

FUTURESCOPE TECH



Design Seminar II

Members

Ibrahim Ammar

Josh Weber

Abdalla Ibrahim

Shakia Shahid

Faculty

Bruce Maxim

Mahmoud Abou-Nasr

I. TABLE OF CONTENTS

I.	Table of Contents	2
II.	Acknowledgements	3
III.	Introduction/Overview	4
	1.1 Goals and Objectives.....	4
	1.2 Statement of Scope.....	4
	1.3 Software Context.....	4
	1.3 Major Constraints.....	4
IV.	Requirements/Analysis Model	5
	2.1 User Profiles.....	5
	2.2 User Stories.....	5-6
	2.3 Data Objects.....	6
	2.4 Data Dictionary.....	7
	2.5 Functional Model and Description.....	8-11
	2.6 Behavior Model and Description.....	12
V.	Hardware/Software Design	13
	3.1 Data Design.....	13-14
	3.2 Architecture Design.....	15
	3.3 Component Design.....	16
	3.3 Software Interface Description.....	17
	3.3 User Interface Design.....	17-20
VI.	Implementation Details	20-21
VII.	Testing/SQA	22
	5.1 Reviews and Audits.....	22-29
	5.2 Quality Tracking.....	30-33
	5.3 Testing Strategy.....	34
	5.3.1 Unit Testing.....	34
	5.3.2 Integration Testing.....	34
	5.3.3 Validation Testing.....	34
	5.3.4 High-Order Testing.....	35
	5.4 Testing Procedures.....	35-37
	5.4.1 Test Case Log.....	37
	5.4.2 Recorded Results.....	38
VIII.	Future Maintenance Suggestions	39
IX.	Delivery/Acceptance Statement	40
X.	References & Bibliography	41
XI.	Appendix A - User Manual	42-43
XII.	Appendix B - Sample Output	44-47
XIII.	Appendix C - Team Member Resumes	48-51
XIV.	Appendix D - Project Plan & Logbook	52-56
XV.	Appendix E - Project Demo Notes	57
XVI.	Appendix F - Final Presentation Slides	58

Acknowledgments

I would like to express my deepest appreciation to Professor Kattan, Professor Bruce Maxim, and Professor Mahmoud Abou Nasr for their expert guidance and unwavering support throughout the CIS 4951 – Design Seminar I and our CIS 4951 – Design Seminar II course. Their dedication to fostering a learning environment has been instrumental in our academic and professional development.

I am also thankful for the hard work and dedication of my team members, and I am proud of our collective achievements in the coding and development of our project website:

- ❖ **Ibrahim Ammar:** Focused on the integration of external APIs and enhancing the security features of our website, ensuring robust and secure user transactions.
- ❖ **Joshua Weber:** Instrumental in developing the back-end infrastructure, focusing on integrating the payment and booking systems which are crucial for the website's functionality.
- ❖ **Abdalla Ibrahim:** Took the lead on the front-end development, ensuring that the website's user interface is both functional and aesthetically pleasing, aligning with the latest design trends.
- ❖ **Shakia Shahid:** Contributed to both front-end and back-end development, specializing in creating responsive design elements that enhanced user experience and accessibility.

Each of us brought unique skills and perspectives to the project, and it has been a tremendous collaborative effort.

Lastly, I extend my gratitude to my family and friends for their constant encouragement and support throughout this challenging yet rewarding project.

1.0 Introduction

This section offers a comprehensive view of the requirement document, outlining the data, functional, and behavioral necessities for the software. The document serves as a roadmap for the development process, ensuring clarity and alignment with project goals.

1.1 Goals and Objectives

The primary goals and objectives of the software project are to revolutionize the online presence of local businesses through FutureScopeTech. This includes providing valuable web development and marketing services. The specific objectives encompass:

- ❖ Achieving a modern website redesign aligned with the latest design trends.
- ❖ Implementing payment integration for seamless and secure transactions.
- ❖ Developing a booking system to streamline communication and collaboration through scheduled appointments.

1.2 Statement of Scope

This section provides a detailed description of the software, highlighting major inputs, processing functionality, and outputs without delving into implementation specifics. The scope includes features such as:

- ❖ Appointment scheduling for clients visiting the website.
- ❖ Information collection, including name, phone, email, and a short project description/link.
- ❖ Possible integration with a scheduling tool, such as <https://calendly.com/>.
- ❖ Diverse payment options, including the simplest method, Stripe, invoicing, one-time payments, subscription models, and project-based custom invoicing.
- ❖ Administration page functionality, ensuring that all content is configurable through a CMS/Admin system.

1.3 Software Context

In the context of our project for FutureScopeTech's website enhancement, our software plays a pivotal role in elevating the online presence of local businesses. Tailored for collaborative software teams, it acts as a catalyst for achieving the project's goals. The software is designed to modernize the website, enable user account creation, integrate seamless payment options, and implement a booking system. It aims to revolutionize the way local businesses engage with web development and marketing services, fostering a more dynamic and interactive online experience.

1.4 Major Constraints

Throughout the year, we have successfully addressed and overcome the initial major constraints for the FutureScopeTech website project. These included ensuring seamless functionality across various web browsers and platforms, managing multiple client services simultaneously, and accommodating team members' varying levels of JavaScript proficiency to ensure effective development. We have resolved the challenge of data permanence by implementing robust alternatives for data storage beyond the browser, ensuring universal access. Integrating diverse payment options such as Stripe and invoicing, along with external scheduling tools like Calendly, has been streamlined to add functionality without complexity. Additionally, we have enhanced our CMS/Admin system to ensure that updates are reflected in real-time and that content remains fully configurable, achieving a dynamic and user-responsive website.

2.0 Requirements/Analysis Model

2.1 User Profiles

The profiles of all user categories are described here

- ❖ **Full Control (Administrator)**
 - Role: Admin Page
 - Responsibilities: Configuring all website content via CMS/Admin system
- ❖ **Read Only (Client/Customer)**
 - Role: Website Visitors
 - Responsibilities: Browsing information pages, accessing portfolio, learning about services.

2.2 User Stories

All the user stories defining the use cases for the software are presented using the user's own words.

User Story #01	User can visit home page
Acceptance Criteria	<ol style="list-style-type: none">1. User will see general information about future scope2. User will be provided with options to navigate to other pages
User Story #02	User can visit pricing page
Acceptance Criteria	<ol style="list-style-type: none">1. User can visit pricing page which contains information regarding all pricing packages
User Story #03	User can visit about page
Acceptance Criteria	<ol style="list-style-type: none">1. User is presented with information about the futurescope company
User Story #04	User visit the contact us page
Acceptance Criteria	<ol style="list-style-type: none">1. User is presented with contact information2. User can schedule an appointment and include details about their reasons for the appointment and other applicable information(to be determined later)
User Story #05	User can view the portfolio page
Acceptance Criteria	<ol style="list-style-type: none">1. User is shown examples of futurescopes work2. User is able to navigate to those sites
User Story #06	User can view terms of service
Acceptance Criteria	<ol style="list-style-type: none">1. User can visit terms of service page and see all legal information and terms of service
User Story #07	User can view privacy page

Acceptance Criteria	1. User can view the privacy policy
User Story #08	User can visit the team page
Acceptance Criteria	1. User will see relevant information about the futurescope team
User Story #09	User can navigate to any page using the top nav bar
Acceptance Criteria	1. The top nav bar will contain aesthetically appealing navigation to the various pages on the website
User Story #10	User can make payments via the payment page
Acceptance Criteria	1. A user will be able to select a payment plan and pay on the website 2. An admin can generate an invoice and send the link to the customer for custom pricing scenarios
User Story #11	Admin can make edits from admin page
Acceptance Criteria	1. Admins can configure the content of the site via an admin dashboard/cms system
User Story #12	Admin generate invoices for custom pricing
Acceptance Criteria	1. An admin can generate custom invoices for ad hoc jobs or special pricing cases
User Story #13	Admin can view invoices and metrics regarding pricing
Acceptance Criteria	1. An admin is able to view metric and data regarding payments and invoices. 2. An admin can issue refunds or credits
User Story #14	Chatbot (potential)
Acceptance Criteria	1. A user can interact with a chatbot that will provide information, schedule appointments, etc. More details to come

2.3 Data Objects

Data objects that will be managed and manipulated by the software are described here, focusing on the new features and enhancements proposed for the FutureScopeTech website.

1. Admin
2. Customer
3. Website
4. Payment (Stripe)
5. Appointments (Calendly)
6. CMS System
7. Traffic (Google Analytics)

2.4 Data Dictionary

As mentioned previously, we will be using third-party tools APIs to implement the CMS system, Payment, and Scheduling because robust free tools already deliver our requirements. The following represent some hypothetical/assumed fields that these tools will store.

Admin

Attributes:

- ❖ ID: Unique identifier for the admin.
- ❖ Name: Name of the admin.
- ❖ Email: Email address of the admin.
- ❖ Password: Encrypted password for security.
- ❖ AnalyticsAccess: Determines if the admin has access to analytics.

Description: Represents the system administrator.

Website

Attributes:

- ❖ SiteURL: The URL of the website.
- ❖ TrafficCount: Number of visitors to the website.
- ❖ Content: Content displayed on the website fetched from CMS.

Description: Represents the main website where users interact.

Customer

Attributes:

- ❖ CustomerID: Unique identifier for each customer.
- ❖ Name: Name of the customer.
- ❖ Email: Email address of the customer.

Description: Represents the end-user or visitor of the website.

CMS System

Attributes:

- ❖ ContentID: Unique identifier for each piece of content.
- ❖ Title: Title of the content.
- ❖ Body: Main content body.
- ❖ LastUpdated: Date when the content was last updated.

Description: Manages and stores the content displayed on the website.

Traffic (Google Analytics)

Attributes:

- ❖ TrafficData: Contains data about website visitors.
- ❖ PageViews: Number of page views.
- ❖ SessionDuration: Average duration of a user session.

Description: Tool to monitor and analyze website traffic.

Payments (Stripe)

Attributes:

- ❖ TransactionID: Unique identifier for each transaction.
- ❖ Amount: Amount of the transaction.
- ❖ CustomerID: ID of the customer making the payment.
- ❖ PaymentStatus: Status of the payment (e.g., completed, pending).

Description: Manages customer payments.

Appointments (Calendly)

Attributes:

- ❖ AppointmentID: Unique identifier for each appointment.
- ❖ Date: Date of the appointment.
- ❖ Time: Time of the appointment.
- ❖ CustomerID: ID of the customer who scheduled the appointment.

Description: Manages and schedules appointments.

2.5 Functional Model and Description

Use Case #1	Client can schedule an appointment
Actors	Client, Admin
Preconditions	No client has schedule the current time slot
Triggers	Client would like to schedule an appointment
Scenario Description	A client will come to the scheduling/contact us page and will fill out information and a time slot to schedule an appointment.
Post Conditions	The client will be scheduled and the appointment will be visible to the admin
Exceptions	A client has already scheduled the time. The time should in this case be unavailable

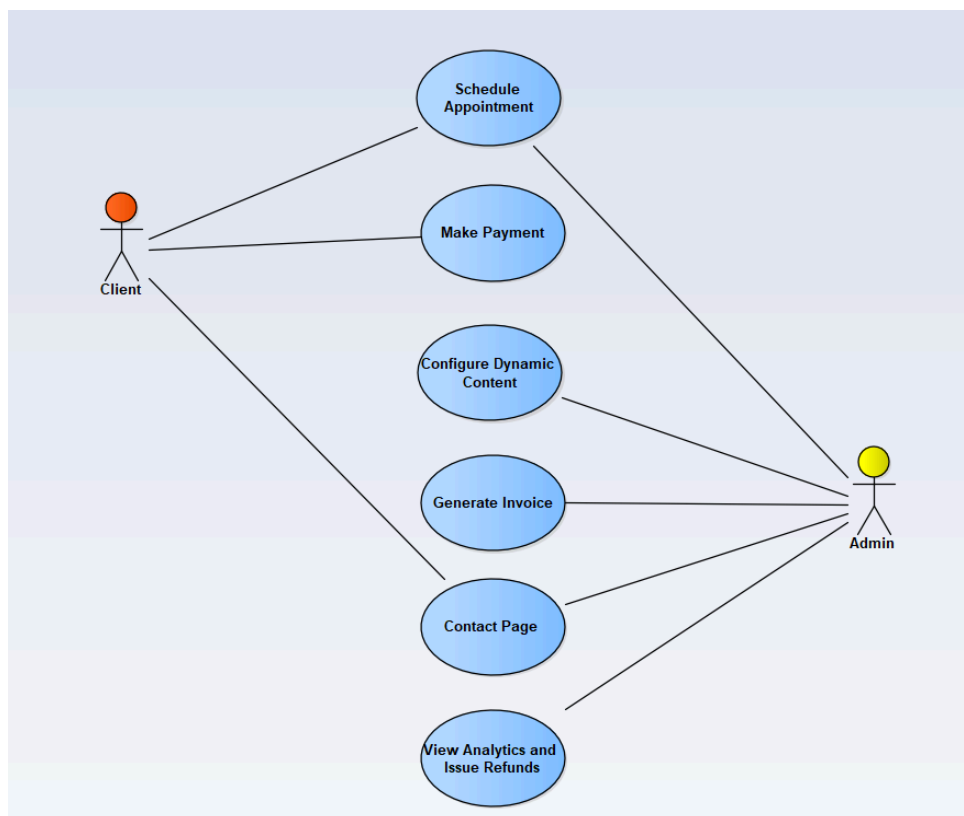
Use Case #2	Client can make a payment from pricing page
Actors	Client
Preconditions	Client has discussed what they needed
Triggers	Client wants to make a payment
Scenario Description	After client appointment and finalizing what they need done, the client will go to the payment page and select what plan they chose.
Post Conditions	After payment goes through client job will go through
Exceptions	Client has a specific price that was agreed upon with the company and will receive specific payment instructions

Use Case #3	Admin can configure dynamic content from cms system
Actors	Admin
Preconditions	Admin can access the cms system
Triggers	Admin wishes to edit content and logs into to cms system
Scenario Description	Admin will access the cms system and edit the relevant field which will be reflected on the site upon changing the content
Post Conditions	The changes will be reflected on the site with the updated content
Exceptions	Errors saving to cms system/database

