Team 1 (Local Services Market)

1. **Skye Rogers**
    - **Student ID**: 027294330
    - **Email**: skye.rogers@student.csulb.edu
2. **John Le**
    - **Student ID**: 031543744
    - **Email**: john.le01@student.csulb.edu
3. **James Young**
    - **Student ID**: 028870086
    - **Email**: james.young01@student.csulb.edu
4. **Bryan Daroy**
    - **Student ID**: 030378086
    - **Email**: bryan.daroy01@student.csulb.edu
5. **Derin Le**
    - **Student ID**: 026994589
    - **Email**: derin.le@student.csulb.edu
6. **Daniel Winn**
    - **Student ID**: 026296931
    - **Email**: daniel.winn@csulb.edu

Preface

This document serves as a detailed guide for the development of the Local Services Market application which aims to connect skilled workers with employers. It is designed for developers, project managers, and any testers involved in the project.

Version	Date	Changes
1.0	11/18/24	Initial Version
1.1	11/20/24	Added Glossary
2.0	TBD	

Purpose

This document serves as a comprehensive guide for the development and comprehension of the software project entitled “Local Services Market.”

Audience

This document is intended for the following stakeholders: project managers, developers, testers, and all other individuals involved in the project lifecycle.

Introduction

The Local Services Market application aims to simplify the job hiring process by providing a platform on which employers can post job requests, and skilled workers can accept them. It has various features for the users such as profile management, job request tracking, and payment processing.

Project Overview

“Local Services Market” is a web platform for job matching and requests. Key features include profile creation, job management, and payment services.

Project Goals

- Enable seamless job matching
- Simplify payment processing
- Provide secure user authentication and data storage

Glossary

- **Employer:** A user posting job requests

- **Worker:** A user looking to accept job requests
- **Job Request:** A posted task or service from an employer

User Requirements and Use Cases

User Stories

1. As a worker, I want to search for jobs that are available so I can find employment.
2. As an employer, I'd like to post a job request so I can find workers that fit my needs.
3. As a user, I want a secure login to protect my data so that my information is secured and safe from hackers.
4. As a worker, I want to view the details of a job to decide if I should accept it.
5. As an employer, I need to approve of a completed job before payment is processed so that I pay for the work correctly.
6. As a worker, I want to track my past earnings and overall income so that I know how much I've gained.
7. As a worker, I want to update my profile info to reflect my improved skills so that it's more appealing to employers.
8. As an employer, I want to receive notifications for job acceptances so that I can hire workers.
9. As a worker, I want to receive notifications for new job requests so that I can apply to jobs quicker.
10. As a user, I want to reset my password so that I could access my account if I forget my password.

Use Case: Accepting a Job

Identifier	UC-3 Accepts Job
Purpose	Allow workers to accept job requests from employers
Requirements	User Story #1, 4

Development Risks	Medium risk, mismatched data
Pre-conditions	Worker has logged in and is browsing available jobs
Post-conditions	Job request is assigned to worker and removed from available jobs

Table 1: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Worker views job details	
2	Worker clicks accept	
3		System notifies employer about job acceptance
4		System receives the employer's agreement to hire the worker
5		System adds the job to the worker
6		Return control to worker

Table 2: Alternate Course of Action

Seq#	Actor's Action	System's Response
1	Worker views job details	
2	Worker clicks accept	
3		System notifies employer about job acceptance
4		System receives the employer's disagreement to hire the worker
5		Return control to worker
6	Worker reapplies to job	
7		System notifies employer about job acceptance
8		System receives the employer's agreement to hire the worker
9		System adds the job to the worker
10		Return control to worker

Table 3: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Worker views job details	
2	Worker clicks accept	
3		System notifies employer about job acceptance
4		System receives the employer's disagreement to hire the worker
5		Return control to worker

System Architecture

(Describe the high-level design of the software. List the components, name the architectural pattern you've used and provide a diagram.)

Components

1. **Frontend**: Flask / Python-based web interface
    - **Details:** Uses HTML/CSS, some Python and JS scripts to build the frontend

2. **Backend**: Flask for API Development
    - **Details:** Uses several tools and libraries; Flask, WTForms, SQLAlchemy, SQLite, Python
3. **Database**: SQLite for data storage.
    - **Details:** Using SQLite we store user data such as job postings, past earnings, etc.
4. **Authentication**: Flask-Login or Flask-Security, Twilio

Architectural Pattern

We used the Pipe-and-Filter pattern for our system. Below is a diagram that shows how we applied it to our system.