SDA Module 3 - Part 2

1. Use Case: Follow User

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student
- **Stakeholders and Interests:**
    - **Student:** Wants to follow another user to receive updates in their feed.
    - **System:** Must enforce privacy rules (private accounts require approval).
- **Preconditions:** User is logged in. Target user's profile exists.
- **Postconditions:** The follow request is pending (if private) or active (if public).
- **Main Success Scenario:**

Actor Action	System Response
1. User navigates to another user's profile.	
2. User clicks "Follow."	
	3. System checks if the profile is public or private.
	4. If public: system adds follow relationship.
	5. If private: system sends follow request notification to the target user.

- **Extensions:**
    - 3a. If already following → show "Unfollow" option.
    - 4a. If blocked by the target user → system denies action.

2. Use Case: Manage Profile

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student
- **Stakeholders and Interests:**
    - **Student:** Wants to personalize their profile and control privacy.
    - **System:** Ensures valid inputs and safe file types.

- **Preconditions:** User is logged in.
- **Postconditions:** Profile information updated.
- **Main Success Scenario:**

Actor Action	System Response
1. User navigates to profile settings.	
2. User edits profile fields (avatar, bio, privacy setting).	
3. User saves changes.	
	4. System validates and stores updates.

- **Extensions:**
    - 2a. Invalid image format → system rejects upload.
    - 2b. Missing mandatory field → system prompts for correction.

3. Use Case: Create Post

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student
- **Stakeholders and Interests:**
    - **Student:** Wants to share a post with classmates.
    - **System:** Ensures only allowed media formats and stores securely.
- **Preconditions:** User is logged in.
- **Postconditions:** Post is created and visible on the user's profile and feed.
- **Main Success Scenario:**

Actor Action	System Response
1. User clicks "New Post."	
2. User enters text and optional hashtags/media.	
3. User submits post.	
	4. System validates input and stores post in database.

- **Extensions:**
    - 2a. Unsupported media format → system rejects.
    - 3a. Empty content → system shows error.

4. Use Case: Like/Unlike Post

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student
- **Stakeholders and Interests:**
    - **Student:** Wants to engage with content via likes.
    - **System:** Ensures uniqueness (one like per user per post).
- **Preconditions:** User is logged in. Post exists.
- **Postconditions:** Post is liked or unliked by user.
- **Main Success Scenario:**

Actor Action	**System Response**
1. User clicks “Like” on a post.	
	2. System checks whether the user already liked it. - If not liked: add like. - If liked: remove like.

- **Extensions:**
    - 2a. Post deleted before like → system shows error.

5. Use Case: View Trending Posts

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student
- **Stakeholders and Interests:**
    - **Student:** Wants to explore trending posts on campus.
    - **System:** Calculates trending posts based on like-rate.
- **Preconditions:** User is logged in.
- **Postconditions:** Trending posts are displayed.
- **Main Success Scenario:**

Actor Action	System Response
1. User navigates to “Trending” section.	

	2. System fetches posts meeting like-rate threshold in past 24 hours. 3. System displays posts in descending order of trending score.

- **Extensions:**
    - 2a. No trending posts → system shows empty state message.

6. Use Case: View Home Feed

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student
- **Stakeholders and Interests:**
    - **Student:** Wants to see recent posts from followed users.
    - **System:** Provides ordered feed with pagination.
- **Preconditions:** User is logged in.
- **Postconditions:** Home feed is displayed.
- **Main Success Scenario:**

Actor Action	**System Response**
1. User opens the app.	
	2. System fetches posts from followed users. 3. Posts are displayed in chronological order.

- **Extensions:**
    - 2a. No follows yet → system shows “Find users to follow.”

7. Use Case: Search for Content

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student (user)
- **Stakeholders and Interests:**
    - **Student:** Wants to find posts, hashtags, or users.
    - **System:** Returns fast, relevant results.
- **Preconditions:** User is logged in.
- **Postconditions:** Search results displayed.

- **Main Success Scenario:**

Actor Action	System Response
1. User types a query in the search bar.	
	2. System fetches matching users, hashtags, and posts.
	3. Results displayed grouped by category.

- **Extensions:**
    - 2a. No results found → system shows “No matches found.”

