Team 25 | September'25

Table of Contents

1 MILESTONE 1 | PAGES 01 - 09

2 MILESTONE 2 | PAGES 10 - 57

3 MILESTONE 3 | PAGES 58 - 86

4 MILESTONE 4 | PAGES 87 - 97

5 MILESTONE 5| PAGES 98 - 107

6 MILESTONE 6| PAGES 108 - 121

Software Engineering Project - September'25

Sync'em

An AI-powered assistant app that helps HR and project managers efficiently allocate projects based on employee skills and availability, while providing a chatbot for employees for quick HR policy queries.

Team Members

1. AKBAR ALI |23f1002997
2. DEON LEVON DMELLO | 21f3002473
3. D NARENDRAN | 22f3002698
4. SAMPRITI RAHA | 22f1000064
5. SAYAN BHOWMICK | 22f3001657
6. SOHAM CHAKRABORTY | 23f1001783
7. TELVIN VARGHESE | 22f2001640
8. TUMMALA NAVEEN | 22f3002759

Table of Contents

User Identification

Primary, secondary, and tertiary users involved in the Project Allocation Assistant App, detailing their roles in managing project assignments, monitoring workload balance, and maintaining the HR chatbot system.

User Stories

Primary, secondary, and tertiary users interact within the Project Allocation Assistant App to allocate projects efficiently, oversee workforce utilization, access HR policy information, and ensure smooth system functionality.

User Identification

1 PRIMARY USERS

- HR Managers: Manage employee information, skill sets, and availability to allocate projects efficiently while ensuring HR policy compliance.
- Project Managers: Assign team members to projects, monitor workload balance, and track progress for timely project delivery.

2 SECONDARY USERS

- Department Leads: Oversee multiple teams to analyze workforce utilization, identify allocation trends, and improve departmental efficiency.
- Employees: View their assigned projects, receive fair workload distribution, and use the chatbot to clarify HR policies or request changes.

User Identification

3 TERTIARY USERS

- Auditors / Compliance Officers - May access reports to ensure HR and project allocation policies are being followed transparently.

- IT Administrators / Security Team - Responsible for configuring, managing, and auditing the system's Role-Based Access Control (RBAC) policies. They ensure that all user types (from executives to auditors) have the appropriate permissions, protecting sensitive data and maintaining system integrity.

User Stories

1 PRIMARY USERS

Without GenAI :

1. **As a Product Manager**, *I* ***want to*** *make structured framework to capture and refine client requirements* ***so that*** *the final solution aligns with their actual needs.*

2. **As a Product Manager**, *I* ***want to*** *access to an employee's skill profile and past performance data* ***so that*** *I can assess their fit before assigning critical work.*

3. **As a Product Manager**, *I* ***want to*** *send client updates (reports, dashboards, alerts)* ***so that*** *clients remain informed about progress and blockers.*

4. **As an HR Manager**, *I* ***want to*** *maintain a centralized HR policy repository* ***so that*** *employees can access information consistently.*

5. **As an HR Manager**, *I* ***want to*** *schedule periodic performance reviews* ***so that*** *employees receive structured feedback throughout the year.*

User Stories

1 PRIMARY USERS

With GenAI :

1. **As a Product Manager**, *I* ***want to*** *use a GenAI-powered tool that can create a structured framework for capturing and refining client requirements,* ***so that*** *the final solution accurately aligns with their true needs.*

2. **As a Product Manager**, *I* ***want to*** *use a GenAI tool that provides access to an employee's skill profile and past performance data,* ***so that*** *I can evaluate their suitability before assigning critical tasks.*

3. **As a Product Manager**, *I* ***want to*** *use a GenAI tool that automates client updates through reports, dashboards, and alerts,* ***so that*** *clients stay informed about project progress and potential blockers.*

4. **As an HR Manager**, *I* ***want to*** *use a GenAI-powered system to manage a centralized HR policy repository* ***so that*** *employees can easily access accurate and up-to-date information at any time.*

5. **As an HR Manager**, *I* ***want to*** *use a GenAI-driven tool to automate and optimize the scheduling of periodic performance reviews* ***so that*** *employees receive timely and structured feedback throughout the year.*

User Stories

2 SECONDARY USERS

Without GenAI :

1. **As a Department Lead**, I **want to** generate workforce utilisation reports manually through filters **so that** I can assess productivity by team or project.

2. **As a Department Lead**, I **want to** view aggregated data of ongoing projects **so that** I can evaluate department efficiency.

3. **As an Employee,** I **want to** browse HR FAQs and documents **so that** I can find answers without waiting for HR responses.

4. **As an Employee,** I **want to** submit leave and reimbursement requests via forms **so that** I can track my requests efficiently.

5. ***As an Employee**, I **want to** search for and view a list of skill improvement and learning courses **so that I** can choose the ones that help me grow in areas relevant to my career.*

User Stories

2 SECONDARY USERS

With GenAI :

1. **As a Department Lead**, I **want to** assign projects automatically based on employees' skills, past performance, and current workload **so that** resource allocation becomes data-driven and efficient.

2. As a **Department Lead**, I **want GenAI to** generate monthly insights on workforce utilization trends so that I can make informed resource and hiring decisions.

3. **As an Employee**, I **want to** receive GenAI based feedback summaries quarterly so that I can track my performance and make proactive improvements.

4. **As an Employee**, I **want GenAI chatbot to** *provide me with instant answers to HR-related queries(e.g., dress code, leave, WFH, travel)* ***so that*** *I don't have to wait for my manager to address trivial questions.*

5. **As an Employee,** I **want GenAI to** give me personalised recommendations for skill improvement and learning courses **so that** I can grow in areas relevant to my career.

User Stories

3 TERTIARY USERS

Without GenAI :

1. **As a Developer,** I **want to** maintain rule-based allocation and reporting modules
so that the system functions reliably.

2. **As a Support Engineer,** I **want to** ensure uptime, security, and data accuracy in all modules
so that user experience remains smooth.

With GenAI :

1. **As a Developer,** I **want to** train and update the GenAI model with new HR policies and company data
so that the chatbot provides accurate, current information.

2. **As a Support Engineer,** I **want to** monitor chatbot performance and retrain models as needed
so that user satisfaction and accuracy remain high.

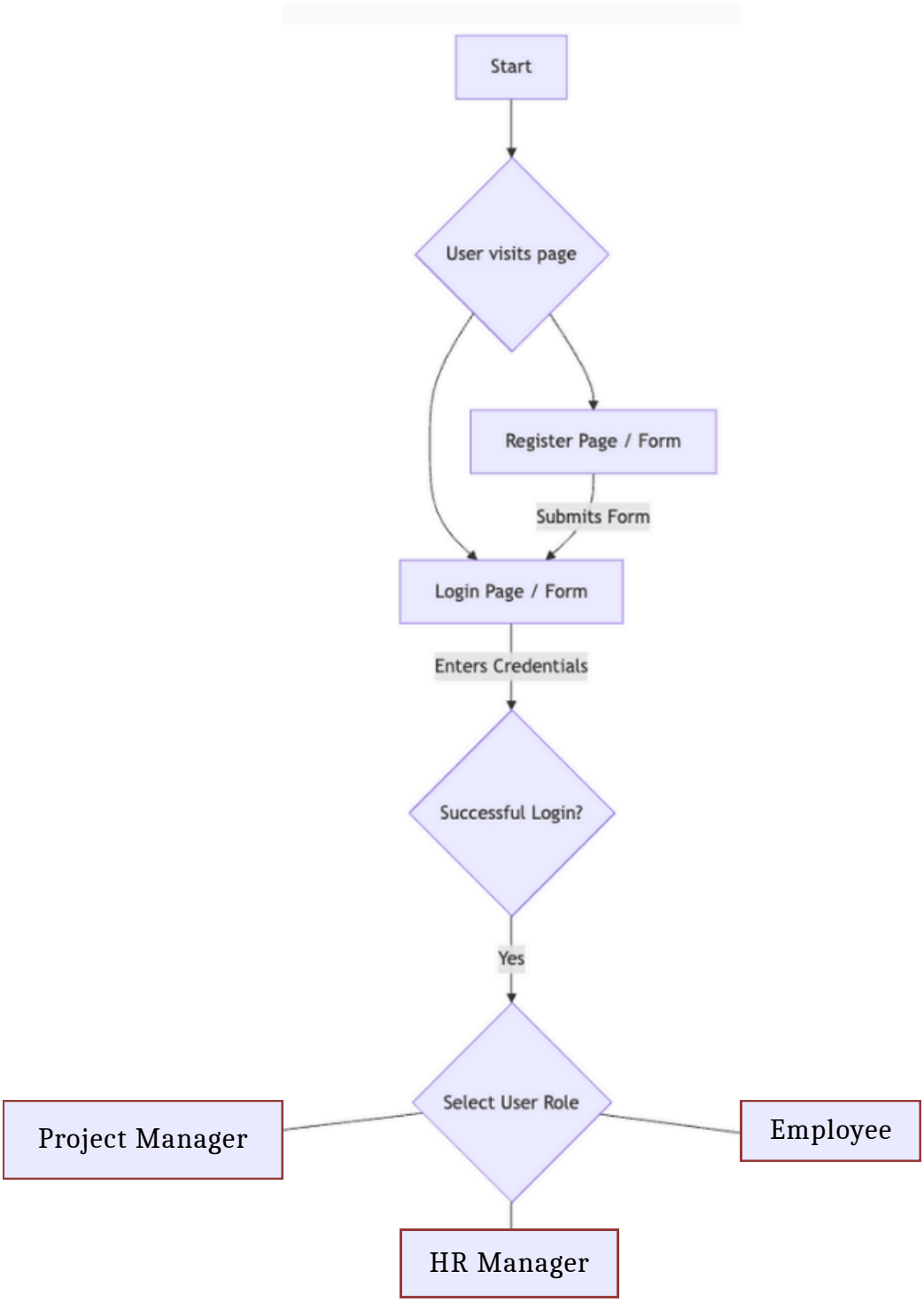
Table of Contents

1 Application Userflow

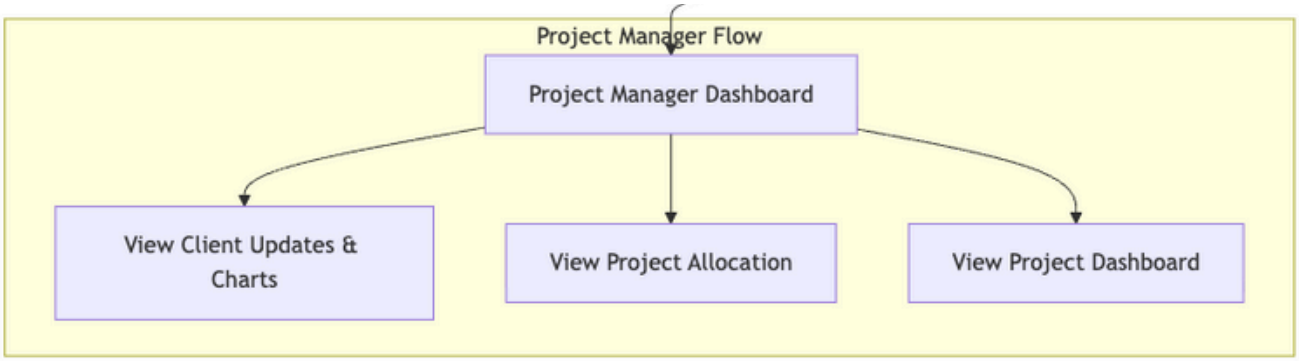
2 Low Fidelity Wireframes

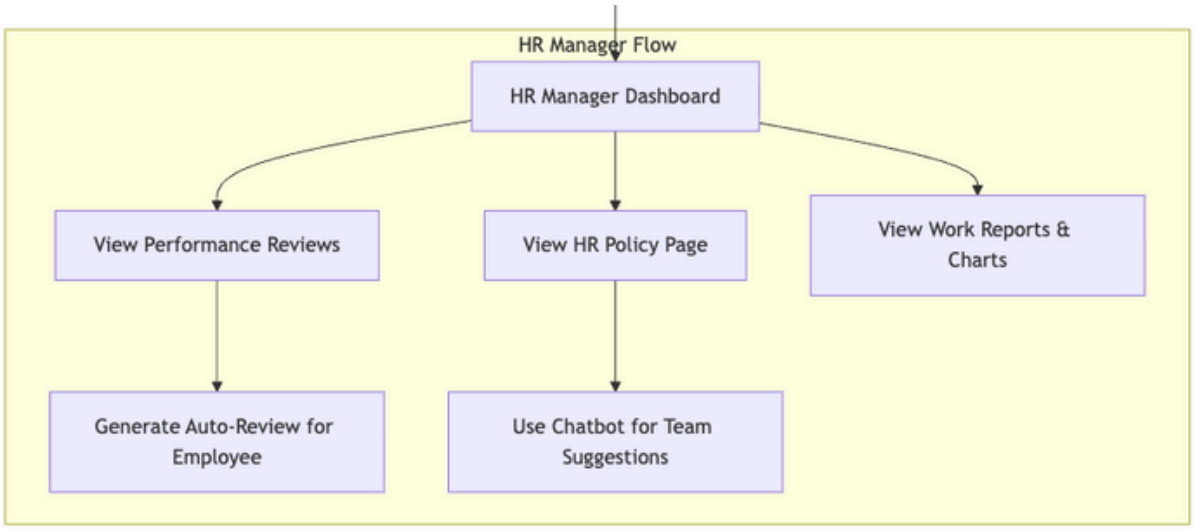
3 Storyboard with User Stories

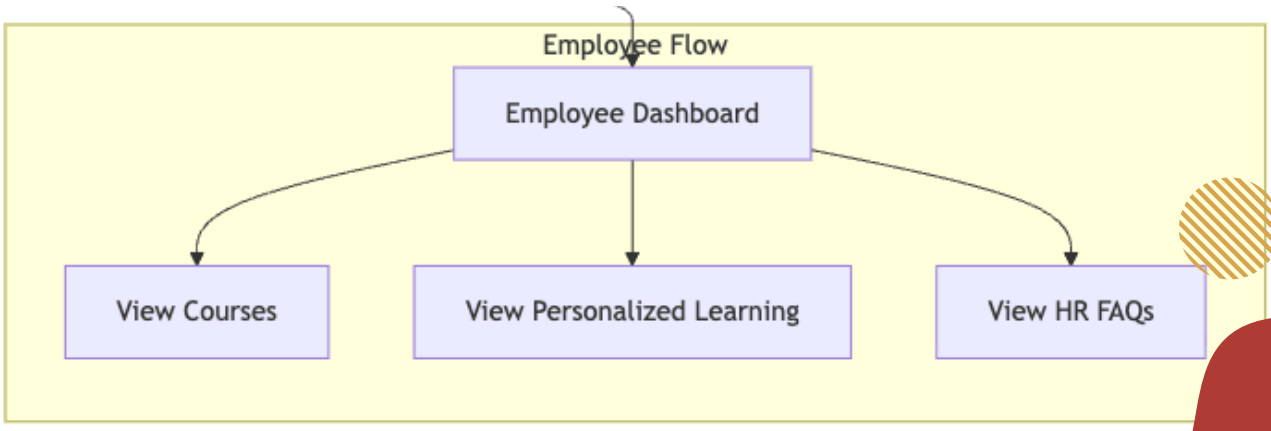
Application Userflow



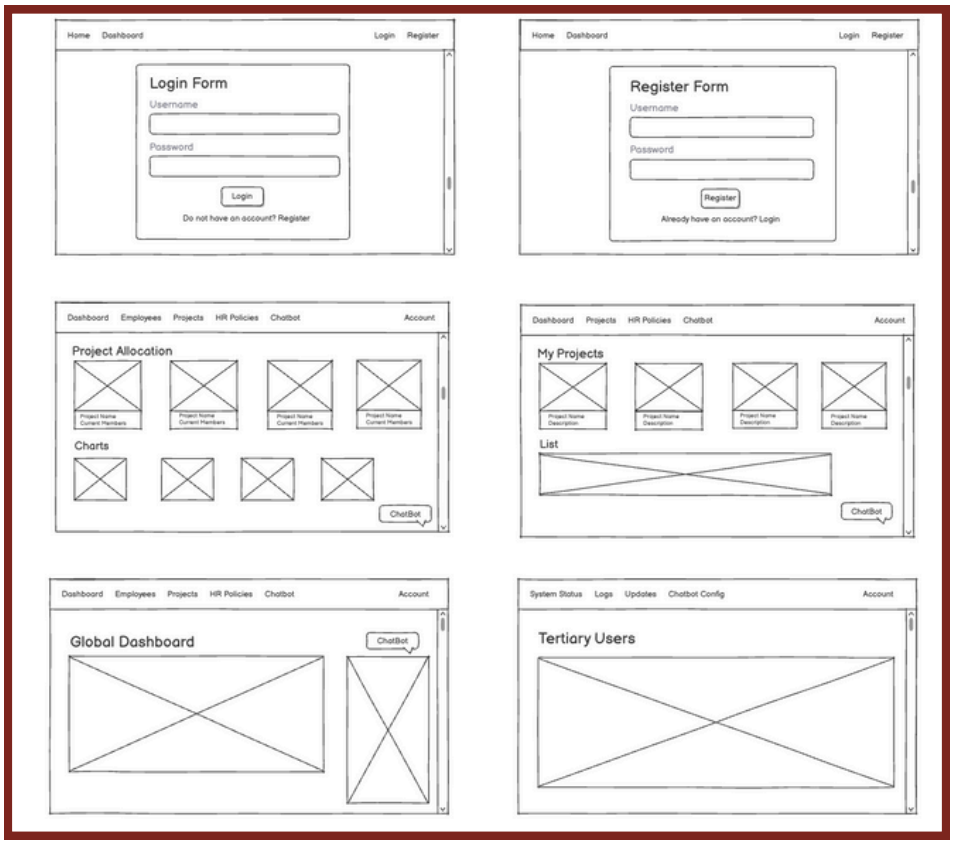
Application Userflow

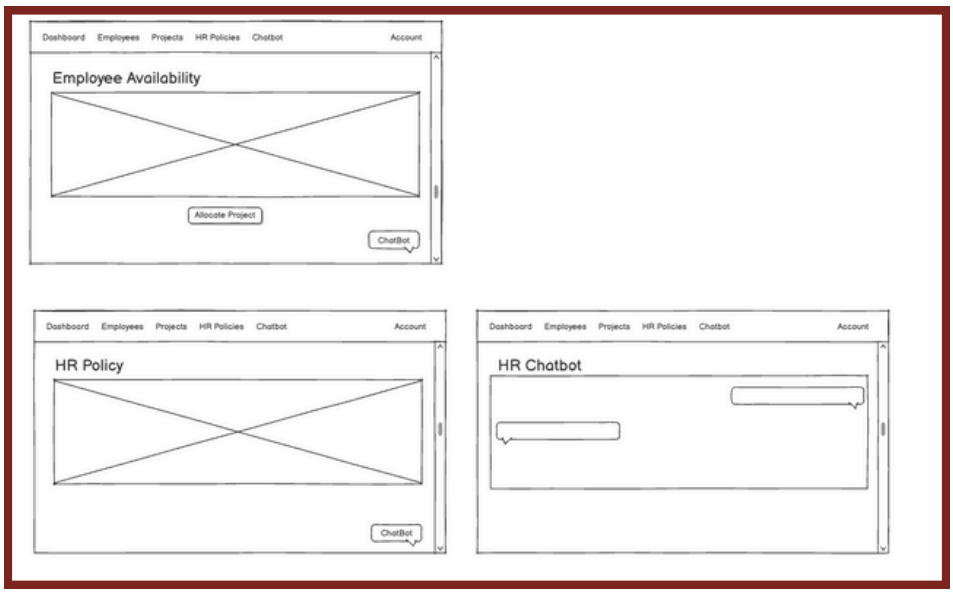




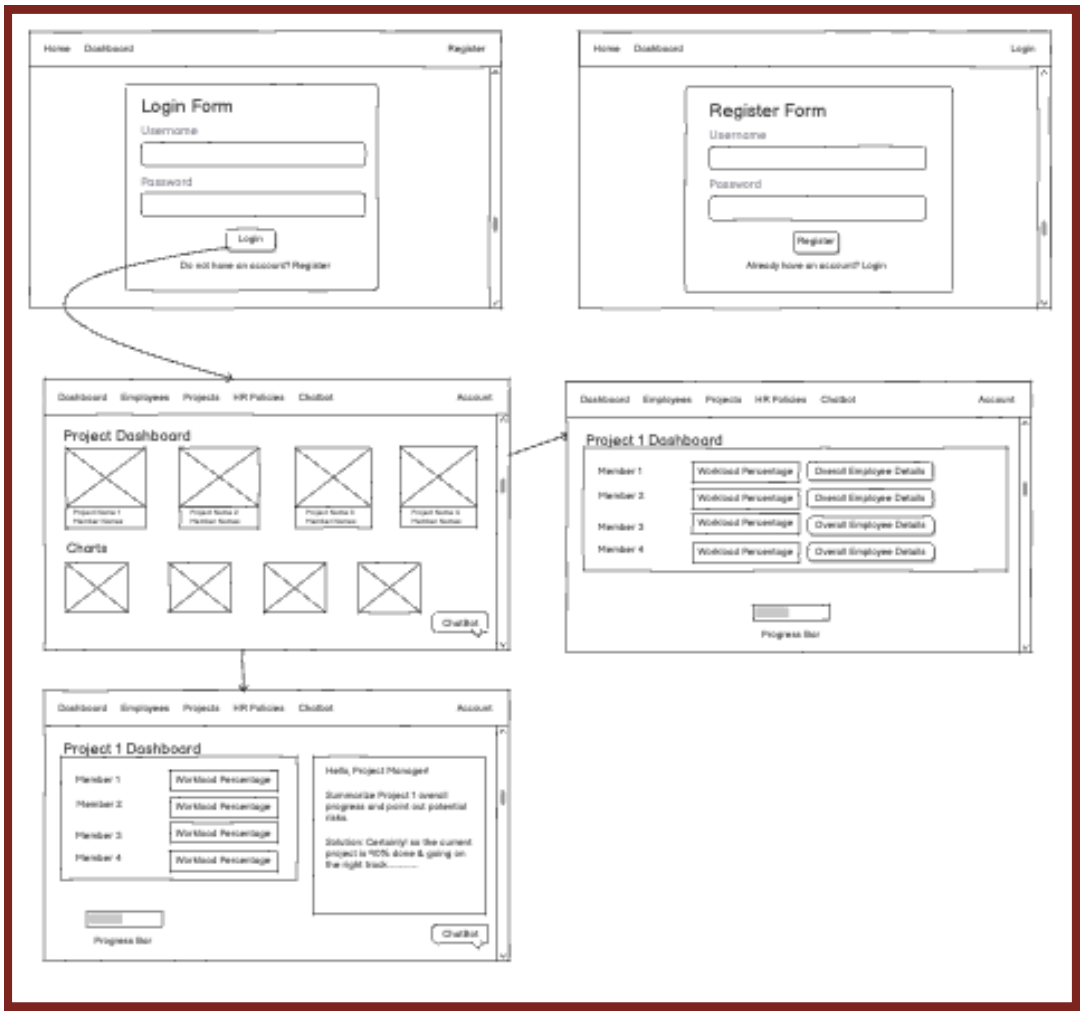


Low Fidelity Wireframes



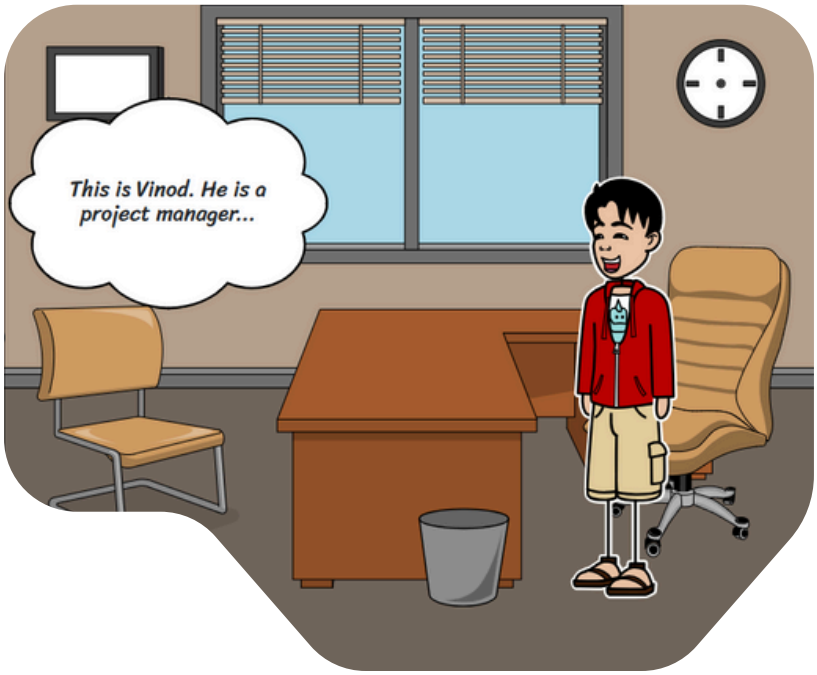


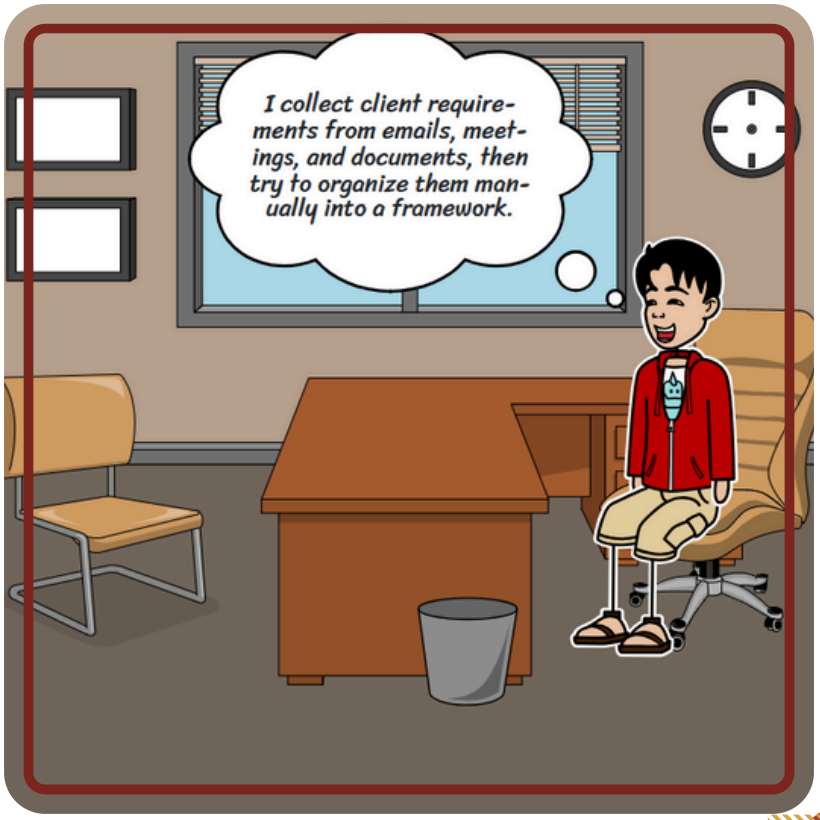
Low Fidelity Wireframes

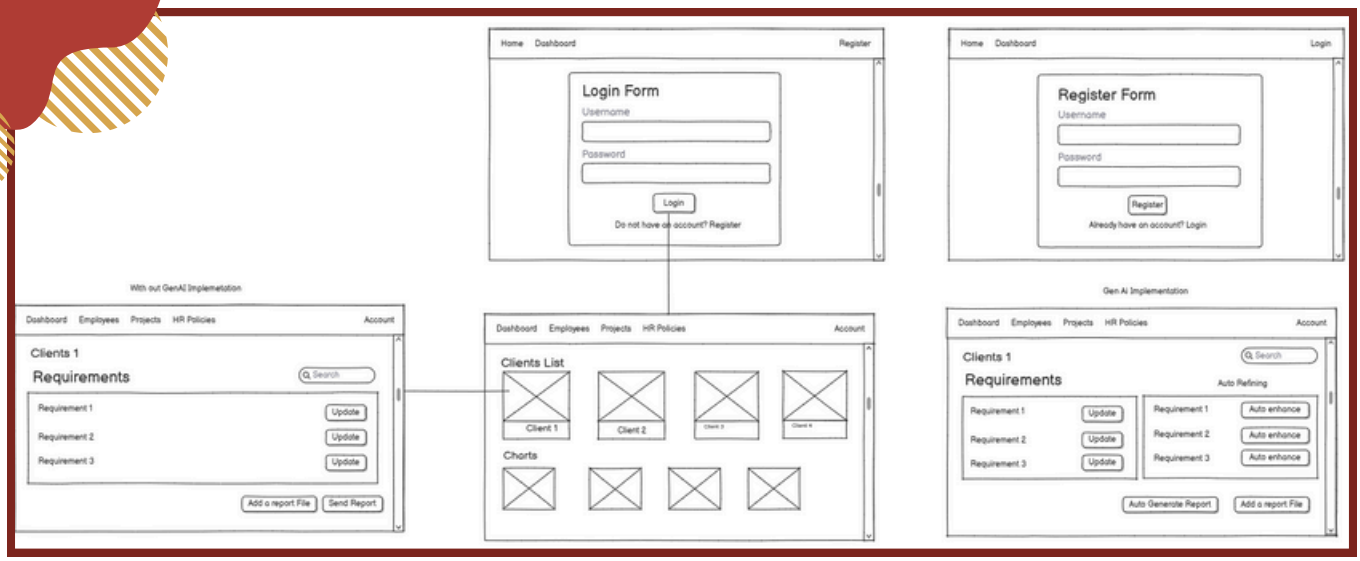


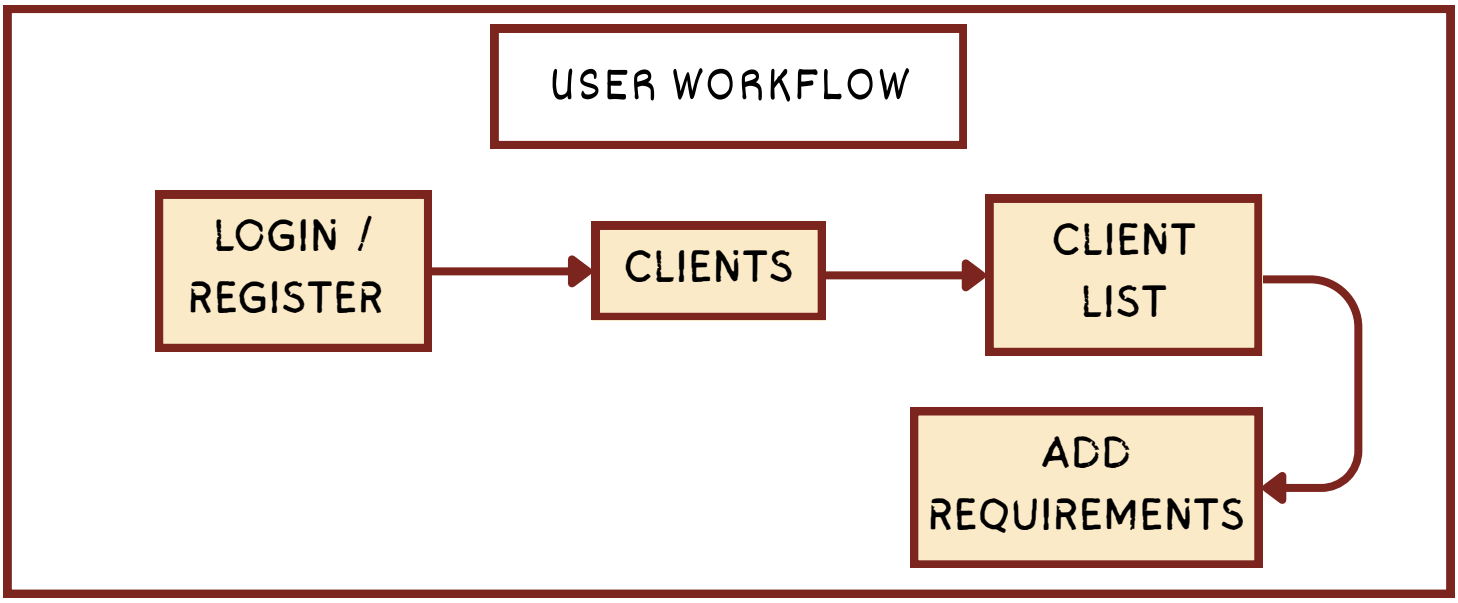
Storyboard with User Stories – Product Manager

Primary User - Scenario 1 - Client Requirements






Home
Dashboard
Register
Login Form
Username
Password
Login
Do not have an account? Register
Home
Dashboard
Login
Register Form
Username
Password
Register
Already have an account? Login
With out GenAI Implemetation
Dashboard
Employees
Projects
HR Policies
Account
Clients 1
Requirements
Search
Requirement 1
Requirement 2
Requirement 3
Update
Add a report File
Send Report
Dashboard
Employees
Projects
HR Policies
Account
Clients List
Client 1
Client 2
Charts
Gen Ai Implementation
Dashboard
Employees
Projects
HR Policies
Account
Clients 1
Search
Requirements
Auto Refining
Requirement 1
Requirement 2
Requirement 3
Update
Auto enhance
Auto Generate Report
Add a report File

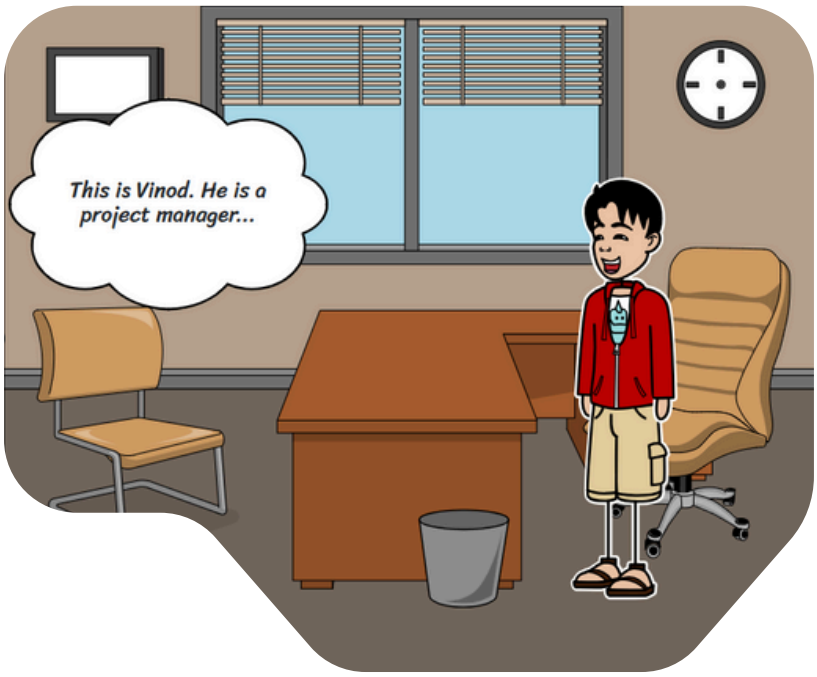

USER WORKFLOW
LOGIN / REGISTER
CLIENTS
CLIENT LIST
ADD REQUIREMENTS

Important details often get lost or misinterpreted, and refining requirements takes too long.

