BoilerTrack

CS 30700
Sprint 1 Planning Document

Team 19

- Anitej Waghray
- Daivik Ghosh
- Daniel Benes-Magana
- Mihika Sharma
- Shlok Bairagi
- Shreya Laxmi Nagendra

Sprint Overview:

In this sprint, we hope to set up the account creation, login, and management user stories on both the student and staff side. Also, adding OAuth with authentik middleware to allow single sign on login later on. We also plan to implement the lost item listing where the students can view, filter and send claim requests whereas the staff can add/manage items in the listings. All, while integrating the processes with our database for proper data entry and retrieval. By the end, we aim to have a basic lost and found system set up and running.

Scrum Master: Daivik Ghosh

Meeting Plan:

- Monday 3:30 PM with Project Coordinator
- Monday 8:00 PM
- Thursday 6:30 PM
- Saturday 5:00 PM

Risks and Challenges:

Our project faces several challenges, particularly in setting up the server, implementing secure user authentication, and organizing the code repository as the codebase grows. While the team has experience in web development, certain tasks, like managing multiple UI forms and ensuring security, will require deeper learning. Additionally, balancing the learning curve with development could affect progress.

Time management will also be a key challenge, as estimating the effort for tasks like server setup and authentication may lead to delays. To mitigate these risks, we plan to dedicate time to learning new technologies, staying organized, and maintaining clear communication. With careful planning and collaboration, we are confident we can overcome these challenges and complete the project successfully.

Current Sprint Details:

User Story #1

As a student user, I want to have the option to create an account with my email.

#	Description	Estimated Time	Owner
1	Create UI for inputting email and password.	4h	Mihika
2	Store email address in relation to a unique user.	2h	Daniel
3	Add verification for email address via confirmation email.	2h	Daniel
4	Store password in relation to a unique user.	1h	Daniel
5	Test user account creation by checking the database	1h	Daniel

Acceptance Criteria:

- Given that the UI for entering email and password are implemented, when the user navigates to the registration page, they should be presented with fields to enter their email and password.
- Given that the user successfully created an account, a confirmation email should be sent to the email address inputted during login.
- The confirmation email should show an alert that the account is confirmed and redirect to the login page.
- Given that the UI for inputting the email and password are implemented, when the user navigates to the login page, they should be presented with fields to enter their email and password.
- The email address and password for a user should be stored in the database appropriately.
- The unit test should ensure account information is being inserted to the database.

User Story #2

As a student user, I want my details (preferred name and pronouns) to be assigned to my account.

#	Description	Estimated Time	Owner
1	Create UI for profile setup	3h	Daivik
2	Add field on registration page for preferred name and pronouns	1h	Daivik
3	Add field in profile setup page for preferred name and pronouns	1h	Daivik
4	Store preferred name and pronouns for the user	1h	Anitej
5	Create unit test for profile registration functionality	1h	Anitej

Acceptance Criteria:

- Given that the UI for profile setup is properly implemented, when the user navigates to the home page, they should be presented with a way to navigate to their profile.
- On the registration page, there should be a field to enter a preferred name and pronouns.
- On the profile page, there should be a field to enter/change preferred name and pronouns.
- The preferred name and pronouns should be stored in the database in conjunction with the user.
- The unit test should ensure profile information is inserted/edited correctly.

User Story #3

As a student user, I want to have a reset password option.

#	Description	Estimated Time	Owner
1	Create a reset password page on login page, with same UI, to input email address	2h	Mihika
2	Add verification for if account exists on the backend and send confirmation or error message	2h	Daniel
3	Add password reset token generation	2h	Daniel

	and store it in the database		
4	Create page to input new password	2h	Mihika
5	Add verification of token and update password in the database	3h	Daniel
6	Create unit test to check correct functionality of password reset by checking database for relevant updates	1h	Daniel

Acceptance Criteria:

- Given the UI for the reset password page has been implemented correctly, the user should be able to navigate to the page and enter their email to be verified
- If the user's account does not exist, an alert should show that it doesn't exist.
- If the user's account does exist, the user should receive an email to get their password reset token
- Given the UI for the new password page is implemented correctly, the user should be able to enter a new password and redirect back to the login page
- When the new password is entered, the password for the user should be updated in the database.

User Story #4

As a student user, I want to be able to deactivate my account.

