Table of Contents

1.0 Preliminary Project Proposal Document

2.0 Proposal Document and Presentation Slides

4.0 Software Development Plan

4.1 Plan Introduction

4.1.1 Project Deliverables

4.2 Project Resources

4.2.1 Hardware Resources

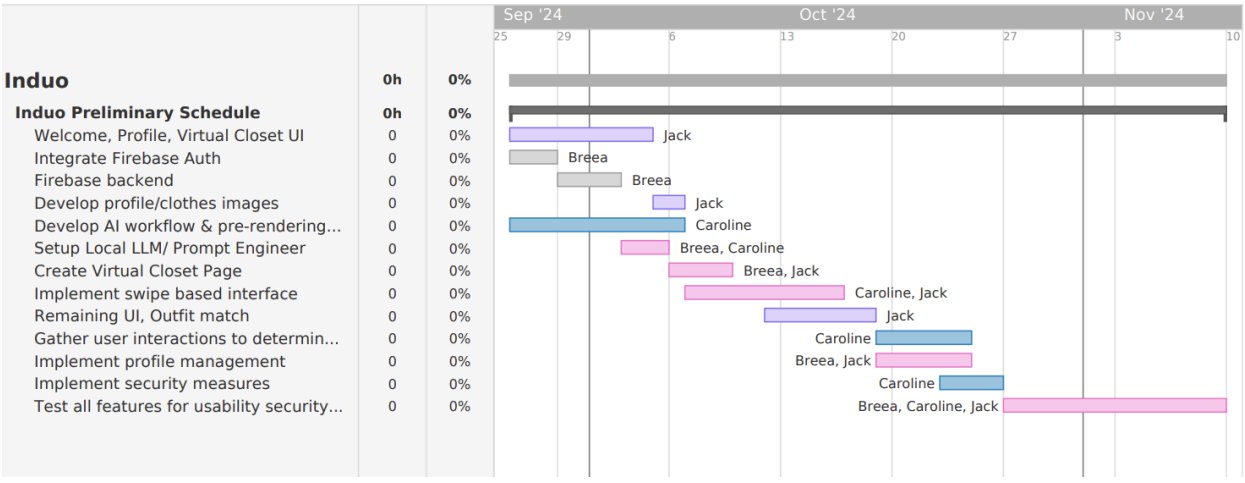
4.2.2 Software Resources

4.3 Project Organization

4.4 Project Schedule

This section provides schedule information for the induō project.

4.4.1 PERT / GANTT Chart



4.4.2 Task / Resource Table

4.4.2 Task / Resource Table

Task	Assigned To	Hardware	Software
Design and implement the user interface (UI) for all pages (Welcome Page, Profile Page, Virtual Closet, Outfit Match).	Jack	MacBooks, iOS devices for testing	Xcode, Swift, GitHub for version control
Integrate Firebase Authentication for user login, third-party login, and profile management.	Breea	MacBooks, iOS devices	Firebase, Swift, Xcode
Develop the functionality for users to upload a profile image as well as clothing images and implement background removal.	Jack	MacBooks, iOS devices	Swift, Xcode, Firebase, image processing libraries
Set up and maintain the Firebase backend for storing profiles, clothing images, and metadata.	Breea	MacBooks	Firebase, Swift, Xcode
Create the Virtual Closet page that displays and manages user clothing items, including search and categorization.	Jack + Breea	MacBooks, iOS devices	Swift, Xcode, Firebase, AI image categorization tools
Setup local LLM and develop prompt engineering for generating detailed descriptions of clothing and outfits, including event-specific details.	Breea + Caroline	MacBooks M3 Pro (for LLM)	Ollama, Firebase, Swift
Develop and integrate the AI workflow for pre-rendering the AI-generated outfit matches using Kolors Virtual Try-On based on the user's virtual closet.	Caroline	MacBooks M3 Pro for LLM hosting, iOS devices	Kolors API, Ollama, LLM, Firebase
Implement the swipe-based interface and display the necessary pre-rendered AI-generated outfits for the clothing item sent to the	Jack + Caroline	MacBooks, iOS devices	Firebase, Swift, Xcode

matching page or for the outfits that match the user's input event prompt.			
Gather user interactions (likes/dislikes) to determine style scores of each outfit, and analyze data for improving AI suggestions.	Caroline	MacBooks, iOS devices	Firebase, Swift
Implement profile management features, including profile picture uploads, edits, and logout functionality.	Jack + Breea	MacBooks, iOS devices	Firebase, Swift, Xcode
Implement security measures like data encryption and ensure compliance with data privacy regulations.	Caroline	MacBooks	Firebase
Test all features for usability, performance, and security, including both manual and automated testing.	Jack + Breea + Caroline	iOS devices, MacBooks	Xcode, testing frameworks (e.g., XCTest)

