Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Name: Apurba Koirala

Reg no: 22BCE3799

Subject Code: BCSE301P

Course Title: Software Engineering Lab

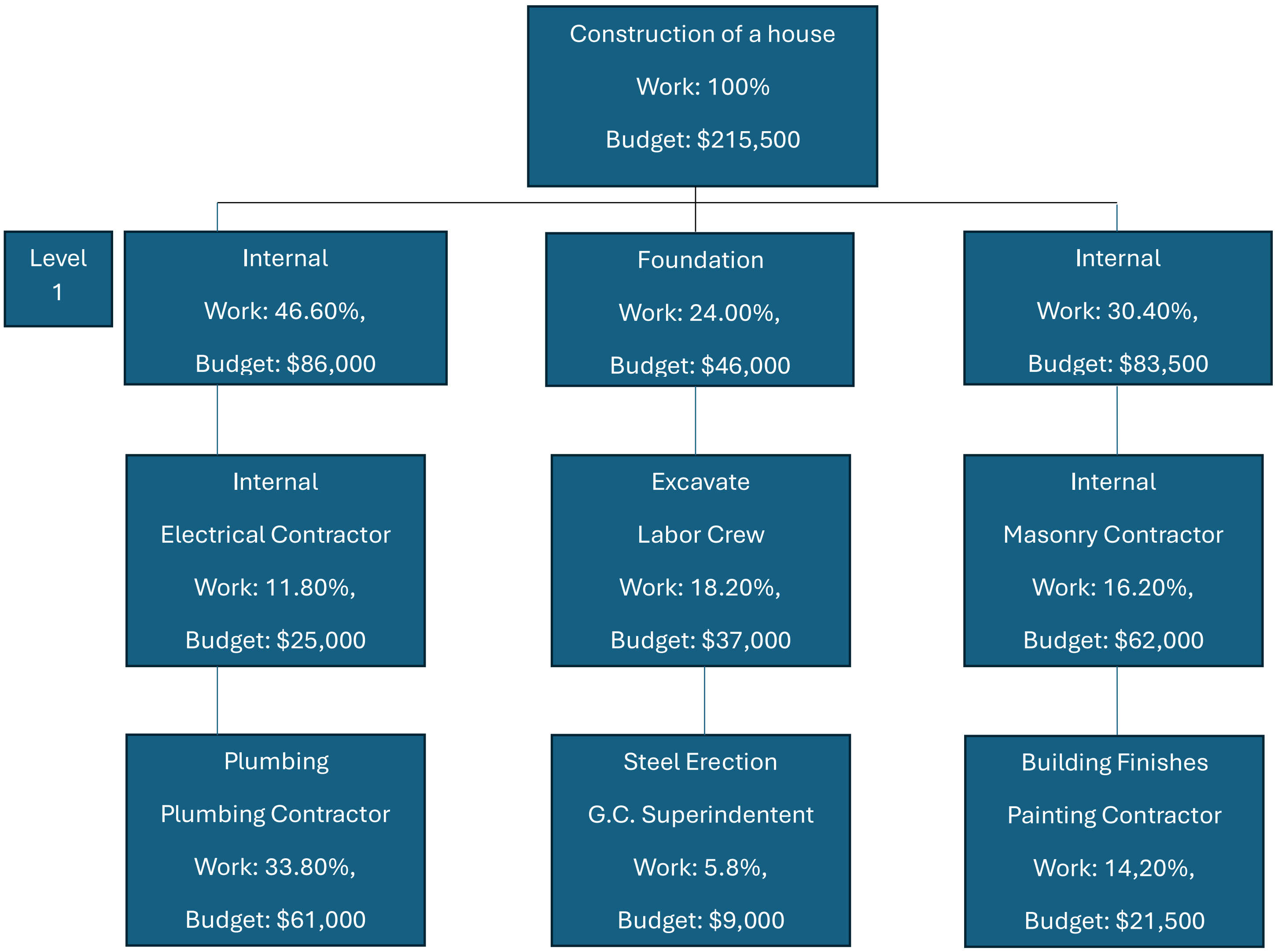
Lab Slot: L45 + L46

Guided by: Dr. PEREPI RAJARAJESWARI

Lab Assessment 2

1. Draw work breakdown structure for any real time application

Fig: WBS for construction of a house (Deliverables)



2. Describe storyboard for any Scenario

Storyboard: Bug Tracking & Fixing in a Software Development Team

It’s a busy morning, and a user tries to log into a company’s web application to check their dashboard. However, instead of accessing their account, they receive an error message: *“Login failed. Please try again.”* Frustrated, they attempt to reset their password, thinking it might be an issue on their end. But after multiple attempts, they still cannot log in. Realizing the issue might be with the application, they navigate to the app’s feedback section and submit a bug report detailing their problem. The report is automatically logged into the company's bug-tracking tool, such as **Jira** or **GitHub Issues**, and marked as a high-priority issue.

