Realistic Social Media Project - Simplified

Let me give you a **much simpler but complete project** that hits ALL requirements without overwhelming you.

Simplified Project: "PostHub - Mini Social Network"

What It Does (Keep It Simple):

- Users can register and login
- Create text posts
- Like posts
- Comment on posts
- View all posts in a feed
- Search posts by keyword
- Everything saves to files

What We're REMOVING:

- No follow/unfollow system (too complex)
- No user profiles (just basic info)
- No media posts (just text)
- No feed algorithms (just show all posts)
- No edit features

Simplified Class Structure (10 Classes Only)

1. User Class

```
Fields: userId, username, password, email
Methods: getters/setters only
Purpose: Store user data
```

2. Post Class (Abstract Parent)

```
Fields: postId, authorName, content, timestamp, likeCount
Methods: abstract display(), addLike(), removeLike()
Purpose: Base for all posts
```

3. TextPost Class (Child)

```
Extends: Post
Additional: wordCount field
```

```
Override: display() method
Purpose: Concrete post implementation
```

4. StatusPost Class (Child)

```
Extends: Post
Additional: mood field (Happy/Sad/Excited)
Override: display() method
Purpose: Second child for inheritance requirement
```

5. Comment Class

```
Fields: commentId, postId, authorName, content, timestamp
Methods: getters/setters, display()
Purpose: Comments on posts
```

6. PostManager Class

```
Fields: ArrayList<Post> posts, ArrayList<Comment> comments
Methods:
  - createPost(content, author) - creates TextPost
  - createPost(content, author, mood) - creates StatusPost [OVERLOADING]
  - deletePost(postId)
  - getAllPosts()
  - searchPosts(keyword)
  - addLike(postId)
  - addComment(postId, comment)
Purpose: Manages all posts and comments
```

7. UserManager Class

```
Fields: ArrayList<User> users
Methods:
  - addUser(user)
  - getUserByUsername(username)
  - getAllUsers()
Purpose: Manages all users
```

8. AuthManager Class

```
Fields: UserManager userManager, User currentUser
Methods:
  - register(username, password, email) - throws DuplicateUserException
  - login(username, password) - throws InvalidLoginException
  - logout()
  - getCurrentUser()
Purpose: Handles authentication
```

9. FileManager Class

```
Static Methods:
  - saveUsers(users) - throws IOException
  - loadUsers() - returns ArrayList<User>
  - savePosts(posts) - throws IOException
  - loadPosts() - returns ArrayList<Post>
  - saveComments(comments) - throws IOException
  - loadComments() - returns ArrayList<Comment>
Purpose: All file operations
```

10. SystemController Class

```
Fields: AuthManager authManager, UserManager userManager, PostManager
postManager, FileManager fileManager
Methods:
  - initialize() - loads all data
  - saveAll() - saves all data
Purpose: Main system coordinator [COMPOSITION with all managers]
```

Interfaces (2 only):

Searchable Interface

```
Methods: search(keyword)
Implemented by: PostManager
```

Likeable Interface

```
Methods: addLike(), removeLike(), getLikeCount()
Implemented by: Post
```

Exception Classes (2 only):

InvalidLoginException - wrong username/password **DuplicateUserException** - username exists

---

OOP Requirements - How They're Met

☑ **Encapsulation:** All fields private, public getters/setters ☑ **Inheritance:** Post → TextPost, StatusPost ☑ **Polymorphism:**

- Overriding: `display()` in TextPost and StatusPost
- Overloading: `createPost()` with different parameters ☑ **Abstraction:** Abstract Post class + 2 interfaces ☑ **Composition:** SystemController owns all managers (can't exist without them) ☑ **Aggregation:** PostManager has ArrayList<Post> (posts can exist independently) ☑ **Exception Handling:** 2 custom + IOException + try-catch-finally ☑ **File Handling:** Save/load from text files

---

File Structure (3 Files Only)

users.txt

```
1|john_doe|pass123|john@email.com
2|jane_smith|pass456|jane@email.com
```

posts.txt

```
1|TextPost|john_doe|Hello World!|2025-11-29 10:30|5|10
2|StatusPost|jane_smith|Great day!|2025-11-29 11:00|3|Happy
```