#	Description	Estimated Time	Owner
1	Create a page that gives the option to close an account	2h	Mihika
2	Add confirmation prompts in the form of "Please type <email/username> to confirm you want to delete your account"	2h	Daniel
3	Add final password verification if a user wants to delete their account	1h	Daniel
4	Server should verify that the logged in user is the account that will be removed	1h	Daniel
5	Server should remove the account from the database by marking it as deleted	2h	Daniel

6	Job should be run to remove accounts marked as deleted in bulk during low traffic hours	3h	Daniel

Acceptance Criteria:

- Given the homepage has an option to manage the user account, there should be an option for deleting the user account
- Given that the UI for deactivating an account is set up properly, when the user navigates to this button in the user profile page, they should be presented with dialog boxes to confirm that they want to deactivate their account.
- Given the user decided to activate their account, the client should password verify the user before deleting the account
- Given that the user successfully deactivates an account, a confirmation email should be sent to the user's email address.
- Given the user information is marked as deleted, the server should remove the data from the database in a regular job

User Story #5

As a student user, I want to be able to search listings with keywords.

#	Description	Estimated Time	Owner
1	Add search bar to the homepage	1h	Mihika
2	Reactively offer keywords in the database to the user as they type	2h	Anitej
3	Allow a user to search multiple tags	1h	Anitej
4	Redirect the user to a results page when they enter the search	1h	Anitej
5	Results page should include information about the item like where it was returned to	2h	Anitej

Acceptance Criteria:

- Given the client has a homepage, there should be a search box
- Given that the client has a search box, when the user types in the search box, they should reactively be offered keyword suggestions that can be added to their search.
- Given the keywords are offered, the user should be able to search by multiple keywords

- Given the user can select multiple keywords, the user should be able to remove keywords from their search
- Given that the user selects a keyword, the search should return all listings with matching (or similar?) keywords.

User Story #6

As a student user, I want to be able to view a list of all lost items on the platform

#	Description	Estimated Time	Owner
1	Create an “all items” page	5h	Mihika
2	Add a link to the page in the homepage	1h	Mihika
3	Add filters for major categories like “water bottle” or “headphones”	1h	Mihika
4	Have a checkbox list for keywords to also filter by	2h	Mihika
5	Include a search option to filter keywords like in story #5	1h	Mihika
6	Allow users to pin entries to the top of the table	1h	Mihika

Acceptance Criteria:

- Given the homepage has a top bar, there should be an option to see items
- Given there is an all items page, there should be a table sorted in alphabetical order
- Given the table is reactive, there should be options to filter major categories
- Given there are options to filter keywords, there should be a search box like story #5
- GIven the table is reactive, users should be able to pin items to the top of the results

User Story #7

As a student user, I want to be able to filter lost items by category.

#	Description	Estimated Time	Owner
1	Create a filter button and place it next to the search box.	1h	Daivik

2	Create a drop down menu that appears when the filter button is pressed.	3h	Daivik
3	Create another drop down menu within the filter widget.	3h	Daivik
4	Add the category options to the category menu	1h	Daivik
5	Sort the items by category	2h	Anitej

Acceptance Criteria:

- Given that the UI for the User Feed has a search box at the top, there should also be a filter drop down menu next to it.
- Given that there is a filter drop down menu next to the search box, there should be another menu to filter the listed items by category.
- Given that there is a menu to filter the listed items by category, there should be a list of the different categories that can be selected.
- Given that a particular category is toggled, the items should be sorted by that category.
- The User Feed should only display the items of that category.

User Story #8

As a student user, I want to be able to filter items by location (where they were lost or found).

#	Description	Estimated Time	Owner
1	Create a “choose location” button within the filter menu.	2h	Daivik
2	Create another drop down menu within the filter widget.	2h	Daivik
3	Add the building options to the drop down menu.	2h	Daivik
4	Sort items by building found.	2h	Anitej

Acceptance Criteria:

- Given that the UI for the User Feed has a search box at the top, there should also be a filter drop down menu next to it.
- Given that there is a filter drop down menu next to the search box, there should be another menu to filter the listed items by the Purdue building they were found in.

- Given that there is a menu to filter the listed items by Purdue buildings, there should be a list of the different buildings that can be selected.
- Given that a building is selected on the map, The items should be sorted based on that location, from nearest to furthest.
- The User Feed should only show the items that were found in or near that area.

User Story #9

As a student user, I want to be able to filter by the date the item was lost.

