**OkHire**

**High-level Design: We will use the Layered architectural pattern for the high-level design of the “OkHire” app**. This pattern is ideal for this application because it separates concerns into distinct sections, making the system more organized and manageable. For example,

Separation of Concerns: Layered architecture divides the system into distinct layers, each responsible for specific functionalities. This separation allows for better management of complexity and facilitates modular development. In the context of "OkHire," separating the presentation, business logic, and data access functionalities into different layers ensures that changes in one layer do not directly impact others, promoting code maintainability and scalability.

Scalability: With layered architecture, it's easier to scale the application horizontally by adding more instances of the same layer without affecting other layers. For instance, if there's a sudden increase in user activity, additional instances of the presentation layer can be deployed to handle the load. In contrast, the business logic and data access layers remain unaffected.

Reusability: Each layer can be developed independently, allowing for greater code reuse across different parts of the application. For example, the business logic layer responsible for matching algorithms can be reused for job seekers and recruiters, promoting efficiency and reducing development time.

Flexibility: Layered architecture allows for flexibility in technology choices within each layer. For instance, the presentation layer can be developed using web technologies for the browser interface and mobile technologies for native mobile applications, without affecting the underlying business logic or data access layers.

Maintainability: Since each layer has well-defined responsibilities and dependencies, maintaining and debugging the application becomes easier. Developers can focus on specific layers without understanding the entire system, which streamlines the development process and reduces the risk of introducing bugs.

By structuring the "OkHire" app using the Presentation Layer, Business Logic Layer, and Data Access Layer, we ensure a clear separation of concerns, scalability, reusability, flexibility, and maintainability, making it an ideal choice for this application.

**Low-Level Design:**

Summary:

To implement our application, we would likely use a pattern from the Behavioral Design Pattern family, such as the Observer Pattern. This pattern allows a one-to-many relationship between objects. THis means that when one object changes, all dependents are notified and updated accordingly. For our platform, OkHire, profiles and chats can be opened and updated on a successful match between a job-seeker and a company recruiter. An Observer Pattern would facilitate real time updating of job postings and candidate profile creation to make sure users always have the most up to date information.

Pseudocode:

| // Subject interface representing the entity being observed  interface Subject {  registerObserver(observer: Observer)  removeObserver(observer: Observer)  notifyObservers()  }  // ConcreteSubject representing the job listing  class JobListing implements Subject {  private observers: List<Observer>  private jobDetails: JobDetails  constructor() {  observers = new ArrayList<Observer>()  }  registerObserver(observer: Observer) {  observers.add(observer)  }  removeObserver(observer: Observer) {  observers.remove(observer)  }  notifyObservers() {  for each observer in observers {  observer.update(jobDetails)  }  }  // Method to update job details and notify observers  updateJobDetails(jobDetails: JobDetails) {  this.jobDetails = jobDetails  notifyObservers()  }  }  // Observer interface representing the observer objects  interface Observer {  update(jobDetails: JobDetails)  }  // ConcreteObserver representing the job seeker  class JobSeeker implements Observer {  update(jobDetails: JobDetails) {  // Update job listings viewed by the job seeker  // Display updated job details to the job seeker  }  } |
| --- |

Class Diagram:

| +--------------------------+ +----------------+  | Subject | | Observer |  +--------------------------+ +----------------+  | + registerObserver() | <---------------- | + update() |  | + removeObserver() | +-----------------+  | + notifyObservers() |  +---------|------------------+  |  +---------|---------------+  | ConcreteSubject |  +---------|---------------+  |  +-----------|--------------------------+  | JobListing |  +--------------------------------------+  | - observers: List<Observer> |  | - jobDetails: JobDetails |  +--------------------------------------+  +---------------+  | JobSeeker |  +---------------+ |
| --- |

**Risk Analysis For OkHire: A Professional Dating App**

User Behavior and Safety:Risk: Inappropriate or harmful interactions between users.Impact: Damage to user trust and reputation, potential legal liabilities.Mitigation: Implement community guidelines and moderation policies to govern user behavior. Provide reporting and blocking mechanisms within the app to empower users to handle abusive behavior. Offer safety tips and educational resources to educate users on safe interactions.

Technical Support and Maintenance:Risk: Insufficient resources allocated for ongoing support and maintenance.Impact: Poor user experience due to unresolved issues or outdated features.Mitigation: Allocate dedicated resources for customer support and bug fixes. Implement automated monitoring and alert systems to detect and address technical issues promptly. Plan regular updates and feature enhancements to keep the app competitive and relevant in the market.

Privacy Concerns:Risk: Users may be wary of sharing personal information on the platform.Impact: Decreased user engagement, negative publicity.Mitigation: Provide clear and transparent privacy policies to assure users of data protection measures. Allow users control over their data with options for anonymization or pseudonymization. Obtain explicit consent for data usage and ensure compliance with relevant privacy regulations.

Technical Challenges:Risk: Development complexities related to matching algorithms, real-time messaging, scalability, and cross-platform compatibility.Impact: Delays in product delivery, increased development costs.Mitigation: Conduct thorough feasibility studies and prototype testing to identify and address technical challenges early in the development process. Utilize agile development methodologies to iterate and adapt to evolving requirements. Employ experienced developers with expertise in relevant technologies to tackle complex development tasks effectively. Regularly review and optimize algorithms for efficiency and accuracy. Implement scalable architecture and ensure compatibility across multiple platforms to accommodate future growth and user needs.

The image above represents a storyboard example of a user accessing the app, either for the first time or as a returning user. Upon opening the application the user will see a simple login screen where they can either choose to sign up or login. We believe that the orange color represents a vibrant, exciting color for users but not too vibrant that it may be seen as unprofessional. In addition, the logo is not too elaborate so it can be easily read by most users. Should the user press the login button, they will be directed to the login screen where they will be prompted to use the username and password to login into their OkHire account. Should the user press the sign up button, they will be prompted to complete several personal questions including creating a username and password as well as providing their name, email, and phone. This is so that they can be in contact with businesses, should they want to get in touch with you. Finally, before an individual can officially become a user, they are required to fill out professional information, similar to that you would see on any job-search app. The user has the option to upload a resume or CV that matches the required format (PDF or DOC) or they can opt to fill in this information manually. This includes education, qualification, work experience, relevant work experience, honors/achievements, etc. The user is notified that they can change this information at any time by going to user settings.