COMPSCI 4ZP6A/B

CAPSTONE SOFTWARE REQUIREMENTS SPECIFICATION (SRS) + PROJECT PLAN

Team 27

October 10, 2025

Table of Contents

Versions, Roles and Contributions

1. Project Title

Vestia - The Retail Experience Platform (RXM)

2. Version

Version #	Authors	Description	Date
0	All	Created first draft of document	October 10, 2025

3. Team Information

Name	Email Id	Student number
Saagar Racharla	racharls@mcmaster.ca	400457787
Rushit Shah	shahr73@mcmaster.ca	400462623
Manush Patel	patem190@mcmaster.ca	400449099
Mahdi Mohammad	moham52@mcmaster.ca	400370587
Garv Rastogi	rastogig@mcmaster.ca	400471597
Partik Singh Dev	devp2@mcmaster.ca	400478418
Gurmandeep Johal	johalg11@mcmaster.ca	400467941

4. Naming Convention and Terminology

Naming Convention	Terminology
SKU (Stock Keeping Unit)	alphanumeric code assigned to a product
JWT (JSON Web Token)	Securely transmit JSON obj

Naming Convention	Terminology
RBAC (Role-based Access Control)	Limits access based on user roles (appears again under security context).
SLA (Service Level Agreement)	A commitment between a service provider and a client that defines expected service standards (e.g., response time).
CRUD (Create, Read, Update, Delete)	Storage operations.
P95	95th Percentile. A performance metric showing latency or response time within which 95% of requests complete.
E2E (End-to-End)	Refers to testing or systems that cover the full process flow from start to finish.
S3 (Simple Storage Service)	AWS cloud object storage service used as a data lake or storage repository.
PCI DSS (Payment Card Industry Data Security Standard)	A standard ensuring that all companies that process, store, or transmit credit card information maintain a secure environment.
GDPR (General Data Protection Regulation)	EU law governing data privacy and protection.
AWS EC2 GPU (Amazon Web Services Elastic Compute Cloud GPU	Virtual cloud machine for graphics processing for AI model

5. Table of contributions

Group member	Contributions (Sections)
Saagar Racharla	10, 15a. - d. , 16
Rushit Shah	6, 7.2, 8, 12, 14
Manush Patel	9
Mahdi Mohammad	4,15,16
Garv Rastogi	1,2,3,18, formatted the whole document
Partik Singh Dev	17, 13, overall editing and revision
Gurmandeep Johal	7.1, 5

6. The Purpose of the Project

The purpose of the project is to revolutionize the traditional outdated fitting room experience by providing an interactive technology, data-driven insights and a seamless customer-store communication. As a customer this project aims to streamline the shopping experience by providing features such as outfit recommendations based on store stock, cross-selling suggestions, and the ability to request additional sizes or colours directly from the screen. Customers can also check out instantly from the changeroom, log in to see personalized recommendations from past purchases, and use a "mix & match" mode to virtually pair tops, bottoms, and shoes. Additional conveniences include queue management, in-changeroom feedback screens, and the option to save or share outfits with friends or on social media. As a retailer, our project provides key insights that previously used to go undocumented. Data such as heatmaps of tried on items, conversion rates of try-on to checkout, real time customer feedback and more can all be leveraged now to improve their stores efficiency while prioritizing customer satisfaction.

Software Requirements Specification (SRS)

7. The Client, and other Stakeholders

7.1 Primary Stakeholders

Client(s):

- The client in this case would be retail stores seeking to boost their store sales by upselling to their customers.
- They would do so by recommending clothing outfit inspirations to customers as well as making shopping more interactive and easy through the system helping revolutionize the fitting room experience.

7.2 Secondary Stakeholders

Stakeholders:

- Shoppers
  - The end users through the kiosks
  - They receive convenience through being able to request sizes, checkout and get personalizations
  - They get instant price matches and time savings
  - Helps with customers who are also introverted and don't want to go up to actual humans and talk.
- Store staff
  - Oversee the operations and help with reviewing analytics to see what items sell the most and get a new form of a heat map within the store to see which areas are targeted the most
  - Reduced time with dealing with customers, rather get ownership of the products they sell through actionable insights metrics
- Store managers / Executives
  - Review overall analytics of the store
  - Get feedback as well through this to help improve clothing styles as well as reduced operational bottlenecks.
- Technology Teams
  - Responsible for the whole SDLC; integration deployment and maintaining the product

        - Provide updates based on feedback to help constantly keep the product at the cutting edge of the fashion industry ensuring product growth is imminent as well as product benefit for both the stores and their customers.
    - Payment Providers
        - Ensure that they can help integrate their payment services seamlessly into the product
        - Get a commission out of this integration
    - Growth Partners
        - These can be anyone such as Venture Capital firms, social media partners, online product promotions
        - Help the brand grow as well as meet their expectations with the product

8. Project Constraints

8.1 Platform

- It will be a kiosk style web application that can run on lightweight devices such as tablets.
- Hosted through the cloud and there will be no on-prem installs so everything can be monitored through the internet.

