Field	Description
Use case name	Register Account
Scenario	Create a new profile for a Freelancer or Client.
Triggering event	A visitor (Student or Business Owner) wants to join Sho8la to post jobs or find work.
Brief description	The user provides their basic personal information (Name, Email, Phone Number), selects their role (Client or Freelancer), and sets a password to create a secure account.
Actors	Client, Freelancer (Student).
Related use cases	*Upload University ID* (Specific to Freelancer, usually follows this).
Stakeholders	**Admin:** Wants verified, real users to prevent spam. **Marketing:** Wants user contact info for engagement.
Preconditions	1. The user must have a valid email address and Egyptian phone number. 2. The user is not currently logged in.
Postconditions	1. A new User record is created in the database. 2. The user is assigned their selected role (Client/Freelancer). 3. The user is redirected to the Login page or Dashboard.
Flow of activities	**Actor (User)** 1. User clicks "Sign Up" on the homepage. 2. User enters full name, email address, and Egyptian phone number. 3. User selects their role: "I am a Student (Freelancer)" or "I am a Client". 4. User creates a password and confirms it. 5. User clicks "Create Account". **System** 1. System displays the registration form. 2. System validates the email format and checks if it already exists. 3. System validates the phone number format (must start with 010, 011, 012, 015). 4. System validates password strength (min 8 chars). 5. System creates the account in the database. 6. System displays a success message and redirects user.
Exception conditions	1. **Email already exists:** System shows error "Email already in use" and asks user to Login. 2. **Weak Password:** System prompts user to use a stronger password. 3. **Invalid Phone:** System rejects non-Egyptian numbers.

Field	Description
Use case name	Login
Scenario	Access the system using registered credentials.
Triggering event	User (Client, Freelancer, or Admin) opens the application and wants to access their personalized dashboard or perform restricted actions.
Brief description	The user authenticates their identity by entering their registered email and password. The system verifies the credentials and redirects them to the appropriate home page based on their role.
Actors	Client, Freelancer (Student), Admin.
Related use cases	*Register Account* (Must happen before Login).
Stakeholders	**Security Team:** Ensures unauthorized users cannot access private data. **Admin:** Needs to ensure banned users cannot log in.
Preconditions	1. The user must have a registered and active account. 2. The application is running and connected to the database.
Postconditions	1. The user is authenticated and a session token (e.g., JWT) is created. 2. The user is redirected to their specific dashboard (Client Dashboard, Freelancer Feed, or Admin Panel).
Flow of activities	**Actor (User)** 1. User clicks the "Login" button. 2. User enters their registered email address. 3. User enters their password. 4. User clicks "Sign In". **System** 1. System validates that inputs are not empty. 2. System hashes the entered password and compares it with the stored hash in the database. 3. System checks the user's role (Client vs. Freelancer vs. Admin). 4. System creates a secure session. 5. System redirects the user to the correct landing page.
Exception conditions	1. **Invalid Email/Password:** System displays "Incorrect credentials" error message. 2. **Account Suspended:** If the Admin has banned the user, the system denies access and shows "Your account has been suspended."

Field	Description
Use case name	Post New Job
Scenario	Client creates a new job listing for freelancers to bid on.
Triggering event	A Client has a specific task (e.g., "Build a website", "Design a logo") and needs to find a student developer.
Brief description	The Client fills out a job form specifying the title, detailed description, expected budget, deadline, and category. Once submitted, the job becomes visible to all verified Freelancers.
Actors	Client.
Related use cases	*Login* (Pre-condition), *Review Proposals* (Follow-up).
Stakeholders	**Freelancers:** Need clear requirements to submit accurate proposals. **Admin:** Needs to ensure no illegal or inappropriate content is posted.
Preconditions	1. The Client must be logged in. 2. The Client's account must be active (not suspended).
Postconditions	1. A new "Job" record is created in the database with status "Open". 2. The job appears in the "Browse Jobs" feed for Freelancers.
Flow of activities	**Actor (Client)** 1. Client navigates to the "Post Job" page. 2. Client enters a descriptive Title (e.g., "Fix Python Bug"). 3. Client writes a detailed Description of the requirements. 4. Client selects a Category (e.g., Web Development, Graphic Design). 5. Client sets a Budget Range (e.g., 500 - 1000 EGP) and Deadline. 6. Client clicks "Publish Job". **System** 1. System validates that all required fields are filled. 2. System checks that the budget is a valid positive number. 3. System saves the job details to the database. 4. System updates the public job feed. 5. System displays a success message "Job Posted Successfully".
Exception conditions	1. **Missing Information:** If the Client leaves the description blank, the system highlights the error. 2. **Inappropriate Content:** If the system detects banned keywords (e.g., "homework cheating"), it rejects the post and flags it for Admin review.

