1. Introduction

The University of Eswatini houses a refectory which serves a variety of generally affordable food products to a large population of staff and students. However, the refectory has difficulties providing accessible and reliable meal services for this population. The availability of some meals is usually in question, and this makes it difficult for students to plan their meals effectively. At times, students and staff miss out on meals due to the running out of food whilst they are in class. Also, apart from at the actual physical location of the refectory, there is no visible price list of the food products, which affects individuals' ability to properly budget and make informed meal choices.

To address these issues, there is a need for the implementation for a digital refectory ordering system. A web-based one would offer the opportunity for the whole population to access it with ease. It would enable the population to view a regularly updated menu of meals. The system would give the capability for staff and students to place orders in advance which would alleviate the problem of food scarcity during busy hours whilst ensuring collection at their convenience. The implementation of this web-based refectory ordering system would significantly enhance the accessibility and predictability of food options by students and staff at the refectory.

2. User/client involvement

Client – University of Eswatini
Users – Students and staff who are consumers at the University of Eswatini refectory

Client

- Provision of comprehensive menu of items with their images, descriptions and pricing. Provided at the beginning of the project, and should be updated regularly.
- Feedback on designs, prototypes and clarification where required. This kind of support should be provided throughout the project when validation is being done in every stage.
- Testing of system functionalities, user interface and user experience to ensure it meets client expectations.
- Personnel who will act as administrators of the system.
- Hardware which will support the system (server for hosting system), computers which refectory staff will use as administrators of the system.

Users

- Accurate personal details. This occurs once the system has been developed.

- Feedback on user interface and user experience of system. This happens after the design phase and also during the testing phase.
- Continued positive use of system within university guidelines
- Reporting of bugs and issues that may occur. This occurs during the lifetime of the system.

3. **Risks**

Categories	**Risks**	**Risk mitigation strategies**
Schedule risks	Delayed gathering of requirements caused by insufficient or delayed information from university may cause a delay in the design and implementation process.	Allocating more time for requirements gathering in order to be thorough, engaging regularly with stakeholders
	There may be challenges integrating new system with existing university databases, resulting in timeline being extended.	Have a separate database for the new system, independent from current university databases
	Testing may not be enough due to unforeseen prolonging of implementation, which may lead to compromised quality of system.	Creation of comprehensive testing plans Testing after each development phase
	Continuous addition of new features and requirements may result in project delays.	Communicate agreed-upon scope Enforce strict evaluation of features before implementation
Technical risks	The selected technologies may be incompatible with university's existing infrastructure. Readjustments may consume additional resources and time.	Thorough compatibility assessment early in the project Consultance of university IT experts
	The system may have loopholes which make room for security breaches. This may compromise the user data and integrity of the system.	Implementation of robust security measures throughout development lifecycle Conduct regular security testing
	System may not be able to perform optimally when user traffic is high, which may lead to system lagging or crashes.	Perform scalability and performance testing Monitor system for bottlenecks, consider load balancing methods Optimize code and architecture for efficiency
	System may not be able to handle future increases in the user load, and this may lead to the need for frequent upgrades.	Design with scalability in mind Use scalable architectures and technologies
Financial risks	The need for additional resources at a certain stage may lead to unforeseen costs. This may force reallocation of funds from other stages, which may then compromise the whole project.	Conduct comprehensive resource planning in early stages Create contingency budget Monitor resource utilization
	There may not be enough resources allocated for development and testing, leading to potential compromise in quality and delay in completion.	Conduct thorough resource estimation at each stage Allocate resources based on accurate estimations Re-evaluate resource needs, adjust accordingly
Operational risks	There may be resistance from staff and students to use the system, which would affect the project's success.	Conduct thorough user involvement and feedback sessions Provide user-friendly interfaces