8.2 Hardware

- Barcode scanning will be done through a usb scanner which will be connected to the application.
- Payments will be handled through Stripe; app will never store card data
- Inventory will be handled through a single source of truth API so that inventory is always consistent and up-to-date.
- Must be able to handle spotty wifi (latency targets should be P95 < 500ms which means that the latency will be in the 95th percentile for the work load)
- All services will be running in a single region (no cross region servers or services)

8.3 Data, Privacy and Compliance

- PII minimization which means that there will need to be consent flags (for users privacy choices regarding their personal data) for any form of email or phone login
- No sensitive biometrics or data will be captured through the machine
- In terms of the payments compliance, it will be relied upon the payments vendor which should follow the Payment Card Industry Data Security Standard (PCI DSS)
- The system should follow Data Subject Access Request (DSAR) which essentially means that there will be transparency with the customer regarding data storage, the processing of the data and the collection.
- Follow GDPR.

8.4 Security

- Session-scoped JWT's will be used for authentication and authorization.
- RBAC will be in effect; staff only gets a certain level of access, developers only get a certain and similar with customers.
    - Customer gets access to scan items, request sizes, give feedback, check out; they will have user based login for personalization
    - Employees get access to other aspect to be discussed through role based access

- Kiosk handles no point of actions; all the kiosks are to be noted as untrusted and all events will occur on the server side.
- Least privilege principal; each user and the systems are only granted a certain amount of access as much as they need:
    - Kiosk
        - Only allowed to read and create customer sessions which include scanning their items, tailoring recommendations, requesting sizes.
    - Staff consoles or access
        - Can only view information pertaining to that certain store
        - Cannot temper with store analytics
    - Manager Dashboard
        - Can only read the analytics
    - Backend microservices
        - The recommendation engine will only be allowed to read the product catalog and purchase histories, nothing more is allowed
    - Checkout service
        - Will only checkout the customer and cannot modify transactions and can only mark orders as paid.
    - All analytics services will just note down the events logs.
- Rate Limiting, so that customers cannot abuse the system sending a lot of requests breaking it.

8.5 Accessibility and UX

- Utilize the WCAG 2.2AA which are the web content accessibility guidelines helping make digital content more accessible to persons with disability.
- Language will be set to english only.

8.6 Testing

- Full testing cycle; unit → integration, selective E2E to ensure that everything works in a real world scenario.

8.7 Legal

- Will adhere to all store policies
- Will follow country based guidelines (Canada in this case)

9. Functional Requirements

The following functional requirements are ranked by priority from high to low.

P0 - Core MVP (Changeroom Session + Item Identification)

- User Story:
    - Customer enters fitting room area, screen turns on and they scan items they would like to try. They are assigned a fitting room and the screen inside the fitting room turns greeting the customer with all data on items scanned. User is also prompted to sign in or create an account as well as continue as guest.
- Frontend:
    - A live tray is portrayed with all scanned items including the SKU, colour, size. User is prompted to sign into account, create an account, or proceed as guest.
- Backend:

    - A device integration service will scan the barcode on the items and maps it to SKU storing it in the session. A session_id is generated once finished scanning and is assigned a room_id.
    - User Service pulls up records of all prior sessions of the users with all items tried, purchases, feedback given etc.
    - Catalogue Service will map each SKU to product metadata(brand, product, colour, size)
- Data:
    - Session Table
        - {session_id, room_id, start_time, end_time, customer_id, {list of items(SKU, brand, colour, size}}
    - Data for all sessions are stored for a particular user under their customer_id

P0 - Request Additional Size/Colour

- User Story:
    - Customer inside the fitting room tried a black small polo and realized it is too tight and would like to request a size larger. They can see a black polo size M with the same SKU is available in store, they can click “request different size/colour” and select size M from the screen and send the request and the staff is alerted.
- Frontend:
    - In the live tray of all products, there exists an item details panel with an option to request different size/colour. Once selected, shows available sizes for the product at the location. Send Request button follows selection.
    - Once associate brings item, screen displays “requested item has arrived”
- Backend:
    - Request Fulfilment Queue Service creates a request_id and enters it into system
    - Communication Protocol pushes request staff - mobile/web
    - Update Inventory db adding product in fitting room
    - Associate marks request as fulfilled, update the system and notification service displays “request item has arrived” message on screen
- Data:
    - Inventory Table(for each SKU), staff_id, session_id, request_id, sku_requested, status, sla_timestamps,
    - Status => Queued, Picked up, Delivered/Cancelled