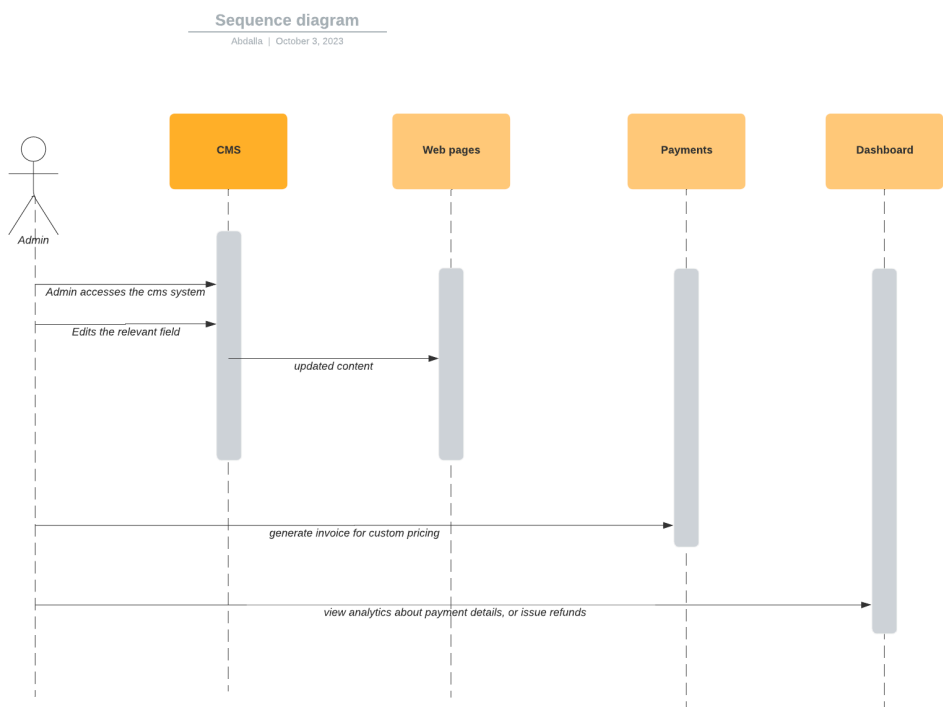
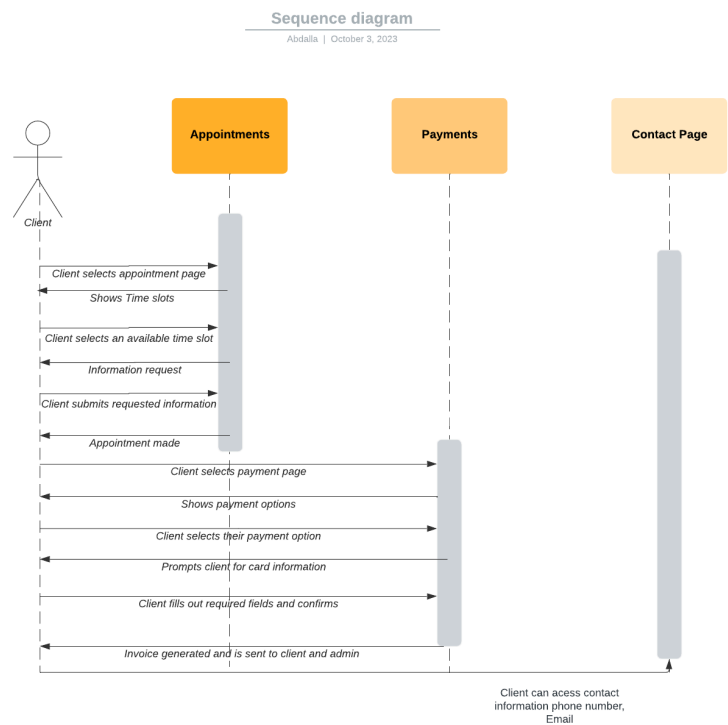
Use Case #4	Admin can generate invoice for custom pricing to send to the client
Actors	Admin, Client
Preconditions	Client and admin have discussed a custom project or pricing
Triggers	Admin wants to bill client for a custom project or pricing scenario
Scenario Description	An admin will login to the admin dashboard or third party software which connects to the website to generate an invoice to send to the client so the client can make the payment
Post Conditions	The client has paid and the admin can view the successful payment from the dashboard
Exceptions	Credit card charge fails

Use Case #5	Admin can manage payments
Actors	Admin
Preconditions	Payments have been made to the company
Triggers	An admin wishes to visit analyze data about the payment they have collected or inspect payment details
Scenario Description	An admin will login to an admin dashboard and can view analytics about the payments or issues refunds to customers if needed
Post Conditions	The admin has analyzed the data or the client has been refunded
Exceptions	The refund fails or there is an issue analyzing the data

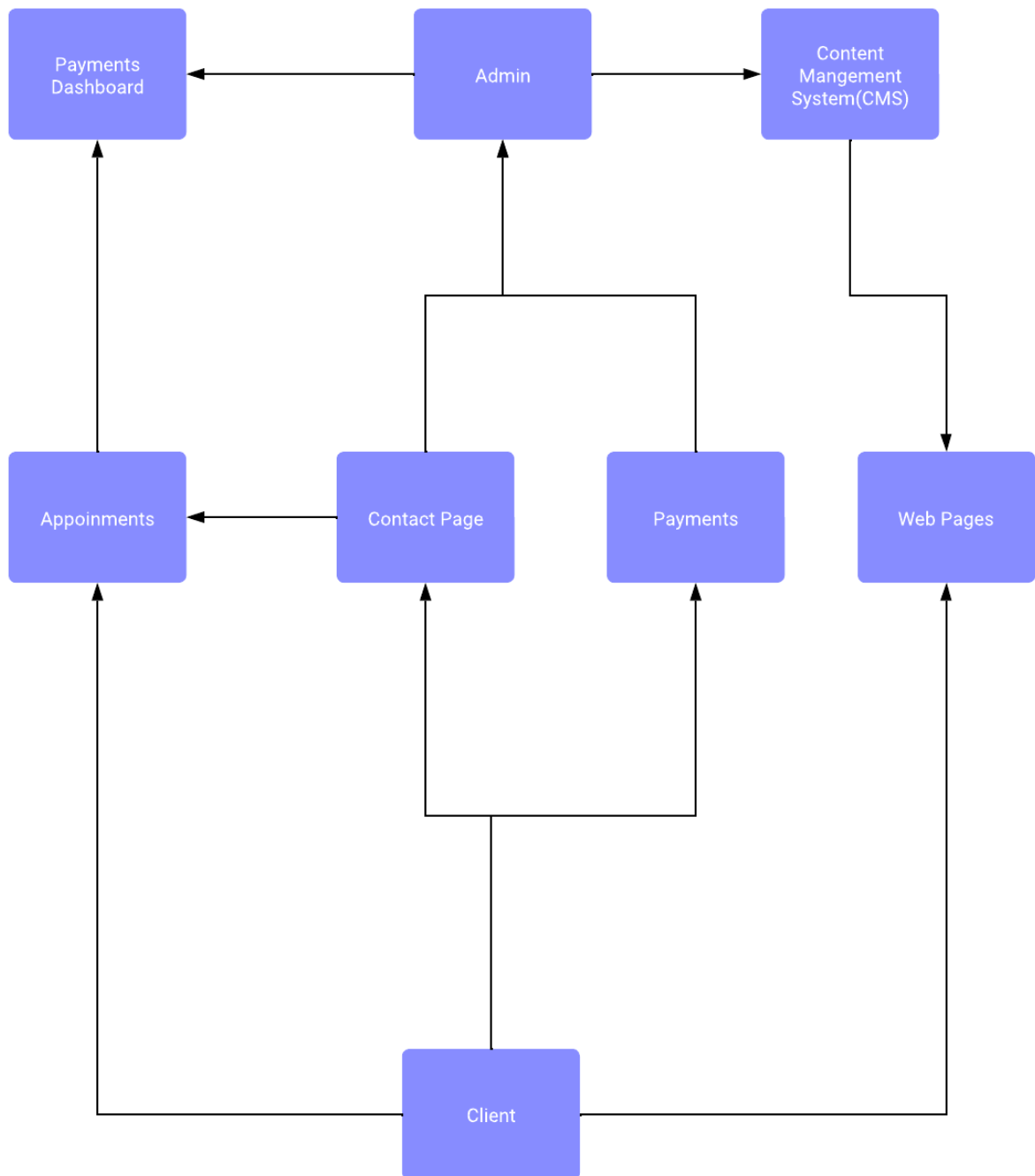
Use Case Diagram



Sequence Diagram



Communication Diagram



2.6 Behavior Model and Description

Events

A listing of events (control, items) that will cause behavioral change within the system is presented.

- Client can schedule an appointment
- Client can make a payment from pricing page
- Admin can configure dynamic content from CMS system
- Admin can generate invoice for custom pricing to send to the client
- Admin can manage payments

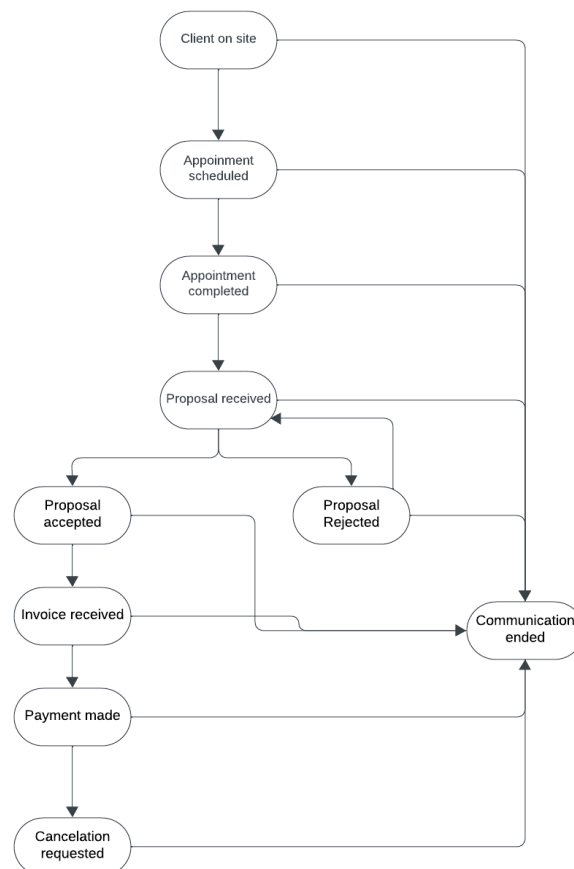
States

A listing of states (modes of behavior) that will result as a consequence of events is presented.

1. Client arrives on site
2. Client schedules appointments with admin
3. Client had appointment
4. Client receives proposal
 - a. Client accepts proposal
 - i. Client receives invoice
 - ii. Client makes a payment
 - iii. Client wants to cancel the maintenance plan
 - iv. Admin cancels plan
 - b. Client reject proposal
 - i. Repeat step 3 or end here
5. Client ends communication

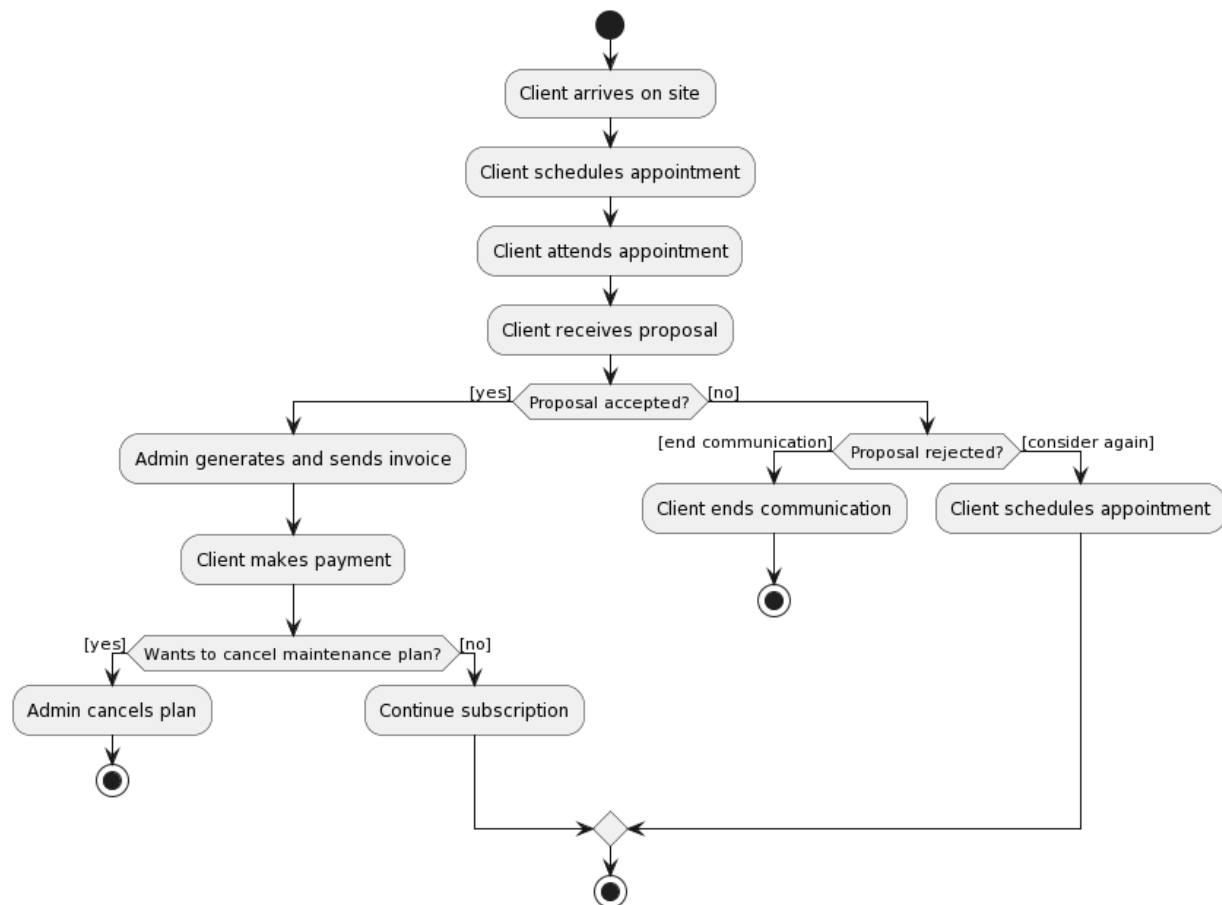
State Transition Diagram

Depict the manner in which the software reacts to external events.



Activity Diagram

Depict the manner in which the software reacts to internal events.



3.0 Hardware/Software Design

3.1 Data Design

A description of all data structures including internal, global, and temporary data structures.

Internal Software Data Structure

Without a backend database, the website will utilize client-side data handling and third-party APIs for dynamic features:

❖ Payment Integration:

- Utilizes secure third-party services (e.g., Stripe, PayPal) for payment processing.
- Client-side JavaScript objects will temporarily hold transaction data before securely transmitting it to the payment service provider.
- Payment confirmation and status will be managed through callbacks from the payment service provider, ensuring secure and seamless transactions.

❖ **Booking System:**

- Integration with external scheduling services (e.g., Calendly) to manage appointments.
- Embedding scheduling widgets directly into the website, allowing clients to book appointments without direct data storage on the site.
- Client-side scripts will handle the interaction with the booking service, providing users with real-time availability and confirmation.

Global Data Structure

Global data structures will focus on content management and site functionality that does not require user-specific data storage:

❖ **Static Content:** Managed through a CMS system in code where we store data as javascript objects, this is what the customer preferred as it keeps code and data in one location.

- About
 - Team members
 - Name
 - Image
 - Title
- Contact
 - Email
 - Phone
 - Whatsapp
- Pricing
 - Name
 - Price
 - Features
 - Stripe link
- Projects
 - Title
 - Image
 - Category
 - Services
 - Website

❖ **User Interactions:** Captured and analyzed through third-party analytics tools like Google Analytics, providing insights without direct data storage on the site.