#	Description	Estimated Time	Owner
1	Create a “choose a date” button within the drop down menu.	2h	Daivik
2	Display a Calendar when the “choose a date” button is pressed.	2h	Daivik
3	Highlight the areas of the Calendar that the user clicks.	2h	Daivik
4	Make the highlighted tiles of the Calendar expandable.	2h	Daivik
5	Sort items by date found.	1h	Anitej

Acceptance Criteria:

- Given that the UI for the User Feed has a search box at the top, there should also be a filter drop down menu next to it.
- Given that there is a filter drop down menu next to the search box, there should be a calendar to choose the date that the item was lost/found.
- Given that there is a Calendar, it should be possible to select the dates on that calendar when the item was found.
- Given that a date was selected, the items should be sorted based on that date.
- The User Feed should only show the items that were lost/found on that date.

User Story #10

As a student user, I want to be able to see detailed information about a lost item.

#	Description	Estimated Time	Owner
1	Add a button to see detailed info for an item on the main item listing page	1h	Mihika
2	Create UI view to see individual items in detail	3 h	Mihika
3	Get item data from the database and display it	3 h	Shlok
4	Write unit tests for the item detail retrieval	2 h	Shlok

Acceptance Criteria:

- Given the main item listing, when a user selects a particular item, then the item details page should load.
- Given the user selected a specific item, then the item details UI page should load up promptly.
- Given an item is selected, when the details page loads, then all relevant information should be visible.
- Given that multiple users are accessing the system, the page should maintain performance and load within 2 seconds.
- Given the UI has loaded up, then the database should correctly fill in all the item details that are stored.
- Given the tests are run, when the backend retrieves item data, then it should pass all data integrity checks.

User Story #11

As a student user, I want to be able to see images of the found items uploaded by desk staff.

#	Description	Estimated Time	Owner
1	Create image placeholder on the item detail page	1 h	Mihika
2	Fetch the image URLs from the backend for a specific found item	2 h	Shlok

3	Integrate the image data into the item details page to display the images	3 h	Shlok

Acceptance Criteria:

- Given the item detail page, when a student views an item with images uploaded, then the images should be fetched from the backend and displayed.
- Given the image data is being retrieved, when a student views the item detail page, then the images should load efficiently with proper resolution.
- Given an item has multiple images, when the user views the item detail page, then all images should be displayed in a scrollable or clickable gallery format.
- Given that no images were uploaded for an item, when a student views the item detail page, then a placeholder or "No images available" message should be displayed.
- Given that image data is corrupted, the system should handle the error gracefully and display a default placeholder without crashing.

User Story #12

As a student user, I want to be able to flag an item that I believe to be mine for further verification.

#	Description	Estimated Time	Owner
1	Add "Submit Claim Request" button to the item detail page	1 h	Mihika
2	Create a "Submit Proof" UI form	2 h	Mihika
3	Implement checks to make sure all the required data is filled	3 h	Shlok
4	Submit request to the backend to process the item data	3 h	Shlok
5	Send a claim approval message to staff	2 h	Shlok

Acceptance Criteria:

- Given the item detail page, when a student believes the item is theirs, then they should be able to click the "Submit Claim Request" button.
- Given the student clicks the "Submit Claim Request" button, when the "Submit Proof" form appears, then the user should be able to input all required information.
- Given the "Submit Proof" form, when the student submits the claim request, then the system should validate that all required fields are filled in before processing the request.

- Given the backend processes the claim, when all data is successfully submitted, then a claim approval message should be sent to the staff for further verification.
- Given the claim request is incomplete, when the student tries to submit the form, then an error message should be displayed prompting them to fill in the missing information.

User Story #13

As a staff user, I want to log in with a secure staff-only authentication

.

#	Description	Estimated Time	Owner
1	Design UI for staff login with input fields for credentials.	2h	Daivik
2	Create database schema to store staff credentials	5h	Daniel
3	Add validation logic to verify correct credentials during login.	2h	Daniel

Acceptance Criteria:

- Given that the staff login UI is implemented, the staff should be able to input credentials and submit for authentication.
- If valid credentials are provided, the staff user should be logged in successfully and redirected to the staff dashboard.
- Given that the staff client as been logged in, staff-only functionalities should be visible on the UI
- If a student attempts to log in as a staff member, they should be prompted to login via the student-only portal.

User Story #14

As a staff user, I want to be able to enter a new found item into the system with details (location, description).

