Project Name: Pelakii Requirements

Team Name: Insulin Intelligence

Team Members:

Raheel Siddiqui : Project Manager
Lukas Hu : Lead Front-End Developer
Tanis Sarbatananda : Team Coordinator + Developer
Vimala Venkata Machiraju : UI/UX Designer + Front-End Developer
Anudeep Hegde : AI/ML and Data Integration Specialist

****Project Updated****

We redirect our project's plan due to on-going and unique circumstances with our sponsor. This has led to some features being removed from the project

Date: 5/17/2024

Table of Contents

Section 1: Requirements

Personas/Scenarios

Persona 1 - ‘Diabetic’ (Diabetes Patient)

Name: Jamie Anderson

Goals
Jamie's goal is to maintain her blood sugar levels and thus wants to buy groceries that can help her achieve her goal.

Needs
Jamie needs to see the ingredients list of each grocery item she is purchasing to ensure that she doesn't consume any form of sugar or sweeteners.

Traits
Age: 34
Occupation: Consultant
Pre-Existing conditions: Type 2 Diabetes

Tech Saviness
Jamie is able to navigate through technology, particularly e-commerce sites with no difficulties. She uses her laptop and mobile phone to access most websites

Scenario
Jamie learnt about Pelakii from one of her coworkers. She decides to check out the Pelakii website and after looking through the site, she decides to look at the list of products she can purchase. She looks through the ingredients list of each grocery item to make sure that there is no sugar, and adds them to a box. She then purchases her first monthly box. is then suggested a monthly box of snacks.

Pains
While shopping for groceries, it's quite annoying for Jamie to look through the ingredients list of every single item she wants to buy to ensure that there are no sugars or sweeteners.

Persona 2 - ‘Blind man’ (Visually Impaired Patient)

Name: Don E. Franklin

Traits
Age: 54
Occupation: Sound Engineer
Pre-Existing conditions: Blindness (Cataracs), Type 2 Diabetes

Tech Saviness
Don has average capability when it comes to technology, but due to his vision loss he relies on assistive technology.

Goals
Don aims to find a simple place to discover and purchase healthy food items. He wants to make informed eating decisions to mitigate his diabetes.

Needs
Don’s blindness means that he needs the app to be accessible. He needs to feature compatibility with screen readers and ideally voice input.

Scenario
Don hears about Pelakii from a friend and decides to check it out. He starts on the landing page by engaging the conversational AI. Since he identifies himself as blind he enables voice input. The AI informs him about his particular form of diabetes where he is then suggested a monthly box of snacks. Ecstatic, he decides to subscribe and get his first box.

Pains
Since type 2 diabetes is really just a multitude of types of diabetes no 1 suggestion of health food services can be tailored to him. He needs a system that can easily suggest what he needs.

A lot of websites on the internet have little or no accessibility features. Don is frustrated because this makes it difficult for him to browse the web.

Persona 3 - Suffering from dairy' (A customer allergic to dairy products)

Name: Hammy Narak

Traits
Age: 19
Occupation: College Student
Pre-Existing conditions: Allergic to dairy products

Tech Saviness
Hammy is passionate about music, especially when it comes to singing Thai songs. She has a lot of experiences with using her phone to do recording and uploading music contents on various social media platforms.

Goals
Hammy Narak's desire is to be on top of her diet. However, she has allergies to foods that contain dairy products and she wants to avoid them at all costs.

Needs
Hammy wants to be able to use a filter search feature where she can only see results of her search items without dairy ingredients.

Scenario
Hammy does not want to go outside for grocery shopping because she is feeling under the weather. She remembers an application called Pelakii that focuses on assisting to select foods with appropriate nutritions. Hammy decides to download the application and place online orders that contain no dairy ingredients.

Pains
Hammy is unhappy with her diet choices because she read online articles on having dairy allergies and she found out that reactions can range from mild to severe, in some cases, they can indeed be life-threatening if she is not careful.

She is tired of shopping for food online and has to manually check the nutrition label if they contain dairy products.

Persona 4 - ‘Gluten-aches’ (A customer with gluten intolerance)

<table>
<tr><td colspan="3">Name: Lucas Bennett</td></tr>
<tr><td> 

<u>Traits</u>
Age: 32
Occupation: High School Teacher
Pre-Existing conditions: Intolerant to products containing gluten

<u>Tech Savviness</u>
He has advanced knowledge of legacy technology and smart devices. He has some knowledge of emerging technology, as his students expose him to new tech.</td><td><u>Goals</u>
Lucas’s desire is to be on top of his diet. His intolerance to gluten means that he can consume items with gluten, but he prefers to avoid it because it will make him groggy and give him digestive issues.

