Lab Manual 1

Course Code: CS-165L **Course Title: Software Engineering**

Course Instructor: Laeeq Khan Niazi

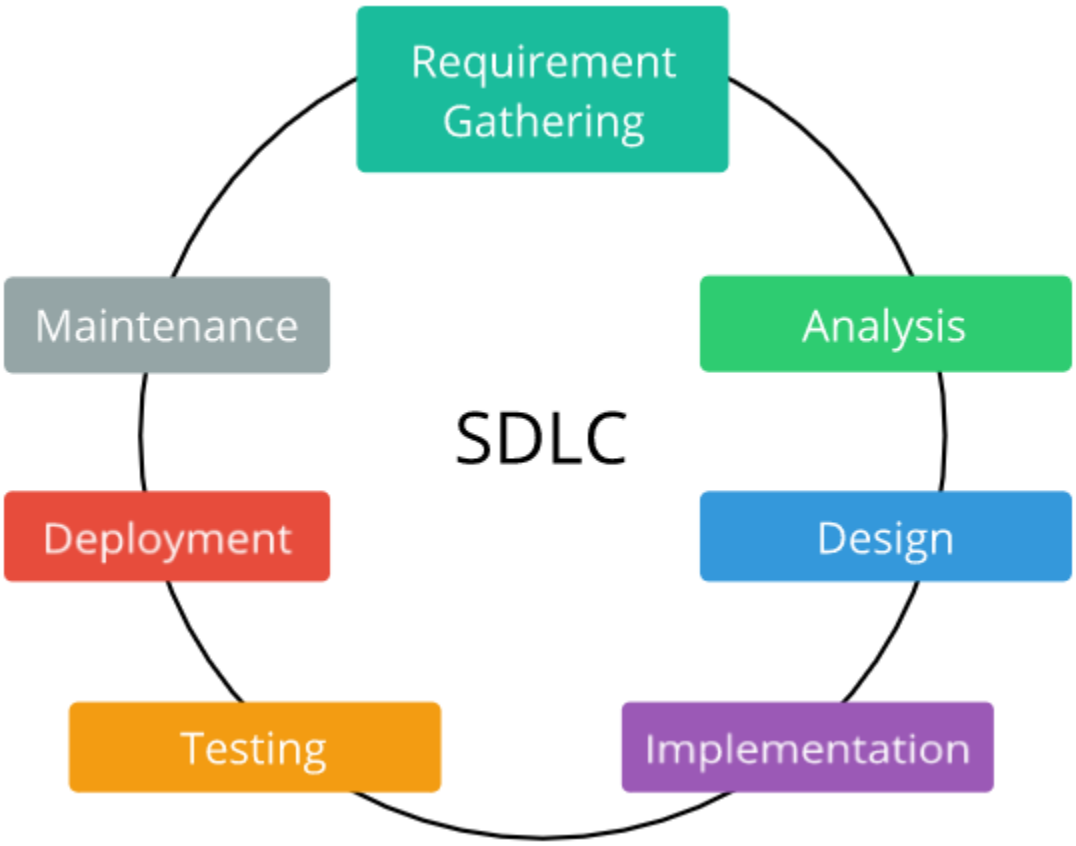
Introduction to the Software Engineering Lab

This laboratory experience will differ from traditional software engineering labs. We will delve into a broader range of cutting-edge technologies, encompassing standard software development methodologies, software architecture design, and the creation of comprehensive software documentation.

In this lab, every student will take on dual roles as both a software developer for one peer and as the owner of a software project for another. To put it simply, you will be tasked with developing software for a fellow student (referred to as "X Student") while simultaneously taking on the role of the owner of a software project for another student (referred to as "Y Student"). This unique approach aims to provide you with a more holistic understanding of software development, encompassing both the development process and the responsibilities of managing a software project from the owner's perspective.