- Location
- Device
- Browser
- Time
- Session Duration

Temporary Data Structure

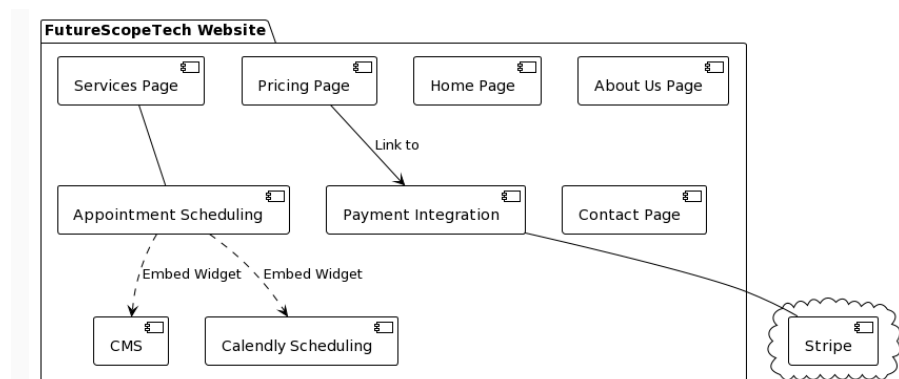
Temporary data related to user interactions and session-specific information are managed using client-side technologies:

- ❖ **Session Information:** Utilizes cookies and the Web Storage API to store session-specific data like user preferences and session IDs.
- ❖ **Form Data:** Information entered in forms, such as contact details or appointment requests, is held temporarily in the browser until processed by the respective third-party service.
 - Full Name
 - Email
 - Subject
 - Message

3.1 Architecture Design

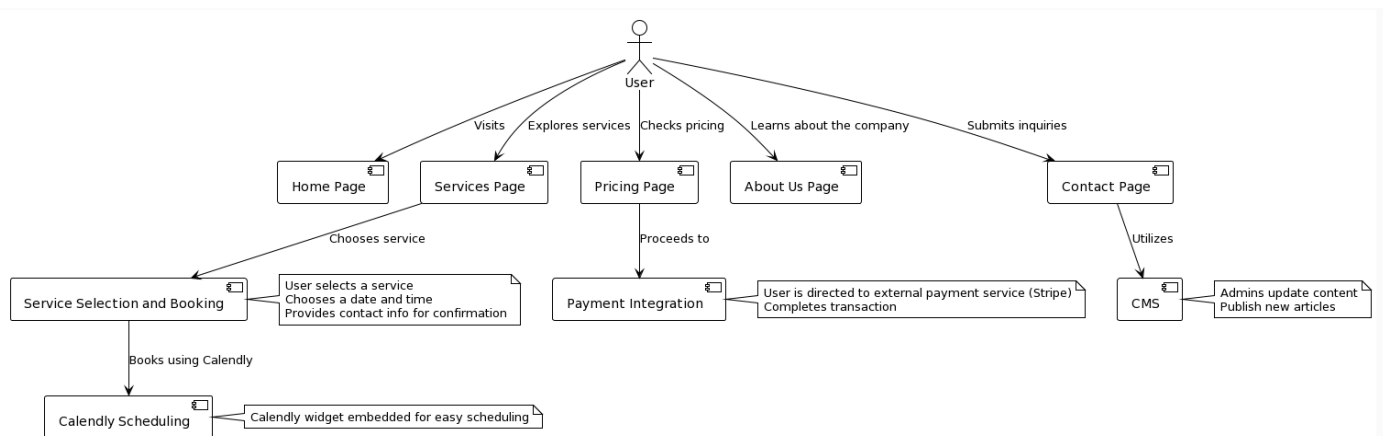
Architecture diagram

The architecture diagram illustrates the main components of the FutureScopeTech website, including the Home Page, Services Page, Pricing Page, About Us Page, and Contact Page. The integration of Calendly for appointment scheduling and Stripe for payment processing is depicted, demonstrating the interaction between these services and the website.



Alternatives

The selected architecture prioritizes simplicity and user-friendliness, employing well-established third-party services to enhance functionality. Alternative architectures considered included building custom in-house solutions for scheduling and payment processing. However, the chosen approach minimizes development time and ensures reliability by leveraging proven platforms.



3.2 Component Design

Each component within the architecture is defined with clear roles and interactions.

Home Page

- ❖ *Major Form(s)*: frmHomePage
- ❖ *Major Action(s)*: Navigate, Display Information
 - The Home Page is the gateway to the FutureScopeTech website, designed to provide a clear and inviting overview of the company's offerings and direct users to navigate to key areas such as Pricing, About, Projects, and Contact Pages.

Pricing Page

- ❖ *Major Form(s)*: frmPricingPage
- ❖ *Major Action(s)*: View Pricing, Redirect to Payment Integration
 - On the Pricing Page, users can explore various service packages, compare pricing, and make informed decisions about purchasing. The page includes a clear pathway for transaction completion through integrated payment solutions.

About Page

- ❖ *Major Form(s)*: frmAboutUsPage
- ❖ *Major Action(s)*: Display Company Information
 - The About Page provides detailed information about FutureScopeTech's mission, values, and the expertise of its team, aiming to build trust and establish a rapport with potential clients.

Projects Page

- ❖ *Major Form(s)*: frmProjectsPage
- ❖ *Major Action(s)*: Showcase Work, Inspire Confidence
 - This page showcases past projects and case studies to demonstrate FutureScopeTech's capabilities and success stories, effectively serving as a portfolio of the company's work.

Contact Page

- ❖ *Major Form(s)*: frmContactPage
- ❖ *Major Action(s)*: Submit Inquiry
 - The Contact Page offers a direct line of communication through a contact form, allowing users to reach out with questions, support requests, or inquiries, which are managed via the CMS for timely responses.

Appointment Scheduling (Calendly Integration)

- ❖ *Major Form(s)*: frmAppointmentScheduling
- ❖ *Major Action(s)*: Book Appointment, Sync Calendar
 - The Calendly integration allows users to book appointments and automatically sync these with their personal calendars, streamlining the scheduling process.

Payment Integration (Stripe)

- ❖ *Major Form(s)*: frmPaymentIntegration
- ❖ *Major Action(s)*: Process Payment
 - Stripe is used for secure payment processing, providing users with a reliable method to complete financial transactions online.

3.4 Software Interface Description

The software's interface(s) to the outside world are described.

External machine interfaces

The website interfaces with user devices, including computers, tablets, and smartphones, through a web browser.

External system interfaces

Integrations with Calendly and Stripe APIs are crucial for appointment scheduling and payment processing, requiring seamless embedding into the site.

Human interface

The human interface consists of the website's front-end, where users interact with the system via graphical elements like buttons, forms, and links.

3.5 User Interface Design

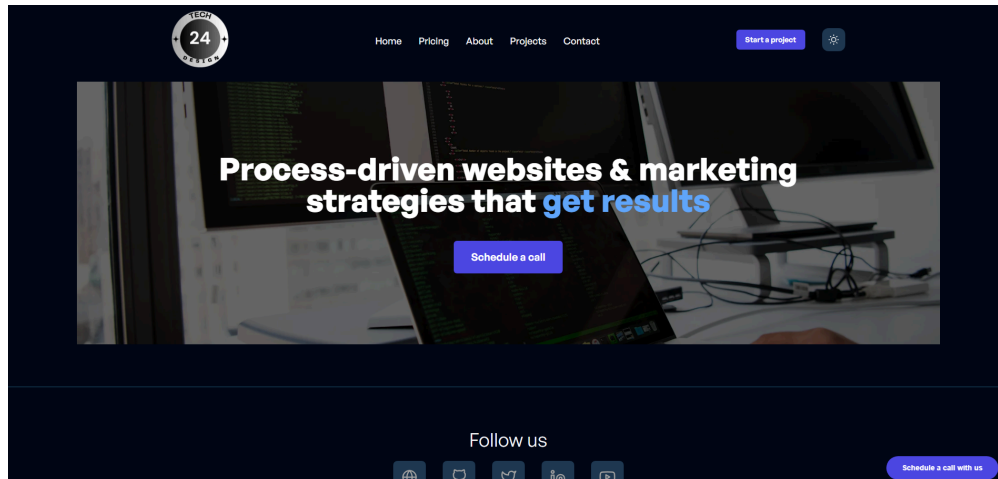
The FutureScopeTech website is designed for optimal user experience, featuring intuitive navigation and clear, engaging content presentation. The design emphasizes accessibility, coherence, and responsiveness, ensuring users can easily access information about the company's tech solutions.

Description of the User Interface

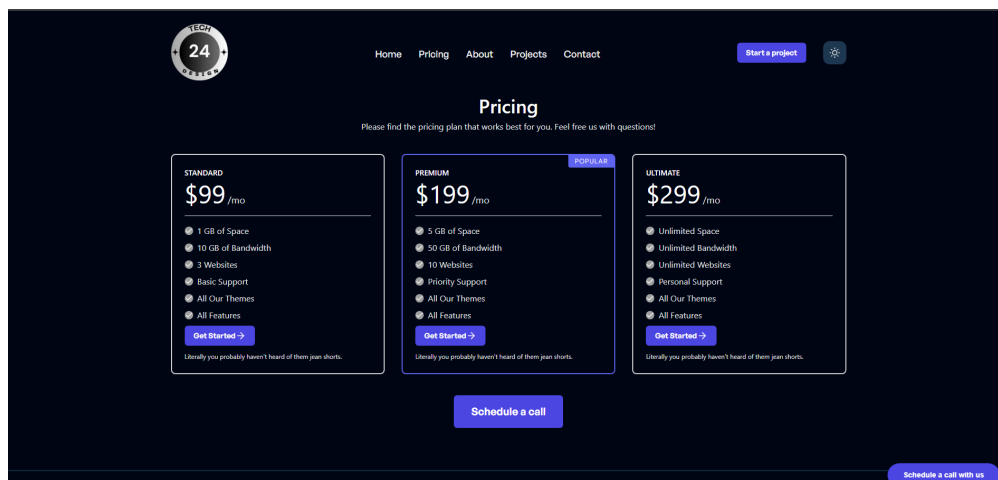
The FutureScopeTech website outlines the company's offerings, team expertise, project showcases, and pricing strategies. Each section, from the Homepage to the Contact Form, is crafted to convey a full spectrum of services and accomplishments, with special emphasis on the Pricing and Projects pages. These sections provide deep dives into the company's value and project successes, underlining their commitment to excellence and customer satisfaction. This comprehensive approach ensures visitors receive a holistic view of FutureScopeTech's capabilities and dedication to quality.

Screen Images

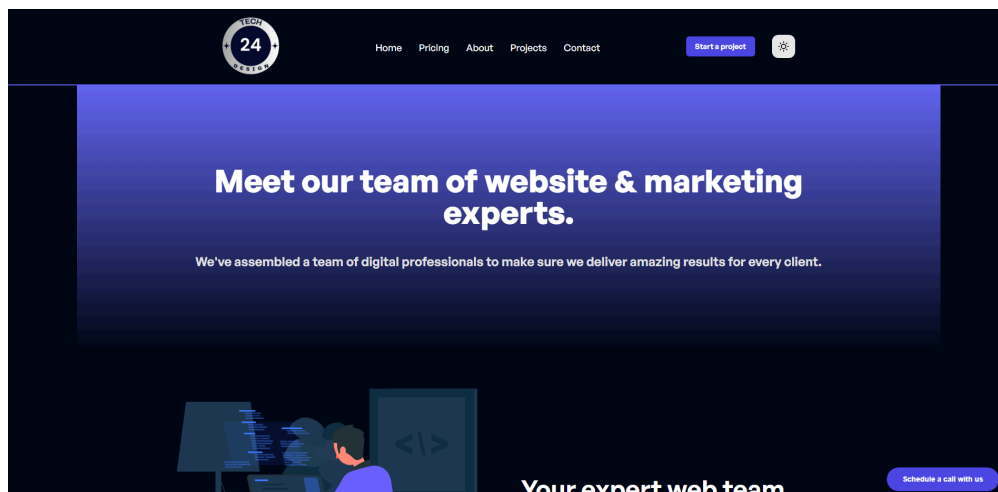
- ❖ **Homepage:** As the portal to the site, the homepage introduces FutureScopeTech with a striking balance of imagery and text, highlighting key services and inviting user engagement with a call to action.



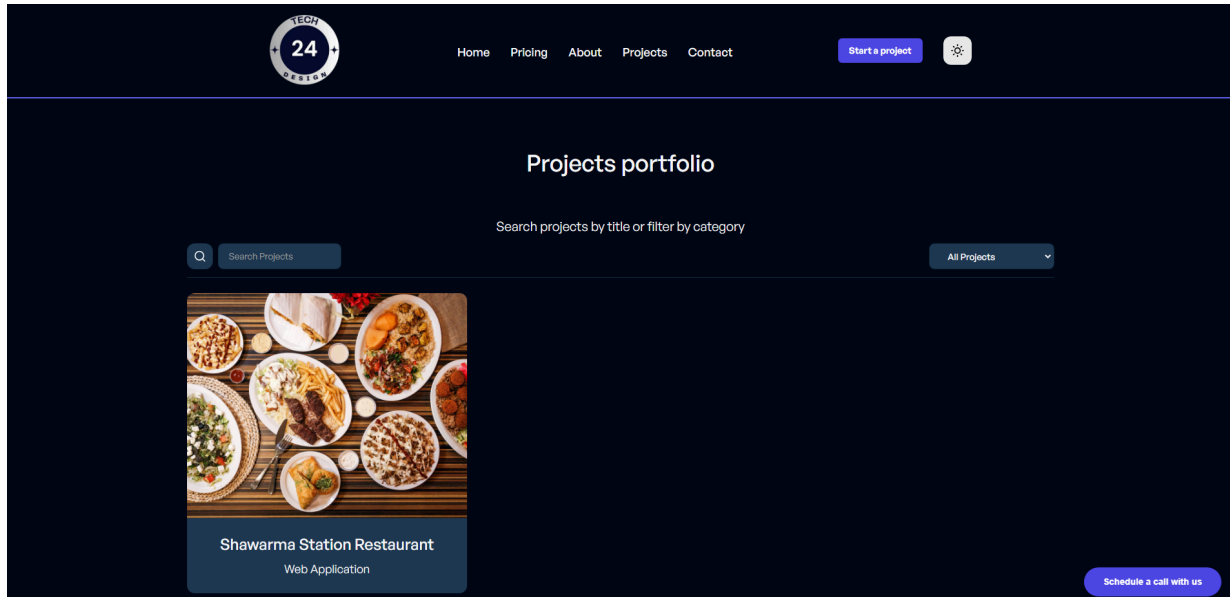
- ❖ **Pricing Page:** This page lays out the service packages in a clear, concise manner, enabling quick comparison and decision-making for potential clients.



- ❖ **About Page:** The About section delves into the company's ethos and the team's expertise, fostering a sense of trust and community with site visitors.



- ❖ **Projects Page:** Showcasing past successes, this page serves as a portfolio, demonstrating the company's capabilities and experience in delivering tech solutions.



- ❖ **Contact Form:** A straightforward form enabling visitors to inquire about services, provide feedback, or ask questions, requiring minimal information for ease of use.

The screenshot shows the 'Contact Form' page of the FutureScopeTech website. The page has a dark blue background. At the top, there is a navigation bar with a logo on the left, a menu with 'Home', 'Pricing', 'About', 'Projects', and 'Contact' in the center, and a 'Start a project' button and a settings icon on the right. Below the navigation bar, the title 'Contact Form' is centered. Underneath the title, there is a form with four input fields: 'Full Name *', 'Email *', 'Subject *', and 'Message *'. Each field has a placeholder text: 'Your Name', 'Your email', 'Subject', and 'Your message'. To the right of the form, there is a section titled 'Contact details' which contains three items: an email address '24techdesign@gmail.com', a phone number '(313) 415-4823', and a WhatsApp link. In the bottom right corner of the page, there is a button labeled 'Schedule a call with us'.

Objects and actions

The interface of FutureScopeTech's website includes various interactive elements:

- ❖ **Navigation Bar:** A constant presence across all pages, providing quick access to different sections of the site.
- ❖ **Call-to-Action Buttons:** Prominently featured, these encourage user engagement, from starting a project to scheduling calls.
- ❖ **Service and Pricing Cards:** These neatly organize the information and allow users to quickly grasp the value propositions.
- ❖ **Contact Inputs:** The contact page presents straightforward fields for message delivery, simplifying direct communication.
- ❖ **Interactive Project Display:** On the projects page, users can interact with the portfolio pieces, offering an immersive experience.
- ❖ **Search Functionality:** Allows users to filter and search for projects, tailoring the display to their interests.