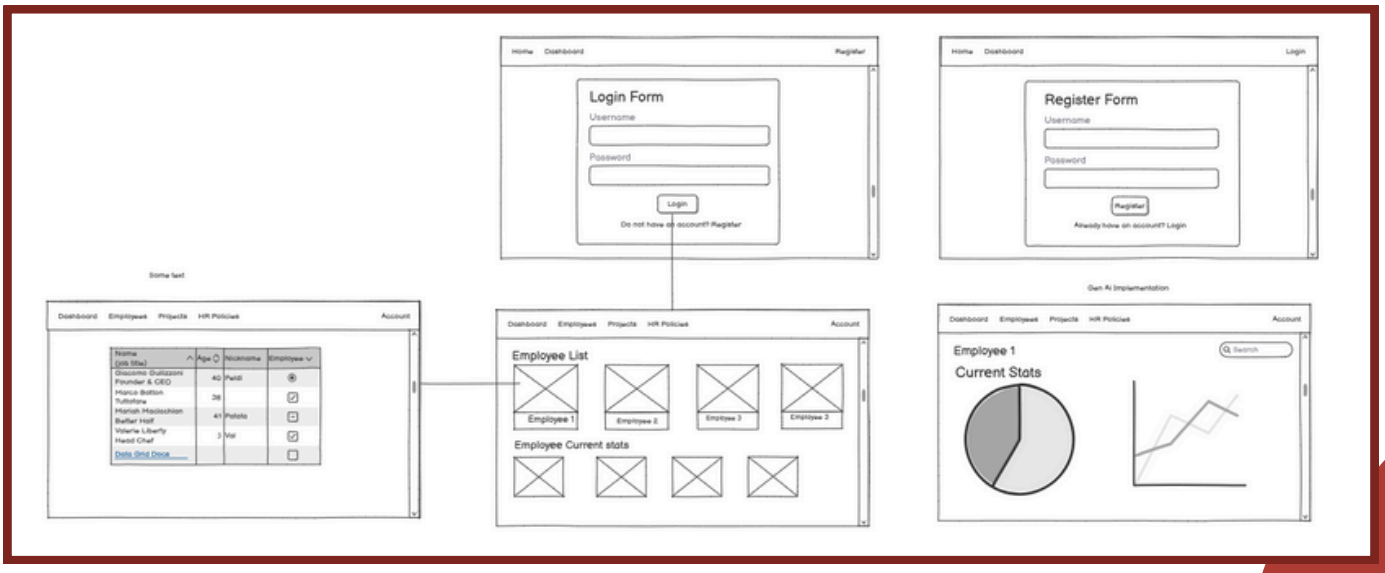
GenAI comes in...

Storyboard with User Stories – Product Manager

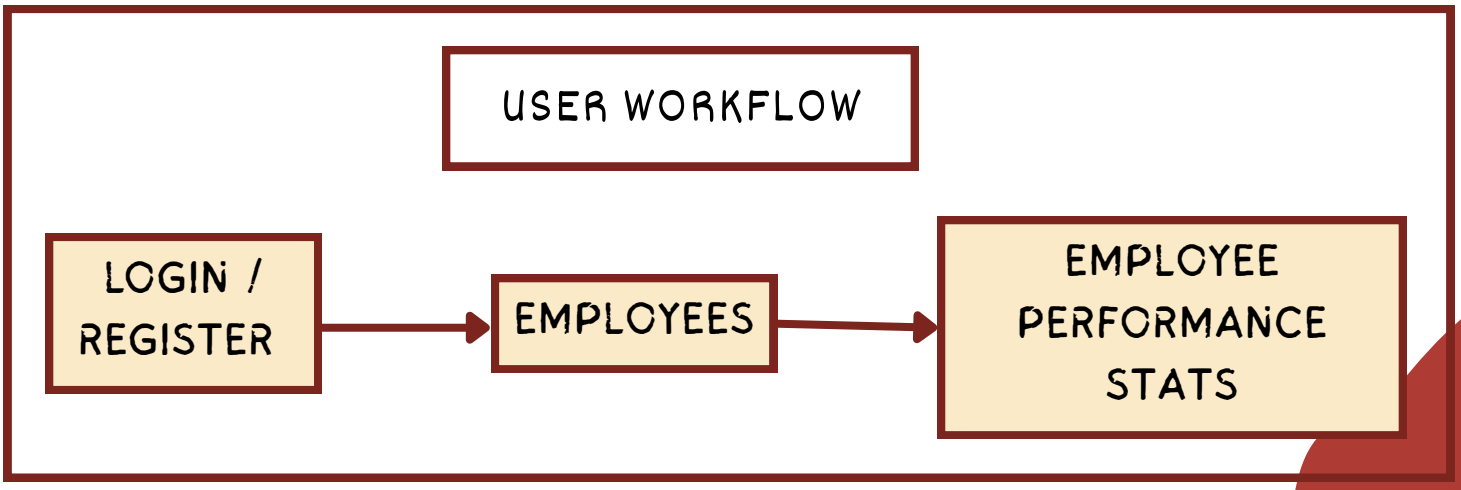
Primary User - Scenario 2 - Query for Best-fit Employees



It's time-consuming and inconsistent — I often miss key insights about an employee's strengths or past project performance.


Login Form
Register Form
Employee List
Employee 1
Employee Current stats
Current Stats

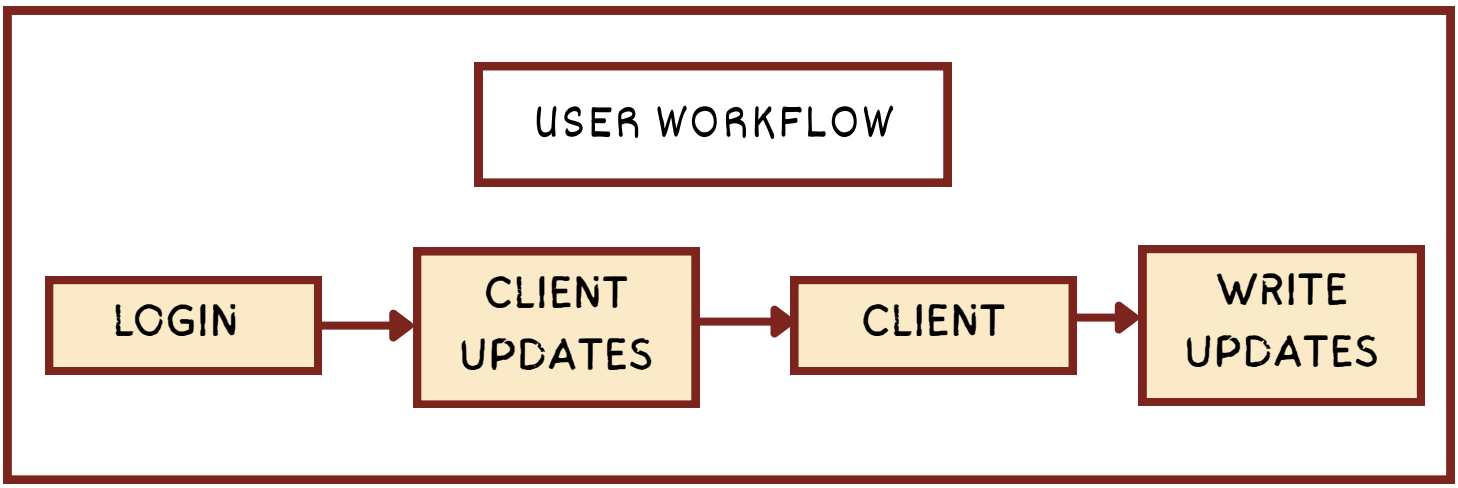
GenAI comes in...

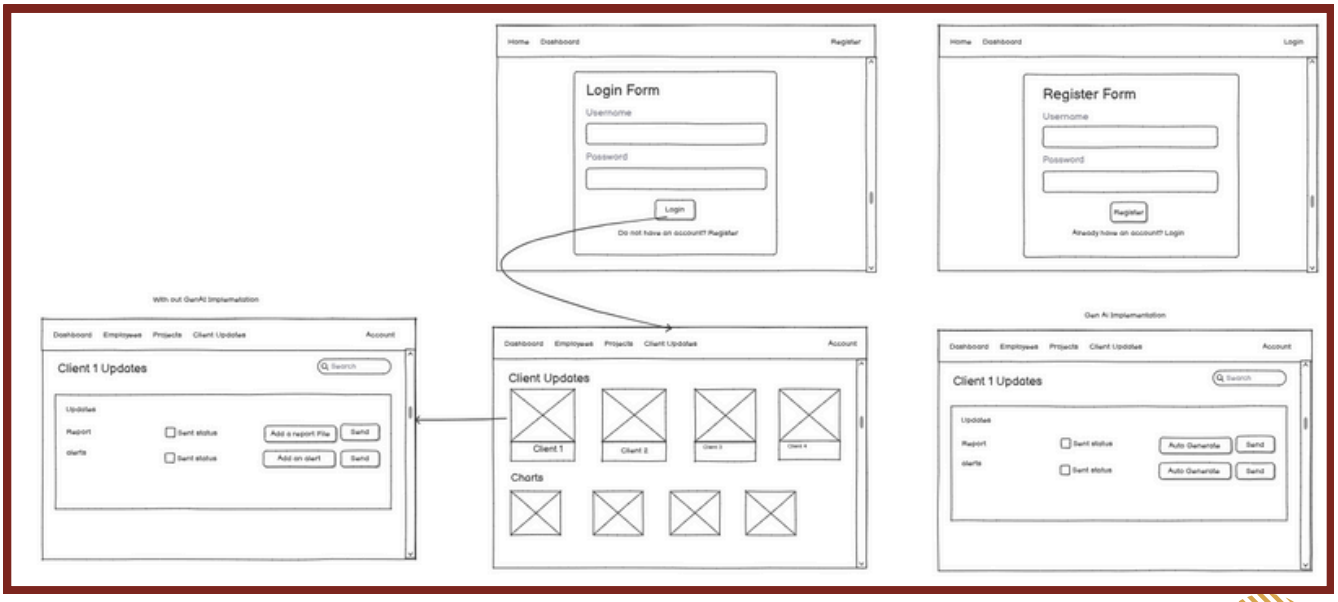


Storyboard with User Stories – Project Manager

Primary User - Scenario 3 - Keeping Clients Informed

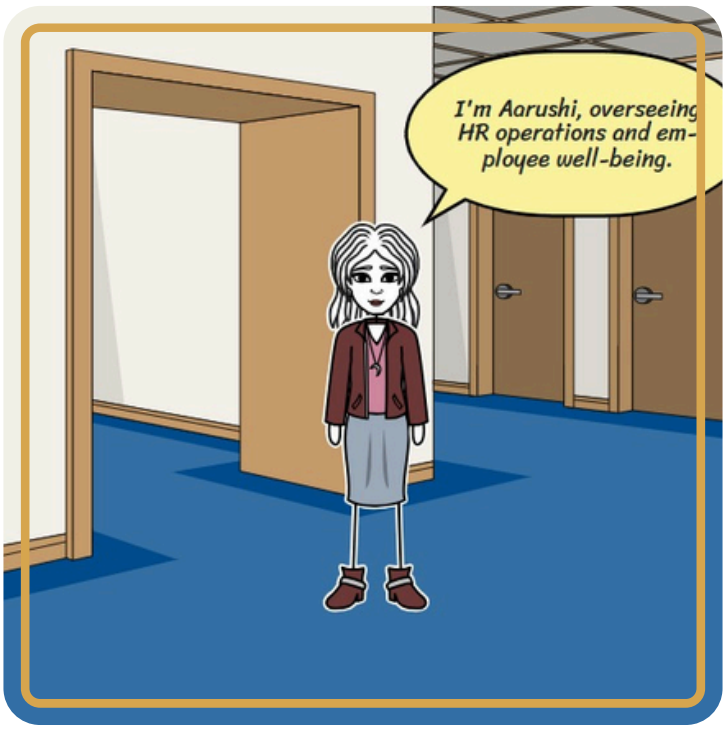
GenAI comes in...

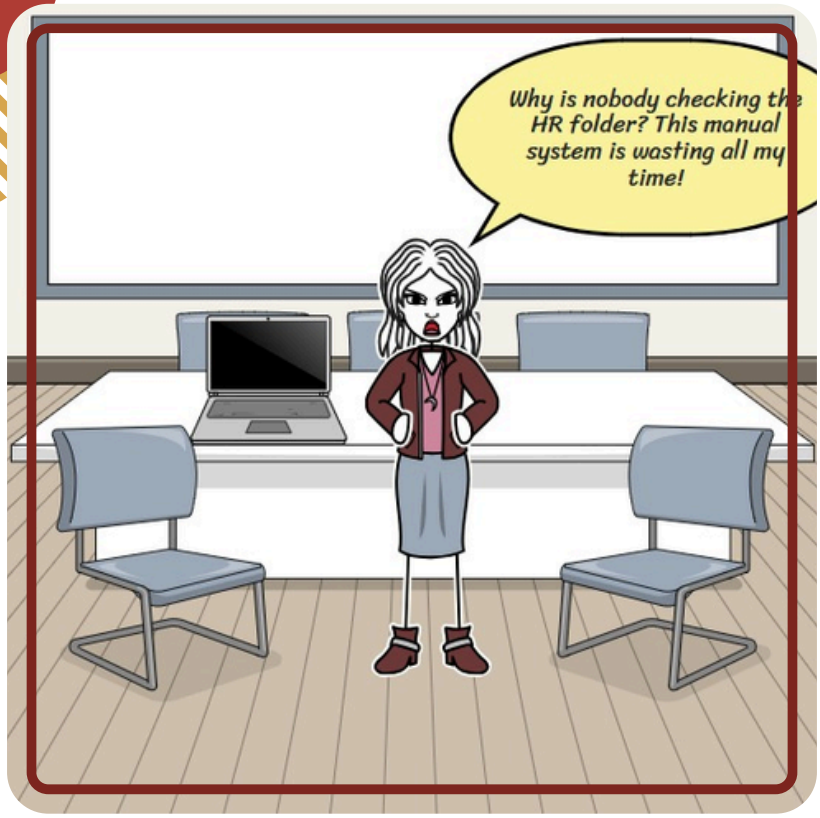

USER WORKFLOW
LOGIN
CLIENT UPDATES
CLIENT
WRITE UPDATES


Login Form
Register Form
Client Updates
Client 1 Updates
Charts

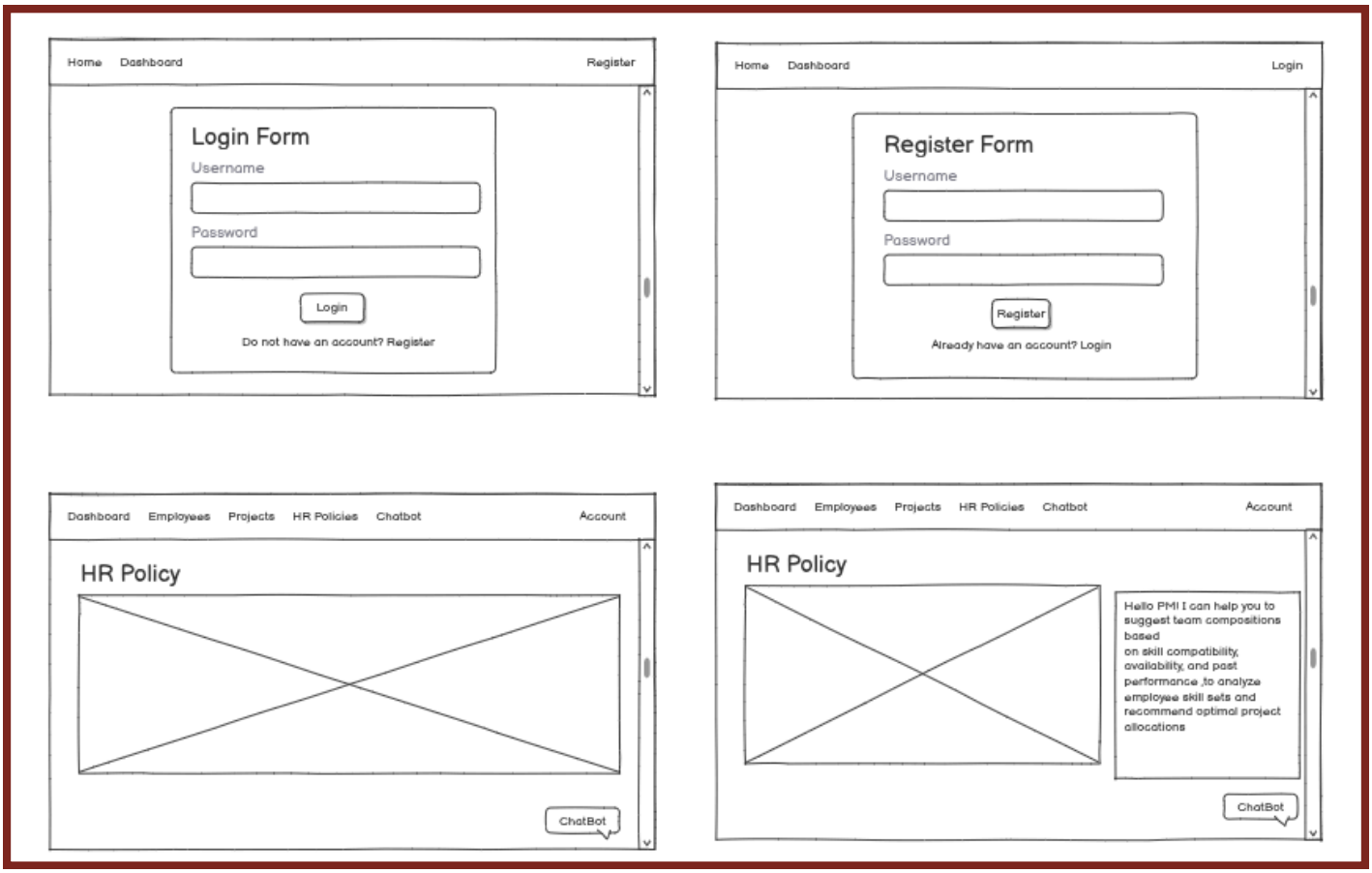
Storyboard with User Stories – HR Manager

Primary User - Scenario 4 - HR Policy Repository



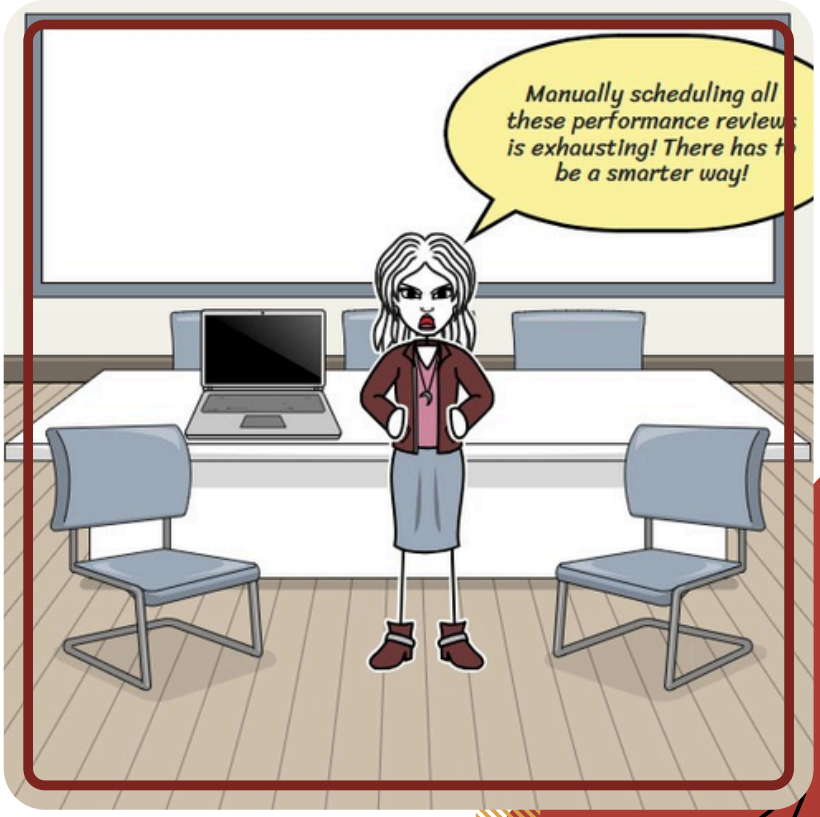


GenAI comes in...

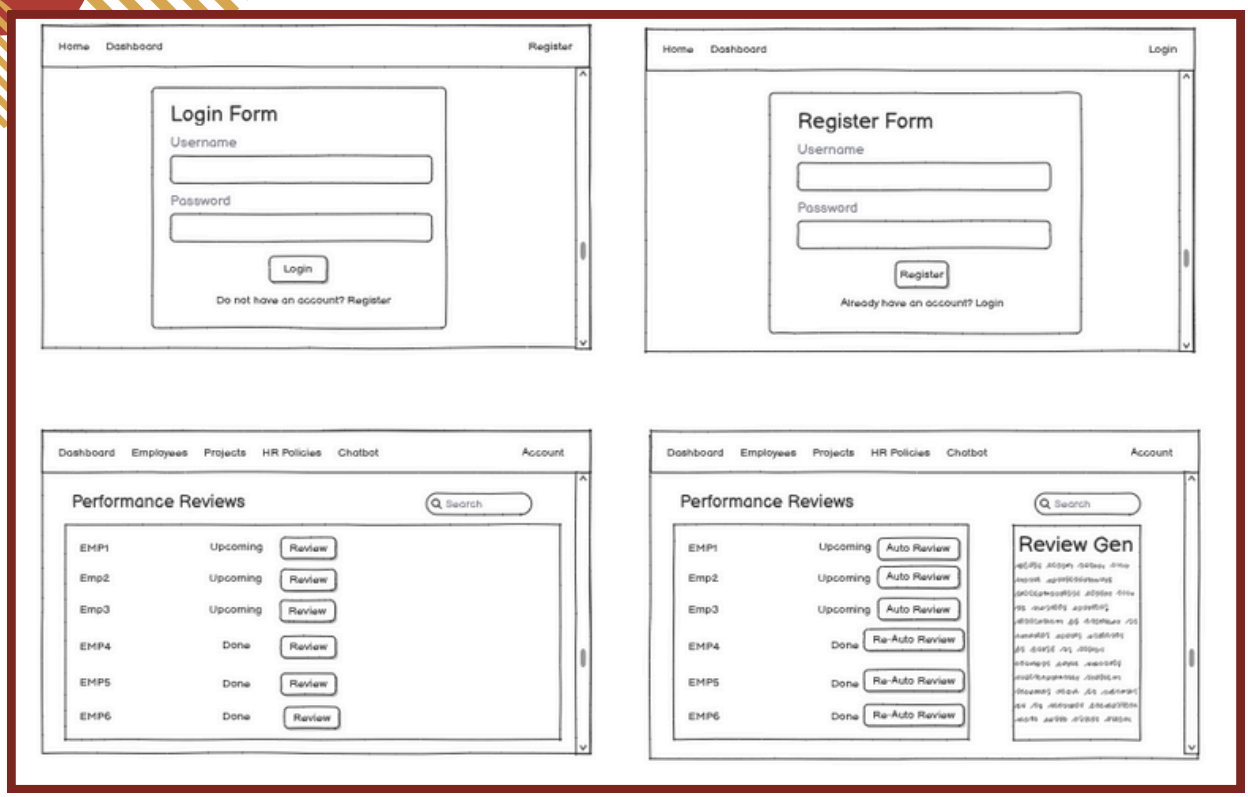

Home
Dashboard
Register
Login Form
Username
Password
Login
Do not have an account? Register
Home
Dashboard
Login
Register Form
Username
Password
Register
Already have an account? Login
Dashboard
Employees
Projects
HR Policies
Chatbot
Account
HR Policy
ChatBot
Dashboard
Employees
Projects
HR Policies
Chatbot
Account
HR Policy
Hello PMI I can help you to suggest team compositions based
on skill compatibility, availability, and past performance ,to analyze employee skill sets and recommend optimal project allocations
ChatBot

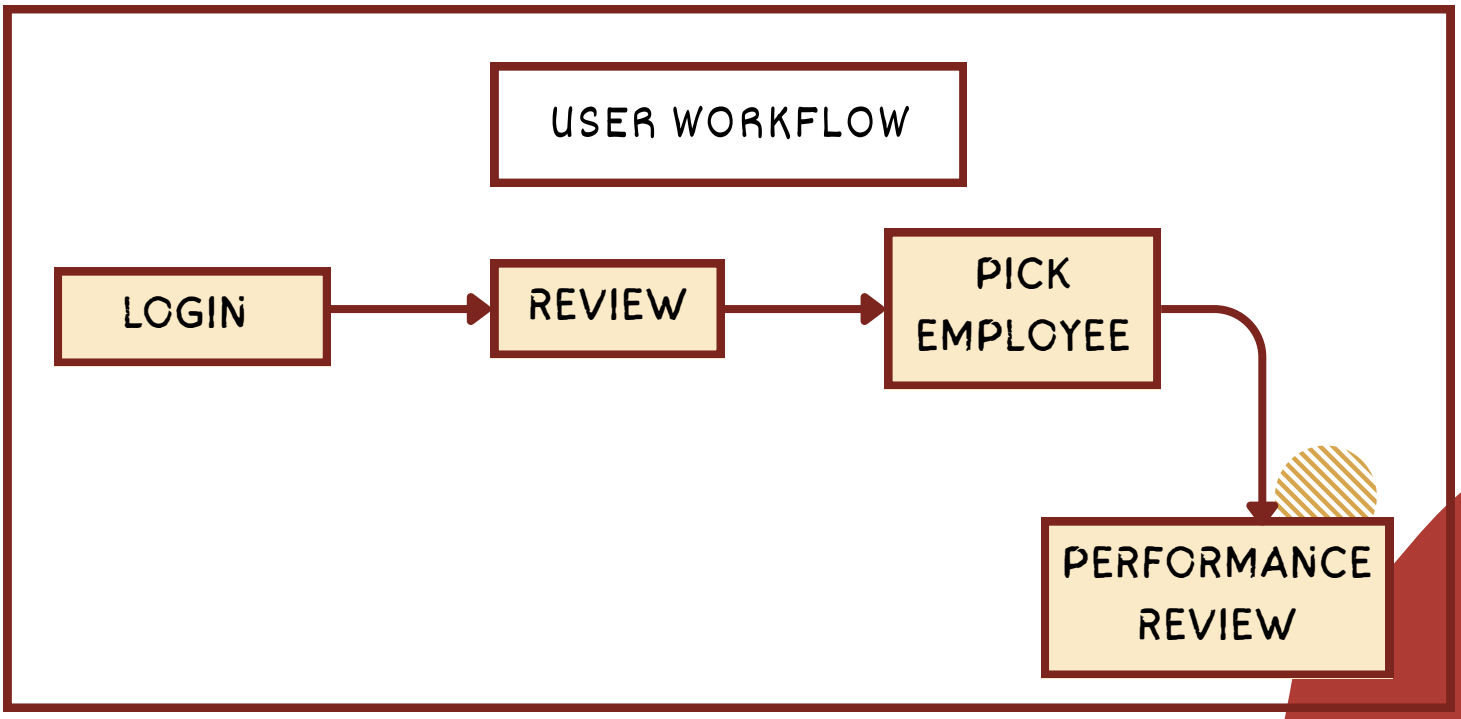
Storyboard with User Stories – HR Manager

Primary User - Scenario 5 - Performance Review



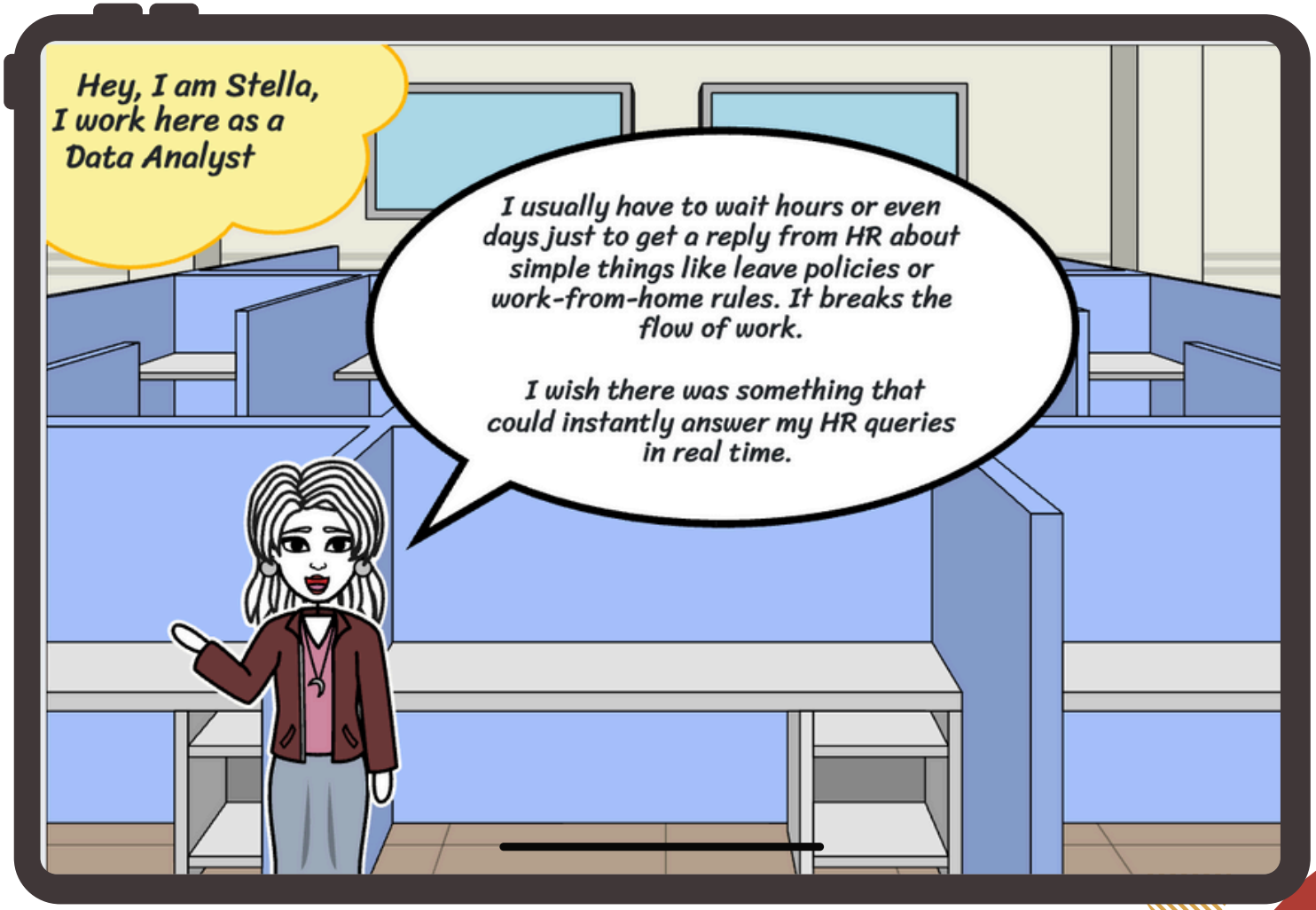
GenAI comes in...


