

**FIX FLEX**

**Chapter1: Introduction......................................................................................5**

**1.1 Overview………………………………................................................6**

**1.2 Objectives………………....................................................................7**

**1.3 Purpose....….....................................................................................8**

**1.4 Scope………………….......................................................................9**

**1.5 General Constraints …………………..............................................10**

**Chapter 2: Planning and Analysis..................................................................11**

**2.1 Project planning.............................................................................12**

**2.1.1 Feasibility............................................................................12**

**2.1.2 Estimated Cost....................................................................13**

**2.1.3 Gantt Chart..........................................................................13**

**2.2 Analysis and limitation of existing system......................................15**

**2.3 Need for new system......................................................................19**

**2.4 Analysis of the new system............................................................20**

**2.4.1 User Requirements..............................................................20**

**2.4.2 System Requirements..........................................................24**

**2.4.3 Domain Requirements..........................................................24**

**2.4.4 Functional Requirements.....................................................25**

**2.4.5 Non- Functional Requirements.............................................28**

**2.5 Advantages of the new system.......................................................30**

**2.6 Risk and Risk Managements..........................................................31**

**Chapter 3: System Design.............................................................................33**

**3.1 Design of database (ERD or Class) Diagram.................................34**

**3.2 Use case diagram...........................................................................36**

**3.3 sequence diagram..........................................................................39**

**3.4 activity diagram...............................................................................40**

**Chapter 4: System Implementation and Results.............................................41**

**4.1 Software Architecture.....................................................................42**

**4.2 Flowchart or Workflow..............................................45**

**Chapter 5: Testing……………………………………………………………….…52**

**5.1 Test Cases.....................................................................................53**

**5.2 Bug Reports .................................................................................54**

**Chapter 6: Result ………………...………………………………………………..56**

**6.1 Expected Results...........................................................................57**

**6.2 Actual Results................................................................................58**

**Chapter 1**

**Introduction**

* 1. **Overview**

What is FixFlex?

FixFlex is a platform that connects people who need tasks done with individuals who are willing to complete those tasks for pay.

It operates like traditional freelancing platforms such as Upwork, Fiverr, and Freelancer, where users can post tasks or jobs and taskers make offers on these tasks.

The user then selects the most suitable tasker for the job. However,

the main difference between fixing and traditional freelancing platforms lies in their primary objectives.

While platforms like Upwork primarily focus on digital services such as programming, writing, and design,

FixFlex's core mission is to facilitate the completion of a wide range of everyday tasks and services.

FixFlex specializes in connecting people who need tasks like (plumbing, electricity, painting, cleaning, moving, and more)

with handymen who are willing to perform these tasks for pay.

For task posters, creating a job listing is straightforward. Upload a detailed description of your task, specify the category it falls under, set your budget, and add any relevant images or documents. With a wide array of categories including home improvement, you're sure to find the right professional for your needs.

The Tasker can set offers that users can choose from

For freelancers and businesses, FixFlex offers a robust platform to showcase your skills. Create a compelling profile highlighting your expertise, experience, and previous work. Browse through a variety of tasks, submit quotes, and communicate directly with potential clients.

The platform emphasizes transparency and trust. All users can view and compare ratings and reviews, ensuring a reliable and quality experience. Payments are secure and straightforward, handled efficiently within the app.

FixFlex offers mobile apps for both iOS and Android devices, allowing users to access the platform conveniently from their smartphones or tablets. This enhances the accessibility of the platform, enabling users to post tasks, make offers, and communicate with each other on the go.

Whether you're looking to get a task done or to offer your skills, FixFlex is your go-to platform for seamless, efficient, and trustworthy service exchanges.

* 1. **Objectives**
* **Accessibility for All**: Fix Flex's primary goal is to be an inclusive platform, welcoming users from various walks of Eygpt. Whether you're a professional seeking more work, or an individual in need of assistance with a task, Fix Flex is designed to cater to your needs. We aim to bridge the gap between demand and supply in the service sector, ensuring easy access for everyone.
* **Support for Varied Needs**: Understanding that needs can range from simple household tasks to complex professional services, Fix Flex is committed to providing a comprehensive range of categories. This ensures that no matter what your requirements are, whether it is repair or refurbishment, you will find a skilled professional to help you.
* **Upholding Privacy and Confidentiality**: We place a high value on the privacy and confidentiality of our users. Personal information and transaction details are securely handled and protected, ensuring a safe and trustworthy environment.
* **Ensuring Quality and Reliability**: To maintain the highest standards, Fix Flex implements a robust system of ratings and reviews. This not only helps in building trust but also ensures that only qualified and reliable professionals are connected with those needing services.
* **Fostering a Community of Support:** By providing a platform where individuals can seek help and professionals can offer their services, Fix Flex aims to foster a sense of community and mutual support. We believe in creating a network where users can share their experiences, help each other grow, and contribute to a more connected and supportive soc

**1.3 Purpose**

**Diversity in services** At Fix Flex, it provides many services that the customer needs, with high quality, so that the customer can find what he needs in one place. **the speed** In Fix Flex, it works to quickly respond to the request when it occurs due to the speed of its repair and ensures its presence in all governorates in Egypt and its rapid arrival.

* 1. **Scope**
* **Broad Geographic Reach**: Fix Flex aims to start by assisting people across various locations and regions . The project's primary focus is on inclusivity, reaching out to individuals in need across different areas and governorates.
* **Verification and Transparency**: ensure the authenticity of the cases and build trust with donors, the platform will support campaigns with photos . This approach is intended to provide donors with confidence and assurance that their contributions are reaching the intended recipients.
* **Open Platform :** Fix Flex will be an open platform where any visitor can submit a case that requires assistance. The platform will then display selected details about these cases
  1. **General Constraint**

Understanding the limitations and constraints of a project is crucial for its successful execution and management. Here are the general constraints for the Fix Flex project:

* **Time Constraint**: The project is subject to a specific timeline, which includes deadlines for each phase and the final rollout date. Adhering to this schedule is vital to ensure timely completion and launch. The time constraint encompasses not only the duration of tasks but also the allocation of resources and coordination among different project components.
* **Scope Constraint**: This defines the boundaries of the project, including its goals, deliverables, features, and functions. The scope outlines the tasks required to complete the project and sets the parameters within which the project must be developed. It is essential to maintain a clear and focused scope to avoid scope creep, which can lead to delays and additional costs.
* **Cost Constraint:** The budget of the project is a critical constraint, encompassing all financial resources required for timely and effective completion within the defined scope. However, the cost constraint is not limited to monetary expenditure alone; it also includes resource allocation, such as time, labor, and material resources. Effective cost management is crucial to ensure that the project does not exceed its financial limitations.
* **Collecting Raw Data**: Gathering the necessary data for the project poses its own set of challenges. These include:

**-** **Concept of Data Collection**: Understanding what data is needed, why it is needed, and how it will be used is fundamental to the data collection process.

