CSCI3100 Tutorial 2: Project Introduction

27th January 2025

LAM, Man Ho

香港中文大學
The Chinese University of Hong Kong

ONE

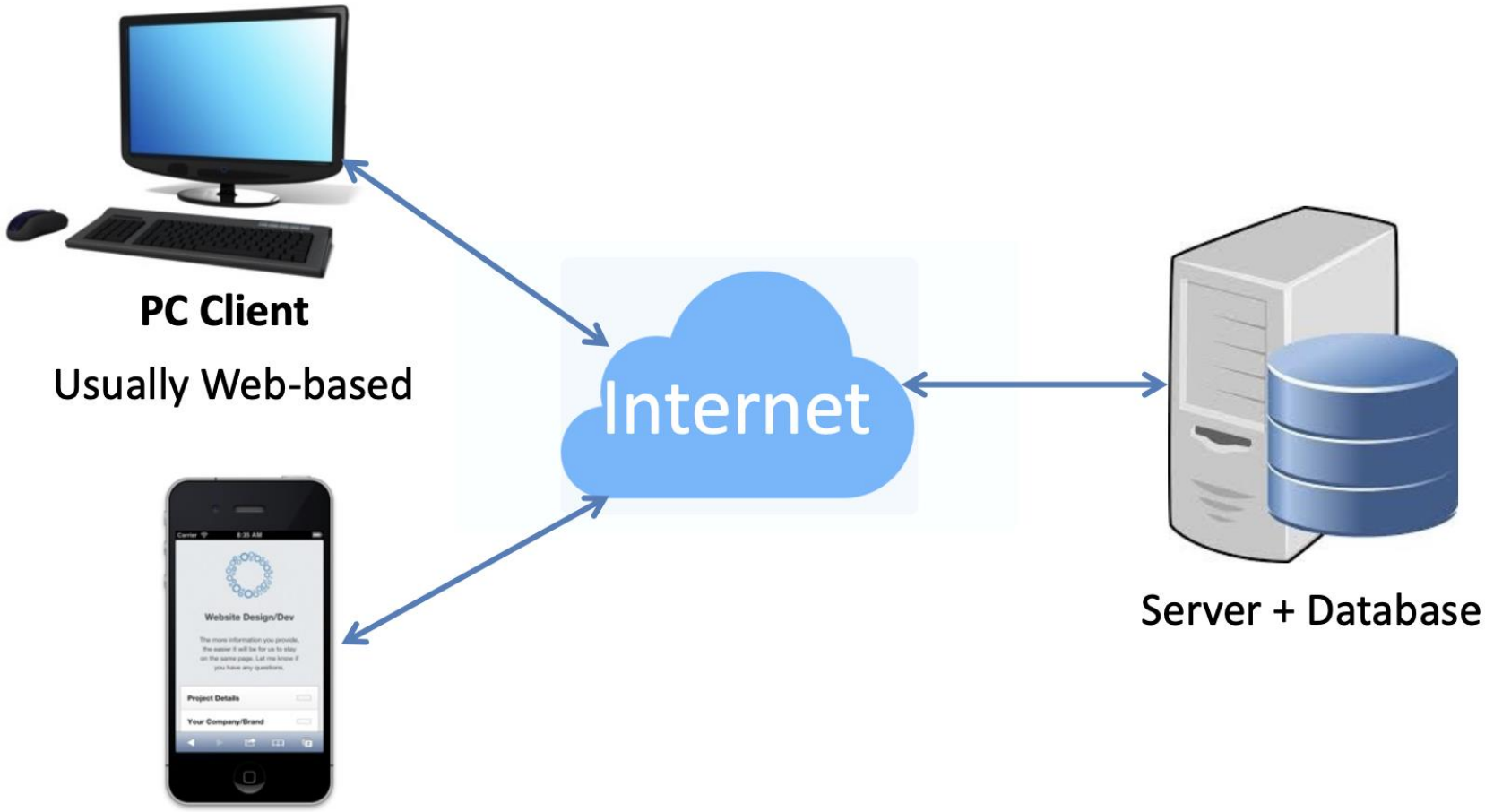
Project Overview

Objective

Practice what you are learning in this CSCI3100 Software Engineering course by specifying, designing, implementing, testing, and documenting a typical software project (e.g., a web-based client-server application, or a software game application).