Home
Dashboard
Register
Login Form
Username
Password
Login
Do not have an account? Register
Home
Dashboard
Login
Register Form
Username
Password
Register
Already have an account? Login
Dashboard
Employees
Projects
HR Policies
Chatbot
Account
Performance Reviews
Search
EMP1 Upcoming Review
Emp2 Upcoming Review
Emp3 Upcoming Review
EMP4 Done Review
EMP5 Done Review
EMP6 Done Review
Dashboard
Employees
Projects
HR Policies
Chatbot
Account
Performance Reviews
Search
EMP1 Upcoming Auto Review
Emp2 Upcoming Auto Review
Emp3 Upcoming Auto Review
EMP4 Done Re-Auto Review
EMP5 Done Re-Auto Review
EMP6 Done Re-Auto Review
Review Gen

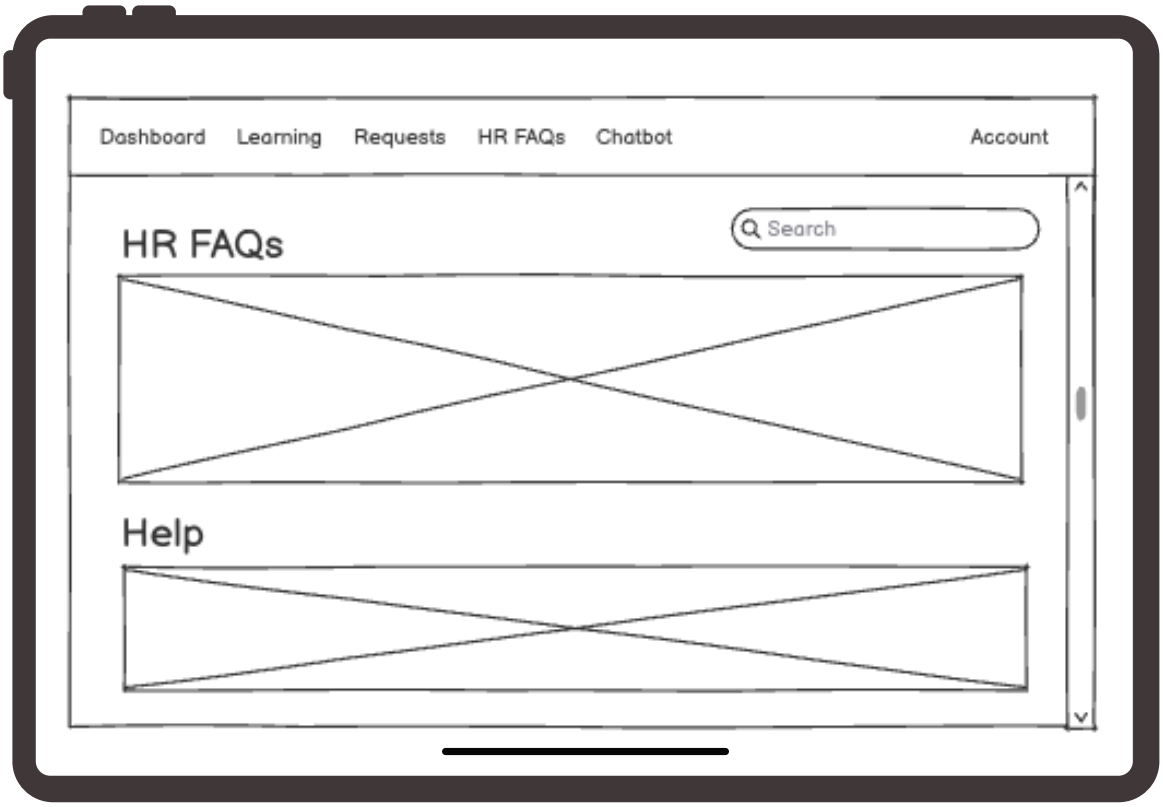

USER WORKFLOW
LOGIN
REVIEW
PICK EMPLOYEE
PERFORMANCE REVIEW

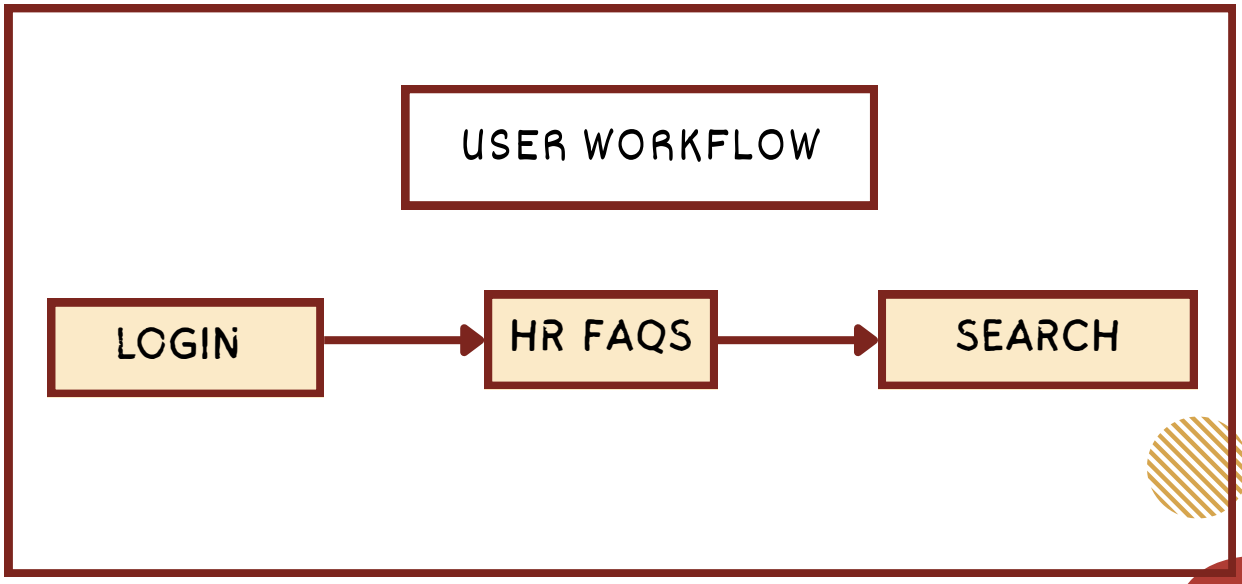
Storyboard with User Stories – Employee

Secondary User - Scenario 1 - Browsing HR-related FAQs



Scenario 1 Solution - Optimised HR-related FAQs section

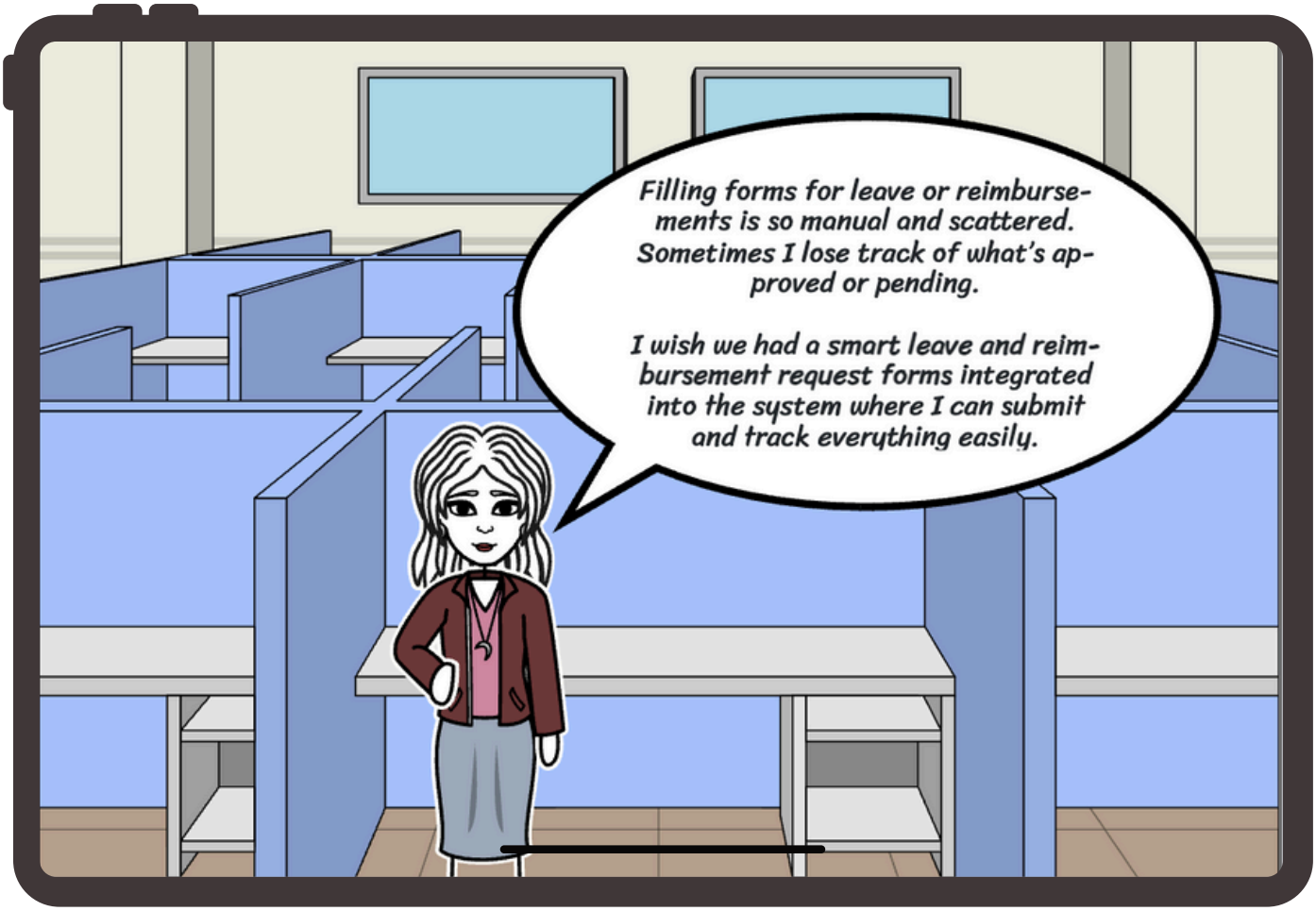




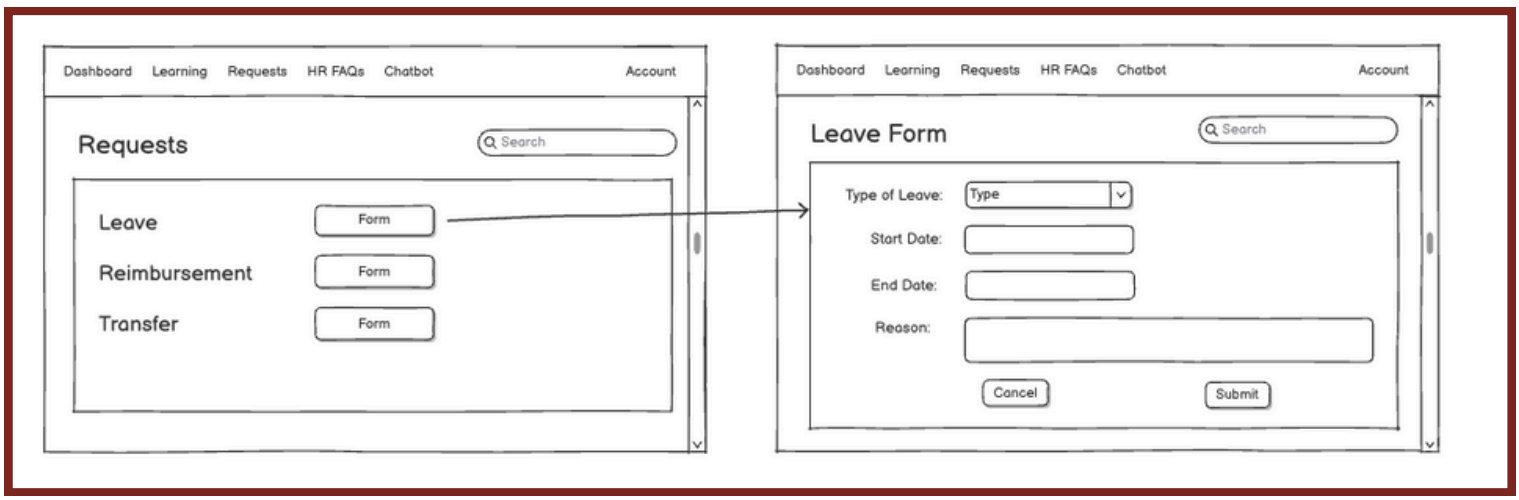
Scenario 1 Solution - Optimised HR-related FAQs section

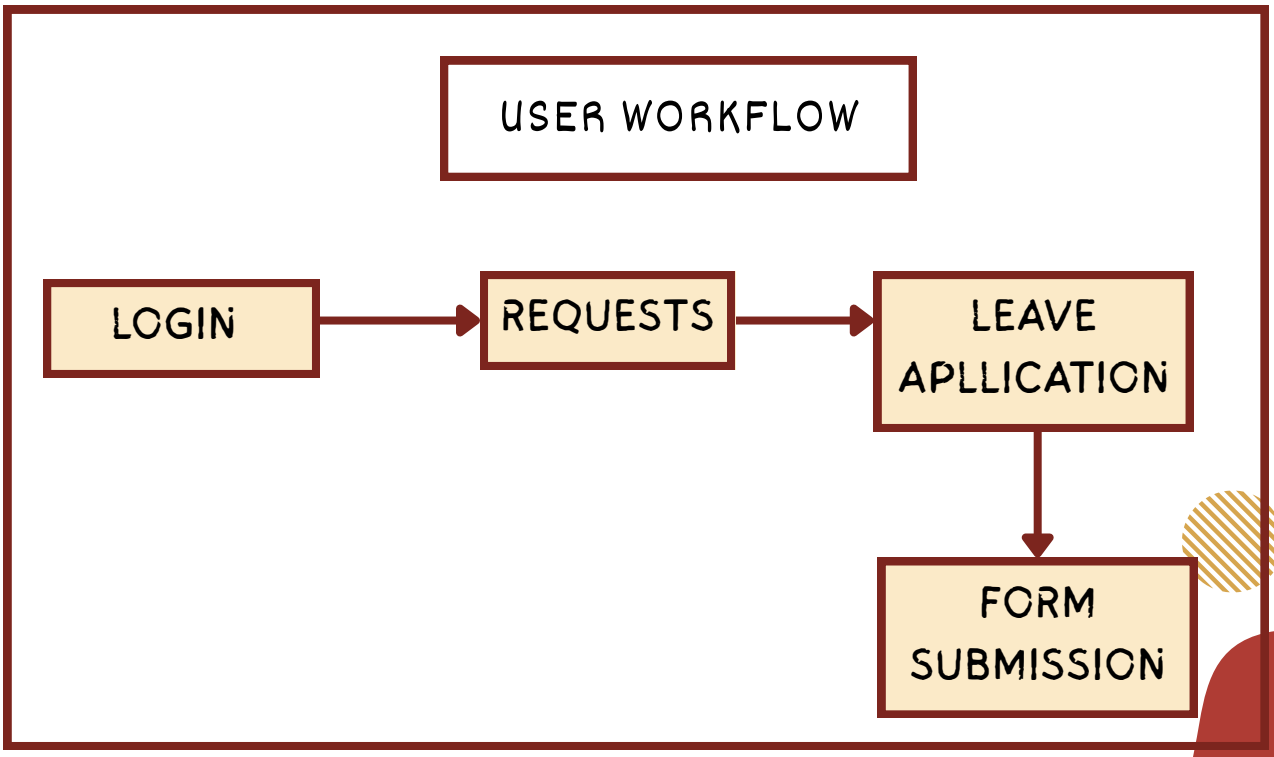
Storyboard with User Stories – Employee

Secondary User - Scenario 2 - Leave Approval Management

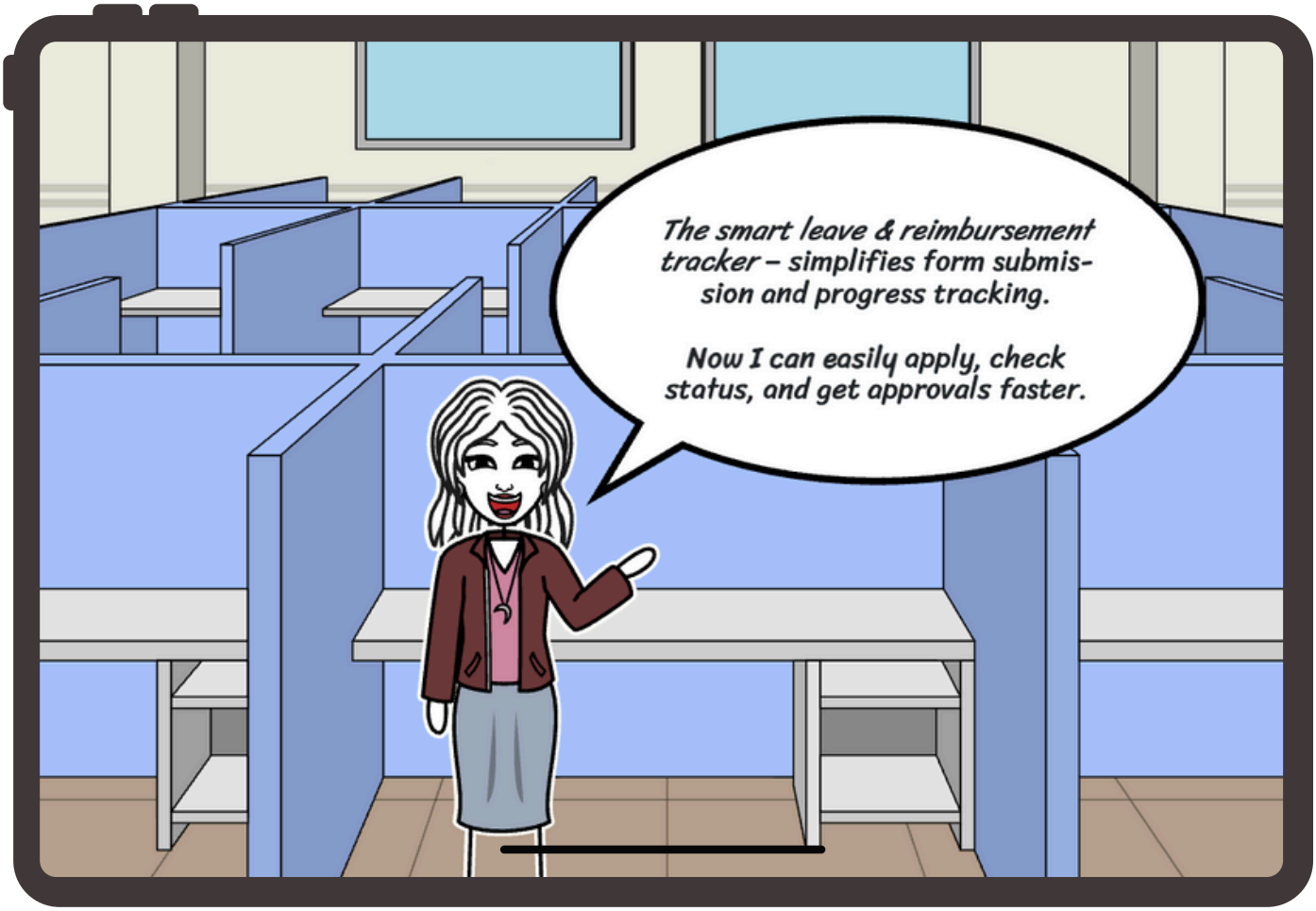


Scenario 2 - Solution - Form-based Leave Approval System



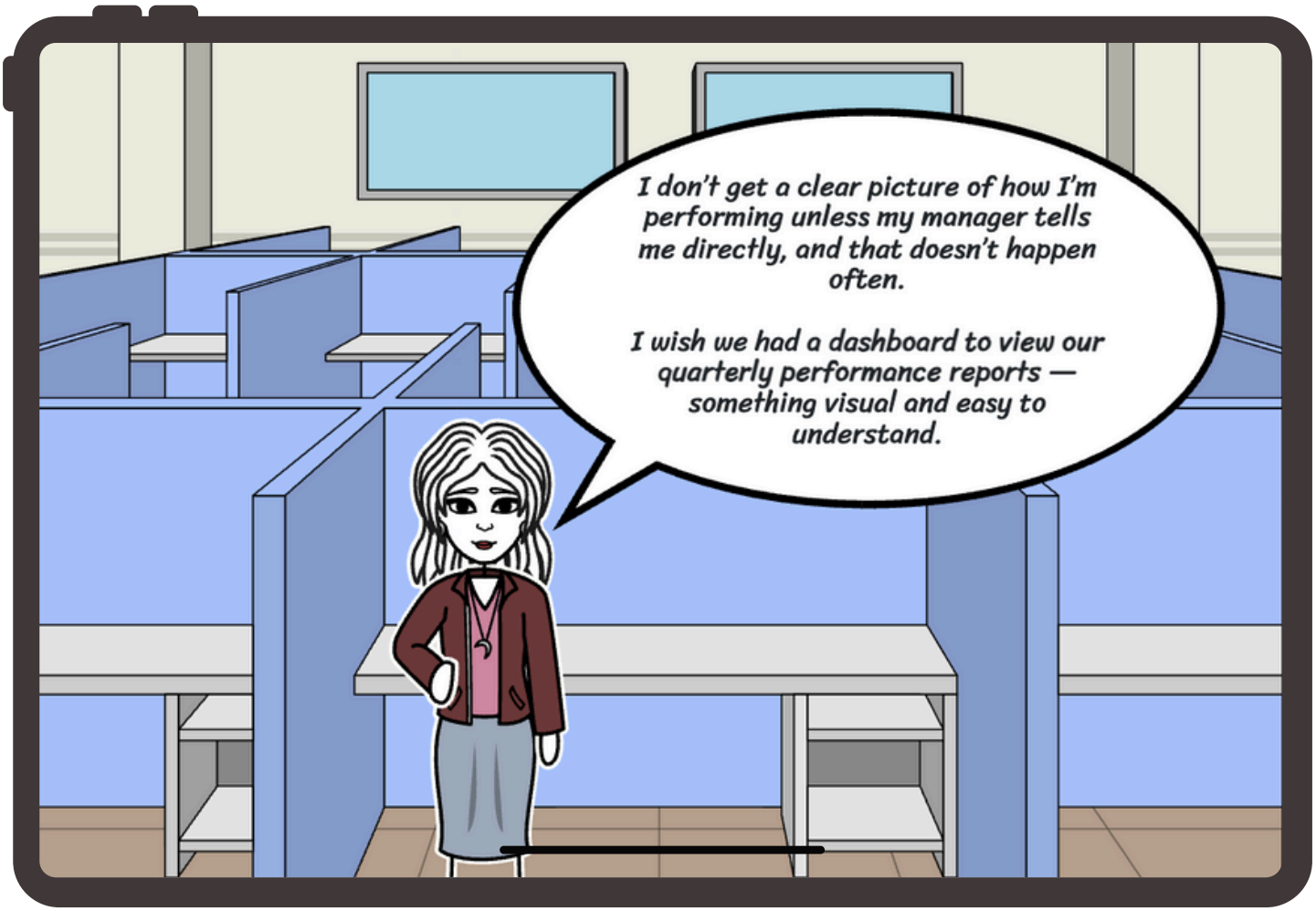


Scenario 2 - Solution - Form-based Leave Approval System

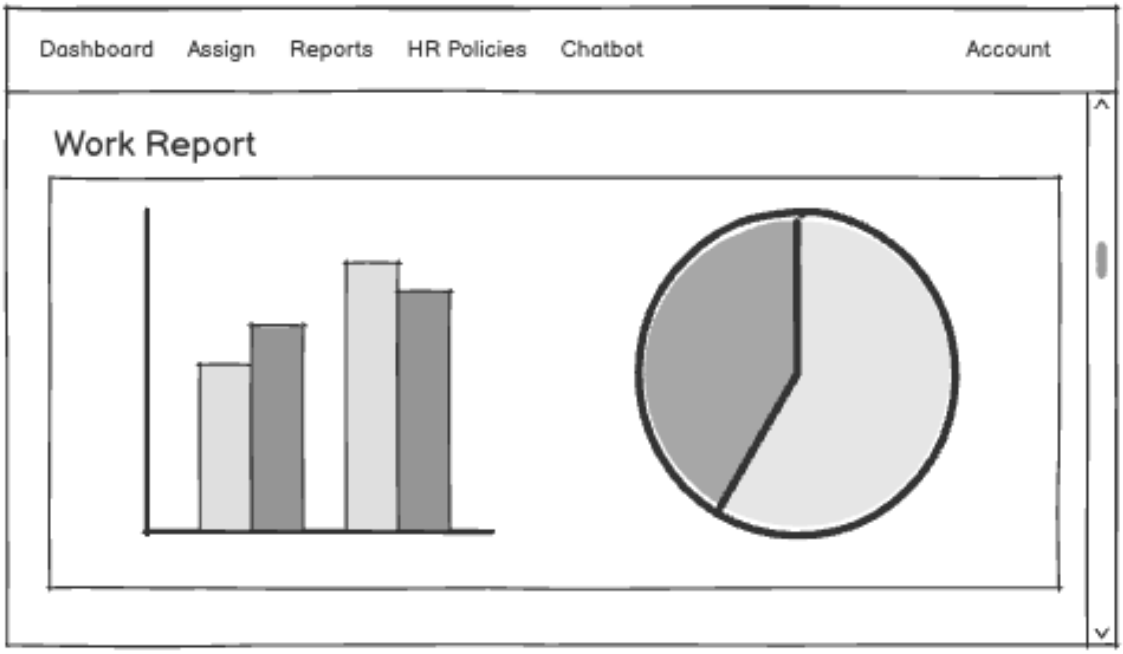


Storyboard with User Stories – Employee

Secondary User - Scenario 3 - Quarterly Performance Report



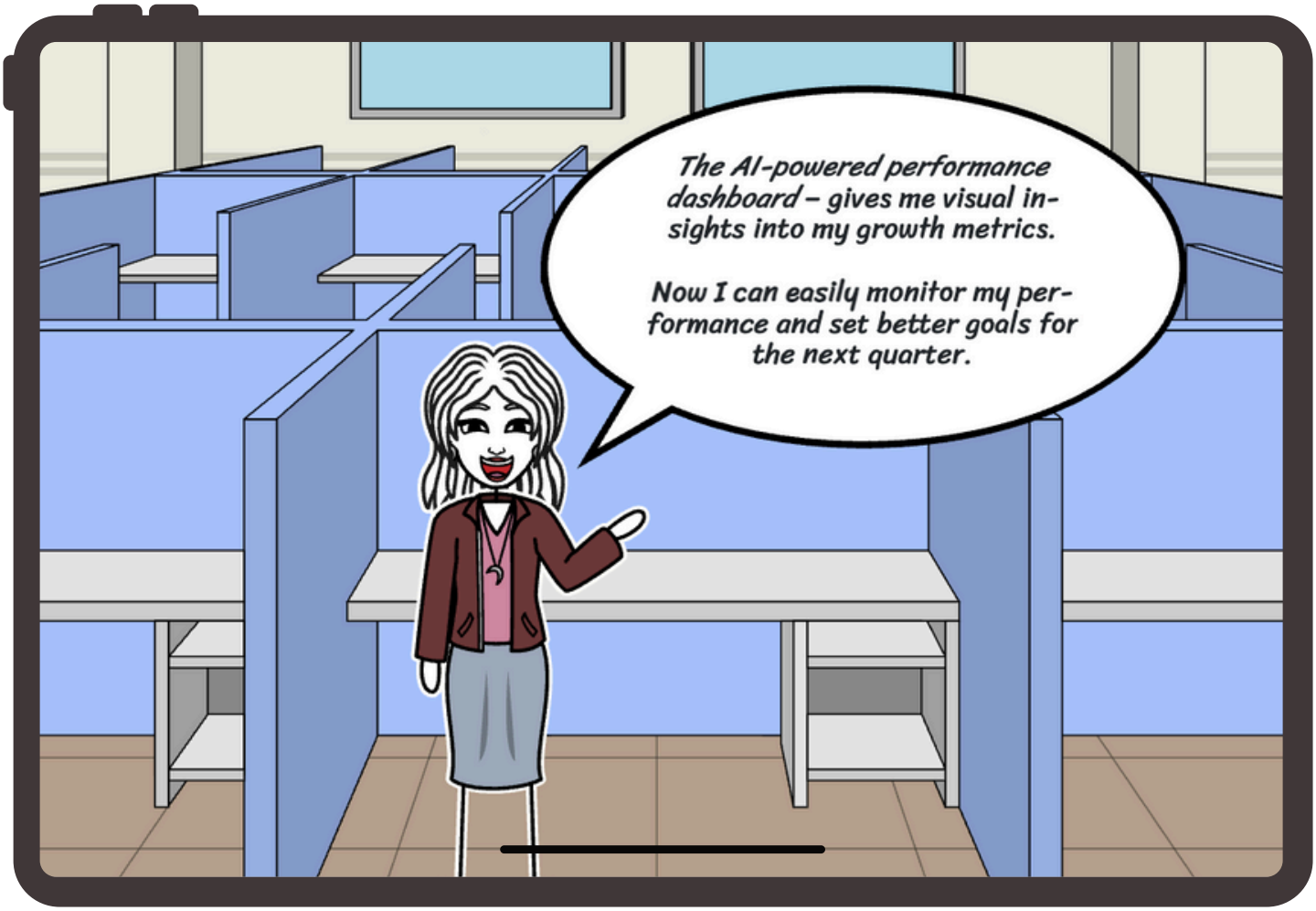
Scenario 3 - Solution - Quarterly Performance Report



