Assignment 1 “Inception”: Deliverables (1/2)

- By **Sun 8 Sep**, submit in Canvas:
  - (ideally, after discussing a draft with your tutor in the preceding consultations)

1. Vision and Scope document for your project

- Required sections from template on slide 6:
  - Sect. 1.5 (Vision statement; see slide 8)
  - One other 1.x section of your choice (except 1.1; see slide 7/10)
  - One other 2.x section of your choice (except 2.1; see slide 11)
- Aim: “Convince an investor to fund your development project”

2. Use Case document (text-only, no UML diagrams)

- Describe three core features of your system as use cases:
  - Use **fully dressed** format template on slides 25-26
  - Include sections 1, 4, 6, 7, 8, 9 and 13
- Describe **at least 6 more use cases** you have come up with in brief format (see slide 20/23)

Assignment 1 “Inception”: Deliverables (2/2)

- By **Sun 8 Sep**, submit in Canvas:
  - (ideally, after discussing a draft with your tutor in the preceding consultations)

3. Project Plan

- List out what use cases (and end-points) will be completed in each of the upcoming phases
  - Use the techniques you learned in HBV401G for Effort Estimation and Prioritization
  - Consider which end-points (see following slides) you need to implement your use cases
  - Schedule a set of end-points you will complete in each sprint, considering the guidelines set in assignments 2-5 (see following slides)

4. Project skeleton

- One team member makes a project skeleton for a Java Spring project (e.g. via IntelliJ)
- Upload that project to Github and add members as contributors
- Each member makes a very small code contribution (via a commit)

Example End-Points & Difficulty - GET

- **Each end-point has a certain difficulty factor given in square brackets, e.g. [1] for easy, [2] for medium, and [3] for hard difficulty**
- Retrieve a single resource by ID (e.g. Item, user, order based on unique identifier) [1]
- Retrieve all resources in a collection (e.g. All users, all items) [1]
- Retrieve all resources with ordering/sorting (sort by specific attribute, e.g. Date, price) [2]
- Retrieve resources based on filtering criteria (e.g. Items under a certain price) [3]
- Retrieve paginated resources (e.g. A subset of resources) [3]
- Retrieve user-specific data (e.g. User profile, user settings) [2]
- Retrieve associated resouces (e.g. User favourites, user orders) [2]
- Retrieve aggregated data (e.g. Total sales, average rating of item) [3]
- Retrieve hierarchical data (e.g. Categories and subcategories) [3]
- Retrieve resource with dependant relationship (e.g. Product and reviews) [3]

Example End-Points & Difficulty – POST & PATCH

- POST
  - Create new resource (e.g. New item, new user account) [2]
  - Authenticate user credentials (e.g. Verify login attempt) [1]
  - Submit form data with media (e.g. Picture, audio upload, profile picture, document) [3]
  - Create an association between resources (e.g. Add item to users favourites) [2]
  - Trigger a complex operation (submit data that initiates complex logic, e.g. Startin a report, initiate a transaction) [3]
  - Submit batch data (e.g. Bulk create items) [3]
- PATCH
  - Update a single attribute of a resource (e.g. change password, update price) [1]
  - Partially update resource details (e.g. update user profile, change multiple item details) [2]
  - Update nested resource (e.g. update specific attributes of items in an order) [3]
  - Update resource with dependency check (e.g. only update if criteria is met) [3]

Example End-Points & Difficulty – DELETE & PUT

- DELETE
  - Delete a resouce (e.g. Delete item, user account) [2]
  - Delete relationship between resources (e.g. Remove association between user and item) [2]
  - Bulk delete resources (e.g. Delete multiple items based on list of unique Ids) [3]
  - Soft delete resource (e.g. Mark a resource as deleted without removing it) [2]
  - Delete based on criteria (e.g. All items older than date) [3]
- PUT
  - Replace a resource entirely (e.g. Update an items details completely) [2]
  - Upload or replace a large file resource (e.g. Replace profile picture, update document) [3]

Assignment Overview

In each assignment (2 - 5), your team will design and implement a set of RESTful API endpoints needed to support your use cases, along with a corresponding user interface (UI) to demonstrate their functionality

Your team is required to implement a different number of endpoints at differing levels of difficulty

For every assignment, pick a number of end-points so that their difficulties add up to the required difficulty (column ‘Total’). Your selection must include a minimum number of easy, medium and hard end-points as indicated by the table (see next slide)

You can use the same end-point type several times (e.g. multiple single entity retrieval by ID) and it will count every time

Assignment Overview (cont.)

