LEXIGO

Group 3:

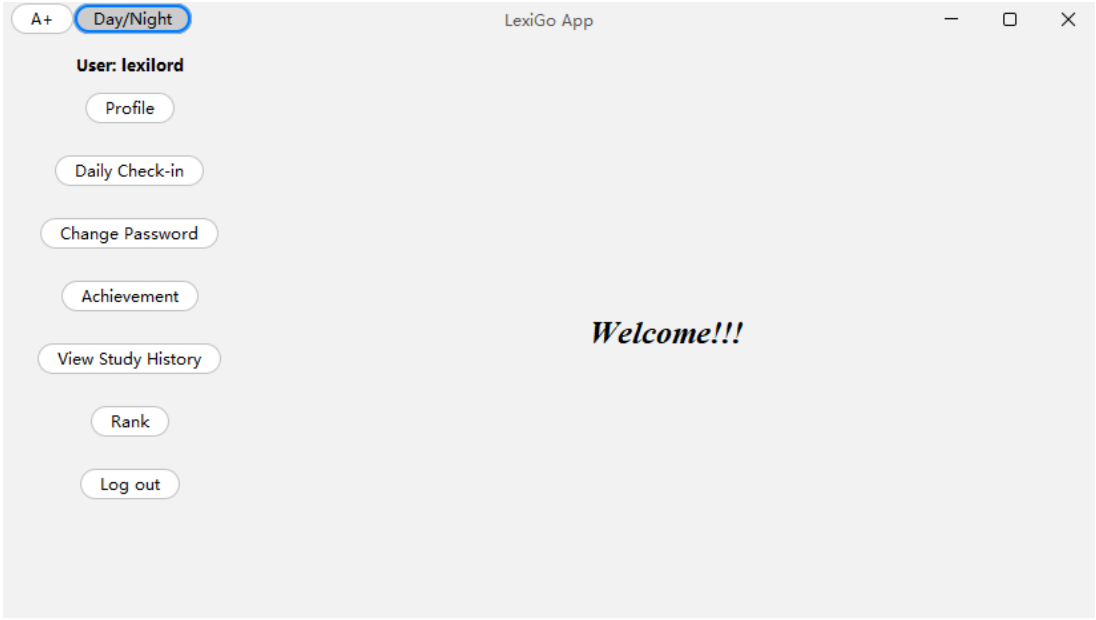
Heyuan Zhou, Jacky Huo,

Jacob Ke, Jingcheng Liang,

Yunzhao Li

SPECIFICATION

- LEXIGO is a cross-platform Java desktop application that helps English speakers learn foreign-language vocabulary through flashcard-based word decks.
- **Main Features:**
  - Flashcard-based vocabulary drills using word decks
  - Profile management for switching target languages
  - Daily check-ins and basic progress tracking
  - Example sentences and translations on every card
  - View study history / Rankings



API USAGE

Provide word details service

Provide translation service

How we use

Return data from two endpoints

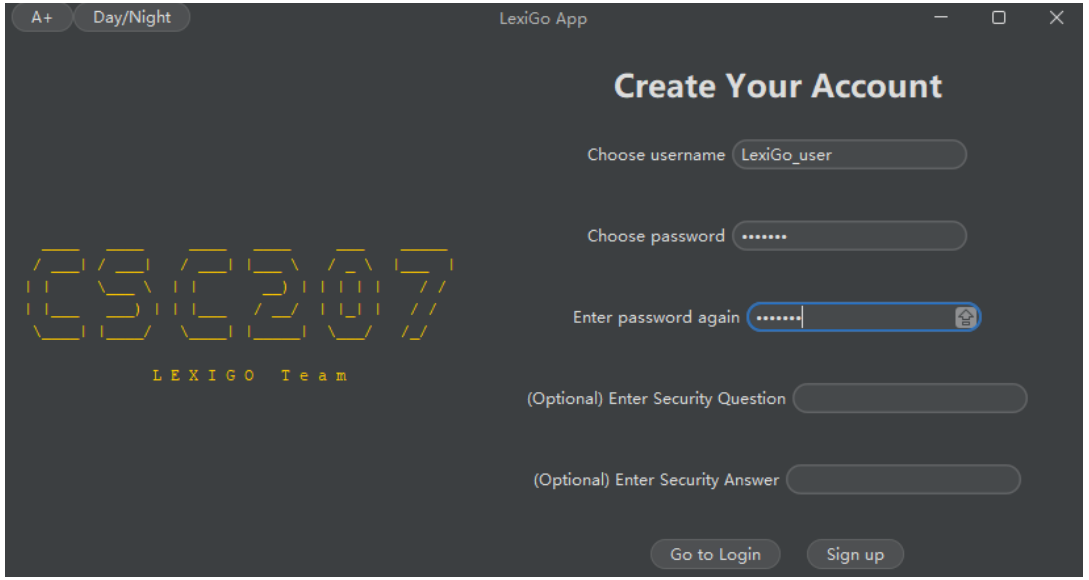

```
{
  "translations": [
    {
      "detected_source_language": "EN",
      "text": "Hallo, Welt!"
    }
```

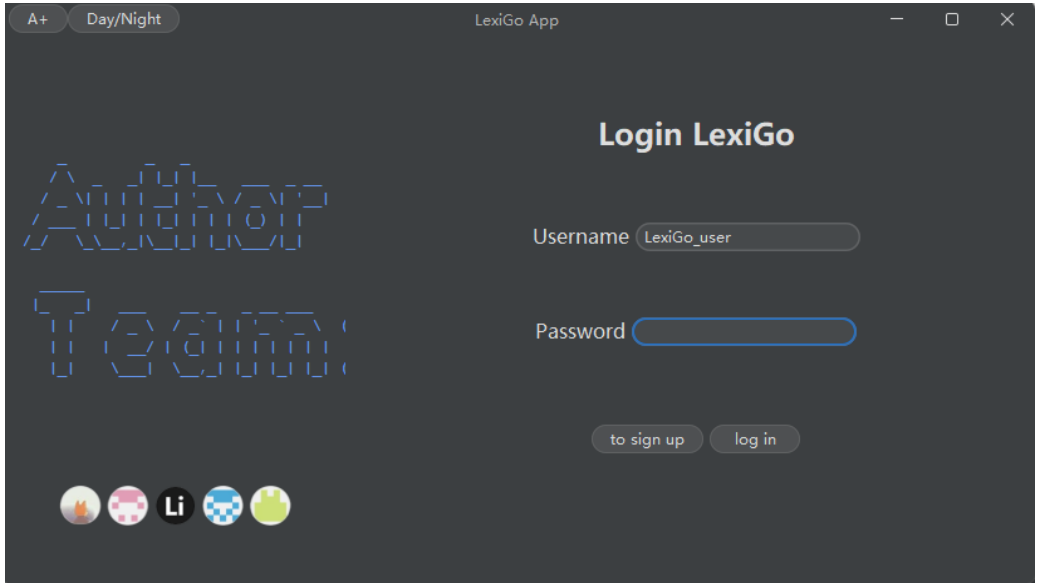


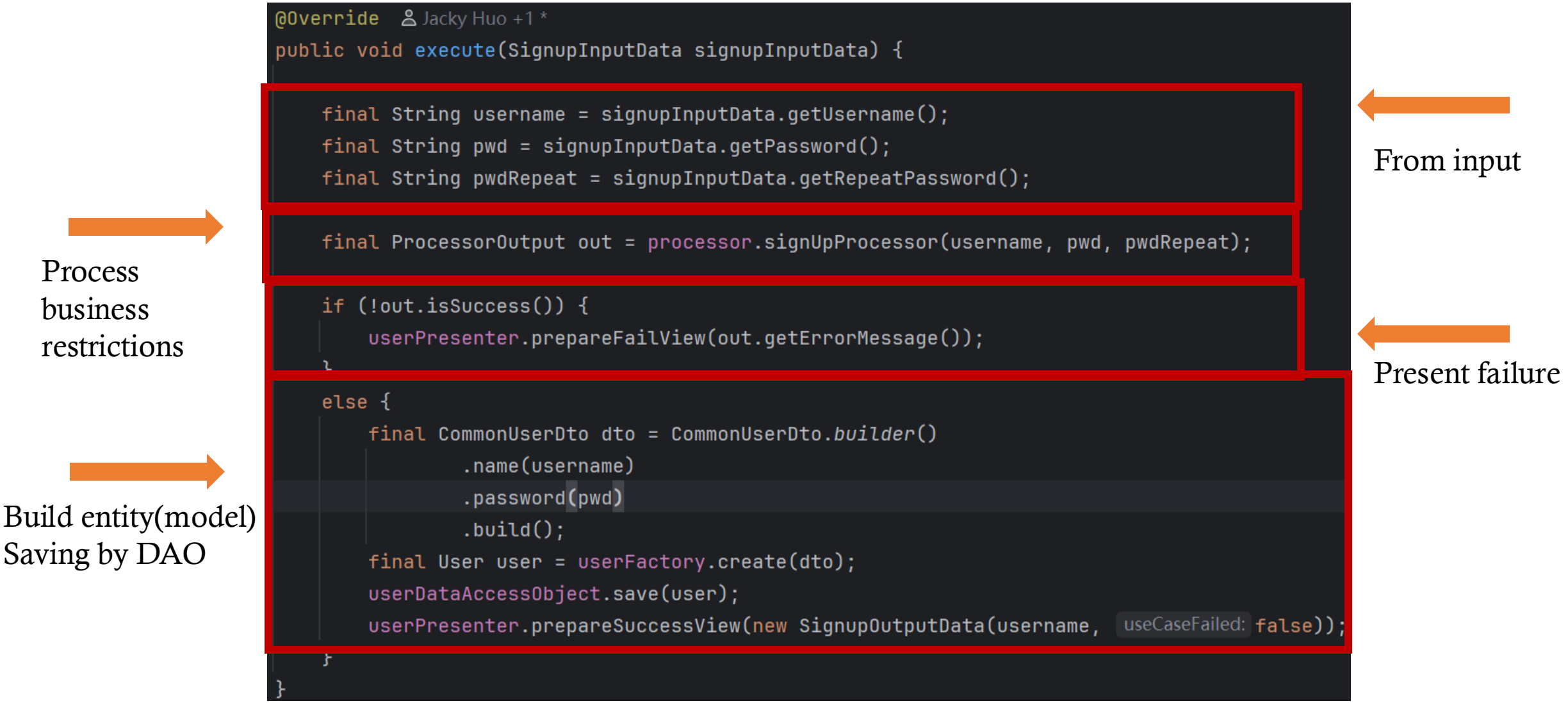
```
      "partOfSpeech": "exclamation",
      "definitions": [
        {
          "definition": "used as a greeting or to begin a phone
          "example": "hello there, Katie!",
```


GROUP CASE: SIGNUP

- As a first-time user, I want to sign up with my name and password, so that I can start using the app.
- I also want my account to be secure so I set a security question and answer to secure my account. (Optional)

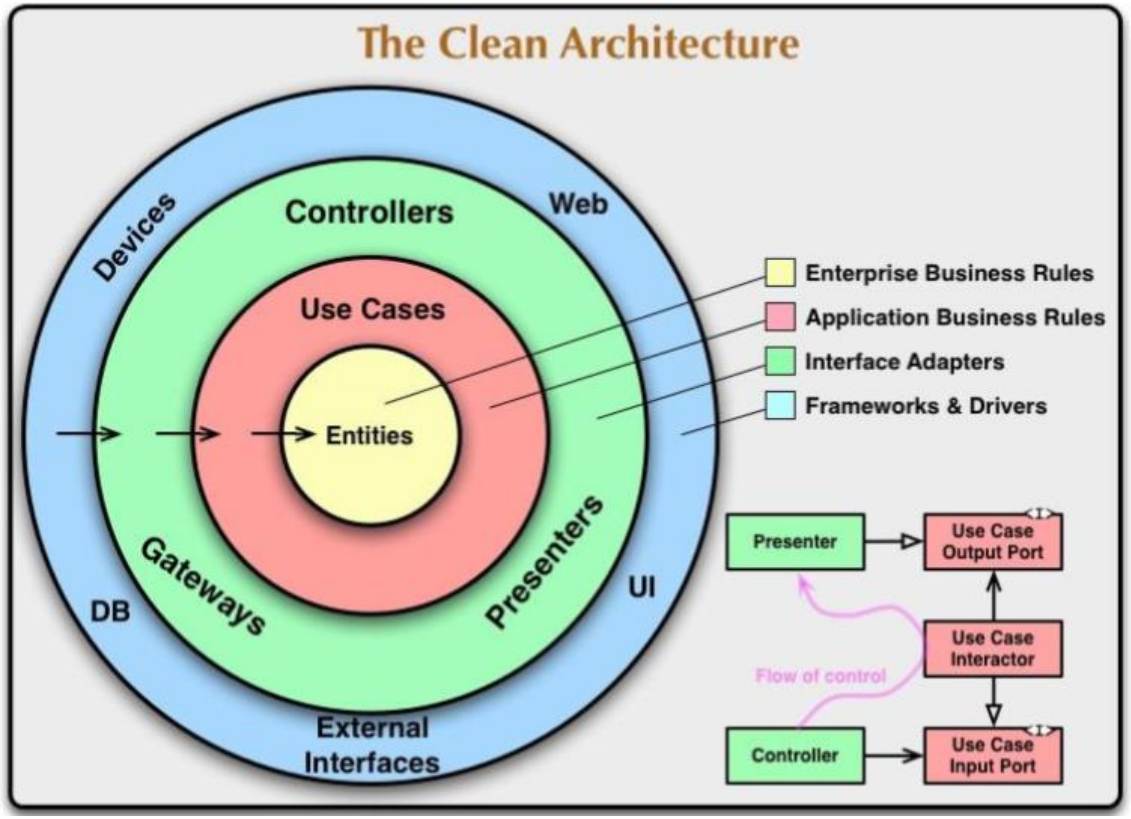






INTERACTOR CODE WALKTHROUGH

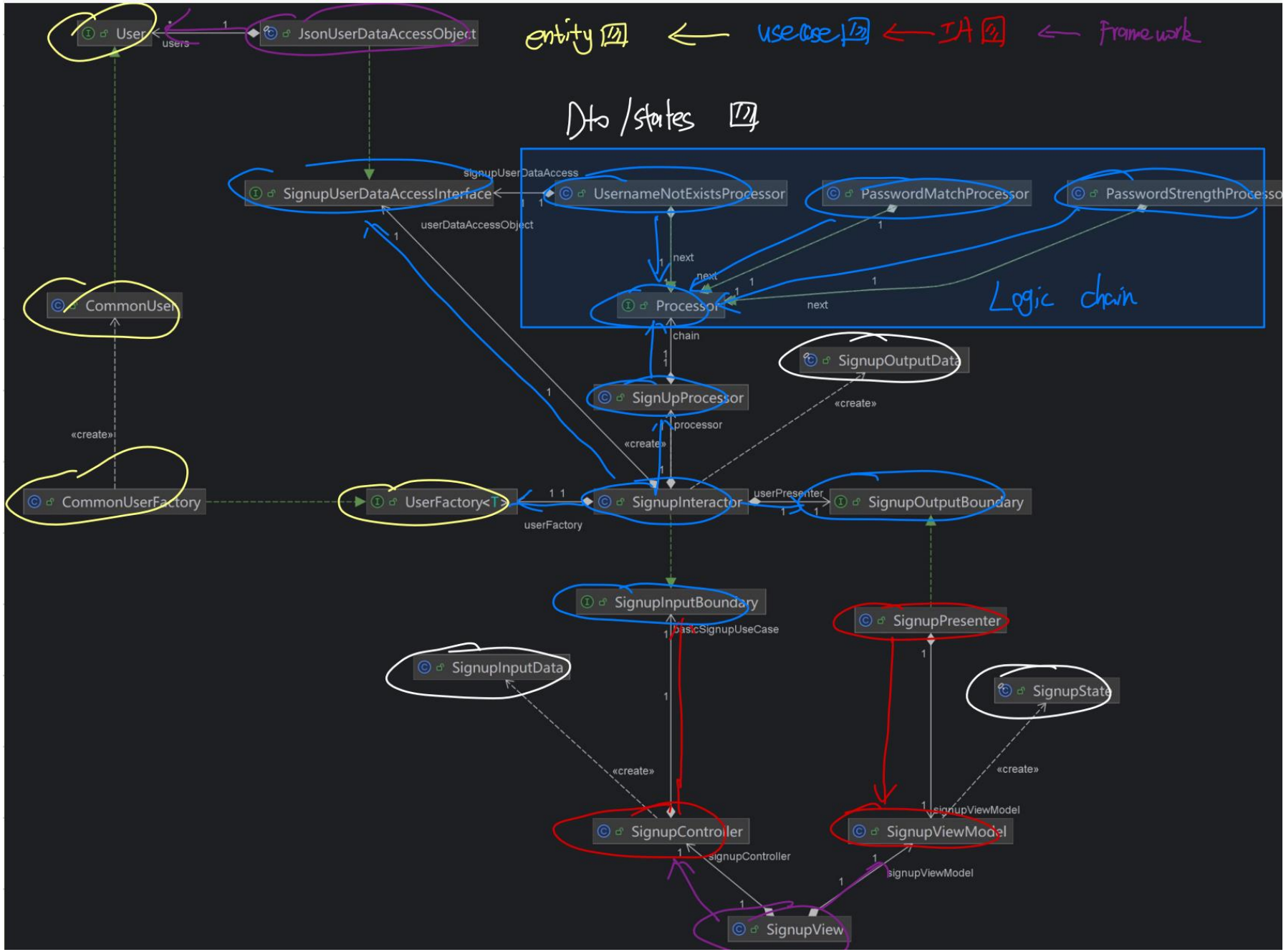
WHY CLEAN ARCHITECTURE WORKS



Our two main points of how CA works:

1. Controller accesses presenter through flow of control
2. All dependency arrows are pointing inward

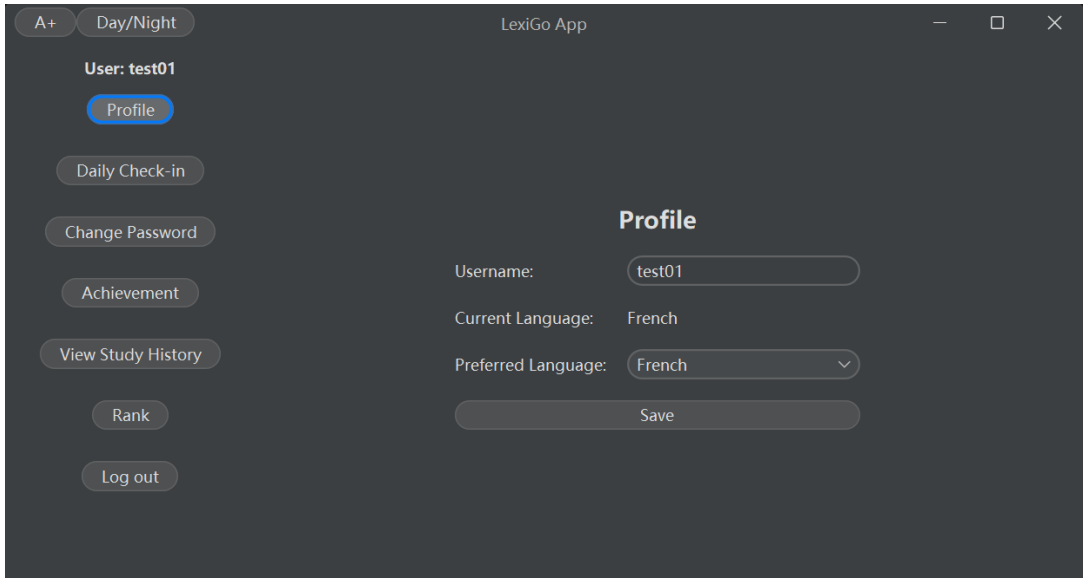
CLEAN ARCHITECTURE

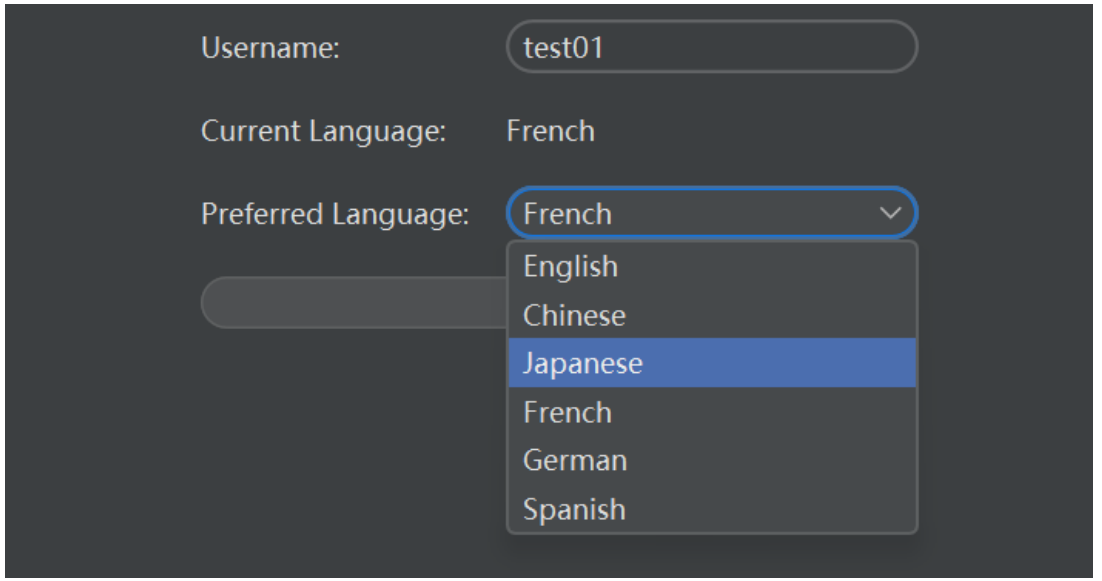


MANAGE PROFILE

Yunzhao Li

- *As a multilingual learner,*
- *I want to change my profile settings (username, target languages),*
- *so that I can switch between language courses flexibly.*





