

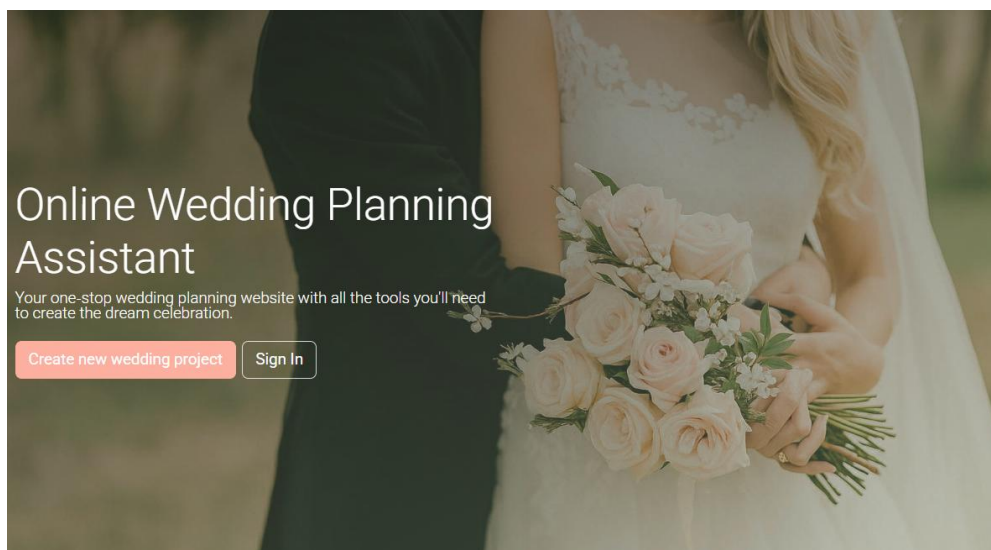


УНИВЕРЗИТЕТ “СВ. КИРИЛ И МЕТОДИЈ” ВО СКОПЈЕ
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО



Проектна задача по предметот:
Напреден Веб Дизајн

Тема: Wedding-Planner App



Изработиле:

Мартина Петковска 223313

Мартина Василевска 211147

Јован Александроски 211248

Предметен наставник:

д-р Бобан Јоксимоски

Мила Додевска

Скопје, 2025 година

Содржина:

Апстракт	4
1. Вовед	5
1.1 Позадина и Мотивација.....	5
1.2 Цели.....	5
2. Процес на имплементација и тешкотии.....	6
2.1 Тек на имплементацијата	6
2.2 Предизвици и потешкотии	6
3. Анализа на барања	7
3.1 Користени технологии.....	7
3.2 Функционални барања.....	7
3.3 Нефункционални барања	8
3.4 User Stories	8
3.5 Ограничувања и претпоставки	8
4. Дизајн на системот.....	9
4.1 Преглед на Архитектура	9
4.2 Frontend дизајн	9
4.2.1 Цел и принципи на дизајнот	9
4.2.2 Структура и навигација	9
4.2.3 Палета на бои и типографија	10
4.2.4 Почетна страница (home page).....	11
4.2.5 Страница за задачи (Task-management page).....	12
4.2.6 Страница за буџет и финансиски преглед (budget-tracker page)	14
4.2.7 Страница за допавувачи (Vendors page)	14
4.2.8 Страница за управување со гости (guest-list page).....	15
4.2.9 Страница за детали на свадбата (wedding-details page).....	17
4.2.10 Респонзивен дизајн	20
4.2.11 Динамички компоненти и сервис слој	21
4.2.12 Тестирање на фронтендот	21
4.3 Backend дизајн.....	22
4.3.1 Технолошки стек и архитектонски стил.....	22
4.3.2 Модули/сервисни граници	22

4.3.3	Модел на податоци и перзистенција	22
4.3.4	Дизајн на API (ресурси, верзионирање, JSON)	23
4.3.5	Автентикација и авторизација	23
4.3.6	Валидација и обработка на грешки	24
4.3.7	Бизнис-логика и сервис слој	24
4.3.8	Интеграции со надворешни системи	24
4.3.9	Кеширање и перформанси	24
4.3.10	Конфигурација и околина	24
4.3.11	Логирање и мониторинг	25
4.3.12	Тестирање на backend	25
4.3.13	Безбедност	25
4.3.14	Деплојмент и CI/CD	25
4.3.15	Скалабилност и идни подобрувања	25
5	Интеграциски слој	26
5.1	Подготовка за интеграцијата	26
5.2	Примена на CORS конфигурација	26
5.3	Замена на хардкодираните податоци со HTTP барања во frontend	27
5.4	Интеграција на сервисите и функционалностите	28
5.5	Улога на интеграцијата во развојниот процес	29
6	Тестирање	29
6.1	Цели на тестирањето	30
6.2	Тестирање на backend со Postman	30
6.3	Тестирање на Frontend	30
5.4	Интеграциско тестирање	31
5.5	Тестирање на корисничкото искуство	31
5.6	Решени проблеми и потешкотии	31
7	План за одржување	32
8	Заклучок	33

Апстракт

Овој проект претставува интернет апликација - Wedding Planner App, наменета да им овозможи на корисниците една едноставна и интуитивна платформа каде ќе можат брзо и лесно да го испланираат најзначајниот настан од нивната љубовна приказна.

Апликацијата е веб-базирана и користи Angular за frontend и Spring Boot за backend. Со неа корисниците можат да креираат персонализиран свадбен планер, каде можат да внесуваат и организираат задачи и временски распоред на настани. Дополнително, системот овозможува раководење со својот буџет и следење на неговото трошење, внесување податоци за услужни обезбедувачи, што ја олеснува комуникацијата и преговорите, како и управување со листа на гости и RSVP потврди.

Фокусот на апликацијата е да обезбеди корисничко искуство кое е лесно за користење, но доволно моќно за комплетно планирање. Реализацијата на овој проект вклучува и интеграција помеѓу frontend-от и backend-от, тестирање на клучните функционалности, како и респонзивност за максимално позитивно корисничко искуство. Резултатот е модерна и функционална веб – апликација, што може да се прошири со идни надградби, со цел да се зголеми удобноста и достапноста за корисниците.

1. Вовед

1.1 Позадина и Мотивација

Планирањето на свадба може да биде доста комплексно и стресно искуство за идните младоженци. Тие треба да координираат различни аспекти, како што се списоци на гости, буџет, распоред на настани, избор на услужни добавувачи (фотографи, декоратори, музика, кетеринг...). Исто така ова искуство може да биде и доста скапо, доколку се изнајми организатор на свадби. Со дигитализација и сè поголема употреба на веб апликации, постои потреба од платформа која ќе им овозможи на корисниците лесно и централизирано да управуваат со сите овие аспекти.

Мотивацијата за избор на оваа тема произлегува од согледувањето на проблемот колку е тешко и предизвикувачко планирањето на свадби, како и од лични интереси кон развој на веб апликации и full-stack проекти и желбата да се создаде практичен алат кој ќе им го олесни целокупното свадбено искуство на корисниците.

1.2 Цели

Целта на апликацијата е да обезбеди иновативно решение за планирање на свадби, кое ќе им овозможи на корисниците:

- Да го креираат текот на настанот и да раководат со задачи
- Да поставуваат и следат буџет, како и трешоците подредени во различни категории
- Да внесат податоци за добавувачи на услуги на едно место
- Да управуваат со листата на гости, RSVP потврди, како и распределба по маси и план мени за секој гостин.

2. Процес на имплементација и тешкотии

2.1 Тек на имплементацијата

Процесот на имплементација започна со споделување на идеи и определување на проблем за кој подоцна ќе се изнајде решение кое ќе биде практично и користено од стана на корисниците. По дискусија на повеќе можни теми, се согласивме дека свадбеното планирање може да биде доста фрустрирачко и да одзема доста време и енергија. Со оваа идеја почна развивањето на full-stack веб апликација со која ќе понудиме решение кое ќе има реална примена.

Откако ја дефиниравме темата, ги определивме главните функционалности кои се потребни за максимално позитивно кориснично искуство. Се фокусиравме повеќе и на автентификација и авторизација на корисниците за максимална безбедност. По регистрација, корисникот избира улога: невеста, младоженец, организатор на настани или корисник и потоа продолжува со планирањето на свадбата, каде на една платформа сè е лесно достапно и убаво организирано. Ги разгледавме и нефункционалните барања како безбедност, едноставен и разбирлив кориснички интерфејс и стабилност.

Следеше одредување на технологиите и поделба на одговорностите. Се одлучивме да користиме Angular за целосна имплементација на frontend и SpringBoot за backend-от. Првично креиравме GitHub репозиториум, со 2 независни гранки за frontend и backend. Се работеше континуирано и на двете гранки, додека не се постигне посакуваната крајна верзија на апликацијата.

Имплементацијата започна со backend – се дизајнираше моделот на податоци и главните ентитети како: „User“, „Budget“, „ItineraryItem“, „Task“, „Vendor“, „WeddingDetails“, како и соодветните контролери за тоа. Фронтендот започна со статички податоци и потоа целосно ги имплементиравме изгледот и функционалностите. Следуваше поврзување на backend-от со frontend-от, бришење на хардкодираните податоци од frontend и нивна замена со HTTP барања кон backend, како и дополнување на service фајловите. Во backend-от се овозможи CORS конфигурација за да се овозможат повици од frontend. Наизменично се тестираа функциите во Postman и целата апликација се прикачи на посебна гранка на GitHub – integration, која се користеше додека да се средат сите неочекувани грешки и bugs. Потоа целата апликација се премести на гранката main. Целосната комуникација на тимот се одвиваше преку Discord, каде правевме состаноци за да го следиме процесот и напредокот.