P0 - User Feedback Analytics

- User Story:
    - Whether use purchases item or leaves it behind, they are asked for feedback on their fitting room experience and product experience.
- Frontend:
    - User is able to click End Session when they are done trying on their products. User selects what they like and what they don't, and feedback on each product and the experience is prompted on the screen. If the user doesn’t fill it out, an email is also sent for user to fill out through mobile.
- Backend:
    - Purchase Service updates db on what the user chose to take and what to leave back, updating the SKU # has been converted from a trial to a purchase.

P1 - Core Retail Analytics

- User Story:
  - Store managers are able to view aggregate analytics on fittings rooms activity including, most items tried, trial-purchase conversion rate, session durations, request fulfilment timings etc. Store Managers are also able to view queue management analytics on the admins screen with empty rooms, occupied rooms, items in each room, duration etc. All of these metrics are key to improving the entire fitting room experience
- Frontend:
  - Admin Dashboard visualizes:
    - Trial-Purchase Ratio
    - Top tried items
    - Most requested sizes/colours
    - Average Session Duration
    - Average Time taken to fulfil request
    - Average Items Tried per Session
    - Repeat Visit Rate
    - Guest vs Signed in Ratio
    - Queue Management Metrics
- Backend:
  - Analytics Service aggregates data from Session Table, Inventory and Purchase Log and presents all information on frontend.
  - Dashboard_API to transfer and display data on admin portal
- Data:
  - Admin_Analytics: {store_id, total_sessions, avg_duration, top_sku, top_size, conversion_rate, request_fulfilment_avg_time, repeat_visit_rate, guest_vs_signed_in_ratio, avg_items_per_session}

P1 - Customer Profile Personalization + Share Outfits

- User Story:
  - Returning customers see personalized greetings based on past purchases, preferences etc. and allow the customer to provide feedback on their past purchase.
  - User can save outfit on their account and share it with others using a qr code/link
- Frontend:
  - Greetings message:
  - “Welcome Back, Peter! Hope you loved your slim-fit denim from last time”
  - “Please leave your feedback on your slim-fit denim”
  - Recommendations are now going to based on set preferences and past purchases
  - Save outfit option for mix-match outfits(QR code, link)
- Backend:
  - Customer Profile Service aggregates past purchases, sessions, feedback
  - Personalization API picks out data from customer profile and adds it to display
- Data:
  - Customer_Profile: {customer_id, preferred_sizes, preferred_colours, brand_affinity, avg_session_duration, purchase_rate, feedback_summary}

P1 - Rule-Based Outfit Recommendation Engine (Non-ML Baseline)

- User Story:
  - A returning customer brings a blue slim-fit shirt into the changeroom. Based on their past purchases and preferences, the system recommends compatible pants and shoes (e.g., neutral chinos, dark jeans, white sneakers) that are available in store. Recommendations are generated using a rule-based scoring algorithm, not hard-coded lists.
- Frontend:
  - On the fitting room screen, under each scanned item, a "Recommended to pair with" panel shows 2–3 suggested items per category (e.g., bottoms, shoes).
  - Each recommendation card displays: product image, brand, price, colour, and an indicator of stock availability.
  - Tapping a recommendation allows the user to request that item from an associate using the existing "Request Additional Size/Colour" style workflow.
- Backend:
  - A dedicated **Recommendation Service** exposes an endpoint such as:

1. Fetches the **base item's attributes** from the catalogue (category, colour family, style tags, brand, price range).
2. Builds or reads a **customer preference profile** from past sessions and purchases (e.g., preferred colours, brands, average price, preferred categories).
3. Filters the catalogue down to **in-stock, compatible categories** (e.g., base = "top" → candidates = "bottoms" and "shoes").
4. Applies a **rule-based scoring function** that combines:
   - colour compatibility between base item and candidate;
   - brand affinity (how often the customer buys that brand);
   - price-range closeness to the customer's usual spend
   - optional style overlap between items (e.g., "slim-fit", "casual").
5. Returns the **top K items per category**, sorted by score.
   - This baseline is **deterministic and explainable** and can later be replaced or extended by a learned ML model without changing the API.