4.0 Implementation Details

The implementation of the FutureScopeTech website involved a series of strategic decisions and the utilization of various technologies to ensure a robust and user-friendly product. Here is a summary of key implementation decisions and resources that were essential for the creation and deployment of the website:

❖ Frontend Development:

- **Technologies Used:** The website's frontend was developed using React.js to create a dynamic and responsive user interface. This choice was driven by React's component-based architecture, which facilitates the modular development of user interfaces.
- **User Experience Design:** Emphasis was placed on creating a user-friendly experience with an intuitive navigation structure. Responsive design principles were applied using CSS frameworks like TailwindCSS, ensuring that the website is accessible and functional across various devices and screen sizes.

❖ Payment and Scheduling Integration:

- **Stripe Integration:** To handle online transactions securely, Stripe was integrated for processing payments. This system supports various payment methods, enhancing the flexibility for users to conduct transactions seamlessly.
- **Calendly Integration:** For scheduling appointments, the website integrates Calendly, allowing users to book appointments without the need for manual intervention. This integration helps manage availability in real-time, improving the efficiency of booking processes.

❖ Content Management System (CMS):

- Dynamic Content Updates: A headless CMS was implemented to manage content updates dynamically. This allows content creators to modify website content directly through the CMS interface without needing to alter the codebase, facilitating timely and efficient content updates.

❖ Security Measures:

- Secure Connections: All data transfers on the website are secured using HTTPS, ensuring that user data is encrypted during transmission. Additionally, security headers and policies are configured to protect against common web vulnerabilities.

❖ Deployment and Maintenance:

- Hosting and Deployment: The website is hosted on Vercel, which provides automatic scaling to handle varying loads, ensuring high availability and performance. The deployment process is streamlined through continuous integration and deployment pipelines, enabling quick and reliable updates to the live environment.
- Performance Optimization: Techniques such as lazy loading of images and asynchronous loading of JavaScript were used to optimize the website's performance. These measures help in reducing the initial load time and improving the user interaction experience.

5.0 Testing/SQA

5.1 Reviews and Audits

In our commitment to delivering top-tier software solutions at FutureScopeTech, we have established a robust system of reviews and audits. This system is designed to uphold our high standards of quality, efficiency, and client satisfaction.

Conducting a Review

FutureScopeTech's review process is divided into two distinct types: client-based reviews and internal team reviews. When modifications to the software are anticipated to impact client performance, it is imperative to seek the client's input initially. However, prior to presenting any such changes to the client, consensus within our team is essential. Furthermore, careful documentation of the project's status both before and after any alterations is crucial for effective project management.

Roles and Responsibilities

SQA Team Structure: Given our team's compact size, each member will assume a multi-faceted role, encompassing various responsibilities and functions.

- ❖ Integration/architecture development: Josh
- ❖ Interface/design development: Abe
- ❖ Integration/functionality tester: Shakia
- ❖ Interface/design tester: Abdalla

Review work products

Weekly Progress Reports

Each team member will compile a detailed weekly report. These reports will not only outline the tasks completed but also delve into the challenges faced, any unresolved issues, and potential risks identified. This documentation is critical in tracking individual contributions and overall team progress.

Change Logs

Any modifications, whether minor tweaks or major overhauls, will be meticulously documented. This change log will include the nature of the change, the reason behind it, who implemented it, and its impact on the project. This provides a transparent history of the project's evolution and assists in tracking the effects of each change.

Client Feedback Documentation

After each client review session, key feedback points, client suggestions, and agreed-upon action items will be formally documented. This ensures that client inputs are accurately captured and addressed in subsequent development phases.

Recorded Meetings

To ensure that all team members are aligned and informed, FutureScopeTech will implement the practice of recording all internal and client meetings. These recordings will capture the full scope of discussions, decisions, and action plans, allowing team members who were unable to attend to stay updated and maintain project continuity.

Formal Technical Reviews

System Specification Review

Description and Focus of the Review:

This review will critically examine the System Specification document to ensure that it accurately represents the intended functionality and overall design of the software. The team will scrutinize the document to identify any potential design flaws, inconsistencies, or areas that lack clarity.

Timing of the Review:

Scheduled immediately after the completion of the System Specification, this review will be one of the first formal technical assessments. Its timing is crucial, as it sets the foundation for the entire project.

Work Products Produced:

Following the review, Josh will compile a comprehensive summary report. This report will detail the findings of the review, highlighting areas of the System Specification that need refinement. It will include a list of identified design flaws, potential impacts on the project, and suggested amendments or enhancements. The team will collaboratively discuss solutions for each identified issue, ensuring that the final System Specification is robust and aligns with the project goals.

Review Checklist:

Are all functional and non-functional requirements clearly documented?

Is there consistency and coherence in the specifications?

Are client concerns adequately addressed?

Are dependencies and constraints clearly defined?

Software Project Plan Review

Description and Focus of the Review:

Led by Josh, this review will evaluate the Software Project Plan for its comprehensiveness, realism, and alignment with project goals. Special attention will be given to resource allocation, milestone planning, and risk management strategies.

Timing of the Review:

Scheduled shortly after the completion of the Software Project Plan, ideally in the early stages of development, to ensure that the project's trajectory is on track.

Work Products Produced:

A summary report will be prepared by Josh, highlighting areas of the project plan that need adjustments. This report will be crucial in guiding the project's direction and ensuring adherence to timeframes and budgets.

Review Checklist:

Is the project plan comprehensive and aligned with organizational goals?

Are roles and responsibilities clearly defined?

Is the timeline realistic and achievable?

Are risk management strategies adequately addressed?

RMMM Review

Description and Focus of the Review:

Shakia will lead the review of the Risk Mitigation, Monitoring, and Management (RMMM) plan. The focus will be on evaluating the identified risks, the effectiveness of monitoring mechanisms, and the proposed mitigation strategies.

Timing of the Review:

Conducted alongside the project planning phase, to ensure that risks are identified and managed from the onset.

Description and Focus of the Review:

The focus is on evaluating the adequacy of risk identification, monitoring mechanisms, and proposed mitigation strategies.

Work Products Produced:

Shakia will compile a report summarizing the RMMM review findings, including any updates or changes needed in the risk management approach.

Review Checklist:

Are all potential risks identified and assessed?

Is there a clear monitoring mechanism for identified risks?

Are mitigation strategies well-defined and feasible?

Is there a contingency plan for high-impact risks?

Requirements Reviews (Models, Specification)

Description and Focus of the Review:

Abe and Abdalla will focus on reviewing the requirements documentation, including models and specifications, to ensure clarity, completeness, and alignment with project objectives.

Timing of the Review:

This review will take place after the initial requirements gathering phase and before design activities begin.

Work Products Produced:

A detailed summary report will be prepared, documenting any changes or enhancements needed in the requirements documentation.

Review Checklist:

Are all functional and non-functional requirements clearly documented?

Are the requirements measurable and testable?

Is there traceability between requirements and project objectives?

Data Design Review

Description and Focus of the Review:

Abe, with support from Shakia, will assess the data design to ensure that it supports system requirements effectively and efficiently.

Timing of the Review:

Scheduled after the completion of requirements and before the start of detailed database implementation.

Work Products Produced:

The review will result in an updated data design document, reflecting necessary adjustments based on the review.

Review Checklist:

Does the data design meet the specified requirements?

Are data structures optimized for performance?

Is there proper normalization and indexing of database tables?

Architectural Design Review

Description and Focus of the Review:

Led by Josh, this review will evaluate the architectural design's alignment with system requirements, scalability, and adherence to best practices.

Timing of the Review:

Conducted after the requirements phase and before detailed component design.

Work Products Produced:

An updated architectural design document will be produced, outlining any necessary modifications.

Review Checklist:

Does the architectural design meet system requirements?

Is the design modular and scalable?

Are key design principles and patterns followed?

Interface (GUI) Design Review

Description and Focus of the Review:

Abe, as the Interface Design specialist, will lead the review of the Graphical User Interface (GUI) design. The review will focus on usability, consistency with design standards, and enhancement of user experience.

Timing of the Review:

This review is scheduled after the architectural design is completed and before detailed component design begins, ensuring the GUI aligns well with the overall software architecture.

Work Products Produced:

Abe will prepare an updated GUI design document, incorporating any changes or enhancements identified during the review.

Review Checklist:

Does the GUI design align with user requirements?

Is the interface intuitive and user-friendly?

Does the design adhere to established style guides?

Component Design Review(s)

Description and Focus of the Review:

Josh and Shakia will collaboratively assess the detailed design of system components, ensuring that they align with the overall architecture and meet specific requirements.

Timing of the Review:

Scheduled after architectural design and before coding begins, to ensure that all components are properly designed for integration.

Work Products Produced:

A report outlining updated component design documentation, including any necessary adjustments for better integration and adherence to design standards.

Review Checklist:

Does each component design align with the architectural design?

Are design patterns and coding standards followed?

Is there proper documentation for interfaces and dependencies?

Code Reviews

Description and Focus of the Review:

Abe and Abdalla will conduct code reviews to ensure efficiency, readability, maintainability, and adherence to coding standards.

Timing of the Review:

These reviews will take place during and after the coding phase, as individual components or modules are completed.

Work Products Produced:

Updated code documentation and revised code incorporating any modifications identified during the review.

Review Checklist:

Does the code adhere to coding standards?

Are variable and function names meaningful?

Are potential security vulnerabilities addressed?

Is the code well-commented for readability?

Test Specification Review

Description and Focus of the Review:

Led by Shakia, this review will evaluate the comprehensiveness and alignment of test specifications with system requirements.

Timing of the Review:

Conducted after the completion of component design and before the initiation of testing activities.

Work Products Produced:

An updated test specification document, reflecting adjustments to ensure thorough coverage of all requirements.

Review Checklist:

Do test specifications cover all requirements?

Are test cases well-documented and executable?

Is there traceability between tests and requirements?

Change Control Reviews and Audits

Description and Focus of the Review:

Josh will oversee reviews and audits focused on the effectiveness of change control processes and their impact on project scope and quality.

Timing of the Review:

Regularly conducted throughout the project lifecycle, with a focus on significant changes or at predetermined milestones.

Work Products Produced:

Updated change control documentation reflecting any necessary adjustments based on the review findings.

Review Checklist:

Are there documented processes for change control?

Is there a clear impact assessment for proposed changes?

Are changes communicated effectively to relevant stakeholders?

SQA Audits

Individual Contribution Tracking

Each team member will submit a bi-weekly summary of their contributions and progress. This report will include completed tasks, any challenges faced, and any assistance needed from the team. This ensures accountability and helps in identifying areas where team members may need support.

Documenting Development Efforts

Team members are responsible for documenting their development efforts, especially when it pertains to specific functionalities they are working on.

Change Management in FutureScopeTech

Before implementing any significant changes that might affect the project's scope or major aspects of the codebase, team members must discuss these changes in a dedicated team meeting. For minor adjustments, communication can be streamlined through an internal messaging system. Regardless of their magnitude, all changes must be recorded in the project's change log.

Client Engagement Strategy for FutureScopeTech

Keeping the client informed is a top priority. Major modifications or updates will be discussed in formal client meetings to obtain feedback and approval. For minor updates, an email summary will be sent to the client's point of contact. This ensures that the client remains engaged and informed about the project's progress.

Version Control and Documentation

Every change, big or small, will be tracked using our version control system. Each update will be documented, detailing what was changed, why, and by whom. This ensures a transparent history of the project's evolution and allows for easy reverting to previous versions if needed. Regular backups of the project will also be maintained as a part of our risk management strategy.

Feedback and Improvement

Regular feedback sessions will be conducted within the team to discuss the effectiveness of current SQA practices. Suggestions for improvements will be encouraged and evaluated. This iterative approach allows the team to continuously refine their methods and enhance the overall quality of the project.

5.2 Quality Tracking

Testing Resources and Staffing

- ❖ **Human Resources:** Our team consists of dedicated test engineers, developers with expertise in Stripe and Calendly APIs, and a project manager to oversee the testing process. For areas requiring specialized knowledge, particularly in security and performance testing, we may consult external experts.
- ❖ **Technical Resources:** We have access to Stripe's sandbox environment for secure payment testing and Calendly's developer accounts for scheduling functionality tests. Our testing will also utilize a variety of devices and browsers to ensure compatibility and tools for performance and security monitoring.
- ❖ **University Resources:** Leveraging the resources provided by our academic institution, we'll have access to university computing labs, which offer a range of software and hardware options for testing environments. This includes servers, desktop computers, and various software tools that can mimic different operating systems and browsers for comprehensive testing.
- ❖ **Training Needs:** Team members will participate in targeted training sessions to deepen their understanding of Stripe and Calendly integrations. These sessions will cover the APIs, best practices for security, and common troubleshooting scenarios to prepare the team for the complexities of testing these integrations.

Test Work Products

- ❖ **Test Plans:** Detailed documents outlining the testing strategy for Stripe integration (covering different payment scenarios, error handling, and transaction security) and Calendly integration (focusing on booking flows, calendar sync, and user notifications).
- ❖ **Test Cases:** A comprehensive set of test cases for each functionality of Stripe and Calendly, including success, failure, and edge cases.
- ❖ **Test Scripts:** Automated scripts for repetitive tests, particularly useful for load and performance testing.
- ❖ **Bug Reports:** Structured reports documenting any issues found during testing, including steps to reproduce, severity, and potential impact on Stripe and Calendly integrations.
- ❖ **Test Summary Reports:** Consolidated documents summarizing the testing efforts, outcomes, and any remaining risks or concerns with the integrations.

Test Record Keeping

- ❖ **Test Logs:** We maintain Excel sheets for logging each test case's execution details, including the date, time, tester, environment, and outcomes. This allows for easy data sorting and analysis.
- ❖ **Defect Logs:** Defects are tracked in a dedicated Excel workbook, capturing their discovery, analysis, corrective actions, and status updates. Conditional formatting highlights unresolved issues for prioritization.

Test Metrics (Using <https://pagespeed.web.dev/>)

- ❖ **PageSpeed Score Improvement:** Tracking the improvement in PageSpeed scores as a primary metric of performance enhancement.
- ❖ **Load Time Reduction:** Measurement of the decrease in page load times post-optimizations.
- ❖ **Resource Optimization:** Tracking the reduction in the size and number of resources loaded (JavaScript, CSS, images) as recommended by PageSpeed.

Testing Tools and Environment

- ❖ **Tools:** Use of automated testing tools for API testing (e.g., Postman for Stripe and Calendly), security testing tools (e.g., OWASP ZAP for web vulnerabilities), and performance testing tools (e.g., JMeter or PageSpeed Insights).
- ❖ **Environment:** Setup of a dedicated testing environment that mimics the production setup as closely as possible, including integration with Stripe's sandbox and Calendly's test accounts, to ensure realistic testing scenarios.

Test Schedule

- ❖ **Timeline:** A detailed timeline aligning testing activities with the overall project schedule. This includes specific periods dedicated to unit testing, integration testing, system testing, and user acceptance testing for Stripe and Calendly features.
- ❖ **Milestones:** Key milestones such as completion of test case development, completion of initial testing rounds, start of user acceptance testing, and final sign-off on the testing phase.
- ❖ **Resource Allocation:** Schedule detailing the allocation of human and technical resources to various testing activities over time, ensuring optimal utilization and coverage of all testing needs.