**-** **Types of Data**: Identifying the types of data and their sources is crucial for informed decision-making and project planning.

**-** **Issues to be Considered for Data Collection**: Addressing potential issues such as data privacy, accuracy, and relevance is essential for effective data collection.

**-** **Methods of Primary Data Collection**: Determining the most suitable methods for gathering primary data is vital for obtaining accurate and useful information.

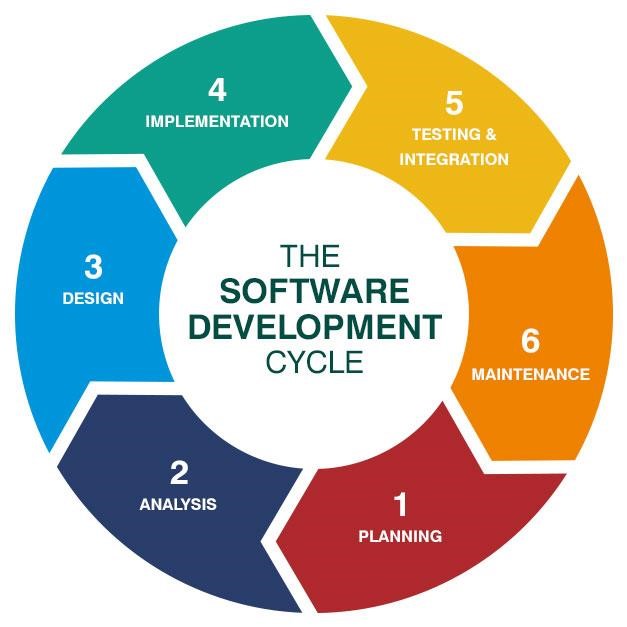
By acknowledging and managing these constraints, the Fix Flex project can be effectively guided towards successful completion, ensuring that it meets its intended goals within the set time frame, scope, and budget, and with the necessary data in hand.

**Chapter 2**

**Planning and Analysis**

# **Development approach and methodology**

## System Development Life Cycle

We followed system development life cycle approach in developing this project with its 7 phases, planning, analysis, design, development, testing, deployment and maintenance.

## Methodology

We followed Agile analysis and design methodology in developing, this project and developed all related UML diagrams, represented in this document later.

**Software development phases:**

**2.4 Analysis of the new system**

* + 1. **User requirements**
* This explains what the user will get through the site or application that

you are using or the user what will it does inside the system?

* Often referred to as user needs, describe what the user does with the

system, such as activities which users should be able to do. User

requirements are documented in the User Requirements

Document (URD) using narrative text. User requirements are signed in Generic by the user and used as the primary input for creating system requirements.

**Our FixFlex will consist of the following users:**

1. **- The User:**

* He can create an account for the platform by registering his basic data (first name, last name, email, password, mobile number, address).
* He can then log in (email, password).
* upload his own profile image.
* update his own profile details.
* Users can post tasks they need help with, providing details (title, category, details, location, date, time, budget).
* The user receives offers from the Tasker and can accept or reject it.
* Users can track the status of their posted tasks and receive notifications on updates.
* Users and taskers can communicate in real-time through the built-in chat system.
* Users can rate taskers and leave reviews based on their experience.
* Users can payment cash or online payment.

1.1-**Tasker**:

It can do everything that the user does in addition to:

* Uploading the type of work he offers, phone number and location to profile.

* Can browse available tasks and choose the ones they want to complete.
* Make an offer to user.
* Send notification to user.

**3-Admin:**

* Delete task.
* Update task.

**2.4.2 System Requirements:**

This is considered the answer to the question - How will this system build? What software, hardware and network which will you work on?

* I mean, what is the software that I use for my son to appreciate this system?
* My data base is where I use it, for example, or where is it?
* What programming languages are used in my system?

**2.4.3 Domain Requirements / Business Requirements**

This is what the business needs and will benefit from by making this application or this web site

* What benefited from the work of this system in terms of the system, I mean, I am now an example of the worker in a system for selling cars and to be able to achieve a certain goal, and this means.
* They agreed to integrate the objectives of the project with the objectives of the stakeholder.
* Ensure that the BRD achieves a very wide and measurable set of process goals and that they meet customer requirements It will be adjustable and changeable Like you can write the BRD:

You write "the name of the project "and then the goal you want to achieve" and then write "Why do you want to achieve this goal."

The “**FixFlex**” platform is based on connects people looking for task services with skilled task runners in their area. Whether you need help with household chores, errands, or any other tasks.

* + 1. **Functional Requirements**

**1- Registration:**

•The User must log in first:

A- The user logs in by (first name, last name, email, password, mobile number, address).

B-The Data will be saved in the database.

C- The user becomes registered in the system.

D-Who uses this:

- Users who are visiting the site for the first time.

**2-Login User:**

A-The user tries to enter the site by Email and Password if the data is correct, if the data is incorrect there appears to him error message.

B- Now the user can use the site.

C-Every User can edit his profile.

D-Upload profile image

**3-Login Admin:**

A- Log in to the system by email and password.

B-Can Update his profile.

C-Can delete any user or tasker.

D- Can See all the details that are specific to a particular post.

E- Can See all requests and can delete them.

**5-Task Posting:**

1. They log in or sign up if they haven't already.
2. After successfully login or sign up, they click on "Post a Task".
3. They describe the task they need help with.
4. They set a budget for the task.
5. They select the category of the task (e.g., cleaning, gardening, IT support).
6. They add additional task details such as location and due date.
7. They preview the task details to ensure accuracy.
8. If the task details are correct, they post the task.