8. Use Case: Report Post

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student
- **Stakeholders and Interests:**
    - **Student:** Wants to flag inappropriate content.
    - **Admin:** Needs to review and act.
    - **System:** Logs reports for auditing.
- **Preconditions:** User is logged in. Post exists.
- **Postconditions:** Post report submitted to admin queue.
- **Main Success Scenario:**

Actor Action	System Response
1. User clicks “Report” on a post.	
2. User selects a reason (spam, offensive, etc.).	
	3. System logs report and notifies admin dashboard.

- **Extensions:**
    - 2a. Duplicate report → system logs but groups reports.

9. Use Case: Comment on Post

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student

- **Stakeholders and Interests:**
    - **Student:** Wants to share their opinion or pass a comment on a post.
    - **System:** Stores comments with correct order.
- **Preconditions:** User is logged in. Post exists.
- **Postconditions:** Comment is created and visible under post.
- **Main Success Scenario:**

Actor Action	System Response
1. User opens a post.	
2. User types a comment.	
3. User submits a comment.	
	4. System validates and saves comment.

- **Extensions:**
    - 2a. Empty comment → system rejects.

10. Use Case: Send Inquiries

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student
- **Stakeholders and Interests:**
    - **Student:** Wants to contact support/admin.
    - **Admin:** Needs to receive inquiries for resolution.
- **Preconditions:** User is logged in.
- **Postconditions:** Inquiry is sent and logged.
- **Main Success Scenario:**

Actor Action	System Response
1. User navigates to “Help/Contact.”	
2. User types message.	
3. User submits.	

	4. System stores inquiry and sends notification to admin.

- **Extensions:**
    - 2a. Empty message → system rejects.

11. Use Case: Register / Login Account

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Student (user)
- **Stakeholders and Interests:**
    - **Student:** Wants secure access with university credentials.
    - **System:** Must validate credentials and store securely.
- **Preconditions:** User has valid university email.
- **Postconditions:** User is registered or logged in.
- **Main Success Scenario:**

Actor Action	System Response
1. User opens app and chooses register or login. - For register: user provides email, password, profile info. - For login: user provides credentials.	
	2. System verifies and establishes session.

- **Extensions:**
    - 3a. Invalid email → system rejects.
    - 4a. Wrong password → system shows error.

12. Use Case: Moderate Reported Content

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Admin
- **Stakeholders and Interests:**
    - **Admin:** Needs tools to review reported content.
    - **Students:** Expect a safe environment.
    - **System:** Must maintain audit log for accountability.
- **Preconditions:** Admin logged in. Reports exist.

- **Postconditions:** Content moderated (hidden/restored) with reason logged.
- **Main Success Scenario:**

Actor Action	System Response
1. Admin opens moderation dashboard.	
	2. System lists reported posts with details.
3. Admin reviews and selects action (hide/restore).	
	4. System updates post state and records action in audit log.

- **Extensions:**
    - 3a. Admin dismisses report → no action taken, log updated.

13. Use Case: Alert All Users

- **Scope:** Ignite system
- **Level:** User goal
- **Primary Actor:** Admin
- **Stakeholders and Interests:**
    - **Admin:** Wants to broadcast important announcements or alerts to all users at once.
    - **Students:** Want to receive timely and relevant alerts (e.g., maintenance, safety, or announcements).
    - **System:** Must ensure reliable delivery and prevent spam or misuse.

- **Preconditions:** Admin is logged in with the correct privileges.
- **Postconditions:** The alert is delivered to all registered users (via in-app notification, email, or both).
- **Main Success Scenario:**

Use Case 13: Alert All Users

Actor Action	System Response
1. Admin navigates to the “Alert all users” option in the dashboard.	

2. Admin enters alert message (title, content, category, optional attachments).

3. System validates the input (checks size limits, required fields, valid file types).

4. Admin confirms and sends the alert.

5. System logs the alert in the database and audit log

6. System delivers the alert to all users via the configured channels.

7. Users receive and can view the alert.

- **Extensions:**
  - **2a. Empty message field** → system rejects and prompts for correction.
  - **2b. Invalid attachment type or size** → system rejects file and shows allowed formats.
  - **4a. Admin cancels before sending** → no alert is delivered.
  - **6a. Delivery channel failure (e.g., email service down)** → system retries and logs partial failure.