2.2 Предизвици и потешкотии

Процесот не помина без предизвици. Еден од првите проблеми беше запознавањето и учење на нови технологии како Angular. Друд значаен проблем беше конфигурацијата на CORS и интеграцијата, но успеавме брзо да надминеме над овие потешкотии.

Имаше и организациски потешкотии – различни распореди и лични обврски доведоа до одложувања во завршувањето на одредени функционалности.

Техничките потешкотии вклучуваа проблеми со форматирање на датуми и времето (1pm -> 13:00), прикажување на динамички листи на гости и добавувачи, како и интерфејсот да биде респонзивен. Овие предизвици ги надминавме со внимателно тестирање, дополнителни стилови и функции.

И покрај овие потешкотии, процесот на имплементација беше успешно завршен. Континуираната комуникација, како и користењето на GitHub за верзиска контрола овозможија сите главни функционалности да бидат навремено имплементирани и тестирани. Проектот ни овозможи да стекнеме вредно практично искуство со работа на реални full-stack апликации, подобрување на нашите технички вештини и решавање на проблеми.

3. Анализа на барања

3.1 Користени технологии

За развој на апликацијата се користеа современи и широко користени технологии кои овозможува брза и ефикасна имплементација:

- **Angular** – како фронтенд framework за изградба на динамичен и интуитивен кориснички интерфејс. Angular ни овозможи лесно управување со компонентите, навигација меѓу страните и поврзување на backend-от преку HTTP барања.
- **Spring Boot** – како backend framework за креирање REST API и ракување со логиката. Spring Boot го поедностави развојот на сервиската страна со автоматска конфигурација и поддршка.
- **SQL Server Management Studio (SSMS)** - алатка за дизајн, управување и одржување на базата со податоци.
- **Postman** – за тестирање на API повици, проверка на одговорите и осигурување дека комуникацијата frontend-backend работи безпречено.
- **Dev интеграција (CORS конфигурација)** – за овозможување конекција меѓу frontend и backend при развој на различни порти, што овозможи независен развој и тестирање на двата дела од апликацијата.

3.2 Функционални барања

1. **Автентификација и авторизација на корисници** – корисниците можат да се регистрираат, да се најават и да го управуваат својот профил. Дополнително при регистрација можат да изберат улога на корисник.
2. **Управување со гости** – додавање, бришење и изменување на гости, како и целосна листа на гости кои може да се сортираат, управување со RSVP потврди и прилагодување на распоредот на седење.
3. **Управување со буџет** – поставување на буџет за свадба, внесување на трошоци, подредени по категории, поставување лимит за секоја категорија, како и автоматско следење на преостанатите средства.

4. **Управување со добавувачи** – додавање и пребарување на услужни добавувачи за свадби (фотографи, музика, кетеринг, слаткари, декоратори...) и нивна контакт информација.
5. **Планирање на задачи** – можност за додавање и менување на задачи, подредени по различни категории. За полесна организација системот овозможува и филтрирање по задолжено лице, сортирање по приоритет, како и поставување потсетник и краен рок за извршување на задачата.
6. **Детали за свадбата и тек на настанот** – внесување на основни податоци за свадбата – како име и презиме на парот и датум и локација на церемонијата. Исто така апликацијата овозможува внесување на текот на настаните, подредени по време.

3.3 Нефункционални барања

Нефункционалните барања ја опишуваат квалитетната страна на апликацијата:

- **Перформанси** – апликацијата треба да се вчитува брзо и да обработува барања без значителни задоцнувања.
- **Употребливост** – системот треба да подржува зголемен број на корисници и податоци без поголеми промени
- **Прилагодливост** - интерфејсот треба да биде интуитивен, разбирлив и лесен за користење
- **Безбедност** – сигурно складирање на кориснички податоци, заштита на лозинки и корисничка безбедност

3.4 User Stories

„Како невеста сакам да додадам гости во мојата свадбена листа за да можам да ги следат RSVP потврдите и лесно да ги распределам гостите по маси за седење“.

„Како организатор на свадби, сакам да раководам со сите свадбени детали и настани на едно место, за ефикасно да ги координирам задачите, буџетот и добавувачите за моите клиенти.“

3.5 Ограничувања и претпоставки

- Апликацијата е веб – базирана и не вклучува мобилна верзија.
- Корисниците се очекува да имаат интернет конекција за пристап до апликацијата
- Сите податоци се чуваат во релациона база на податоци
- Не се вклучени автоматски известувања преку е-пошта или SMS во оваа верзија на проектот
- Не е вклучен AI асистент кој би можел да им помогне на корисниците со корисни совети.

4. Дизајн на системот

4.1 Преглед на Архитектура

Системската архитектура на апликацијата е базирана на клиент-сервер модел со јасно одвоени слоеви за frontend и backend. Frontend-от овозможува динамичен и респонзивен кориснички интерфејс, како и едноставно управување со компоненти и рутирање помеѓу страниците. Backend-от овозможува сигурна и стабилна платформа со веб сервиси и одработена деловна логика.

Овој пристап овозможува модуларност, лесно одржување и можност за идно проширување на системот.

4.2 Frontend дизајн

Frontend делот од апликацијата обезбедува модуларна структура, брзо рендерирање на компонентите и лесно одржување. Целта е да се овозможи интуитивно и визуелно привлечно корисничко искуство, со едноставна навигација и функционалности кои ги задоволуваат потребите на корисниците при планирањето на свадби.

4.2.1 Цел и принципи на дизајнот

При планирањето на фронтендот се водеа неколку главни принципи:

- **Употребливост** – лесно разбирлив и достапен дизајн
- **Козистентност** – Сите страници користат хармонизирана палета на бои, стилови за копчиња и форми и типографија
- **Респонзивен дизајн** – Оптимизирано за изгледа и функционира на екрани со различни големини
- **Интерактивност** – динамично ажурирање на податоците

4.2.2 Структура и навигација

Навигацијата на апликацијата е имплементирана со routing, што овозможува лесно движење помеѓу клучните страници.

Секој кориснит треба да си има свој профил, што овозможува персонализирано искуство и лесен пристап до потребните информации. При прв пристап до апликацијата, корисникот е пренасочен кон **страницата за регистрација**, каде што треба да внесе основни информации (име, е-пошта, лозинка). По успешна регистрацијата корисникот се пренасочува кон **страницата за избор на улога** (bride, groom, event planner, other), па потоа на **страницата за најавување**.

Слика 1: Регистрација

Слика 2: Избирање на улога

Слика 3: Најавување

По успешно најавување, корисникот има пристап до целосната апликација. Главното мени е сместено во горниот дел од интерфејсот и содржи линкови до сите страници. Во десниот агол е сместено копчето за профилот на корисникот со можност за одјавување.



Слика 4: Навигациско мени

4.2.3 Палета на бои и типографија

Дизајнот содржи нежна и елегантна палета на бои, инспирирана од свадбени теми, со нежни пастелни тонови за позадина и контрасни темни акценти за текстот и копчињата.

