Staffing of Multiple Projects across Different Locations

TUM School of Computation, Information and Technology and fortiss GmbH, SS 2024

Digital Product Innovation and Development
Prof. Dr. Andrea Stocco
Parisa Elahidoost
Florian Angermeir
Dr. Jannik Fischbach

In Cooperation with itestra GmbH
Noah Kaspereit
Tobias Simon

Fober, Luca luca.fober@tum.de
Leschke, Laura laura.leschke@tum.de
Stierlen, Martin martin.stierlen@tum.de
Yildiz, Selin selin.yildiz@tum.de
Yılmaz, Hande hande.ylmaz@tum.de

I. Introduction and Basics

Itestra is a fast-growing IT company specializing in agile software engineering and delivering custom software systems for digital businesses. With currently approximately 150 employees across 13 locations, itestra is expanding its operations to multiple locations and increasing the number of projects handled. As the company scales, managing staffing across different locations has increased in complexity. The current approach, which relies on manual staffing, has been effective in the past but might reach its limits.

A. Background and Context

Project staffing refers to the process of assigning suitable employees to projects. In doing so, employee skills are matched to required skills of a project. Additional criteria such as employee availability, location and working hours need to be met. As a foundation, information on both project specifications and requirements as well as employee data are required. When approaching us, itestra envisioned a web application to support exactly this process by providing an interface to assist in effectively staffing projects. The solution was expected to provide a supportive tool for staffing. It should provide both an overview over the workforce and project staffing as well as assist in actual staffing processes.

B. Report Objectives and Outline

This report aims to provide both, a detailed description of challenges faced by itestra in managing staffing across multiple locations and outline the corresponding solution. Therefore, this report will first delve into the problem description and provide an overview over specific challenges that required the development of a new solution. It will then discuss key epics, user stories and acceptance criteria to be met as part of the requirements. Based on those, the architecture of the developed web application will be examined. As part of this, technical design, UI/UX considerations and the deployment strategy will be described. Finally, the report will conclude by providing recommendations for future work. This includes recommendations for both the rollout process as well as suggestions for future features to be implemented.

II. Problem description

As itestra continues to expand across multiple locations and the complexity of staffing increases, managing project assignments using the existing process has become more difficult.

A. Current Staffing Landscape

The current status-quo of project staffing includes first that project leads add project specifications to an Excel sheet. As part of this, required number of full-time employees (FTE) and corresponding skill-levels are estimated. Staffing decisions are then made manually by two experienced employees. The matching is made based on employee's skills and project requirements. To date, this approach is successful. However, with increasing numbers of both employees and projects, there is a need for a more scalable and efficient solution to manage project staffing processes.

A significant challenge lies in matching employees to projects in a way that fully leverages their skills to successfully complete projects while additionally considering employee's personal interests and career development goals.

B. Impact on Project Delivery and Employee Development

Our solution aims to address the aforementioned challenges. Thus, operational efficiency as well as employee satisfaction and growth should be increased. First, with more projects to manage, the currently manual process is not scalable. Potential negative impacts on timeliness and accuracy of staffing of the current process are expected to be met by our solution. Further, by aligning project assignments with career goals, employee satisfaction could potentially increase, and professional development goals could be supported.

III. Requirements

For this project to be successful, it was essential to start with a thorough understanding of the customer's situation and objectives. Requirements in the form of epics, user stories and key acceptance criteria were therefore derived resulting in two key epics and corresponding user stories.

A. Epic 1: Centralized Employee and Project Management

First, an overarching requirement identified is the creation of a unified platform for managing employee profiles and project data. This includes not only the overall management of employee profiles and project information, but also the development of an accurate central repository for all staffing-related information. Data accuracy for staffing decisions must be ensured. Subsequently, four key user stories were identified as part of this epic.