- Data:
  - Customer_Profile: {customer_id, preferred_sizes, preferred_colours, brand_affinity, avg_session_duration, purchase_rate, feedback_summary}

P2 - Outfit Recommendations + Mix and Match

- User Story:
  - A customer wants to build a full outfit inside the changeroom. They start with a top they scanned, open Mix & Match mode, and see compatible bottoms and shoes suggested based on their preferences and store inventory. They can tap different combinations to preview outfits and request any combination to be brought into the room.
- Frontend:
  - A dedicated **"Mix & Match"** screen with three slots:
    - Top / Outerwear
    - Bottom
    - Shoes / Accessories

    - Each slot opens a **scrollable carousel** of recommended items in that category, populated by the Rule-Based Recommendation Engine (P1).
    - As the customer selects items in each slot, the UI updates a simple **outfit preview** layout showing the chosen top + bottom + shoes.
    - A **“Request this outfit”** button sends item requests for all selected SKUs to the staff dashboard.
    - Users can also **save** a Mix & Match outfit to their profile, which integrates with the “Save & Share Outfits” requirement.

- Backend:
  - Mix & Match Mode calls the same `GET /recommendations` endpoint but in a **multi-category** context (e.g., base_sku = top, categories = [“bottom”, “shoes”]).
  - Backend rules enforce:
    - Items in the same outfit must belong to **distinct categories** (no top+top+top).
    - All recommended items must be **in stock** for that store location.
    - Colour combinations must pass a basic **colour compatibility check** using configuration (e.g., avoid strongly clashing colours; favour neutral bottoms with bright tops).
  - When the user taps **“Request this outfit”**, the system:
    - Creates multiple entries in the **Request Fulfilment Queue** (one per SKU).
    - Associates them with a shared `outfit_request_id` so staff can see that these items belong to the same customer request.
    - Updates status changes (Queued → Picked Up → Delivered) that are reflected both in the staff dashboard and the changeroom kiosk.
- Data:
  - MixMatch_Session table (or collection) with structure such as: { outfit_request_id, session_id, customer_id, top_sku, bottom_sku, shoes_sku, created_at }
  - Reuses Recommendation_Log to track which outfits and items were:
    - viewed in Mix & Match,
    - requested from staff,
    - purchased at checkout.
  - This data feeds into future analytics (e.g., “Most popular outfit combinations”, “Conversion rate of Mix & Match recommendations”).

10. Data and Metrics

10.1 AI Recommendation Engine (core ML feature)

- **Data used to train/build**
  - A dataset that provides full complete outfits, with the dataset already split into training, validation and testing
    - This data will be used to build the model that recommends cross sells that go along with their clothes they brought into the changeroom

    - The data must consists of different articles of clothing along with a description that will be used to train the model
- **Links / Clear Plan to obtain or simulate data**
    - Polyvore dataset: https://github.com/xthan/polyvore-dataset?
- **Performance metrics (with goals)**
    - Over 75% accuracy that the top 5 recommendations are compatible with the items they brought into the change room
    - This is relevant the customer will see a large list of relevant items, but we want to focus on the top and ensuring its accuracy

10.2 Mix & Match Mode

- **Data used to train/build**
    - A dataset that has a full store catalog of images organized into categories shirts, pants, shoes, shorts, etc
- **Links / Clear Plan to obtain or simulate data**
    - Clothing dataset: https://github.com/switchablenorms/DeepFashion2
- **Performance metrics (with goals)**
    - The first 3 items of each category are compatible with each other, such as the shirts, pants and shoes
    - This is relevant as customers will most likely focus on the first couple of shirts and pants rather than endlessly scrolling

10.3 Request Additional Sizes/Colours

- **Data used to train/build**
    - The data on every request must pull the real time live inventory of what is currently in the retail store
        - This must include the SKU, size, color, stock quantity, location in the store that will be sent to the store associate
- **Links / Clear Plan to obtain or simulate data**
    - Simulate data by using a sample dataset where the live inventory is constantly being updated with a random function.
- **Performance metrics (with goals)**
    - Over 95% fulfillment rate of total requests and fulfilled requests
    - Under 5s delay in when the request is sent and received by the store associate
    - These metrics are relevant as a low fulfillment rate would create a terrible customer experience and they will be more inclined to just get it themselves.
    - Additionally, if the delay between the request and when the store associate is received is too long, it will create a bottleneck.