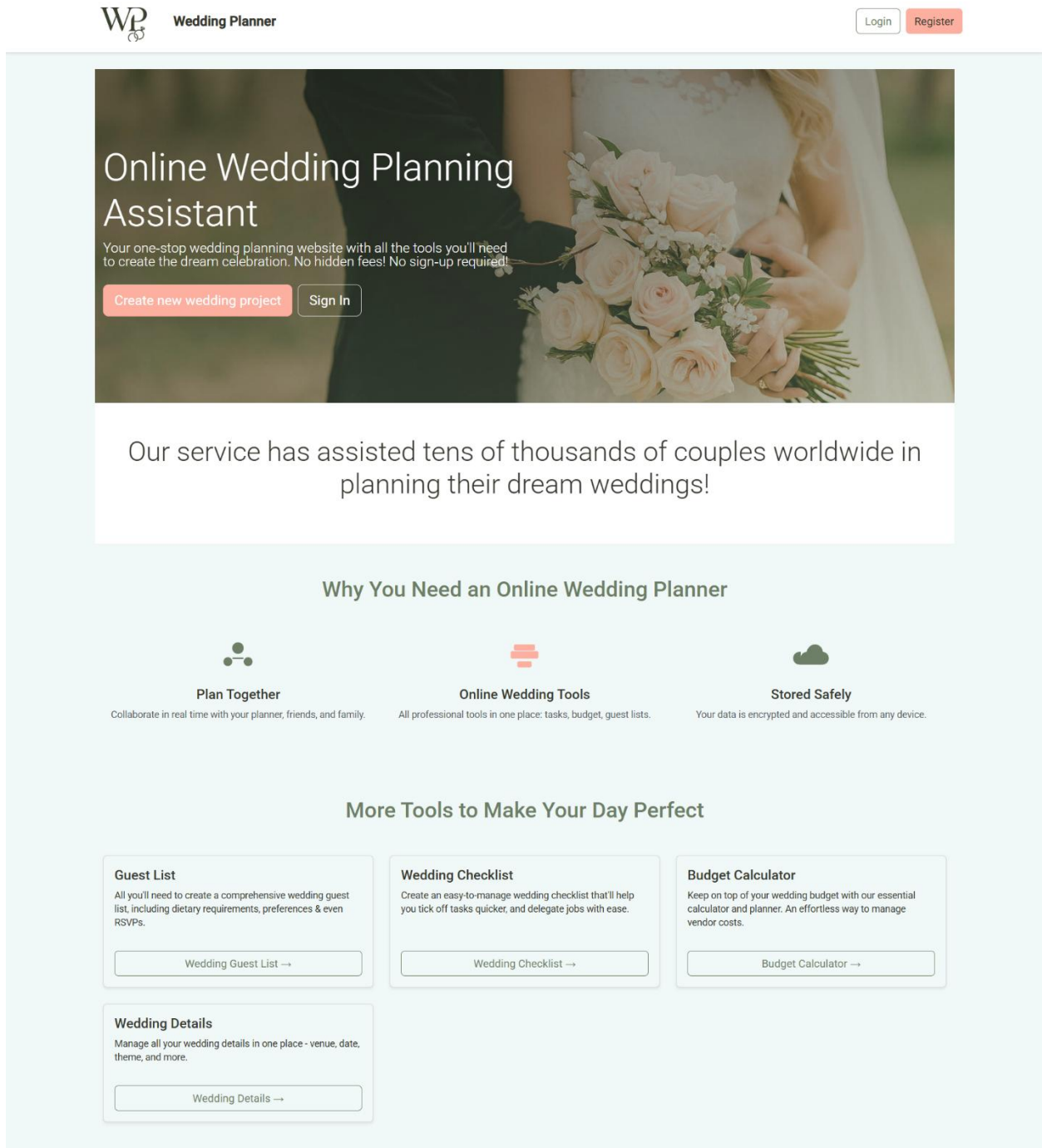


Слика 5: Палета на бои

Примарен фонд кој се користеше е имплементиран од Google Fonts во index.html и е конфигуриран во styles.scss - **Roboto**, **"Helvetica Neue"**, sans-serif. Со оваа типографија се постигна едноставна и разбирлива содржина, и се создаде баланс меѓу елеганција и читливост.

4.2.4 Почетна страница (home page)

Прва точка на контакт за корисниците, која и воедно вклучува и краток вовед во функционалностите на апликацијата. Таа содржи и брзи линкови и копчиња до останатите страници.



Слика 6: home page

4.2.5 Страница за задачи (Task-management page)

Следната страница им овозможува на корисниците да креираат, управуваат и менуваат со сите задачи поврзани со планирање на настанот. Корисникот може да додава нови задачи со детали како наслов, опис, рок, приоритет и да ја зададе задачата на одредена личност. Освен тоа може да се зададе и потсетник на секоја задача.

Постојаните задачи се прикажани во организирана листа со можност за уредување или бришење. Исто така е овозможено додавање на подзадачи (subtasks) под секоја задача.

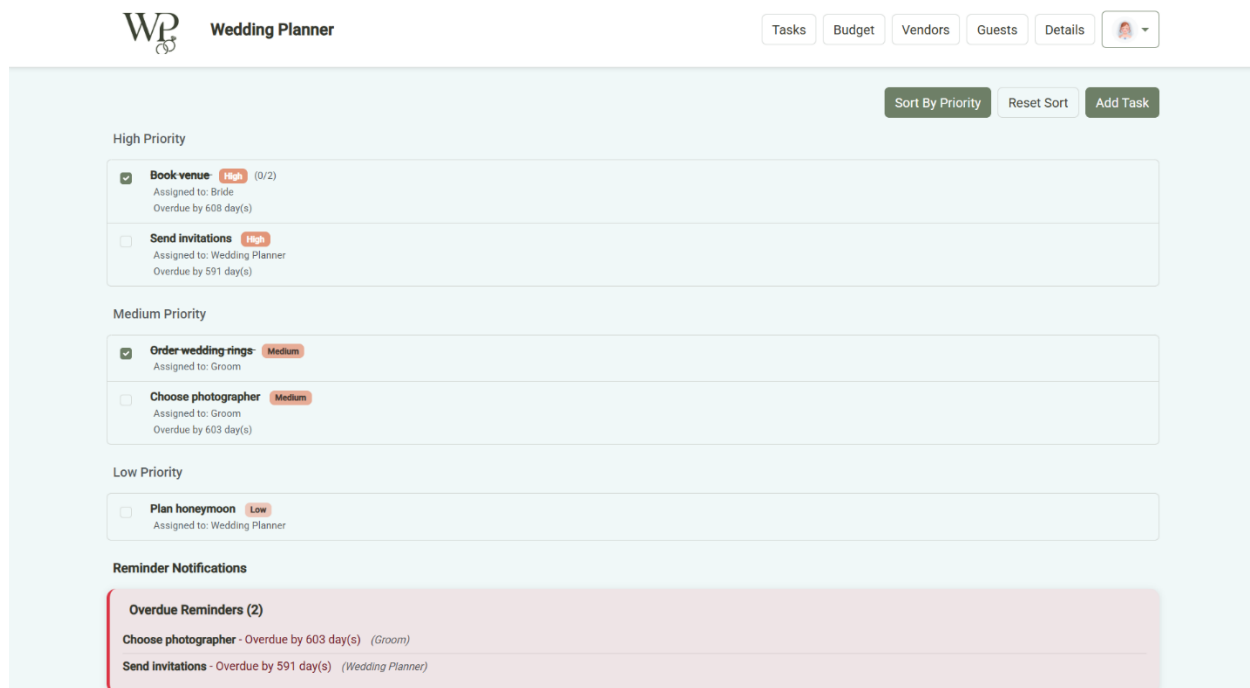
Статусот на секоја задача може да се менува во завршена/не завршена и според него под листата со задачи има прогрес бар кој визуелно им претставува на корисниците колку задачи имаат завршено.

The screenshot displays the 'Wedding Planner' interface. At the top, there's a navigation bar with tabs: Tasks, Budget, Vendors, Guests, Details, and a user profile icon. Below this, a 'Sort By Priority' button and a 'Close' button are visible. A filter section allows users to 'Filter by assigned person' with a dropdown menu set to 'All tasks'. The main task list has columns for Title, Due date, Priority, and Assign to. Below these columns are input fields for 'Task title', 'mm/dd/yyyy' for the due date, a priority dropdown set to 'Medium', and a 'Select person' dropdown. There are 'Add' and 'Cancel' buttons. A description field with 'Optional description' is also present. A checkbox for 'Enable reminder' is shown. The task list includes: 'Book venue' (High priority, 3/3 subtasks, assigned to Jane Smith, overdue by 608 days), 'Choose photographer' (Medium priority, 0/2 subtasks, assigned to John Doe, overdue by 603 days), 'Order wedding rings' (Medium priority, assigned to Jane Smith), 'Send invitations' (High priority, assigned to Event Planner, overdue by 591 days), and 'Plan honeymoon' (Low priority, assigned to John Doe). A progress bar at the bottom shows 'Overall Progress' at 40%, with '2 completed out of 5 total tasks'. A 'Reminder Notifications' section highlights 'Overdue Reminders (2)' with details for 'Choose photographer' and 'Send invitations'.

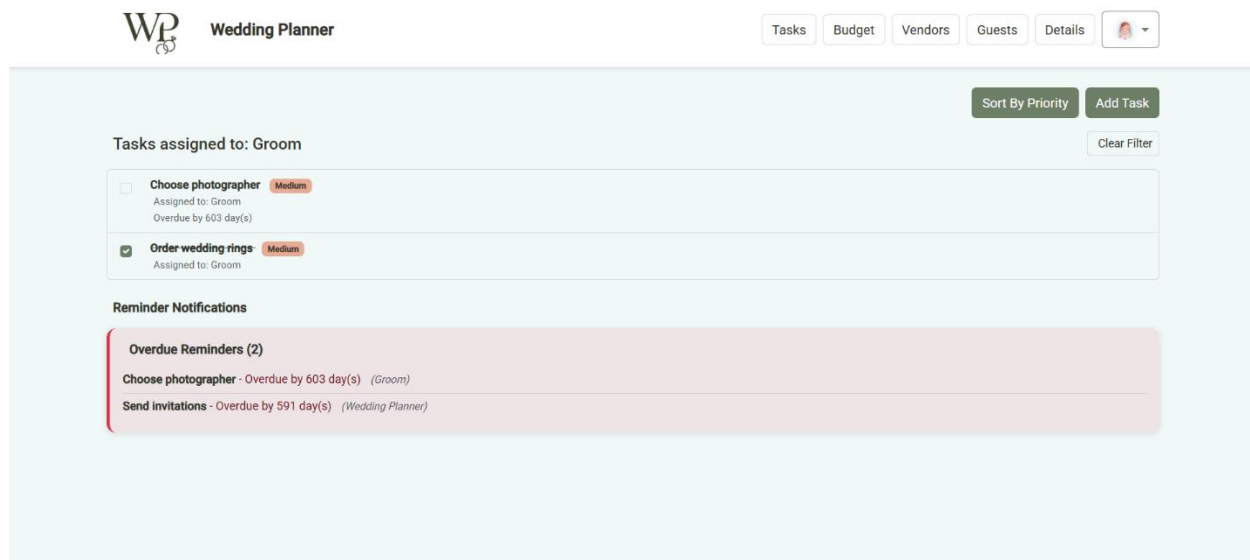
Слика 7: task-management page

Дополнителни функционалности на оваа страница, кои допринесуваат за подобра организација се:

- Филтрирање по задолжено лице
- Сортирање по приоритет
- Поргрес бар
- Потсетник за задачи кои не се завршени, но им поминал рокот.