Introduction to Software Engineering

Software engineering is a systematic and disciplined approach to designing, developing, testing, and maintaining software. The steps involved in software engineering can vary depending on the specific development methodology you're following (e.g., Waterfall, Agile, DevOps), but here's a general sequence of steps that are commonly followed:



Requirements Gathering and Analysis:

- Understand and document the needs and requirements of the software from stakeholders.
- Create user stories, use cases, or functional specifications.

System Design:

- Create a high-level architectural design of the software.
- Design data structures, algorithms, and user interfaces.
- Break down the system into smaller components or modules.

Detailed Design:

- Develop detailed specifications for each component or module.
- Define data models and databases if applicable.
- Create class diagrams, sequence diagrams, and other design documentation.

Implementation (Coding):

- Write the actual code for each component or module.
- Follow coding standards and best practices.
- Perform code reviews to ensure quality and consistency.

Testing:

- Develop a test plan that outlines how the software will be tested.
- Perform unit testing to check individual components.
- Conduct integration testing to ensure that components work together.
- Execute system testing to evaluate the entire system's functionality.
- Perform user acceptance testing (UAT) with stakeholders.

Debugging and Bug Fixing:

- Identify and fix defects and issues discovered during testing.
- Regression testing may be necessary after bug fixes.

Documentation:

- Create comprehensive documentation, including user manuals and developer guides.
- Document code, APIs, and architecture for future reference.

Deployment:

- Prepare the software for deployment to the target environment.
- Deploy the software to production or staging servers.

Maintenance and Updates:

- Monitor the software in the production environment.
- Address any issues, bugs, or security vulnerabilities as they arise.

- Implement updates and enhancements based on user feedback and evolving requirements.

Feedback and Iteration:

- Gather feedback from users and stakeholders.
- Plan and prioritize new features or improvements.
- Iterate through the development process as needed to incorporate changes.

Task 1:

Write down the problem statement of your project (You can make web, desktop, or mobile application)

Task2:

Write down the functional and non-functional requirements of your project.

Example of LMS functional and non-functional requirements.

Functional Requirements:

1. **User Registration and Authentication:**
    - Users can create accounts with unique usernames and passwords.
    - Users can log in securely using their credentials.
    - Password reset and recovery mechanisms are in place.
2. **User Roles:**
    - The system supports different user roles, such as students, instructors, administrators, and parents (if applicable).
    - Each role has specific permissions and access levels.
3. **Course Management:**
    - Instructors can create and manage courses.
    - Course information includes title, description, schedule, and materials.
    - Courses can be categorized by subject, level, or department.
4. **Enrollment and Registration:**
    - Students can enroll in courses.
    - Instructors can approve or deny enrollment requests.
    - Automated waitlisting and capacity management (if applicable).
5. **Content Management:**

- Instructors can upload course materials (lectures, documents, videos, quizzes, assignments).
- Support for various content formats (PDF, video, audio, HTML, etc.).
- Version control for content updates.

6. **Learning Resources:**
   - Discussion forums and chat features for student-instructor and student-student interaction.
   - Notification system for updates, announcements, and discussions.
7. **Assessment and Grading:**
   - Support for various assessment types, including quizzes, assignments, and exams.
   - Automated grading and feedback.
   - Gradebook for tracking student progress.
8. **Progress Tracking and Reporting:**
   - Students can track their progress, including completed modules and grades.
   - Instructors and administrators can access detailed reports on student performance.
9. **Communication and Notifications:**
   - Messaging system for communication between users.
   - Email or SMS notifications for announcements, deadlines, and forum activity.
10. **User Profile and Portfolio:**
    - Users can create and update their profiles.
    - Portfolios to showcase achievements, certificates, and skills.
11. **Integration:**
    - Integration with external systems for single sign-on (SSO), payment processing, and analytics.
12. **Accessibility:**
    - Compliance with accessibility standards (e.g., WCAG) for users with disabilities.

Non-Functional Requirements:

1. **Performance:**
   - The system must support many concurrent users.
   - Response times should be within acceptable limits, even during peak usage.
2. **Scalability:**
   - The system should be able to scale horizontally or vertically to accommodate growth.
3. **Security:**
   - User data, including personal information and grades, must be securely stored and encrypted.
   - Role-based access control (RBAC) to ensure data privacy and prevent unauthorized access.