The following is the tentative schedule for the testing of FutureScopeTech:

- ❖ **Week 1 (Mar 13-19) - Initialization Week:** Finalizing test plan and setting up the environment.
- ❖ **Week 2 (Mar 20-26) - Component Week:** Unit testing of frontend, backend, and APIs.
- ❖ **Week 3 (Mar 27-Apr 2) - Integration Week:** Testing integration of all system components.
- ❖ **Week 4 (Apr 3-9) - Validation Week:** Full system validation against requirements.
- ❖ **Week 5 (Apr 10-14) - Evaluation Week:** Stress, security, and usability testing.
- ❖ **Due Date (Apr 15) - Delivery Day:** Final review and project submission.

Integration with Risk Management:

Quality tracking will incorporate specific strategies from the Risk Mitigation, Monitoring, and Management Plan, particularly focusing on the identified risks of software defects and performance bottlenecks. For instance:

- ❖ Defect Tracking: We will utilize defect logs and regular review meetings to ensure any emerging issues are quickly identified and addressed. This process is part of our broader risk management strategy, which emphasizes preemptive actions to mitigate risks associated with software quality.

Alignment with Software Requirements and Specifications:

- ❖ Our quality assurance practices are tightly coupled with the software's functional requirements and design specifications to ensure compliance and performance:
 - ❖ Specification Verification: Each component's implementation is verified against the detailed specifications provided in the Software Design Specification document. This includes checking functionalities against the stated requirements to ensure they function as intended without deviation.
 - ❖ Requirements Traceability: We employ a systematic approach to trace each test case back to its corresponding requirement as outlined in the Software Requirements Specification. This ensures comprehensive coverage of all features and functionalities and supports our quality control processes.

User Feedback Integration:

- ❖ User feedback is an integral part of our quality tracking. We actively collect and analyze user input during the testing phases, which helps refine the software:
 - Beta Testing Feedback: Feedback gathered during beta testing phases is used to make adjustments before final release, ensuring the software meets both technical and user expectations.

Quality Assurance Processes:

- ❖ Testing Protocols: Our Software Quality Assurance Plan details the use of both automated and manual testing procedures to ensure comprehensive coverage. Automated tests are conducted for regression, performance, and security, while manual tests focus on usability and interface design.
- ❖ Quality Metrics: We track several quality metrics, including defect densities and test coverage, to measure our progress and guide quality improvements.

Documentation and Reporting:

- ❖ Detailed documentation of all QA activities is maintained as part of our project governance model. Regular quality reports are provided to stakeholders, outlining the current quality status and any issues identified during the testing phases.

Feedback Loop and Iterative Improvement:

- ❖ We implement a structured feedback loop with stakeholders, using real-time quality metrics and test results to make informed decisions about software improvements, ensuring a continuous improvement cycle throughout the project lifecycle.

5.3 Testing Strategy

The testing strategy for FutureScopeTech is designed to validate the functionality, reliability, performance, and security of the website across all its components, ensuring a seamless and intuitive user experience. This strategy focuses on critical areas such as user interface components, third-party integrations, and the overall architectural design.

5.3.1 Unit Testing

- ❖ **Objective:** Test individual components for functionality and reliability, particularly focusing on the internal software data structures as outlined in the specification. This includes client-side data handling and interaction with third-party APIs for dynamic features.
- ❖ **Components to be Tested:** Home Page, Pricing Page, About Page, Projects Page, Contact Page, Appointment Scheduling, and Payment Integration.
- ❖ **Approach:** Developers will use JavaScript testing frameworks like Jest to test React components and functions, ensuring that each component behaves as expected in isolation.

5.3.2 Integration Testing

- ❖ **Objective:** Ensure that the components work together seamlessly and third-party services like Calendly and Stripe are integrated properly, as per the architectural design.
- ❖ **Components to be Tested:** Integration of the Appointment Scheduling and Payment Integration components with external services, and their interaction with the rest of the website.
- ❖ **Approach:** Use tools like Playwright for end-to-end testing to simulate user actions and interactions between components and external APIs.

5.3.3 System Testing

- ❖ **Objective:** Validate the complete system against requirements, focusing on the user interface design, responsiveness, and the overall user journey across the website.
- ❖ **Components to be Tested:** Entire website, including all pages and functionalities mentioned in the specification document.
- ❖ **Approach:** Conduct comprehensive system testing to assess the website's functionality, usability, accessibility, and compatibility across different devices and browsers.

5.3.4 Validation Testing

- ❖ **Objective:** Confirm that the website meets business needs and user expectations, with a particular focus on the seamless operation of appointment scheduling, payment processing, and content management through the CMS/Admin system.
- ❖ **Components to be Tested:** The entire website, with a focus on user interaction points like forms, scheduling interfaces, and payment gateways.
- ❖ **Approach:** User Acceptance Testing (UAT) with stakeholders and selected end-users to validate the usability, efficiency, and effectiveness of the website.

5.3.5 High-Order Testing

- ❖ **Stress Testing:** Evaluate the website's performance under extreme conditions, focusing on appointment scheduling and payment processing functionalities to ensure stability.
- ❖ **Security Testing:** Conduct security assessments, especially on payment processing and user data handling, to identify vulnerabilities and ensure data protection.
- ❖ **Performance Testing:** Measure the website's response times and resource utilization, ensuring it meets the performance bounds specified in the design document.
- ❖ **Recovery Testing:** Test the website's ability to recover from failures, particularly focusing on data integrity during payment transactions and appointment bookings.

5.4 Testing Procedures

This section describes as detailed test procedure including test tactics and test cases for the software.

Software (SCIs) to be tested

Refer to Section 2.1 for a detailed enumeration of the Software Configuration Items (SCIs) targeted in this testing phase. This includes all UI components, client-side functionalities, third-party integrations, and security features crucial to the website's operation.

Testing Procedure

Unit Test Cases

- ❖ *Objective*
 - To validate the individual units of code, particularly focusing on client-side scripts and their interactions with Stripe and Calendly APIs.
- ❖ *Components*
 - Home Page scripts for dynamic content loading
 - Pricing Page scripts for service options display
 - Contact Form validation scripts
 - Appointment Scheduling scripts interfacing with Calendly
 - Payment Processing scripts integrating with Stripe
- ❖ *Approach*
 - Employ Jest to create and execute test cases for each function within the components, ensuring each unit performs as expected in isolation.
- ❖ *Expected Results*
 - Each unit test should pass, confirming that individual components function correctly without errors.

Integration Testing

- ❖ *Objective:*
 - To ensure seamless cooperation between the website's various client-side components and the integrated services of Stripe and Calendly.
- ❖ *Components*
 - Integration of the Contact Page with the backend email service
 - Interaction between the Appointment Scheduling feature and Calendly's API
 - Integration of the Payment Processing feature with Stripe's API
- ❖ *Approach*
 - Utilize Playwright for end-to-end testing to simulate real-world user scenarios, verifying the integrated components work harmoniously.
- ❖ *Expected Results*
 - Successful completion of test scenarios without integration issues, demonstrating flawless interaction between components and third-party services.

Validation Testing

- ❖ *Objective:*
 - To validate the entire website against the specified requirements and user expectations, with a keen focus on user experience and overall functionality.
- ❖ *Approach:*
 - Conduct User Acceptance Testing (UAT) with a select group of end-users and stakeholders, using predefined scenarios that cover all website functionalities.
 - Gather feedback on the usability, efficiency, and effectiveness of the website.
- ❖ *Expected Results:*
 - Positive feedback from users, confirming the website meets the outlined requirements and provides a satisfactory user experience.

High-Order Testing

Stress Testing:

- ❖ Objective: Assess the website's performance under peak load conditions, especially for appointment scheduling and payment processing.
- ❖ Approach: Simulate high traffic scenarios using tools like JMeter to stress test the website's responsiveness and stability.
- ❖ Expected Results: The website remains responsive and stable, with no significant performance degradation under high load.

Security Testing:

- ❖ Objective: Ensure stringent security measures are in place for user data handling and transactions.
- ❖ Approach: Conduct vulnerability scans and penetration testing to identify potential security flaws.

- ❖ Expected Results: No critical vulnerabilities found, with all transactions securely processed.

Performance Testing:

- ❖ Objective: Evaluate the efficiency and speed of the website, focusing on load times and interaction responsiveness.
- ❖ Approach: Use performance testing tools to measure key performance indicators (KPIs) under normal and peak loads.
- ❖ Expected Results: The website meets predefined performance benchmarks, ensuring a smooth user experience.

Recovery Testing:

- ❖ Objective: Test the website's ability to recover from failures, particularly in critical functionalities like payment processing and appointment scheduling.
- ❖ Approach: Implement scenarios that simulate failure modes and recovery processes.
- ❖ Expected Results: The website recovers gracefully from failures, maintaining data integrity and user continuity.

5.4.2 Record Results of Specialized Testing Procedures

This subsection outlines the methods and best practices for documenting the results of specialized testing procedures including performance, usability, playtesting, and acceptance tests.

Documentation Approach:

- ❖ **Test Case Documentation:** Each test case will be documented in a Test Case Management Tool, detailing the test scenario, expected outcomes, actual outcomes, and any discrepancies.
- ❖ **Results Logging:** Log detailed results of each test session, including performance metrics, user feedback, and system behavior during the test. Use structured formats such as spreadsheets or dedicated software to maintain consistency.
- ❖ **Error Reporting:** For any failures or bugs identified during testing, use a standardized bug-tracking system to record details such as error description, steps to reproduce, severity, and screenshots if applicable. Assign priority levels for bug resolution.
- ❖ **Performance Benchmarks:** Record performance test results against predefined benchmarks. Document metrics such as load times, response times, and resource utilization in a performance monitoring tool.
- ❖ **Usability Feedback:** Collect and catalog user feedback during usability tests. Use surveys, interviews, and observation notes to capture detailed user interactions and reactions to the software.
- ❖ **Playtesting Sessions:** Document specific observations, player behavior, and feedback during playtesting rounds, focusing on game mechanics, narrative engagement, and overall player satisfaction.
- ❖ **Acceptance Sign-off:** For acceptance tests, prepare a detailed report based on the acceptance criteria agreed upon with stakeholders. Include a sign-off section for stakeholders to approve the final product post-testing.

Data Analysis and Reporting:

- ❖ **Regular Reviews:** Schedule regular review meetings to analyze the testing data collected, identify trends, and discuss the implications for the software development lifecycle.
- ❖ **Performance Trends:** Use graphical tools to visualize performance trends over time. Identify areas where performance may degrade and require optimization.
- ❖ **Quality Assurance Reports:** Generate comprehensive quality assurance reports summarizing the outcomes of all tests conducted, highlighting any potential quality issues.
- ❖ **Stakeholder Reporting:** Prepare tailored reports for different stakeholders, providing relevant test results that align with their specific interests or areas of concern.

Storage and Access:

- ❖ **Secure Storage:** Ensure all test documentation is securely stored in a centralized repository that maintains version control and is accessible to team members based on their roles and responsibilities.
- ❖ **Data Retention Policy:** Define a data retention policy that complies with regulatory requirements and organizational best practices for maintaining and disposing of test records.

Tools and Resources:

- ❖ **Testing Software:** Utilize tools such as JIRA for bug tracking, Jest or Cypress for test case management, and Tableau for data visualization to support effective documentation and reporting.
- ❖ **Integration with Development Tools:** Integrate testing tools with continuous integration/continuous deployment (CI/CD) pipelines to automatically capture and log test results during each deployment cycle.

6.0 Future Maintenance Suggestions

1. Regular Software Updates:

- a. Action: Consistently update the website's software stack, including all frameworks (e.g., React.js), libraries, and third-party APIs (Stripe and Calendly).
- b. Purpose: This will patch security vulnerabilities, improve functionality, and ensure compatibility with new browser updates and technologies.

2. Performance Monitoring and Optimization:

- a. Action: Implement continuous monitoring tools to track the website's performance. Analyze metrics such as page load times, server response times, and user engagement.
- b. Purpose: Quick identification and resolution of performance bottlenecks to maintain a smooth user experience.

3. Security Audits and Enhancements:

- a. Action: Schedule regular security audits and reviews, including penetration testing and vulnerability scanning.
- b. Purpose: To protect user data and prevent security breaches, ensuring that the website complies with the latest security standards and regulations.

4. Content Updates and Management:

- a. Action: Establish a routine for updating and reviewing website content through the CMS to ensure accuracy and relevancy.
- b. Purpose: Keeping the website content fresh and relevant improves SEO, engages users, and supports marketing efforts.

5. User Experience (UX) Improvements:

- a. Action: Collect and analyze user feedback regularly to identify areas for UX improvements. Implement A/B testing to test changes and innovations in the UI.
- b. Purpose: Enhancing user satisfaction and engagement, leading to increased usage and conversions.

6. Backup and Disaster Recovery Planning:

- a. Action: Enhance and test the website's backup and recovery procedures to ensure data integrity.
- b. Purpose: Quick recovery from data loss or website downtime, minimizing impacts on users and the business.

7.0 Delivery/Acceptance Statement



24TECHDESIGN

INNOVATIVE DESIGN,
LASTING IMPRESSIONS

+1 (313) 415-48
24techdesign@gmail.com
Allen Park, MI, USA

March 25, 2024

To:

Bruce R. Maxim, PhD
Nattu Natarajan Professor of
Engineering
Computer and Information Science
University of Michigan-Dearborn
313-436-9155

Dear DR Maxim,

I hope this letter finds you well. I am pleased to provide my final evaluation comments on the senior design project completed by Josh, Ibrahim, Shakia, and Abdalla, in accordance with your instructions. Please find enclosed the evaluation report addressing the three points outlined in your communication.

1. Project Environment Compliance:

- The project ran seamlessly within the specified environment outlined in the requirements document.

2. Alignment with Expectations:

- The delivered products met our negotiated expectations admirably. The team demonstrated exceptional dedication to fulfilling the outlined requirements.

3. Delivery of Essential Resources:

- The team provided all necessary deliverables, including source code, program executables, documentation, passwords, and relevant resources essential for post-completion maintenance.

I commend the team for their exemplary dedication and teamwork throughout the project duration. They have showcased outstanding problem-solving skills and commitment to delivering a quality product.

Thank you for the opportunity to work with your students. I eagerly anticipate future collaborations.

Regards,

Amr Mohsin
24TechDesign Founder

8.0 References & Bibliography

❖ Books and Articles:

- Learning React: Modern Patterns for Developing React Apps by Alex Banks and Eve Porcello - Provides foundational knowledge in React used in the project.
- CSS in Depth by Keith J. Grant - Helped with advanced CSS techniques implemented using Tailwind CSS.

❖ Web Sites and Tutorials:

- Calendly. Used for scheduling appointments within the system, integrating seamlessly with the website to avoid booking conflicts.
- Stripe. Payment processing service used for handling transactions, including one-time payments, subscriptions, and custom invoicing.
- Vercel. Deployment and continuous integration/continuous deployment (CI/CD) platform used for hosting the project and managing deployment pipelines.
- GitHub. Code repository hosting service used for version control and collaboration, featuring integration with Vercel for CI/CD.
- Next.js. The React framework used for building the user interface of the website, enabling server-side rendering and generating static websites.
- Tailwind CSS. A utility-first CSS framework used for designing modern, responsive layouts for the project.
- Builder.io. Mentioned as the Content Management System (CMS) for dynamically managing website content.
- Google Analytics. Service used for tracking website traffic and user interactions to help measure the effectiveness of the website.
- Sanity.io and Hygraph. External CMS systems mentioned for their integration capabilities with the project for managing dynamic content.