Слика 8: Сортирање по приоритет

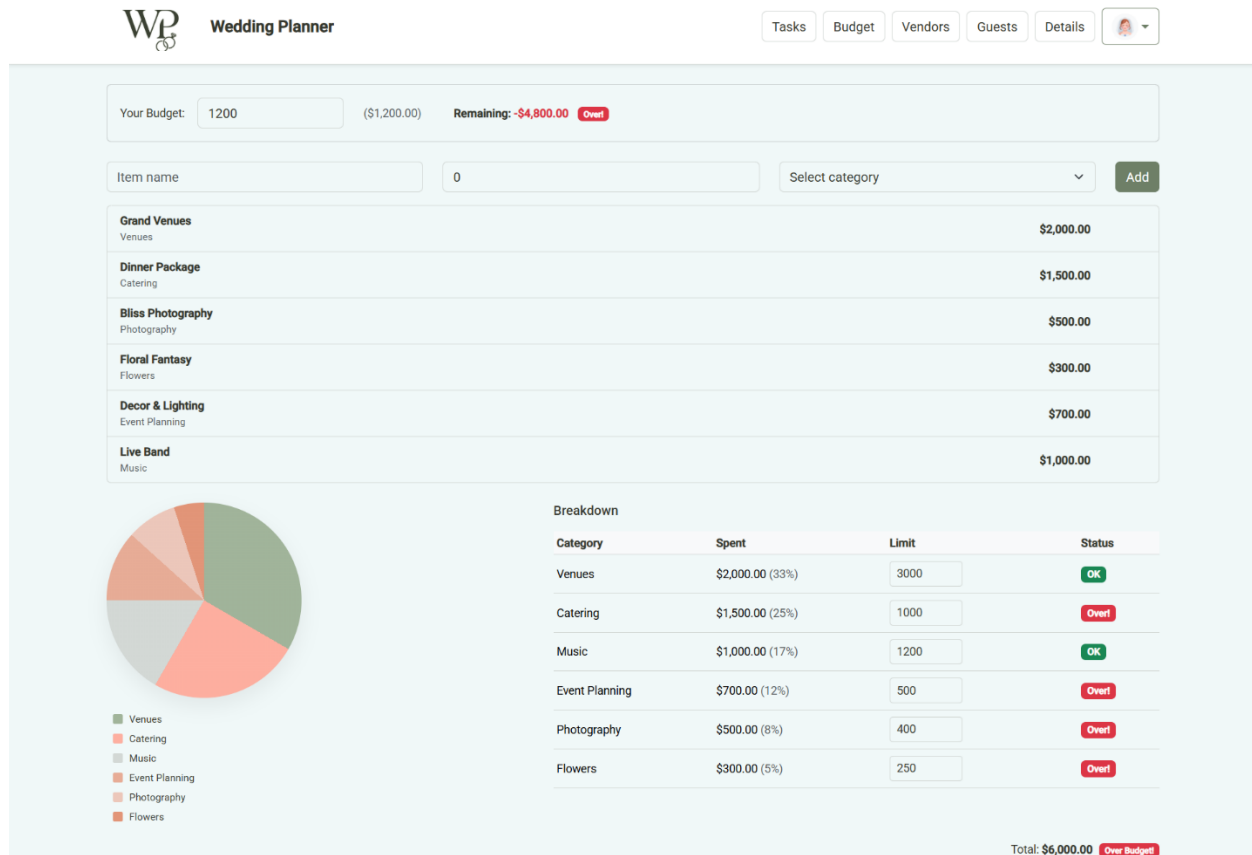


Слика 9: Филтрирање по лице

4.2.6 Страница за буџет и финансиски преглед (budget-tracker page)

Овој дел им овозможува на корисниците да внесуваат трошоци и автоматски да го следат преостанатиот буџет. Страницата содржи:

- Форма за поставување на буџет
- Формулар за додавање нов трошок со категорија и износ
- Визуелизација на потрошувачката преку пита – дијаграм
- Сумиран приказ на потрошениот буџет по категории и поставување на лимит
- Визуелно предупредување за надминување на буџетот

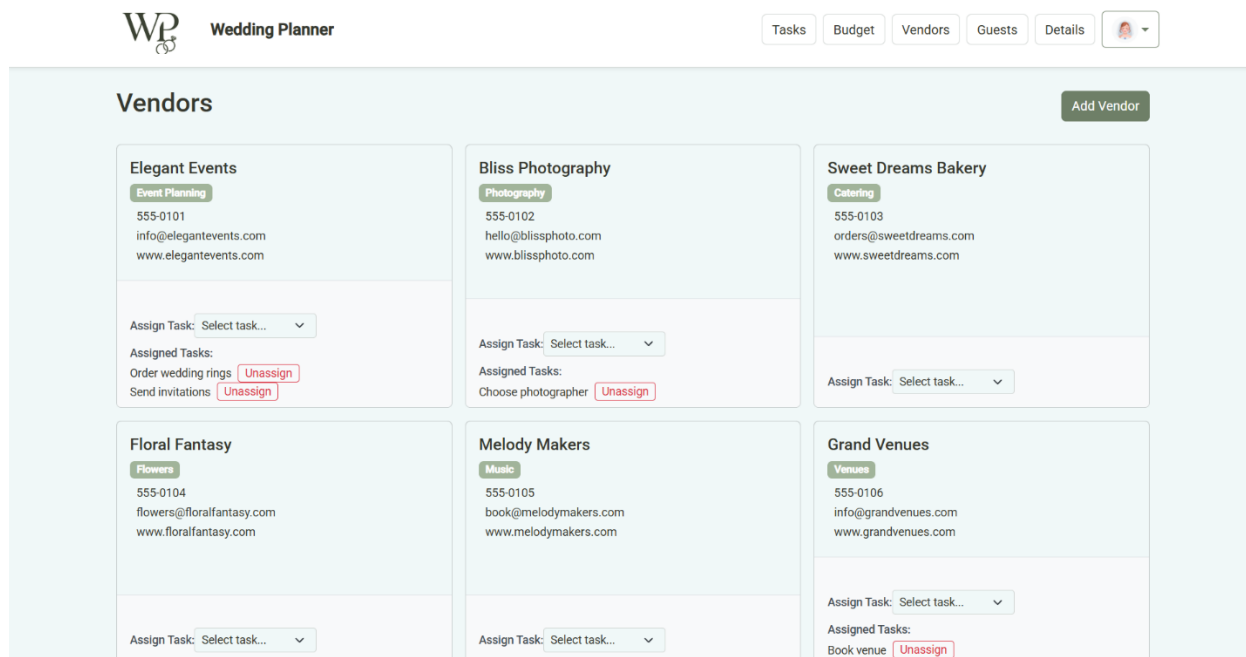


Слика 10: budget-tracker page

4.2.7 Страница за допавувачи (Vendors page)

Опција за додавање и зачувување на податоци за доставувачи со детали како име, контакт, е-пошта, веб страница, тип на услуга. За секој зачуван доставувач има опција да се уреди или избрише.

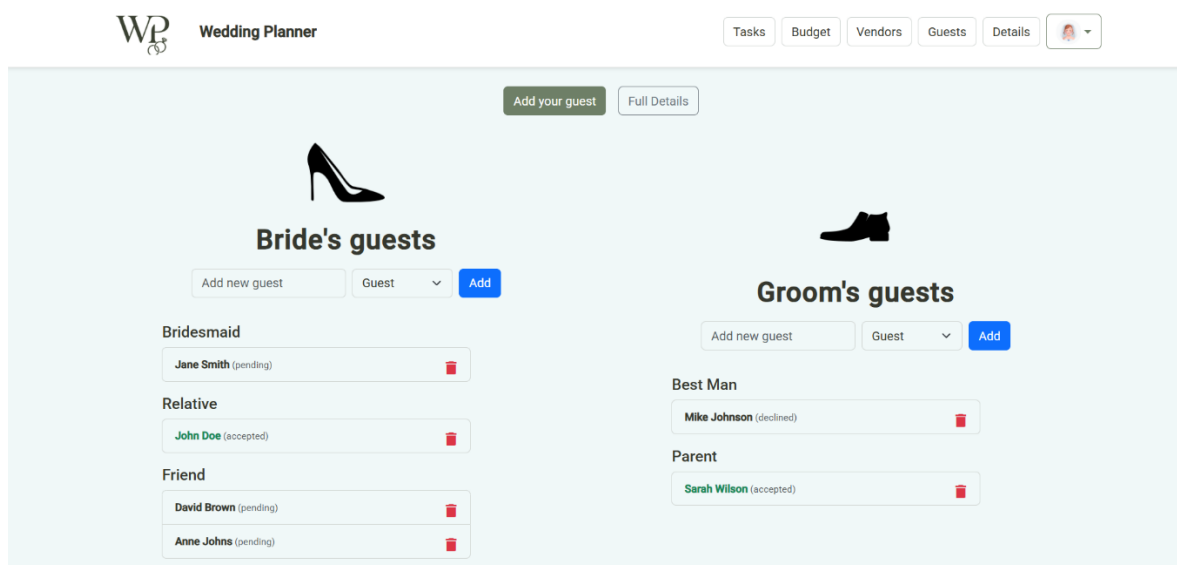
Дополнително под секој доставувач се прикажува листа со задачи од task-management page и има можност одредена задача да се определи за одреден доставувач.