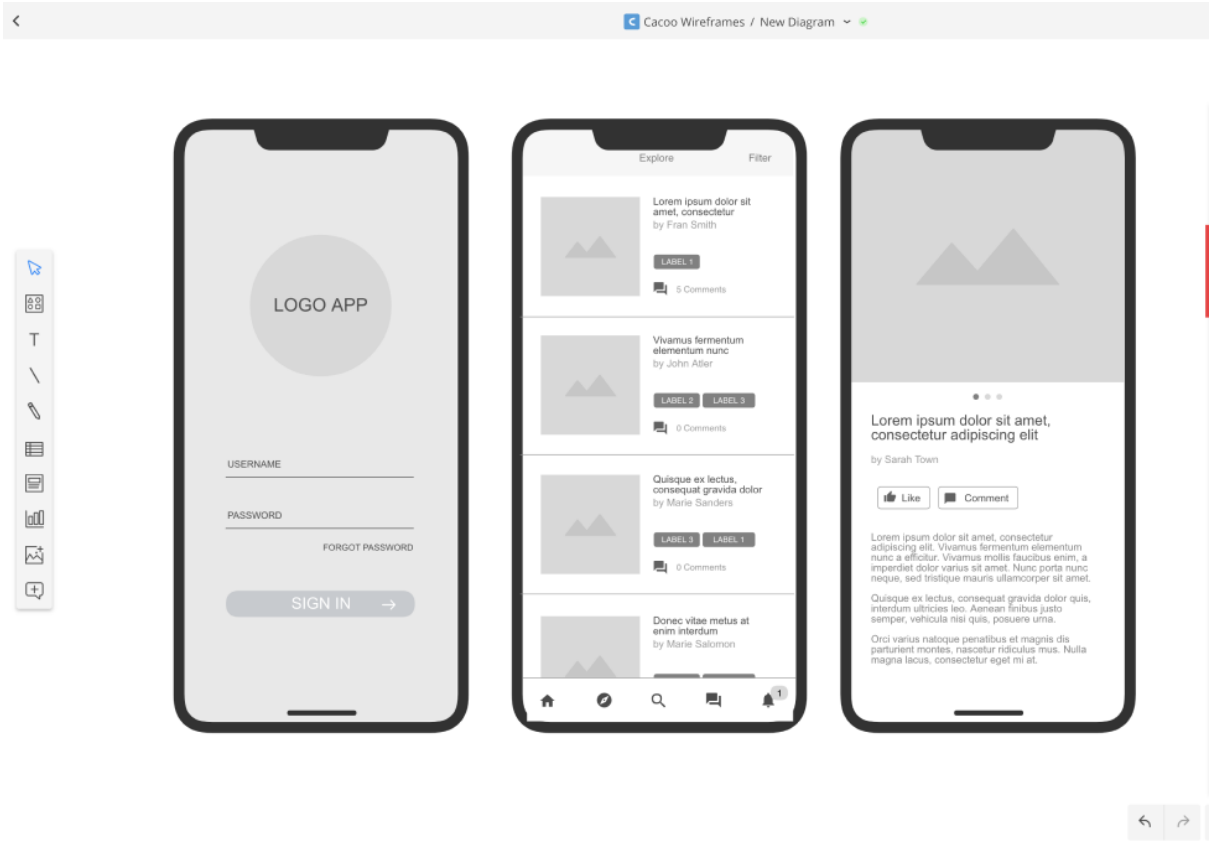
4. **Reliability:**
    - The system should be available and reliable with minimal downtime.
    - Regular data backups and disaster recovery plans.
5. **Usability:**
    - The user interface should be intuitive and user-friendly.
    - Support for multiple languages and accessibility features.
6. **Compatibility:**
    - Compatibility with various web browsers and devices (desktop, tablet, mobile).
7. **Scalability:**
    - The system should be able to handle a growing number of users, courses, and content.
8. **Compliance:**
    - Compliance with relevant educational standards (e.g., SCORM, xAPI).
9. **Performance Monitoring:**
    - Monitoring tools for tracking system performance and user behavior.
10. **Data Retention and Privacy:**
    - Clear data retention policies and compliance with data privacy regulations (e.g., GDPR).
11. **Reporting and Analytics:**
    - Robust reporting and analytics tools for administrators and instructors to assess system usage and student performance.
12. **Documentation and Support:**
    - Comprehensive documentation for users and administrators.
    - Support channels for issue resolution and assistance.