10.4 Personalized Login (past-purchase-based recs)

- **Data used to train/build**
    - User profiles, that consists data of all demographics, including age, gender, past purchases, purchase data, amount, etc
    - Be able to save their data for the next session (saving clothes to a wishlist)
- **Links / Clear Plan to obtain or simulate data**
    - User Purchase Behaviour: User Profile Dataset

    - Further data can be simulated based on the Polyvore dataset above by combining both to create a valid purchase behaviour dataset
- **Performance metrics (with goals)**
    - Over a 30% retention rate of when customers log in again within 90days
    - This metric is relevant as it will allow us to see the success of seeing if customers value the personalized login

10.5 Save & Share Outfits

- **Data used to train/build**
    - Saved outfits that consist of the user, timestamp, clothes, and images
    - Share events (SMS link / QR to phone).
- **Links / Clear Plan to obtain or simulate data**
    - Generate a link/QR code of the outfit with links to each product that can be saved to the user profiles.
- **Performance metrics (with goals)**
    - Over a 20% save rate of outfits built and tried on in the changeroom specifically
    - Over a 25% click through rate of the link/QR code within 14 days
    - These metrics are relevant we need to keep track of the engagement and store the data after and outside the changeroom

10.6 Queue Management (Customer Perspective)

- **Data used to train/build**
    - Data that shows queue events that include when they entered, and notifications on how long they have been inside.
- **Links / Clear Plan to obtain or simulate data**
    - Simulate traffic data by creating mock customers who randomly enter changerooms at specific timestamps and randomly come out anytime from 5 to 10 minutes.
- **Performance metrics (with goals)**
    - Over 95% accuracy over all queue events
    - This metric is relevant as anything lower will result in wasted time/changeroom in between customers.

10.7 In-Changeroom Feedback + Store Dashboard

- **Data used to train/build**
    - Dataset of a short feedback survey of sizing, colour, fit, etc
    - Datapoints include abandonment rate, Try out → Checkout %, Heat maps of what people are trying on in the store and more.
- **Links / Clear Plan to obtain or simulate data**
    - Simulate data by randomizing the feedback, each time someone enters the fitting room and has feedback
    - From here we can take the simulate the datapoints for the dashboard
- **Performance metrics (with goals)**
    - Over 50% response rate, and under 5 minutes dashboard update latency
    - These metrics are relevant as we need high response rate, and we need the dashboard to update as close to real time as possible so the owner can act

accordingly.

11. Non-functional requirements.

11.1 Look and Feel Requirements

- The changeroom screen interface must have a modern, minimalistic design with large, touch-friendly buttons suitable for quick use.
- Visual branding should align with the retailer's style (colors, fonts, logo).
- Outfit recommendations and Mix & Match previews must display clear, high-resolution images of clothing items.
- Interfaces must adapt to multiple screen sizes (e.g., 24–32 inch kiosk displays, tablets for staff).

11.2 Usability and Humanity Requirements

- Customers with no technical background should be able to complete core tasks (request a size, checkout, leave feedback) in 3 steps or fewer.
- Accessibility: screens must support large text mode, high contrast, and voice guidance for visually impaired users (WCAG 2.1 compliance baseline).
- The system must support multilingual UIs (at minimum English + French for Canada; extensible to others).
- Feedback prompts should use friendly, non-technical language (e.g., "Didn't fit right?" instead of "Report error").
- Customers must never feel "watched"; the system must clarify that no cameras or biometrics are used inside changerooms.

11.3 Performance and speed requirements

- AI recommendations must load in $\leq$ 1.0 second for 95% of queries.
- Queue updates and staff alerts must propagate in under 2 seconds end-to-end.
- Checkout transactions must complete in $\leq$ 90 seconds on average.
- System must support at least 50 concurrent changerooms per store without performance degradation.
- Daily analytics dashboards must update within 15 minutes of events.

11.4 Security and Privacy

- All customer logins, feedback, and checkout data must be encrypted in transit (TLS 1.2+) and at rest (AES-256).
- The system must use tokenized identifiers (no plain PII storage).
- Payment processing must comply with PCI-DSS standards; sensitive card details are never stored by the system.
- Login sessions must auto-timeout after 5 minutes of inactivity.
- Only authorized store staff can access dashboards and feedback, protected by role-based access control.

11.5 Legal + Compliance

- The system must comply with **consumer privacy laws** (GDPR, CCPA if deployed in the U.S., and PIPEDA in Canada).
- Accessibility must comply with **Accessibility for Ontarians with Disabilities Act (AODA)** in Ontario, or equivalent in other jurisdictions.
- Store-collected data must be stored only on **approved cloud/data centers** (e.g., AWS Canada for Canadian operations).
- Customer feedback cannot be used for marketing without **explicit opt-in consent**.
- Proper licensing for **datasets** (Polyvore is research-only) must be acknowledged and limited to prototyping; production datasets must come from retailer-owned stock images.