		Provide incentives for first users Offer training and support during transition
	Training for users may be insufficient, leading to operational inefficiency or decreased usage over time.	Offer training and support during transition Gather feedback for improvement
	Changes in the market trends may affect the rates of system use, causing it to be obsolete.	Design system with flexibility and adaptability in mind Plan for regular system updates and align system with changing needs
Legal risks	The system's data handling techniques may fail to comply with data protection guidelines, which may result in fines and other legal action.	Conduct regular audits to ensure compliance Involve experts to review and validate data handling techniques
	There may be disputes regarding the deliverables of the project, and this may result in delay of project timeline.	Establish clear and detailed scope and deliverables in the initial project documentation Regularly communicate with stakeholders Document agreements and changes
	Unauthorized use of intellectual property could result in legal disputes.	Conduct regular audits to detect and prevent unauthorized use
Human resources risks	Key team members may leave the project or be complacent when doing the job, resulting in delays.	Cross train team members to ensure knowledge sharing Thoroughly document processes and knowledge for smoother transitions Develop strategies for keeping members motivated
	There may be insufficient expertise to cover certain areas of the project, resulting in a steep learning curve for other members and potentially compromised quality in the output.	Provide training and mentorship to bridge knowledge gaps Encourage knowledge sharing among team members
	There may be misunderstandings between personnel and stakeholders due to lack of communication, resulting in delays and misaligned expectations.	Establish clear communication channels and protocols Document all decisions and agreements Conduct regular meetings

4. Standards, guidelines and procedures

a. Model

Of the numerous models that exist, we saw it fit that the Waterfall model be used for this system. It is a software development model characterized by its sequential approach. All activities are planned before the commencement of the software development, with each phase documented and approved before the next phase begins.

There are numerous factors which informed our choice of going with the waterfall model. The model is suitable for systems which have requirements that are well-defined and understood. Most, if not all, requirements and expectations of this ordering system are clear, with the proposed functionalities of the system well-defined and explained. Its clarity and structure makes it easy to track the progress of the project, which simplifies the process of setting well-defined milestones and timelines as the project manager. The model's emphasis on documentation in every stage makes it easy to provide feedback to stakeholders in order to make changes before moving forward, creating a rapport between stakeholders and the developing team.

In addition to that, the model is the most common method of developing software having been developed decades ago, so its adoption by the team developing the system will present just a narrow learning curve. Also, our team is comprised of very few people, so easy adoption of the methodology will mean there will be more time to focus on how the tasks will be shared amongst the existing members.

Using this methodology allows both stakeholders and developing team to be able to comprehensively identify and manage potential risks early on in the software development process, which minimizes the chance for unforeseen changes down the line.

5. Organization of the project

The following table outlines the different personnel working on the project and the different roles that they take up. Lack of adequate personnel means that the existing personnel have been assigned multiple secondary roles. Training for these roles will take place on the first week.

Member	**Primary role**	**Secondary role(s)**
Mayenziwe Maseko	Project manager	Front-end programmer, back-end programmer, UI designer
Nosisa Dlamini	UI designer	UX researcher, quality assurance officer
Melusi Magagula	Front-end programmer	Technical writer
Sinethemba Mndzebele	Architect	Back-end programmer, technical writer

The following table provides a guide as to what is expected from the personnel in the different roles that they were assigned to:

Role	**Responsibilities**
Project manager	1. Oversees the entire project, ensures goals, schedules and quality standards align 2. Facilitates communication, resolves conflicts and supports team members 3. Sets objectives, allocates tasks and ensures the project meets stakeholders' expectations
UI designer	1. Focuses on the user interface design, creating visually appealing and intuitive interfaces 2. Designs layouts, typography, color schemes and visual elements for the system 3. Ensures consistency and usability across all user elements
UI researcher	1. Conducts user research to understand user behaviors, needs and concerns 2. Creates user personas, user journey maps and conducts usability testing 3. Provides insights to enhance the user experience based on research findings
Front-end programmer	1. Implements UI designs and functionalities on the client side 2. Utilizes HTML, CSS, JavaScript to create responsive and interactive user interfaces
Back-end programmer	1. Manages server-side logic, databases and application integration 2. Develops APIs, handles data storage and ensures system functionality and security on the server
Technical writer	1. Creates technical documentation, guidelines and manuals for the system 2. Ensures comprehensive and understandable documentation for developers, users and other stakeholders
Architect	1. Designs the overall structure and framework of the system 2. Focuses on scalability, performance, security and integrations between components 3. Selects appropriate technologies and sets guidelines for development based on the system's requirements
Quality assurance officer	1. Ensures system meets specified requirements and quality standards. 2. Develops and executes test plans which enable testing to identify and resolve issues before deployment.