MANAGE PROFILE INTERACTOR CODE I

```
public void execute(ProfileInputData profileInputData) {
    presenter.prepareSuccessView(new ProfileOutputData(profileInputData.getUsername(),
            // get language from dao
            userDataAccess.getLanguage(profileInputData.getUsername()),
            // Languages
            Language.values()));
    // extension: when getLanguage fails, should create dtos down here to present Failure.
}
```

Presenter

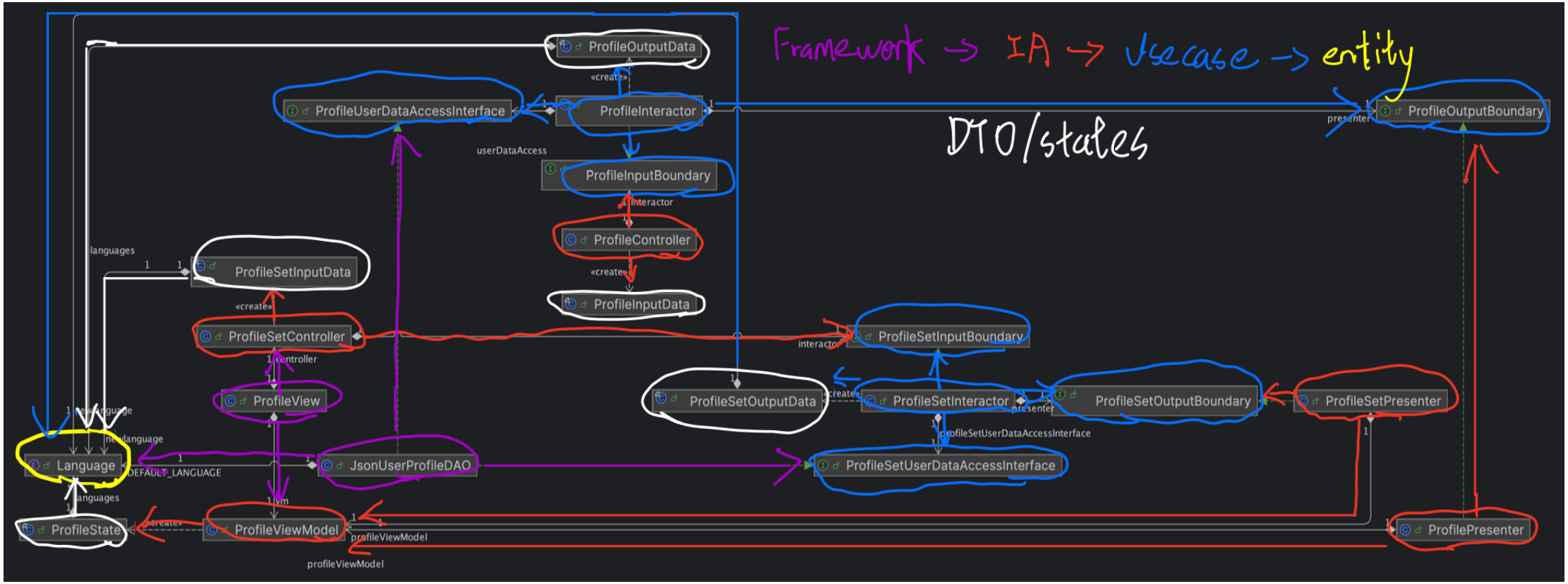
DAO

MANAGE PROFILE INTERACTOR CODE II

```java
@Override  yunzhaol +1 *
public void execute(ProfileSetInputData profileSetInputData) {
    // first time using, use default language
    if (profileSetInputData.getOldlanguage() == null) {
        // manipulate profile entity
        final PersonalProfile personalProfile = profileFactory.createPersonalProfile(profileSetInputData
                .getUsername(), profileSetInputData.getNewlanguage());
        // save through dao(here dai)
        profileSetUserDataAccessInterface.save(personalProfile);
        // present
        presenter.prepareSuccessView(new ProfileSetOutputData(profileSetInputData.getUsername(),
                profileSetInputData.getNewlanguage()));
    }
    else if (profileSetInputData.getOldlanguage().code().equals(profileSetInputData.getNewlanguage().code())) {
        // same language failure
        presenter.prepareFailView( errorMessage: "New language must be different from the old language.");
    }
    else {
        // normal case
        final PersonalProfile personalProfile = profileFactory
                .createPersonalProfile(profileSetInputData.getUsername(), profileSetInputData.getNewlanguage());
        profileSetUserDataAccessInterface.save(personalProfile);
        presenter.prepareSuccessView(new ProfileSetOutputData(profileSetInputData.getUsername(),
                profileSetInputData.getNewlanguage()));
    }
}
```

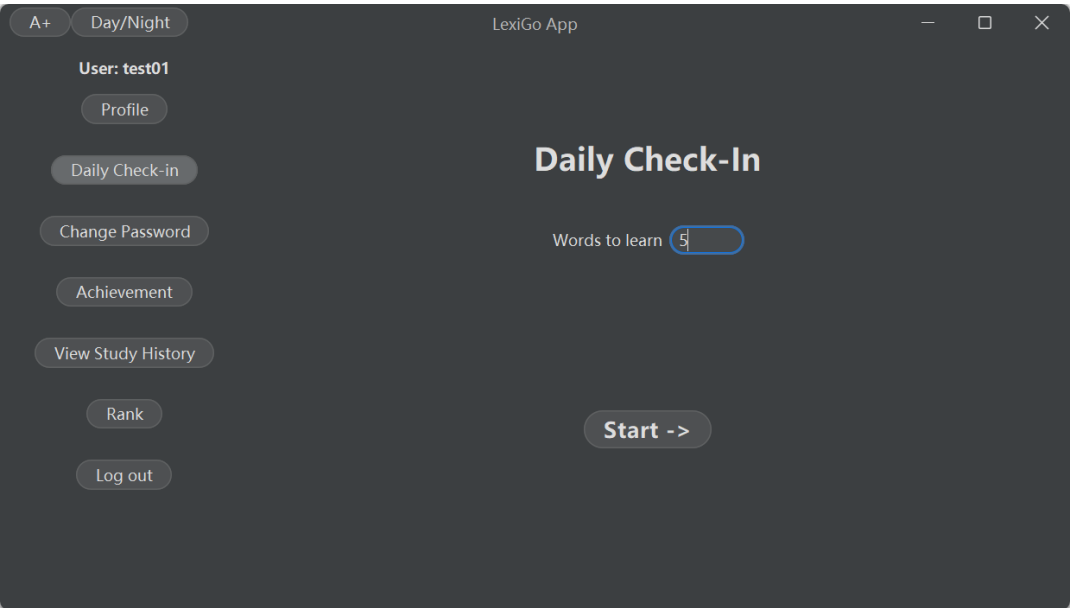
Factory creates entity

CLEAN ARCHITECTURE
PROFILE CASE



WordPlay I (Case name: Start Check in) - Haozhe Huo

Study session ready!



As a language learner,

I want to study with flip cards showing target-language words and mother-tongue translations, **so that** I can memorize vocabulary efficiently.

INTERACTOR CODE BRIEF WALKTHROUGH

```java
public void execute(StartCheckInInputData input) {
    // Format detector, as it might change, put it out of interactor
    if (formatter.execute(input.getLength())) {
        /* 1. Load domain data */
        // get wordbook dao from dto
        final WordBookAccessInterface wordbookdao;
        final UserRecordDataAccessInterface userrecorddao;

        wordbookdao = daoDto.getWordBookAccessObject();
        userrecorddao = daoDto.getUserDataAccessObject();

        // get data from DAI(injected DAO)
        final WordBook wordBook = wordbookdao.get();
        final List<LearnRecord> history = userrecorddao.get(input.getUsername());

        /* 2. Check remaining words in the book */
        // Note: as logic might not be extended, there's no strategic relocation
        final int remaining = getRemaining(history, wordBook);
        if (remaining < Integer.parseInt(input.getLength())) {
            // business logic: u cannot learn words u learnt before :)
            presenter.prepareFailView( errorMessage: "No more words to learn");
        }
        else {
            /* 3. Use strategy to pick word IDs */
            // strategy might change, so put it outside
            final List<UUID> wordIds = generator.generate(
                    wordBook,
                    history,
                    input.getLength()
            );
```

DAOs

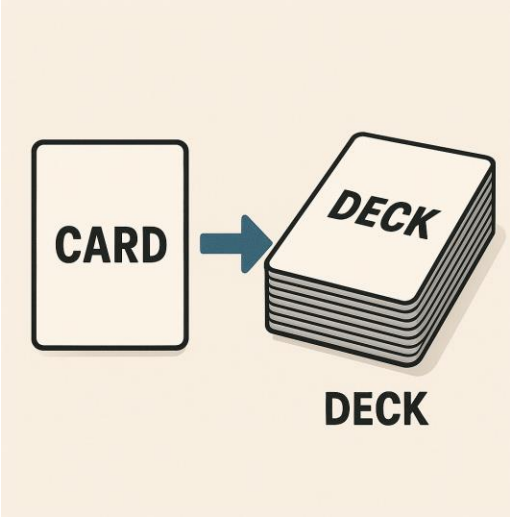
Failure case

Gnerate list based on injected EI

INTERACTOR CODE BRIEF WALKTHROUGH II

Build & manipulate entity

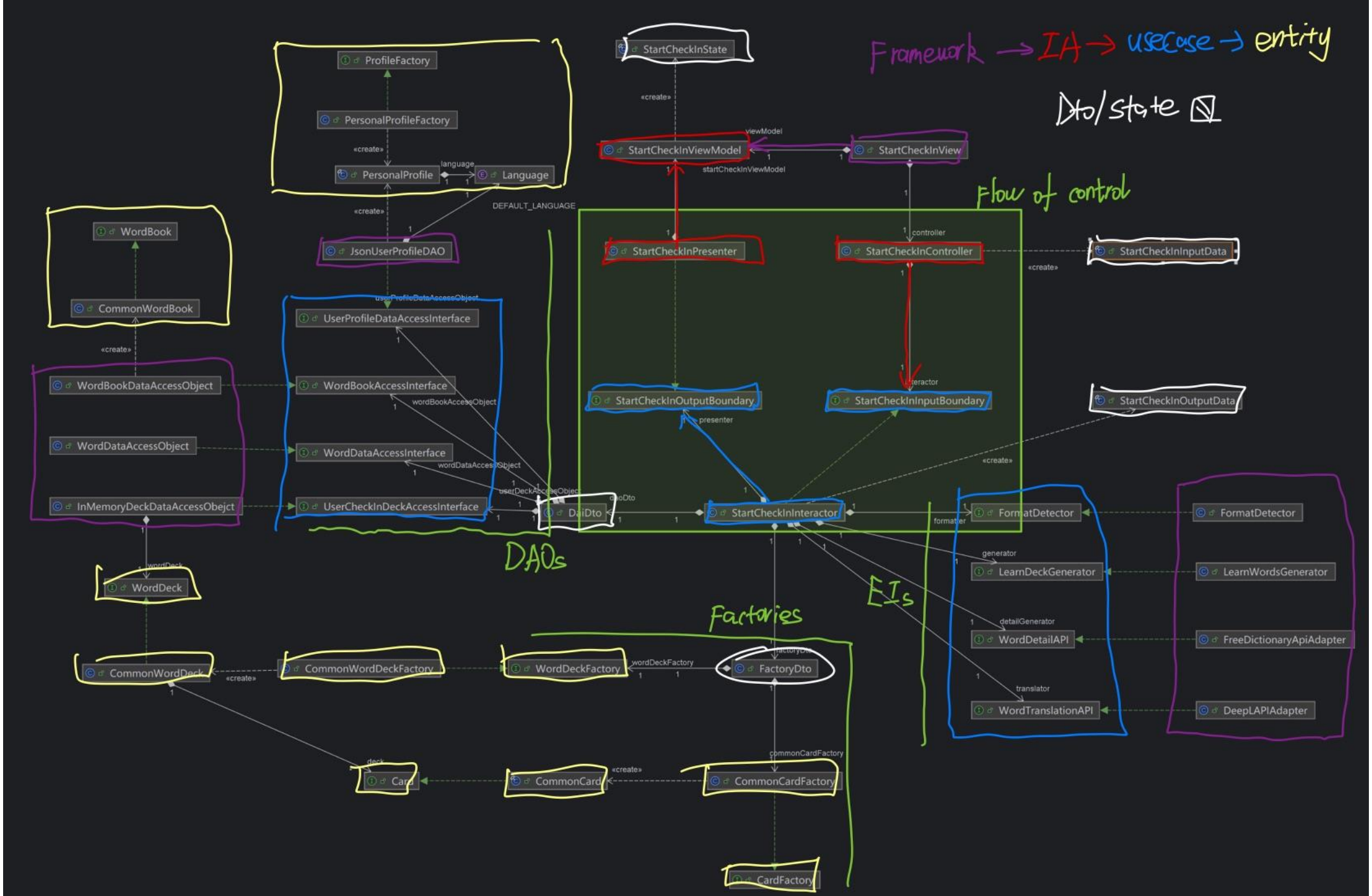
Entity factory & Dao saving



Injected Api in buildCard(wordid, username)

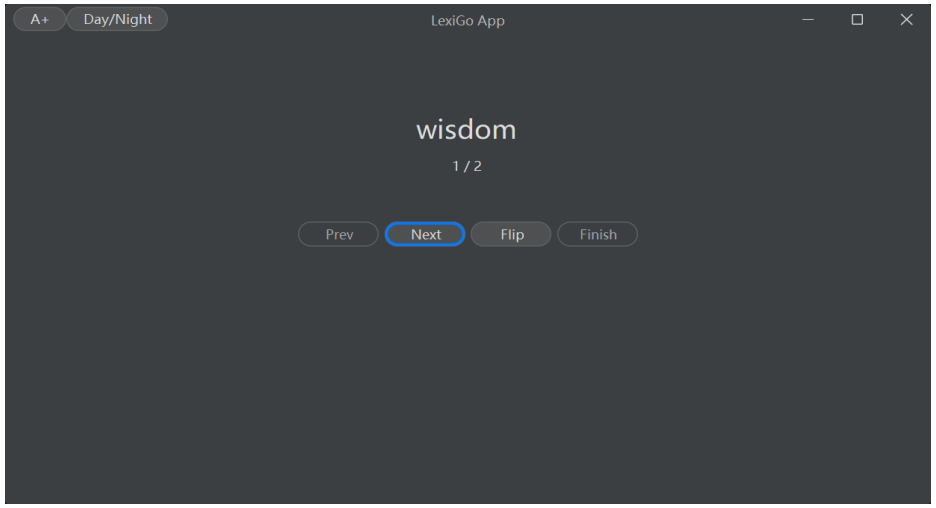
```
        /* 4. Build Card objects one by one (no stream API) */
        final List<Card> cards = new ArrayList<>();
        for (UUID wordId : wordIds) {
            cards.add(buildCard(wordId, input.getUsername()));
        }

        /* 5. Create deck via factory and persist */
        // get factory from dto
        final WordDeckFactory wordDeckFactory = factoryD  getWordDeckFactory();

        // get dao(dai)
        final UserCheckInDeckAccessInterface deckAccess;
        deckAccess = daoDto.getUserDeckAccessObject();

        // factory logic && DAO logic
        final WordDeck deck = wordDeckFactory.create(cards);
        deckAccess.save(deck);

        /* 6. Inform presenter of success */
        presenter.prepareSuccessView(
                new StartCheckInOutputData(input.getLength(), useCaseF  d: false, input.getUsername())
        );
    }
}
else {
    presenter.prepareFailView( errorMessage: "You should input valid positive  nterger");
}
}
```

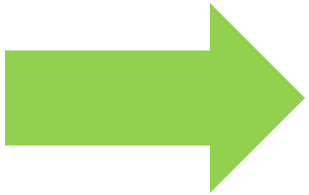
```
final String translation = translator.getTranslation(text, targetLang);

final String example = detailGenerator.getWordExample(text);
```

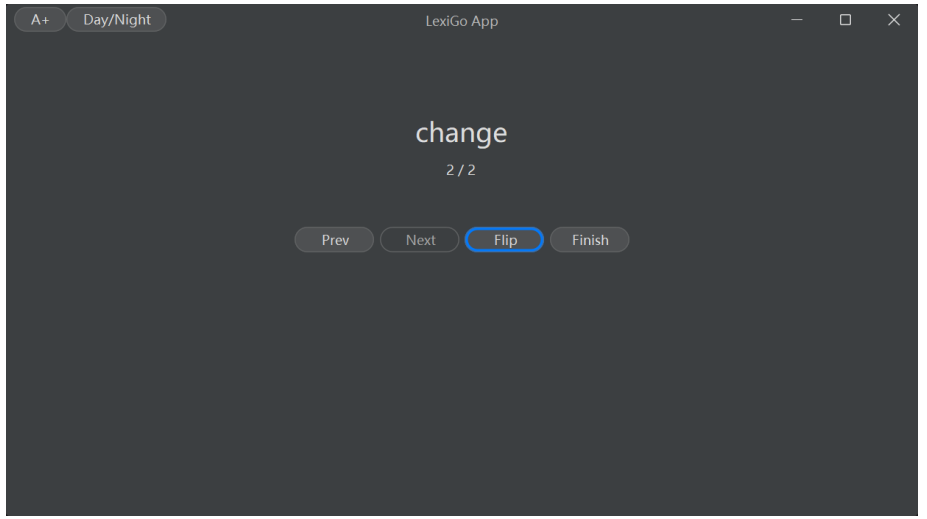

Framework → IA → usecase → entity
Dto/state
Flow of control
DAOs
EIs
Factories
ProfileFactory
PersonalProfileFactory
PersonalProfile
Language
DEFAULT_LANGUAGE
JsonUserProfileDAO
WordBook
CommonWordBook
WordBookDataAccessObject
WordDataAccessObject
InMemoryDeckDataAccessObejct
UserProfileDataAccessInterface
WordBookAccessInterface
WordDataAccessInterface
UserCheckInDeckAccessInterface
DaiDto
WordDeck
CommonWordDeck
CommonWordDeckFactory
WordDeckFactory
FactoryDto
Card
CommonCard
CommonCardFactory
CardFactory
StartCheckInState
StartCheckInViewModel
StartCheckInView
StartCheckInPresenter
StartCheckInController
StartCheckInInputData
StartCheckInOutputBoundary
StartCheckInInputBoundary
StartCheckInOutputData
StartCheckInInteractor
FormatDetector
LearnDeckGenerator
WordDetailAPI
WordTranslationAPI
LearnWordsGenerator
FreeDictionaryApiAdapter
DeepLAPIAdapter
«create»
language
viewModel
startCheckInViewModel
controller
presenter
generator
detailGenerator
translator
formatter
wordDeckFactory
commonCardFactory
deck