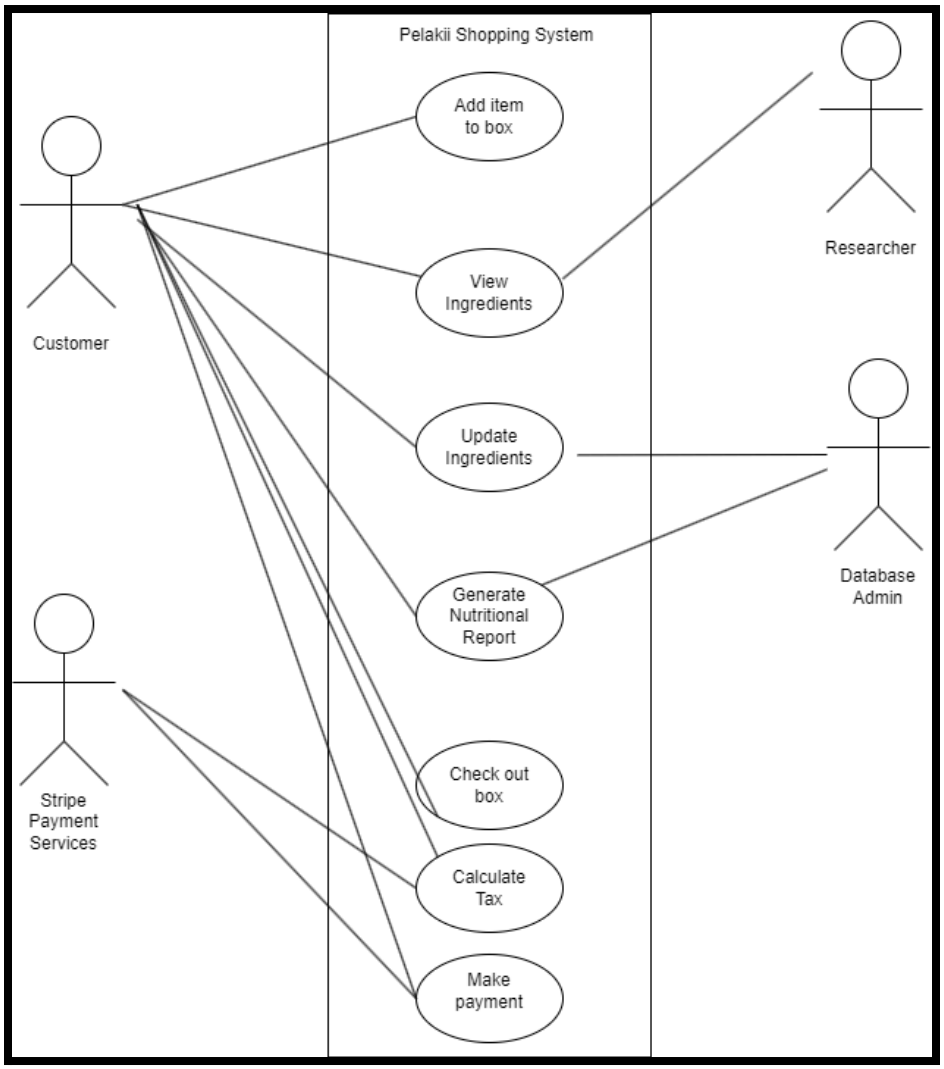
<u>Scenario</u>
Lucas has a busy schedule and his school is relatively out of the way from grocery and convenience stores. She remembers an application called Pelakii that focuses on assisting in selecting foods with appropriate nutrition. Lucas decides to use the application and orders a monthly box that has gluten-free items delivered to him bi-monthly.</td><td><u>Needs</u>
Lucas wants to be able to use a filter search feature where he can only see the results of her search items with low or no gluten content. Using the ingredient and nutrition lists to further inform his decisions.

<u>Pains</u>
Lucas is unhappy with his diet choices because he is suffering from constant stomach aches, headaches, and tiredness.

He is too busy to shop in-person for food, he wants food delivered to him that fits his needs. He wants to remove the guesswork out of snack shopping.</td></tr>
</table>

Functional Requirements

(**Updated**)Use Case Diagram



(**Updated**) User requirements (in use case description format)

Use Case Name	Check Out Box
Author	Vimala Machiraju
Priority	High
Description	Customer must be able to check out the box after adding desired items
Primary Actor	Customer
Secondary Actor(s)	N/A
Preconditions	Users must be logged into Pelakii. Users must have items added in the box.
Success End Condition	User successfully checks out the box and proceeds to make payment.
Failed End Condition	User is unable to check out the box.
Trigger	User adds items to the box and clicks on “check out”.
Basic Flow (Main Success Scenario)	1. User clicks the “checkout” button 2. User views all the items in the box, along with the ingredients list, the price of each item, and total cost. 3. User has the choice to add or remove an item. 4. User checks out the item and proceeds to make payment.
Alternative Flows	1. User clicks the “checkout” button 2. User views all the items in the box, along with the ingredients list, the price of each item, and total cost. 3. User decides to add another item so they go back to select items to add to their box.

Exception Flows	N/A
Relationship to Other Use Cases	Relates to ‘Add Item to Box’ since items must be added in order to be checked out.

Use Case Name	Make Payment
Author	Vimala Machiraju
Priority	High
Description	User must be able to make payment after checking out their box
Primary Actor	Customer
Secondary Actor(s)	Payment Services
Preconditions	Users must be logged into Pelakii. Users must have items added in the box. Users must have checked out the box
Success End Condition	User successfully makes the payment and gets a receipt
Failed End Condition	User is unable to make payment.
Trigger	User checks out the box with their added items.
Basic Flow (Main Success Scenario)	1. User checks out box 2. User enters shipping address 3. User enters payment information (credit card number, etc) 4. User confirms order 5. System sends payment for authorization 6. Order is confirmed and system generates a receipt of the order
Alternative Flows	N/A
Exception Flows	1. User checks out box

	2. User enters shipping address 3. User enters payment information (credit card number, etc) 4. User confirms order 5. System sends payment for authorization 6. Payment gets declined
Relationship to Other Use Cases	Relates to ‘Check Out Box’ since payment can only be made after the user has successfully checked out the box.

Use Case Name	Calculate Tax
Author	Raheel Siddiqui
Priority	High
Description	User must be able to calculate tax before making a payment
Primary Actor	Stripe
Secondary Actor(s)	Customer
Preconditions	Users must be logged into Pelakii. Users must have items added in the box. Users must have checked out the box User must be in process of making a payment
Success End Condition	User successfully makes the payment with the correct tax
Failed End Condition	User is unable to get tax percent.
Trigger	User continues to checkout for selected box
Basic Flow (Main Success Scenario)	7. User checks out box 8. User enters shipping address 9. User receives tax amount 10. User enters payment information (credit card number, etc) 11. User confirms order 12. System sends payment for authorization 13. Order is confirmed and system generates a receipt of the order
Alternative Flows	N/A
Exception Flows	1. User checks out box

	2. User enters shipping address 3. Tax not calculated
Relationship to Other Use Cases	This relates to 'Make Payment' since adding tax is part of the payment process.

Use Case Name	Interact with Conversational AI
Author	Tanis Sarbatananda
Priority	High
Description	A user interacts with a chatbot feature where it offers personalized recommendation
Primary Actor	Customer
Secondary Actor(s)	researcher/tester
Preconditions	N/A
Success End Condition	A user clicks on "click here to chat with us for personalized guidance" and a new window pops up where the user is able to ask questions.
Failed End Condition	A user clicks on "click here to chat with us for personalized guidance" and nothing pops up, offering no way for the user to ask questions.
Trigger	A user clicking on the "click here to chat with us for personalized guidance" button.
Basic Flow (Main Success Scenario)	1. A user clicks on the "click here to chat with us for personalized guidance" button. 2. A new screen pops up (chat-like feature) where the user is able to go back and forth with the AI to ask for personal recommendations.
Alternative Flows	N/A
Exception Flows	1. A user clicks on the "click here to chat with us for personalized guidance" button. 2. A new screen does not pop up.