USER WORKFLOW

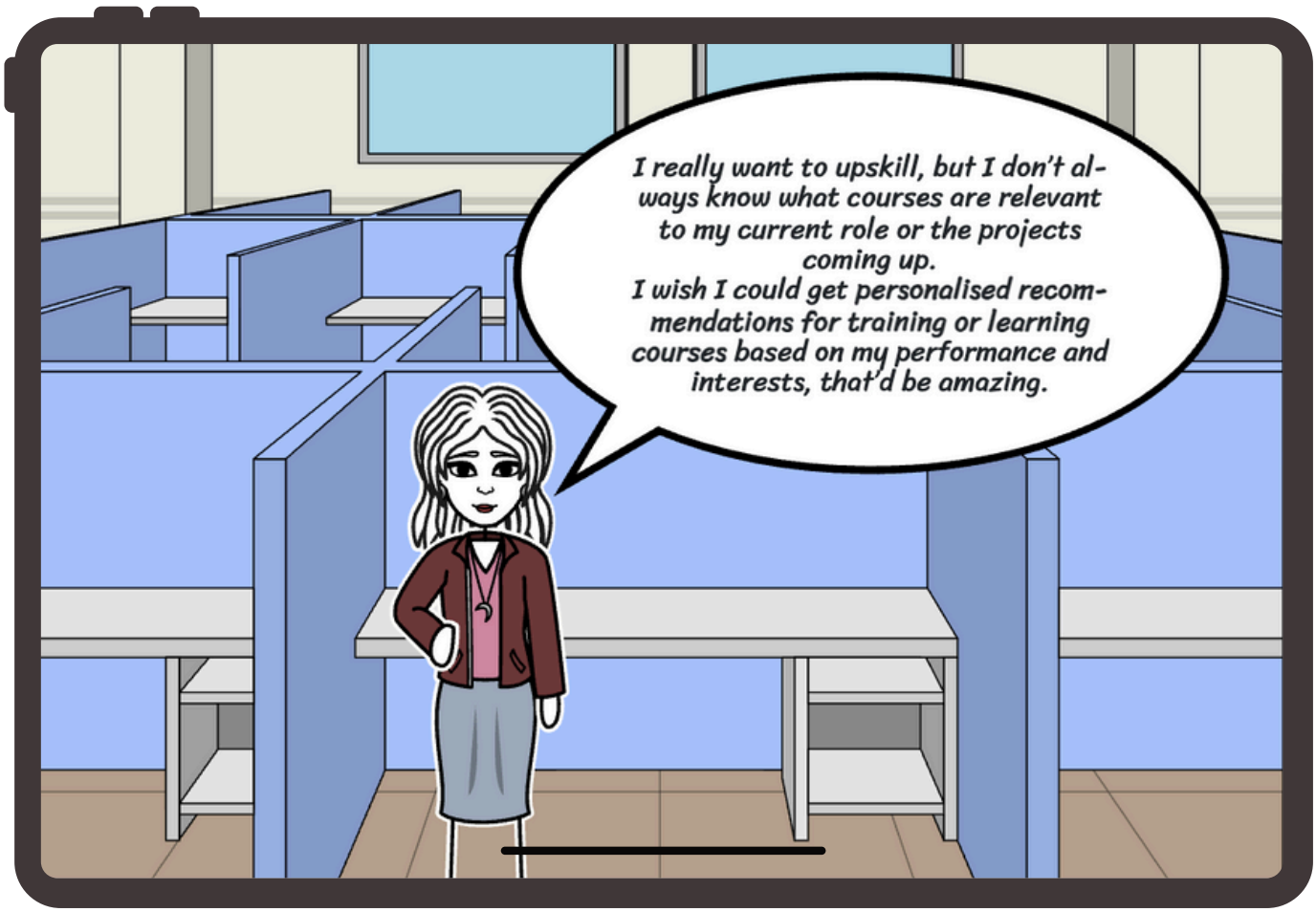
LOGIN → REPORTS → PROJECT WISE REPORTS

Scenario 3 - Solution - Quarterly Performance Report

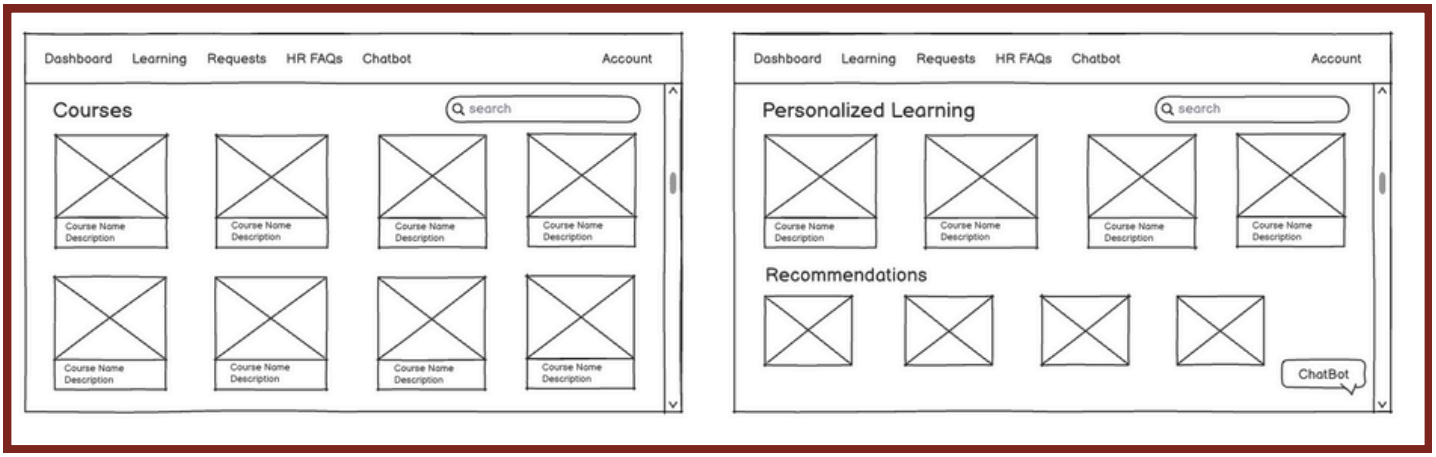


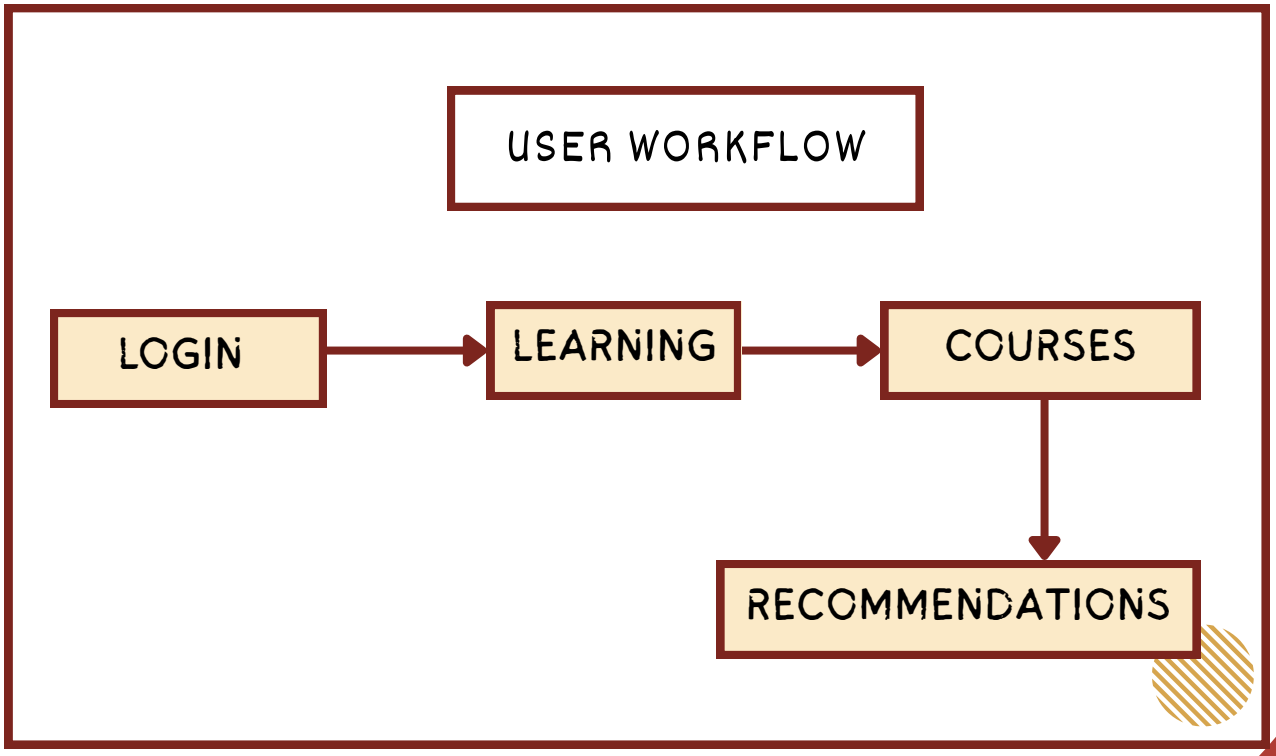
Storyboard with User Stories – Employee

Secondary User - Scenario 4 - Personalised Skill Improvement Report

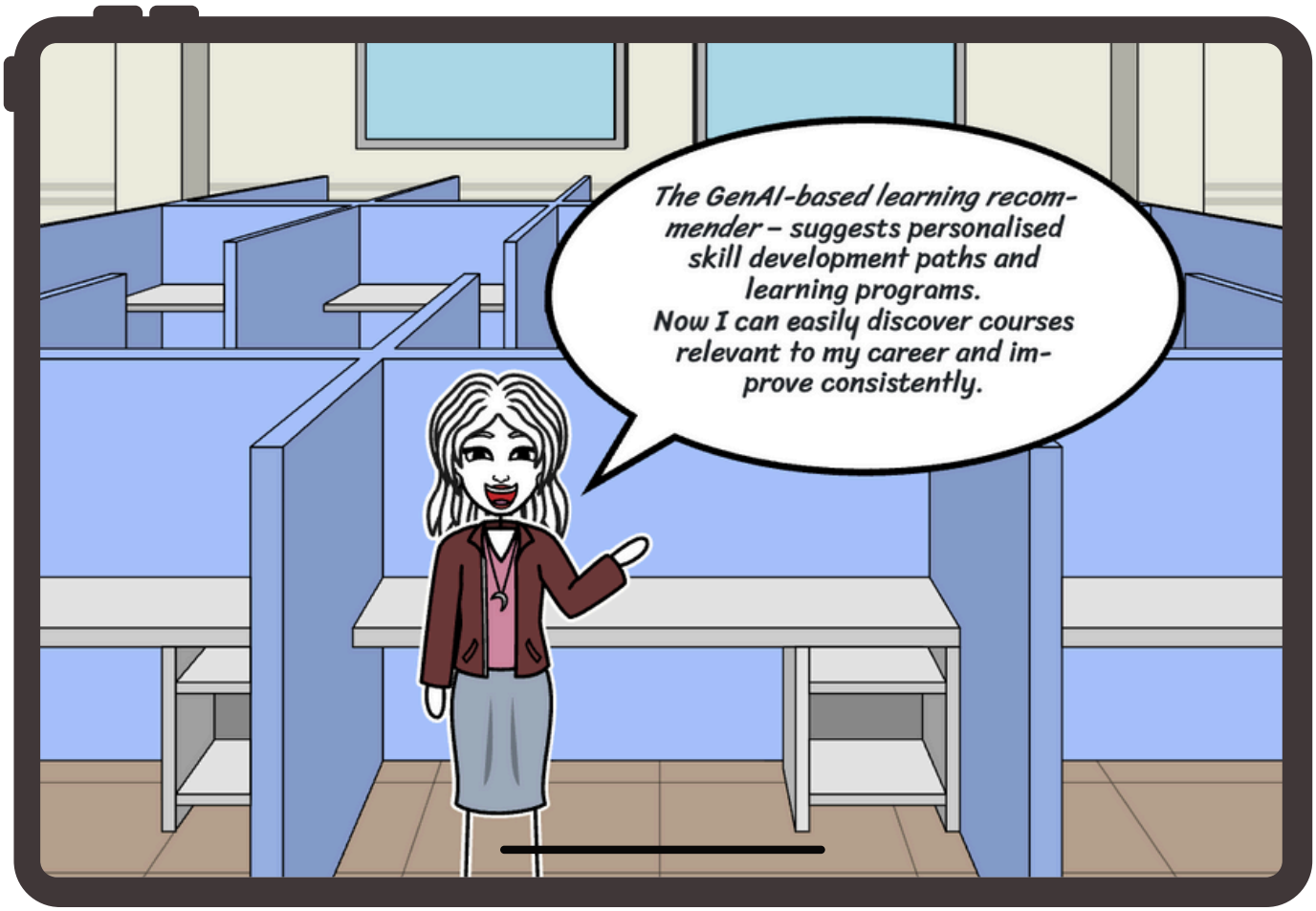


Scenario 4 - Solution - Personalised Skill Improvement Report



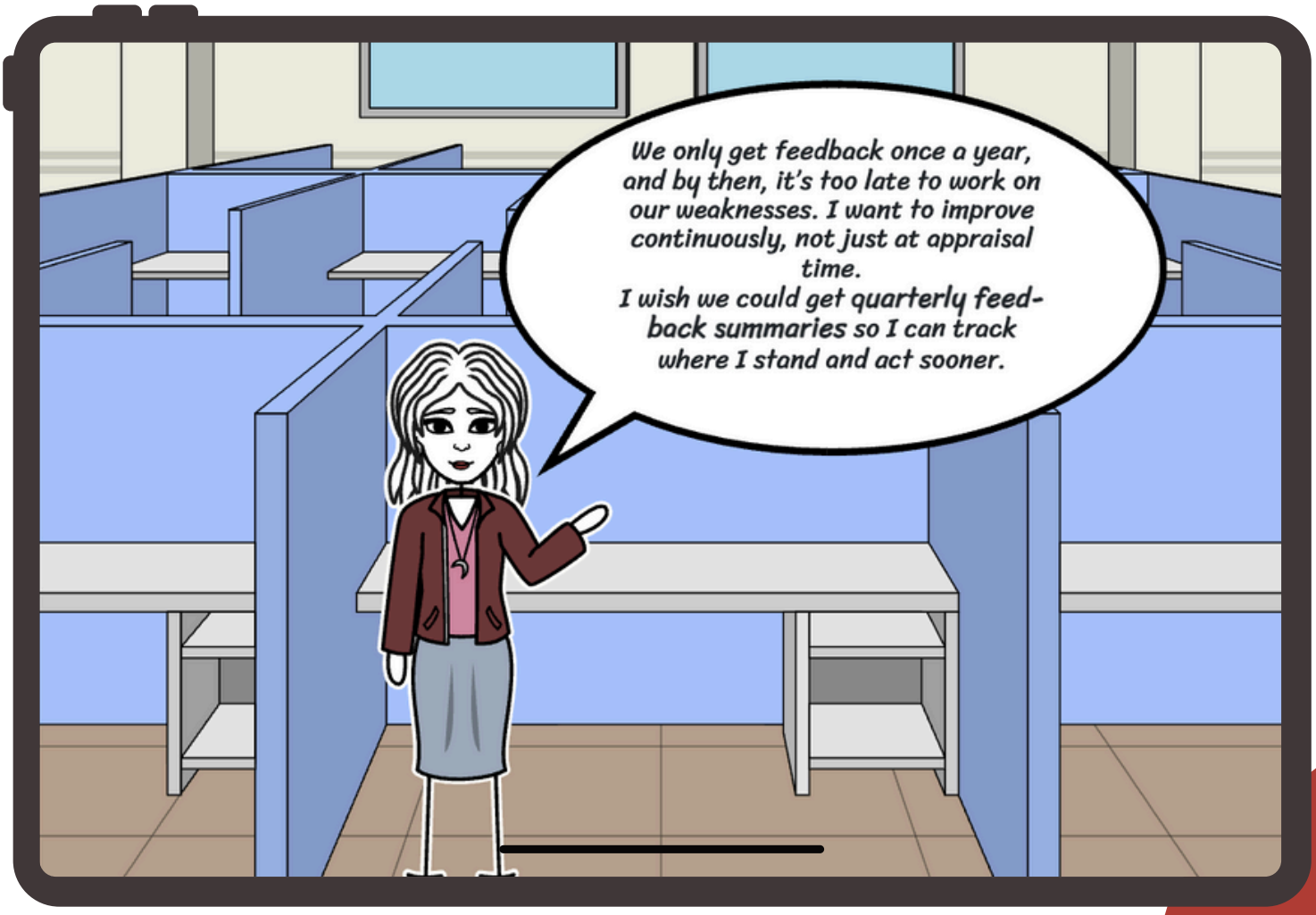


Scenario 4 - Solution - Personalised Skill Improvement Report

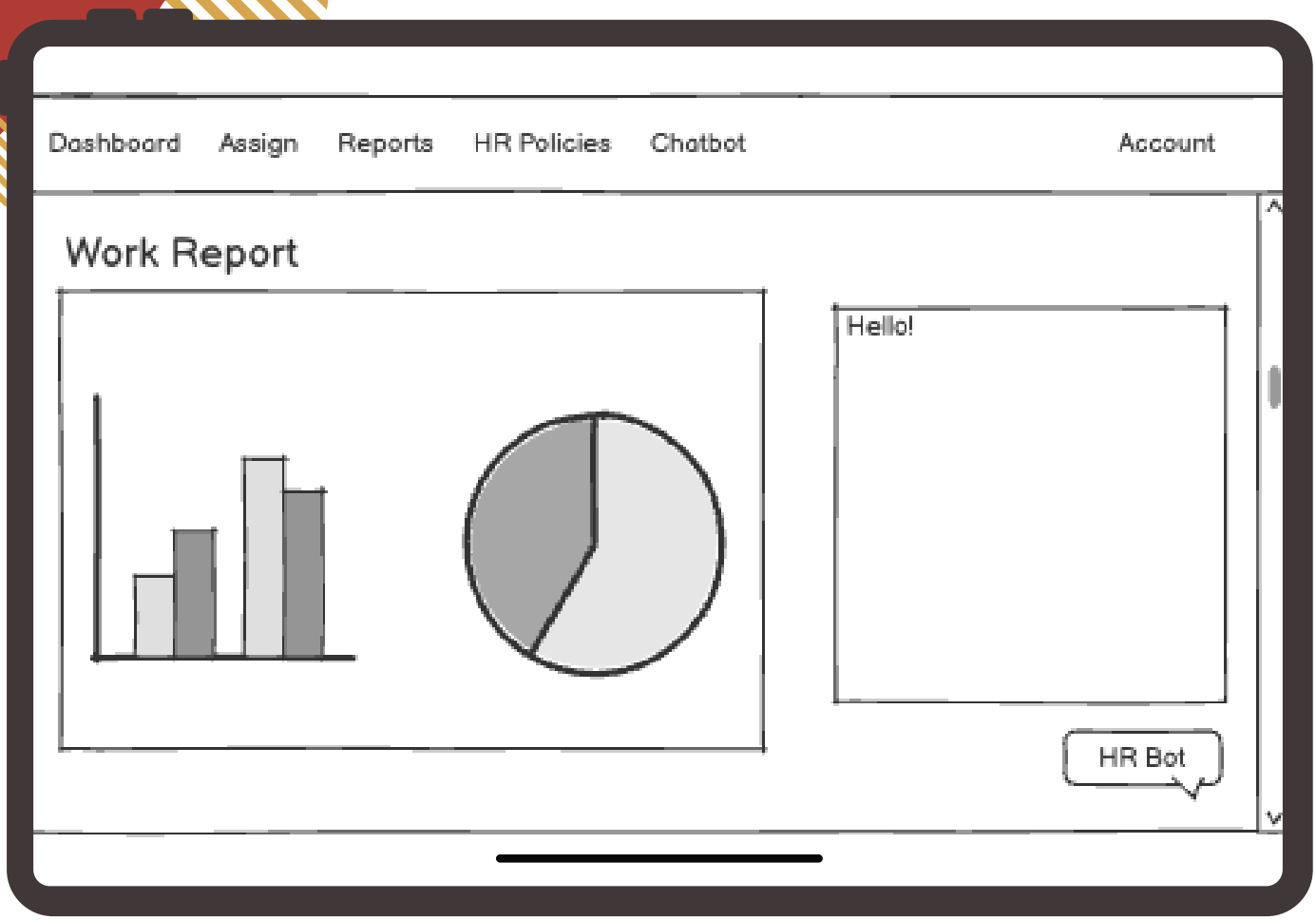


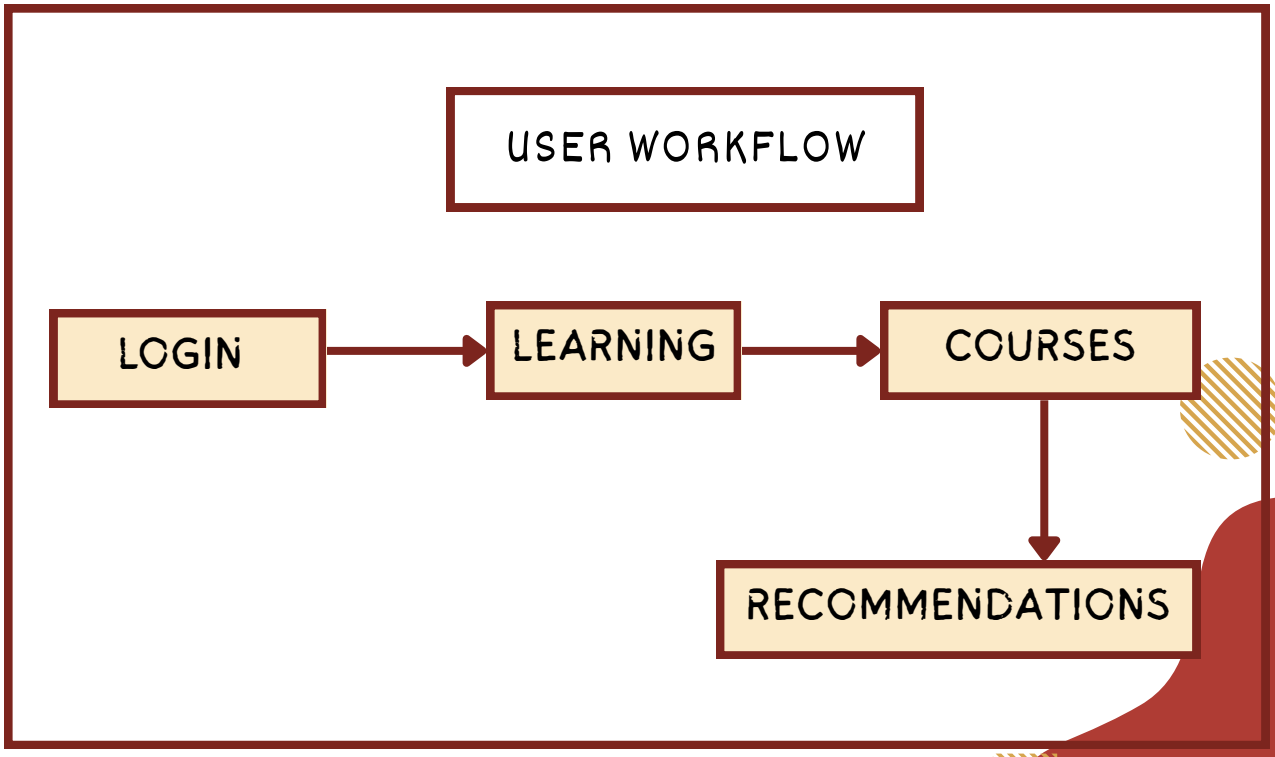
Storyboard with User Stories – Employee

Secondary User - Scenario 5 - Personalised Quarterly Feedback

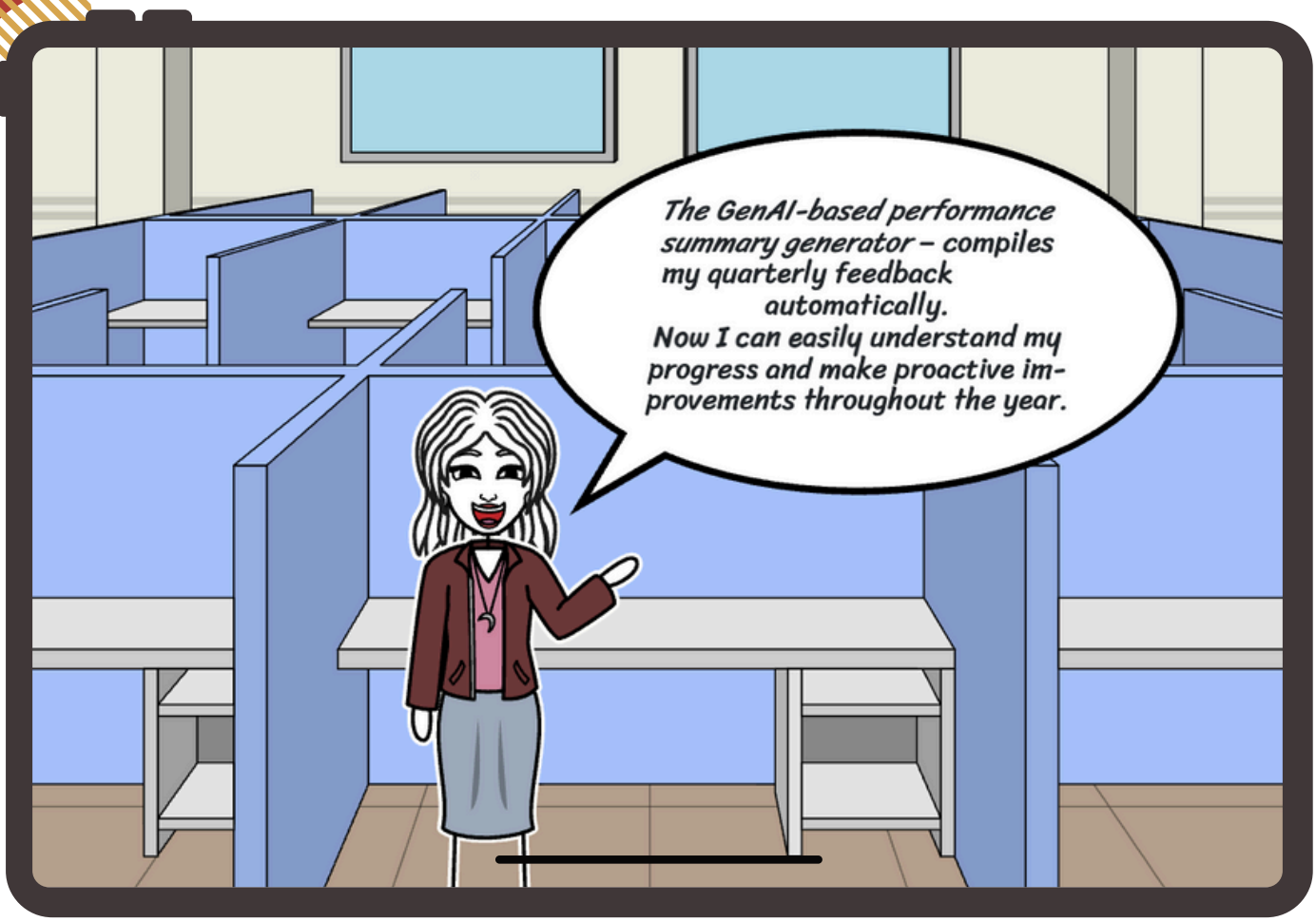


Scenario 5 - Solution - Personalised Quarterly Feedback



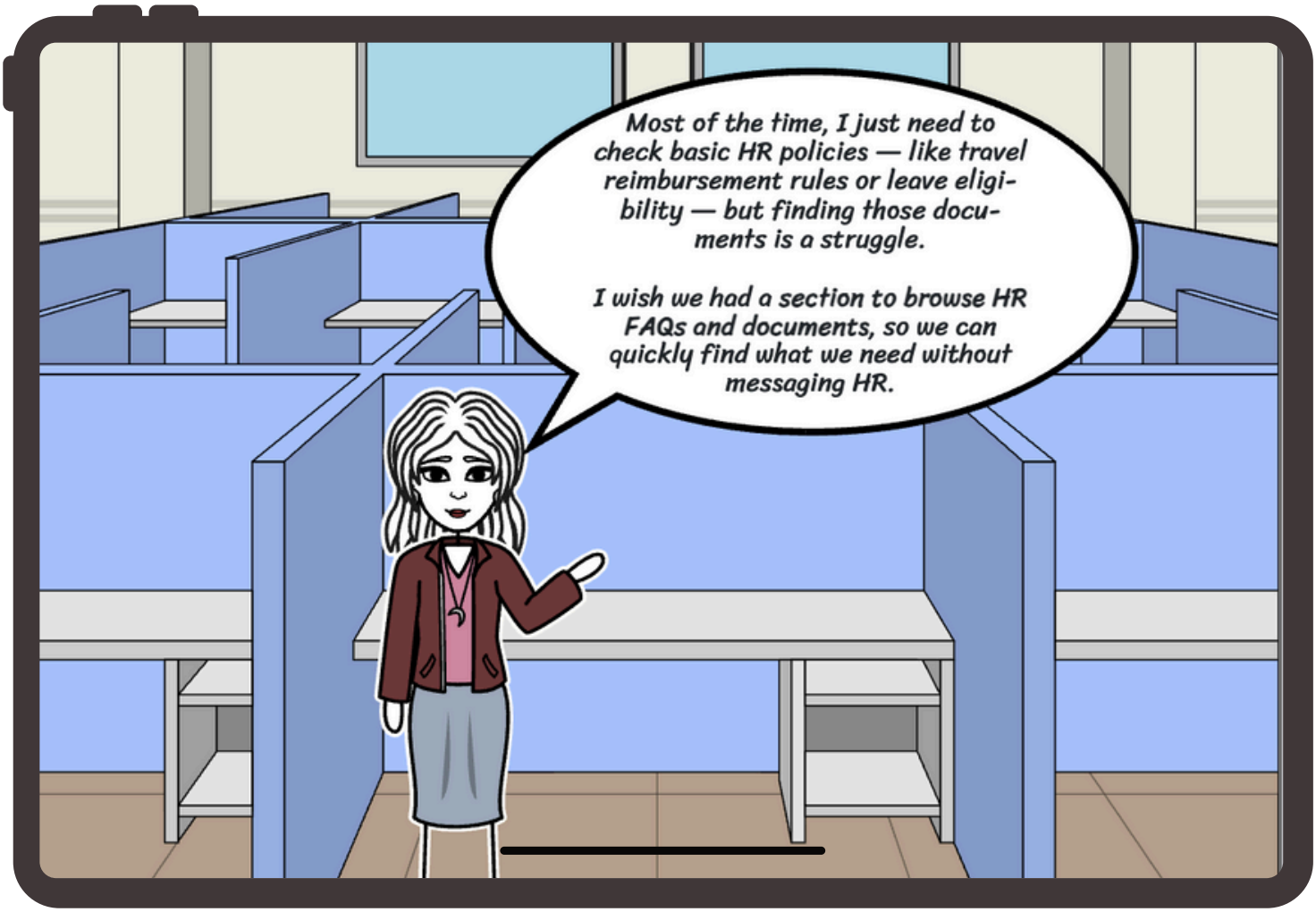


Scenario 5 - Solution - Personalised Quarterly Feedback

