**ASSIGNMENT REPORT**

**for**

**CS-3009: Software Engineering**

**Prepared by Ahmed Murtaza Malik, i22-0985**

**FAST NUCES**

**April 30th, 2025**

**Group Members**

Ahmed Murtaza Mali k i22-0985

Hamza Ahmad Mir i22-1003

Ataa Ur Rasool i21-2987

**Blackbox Testing for UI Fields**

**Field:** Comment

**User Story:** Story 1

**Users:** Student

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Non-empty, within char limit | “Great post!” |
| Invalid | Weak | Empty comment | “” |
| Invalid | Weak | Exceeds char limit (500 chars) | Any string >500 characters |

**Test Cases**

| Test Case ID: TC-1-1  Test Case Description: Verify valid comment submission (non-empty, within 500 characters).  Preconditions: User is logged in and on a post page.  Inputs: Comment: "Great post!"  Expected Results: Comment is posted successfully.  Steps:   1. Navigate to the post. 2. Enter "Great post!" in the comment field. 3. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-1-2  Test Case Description: Verify invalid comment submission (empty).  Preconditions: User is logged in and on a post page.  Inputs: Comment: "" (empty string)  Expected Results: Error message: "Comment cannot be empty."  Steps:   1. Navigate to the post. 2. Leave the comment field empty. 3. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-1-3  Test Case Description: Verify invalid comment submission (exceeds 500 characters).  Preconditions: User is logged in and on a post page.  Inputs: Comment: A string of 501 characters.  Expected Results: Error message: "Comment exceeds 500 characters."  Steps:   1. Navigate to the post. 2. Enter a 501-character string. 3. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Edit Comment

**User Story:** Story 3

**Users:** Student

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Non-empty, within char limit | “Great post!” → “Bad post” |
| Invalid | Weak | Empty after edit | “Great post” → “” |
| Invalid | Weak | Exceeds char limit (500 chars) | “Great post” → Any string >500 characters |

**Test Cases**

| Test Case ID: TC-3-1  Test Case Description: Verify valid comment edit (non-empty, within 500 characters).  Preconditions: User has an existing comment.  Inputs: Edited comment: "Bad post"  Expected Results: Comment is updated successfully.  Steps:   1. Navigate to the comment. 2. Click "Edit". 3. Replace text with "Bad post". 4. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-3-2  Test Case Description: Verify invalid comment edit (empty after edit).  Preconditions: User has an existing comment.  Inputs: Edited comment: ""  Expected Results: Error message: "Comment cannot be empty."  Steps:   1. Navigate to the comment. 2. Click "Edit". 3. Delete all text. 4. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-3-3  Test Case Description: Verify invalid comment edit (exceeds 500 characters).  Preconditions: User has an existing comment.  Inputs: Edited comment: A string of 501 characters.  Expected Results: Error message: "Comment exceeds 500 characters."  Steps:   1. Navigate to the comment. 2. Click "Edit". 3. Enter a 501-character string. 4. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Delete Comment

**User Story:** Story 4

**Users:** Student

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | User owns comment | User clicks delete button → Comment is deleted |
| Invalid | Weak | User doesn’t own comment | Delete button disabled |

**Test Cases**

| Test Case ID: TC-4-1  Test Case Description: Verify valid comment deletion (user owns comment).  Preconditions: User has an existing comment.  Inputs: N/A  Expected Results: Comment is deleted.  Steps:   1. Navigate to the comment. 2. Click "Delete". 3. Confirm deletion.   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-4-2  Test Case Description: Verify invalid comment deletion (user does not own comment).  Preconditions: Comment exists, user is not the owner.  Inputs: N/A  Expected Results: Delete button is disabled.  Steps:   1. Navigate to another user’s comment. 2. Check for "Delete" button visibility.   Actual Results:  Status:  Notes: |
| --- |

**Field:** Check Application Status

**User Story:** Story 5

**Users:** Student

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Student has submitted an application | Student ID: i22-0985 → Displays "Pending" |
| Invalid | Weak | No application exists for student | Student ID: i22-0000 → "No applications found" |
| Invalid | Weak | Unauthorized access (invalid studentID) | Student ID: i23-4567 → "Access denied" |

**Test Cases**

| Test Case ID: TC-5-1  Test Case Description: Verify valid application status check (student has submitted an application).  Preconditions: Student has a pending application.  Inputs: Student ID: i22-0985  Expected Results: Displays "Pending".  Steps:   1. Navigate to "Application Status" page. 2. Enter Student ID. 3. Click "Check Status".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-5-2  Test Case Description: Verify invalid application status check (no application exists).  Preconditions: Student has no applications.  Inputs: Student ID: i22-0000  Expected Results: Displays "No applications found".  Steps:   1. Navigate to "Application Status" page. 2. Enter Student ID. 3. Click "Check Status".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-5-3  Test Case Description: Verify unauthorized access (invalid student ID).  Preconditions: User is not the student.  Inputs: Student ID: i23-4567  Expected Results: Displays "Access denied".  Steps:   1. Navigate to the "Application Status" page. 2. Enter invalid Student ID. 3. Click "Check Status".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Cancel Application Status

**User Story:** Story 6

**Users:** Student

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Student has submitted an application | Application ID: APP-001 → Status changes to "Cancelled" |
| Invalid | Weak | No application exists for student | Application ID: APP-999 → "Application not found" |

**Test Cases**

| Test Case ID: TC-6-1  Test Case Description: Verify valid cancellation (pending application).  Preconditions: Application is pending.  Inputs: Application ID: APP-001  Expected Results: Status changes to "Cancelled".  Steps:   1. Navigate to "Application Details". 2. Click "Cancel Application". 3. Confirm cancellation.   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-6-2  Test Case Description: Verify invalid cancellation (non-existent application).  Preconditions: Application does not exist.  Inputs: Application ID: APP-999  Expected Results: Error message: "Application not found".  Steps:   1. Navigate to "Application Details". 2. Enter invalid Application ID. 3. Click "Cancel".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Edit Post