WordPlay II & III (Case name: Study Session & Word detail)

Haozhe Huo (study session case)
&
Jingcheng Liang (word detail case)







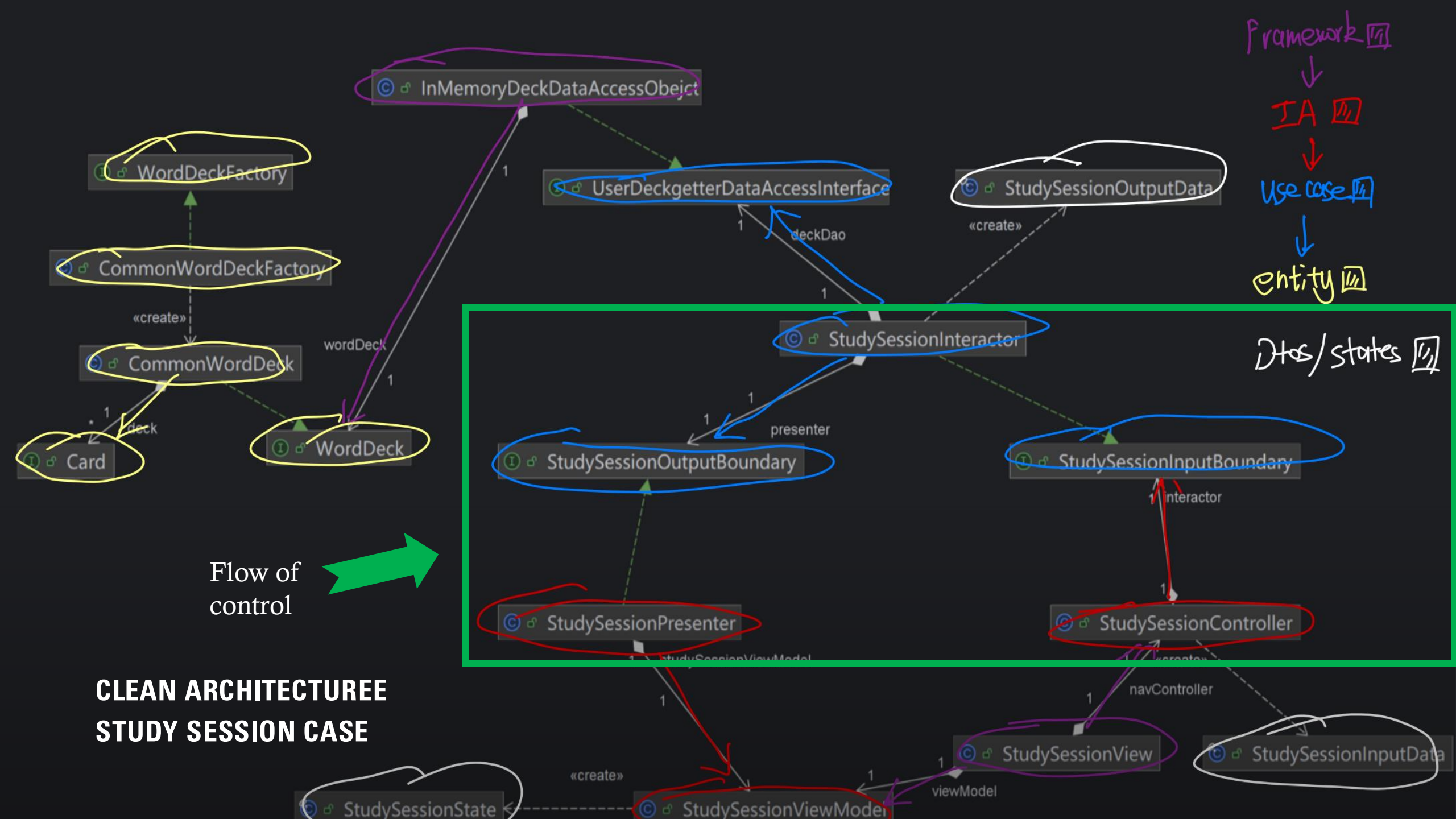
INTERACTOR CODE BRIEF WALKTHROUGH

```java
public void handleNextRequest(StudySessionInputData inputData) {

    // pagenum && totalpage from input
    final int currentPage = Integer.parseInt(inputData.getPagenumber());
    final int totalPages = Integer.parseInt(inputData.getTotalpage());

    // curr_page = curr_index + 1 = next_index
    final int nextIndex = currentPage;

    // get word using dao(using index)
    final String text = deckDao.getText(nextIndex);

    // if page is first / last As it's simple and do not need extension, so put it here
    final boolean reachLast = nextIndex == totalPages - 1;
    final boolean reachFirst = nextIndex <= 0;

    // output
    final StudySessionOutputData out = new StudySessionOutputData(
            String.valueOf( i: currentPage + 1),
            text,
            reachLast,
            reachFirst,
            String.valueOf(totalPages)
    );
    // when first page, prev button is unreachable, at last page , next button is unreachable
    // so only success is needed
    presenter.prepareSuccessView(out);
}
```

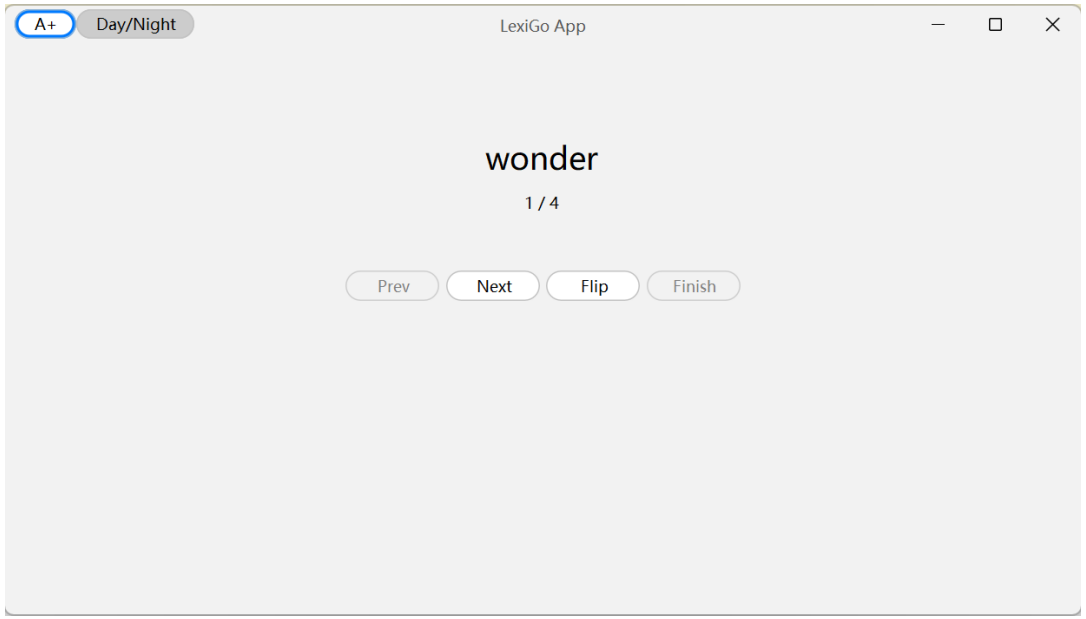
Dto data

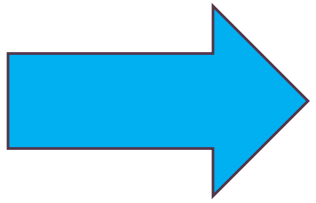
DAO

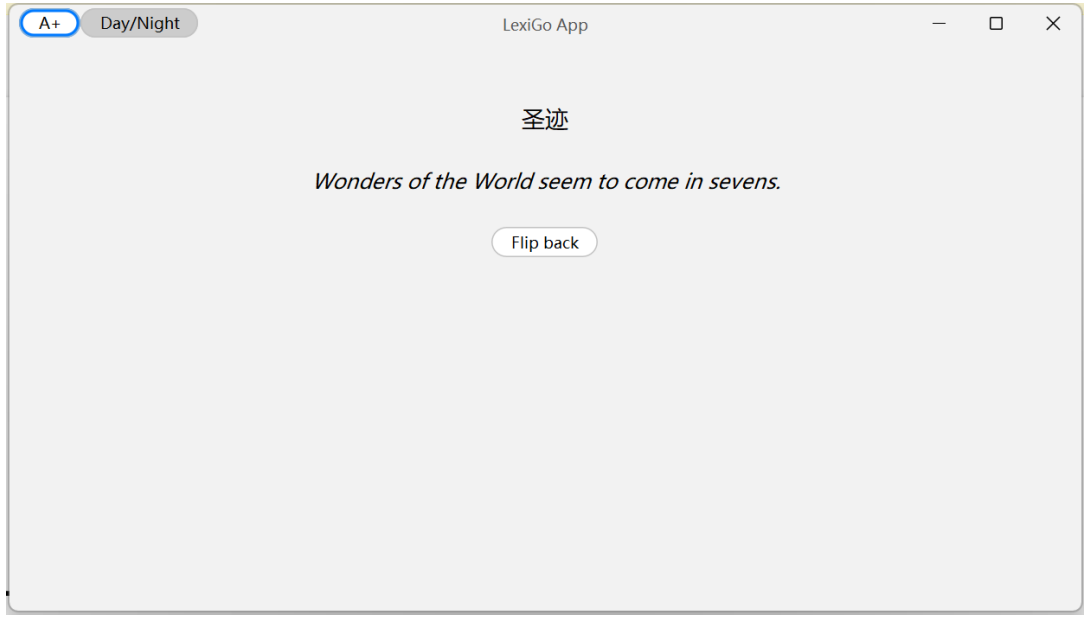
Build output model and present



CLEAN ARCHITECTUREE
STUDY SESSION CASE







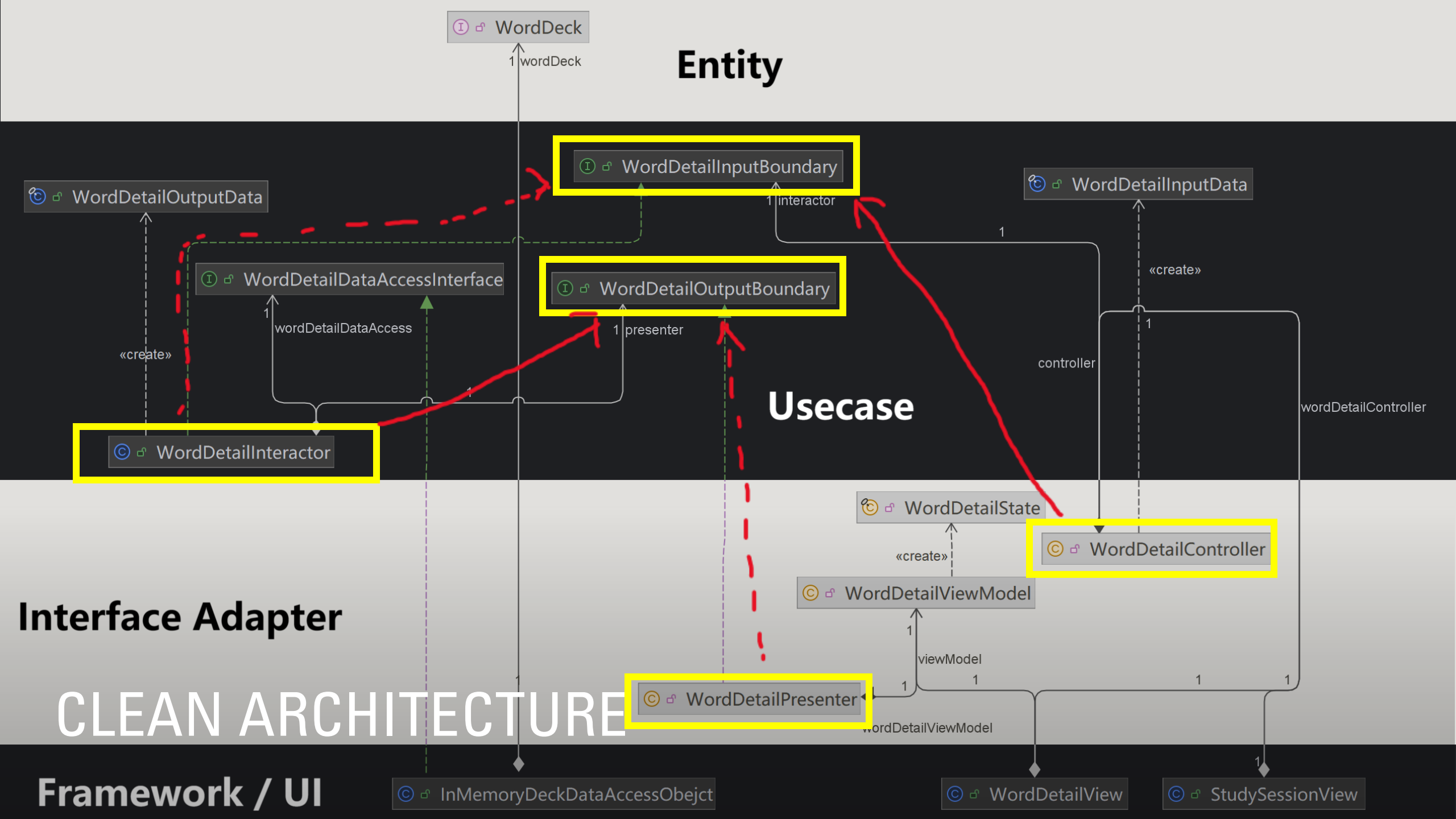
WORDDETAIL

Jingcheng Liang

TAP TO FLIP → BACK: EXAMPLE SENTENCE, TRANSLATION.

```java
@Override  Jingcheng Liang
public void execute(WordDetailInputData inputData) {
    final String translation = wordDetailDataAccess.getTranslation(
            input: Integer.parseInt(inputData.getCurrpage()) - 1);
    final String example = wordDetailDataAccess.getExample(
            input: Integer.parseInt(inputData.getCurrpage()) - 1);

    final WordDetailOutputData wordDetailOutputData =
            new WordDetailOutputData(translation, example);
    presenter.prepareSuccessView(wordDetailOutputData);
}
```

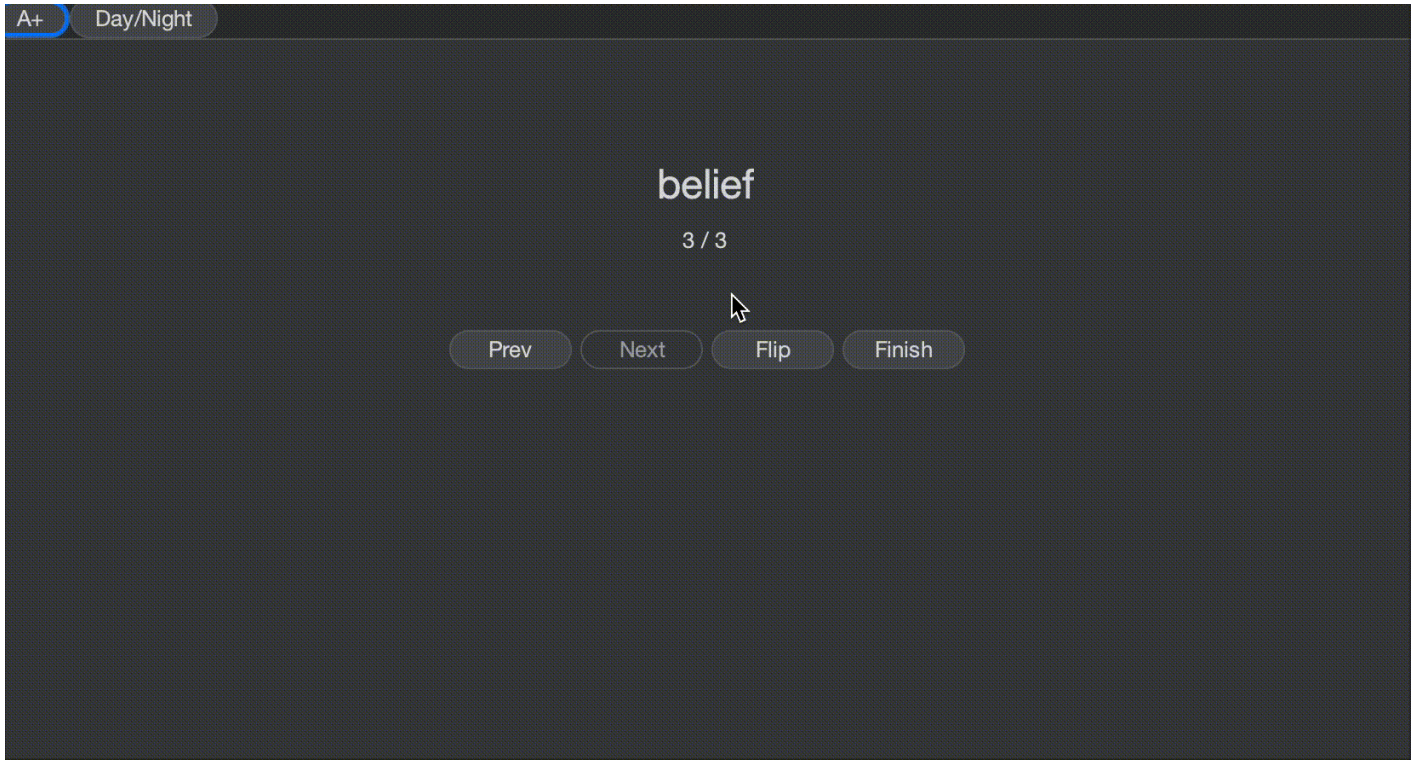
INTERACTOR CODE WALK THROUGH


WordDeck
1 wordDeck
Entity
WordDetailInputBoundary
WordDetailInputData
WordDetailOutputData
1 interactor
1
WordDetailDataAccessInterface
WordDetailOutputBoundary
«create»
1 wordDetailDataAccess
1 presenter
1
controller
wordDetailController
Usecase
WordDetailInteractor
WordDetailState
«create»
WordDetailController
WordDetailViewModel
Interface Adapter
1
viewModel
1
1
1
1
CLEAN ARCHITECTURE
WordDetailPresenter
wordDetailViewModel
Framework / UI
InMemoryDeckDataAccessObejct
WordDetailView
StudySessionView

FINISH CHECK-IN

Yunzhao Li

- *In the Learner finishing session,*
- *I want to complete a session & finalize check-in,*
- *so that I can log progress and update streak.*



FINISH CHECK-IN INTERACTOR CODE I

```java
package use_case.finish_checkin;

import ...

public class FinishCheckInInteractor implements FinishCheckInInputBoundary {  4 usages  yunzhaol +2

    private final FinishCheckInOutputBoundary presenter;  2 usages
    private final UserSaveRecordDataAccessInterface recordSaver;  2 usages
    private final UserDeckGetTextDataAccessInterface textsdataGetter;  2 usages
    private final LearnRecordFactory learnRecordFactory;  2 usages
    private final TimeGetter timegenerator;  2 usages

    public FinishCheckInInteractor(FinishCheckInOutputBoundary presenter,  2 usages  yunzhaol
                                   UserSaveRecordDataAccessInterface recordSaver,
                                   UserDeckGetTextDataAccessInterface textsdataGetter,
                                   LearnRecordFactory learnRecordFactory,
                                   TimeGetter timegenerator) {
        this.presenter = presenter;
        this.recordSaver = recordSaver;
        this.textsdataGetter = textsdataGetter;
        this.learnRecordFactory = learnRecordFactory;
        this.timegenerator = timegenerator;
    }
```

