CIS 1512: Software Engineering

Gig App

(A Local Gig Jobs Platform)

Software Requirements Specification (SDD)

Prepared by: Gig App Team

Cordero Ebberhart
Lucas Huff
Melissa Wright-Uzoigwe
Sean Tucker
Tamara Nicholas

Professor: Hadi Nasser

Table of Contents

1. Introduction

This Software Design Document (SDD) defines the architecture and component-level design for the Gig App, a local gig jobs platform that connects clients (job posters) with workers (job seekers).

1.1 Purpose

This Software Design Document (SDD) defines the system architecture and component-level design for the Gig App, which connects clients (job posters) with independent workers (job seekers) for short-term local jobs.

This document translates the requirements defined in the Software Requirements Specification (SRS) and the Software Project Management Plan (SPMP) into a detailed, implementable design. It ensures technical feasibility, scalability, maintainability, and compliance with all Phase 1 objectives.

1.2 Scope

The Gig App enables users to register, authenticate, post, apply for gigs, communicate via notifications, and leave reviews. This is designed to be responsive, secure, and modular, suitable for deployment on cloud hosting platforms like Heroku or AWS.

1.3 References

- Pressman, R. S., Software Engineering: A Practitioner's Approach
  http://www.rspa.com/docs/Designspec.html
- Gig App – Software Requirements Specification (SRS)
- Gig App – Software Project Management Plan (SPMP)