At the development team’s stand-up meeting, the project manager reviews the backlog and notices multiple users reporting the same login issue. Recognizing the urgency, they assign the bug to **Alex**, a backend developer, and tag it as a "critical" issue. Alex receives an automated Slack notification: *“New bug assigned: Login failure issue. Please investigate.”* Without delay, Alex opens the bug tracker, reads the description, and starts analyzing recent changes in the authentication system.

To debug the issue, Alex first attempts to reproduce the bug in a **local development environment**. He sets up test user credentials and tries logging in, but everything appears to work fine locally. Suspecting that the issue might be environment-specific, he checks the logs from the **production server**. The logs reveal a series of **authentication failures** due to a recent database migration that inadvertently altered user session handling. He realizes that a small but critical change in the database schema caused existing user sessions to become invalid, leading to failed logins.

With the root cause identified, Alex formulates a fix. He modifies the authentication logic to properly handle existing sessions and writes a migration script to update the affected database records. After implementing the fix, he pushes the changes to a **feature branch** and creates a pull request. The **CI/CD pipeline** automatically runs unit tests, integration tests, and security checks to ensure the fix doesn’t introduce new issues.

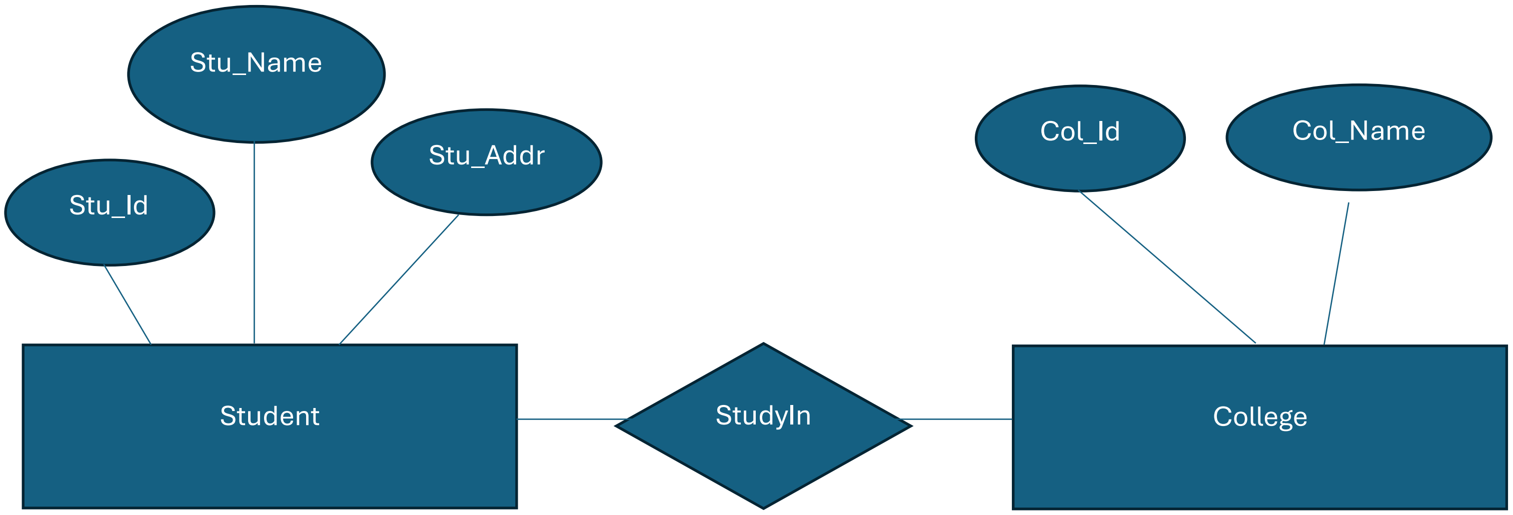
Once the tests pass, the pull request is reviewed by **Sophia**, another backend developer. She inspects the code, runs additional tests, and approves the changes. The fix is then merged into the **main branch**, triggering an automatic deployment to a **staging environment**. The QA team performs a final round of testing and confirms that the issue has been resolved. The fix is then deployed to **production**, restoring login functionality for all users.