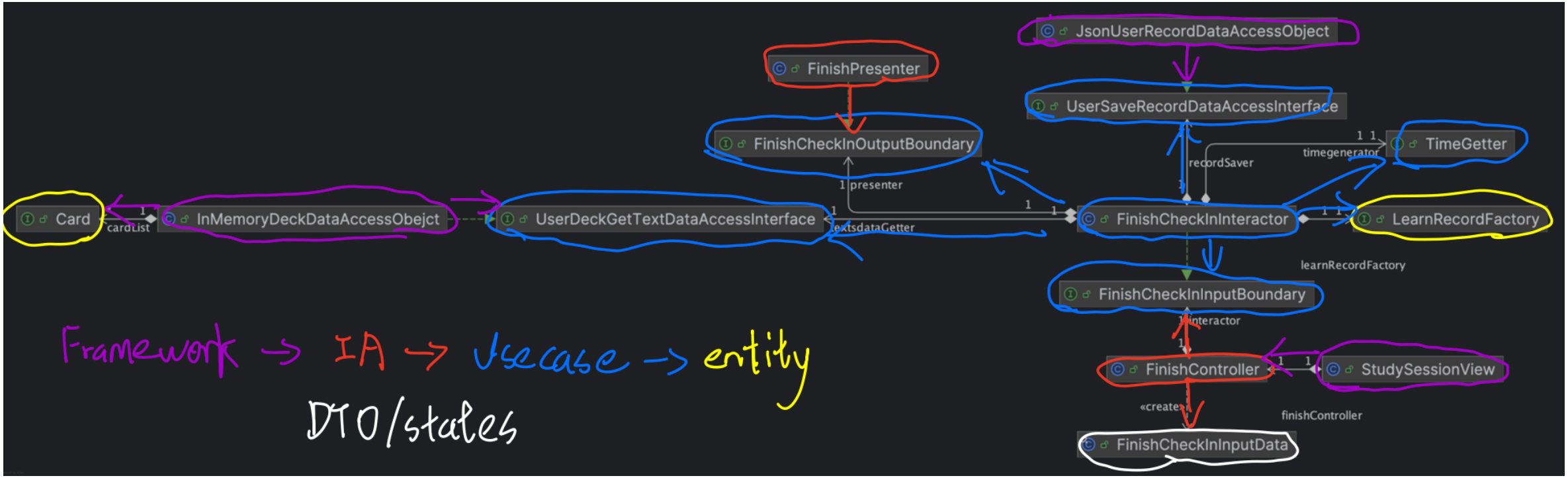
FINISH CHECK-IN INTERACTOR CODE II

```java
public void execute(FinishCheckInInputData inputData) {
    final List<UUID> uuidList = new ArrayList<>();
    // get all word Ids that previous session learnt
    for (Card eachCard : textsdataGetter.getWordDeck()) {
        uuidList.add(eachCard.getWordId());
    }
    // use factory to construct record entity
    final LearnRecord learnRecord = learnRecordFactory.create(inputData.getUsername()
            timegenerator.generate(), uuidList);
    // use learn record dao to save data
    recordSaver.save(learnRecord);
    // present
    presenter.prepareSuccessView();
}
```

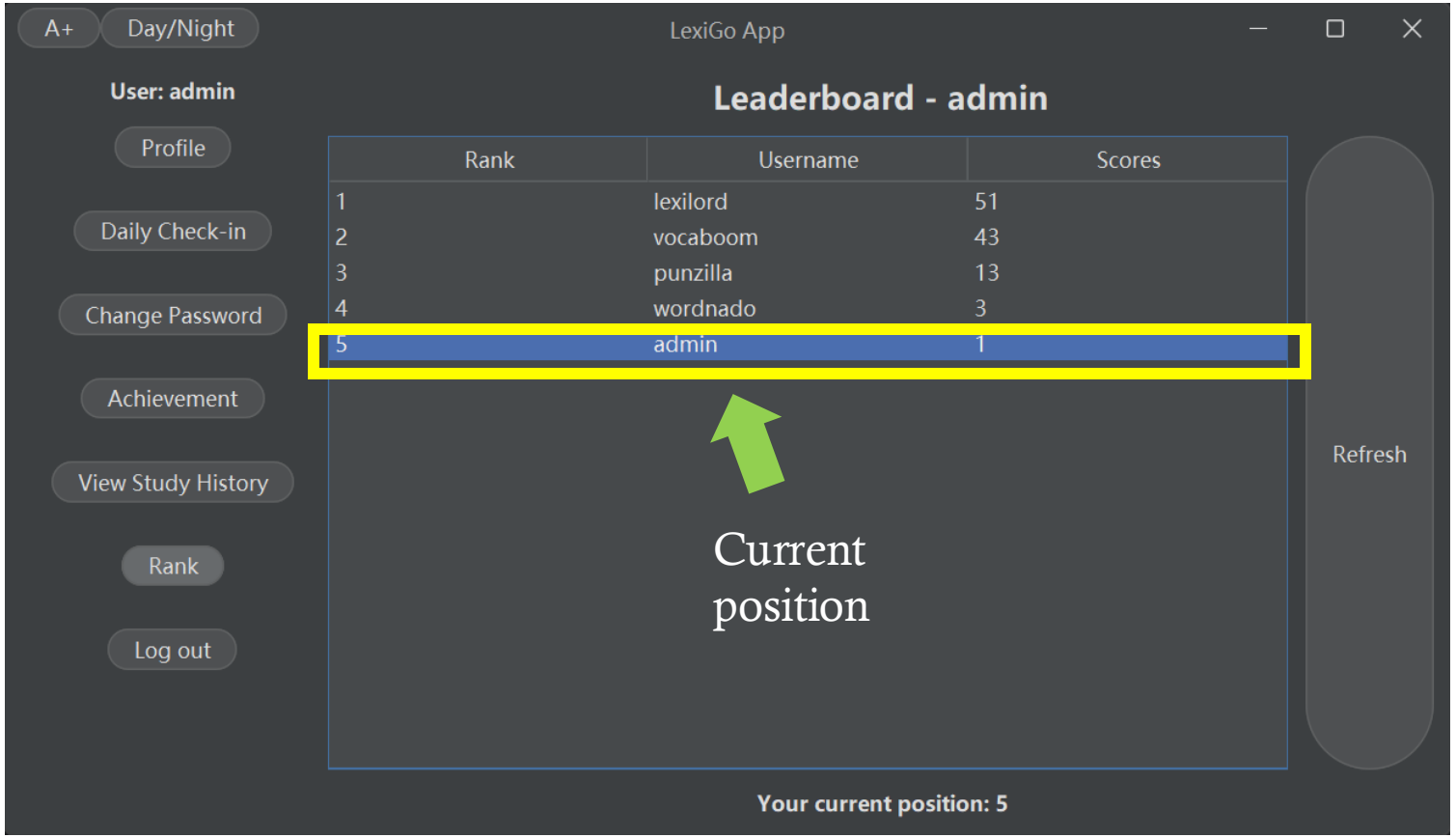
DAO

Factory creates entity

Presenter



CLEAN ARCHITECTURE
FINISH CHECK-IN CASE



RANK

Jingcheng Liang

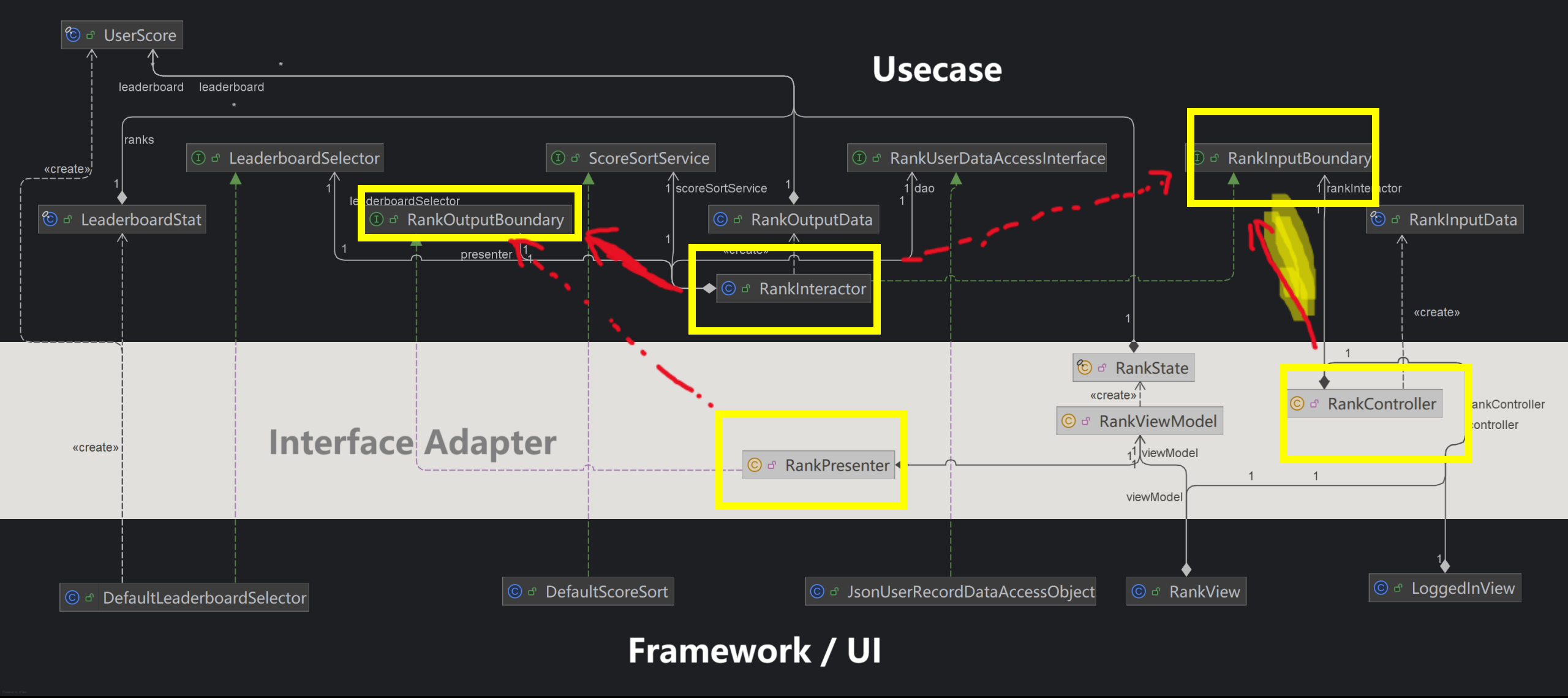
SEE HOW YOU COMPARE WITH FRIENDS OR THE GLOBAL COMMUNITY ALL-TIME LEADERBOARDS.

```
@Override  Jingcheng Liang *
public void execute(RankInputData in) {
    final List<Map.Entry<String, Integer>> users =
            scoreSortService.getSortedRank(dao.getAllUsers());

    final LeaderboardStat leaderboardStat = leaderboardSelector
            .getSortedRank(users, in.getUsername(), limit);

    presenter.prepareSuccessView(
            new RankOutputData(in.getUsername(),
                    leaderboardStat.getRanks(), leaderboardStat.getPosition()));
}
```

External interface 1: sort all users on app

External interface 2: extract rank data

INTERACTOR CODE WALK THROUGH



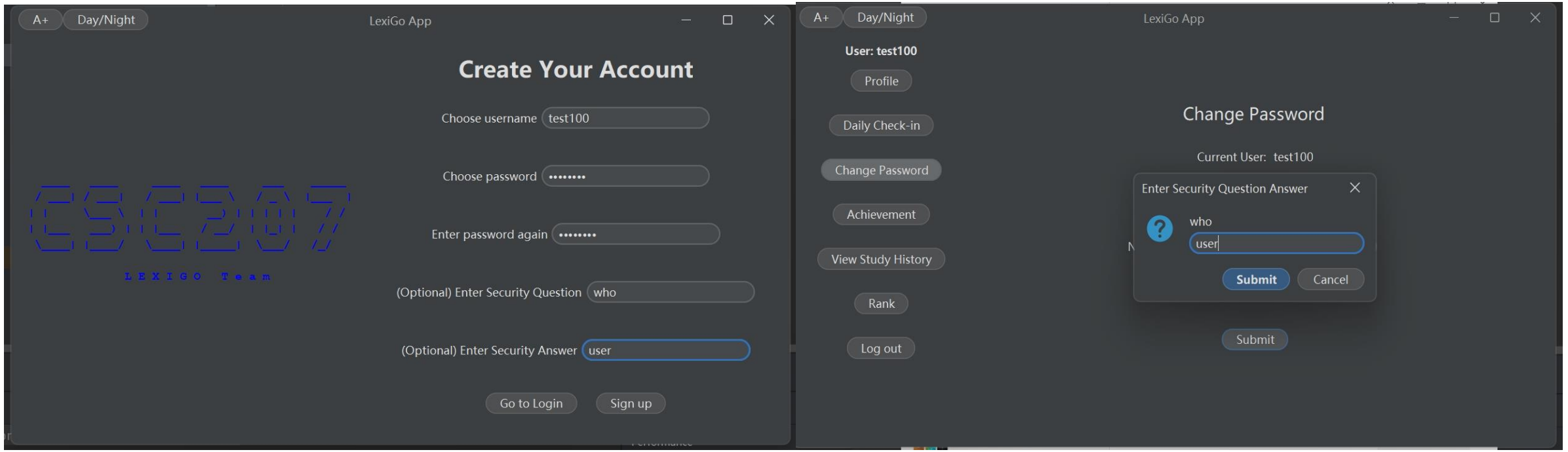
CLEAN ARCHITECTURE

CHANGE PASSWORD

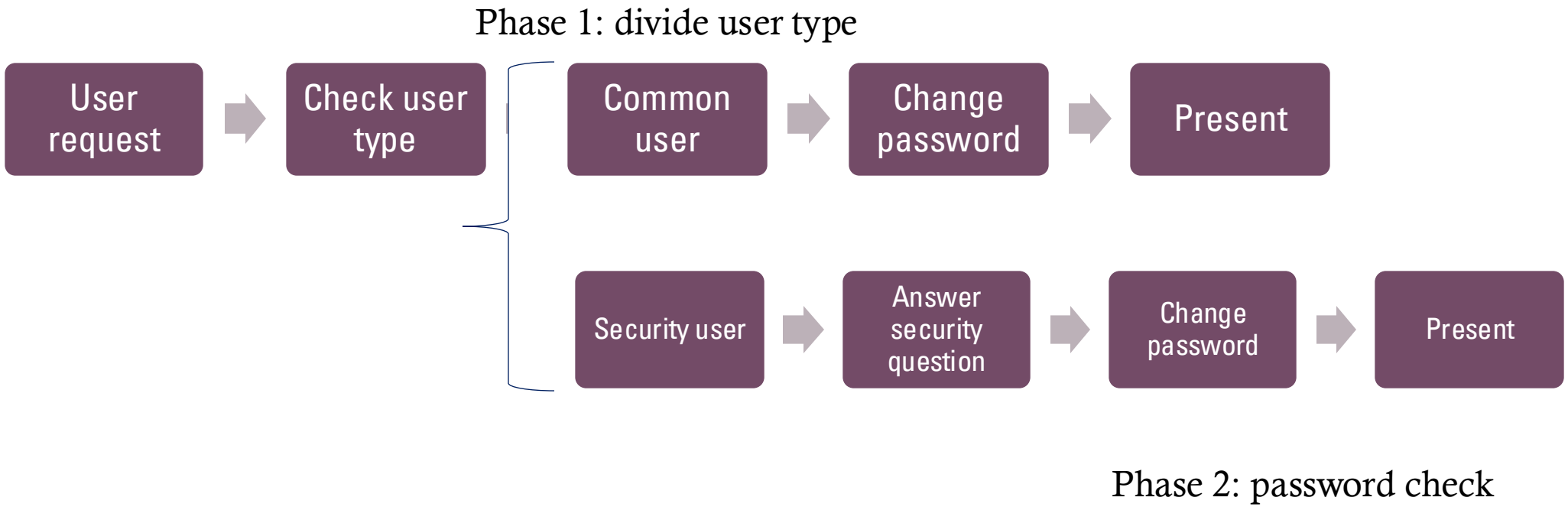
JAKOB KE

- *User story: As a security-conscious user,*
- *I want to change my password with identity verification,*
- *so that I can maintain account security after suspected breaches.*

For Security User



WORKFLOW


Phase 1: divide user type
User request
Check user type
Common user
Change password
Present
Security user
Answer security question
Change password
Present
Phase 2: password check

Security case: Answer question

Common case

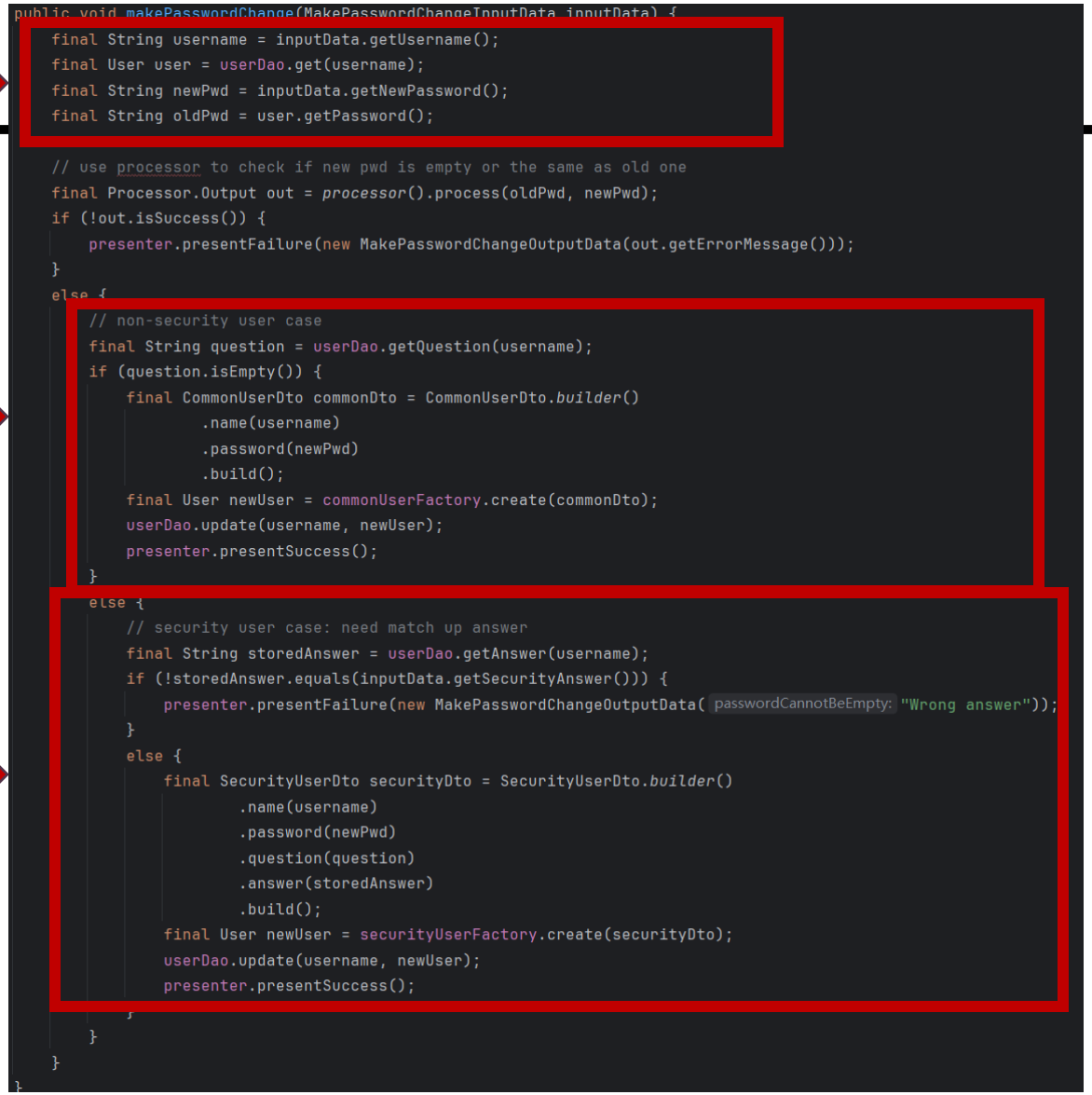
```java
@Override
public void execute(ChangePasswordInputData inputData) {

    boolean verification = false;
    final String type = userdatagetter.getType(inputData.getUsername());

    switch (type) {

        case "SECURITY":
            verification = true;
            final String securityQuestion = userdatagetter.getSecurityQuestion(inputData.getUsername());
            userPresenter.preparePage(new ChangePasswordOutputData(inputData.getUsername(), verification,
                    securityQuestion));
            break;
        default:
            verification = false;
            userPresenter.preparePage(new ChangePasswordOutputData(inputData.getUsername(), verification,
                    securityQuestion: null));
            break;
    }
}
```

