7SENG013C.Y Software Development Project

Project Progress Report (PPR)

Gig Work Marketplace Software Project

Student: Ms. Poddiwela Keerthirathna (20274450)

Supervisor: Ms. Dileeka Alwis

Date: 04/02/2025

MSc Information Technology
School of Computer Science & Engineering
College of Design, Creative and Digital Industries
University of Westminster

Table of Contents

Table of Figures

1 Overview & changes to the project

Gig work spans across a wide range of tasks, from handyman work, babysitting to online freelancing (De Stefano and Aloisi, 2018). Gig economy is rapidly evolving with technological advancement, emergence of the Gen Y and Gen Z workforce and the Covid-19 pandemic (Ray, Sengupta and Varma, 2024). Amisdt this volatality of the gig economy, both employers and part-time workers struggle to find suitable matches efficiently as the current digital platforms present an unmanageable amount of job opportunities and candidates, which makes finding the right match challenging. Technology is called for in the gig economy for efficient work allocation.

This software development project intends to bridge this gap by developing a web application that connects employers with part-time workers. This digital platform will facilitate job posting, job picking by gig workers, and seamless communication between employers and workers. This solution enhances gig economy participation by narrowing its focus on gig work, unlike platforms that caters to both full-time and flexible jobs. Employers will benefit from efficiently sourcing reliable part-time talent for numerous tasks, ranging from household work to short-term professional work. Gig employees will find the right opportunity to earn additional income by rendering the posted service.

This application will serve the needs of below users.
1) Employers: Businesses and individuals requiring reliable part-time workers for various tasks.
2) Gig Workers: Individuals, including students, freelancers seeking short-term job opportunities that fit their schedules.

Due to the limitations in time and skills, the following areas are deprioritized to be handled in a subsequent iteration of the project.
- Enhancing the performance of the application to handle a growing number of users will be delayed to ensure that the core functionalities are working as expected.
- Designing an intuitive user interface to ensure both employers and gig workers have a seamless experience will be set aside to focus on the basic usability.

With the above refinements, this gig work intermediary platform will cover the following core features.
- Allow employers to post part-time jobs with skills required and payment offered.
- Enable gig workers to search for gigs and apply for suitable opportunities.
- Facilitate in-app messaging for clear communication between employers and workers.
- Send notifications to users about job offers, application statuses, and messages.

2 Project aims and objectives

The purpose of this digital platform is to streamline the flexible work allocation process by connecting employers looking for part-time workers and individuals seeking part-time employment. The proposed aims of this development project remain unchanged. To address the current inefficiencies in the part-time job market the essential functional requirements should be prioritized. Hence, the project objectives are updated to reflect the changes of the project scope. The objectives of ensuring the scalability of the application to handle a growing amount of user data (PO1.3 and PO2.3) were deprioritized, to retain focus on the essential functional requirements of the application. The updated list of project aims and objectives are as follows.

The aims of the project are:

Aim 1 (PA1) – Design and develop a user friendly, simple and intuitive digital platform that enables seamless task outsourcing for employers.

Aim 2 (PA2) – Design and develop a user friendly, simple and intuitive digital platform that facilitates efficient job seeking for part-time workers.

Aim 3 (PA3) - Enhance communication between employers and part-time workers to improve task outsourcing.

The objectives for PA1 are as follows.
- (PO1.1) Implement a responsive user interface that allows employers to easily post jobs.
- (PO1.2) Develop a database to manage employer data and job postings securely and efficiently.

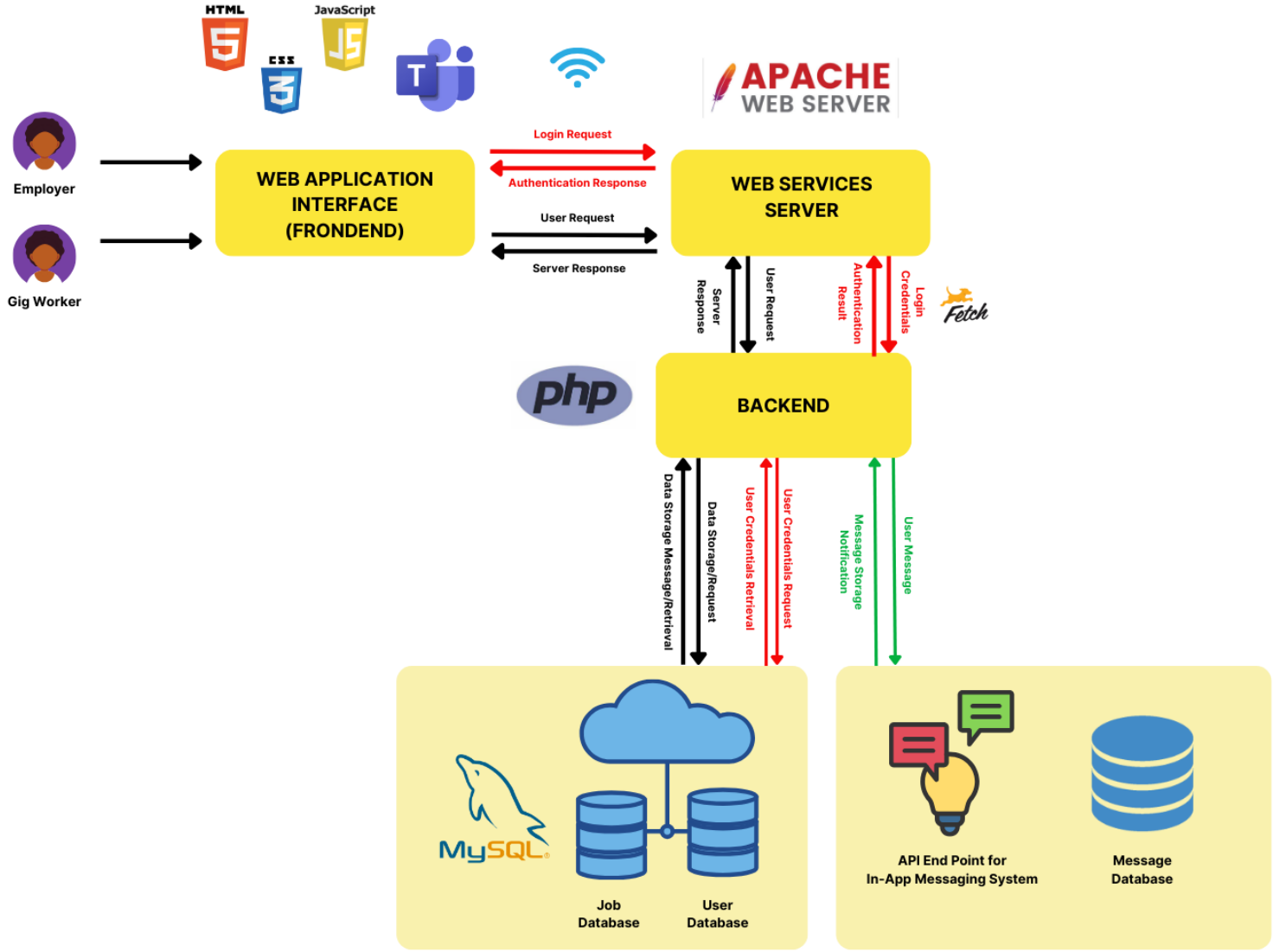
The objectives for PA2 are as follows.
- (PO2.1) Implement a responsive user interface that allows part-time workers to easily search and apply for jobs.
- (PO2.2) Develop a database to manage employee data and applications securely and efficiently.