(1) An Admin requires being able to create, delete and edit user accounts, so that log-in credentials are generated, and employees are able to access their profiles. Subsequently, acceptance criteria include, that admins can create new employee profiles, edit existing ones, and delete profiles if necessary. Further, admins need to be able to input user details, including E-mail, working time, and working location. The system needs to validate data input.
(2) Employees need to be able to edit their corresponding profiles, including skills-levels, weekly availability and working location. Only then suitable projects may be identified. In consequence, it is necessary, that the employee profile includes fields for skills, location, and availability. Further, profile changes need to be saved in real-time. The system should also validate data input.
(3) Project Managers are required to be able to create and manage project specifications. This includes required skills and corresponding skill-levels, deadlines, and priorities, so that projects can be staffed accordingly. Acceptance criteria thus include a project creation interface which allows the input of project name, required skills, timeline, priority, and location. Project managers then need to be able to edit project details as needed, with changes reflected in real-time.
(4) Admins further require being able to view and manage all projects within the system. Then, progress monitoring and staffing adjustments can be made as needed. Subsequently, admins need to be able to access a dashboard displaying all ongoing projects, including key information such as staffing status and deadlines. Further, admins need to be able to edit project details, assign or reassign employees, and close projects upon completion. The system is required to provide real-time updates on project status and employee assignments.

B. Epic 2: Skill Matching and Project Staffing

Key to successful implementation is the development of a skill matching algorithm and a project staffing interface. Only then can employees accurately be assigned to projects based on their skills, availability, and other relevant factors. Three user stories result as part of this epic.

(1) Admins require the system to suggest best-matched employees for a project based on required skills and availability. Then efficient staffing of projects can be ensured. Resulting are acceptance criteria including the consideration of relevant information such as required skills, employee availability, and location when generating staffing suggestions. The system is expected to present ranked suggestions for each project, allowing admins to manually assign best-fitting employees to projects with one click. The system tracks and logs any changes made in real time.
(2) Project Managers expect the system to consider employee career goals and preferences in the matching process. This is aimed at motivating employees and aligning matching with professional development goals. Therefore, the system needs to allow employees to indicate career goals and corresponding target skill-levels to specific skills. Further, the project staffing interface needs to display employee preferences. Subsequently, project managers will be able to view how well the team composition aligns with employee goals.
(3) Correspondingly, employees need a clear and simple interface to view project assignments and update current and target skill-levels, so that profiles are kept current and aligned with career goals. Therefore, employees need to be able to easily navigate to their profile to update skills, availability, and preferences. The system further needs to provide a clear overview of current and past project assignments.

C. Acceptance Criteria

Resulting from both epics and user stories, three key acceptance criteria need to be met for the implementation to be successful.

(1) The staffing application needs to be accurate and efficient. This includes first an accurate matching of employees to projects based on predefined criteria especially skills employee preferences. Second, all data including employee profiles, project details and staffing assignments must be updated in real-time.
(2) Further, the application must ensure user experience criteria are met. Subsequently, the user interface must be simple and intuitive. Further, ease of use in regard to key actions such as updating profiles, managing projects and making staffing decisions needs to be ensured. Therefore, those key actions must be easily accessible and performable with minimal clicks.
(3) As stated, itestra is a fast-growing company. Resulting from this are scalability requirements. The system must be able to handle growth including an increasing number of employees, projects and locations.

IV. SOLUTION ARCHITECTURE

From our final problem description and corresponding requirements, we were able to derive implications for implementation in the form of the solution architecture. Thus, requirements identified were translated into feasible designs and technical solutions.

A. UI/UX Considerations

First, UI/UX considerations needed to be made. Therefore, as a first implementation step, complex requirements made by the clients needed to be translated into a straightforward application. This translation step was handled in the design tool Figma, where the application was first holistically designed. Over the course of the project and by considering continuous client feedback, in total four versions were iteratively designed.

One criterion to be met was to have a clean and intuitive user-centric design. Therefore, designs needed to focus on ease of use and include powerful functionalities despite simplicity. Users must be able to complete tasks and meet their requirements by taking minimal steps and clicks. Thus, unnecessary complexity was minimized in the design phase.