Format: `postId|type|author|content|timestamp|likeCount|wordCount/mood`

comments.txt

```
1|1|jane_smith|Nice post!|2025-11-29 10:35
2|1|john_doe|Thanks!|2025-11-29 10:40
```

Format: `commentId|postId|author|content|timestamp`

Simple GUI Structure (5 Screens Only)

Screen 1: Login (LoginScreen.fxml)

Components:

- TextField (username)
- PasswordField (password)
- Button (Login)
- Button (Register)
- Label (error message)

Actions:

- Login → validate → open Feed
- Register → open Registration

Screen 2: Registration (RegisterScreen.fxml)

Components:

- TextField (username, email)
- PasswordField (password)
- Button (Register)
- Button (Back)
- Label (error/success)

Actions:

- Register → save user → back to Login

Screen 3: Feed (FeedScreen.fxml)

Components:

- MenuBar (Logout, Create Post, Search)
- ListView (all posts)
- Label (welcome message)
- Button (Refresh)

Actions:

- Click post → open Post Detail
- Create Post → open Create Post screen
- Logout → back to Login

Screen 4: Create Post (CreatePostScreen.fxml)

Components:

- TextArea (content)
- RadioButton (Text Post / Status Post)
- ComboBox (mood - only if Status Post)
- Button (Post)
- Button (Cancel)
- Label (character count)

Actions:

- Post → save → back to Feed

Screen 5: Post Detail (PostDetailScreen.fxml)

Components:

- Label (author, content, timestamp, likes)
- Button (Like)
- ListView (comments)
- TextArea (new comment)
- Button (Add Comment)
- Button (Back)

Actions:

- Like → increment count → save
- Add Comment → save → refresh list

Realistic 7-Day Plan

Day 1 (Today) - Entity Classes

Time: 3-4 hours

Morning (2 hours):

- Create User class (private fields + getters/setters)
- Create Comment class
- Test by creating objects

Afternoon (1-2 hours):

- Create abstract Post class
- Create TextPost extending Post
- Create StatusPost extending Post
- Test inheritance

✅ **Checkpoint:** Can create User, TextPost, StatusPost objects

---

Day 2 - Interfaces + Managers

Time: 4-5 hours

Morning (2 hours):

- Create Searchable interface
- Create Likeable interface
- Make Post implement Likeable
- Test

Afternoon (2-3 hours):

- Create UserManager class
- Create PostManager class (implement Searchable)
- Implement all basic methods
- Test: create users, create posts, search

✅ **Checkpoint:** Can manage users and posts in memory

Day 3 - Auth + System + Exceptions

Time: 4-5 hours

Morning (2 hours):

- Create InvalidLoginException
- Create DuplicateUserException
- Create AuthManager with try-catch blocks
- Test registration and login

Afternoon (2-3 hours):

- Create SystemController
- Wire everything together
- Test complete flow in console (no GUI)

✅ **Checkpoint:** Backend complete - can register, login, post, comment, like (in memory)

Day 4 - File Handling (CRITICAL)

Time: 5-6 hours

All Day:

- Create FileManager class
- Implement saveUsers/loadUsers
- Test: save → clear → load → verify
- Implement savePosts/loadPosts (handle both TextPost and StatusPost)
- Test: save → clear → load → verify
- Implement saveComments/loadComments
- Test: save → clear → load → verify

CRITICAL TEST:

- Run program → register → login → create posts → comment → like
- Close program
- Reopen program → ALL DATA SHOULD BE THERE

☑ **Checkpoint:** Data persists - THIS IS MANDATORY

Day 5 - GUI Part 1 (Login + Register + Feed)

Time: 5-6 hours

Morning (2-3 hours):

- Install Scene Builder
- Create LoginScreen.fxml (design in Scene Builder)
- Create LoginController.java
- Connect and test login/register navigation
- Implement login with try-catch, show errors

Afternoon (2-3 hours):