**User Story:** Story 10

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Non-empty, within char limit | “Great post!” → “Bad post” |
| Invalid | Weak | Empty after edit | “Great post” → “” |
| Invalid | Weak | Exceeds char limit (1000 chars) | “Great post” → Any string >1000 characters |

**Test Cases**

| Test Case ID: TC-10-1  Test Case Description: Verify valid post edit (non-empty, within 1000 characters).  Preconditions: User owns the post.  Inputs: Edited post: "Bad post"  Expected Results: Post updated successfully.  Steps:   1. Navigate to the post. 2. Click "Edit". 3. Replace text with "Bad post". 4. Click "Save".   Actual Results:  Status:  Notes:  Notes: |
| --- |

| Test Case ID: TC-10-2  Test Case Description: Verify invalid post edit (empty after edit).  Preconditions: User owns the post.  Inputs: Edited post: ""  Expected Results: Error message: "Post cannot be empty".  Steps:   1. Navigate to the post. 2. Click "Edit". 3. Delete all text. 4. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-10-3  Test Case Description: Verify invalid post edit (exceeds 1000 characters).  Preconditions: User owns the post.  Inputs: Edited post: A string of 1001 characters.  Expected Results: Error message: "Post exceeds 1000 characters".  Steps:   1. Navigate to the post. 2. Click "Edit". 3. Enter 1001-character string. 4. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Delete Post

**User Story:** Story 11, Story 12

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | User owns post / User is admin | User clicks delete button → post is deleted |
| Invalid | Weak | User does not own post / User is not admin | Delete button disabled |

**Test Cases**

| Test Case ID: TC-11-1  Test Case Description: Verify valid post deletion (user owns post).  Preconditions: User has an existing post.  Inputs: N/A  Expected Results: Post is deleted.  Steps:   1. Navigate to the post. 2. Click "Delete". 3. Confirm deletion.   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-11-2  Test Case Description: Verify valid post deletion (admin).  Preconditions: Admin user.  Inputs: N/A  Expected Results: Post is deleted.  Steps:   1. Navigate to the post. 2. Click "Delete". 3. Confirm deletion.   Actual Results:  Status:  Notes: |
| --- |

**Field:** Attach File

**User Story:** Story 13

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Valid file size and type | 2MB PDF |
| Invalid | Weak | Invalid file type | EXE attached |
| Invalid | Weak | File exceeds size limit (20 MB) | >20 MB file size |

**Test Cases**

| Test Case ID: TC-13-1  Test Case Description: Verify valid file attachment (valid type and size).  Preconditions: User is on a post/comment page.  Inputs: File: 2MB PDF  Expected Results: File attached successfully.  Steps:   1. Navigate to the post/comment. 2. Click "Attach File". 3. Select valid file. 4. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-13-2  Test Case Description: Verify invalid file attachment (invalid type).  Preconditions: User is on a post/comment page.  Inputs: File: EXE file  Expected Results: Error message: "Invalid file type".  Steps:   1. Navigate to the post/comment. 2. Click "Attach File". 3. Select invalid file. 4. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-13-3  Test Case Description: Verify invalid file attachment (exceeds 20MB).  Preconditions: User is on a post/comment page.  Inputs: File: 21MB PDF  Expected Results: Error message: "File exceeds size limit".  Steps:   1. Navigate to the post/comment. 2. Click "Attach File". 3. Select an oversized file. 4. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Update Profile Picture

**User Story:** Story 14

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Valid image format and size | 1 MB JPEG |
| Invalid | Weak | Unsupported image format | TXT attached |
| Invalid | Weak | Image size exceeds limit (10 MB) | 15MB file attached |

**Test Cases**

| Test Case ID: TC-14-1  Test Case Description: Verify valid profile picture update (valid format and size).  Preconditions: User is logged in.  Inputs: Image: 1MB JPEG  Expected Results: Profile picture updated successfully.  Steps:   1. Navigate to "Profile Settings". 2. Click "Upload Photo". 3. Select valid image. 4. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-14-2  Test Case Description: Verify invalid profile picture update (unsupported format).  Preconditions: User is logged in.  Inputs: File: TXT file  Expected Results: Error message: "Unsupported image format".  Steps:   1. Navigate to "Profile Settings". 2. Click "Upload Photo". 3. Select invalid file. 4. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-14-3  Test Case Description: Verify invalid profile picture update (exceeds 10MB).  Preconditions: User is logged in.  Inputs: Image: 15MB JPEG  Expected Results: Error message: "Image exceeds size limit".  Steps:   1. Navigate to "Profile Settings". 2. Click "Upload Photo". 3. Select the oversized image. 4. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Update Bio

**User Story:** Story 15

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Non-empty, within char limit | “CS student at FAST” |
| Invalid | Weak | Empty bio | “” |
| Invalid | Weak | Exceeds char limit (144 chars) | Any string > 144 characters |

**Test Cases**

| Test Case ID: TC-15-1  Test Case Description: Verify valid bio update (non-empty, within 144 characters).  Preconditions: User is logged in.  Inputs: Bio: "CS student at FAST"  Expected Results: Bio updated successfully.  Steps:   1. Navigate to "Profile Settings". 2. Enter valid bio. 3. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-15-2  Test Case Description: Verify invalid bio update (empty).  Preconditions: User is logged in.  Inputs: Bio: ""  Expected Results: Error message: "Bio cannot be empty".  Steps:   1. Navigate to "Profile Settings". 2. Delete bio text. 3. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-15-3  Test Case Description: Verify invalid bio update (exceeds 144 characters).  Preconditions: User is logged in.  Inputs: Bio: A string of 145 characters.  Expected Results: Error message: "Bio exceeds 144 characters".  Steps:   1. Navigate to "Profile Settings". 2. Enter an oversized bio. 3. Click "Save".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Change Password

**User Story:** Story 16

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Correct old password, New password >10 and <30 chars | Old expected: “abcd1234defg”  Old actual: “abcd1234defg”  New password: “qwertywasdzxcv” |
| Invalid | Weak | Incorrect old password | Expected: “123”  Actual: “1234” |
| Invalid | Weak | New password <10 chars | “abc” |
| Invalid | Weak | New password >30 chars | Any string > 30 chars |