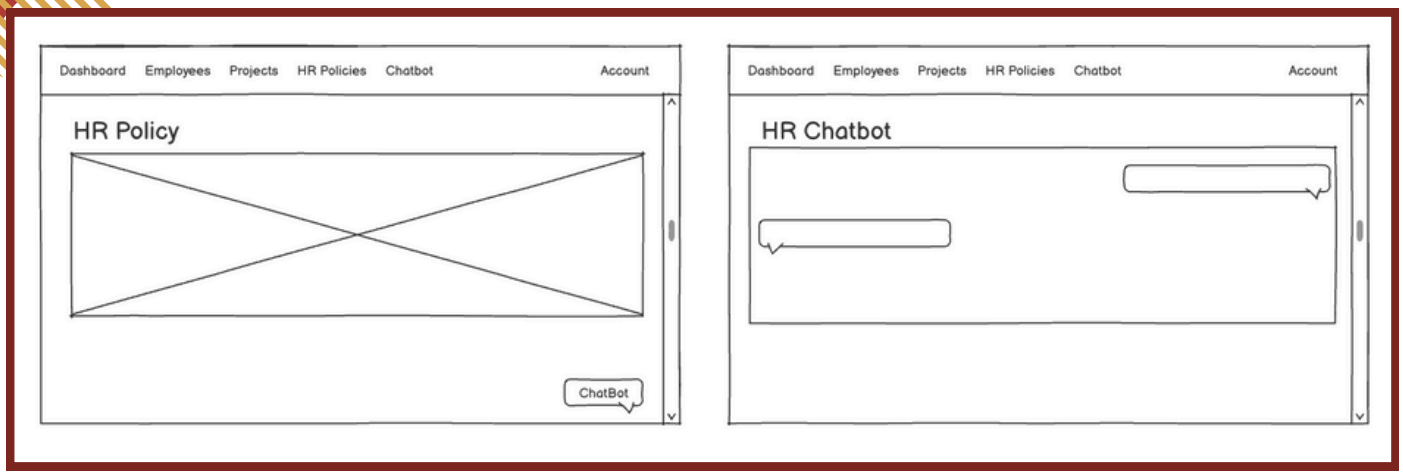


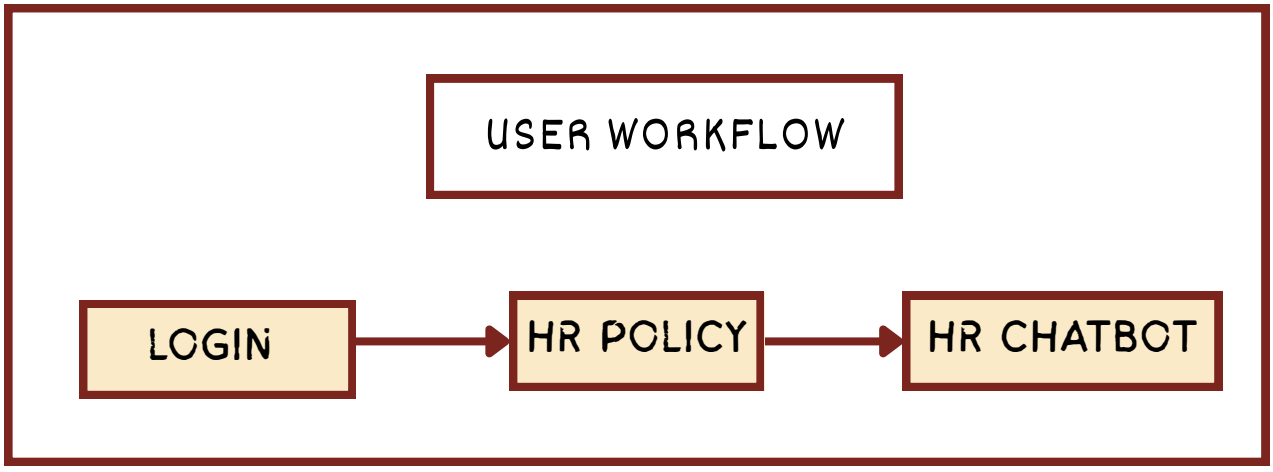
Storyboard with User Stories – Employee

Secondary User - Scenario 6 - HR-related Query ChatBot

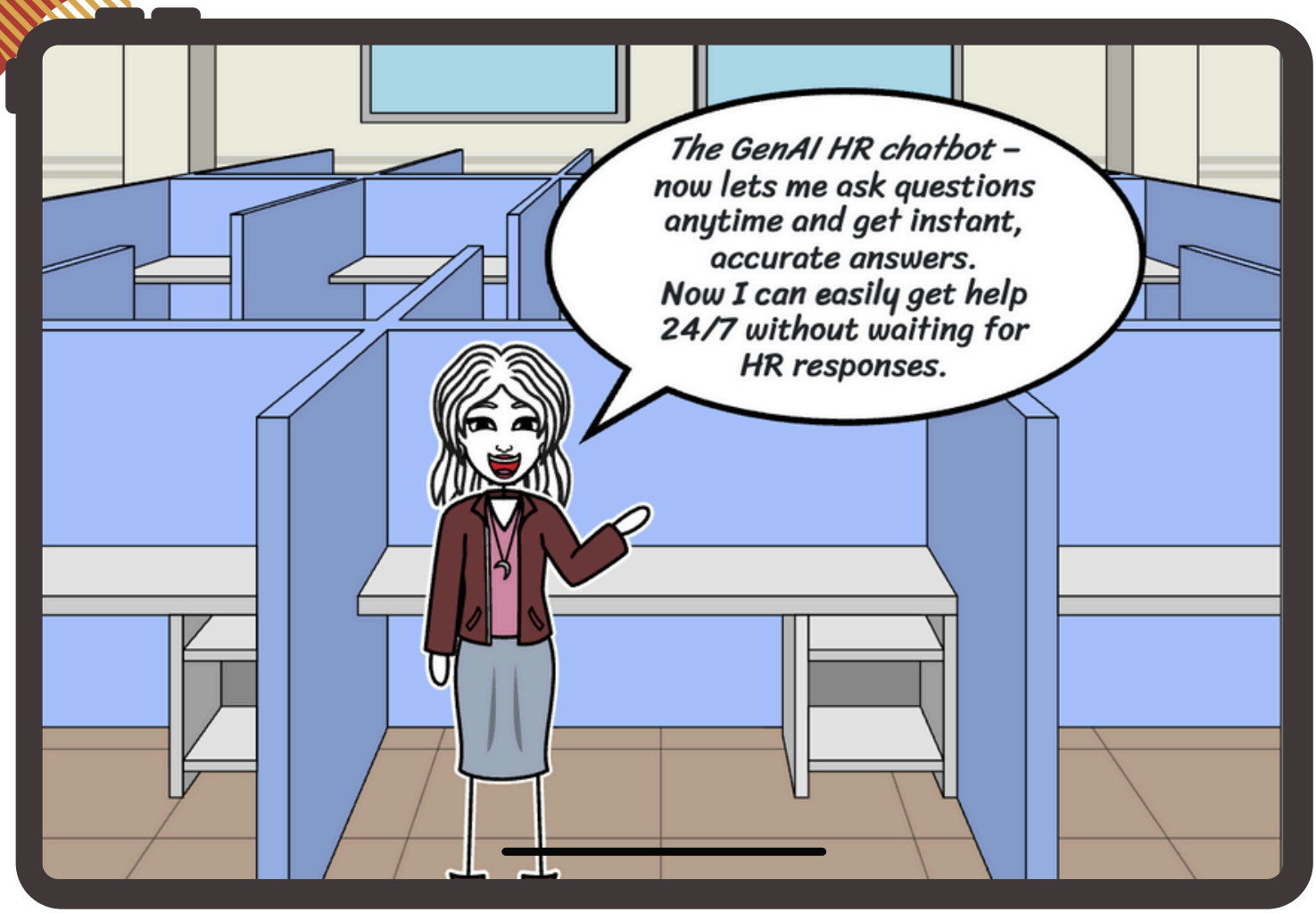


Scenario 6 - Solution - HR-related Query ChatBot



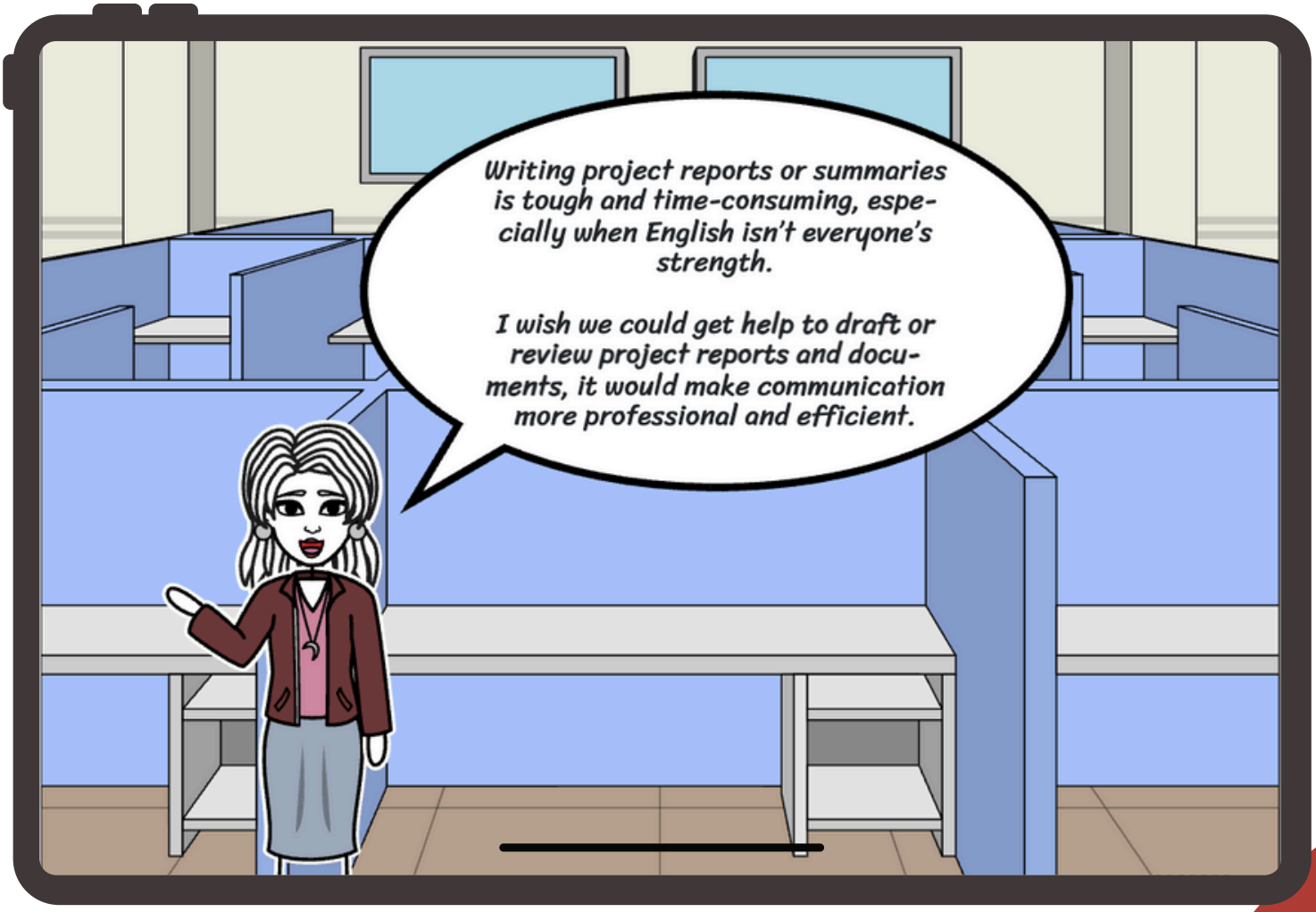


Scenario 6 - Solution - HR-related Query ChatBot

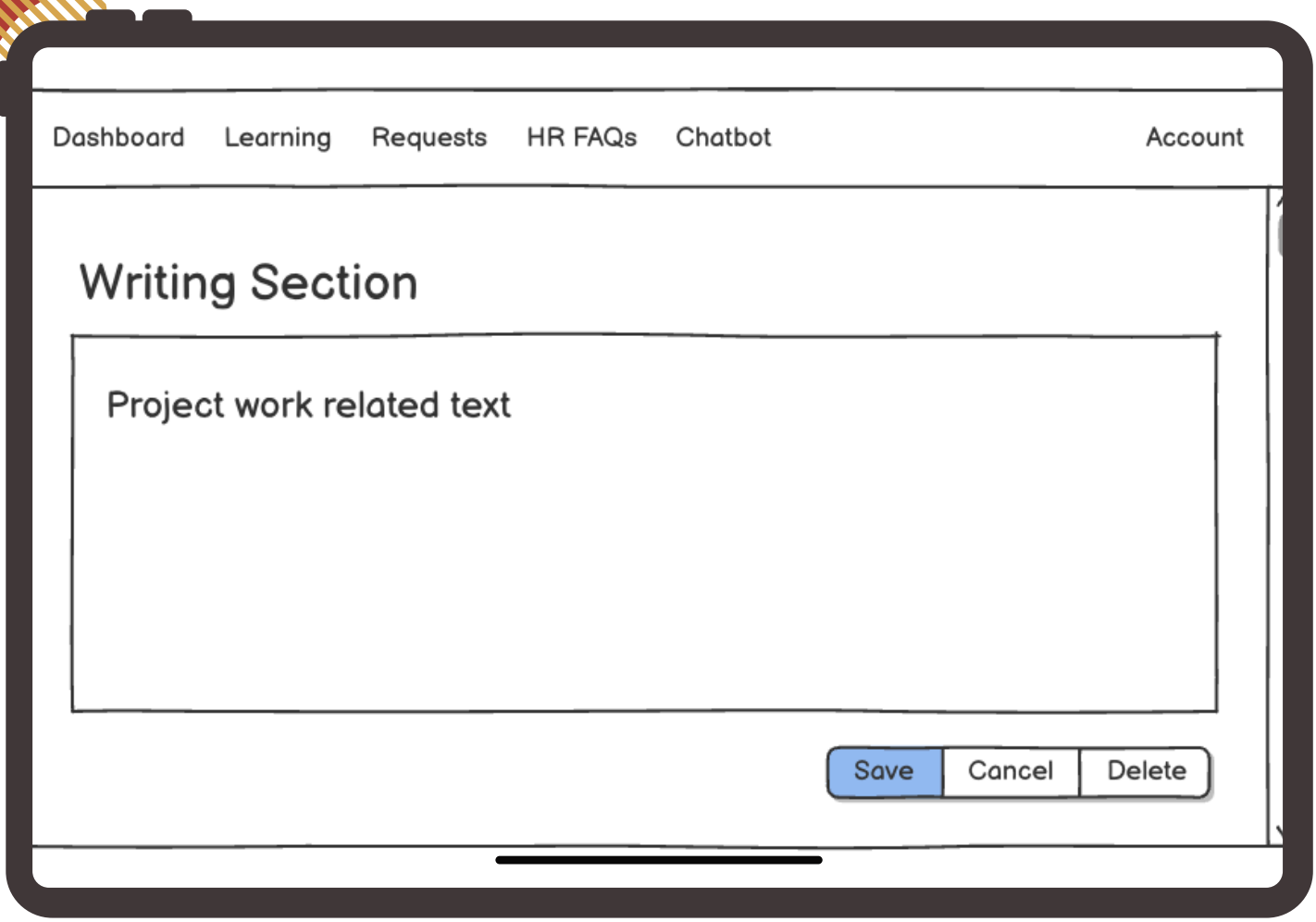


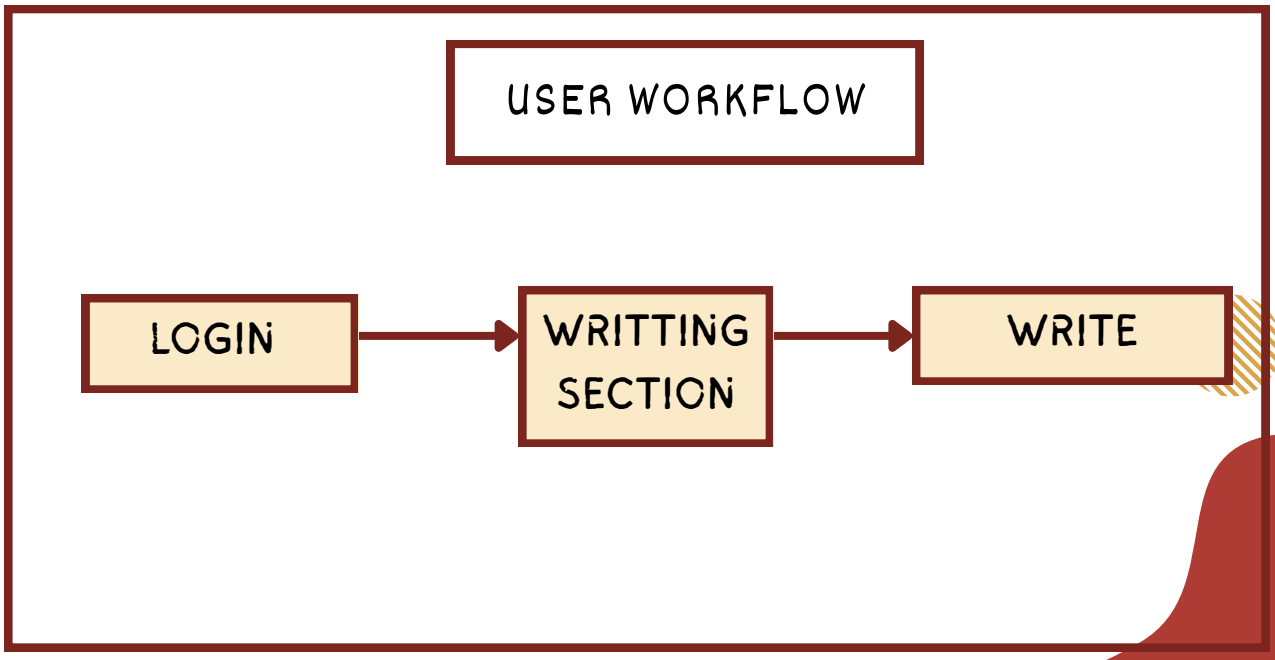
Storyboard with User Stories – Employee

Secondary User - Scenario 7 - Writing Assistant

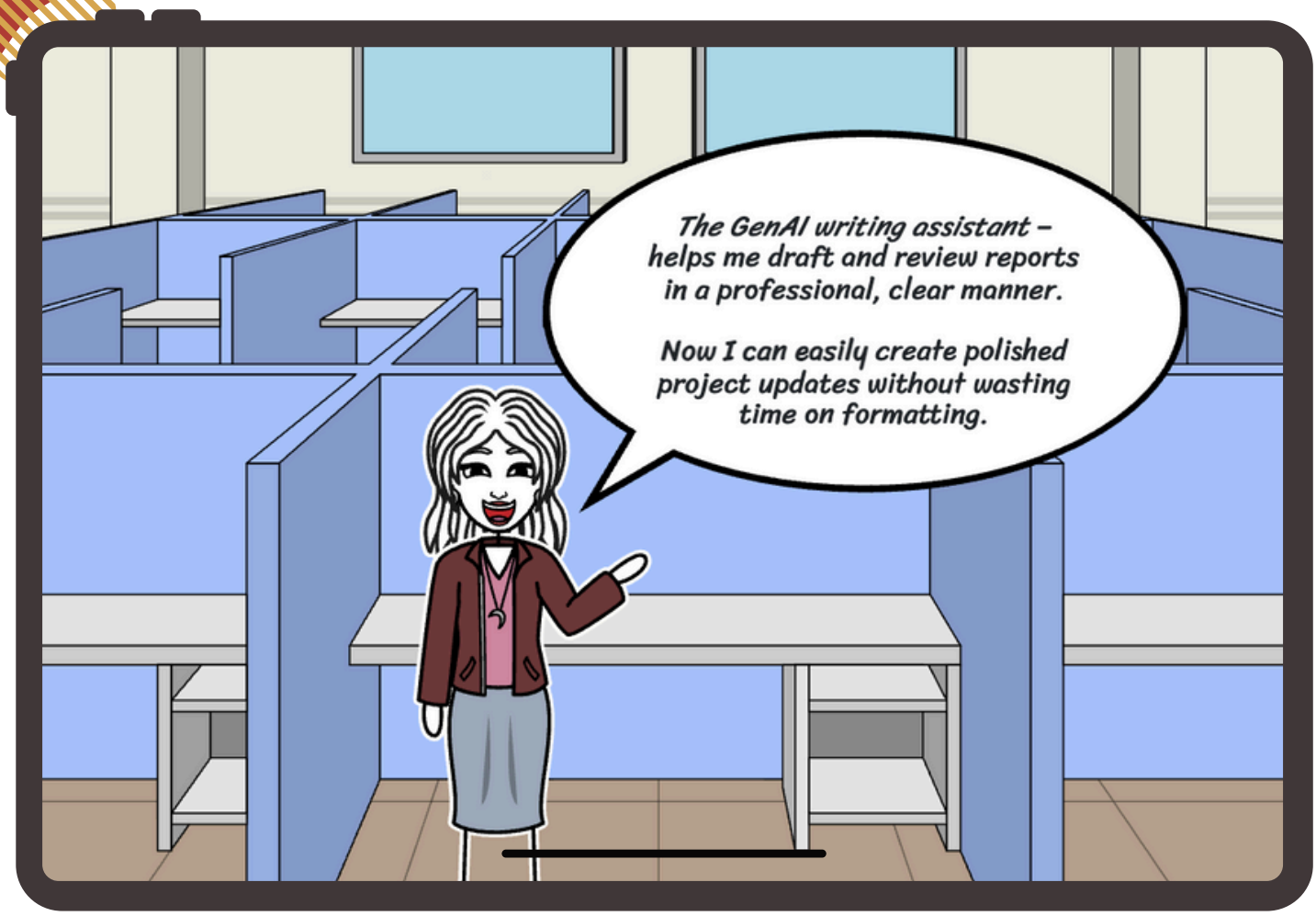


Scenario 7 - Solution - Writing Assistant

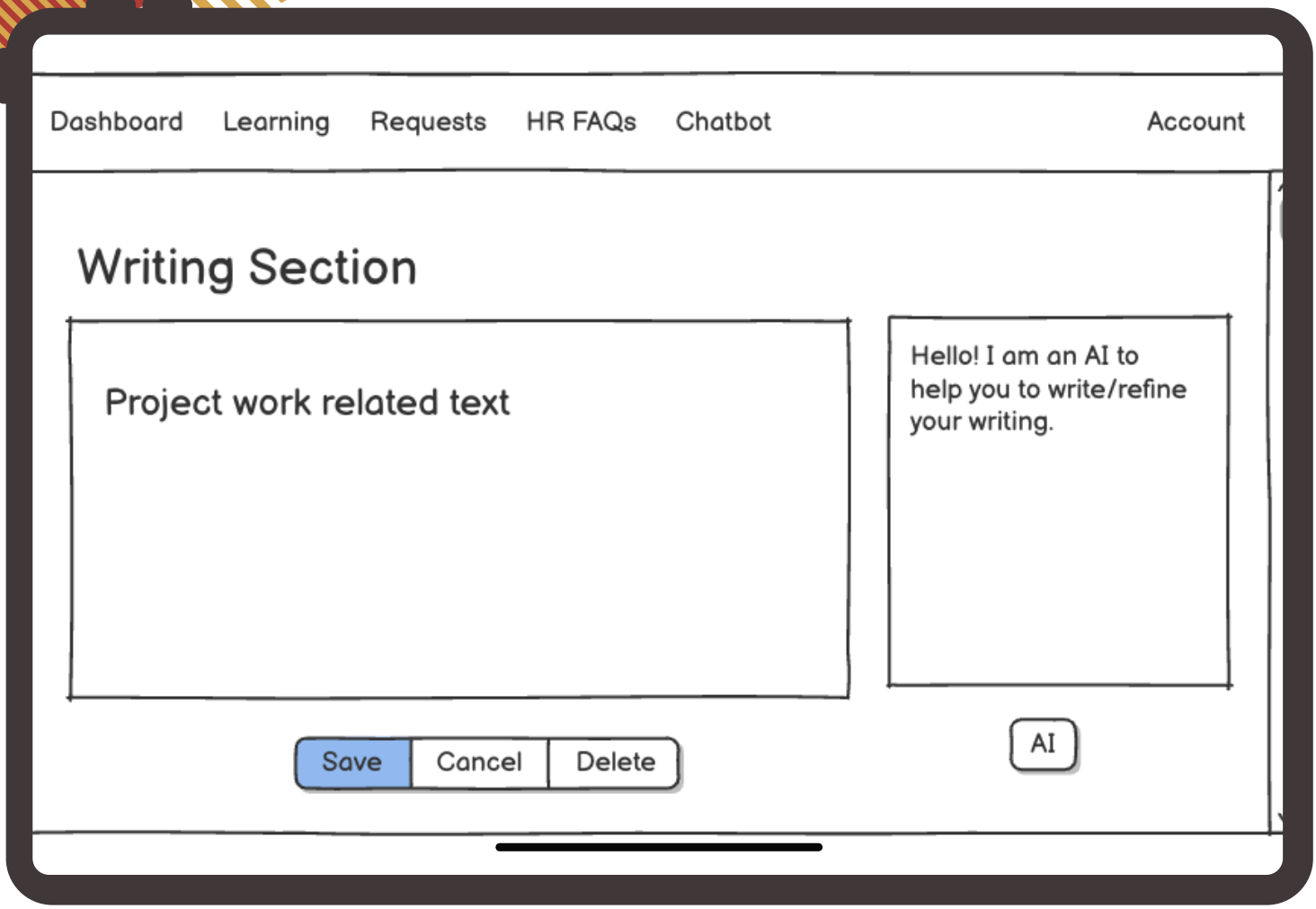


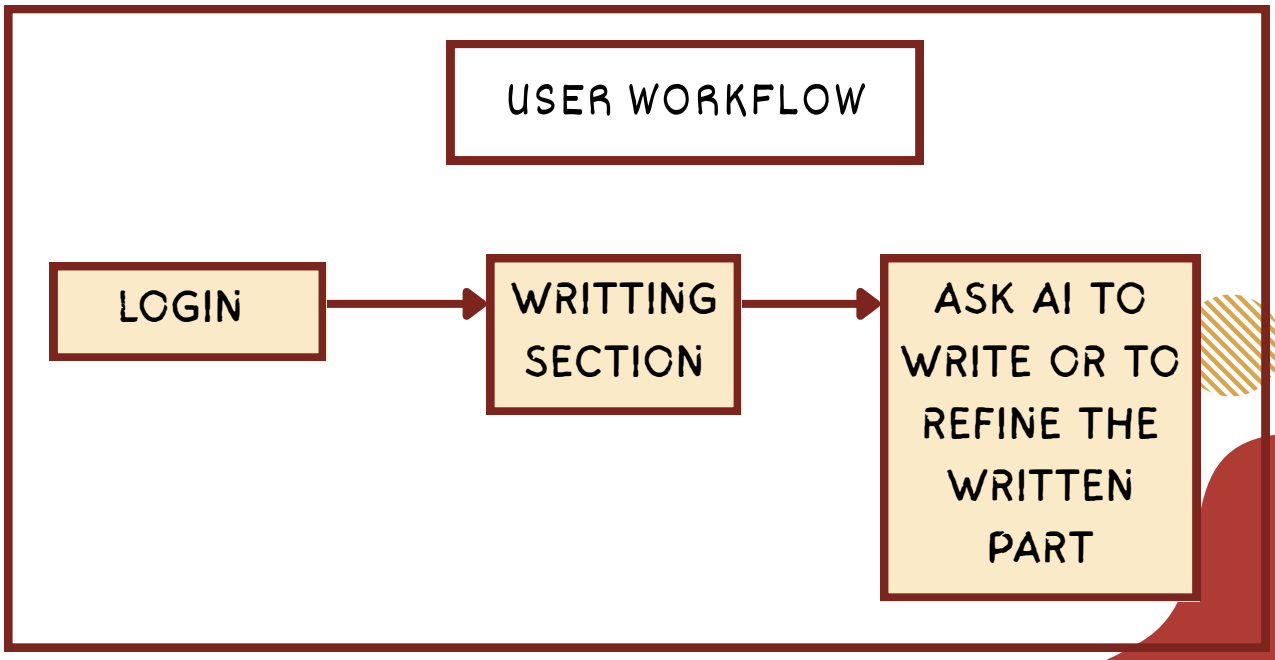


Scenario 7 - Solution - Writing Assistant



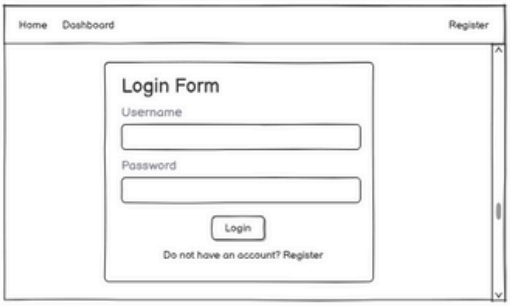
Scenario 7 - Solution - Writing Assistant

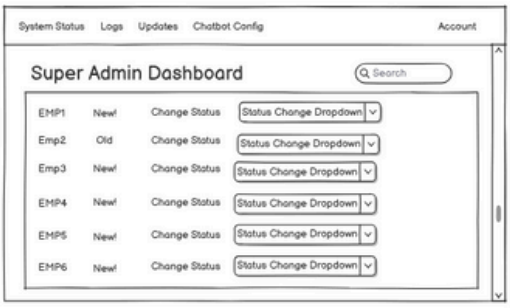


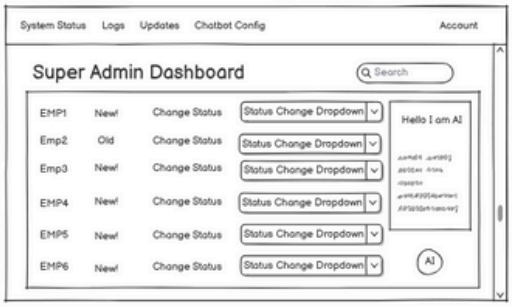


Tertiary User - Scenario 1 - Developer

GenAI comes in...