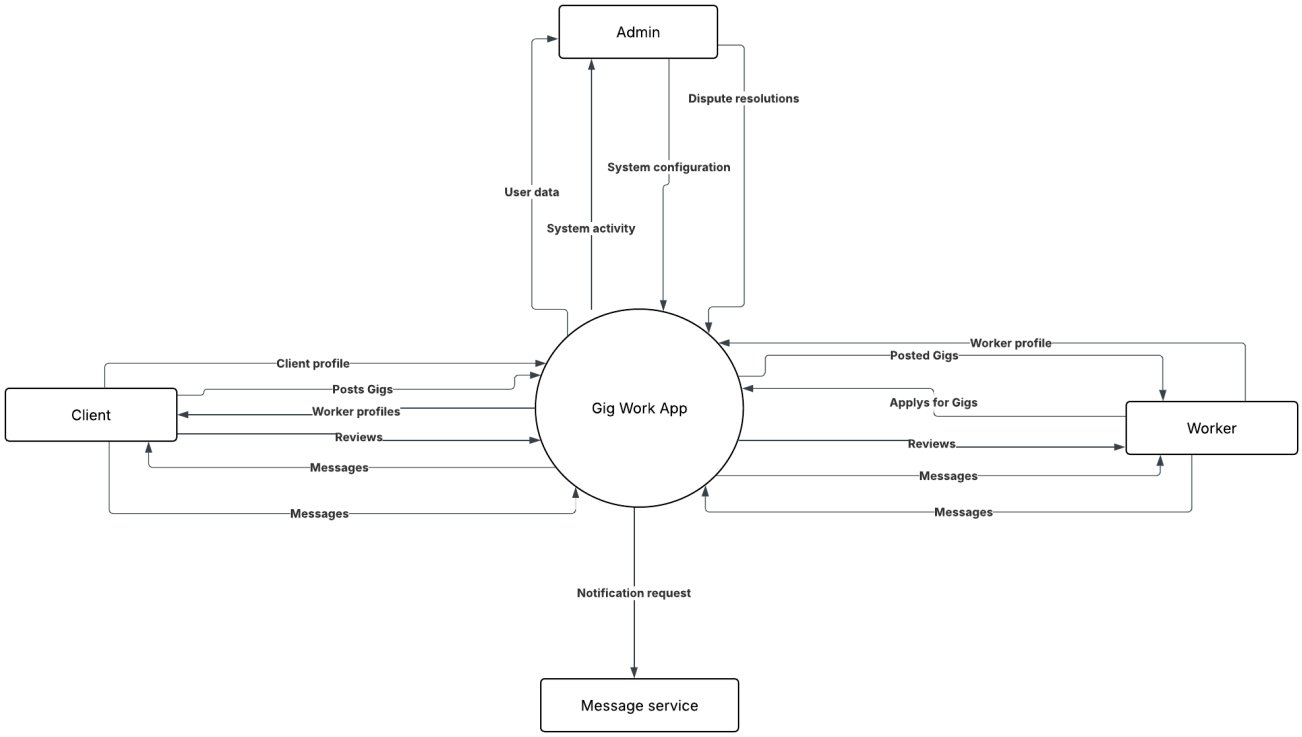
2. Data Design

2.1 Data Structures

The system uses relational data structures. Below are the core tables and brief descriptions.

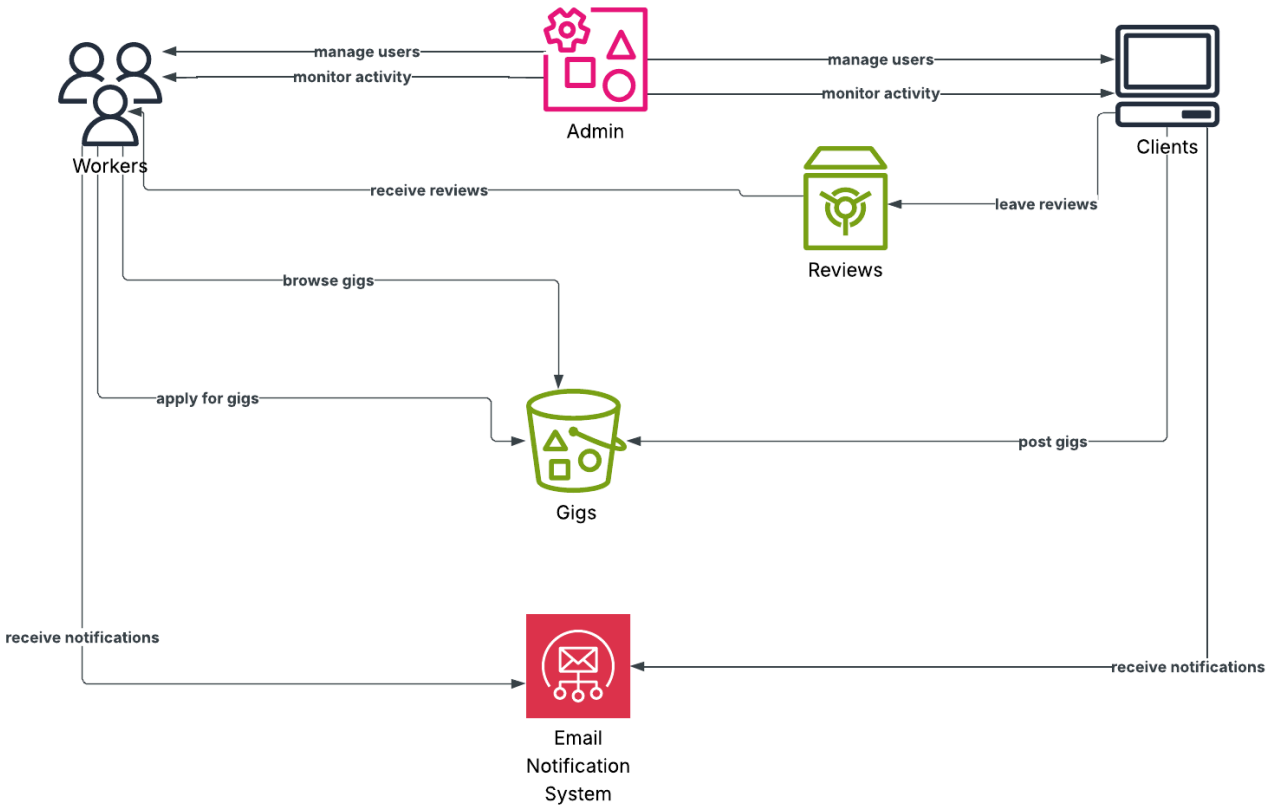
Table	Purpose / Key fields
Users	Stores user accounts: id, name, email, role (client/worker/admin), password_hash, created_at
Gigs	Gig postings: id, client_id, title, description, location, payment, created_at, status
Applications	Worker applications: id, gig_id, worker_id, message, status, created_at
Reviews	Client reviews: id, gig_id, reviewer_id, rating, text, created_at
Notifications	In-app/email notifications: id, user_id, type, payload, read_flag, created_at

2.2 Data Flow Diagram



Description: The diagram should depict major data flows between external entities (Client, Worker, Admin) and the central system.