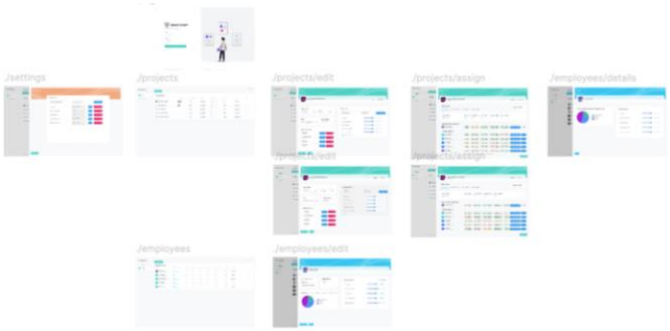

Figure 1: Design Overview of Web Application Design Drafts made in Figma

Furthermore, consistency in visual elements to ensure a user-centric UI/UX was considered. This includes a consistent color scheme to enhance navigation and reduction of cognitive user load as well as standardized iconography, typography and spacing, making the interface both visually appealing and functional regarding user recognition.

B. Technical Overview

The identified requirements and designs were then implemented. Main objectives were to provide a robust technical foundation to ensure scalability, security and performance of the application. As a first step, the requirements were mapped to a UML diagram.

Figure 2: UML Diagram of application solution structure

As Technology Stack, the MERN Stack was implemented. This includes a MongoDB database chosen as a starting point. Employee profiles, project details and staffing assignment are stored in corresponding documents in the database. Express.js serves as the web application framework for Node.js and handles HTTP requests, routing and middleware functions. Therefore, it processes requests, returns responses and connects with MongoDB. The frontend is powered by React.js, a library for building user interfaces. This enables dynamic and reactive single-page applications. UI rendering, user interactions and state management as well as backend API integration are managed in React.js. Lastly, Node.js handles server-side functionality and manages API communication with the frontend.

Accurate state management and data handling are ensured by the implementation of the Redux Toolkit and RTK Queries. The Redux Toolkit ensures centralized storage for all application data to simplify state updates. By doing so, predictable state transitions are ensured. API integration is primarily handled through RTK Query, which enhances data fetching and caching. In consequence, a seamless way to interact with the backend and improved application performance are ensured.

Authentication and Security are handled via JSON Web Tokens (JWT). User sessions are thus securely protected, and access is restricted to authorized users. RESTful endpoints for all key operations, such as managing profiles, projects, and staffing assignments, are exposed by the backend. Those API endpoints following the REST (Representational State Transfer) principles allow for stateless operations.

Lastly, all components from frontend and backend are containerized using Docker. Consistency across development, and production environments is therefore ensured.

C. Deployment, Testing and CI/CD

Measures to ensure a seamless deployment process and successful adoption by the client were undertaken. This includes a testing suite, a continuous integration/continuous deployment (CI/CD) pipeline and separate environments for development and production. Additionally, a comprehensive WIKI is provided.

First, the testing suite comprises both unit and end-to-end (E2E) tests. This ensures comprehensive testing for expected functionality. Backend unit tests verify the functionality of individual components of the backend system. Each component of the application is tested in isolation to ensure correct functionality. Tests are implemented in the testing framework Jest and Mockingdose. Further, E2E tests are conducted to simulate user interactions and real case user scenarios. This ensures the application behaves as expected from the user perspective. Therefore, the entire application stack is tested using E2E-tests which are implemented with Playwright.

Regarding the CI/CD pipeline, GitHub Actions automate the testing and deployment process. Therefore, every change on main in code triggers a series of the previously mentioned automated tests. If passed, the code is deployed to the according environment. Our GitHub repository uses two self-hosted runners: the development runner and the production runner. Both runners are configured on AWS EC2 Ubuntu instances with the type t3. medium (2 CPU, 4 GiB Memory). Branch protection protects the main branch by requiring tests before any changes can be merged. Subsequently, only functional code reaches production.

Separate environments for differing purposes were implemented using two AWS EC2 instances. The development environment with a separate database is built to ensure a safe development space to work on for new features without affecting the live system. The production environment on the other hand is the live application environment. As mentioned, only thoroughly tested and

approved code is deployed here. To ensure consistency between both environments, the development database is reset to the production status every day at 2 AM.

