Team Roles

Scrum Master: Aum Palande
Product Owner: Nishant Basu
Individual Contributor: Mukil Senthilkumar, Raunak Kharbanda, Ellika Mishra, Prakhar Suryavansh, Susheel Vadakkekuruppath

Links

Deployed App: https://tamu-sks.herokuapp.com/
Github Repository: https://github.com/nishant-basu-tamu3/student_knowledge_system
Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2721032
Join Slack Channel:
https://join.slack.com/t/csce606studen-has4149/shared_invite/zt-2riag3qhu-2ykYDzif4hoiZcmwC~rZLw

Customer Meeting Information

- Recurring Meeting (starting on 9/25): Peterson 426, 4:15 - 5:45
- Met with Dr. Lightfoot and whole team
- Meeting Summary
    - Product already created, and can develop on top of that
    - Main task: Update project to work for new UI of Howdy
        - Rosters look different
        - Export instructions should change
    - Expansion of the Quiz Task:
        - Matching Tasks
        - Students that the professor gets correct should show up less
    - Other tasks:
        - Wants to be able to have current students
        - Add tags or flags for whether the student is applicable for a LOR or to be a TA
    - Overall Project Flow
        - Verify that the product works
        - Find gaps and issues with the product
        - Fix the Gaps
        - Change data for the new Howdy
        - Flags and Quiz changes

Summary

Professors have lots of students. A professor should be able to remember what they thought of the student when they took a course, especially when a student asks for a Letter of Recommendation, a TA position, or anything else. The Student Knowledge System is a way for professors to be able to connect with their students more, and allows for the professors to remember legacy students. The stakeholders, professors, should be able to view previous students, as well as view their photo, see comments about what the professor thought about them, and see when students took specific classes. The application meets these requirements, as professors are able to upload rosters directly from Howdy, and then edit comments about them which are saved in a database. The professor should be able to filter and find their students easily. Similarly, the professor can also use quizzes in order to learn the names and faces of the students in their current sections.

As this is a legacy project, we are developing on top of a code base that is already created. The previous teams have set up a solid product that tends to have some necessary features. Our main focus for the project will be to add features, such as the student page and fixing some issues with the login UI. We think that while the project is a solid starting place, there is a lot of room for improvement. We will first start by breaking down the project, and seeing any bugs or gaps we find, and fix those as well as the other stories by previous teams that were not finished.

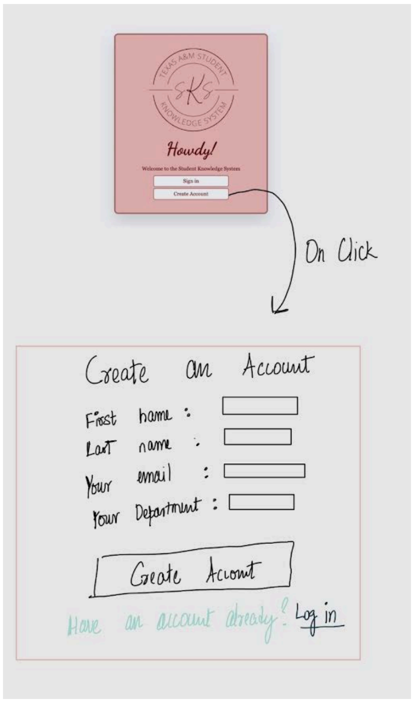
User Stories

- Feature: See When a Student took a class
  - As a professor
  - The information I wrote after the student took the first class should copy over to the second time the student takes the class
  - Even if I remember students, knowing when and which classes they took. This would help a lot if the student has chosen to take me for different classes, which would affect how I write my referrals.
- Feature: Create account redirect at the login page
  - As a professor
  - When I don't have an account, and I click Create account
  - Redirect to the Create Account page
  - I should see text fields for Name, Mail ID, and Department.
- Feature: Ensure that only valid image files can be uploaded
  - As a professor,
  - When I create a new student,
  - And I click the "Browse" button to upload an image of the student,
  - Then only valid image file formats (e.g., .jpg, .png, .gif) should be allowed
- Feature: Validation on creating Students(Without any input a student is created)
  - As a professor,
  - When I create a new student without any details