The objectives for PA3 are as follows.
- (PO3.1) Implement an in-app messaging system to facilitate seamless communication between employers and part-time workers.
- (PO3.2) Develop a notification system to notify users about application statuses and messages.

3 Project system architecture diagram

The system architecture diagram of this web application is shown in Figure 1.

Figure 1


HTML
CSS
JavaScript
APACHE WEB SERVER
Employer
Gig Worker
WEB APPLICATION INTERFACE (FRONDEND)
Login Request
Authentication Response
User Request
Server Response
WEB SERVICES SERVER
Server Response
User Request
Authentication Result
Login Credentials
Fetch
php
BACKEND
Data Storage Message/Retrieval
Data Storage/Request
User Credentials Retrieval
User Credentials Request
Message Storage Notification
User Message
MySQL
Job Database
User Database
API End Point for In-App Messaging System
Message Database

When a user (i.e. employer or gig worker) engages with the web interface (e.g., log in, post a job, search for jobs, pick a task), an HTTP request is sent from the frontend to the backend. When a user attempts to log in, username and password are obtained from the user, and these are compared against the user details retrieved from the user database. If the user-provided details and database details match, the user is allowed to obtain other services from the web application. When a user looks for any other service provided by the applocation (e.g., posting a job, searching for jobs, picking a task), the backend server interacts with job database to retrieve or store data. After processing such requests, the backend sends a response to the user via the web services server. When a user messages another user, the backend server interacts with in-app messaging system to store the message and provide notifications to the recipient.

4 Research

4.1 Research impact

During the user requirements analysis stage of this project, existing applications that connects employers with part-time workers, including GigSmart, Airtasker and Indeed, were reviewed. Based on the comparison of similar applications, the features that users of a gig work intermediary platform would benefit from were identified. Those are listed below.

1) Job Posting: Employers should be able to post jobs quickly without the hassle of creating a user account. The risk of fradulent job postings can be minimized by enabling workers and employers to communicate with each other via the in- app messaging system.
2) Categorizing tasks: Tasks in the platform should be categorized based on the nature of the service to enable employers to easily post their jobs and workers to efficiently find suitable opportunities. The interface should be visually appealing and easy to navigate.
3) Job search: Gig workers should be able to search for jobs using keywords, job titles and locations without creating a user account.
4) Task picking: Gig workers should be able to pick tasks that match their skills, experience, and availability by logging in from a user account.
5) User Profiles: Both employers and gig workers should be able to create user accounts, where worker profiles include details on skills, ratings, and completion rates to help employers make informed decisions.
6) Communication Tools: Both employers and workers should be able to maintain clear and efficient communication with an in-app messaging system.

Considerable research was conducted on frontend and backend development technologies. This research enhanced the frontend web architecture by understanding the features of HTML5 to define organized web structures (Awasthi, More and Viegas, 2022). The evaluation of data processing capabilities, programming language considerations and volume of different frontend development tools leads to the development of a user-friendly solution (Awasthi, More and Viegas, 2022). The review of elements of relational database management systems results in a robust and efficient application that

promptly responds to user requests originating from the frontend (Vishesh et al., 2017).

Additionally, significant research has been done on the regulatory impact of labour laws to ensure that the application promotes ethical labour practices. This results in the application being legally compliant ethically acceptable. Research on regulatory frameworks revealed that the labour laws are applicable to this web application only if the system maintain managerial control over gig workers (Stewart and Stanford, 2017). As this platform only acts as an intermediary between gig workers without taking control over either party, no additional measures will be taken to comply with laws and regulations.

Furthermore, user interface (UI) research has been carried out to improve the user experience and satisfaction. UI research focused on understanding user needs which leads to optimum user flows and higher coversion rate (Aeologic Technologies, 2025). The importance of structuring information on the application was identified that would result in gaining user's attention to important elements of the application (Aeologic Technologies, 2025). Graphic elements will be made painly visible and consistent throughout the application to make the application visually appealing and organized (Aeologic Technologies, 2025).