❖ Tutorials:

- React Documentation - Official tutorials that provided guidance on React components, hooks, and state management.
- Next.js Tutorials by Vercel - Step-by-step guides to building and deploying sites with Next.js.

❖ Code Libraries and Frameworks:

- React Router - Used for navigating pages in the single-page application.

9.0 Appendix A - User Manual

Introduction

- ❖ Purpose of Product: The website is designed to facilitate seamless interactions with our services, specifically tailored for appointment scheduling and information dissemination without the need for user logins.
- ❖ Operating Environment: Accessible through modern web browsers such as Chrome, Firefox, Safari, and Edge on desktop, tablet, and smartphone devices.
- ❖ General Functionality: Users can view service details, schedule appointments, and access payment options directly on the website.
- ❖ Special Features: Real-time scheduling via Calendly and secure payment processing through Stripe ensure a streamlined user experience.
- ❖ Limitations: The site requires an active internet connection and does not store user data or provide account management functionalities due to the absence of a user database.

Installation

- ❖ Physical Requirements: There are no physical installation requirements, as the website operates entirely online.
- ❖ Copying & Backup Procedures: Not applicable to end-users, as all content is managed and updated centrally without user intervention.
- ❖ Software Installation Instructions: No software needs to be installed by users, as the website is accessed directly through a web browser.

Tutorial

- ❖ Walkthrough of Example Session: Includes navigating to the service scheduling page, selecting a service, choosing a time for an appointment, and completing a payment transaction.
- ❖ Explanation of Example Session: The tutorial will detail each step, demonstrating how users can interact with the website functionalities smoothly.
- ❖ Help Facilities Provided: Online help includes a FAQ section that addresses common questions and concerns. Direct support can be accessed via email.

Detailed Instructions

- ❖ Operation of Product: Direct access to the website allows users to interact with the available services without the need for registration or login.
- ❖ How to Run All Product Functions: Users can select services, schedule appointments, and make payments using straightforward, guided processes on the website.
- ❖ Sample Output from Product: Users will receive email confirmations detailing their appointments and payment receipts directly to their provided email addresses.
- ❖ Sample Input to Product: Required inputs include selecting service options, appointment times, and payment details.
- ❖ Error Handling Supported by Product: If errors occur during appointment scheduling or payment, the website provides clear error messages and steps for resolution.

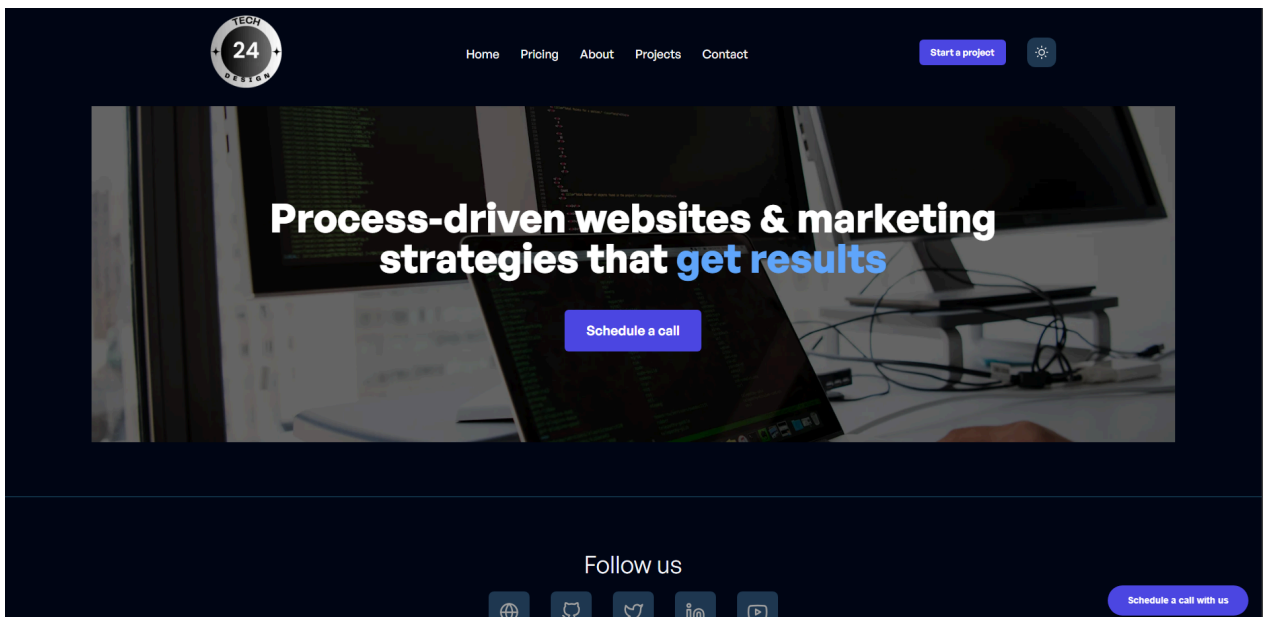
Technical Details

- ❖ Advanced Features: Integration with third-party services like Calendly for scheduling and Stripe for payments enhances functionality without the need for internal data management.
- ❖ Modification of Product: Users can modify their scheduled appointments through links provided in their confirmation emails.
- ❖ Support Information: For assistance, users can contact support through an embedded form on the website or by sending an email to 24techdesign@gmail.com. Support is available during business hours, with responses typically within 24 hours.

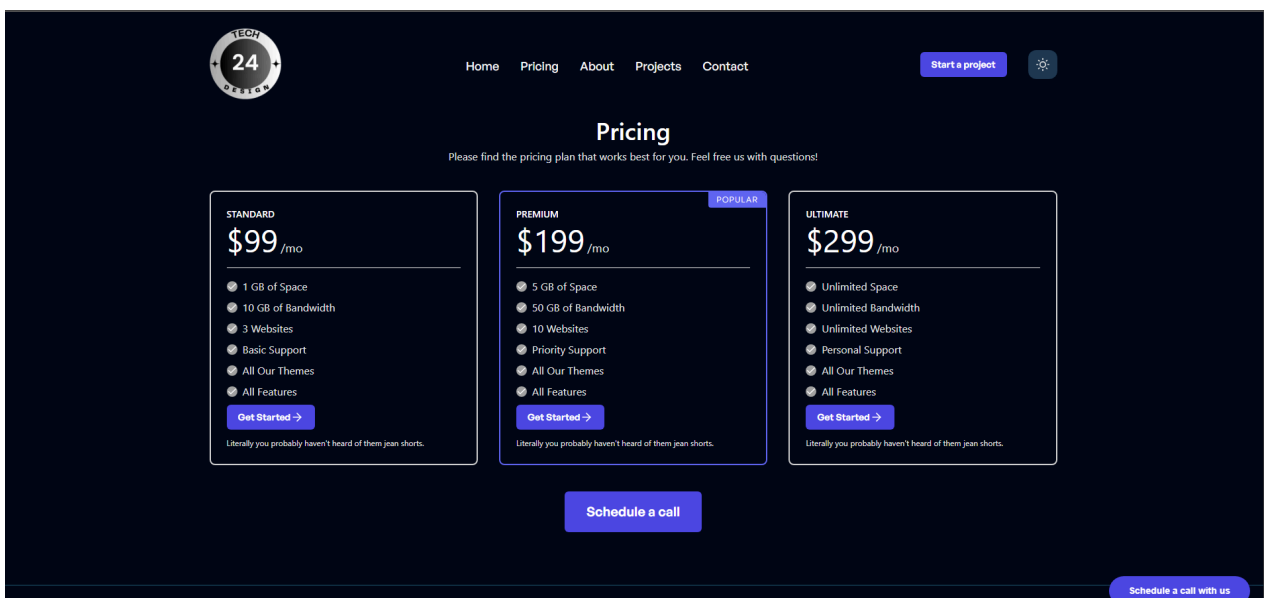
10.0 Appendix B - Sample Output

❖ Homepage

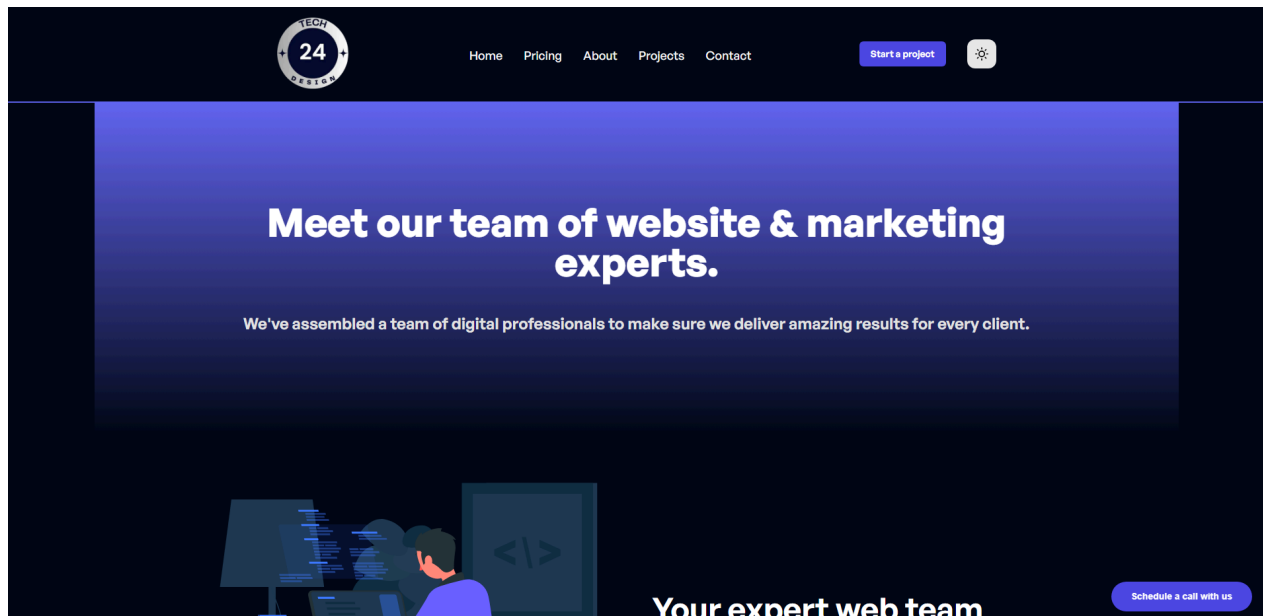
- The homepage serves as the gateway to the website, featuring a clean and modern design that highlights key services with vibrant imagery and compelling call-to-action buttons. Navigation is intuitive, with a top menu that provides easy access to all sections of the site. Prominent features include a dynamic slideshow or video background that showcases the company's expertise and areas of focus.



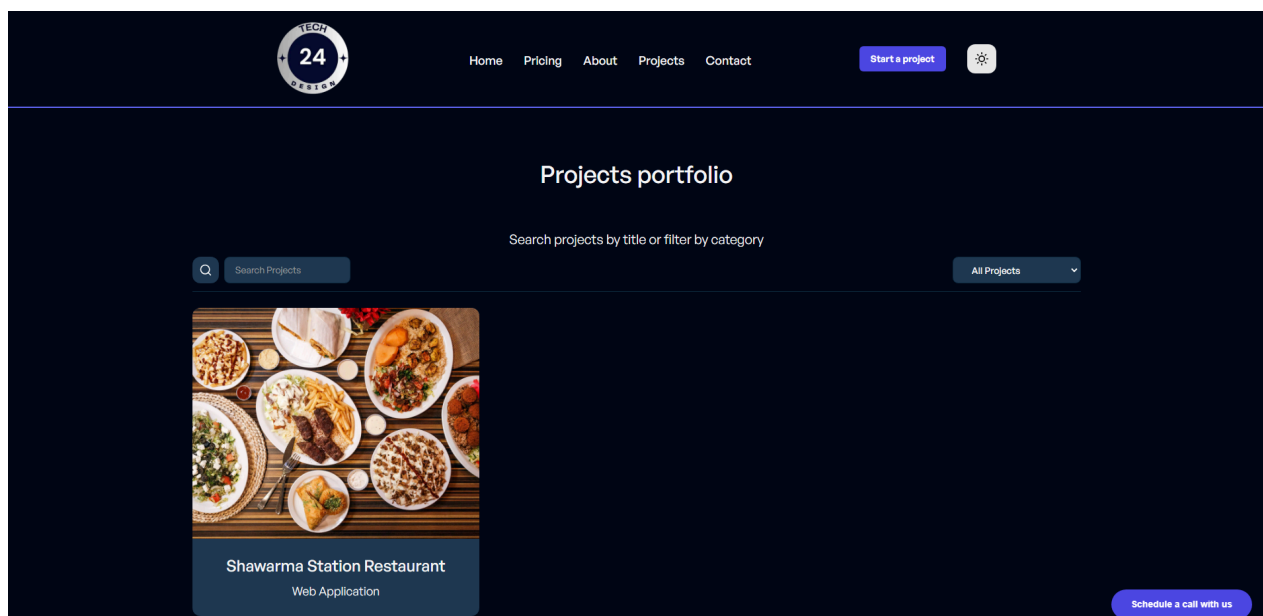
- ❖ **Pricing Page:** This page outlines the various service packages offered, designed to cater to a range of client needs from basic to premium options. Each package is clearly presented with detailed descriptions of the features included, price points, and benefits. Visual cues such as icons or comparison charts help users quickly grasp the value proposition of each option. A direct link or button for users to sign up or contact for more details is also featured prominently.



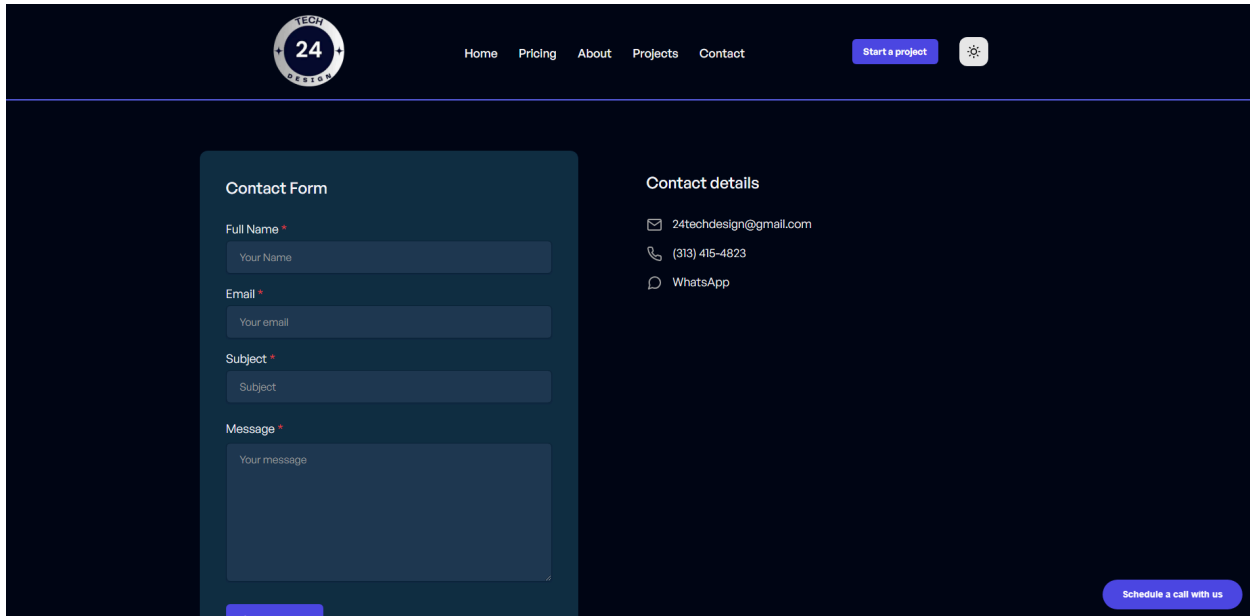
- ❖ **About Page:** The About page provides comprehensive information about the company's background, mission, and the team behind the services. It includes professional portraits of team members, brief biographies, and their roles within the company. This page aims to build trust by highlighting the company's commitment to quality and customer satisfaction, supported by testimonials and achievements.