12. Risks and issues predicted.

- User adoptability (A lot of users will be weary and concerned at first with having a tablet inside their fitting rooms or anywhere nearby)
  - **Impact:** Could impact growth and user trust
  - **Mitigation:** Clear signs explaining what the system does with exclamations stating no video or camera is used
- The recommendation engine might not be as accurate as needed
  - **Impact:** Might lose customer trust quickly if they are not getting good recommendations
  - **Mitigation:** Start off with a simple version that works for sure before rolling out the full thing. Constantly work on improving it so that this doesn't happen.
- Staff resistance to the workflow change
  - **Impact:** Staff might be dismissive of the machine
  - **Mitigation:** Involve staff in the pilot project helping them understand how their roles and responsibilities will change. Show the staff the clear productivity gains which will help incentivize the adoption.

Project Development Plan

13. Team Meeting and Communication Plan

- **Meetings:** Weekly meetings on Wednesday at 3pm for 1-2 hours will be held for quick updates, progress, and discussing priorities for the coming week.
- **Communication Channels:** Several methods of communication will be used, but the primary for daily coordination, quick questions, and team-wide announcements will be held on Discord. Discord will allow us to set up a server that has channels for each section of the project. For instance, we have designed the following channels: #general, #fortend, #backend, #ML, and #design. In addition to discord, the group plans on using Microsoft Teams to communicate with the TA and professor.

  Document Sharing and Collaboration:

  For documentation and file sharing such as the SRS, we will be using Google Drive as our central repository system. For prototype designs of the project, we will be using Figma which gives us the opportunity to work with interactive user interface mockups. For source code, we will be storing all files in a GitHub repository. In the repository we will manage version control and issue tracking. Peers will be notified and will review any changes members have pushed on to the repository. We will ensure each push and pull request is properly documented in the branch system.

Coding Environments:

- Frontend systems will use Next.js with integration of Tailwind & Figma.
- Backend systems will use Node.js/Express (API services)/FastAPI.
- For machine learning prototypes we will begin using Jupyter Notebooks for initial model training & evaluation.
- For the final machine leanring system, we will use Python scripts packaged in Docker containers, run on cloud/GPU when needed.

14. Team Member Roles

a) Who is responsible for each of the functional and non-functional requirements?

Name	Role	Functional Requirements	Non-Functional Requirements
Rushit Shah	Software & Platform Engineer	P0 – Changeroom Session + Item Identification P1 – Core Analytics AI Recommendation Engine Mix & Match Mode	Security, Privacy, Performance (latency < 500 ms), Cloud Deployment
Saagar Racharla	Data Engineer & Solutions Architect	P0 – Session/Inventory Data Models P1 – Request Additional Size/Colour Analytics Data Pipelines	Compliance (GDPR/PIPEDA/PCI-DSS), Data Governance, Cloud Infrastructure (AWS/DynamoDB/S3)
Manush Patel	Backend Engineer & BI Analytics Lead	P0 – Changeroom Session + Item Identification P1 – Core Analytics + In-Changeroom Feedback & Dashboard	Data Visualization, Reporting Dashboards, Database Optimization
Garv Rastogi	Frontend & UI/UX Engineer	P0 – Changeroom UI + Live Tray P1 – Request Size/Colour UI Save & Share Outfits	Usability, Accessibility (WCAG/AODA), Multilingual Interface
Partik Singh Dev	Backend & API Developer	P0 - Session ID / Checkout Service P1 - Request Fulfilment Queue Backend	Look & Feel, Testing (Unit/E2E), Responsiveness
Gurmandeep	Frontend & QA	P0 - Changeroom Session	System Reliability,

Johal	Tester	Interface Testing P1 - Queue Management (Customer Side)	Logging, Error Handling
Mahdi Mohammad	Machine Learning & AI Engineer	P0 - Data Simulation for SKU + Session Events P1 - AI Recommendation Engine + Personalized Login	ML Performance Metrics, Model Evaluation

b) **Coordinator/program manager:** Saagar Racharla

15. Workflow Plan

GitHub will be where the codebase is stored, a different folder will be used for each subsection of the project, including frontend, backend, login, recommendation, mix and match and more. On Github we will have two core branches which include main and develop. Main will be the release ready branch and develop will be the branch used for integration. From here for each feature another branch will be made off of the develop branch. Pull requests will require two approvals for any pull request onto the develop branch, and pull requests onto the main branch will require the whole team's approval. Issue management will be documented on a Kanban board in GitHub Projects and will be addressed as features get implemented and pulled onto the develop and main branch. Our team will follow two-week sprints, where every two weeks each member will be responsible for delivering the tickets they are assigned. At the end of the two weeks, we will be having sprint reviews and retrospectives to ensure maximum efficiency.