Field	Description
Use case name	Review Proposals
Scenario	Client evaluates bids submitted by freelancers for a specific job.
Triggering event	One or more Freelancers have submitted proposals to a job the Client posted, and the Client wants to select a candidate.
Brief description	The Client views a list of all proposals for their job. They can see the Freelancer's name, price quote, estimated timeline, and university verification status. The Client can then choose to accept a proposal or ignore it.
Actors	Client.
Related use cases	*Post New Job* (Predecessor), *Pay Deposit* (Successor).
Stakeholders	**Freelancers:** Waiting for a response to start work. **Platform Admin:** Wants matches to occur so transaction fees are generated.
Preconditions	1. The Client must have at least one active job posted. 2. At least one Freelancer must have submitted a proposal.
Postconditions	1. If accepted: The Proposal status updates to "Accepted" and the Client is taken to the Payment screen. 2. If rejected/ignored: The Proposal remains pending or is marked as "Rejected".
Flow of activities	**Actor (Client)** 1. Client navigates to their "My Jobs" dashboard. 2. Client clicks on a specific job title. 3. Client scrolls through the list of received proposals. 4. Client reviews the details (Price, Message, Student Profile). 5. Client clicks "Accept Proposal" on the chosen candidate. **System** 1. System retrieves all proposals linked to that Job ID from the database. 2. System displays proposals in a list. 3. When "Accept" is clicked, system locks the proposal. 4. System redirects the Client to the "Pay Deposit" page to fund the escrow.
Exception conditions	1. **No Proposals:** System displays "No proposals yet" message. 2. **Job Closed:** If the job was already cancelled or completed, the system prevents reviewing new proposals.

Field	Description
Use case name	Pay Deposit (Escrow)
Scenario	Client transfers funds to the secure system holding account to start the project.
Triggering event	Client has clicked "Accept Proposal" and must fund the project before the Freelancer can begin work.
Brief description	The Client is redirected to the payment gateway (Paymob integration). They choose a payment method (InstaPay, Wallet, or Card) and transfer the agreed amount. The system holds this money (Escrow) and notifies the Freelancer to start.
Actors	Client, External Payment Gateway (Paymob).
Related use cases	*Review Proposals* (Predecessor), *Process Payments* (System internal).
Stakeholders	**Freelancer:** Needs assurance that the money exists before working. **Sho8la (Platform):** Holds the money to ensure safety and takes a commission fee.
Preconditions	1. A proposal has been accepted by the Client. 2. The Client has sufficient funds in their external wallet/bank.
Postconditions	1. The transaction is recorded in the database. 2. The Job Status updates from "Open" to "In Progress". 3. The funds are added to the system's "Escrow Wallet" (not directly to the freelancer).
Flow of activities	**Actor (Client)** 1. Client reviews the total amount (Project Price + Service Fee). 2. Client selects payment method (e.g., "Mobile Wallet"). 3. Client enters their wallet number. 4. Client confirms the transaction via OTP or PIN (External step). **System** 1. System calculates the total fee. 2. System initiates a secure session with Paymob API. 3. System receives "Success" signal from Paymob. 4. System marks the money as "Held in Escrow". 5. System notifies the Freelancer: "Project Funded! You can start working now."
Exception conditions	1. **Payment Failed:** If Paymob returns "Insufficient Funds," the system shows an error and does not start the job. 2. **Timeout:** If the user takes too long, the secure session expires and must be restarted.