4.2 Remaining research topics to be investigated

There are several other areas to be investigated to further refine this web application including:

- Integration of pre-built functionalities offered by external messaging services via RESTful APIs (Zetaton, 2024).
- Web application security measures to protect sensitive data of users and reduce the risk of cyber attacks (Katz, 2024).

5 Functional requirements

The final list of functional and non-functional requirements for the web application, categorized into essential, desirable and luxury, are as follows. Desirable and luxury requirements are grouped together as desirable. At the end of the current project iteration, all the essential functional and non-functional requirements are planned to be implemented. The desirable requirements will be addressed in a subsequent iteration.

<u>Essential Functional Requirements</u>

1) User Registration and Authentication

- Employers and workers can create accounts.
- Secure login and password maintenance.

2) Profile Management

- Employers can create and manage company or individual profiles.

- Workers can create and update personal profiles, including skills and experience.

3) Job Posting and Management
- Employers can post jobs with details such as job description, requirements, and budget.
- Employers can manage and edit job postings.

4) Job Search and Application
- Gig workers can search for jobs based on various filters (location, pay, job type).
- Workers can apply for jobs directly through the platform.

5) Communication Tools
- In-app messaging between employers and workers.
- Notifications for job postings, applications, and messages.

6) Application Tracking
- Employers can track applications and manage the hiring process.
- Workers can track the status of their applications.

7) Ratings and Reviews
- Employers can rate and review workers.

Desirable Functional Requirements

1) Advanced Search Filters
- More detailed filters for job searches (e.g., remote/on-site, date posted).

2) Calendar Integration
- Integration with calendar apps for scheduling interviews and job start dates.

Essential Non-Functional Requirements

1) Usability
- Organized, visually appealing, and easy to navigate.

Desirable Non-Functional Requirements

1) Performance
- Fast load times and responsive design.
- Scalability to handle a large number of users.

2) Security
- Data encryption for sensitive user information.
- Regular security audits and updates.

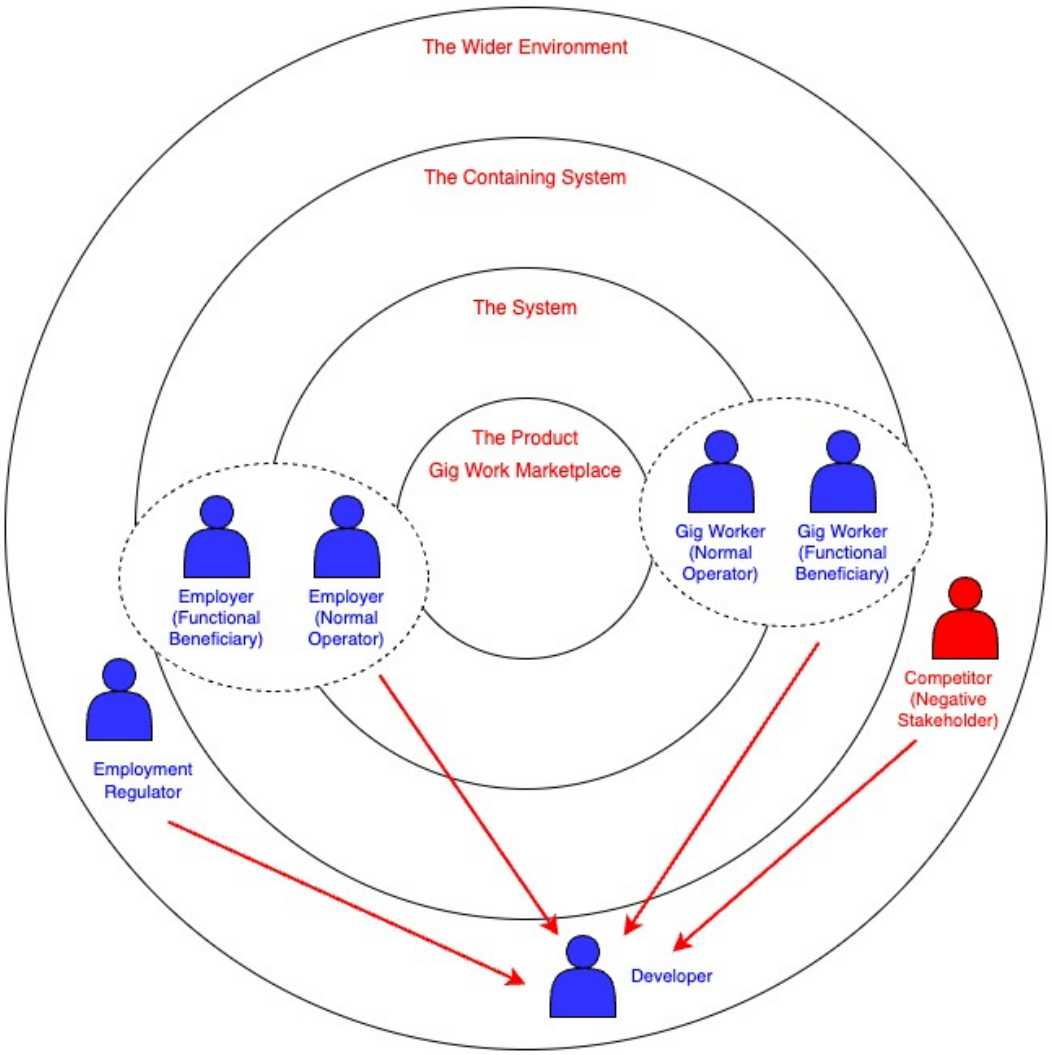
3) Usability
- Intuitive and user-friendly interface.