Takeaways

1. Learn Figma Tool
2. Learn Start UML Tool
3. Learn Trello (Project Management Tool)
4. Create Account on Overleaf for document management.
5. Create a GitHub repo for you project.
6. Install Slack for your project communication.

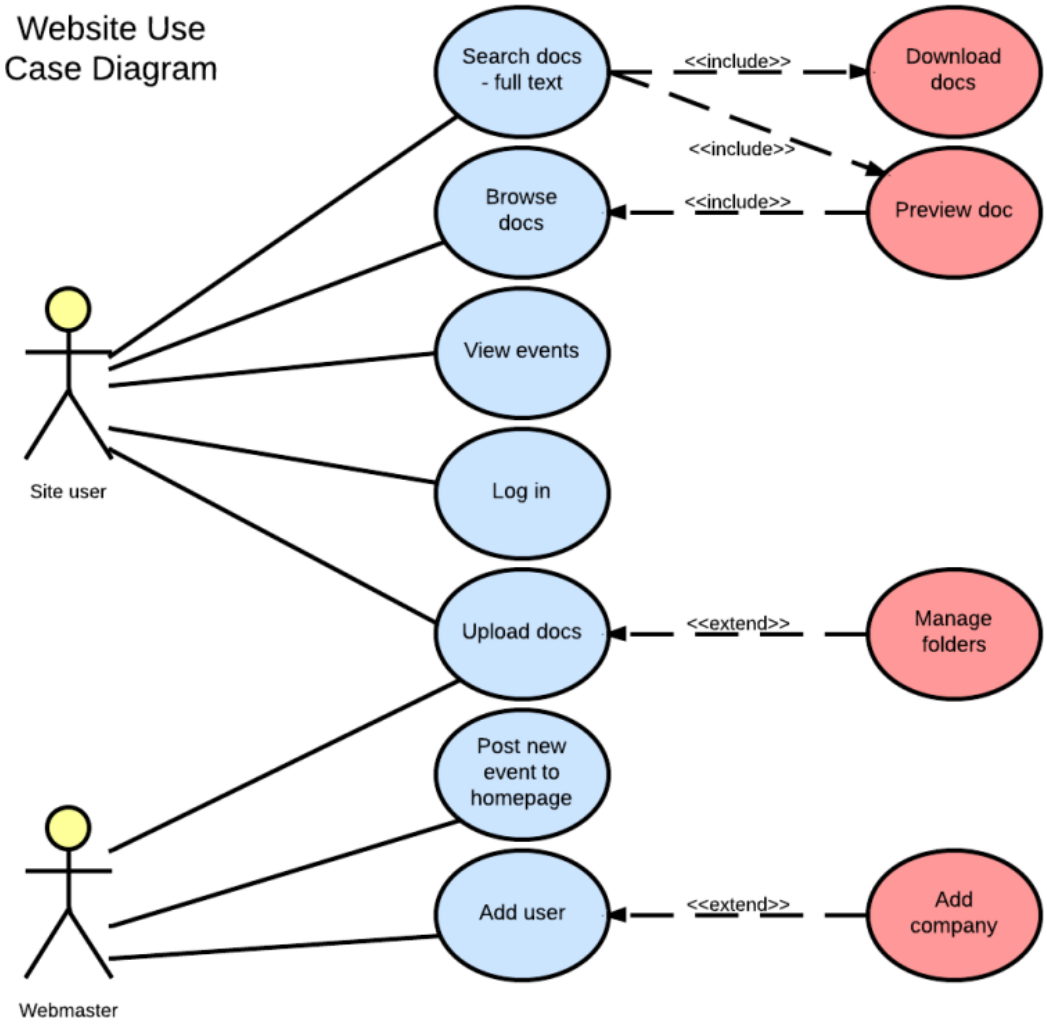
Task 1:

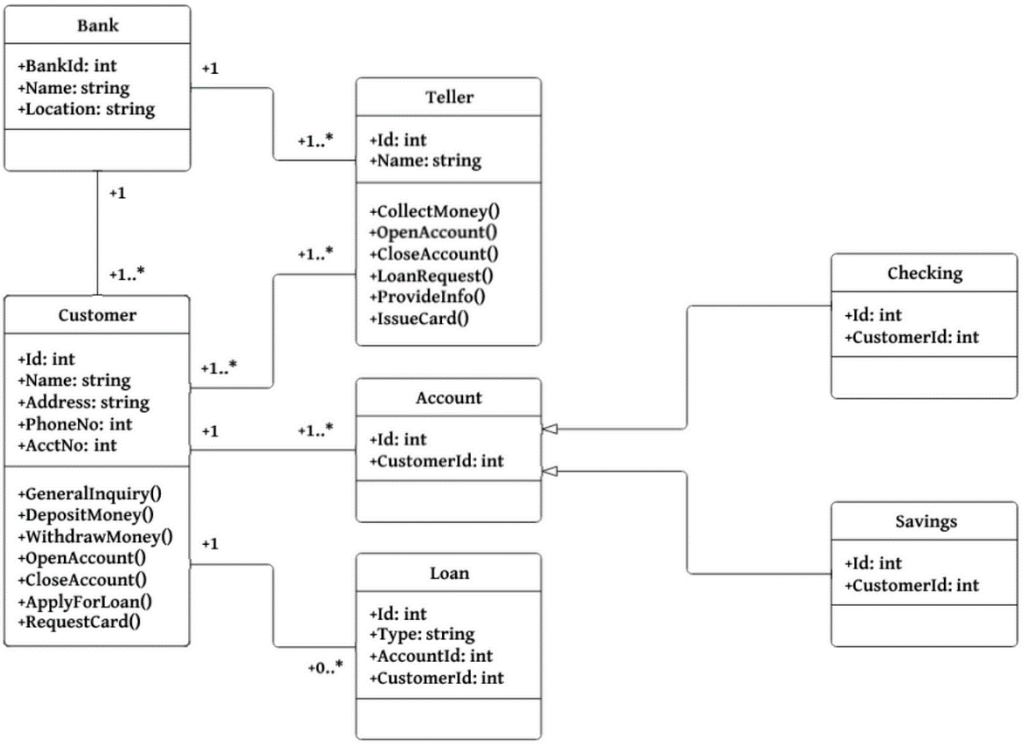
Make following screens in the Figma.