Modern Application

Software Architecture



Important Stats

- Project accounts for 30% of the course grade
- A journaled, auditable software development process
  - Documentation (Report, code) – 55%
    1. Requirements Specification
    2. Design and Implementation
    3. Testing Document
    4. Release Notes and User Manual
  - Software product (Demo day) – 45%

Schedule

Date	Phase Deliverables	Weightings
7 February, 2025	Requirements Specification (1st Version)	10%
7 March, 2025	Design and Implementation (1st Version)	15%
18 April, 2025	Requirements Specification (Final Version)	5%
	Design and Implementation (Final Version)	10%
2 May, 2025	Project Demo	45%
9 May, 2025	Testing Document	10%
	Release Notes and User Manual	5%

Phase 0: Grouping

- 3 - 5 students for each group
- All students in a group work on the same project for the entire project duration
- No joint work over any technical aspects of the project is allowed between any two teams

- **No Free-rider**
  - **Group members can report free-riders**
  - **Instructors will verify the validity of the complaints**

Phase 1: Requirements Specification

- First Version: Duration - 2 weeks (**7 Feb 23:59**)
- Final Version: Duration - 6 weeks (**18 Apr 23:59**)
- Total Weighting: 15%

- Submit a document following the Software Requirements Specification (SRS) format. The document should include descriptions of requirements, functionalities, features, and a high-level overview of the system architecture.

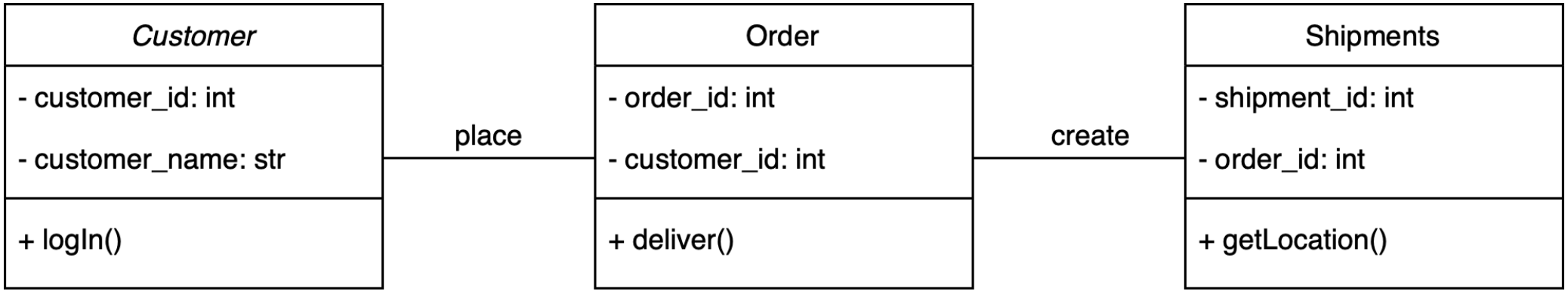
Phase 2: Design and Implementation

- First Version: Duration - 4 weeks (**7 Mar 23:59**)
- Final Version: Duration - 6 weeks (**18 Apr 23:59**)
- Weighting: 25%

- Submit a developer-oriented document that specifies all the design details and implementation considerations. Including
  - UML Diagrams (**Recommended**)
  - Module-level design with responsibilities and interactions
  - Database design
  - Coding standards
  - ...

