Report *on*

AllyHub

Prepared for

Mr. Taslim Taher **Assistant Professor** **Department of CSE**	**Ms. Tasnuva Binte Rahman** **Lecturer** **Department of CSE**

Course No: CSE 3224
Course Name: Information System Design & Software Engineering Lab

Prepared by

Lab Section: A2

Group Name/No: A202

ID: 20210104032 Name: Afia Fahmida
ID: 20210104040 Name: Ashikul Islam
ID: 20210104047 Name: Zenun Chowdhury

Date: 29 May, 2024

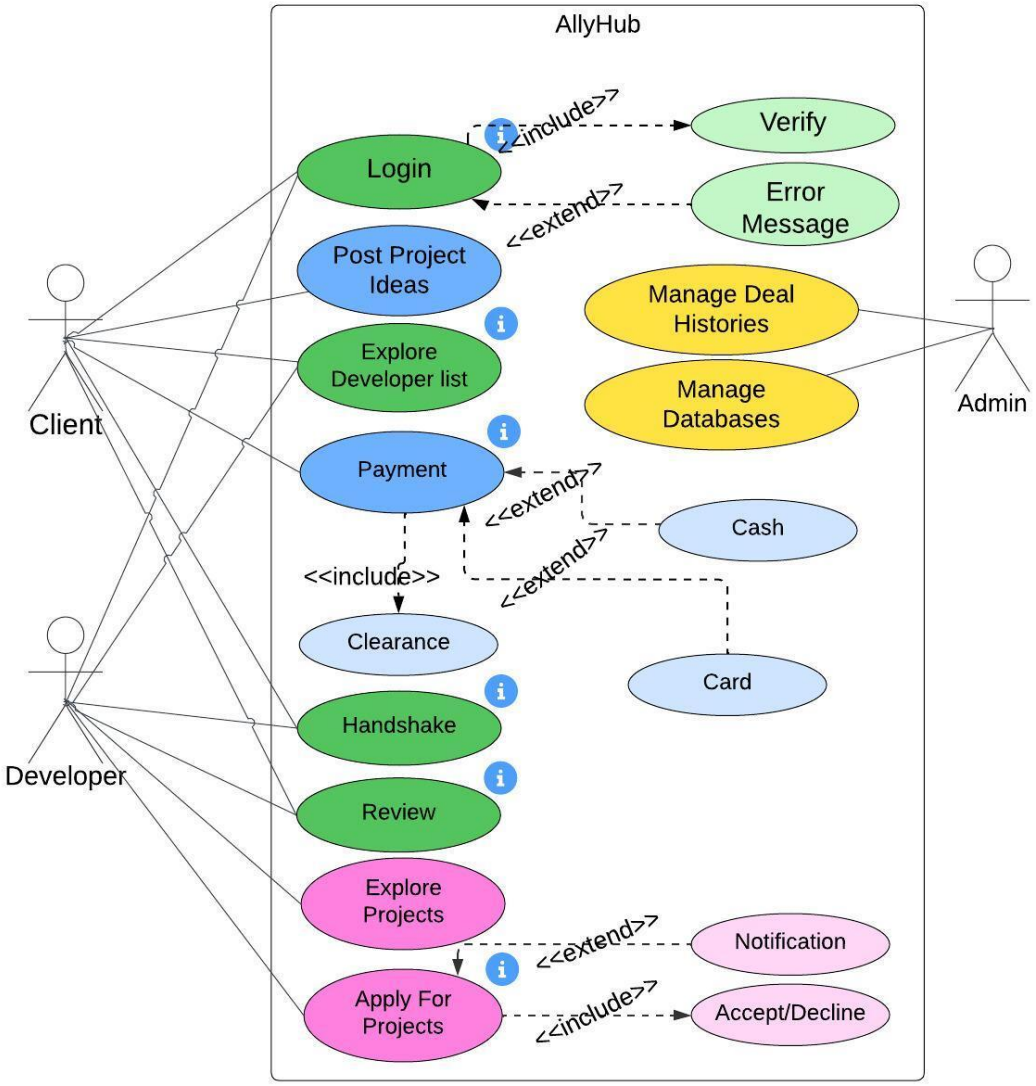
Department of Computer Science and Engineering

Actors

We have two types of actors in our project. They are:

- *Primary Actors*: Clients and Developers the two users are the prime actors of AllyHub. In our diagram below you will find them at the left side of the system.
- *Secondary Actors*: Admins are the secondary actors here. But earlier we have mentioned in our reports that we do not specifically have any admin panel. So the team members of our project are the potential admins who will handle the scenarios mentioned in our Use-Case diagram. In the diagram below, you will find the secondary users on the right side of the system.

Use-Case Diagram



Here, you will see the rectangular system in the middle where all the use cases can be found in it connected with their assigned actors. As mentioned earlier, primary

actors are on the left and secondary actors are on the right. The green use-cases have multiple dependencies. The blue ones are assigned to Clients and the pink ones are assigned for Developers. The yellow colored use cases are for the admins.