**6-Browse Task:**

1. They browse available tasks on the platform.
2. If they find a task of interest, they view the task details.
3. If they're interested in the task, they make an offer to the task poster.
4. If the offer is accepted, they proceed to complete the task.
5. After completing the task, they request payment from the task poster.
6. If payment is received, the task is considered completed successfully.
7. If payment is not received, they follow up for payment.

**7-Real-time Chat:**

1. After the user and tasker logs in to the system.
2. The user can communicate with the tasker in real-time through the built-in chat system.
3. Who uses this:

- Any user registered in the system wants to talk to Tasker

**8-Task Status Tracking:**

1. After the user logs in to the system and posted task.
2. Users can track the status of their posted tasks and receive notifications on updates.
3. Who uses this:

- The user who posted the post

**9-Ratings and Review**

A. After the user logs in to the system and posted task

and carry out his mission.

B. Users can rate task runners and leave reviews based

on their experience

C. Who uses this:

- Any user to whom Tasker provided a service

**2.4.5 Non- Functional Requirements**

**1- Performance Requirements:**

1-The system supports the use of multiple users at the same time.

2- The database should be normalized to prevent redundant data and

improve the performance.

3- The database should be distributed to prevent outages.

4- We will make database backup several times.

**2- Safety Requirements:**

The database may crash at any certain time due to virus or operating

system failure. Therefore, it is required to take the database backup.

**3- Security Requirements:**

1- The system should be completely Consistent and Secure.

2- Keep specific history data sets.

3- Communication needs to be restricted when the application is validating

the user.

### 4-Usability:

### 1-user can achieve their goal easily.

### 2-Speed of performing tasks for the user.

### 3-Ease and simplicity of design

**2.5 Advantages of the new application**

* First, Fix Flex aims to bring together home repair services in one easy-to-access place
* The system will not focus on one service, but will focus on more than one service it provides

**2.6 Risk and Risk Managements**

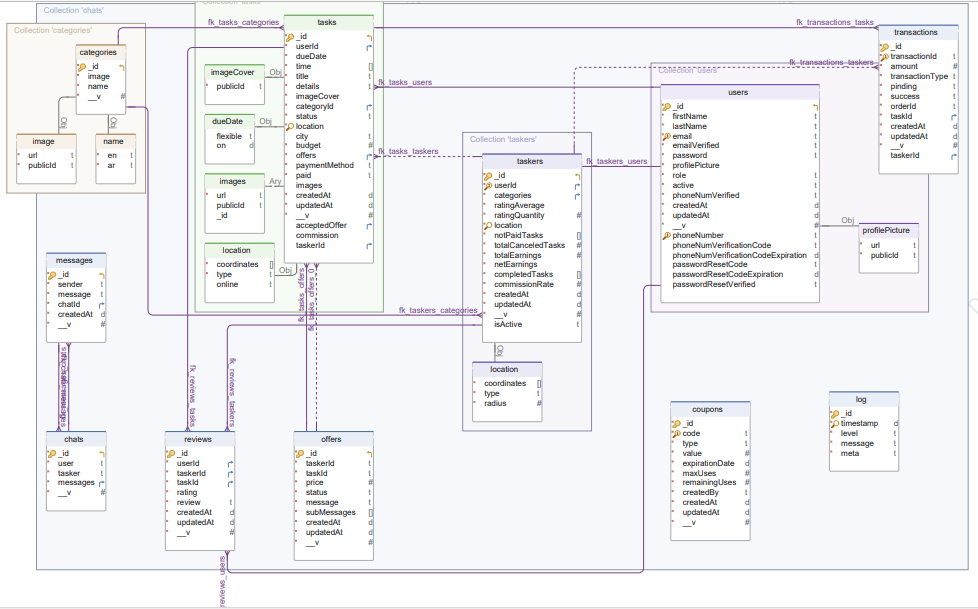
**Risks:**

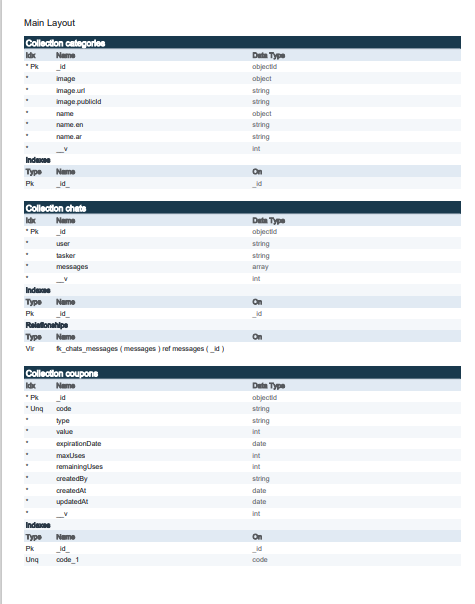
1. Not taking trust from customers
2. Great responsibility due to providing many services