Relationship to Other Use Cases	This use case is significant with “adding to box” because the user wants to know which item would be beneficial to their circumstance.

Updated wording from “nutritional report” to “nutritional facts/ingredients” based on sponsor’s feedback

Use Case Name	Generate Nutritional Facts & Ingredients
Author	Anudeep Hegde
Priority	High
Description	Users can generate a report summarizing their list nutritional intake
Primary Actor	Customer
Secondary Actor(s)	Data Analysis System
Preconditions	User has added items to their list
Success End Condition	A report is generated displaying the total intake of calories, proteins, carbs, and fats, along with a calculation of glucose intake.
Failed End Condition	The report cannot be generated due to a lack of data or system error.
Trigger	User selects the ‘Generate Report’ option in the list.
Basic Flow (Main Success Scenario)	1. Users access the reporting feature in the app. 2. User selects ‘Generate Report’ 3. System collects all nutritional data entered by the user into the list 4. System calculates the total intake and compares it with recommended values 5. System presents the report in a user-friendly way 6. User reviews their nutritional report for the list at hand

Alternative Flows	1. User selects ‘Generate Report’ for a previous list using calendar view
Exception Flows	1. If no data is available for the selected list, inform the user and suggest data entry. 2. If the system encounters an error during report generation, notify the user and offer to retry.
Relationship to Other Use Cases	1. Complements ‘Add item to box’ by using the added items data for reports.

(**Updated**) Other functional requirements

- Home Page
  - Brief Introduction Page about the company Pelakii
- About Page (*NEW*)
  - Contents
    - Details and images that convey Pelakii’s details, values, and solutions.
  - Footer
    - located at the bottom of the page, including hyperlinks to email, instagram, and Linkedin
  - Conversational AI Feature
    - is on the right bottom of the About page, users are able to chat with AI to ask questions or any recommendations.
- Shopping Page (*NEW*)
  - A page where user will be able to choose items based on their needs and desired
  - Footer located at the bottom of the page, including hyperlinks to email, instagram, and Linkedin
  -
  -
- Log In
  - Everytime a user login, a user must be able to change their personal information so that their personalized aisle is changed accordingly
  - User must be able to log in to their account with username and password
  - A program must be able to verify that all users have to be at least 21 years of age (or under the supervision of one)
- Pre-Ordering
  - API for all ingredients list must be implemented to display what ingredients each item contain
- Conversational AI
  - AI must be able to update at intervals to log new information
  - AI must avoid providing obviously incorrect suggestion to user (this would depend on user’s personal information and preferences)
  - User must be able to communicate with the AI through GUI

- Discover Box (Purchasing items)
    - User must be able to do manual updates to information and cart, not just conversationally

Non-Functional Requirements

Requirements for each relevant quality, e.g., security, reliability, availability, performance, usability, maintainability, correctness, etc.

1. Security: Customers will have personal information stored on Pelakii such as emails, IDs, and credit card numbers so it is vital that it is protected. Failure to secure users' personal information could result in many lawsuits and the compromise of many peoples' accounts.

2. Scalability: Customers should not be limited by their geographical location to have access to Pelakii's services. The system must be able to be accessible to users across the country.

3. Accessibility: The system must be able to be used by those with disabilities given that they are the target audience. Special consideration must be made as to users who may be blind, low vision, hearing impaired, and/or have mobility restrictions (as it relates to typing and use of input devices). Many users are health-conscious due to their disabilities; the Pelakii web app must be designed around these needs. It is important that visual and hearing disabilities are not an obstacle for customers that have them when it comes to using Pelakii.

4. Reliability: The Pelakii web app must be responsive and functional to present Pelakii's products in the best possible way and to allow for a frustration-free customer experience. Pelakii users will depend on the app on a weekly basis so it is important that the system is consistently working properly. Without reliability, many users may not be able to depend on Pelakii to deliver them food. This web app will be the main place where products are sold so it must be ensured that customers are not lost due to an unreliable or slow application.

5. Usability: Pelakii must be able to be used by the average person so that they may be able to buy and receive products. Failure to create a usable system would prevent users from being able to interact with Pelakii.

6. Conversational: This product's integration with AI means that this product will be designed in the form of a conversation. Using this web app should feel more like a conversation with a nutritionist or doctor. with an AI rather than the standard flow of a store as with many other similar products.

7. Modularity: This web app will be developed further after this team is done working on this project. Therefore there must be consideration made as to what will later be added to the product. We were given a list of functional requirements that are not within the scope of this current project course, therefore we built in the ability to augment the app. This applies to both the back end and front end of the software.

(**Updated**) Project/Process/Development Requirements

- Accessibility
    - Using ADA Guard to handle accessibility features
- Inventory
    - ~~Faire~~
    - Database
    - ~~Nutrition Facts~~
    - Ingredient List
    - ~~CRUD (Create Read Update Delete) operations database~~
- Payment System
    - Stripe Tax~~/Avalara~~
- Shipping
    - B2B Drop Shipping
- Ordering
    - Customized Box
        - It is a standard box, but users should be able to add/delete products
    - One item should be possible to pre-order (standard box)