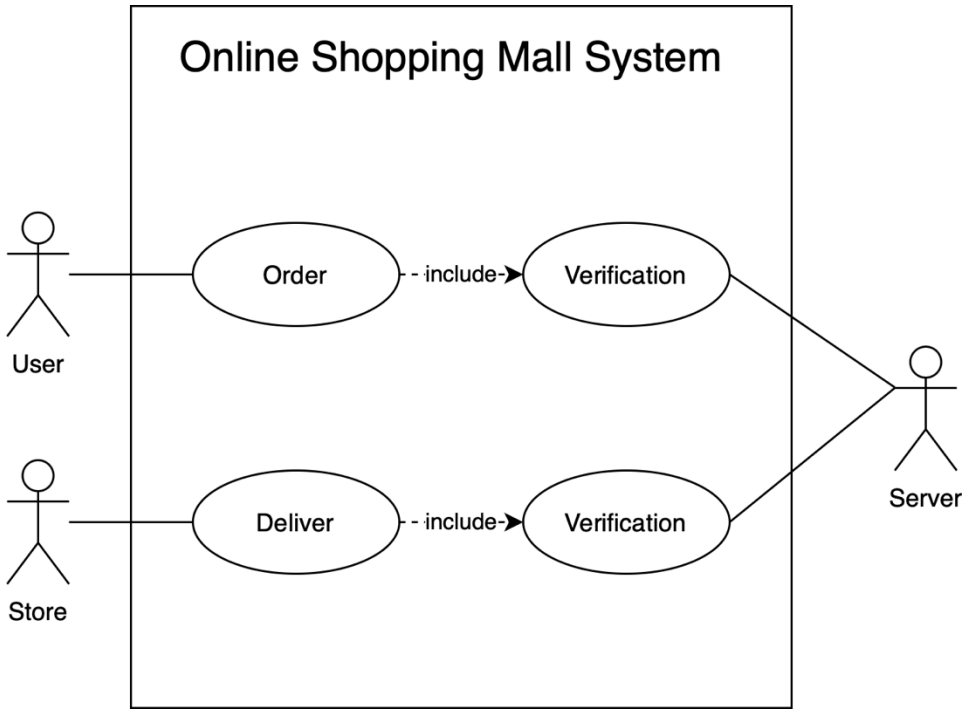
Phase 2.1: Class Diagrams

- Class Diagram (Module-level design)
  - Show attributes (fields) and methods (functions)