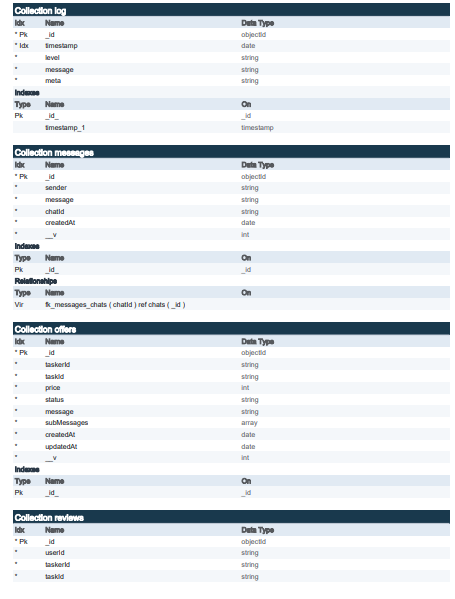
**Chapter3**

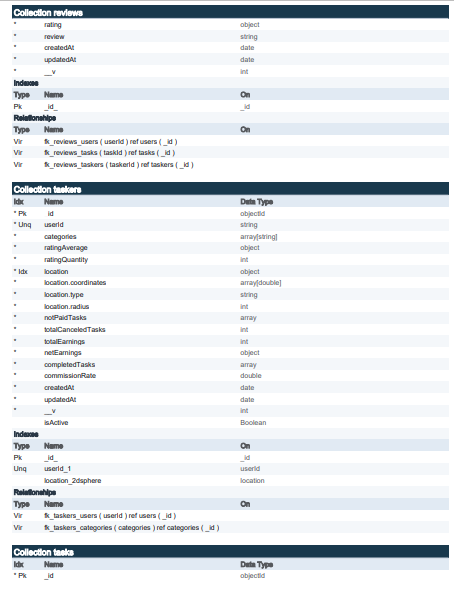
**Software Design**

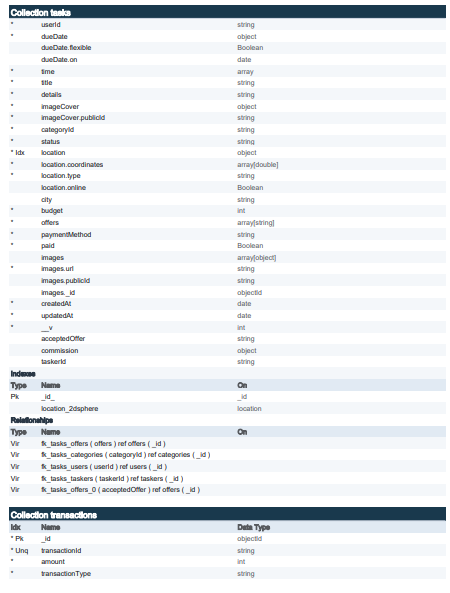
**3.1 Design of Database**

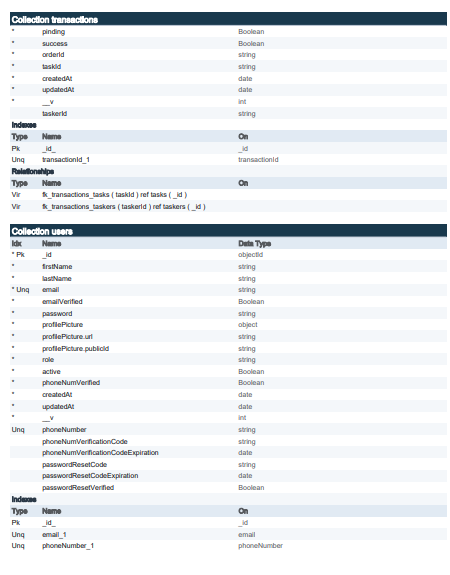
****

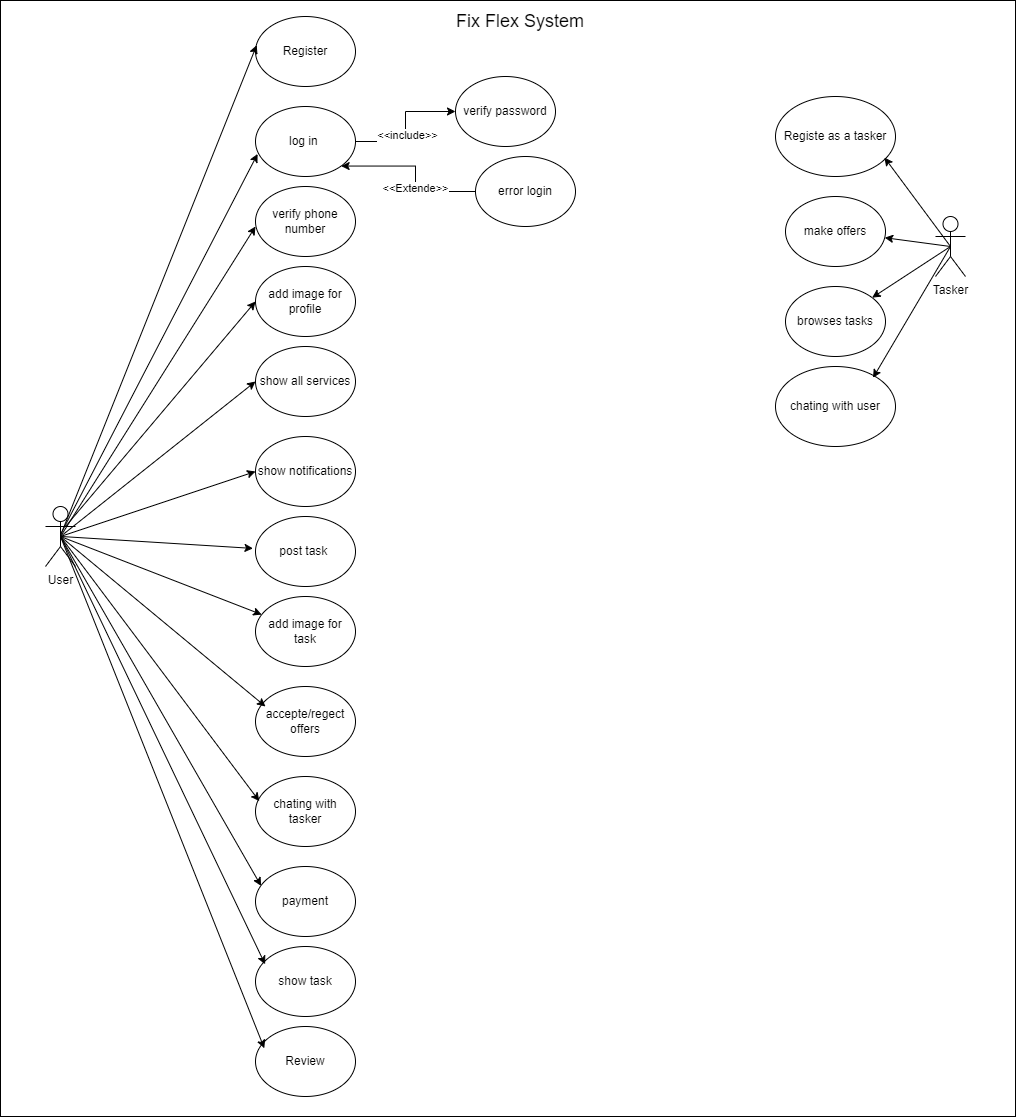
****

****

****

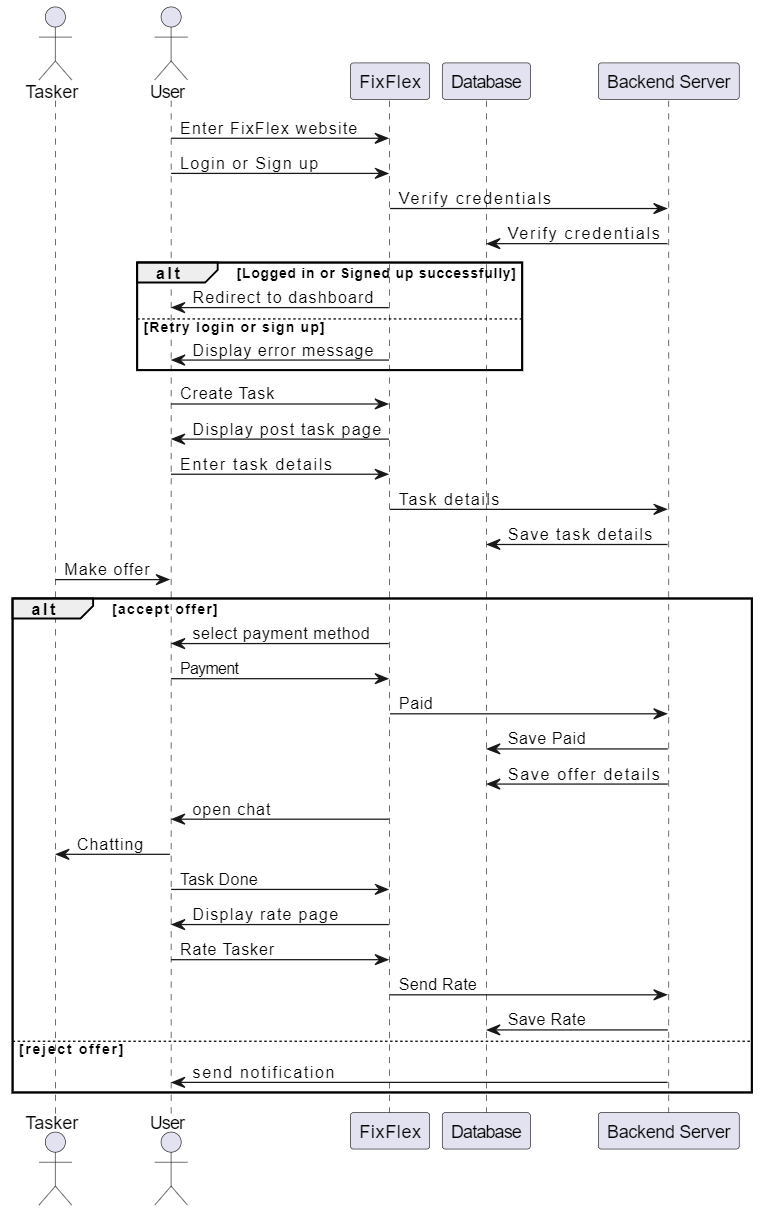






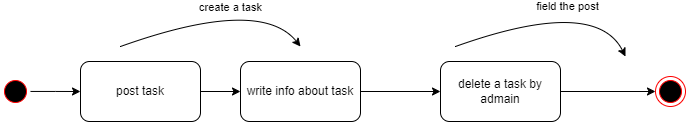
**3.2 Use Case Digram**

****

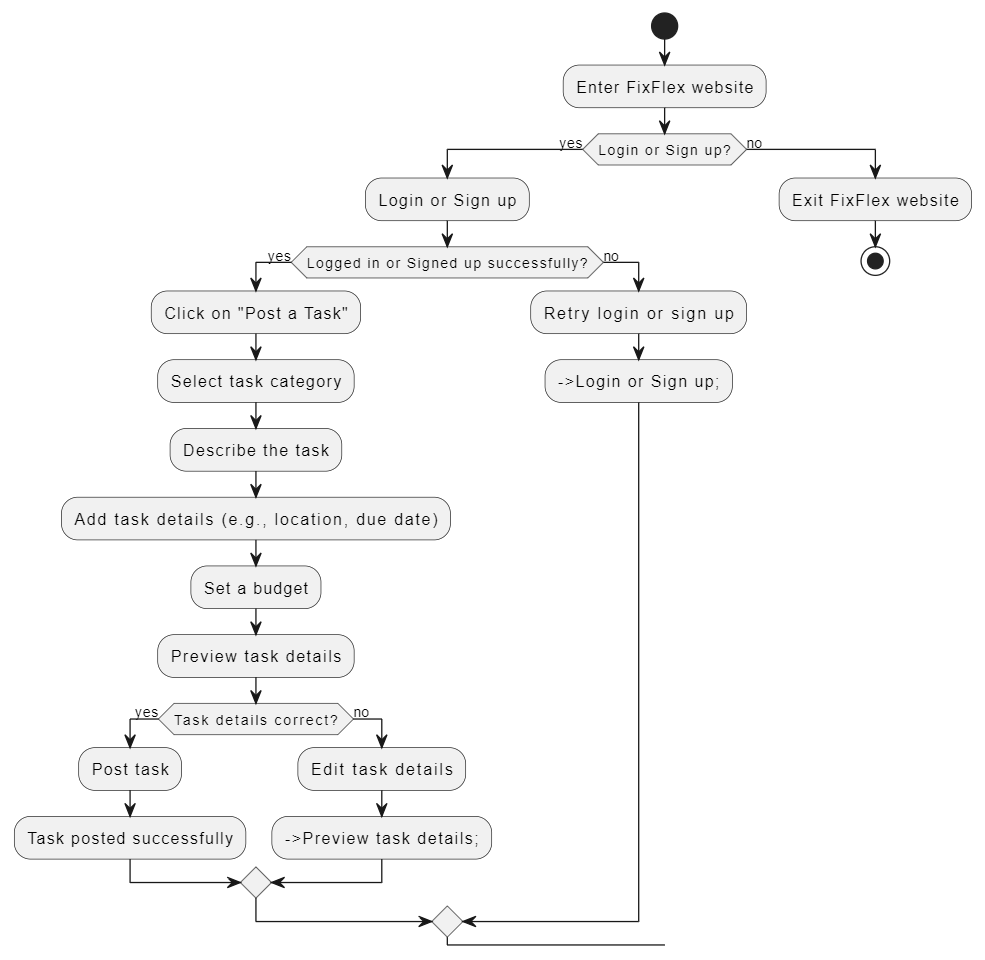
****

3.4 **State Digram**

* **Create a task**
* **Delete the post**

****

3.5 **Activity Digram**

****

**Chapter 4**

**System Implementation and Results**

**Software Architecture:**

Backend:

The backend of FixFlex follows a modular, scalable, and maintainable architecture, leveraging TypeScript, Express.js, and MongoDB. The architecture is designed to ensure code reusability, separation of concerns, and scalability to accommodate future growth and changes in the application's requirements.