Слика 11: vendors page

4.2.8 Страница за управување со гости (guest-list page)

Управувањето на листата со гости е најчесто една од најстресните активности при планирањето на свадбата. За да се олесни целокупното искуство, guest-list страницата е внимателно дизајнирана за едноставност и прегледност. Страницата е поделена на две половини – една за внес на гостите од страна на невестата, другата за гостите од младоженецот. При додавање на нов гостин, корисникот може да избере улога (на пример: Bridesmaid, Best Man, Parent, Relative, Friend, Guest), со што уште од самиот почеток се категоризира листата.

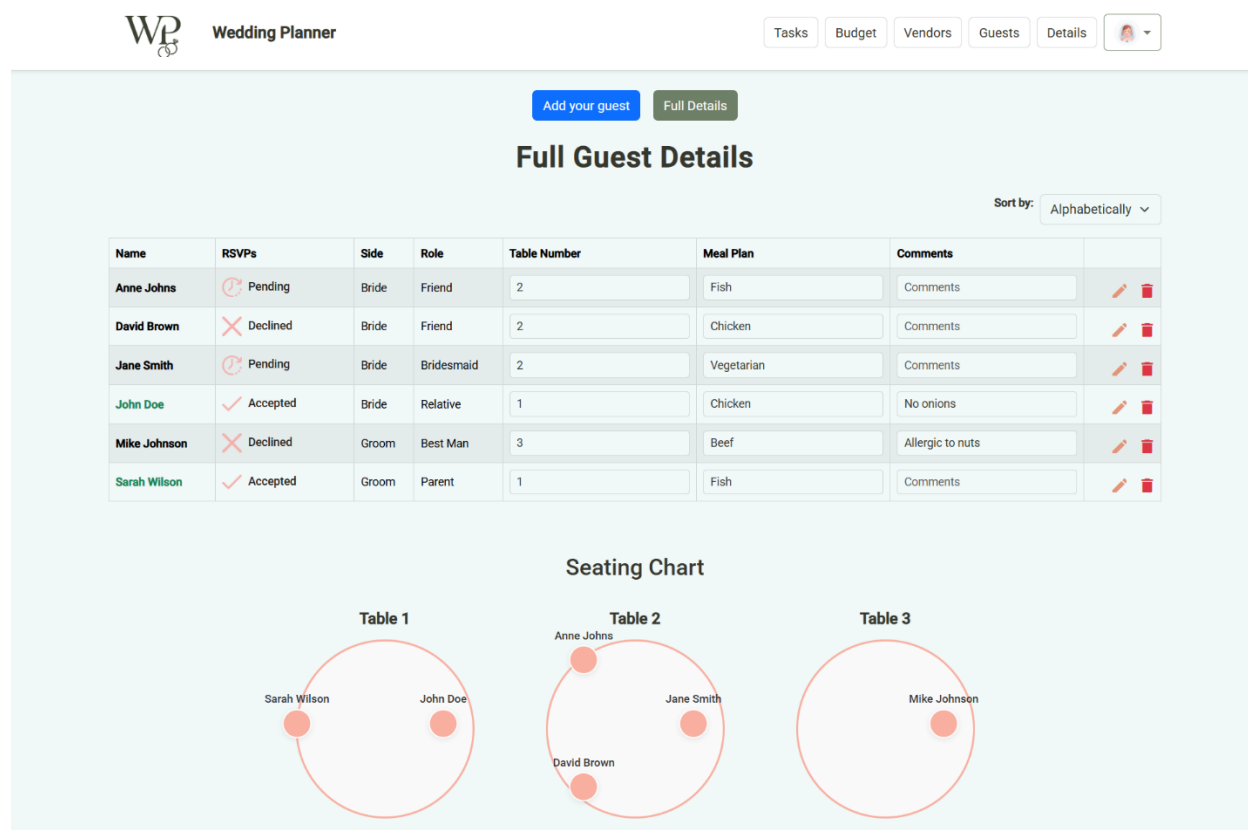


Слика 12: guest-list page – Форма за внес на гостите

На горниот дел од страната има и копче – Full Details која води до нова секција каде уредно во табела се прикажани сите гости. Табелата ги има следните функционалности:

- Раководење со RSVPs потврди
- Филтрирање на гостите по: страна, азбучен ред, улога и потврда за доаѓање
- Сместување на гостите по маси, кои подоцна визуелно ги прикажува во визуелен sitting chart
- Внесување мени за исхрана за секој гостин

Овој пристап овозможува лесна навигација и филтрација, брз преглед и ефикасно управување со сите детали поврзани со гостите, со што се намалува стресот и обезбедува подобра организација на целиот настан.



WP Wedding Planner

Tasks Budget Vendors Guests Details

Add your guest Full Details

Full Guest Details

Sort by: Alphabetically

Name	RSVPs	Side	Role	Table Number	Meal Plan	Comments	
Anne Johns	Pending	Bride	Friend	2	Fish	Comments	
David Brown	Declined	Bride	Friend	2	Chicken	Comments	
Jane Smith	Pending	Bride	Bridesmaid	2	Vegetarian	Comments	
John Doe	Accepted	Bride	Relative	1	Chicken	No onions	
Mike Johnson	Declined	Groom	Best Man	3	Beef	Allergic to nuts	
Sarah Wilson	Accepted	Groom	Parent	1	Fish	Comments	

Seating Chart

Table 1

Sarah Wilson John Doe

Table 2

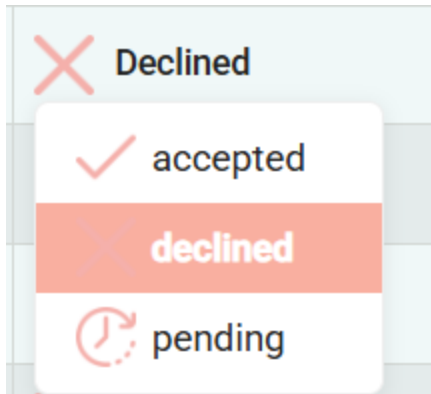
Anne Johns Jane Smith David Brown

Table 3

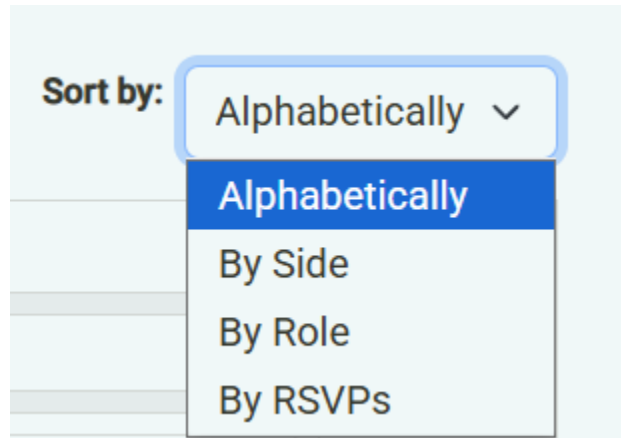
Mike Johnson

Слика 13: guest-list page Приказ на целосно детали

Дополнителни функции во guest-list страницата:



Слика 14: RSVPs потврди



Слика 15: Сортирање

4.2.9 Страница за детали на свадбата (wedding-details page)

Последната страница од Wedding Planner претставува централен екран за внес и визуелен приказ на сите важни детали од самата свадба. Функционалностите се прикажани во scroll-down листа, каде корисникот може лесно да пристапи до следниве секции:

- Форма за внес на податоци за младоженците и визуелен приказ
- Форма за внес и приказ на датум и локација на церемонијата
- Сумиран приказ на клучни елементи - листата со гости, завршени задачи и буџет
- Форма за организација на текот на настаните според време на случување и визуелен приказ за лесна прегледност на редоследот на активностите.

Со оваа страница, сите аспекти од свадбата се централно организирани, што им овозможува на корисниците брз пристап и лесна контрола над секој дел од планирањето.