- ❖ **Projects Page:** Showcasing previous work, this page acts as a portfolio highlighting detailed case studies and examples of completed projects. Each project is displayed with high-quality images or videos, a brief overview of the project scope, the solutions provided, and the outcomes achieved. Links to detailed case studies or client testimonials provide deeper insights into the company's capabilities.

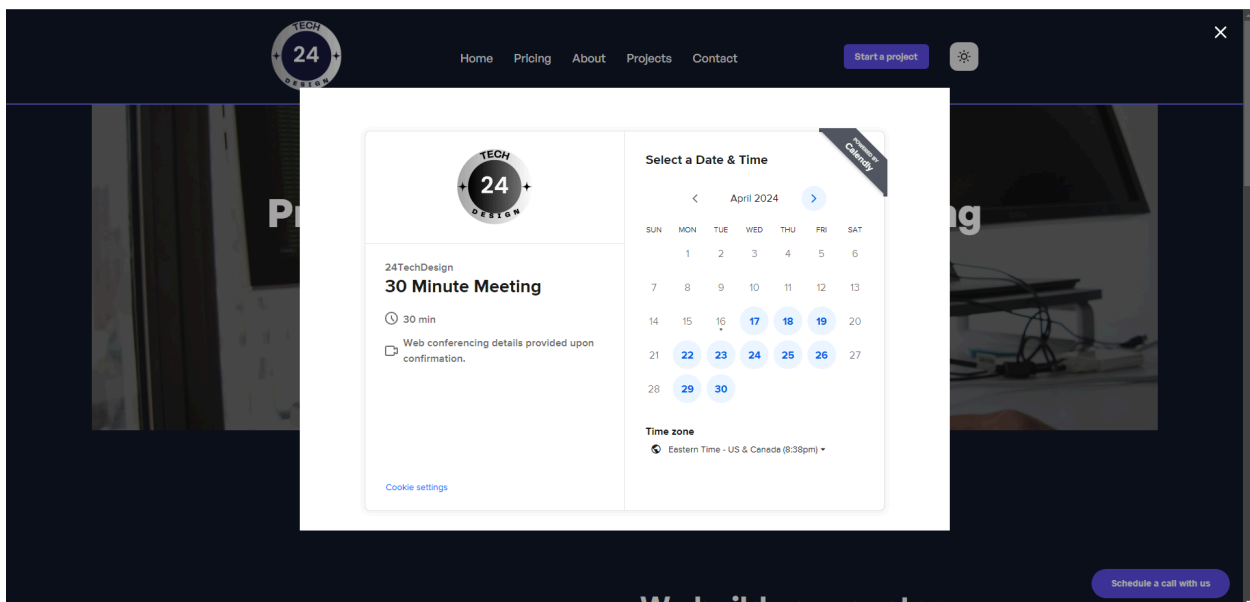


- ❖ **Contact Form:** The contact form is designed for simplicity and ease of use, encouraging visitors to get in touch with minimal effort. The form includes fields for the user's name, email address, subject, and a message box. Instructions for filling out the form are clear, and information on privacy and how the data will be used is provided to reassure users. A submission button labeled clearly indicates the next step, and users receive immediate confirmation that their message has been sent.



The screenshot shows the contact form on the 24TechDesign website. The form is titled "Contact Form" and includes fields for "Full Name", "Email", "Subject", and "Message". To the right of the form, under "Contact details", the email "24techdesign@gmail.com", phone number "(313) 415-4823", and "WhatsApp" are listed. A "Start a project" button is visible in the top right corner, and a "Schedule a call with us" button is at the bottom right.

- ❖ **Scheduling with Calendly:** The image captures the Calendly appointment scheduling interface overlaid on the website's interface, offering a seamless user experience. Users can effortlessly book a '30 Minute Meeting' with 24TechDesign by selecting a date and time from the embedded calendar. The current month is displayed, with available dates highlighted for user convenience. Time slots are presented below the calendar, allowing the user to choose a time that suits their schedule. The selected date and time are prominently displayed, along with a note that web conferencing details will be provided upon confirmation. The process simplifies booking an appointment, indicating the company's dedication to efficient customer service.



The screenshot shows the 24TechDesign website with a Calendly scheduling overlay. The overlay displays a "30 Minute Meeting" with 24TechDesign, a 30-minute duration, and a note that web conferencing details will be provided upon confirmation. The calendar shows the month of April 2024, with dates 17, 18, 19, 22, 23, 24, 25, 26, 29, and 30 highlighted. The time zone is set to "Eastern Time - US & Canada (8:38pm)". A "Cookie settings" link is visible at the bottom left of the overlay. The background shows the website's navigation bar and a "Start a project" button.

- ❖ **Paying with Stripe:** This image showcases the Stripe-powered payment form for a 'Standard Website Maintenance' subscription priced at \$99.00 per month. The form is divided into sections for clear organization, starting with 'Contact information' where users enter their email and phone number. The 'Payment method' section offers options like credit card or alternative payment services, such as Cash App Pay. Users are guided to input their card details, billing address, and are given the option to save their information for future transactions, emphasizing convenience with security. The layout is clean, professional, and concludes with a prominent 'Subscribe' button, assuring users of a secure, smooth, and transparent transaction facilitated by Stripe's robust infrastructure.

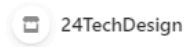
The screenshot displays a Stripe payment form for a 'Standard Website Maintenance' subscription. The form is divided into two main sections: a pricing summary on the left and a payment form on the right.

Pricing Summary (Left):

- 24TechDesign logo
- Subscribe to Standard Website Maintenance
- \$99.00** per month
- Standard Website Maintenance** \$99.00
This package includes creating the website based off the given requirements, hosting services, and one year act...
Billed monthly
- Subtotal \$99.00
- Tax ⓘ Enter address to calculate
- Total due today \$99.00
- Powered by stripe | Terms | Privacy

Payment Form (Right):

- Contact information
- Contact information
email@example.com
(201) 555-0123 ⓘ
- Payment method
Card Cash App Pay
- Card information
1234 1234 1234 1234
MM / YY CVC
Cardholder name
Full name on card
- Billing address
United States
Address
Enter address manually
- ☒ Save my info for 1-click checkout with Link
Securely pay on 24TechDesign and everywhere Link is accepted.
link · More info
- Subscribe**
- By confirming your subscription, you allow 24TechDesign to charge you for future payments in accordance with their terms. You can always cancel your subscription.
- By clicking Subscribe, you agree to Link's terms and privacy policy.



Subscribe to Premium Website Maintenance

\$199.00 per month

Premium Website Maintenance \$199.00
Payment for monthly maintenance which includes minor changes (changing images, updating content, social...
Billed monthly

Subtotal \$199.00

Tax ⓘ Enter address to calculate

Total due today \$199.00



Subscribe to Ultimate Website Maintenance

\$299.00 per month

Ultimate Website Maintenance \$299.00
This package includes creating the website based off the given requirements, hosting services, and one year act...
Billed monthly

Subtotal \$299.00

Tax ⓘ Enter address to calculate

Total due today \$299.00

11.0 Appendix C - Team Member Resumes

1. Resume - Ibrahim Ammar

Ibrahim Ammar

Dearborn, MI | [LinkedIn](#) | [GitHub](#)
imammar@umich.edu | (313) 558-3804

EDUCATION

University of Michigan - Dearborn

Dearborn, MI

Bachelor of Science: Computer and Information Science, concentration in Computer Science

Expected Apr.2024

- GPA: 3.5/4.0, Dean's List (7 semesters, 2021-2023)
- Relevant Coursework: Mobile Applications, Software Security, Computer Networks, Data Structure/Algorithms

Henry Ford College

Dearborn, MI

Michigan Transfer Agreement: Computer Science

May, 2021

GPA: 3.7/4.0, Dean's List (7 semesters, 2018-2021), Received H. Hackett Scholarship, Henry Ford Engineering Club

PROJECT/RESEARCH

University of Michigan – Dearborn

Dearborn, MI

COVID-19 Research (Probability and Statistics)

Feb. 2022 – Apr. 2022

- Conducted in-depth COVID-19 research with a collaborative team, focusing on Probability and Statistics.
- Created insightful visuals in Tableau, performed advanced analyses with Minitab, and ensured data quality using Excel, contributing crucial findings to the research.
- Crafted a detailed 300-slide PowerPoint using Tableau and Minitab visuals, effectively presenting COVID-19 data

Online Crypto Currency Trading (Socket Programming; TCP sockets; SQL)

Dec. 2022 – Apr. 2023

- Collaborated with a team to develop a crypto currency trading application using network sockets.
- Programmed client and server portions of the application using C++.
- Implemented commands client and server portions: BUY, SELL, DEPOSIT, BALANCE, LIST, QUIT, SHUTDOWN, LOGIN, LOGOUT, WHO, LOOKUP.

Full Stack Web Platform Development – Abe's Burger Shop (Front-End, Node/PHP, Databases) Nov. 2023 – Dec. 2023

- Developed the creation of Abe's Burger Shop's web platform, integrating HTML, CSS, and JavaScript (React) for a dynamic front-end, coupled with a Node/PHP-powered back-end interfacing with SQL databases.
- Enhanced user engagement through intuitive order customization and real-time tracking functionalities.

Client Collaboration: FutureScopeTech Web Revamp (Senior Design Project)

Aug. 2023 – Present

- Collaborating on a modern website redesign with FutureScopeTech, incorporating latest trends with React.js.
- Contributing to the implementation of user account creation, integration of secure payment features, and development of a booking system for improved personalized interactions and seamless transactions.

WORK EXPERIENCE

Primealete Nutrition

Dearborn, MI

Manager

Dec. 2019 – Dec. 2023

- Trained kitchen staff in safely operating and handling equipment, machinery, and utensils.
- Ensured high food quality standards by checking delivery contents to verify product quality and quantity
- Demonstrated consistent care, attention to detail, and quality while creating identical dishes daily.
- Initiated cost-saving measures while maintaining culinary excellence, significantly reducing operational expenses.

Gas Station

Royal Oak, MI

Cashier


Jan. 2022 - Aug. 2022

- Provided exceptional customer service by welcoming customers and assisting them in finding store items.
- Processed cash and credit payments for gasoline, food, and other merchandise.
- Managed in-store food and general merchandise inventory, stocked goods, and documented low stock levels.

TECHNICAL SKILLS

- **Programming Languages:** C++, Java, Python, SQL (MySQL Workbench)
- **Web Development:** HTML, CSS, JavaScript (React.js); Experienced in Full-Stack Development
- **Software & Tools:** GitHub, Jira, Tableau, Minitab, PowerBI, Microsoft Office Suite, Agile Methodologies, SCRUM
- **Development Environments:** Microsoft Visual Studio, Linux, Eclipse, Dev C++, IntelliJ IDEA, VS Code

2. Resume - Josh Weber

<div><h1>Josh Weber</h1><p>Software Engineer</p><h3>Contact</h3><p> 734-664-5264</p><p> jweb89@yahoo.com</p><p> www.joshweber.netlify.app</p></div>	<h3>Skills</h3> <p>React, Go, Redux, Cypress, React Native, CI/CD, Bootstrap, HTML, SQL, AWS, Git, C++, CSS, Node, Javascript, Typescript, SEO, PostgreSQL, Wordpress, MongoDB, Docker</p>
	<h3>Work History</h3>
	<h4>Full Stack Engineer</h4> <p><i>Seniorly(remote), San Francisco, CA - 03/21 - Current</i></p> <ul style="list-style-type: none">• Created new automated testing infrastructure to free up developer resources and increase test coverage• Revamped authentication system using React, Golang, and SQL• Designed and implemented user event tracking on frontend and backend API to obtain better insights
	<h4>Full Stack Engineer</h4> <p><i>Ziggurat Technologies (remote), San Francisco, CA - 11/20 – 03/21</i></p> <ul style="list-style-type: none">• Developed new Node API using Express to deliver stock picks to customers using the mobile app.• Managed two interns who made significant contributions to the authentication flow.• Created reusable libraries in React and React Native
	<h4>Owner and Web Developer</h4> <p><i>PC Swaps – Online Marketplace, pcswaps.com - 03/18 – 11/21</i></p> <ul style="list-style-type: none">• Grew website to over 10,000 users and 500 transactions using SEO, social media, and organic growth• Developed scalable code to handle increasing traffic and transactions.• Managed all facets of the business including accounting, customer support, marketing, and development.
	<h3>Education</h3>
	<h4>University of Michigan - Dearborn</h4> <p><i>BS: Computer Science and Data Science – 08/21 – 12/24</i></p> <ul style="list-style-type: none">• Chancellors Scholarship• CECS Rebecca Campbell Memorial Scholarship• BS-CSDS Honors Scholar Award

3. Resume - Abdalla Ibrahim

Abdalla Ibrahim

734-757-3243 | abdallaibrahim2000@outlook.com | Dearborn, MI | [LinkedIn](#)

OVERVIEW

Motivated computer science major with excellent problem solving and leadership skills. Expertise in many programming languages and integrated development environments. Demonstrated results in working with teams and accomplishing goals. In search of an entry level job with an organization focused on individual growth and skill development in computer and information science.

EDUCATION

Bachelor of Science in Computer & Information Science: University of Michigan – Dearborn

- **Major:** Computer Science | **GPA:** 3.2 | **Expected Graduation:** April 2024
 - College of Engineering & Computer Science University Honors | Fall 2020, Winter 2021, Summer 2021

EXPERTISE

Programming Languages: C++ | Java | Python | SQL

Web Development: JavaScript | CSS | HTML

Applications: Microsoft Visual Studios | JetBrains IntelliJ IDEA | Oracle VM VirtualBox | Android Studio | Oracle SQL Developer | Git Bash | Apache NetBeans | Atom | Eclipse | Enterprise Architect | Linux | Jira | Microsoft Office Tools

PROJECT EXPERIENCE

Error Tracker Website | Software Engineering - CIS375 | Fall 2022

- Created a functional error tracking system website that allows users to search, add, edit errors, and update existing errors.
- Developed a web-based application using JavaScript that is compatible with multiple browser applications and is mobile friendly.
- Performed all levels of the software development lifecycle that included specs, design construction, test planning, execution, and validation.
- Collaborated with team members from start of development through end of product lifecycle.
- Presented at regular intervals throughout project development.

Arithmetic Game | Mobile Application Development- CIS436 | Winter 2023

- Created a two-player game using Kotlin that incorporates a die, and depending on the number rolled, would provide a math equation for the player to solve.
- Developed game application using Android Studio to create backend code for game functionality (i.e., roll, guess, type of problem) and designed user interface.
- Created parameters for addition, subtraction, and multiplication equations.
- Designed test cases to check functionality of application across multiple android devices.

Linux Bash Shell Commands | Software Security | CIS449 | Winter 2023

- Used Virtual Machine running SEED Ubuntu to allow the use of shell commands in LINUX terminal.
- Utilized PuTTY and FileZilla to enable SSH and SFTP that allowed the modification, search, and replacement of system files and navigation of directories.
- Led demo instruction of shell commands to peers and professor.