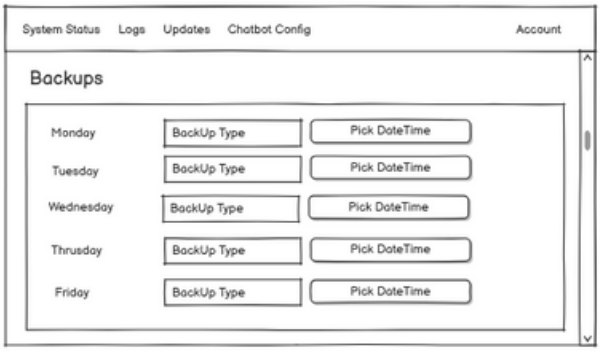






Tertiary User - Scenario 2 - Developer

Home
Dashboard
Register
Login Form
Username
Password
Login
Do not have an account? Register


System Status
Logs
Updates
Chatbot Config
Account
Backups
Monday
Tuesday
Wednesday
Thrusday
Friday
BackUp Type
Pick DateTime

By constantly auditing logs, validating data, and coordinating with developers, I make sure every module stays secure, accurate, and reliable for users.

Centralized monitoring tool comes in...

Client Meeting Links

Name	Roll No.	Interview Links
DEON LEVON DMELLO	21f3002473	2025-10-16 23-07-26.mp4
TUMMALA NAVEEN	22f3002759	SE Interview.mp4
SAYAN BHOWMICK	22f3001657	22f3001657 - Interaction Video Full.mp4
SOHAM CHAKRABORTY	23f1001783	Interview 1 Video.mp4
SAMPRITI RAHA	22f1000064	ofj-xiav-btb (2025-10-14 16:27 GMT+5:30)
TELVIN VARGHESE	22f2001640	Vasu's SE Interview.m4a
D NARENDRAN	22f3002698	user interview narendran.mp4
AKBAR ALI	23f1002997	VID_20251016222141.mp4

Client Interview Video Combined

Table of Contents

1 PROJECT SCHEDULE

2 SCRUM MEETINGS

3 PROJECT SCHEDULING TOOLS

4 COMPONENTS DESIGN

5 SOFTWARE DESIGN

6 USER INTERFACE

Project Schedule

SPRINTS SCHEDULE			
Sprint No.	Task	Start Date	End Date
1	User Identification and Requirement Analysis	Sep 19, 2025	Sep 28, 2025
2	User Interface and Functionalities	29 Sep 2025	19 Oct 2025
3	Designing the frontend	20 Oct 2025	Nov 2, 2025
4	Creating Backend API Endpoints	03 Nov 2025	09 Nov 2025
5	Integrating the Frontend & Backend	10 Nov 2025	16 Nov 2025
6	Testing	17 Nov 2025	30 Nov 2025

Scrum Meeting

Details:

Scrum Meetings Schedules Every Wednesday and Saturday (9:00 PM onwards)
Additionally on Mondays, Sundays (as per requirement)

Mode:

Virtual

Attendees:

1. AKBAR ALI | 23f1002997
2. DEON LEVON DMELLO | 21f3002473
3. D NARENDRAN | 22f3002698
4. SAMPRITI RAHA | 22f1000064
5. SAYAN BHOWMICK | 22f3001657
6. SOHAM CHAKRABORTY | 23f1001783
7. TELVIN VARGHESE | 22f2001640
8. TUMMALA NAVEEN | 22f3002759

About the Sprints

Sprint 1
User Identification and Requirement Analysis

This began with the team introduction and then we sat further in the second meeting and then selected the sector of our app and then in the next meat, We identified all user categories (HR Managers, Project Managers, Employees, ,admin etc) and define their roles and interactions within the system.
After the identification of all types of users of the app we then check for the requirements on our end that we see and the fuctionalities that we need to provide to our Users.
Finalize and submit the Milestone 1 Report, consolidating the user identification and requirement documentation.
Dates: 19 September 2025 – 28 September 2025

Sprint 2
User Interface and Functionalities

For better understanding of the sector as well as the requirements of our users each of us conducted various type of user's Intervies. Develop detailed user stories following the SMART framework to capture functional and non-functional requirements effectively.
Design the user journey using storyboards and low-fidelity wireframes to represent workflows for each user type. Implement usability guidelines to ensure clarity and consistency in navigation.Map user stories to interface components to define how functionalities (e.g., project allocation, HR chatbot, performance feedback) will appear in the system.
Complete and submit the Milestone 2 Report, including finalized wireframes and user flow diagrams.
Dates: 29 September 2025 – 19 October 2025

About the Sprints

Sprint 3
Designing the Frontend

This is a frontend code focused sprint. We develop the major frontend pages using Vue.js, ensuring modular and reusable component architecture. Implement responsive layouts for HR Manager, Employee, and Project Manager dashboards.
Set up project routing and redirections between pages such as Login, Dashboard, Allocation, Chatbot, and Reports.
Document the project schedule, sprint plan, scrum meeting calendar, and create a Gantt chart and Kanban board for task tracking.
Complete and submit the Milestone 3 Report with screenshots of UI pages and planning tools.
Dates: 20 October 2025 - 02 November 2025

Sprint 4
Creating Backend API Endpoints

We are using Flask for backend. This is a core backend oriented and focusses on Designing and implementing RESTful APIs using Flask for essential modules such as user authentication, project allocation, chatbot responses, and performance tracking.
Document all endpoints with detailed specifications, including parameters, methods, and sample JSON responses.
Prepare and submit the Milestone 4 Report, showcasing endpoint documentation and code integration readiness.
Dates: 3 November 2025 – 9 November 2025

About the Sprints

Sprint 5
Integrating the Frontend & Backend

This one has the connectivity of the backend and frontend. We first do unit testing here of individual components and then we connect the Vue.js frontend with Flask backend APIs to enable real-time data flow. Ensure synchronisation of authentication tokens, role-based dashboards, and chatbot interactions.
Resolve CORS and routing issues, verify form submissions, and validate data exchange between modules.
Conduct internal reviews and prepare for functional testing.
Dates: 10 November 2025 – 16 November 2025

Sprint 6
Testing

This is the basic testing based sprint here the whole coding part is done and the app is ready here We perform integration, and usability testing across all components to ensure functional correctness and performance.
Use test cases mapped to each user story to validate API reliability, UI responsiveness, and chatbot accuracy.
Document testing results and bug fixes, and prepare the Milestone 6 Report for submission.
Dates: 17 November 2025 – 30 November 2025

Scrum Meeting Details

Scrum Meeting Details conducted for Sprint 1

Started with a Sprint 1, the team held multiple scrum meetings to establish the project foundation and align on objectives. The initial meeting began with team introductions, discussion of individual roles, and brainstorming of potential project domains. After a collaborative evaluation of ideas, the team selected the HR and Project Allocation sector for the app, later titled “Sync’em.”

In the subsequent meetings, the focus shifted toward identifying and defining all user categories — HR Managers, Project Managers, Employees, Department Leads, and Administrators — and analyzing their roles and interactions within the system. Each team member contributed by researching and documenting user requirements and expected functionalities.

The team then conducted a collective review to verify that the user identification covered every functional area of the application. Following this, members worked together to determine key functional and non-functional requirements, emphasizing usability, scalability, and security.

Scrum Meeting Details

Scrum Meeting Details conducted for Sprint 2

During Sprint 2, the team focused on translating the identified user requirements and stories into a structured user interface (UI) flow and defining key functional components of the application.
The first scrum meeting for this sprint began with a review of the finalized requirements from Sprint 1. And we focused on the interviews for better understanding of the Sector.
Finally, the team collaborated to draft and refine user stories for all identified users, ensuring that they adhered to SMART guidelines (Specific, Measurable, Achievable, Relevant, and Time-bound).
Team members collaboratively mapped each user story to corresponding UI elements and screens. Based on this mapping, the team discussed and agreed upon a consistent navigation structure to ensure a smooth user journey for all user types – including HR Managers, Project Managers, and Employees.
In the following meeting, members presented their initial low-fidelity wireframes for critical modules such as login, dashboard, project allocation, chatbot, and performance reports. Feedback was collected through peer review, focusing on improving clarity, reducing redundancy, and ensuring alignment with usability principles and design heuristics.
Subsequent discussions focused on refining the application user flow, connecting different screens to represent real-world interactions, and ensuring that each functionality was linked to a specific user goal. The team also finalized color themes, layout consistency, and interaction logic to maintain a uniform design language across all pages.

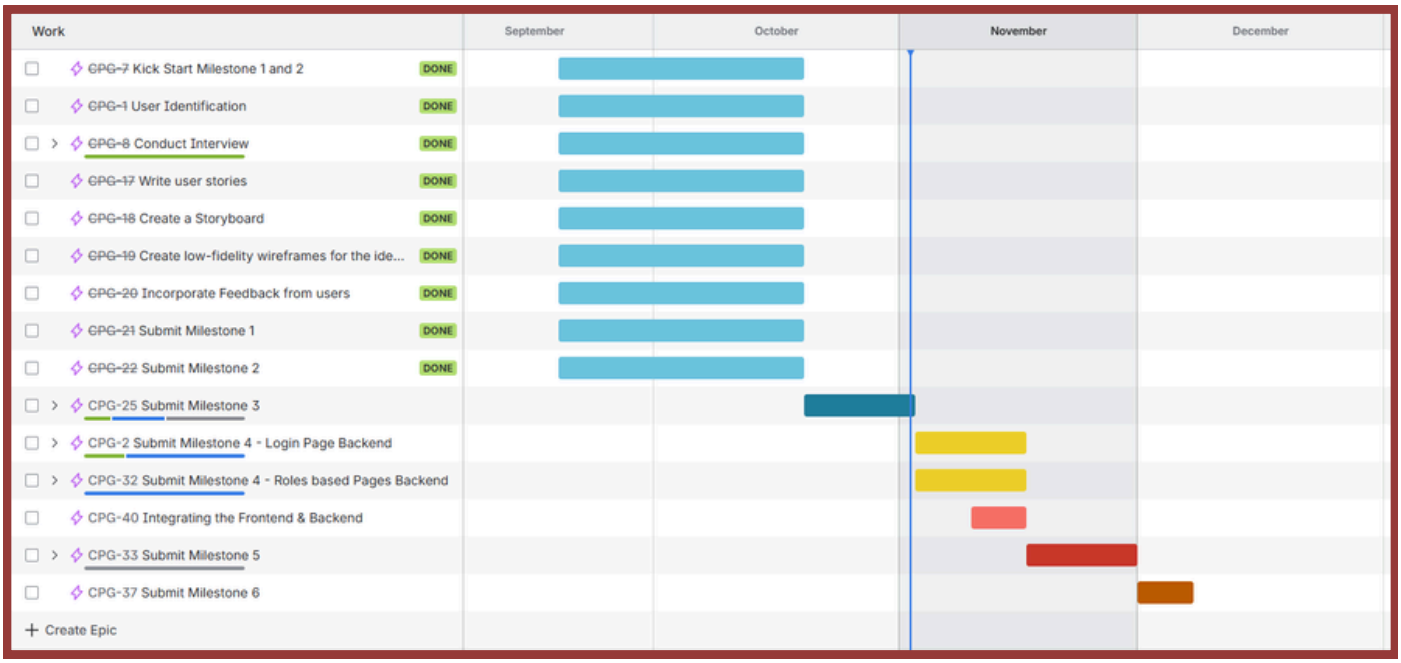
Scrum Meeting Details

Scrum Meeting Details conducted for Sprint 3

In Sprint 3, the team focused on designing and building the main frontend of the app using Vue.js. The first meeting began with a review of the wireframes from Sprint 2 and assigning pages to each member, such as the login page, dashboard, chatbot, and reports page.
In the next few meetings, everyone shared their progress, discussed layout consistency, and connected all pages using Vue Router for smooth navigation. The team also worked on making the design responsive and user-friendly across devices. Towards the end of the sprint, the team finalized all pages, added placeholder data for testing, and prepared screenshots of the Gantt chart and Kanban board. The sprint concluded with compiling and submitting the Milestone 3 Report, completing the frontend design phase successfully.

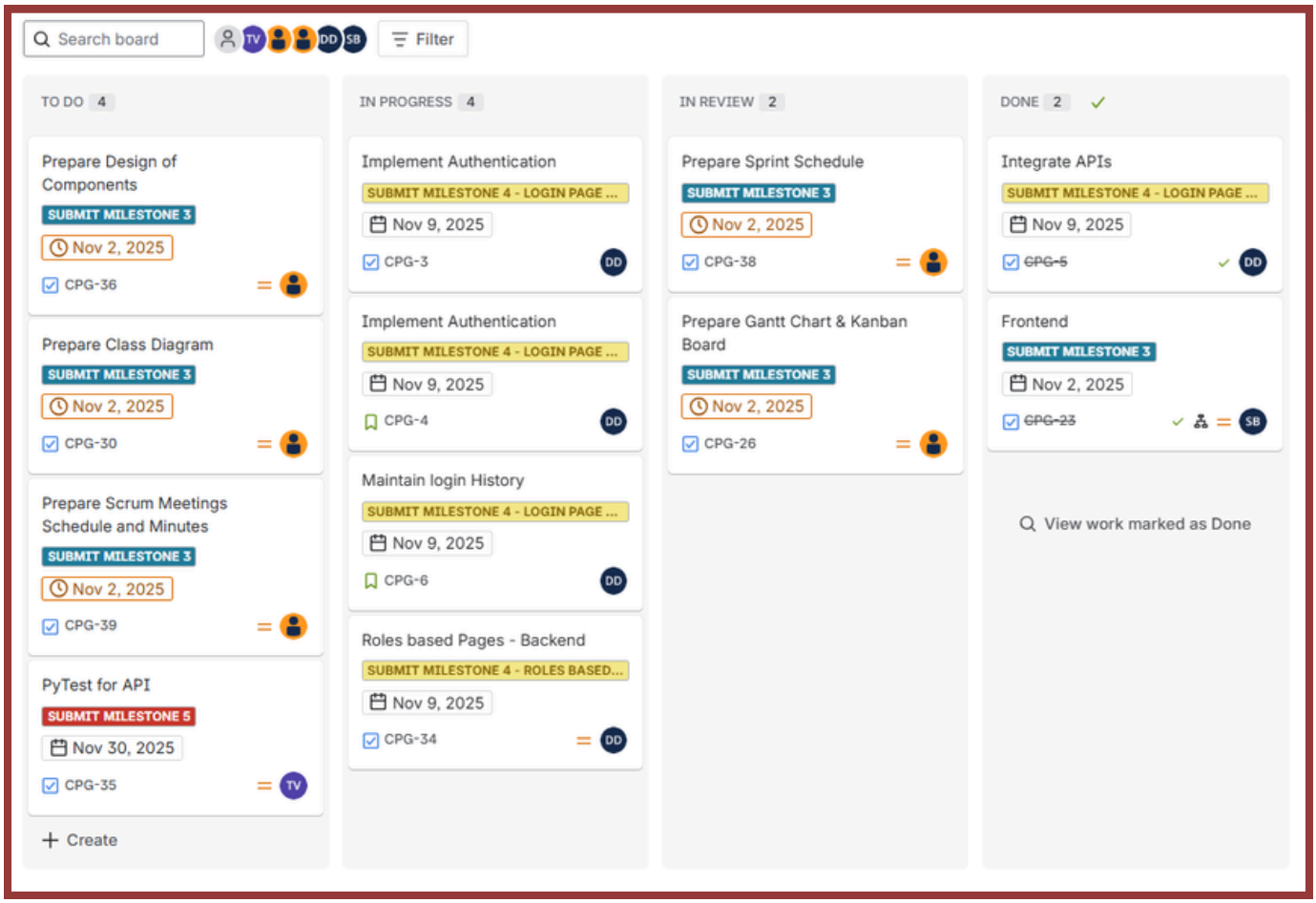
Project Scheduling Tools

Gnatt Chart



Project Scheduling Tools

Kanban Board



COMPONENT DESIGN

Admin Component

- Read employee directory list and filter results via search bar bound to searchQuery with computed filtering and input watcher.
- Update employee status by clicking Change Status and selecting a new value from the status dropdown for each employee item.
- Create a chat session by toggling the floating AI button to open the chat window for the current route.
- Create a user message with the send action, then auto-generate AI reply and auto-scroll the chat body.
- Read chat history by rendering messages array and close the dialogue to end the session.
- Read data on mount via fetchData stub and log actions for search and status changes.
- Read admin profile panel with placeholders for account details.
- Update credentials through a password change form placeholder.
- Create/rotate/delete API keys via management placeholder section.
- Create weekly backup schedule entries per day with type selector (full, incremental, differential) and date time picker.
- Update backup type and runtime by editing v-model fields for each item.
- Read current backup plan by listing configured items in the backup container.
- Save configuration to persist changes using saveConfig stub when wiring an API.
- Read live or historical logs via placeholder tail view.
- Filter or search logs when inputs are added in future iterations.
- Read system health summaries, including uptime and resource usage placeholders.
- Update status visuals when backing data sources are connected in future.
- Read current versions and update status in the updates panel.
- Trigger check for updates action when the control is wired to backend.
- Read and navigate between routes using the top navigation header and router-view outlet.
- Create global chat session available across pages, send user messages, and update conversation with simulated AI responses.
- Close chat to end session and retain layout-level state while switching routes.

COMPONENT DESIGN

HR Component

- Displays comprehensive workforce statistics, key HR metrics, and quick navigation panels for managing employees, projects, and policies from a unified dashboard interface.
- Reads and lists detailed employee records, supporting dynamic filtering, sorting, and real-time search bound to reactive inputs and computed properties for efficient performance tracking.
- Shows active and completed projects with live search, filtering, and progress indicators, enabling HR managers to oversee project timelines, team assignments, and managerial responsibilities.
- Renders official HR policy documents within an embedded PDF viewer, while the integrated “Ask HR” chatbot provides instant, interactive assistance for employee policy-related queries.
- Fetches, updates, and synchronizes data for employees, projects, and policies between the frontend components and backend APIs, handling all CRUD operations and ensuring real-time consistency.
- Creates and manages chat sessions for HR-related discussions, processes user messages, generates AI-driven responses, and maintains session history and conversational flow across views.
- Automates the generation of employee performance reviews, tracks evaluation progress, logs historical reviews, and updates performance metrics dynamically for HR analysis.
- Filters and refines tabular data across Employees and Projects views using reactive watchers and computed search bindings, ensuring seamless, real-time data exploration and quick accessibility.

COMPONENT DESIGN

Product Manager Component

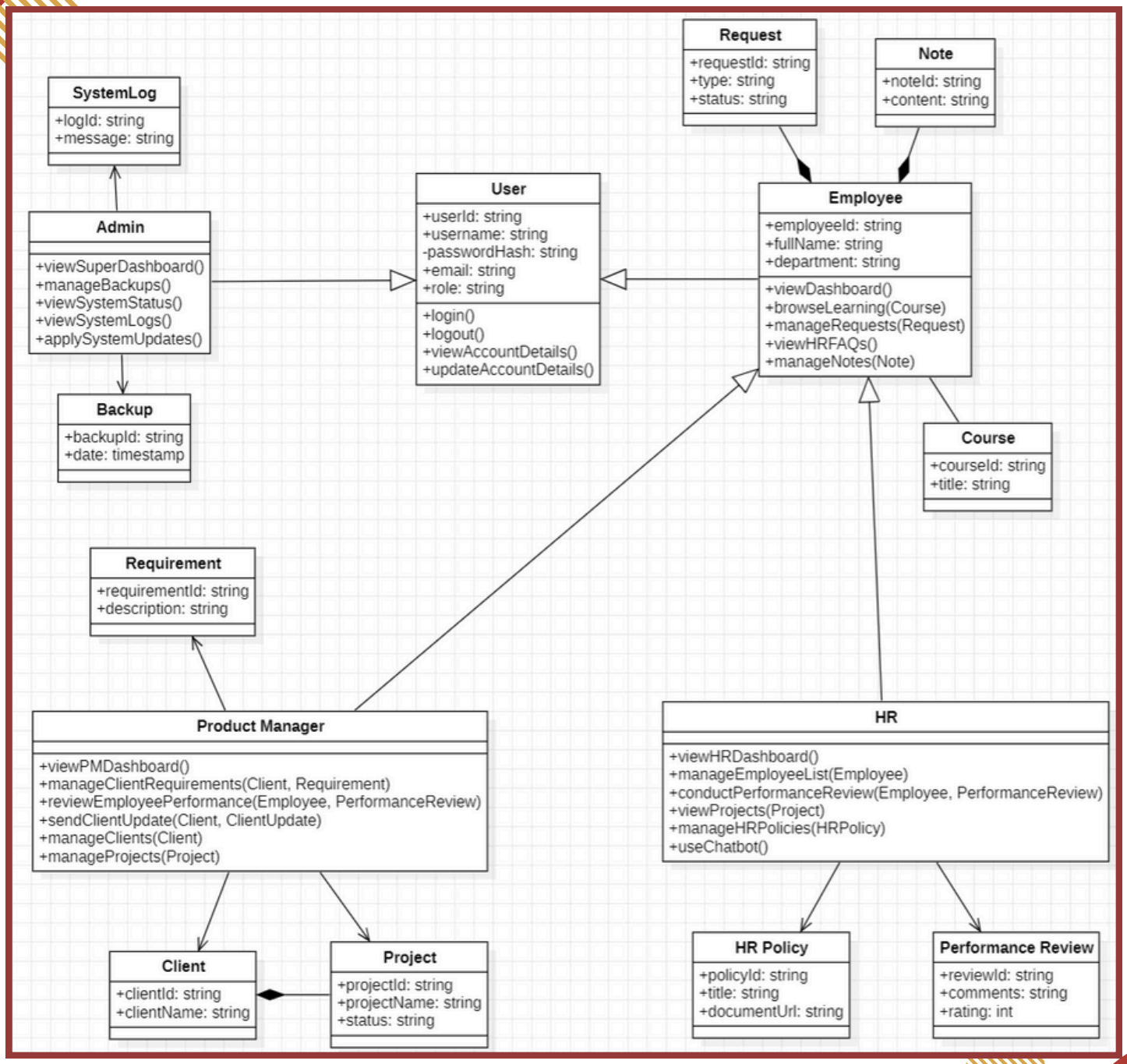
- Displays an organized overview of all clients through interactive cards, enabling product managers to access client-specific dashboards, review requirements, track progress, and manage updates from a single interface.
- Lists detailed client requirements in a structured format, supporting seamless navigation and quick access to individual requirement descriptions for efficient planning and prioritization.
- Presents project updates and client communications in a consolidated view, allowing managers to monitor progress, review milestones, and ensure transparent client engagement throughout the project lifecycle.
- Integrates an intelligent AI Agent that assists with requirement clarification, project updates, and performance insights — capable of responding contextually, maintaining conversation history, and supporting decision-making workflows across views.
- Visualizes key performance indicators through interactive line and pie charts, tracking employee or project trends over time to assess productivity, goal achievement, and overall delivery health.
- Synchronizes client, requirement, and performance data between frontend components and backend APIs, executing all CRUD operations and maintaining real-time consistency across sessions.
- Enhances productivity with reactive data binding and computed properties for instant filtering, search, and sorting within client, update, and performance views.
- Provides a cohesive, role-specific dashboard for product managers to efficiently manage clients, track outcomes, and leverage AI-driven insights for improved coordination and strategic alignment.

COMPONENT DESIGN

Employee Component

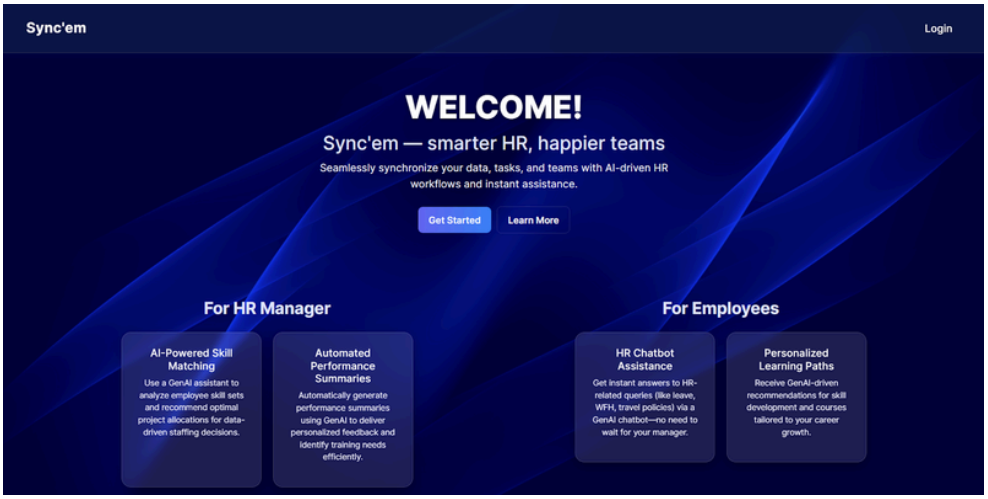
- Allows users to perform CRUD operation on different request types like leave, reimbursement, and transfer.
- Lets users search and filter requests dynamically using a reactive search bar with computed properties.
- Opens and edits request forms within the same page through Vue Router's nested routes and router-view.
- Provides smooth navigation to each form using dynamic router links for quick access.
- Includes a quick notes feature to create, read, update, and delete personal notes or reminders.
- Shows confirmation prompts for deleting notes and alerts for saving updates.
- Displays user account details like name, email, phone, role, and department.
- Allows users to edit personal info while keeping restricted fields like department and role locked.
- Supports resetting form changes using backed-up data for safe reversion.
- Automatically generates avatar initials and editable fields using computed properties.
- Lets users toggle preferences such as email notifications and two-factor authentication.
- Provides save and reset actions via placeholder methods for future backend integration.
- Uses a modular structure with each view (Requests, Notes, Account) handling its own CRUD logic and state.
- Ensures live updates and reactivity across the UI with computed properties and watchers.
- Prepares the app for backend features like authentication, data storage, and API-based CRUD operations.
- Creates a unified, interactive interface that combines user management, request handling, and note-taking.

CLASS DIAGRAM

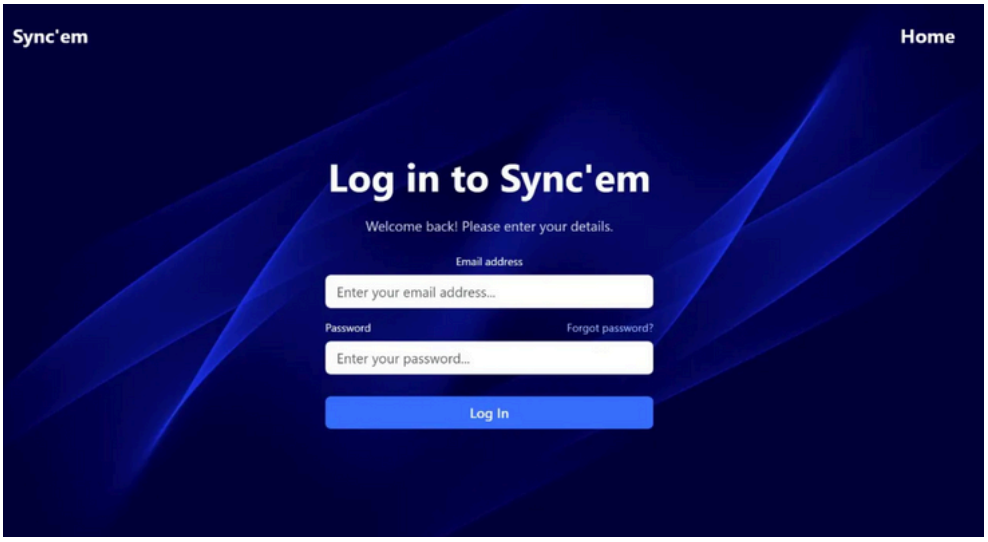


UI PAGES

Landing & Login Page



The Landing Page is the first page users see when they visit Sync'em.
It introduces the platform's purpose – an AI-powered assistant that helps HR and Project Managers allocate projects efficiently and supports employees with quick HR-related assistance.
The page highlights the core features of the system, such as Project Allocation, HR Chatbot, and Performance Insights, along with brief descriptions and visuals for each.