6 Design overview

6.1 Onion model for stakeholder analysis

The onion model given in Figure 2 represents the stakeholders of this web application. The innermost ring is the Gig Work Marketplace software that this project intends to develop, the system ring represents the end users who directly interact with the system, the containing system ring denotes the generalized parties who are functional beneficiaries, and the outermost wider environment ring represents the affected external stakeholders including negative stakeholders.

Figure 2

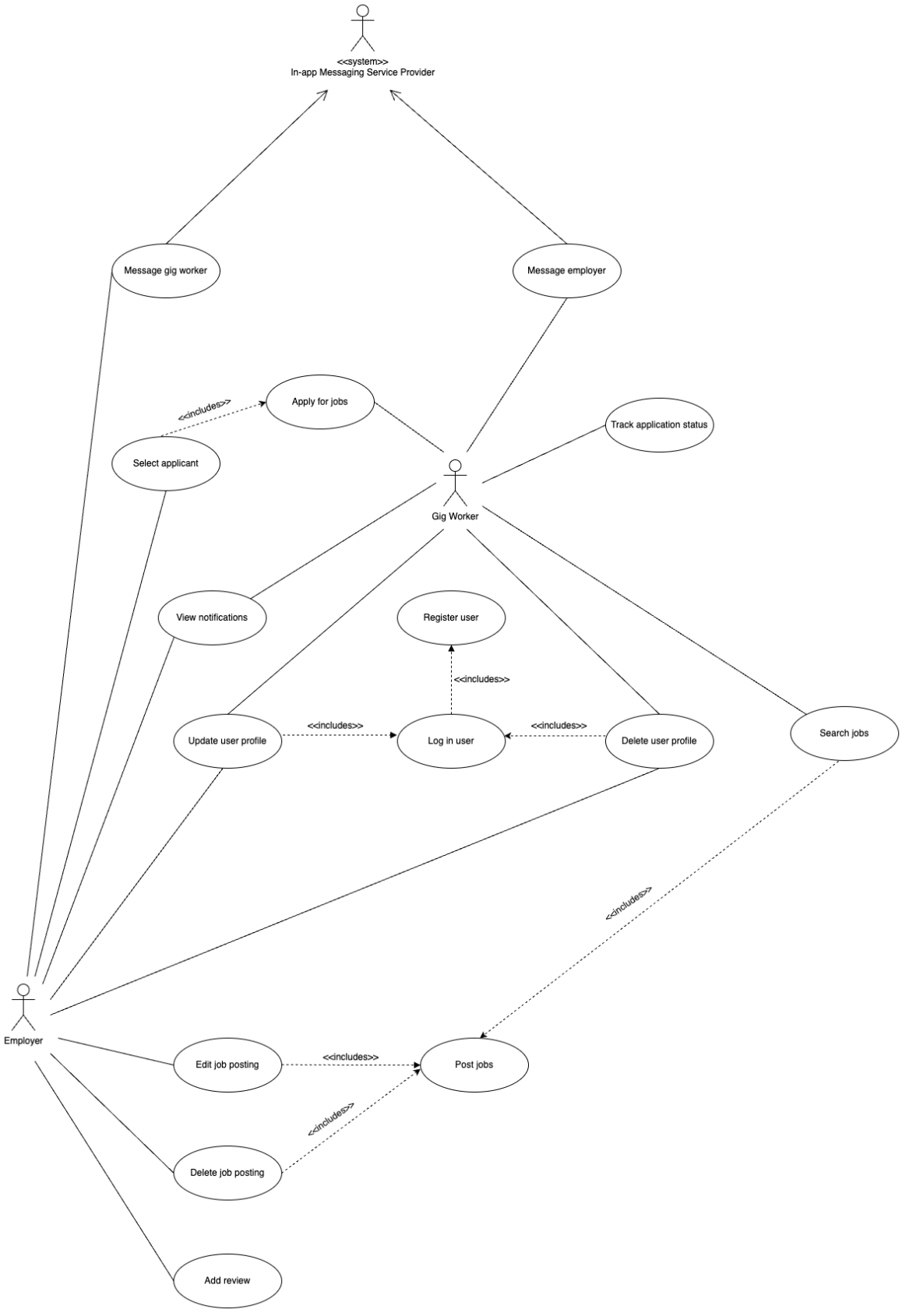


6.2 Use case diagram

Figure 3 illustrates the use case diagram of this web application. There are 3 actors in this model who are associated with multiple use cases. There are several include relationships among use cases in this system. To log in to the web application, users have register with application which is demonstrated as

a include relationship between “Log in user” and “Register user” use cases. Updating and deleting user profiles include the behaviour of “Log in user” use case. Editing and deleting of job postings use the common behaviour of “Post jobs” use case. In addition, searching jobs by gig workers include the behaviour of “Post jobs” use case carried out by employers. The “Select applicant” base use case uses the behaviour of “Apply for jobs”, as the gig workers have to apply for jobs for the employers to review applications.

Figure 3


<<system>>
In-app Messaging Service Provider
Message gig worker
Message employer
<<includes>>
Apply for jobs
Track application status
Select applicant
Gig Worker
View notifications
Register user
<<includes>>
Update user profile
<<includes>>
Log in user
<<includes>>
Delete user profile
Search jobs
<<includes>>
Employer
Edit job posting
<<includes>>
Post jobs
<<includes>>
Delete job posting
Add review

6.3 Class diagram

The class diagram of this web application is given in Figure 4. There are several association relationships among classes in this system, which are listed below.

- Employer creates Job Posting.
- Job Posting belongs to Job Category.
- Gig Worker creates Job Application.
- Gig Worker applies for Job Posting.
- User sends Message.
- User receives Message.
- User receives Notification.
- Employer gives Rating.

The aggregation relationships among the classes in this system are as follows.

- User has a User Account.
- Notification has a message.