5.0 Requirements Specification

5.1 Introduction

The induō system is a mobile application designed to assist users in managing their wardrobe and enhancing their fashion choices. It enables users to upload images of their clothing items, generate AI-based outfit recommendations, and visualize these outfits through a virtual try-on interface. By incorporating advanced AI technologies, such as hosting a large language model (LLM) for generating detailed clothing descriptions and integrating with an API-based virtual try-on feature, induō aims to streamline wardrobe management, increase creativity in outfit selection, and encourage sustainable fashion consumption. The application allows users to swipe through AI-generated outfits, upload clothing images with background removal, and store user preferences and wardrobe data securely using Firebase. The system is built as an iOS application using the Swift programming language and integrates Firebase as the backend to store user profiles and clothing metadata.

The system is composed of several key components, including the Graphical User Interface (GUI) for user interaction, AI processing for outfit generation and clothing descriptions including AI Outfit Matching Subsystem and AI Clothing Description Subsystem, and Firebase Backend Subsystem for storing data. Each component has specific functional and performance requirements, which are detailed in the subsequent sections of this document. The high-level diagram below represents the primary system components and their interactions.

The high-level system diagram below outlines the main components of the induō application:

System Components Overview

1. **Graphical User Interface (GUI)**: This includes various user-facing screens where users interact with their wardrobe, upload clothing items, and assess AI-generated outfits.
2. **AI Outfit Matching Subsystem**: The AI-driven subsystem responsible for generating outfits and managing the swipe-based interface for user selections.
3. **AI Clothing Description Subsystem**: Generates detailed descriptions of individual clothing items and complete outfits using the locally hosted LLM.
4. **Firebase Backend Subsystem**: Manages the storage of all user data, including profile information, wardrobe data, and AI-generated outfits, securely in Firebase.

5.2. CSCI Component Breakdown

CSCI induō is composed of the following CSCs:

5.2.1 Graphical User Interface (GUI) CSC

This CSC is responsible for the user-facing portion of the app, where users interact with their wardrobe, upload clothing items, and prompt/assess generated images of themselves in outfits.

- **5.2.1.1 Virtual Closet Main Screen CSU**
  Displays the user's wardrobe and facilitates interaction with individual clothing items.
  - **5.2.1.1.1 WardrobeDisplay module**: Displays all clothing items as scrollable images.
  - **5.2.1.1.2 ClothingItem module**: Represents individual clothing items, enabling selection for matching.
  - **5.2.1.1.3 SearchBar module**: Allows users to input event/weather prompts to find suitable outfits.
- **5.2.1.2 Clothing Item Upload Panel CSU**
  Manages image uploading, background removal, and local storage.
  - **5.2.1.2.1 ImageUpload module**: Handles image selection and upload.
  - **5.2.1.2.2 BackgroundRemoval module**: Removes backgrounds using image processing.
  - **5.2.1.2.3 FirebaseStorage module**: Stores filenames and metadata in Firebase.
- **5.2.1.3 Profile Page CSU**
  Handles user profiles, including uploading a full-body image and managing profile data.
  - **5.2.1.3.1 ProfileInput module**: Manages full-body image uploads.
  - **5.2.1.3.2 ProfileEdit module**: Allows users to update personal information.
- **5.2.1.4 Swipe Screen UI CSU**
  This CSU is responsible for managing the swipe interaction where users like or dislike AI-generated outfits.
  - **5.2.1.4.1 SwipeGestureHandler module**: Manages the gesture input for swipe left (dislike) or swipe right (like) interactions.
  - **5.2.1.4.2 SwipeFeedback module**: Visually highlights swipes, provides feedback for liked/disliked outfits (e.g., animations, color change).

5.2.2 AI Outfit Matching CSC

This CSC handles the AI-generated outfit generation and the "Tinder-style" swiping interface.

- **5.2.2.1 AI-Generated Outfit Rendering CSU**
  Handles API calls to Kolors Virtual Try-On for AI outfit generation.
  - **5.2.2.1.1 APIRequest module**: Sends API requests to Kolors with user data and clothing images.
  - **5.2.2.1.2 ImageProcessing module**: Receives and processes generated outfit images.