## Model-View-Controller (MVC) Pattern

The backend architecture of FixFlex follows the Model-View-Controller (MVC) pattern, which divides the application into three main components:

* Models: Represents the data structure of the application, interacts with the database using Mongoose, and defines the schema for each resource.
* Controllers: Processes incoming requests, interacts with services, and returns responses to clients.
* Routes: Defines the API endpoints, maps HTTP methods to controller actions, and provides a standardized interface for clients to interact with the application.

## Data Access Object (DAO) Pattern

The backend architecture of FixFlex employs the Data Access Object (DAO) pattern to separate the data access logic from the business logic. The DAO pattern encapsulates the database operations for each resource into a separate file, providing a clean and modular structure for managing data access.

## Data Transfer Object (DTO) Pattern

The backend architecture of FixFlex uses Data Transfer Objects (DTOs) to transfer data between layers and components. DTOs define the structure of data exchanged between the client and server, ensuring consistency and type safety throughout the application.

## Service Layer

The backend architecture of FixFlex includes a service layer that encapsulates the business logic of the application, interacts with the data access layer, and provides a centralized interface for controllers to access application functionality. The service layer helps maintain separation of concerns, improves code reusability, and facilitates testing and debugging.

## Middleware

The backend architecture of fixflex includes middleware functions that intercept incoming requests, perform actions, and pass control to the next middleware or route handler. Middleware functions are used for tasks such as authentication, error handling, logging, input validation, and more, providing a modular and extensible architecture for managing request processing.

## Dependency Injection

The backend architecture of FixFlex employs dependency injection using the tsyringe package to manage component dependencies and improve code maintainability, flexibility, and testability. Dependency injection helps decouple components, promote code reuse, and facilitate the implementation of inversion of control (IoC) principles.

## Database Schema

The database used for FixFlex is MongoDB, a NoSQL database that stores data in a flexible, JSON-like format.

See [Database Schema](https://github.com/fixflex/backend/blob/stage/MongoDB-schema-visualization.md) file for more details.

The database schema for FixFlex consists of the following collections:

* Users: Stores user information such as name, email, phone number, password, role, profile picture, and verification status.
* Taskers: Stores tasker information such as user ID, description, skills, qualifications, availability, ratings, and earnings.
* Tasks: Stores task information such as user ID, title, description, category, location, status, budget, images, and timestamps.
* Categories: Stores category information such as name, description, parent category, and subcategories.
* Coupons: Stores coupon information such as code, discount amount, expiry date, and usage limit.
* Offers: Stores offer information such as task ID, tasker ID, amount, status, and timestamps.
* Reviews: Stores review information such as task ID, tasker ID, user ID, rating, comment, and timestamps.
* Chats: Stores chat information such as user IDs, task ID, tasker ID, and timestamps.
* Messages: Stores message information such as chat ID, user ID, tasker ID, content, and timestamps.
* Transactions: Stores transaction information such as user ID, tasker ID, amount, type, status, and timestamps.

## RESTful API Design

The API design for FixFlex follows RESTful principles, with clear and predictable URL structures, HTTP methods, and status codes. The API endpoints are organized into resource-based routes, with each route corresponding to a specific resource or entity in the system. The API design includes the following key features:

* Resource-Based Routes: Organizes API endpoints into resource-based routes such as /users, /tasks, /taskers, /offers, /reviews, /categories, and /coupons.
* CRUD Operations: Implements Create, Read, Update, and Delete operations for all resources using HTTP methods such as GET, POST, PUT, PATCH, and DELETE.
* Pagination and Filtering: Supports pagination and filtering for listing resources, allowing users to limit the number of results and filter based on specific criteria.
* Search and Sorting: Allows users to search for resources based on keywords, categories, locations, and other criteria, enhancing resource discovery.
* Error Handling: Provides informative error messages, status codes, and error responses to help clients understand and handle errors effectively.

## Modular Structure

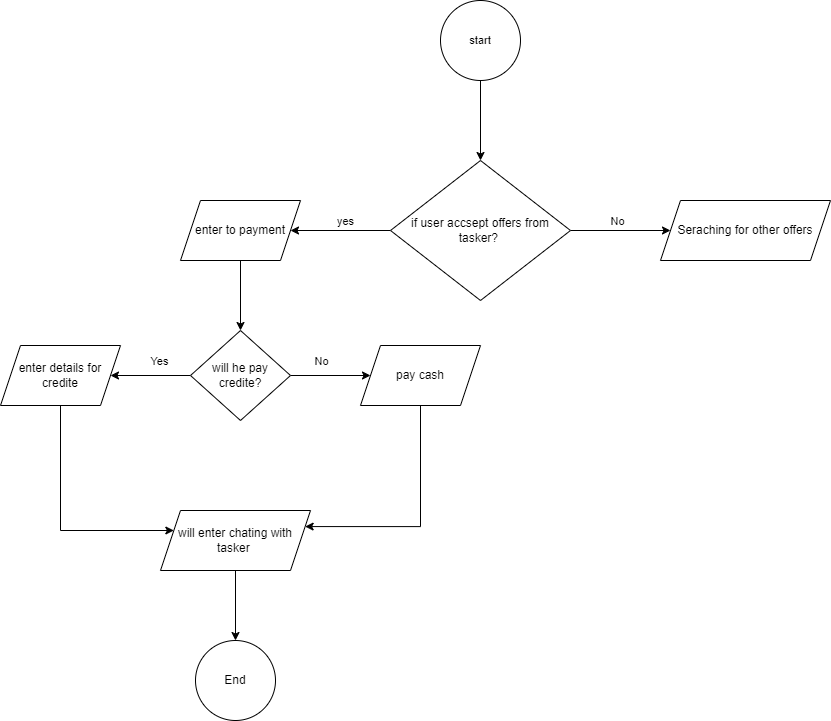
* **Controllers**: Handle incoming requests, process data, and send responses to clients.
* **Services**: Contain business logic, interact with the database, and perform operations on data.
* **Models**: Define data structures and schemas for MongoDB collections using Mongoose.
* **Routes**: Define API endpoints, route requests to controllers, and handle HTTP methods.
* **Middleware**: Implement custom middleware functions for authentication, validation, error handling, etc.
* **Exceptions**: Define custom exception classes to handle errors and exceptions in the application.
* **Helpers**: Contain utility functions, helper classes, and third-party integrations to assist with common tasks.
* **Config**: Store configuration settings, environment variables, and validation rules.
* **DB**: Contain data access objects (DAOs) to interact with the MongoDB database.
* **Docs**: Define Swagger documentation files for API endpoints and models.
* **DTOs**: Define data transfer objects (DTOs) to transfer data between layers and components.
* **Interfaces**: Define TypeScript interfaces for data structures, models, and API requests and responses.
* **Middlewares**: Implement custom middleware functions for authentication, validation, error handling, etc.
* **Routes**: Define API endpoints, route requests to controllers, and handle HTTP methods.
* **Services**: Contain business logic, interact with the database, and perform operations on data.
* **Sockets**: Implement real-time communication between clients and servers using Web Sockets.
* **Types**: Define custom TypeScript types, interfaces, and declarations for better type safety and code quality.