INTERACTOR WALKTHROUGH I

DAO Model

Non-security question case

Security user case

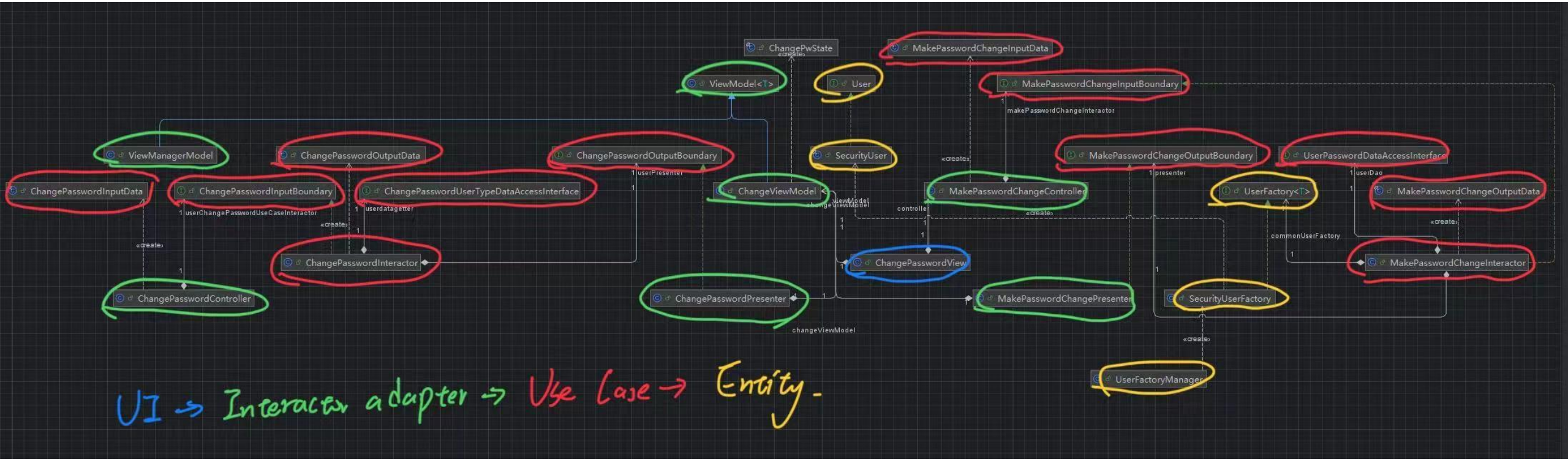


```
public void makePasswordChange(MakePasswordChangeInputData inputData) {
    final String username = inputData.getUsername();
    final User user = userDao.get(username);
    final String newPwd = inputData.getNewPassword();
    final String oldPwd = user.getPassword();

    // use processor to check if new pwd is empty or the same as old one
    final Processor.Output out = processor().process(oldPwd, newPwd);
    if (!out.isSuccess()) {
        presenter.presentFailure(new MakePasswordChangeOutputData(out.getErrorMessage()));
    }
    else {
        // non-security user case
        final String question = userDao.getQuestion(username);
        if (question.isEmpty()) {
            final CommonUserDto commonDto = CommonUserDto.builder()
                    .name(username)
                    .password(newPwd)
                    .build();
            final User newUser = commonUserFactory.create(commonDto);
            userDao.update(username, newUser);
            presenter.presentSuccess();
        }
        else {
            // security user case: need match up answer
            final String storedAnswer = userDao.getAnswer(username);
            if (!storedAnswer.equals(inputData.getSecurityAnswer())) {
                presenter.presentFailure(new MakePasswordChangeOutputData( passwordCannotBeEmpty: "Wrong answer"));
            }
            else {
                final SecurityUserDto securityDto = SecurityUserDto.builder()
                        .name(username)
                        .password(newPwd)
                        .question(question)
                        .answer(storedAnswer)
                        .build();
                final User newUser = securityUserFactory.create(securityDto);
                userDao.update(username, newUser);
                presenter.presentSuccess();
            }
        }
    }
}
```

INTERACTOR WALKTHROUGH II

CLEAN ARCHITECTURE

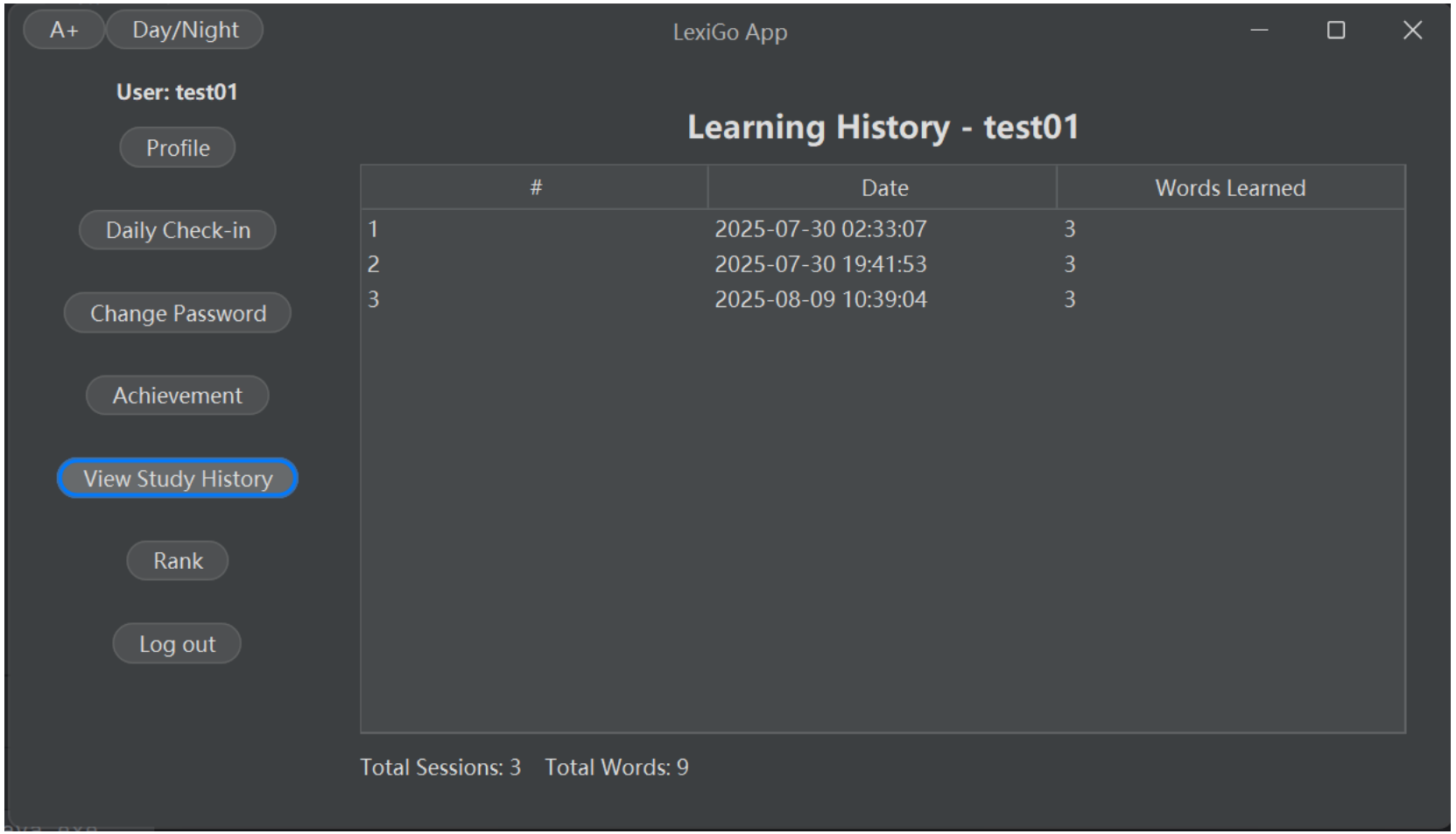
- InputBoundary → Interactor processes business logic via DAO → calls OutputBoundary → Presenter formats results.



VIEW HISTORY

JACOB KE

- *As a returning learner, I want to view my past study sessions, so that I can track my learning progress and adjust my study plan.*

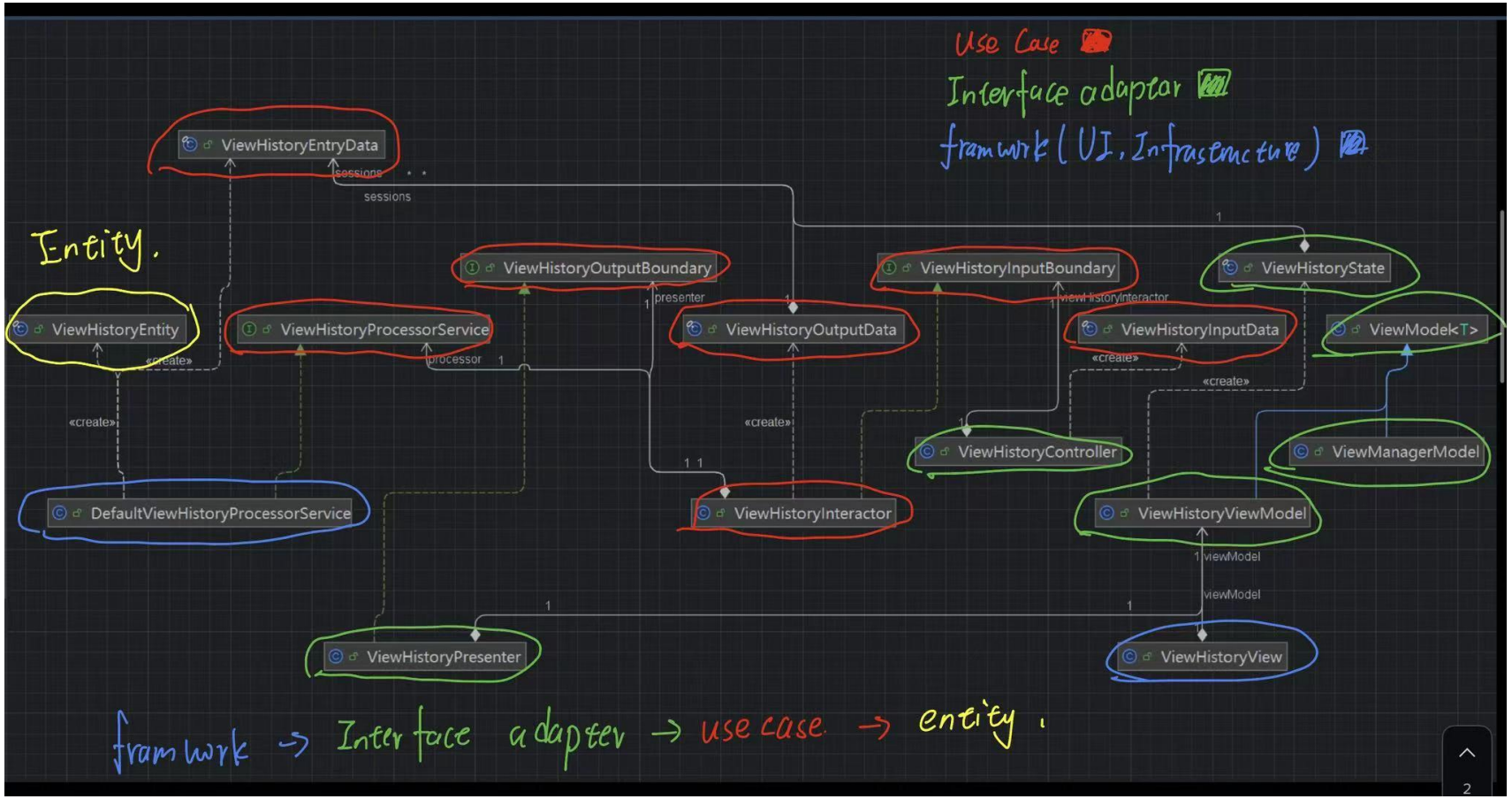


DAO

```
public void execute(ViewHistoryInputData inputData) {
    final String username = inputData.getUsername();

    // Retrieve records from data layer
    final List<LearnRecord> records = dataAccess.get(username);

    // Early return if no records are found
    if (records.isEmpty()) {
        presenter.prepareFailView( errorMessage: "No learning records found for user: " + username);
    }
    else {
        // Delegate processing of raw records
        final List<ViewHistoryEntryData> processedSessions = processor.processRecords(records);

        // Compute aggregates
        final int totalSessions = processedSessions.size();
        final int totalWords = calculateTotalWords(processedSessions);

        // Build and present the output data
        final ViewHistoryOutputData result = new ViewHistoryOutputData(
                username,
                processedSessions,
                totalSessions,
                totalWords
        );
        presenter.prepareSuccessView(result);
    }
}
```

INTERACTOR CODE WALK THROUGH

CLEAN ARCHITETURE



ACHIEVEMENT

Heyuan Zhou

As a motivated learner,

I want to unlock achievements based on my learning progress,

so that I can celebrate my milestones and stay engaged.

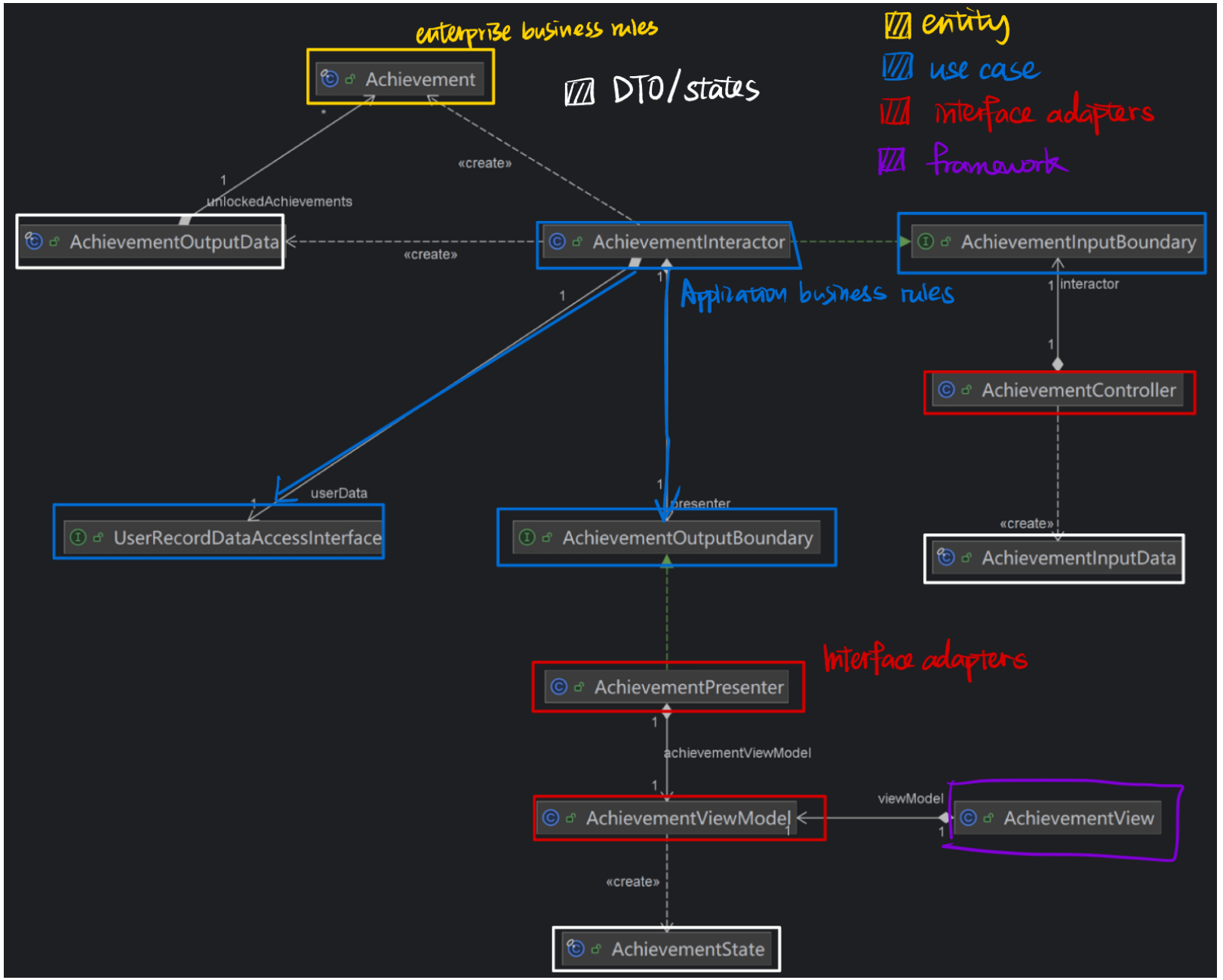
```java
@Override
public void evaluate(AchievementInputData inputData) {
    final String username = inputData.getUsername();
    // Fetch all learning session records for the user.
    final List<LearnRecord> wordsLearned = userData.get(username);
    // This list will store any newly unlocked achievements.
    final List<Achievement> newlyUnlocked = new ArrayList<>();
    // Analyze the data and determine which achievements to unlock.
    calculateAchievements(newlyUnlocked, wordsLearned);
    // Package the result into an output data object and pass it to the preser
    final AchievementOutputData response = new AchievementOutputData(newlyUnlo
    presenter.present(response);
}

/**
 * This helper method calculates which achievements the user should receive
 * @param newlyUnlocked a list to hold newly earned achievements.
 * @param wordsLearned  the user's full learning history.
 */
private void calculateAchievements(List<Achievement> newlyUnlocked,
                                   List<LearnRecord> wordsLearned) {
    // Number of learning sessions.
    final int totalLearnedTimes = wordsLearned.size();
    // Total number of words learned (sum of word IDs in all records).
    int wordsLearnedCount = 0;
    for (LearnRecord record : wordsLearned) {
        wordsLearnedCount += record.getLearnedWordIds().size();
    }
    // Check if the user qualifies for session-count achievements.
    checkTimeLearned(newlyUnlocked, totalLearnedTimes);
    // Check if the user qualifies for word-count achievements.
    checkWordLearned(newlyUnlocked, wordsLearnedCount);
}
```