Assignment	Total	Minimum # easy end-points	Minimum # medium end-points	Minimum # hard end-points
Assignment 2	2	2	0	0
Assignment 3	20	4	3	1
Assignment 4	20	4	2	2
Assignment 5	10	4	1	0

Example for Assignment 3:
6 easy end points (x1)
+ 4 medium end points (x2)
+ 2 hard end points (x3)
Gives a total difficulty score of 20

Or another example for Assignment 3:
4 easy end points (x1)
+ 5 medium end points (x2)
+ 2 hard end points (x3)
Gives a total difficulty score of 20

Grading Criteria

Your final grade for everything related to development and implementation will be based on the following criteria:

1. **Functionality of the API Endpoints (70%)**
    1. This portion of the grade will evaluate the correctness, performance, and robustness of your API endpoints.
    2. Each endpoint should correctly handle requests and provide appropriate responses, including error handling and edge cases.
2. **Implementation of the User Interface (UI) (30%)**
    1. This portion of the grade will evaluate the effectiveness and usability of the UI created to interact with your API endpoints.
    2. The UI can be a web page or application built using any front-end technology (e.g., Spring MVC, React, Angular, Vue).
    3. The UI should provide a clear and intuitive way for users to interact with each endpoint. It should include forms, buttons, and other elements to facilitate API requests and display responses.

Assignment 1 “Inception”: Presentation

- On **Thu 12 Sep**, present and **explain** your deliverables to your tutor:
  - What considerations are behind your vision and scope document?
  - How did you come up with your use cases and scenarios?
  - What considerations influenced the schedule of use cases in your project plan?
  - Show that all team members have access and contributed something to the code repository.

Assignment 1 “Inception”: Grading Criteria

- **Vision and Scope document (20%)**
  - Precise vision statement in Sect. 1.5 (7%)
  - Plausible argument in chosen Sect. 1.x (7%)
  - Plausible argument in chosen Sect. 2.x (6%)
- **Use Case document (60%)**
  - 3 core use cases in fully-dressed format (16% per use case)
    - Precise and plausible information in Sect. 1, 4, 6, 7, 13 (2% per use case)
    - Precise formulation of main scenario in Sect. 8 (7% per use case)
    - Precise formulation of alternative scenarios in Sect. 9 (7% per use case)
  - At least 6 more use cases in brief format (12% total)
- **Project plan (15%)**
  - Use cases plausibly assigned to project phases, considering the end-point difficulty factor (15%)
- **Code repository (5%)**
  - All team members committed at least a trivial input to the repository (5%)

Assignment 2 “Elaboration”: Deliverables

- By **Sun 29 Sept**, submit in Canvas:
  - (ideally, after discussing a draft with your tutor in the preceding consultations)

1. A **UML state machine diagram** showing the navigation between your system’s web pages
   - Or, if you’re using a client-side UI framework: Illustrating how a particular web page changes as the user is performing a use case
2. A **UML sequence diagram** showing the control flow between your project’s components
   - Collaboration of JSPs, controllers, services, repository, and potential client-side components in one exemplary request-response cycle
     - Make sure the method calls in the sequence diagram reflect the methods specified in the class diagram
3. A **UML class diagram** showing the complete design model of your project
   - Showing all the classes your final system shall consist of on the server side
     - with attributes, data types, methods, relations, multiplicities
4. Planned **end-points** (as per your assignment 1 project plan)
5. A link to your team’s **github repository**
   - Showing the implementation of your project’s backbone architecture and first features
   - Brief notes on project status, progress, exceptions

Assignment 2 “Elaboration”: Presentation

- On **Thu 3 Oct**, your current P.O. presents and **explains** your deliverables to your tutor:
  - What considerations are behind your UML diagrams, e.g.:
    - How is the scope of the features reflected in the state diagram?
    - How is the MVC pattern reflected in the sequence diagram?
    - How are the principles of encapsulation, separation of concerns etc. reflected in the class diagram?
  - How did the implementation proceed?
    - How far along are you in relation to the project plan?
    - What are the reasons for delays, and how are you planning to deal with them?
    - Demo the newly implemented features