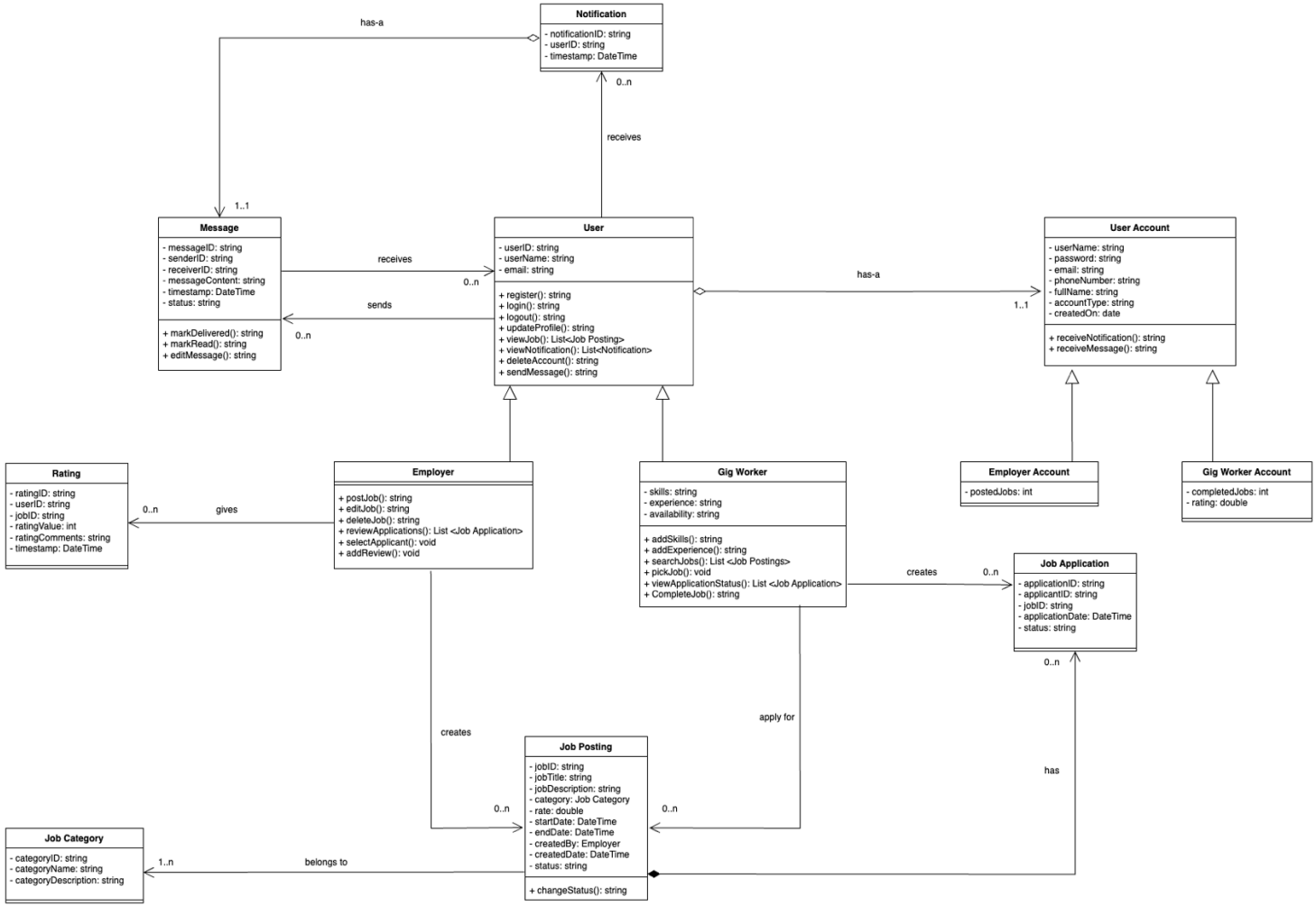
Composition relationships exist between two classes in this system, which is given below.

- Job Application has a Job Posting.

The generalization relationships between classes in this web application are listed below.

- Employer is a User.
- Gig Worker is a User.
- Employer Account is a User Account.
- Gig Worker Account is a User Account.

Figure 4


Notification
- notificationID: string
- userID: string
- timestamp: DateTime
has-a
0..n
receives
1..1
Message
- messageID: string
- senderID: string
- receiverID: string
- messageContent: string
- timestamp: DateTime
- status: string
+ markDelivered(): string
+ markRead(): string
+ editMessage(): string
receives
0..n
sends
0..n
User
- userID: string
- userName: string
- email: string
+ register(): string
+ login(): string
+ logout(): string
+ updateProfile(): string
+ viewJob(): List<Job Posting>
+ viewNotification(): List<Notification>
+ deleteAccount(): string
+ sendMessage(): string
has-a
1..1
User Account
- userName: string
- password: string
- email: string
- phoneNumber: string
- fullName: string
- accountType: string
- createdOn: date
+ receiveNotification(): string
+ receiveMessage(): string
Rating
- ratingID: string
- userID: string
- jobID: string
- ratingValue: int
- ratingComments: string
- timestamp: DateTime
0..n
gives
Employer
+ postJob(): string
+ editJob(): string
+ deleteJob(): string
+ reviewApplications(): List <Job Application>
+ selectApplicant(): void
+ addReview(): void
Gig Worker
- skills: string
- experience: string
- availability: string
+ addSkills(): string
+ addExperience(): string
+ searchJobs(): List <Job Postings>
+ pickJob(): void
+ viewApplicationStatus(): List <Job Application>
+ CompleteJob(): string
Employer Account
- postedJobs: int
Gig Worker Account
- completedJobs: int
- rating: double
creates
0..n
Job Application
- applicationID: string
- applicantID: string
- jobID: string
- applicationDate: DateTime
- status: string
0..n
creates
0..n
Job Posting
- jobID: string
- jobTitle: string
- jobDescription: string
- category: Job Category
- rate: double
- startDate: DateTime
- endDate: DateTime
- createdBy: Employer
- createdDate: DateTime
- status: string
+ changeStatus(): string
0..n
apply for
has
Job Category
- categoryID: string
- categoryName: string
- categoryDescription: string
1..n
belongs to