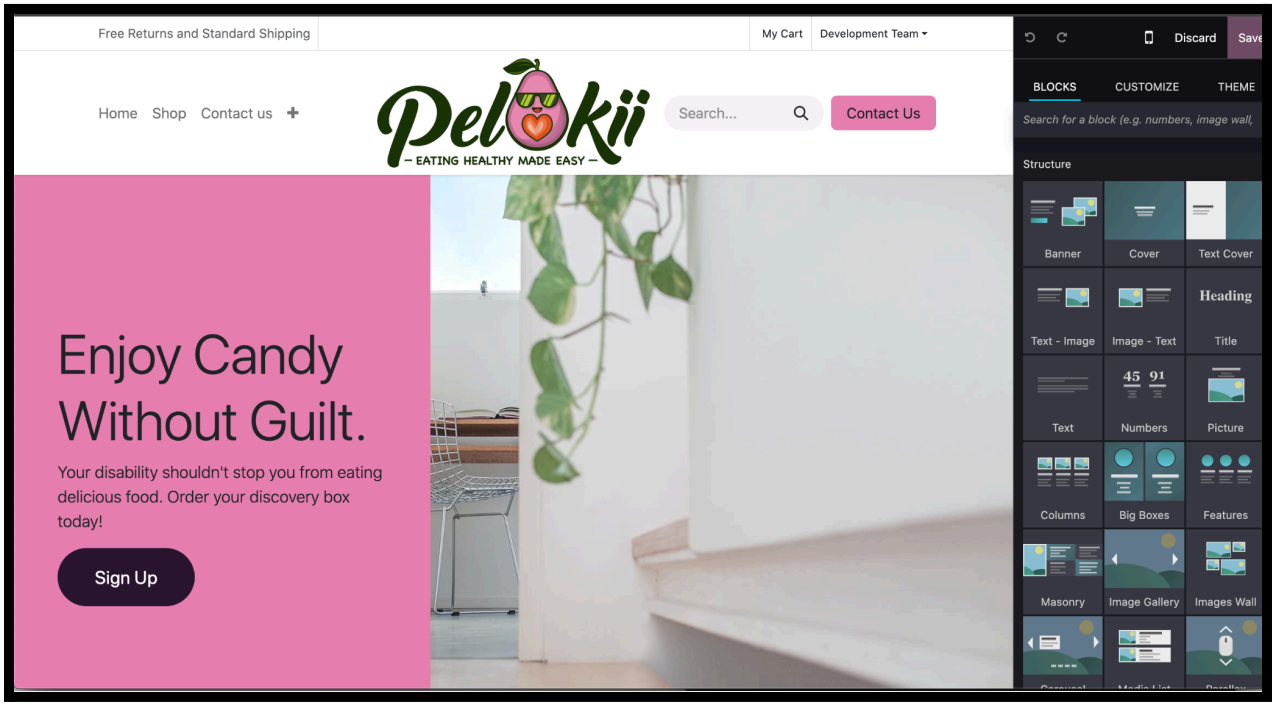
(**Updated**) Constraints and Assumptions

- Constraints
  - ~~Users must be at least 21 years old~~
  - Users must not be able to purchase that they are allergic to, without warning
  - The payment method must be correct and acceptable for the items to be successfully purchased and delivered to processed
  - Limits on the quantity of certain products that can be purchased in a single transaction to prevent misuse or hoarding.
  - Secure authentication mechanisms to verify the identity of users before allowing them to make a purchase.
- Assumptions
  - Features
    - Users have the ability to keep track of what food they each in a day separated into each meal
    - Users can view recipe analysis based on the item they are purchasing
    - There is an option to categorize different types of food
  - Accessibility
    - Accessibility for disabled users
  - Learning & Community Engagement
    - There will be a feature dedicated to educating users on exercises
  - Ordering
    - Boxes (item cart) are customizable (option to add, modify, delete an item)
  - Return
    - Communicate the return and refund policy to users before they make a purchase. Provide options for returning or exchanging items that are damaged, defective, or not as described

Section 2: UI Design

Mockups

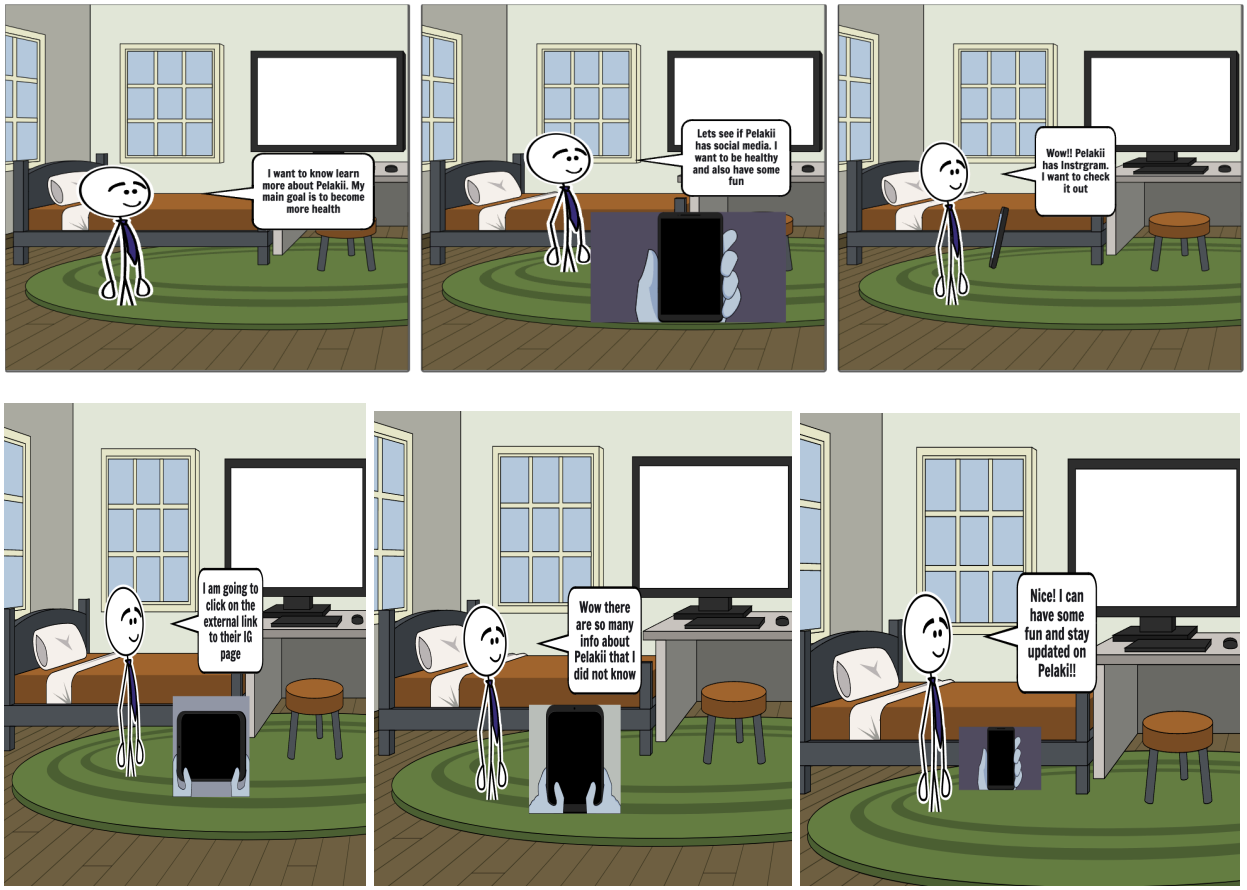
We created the mockups of the Landing Page on Odoo itself since its CMS was fairly easy to use.