Use-Cases:

1. *Login*: A Client or a developer first need to sign up and login into their respective accounts.
    - Verify: This case includes a verification case too.
    - Error Message: If the procedure fails it will extend to provide an error message.
2. *Post Project Proposal*: A client can post project ideas and proposals in the post section that will be available in the UI of client's panel.
3. *Explore Developer List*: Clients and developers are allowed to explore and find developers according to their needs to assign and team up respectively.
4. *Payment*: Clients after making deals and finishing a project will be able to pay the salaries for the assigned developers whether using cash or cards.
    - Cash: The system includes pay in cash.
    - Card: The system also includes pay using cards.
    - Clearance: Once payment is done a clearance notification is extended.
5. *Handshake*: This one is for both developers and clients. It means making a deal on a particular project.
6. Review:
7. *Explore Projects*: Developers are allowed to explore projects proposed or posted by clients.
8. *Apply For Projects*: Only developers can apply to work on a project posted by the clients.
    - Accept/Decline: We have included accept or decline options.
    - Notification: A notification will be sent if the application is accepted or declined.
9. *Manage Deal Histories*: Admin as in our team will manage the deal histories from backend.
10. *Manage Databases*: Our team will also alter or update the database if needed.

Relations:

- Association (one actor and one-use case):
    1. Client->Post Project Ideas: Only clients can post projects.
    2. Client->Payment: only a client needs to pay the salaries of assigned developers.
    3. Developer->Explore Projects: Developers can explore the posted project ideas by clients.
    4. Developer->Apply For Projects: Only developers can apply for jobs.
    5. Admin->Manage Deal Histories: Only an admin as in team member who has access to backend codes can manage this part.
    6. Admin->Manage Databases: Only an admin as in team member who has access to database SQL queries can manage this part.

- Generalization (many actors and different use cases, but independent):
  1. Client->Developer->Login: Both the primary actors need to sign up and log into their accounts first.
  2. Client->Developer->Review: Both the primary actors can share their experiences in the review section.
  3. Client->Developer->Explore Developer list: Both clients and developers can search up Developers List.

- Dependency (multiple dependencies):
  1. Client->Handshake<-Developer: A client need to send handshake request so that a developer can handshake back. So they are dependent on each other here.

Description of Scenarios:

- *Login*:
  Actor: (Primary) Client, Developer.
  Description: The actors need to login into their individual accounts to get facilitated with the other features.
  Precondition: Both the actors need to create an account first and then login.
  Priority: High.
  Exception: Includes Verification and extends to give an error message if fails to Login.

- *Post Project Ideas*:
  Actor: (Primary) Client.
  Description: The actor can post project proposal for clarifications and judgements of the developers. These project ideas will go to the feeds of developers. Also this will create exceptionality of the project idea so others can know that an idea like this already on its way.
  Precondition: The actor need to login first because only a client can post not developer.
  Priority: Medium.
  Exception: None.

- *Explore Developers List*:
  Actor: (Primary) Client.
  Description: The actor can explore and search for their desired developer by filtering according to programming language or developing domain.
  Precondition: The actor need to login first.
  Priority: Medium.
  Exception: None.

- *Payment:*
  Actor: (Primary) Client.
  Description: After the project is done the clients are supposed to pay salaries to the developers who worked under them.

Precondition: The actor need login first because only a client need to pay but not developers. Also the client and developers need to be under deal previously and the developers are supposed to be done with their assigned tasks.
Priority: High.
Exception: Includes pay in cash or cards options and extends to give a clearance message once payment is done.

- *Handshake:*
  Actor: (Primary) Client, Developer.
  Description: A client can select their desired developer and send them Handshake request as a symbol of deal offer. Developers can handshake back which will be accounted as deal done.
  Precondition: The both actors need to do the act from being logged in their individual accounts for history capture.
  Priority: High
  Exception: None.

- *Review:*
  Actor: (Primary) Client, Developer
  Description: The developers and clients can share their experience of working on a project together on review section.
  Precondition: Both the actors need to be logged into their accounts.
  Priority: Low.
  Exception: None.

- *Explore Projects:*
  Actor: (Primary) Developer.
  Description: A developer can explore and search their preferred projects.
  Precondition: The actor need to stay logged into their own account and needs to be a developer.
  Priority: Low.
  Exception: None.

- *Apply For Projects:*
  Actor: (Primary) Developer.
  Description: A developer can apply for affiliations with a project to the clients. They can handshake once a client accepts the application and sends a handshake request.
  Precondition: The actor needs to be a developer and needs to stay logged into their individual account.
  Exception: Includes acceptance or declination and extends to give a notification message once accepted or declined.

- *Manage Deal Histories:*
  Actor: (Secondary) Admin.
  Description: As we do not have an admin panel our teammates will handle this scenario from the backend.

Precondition: The admin needs to be a team member who has access to our backend codes.
Priority: High.
Exception: None.

- *Manage Databases:*
  Actor: (Secondary) Admin.
  Description: As we do not have an admin panel our teammates will handle this scenario from the backend.
  Precondition: The admin needs to be a team member who has access to our database SQL queries.
  Priority: High.
  Exception: None.

Conclusion

We have emphasized the use-case diagram for AllyHub in this report. The main users (actors) are identified, the various tasks they carry out (use cases) are described, and scenarios detailing the system's operation under various circumstances are detailed in this diagram. Requirements are included in each scenario to guarantee seamless functionality. The use cases have been evaluated in order of usefulness to the project. All things considered, this use-case diagram gives both administrators and users alike a clear idea of how our system effectively supports interactive connection between Clients and Developers.