Field	Description
Use case name	Approve Work & Release Funds
Scenario	Client accepts the delivered work and completes the job.
Triggering event	The Freelancer has marked the job as "Delivered" and uploaded the files/links, and the Client has verified the quality.
Brief description	The Client reviews the final deliverables. If satisfied, they click the "Approve" button. This action triggers the system to move the funds from the "Escrow" holding account directly to the Freelancer's available balance.
Actors	Client.
Related use cases	*Pay Deposit* (Predecessor), *Submit Deliverables* (Pre-condition from Freelancer).
Stakeholders	**Freelancer:** Gets paid immediately upon approval. **Sho8la:** Records the completed transaction and marks the job as closed.
Preconditions	1. The job status must be "In Progress". 2. The Freelancer must have submitted work. 3. Funds must be currently held in Escrow.
Postconditions	1. Funds are transferred to Freelancer's wallet. 2. Job status updates to "Completed". 3. Client and Freelancer can now leave a review.
Flow of activities	**Actor (Client)** 1. Client receives notification "Work Delivered". 2. Client downloads files or checks the link provided. 3. Client verifies the work meets requirements. 4. Client clicks "Approve & Release Payment". 5. Client rates the freelancer (1-5 stars). **System** 1. System validates the approval action. 2. System executes the transfer: deducts from Escrow, adds to Freelancer's "Available Balance". 3. System closes the job. 4. System notifies Freelancer: "Payment Received".
Exception conditions	1. **Dispute:** If the work is bad, Client clicks "Request Revision" instead of "Approve." The funds remain in Escrow until the Freelancer updates the work.

Field	Description
Use case name	Chat with Client
Scenario	Real-time text communication between the two parties to discuss project details.
Triggering event	A Freelancer needs to ask a clarification question, or a Client wants to give specific instructions after a proposal is accepted.
Brief description	Provides a secure, internal messaging interface where the Client and Freelancer can exchange text messages and share links regarding the ongoing job. It ensures all communication stays on the platform for safety.
Actors	Client, Freelancer (Student).
Related use cases	*Submit Proposal* (Usually triggers the start of a conversation), *Approve Work* (Discussed in chat).
Stakeholders	**Admin:** Needs chat logs to resolve disputes if they happen. **Sho8la:** Wants to prevent users from sharing phone numbers to take the deal off-platform.
Preconditions	1. Both users must be logged in. 2. A Proposal must be active (submitted or accepted) linking these two users.
Postconditions	1. The message is stored in the database. 2. The recipient receives a notification (real-time).
Flow of activities	**Actor (Sender)** 1. User clicks the "Chat" icon on the Job or Proposal page. 2. User types a message in the text box. 3. User clicks "Send". **System** 1. System validates the message content (checks for banned words/spam). 2. System saves the message timestamp and content. 3. System pushes the message to the Recipient's screen instantly (via WebSockets). 4. System updates the "Last Active" status.
Exception conditions	1. **Off-Platform Detection:** If the system detects a phone number or email in the chat (e.g., "010xxxx"), it may block the message and show a warning: "Sharing contact info is forbidden." 2. **Connection Lost:** If the internet fails, the message shows a "Failed to Send" red icon.

Field	Description
Use case name	Upload University ID
Scenario	Student validates their identity to gain the "Verified Student" badge.
Triggering event	After registration, the Freelancer tries to access full features (like bidding), but the system restricts them until verification is submitted.
Brief description	The Freelancer takes a photo of their valid university ID card (e.g., Cairo University, FCI) and uploads it to the system. This allows the Admin to confirm they are a real student, building trust with clients.
Actors	Freelancer (Student).
Related use cases	*Register Account* (Predecessor), *Verify Student ID* (Successor - performed by Admin).
Stakeholders	**Clients:** Want assurance they are hiring educated students, not random scammers. **Admin:** Needs to maintain platform integrity.
Preconditions	1. The Freelancer must be logged in. 2. The Freelancer's account status is currently "Unverified".
Postconditions	1. The ID image is securely stored in the database/cloud storage. 2. The User's verification status changes to "Pending Approval". 3. The user sees a "Verification in Progress" banner.
Flow of activities	**Actor (Freelancer)** 1. User navigates to "Profile Settings" or clicks the "Verify Now" banner. 2. User selects "Upload ID". 3. User browses their device to select a photo (JPG/PNG) of the ID card. 4. User clicks "Submit for Review". **System** 1. System validates the file type (must be image) and size (max 5MB). 2. System uploads the file to the secure server. 3. System updates the database record to "Pending". 4. System displays a success message: "ID submitted. Review takes 24 hours."
Exception conditions	1. **File Too Large:** If the image is > 5MB, system shows "File too large, please compress it." 2. **Invalid Format:** If user tries to upload a PDF or Word doc, system rejects it: "Images only."