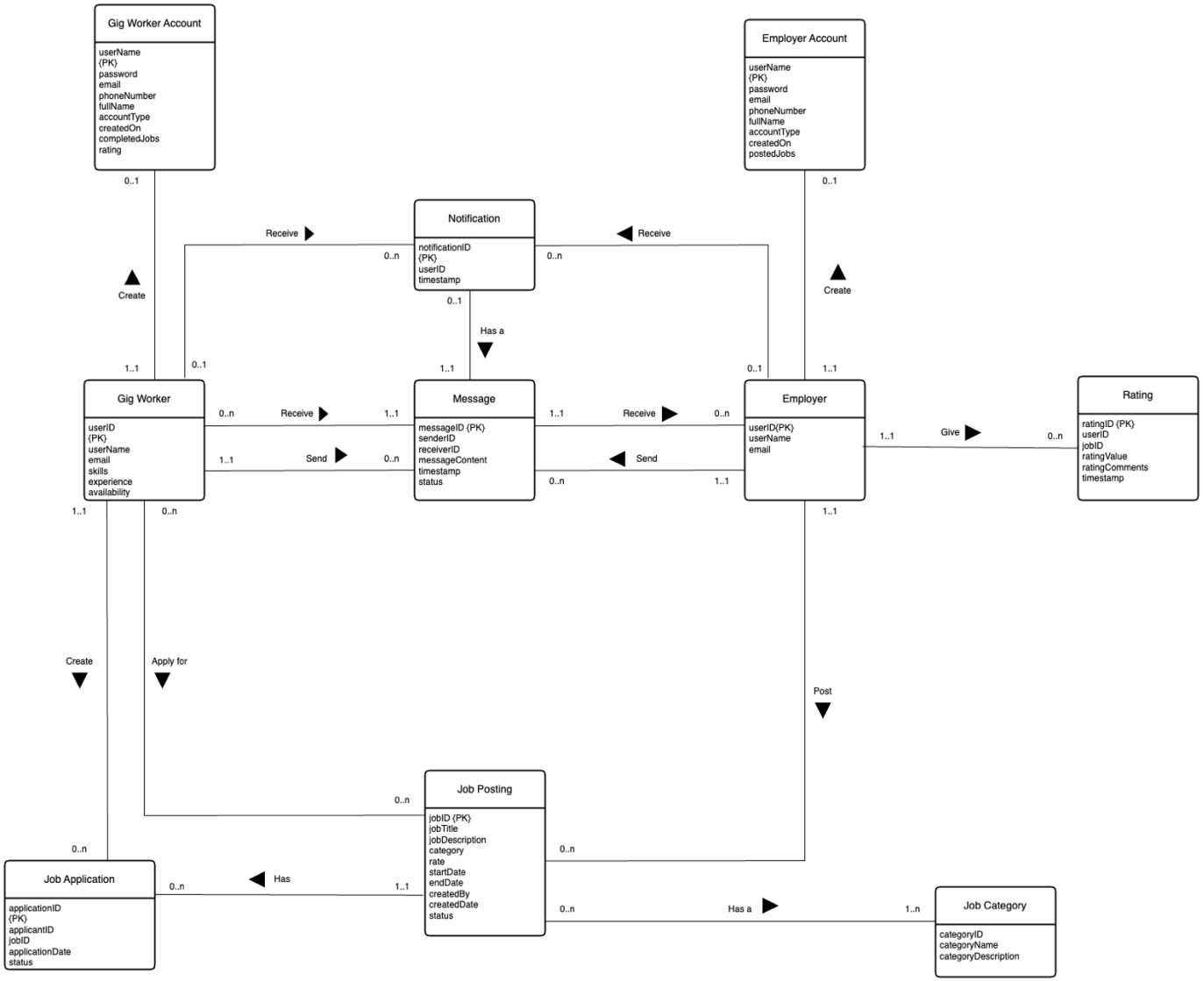
6.4 Entity-Relationship diagram

The entity-relationship diagram of this system given in Figure 5. The main relationships in the database of this web application are,

- Gig Worker create Gig Worker Account.
- Employer create Employer Account.
- Employer post Job Posting.
- Gig Worker create Job Application.
- Gig Worker apply for Job Posting.
- Job Posting has Job Application.
- Job Posting has a Job Category.
- Gig Worker receive Message.
- Gig Worker send Message.
- Employer receive Message.
- Employer send Message.
- Notification has a Message.
- Gig Worker receive Notification.
- Employer receive Notification.
- Employer give Rating.

Cardinality and participation of each relationship, and primary keys of each entityt are represented in the diagram.

Figure 5


Gig Worker Account
userName
{PK}
password
email
phoneNumber
fullName
accountType
createdOn
completedJobs
rating
Employer Account
userName
{PK}
password
email
phoneNumber
fullName
accountType
createdOn
postedJobs
Notification
notificationID
{PK}
userID
timestamp
Receive
Receive
Create
Create
Has a
Gig Worker
userID
{PK}
userName
email
skills
experience
availability
Message
messageID {PK}
senderID
receiverID
messageContent
timestamp
status
Receive
Receive
Send
Send
Employer
userID{PK}
userName
email
Give
Rating
ratingID {PK}
userID
jobID
ratingValue
ratingComments
timestamp
Create
Apply for
Post
Job Posting
jobID {PK}
jobTitle
jobDescription
category
rate
startDate
endDate
createdBy
createdDate
status
Job Application
applicationID
{PK}
applicantID
jobID
applicationDate
status
Has
Has a
Job Category
categoryID
categoryName
categoryDescription

7 Implementation Issues, Techniques & Technologies

To complete the implementation of this project, the following issues need to be researched and solved.