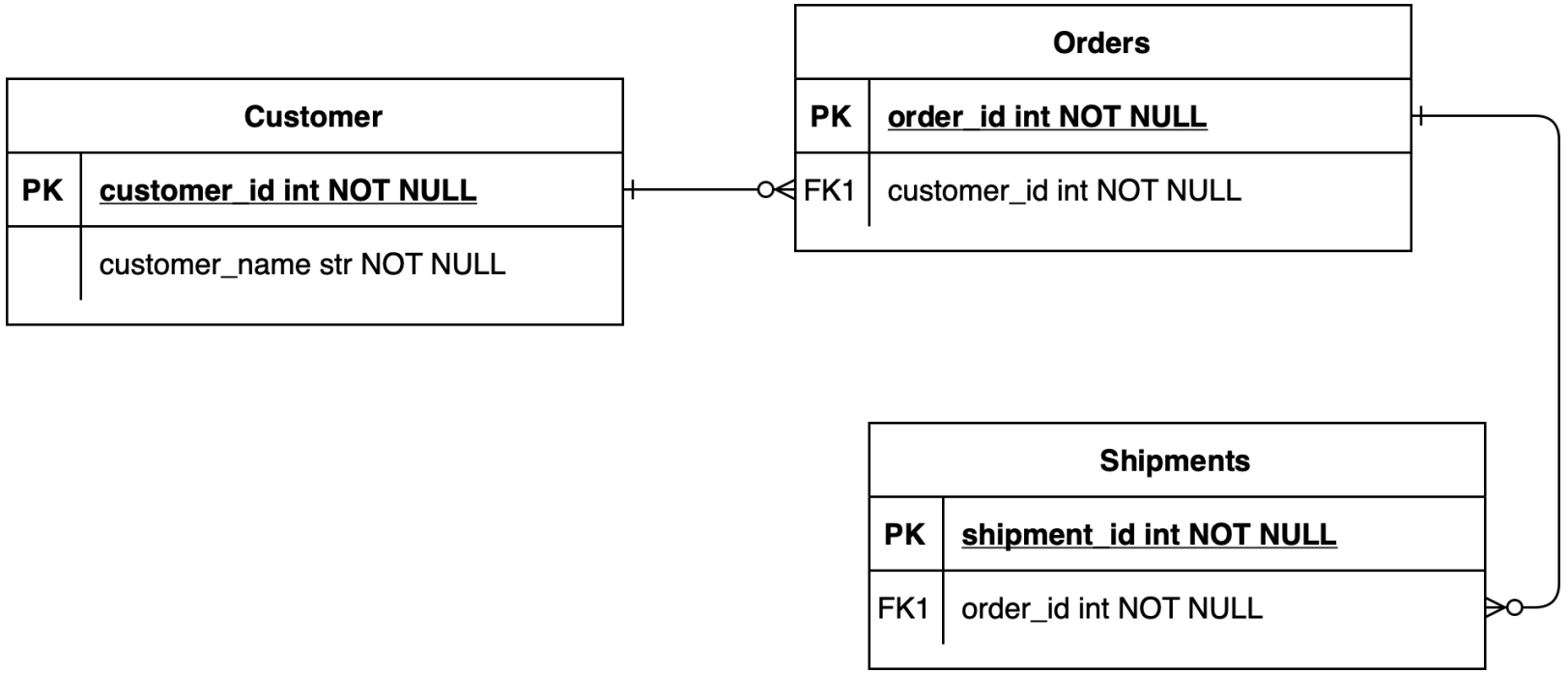
Phase 2.1: UML Diagrams

- UML Diagram
  - Captures different aspects of the system's behavior and interactions



Phase 2.1: ER Diagrams

- ER Diagram (Database design)
  - Visualizes the database schema



Phase 3: Project Demo

- Demo Day: Duration: 2 weeks (**2 May**)
- Weighting: 45%

- In this phase, you are completing your project. You will need to make a **demonstration** of your complete application (15 minutes per group)

- Signup schedule for demonstration will be announced later

Phase 4: Testing

- Testing Document: Duration – 1 weeks (**9 May**)
- Weighting: 10%

- Submit a developer-oriented document that describes the testing process and strategies in detail. The document must include:
  1. Test Plan
     - Specifies which components were tested and which were not with reasons for exclusions.
  2. Representative Test Cases
     - Includes test cases designed to evaluate the system's key functionalities
     - Describe the reason behind the design of these test cases and the approach for testing
     - Examples should cover typical, edge, and exceptional scenarios to ensure comprehensive validation

Phase 5: Delivery

- Release Notes and User Manual: Duration – 1 weeks (**9 May**)
- Weighting: 5%

- Coding should be frozen
- Submit **release notes** that includes release notes with version information, included and excluded features and, bug fixes
- Along with a **user manual** detailing the software overview, system requirements, installation steps, and feature usage instructions

Project Topic

- **No specific topic requirements**
- Choose any kind of software project that excites you

- Projects from last few years for your reference:
  - Content-oriented application: X (Twitter), Facebook
  - Systems: Online Shopping Mall System, Course Registration System
  - Gaming Software : Pac-Man, Snake.io, Gobang

Requirements and Grading Criteria

- **Architecture, Design and Implementation**
  1. Software architecture is scalable (**4 points**)
  2. Design and implementation are scalable (**4 points**)
  3. The design is consistent with the Requirements Specification (**4 points**)
  4. Easy to use (**2 points**)

Technical Requirement: Global Database

A global database must be employed for storing data, for operations like user management, licence management and system configuration.

- SQL database (e.g., MySQL, or Sqlite)
- NoSQL database (e.g., MongoDB, or Redis)
- Worth for **4 points**

Technical Requirement: User Interface

An user interface for users to interact with the system.

- Web-Based System: Accessible via web browsers
- Game Interface: Accessible through gaming platforms or installation
- Graphical, beautiful, and user-friendly
- Worth for **6 points**

Technical Requirement: User Management

- **Signup** Operation (**2 points**)
  - Allows users to create new profiles for personalized access
- **Login** and **Logout** Operations (**2 points**)
  - Access to core functions is restricted to logged-in users
  - Users must log in using valid credentials and can securely log out when finished

Technical Requirement: License Management

- Authorization by key or key-file (**4 points**)
  - Authorization: Implemented via key or key-file
  - Restricts system access to users with valid authorization
    - System validates the key or key-file before allowing usage
    - Ensures only licensed users can access the software

Technical Requirement: Code Documentation

- Worth for **4 points**
- Provide clear explanations of your code for better understanding
- Critical during development and future maintenance
  - Helps other developers understand and work with your code
  - Enhances code readability and maintainability

```
// Calculate the compound interest
// Formula: A = P(1 + r/n)^(nt)
// Where: A = final amount, P = principal, r = annual interest rate,
// n = number of times interest is compounded per year, t = number of years

let finalAmount = principal * Math.pow((1 + (rate / compoundingFrequency)),
(compoundingFrequency * time));
```

Application-specific Requirements

- Completing the system
- For a group of n **members**, you must implement at least $n - 1$ **externally observable features**
- Each of the $n - 1$ application-specific features will contribute $\frac{9}{n-1}$ **points** to the overall score