Field	Description
Use case name	Browse Jobs
Scenario	Freelancer searches for available projects that match their skills.
Triggering event	Freelancer logs in and wants to find work to earn money.
Brief description	The Freelancer views a feed of active job postings. They can filter these jobs by category (e.g., Web, Mobile, Design), budget range, or search by keywords to find the most suitable tasks.
Actors	Freelancer (Student).
Related use cases	*Post New Job* (The source of data), *Submit Proposal* (The next step).
Stakeholders	**Clients:** Want their jobs to be seen by the right talent. **Freelancers:** Need an easy way to find relevant work without wasting time.
Preconditions	1. The Freelancer must be logged in. 2. There must be active "Open" jobs in the database.
Postconditions	1. A list of job summaries is displayed to the user. 2. The user can click a job to view full details.
Flow of activities	**Actor (Freelancer)** 1. User clicks the "Find Work" or "Jobs" tab. 2. User optionally enters a keyword (e.g., "Python"). 3. User applies filters (e.g., Budget > 1000 EGP). 4. User scrolls through the results list. 5. User clicks on a specific job card. **System** 1. System queries the database for jobs with status "Open". 2. System filters results based on user input. 3. System displays the job cards (Title, Client Name, Budget, Time posted). 4. When a card is clicked, system loads the detailed view.
Exception conditions	1. **No Results:** If the search matches nothing, system displays "No jobs found. Try different keywords." 2. **Database Error:** If the job feed fails to load, system shows "Service unavailable, please refresh."

Field	Description
Use case name	Submit Proposal
Scenario	Freelancer applies for a specific job by offering a price and timeline.
Triggering event	Freelancer finds a job they like in the "Browse Jobs" list and decides they can complete it.
Brief description	The Freelancer fills out a proposal form, specifying their bid amount (Price), estimated delivery time, and a cover letter explaining why they are the best fit. Once submitted, the Client receives a notification.
Actors	Freelancer (Student).
Related use cases	*Browse Jobs* (Predecessor), *Chat with Client* (Follow-up if interested).
Stakeholders	**Client:** Receives offers to choose from. **Sho8la:** Ensures the bidding process is competitive and fair.
Preconditions	1. The Freelancer must be logged in. 2. The Freelancer must be "Verified" (ID approved) to bid on jobs above a certain budget (if applicable). 3. The Job must still be "Open".
Postconditions	1. A new Proposal record is created in the database. 2. The Client receives a notification: "New Proposal Received." 3. The Freelancer sees the job in their "My Proposals" list.
Flow of activities	**Actor (Freelancer)** 1. User clicks "Apply Now" on the job page. 2. User enters their Bid Amount (e.g., 600 EGP). 3. User specifies Delivery Time (e.g., 3 Days). 4. User writes a Cover Letter/Message. 5. User clicks "Submit Proposal". **System** 1. System validates inputs (Bid must be positive). 2. System checks if Freelancer has already applied to this job (prevents duplicates). 3. System saves the proposal to the database. 4. System increments the "Proposal Count" on the job listing. 5. System displays success message.
Exception conditions	1. **Already Applied:** If user tries to bid twice on the same job, system shows error "You have already submitted a proposal." 2. **Unverified:** If a non-verified student tries to bid on a high-value project, system prompts: "Please verify your ID first."

Field	Description
Use case name	Request Withdrawal
Scenario	Freelancer moves their earned money from the Sho8la wallet to their personal account.
Triggering event	Freelancer has completed jobs, accumulated a balance, and wants to cash out.
Brief description	The Freelancer views their "Available Balance" and requests a payout. They specify the amount and the destination (e.g., Vodafone Cash number or InstaPay address). The Admin then reviews and processes this request.
Actors	Freelancer (Student).
Related use cases	*Approve Work* (Source of funds), *Approve Payout* (Successor - performed by Admin).
Stakeholders	**Admin:** Needs to verify the request to prevent fraud. **Freelancer:** Wants quick access to their earnings.
Preconditions	1. Freelancer must have a positive "Available Balance". 2. Freelancer must not have any active disputes locking their funds.
Postconditions	1. A "Withdrawal Request" record is created with status "Pending". 2. The requested amount is deducted from "Available Balance" and moved to "Frozen/Pending Withdrawal".
Flow of activities	**Actor (Freelancer)** 1. User navigates to the "Wallet" page. 2. User clicks "Withdraw Funds". 3. User enters the Amount (e.g., 500 EGP). 4. User selects Method (Vodafone Cash / InstaPay). 5. User enters the destination number/address. 6. User clicks "Confirm Withdrawal". **System** 1. System checks if (Requested Amount <= Available Balance). 2. System validates the phone number/InstaPay format. 3. System creates the transaction record. 4. System updates the wallet UI to show the new pending state. 5. System notifies Admin: "New Payout Request".
Exception conditions	1. **Insufficient Funds:** If user requests 1000 EGP but has only 500 EGP, system shows error "Insufficient Balance". 2. **Invalid Number:** If the Vodafone Cash number is not 11 digits, system rejects the request.