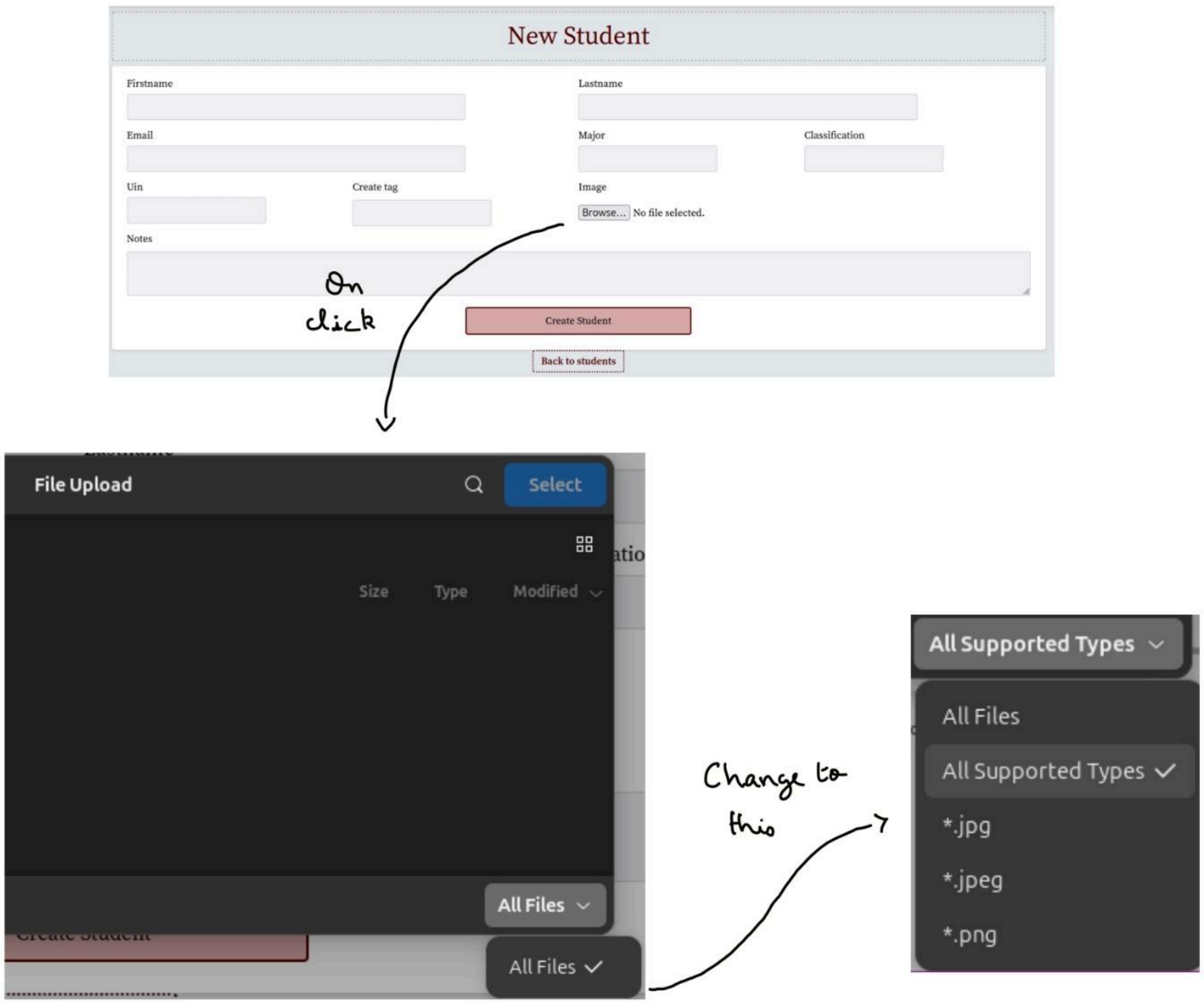
    - And I click the "submit" button to add a new student then a new student is added without any input data.
    - Student input should be proper, every field should be mandatory and validated before adding a new student. Once a student is added then the same student input should not be allowed.
- Feature: Students page doesn't display students.
    - As a professor,
    - When I create a new student,
    - And I go to the "Students" page,
    - Then the students page doesn't display the students
- Feature: Modify Login
    - As a student
    - The current system generates a magic link and uses it for verification. I prefer having the user redirect through the verification link instead first verifying a magic link and later navigate back to the login page.
    - Requirement is to verify an account with an in-mail link and use this link for further account verification. The link should redirect to the login page for the user.