**Test Cases**

| Test Case ID: TC-16-1  Test Case Description: Verify valid password change (correct old password, new password 10-30 chars).  Preconditions: User is logged in.  Inputs: Old Password: "abcd1234defg", New Password: "qwertywasdzxcv"  Expected Results: Password updated successfully.  Steps:   1. Navigate to "Change Password". 2. Enter old and new passwords. 3. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-16-2  Test Case Description: Verify invalid password change (incorrect old password).  Preconditions: User is logged in.  Inputs: Old Password: "1234", New Password: "newpassword123"  Expected Results: Error message: "Incorrect old password".  Steps:   1. Navigate to "Change Password". 2. Enter the wrong old password. 3. Enter a new password. 4. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-16-3  Test Case Description: Verify invalid password change (new password <10 chars).  Preconditions: User is logged in.  Inputs: New Password: "abc"  Expected Results: Error message: "Password must be 10-30 characters".  Steps:   1. Navigate to "Change Password". 2. Enter a valid old password. 3. Enter a short new password. 4. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-16-4  Test Case Description: Verify invalid password change (new password >30 chars).  Preconditions: User is logged in.  Inputs: New Password: A string of 31 characters.  Expected Results: Error message: "Password must be 10-30 characters".  Steps:   1. Navigate to "Change Password". 2. Enter a valid old password. 3. Enter a long new password. 4. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Schedule Announcement

**User Story:** Story 17

**Users:** Society

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Valid date and content | Date: 2025-05-01, Content: "Annual Meet" → Scheduled |
| Invalid | Weak | Missing date or content | Date: Empty → "Date required" |
| Invalid | Weak | Past date used | Date: 2024-12-31 → "Invalid date" |

**Test Cases**

| Test Case ID: TC-17-1  Test Case Description: Verify valid announcement scheduling (valid date and content).  Preconditions: Society member is logged in.  Inputs: Date: 2025-05-01, Content: "Annual Meet"  Expected Results: Announcement is scheduled.  Steps:   1. Navigate to "Schedule Announcement". 2. Enter date and content. 3. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-17-2  Test Case Description: Verify invalid scheduling (missing date or content).  Preconditions: Society member is logged in.  Inputs: Date: Empty, Content: Empty  Expected Results: Error message: "Date required".  Steps:   1. Navigate to "Schedule Announcement". 2. Leave fields empty. 3. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-17-3  Test Case Description: Verify invalid scheduling (past date).  Preconditions: Society member is logged in.  Inputs: Date: 2024-12-31  Expected Results: Error message: "Invalid date".  Steps:   1. Navigate to "Schedule Announcement". 2. Enter the past date. 3. Click "Submit".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Assign Role

**User Story:** Story 18

**Users:** Society

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Assign valid role to society member | Member ID: MEM-001 → Role: "Operations Team Member" |
| Invalid | Weak | Assign higher role without permission | Role: "President" → "Permission denied" |

**Test Cases**

| Test Case ID: TC-18-1  Test Case Description: Verify valid role assignment (society member).  Preconditions: User has permission to assign roles.  Inputs: Member ID: MEM-001, Role: "Operations Team Member"  Expected Results: Role assigned successfully.  Steps:   1. Navigate to "Manage Roles". 2. Select member and role. 3. Click "Assign".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-18-2  Test Case Description: Verify invalid role assignment (non-member).  Preconditions: User is not a society member.  Inputs: User ID: 122-0000  Expected Results: Error message: "Not a society member".  Steps:   1. Navigate to "Manage Roles". 2. Enter non-member ID. 3. Click "Assign".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Check Role in Society

**User Story:** Story 19

**Users:** Student

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Student is a valid member of that society | Student ID: 122-0985 → Role: "Member" |
| Invalid | Weak | No society selected | Error: No society selected |

**Test Cases**

| Test Case ID: TC-19-1  Test Case Description: Verify valid role check (society member).  Preconditions: Student is a society member.  Inputs: Student ID: 122-0985  Expected Results: Displays role: "Member".  Steps:   1. Navigate to "My Roles". 2. Enter Student ID. 3. Click "Check Role".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-19-2  Test Case Description: Verify invalid role check (no society selected).  Preconditions: Student has not joined any society.  Inputs: N/A  Expected Results: Error message: "No society selected".  Steps:   1. Navigate to "My Roles". 2. Click "Check Role".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Update Venue Booking

**User Story:** Story 20

**Users:** Society

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Updating to an available slot | Booking ID: VEN-001 → New date: 2025-05-02 |
| Invalid | Weak | Updating to a booked slot | New date: 2025-05-01 (already booked) → "Slot unavailable" |
| Invalid | Weak | Updating after event date | Event date: 2025-04-25 → "Cannot update past events" |

**Test Cases**

| Test Case ID: TC-20-1  Test Case Description: Verify valid update (available slot).  Preconditions: Venue booking exists.  Inputs: Booking ID: VEN-001, New date: 2025-05-02  Expected Results: Booking updated successfully.  Steps:   1. Navigate to "Manage Bookings". 2. Select booking. 3. Enter a new date. 4. Click "Update".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-20-2  Test Case Description: Verify invalid update (booked slot).  Preconditions: Slot is already booked.  Inputs: New date: 2025-05-01  Expected Results: Error message: "Slot unavailable".  Steps:   1. Navigate to "Manage Bookings". 2. Select booking. 3. Enter booked slot date. 4. Click "Update".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-20-3  Test Case Description: Verify invalid update (past event date).  Preconditions: Event date has passed.  Inputs: New date: 2025-04-25  Expected Results: Error message: "Cannot update past events".  Steps:   1. Navigate to "Manage Bookings". 2. Select booking. 3. Enter the past date. 4. Click "Update".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Cancel Venue Booking