#	Description	Estimated Time	Owner
1	Design UI for entering found item details (location, description, etc.).	4h	Shreya
2	Implement database schema to store	4h	Shlok

	found item details.		
3	Implement input validation for required fields (e.g., location, description).	2h	Shlok
4	Display entered items in an updated system's staff view.	3h	Shreya
5	Create unit tests to check that database is updated accordingly	1h	Shlok

Acceptance Criteria:

- Given the user has access to the found item entry form, they should be able to enter all necessary details (location, description).
- On submitting the form, the new found item should be stored in the database and visible in the staff view.
- If any required field is missing, an error message should be displayed, prompting the user to complete the form.
- All implemented unit tests should run successfully.

User Story #15

As a staff user, I want to upload an image of a found item to the system.

#	Description	Estimated Time	Owner
1	Design UI for uploading an image with a file input option	3h	Anitej
2	Implement backend API to handle image uploads.	3h	Anitej
3	Implement validation to ensure only image files (JPG, PNG, etc.) can be uploaded.	2h	Anitej
4	Display uploaded image in the system’s staff view alongside the item details.	3h	Anitej

Acceptance Criteria:

- The staff user should be able to upload an image of a found item through the UI.
- Only valid image formats (e.g., JPG, PNG) should be accepted.
- Once uploaded, the image should be linked to the found item in the system.

- The uploaded image should be viewable in the system's staff view alongside the found item's details.

User Story #16

As a staff user, I want to tag the item with specific characteristics (e.g., size, brand, color).

#	Description	Estimated Time	Owner
1	Design UI for displaying and selecting possible tags for items	3h	Shreya
2	Store item characteristics in the database	1h	Shreya
3	Display the characteristics in the system's student view alongside the item details.	1h	Shreya
4	Create unit tests to check that the database is being updated accordingly	1h	Shreya

Acceptance Criteria:

- Given the staff client has a homepage to input items into the system, there should be text fields for the staff client to specify about the item like the size, color, shape, etc of the item.
- These fields should be clearly labeled, allowing the staff client to easily input necessary information about the item.
- Given these fields, the user should be able to be able to change these details at any given point in the future.
- The staff client should also be able to select generated/popular characteristics for the item via a drop down.
- The staff client should also be allowed to enter "additional notes" for an item.

User Story #17

As a staff user, I want to transfer an item to the central lost and found facility in the system.

#	Description	Estimated Time	Owner
1	Design UI for transfering an item to the central lost and found facility	2h	Shreya

2	Transfer an item from the lost items list to the archived items list in the database	3h	Shreya
3	Create unit test to check that database is getting updated accordingly	1h	Shreya

Acceptance Criteria:

- Given the staff client has a page to view all the lost items, there should be a button to archive / move the item to the central lost and found facility.
- Given this button, the staff client should be able to click it to archive / move the item to the central lost and found facility.
- The staff client should be able to view the location of the lost and found facility/select a specific lost and found facility from a list of facilities if applicable.
- The staff client should be able to undo this action and make changes at any point in the future

User Story #18

As a staff user, I want to be able to modify pre-existing items in the system.

#	Description	Estimated Time	Owner
1	Design UI for modifying an item	3h	Anitej
2	Update item details in the database	3h	Shreya
3	Add validation checks for modified data (format, required fields, etc)	3h	Shreya

Acceptance Criteria:

- Given the staff user is on the item listings page, there should be a “modify” button next to each item.
- When the staff user hits the “modify” button, they should be redirected to a form.
- Given that the staff user is redirected to the modify form, the current item details should be pre-filled in the form.
- Given that the form is filled, the system should validate necessary fields before saving changes.
- Given that the form is submitted, the item details should be updated in the database.

User Story #19

As a staff user, I want to flag items for special attention (e.g., high-value items).

#	Description	Estimated Time	Owner
1	Design UI for flagging an item in item listings page	3h	Anitej
2	Implement backend functionality to update item in database	3h	Shlok
3	Implement “Claim request” form to be visible to student client	3h	Shreya
4	Implement “Claim review” for staff client to review that claim request sent by student	3h	Shreya
5	Create unit tests to check that the database reflects interim stages of the claim process	2h	Shlok

Acceptance Criteria:

- Given the staff client is displayed the option to flag an item as “high-value”, the database should reflect this change for the specific item subsequently.
- When a student client sends a request to claim an item flagged “high value”, they should be prompted to fill a “claim request” form.
- The claim request form should allow the student to submit proof of ownership of the item (photos, videos, etc.) or select an option to provide proof in person (via questions about the appearance of the item, location found, etc.)
- The claim request should be visible to the staff client once it is submitted. The staff client will have the ability to review, accept, or proceed (in case the in-person proof option is selected by the student client) with the claim process.
- The database should be updated with what interim stage of the review process the item is in.