- Example (content-oriented application):
  - Posting text and media content
  - Follow and Unfollow system

Grading Criteria

Aspect	Functionality	Points
Architecture, Design and Implementation	Software architecture	4
	Design and implementation	4
	Consistency with Req. Spec.	4
	Easy to use	2
Technical Requirements	Global Database	4
	User Interface	6
	User Management System	2 + 2
	License Management	4
	Code Documentation	4
Application-specific Requirements	n – 1 externally observable features	9
	Total	**45**

Submission (Documents)

- Documents must be submitted alongside signed VeriGuide receipts

1. Requirements Specification:
   - “Group**_Requirements_Specification{_VeriGuide}”
2. Design and Implementation
   - “Group**_Design_Implementation{_VeriGuide}”
3. Testing Document:
   - “Group**_Testing_{VeriGuide}”
4. Release Notes and User Manual
   - “Group**_Release_Notes_User_Manual_{VeriGuide}”

- Replacing ** with your group ID (without quotes)
- For example: Group00_Testing.pdf and Group00_Testing_VeriGuide.pdf

Submission (Code)

- **All** project materials (including source code, images, flash files, database files, documentation, etc.) must be maintained using **GitHub**
- You MUST submit your project to GitHub and faithfully **record your coding activities**
- We will **NOT** accept submissions via other approaches
- Tutors will **NOT** help you debug your code

TWO

Requirements Specification

Documentations

- Font, Font Size, and Styles:
  - The document must use Times New Roman, size 11 throughout
- Cover page:
  - Name of the document
  - Project title (you may freely name your own project)
  - Document version number and revision history
  - Printing date
  - Group ID
  - Names and SIDs of group members
  - Department

Contents

1. Document Revision History
2. Introduction
   - Overview
   - Definitions, acronyms, and abbreviations
   - References
3. Assumptions and Dependencies
4. High-Level System Architecture
5. Functional Requirements
   - Summarization of functional requirements
   - Actors
   - Use Cases with diagrams

Part 1: Document Revision History

- Provide a record of all changes made to the document over time
- Include details such as the revision number, the person or team responsible for the updates, date, and description of changes

Version	Revised By	Revision Date	Comments
0.1	Osamah Yacoub, AlliedSoft	Nov 21 2002	Initial draft.
0.11	Osamah Yacoub	Dec 2 2002	Minor modifications for information consistency.
0.2	Osamah Yacoub	Dec 12 2002	Changed according to Dec 3rd stakeholders' meeting results.
1.0	Osamah Yacoub, Ahmad Arrabi	Dec 31 2002	Finalized requirements, use cases, and component Business Priorities.

Part 2: Introduction

1. **Introduction of your software product**
2. **Overview**

   The evil-minded lecturer Kei Kaiju is seeking a software system to aid in his quest for world domination. This system must include a user interface accessible to users, which can be web-based, a desktop application, or a mobile application. There will be various interactions between the system and the user, with feedback provided through multiple channels. Any functional system that meets these requirements could contribute to Kei's nefarious plans.

3. **Definitions, acronyms, and abbreviations**
   - Give a list of definitions, acronyms, and abbreviations used in your doc.
   - For example: UI – User Interface
4. **References**
   - A list of all references used to produce this document