User Interface

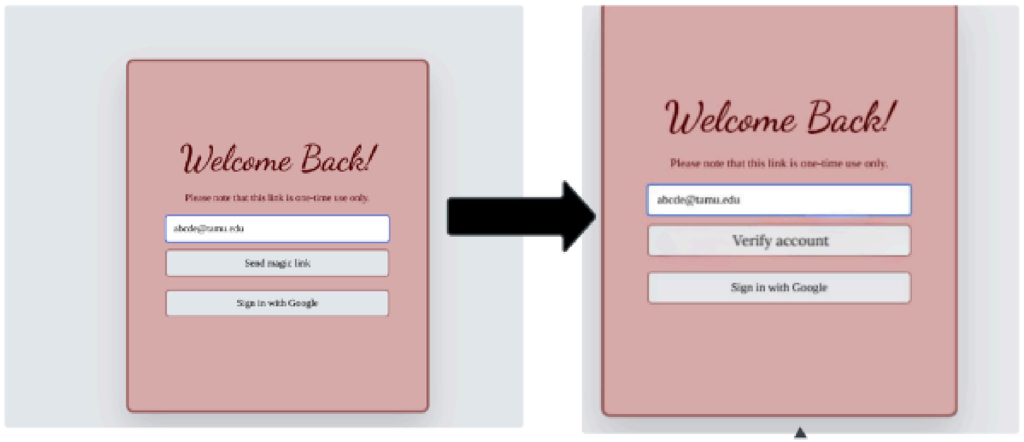
Mockup for Feature: Create account redirect at the login page



Mockup for Feature: Ensure that only valid image files can be uploaded



Mockup for Feature: Modify Login

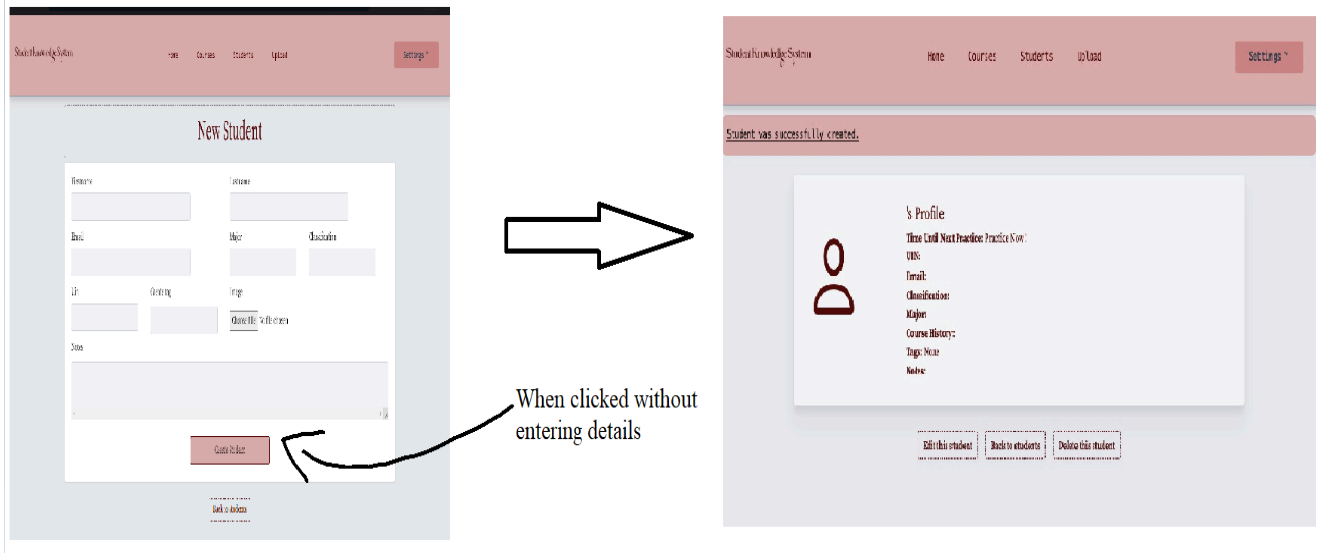


Please note that this link is one-time use only.