Our primary data lake will be AWS S3, and from here we will leverage dynamoDB, and more for each feature. Heavy tasks will be run on certain members laptops, if needed we will utilize AWS Cloud GPUs for training models. To achieve the performance metrics and requirements outlined in our SRS, the frontend will be developed with Next.js and Tailwind CSS to ensure fast load times and WCAG 2.2AA accessibility compliance. Real-time communication between the kiosk, backend, and staff dashboard will be handled using Socket.IO, while Redis will be used for caching and queue management to maintain low latency and responsive updates. The backend will be powered by FastAPI and hosted on AWS Elastic Beanstalk for scalability and 99% uptime, with DynamoDB managing real-time inventory data through AWS Lambda triggers. An AI recommendation engine built with PyTorch will enhance personalization, and all payment transactions will be securely processed via Stripe with JWT-based authentication for user security. Automated testing using Jest and Playwright, combined with CI/CD pipelines through GitHub Actions, will ensure high reliability and code quality.

16. Proof of Concept Demonstration Plan

During the Proof-of-Concept (PoC) demonstration, the team will present a functional end-to-end prototype that highlights both the core changeroom workflow and a clear element of algorithmic depth through our rule-based recommendation engine.

Customer Workflow:
A customer will scan a product barcode at the kiosk, initiating a changeroom session. The screen displays the scanned items with details such as name, colour, and size, and allows the user to

request alternate sizes or colours. These requests appear instantly on the staff dashboard, where store associates can mark each as “Picked Up” or “Delivered.” The kiosk interface updates the status in real time to reflect these changes. Once the session ends, the customer will be prompted to provide short feedback, which automatically updates the analytics dashboard.

Algorithmic Component (Rule-Based Recommendation Engine):
To demonstrate technical depth, the PoC will include a working rule-based recommendation system that generates outfit suggestions dynamically. When a customer scans an item, the backend executes a scoring algorithm that evaluates catalogue items based on colour compatibility, brand affinity, and price-range similarity. The top-scoring items are displayed under a “Recommended for You” panel on the kiosk screen. This component provides tangible algorithmic reasoning and will include a short code excerpt or console output illustrating how the system calculates and ranks recommendations.

Analytics and Dashboard:
The store-side dashboard will visualize real-time and simulated data such as total active sessions, most-requested sizes, average fulfilment time, and feedback ratings. Even if some values are mock data, they will demonstrate the full data flow between kiosk, backend, and analytics layers.

Goal for the PoC:
By the PoC milestone, the system will provide a cohesive demonstration of:
1. Item scanning and session creation
2. Real-time request fulfilment between kiosk and staff dashboard
3. Dynamic rule-based outfit recommendations (algorithmic depth)
4. Feedback submission updating the analytics dashboard

This ensures the PoC highlights both functional progress and computational intelligence, aligning with capstone expectations for technical and algorithmic rigor.

17. Technology

17.1 Programming Languages, Environments, and Testing

Our project will be using TypeScript with Next.js to construct an interactive kiosk system as our frontend interface that will allow users to browse, scan, and receive assitance. The frontend interface will showcase accessibility, responsiveness, and quick interactions with the use of TailwindCSS for design and Next.js built-in routing for navigation systems. For the backend, we will be running Express on Node.js that will be written in TypeScript so that it can manage session handling, API requests, and interactions with the AI recommendation system. Using Typescript across the system will allow for consistency and strong type development throughout the stack making the system easier to maintain and debug.

The development process will primarily be covered in Visual Studio Code with additional extensions such as ESLint, Prettier, and GitLens to ensure that the code is standardized and presentable. Repository management will be managed using GitHub. We will follow a branching workflow where all changes are made on separate branches and pull requests require mandatory peer review. Moreover, we will be using the Jest and Playwright testing library to ensure unit and integration testing for the frontend system. For backend logic and API testing, we will use Mocha and Chai as it works well with Node.js. For continued testing and automation, we will be using GitHub Actions with Docker to have consistent environments across local, test and production systems.