Tech 24 Design Website - Senior Design Project | CIS4952 | Winter 2024

- Developed a modern website using React.js for a local tech company that integrated diverse payment options (Stripe), use of an external scheduling tool (Calendly), and real-time update admin system for content management.
- Coordinated bi-weekly meetings to ensure client needs are met and any feedback on design is provided.

EMPLOYMENT & EXTRA-CURRICULAR ACTIVITIES

Melvindale Pharmacy, Melvindale, MI, November 2020 – Current

- Interact with customers, stock shelves and deliver medications to patients.
- Work with pharmacists, pharmacy technicians and owners.

Volunteer, Children's Hospital of Michigan - Detroit, MI | January 2018 – March 2020

- Provided companionship to children undergoing cancer treatment and bone marrow transplants by offering personal insight from my experience as a pediatric cancer patient.

HONORS AND AWARDS

- Impact Opportunity Scholarship
- Kohler Cancer TRC Scholarship
- Henry Ford Village Scholarship

4. Resume - Shakia Shahid

Shakia Shahid

shakiashahid@gmail.com | 586-438-7786 | Madison Heights, MI

EXPERIENCE

eInfochips Limited - Infotainment Validation Engineer

May 2022 - present

Farmington Hills, MI

- Performed smoke testing, sanity testing, and Basic Functional Test (BFT) validation on automotive infotainment systems for both vehicle and bench to ensure adherence to industry standards provided in the requirement documents.
- Used tools like Vector CANoe/CANalyzer and diagnostic tools like CANape to analyze Controller Area Network (CAN) communication and troubleshoot software/hardware problems.
- Utilized Pytest to execute automated test scripts for regression testing, UI testing, and integration testing of infotainment software system.
- Applied network simulation tools such as CarSim to accurately replicate real-world driving scenarios and validate the functionality of multimedia features like navigation, voice recognition, and connectivity.
- Monitored system performance using performance monitoring tools such as Wireshark to improve infotainment system performance.
- Efficiently resolved bugs and ensured timely completion of validation tasks by managing issues throughout the validation process.

Aerotek Staffing - Software Quality Assurance Test Engineer

November 2020 - April 2022

Wixom, MI

- Created test plans and strategies to ensure the functionality, performance, and validation of system features such as infotainment software, navigation, connectivity, and power management features.
- Identified defects, bugs, and issues during testing and carefully recorded them in defect tracking provided to help developers resolve issues promptly.
- Actively involved in ongoing efforts to improve testing processes, methodologies, and tools, with the goal of increasing efficiency and enhancing quality throughout the development lifecycle.

EDUCATION

University of Michigan - B.S., Computer Science

September 2017 - May 2024

Dearborn, MI

SKILLS

- **Automation tools:** Python, Git, Jenkins, Nexus
 - **Flashing tools:** Fastboot, QFIL, Renesas Flash Programmer, Vector Flash Bootloader.
 - **Diagnostic/Simulation tools:** Vector CANoe/CANalyzer, Vector VT System, Vehicle Spy, CarSim, Wireshark, Simulink.
 - **Code Analysis tools:** Coverity
 - **Defect Management tools:** Jira, RQM, IBM Rational DOORS, Bugzilla/Buganizer
-

12.0 Appendix D - Project Plan & Logbook

1. Project Overview

- ❖ Project Name: FutureScopeTech Website Enhancement
- ❖ Project Goals: Revamp the online presence of local businesses through a modern, interactive website featuring advanced booking and payment systems.
- ❖ Technologies Used: React (Next.js), Tailwind CSS, Stripe, Calendly, Vercel, GitHub, Google Analytics, Sanity.io, and Hygraph.

2. Project Timeline and Milestones

- ❖ Initiation: Early September 2023
 - Project kickoff meeting to discuss project objectives, stakeholder expectations, and gather initial requirements.
 - Stakeholder interviews to refine goals and gather more detailed system requirements.
 - Completion and approval of the project scope document.
- ❖ Planning: Mid-September 2023
 - Development of the comprehensive project plan including detailed timeline, resource allocation, and budget.
 - Risk management planning, identifying potential project risks and devising mitigation strategies.
 - Finalizing the technology stack and tool integrations (GitHub for version control, Vercel for deployment).
- ❖ Execution: Late September 2023 - March 2024
 - Development sprints, each lasting two weeks, with the following major focuses:
 - Sprint 1-2: Setup of project environment, initial prototypes of website layout using Next.js and Tailwind CSS.
 - Sprint 3-6: Integration of Calendly for appointment scheduling and Stripe for payment processing.
 - Sprint 7-12: Implementation of dynamic content management using Sanity.io and Hygraph; rigorous testing and refinement.
 - Sprint 13-14: Final user interface improvements, security enhancements, and preparation for launch.
 - Bi-weekly sprint review meetings to discuss progress, challenges, and adjust plans as necessary.
 - Mid-project review in December to evaluate project status against milestones and adjust the project trajectory as required.
- ❖ Monitoring & Control: Ongoing throughout the project
 - Regular project status updates and adjustments based on stakeholder feedback and technical challenges.
 - Use of Google Analytics to monitor user engagement during the beta release phase and refine user experience based on real-time data.

- ❖ Closure: April 12, 2024
 - Final project review and closure meeting to ensure all project deliverables are met and stakeholder satisfaction is achieved.
 - Deployment of the final version of the website.
- ❖ Project documentation and handover to the client's technical team for ongoing maintenance.
 - Reflection session with the project team to discuss lessons learned and document insights for future projects.

3. Logbook Entries

- ❖ Weekly Entries:
 - Every week, document key activities, achievements, and challenges faced.
 - Record decisions made during the team meetings and their rationales.
 - Note any changes in scope or project direction approved by stakeholders.
- ❖ Special Entries:
 - Document significant breakthroughs or setbacks in detail.
 - Entry for each milestone reached to evaluate its completion against the expected outcomes.

Project Estimates

This portion of the document provides cost, effort and time estimates for the project using various estimation techniques, which will be elaborated in the appropriate section.

2.1 Historical data used for estimates

Describes the historical data that is relevant to the estimates presented.

Job Function : Application Development

Experience : Five years or less

Company Size : Below 20 Employees

Low US: \$60,000

Median US: \$70,000

High US: \$80,000

\$70,000/12= \$5,833 per month

Estimation techniques applied and results

A description of each estimation technique and the resultant estimates are presented here.

Estimation technique m

- Process - based estimation

Estimation technique *process based*

- | | |
|-------------------------------|-----|
| ● Client Schedule Appointment | CSA |
| ● Client Payments | PAY |
| ● Dynamic Content | DYC |
| ● Custom Invoices | CI |
| ● Admin Manage Payments | AMP |

Activity	Cust. Comm.	Planning	Risk Analysis	Engineering	Engineering	Constructi on Release	Constructi on Release	Cust. Eval.	Totals
Task				Analysis	Design	Code	Test		
Function									
CSA	0.3	0.1	0.1	0.8	1	0.8	0.5	0.1	3.7
CP	0.2	0.1	0.4	1.5	1.2	1.4	1	0.1	5.9
DC	0.1	0.1	0.1	0.9	0.7	1	0.5	0.1	3.5
CI	0.3	0.1	0.3	1.1	0.8	0.5	0.8	0.1	4
AMP	0.2	0.1	0.2	0.4	0.6	0.7	0.4	0.1	2.7
Total	1.1	0.5	1.1	4.7	4.3	4.4	3.2	0.5	19.8
Effort %	0.05555555556	0.02525252525	0.05555555556	0.2373737374	0.2171717172	0.2222222222	0.1616161616	0.02525252525	1

- Based on the median salary of an engineer, using process-based techniques we estimate that it will take 19.8 in person months with a total cost of $19.8 * \$5,833$ which is approximately \$115,493.4 which is about \$115,000

Estimate for technique m

Step 1: You have to compute the count-total which will be used to define the complexity of a project. You will do that by completing the table below:

Information Domain Values							
Measurement Parameter	Count		Simple	Average	Complex		Total
Number of user inputs	12	X	3	4	6	=	48.00
Number of user outputs	6	X	4	5	7	=	30.00
Number of user inquiries	4	X	3	4	6	=	16.00
Number of files	0	X	7	10	15	=	.0
Number of external interfaces	3	X	5	7	10	=	21.00
Count=Total							115.00

Count Total

Step 2: You have to find the complexity adjustment values based on responses to the questions below:

Complexity Weighting Factors						
// heading of the second table Rate each factor on a scale of 0 to 5:						
(0 = No influence, 1 = Incidental, 2 = Moderate, 3 = Average, 4 = Significant, 5 = Essential):						
Question	0	1	2	3	4	5
1. Does the system require reliable backup and recovery?	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2. Are data communications required?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
3. Are there distributed processing functions?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4. Is performance critical?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
5. Will the system run in an existing, heavily utilized operational environment?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6. Does the system require on-line data entry?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
8. Are the master file updated on-line?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
9. Are the inputs, outputs, files, or inquiries complex?	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
10. Is the internal processing complex?	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
11. In the code designed to be reusable?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
12. Are conversion and installation included in the design?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
13. Is the system designed for multiple installations in different organizations?	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
14. Is the application designed to facilitate change and ease of use by the user?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Total						
23.00						

Show Total of weighting Factor

The Function Points is:

Step 3: You have to find LOC (Lines of Code), and you do this by choosing a programming language that you will using when developing a project:

Programming Language	LOC/TP (average)	Select
Assembly Language	320	<input type="radio"/>
C	128	<input type="radio"/>
COBOL	105	<input type="radio"/>
Fortran	105	<input type="radio"/>
Pascal	90	<input type="radio"/>
Ada	70	<input type="radio"/>
Object-Oriented Languages	30	<input checked="" type="radio"/>
Fourth Generation Languages (4GLs)	20	<input type="radio"/>
Code Generators	15	<input type="radio"/>
Spreadsheets	6	<input type="radio"/>
Graphical Languages (icons)	4	<input type="radio"/>

LOC/FP: 3036.00

Step 4: Final Step is to select complexity of the software project:

Software Project	a _b	b _b	c _b	d _b	Select
Organic	2.4	1.05	2.5	0.38	<input checked="" type="radio"/>
Semi-detached	3.0	1.12	2.5	0.35	<input type="radio"/>
Embedded	3.6	1.20	2.5	0.32	<input type="radio"/>

Calculate Effort and Duration

Effort (E) = a_b(KLOC)^{b_b} = Duration (D) = c_b(E)^{d_b} =

Reset Data

Putting the LOC estimate into the cocomo calculator

YOUR BASIC COCOMO RESULTS!!								
MODE	"A" variable	"B" variable	"C" variable	"D" variable	KLOC	EFFORT, (in person/months)	DURATION, (in months)	STAFFING, (recommended)
embedded	3.6	1.2	2.5	0.32	3.036	13.647861966911455	5.769799045724953	2.365396413073283

Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm,81). The final estimates are determined in the following manner:

effort = a*KLOC^b, in person/months, with KLOC = lines of code, (in the thousands), and:


duration = c*effort^d, finally:

staffing = effort/duration

For further reading, see Boehm, "Software Engineering Economics", (81)

WARNING: If you see "NaN" in any field above, you have entered an **INVALID** value for KLOC!! Hit the "BACK" button on your browser, hit the "RESET" button, and enter a **DECIMAL NUMBER** in the KLOC input text box!

Thank you, and happy software engineering!



- Using the function cost estimation we get 13.6 in person months using 2.4 engineers giving a duration of 5.77 months. Therefore the cost estimate is 2.4 * 5.77 * \$5,833 yields \$80,775.38 which we'll round to \$90,000







Reconciled Estimate

- ❖ The final cost, effort, time (duration) estimate for the project (at this point in time) is presented here.
- ❖ The final cost, effort, time (duration) estimate for the project (at this point in time) is presented here.
 - Process based:
 - Effort: 19.8 in person months
 - Total Cost : \$115,000
 - Function Point based:
 - Effort: 13.6 in person months
 - Duration: 6 months
 - Total Cost: \$90,000
- ❖ We are going to average the estimation methods to get the final estimate because they are relatively close in cost and duration.
- ❖ Estimated cost: $(115000 + 90000) / 2 = \textbf{\$102,500}$
- ❖ Effort = $(19.8 + 13.6) / 2 = \textbf{16.7 in person months}$
- ❖ Using variable c and d from our basic cocomo and the formula $c * \text{effort}^d$
- ❖ Duration = $2.5 * 16.7^{.32} = \textbf{6.2 months}$

Project Resources

- ❖ People, hardware, software, tools, and other resources required to build the software are noted here.
- ❖ People - The 4 group members will all try to distribute work evenly and perform a variety of tasks such as coding, planning, debugging etc. to all get experience and gain knowledge about these types of tasks.
- ❖ Hardware - No physical hardware will be required for this project
- ❖ Software - We will use a variety of open source software such as Next Js, Tailwind, and other javascript libraries. We will also use a CMS system such as Builder.io for dynamic content.
- ❖ Tools - We will use github for source control, Jira for task management, and vercel for CI/CD deployments.

Commit Logs

Add @next/third-parties dependency jweb89 committed 3 months ago · ✓ 1 / 1	1112e29	 
Add Google Analytics for production environment jweb89 committed 3 months ago · ✗ 0 / 1	b6a67a5	 
Add Link component to HeaderLink jweb89 committed 3 months ago · ✓ 1 / 1	5440ba3	 

13.0 Appendix E - Project Demo Notes

1. Introduction:

- ❖ Briefly introduce the project, its goals, and the main technologies used (e.g., Next.js, Tailwind CSS, Stripe, Calendly).
- ❖ Overview of the project scope and objectives.

2. Website Overview:

- ❖ Demonstrate the website's layout and design.
- ❖ Navigate through the home page, about page, pricing page, and contact page.
- ❖ Highlight responsive design and usability features.

3. CMS and Content Management:

- ❖ Show the Builder.io dashboard and demonstrate how content is managed and published.
- ❖ Edit a sample page to show real-time updates on the website.

4. Appointment Scheduling with Calendly:

- ❖ Demonstrate how to schedule an appointment using the integrated Calendly widget.
- ❖ Show the process from both the client and admin perspectives.
- ❖ Explain how appointments are managed and synchronized to avoid conflicts.

5. Payment Integration with Stripe:

- ❖ Walk through the payment process on the website.
- ❖ Show different payment options (one-time, subscription, custom invoicing).
- ❖ Demonstrate the backend processing of payments and how transactions are logged.

6. Dynamic Content with Sanity.io and Hygraph:

- ❖ Explain how dynamic content is handled using Sanity.io and Hygraph.
- ❖ Show a live example of content being updated from the CMS and reflected on the website.

7. Analytics with Google Analytics:

- ❖ Present the Google Analytics dashboard.
- ❖ Discuss key metrics tracked (e.g., page views, session duration, user demographics).
- ❖ Explain how this data informs business decisions.

8. Deployment with Vercel:

- ❖ Explain the CI/CD process using GitHub and Vercel.
- ❖ Show how updates are pushed to the live site through Vercel.

9. Q&A:

- ❖ Open the floor for questions.
- ❖ Provide detailed answers to technical and user experience questions.

10. Conclusion:

- ❖ Summarize the key features and benefits of the project.
- ❖ Discuss future enhancements or potential additions.

14.0 Appendix F - Final Presentation Slides

❖ Final Project Demo Presentation Slides

- <https://docs.google.com/presentation/d/1arQ-k3HZ8zGekxGPp-LeHGfeBoNFBYUA2O-WjIzi0kY/edit?usp=sharing>

❖ Team Post Mortem

- https://docs.google.com/presentation/d/1FNPdE1riHIlgQw-lArHjoKIPQuqZO_HcxMDsG1xwaEO/edit?usp=sharing