Field	Description
Use case name	Verify Student ID
Scenario	Admin reviews and validates the university ID uploaded by a freelancer.
Triggering event	A Freelancer has uploaded their ID card, generating a "Pending Verification" request in the Admin panel.
Brief description	The Admin views the queue of pending verifications. They examine the uploaded image to ensure it is a valid, readable Egyptian university ID and matches the user's name. They then approve or reject the request.
Actors	Admin.
Related use cases	*Upload University ID* (Predecessor), *Login* (Precondition).
Stakeholders	**Sho8la:** Ensures only real students are on the platform. **Freelancer:** Needs the "Verified" badge to unlock full account features.
Preconditions	1. The Admin is logged in to the Admin Dashboard. 2. There is at least one "Pending" verification request in the database.
Postconditions	1. **If Approved:** User's status updates to "Verified" and they receive a badge. 2. **If Rejected:** User's status resets to "Unverified" and they receive a reason (e.g., "Blurry Image").
Flow of activities	**Actor (Admin)** 1. Admin clicks on the "Verification Requests" tab. 2. Admin selects a specific user from the list. 3. Admin views the uploaded ID image and compares the name with the profile name. 4. Admin clicks either "Approve" or "Reject". 5. (If Rejecting) Admin types a reason for the rejection. **System** 1. System retrieves the image from secure storage. 2. System updates the User table in the database. 3. System sends a notification to the Freelancer: "Congratulations, you are verified!" or "Verification Failed."
Exception conditions	1. **Image Load Error:** If the file is corrupted, Admin sees "Error loading image" and automatically rejects the request asking the user to re-upload. 2. **Duplicate ID:** If the Admin flags that this ID number is already in use by another account, the system bans both accounts for fraud.

Field	Description
Use case name	Approve Payout
Scenario	Admin authorizes the transfer of funds to a Freelancer's external wallet.
Triggering event	A "Pending Withdrawal" request appears in the Admin Dashboard after a Freelancer requests their money.
Brief description	The Admin reviews the pending financial request to ensuring there are no security flags on the account. They then approve the transaction, which triggers the actual money transfer (via Paymob/InstaPay integration or manual transfer) and marks the request as "Completed" in the system.
Actors	Admin.
Related use cases	*Request Withdrawal* (Predecessor), *Login* (Precondition).
Stakeholders	**Freelancer:** Waiting for their money. **Sho8la Finance:** Needs accurate records of money leaving the system.
Preconditions	1. Admin is logged in. 2. A Freelancer has submitted a valid withdrawal request. 3. The system's central bank account has sufficient funds.
Postconditions	1. Funds are transferred to the Freelancer. 2. The transaction status in the database changes from "Pending" to "Paid". 3. An email/SMS receipt is sent to the Freelancer.
Flow of activities	**Actor (Admin)** 1. Admin navigates to "Financial Requests". 2. Admin reviews the request details (User Name, Amount, Destination Number). 3. Admin checks for any fraud alerts. 4. Admin clicks "Approve Transfer". **System** 1. System locks the transaction to prevent double-payment. 2. System calls the Payout API (or records the manual transfer reference). 3. System deducts the amount from the platform's pending liability. 4. System notifies the Freelancer: "Your funds have been sent."
Exception conditions	1. **Suspected Fraud:** If the user has suspicious activity, Admin clicks "Reject" or "Hold". The funds are returned to the Freelancer's Sho8la wallet, and their account is flagged. 2. **Gateway Failure:** If the banking API is down, the system shows "Transfer Failed, try again later" and keeps the request in "Pending" status.