- Create RegisterScreen.fxml
- Create RegisterController.java
- Implement registration with exception handling
- Create FeedScreen.fxml
- Create FeedController.java
- Load posts into ListView from file

☑ **Checkpoint:** Can login → see feed with posts

Day 6 - GUI Part 2 (Create Post + Post Detail)

Time: 5-6 hours

Morning (2-3 hours):

- Create CreatePostScreen.fxml
- Create CreatePostController.java
- Implement RadioButton logic (show mood ComboBox only for StatusPost)
- Implement Post button → save → back to feed
- Test creating both post types

Afternoon (2-3 hours):

- Create PostDetailScreen.fxml
- Create PostDetailController.java
- Load post details + comments
- Implement Like button
- Implement Add Comment button
- Save changes to file

✅ **Checkpoint:** All features work through GUI

Day 7 - Testing + UML + Report

Time: 6-8 hours

Morning (2 hours): TESTING:

- Test full flow: Register → Login → Create Post → Like → Comment → Logout → Login → Data still there
- Test exceptions: wrong password, duplicate username
- Test edge cases: empty fields, special characters
- Fix bugs

Afternoon (4-6 hours):

Hour 1: UML (use my PlantUML code, simplified version below)

Hours 2-4: Report (4-5 pages)

Page 1:

- Title page
- Introduction: "PostHub is a mini social network where users can create posts, like, and comment."
- System overview: briefly describe features

Page 2:

- **Encapsulation:** Explain User class with code snippet
- **Inheritance:** Post → TextPost, StatusPost. Show UML snippet + code
- **Polymorphism:**
  - Overriding: display() method with code
  - Overloading: createPost() method with code

Page 3:

- **Abstraction:** Abstract Post + interfaces with justification
- **Composition:** SystemController owns managers. UML snippet with filled diamond
- **Aggregation:** PostManager has posts. UML snippet with hollow diamond
- **Exception Handling:** Show 2 custom exceptions + try-catch code + IOException

Page 4:

- **File Handling:** Explain 3 files, show format, show save/load code
- **Class Descriptions:** 1-2 sentences per class

Page 5:

- UML diagram (full page)

Page 6:

- 5 GUI screenshots with captions
- Conclusion: "Successfully implemented all OOP requirements"

Hour 5-6:

- Proofread report
- Test demo one more time
- Prepare to explain code

☑ **Checkpoint:** PROJECT COMPLETE! 🎉

Time Breakdown Summary

Day	Focus	Hours	Cumulative
1	Entity Classes	3-4	4
2	Interfaces + Managers	4-5	9
3	Auth + System	4-5	14
4	File Handling	5-6	20
5	GUI Part 1	5-6	26

Day	Focus	Hours	Cumulative
6	GUI Part 2	5-6	32
7	Testing + Docs	6-8	40

Total: ~40 hours over 7 days

What Makes This Realistic

☑ **Only 10 classes** (vs 15 in previous plan) ☑ **Only 5 GUI screens** (vs 8) ☑ **Only 3 text files** (vs 5) ☑ **No complex features** (no follows, no profiles, no media) ☑ **Still meets ALL requirements** from your PDF ☑ **Each day has clear, achievable goals** ☑ **Buffer time built in** for bugs and breaks

---

Emergency Shortcuts (If Behind)

Skip if necessary:

- StatusPost class (just use TextPost, explain in report you could extend to StatusPost)
- Search functionality (implement but keep very basic)
- Mood ComboBox (just use TextField)

Never skip:

- File handling
- At least one inheritance (Post → TextPost)
- Exception handling with try-catch
- Basic GUI with login and create post

Success Checklist

By Day 3: ☑ Backend works in console **By Day 4:** ☑ Data saves and loads from files **By Day 6:** ☑ GUI works for all features **By Day 7:** ☑ Report done, ready to demo

---

This is **actually doable in one week** if you dedicate 5-6 hours per day. The project is simpler but still impressive and meets every single requirement.

Start TODAY with Day 1. Don't overthink it. Just code. 💪

Good luck! You got this! 🚀