Total Hours Per Person for Sprint 1:

#	**Name**	**Total Time**
1	Anitej	30
2	Daivik	31
3	Daniel	30
4	Mihika	30
5	Shlok	30
6	Shreya	31

Gantt Chart for Sprint 1:

Included at the end.

Remaining Backlog:

Functional Requirements:

Student Side -

1. ~~As a user, I want to have the option to create an account with my email.~~
2. ~~As a user, I want my details (preferred name and pronouns) to be assigned to my account.~~
3. ~~As a user, I want to have a reset password option.~~
4. ~~As a user, I want to be able to deactivate my account.~~
5. ~~As a user, I want to view a list of all lost items on the platform.~~
6. ~~As a user, I want to be able to search listings with keywords.~~
7. ~~As a user, I want to filter lost items by category.~~
8. ~~As a user, I want to see detailed information about a lost item.~~
9. ~~As a user, I want to see images of the found items uploaded by desk staff.~~
10. ~~As a user, I want to be able to filter items by location (where they were lost or found).~~
11. ~~As a user, I want to be able to filter by the date the item was lost.~~
12. ~~As a user, I want to flag an item that I believe to be mine for further verification.~~
13. As a user, I want to view the current status of my flagged items (claimed, in dispute, etc.).
14. As a user, I want to pre-input details like description, date, and location about my lost item to help match found items.
15. As a user, I want to receive a notification when an item matching my description of a lost item is found.
16. As a user, I want to be able to post a template of my lost item request on social media (if time allows).
17. As a user, I want to claim my item through a verification process (e.g., providing additional details or proof).
18. As a user, I want to report an item I have found.
19. As a user, I want to see a map of the nearby help desks to drop off the items I found.
20. As a user, I want to view the items I have reported as found.
21. As a user, I want to modify my claim on a lost item in case of error.
22. As a user, I want to get a map view of the locations where lost items were reported.
23. As a user, I want to see the top items being reported as lost (based on trends).
24. As a user, I want to be able to “register” a personal item: generate a unique identifier/QR code for a personal item that maps directly to my user account and item details.
25. As a user, I want to be able to save the QR as a printable sticker and be provided steps on printing at a designated location (WALC).
26. As a user, I want to be able to view all, add and delete my registered items.
27. As a user, I want to see a list of all previously claimed items.

28. As a user, I want to have my recently claimed items to be automatically registered in my account.
29. As a user, I want to start a claim dispute form on wrongly claimed items.
30. As a user, I want to be able to communicate with the help desk in case of disputes.
31. As a user, I want to leave a review or feedback on the lost and found service.
32. As a user, I want to be able to view a listing of items moved to the surplus store for sale (if time allows).
33. As a user, I want to be able to buy items online from the surplus store (if time allows).

Front Desk/Staff Side -

1. ~~As a staff member, I want to log in with a secure staff-only authentication.~~
2. ~~As a staff member, I want to enter a new found item into the system with details (location, description).~~
3. ~~As a staff member, I want to upload an image of a found item to the system.~~
4. ~~As a staff member, I want to tag the item with specific characteristics (e.g., size, brand, color).~~
5. As a staff member, I would like to generate searchable keyword descriptions for listings using visual search AI.
6. ~~As a staff member, I want to transfer an item to the central lost and found facility in the system.~~
7. As a staff member, I want to notify the system when an item has been claimed by its rightful owner.
8. ~~As a staff user, I want to be able to modify pre-existing items in the system.~~
9. ~~As a staff member, I want to flag items for special attention (e.g., high-value items).~~
10. As a staff member, I want to log disputes between students over claimed items.
11. As a staff member, I want to see a list of all unclaimed items that have been in the system for more than 1 week.
12. As a staff member, I want to track the history of the movement of an item across help desks.
13. As a staff member, I want to archive items that have been claimed or moved to the surplus store for sale.
14. As a staff member, I want to manage listings on the online surplus store marketplace (if time allows).
15. As a staff member, I want to print a receipt for students when they claim a lost item.
16. As a staff member, I want to see reports on lost and found activity (number of items, most common locations).
17. As a staff member, I want to be able to scan pre registered items for easy verification of ownership.
18. As a staff member, I want to send alerts to students who have claimed items for pickup.
19. As a staff member, I want to verify students' identification before releasing an item.