- **5.2.2.2 Outfit Matching Swipe Screen CSU**
  Displays pre-rendered AI-generated outfit combinations for users to swipe through.
    - **5.2.2.2.1 OutfitDisplay module**: Collects and displays pre-rendered AI outfits for swiping display.
    - **5.2.2.2.2 StyleScore module**: Updates style scores in database table based on user interactions.

5.2.3 AI Clothing Description CSC

Responsible for generating detailed descriptions of clothing and outfits using Ollama-hosted LLM locally on a Macbook M3 Pro acting as our server.

- **5.2.3.1 Clothing Description Generation CSU**
  Generates descriptions for individual clothing items.
    - **5.2.3.1.1 LLMRequest module**: Sends images to the LLM for clothing descriptions.
    - **5.2.3.1.2 FirebaseSync module**: Syncs descriptions with Firebase.
- **5.2.3.2 Outfit Description Generation CSU**
  Generates descriptions for complete outfits and provides event-specific suggestions.
    - **5.2.3.2.1 LLMOutfitRequest module**: Requests descriptions and event/environment recommendations for full AI-generated outfits.
    - **5.2.3.2.2 EventRecommendation module**: Sends user search bar prompt to LLM with descriptions of all outfits to determine which outfits best align with the user's request.

5.2.4 Firebase Backend CSC

Handles the storage of images, metadata, user profiles, and other application data in Firebase.

- **5.2.4.1 User Profile Management CSU**
  Manages user profiles, including full-body images and wardrobe data.
    - **5.2.4.1.1 FirebaseAuth module**: Handles user authentication.
    - **5.2.4.1.2 ProfileStorage module**: Stores user profiles, including clothing data.
- **5.2.4.2 Wardrobe Database Storage CSU**
  Manages the storage of clothing and outfit image filenames, and associated metadata.
    - **5.2.4.2.1 ClothingImage module**: Stores table of uploaded clothing item image filenames and descriptions.
    - **5.2.4.2.2 OutfitImage module**: Stores table of AI-generated outfit image filenames and style scores.

5.3 Functional Requirements by CSC

5.3.1 Graphical User Interface (GUI)

The GUI subsystem serves as the primary interface through which users interact with the induō application. It enables users to navigate between different features, upload clothing items, view outfit suggestions, and manage their profiles. The GUI will provide intuitive interactions and visual feedback to enhance user experience.

5.3.1.1 The GUI subsystem shall be displayed as an iOS application using the Swift programming language.

5.3.1.2 The subsystem shall consist of four key pages: Welcome Page, Profile Page, Virtual Closet Page, Outfit Match Page

5.3.1.3 The GUI subsystem shall provide clear navigation elements (e.g., buttons, tabs) for moving between pages of the application..

5.3.1.4 The GUI subsystem shall include a Welcome Page with the app name “induō” prominently displayed.

5.3.1.4.1 The subsystem shall display a login page with fields for both username and password inputs.

5.3.1.4.2 The subsystem shall provide a “forgot password” feature, allowing users to reset their password via email.

5.3.1.4.3 The subsystem shall validate user credentials and provide error messages for incorrect input.

5.3.1.4.4 The subsystem shall allow users to log in via third-party services (e.g., Google, Apple).

5.3.1.4.5 The subsystem shall display a welcome message upon successful login.

5.3.1.4.6 The subsystem shall provide an option for users to create a new account.

5.3.1.5 The GUI subsystem shall include a Profile Page that has input for and displays the user’s name, account information, and profile picture.

5.3.1.5.1 The subsystem shall allow users to upload a full-body picture, which will remain displayed on the profile.

5.3.1.5.2 The subsystem shall provide an option for users to edit their profile information (e.g., name, email).

5.3.1.5.3 The GUI subsystem shall provide a logout option from the Profile Page.

5.3.1.6 The GUI subsystem shall display the Virtual Closet Page with a scrollable view of the user's clothing items.

5.3.1.6.1 The GUI subsystem shall react to user gestures, including swiping and tapping, for interaction with clothing items.