**User Story:** Story 21

**Users:** Society

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Cancel before deadline | Booking ID: VEN-001 → Status: "Cancelled" |
| Invalid | Weak | Cancel non-existent booking | Booking ID: VEN-999 → "Booking not found" |

**Test Cases**

| Test Case ID: TC-21-1  Test Case Description: Verify valid cancellation (before deadline).  Preconditions: Booking is active.  Inputs: Booking ID: VEN-001  Expected Results: Status changes to "Cancelled".  Steps:   1. Navigate to "Manage Bookings". 2. Select booking. 3. Click "Cancel". 4. Confirm cancellation.   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-21-2  Test Case Description: Verify invalid cancellation (non-existent booking).  Preconditions: Booking does not exist.  Inputs: Booking ID: VEN-999  Expected Results: Error message: "Booking not found".  Steps:   1. Navigate to "Manage Bookings". 2. Enter invalid Booking ID. 3. Click "Cancel".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Check Event Attendance

**User Story:** Story 23

**Users:** Society

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Past event | Event ID: EVT-001 → Attendance: 27 |
| Invalid | Weak | Future event | Event ID: EVT-002 (scheduled) → "Data unavailable" |
| Invalid | Weak | Unauthorized access | User ID: 122-0000 → "Access denied" |

**Test Cases**

| Test Case ID: TC-23-1  Test Case Description: Verify valid attendance check (past event).  Preconditions: Event has concluded.  Inputs: Event ID: EVT-001  Expected Results: Displays attendance: 27.  Steps:   1. Navigate to "Event Attendance". 2. Enter Event ID. 3. Click "Check Attendance".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-23-2  Test Case Description: Verify invalid attendance check (future event).  Preconditions: Event is scheduled.  Inputs: Event ID: EVT-002  Expected Results: Error message: "Data unavailable".  Steps:   1. Navigate to "Event Attendance". 2. Enter Event ID. 3. Click "Check Attendance".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-23-3  Test Case Description: Verify unauthorized access.  Preconditions: User is not authorized.  Inputs: User ID: 122-0000  Expected Results: Error message: "Access denied".  Steps:   1. Navigate to "Event Attendance". 2. Enter unauthorized User ID. 3. Click "Check Attendance".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Approve Venue Booking

**User Story:** Story 24

**Users:** Admin

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Approve booking | Booking ID: VEN-001 → Status: "Approved" |
| Valid | Strong | Reject booking | Booking ID: VEN-001 → Status: "Rejected" |

**Test Cases**

| Test Case ID: TC-24-1  Test Case Description: Verify valid approval (admin).  Preconditions: Booking is pending.  Inputs: Booking ID: VEN-001  Expected Results: Status changes to "Approved".  Steps:   1. Navigate to "Approve Bookings". 2. Select booking. 3. Click "Approve".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-24-2  Test Case Description: Verify invalid approval (already approved).  Preconditions: Booking is already approved.  Inputs: Booking ID: VEN-001  Expected Results: Error message: "Already approved".  Steps:   1. Navigate to "Approve Bookings". 2. Select approved booking. 3. Click "Approve".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Login

**User Story:** None

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | Correct Email and Password | “Great post!” → “Bad post” |
| Invalid | Weak | Incorrect credentials | “Great post” → “” |
| Invalid | Weak | Empty email | Email expected: “i220985@nu.edu.pk”  Email actual: “” |
| Invalid | Weak | Invalid email format | Email actual: “abcd@” |

**Test Cases**

| Test Case ID: TC-LOGIN-1  Test Case Description: Verify valid login with correct email and password.  Preconditions: User account exists.  Inputs: Email: "f220985@nu.edu.pk", Password: "123"  Expected Results: Successful login; redirected to dashboard.  Steps:   1. Navigate to the login page. 2. Enter valid email and password. 3. Click "Login".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-LOGIN-2  Test Case Description: Verify invalid login with incorrect credentials.  Preconditions: User account exists.  Inputs: Email: "f220985@nu.edu.pk", Password: "wrongpassword"  Expected Results: Error message: "Invalid credentials".  Steps:   1. Navigate to the login page. 2. Enter valid email and incorrect password. 3. Click "Login".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-LOGIN-3  Test Case Description: Verify login attempt with empty email.  Preconditions: None.  Inputs: Email: "", Password: "123"  Expected Results: Error message: "Email required".  Steps:   1. Navigate to the login page. 2. Leave the email field empty. 3. Enter password. 4. Click "Login".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-LOGIN-4  Test Case Description: Verify login attempt with invalid email format.  Preconditions: None.  Inputs: Email: "abcd@", Password: "123"  Expected Results: Error message: "Invalid email format".  Steps:   1. Navigate to the login page. 2. Enter invalid email. 3. Enter password. 4. Click "Login".   Actual Results:  Status:  Notes: |
| --- |

**Field:** Signup

**User Story:** None

**Users:** Student

**Equivalence Classes:**

| **Type** | **Strength** | **Partition** | **Test Cases** |
| --- | --- | --- | --- |
| Valid | Strong | New user with valid details | Email: "test@nu.edu.pk", Password: "Abc@1234" → Account created |
| Invalid | Weak | Existing key fields used | Email: "ahmed@nu.edu.pk" → "Email already exists" |
| Invalid | Weak | Missing required fields | Email: Empty → "Email required" |

**Test Cases**

| Test Case ID: TC-SIGNUP-1  Test Case Description: Verify valid signup (new user).  Preconditions: User does not exist.  Inputs: Email: "test@nu.edu.pk", Password: "Abc@1234"  Expected Results: Account created successfully.  Steps:   1. Navigate to the "Signup" page. 2. Enter email and password. 3. Click "Register".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-SIGNUP-2  Test Case Description: Verify invalid signup (existing email).  Preconditions: Email is already registered.  Inputs: Email: "ahmed@nu.edu.pk"  Expected Results: Error message: "Email already exists".  Steps:   1. Navigate to the "Signup" page. 2. Enter an existing email. 3. Click "Register".   Actual Results:  Status:  Notes: |
| --- |

| Test Case ID: TC-SIGNUP-3  Test Case Description: Verify invalid signup (missing fields).  Preconditions: N/A  Inputs: Email: Empty, Password: Empty  Expected Results: Error message: "Email required".  Steps:   1. Navigate to the "Signup" page. 2. Leave fields empty. 3. Click "Register".   Actual Results:  Status:  Notes: |
| --- |