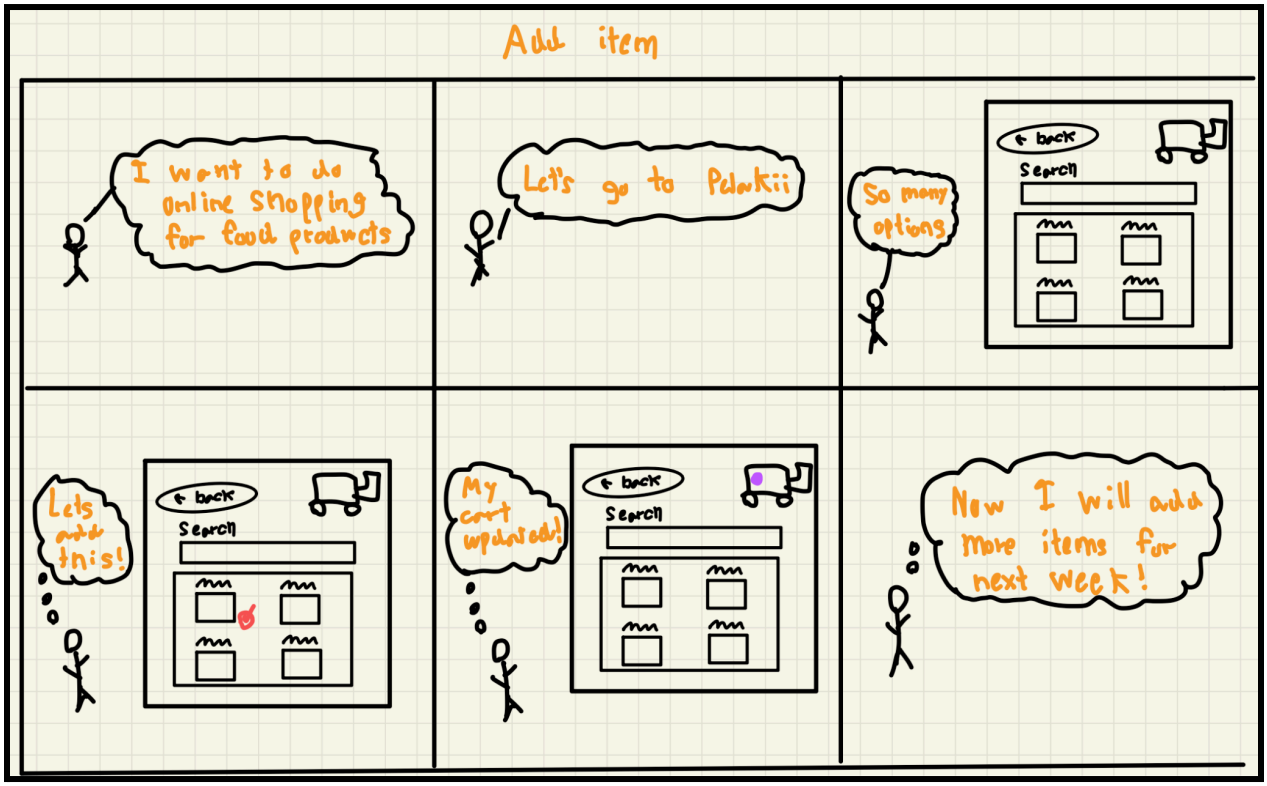

Placeholder
Embark on this journey with us, conquering diabetes your way, without restrictions. Join a community dedicated to vitality and well-being. Be part of something extraordinary by signing up today!
Contact us
https://pelakii.odoo.com/landing-page#myCarousel1714341208019
Structure
Banner
Cover
Text Cover
Text - Image
Image - Text
Heading
Title
Text
45 91
Numbers
Picture
Columns
Big Boxes
Features
Masonry
Image Gallery
Images Wall

(**Updated**) Storyboard

“Click on External Link” Use Case

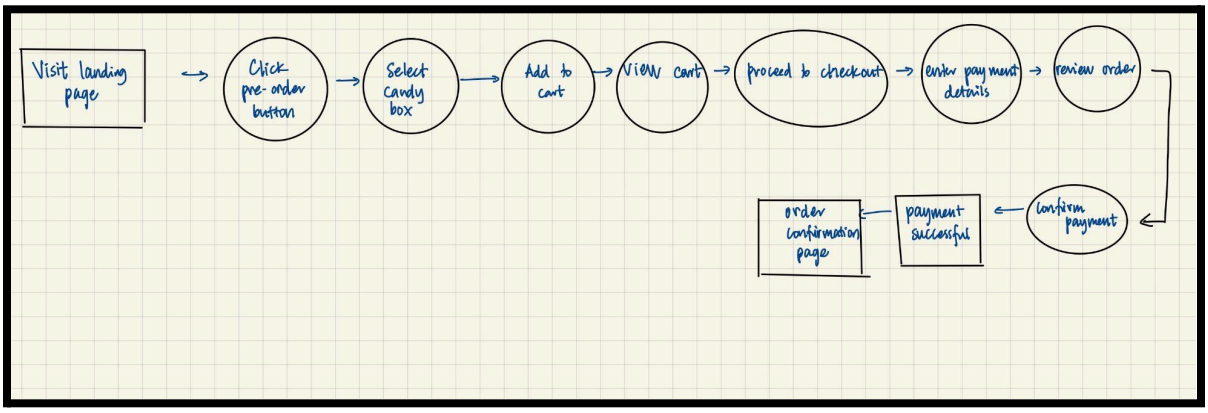


“Add Item” Use Case



User Flow

The user flow below displays how the user will be able to visit the landing page, pre-order a discovery box, and make a payment. It has been approved by the sponsor.



The user flow below displays how the user can add a discovery box, ~~add/delete items from the discovery box~~, and continue with the payment. It has been approved by the sponsor for this version. More iterations will be made as we work towards the next sprint. This user flow is in text format to show what elements will be present on each page.