20. As a staff member, I want to delete or edit incorrect entries in the system.
21. As a staff member, I want to categorize found items in bulk (for batch uploads).
22. As a staff member, I want to use filters to quickly find items that have been reported as lost by a specific student.
23. As a staff member, I want to see the real-time status of all items at my location.
24. As a staff member, I want to see which items have been flagged for review by students.
25. As a staff member, I want to receive automated reminders to transfer items to the central lost and found.
26. As a staff member, I want to view activity logs for each item to ensure accountability.
27. As a staff member, I want to post weekly unclaimed item listings to social media (if time allows).

Non-functional Requirements:

- **Architecture and Performance**

  Database: sqlite3
  Frontend: React
  Backend: Flask

  BoilerTrack will use a modern web architecture with a React frontend and a Flask backend. We will use SQLite for data storage. The frontend built with React will provide a user-friendly interface for the reporting and claiming of lost items. The Flask backend, powered by Python, will handle the core application logic, manage data processing, API requests, and integrations with the database. The lightweight SQLite database will store all item-related data and user-account related data. RESTful APIs will ensure smooth communication between the front and back ends, handling tasks like user authentication and claim management. This architecture will provide an efficient and scalable solution to centralize lost and found operations across Purdue University.

- **Security**

  For BoilerTrack, we need to securely store user accounts. We plan on using Supabase for security of user account information. We will use OAuth with authentik. University Staff will have special permissions to add/update/delete items that regular users will not have.

- **Usability**

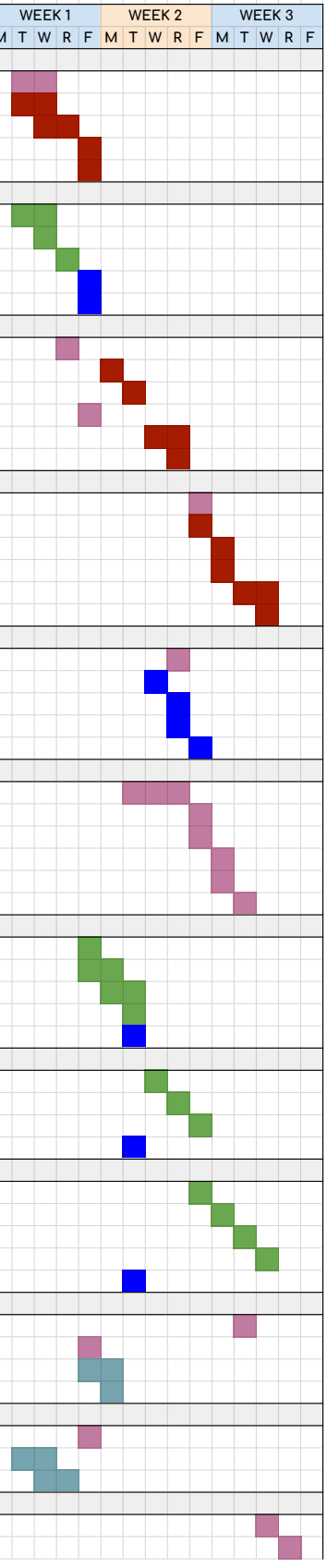
  The UI should be minimalist, showing only essential information like search fields and item lists. We want the main page to load in 500 ms. Navigation must be easy, with users able to browse, search, and report items within a few clicks. Consistent design across all pages is key. Feedback should include clear success and error messages. Search should allow filtering by date, location, and item type, with fast, relevant results. A “Help” or FAQ section will address common questions. The app must stay under 1 GB by limiting image sizes and unnecessary code, ensuring no lag with efficient API calls and database management. We want to load 100 concurrent instances.