5.3.1.6.2 The GUI subsystem shall include input fields for users to enter search queries for specific outfit requests.

5.3.1.6.3 The GUI subsystem shall provide error messages for invalid inputs in search fields.

5.3.1.6.4 The GUI subsystem shall display loading indicators during image uploads and AI processing.

5.3.1.6.5 The subsystem shall provide a button for users to upload images of their clothing items in a pop-up window.

5.3.1.6.6 Each clothing item in the virtual closet shall be clickable, launching a pop-up window with a view of the selected item.

5.3.1.6.6.1 The GUI subsystem shall allow users to view detailed information about each clothing item in the pop-up when clicked.

5.3.1.6.6.2 The GUI subsystem shall include a “Match” button on the pop-up that will take users to the Outfit Match Page where you will swipe suggested outfits using that article of clothing.

5.3.1.6.7 The subsystem shall provide a categorization feature, allowing users to navigate clothing by type (e.g., shirts, pants).

5.3.1.6.8 The subsystem shall include a tab that displays the user's previously saved outfits.

5.3.1.6.9 The GUI subsystem shall display help information or tooltips to guide users in using the app.

5.3.1.6.10

5.3.1.6.11 The GUI subsystem shall provide an option for users to pin their favorite clothing items.

5.3.1.6.12 The GUI subsystem shall allow users to edit or delete clothing items from their wardrobe.

5.3.1.6.13 The GUI subsystem shall provide user prompts to confirm actions such as deletion of items.

5.3.1.7 The GUI subsystem shall include an Outfit Match Page that displays clothing matches based on user preferences and uploaded items.

5.3.1.7.1 The GUI subsystem shall provide feedback animations when users swipe left (dislike) or right (like) on outfits.

5.3.1.7.2 The user shall be able to swipe through various outfit matches generated by the app.

5.3.1.7.3 The AI Outfit Matching subsystem shall display a pop-up of the clothes within the user's virtual closet that can be used to create a similar look for the outfits they swipe right on (like).

5.3.2 AI Outfit Matching Subsystem

The AI Outfit Matching subsystem is responsible for generating outfit combinations from the virtual closet onto the user, using their profile photo, and recommending outfits based on user preferences. This subsystem incorporates a swipe-based interface that allows users to easily express their likes and dislikes regarding the suggested outfits.

5.3.2.1 The AI Outfit Matching subsystem shall generate outfit suggestions based on user-uploaded clothing items, style scores of outfits based on swipe history, and if the user included an event prompt.

5.3.2.2 The AI Outfit Matching subsystem shall allow users to swipe through generated outfit suggestions.

5.3.2.3 The AI Outfit Matching subsystem shall record user preferences for outfits selected as “liked.”

5.3.2.4 The AI Outfit Matching subsystem shall save the user's favorite outfits for future reference.

5.3.2.5 The AI Outfit Matching subsystem shall update the outfit recommendations based on user interactions over time.

5.3.2.6 The AI Outfit Matching subsystem shall integrate weather and event data to provide relevant outfit suggestions.

5.3.2.7 The AI Outfit Matching subsystem shall display style scores for each outfit based on user feedback.

5.3.2.8 The AI Outfit Matching subsystem shall allow users to filter outfit suggestions by clothing type or occasion.

5.3.2.9 The AI Outfit Matching subsystem shall analyze user preferences to improve outfit recommendations.

5.3.3 AI Clothing Description Subsystem

The AI Clothing Description subsystem generates detailed descriptions for individual clothing items and outfits. This subsystem uses a locally hosted large language model to provide users with contextual information and recommendations.

5.3.3.1 The AI Clothing Description subsystem shall generate descriptions for each clothing item uploaded by the user.

5.3.3.2 The AI Clothing Description subsystem shall provide outfit descriptions based on selected clothing items.

5.3.3.3 The AI Clothing Description subsystem shall incorporate event and weather information into outfit descriptions.

5.3.3.4 The AI Clothing Description subsystem shall suggest styling tips based on the generated outfit descriptions.

5.3.3.5 The AI Clothing Description subsystem shall provide feedback to users on the suitability of outfits for specific events.

5.3.3.6 The AI Clothing Description subsystem shall sync generated descriptions with the Firebase backend for user access.

5.3.3.7 The AI Clothing Description subsystem shall support requests for outfit recommendations based on user preferences.

5.3.3.8 The AI Clothing Description subsystem shall utilize user feedback to refine and improve clothing descriptions over time.