1. Accessing the Products Page

- Navigate to the website and click on the "Products" tab in the main navigation bar.
- System Response: Load the "Products" page displaying all available products.

2. Viewing the Discovery Box

- Scroll through the list of products or use the search bar to find the "Discovery Box".
- System Response: Display the "Discovery Box" product with an image, brief description.

3. Selecting the Discovery Box

- Click on the "Discovery Box".
- System Response: Load the "Discovery Box" detail page, displaying the following:
  - Image of the Discovery Box
  - Detailed description of the Discovery Box
  - List of items currently included in the Discovery Box
  - Options to add or remove items
  - "Add to Cart" button
  - "Save for Later" button

(We have decided to discard steps on the left were due to changes in our project)

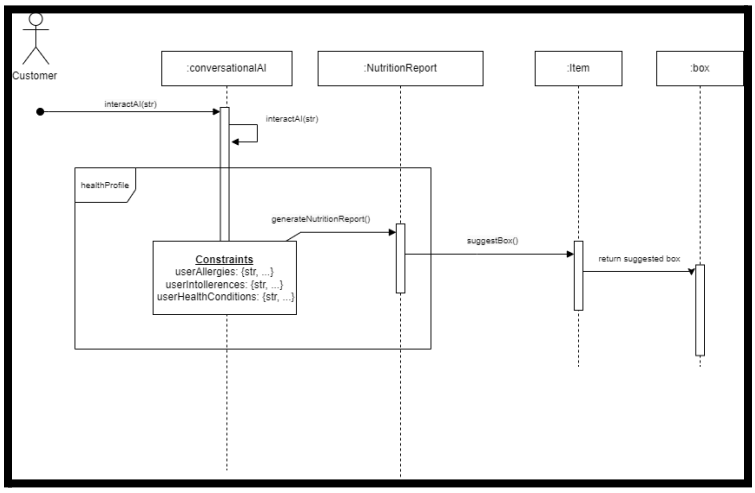
8. Finalizing the Discovery Box

- Review all items in the Discovery Box, adjust quantities if needed, and click "Add to Cart".

Section 3: Software Design

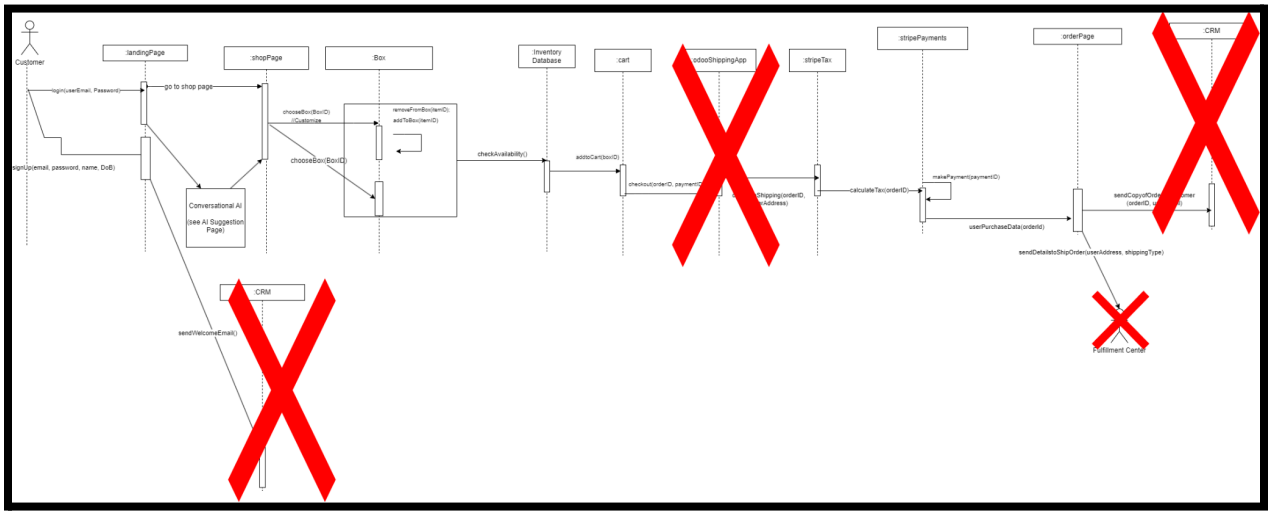
(**Updated**) UML Sequence Diagram

AI Interaction



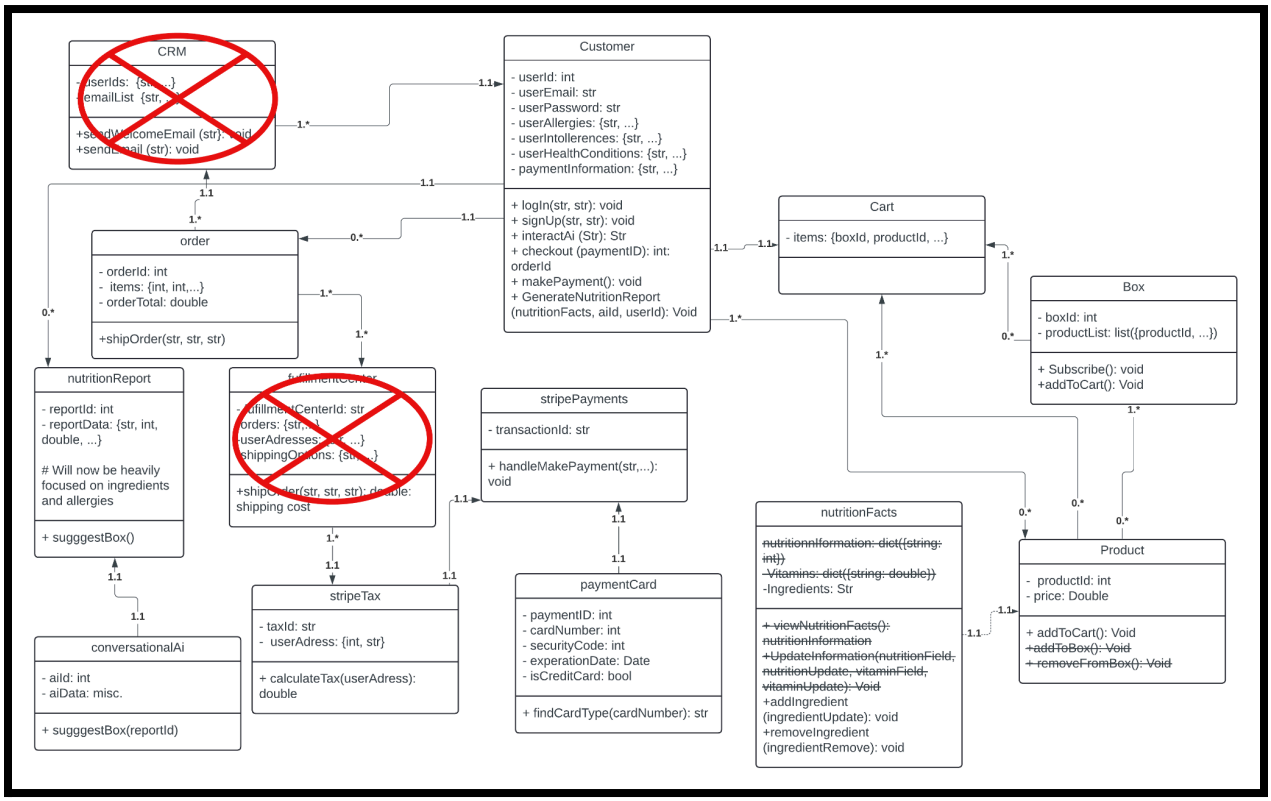
(Note: click on image to view in a higher picture quality)