Assignment 2 “Elaboration”: Grading Criteria

- **UML State Machine Diagram (10%)**
  - ✓ Correct syntax: Clean, syntactically correct UML (2%)
  - ✓ Plausible scope: Scope of diagram matches scope of previously submitted use cases (3%)
  - ✓ Clear specification: Clearly shows different pages and the conditions (triggers, guards) under which they are reached (5%)
- **UML Sequence Diagram (20%)**
  - ✓ Correct syntax: Clean, syntactically correct UML showing relevant call structures (methods and parameters) (5%)
  - ✓ Technical accuracy: Shows how your controller, application logic and data model classes collaborate (i.e. call each other) in a request-response cycle (15%)
- **UML Class Diagram (35%)**
  - ✓ Correct syntax: Clean, syntactically correct UML, including attributes and methods of classes (5%)
  - ✓ Plausible scope: Includes all necessary controllers, business logic, entities, persistence logic / interfaces for the previously submitted use cases (10%)
  - ✓ Clean design: Classes connected by plausible associations with multiplicities, class structure reflects MVC pattern, design principles of encapsulation, separation of concerns etc. (20%)
- **Implementation Progress (35%)**
  - ✓ Team activity and progress in relation to the project plan (20%)
  - ✓ Team awareness of project status, challenges and work ahead (10%)
  - ✓ Code quality: Clean, documented code (5%)

Assignment 3 “Construction I”

- By **Sun 20 Oct**, submit in Canvas:
  - A link to your team’s **github repository**
    - Showing the implementation of your project (end-points and UI)
    - Brief notes on project status, progress, exceptions

- On **Thu 24 Oct**, the phase’s P.O. presents and **explains** your deliverables:
  - How did the implementation proceed?
    - How far along are you in relation to the project plan?
    - What are the reasons for delays, and how are you planning to deal with them?
    - Demo the newly implemented features

- **Grading criteria: Implementation Progress (100%)**
  - ✓ Team activity and progress in relation to the project plan (75%)
  - ✓ Team awareness of project status, challenges and work ahead (15%)
  - ✓ Code quality: Clean, documented code (10%)

Assignment 4 “Construction II”

- By **Sun 10 Nov**, submit in Canvas:
  - A link to your team’s **github repository**
    - Showing the implementation of your project (end-points and UI)
    - Brief notes on project status, progress, exceptions

- On **Thu 14 Nov**, the phase’s P.O. presents and **explains** your deliverables:
  - How did the implementation proceed?
    - How far along are you in relation to the project plan?
    - What are the reasons for delays, and how are you planning to deal with them?
    - Demo the newly implemented features

- **Grading criteria: Implementation Progress (100%)**
  - ✓ Team activity and progress in relation to the project plan (75%)
  - ✓ Team awareness of project status, challenges and work ahead (15%)
  - ✓ Code quality: Clean, documented code (10%)

Assignment 5: Final Product – Presentation

- **On Thu 23 Nov**, each team presents their product to their tutor and other teams
  - 20 minutes per presentation (~5 minutes for each of the following parts, 5 minutes for questions)
- Your presentation must cover the following parts (in an order of your choice):
  - **A live demonstration of your final software (no videos, mockups etc.)**
    - Should include product's key use cases
  - **An overview of your system architecture**
    - What are the key components, and how do they communicate?
    - What aspects are client and server responsible for?
    - How are you storing and accessing data?
    - Any particular aspects of your design you would like to highlight?
  - **A retrospective on your project work**
    - What went well? What difficulties did you encounter?
    - How did you plan to structure / manage your work? How did that turn out?
    - What would you do differently next time?
    - How would you avoid any difficulties you encountered?

Assignment 5: Grading Criteria

1. **Product Implementation (70%)**
   - ✓ Software runs smoothly
   - ✓ Key use cases are working
2. **Deployment (5%)**
   - ✓ Is the app accessible via the web
3. **Retrospective & Discussion (25%)**
   - ✓ Architecture described, illustrated clearly
   - ✓ Key design decisions described and illustrated clearly
   - ✓ Room for improvement discussed critically
   - ✓ Design & development process over the course of the semester described clearly
   - ✓ Handling of technical / methodical / collaboration challenges discussed critically

- **Submitted code**
  - Not graded explicitly, but checked for conformity with presented prototype and architectural requirements
    - Inconsistency of submitted code with presented product may lead to reduction of Product Demonstration grade
    - Completely messy code may lead to reduction of Architecture & Design grade
  - **No need to polish after presentation!**
  - **Focus on exam prep instead!**

- **Presentation Grade**
  - Only counted for a product owner giving the whole presentation
  - Not counted for shared presentations