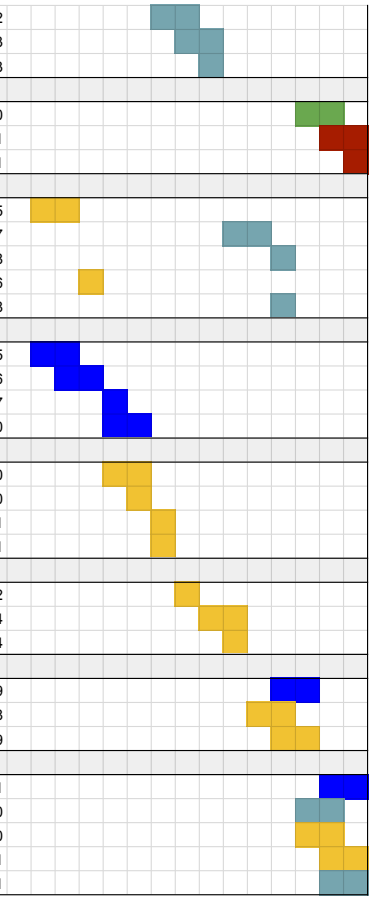
- **Hosting/Deployment**

  We will be hosting in containers for easy reproducibility using docker-compose on a free Oracle vps. This will allow us to bring up all the services (like the container for authentik, and its supporting entourage) for the application in one command.

TEAM	19
SPRINT	1
TEAM MEMBERS	Anitej, Daivik, Daniel, Mihika, Shlok, Shreya

STORY NUMBER	TASK DESCRIPTION	OWNER	START DATE	DUE DATE
1	**Student User: Account Creation and Login**			
	Create UI for inputting email and password.	Mihika	9/24	9/25
	Store email address in relation to a unique user.	Daniel	9/24	9/25
	Add verification for email address via confirmation email.	Daniel	9/25	9/26
	Store password in relation to a unique user.	Daniel	9/27	9/27
	Test user account creation by checking the database	Daniel	9/27	9/27
2	**Student User: Account Management**			
	Create UI for profile setup	Daivik	9/24	9/25
	Add field on registration page for preferred name and pronouns	Daivik	9/25	9/25
	Add field in profile setup page for preferred name and pronouns	Daivik	9/26	9/26
	Store preferred name and pronouns for the user	Anitej	9/27	9/27
	Create unit test for profile registration functionality	Anitej	9/27	9/27
3	**Student User: Forgot Password**			
	Create a reset password page on login page, with same UI, to input email address	Mihika	9/26	9/26
	Add verification for if account exists on the backend and send confirmation or error message	Daniel	9/30	9/30
	Add password reset token generation and store it in the database	Daniel	10/1	10/1
	Create page to input new password	Mihika	9/27	9/27
	Add verification of token and update password in the database	Daniel	10/2	10/3
	Create unit test to check correct functionality of password reset by checking database for relevant updates	Daniel	10/3	10/3
4	**Student User: Deactivate Account**			
	Create a page that gives the option to close an account	Mihika	10/4	10/4
	Add confirmation prompts in the form of “Please type <email/username> to confirm you want to delete your account”	Daniel	10/4	10/4
	Add final password verification if a user wants to delete their account	Daniel	10/7	10/7
	Server should verify that the logged in user is the account that will be removed	Daniel	10/7	10/7
	Server should remove the account from the database by marking it as deleted	Daniel	10/8	10/9
	Job should be run to remove accounts marked as deleted in bulk during low traffic hours	Daniel	10/9	10/9
5	**Student User: Search Listing by Keywords**			
	Add search bar to the homepage	Mihika	10/3	10/3
	Reactively offer keywords in the database to the user as they type	Anitej	10/2	10/2
	Allow a user to search multiple tags	Anitej	10/3	10/3
	Redirect the user to a results page when they enter the search	Anitej	10/3	10/3
	Results page should include information about the item like where it was returned to	Anitej	10/4	10/4
6	**Student User: View Whole Item Listing**			
	Create an “all items” page	Mihika	10/1	10/3
	Add a link to the page in the homepage	Mihika	10/4	10/4
	Add filters for major categories like “water bottle” or “headphones”	Mihika	10/4	10/4
	Have a checkbox list for keywords to also filter by	Mihika	10/7	10/7
	Include a search option to filter keywords like in story #5	Mihika	10/7	10/7
	Allow users to pin entries to the top of the table	Mihika	10/8	10/8
7	**Student User: Filter Listing by Category**			
	Create a filter button and place it next to the search box.	Daivik	9/27	9/27
	Create a drop down menu that appears when the filter button is pressed.	Daivik	9/27	9/30
	Create another drop down menu within the filter widget.	Daivik	9/30	10/1
	Add the category options to the category menu	Daivik	10/1	10/1
	Sort the items by category	Anitej	10/1	10/1
8	**Student User: Filter Listing by Location**			
	Create a “choose location” button within the filter menu.	Daivik	10/2	10/2
	Create another drop down menu within the filter widget.	Daivik	10/3	10/3
	Add the building options to the drop down menu.	Daivik	10/4	10/4
	Sort items by building found.	Anitej	10/1	10/1
9	**Student User: Filter Listing by Date**			
	Create a “choose a date” button within the drop down menu.	Daivik	10/4	10/4
	Display a Calendar when the “choose a date” button is pressed.	Daivik	10/7	10/7
	Highlight the areas of the Calendar that the user clicks.	Daivik	10/8	10/8
	Make the highlighted tiles of the Calendar expandable.	Daivik	10/9	10/9
	Sort items by date found.	Anitej	10/1	10/1
10	**Student User: Individual Item Details**			
	Add a button to see detailed info for an item on the main item listing page	Mihika	10/8	10/8
	Create UI view to see individual items in detail	Mihika	9/27	9/27
	Get item data from the database and display it	Shlok	9/27	9/30
	Write unit tests for the item detail retrieval	Shlok	9/30	9/30
11	**Student User: View Item Pictures**			
	Create image placeholder on the item detail page	Mihika	9/27	9/27
	Fetch the image URLs from the backend for a specific found item	Shlok	9/24	9/25
	Integrate the image data into the item details page to display the images	Shlok	9/25	9/26
12	**Student User: Submit Claim Request**			
	Add “Submit Claim Request” button to the item detail page	Mihika	10/9	10/9
	Create a “Submit Proof” UI form	Mihika	10/10	10/10