6. Project phases

	11 Dec – 17 Dec							18 Dec – 24 Dec							25 Dec – 31 Dec							1 Jan – 7 Jan							8 Jan – 15 Jan						
Phase	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
Requirements Analysis and definition	🟦	🟦	🟦	🟦	🟦	🟦	🟦																												
System and software design								🟨	🟨	🟨	🟨	🟨	🟨	🟨																					
Implementation and unit testing															🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩							
Integration and system testing																									🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Operation and maintenance																									🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪

7. Requirements analysis and design

a. <u>Requirements analysis</u>
   i. Gathering requirements
      1. Interviews with university management, information will be recorded on notebooks.
      2. Surveys distributed to staff and students to gather broader feedback, with the use of Google Forms.
   ii. Documentation of findings
      1. Use of use case diagram to show proposed interaction of user and system. Visual Paradigm will be used.
      2. Requirements workshops with the project team to refine and finalize the requirements. Whatsapp and face-to-face meetings will be used.
   iii. Validation of documentation
      1. Blended communication will be use to give stakeholders feedback on documentation.

b. <u>Design</u>
   i. Architecture design – diagrams like DFD, context diagrams, use case diagrams, UML diagrams etc. will be used to show the system's architecture. Tools like Visual Paradigm and ArgoUML will be used.
   ii. Database design – Entity-Relationship(ER) diagrams will be used to document the manner the database is structured, as well as how the data relates to each other.
   iii. User interface – wireframes and mockups will be constructed to show the visual layout of the system using a design tool called Figma.

8. **Implementation**
    a. Platform – this will be a web-based system
    b. Coding
        i. Front-end
            1. HTML for structuring the web content
            2. CSS for styling
            3. JavaScript as the client-side scripting language
        ii. Back-end – the system back-end will be developed using the LAMP stack
            1. Linux as the operating system
            2. Apache as the web server
            3. MySQL as the database
            4. PHP as the server-side scripting language
        iii. Tools
            1. IDE: Visual Studio Code
            2. Version control: GitHub
9. **Testing**

Our test area will be a dedicated computer with Apache server hosting the web application. Devices (laptops, tablets and smartphones) connected to the same network access the web app via a link and utilize the web app in an attempt to ascertain that it carries no bugs, it is robust, available and even secure as possible.

Testing procedures to be followed

After the test plan is drawn up, here is how testing in the test area will be done:

a. *Unit testing:*

- Individual units of code (e.g., functions, modules) shall be tested in isolation to ensure they work as expected. Black-box testing will be used.

b. *Integration testing:*

- Individual units of code will be combined and tested together to ensure they work together correctly, here are the specifics:
    - User interface testing - the tester tests each interface function
    - Use scenario testing - the tester tests each use scenario
    - Data flow testing - tests each process in a step-by-step fashion
    - System interface testing - tests the exchange of data with other systems

The order in which components are integrated and tested has to be stated.

c. *System testing:*

- The entire software system will be tested as a whole to ensure it meets all the requirements. This may include functional testing and non-functional testing, respectively:
    - Requirements testing - tests whether original business requirements are met.
    - Usability testing- tests how convenient the system is to use.
    - Security testing- tests disaster recovery and unauthorized access.
    - Performance testing- examines the ability to perform under high loads.
    - Documentation testing- tests the accuracy of the documentation.

d. *Acceptance testing:*

- The software will then be tested by the end users or stakeholders to ensure it meets their needs and expectations, here are the criteria:
    - Alpha testing - conducted by users to ensure that they accept the system
    - Beta testing - uses real data, not test data.

Crucial Tasks

Defect fixing:

- Any bugs or defects that are found during testing are fixed by the development team in the development area.

Regression testing:

- After defects are fixed, the web app is re-tested in the test area to ensure that the fixes have not introduced any new problems.

Once all the program and system tests have been run and passed, programs will be copied into the production area—the location where the final system will reside.