Wedding Overview

Complete overview of your special day

Couple Details

Save

Bride

Groom

First Name

First Name

Last Name

Last Name

Age

Age

0

0

Event Information

Wedding Date

mm/dd/yyyy

Location

City, State

Guests

Manage Guests

3

Bride's Guests

2

Groom's Guests

Wedding Party:
5 total guests

Confirmed RSVPs:
2

Checklist

Manage Tasks

2 of 5 tasks completed

40%

Budget

Manage Budget

\$0
Total Budget

\$6,000
Spent

\$0
Remaining


Budget utilization: 0%


Event Itinerary

2:00 PM  **Ceremony**
Wedding ceremony at the venue

3:00 PM  **Cocktail Hour**
Drinks and appetizers

4:00 PM  **Reception**
Dinner and dancing


5:00 PM  **First Dance**
Couple's first dance

6:00 PM  **Cake Cutting**
Wedding cake ceremony

7:00 PM  **Bouquet Toss**
Traditional bouquet toss

8:00 PM  **Garner Toss**
Traditional garner toss

9:00 PM  **Open Dancing**
Party continues

11:00 PM  **Grand Exit**
Sparkler send-off

Add New Item

12 : 00 PM

Event

Description

+

Слика 16: wedding-details page

Wedding Overview

Complete overview of your special day

Couple Details

Save

Ana Janevska & Marko Markovski



Edit Details

Event Information

Wedding Date

September 30, 2025

Edit

Location

Skopje

Edit

Guests

Manage Guests

3

Bride's Guests

2

Groom's Guests

Wedding Party:
5 total guests

Confirmed RSVPs:
2

Checklist

Manage Tasks

2 of 5 tasks completed

40%

Budget

Manage Budget

\$1,200
Total Budget

\$6,000
Spent

\$0
Remaining

Budget utilization: 100%

Event Itinerary

2:00 PM **Ceremony**
Wedding ceremony at the venue

3:00 PM **Cocktail Hour**
Drinks and appetizers

4:00 PM **Reception**
Dinner and dancing

5:00 PM **First Dance**
Couple's first dance

6:00 PM **Cake Cutting**
Wedding cake ceremony

7:00 PM **Bouquet Toss**
Traditional bouquet toss

8:00 PM **Garter Toss**
Traditional garter toss

9:00 PM **Open Dancing**
Party continues

11:00 PM **Grand Exit**
Sparkler send off

Add New Item

12 : 00 PM

Event

Description

+

Слика 17: wedding details page по зачувување на податоците

4.2.10 Респонзивен дизајн

Дизајнот е тестиран на различни екрани за да обезбеди добра употребливост. Техники кои се користени за да се добие максимално позитивно корисничко искуство:

- SCSS Grid за респонзивни мрежи

```
// role-selection.scss
- .roles-grid {
-   display: grid;
-   grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
-   gap: 2rem;
-   margin-top: 2rem;
- }
```

Објаснување: `auto-fit` и `minmax()` овозможуваат автоматско прилагодување на бројот на колони според достапниот простор.

- Media Queries за различни екрани

```
// role-selection.scss
- @media (max-width: 768px) {
-   .main-title {
-     font-size: 2.5rem;
-     margin-bottom: 2rem;
-   }
-
-   .roles-grid {
-     grid-template-columns: 1fr;
-     gap: 1.5rem;
-   }
-
-   .role-card {
-     padding: 1.5rem;
-   }
- }
-
- @media (max-width: 480px) {
-   .main-title {
-     font-size: 2rem;
-   }
-
-   .role-card {
-     padding: 1rem;
-   }
- }
```

- Bootstrap Grid System

```
<!-- task-manager.html -->
- <div class="row g-2 align-items-center">
-   <div class="col-md-3">
-     <label class="form-label">Filter by assigned person:</label>
-   </div>
-   <div class="col-md-4">
-     <select class="form-select" (change)="filterByAssignee($any($event.target).value)">
-       <option value="">All tasks</option>
-       <option *ngFor="let user of availableUsers" [value]="user">{{ user }}</option>
-     </select>
-   </div>
- </div>
```

```
-     </select>
-   </div>
- </div>
```

Објаснување: `col-md-*` класите се прилагодуваат на различни големини на екрани.

- Flexbox за респонзивни компоненти

```
- // navbar.scss
- .main-layout {
-   display: flex;
-   flex-direction: row;
-   min-height: 100vh;
- }
```

- Viewport Meta Tag

```
- <!-- index.html -->
- <meta name="viewport" content="width=device-width, initial-scale=1">
```

- Responsive Typography

```
- // role-selection.scss
- .main-title {
-   font-size: 3.5rem;
-   font-weight: 700;
-   color: #2c3e50;
-   margin-bottom: 3rem;
- }
-
- @media (max-width: 768px) {
-   .main-title {
-     font-size: 2.5rem;
-     margin-bottom: 2rem;
-   }
- }
-
- @media (max-width: 480px) {
-   .main-title {
-     font-size: 2rem;
-   }
- }
```

- Responsive Form Layout
- Responsive Image Layout

4.2.11 Динамички компоненти и сервис слој

Со цел одржливост, секога поголема функционалност е имплементирана како независна компонента. Angular сервисите се користени за комуникација со backend-от преку HTTP барања. Ова овозможува лесно додавање нови функционалности и едноставно тестирање на компонентите.

4.2.12 Тестирање на фронтендот

Функционалностите се тестираа со симулации за внесување статички податоци и проверки дали истите се правилно прикажани и ажурирани. Дополнително навигацијата и responsive дизајнот беа проверувани на различни уреди и прелистувачи.

4.3 Backend дизајн

4.3.1 Технолошки стек и архитектонски стил

Backend делот е изграден со Java 21 (JDK 21) и Spring Boot 3, следејќи REST архитектонски стил со JSON размена на податоци. Апликацијата користи повеќеслојна архитектура со јасно разделени одговорности:

- Controllers: изложени REST ендпоинти и мапирање на HTTP.
- Services: бизнис-логика и доменски правила.
- Repositories: пристап до база преку Spring Data JPA.
- Models/Entities: мапирање на табели во ентитети.
- DTOs: влезно/излезни модели за стабилен API договор.

Базата на податоци е Microsoft SQL Server. Верзонирањето на API е со base path „/api/v1“. CORS е конфигуриран за безбедна FE↔BE комуникација. Во оваа фаза, извршувањето е локално развојно.

4.3.2 Модули/сервисни граници

Системот е организиран по домени со јасни граници и сопствен сервисен слој:

- Tasks: креирање/ажурирање/бришење задачи; статуси, рокови, приоритети.
- Budget: трошоци и приходи; резиме/агрегати по период и категорија.
- Vendors: управување со добавувачи и нивните контакт/услужни податоци.
- Guests: листа на гости, RSVP статус, посебни барања.
- (Опционално) Auth: основи за регистрација/најава и улоги.

Секој модул следи конвенција Controller → Service → Repository и користи DTO за I/O. Ова овозможува одржливост и ја олеснува идната еволуција.

4.3.3 Модел на податоци и перзистенција

Податочниот модел ги покрива клучните домени и релациите меѓу нив.

- Клучни ентитети: User, Task, BudgetItem, Vendor, Guest.
- Релации:
 - User–Task: 1:N (еден корисник има повеќе задачи).
 - User–BudgetItem: 1:N (буџет по корисник).
 - User–Guest: 1:N (гости по корисник).
 - Vendor–BudgetItem: 1:N (еден добавувач на повеќе ставки).

- ORM: Spring Data JPA (Hibernate) со SQL Server дијалект. Репозиториуми изведени од JpaRepository за CRUD/пагинација/сортирање.

- Миграции: во оваа фаза DDL генериран од Hibernate; планирано воведување Flyway/Liquibase за верзионирање на шеми.

- Интегритет и перформанси: not-null/unique ограничувања, надворешни клучеви, индекси на чести филтри (на пр. status, dueDate, category).

4.3.4 Дизајн на API (ресурси, верзионирање, JSON)

- Base path: /api/v1; формат: application/json; UTF-8; датуми ISO-8601 (UTC).

- Именување: множина за колекции (/tasks, /vendors, /guests, /budget/items).

- Методи/кодови: GET/POST/PUT/PATCH/DELETE со 200/201/204/400/404/409/422/500.

- Валидација: Bean Validation на DTO (@NotBlank, @Email, @Positive...).

- Пагинација/филтри/сортирање: ?page=&size=&sort=field,asc|desc + доменски филтри (на пр. status, dueBefore, type).

- Грешки: стандардизиран облик (Problem Details, RFC 7807) со type/title/status/detail.

- Примери на ендпоинти:

- Tasks: GET /api/v1/tasks, GET /api/v1/tasks/{id}, POST /api/v1/tasks, PUT/PATCH /api/v1/tasks/{id}, DELETE /api/v1/tasks/{id}

- Budget: GET /api/v1/budget/items, GET /api/v1/budget/items/{id}, POST /api/v1/budget/items, PUT/DELETE /api/v1/budget/items/{id}, GET /api/v1/budget/summary?from=&to=

- Vendors: GET /api/v1/vendors, GET /api/v1/vendors/{id}, POST /api/v1/vendors, PUT/DELETE /api/v1/vendors/{id}

- Guests: GET /api/v1/guests, GET /api/v1/guests/{id}, POST /api/v1/guests, PUT /api/v1/guests/{id}, PATCH /api/v1/guests/{id}/rsvp, DELETE /api/v1/guests/{id}

- (Опционално) Auth: POST /api/v1/auth/register, /login, /logout

- CORS: ограничени доверливи origin-и за фронтенд.

4.3.5 Автентикација и авторизација

Во тековната фаза системот работи во локално опкружување без комплексна авторизација. Планирано:

- Аутентикација: JWT bearer токени или session-based (Spring Security).

- Авторизација: улоги (на пр. USER/ADMIN) и методно/ендпоинт ниво ограничувања (@PreAuthorize).

- Заштита на чувствителни рути (создавање/ажурирање/бришење) и минимални јавни GET рути.

4.3.6 Валидација и обработка на грешки

- Валидација: анотации на DTO; конзистентни пораки за грешки.