**Sprint Meetings**

**Sprint 1 Planning Meeting**

| **Name** | Sprint Planning Meeting |
| --- | --- |
| **About** | Planning Meeting for Sprint 1: Core Features & User Interaction |
| **Date Time Stamp** | March 13, 2025 10:00AM - 10:30AM |
| **Attendees** | | **Present** | **Absent** | | --- | --- | | Murtaza  Hamza  Ataa | – | |
| **Topics Discussed** | * Murtaza: Reviewed backlog items for Sprint 1 (Login, Comment, Edit/Delete Post). * Hamza: Emphasized prioritizing authentication security. * Ataa: Suggested using React for frontend to speed up UI development. |
| **Conclusions** | * Finalized task assignments: Hamza (UI), Murtaza (Backend), Ataa (Security). * Agreed to use JWT for session management. * Set daily standups per convenience. |
| **Overall Duration** | 30 Minutes |
| **Next Meeting** | Sprint 1 Review on March 24, 2025 |

**Sprint 1 Review Meeting**

| **Name** | Sprint Review Meeting |
| --- | --- |
| **About** | Review Meeting for Sprint 1: Core Features & User Interaction |
| **Date Time Stamp** | March 24, 2025 10:00AM - 10:30AM |
| **Attendees** | | **Present** | **Absent** | | --- | --- | | Murtaza  Hamza  Ataa | – | |
| **Topics Discussed** | * Murtaza: Demo of login flow and role-based redirection. * Hamza: Highlighted delays in API integration. * Ataa: Praised UI responsiveness but noted styling inconsistencies. |
| **Conclusions** | * 8/10 stories completed (Login, Comment, Delete Post, etc.). * Session management and "Remember Me" delayed to Sprint 2. * UI styling to be refined in Sprint 2. |
| **Overall Duration** | 30 Minutes |
| **Next Meeting** | Sprint 2 Planning on April 16, 2025 |

**Sprint 2 Planning Meeting**

| **Name** | Sprint Planning Meeting |
| --- | --- |
| **About** | Planning Meeting for Sprint 2: Society Membership & Events |
| **Date Time Stamp** | April 16, 2025 9:00PM - 9:30PM |
| **Attendees** | | **Present** | **Absent** | | --- | --- | | Murtaza,  Hamza,  Ataa | - | |
| **Topics Discussed** | * Murtaza: Prioritized Society features (Check Application Status, Assign Roles). * Hamza: Proposed using MongoDB for realtime society member tracking. * Ataa: Requested stricter input validation for announcements. |
| **Conclusions** | * Assigned tasks: Hamza (UI for Society Dashboard), Murtaza (Backend APIs). * Agreed to use MongoDB for real-time DB communication. * Set daily standups per convenience. |
| **Overall Duration** | 30 Minutes |
| **Next Meeting** | Sprint 2 Review on April 29, 2025 |

**Sprint 2 Review Meeting**

| **Name** | Sprint Review Meeting |
| --- | --- |
| **About** | Review Meeting for Sprint 2: Society Membership & Events |
| **Date Time Stamp** | April 29, 2025 11:00PM - 11:30PM |
| **Attendees** | | **Present** | **Absent** | | --- | --- | | Murtaza,  Hamza | Ataa | |
| **Topics Discussed** | * Murtaza: Flagged minor bugs in venue booking updates. * Hamza: Demo of application approval flow and role assignment. |
| **Conclusions** | * 9/10 stories completed (Check Application Status, Assign Roles, etc.). * Venue booking bugs to be fixed between Sprint 2 and 3. * All remaining stories to be done in Sprint 3. |
| **Overall Duration** | 30 Minutes |
| **Next Meeting** | Sprint 3 Planning on May 1, 2025 |

**Sprint Backlogs**

**SPRINT 1**

**Backlog Item 1**

User Story: As a user, I want to log in or sign up to securely access the platform.

Major Task: Foundational user authentication (Login/Signup)

Subtasks:

1. Design and implement a secure login/signup API.
2. Create UI for login/signup with validation.
3. Integrate password reset ("Forgot Password?") functionality.

Associated User Stories: None

Rules Associated: Password complexity requirements, encryption for sensitive data.

**Backlog Item 2**

User Story: As a user (Student/Society/Admin), I want to comment on posts to engage in discussions.

Major Task: Implement comment functionality

Subtasks:

1. Add frontend comment input field and submission button.
2. Create a backend API to save/retrieve comments.
3. Ensure role-based permissions (e.g., Admins can delete inappropriate comments).

Associated User Stories: Story 1

Rules Associated: Comments must adhere to moderation policies.

**Backlog Item 3**

User Story: As a user, I want to edit my comments to correct mistakes or update information.

Major Task: Enable comment editing

Subtasks:

1. Add the "Edit" button to comments (visible only to the author and Admins).
2. Implement backend logic to update comments.
3. Add audit trail for edited comments.

Associated User Stories: Story 3

Rules Associated: Edits are allowed only within 24 hours of posting.

**Backlog Item 4**

User Story: As a user, I want to delete my comments to remove unwanted content.

Major Task: Enable comment deletion

Subtasks:

1. Add the "Delete" button to comments (visible to author, Admins, and Societies).
2. Implement soft/hard delete logic in the backend.
3. Update UI to reflect deletions in real time.

Associated User Stories: Story 4

Rules Associated: Admins/Societies can delete any comment in their scope.

**Backlog Item 5**

User Story: As an Admin, I want to edit posts to correct errors or update information.

Major Task: Allow post editing for Admins

Subtasks:

1. Add the "Edit Post" button (Admin-only visibility).
2. Create a backend API for post updates.
3. Add revision history tracking.

Associated User Stories: Story 10

Rules Associated: Edited posts trigger a notification to followers.