Shortly after deployment, the support team sends out an email notification to affected users, informing them that the issue has been resolved. The original user who reported the bug receives the email and attempts to log in again. This time, they successfully access their account. Relieved, they continue using the app.

Meanwhile, Alex updates the bug tracker, marking the issue as **resolved**. The project manager reviews the resolution, closes the ticket, and thanks Alex for his quick response. In the next retrospective meeting, the team discusses the incident and decides to implement an additional **automated test case** to catch similar issues before deployment. The incident serves as a reminder of the importance of thorough testing and **continuous monitoring** in software development

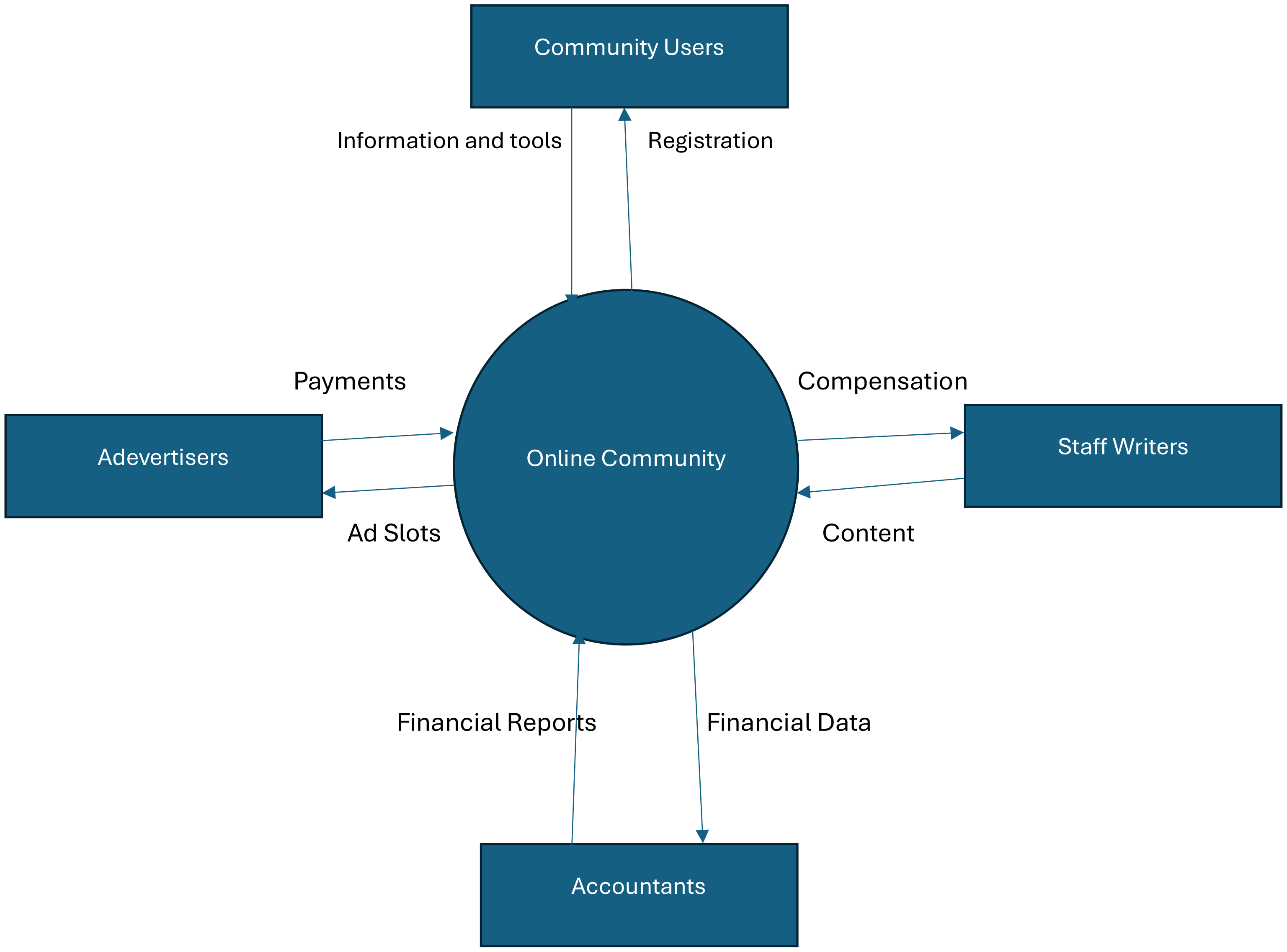
3. Requirement modelling using Entity Relationship Diagram (Structural Modeling)

Fig: ER diagram for student-college database.