Part 3: Assumptions and Dependencies

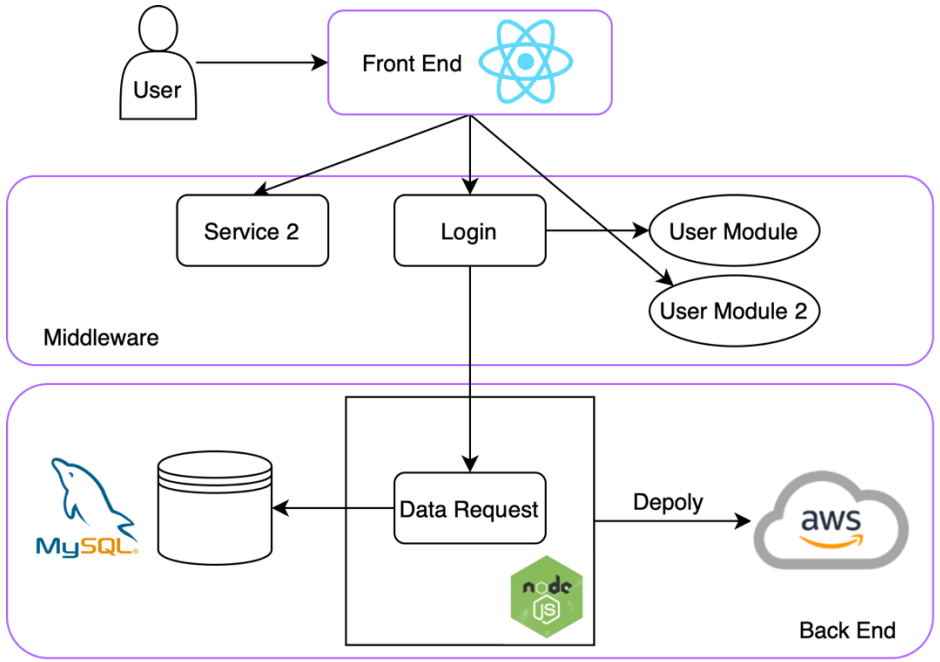
- Examples:
  - Web browser (Microsoft Internet Explorer 11.0 or above)
  - Operation Systems (Window 8.1 or above)
  - Supported Languages
  - User Connectivity (CSE Intranet)
  - Window Size & Resolution

- Our requirements:
  - Able to run on Linux, Windows, or Android released within 5 years
  - Able to support major decent browsers, including Firefox and Chrome

Part 4: High-Level System Architecture

- Provides an **abstract overview** of your system's structure, including the components, interactions, and actions
- It should comprehensively outline the major modules, data flow and integration points to ensure the clarity in system design, serving as a **blueprint** for further implementation

Three-tire Architecture



Part 5: Functional Requirements

1. **A list to summarize the functional requirements of your system**
   - User Management
   - User Operations

Req. No	Title	Description
R1	User Management	
R1.1	User Sign up	Allow new users to create accounts.
…	…	…
R2	User Operations	
R2.1	Posting	Allow users to post text and media content.

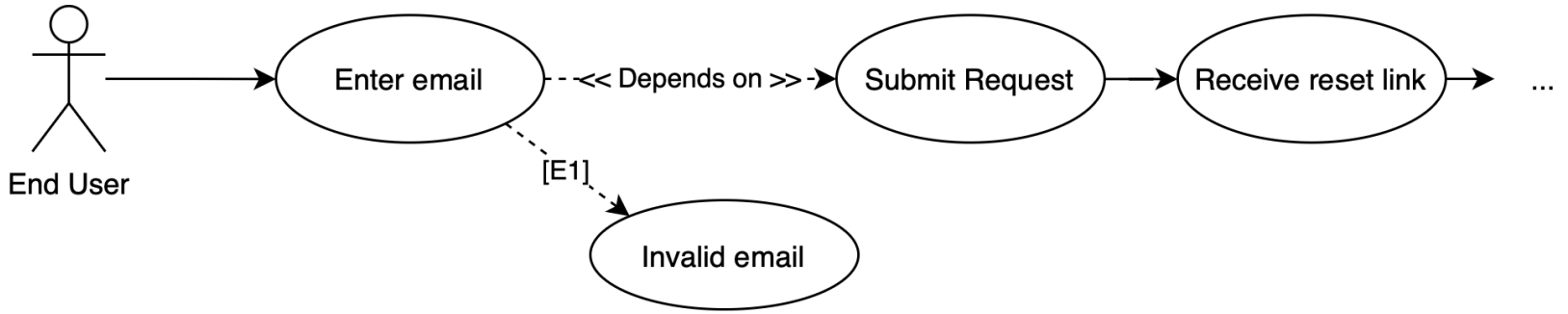
2. **Actors** (entities that interact with a system by exchanging data)
   - End User
   - System Administrator
   - …

Part 5: Functional Requirements

3. Use Cases

- **Forgot Password and Change Password Using Email**
  - Initiator: End user
  - Description: This use case allows users to reset their password by requesting a password reset link sent to their registered email address
  - Basic Flow of Events: The system asks the actor to enter their email address. The actor enters their email address and submits the request [E1] ...
  - Exceptional Flow of Events:
    - [E1] If the email address does not exist in the system, the system displays: "Email not found."
    - [E2] …

4. Use Case Diagrams



THREE

Demonstration

Thank you!

香港中文大學
The Chinese University of Hong Kong