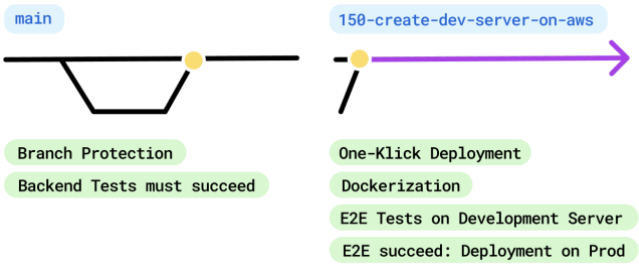

Figure 3: Conceptualized CI/CD Pipeline, One Klick Deploy

Lastly, itestra is provided with a comprehensive WIKI site on GitHub, which offers comprehensive internal documentation. Step-by-step guides for using the system, troubleshooting common issues, and extending the platform's capabilities are handed over as part of this WIKI.

V. Next Steps And Future Work

The handover includes a fully functioning web application addressing the client challenges. However, necessary implementation steps need to be undertaken for rollout. Furthermore, prioritization of client requirements was made resulting in not yet implemented but possibly helpful additional features for future works. Our recommendations on how the client could proceed will be described in the following.

A. Rollout Strategy

As part of Rollout, first migration of existing data needs to be handled. Therefore, a detailed plan for migrating current employee profiles, project data, and other relevant information from existing systems (e.g., Excel sheets) into the new platform needs to be set up. Additionally, data validation to ensure information accuracy and correct mapping to system infrastructure should be conducted.

Further, rollout should begin with the onboarding of admins and project managers, as those are the primary users of the current state of the application. This should include relevant trainings in which functionality is explained, but also feedback collection mechanisms for future development.

In the future after admins and project managers, employees could gradually be introduced to the system. As part of this, data accuracy must be assured. Thus, employees should be motivated to update profiles, manage skills and development goals. Additionally, user adoption rates could be monitored, and additional support should be offered to ensure smooth transitions.

B. Future Outlook and Development

As time and resources were limited, prioritization of functionalities needed to be made. Therefore, further adjustments and additional functionalities may be implemented in the future.

(1) The HR management tool currently used by itestra, Odoo, operates on a PostgreSQL-based database. Therefore, one change could be a transition from MongoDB to PostgreSQL. This change was not made yet due to prioritization from client side. Alternatively, APIs to facilitate smooth data exchange between systems may be created. By doing so, the application could be integrated into existing HR and project management systems currently in use.

(2) The app today is primarily conceptualized to aid itestra employees with project staffing authority in general. A role-based access control (RBAC) system that supports granular user roles could be developed and implemented. Example roles could include HR managers, project managers, and employees. The roles must have tailored access to specific features and data. Additionally, each role may be provided with a customized dashboard that displays the most relevant information and tools for their specific tasks.

(3) To further ease the usage and accessibility of the app, a mobile app enabling employees and project managers to access the system on the go could be developed. This could include mobile access to profiles, project assignments, notifications and the ability to update availability and skills.

(4) Further, advanced analytics and AI may be integrated. Regarding staffing, machine learning models trained to predict staffing needs and anticipate future skill shortages may be implemented. This may include implementing an AI-powered system that suggests optimal team composition and potential project assignments for employees. Privacy and security concerns need to be evaluated. Regarding analytics, the application may be extended to include reporting capabilities. The system could provide detailed analytics on e.g. staffing efficiency, skill development trends and employee utilization. As a next step to reports, dashboards with real-time analytics on ongoing projects could provide advanced analytics capabilities.

VI. New Insights

Over the course of the project, numerous learnings regarding project, team management and technical solution space were gathered. This section will thus provide a comprehensive reflection on the lessons learned.

A. Project Management Insights

Regarding project management, lessons learned range from team dynamics, communication, stakeholder management and agile methodology to translating client requirements into technical specifications.