**Backlog Item 6**

User Story: As an Admin/Society, I want to delete posts to remove irrelevant or inappropriate content.

Major Task: Enable post deletion

Subtasks:

1. Add the "Delete Post" button (visible to Admins/Societies).
2. Implement confirmation modal before deletion.
3. Archive deleted posts for audit purposes.

Associated User Stories: Story 11

Rules Associated: Societies can only delete their own posts.

**Backlog Item 7**

User Story: As a Society, I want to attach images/files to posts to share resources.

Major Task: Add file/image upload to posts

Subtasks:

1. Design UI for file upload (drag-and-drop or browser).
2. Integrate cloud storage (e.g., AWS S3).
3. Validate file types and sizes.

Associated User Stories: Story 13

Rules Associated: File size limit: 10MB; allowed formats: JPG, PNG, PDF.

**Backlog Item 8**

User Story: As a user, I want to update my profile picture to personalize my account.

Major Task: Implement profile picture updates

Subtasks:

1. Add "Edit Profile Picture" UI component.
2. Integrate image cropping/resizing tool.
3. Store images in a CDN.

Associated User Stories: Story 14

Rules Associated: Images must be under 5MB and appropriate.

**Backlog Item 9**

User Story: As a user, I want to update my bio to share information about myself.

Major Task: Enable bio editing

Subtasks:

1. Add an editable bio field in the profile section.
2. Implement character limit (e.g., 500 characters).
3. Save updates to the database.

Associated User Stories: Story 15

Rules Associated: Bio content must follow community guidelines.

**SPRINT 2**

**Backlog Item 10**

User Story: As a user, I want to change my password to maintain account security.

Major Task: Add password change functionality

Subtasks:

1. Design "Change Password" form in user settings.
2. Verify old password before updating.
3. Send confirmation email after password change.

Associated User Stories: Story 16

Rules Associated: New password must differ from the last 3 passwords.

**Backlog Item 11**

User Story: As an Admin, I want to moderate comments to ensure compliance with policies.

Major Task: Build comment moderation tools

Subtasks:

1. Add the "Flag Comment" button for users.
2. Create an Admin dashboard to review flagged comments.
3. Enable bulk actions (approve/delete).

Associated User Stories: Story 2

Rules Associated: Flagged comments are hidden pending review.

**Backlog Item 12**

User Story: As a Student, I want to check my society application status to track progress.

Major Task: Display application status

Subtasks:

1. Add the "Applications" section to the user dashboard.
2. Show status (Pending/Approved/Rejected) with timestamps.
3. Send email notifications for status changes.

Associated User Stories: Story 5

Rules Associated: Students can only view their own applications.

**Backlog Item 13**

User Story: As a Student, I want to cancel my society application if I change my mind.

Major Task: Allow application cancellation

Subtasks:

1. Add the "Cancel Application" button in the Applications section.
2. Implement confirmation modal.
3. Notify Society Admins of cancellation.

Associated User Stories: Story 6

Rules Associated: Cancellation is allowed only while status is "Pending."

**Backlog Item 14**

User Story: As a Society Admin, I want to approve/reject applications to manage members.

Major Task: Build application review system

Subtasks:

1. Create an Admin dashboard to view pending applications.
2. Add "Approve/Reject" buttons with optional feedback.
3. Automatically update applicant status.

Associated User Stories: Story 7

Rules Associated: Rejected applications require a reason.

**Backlog Item 15**

User Story: As a member, I want a confirmation prompt before leaving a society to avoid mistakes.

Major Task: Add exit confirmation

Subtasks:

1. Trigger modal when clicking "Leave Society."
2. Include consequences (e.g., data loss).
3. Log exit action in member history.

Associated User Stories: Story 8

Rules Associated: Members cannot rejoin for 30 days after leaving.

**Backlog Item 16**

User Story: As a Society Admin, I want to track member leave history to analyze retention.

Major Task: Implement leave history tracking

Subtasks:

1. Create a database table for member exits.
2. Add the "Leave History" tab in the Admin dashboard.
3. Export data as CSV.

Associated User Stories: Story 9

Rules Associated: Historical data is retained for 2 years.

**Backlog Item 17**

User Story: As a Society Admin, I want to schedule announcements to share updates in advance.

Major Task: Build announcement scheduler

Subtasks:

1. Add a calendar picker for scheduling.
2. Create a backend cron job for timed posts.
3. Notify Admins before publication.

Associated User Stories: Story 17

Rules Associated: Scheduled posts can be edited until published.

**Backlog Item 18**

User Story: As a Society Admin, I want to assign leadership roles to manage responsibilities.

Major Task: Enable role assignments

Subtasks:

1. Design role management UI (e.g., dropdowns).
2. Add backend logic for role permissions.
3. Send notifications to assigned members.

Associated User Stories: Story 18

Rules Associated: Admins can assign roles only within their society.

**Backlog Item 19**

User Story: As a user, I want to view my assigned role in a society to understand my responsibilities.

Major Task: Display role information

Subtasks:

1. Show role on society profile pages.
2. Ensure real-time updates when role changes.

Associated User Stories: Story 19

Rules Associated: Roles are visible only to society members.

**Backlog Item 20**

User Story: As a user, I want to view a list of all societies to explore communities.

Major Task: Create society directory

Subtasks:

1. Design searchable list UI.

Associated User Stories: Story 25

Rules Associated: Private societies are excluded unless the user is a member.