5.3.3.9 The AI Clothing Description subsystem shall maintain a history of generated descriptions for user review.

5.3.3.10 The AI Clothing Description subsystem shall allow users to edit or update clothing descriptions manually.

5.3.3.11 The AI Clothing Description subsystem shall provide users with seasonal outfit suggestions.

5.3.4 Firebase Backend Subsystem

The Firebase Backend subsystem is responsible for managing user data, including profile information, wardrobe data, AI-generated outfit data, and local image files' location/name. This subsystem ensures secure storage and retrieval of all application data.

5.3.4.1 The Firebase Backend subsystem shall authenticate user profiles securely using Firebase Authentication.

5.3.4.2 The Firebase Backend subsystem shall store user profiles, including full-body image filename and wardrobe data.

5.3.4.3 The Firebase Backend subsystem shall securely store filenames and metadata for uploaded clothing images.

5.3.4.4 The Firebase Backend subsystem shall sync user data across devices to ensure data consistency.

5.3.4.5 The Firebase Backend subsystem shall allow users to update their profile information as needed.

5.3.4.6 The Firebase Backend subsystem shall manage the retrieval of saved outfits for user access.

5.3.4.7 The Firebase Backend subsystem shall support efficient querying of wardrobe items based on user preferences.

5.3.4.8 The Firebase Backend subsystem shall maintain logs of user interactions with the system for analytics.

5.3.4.9 The Firebase Backend subsystem shall ensure data privacy and compliance with relevant regulations.

5.3.4.10 The Firebase Backend subsystem shall provide backup and recovery options for user data.

5.3.4.11 The Firebase Backend subsystem shall provide data encryption for user information stored in the database.

5.4 Performance Requirements by CSC

5.4.1 Graphical User Interface (GUI)

5.4.1.1 The GUI shall be intuitively designed in order to support seamless user interaction.

5.4.1.2 The application should be responsive and function smoothly on various iOS devices (iPhones and iPads).

5.4.1.3 Image loading and outfit generation should occur within a reasonable timeframe given backend processing capabilities. (e.g., about 30 seconds per outfit).

5.4.1.4 The application should be memory efficient (100-150 MB RAM) and not drain the users battery excessively (5-10 percent per hour).

5.4.1.5 The application should be compatible with a wide range of iOS devices, including both iPhones and iPads, as well as older models (iPhone 8 or later) and different screen sizes.

5.4.2 AI Outfit Matching

5.4.2.1 Generated outfit images should be realistic and visually appealing.

5.4.2.2 The outfit generator should be able to handle a wide range of clothing item types (tops, shirts, pants, skirts, etc.) and styles.

5.4.2.3 The ML model should be adaptable to evolving fashion trends and styles.

5.4.3 AI Image Generation

5.4.3.1 Generated images should be consistent with the user's selected environment and outfit style.

5.4.3.2 The AI image generation tool should be able to produce high-quality images even on mobile devices with limited processing power.

5.4.4 Firebase Backend Subsystem

5.4.4.1 The database (Firebase) should be optimized for efficient data retrieval and storage. Under a reliable network connection, users should receive data to their device in 2-5 seconds.

5.4.4.2 The Firebase backend should have robust security measures in place to protect user data. Built-in Firebase authentication will be used as well as the implementation of granular security rules in order to control who can read/write data.

5.4.4.3 The Firebase backend database shall accommodate up to 10,000 monthly active users for authentication.

5.4.4.4 The Firebase backend database shall allow up to 50,000 reads, 20,000 writes, and 20,000 deletes per day.

5.4.4.5 The Firebase backend will accommodate up to 100 simultaneous connections.

5.4.4.6 Firebase cloud functionality shall accommodate up to 2 million invocations per month.

5.5 Project Environment Requirements

5.5.1 Development/Deployment

5.5.1.1 Hardware:

Category	Requirement
Processor	M2
GPU Memory	Optional
RAM	32GB
System Swap	Required

5.5.1.2 Software:

Category	Requirement
Operating System	macOS Monterey 12
IDE/Compiler	XCode 13.3
LLM (Text Generation)	ClaudeAI
LLM (Image Generation)	Stable Diffusion
Backend	Firebase SDK
Image Masking	Swift Core ML
VC/Repository	Github

5.5.2 Execution

5.5.2.1 Hardware:

Category	Requirement
Device Type	Mobile iOS
Processor	A11 Bionic
RAM	3GB

5.5.2.2 Software:

Category	Requirement
Operating System	iOS 12.4
Network	Active Internet
Offline Functionality	Limited to pre-loaded data.