INTERACTOR WALKTHROUGH

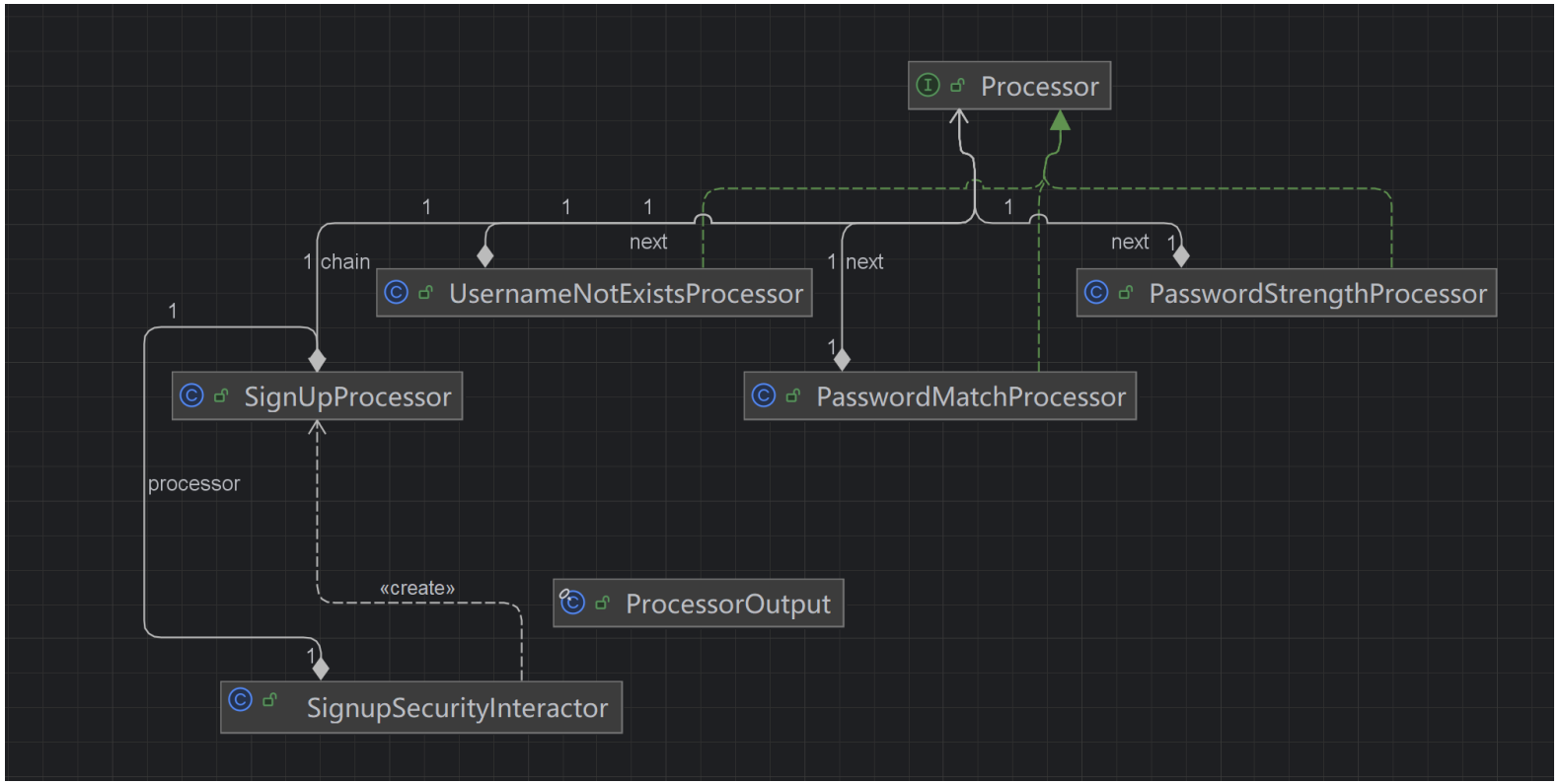
```java
/**
 * Adds word-based achievements to the list if thresholds are met.
 * @param newlyUnlocked unlocked
 * @param wordsLearnedCount count
 */
private void checkWordLearned(List<Achievement> newlyUnlocked, int wordsLearnedCount) {
    final Map<Integer, Achievement> wordAchievements = new HashMap<>();
    wordAchievements.put(1, new Achievement( id: "W1",  name: "First Word",
            description: "First Word I Learned!",  iconUnicode: "\uD83D\uDC4B"));
    wordAchievements.put(INT1, new Achievement( id: "W5",  name: "5 Words Learned",
            description: "5 Words Learned!",  iconUnicode: "\uD83C\uDF89"));
    wordAchievements.put(INT2, new Achievement( id: "W10",  name: "10 Words Learned",
            description: "10 Words Learned!",  iconUnicode: "\uD83D\uDD25"));
    wordAchievements.put(INT3, new Achievement( id: "W20",  name: "20 Words Learned",
            description: "20 Words Learned!",  iconUnicode: "\uD83D\uDC53"));

    // If the user's total words learned reaches the threshold, add the achievement.
    for (Map.Entry<Integer, Achievement> entry : wordAchievements.entrySet()) {
        if (wordsLearnedCount >= entry.getKey()) {
            newlyUnlocked.add(entry.getValue());
        }
    }
}
```

```java
/**
 * Adds session-based achievements to the list if thresholds are met.
 * @param newlyUnlocked unlocked
 * @param totalLearnedTimes count
 */
private void checkTimeLearned(List<Achievement> newlyUnlocked, int totalLearnedTimes) {
    final Map<Integer, Achievement> timeAchievements = new HashMap<>();
    timeAchievements.put(1, new Achievement( id: "A1",  name: "1 Time Learned",
            description: "You have learned 1 time!",  iconUnicode: "\uD83D\uDCDA"));
    timeAchievements.put(2, new Achievement( id: "A2",  name: "2 Times Learned",
            description: "You have learned 2 times!",  iconUnicode: "\uD83E\uDDE0"));
    timeAchievements.put(INT, new Achievement( id: "A3",  name: "3 Times Learned",
            description: "You have learned 3 times!",  iconUnicode: "\uD83C\uDFC6"));

    for (Map.Entry<Integer, Achievement> entry : timeAchievements.entrySet()) {
        if (totalLearnedTimes >= entry.getKey()) {
            newlyUnlocked.add(entry.getValue());
        }
    }
}
```

INTERACTOR WALKTHROUGH

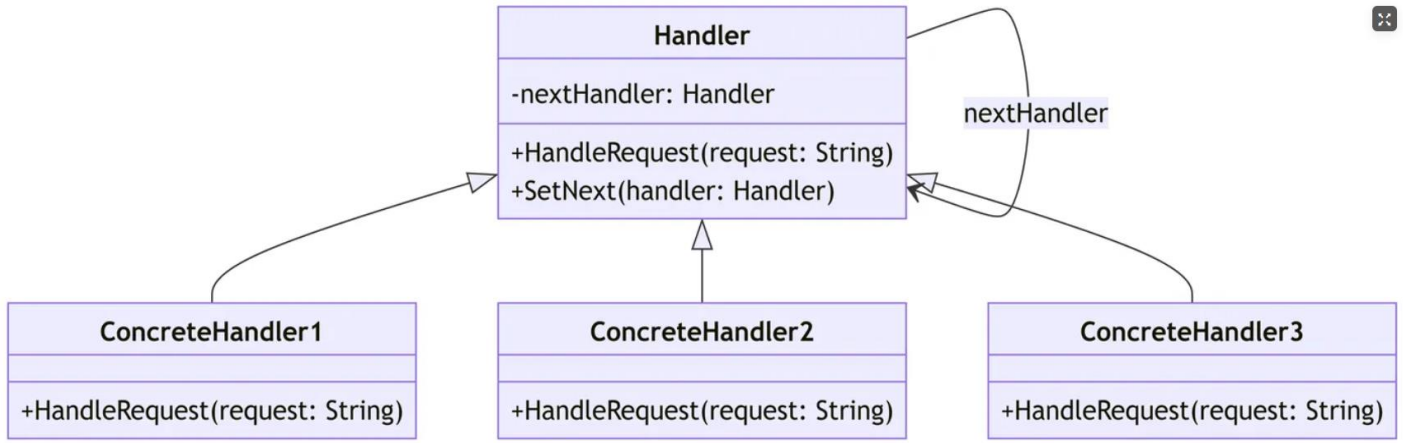


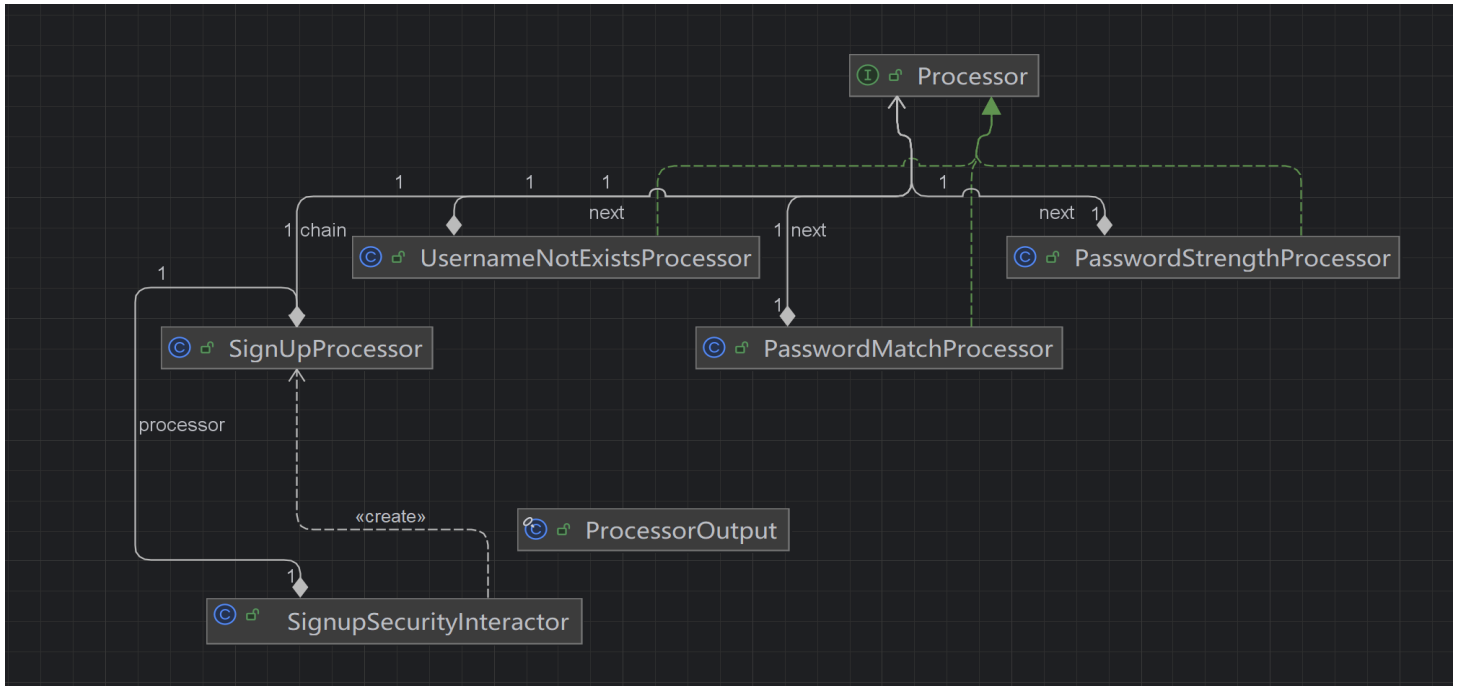
CA & FLOW OF CONTROL



DESIGN PATTERN INTRODUCTION

Responsibility Chain

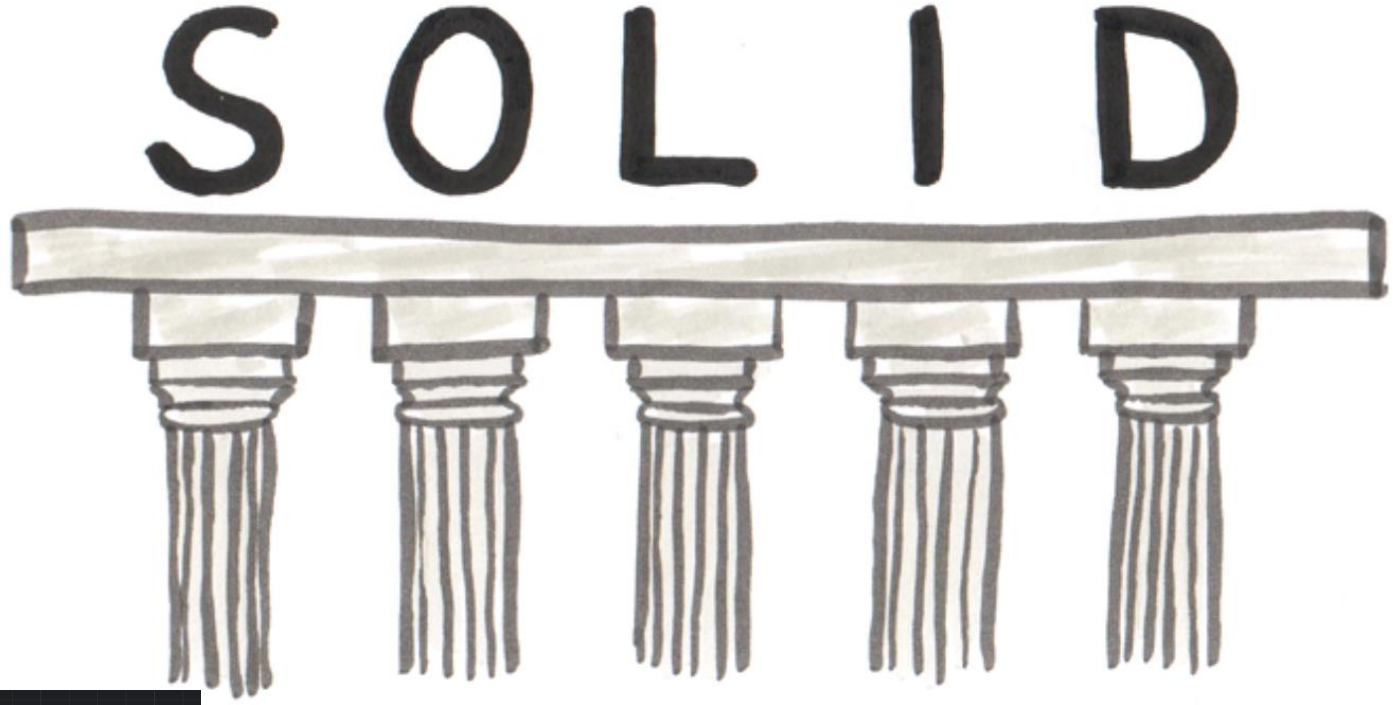




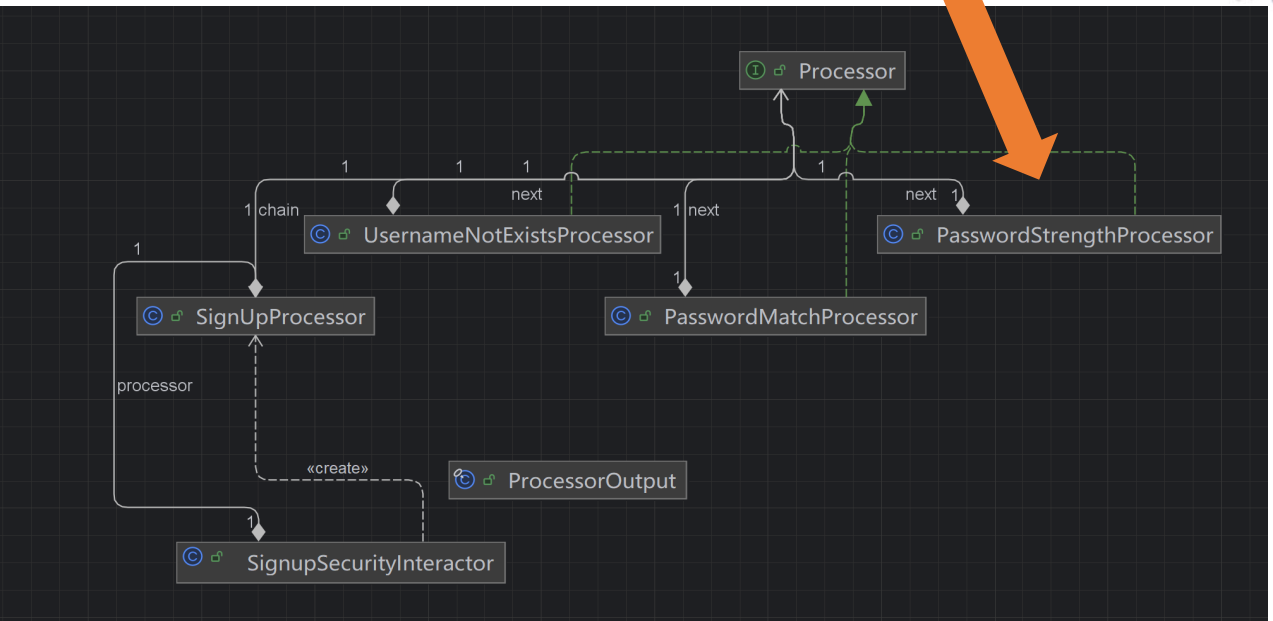
IDEAL & ACTUAL

The design is better for

SRP



Adjust password security level

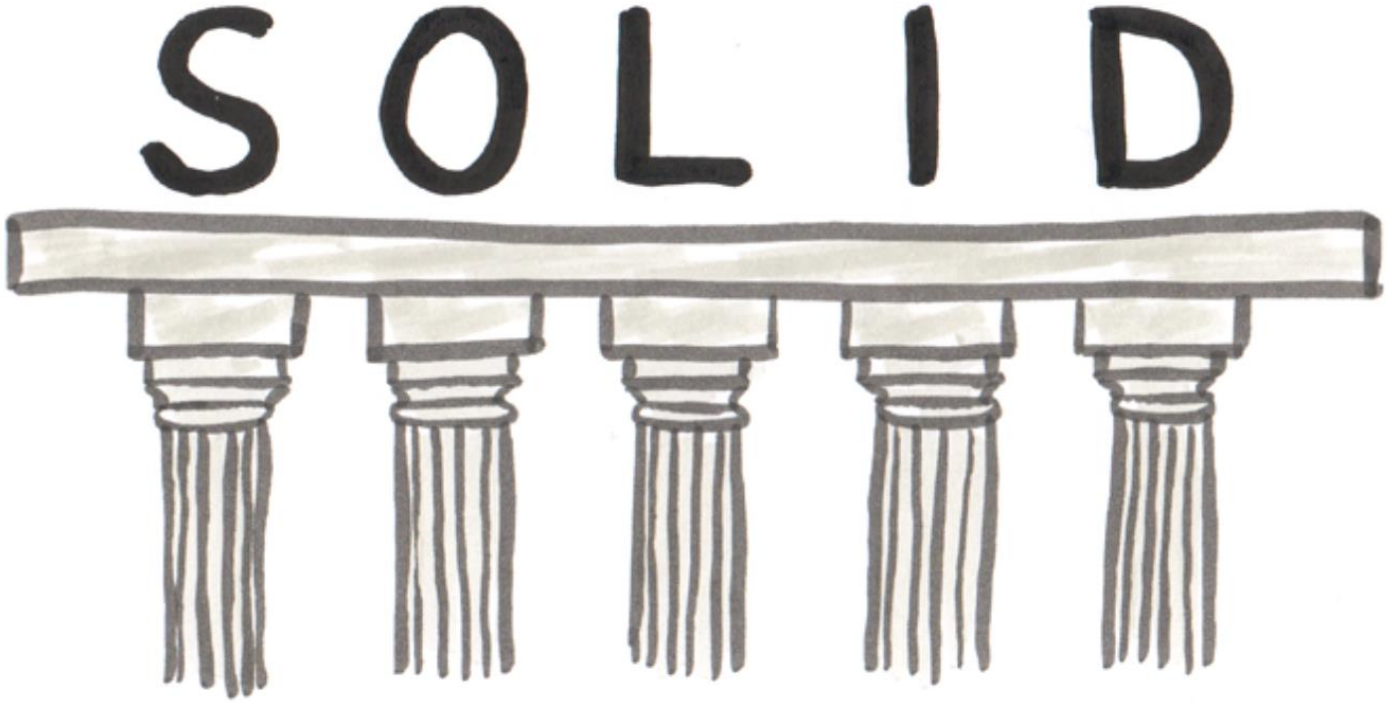


V S

The design is better for

OCP

—— Two extension examples

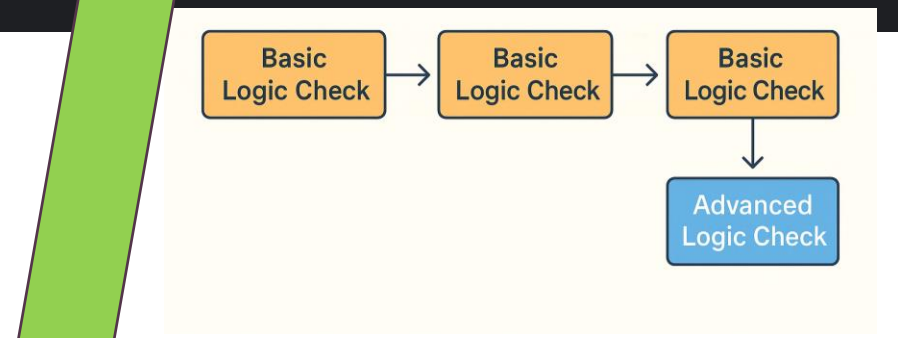


```
    // Build the chain: username uniqueness → password match → password strength
    this.chain = builder()
            //.addUsernameCannotHaveCurseWordCheck()
            .addUserExistsCheck(dao)
            .addPasswordCheck()
            .addPasswordStrengthCheck(passwordRule)
            .build();
}
```