## Error Handling

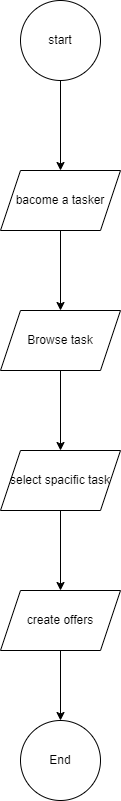
The backend architecture of FixFlex includes a robust error handling mechanism to catch and handle exceptions gracefully, providing informative error messages, status codes, and error responses to clients. The error handling mechanism includes the following key features:

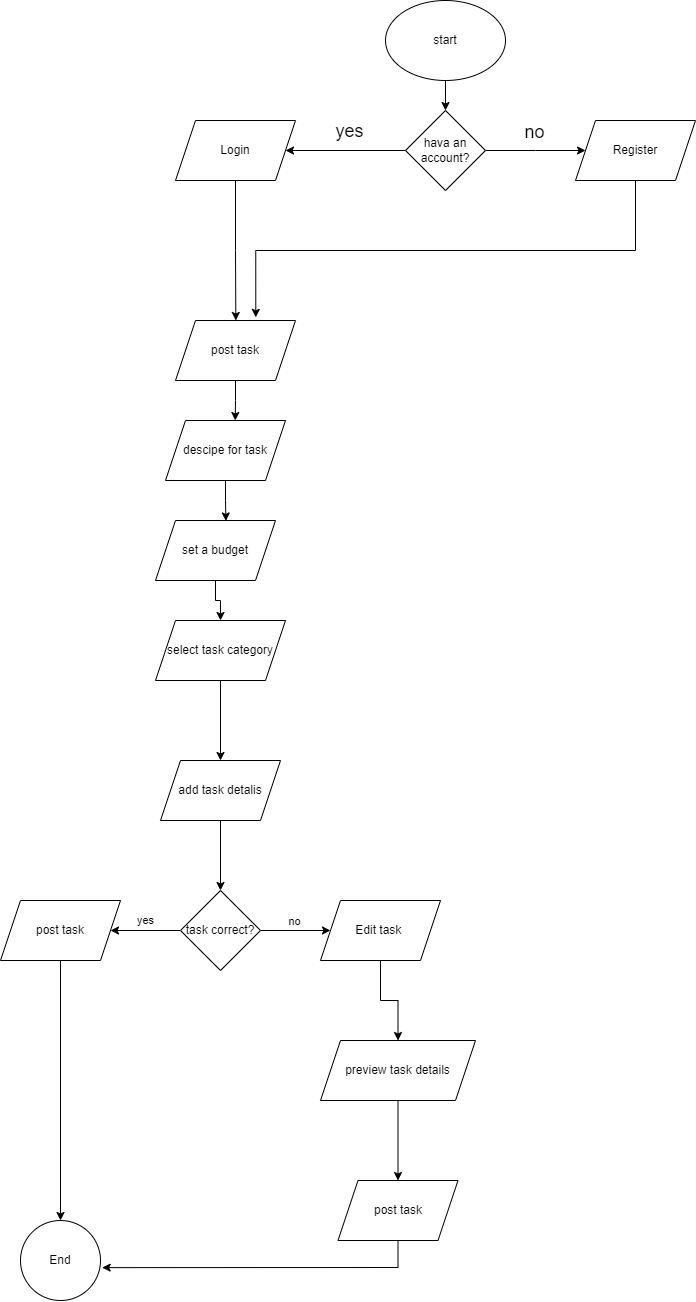
* Custom Exceptions: Defines custom exception classes to handle errors and exceptions in the application, providing detailed error messages and status codes.
* Error Middleware: Implements error middleware functions to catch and handle exceptions, log errors, and send error responses to clients.
* Global Error Handling: Centralizes error handling logic in a global error handler middleware to ensure consistent error responses across the application.
* Error Logging: Logs errors and exceptions to track application behavior, monitor performance, and debug issues effectively.
* Error Response Format: Standardizes error responses with a consistent format, including status codes, error messages, error details, and stack traces.

