca Crumpler, MD: Dr. Crumpler was the first Black woman in the US to earn an MD. She became a doctor in 1864 and published a book of medical advice for women and children in 1883.

2. **Joycelyn Elders Society**: 10-20 years post residency ($3200/yr)

   M. Joycelyn Elders, MD: Dr. Elders was the first person in the state of Arkansas to become board certified in pediatric endocrinology, the 15th Surgeon General of the United States, and the second woman to head the US Public Health Service.

3. **Barbara Ross Society**: <10 years post residency ($1600/yr)

   Barbara Ross-Lee, DO: Dr. Ross-Lee was the first Black woman to be appointed dean of a US medical school, Diana Ross's sister

4. **Alexa Canaday Society**: Residents and Fellows ($800/yr)

   Alexa Irene Canady, MD: In 1981, Dr. Canady became the first Black woman to become a neurosurgeon in the United States.

5. **Mae Jemison Society**: medical students ($400/yr)

   Mae C. Jemison, MD: Dr. Jemison is a physician, scientist, teacher, chemical engineer, and astronaut.

6. **Nancy Oriol Society**: premedical students ($200/yr)

   Nancy E. Oriol, MD: Dr. Nancy Oriol is a Harvard anesthesiologist, inventor, community advocate, and the co-founder of MedScience program for high school students.

7.3 Appendix C – Team Information

The Black Women M.D. Network is a ground-breaking platform designed to empower Black women doctors, and our group recognizes the immense value and impact that this network has in fostering an inclusive, supportive, and inspiring community of professionals. We believe that Black women physicians deserve a dedicated and specialized space to connect, collaborate, and grow professionally, and we are excited to contribute our technical expertise to help bring this vision to life.

As a group of passionate and driven professionals, we are deeply committed to creating positive change in our communities, and we are inspired by the mission and vision of the Black Women MD Network. Our team is excited to leverage our technical skills and experience to develop a robust, user-friendly platform that enables Black women physicians to network, discover new opportunities, and advance their careers. We are honored to be a part of this important initiative and look forward to contributing to the growth and success of the Black Women MD Network.

Paige DeVeau – Paige is graduating as a Computer Engineer. She will be joining Dell Technologies in Hopkinton, MA as Software Engineer I upon graduation.

Mya Turner – Mya Turner is a Senior studying Computer Engineering. She is excited to be graduating this May and begin her post-graduate career as a Software Engineer at Netflix. Besides coding, Mya loves to dance and be a mentor to other underrepresented minorities like her!

Akhil Sehgal – Akhil is graduating as a Computer Engineer. He is passionate about building products and flying Cessna 172s in his free time. He will be joining Cisco Systems in San Jose, CA as an Engineering Product Manager upon graduation.

Vinay Metlapalli – Vinay is graduating as a Computer Engineer. He is passionate about working with clients to develop software solutions for them. He will be joining Beacon Platform, Inc. in New York City, NY as a Sales Engineer upon graduation.

Abhinoor Singh – Abhinoor is graduating as a Computer Engineer. He is passionate about writing software and working with computer hardware. He will be joining Amazon in Boston, MA as a Software Development Engineer upon graduation.