- Data Security: Ensuring that user data is protected from unauthorized access and implement advanced security measures to prevent data breaches.
- User Authentication: Implementing secure and efficient user authentication mechanisms.
- API Integration: Seamlessly integrating with third-party APIs, including Microsoft Teams for messaging services.
- Performance Optimization: Enhancing the application's performance to handle moderate loads and complex operations.

PostgreSQL was planned to be used as the database management system of this web application to manage user data securely and efficiently. However, the plan was revised to use the XAMPP server as the local host, and MySQL as the database management system which is embedded in the XAMPP distribution to streamline the development process.

The technologies outlined below will be used to develop this web application.

- Web development environment: XAMPP web server.
- Frontend development: HTML to structure the web content (AppMaster, 2023), CSS to style the application (Abramowski, 2023), and JavaScript to add interactivity (Joshi, 2023).
- Backend development: PHP scripting language.
- Database: MySQL as the database management system.
- Web server: Apache web server.
- Security: HTTPS for secure data transmission and OAuth 2.0 for authentication.
- API Integration: Gig Work Marketplace web application will be integrated with Microsoft Teams to leverage the in-app messaging capabilities utilizing RESTful APIs (Microsoft, 2023).

8 Testing strategy

Evaluating the quality of the web application and its components (International Software Testing Qualifications Board, no date) is important to make a judgement about the acceptability of the system and discover problems (Jorgensen, 2014). The following types of testing will be conducted to ensure the quality of this Gig Work Marketplace web application.

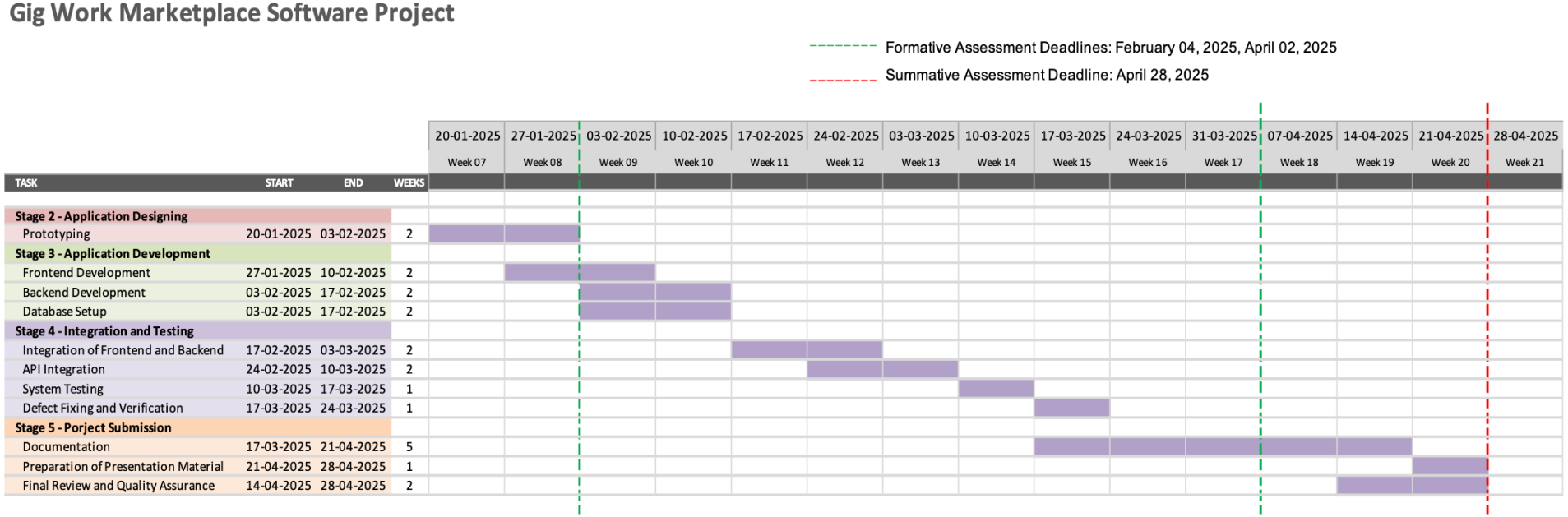
- Risk based testing will be done by prioiritizing test scenarios with highest level of risk considering the tight timeline of the project (International Software Testing Qualifications Board, no date).
- Functional testing to evaluate whether the web application satisfies the functional requirements outlined in the list of requirements (International Software Testing Qualifications Board, no date).

- Use case testing where testing is based on the execution of use cases outlined at the design phase of the project (International Software Testing Qualifications Board, no date).
- GUI testing will be conducted to review the responsiveness and intuitivity of the graphical user interface (International Software Testing Qualifications Board, no date).
- Integration testing will focus on the interaction between frontend, backend and APIs (International Software Testing Qualifications Board, no date).
- End-to-end testing where busiess processes are executed from start to finish (International Software Testing Qualifications Board, no date).
- Exploratory testing at the end of the implementation based on the knowledge of the system and results of previous tests (International Software Testing Qualifications Board, no date).
- Regression testing to identify the defects introduced in unchanged areas of the application (International Software Testing Qualifications Board, no date).

9 Remaining tasks

Stage 1 – 'Research and Requirements Gathering' and several tasks in stage 2 – 'Application Designing' of the project were completed during the first 6 weeks. The remaining tasks and milestones that need to be completed with timescales are given in Figure 6 below.