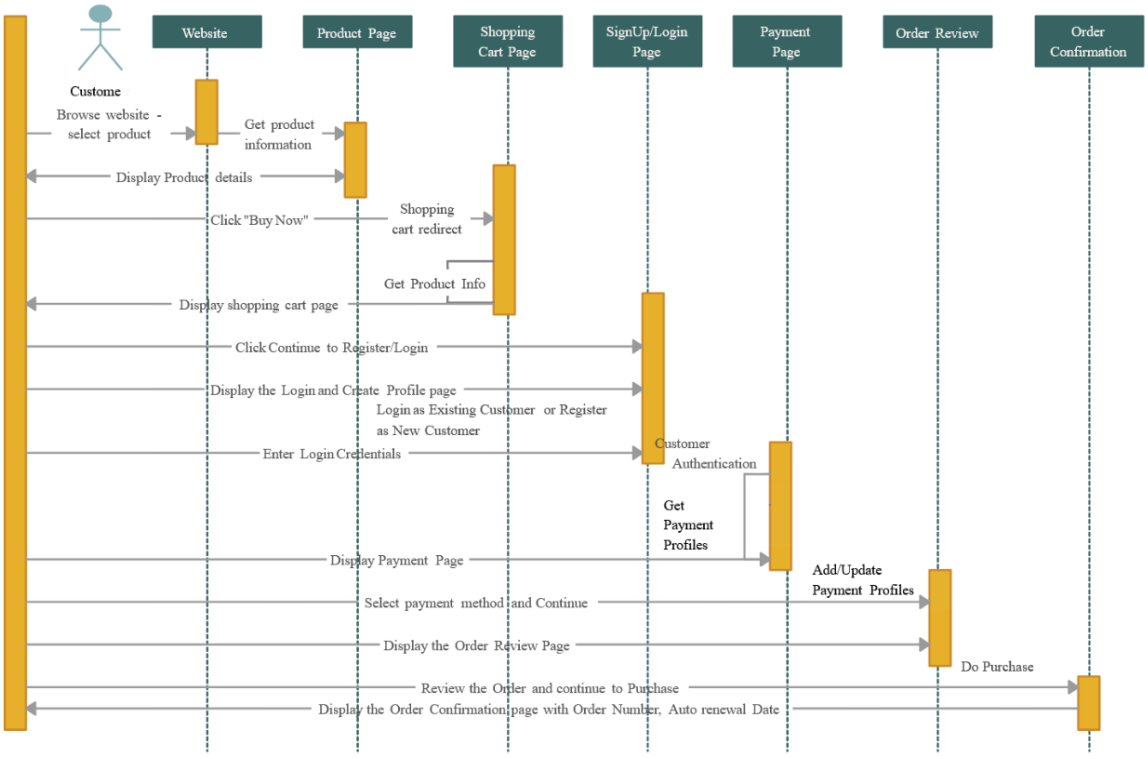


Task 2:

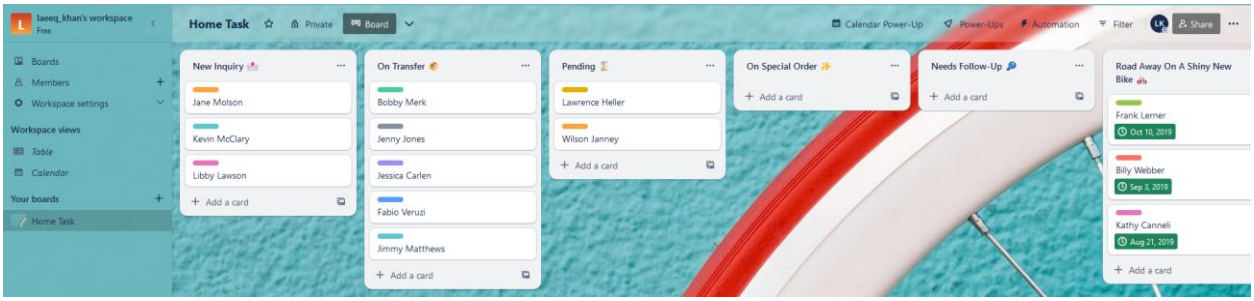
Draw following diagrams in STAR UML


Website Use Case Diagram
Search docs - full text
Download docs
<<include>>
<<include>>
Browse docs
<<include>>
Preview doc
View events
Site user
Log in
Upload docs
<<extend>>
Manage folders
Post new event to homepage
Add user
<<extend>>
Add company
Webmaster





Make following board in Trello.


laeeq_khan's workspace
Free
Boards
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
Home Task
Home Task
Private
Board
Calendar Power-Up
Power-Ups
Automation
Filter
Share
New Inquiry
Jane Molson
Kevin McClary
Libby Lawson
Add a card
On Transfer
Bobby Merk
Jenny Jones
Jessica Carlen
Fabio Veruzi
Jimmy Matthews
Add a card
Pending
Lawrence Heller
Wilson Janney
Add a card
On Special Order
Add a card
Needs Follow-Up
Add a card
Road Away On A Shiny New Bike
Frank Lerner
Oct 10, 2019
Billy Webber
Sep 3, 2019
Kathy Canneli
Aug 21, 2019
Add a card