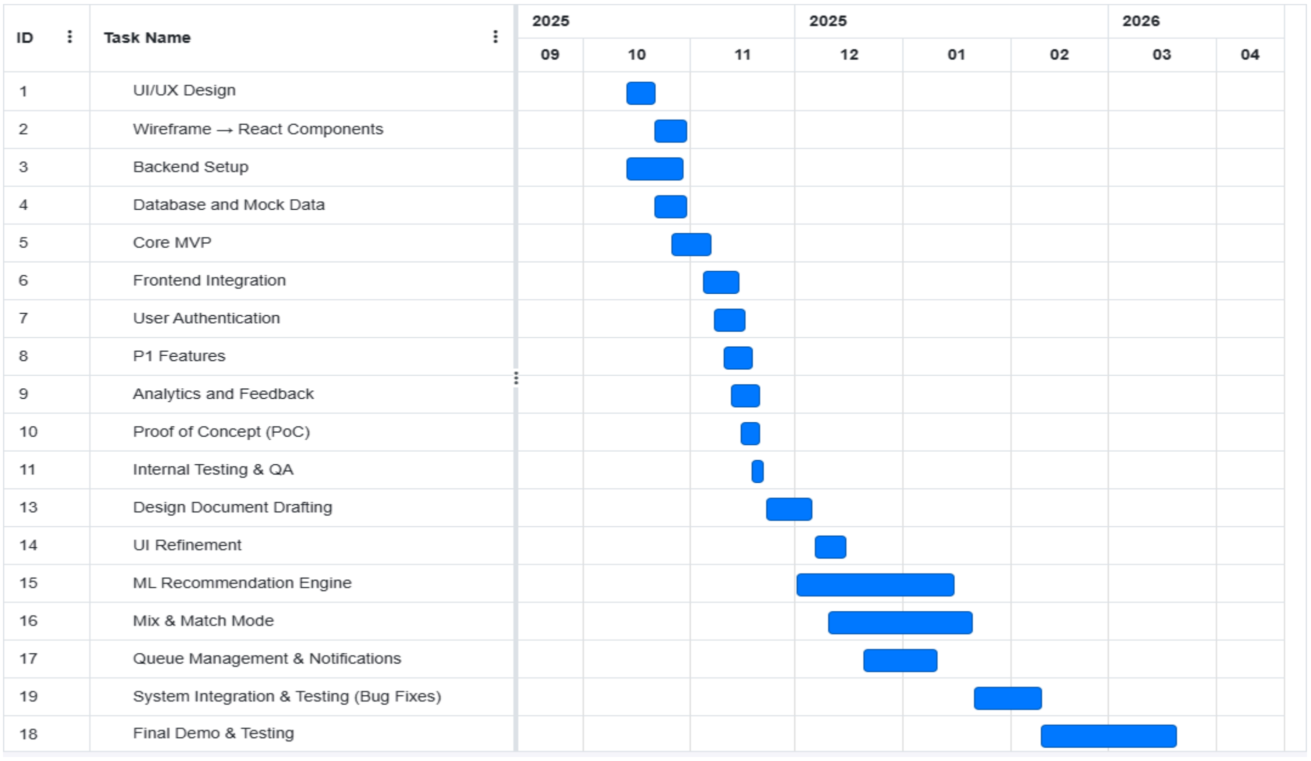
17.2 Machine Learning Libraries

We will be using Python to develop our machine learning recommendation system along with PyTorch and scikit-learn to build, train, and test the model. The frameworks will help the development of neural models that will have the ability to recommend complementary clothing items based on visuals and text features. The dataset, Polyvore Fashion Dataset, will be used to train and test our model on the focus of outfit compatibility prediction. Data will be processed with the use of Pandas and NumPy along with matplotlib for plotting the visual performance and metrics. We will deploy our model through FastAPI and integrate it into our main system using RESTful endpoints. For evaluation we will be using metrics such as Precision@K and Recall@K to measure our recommendation accuracy and satisfaction.

17.3 GPU Usage and Infrastructure

The use of GPU acceleration will be primarily used for model training to reduce the computation time. We will be employing AWS EC2 GPU or Google Colab. After training the model, we will have the machine learning service run in Docker containers hosted on AWS DynamoDB will manage real-time inventory and session data while Amazon S3 for image storing. Additionally, the use of fully automated CI/CD will be pipelined through GitHub Actions; it will manage, test, and have staged developments. Environment variables will be securing sensitive credentials while AWS CloudWatch and Sentry will be responsible for monitoring the performance logs, and errors. It will ensure reliability and scalability for any future integrations. Lastly, the kiosk system will be equipped with a USB bar code scanner that will ensure low-latency and seamless interactions.

18. Project Scheduling: Gantt chart.



ID	Task Name
1	UI/UX Design
2	Wireframe → React Components
3	Backend Setup
4	Database and Mock Data
5	Core MVP
6	Frontend Integration
7	User Authentication
8	P1 Features
9	Analytics and Feedback
10	Proof of Concept (PoC)
11	Internal Testing & QA
13	Design Document Drafting
14	UI Refinement
15	ML Recommendation Engine
16	Mix & Match Mode
17	Queue Management & Notifications
19	System Integration & Testing (Bug Fixes)
18	Final Demo & Testing