**GoLance**

**✅ Model Used:**

You’ll be using a **Client–Server Model** with a layered architecture:

**1. Presentation Layer (Frontend):**

Handles user interface using **HTML, CSS, JavaScript**. Users (students and admins) interact via web/mobile UI.

**2. Application Layer (Backend):**

Business logic built with **Java (Servlets / JSP / Spring)** handles task posting, bidding, credit transfers, and user authentication.

**3. Data Layer (Database):**

Stores all data using **SQL (MySQL/PostgreSQL)**—manages users, tasks, bids, reviews, credits, etc.

**🔧 System Model (High-Level)**

Your platform will follow a **Client-Server Model** with the following layers:

1. **Frontend (Client-Side)**
   * HTML, CSS, JavaScript (with optional Bootstrap, AJAX)
   * User Interfaces for students and admins
2. **Backend (Server-Side)**
   * Java (Servlets, JSP or Spring Boot)
   * Handles user sessions, task postings, bids, credit transfers, authentication
3. **Database (SQL-based)**
   * Stores users, tasks, bids, messages, credits, reviews, etc.
4. **Admin Panel**
   * Web-based interface to manage users, approve tasks, monitor disputes, etc.

**📦 Database Tables (with Primary Keys and Relations)**

**1. users**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| user\_id (PK) | INT (AUTO) | Unique ID |
| name | VARCHAR | Full name |
| email | VARCHAR | College-verified email |
| password\_hash | VARCHAR | Encrypted password |
| role | ENUM | requester, freelancer, admin |
| credits | INT | App credits |
| rating | FLOAT | Average user rating |
| registered\_on | TIMESTAMP | Registration timestamp |

**2. tasks**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| task\_id (PK) | INT (AUTO) | Unique task ID |
| user\_id (FK) | INT | Task creator |
| title | VARCHAR | Task title |
| description | TEXT | Task description |
| category | VARCHAR | Design, Writing, Coding, etc. |
| posted\_on | TIMESTAMP | Post timestamp |
| deadline | DATE | Requested deadline |
| status | ENUM | open, assigned, completed |
| assigned\_to (FK) | INT | Freelancer ID |
| credits\_offered | INT | Budgeted credits |

**3. bids**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| bid\_id (PK) | INT (AUTO) | Unique bid ID |
| task\_id (FK) | INT | Task reference |
| freelancer\_id (FK) | INT | User placing the bid |
| bid\_credits | INT | Credits requested for task |
| message | TEXT | Pitch or offer message |
| bid\_time | TIMESTAMP | Bid timestamp |

**4. messages**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| message\_id (PK) | INT (AUTO) | Unique message ID |
| sender\_id (FK) | INT | Sender user ID |
| receiver\_id (FK) | INT | Receiver user ID |
| message\_text | TEXT | Chat content |
| sent\_at | TIMESTAMP | Message time |

**5. task\_attachments**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| attachment\_id (PK) | INT (AUTO) | Unique file ID |
| task\_id (FK) | INT | Related task |
| file\_url | VARCHAR | File path or URL |
| uploaded\_by (FK) | INT | Uploader's user ID |
| uploaded\_at | TIMESTAMP | Upload timestamp |

**6. reviews**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| review\_id (PK) | INT (AUTO) | Unique review ID |
| task\_id (FK) | INT | Reviewed task |
| reviewer\_id (FK) | INT | User giving the review |
| reviewee\_id (FK) | INT | User being reviewed |
| rating | INT | Rating (e.g., 1-5) |
| comment | TEXT | Optional feedback |
| reviewed\_at | TIMESTAMP | Review timestamp |

**7. credit\_transactions**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| transaction\_id (PK) | INT (AUTO) | Unique ID |
| user\_id (FK) | INT | Affected user |
| amount | INT | Credits added/deducted |
| type | ENUM | earned, spent, withdrawn |
| task\_id (FK) | INT (nullable) | Linked task |
| timestamp | TIMESTAMP | When it happened |

**8. admin\_logs (optional for audits)**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| log\_id (PK) | INT (AUTO) | Unique log ID |
| admin\_id (FK) | INT | Admin performing action |
| action | TEXT | Description of action |
| target\_user\_id (FK) | INT (nullable) | Target user, if any |
| log\_time | TIMESTAMP | Log timestamp |

**✅ Relationships**

* One user can post many tasks
* One task can have many bids
* One task is assigned to **one** freelancer
* One task can have many task\_attachments
* One user can receive/send many messages
* One task results in one review for the freelancer

**🔄 Project Flow of GoLance – Campus-Focused Freelancing Platform**

**1. User Registration & Login**

* **Flow**:
  + User visits platform.
  + Registers using **college email ID**.
  + Receives OTP or verification link.
  + Logs in upon successful verification.
* **Tables Used**:
  + users – stores user credentials and profile details.
  + auth\_tokens – stores OTP or email verification tokens (optional).
  + user\_roles – differentiates between admin, task requester, and freelancer.

**2. Posting a Task (By Requester)**

* **Flow**:
  + Logged-in user fills task posting form (title, description, deadline, budget).
  + Task is saved as “Open” status.
  + Other users can now view and bid.
* **Tables Used**:
  + tasks – contains task details and status.
  + task\_categories – optional: categorize tasks (e.g., writing, design).

**3. Bidding on Task (By Freelancer)**

* **Flow**:
  + Logged-in freelancers view open tasks.
  + Places a bid with estimated time and proposed credits.
  + Task owner reviews all bids.
* **Tables Used**:
  + bids – stores bid amount, time estimate, freelancer ID, and task ID.

**4. Awarding Task**

* **Flow**:
  + Task requester selects a freelancer from bids.
  + Task status changes to “Assigned”.
  + Assigned freelancer gets notification.
* **Tables Used**:
  + tasks – task status is updated.
  + bids – selected bid is marked “Accepted”.

**5. Task Completion**

* **Flow**:
  + Freelancer uploads completed work.
  + Requester reviews and accepts/rejects.
  + Upon acceptance, credits are transferred.
* **Tables Used**:
  + submissions – holds uploaded files and submission notes.
  + credits\_transactions – logs the credit transfer from requester to freelancer.
  + task\_reviews – stores review and rating.

**6. Credit Management**

* **Flow**:
  + Users earn/spend credits through task transactions.
  + Admin can manually add/remove credits if needed.
  + Future phase: Withdrawal to real money.
* **Tables Used**:
  + user\_credits – balance of each user.
  + credits\_transactions – transfer logs.

**7. Admin Functions**

* **Flow**:
  + Admin logs into dashboard.
  + Manages users, tasks, complaints.
  + Views reports, disputes, manages credit flow.
* **Tables Used**:
  + admin\_logs – record of admin activities.
  + complaints – user-submitted issues.
  + users, tasks, bids, credits\_transactions – accessed for monitoring.

**📊 Summary of Tables Used per Phase**

| **Phase** | **Tables Used** |
| --- | --- |
| Registration & Login | users, auth\_tokens, user\_roles |
| Task Posting | tasks, task\_categories |
| Bidding | bids |
| Task Assignment | tasks, bids |
| Task Completion | submissions, credits\_transactions, task\_reviews |
| Credit Management | user\_credits, credits\_transactions |
| Admin Functions | admin\_logs, complaints, all primary tables for monitoring and control |