**Burndown Charts**

**SPRINT 1**

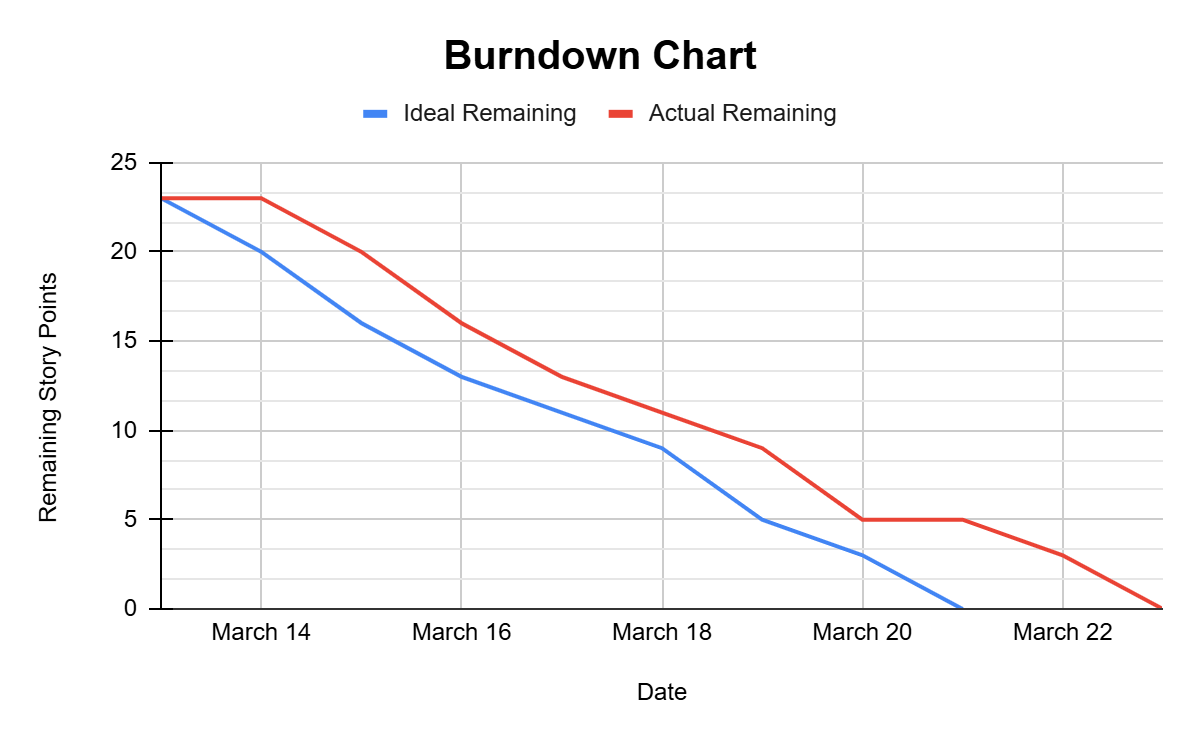
**Sprint Tasks and Estimates**

| **Task** | **Story Points** |
| --- | --- |
| Login/Signup | 3 |
| Comment functionality | 4 |
| Comment editing | 3 |
| Comment deletion | 2 |
| Post editing | 2 |
| Post deletion | 4 |
| File attachment to posts | 2 |
| Profile picture updation | 3 |
| **Total** | **23** |

**Remaining Work over Time**

| **Date** | **Ideal Remaining** | **Actual Remaining** |
| --- | --- | --- |
| March 13 | 23 | 23 |
| March 14 | 20 | 23 |
| March 15 | 16 | 20 |
| March 16 | 13 | 16 |
| March 17 | 11 | 13 |
| March 18 | 9 | 11 |
| March 19 | 5 | 9 |
| March 20 | 3 | 5 |
| March 21 | 0 | 5 |
| March 22 | - | 3 |
| March 23 | - | 0 |

**Burndown Chart**

****

**SPRINT 2**

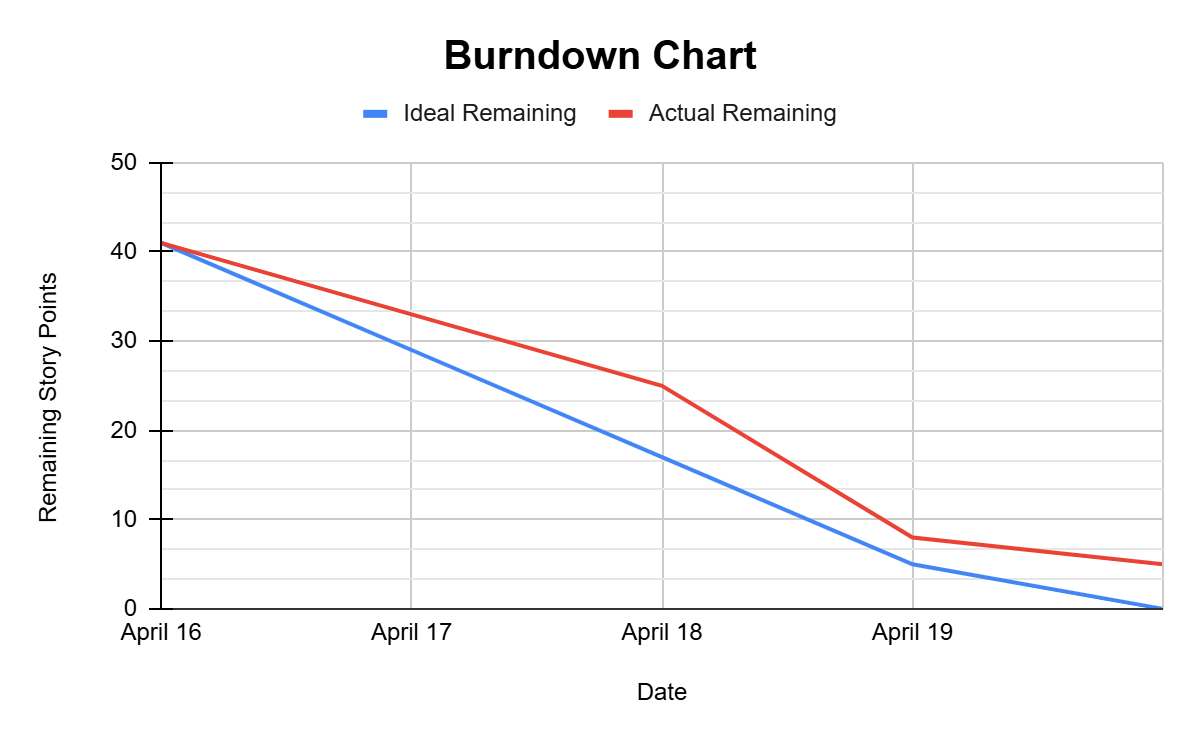
**Sprint Tasks and Estimates**

| **Task** | **Story Points** |
| --- | --- |
| Password change | 5 |
| Comment moderation | 4 |
| Application status checking | 4 |
| Application cancellation | 3 |
| Application management | 4 |
| Exit confirmation | 3 |
| History tracking | 5 |
| Announcement scheduling | 5 |
| Role assignment | 4 |
| Role info display | 2 |
| Society directory | 2 |
| **Total** | **41** |