Purchase



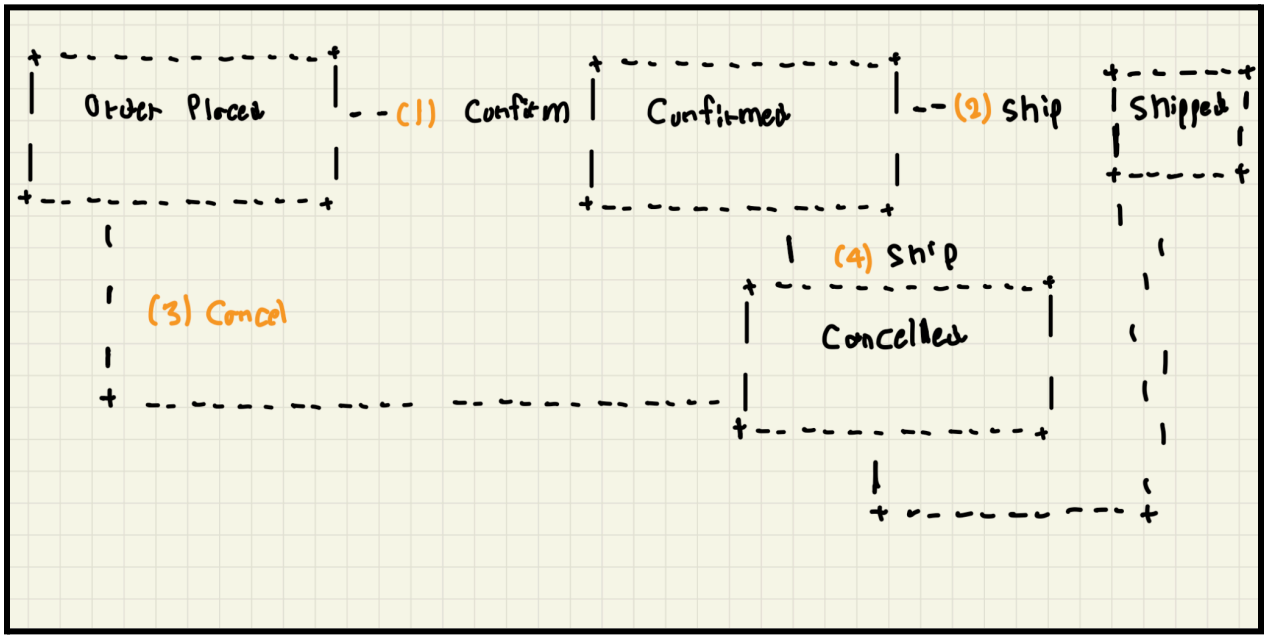
(Note: click on image to view in a higher picture quality)

(**Updated**) UML Class Diagram



One Additional Diagram

State Diagram of Shopping/Shipping Process



Backend Module Description

Functional Modules

- **main.py**
  - The block comment commented-out code is the code for the **address_ui.py** to function as the main method of address input.
  - The code that uses JSON fine for address entry is marked in the code, is set to run using this method
  - *Calc_1* is what actually creates the base calculations for tax and other information gotten from the stripe servers
  - The code following *calc_1* calculates the various amounts not directly given by stripe and allows them to be saved to *transactionDict*.
  - *transactionDict* stores all the relevant information to a transaction in the files
  - Then write *transactionDict* to the **receipt.json** file to be read by the frontend UI
  - *except stripe._error.InvalidRequestError:* Handles what happens when an invalid address is entered, printing an error message to the console.

JSON Files

- **Checkout.json** is structured as:
  - {"address":
    - {"line1": "",
    - "city": "",
    - "state": "",
    - "postal_code": "",
    - "country": ""},
  - "cart":
    - [{"amount": #,
    - "tax_code": "",
    - "reference": ""}],
    - // Is a list because it allows for multiple items in cart
  - "shipping":#}
- **Receipt.json** is structured as:
  - {"item cost": #,
  - "shipping": #,
  - "subtotal": #,
  - "taxPercent": #.##,
  - "taxAmnt": #,
  - "total": #}

Testing Modules

- **stripe_tax.py**

    - *print(stripe.tax.Settings.retrieve()):* prints the current settings from the stripe tax dashboard
    - *stripe.tax.Settings.modify:* Allows for modification of settings, as is set up modifies HQ address and tax code behavior
- **address_ui.py :** The only class in this file creates and runs a UI with text entry fields for every respective field in the address dictionary. The address is stored in address_current list as ["Street Address", "City", "State", "Postal Code", "Country"]. The submit button closes the UI and allows the rest of **Main.py** to continue.
- **createMockAddress.py**
    - *writeToJson():* Writes a dictionary to a JSON file with the address of a WAWA in Orlando.
- **printReceiptConsole.py**
    - The only function in this module takes in a receipt dictionary (see receipt.json) and prints it, some of the calculations for the numbers are made in **main.py**:
        - item cost: 515
          shipping: 300
          _____+
          subtotal: 815
          Tax %: 6.60%
          Tax amnt: 34
          _____+
          total: 849

Frontend Module Description

about.html

- Header
  - **Purpose**: The header contains the navigation menu and the Pelakii logo.
  - **Classes**: `top-nav`, `pelakii-logo`, `top-nav-right`.
  - **Links**: Navigation links to Home, Shop, and About Us pages.