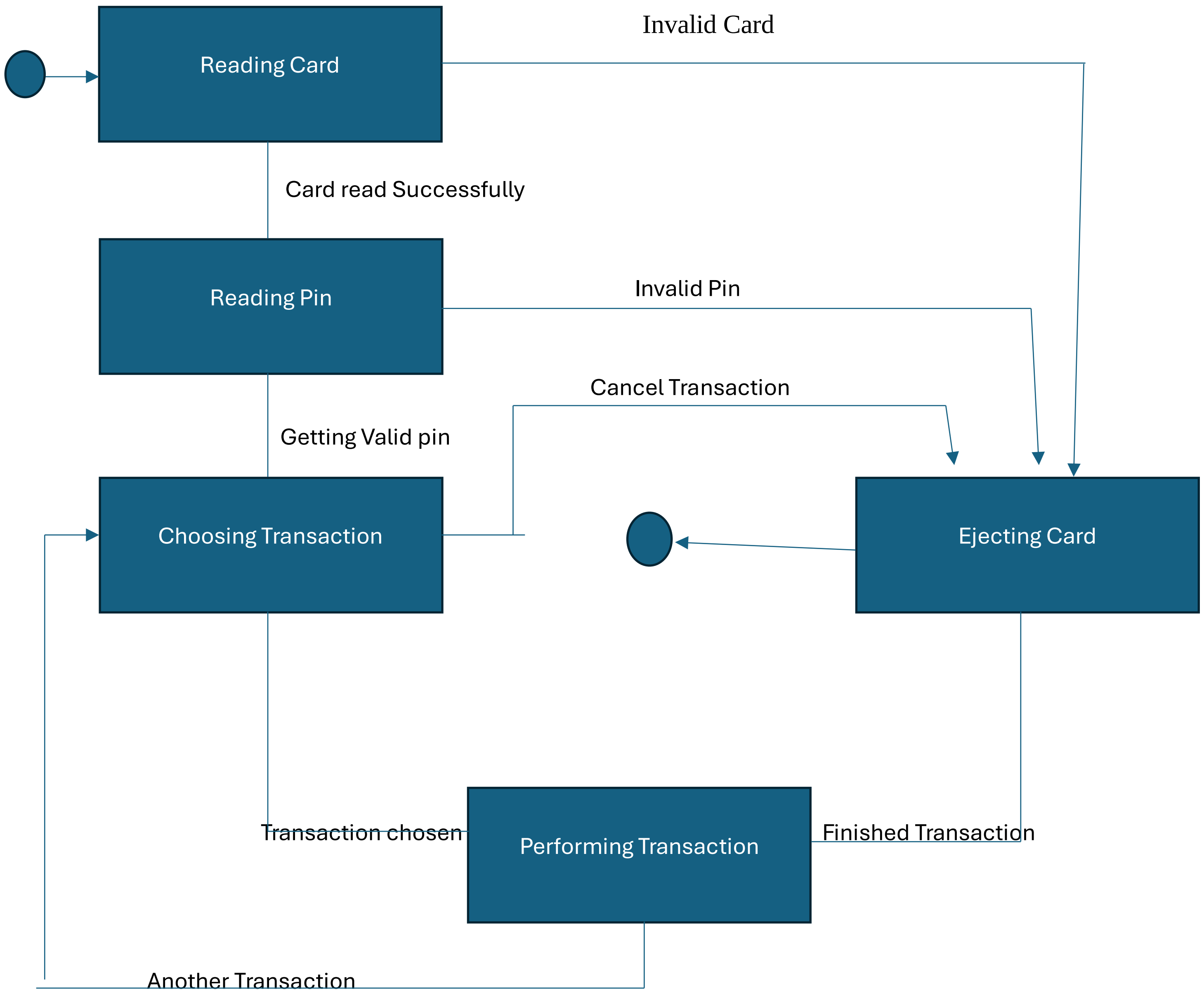
4. Requirement modelling using Context flow diagram, DFD (Functional Modeling)

Fig: Context Flow Diagram for an Online Community Platform.



5. Requirement modelling using State Transition Diagram (Behavioral Modeling)

Fig: State Transition Diagram for ATM system


Reading Card
Invalid Card
Card read Successfully
Reading Pin
Invalid Pin
Cancel Transaction
Getting Valid pin
Choosing Transaction
Ejecting Card
Transaction chosen
Performing Transaction
Finished Transaction
Another Transaction