**Remaining Work over Time**

| **Date** | **Ideal Remaining** | **Actual Remaining** |
| --- | --- | --- |
| April 16 | 41 | 41 |
| April 17 | 29 | 33 |
| April 18 | 17 | 25 |
| April 19 | 5 | 8 |
| April 20 | 0 | 5 |

**Burndown Chart**

****

**Project Planners**

**SPRINT 1**

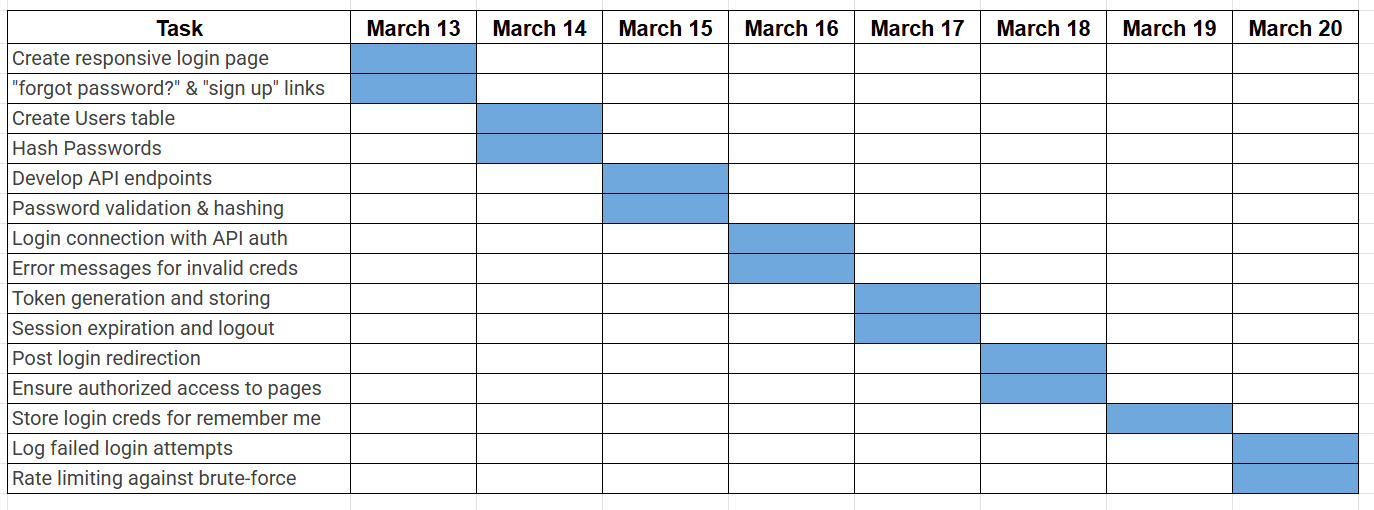
| **Date** | **Backlog Item** | **Task** | **Assignee** |
| --- | --- | --- | --- |
| March 13 | UI for Login Page | Create responsive login page | Hamza |
| March 13 | UI for Login Page | Add "forgot password?" & "sign up" links | Hamza |
| March 14 | Database Setup | Create Users table | Murtaza |
| March 14 | Database Setup | Hash passwords | Murtaza |
| March 15 | Backend Authentication | Develop API endpoints | Hamza |
| March 15 | Backend Authentication | Password hashing and validation | Hamza |
| March 16 | Login/Frontend integration | Login connection with auth API | Murtaza |
| March 16 | Login/Frontend integration | Error messages for invalid credentials | Murtaza |
| March 17 | Session management | Token generation and storing | Hamza |
| March 17 | Session management | Session expiration and logout | Hamza |
| March 18 | Role-based redirection | Post-login redirection | Ataa |
| March 18 | Role-based redirection | Ensure authorized access to restricted pages | Ataa |
| March 19 | Remember Me | Store login credentials for remember me | Murtaza |
| March 20 | Logging & Security | Log failed login attempts | Ataa |
| March 20 | Logging & Security | Rate limiting against brute-force attacks | Ataa |

**SPRINT 2**

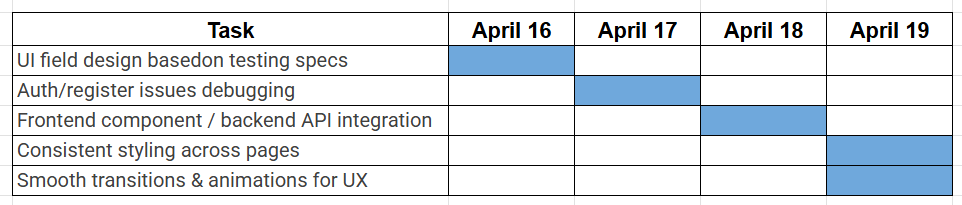
| **Date** | **Backlog Item** | **Task** | **Assignee** |
| --- | --- | --- | --- |
| April 16 | UI Fields for Blackbox testing | UI field design based on testing specs | Hamza |
| April 17 | Auth/register error fixing | Auth/registration issues debugging | Murtaza |
| April 18 | UI/Backend Connection | Front component / backend API integration | Hamza |
| April 19 | UI Styling & Animations | Consistent styling across all pages | Hamza |
| April 19 | UI Styling & Animations | Smooth transitions and animations for UX | Hamza |

**Gantt Charts**

**SPRINT 1**

****

**SPRINT 2**

****