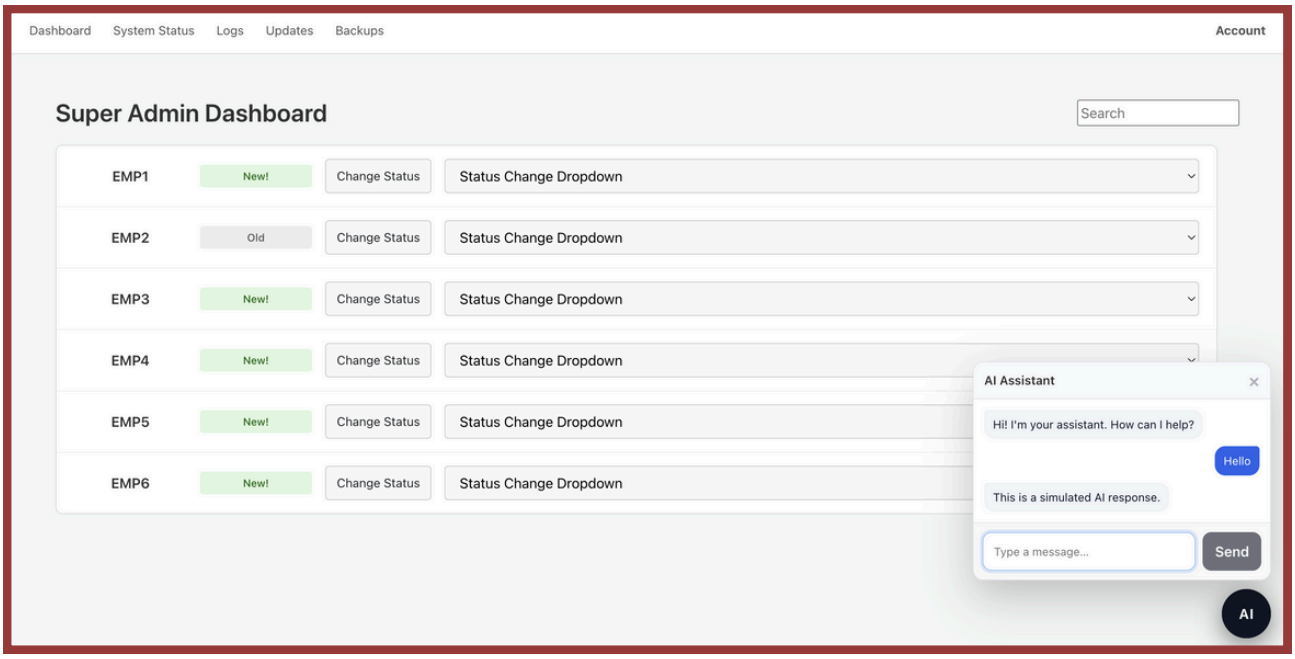


The Login Page serves as the secure entry point for all users – Admins, HR Managers, Project Managers, and Employees.
It verifies credentials through the authentication module and redirects users to their respective dashboards based on their assigned roles.

UI PAGES

Admin Pages

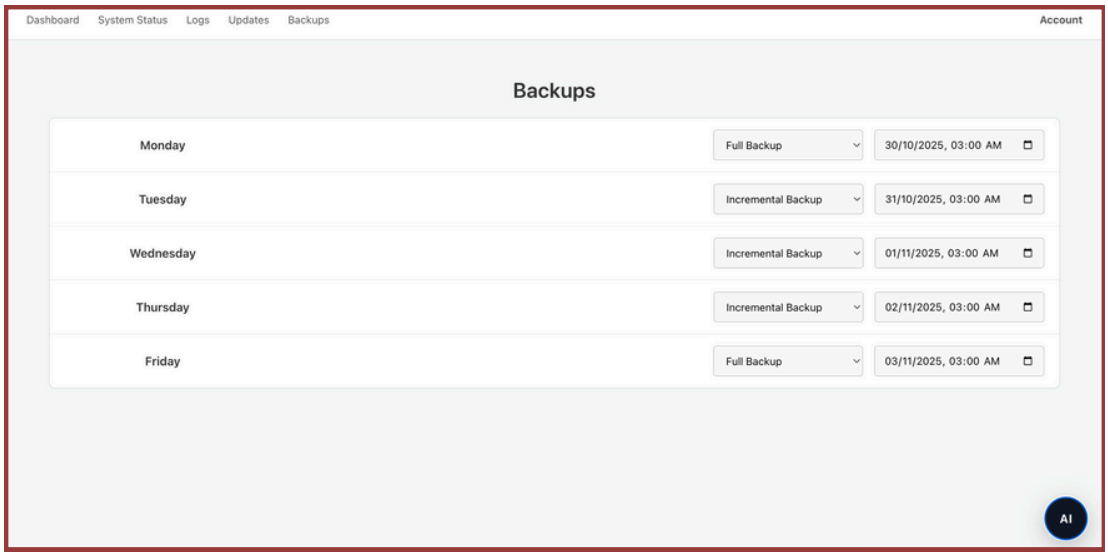
Dashboard



The Admin Dashboard serves as the central control panel for managing the overall system.
It provides a high-level overview of platform activity, including user registrations, project allocations, and chatbot interactions. The admin can view and manage all users (HR managers, project managers, and employees), monitor data flow, and oversee system performance.

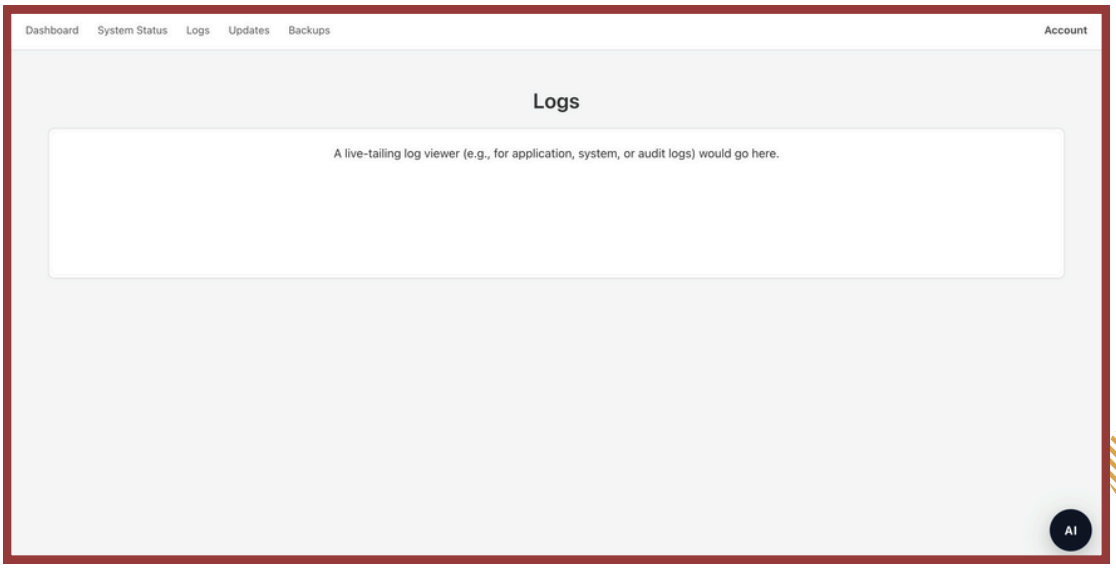
UI PAGES

Back Up



For dealing with the case of any type of failure of the the system the admin is facilicated to keep the the backup of the data of the company.

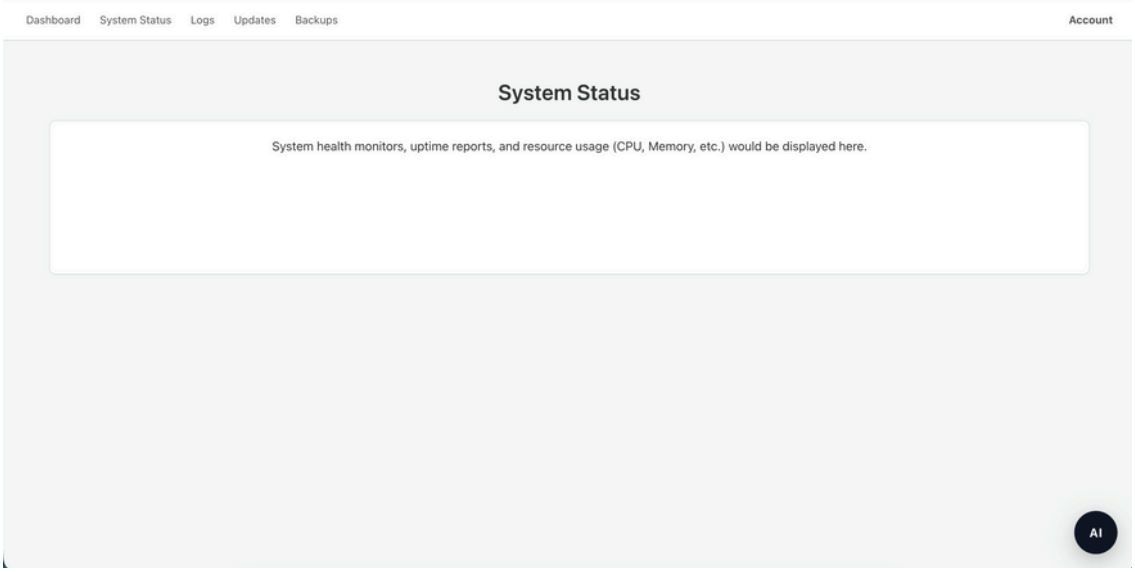
Logs



Admin also have an access of the Logs of the app . This page helps admin with the activities going on in the software what is the flow of tasks going on.

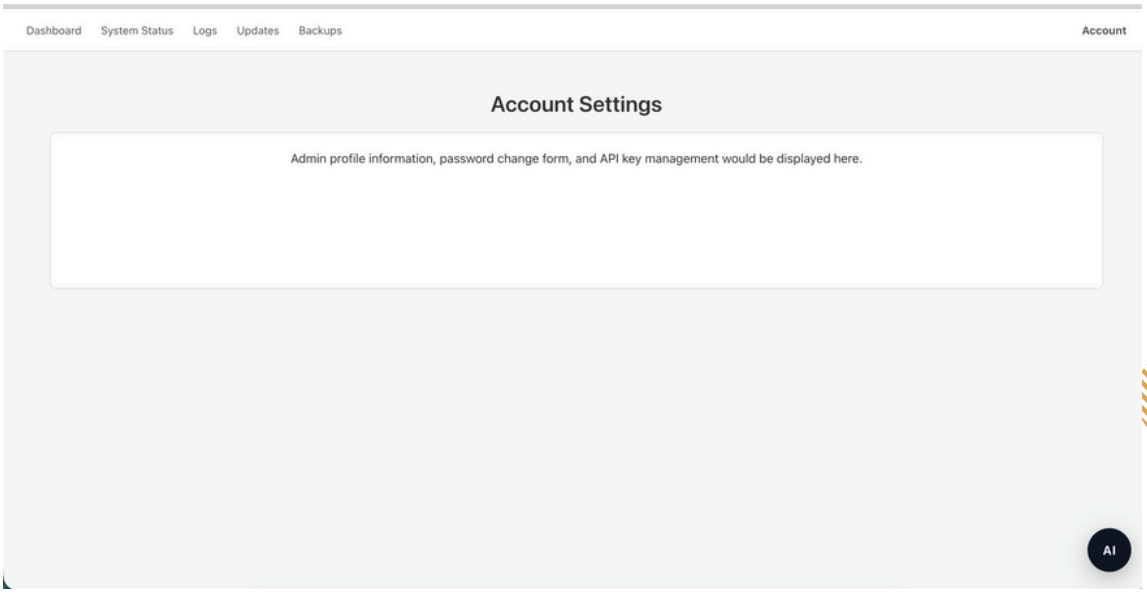
UI PAGES

System Status



Visual reports showing allocation statistics and chatbot usage trends. This page helps the user with the quick understandings and the overview as this visually represents all tasks. Track all major actions across the platform for transparency.

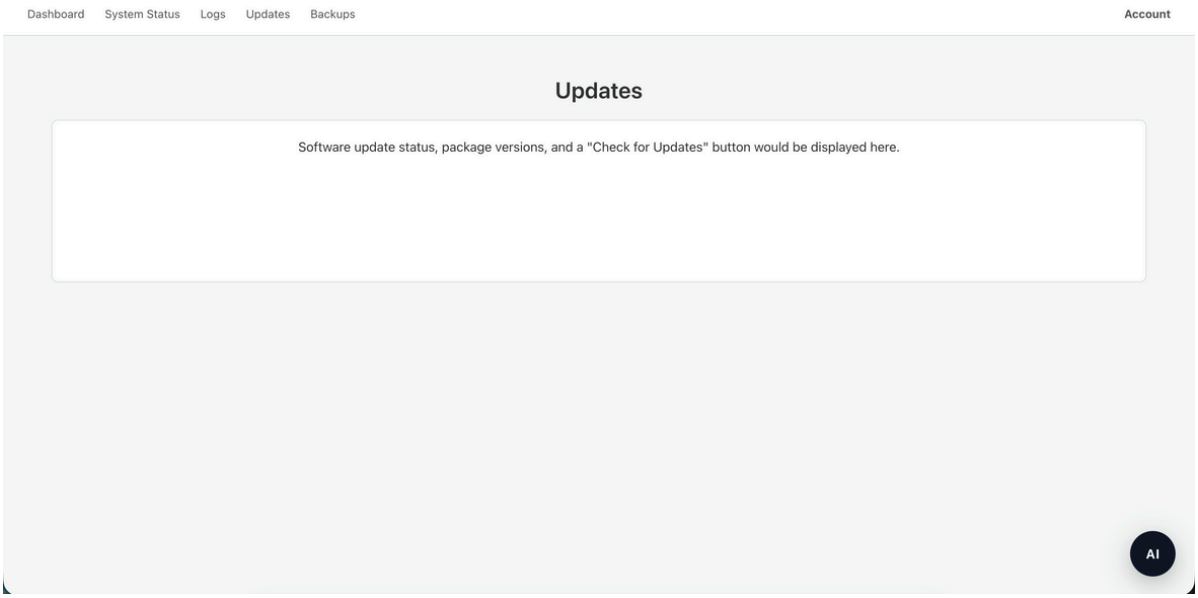
Account settings



Manage permissions and ensure data security. Secrets like Profile info, API Keys and Passwords are also managed here.

UI PAGES

Updates

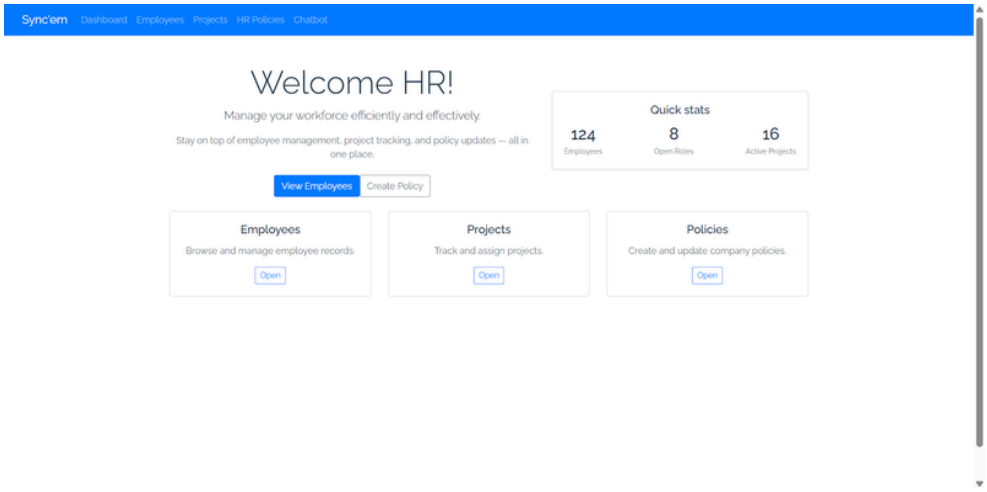


This page provides the admin with all the further updates of the System Software.
Here admin can also check that if the system that he is using is already upto date or not

UI PAGES

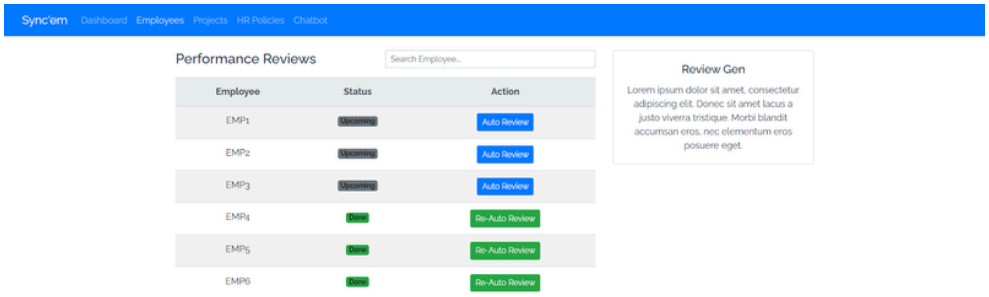
HR Page

Dashboard



The HR Dashboard provides HR Managers with tools to manage employees, monitor workloads, and oversee performance.
It acts as the main interface for HR-related operations, integrating both data management and GenAI assistance.

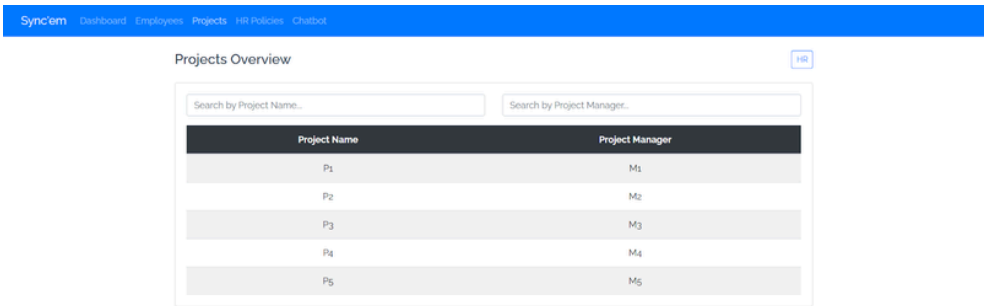
Performance Review



At this page HR can manage the employees and also see the status of their performance. Schedules and records periodic feedback and review sessions. And the review part can also be done via Gen AI.

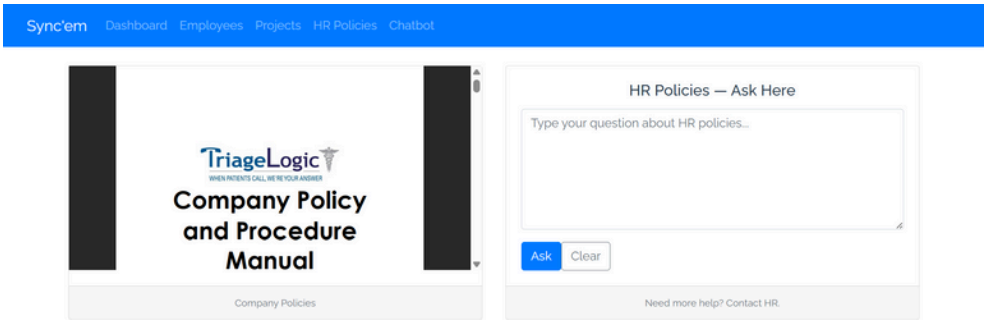
UI PAGES

Projects Overview



This page is dedicated for the Overview of the projects the which are allotted and the management of those projects. Also this page helpd HR by automating assignment of project manager to projects using GenAI-based recommendations.

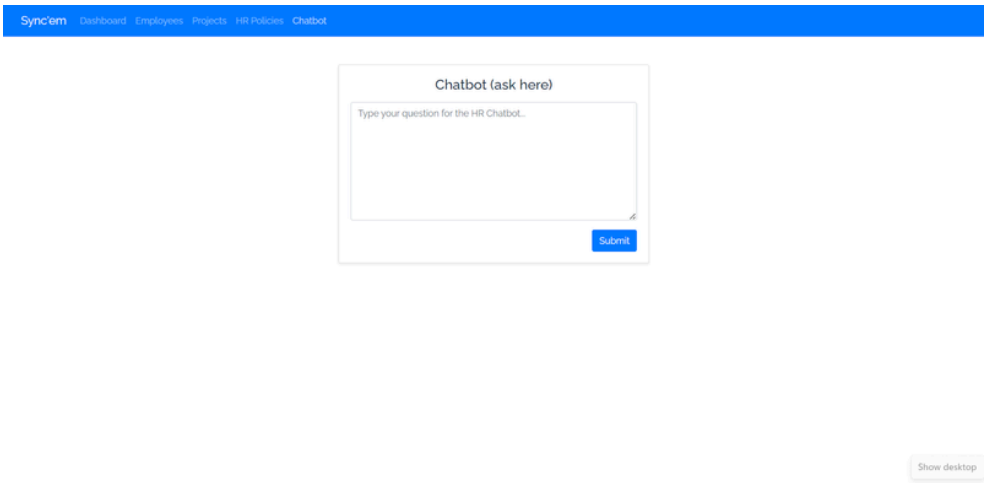
HR Policy



This page is dedicated for helping the HR with the HR Policies. In here we have all the details regarding the company details and policies, procedures manuals and rest all the policies that are required.
Also we have provided a chatbot which will saves time of finding the relevant policy using GenAI-based recommendations.

UI PAGES

Chatbot

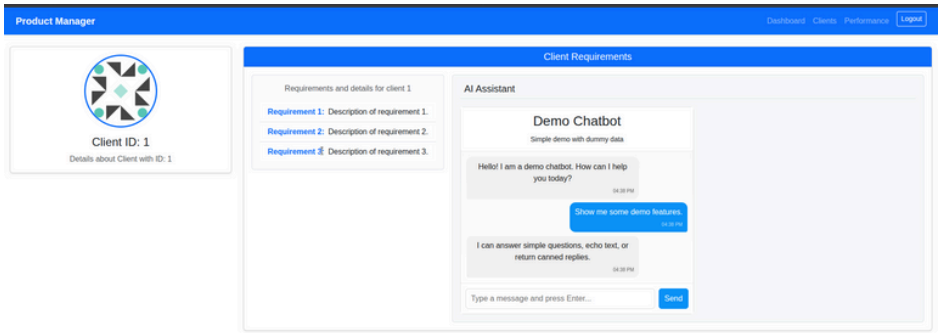


This page provides an interactive HR Chatbot interface where questions can be asked related to HR policies or procedures, instantly resolving HR queries without human intervention. It reduces HR workload, improves employee engagement, and serves as a centralized hub for quick information retrieval on company policies and procedures.

UI PAGES

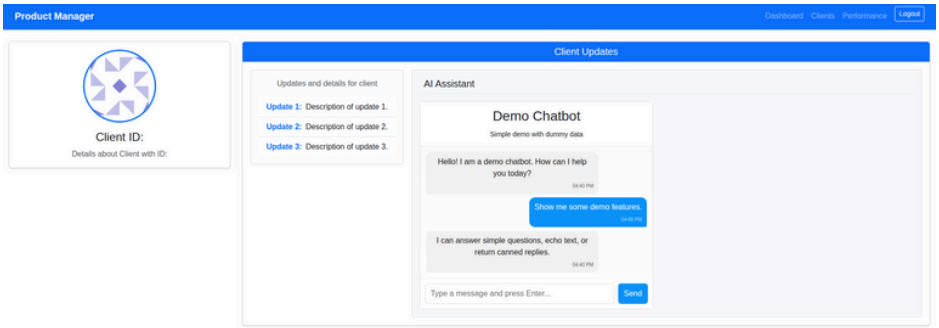
Project Manager Page

Client Requirements



This page tells the project manager about the client requirements. As well as it provides with the GenAI enhancement of the requirements.

Client Updates



This page automatically sends clients detailed milestone updates through agents, helping product managers track progress efficiently.

UI PAGES

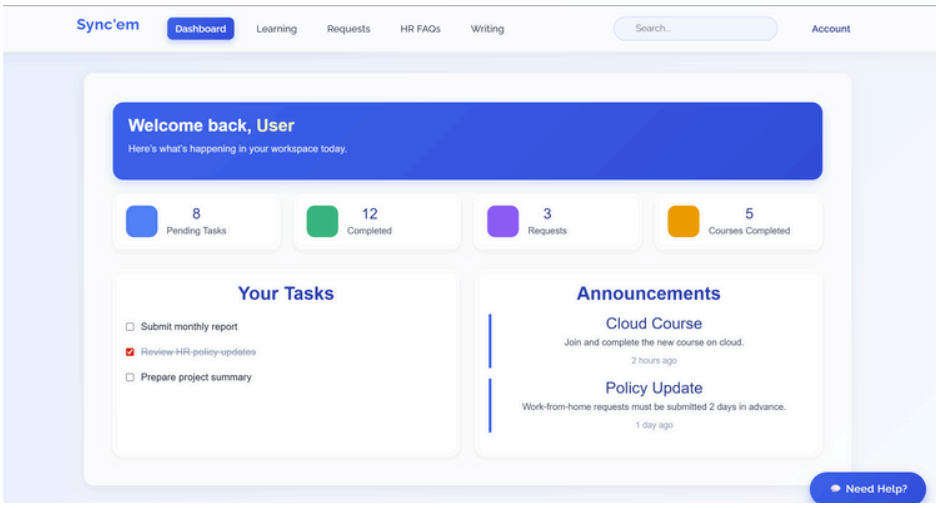
Performance Review

This page uses AI agents to automatically generate and track employee performance reviews, providing mangers with real-time insights and progress analytics.

UI PAGES

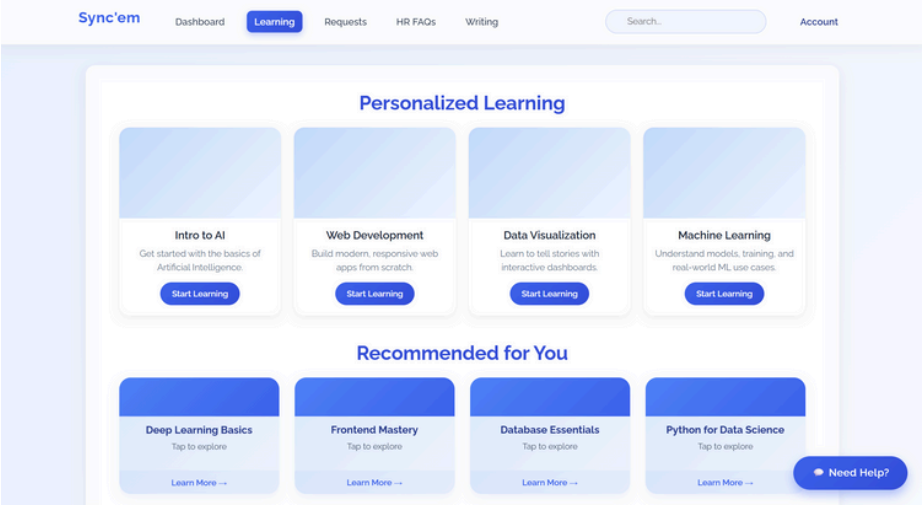
Employee Pages

Dashboard



The User Dashboard is designed for employees to view their assigned projects, performance reports, and HR-related updates in one place. It provides a personalized workspace for every employee, enhancing engagement and transparency.

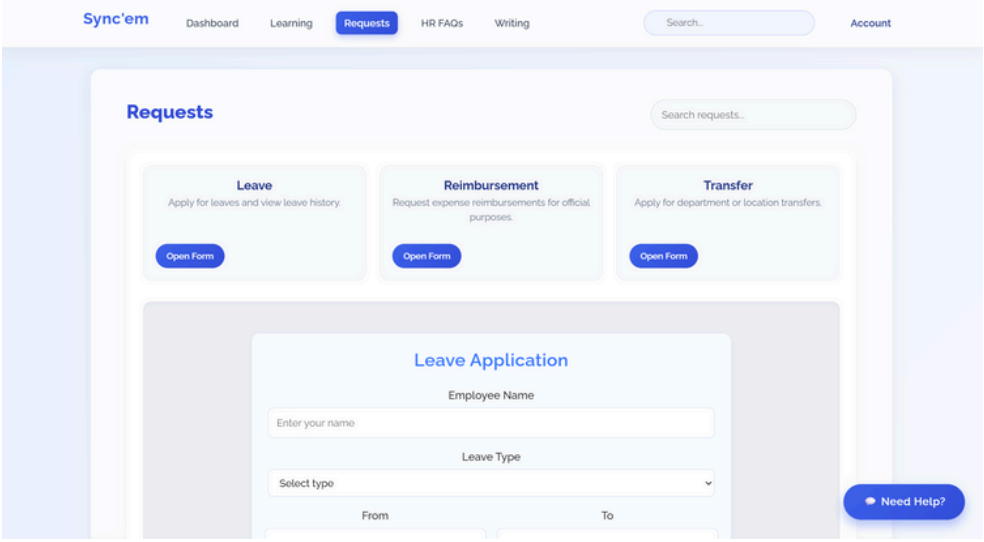
Personalised Learning



This page is dedicated for the improvements of the employees. Here they can learn new things via courses provided.
This page also suggests personalized courses based on GenAI analysis.

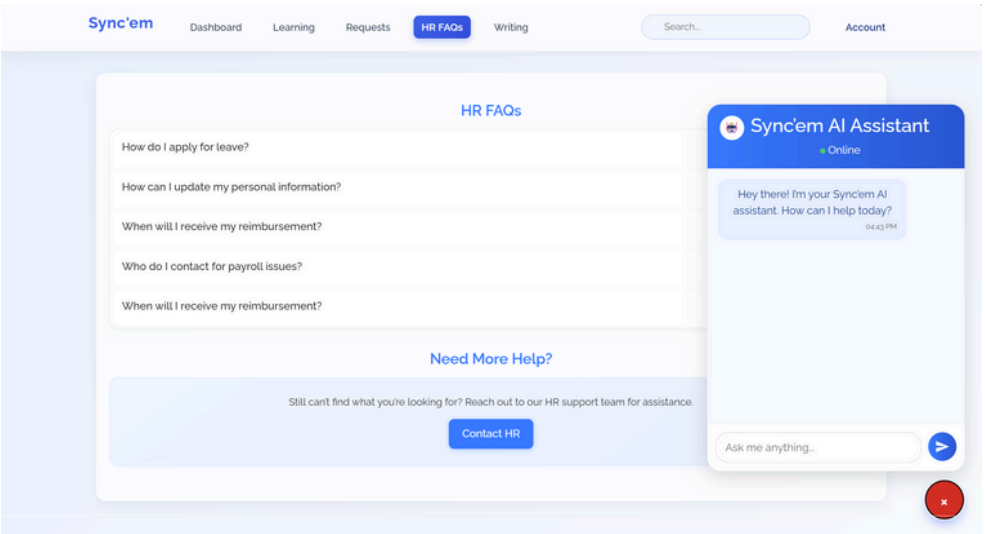
UI PAGES

Requests



This page is focused for routine interactions of the HR with the employee. Here the employee is facilitated with all sorts of requests and requirements. Also suggests personalized requests based on GenAI analysis.