- Глобален error handler: `@ControllerAdvice` за мапирање на исклучоци во 4xx/5xx одговори.

- Типични случаи: 400 (невалидни податоци), 404 (ресурс не постои), 409 (конфликт – unique), 422 (бизнис правила).

- Логирање на грешки со контекст (request URI, метод, корисник ако има).

4.3.7 Бизнис-логика и сервис слој

- Сервисите инкапсулираат доменски правила (на пр. Task не може со dueDate во минато при креирање; агрегации за буџет; RSVP транзиции).

- Транзакции: `@Transactional` околу операции со повеќе репозиториуми.

- Мапирање: DTO↔Entity со мапер (рочно или MapStruct) за чист separation.

4.3.8 Интеграции со надворешни системи

Во оваа фаза нема активни надворешни интеграции. Планирани можности:

- E-mail/SMS известувања (потврди, потсетници).

- Webhooks или 3rd-party APIs (плаќања, календари).

- Отпорност: timeouts/retries/circuit-breaker при надворешни повици.

4.3.9 Кеширање и перформанси

- Пагинација на листи (избегнување големи одговори).

- Индекси на колони со чести филтри/сортирања.

- Избегнување N+1 преку fetch стратегии/проекции кога е потребно.

- (Планирано) Апликативно кеширање на read-heavy рути (`@Cacheable`) и ETag/Cache-Control за GET одговори.

4.3.10 Конфигурација и околина

- Конфигурација преку `application.properties/.yaml`; чувствителни параметри преку `environment variables`.

- Профили: dev (локален), со можност за prod/stage во иднина.

- Параметри: серверски порт, CORS дозволени origin-и, JDBC URL/credentials за SQL Server.

- Secrets не се чуваат во VCS; користење на `.gitignore` и `env-specific` конфигурации.

4.3.11 Логирање и мониторинг

- Логирање: нивоа (INFO/DEBUG/ERROR), структуриран формат кога е можно.
- Трасабилност: корелациски идентификатор (X-Request-Id) во иднина.
- (Планирано) Spring Boot Actuator за health/info/metrics endpoints.
- Анализа на логови локално; можност за централизација при деплојмент.

4.3.12 Тестирање на backend

- Рачно тестирање: Postman (колекции со позитивни/негативни сценарија; околина за base URL/vars).
- Интеграциски проверки: browser console при FE↔BE повици (CORS, статуси, JSON).
- (Планирано) Unit тестови со JUnit/Mockito за сервис слој и WebMvc тестови за контролери.
- (Планирано) Тест податоци преку @Sql или Testcontainers за SQL Server.

4.3.13 Безбедност

- Валидација на влез (server-side), централизирано ракување со грешки без откривање имплементациски детали.
- Заштита од типични ранливости: SQL injection (JPA параметризирани queries), XSS (сервер не рендерира HTML), CSRF (ако се користат сесии).
- CORS: ограничени доверливи origin-и и методи.
- (Планирано) Rate limiting на write-пути и auditing на осетливи операции.

4.3.14 Деплојмент и CI/CD

- Тековно: локално извршување (dev профил).
- Репозиториум: GitLab на факултетот; можност за втор remote (GitHub/GitLab public).
- (Планирано) Docker за изедначени околина и GitLab CI за build/test/lint.
- Конфигурација преку околиински варијабли на сервер; без хардкодирани креденцијали.

4.3.15 Скалабилност и идни подобрувања

- Хоризонтално проширување: разделување на домени во независни сервиси ако расте обемот.
- Перформанси: апликативно/HTTP кеширање, проекции за read-heavy рути, async обработка за спори задачи.
- Одржливост: покриеност со тестови, автоматски миграции (Flyway/Liquibase), Actuator/metrics.
- Функционални проширувања: нотификации (e-mail/SMS), календари, увоз/извоз на податоци, улоги/пермисии.

5 Интеграциски слој

Последна фаза во градењето на овој проект е интеграцијата меѓу frontend-от и backend-от, каде што одделно развиените компоненти на frontend-от (во Angular) и backend-от (во SpringBoot) треба да се поврзат во функционална целина. Во случајот на Wedding Planner системот, Angular frontend-от е одговорен за визуелизација и интеракција со корисникот, додека backend делот обезбедува обработка на податоците, бизнис-логика и трајно чување на информации во базата на податоци. Целта на интеграцијата е да се овозможи безпречна комуникација помеѓу овие два слоја, така што податоците кои се внесуваат, менуваат или бришат на корисничкиот интерфејс да бидат веднаш отсликани во backend и базата на податоци.

5.1 Подготовка за интеграцијата

Првичните фази од развојот беа извршувани на две одделни гранки на GitHub – една за frontend и една за backend. Во frontend-от се користеа мок (mock) податоци за да ги развива компонентите и да го тестира изгледот на страниците без активен сервер. Во меѓувреме во backend-от паралелно се развиваше базата на податоци, REST API ендпоинтите и сервисната логика. Оваа поделба овозможи паралелен развој и побрз напредок без меѓусебно блокирање. Следно требаше да постепено да се спојуваат двата дела.

Angular фронтенд апликацијата е организирана во различни компоненти и сервиси. На пример, Wedding Details компонентата управува со целокупниот преглед на свадбата, вклучувајќи детали за парот, информации за настанот, гости, чеклисти, буџет и временска програма. Секој дел користи свои Angular сервиси за комуникација со backend API-то преку HttpClient модулот.

Spring Boot користи SQL Server база на податоци за складирање на податоците:

```
spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=WeddingPlannerDB
```

Ова значи дека сите податоци (корисници, задачи, гости, буџет) се чуваат во SQL Server базата на податоци.

Backend API-то е изградено со REST архитектура и овозможува GET, POST, PUT и DELETE барања. Овие барања овозможуваат:

- Превземање на податоци од базата (пр. листа на гости)
- Внесување и зачувување на нови податоци (пр. додавање на нов гостин или нов настан во временската програма)
- Ажурирање на постоечки податоци (на пр. менување на податоците на веќе постоечки гостин)
- Бришење на податоци (на пр. отстранување на непотребни задачи или гости)

5.2 Примена на CORS конфигурација

За време на развој, Angular апликацијата се извршува на `http://localhost:4200`, додека backend серверот работи на различен порт, на `http://localhost:8080`. За HTTP

барањата од фронтендот кон API-то да не бидат блокирани поради користењето на различни порти, се воведува CORS конфигурација (Cross-Origin Resource Sharing) на backend-от.

```
// WebConfig.java
@Configuration
public class WebConfig implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
            .allowedOrigins("http://localhost:4200")
            .allowedMethods("*");
    }
}
```

Бидејќи Angular работи на порт 4200, а Spring Boot на порт 8080, тие се сметаат за различни "домени". За да можат да комуницираат, Spring Boot мора да дозволи барања од Angular:

```
registry.addMapping("/**")
    .allowedOrigins("http://localhost:4200")
    .allowedMethods("*");
```

Ова значи: „Дозволи ги сите барања кои доаѓаат од порта 4200“.

Со ова, се овозможува непречена интеракција помеѓу двата слоја и се елиминираат грешки поврзани со блокирани повици. На пример:

1. Корисникот внесува email и лозинка во login формата
2. Angular испраќа POST барање до `http://localhost:8080/api/login`
3. Spring Boot го прима барањето, проверува ги креденцијалите во базата на податоци
4. Серверот враќа одговор (успешен или неуспешен)
5. Angular го прима одговорот и го прикажува резултатот на корисникот

5.3 Замена на хардкодираните податоци со HTTP барања во frontend

Откако backend-от беше стабилен, сите хардкодирани податоци од frontend-от постепено се заменуваа со реални повици преку HTTPClient сервисите.

HTTP Configuration:

```
// main.ts
import { provideHttpClient, withInterceptors } from '@angular/common/http';
import { AuthInterceptor } from '../app/shared/auth.interceptor';

bootstrapApplication(App, {
  providers: [
    provideRouter(routes),
    provideHttpClient(withInterceptors([AuthInterceptor]))
  ]
});
```

Се користеше Authentication Interceptor кој автоматски додаваше информации на секое барање:

```
if (token) setHeaders['Authorization'] = `Bearer ${token}`;  
if (userId) setHeaders['X-User-Id'] = userId;
```

Ова значи: „На секое барање до серверот, автоматски додај го токенот за автентификација“.

Services – Бизнес логика

Се користеа различни service фајлови:

-AuthService – за најавување и регистрација

-TaskService – за управување со задачи

-GuestService – за управување со гости

```
// task.service.ts  
@Injectable({ providedIn: 'root' })  
export class TaskService {  
  private readonly API_BASE_URL = 'http://localhost:8080/api';  
  
  getAllTasks(): Observable<Task[]> {  
    return this.http.get<TaskDTO[]>(`${this.API_BASE_URL}/tasks`).pipe(  
      map(tasks => this.mapTasksFromBackend(tasks))  
    );  
  }  
  
  createTask(task: Task): Observable<Task> {  
    const taskDTO = this.mapTaskToBackend(task);  
    return this.http.post<TaskDTO>(`${this.API_BASE_URL}/tasks`, taskDTO).pipe(  
      map(response => this.mapTaskFromBackend(response))  
    );  
  }  
  
  updateTaskCompletion(task)
```

Исто така се користи и Route Protection:

```
export const routes: Routes = [  
  { path: '', redirectTo: 'home', pathMatch: 'full' },
```

5.4 Интеграција на сервисите и функционалностите

Секој сервис од frontend-от беше поврзан со соодветниот REST контролер на backend-от. Овој процес вклучуваше:

- Ажурирање на сите Angular сервиси со вистински API URL адреси.
- Тестирање на секоја функција во Postman пред да се повика од frontend-от, за да се елиминираат грешки во серверската логика.
- Привремено додавање на логови (console.log) во компонентите за следење на добиените одговори.