- Intro
  - **Purpose**: Displays an introductory image for the page.
  - **Classes**: `about-images`, `id="about-page-images"`.
  - **Image Source**: `"/Fixing Portfolio/Pelakii-Pictures/Pelakii_Logo.png"`.

- What is Pelakii
  - **Purpose**: Provides a textual introduction about Pelakii.
  - **Classes**: `literal-What-is-Pelakii-text`, `actual-What-is-Pelakii-text`.

- Our Values
  - **Purpose**: Describes the core values of Pelakii.
  - **Classes**: `literal-our-values-text`, `actual-our-values-text`, `about-images`.
  - **Image Source**: `"/Fixing Portfolio/Pelakii-Pictures/integrity.jpg"`.

- Big Picture
  - **Purpose**: Highlights important statistics and issues related to food choices and health.
  - **Classes**: `literal-big-picture-text`, `actual-big-picture-text`, `issue-list`, `about-images`.
  - **Image Source**: `"/Fixing Portfolio/Pelakii-Pictures/Big-Picture.webp"`.

- Issues
  - **Purpose**: Details specific issues faced by the target audience.
  - **Classes**: `literal-issues-text`, `actual-issues-text`, `about-images`.
  - **Image Source**: `"Problem.png"`.

- Solutions
  - **Purpose**: Presents the solutions offered by Pelakii to address the outlined issues.
  - **Classes**: `literal-solutions-text`, `actual-solutions-text`, `about-images`.
  - **Image Source**: `"/Fixing Portfolio/Pelakii-Pictures/Solution1.png"`.

- AI Feature
    - **Purpose**: Introduces the AI feature available on the site for personalized interaction.
    - **Classes**: literal-AI-Feature-text, actual-AI-Feature-text, about-images.
    - **Image Source**: "/Fixing Portfolio/Pelakii-Pictures/AI-Chatbot1.jpeg".

- Footer
    - **Purpose**: Provides contact information and links to social media.
    - **Classes**: `footer-about-container`, `nav`, `nav-item`, `nav-link`, `text-body-secondary`.

- Images

    - Images are stored in the /Fixing Portfolio/Pelakii-Pictures/ directory or other specified paths.
        - Logo: Pelakii_Logo.png
        - Integrity Image: integrity.jpg
        - Big Picture Image: Big-Picture.webp
        - Problem Image: Problem.png
        - Solution Image: Solution1.png
        - AI Chatbot Image: AI-Chatbot1.jpeg

- Alt Text
    - Alt text is provided for each image for accessibility and SEO purposes.

Style.css (line 150 - 338)

1. Images Styling

General Image Styling

Description: This style centers images horizontally within their container.

Details:

- Makes the image a block-level element to ensure it occupies its own line.

- Automatically adjusts left and right margins to center the image within its container.

About Page Image Styling

Description: This style applies specific dimensions and margins to images on the About page.

Details:

- Sets a fixed width for the image to ensure consistent sizing.
- Adds a top margin to provide spacing between the image and any content above it.

2. Text Styling

What is Pelakii Text

Literal Text Styling

Description: This style is applied to the heading "What is Pelakii?" to highlight it.

Details:

- Centers the text horizontally.
- Sets the text color to a specific shade of pink for brand consistency.
- Adds a top margin for spacing from the content above.

Actual Text Styling

Description: This style formats the paragraph text that describes Pelakii.

Details:

- Aligns the text to the left to ensure readability.
- Centers the text container horizontally and provides margins at the top and bottom for spacing.
- Sets a readable font size and limits the maximum width to ensure it doesn’t stretch too far across the page.

Our Values Text

Literal Text Styling

Description: This style is used for the heading "Our Values".

Details:

- Centers the text horizontally.
- Uses the same pink color to maintain a consistent look.
- Adds a top margin to separate it from preceding sections.

Actual Text Styling

Description: This style formats the content explaining Pelakii's values.

Details:

- Left-aligns the text for standard readability.
- Centers the text container horizontally and sets margins for spacing.
- Uses a readable font size and limits the width for a neat appearance.

Big Picture Text

Literal Text Styling

Description: This style is applied to the heading "Big Picture".

Details:

- Centers the text horizontally.
- Maintains the same pink color for visual consistency.
- Adds a top margin to space it from the previous section.

Actual Text Styling

Description: This style formats the content detailing the bigger picture regarding food and health.

Details:

- Aligns the text to the left for clarity.
- Centers the text container and provides margins to space it vertically.
- Uses a consistent font size and width restriction for visual harmony.

Issues Text

Literal Text Styling

Description: This style is applied to the heading "Issues".

Details:

- Centers the text horizontally.
- Uses the same pink color for thematic consistency.
- Adds a top margin for separation from other sections.

Actual Text Styling

Description: This style formats the text describing the issues faced by the target audience.

Details:

- Aligns the text to the left for clear readability.
- Centers the text container horizontally and provides vertical margins for spacing.
- Uses a standard font size and width limitation to maintain a clean layout.

Solutions Text

Literal Text Styling

Description: This style is applied to the heading "Solutions".

Details:

- Centers the text horizontally.
- Uses the pink color to match other headings.
- Adds a top margin for proper separation from adjacent sections.

Actual Text Styling

Description: This style formats the content outlining the solutions provided by Pelakii.

Details:

- Left-aligns the text for standard readability.
- Centers the text container and provides margins for appropriate spacing.
- Maintains a consistent font size and restricts width for a tidy appearance.

AI Feature Text

Literal Text Styling

Description: This style is applied to the heading "AI Feature".

Details:

- Centers the text horizontally.
- Utilizes the same pink color for uniformity.
- Adds a top margin to ensure it is distinct from the previous content.

Actual Text Styling

Description: This style formats the text describing the AI features available in the application.
Details:

- Aligns the text to the left to facilitate easy reading.
- Centers the text container and provides margins for balanced spacing.
- Uses a standard font size and restricts the container's width to keep the layout organized.