2.3 Level 1 Data Flow Diagram



Description: Show decomposed workflows for Posting, Applying, Reviewing, and Notifying between system components.

3. Architectural and Component-Level Design

3.1 Architectural Style

The system adopts a three-tier client-server architecture with a RESTful API backend. Frontend: HTML/CSS/JS; Backend: Flask (Python) or Node.js (Express); Database: SQLite/MySQL.

3.2 Component Decomposition

Module	Description
Authentication Module	Handles registration, login, JWT-based sessions, and password encryption.
Gig Management Module	Manages CRUD operations for gig listings.
Application Module	Enables workers to apply to gigs and track status.
Review & Badge Module	Handles ratings and badge logic.
Notification Module	Manages message queues for in-app and email notifications.
Admin Module	Moderation and reporting tools.

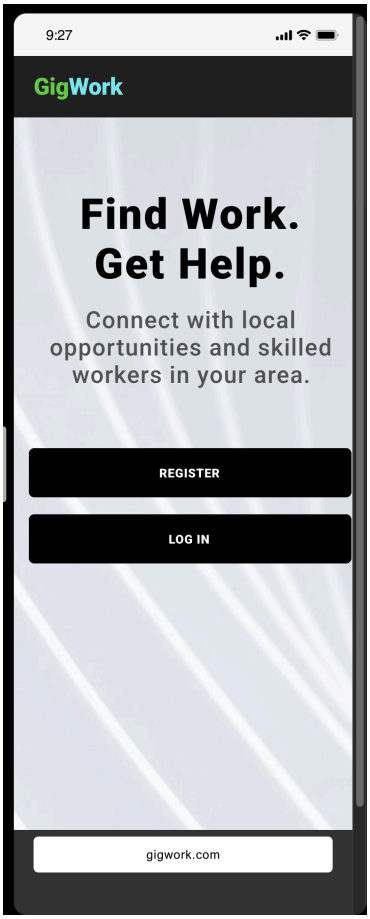
4. User Interface Design

4.1 UI Design Principles

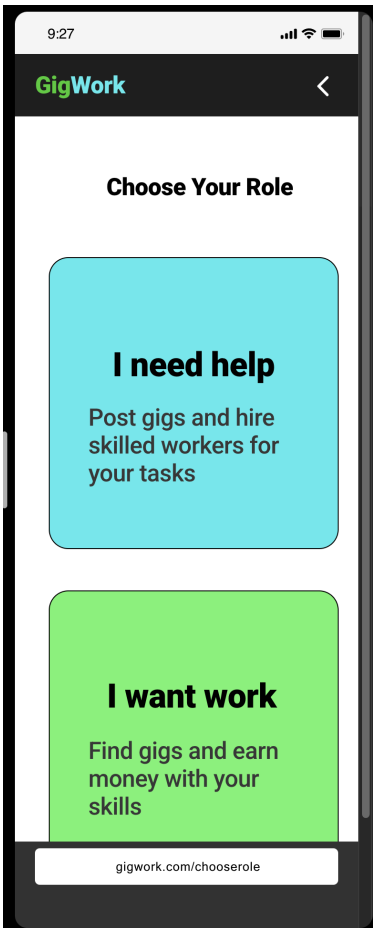
Focus on simplicity, consistency, responsiveness, and accessibility. Design should be mobile-first and intuitive for all user roles.