	Implement checks to make sure all the required data is filled	Shlok	10/1	10/2	
	Submit request to the backend to process the item data	Shlok	10/2	10/3	
	Send a claim approval message to staff	Shlok	10/3	10/3	
13	**Staff User: Account Creation and Login**				
	Design UI for staff login with input fields for credentials.	Daivik	10/9	10/10	
	Create database schema to store staff credentials	Daniel	10/10	10/11	
	Add validation logic to verify correct credentials during login.	Daniel	10/11	10/11	
14	**Staff User: Add New Item to Listing**				
	Design UI for entering found item details (location, description, etc.).	Shreya	9/24	9/25	
	Implement database schema to store found item details.	Shlok	10/4	10/7	
	Implement input validation for required fields (e.g., location, description).	Shlok	10/8	10/8	
	Display entered items in an updated system's staff view.	Shreya	9/26	9/26	
	Create unit tests to check that database is updated accordingly	Shlok	10/8	10/8	
15	**Staff User: Upload Item Picture**				
	Design UI for uploading an image with a file input option	Anitej	9/24	9/25	
	Implement backend API to handle image uploads.	Anitej	9/25	9/26	
	Implement validation to ensure only image files (JPG, PNG, etc.) can be uploaded.	Anitej	9/27	9/27	
	Display uploaded image in the system's staff view alongside the item details.	Anitej	9/27	9/30	
16	**Staff User: Include Basic Item Details**				
	Design UI for displaying and selecting possible tags for items	Shreya	9/27	9/30	
	Store item characteristics in the database	Shreya	9/30	9/30	
	Display the characteristics in the system's student view alongside the item details.	Shreya	10/1	10/1	
	Create unit tests to check that the database is being updated accordingly	Shreya	10/1	10/1	
17	**Staff User: Flag High Value Items**				
	Design UI for transfering an item to the central lost and found facility	Shreya	10/2	10/2	
	Transfer an item from the lost items list to the archived items list in the database	Shreya	10/3	10/4	
	Create unit test to check that database is getting updated accordingly	Shreya	10/4	10/4	
18	**Staff User: Modify Pre-existing Items**				
	Design UI for modifying an item	Anitej	10/8	10/9	
	Update item details in the database	Shreya	10/7	10/8	
	Add validation checks for modified data (format, required fields, etc)	Shreya	10/8	10/9	
19	**Staff User: Transfer Item to Central Lost & Found**				
	Design UI for flagging an item in item listings page	Anitej	10/10	10/11	
	Implement backend functionality to update item in database	Shlok	10/9	10/10	
	Implement “Claim request” form to be visible to student client	Shreya	10/9	10/10	
	Implement “Claim review” for staff client to review that claim request sent by student	Shreya	10/10	10/11	
	Create unit tests to check that the database reflects interim stages of the claim process	Shlok	10/10	10/11	