Figure 6



TASK	START	END	WEEKS
Stage 2 - Application Designing			
Prototyping	20-01-2025	03-02-2025	2
Stage 3 - Application Development			
Frontend Development	27-01-2025	10-02-2025	2
Backend Development	03-02-2025	17-02-2025	2
Database Setup	03-02-2025	17-02-2025	2
Stage 4 - Integration and Testing			
Integration of Frontend and Backend	17-02-2025	03-03-2025	2
API Integration	24-02-2025	10-03-2025	2
System Testing	10-03-2025	17-03-2025	1
Defect Fixing and Verification	17-03-2025	24-03-2025	1
Stage 5 - Porject Submission			
Documentation	17-03-2025	21-04-2025	5
Preparation of Presentation Material	21-04-2025	28-04-2025	1
Final Review and Quality Assurance	14-04-2025	28-04-2025	2

10 Conclusions and reflections

Stage 1 – 'Research and Requirements Gathering' took more time than planned resulting in a delay in starting the stage 2 – 'Application Designing' of the project. This could have been minimized by realistic planning at the initiation of the project. However, after this delay was identified the velocity of project deliverables were continuously monitored and plans were adjusted to stay within formative and summative assessment deadlines. Most of the learning materials on API integration focuses on enhancing the capabilities of existing applications by integrating our own custom applications, which is the exact opposite of the requirement of this Gig Work Marketplace web application. Hence, more research need to be conducted on integrating existing messaging applications into the web application I am building.

Continuous learning of frontend and backend development, and database management went according to the project plan. Additionally, knowledge of the macro environment, stakeholders and laws and regulations was acquired during stage 1 of the project, which resulted in prioritizing requirements for the seamless execution of the project. Furthermore, existing application designing skills were enhanced during stage 2 of the project, that led to structure the frontend and backend of this web application for efficient project implementation. Moreover, the research on UX design resulted in categorizing the user interface requirements into the basic and extended, which made the requirements prioritization of the project more realistic.

At the end of this project, I will be able to complete the development of a web application that meets the essential functional requirements listed in section 5 above. Also, I will gain the experience of planning and executing a project inline with the software development lifecycle which will help me to achieve my career objectives.

11 References

Abramowski, N. (2023). What is CSS? A Beginner's Guide. *CareerFoundry*. Available from https://careerfoundry.com/en/blog/web-%20development/what-is-css/ [Accessed 18 January 2025].

Aeologic Technologies (2025). The Role of UX/UI Design in Successful Web Development Projects. *Aeologic.* Available from https://www.aeologic.com/blog/the-role-of-ux-ui-design-in-successful-web-development-projects/ [Accessed 18 January 2025].

AppMaster (2023). The Role of HTML in Website Development. *AppMaster*. Available from https://appmaster.io/blog/html-website-development [Accessed 18 January 2025].

Awasthi, A. More, S. and Viegas, W. (2022). Research and Analysis of the Front-end Frameworks and Libraries in Web Development. *International Journal for Research in Applied Science & Engineering Technology,* 10 (4), 411-416. Available from https://www.ijraset.com/best-journal/research-and-analysis-of-frontend-frameworks-and-libraries [Accessed 18 January 2025].

De Stefano, V. & Aloisi, A. (2018). *European legal framework for 'digital labour platforms'*. Luxembourg: Publications Office of the European Union. Available from https://www.researchgate.net/publication/330683831_European_legal_framework_for_digital_labour_platforms [Accessed 03 February 2025].

International Software Testing Qualifications Board (no date). Glossary. *ISTQB.* Available from https://glossary.istqb.org/en_US/search?term=testing&exact_matches_first=true [Accessed 18 January 2025].

Jorgensen, P. C. (2014). *Software Testing: A Craftsman's Approach,* 4th ed. Boca Raton: Taylor & Francis Group. Available from https://malenezi.github.io/malenezi/SE401/Books/Software-Testing-A-Craftsman-s-Approach-Fourth-Edition-Paul-C-Jorgensen.pdf [Accessed 18 January 2025].

Joshi, M. (2023). A detailed guide on JavaScript Web Development. *BrowserStack*. Available from https://www.browserstack.com/guide/javascript-web-development [Accessed 18 January 2025].

Katz, E. (2024). 8 Web Application Security Best Practices for 2024. *open-appsec*. Available from https://www.openappsec.io/post/web-application-security-best-practices [Accessed 18 January 2025].

Microsoft (2023). Integrate web apps. *Microsoft*. Available from https://learn.microsoft.com/en-us/microsoftteams/platform/samples/integrate-web-apps-overview [Accessed 18 January 2025].

Ray, B., Sengupta, A. and Varma, A. (2024). The gig verse: building a sustainable future. *International Journal of Organizational Analysis*, 32 (*10*), 2275-2298. Available from https://www.emerald.com/insight/content/doi/10.1108/ijoa-08-2023-3946/full/pdf?title=the-gig-verse-building-a-sustainable-future [Accessed 26 January 2025].

Stewart, A. and Stanford, J. (2017). Regulating work in the gig economy: What are the options?. *The Economic and Labour Relations Review*, 28 (*3*), 420–437. Available from https://journals.sagepub.com/doi/epub/10.1177/1035304617722461 [Accessed 26 January 2025].

Vishesh, S., Hathwar, K.P., Ravishankar, R., Nandhishwara, B.N., Hema, R. and Amulya H.P. (2017). Back-End Web-Application Development and the Role of an Admin. *International Journal of Advanced Research in Computer and Communication Engineering,* 6 (9), 60-65. Available from https://ijarcce.com/upload/2017/september-17/IJARCCE%2011.pdf [Accessed 18 January 2025].

Zetaton (2024). The Role of API Integration in Modern Web Development. *Zetaton.* Available from https://www.zetaton.com/blog/the-role-of-api-integration-in-modern-web-development [Accessed 18 January 2025].