CODING EXTENSION EXAMPLE1

- 1 Write a new processor and update builder
- 2, Add to the chain managed by signupProcessor

```
final ProcessorOutput output = processor.signUpProcessor(username, password, repeatPassword);

if (!output.isSuccess()) {
    // add security checks here, like a processor
    // e.g output = processorSecurity.process(securityQuestion, securityAnswer), the same design.
    userPresenter.prepareFailView(output.getErrorMessage());
}
else {
    final SecurityUserDto dto = SecurityUserDto.builder()
            .name(username)
            .password(password)
            .question(signupInputData.getSecurityQuestion
            .answer(signupInputData.getSecurityAnswer())
            .build();
```



1, Add a new featureProcessor

2, Create a new responsibility chain for new features

CODING EXTENSION EXAMPLE2

DIP—— VIEW HISTORY

Abstract dependencies

```
public class ViewHistoryInteractor implements ViewHistoryInputBoundary {  8 usages  Jingcheng Liang +1

    private final UserRecordDataAccessInterface dataAccess;  2 usages
    private final ViewHistoryOutputBoundary presenter;  3 usages
    private final ViewHistoryProcessorService processor;  2 usages

    /**
     * Constructor with injectable dependencies for testing/customization.
     *
     * @param dataAccess the data access interface for retrieving user records
     * @param presenter  the output boundary for presenting results
     * @param processor  the service for processing raw history records
     */
    public ViewHistoryInteractor(UserRecordDataAccessInterface dataAccess,  2 usages  Jacob Ke
                                 ViewHistoryOutputBoundary presenter,
                                 ViewHistoryProcessorService processor) {
        this.dataAccess = dataAccess;
        this.presenter = presenter;
        this.processor = processor;
    }
```

ISP

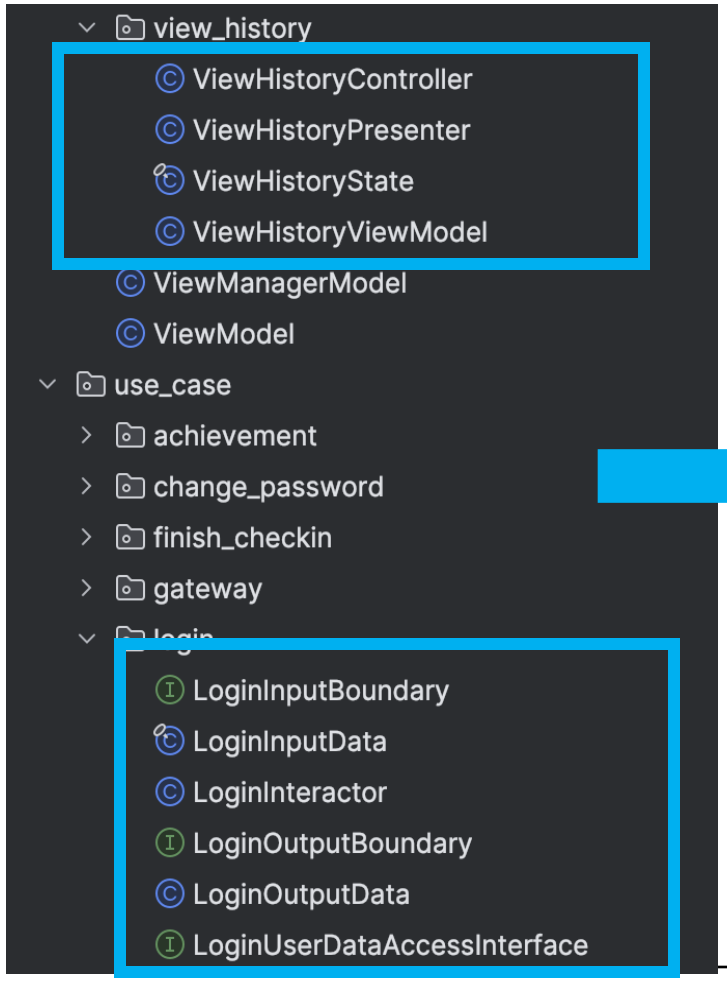
```
public class JsonUserRecordDataAccessObject  43 个用法
        implements UserRecordDataAccessInterface,
        UserSaveRecordDataAccessInterface,
        RankUserDataAccessInterface {
```

```
public interface UserRecordDataAccessInterface {  16 个用法  1 个实现  Jingcheng Liang
    /**
     * Fetches all {@link LearnRecord} instances associated with the given user.
     *
     * @param username the username whose history to retrieve; must not be {@code null
     * @return a {@link java.util.List} of {@link LearnRecord}; never {@code null}
     */
    List<LearnRecord> get(String username);  1 个实现  Jingcheng Liang
}
```

```
public interface UserSaveRecordDataAccessInterface {  4 个用法  1 个实现  Jingcheng Liang +1
    /**
     * Saves a completed learning session record.
     *
     * @param learnRecord the {@link LearnRecord} representing the session to persist
     *                    must not be {@code null}
     */
    void save(LearnRecord learnRecord);  1 个实现  yunzhaol
}
```

```
public interface RankUserDataAccessInterface {  4 个用法  1 个实现  Jingcheng Liang
    /**
     * Retrieves a map containing usernames and their corresponding to learn records.
     *
     * @return a map where each key is a username, and each value is a list of LearnRecord obje
     */
    Map<String, List<LearnRecord>> getAllUsers();  4 个用法  1 个实现  Jingcheng Liang
}
```