Please note that this link is one-time use only.

Mockup for Feature: Validation on creating Students(Without any input a student is created)

Current



Student was successfully created.

's Profile

Time Until Next Practice: Practice Now!

UIN:

Email:

Classification:

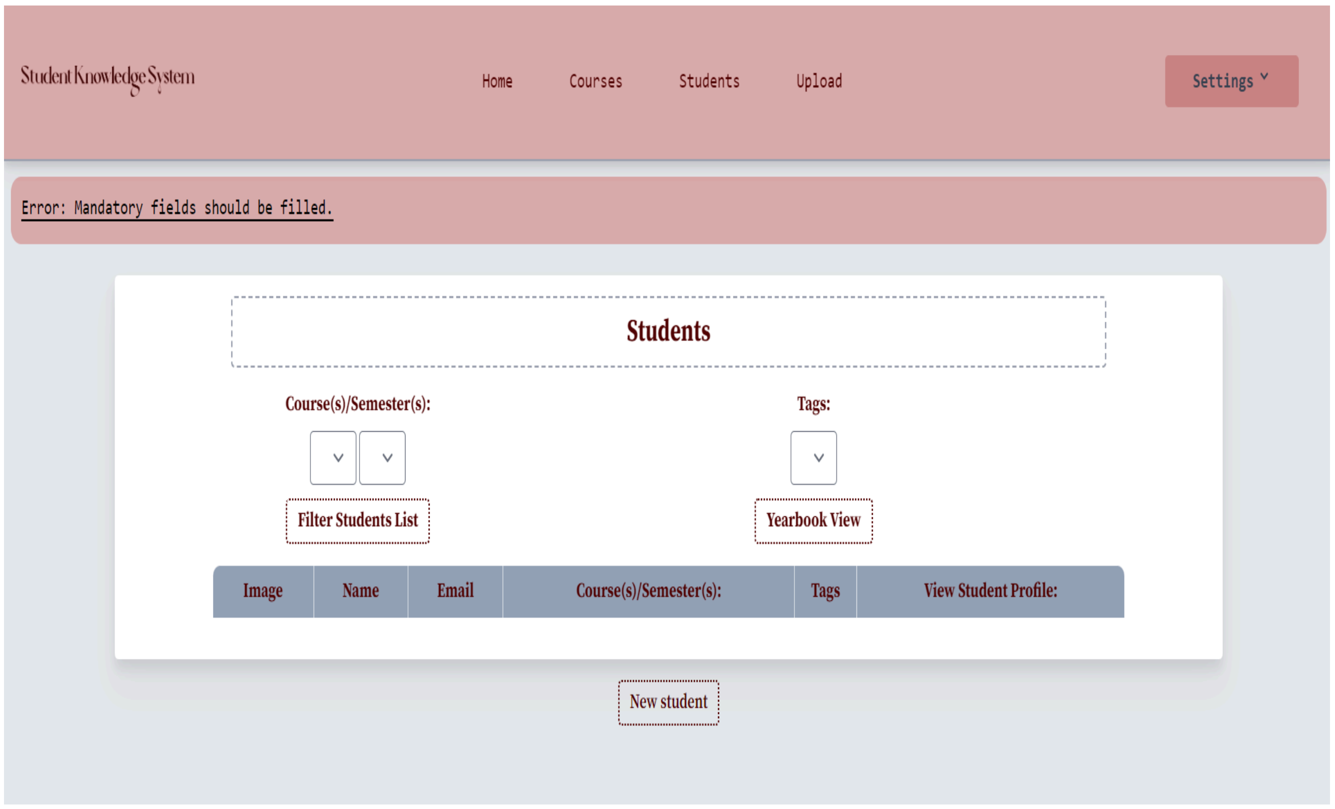
Major:

Course History:

Tags: None

Notes:

Updated:

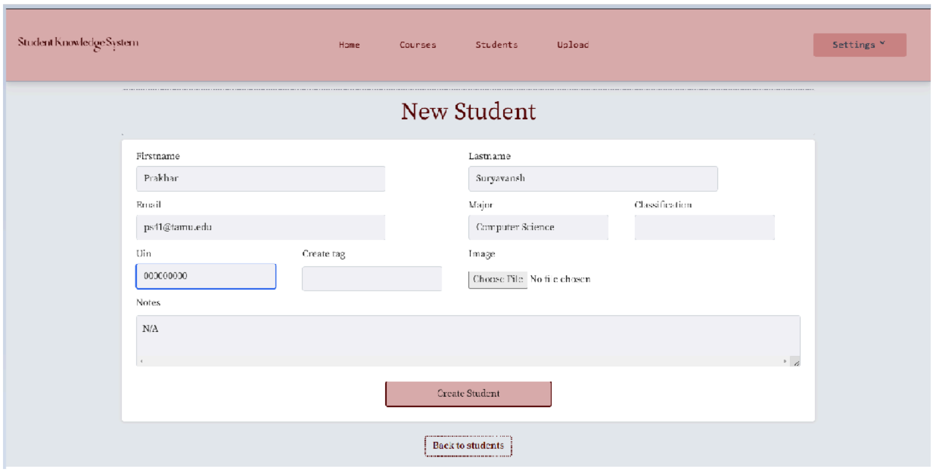


Error: Mandatory fields should be filled.

Image	Name	Email	Course(s)/Semester(s):	Tags	View Student Profile:

Mockup for Feature: Students page doesn't display students

Current



Student Knowledge System · Home · Courses · Students · Upload · Settings

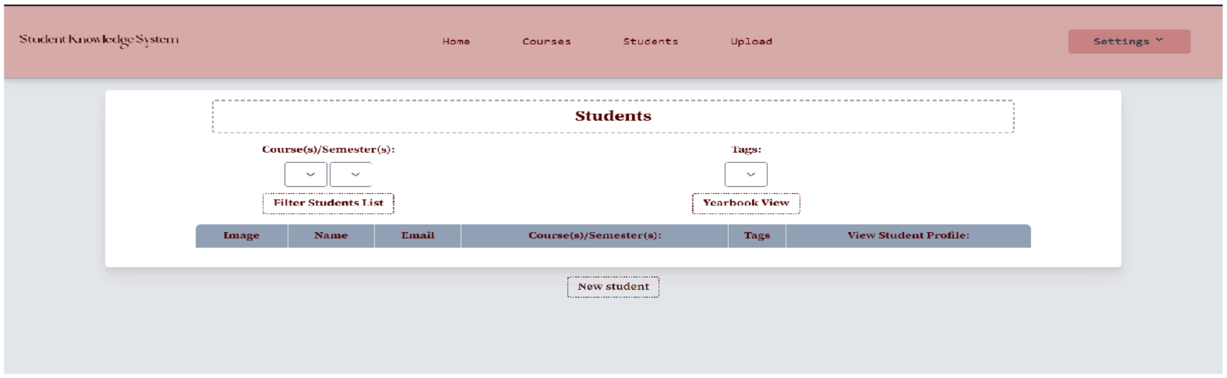
New Student

Firstname: Prakhar	Lastname: Suryavansh	
Email: ps41@tamu.edu	Major: Computer Science	Classification:
Uin: 000000000	Create tag:	Image: Choose File No file chosen
Notes: N/A		

Create Student

Back to students

Student info not displayed



Student Knowledge System · Home · Courses · Students · Upload · Settings

Students

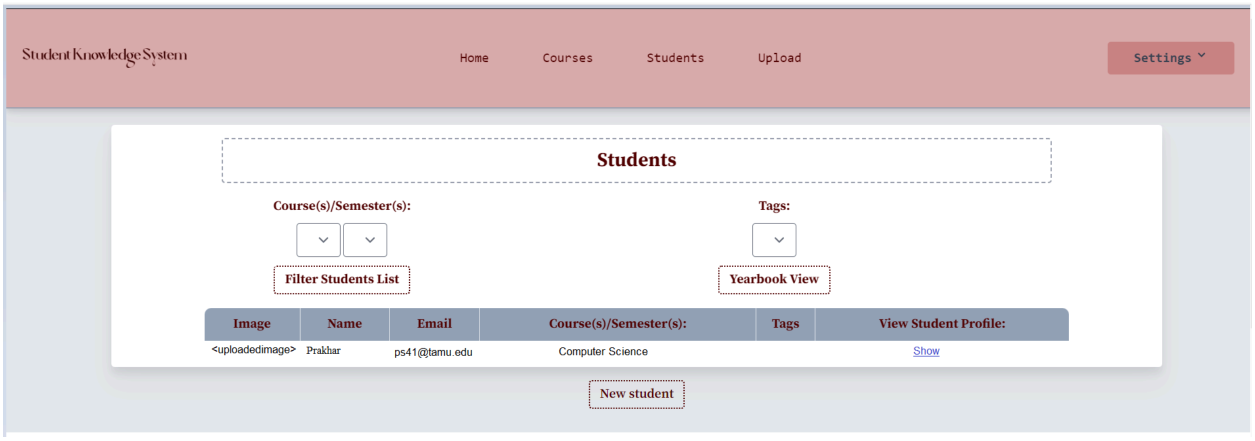
Course(s)/Semester(s): · Tags:

Filter Students List · Yearbook View

Image	Name	Email	Course(s)/Semester(s):	Tags	View Student Profile:

New student

Updated: (To display student details)



Student Knowledge System · Home · Courses · Students · Upload · Settings

Students

Course(s)/Semester(s): · Tags:

Filter Students List · Yearbook View

Image	Name	Email	Course(s)/Semester(s):	Tags	View Student Profile:
<uploadedimage>	Prakhar	ps41@tamu.edu	Computer Science		Show

New student

Sprint Backlog

- Change create account page
  - Get name and other information
- Students page does not display students
- Limit image upload file type
- Validation on create student
- Change login page
  - Magic Link -> Verify Account
- Explain the export data more clearly
- Professor Profile Page
- Login through OAuth requires you to create an account
- Update for new version of Howdy

Strategy for Learning

For legacy projects, the approach to learning and improving the prior code will be divided into five phases. These include an assignment and planning phase, two learning phases, and two improvement phases. Below is a breakdown of this strategy:

(Assignment Phase) Phase 1: Team Setup and Initial Planning

The first phase will involve assigning team roles such as Product Manager, Scrum Masters, and Individual Contributors. During this time, we will also gather key information, including access to the repository and client details. A high-level view of the project will be discussed with the client through weekly meetings, gaining insights into the current state, stakeholders, and prior accomplishments. We will also clarify client expectations for improvements in this iteration.In this phase, we will begin creating boilerplate user stories and tasks, each with a general acceptance criterion. These tasks will be time-bound and broken down into smaller tasks as our understanding of the project deepens.

(Learning Phase - 1) Phase 2: Local Setup and Initial Deployment

In the first learning phase, the team will set up local environments and deploy the initial version of the application to a Heroku instance. We will seek any documentation left by previous teams and, if possible, connect with past developers for clarification. This phase will ensure that every team member can run the application locally, as we will rotate roles in future sprints. Special attention will be given to configuring external dependencies like AWS, Google APIs, and Heroku. Support may be sought from TAs or professors, particularly when managing shared accounts among the team.

(Learning Phase - 2) Phase 3: In-Depth Understanding of the Project

In the second learning phase, after completing the initial setup, we will aim to gain a deeper understanding of the application. We will critically assess processes and technologies used so far, asking key questions like whether we should continue using AWS or explore alternatives. Each team member will become familiar with the full tech stack, unit and integration testing, schema structures, external APIs, and how the app interacts with other systems like the Howdy Portal. By the end of this phase, every team member should be comfortable navigating all aspects of the project.

(Improvement Phase - 1) Phase 4: Process Improvement

The first improvement phase will focus on streamlining processes. On the product side, we will establish a standardized template for creating user stories, assigning story points, and conducting daily Scrum meetings. We will assign specific approval roles and create a timeline for deployments. On the infrastructure side, we will evaluate testing frameworks and improve test coverage, fixing failing tests where needed. We'll aim to implement a CI/CD pipeline, ensuring that only tested code is merged into the master branch. We will also refine documentation, aiming for simplicity and efficiency in setting up the local project by implementing a one-step build process, in line with Joel's test.

(Improvement Phase - 2) Phase 5: Product Improvement

In the final improvement phase, the team will focus on implementing the actual features requested by the client, balancing what is feasible with the project timeline. Each user story will be broken down into independent tasks, tracked for progress, and paired with associated test stories. Features will be tested in a staging environment by a designated testing team and undergo UAT (User Acceptance Testing) with the client before deployment. We will follow a structured release schedule to ensure smooth rollouts of the final features.