Efficient team management made regular team meetings, clear role definition and the usage of collaboration tools necessary. In the beginning of the project, we set up two meetings per week. Those meetings were crucial for keeping the team aligned and ensuring everyone was aware of their responsibilities. Additionally, assigning roles such as frontend and backend responsibilities early in the project helped to avoid overlaps. By assigning specific areas of ownership, every team member was aware of their area of responsibility.

Team management was efficiently managed in collaboration tools. Especially, Notion served as a central knowledge base. All project related documentation, including an internal WIKI on both technical implementation details as

well as task management, were stored and updated there. Additionally, a Kanban board was managed in Notion. This allowed for efficient assignment overviews and progress tracking. Additionally, GitHub Project was used. By doing so, sprints were managed effectively.

Furthermore, crucial for success were regular stakeholder meetings. In monthly sessions with our clients from itestra, feedback was gathered, requirements were discussed, and overall client expectations were managed. Additionally, beta versions were shared with itestra to gather comprehensive feedback and enable for testing. Regarding scope and requirement changes, flexibility was necessary. Re-prioritizing features posed a continuous challenge. Further, as some client needs were initially vague, breaking them down into actionable user stories was necessary.

Agile methodology was an additional learning, as the project among others was divided into sprints. Each sprint defined clear goals and deliverables. This approach facilitated trackable progress, which also facilitated client communication.

Lastly, external inputs and feedback from regular meetings with our supervisor Parisa Elahidoost helped us in further improvements.

B. Technical Insights

Technical Insights acquired range over the whole solution architecture. Working with the MERN stack provided hands-on experience in modern web development. From designing frontend UI to managing backend services, the team gained deeper understanding of building a full-stack JavaScript application. This includes implementing state management and data fetching through RTK Query, as well as overall understanding for flows of data from frontend to backend to ensure consistent data.

Setting up a CI/CD pipeline allowed for automated testing and deployment. This facilitated the development process. Further, catching bugs early and ensuring a stable codebase showed the importance of early testing. The team further learned the importance of containerization as the ‘works on my machine’ problem was reduced.

Database management of the NoSQL database also taught us important lessons in handling data migrations.

C. Recommendations for future projects

When reflecting on what could be done differently, challenges faced result in actionable lessons learned. While the project ultimately succeeded, initial stages encountered some challenges. These resulted among others from evolving requirements. In future projects, we would place a stronger emphasis on early requirement gathering. This would ensure a better and clearer understanding of project requirements and stakeholder alignment to reduce reiterations.

Further, as the project scope expanded at times due to changing client needs, we would today implement a clearer process on handling scope management and especially engage in more frequent scope reviews with the clients.

VII. Conclusion

In conclusion, the scalable, centralized platform for project staffing envisioned by itestra is successfully implemented and ready for rollout. The web application deployed addresses the specific requirements identified. Therefore, by implementing our solution, itestra can immediately handle project staffing processes more accurately and efficiently. Additional benefits from our solution include potentially increased employee satisfaction and development goal-oriented project staffing.

However, for the future we recommend further development of the application. Potential future enhancements may include the development of a mobile app, the inclusion of advanced analytics and especially the advancements of the matching algorithm.

Overall, the application sets the foundation for sustainable growth, ensuring itestra can continue to scale effectively.

Acknowledgment

We thank itestra and especially Noah Kaspereit and Tobias Simon for providing us with this real and relevant challenge. We appreciate your help, feedback and discussions in the continuous meetings over the course of this project. Moreover, we thank Parisa Elahidoost for the continuous support and help throughout the course of this inspiring hands-on seminar.

Contribution statement

In the following, you find responsibilities for the key areas of contribution. However, it is important to note, that each team member worked on other areas and supported if necessary.

Area of Contribution	**Responsible Contributor**
Frontend	Hande Yilmaz
Backend, Database	Laura Leschke, Martin Stierlen
CI/CD Pipeline	Luca Fober
Testing	Laura Leschke, Selin Yildiz Luca Fober, Hande Yilmaz
Presentations, Documentation	Selin Yildiz, Luca Fober
Requirements, Design	Selin Yildiz, Luca Fober, Martin Stierlen
Report	Selin Yildiz