HR FAQs



All the Frequent question and their answers are listed at this page and we also provide a chat bot for further assistance that personalizes the answers based on the person via GenAI

Frontend Source Code

Link

Frontend code

Table of Contents

1 ADMIN APIs

2 EMPLOYEE APIs

3 PROJECT MANAGER APIs

4 HUMAN RESOURCE APIs

DELETE /api/hr/course/assign/edit/{assign_id}
Courseassignmentdetailresource.Delete

GET /api/hr/course/assign/edit/{assign_id}
Courseassignmentdetailresource.Get

PUT /api/hr/course/assign/edit/{assign_id}
Courseassignmentdetailresource.Put

POST /api/hr/faq Hrfaqcreateresource.Post

DELETE /api/hr/faq/{faq_id} Hrfaqdetailresource.Delete

GET /api/hr/faq/{faq_id} Hrfaqdetailresource.Get

PUT /api/hr/faq/{faq_id} Hrfaqdetailresource.Put

Product Manager

PM-related endpoints for overseeing product workflows and updates.

GET /api/pm/dashboard Prdashboardresource.Get

DELETE /api/pm/clients Clientsresource.Delete

GET /api/pm/clients Clientsresource.Get

POST	/api/hr/policies	Hrpolicycollectionresource.Post
DELETE	/api/hr/policy/{policy_id}	Hrpolicydetailresource.Delete
GET	/api/hr/policy/{policy_id}	Hrpolicydetailresource.Get
PUT	/api/hr/policy/{policy_id}	Hrpolicydetailresource.Put
GET	/api/hr/policy/create	Hrpolicycollectionresource.Get
POST	/api/hr/policy/create	Hrpolicycollectionresource.Post
GET	/api/hr/employees	Employeelistresource.Get
DELETE	/api/hr/employee/{emp_id}	Employeedetailresource.Delete
GET	/api/hr/employee/{emp_id}	Employeedetailresource.Get
PUT	/api/hr/employee/{emp_id}	Employeedetailresource.Put
GET	/api/hr/annoucement/{user_id}	Announcementadminlistcreateresource.Get

POST	/api/hr/annoucement/{user_id}	Announcementadminlistcreateresource.Post
DELETE	/api/hr/annoucement/edit/{ann_id}	Announcementadmindetailresource.Delete
GET	/api/hr/annoucement/edit/{ann_id}	Announcementadmindetailresource.Get
PUT	/api/hr/annoucement/edit/{ann_id}	Announcementadmindetailresource.Put
GET	/api/hr/course	Courseadminlistcreateresource.Get
POST	/api/hr/course	Courseadminlistcreateresource.Post
DELETE	/api/hr/course/{course_id}	Courseadmindetailresource.Delete
GET	/api/hr/course/{course_id}	Courseadmindetailresource.Get
PUT	/api/hr/course/{course_id}	Courseadmindetailresource.Put
GET	/api/hr/course/assign/{user_id}	Courseassignmentlistresource.Get
POST	/api/hr/course/assign/{user_id}	Courseassignmentlistresource.Post

GET	/api/employee/account	Accountresource.Get
PUT	/api/employee/account	Accountresource.Put
DELETE	/api/employee/account	Accountresource.Delete
POST	/api/employee/assistant	Aiassistantresource.Post

Human Resource

HR-focused endpoints for managing employees and internal processes.

GET	/api/hr/reviews	Hrreviewslistresource.Get
POST	/api/hr/reviews	Hrreviewslistresource.Post
GET	/api/hr/reviews/{user_id}	Hrreviewsbyuserresource.Get
DELETE	/api/hr/review/{review_id}	Hrreviewdetailresource.Delete
PUT	/api/hr/review/{review_id}	Hrreviewdetailresource.Put
GET	/api/hr/policies	Hrpolicycollectionresource.Get

GET	/api/employee/requests/reimbursement/{reimbursement_id}	Reimbursementrequestresource.Get
PUT	/api/employee/requests/reimbursement/{reimbursement_id}	Reimbursementrequestresource.Put
DELETE	/api/employee/requests/transfer/{transfer_id}	Transferrequestresource.Delete
GET	/api/employee/requests/transfer/{transfer_id}	Transferrequestresource.Get
PUT	/api/employee/requests/transfer/{transfer_id}	Transferrequestresource.Put
GET	/api/employee/hr-faqs	Hrfaqlistemployeeresource.Get
GET	/api/employee/writing	Allquicknotesresource.Get
POST	/api/employee/writing	Allquicknotesresource.Post
DELETE	/api/employee/writing/{note_id}	Quicknotesresource.Delete
GET	/api/employee/writing/{note_id}	Quicknotesresource.Get
PUT	/api/employee/writing/{note_id}	Quicknotesresource.Put

GET	/api/employee/recommendations	Courserecommendationresource.Get
GET	/api/employee/requests/leave	Allleaverequestresource.Get
POST	/api/employee/requests/leave	Allleaverequestresource.Post
GET	/api/employee/requests/reimbursement	Allreimbursementrequestresource.Get
POST	/api/employee/requests/reimbursement	Allreimbursementrequestresource.Post
GET	/api/employee/requests/transfer	Alltransferrequestresource.Get
POST	/api/employee/requests/transfer	Alltransferrequestresource.Post
DELETE	/api/employee/requests/leave/{leave_id}	Leaverequestresource.Delete
GET	/api/employee/requests/leave/{leave_id}	Leaverequestresource.Get
PUT	/api/employee/requests/leave/{leave_id}	Leaverequestresource.Put
DELETE	/api/employee/requests/reimbursement/{reimbursement_id}	Reimbursementrequestresource.Delete

GET /api/pm/employee/performance/{employee_id} Employeeperformanceresource.Get

DELETE /api/pm/projects Projectsresource.Delete

GET /api/pm/projects Projectsresource.Get

POST /api/pm/projects Projectsresource.Post

PUT /api/pm/projects Projectsresource.Put

General

Common endpoints such as authentication and basic utilities.

GET /user/login Userloginresource.Get

POST /user/login Userloginresource.Post

GET /me Protectedresource.Get

POST /token Login For Access Token

GET / Index

POST	/api/pm/clients	Clientsresource.Post
PUT	/api/pm/clients	Clientsresource.Put
DELETE	/api/pm/client/requirements/{client_id}	Clientrequirementresource.Delete
GET	/api/pm/client/requirements/{client_id}	Clientrequirementresource.Get
POST	/api/pm/client/requirements/{client_id}	Clientrequirementresource.Post
PUT	/api/pm/client/requirements/{client_id}	Clientrequirementresource.Put
DELETE	/api/pm/client/updates/{client_id}	Clientupdatesresource.Delete
GET	/api/pm/client/updates/{client_id}	Clientupdatesresource.Get
POST	/api/pm/client/updates/{client_id}	Clientupdatesresource.Post
PUT	/api/pm/client/updates/{client_id}	Clientupdatesresource.Put
GET	/api/pm/employees	Employeesresource.Get

Employee Endpoints for employee-related operations and interactions.

GET	/api/employee/dashboard	Dashboardresource.Get
GET	/api/employee/todo	Alltodoresource.Get
POST	/api/employee/todo	Alltodoresource.Post
DELETE	/api/employee/todo/{task_id}	Todoresource.Delete
GET	/api/employee/todo/{task_id}	Todoresource.Get
PUT	/api/employee/todo/{task_id}	Todoresource.Put
GET	/api/employee/annoucements	Announcementemployeeresource.Get
GET	/api/employee/learning	Learningresource.Get
GET	/api/employee/courses	Courseassignmentemployeeresource.Get
PUT	/api/employee/course/{course_id}	Employeecourseupdatebycourseidresource.Put

Administration Administrative endpoints for managing system-wide configurations.

POST	/api/admin/register	Adminregistrationresource.Post
GET	/api/admin/summary	Admindashboardresource.Get
GET	/api/admin/employees	Adminemployeeresource.Get
POST	/api/admin/employees	Adminemployeeresource.Post
GET	/api/admin/backup-config	Adminbackupresource.Get
PUT	/api/admin/backup-config	Adminbackupresource.Put
GET	/api/admin/updates	Adminupdatesresource.Get
GET	/api/admin/account	Adminaccountresource.Get
PUT	/api/admin/account	Adminaccountresource.Put

1. Admin APIs

1.1

```
Route/Path: /api/admin/employees
```

Request Method: GET
Inputs:
JSON: not applicable
Header: {“Authorization”: f"Bearer {admin_token}"}

admin/employees

Expected Output:
HTTP Status Code: 200
JSON: List of All the employees, including the current user, in a JSON format.

Actual Output:
HTTP Status Code: 200
JSON:


```
{  "id": 2,  "name": "pm",
"email": "pm@gmail.com",  "role": "product_manager"},

{  "id": 3,  "name": "hr",
"email": "hr@gmail.com",  "role": "human_resource"},

{  "id": 5,  "name": "Admin",
"email": "admin429@gmail.com",  "role": "root"},

{  "id": 4,  "name": "New Automated Test Name",
"email": "emp@gmail.com",  "role": "employee"},

{  "id": 1,  "name": "Root",
"email": "root@gmail.com",  "role": "root"},

{  "id": 7,  "name": "Admin",
"email": "admin678@gmail.com",  "role": "root"}
```


Result: Success

1.2

```
Route/Path: /api/admin/account
```

Request Method: GET

Inputs:
JSON: not applicable
Header: {“Authorization”: f"Bearer {admin_token}"}

admin/account

Expected Output:
HTTP Status Code: 200
JSON: List having id, name, role etc for the current user (admin)

Actual Output:
HTTP Status Code: 200
JSON:

```
{  "id": 1,  "name": "Root",
"email": "root@gmail.com",  "role": "root"  }
```

Result: Success

2. Employees APIs

2.1

```
Route/Path: /api/employee/dashboard
```

Request Method:

Input: not applicable
Header Token: {“Authorization”: f"Bearer {employee_token}"}

employee/dashboard

Expected Output:
HTTP Status Code: 200
JSON: List of all message, stats, requests, tasks, announcements of the Employee

Actual Output:
HTTP Status Code: 200
JSON:

{ "message": "Dashboard data retrieved successfully",
"stats": { "pending_tasks": 1, "completed_tasks": 1,
"requests": 34, "courses_completed": 0 },

"tasks": [{ "id": 4, "task": "Updated task",
"status": "completed", "deadline": null,
"date_created": "2025-11-30T17:33:45.307489" },
{ "id": 2, "task": "Buy office supplies",
"status": "pending", "deadline": null,
"date_created": "2025-11-30T17:11:48.691191" }],

"announcements": [{ "id": 2,
"announcement": "Office will remain closed on Friday",
"created_at": "2025-11-30T17:12:24.051698" }],

"user": { "id": 4, "name": "New Automated Test Name",
"email": "emp@gmail.com", "role": "employee" } }

Result: Success

2.2

```
Route/Path: /api/employee/requests/leave
```

Request Method: POST

Input:

{
"leave_type": "Sick Leave",
"from_date": "2025-01-20T00:00:00",
"to_date": "2025-01-22T00:00:00",
"reason": "Fever & cold"
}

Header Token: {“Authorization”: f"Bearer {employee_token}"}

[employee/request/leave]

Expected Output:
HTTP Status Code: 200
JSON: List of the message, request_id and leave_id

Actual Output:
HTTP Status Code: 200
JSON:


```
{  "message": "Leave request submitted",
"request_id": 37,  "leave_id": 23  }
```


Result: Success

3. Project Manager APIs

3.1

```
Route/Path: /api/pm/dashboard
```

Request Method: GET

Input: not applicable
Header Token: {“Authorization”: f"Bearer {pm_token}"}

pm/dashboard

Expected Output:
HTTP Status Code: 200
JSON: List of clients, projects and stats

Actual Output:
HTTP Status Code: 200
JSON:


```
{  "message": "Dashboard data retrieved successfully",
"data":
{  "user":
{  "id": 2,  "name": "pm",
"email": "pm@gmail.com",  "role": "product_manager" },

 "ClientList": [
 {  "id": 1,
 "clientname": "Test Client Pvt Ltd",
 "description": "dGVzdCBkZXRhaWw="  },

 {  "id": 2,  "clientname": "Test Client Pvt Ltd",
```

"description": "dGVzdCBkZXRhaWw=" },
],

"projects": [
{ "id": 2, "project_id": "P3892",
"project_name": "Updated Project Pvt Ltd",
"description": "Updated Project Pvt Ltd",
"status": "PENDING",
"client_id": 2, "manager_id": 1 },

{ "id": 4, "project_id": "P6177",
"project_name": "Test Project Pvt Ltd",
"description": "Test Project Pvt Ltd",
"status": "PENDING", "client_id": 10, "manager_id": 1 }],

"stats":
{ "total_projects": 4, "active_projects": 0,
"completed_projects": 0, "pending_projects": 4 } } }

Result: Success

3.2

```
Route/Path: /api/pm/employee/performance/1
```

Request Method: GET

Input: not applicable
Header Token: {“Authorization”: f"Bearer {pm_token}"}

pm/employee/performance

Expected Output:
HTTP Status Code: 200
JSON: List of message, employee and their current stats and performance trends

Actual Output:
HTTP Status Code: 200
JSON:

{ "message": "Employee performance retrieved successfully",
"data":
{ "employee":

```
{  "id": 1,  "name": "Updated Name",
"email": "root@gmail.com",  "role": "root"  },

"current_stats": {  "completed": 0,  "in_progress": 0,
"pending": 0,  "total": 0,  "completed_percentage": 0,
"in_progress_percentage": 0,  "pending_percentage": 0  },

"performance_trends": [
{  "month": "Jan",  "score": 65  },
{  "month": "Feb",  "score": 72  },
{  "month": "Mar",  "score": 68  },
{  "month": "Apr",  "score": 78  },
{  "month": "May",  "score": 85  },
{  "month": "Jun",  "score": 88  }
]  }  }
```


Result: Success

4. Human Resource APIs

4.1

```
Route/Path: /api/hr/course
```

Request Method: GET

Input: not applicable
Header Token: {“Authorization”: f"Bearer {hr_token}"}

[hr/course](#)

Expected Output:
HTTP Status Code: 200
JSON: List of all courses available

Actual Output:
HTTP Status Code: 200
JSON:


```
[
{  "id": 9,
"course_name": "Duplicate Assignment Test Course",
```

```
"course_link": "https://example.com/dup-course",
"topics": "Duplicate, Testing"  },

{  "id": 8,
"course_name": "Test Course for Assignment",
"course_link": "https://example.com/test-course",
"topics": "Testing, Assignment"  },

{  "id": 6,
"course_name": "Duplicate Assignment Test Course",
"course_link": "https://example.com/dup-course",
"topics": "Duplicate, Testing"  }
]
```


Result: Success

4.2

```
Route/Path: /api/hr/policies
```

Request Method: POST

Input:


```
{
"id":"1",
"title":"first quality check",
"content":"anything"
}
```


Header Token: {“Authorization”: f"Bearer {pm_token}"}

hr/policies

Expected Output:
HTTP Status Code: 200
JSON: List of message and policy information

Actual Output:
HTTP Status Code: 200
JSON:

```
{
  "message": "Policy created",
  "policy": {
    "created_at": "2025-11-30T18:35:00.615982",
    "id": 4,
    "title": "first quality check",
    "content": "anything"
  }
}
```

Result: Success

5. Other Testcase

5.1 Where the expected output does not actual output

```
Route/Path: /api/employee/requests/reimbursement
```

Request Method: POST

Input:

```
{
"expense_type": "Travel",
"amount": "500$",
"date_expense": "2025-01-22T00:00:00",
"remark": "Taxi",
}
```

Header Token: {“Authorization”: f"Bearer {emp_token}"}

[request/reimbursement]

Expected Output:
HTTP Status Code: 200
JSON: amount reimbursed

Actual Output:
HTTP Status Code: 422
JSON:

{ "detail": [
{ "type": "json_invalid",
"loc": ["body", 106],
"msg": "JSON decode error",
"input": {},
"ctx": { "error": "Illegal trailing
comma before end of object" } }] }

Result: Failed

Table of Contents

1 MEET OUR TEAM

2 HOW TO RUN THE APPLICATION

3 TECH STACK

4 ISSUE REPORTING & PULL REQUESTS

Meet Our Team: Frontend & Backend

Deon Levon Dmello
21f3002247 3

Akbar Ali
23f1002997

Sayan Bhowmick
22f3001657

Tummala Naveen
22f3002759

Meet Our Team: Tester, PM & Scrum Master

SCRUM MASTER

D Narendran
22f3002698

UI/UX DESIGNER

Soham Chakraborty
23f1001783

TESTING LEAD

Telvin Varghese
22f2001640

PROJECT MANAGER

Sampriti Raha
22f1000064

How to Run the Application

Pre-requisites

- Python 3.10+ installed
- Node.js and npm installed
- Mailhog, redis and celery installed

Run the Backend (flask)

1. Open a terminal and navigate to the backend directory:
cd backend

2. (Recommended) Create and activate a virtual environment:
python -m venv venv
On Windows: venv\Scripts\activate
On macOS/Linux: source venv/bin/activate

3. Install Python dependencies:
pip install -r requirements.txt

4. Start the Flask server:
python main.py

5. Notes:
Default demo credentials (as per README):
admin: admin@example.com
pm: pm@example.com
hr: hr@example.com
employee: employee@example.com

Our Tech Stack

Our Tech Stack

FRONTEND

- Framework: Vue.js (SPA with role-specific dashboards for human resources, product manager, admin and employee)
- UI/UX: User-friendly components
- Authentication : JWT Bearer tokens in Authorization headers
- Gen-AI: Simple and lucid answers to FAQs

BACKEND

- **Runtime/Language:** Python 3.10+
- **Web Framework:** FastAPI
- **CORS:** FastAPI-CORS for secure cross-origin frontend-backend communication
- **Authentication/Authorisation:** FastAPI-jwt-extended (JWT issuance, verification, role-based access)
- **Password Security:** werkzeug.security (hashing, verification)
- **Session Management:** JWT sessions with decay timer
- **Input Validation:** Regex-based checks; endpoint-level validation

Our Tech Stack

AI/LLM INTEGRATION

Providers: Google Generative AI (Gemini 2.5 Flash, Gemini Embedding 001) and ChatGroq (Llama 3.3 70B Versatile)
Use Cases:

- Employee Agent: Utilizes a RAG pipeline with structured context to provide personalized guidance.
- HR Agent: Deploys lightweight FAISS retrieval for fast, broad HR knowledge lookups.
- PM Agents: Leverages LangGraph to automate structured workflows like requirements analysis and reporting.

DATABASE & ORM

- **ORM:** SQLAlchemy integrated with FastAPI
- **Database Engine:** PostgreSQL
- **Models:**
  - **Employee Models:** FAQ, Announcement, Attendance, AttendanceStatusEnum, Chat, Course, Department, Leave, QuickNote, Reimbursement, Request, RequestStatusTypeEnum, RequestTypeEnum, RoleEnum, StatusTypeEnum, ToDo, Transfer, User, UserCourse
  - **HR Models:** HRPolicy, PerformanceReview
  - **Product Manager Models:** Client, EmpTodo, Project, Requirement, PMStatusEnum, Update, UserProject
  - **Admin Models:** Backup, BackupTypeEnum, Log

Our Tech Stack

API & SERVICES

- **REST Endpoints:** Modular routes for Auth, Chat, Tasks, To-dos, Course Recommendation, Leave Application, Reimbursements, Project Tracking, Admin etc
- **External API:** Gemini API and ChatGroq for the Gen-AI features
- **File Handling:** Chatbot based on the policy documentation pdf which answers based on the pdf content

SAFETY & SECURITY

- **JWT Error Handlers:** Expired, invalid, missing, or revoked token responses (HTTP 403 Forbidden).
- **RBAC:** Role-protected endpoints for Root, Human Resource, Product Manager, and Employee.
- **Prompt Sanitization:** Safe AI responses via structured message templates (SystemMessage vs. HumanMessage).
- **Secret Management:** Config-driven keys with fallback generation (e.g., secrets.token_hex).
- **Data Integrity:** Enforced parent-child relationships and foreign key constraints.

Our Tech Stack

ANALYTICS & ADMIN

- **Aggregations:** Dashboard stats, task analytics, user activity metrics, employee performance metrics
- **Admin Tools:** User CRUD (admin-only), cascade deletes, migration utilities

DEV & TESTING

- **Dev Server:** FastAPI debug mode on localhost:8000
- **Testing Framework:** Pytest
- **Test Patterns:** Use bearer-token authentication, mock external service calls, and run FastAPI in debug mode with in-memory.Tests run against a PostgreSQL Db.

Our Tech Stack

UTILITIES & LIBRARIES

- **Core Stack:** FastAPI and Vue.js 3 form the backbone, supported by Celery for background tasks and SQLModel for database management.
- **AI & Retrieval:** LangChain and LangGraph drive agent workflows, using ChromaDB and FAISS for vector search.
- **UI & Utilities:** Bootstrap/Tailwind and Chart.js handle design and analytics, secured by tools like Axios and DOMPurify.

CONFIGURATION

- **Environment-driven:** API keys (Gemini), base URLs, secret keys via Config
- **Instance Directory:** Created on startup for persistence when needed

Issue Reporting & Pull Requests

ISSUES SUMMARY

Issue Summary – Tokens Invalidation During App Reload

Focus Area:
Authentication & Developer Experience

- The issue centers on token stability during development. When the backend reloads, the frontend-held tokens are immediately invalidated, disrupting all authenticated sessions.

Core Problem:

- Tokens stored by the frontend become unusable every time the backend restarts, even for minor code changes. This forces developers to re-login repeatedly during active development.

Impact & Importance:

- High Impact on Developer Productivity:
- Frequent token invalidation significantly slows down iterative development, as developers must authenticate again after each refresh or rebuild.
- Interrupts Testing & Debugging:

Breaks the workflow for validating authenticated routes, form submissions, dashboards, and user flows.

- Reduces Development Efficiency:
- Forces unnecessary friction in everyday tasks, extending overall dev timelines and obscuring genuine bugs.
- Affects DX (Developer Experience):
- Creates frustration and lowers development velocity, especially for teams frequently modifying backend logic.

Type:

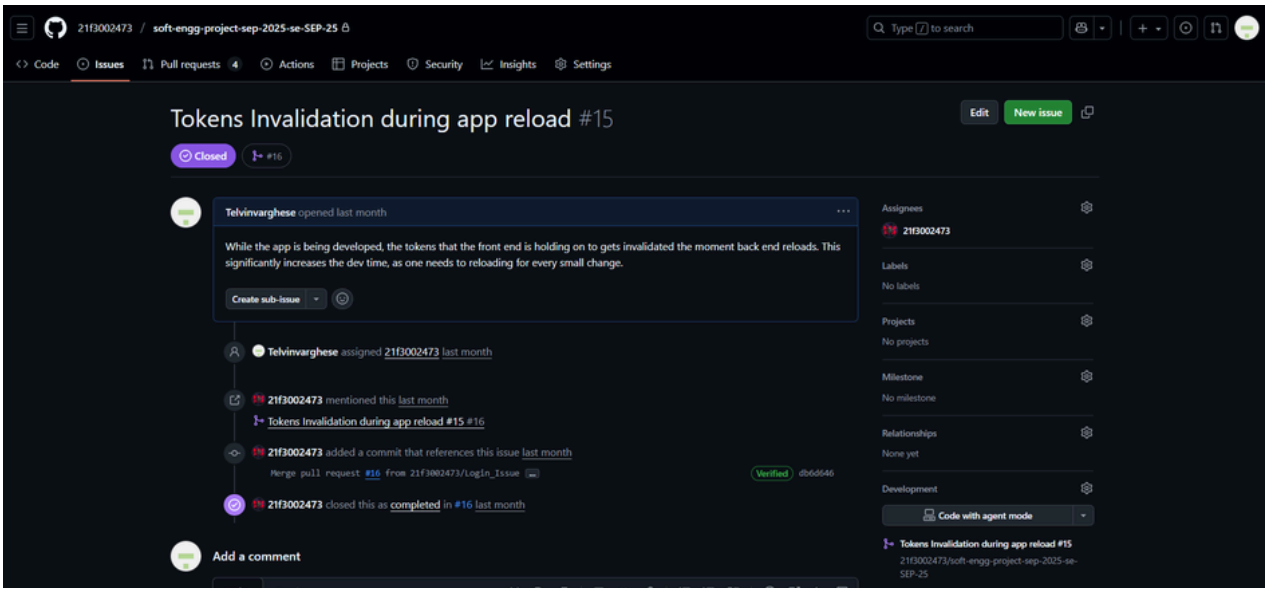
- Bug fix – relates to authentication session persistence during local development.

Status:

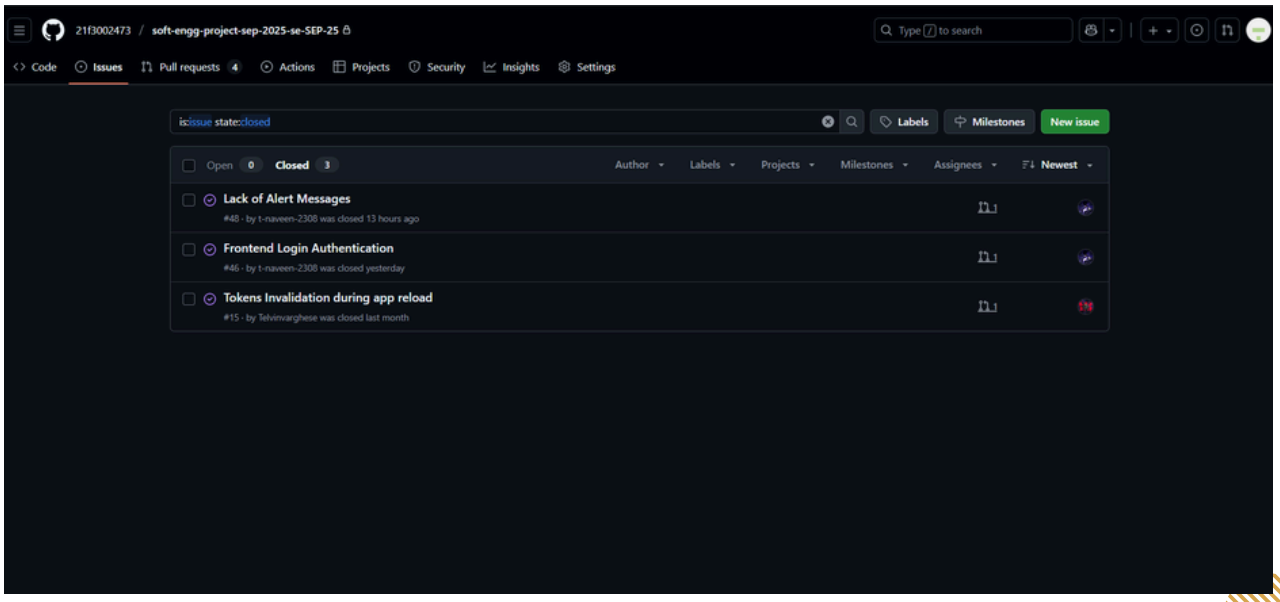
- Closed – an important fix given its impact across the entire dev workflow.

Issue Reporting & Pull Requests

ISSUES SUMMARY



Other Issues



Issue Reporting & Pull Requests

PULL REQUESTS SUMMARY

0 Open ✓ 50 Closed | Author | Label | Projects | Milestones | Reviews | Assignee | Sort

- Frontend — #1 by 21f3002473 was merged on Oct 29 — 8
- Backend — #2 by 21f3002473 was merged on Oct 29 — 3
- Users View PR ✓ enhancement — #3 by t-naveen-2308 was merged on Nov 2 • Approved — 1
- Raise PR for Project Manager View ✓ enhancement — #4 by 21f3002473 was merged on Nov 2 • Approved — 7
- Admin View PR ✓ — #5 by SayanBhowmick-22f3001657 was merged on Nov 2 • Approved — 1
- Hr views — #6 by 23f1001783 was closed on Oct 31
- HR Page done so far ✓ — #7 by 23f1001783 was merged on Nov 2 • Approved — 3
- Added Forms and Improved the UI ✓ — #8 by t-naveen-2308 was merged on Nov 3 • Approved — 2

0 Open ✓ 50 Closed | Author | Label | Projects | Milestones | Reviews | Assignee | Sort

- Enhanced Allocation System, UI Fixes, PostgresDB Improvements & Ngrok Integration ✓ — #53 by t-naveen-2308 was merged 31 minutes ago • Approved — 1
- updated tests file ✓ — #52 by Telvinvarghese was closed 9 hours ago
- Chatbots Agents of HR ✓ — #51 by 23f1001783 was merged 5 hours ago — 2
- Refactored admin and employee frontend files. ✓ — #50 by t-naveen-2308 was merged 2 days ago
- Adding Toaster ✓ — #49 by t-naveen-2308 was merged 2 days ago • Approved — 1
- Fixed authentication issue in the frontend. ✓ — #47 by t-naveen-2308 was merged 2 days ago — 1
- HR Frontend Integration ✓ — #45 by t-naveen-2308 was merged yesterday
- frontend & backend integration is done for Admin & the admin dashboar... — #44 by SayanBhowmick-22f3001657 was merged 2 days ago • Approved — 1
- Employee Agents Refinement — #43 by t-naveen-2308 was closed yesterday • Draft

Issue Reporting & Pull Requests

PULL REQUESTS SUMMARY

Enhanced Allocation System, UI Fixes, PostgresDB Improvements & Ngrok Integration #53

Edit | Code

Merged | t-naveen-2308 merged 23 commits into main from client_feature | 34 minutes ago

Conversation 1 | Commits 23 | Checks 4 | Files changed 51 | +4,729 −1,685

t-naveen-2308 commented 4 hours ago • edited | Collaborator

- Added skill-based employee allocation
- Implemented automatic employee allocation
- Fixed multiple UI issues
- Resolved PostgresDB errors and added troubleshooting commands
- Integrated ngrok support and addressed related JSON parsing issues

t-naveen-2308 and others added 23 commits 4 hours ago

Used toast, sweetalert in PM frontend. c2d71d9

Reviewers: 21f3002473

Assignees: t-naveen-2308

Labels: None yet

Projects: None yet

Milestone

Frontend Initial Setup: Routing, Auth, Landing Page & RBAC Integration #1

Edit | Code

Merged | 21f3002473 merged 18 commits into main from frontend | on Oct 29

Conversation 8 | Commits 18 | Checks 0 | Files changed 26 | +13,433 −1

21f3002473 commented on Oct 11 • edited by t-naveen-2308 | Owner

- Set up the full frontend structure with routing and layouts
- Added landing page with background visuals and UI improvements
- Implemented Login and Registration pages with working auth flow
- Added token validation, refresh handling, and role-based route protection
- Created and updated global state management (appState.js, store.js)
- Completed RBAC integration across the app
- Cleaned up files and added minor UI enhancements

initial commit for frontend 4d7b975

21f3002473 assigned t-naveen-2308 and SayanBhowmick-22f3001657 on Oct 11

Reviewers: t-naveen-2308, SayanBhowmick-22f3001657

Assignees: t-naveen-2308, 23f1001783

Labels: None yet

Projects: None yet