**Flowchart:**

** -Accept Offers**

* + **Tasker make offers**

****

* + **post task**

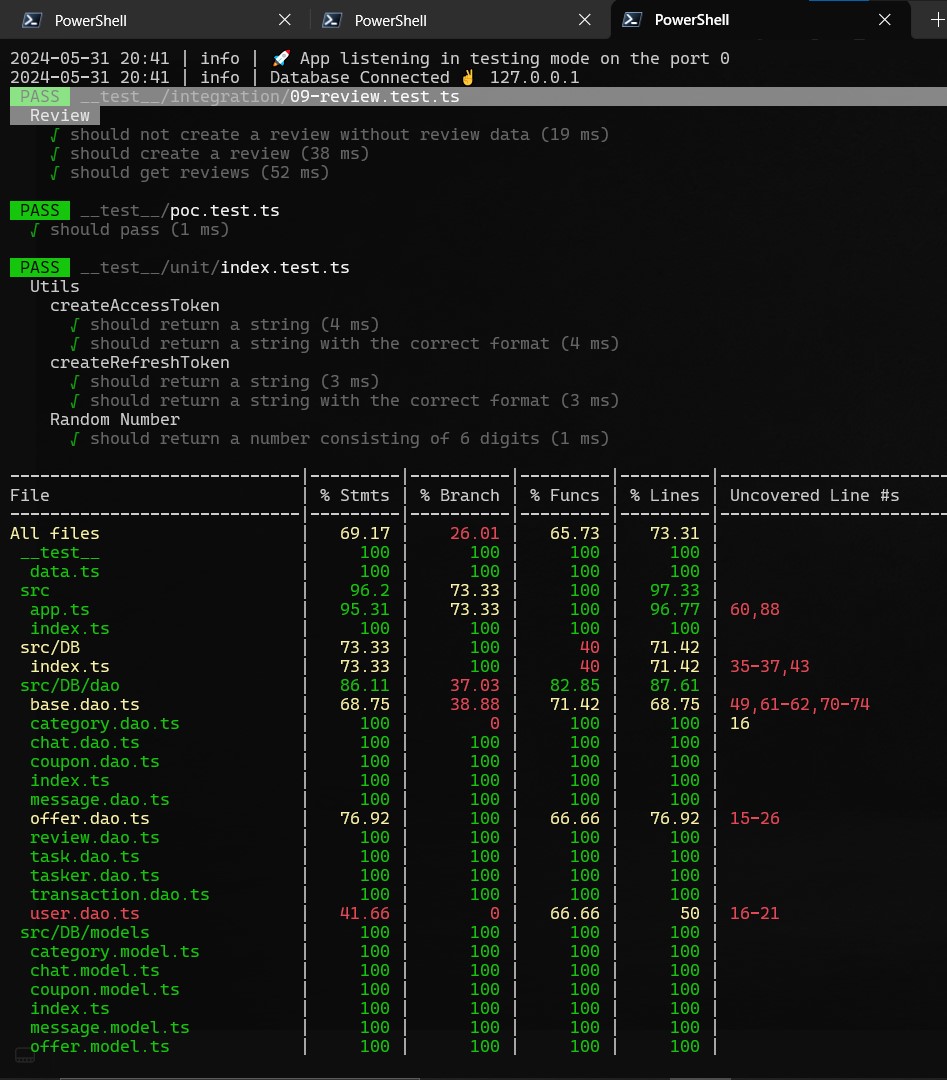
**Chapter 5**

**Testing**

****

****

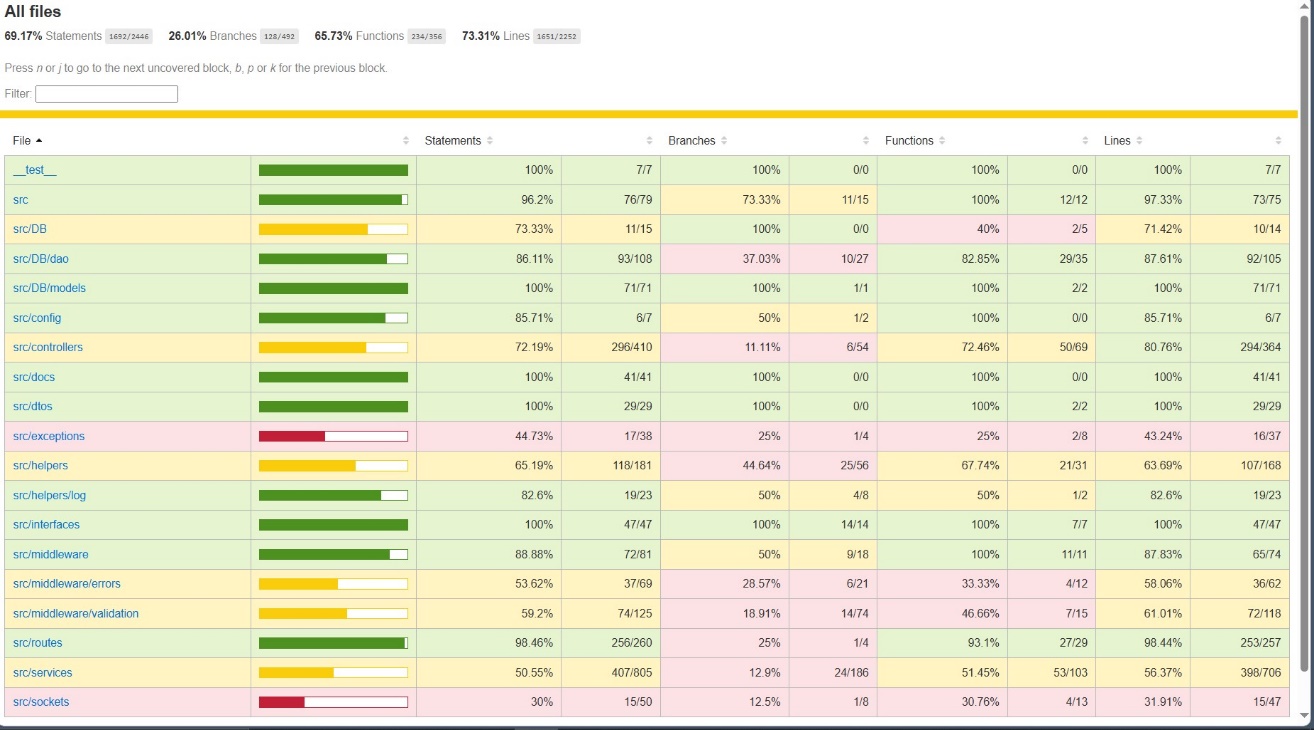
****

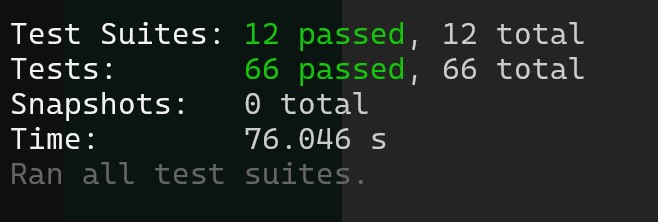












**Chapter 6**

**Result**

**Architecture in Mobile Phone App:**

****