4.2 Key Screens and Navigation

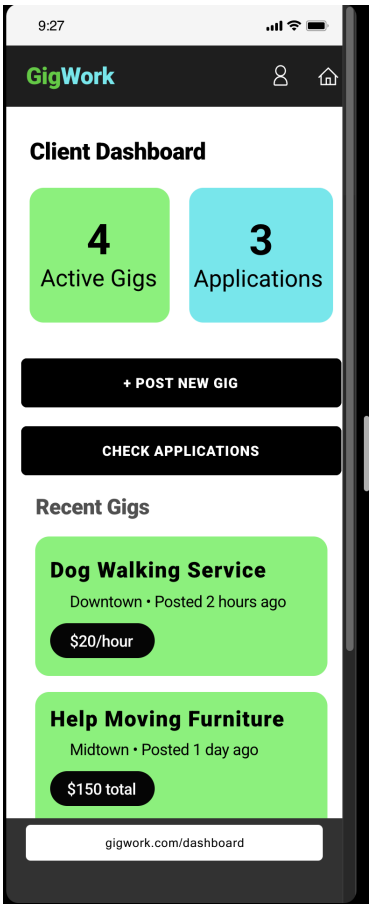
4.2.1 Splash Screen:

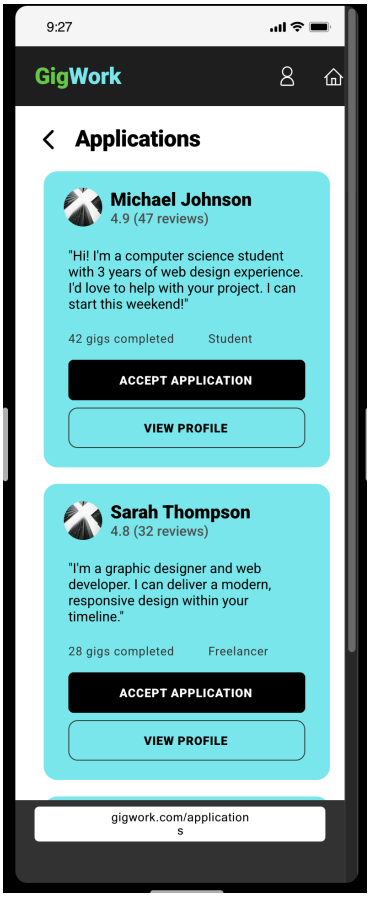


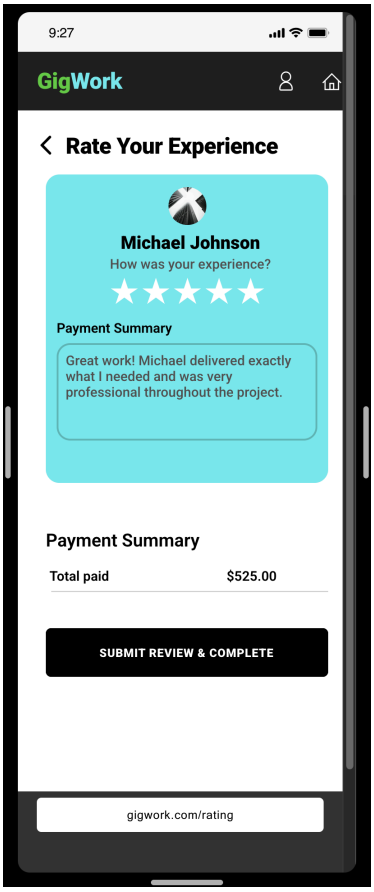
4.2.2 Role Selection Screen:



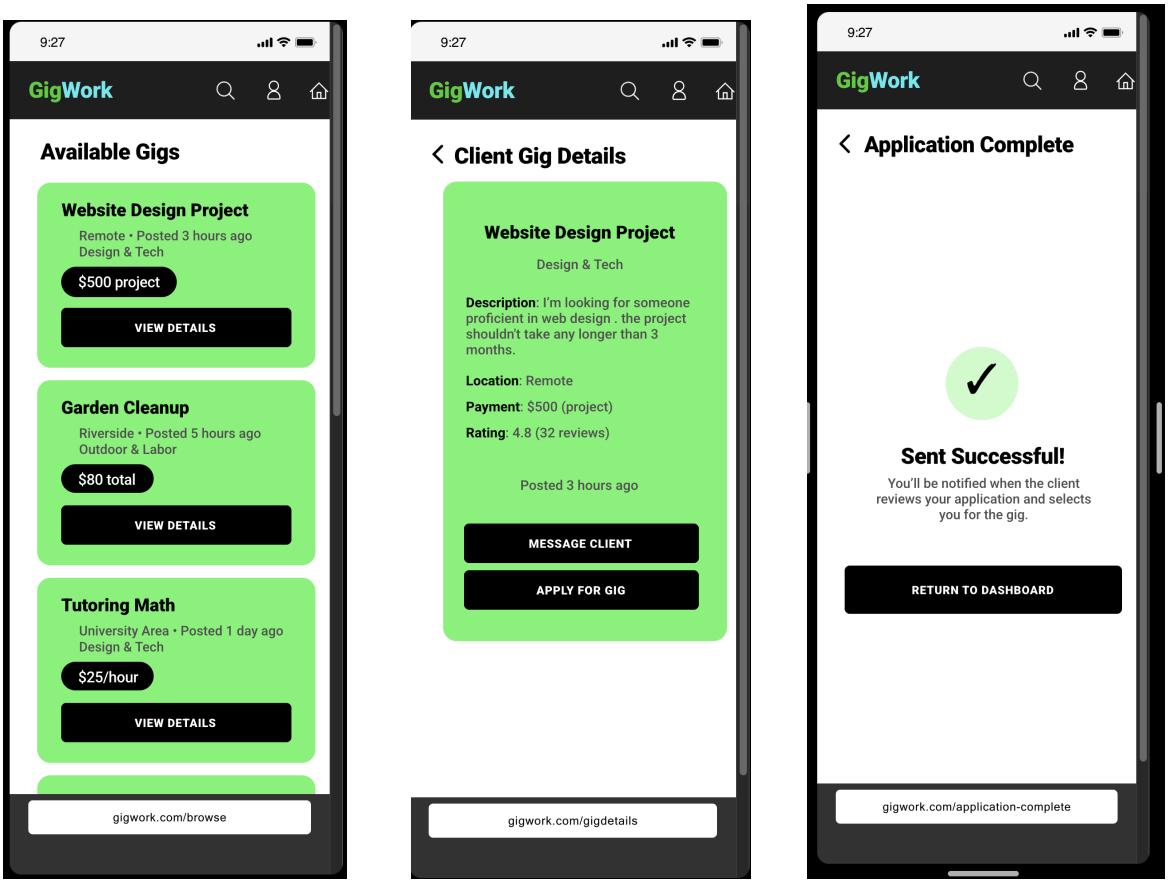
4.2.3 Client Dashboard & Application Acceptance:







4.2.4 Worker Dashboard & Gig Acceptance:



5. Design Constraints and Limitations

- Backend limited to Flask or Node.js.
- Relational Database required (SQLite/MySQL).
- HTTPS is mandatory for deployment.
- No payment gateway integration in Phase 2.
- Supported browsers: latest Chrome and Firefox.

6. Testing Considerations

6.1 Testing Types

Includes unit, integration, system, and acceptance testing.6.2 Example Test Cases

Test	Objective	Expected Result
Log in with valid credentials	Verify authentication	User receives a valid JWT token
Post a new gig	Validate gig creation	Gig saved and appears in the listing
Apply for a gig	Test application process	Application stored and client notified
Submit review	Confirm the review system	Review saved and visible on profile

7. Appendices

7.1 Packaging and Installation

Deployment target: Cloud hosting (Heroku/AWS). Installation Steps:

1. Clone the repository
2. Install dependencies
3. Set environment variables
4. Run migrations
5. Launch application

7.2 References

- Pressman Design Specification Template – http://www.rspa.com/docs/Designspec.html
- Gig App SRS & SPMP – Phase 1 Deliverables