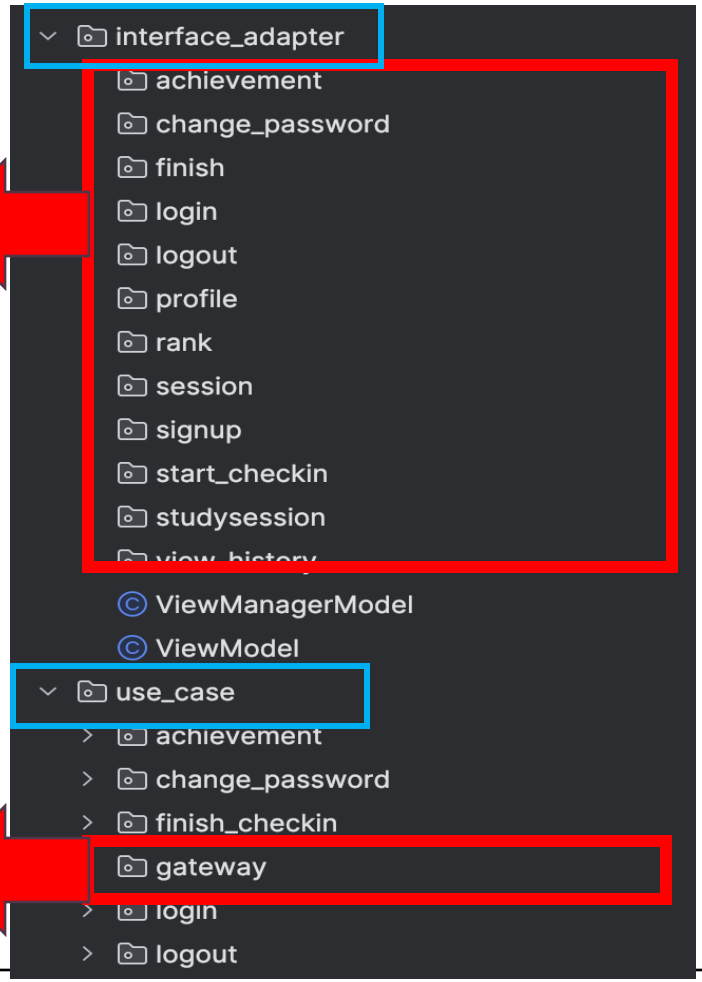
CODE ORGANIZATION

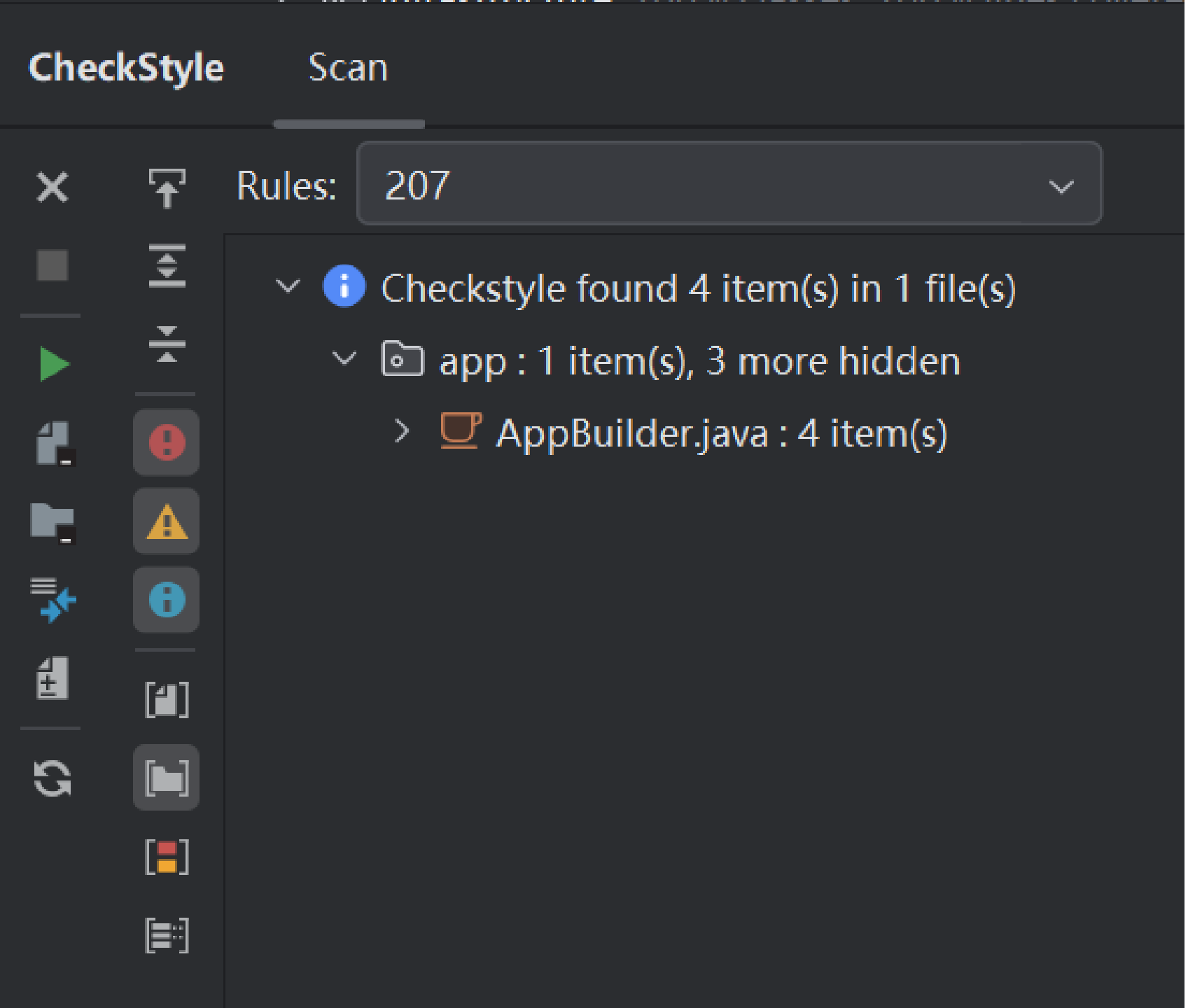


Naming follows CA convention

Files organized based on user story

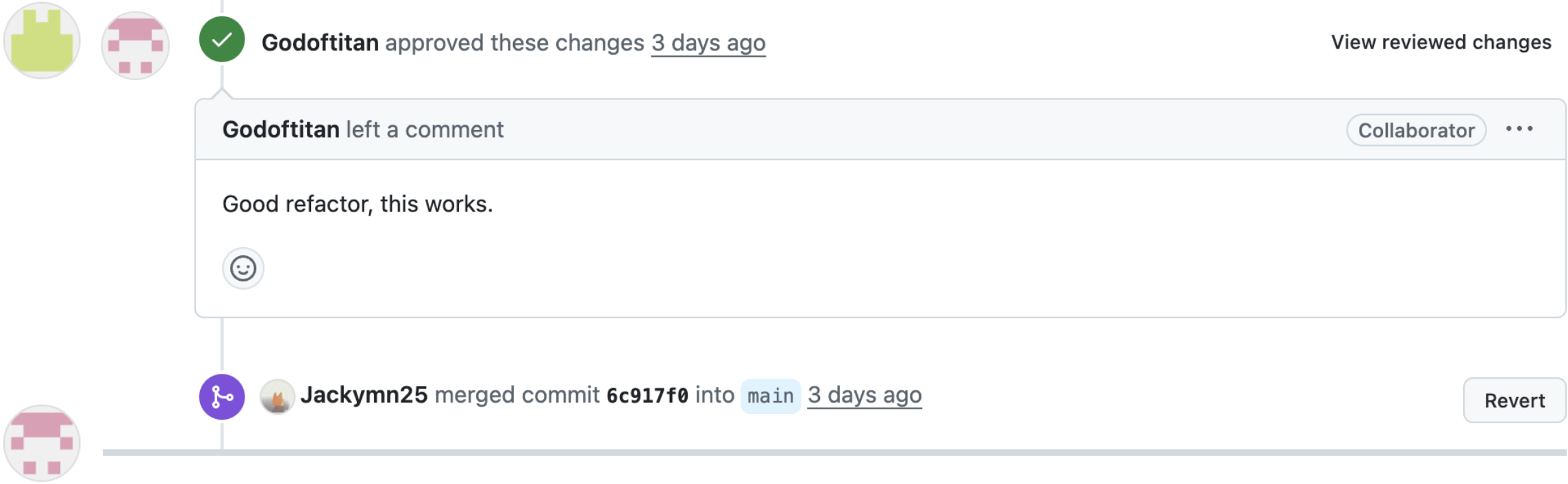
Shared DAI between user stories



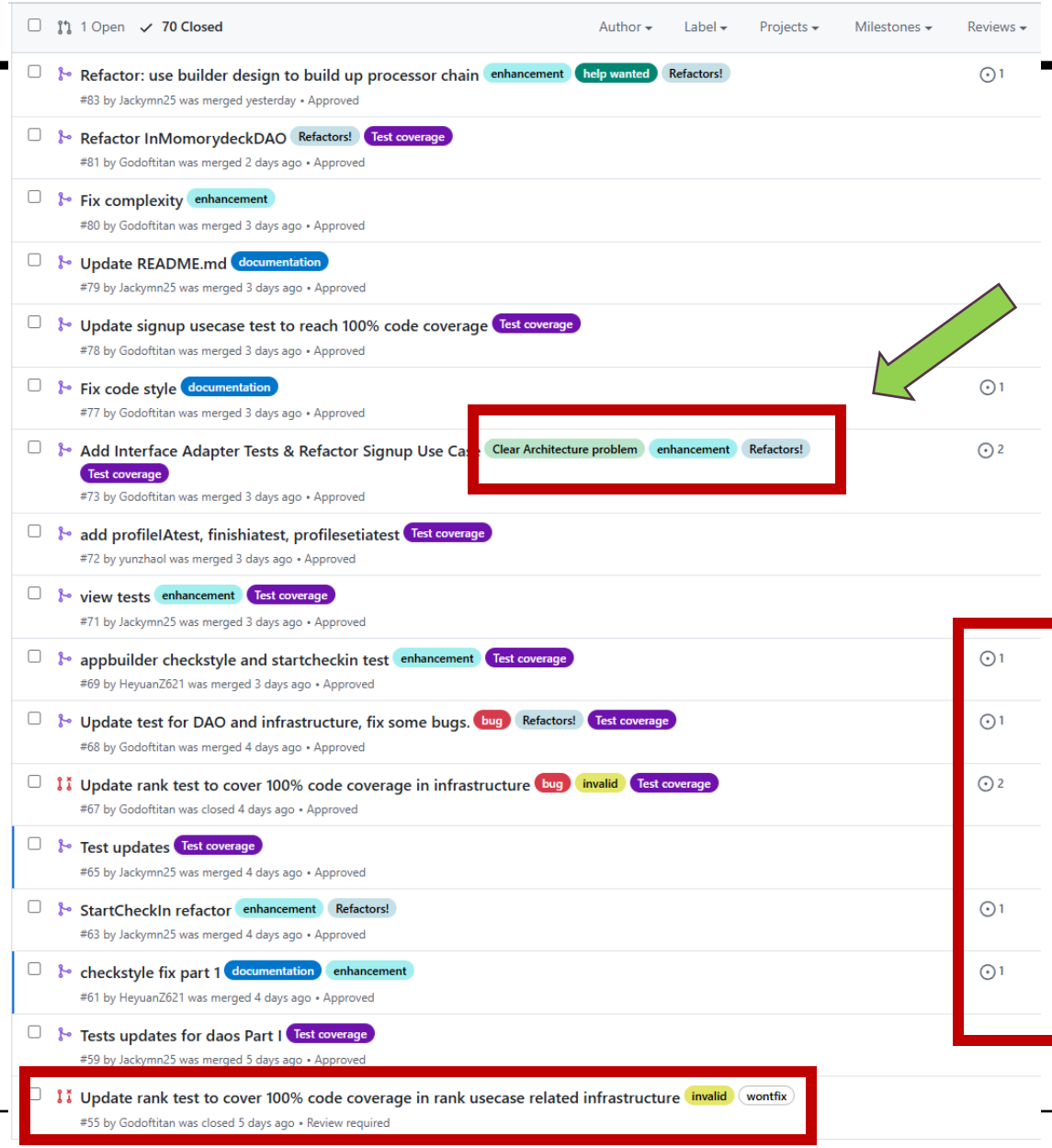


CODE QUALITY – Checkstyle

CODE QUALITY – Pull request



Pull requests examples

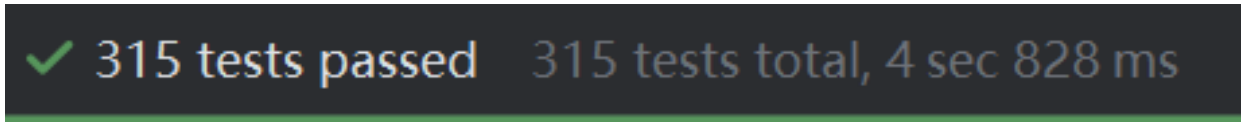


Labels

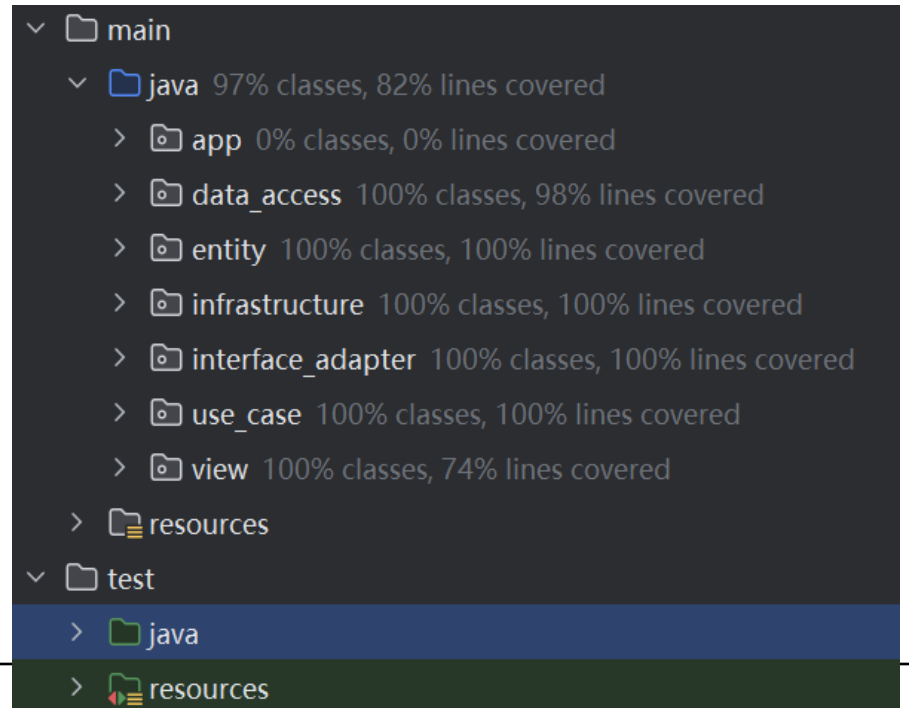
Closed issues

Buggy design rejected

TESTING



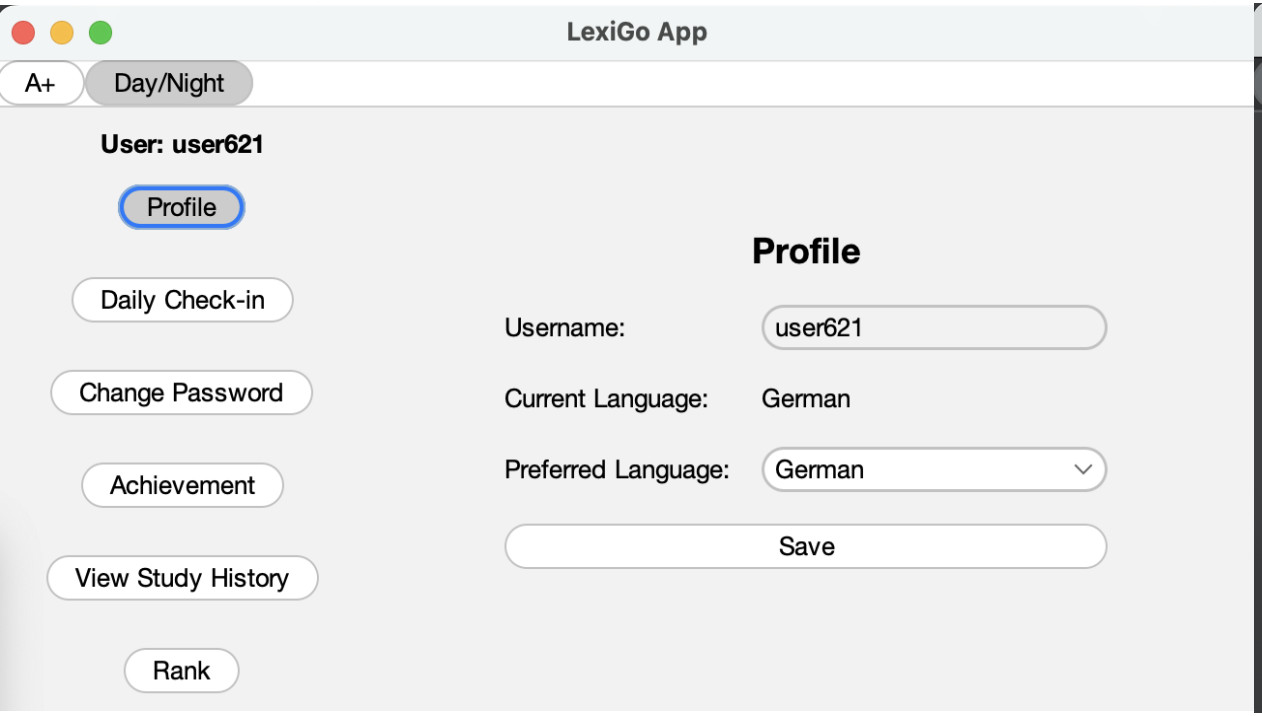
- 100% test coverage for entity, use case, interface adapter
- 82% coverage for overall code base

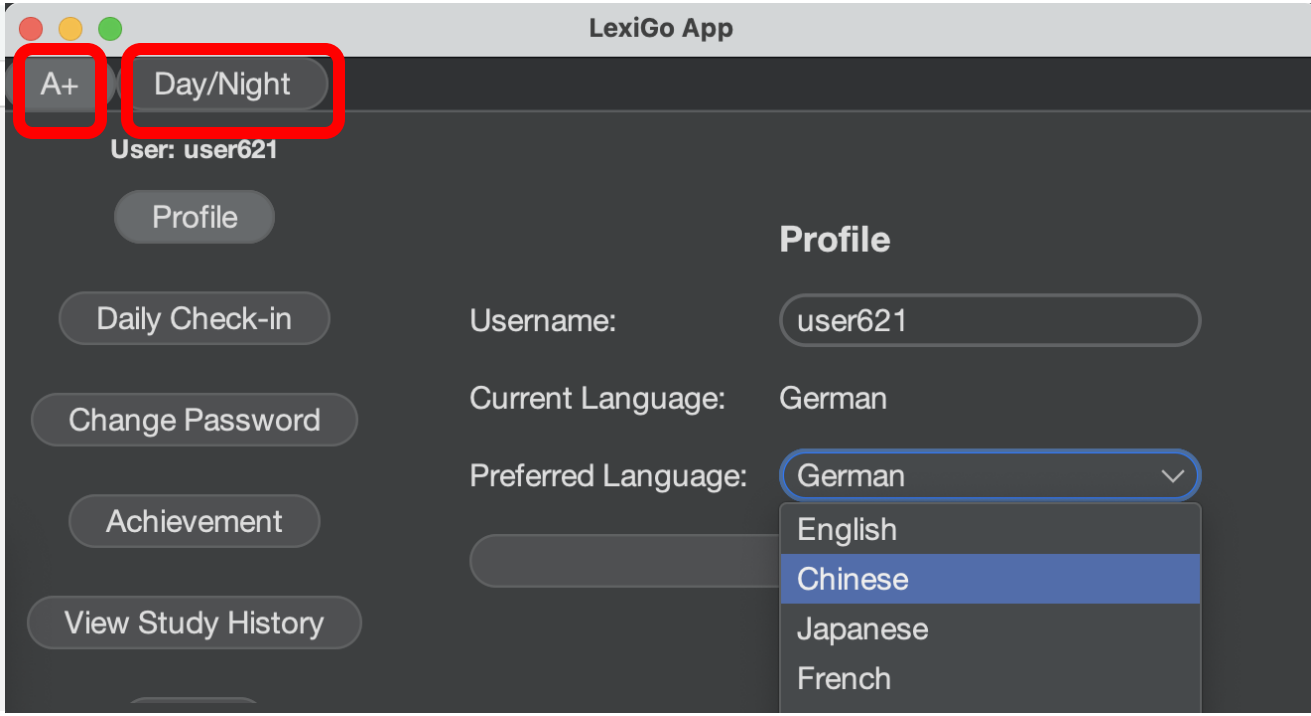


Element	Class, %	Method, %	Line, %
all	97% (182/186)	86% (610/709)	82% (1938/2336)
app	0% (0/4)	0% (0/23)	0% (0/207)
data_access	100% (11/11)	100% (61/61)	98% (202/205)
entity	100% (24/24)	100% (74/74)	100% (154/154)
infrastructure	100% (8/8)	100% (14/14)	100% (126/126)
interface_adapter	100% (52/52)	100% (188/188)	100% (416/416)
use_case	100% (53/53)	100% (164/164)	100% (494/494)
view	100% (34/34)	58% (109/185)	74% (546/734)

ACCESSIBILITY REPORT

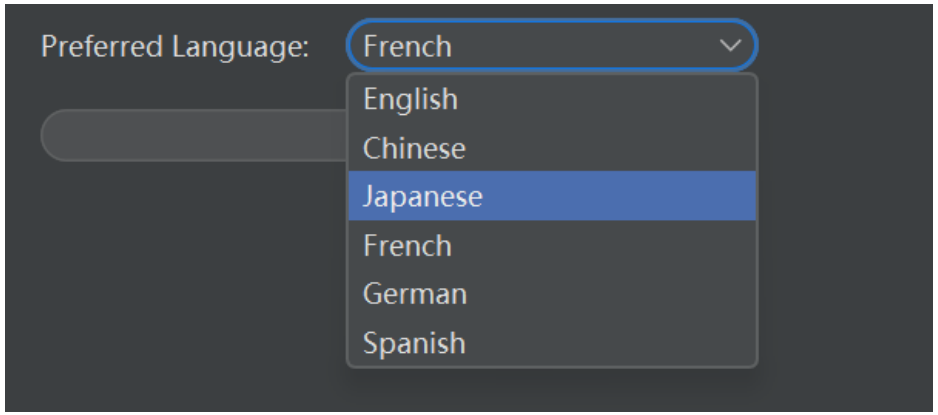
Principle 2: Flexibility in use





ACCESSIBILITY REPORT

- Fluent English speakers who want to learn a new foreign language in:
- Users who prefer a desktop experience



- **Social model**: Limitation caused by restricted language set
- Future extension can expand available target languages to reduce barrier

Full Accessibility report on Readme.md

Accessibility Report

Audience. Introductory-level English learners using a desktop vocabulary app. **Model.** We follow the **social model of disability**: barriers live in the product environment, so we design the default UI for everyone—no separate "accessible mode".

Universal Design principles — current status

Principle	Status	What's in LexiGO today	Next steps (high-impact)
1. Equitable Use	Partial	One UI for all users; no segregated interface.	Programmatic **name/role/state** for every control; avoid color-only cues; text alternatives for decorative images.
2. Flexibility in Use *(focus)*	Strong	**200% text zoom** without content loss; **Dark/Light themes**; mouse and most screens support keyboard.	Guarantee a **keyboard-only path** for all core tasks; shortcuts (Space=flip, ←/→=prev/next, Enter=submit).
3. Simple & Intuitive *(focus)*	Strong	Plain labels ("Daily Check-in", "Profile"); predictable flows.	60-second first-run tips; clearer, actionable error messages.
4. Perceptible Information	Partial	High-contrast themes; icons **and** text labels.	Verify contrast ≥ **4.5:1** on both themes; label inputs via `setLabelFor`; accessible descriptions for logos/illustrations.
5. Tolerance for Error	Improving	**Logout confirmation dialog** prevents accidental sign-out (`JOptionPane.YES_NO_OPTION`; **Esc/No** cancels). Duplicate-password check; informative alerts.	Confirm/undo for other destructive actions; link form errors to fields; **Esc** closes dialogs and returns focus consistently.
6. Low Physical Effort	Strong	Few steps to start studying; no drag/long-press; generous click targets.	Optional "auto-start study" to cut routine clicks; keyboard shortcuts to reduce hand travel.
7. Size & Space for Use	Partial	Resizable window; large-text mode keeps layout usable.	Add a min-window size and scrolling in tiny layouts; wrap long labels; enlarge small toggles for touch devices.

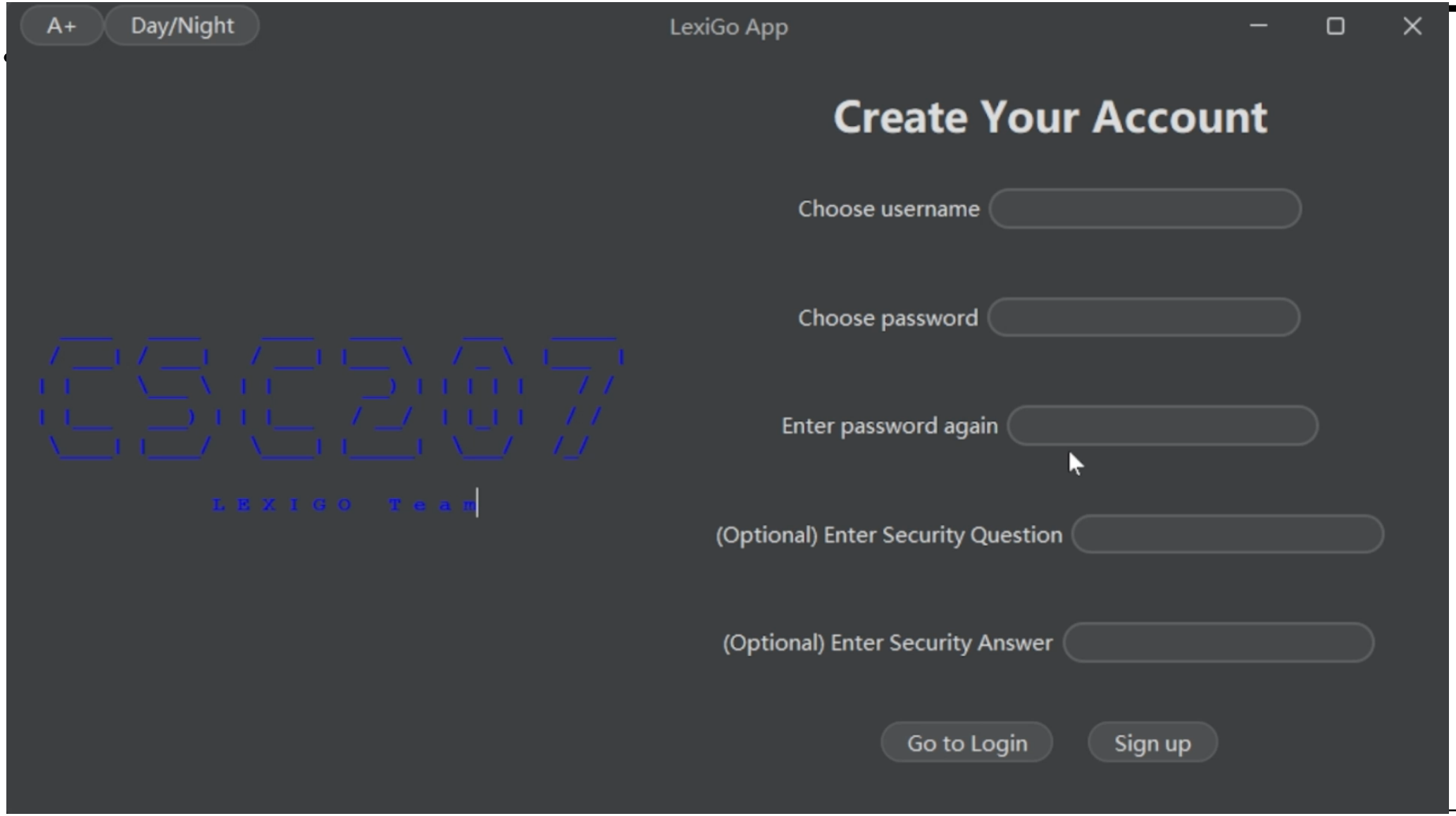
Per-principle notes

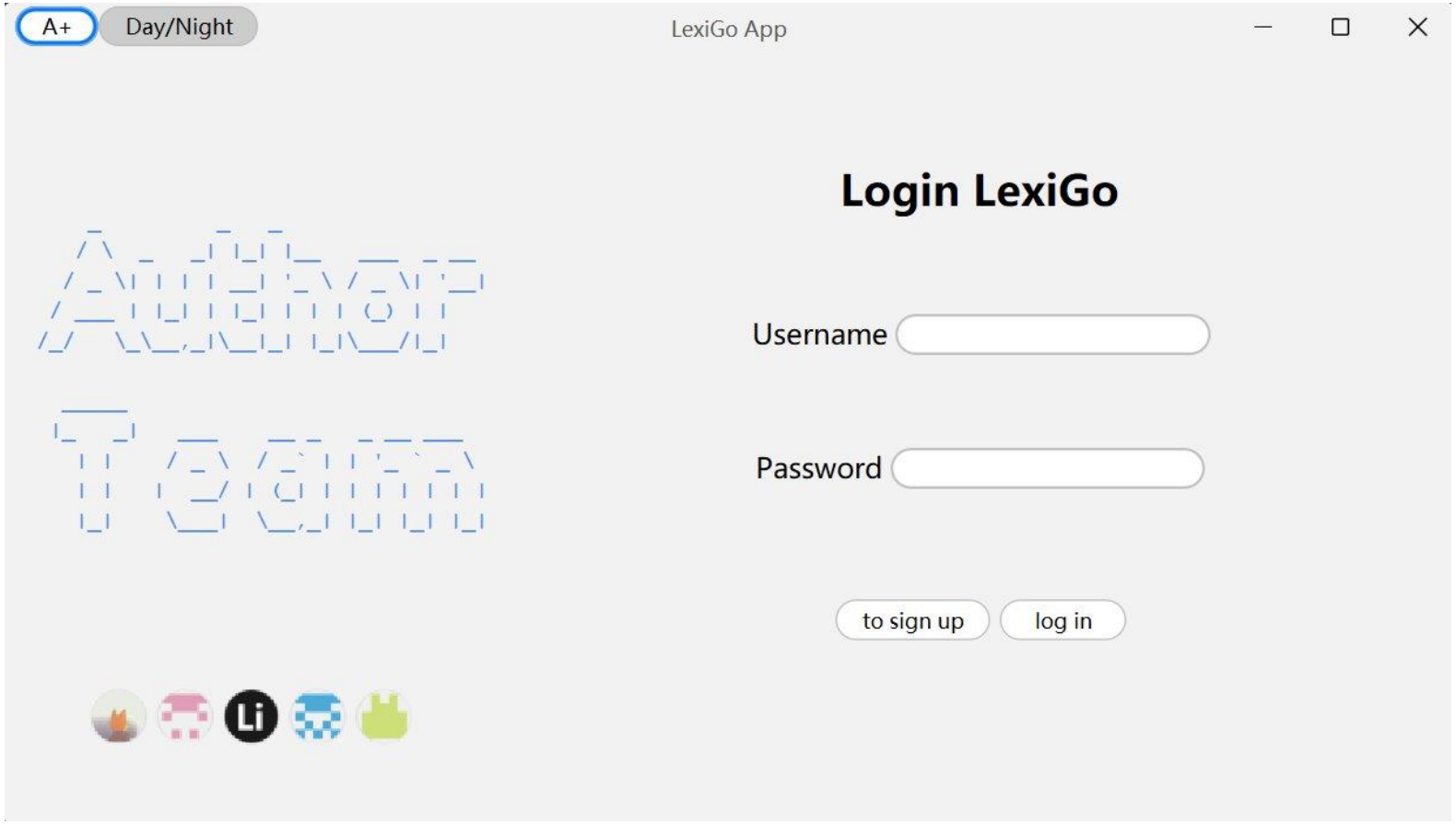
- **Equitable Use:** One UI for all; themes and large text benefit everyone. • Future: complete AccessibleContext naming and alt text.
- **Flexibility in Use:** 200% zoom + keyboard/mouse on core screens. • Future: full keyboard paths + shortcuts.
- **Simple & Intuitive:** Clear labels and flows. • Future: brief first-run tips + actionable errors.
- **Perceptible Information:** High contrast, text + icon signals. • Future: verify 4.5:1 contrast; add input labelling and logo descriptions.
- **Tolerance for Error: Logout requires confirmation; Esc/No aborts**, plus duplicate-password check. • Future: confirm/undo elsewhere; field-linked errors; consistent Esc handling.
- **Low Physical Effort:** Minimal steps; no precision gestures. • Future: auto-start and shortcuts.
- **Size & Space:** Resizable; large text preserves layout. • Future: min size, scroll when tiny, bigger toggles for touch.

Market / Audience. LexiGO targets **native English speakers aged 10+** who want a fast, low-friction way to build foreign-language vocabulary—students preparing for exams, adult self-learners, travelers, and language-club members. It suits users who prefer a **distraction-free desktop** experience with short daily sessions and gentle motivation (streaks and badges). Schools and clubs can deploy LexiGO in labs because it runs as a **single JAR** with local storage. Privacy-minded learners who dislike cloud lock-in appreciate **offline-friendly** study and on-device data.

Access & ethics. Today the app may be **less usable for screen-reader and keyboard-only users** because some controls lack programmatic names/roles and a complete keyboard path. Learners whose **UI language isn't supported** face an environmental barrier; we are expanding localization. We are closing gaps by adding accessible names, visible focus, dialog **Esc** handling, and field-linked error messages. Desktop-only distribution may also exclude people who rely solely on mobile; a lightweight web/desktop hybrid is on our roadmap.

FUNCTIONALITY DEMO




A+
Day/Night
LexiGo App
Login LexiGo
Username
Password
to sign up
log in

THANK YOU