API Endpoints (REST Controllers):

```
// UserController.java
@RestController
@RequestMapping("/api")
public class UserController {

    @PostMapping("/register")
    public ResponseEntity<RegisterResponseDTO> registerUser(@RequestBody RegisterDTO
registerDTO) {
        User user = userService.createUser(registerDTO.name(), registerDTO.email(),
registerDTO.password());
        RegisterResponseDTO responseDTO = new RegisterResponseDTO(
            user.getId(), user.getName(), user.getEmail()
        );
        return ResponseEntity.ok(responseDTO);
    }

    @PostMapping("/login")
    public ResponseEntity<LoginResponseDTO> loginUser(@RequestBody LoginDTO loginDTO) {
        User user = userService.login(loginDTO.email(), loginDTO.password());
        LoginResponseDTO responseDTO = new LoginResponseDTO(
            user.getId(), user.getEmail()
        );
        return ResponseEntity.ok(responseDTO);
    }
}
```

5.5 Улога на интеграцијата во развојниот процес

Оваа архитектура обезбедува чиста распределба помеѓу frontend и backend, со јасно дефинирани API endpoints и сигурна комуникација преку HTTP протоколот.

Интеграцијата не беше само техничко поврзување на два дела од апликацијата – таа беше и критичен момент за идентификување на слабости во дизајнот и имплементацијата. Некои функционалности, како филтрирањето на гостите или сортирањето на задачите, бараа дополнителни API прилагодувања за да се обезбеди ефикасна работа. Исто така, откриени беа ситуации каде што е потребно да се враќаат пораки за грешка од backend-от, за да се овозможи прикажување на кориснички известувања на frontend-от.

6 Тестирање

Тестирањето беше суштински дел од развојниот процес на Wedding Planner апликацијата за да се осигураме дека сите функционалности се стабилни, точни и кориснички пријателски пред финалната верзија. Со оглед на тоа што проектот беше поделен на frontend и backend, тестирањето се одвиваше во повеќе фази – изолирано тестирање на frontend и backend, интеграциско тестирање и финална проверка на целосната апликација.

6.1 Цели на тестирањето

Главните цели беа:

- Проверка на функционалностите на системот и исполнување на барањата дефинирани во анализата.
- Откривање и исправување на грешки и недоследности во логиката или интерфејсот.
- Осигурување на стабилна комуникација помеѓу frontend-от и backend-от.
- Потврда дека податоците се точни, правилно се прикажуваат и се ажурираат.
- Гарантирање на позитивно корисничко искуство преку визуелна и функционална проверка на интерфејсот.

6.2 Тестирање на backend со Postman

Првиот чекор беше тестирање на REST API ендпоинтите користејќи Postman. Овој процес вклучуваше:

- CRUD операции за гости – се тестираше додавање на нов гостин (POST), прикажување на сите гости (GET), ажурирање на постоечки податоци (PUT) и бришење (DELETE).
- Валидација на податоци – внесување невалидни или непотполни податоци за да се провери дали серверот враќа соодветни пораки за грешка.
- Буџет и задачи – проверка на додавање и ажурирање на задачи и трошоци, како и правилното враќање на пресметаните вредности.
- Одговори за грешка – тестирање на сценарија каде што се повикуваат непостоечки ресурси или недозволен барања.

Со оваа фаза, backend логиката беше потврдена како стабилна и подготвена за интеграција.

6.3 Тестирање на Frontend

Frontend-от беше тестиран паралелно со развојот на страниците. Првично се користеа хардкодирани податоци за визуелна проверка на изгледот, по што се премина на реални HTTP барања. Проверувани беа:

- Рутирање и навигација – сите линкови и менија правилно водеа до соодветните страници.
- Форми и валидација - формите за регистрација, логирање, додавање гости или задачи правилно ја проверуваа точноста на внесените податоци.
- Интерактивност - функционалности како сортирање, филтрирање и пребарување на гости или задачи беа испробани со различни сценарија.
- Визуелна конзистентност - проверено беше дали сите страници го следат истиот дизајн (палета на бои, типографија, стил на копчиња).

5.4 Интеграциско тестирање

По замена на мок податоците со реални HTTP повици и конфигурирање на CORS дозволите, беше извршено интеграционо тестирање за да се потврди правилната комуникација помеѓу frontend-от и backend-от. Во оваа фаза се проверуваше:

- Дали податоците внесени од корисникот преку frontend навистина се зачувуваат во базата на backend-от.
- Дали бришењето или уредувањето на податоците на backend-от веднаш се одразува на интерфејсот.
- Справување со грешки – како системот реагира кога серверот е недостапен или враќа порака за грешка.

Сите функционалности беа тестирани на интеграциската гранка `integration` на GitHub. По потврдување на стабилноста, кодот беше преместен на `main`.

5.5 Тестирање на корисничкото искуство

За да се осигури позитивно корисничко искуство, беше направено мануелно тестирање од перспектива на различни улоги. Се обрна внимание на тоа дали интерфејсот е интуитивен, дали формите се лесни за користење и дали визуелните прикази (табели, карти и менија) ги олеснуваат задачите.

5.6 Решени проблеми и потешкотии

Некои од клучните проблеми што беа откриени и решени за време на тестирањето:

- Проблем со CORS дозволите – при првите тестови, некои HTTP барања беа блокирани. Проблемот беше решен со глобална CORS конфигурација во Spring Boot.
- Интеграција помеѓу бекенд и фронтенд - Најзначајниот предизвик беше интегрирањето на бекендот (API-то изградено со Spring Boot) со Angular фронтендот.
- Работа со хардкодирани податоци и премин на HTTP барања - Кога преминавме на реални HTTP барања, се појавија некои несогласувања меѓу моделите во фронтендот и податоците што ги враќаше бекендот. За ова, направивме усогласување на интерфејсите во Angular и DTO објектите на бекендот, како и дополнителни проверки со `console.log` и дебагирање во DevTools.
- Багови во неправилно конфигурирани Angular сервиси - За време на тестирањето, откривме багови поврзани со неправилно конфигурирани Angular сервиси.
- Валидација на форми – Додадена е дополнителна Angular валидација за задолжителни полиња, правилна е-пошта и валидна лозинка.

7 План за одржување

За долгорочна функционалност и стабилност на Wedding Planner App се развиваат детални планови за редовно одржување, функционални проширувања, подобрување на корисничкото искуство и склабилност.

1. Редовно одржување

- Моментално справување со технички проблеми и багови одма по нивното откривање
- Константно правење на резервна копија на базата на податоци за заштита од нивно бришење
- Редовно ажурирање на сите frontend и backend библиотеки за одржување на перформансите

2. Функционални проширувања

Во иднина се планира интеграција на AI асистент кој ќе им помага на корисниците за навигацијата на нашата веб страна, да добиваат AI генерирани препораки и идеи, како и автоматско одговарање на најчесто поставувани прашањата кои ги поставуваат корисниците. Дополнително се разгледува и опцијата за мобилна верзија со цел за полесен пристап од мобилните телефони со можност да управуваат со нивната свадба од каде било, се услов да имаат интернет конекција

3. Подобрување на корисничкото искуство

Оптимизирање на корисничкиот интерфејс и визуелна прилагодливост на екранот за полесна употреба од различни уреди и додавање на систем за известување на корисниците и информирање за важни промени, задачи и рокови преку смс пораки или електронски e-mail пораки

4.Склабилност и инфраструктура

- Со зголемувањето на бројот на корисници на нашата апликација земено е во предвид преминувањето на cloud платформи за подобро складирање и често ажурирање на податоци за непречено функционирање
- Имплементацијата на мониторинг систем ќе овозможи следење на перформансите на серверите и целокупниот системот со кој што ќе се идентификуваат проблемите и баговите во системот и ќе доведе до нивно навремено решавање

5. Редовна ревизија и документација

Се извршува редовна документација на секоја промена за да се обезбеди јасен преглед на имплементациите и процедурите. Со помош на документираните прегледи собрани во текот на целата година се оценува ефикасноста на тековните функции и донесуваме одлуки за нивно подобрување и унапредување. Ова ни овозможува континуирано усовршување на платформата.

8 Заклучок

Во текот на развојот на Wedding Planner беа надминати многу предизвици, но исто така се стекнавме и со нови знаења.

Со користењето на Angular како frontend и Spring Boot за backend се доведе до развивање на флексибилна платформа која ги задоволува барањата на корисниците на сите аспекти на свадбени подготовки. Wedding Planner App на располагање нуди од управување со гости и буџет до следење на планот на настаните. Техничката структура е изработина со оставен простор за идно проширување, а неговиот дизајн го олеснува додавањето на мобилната поддршка.

Wedding Planner ги користи современите веб технологии и нуди практичен, лесен и ефикасен начин за управување со еден од вашите најважни настани во животот и придонесува подобра организација за идните младеници.